MSIN0094 Second Assignment Answer Sheet

Anonymous Candidate Number: RWFD2

Self-reported word count: 1421 words

```
Rows: 5,000
Columns: 15
$ user_id        <int> 10006, 10009, 10041, 10055, 10073, 10083, 10092, 10100,…
$ gender         <chr> "F", "F", "M", "F", "M", "F", "F", "M", "F", "M", "F", …
$ first          <int> 7, 65, 31, 77, 41, 53, 37, 17, 3, 59, 19, 61, 11, 49, 3…
$ last           <int> 7, 5, 17, 5, 15, 7, 3, 1, 3, 7, 9, 1, 5, 11, 9, 13, 9, …
$ electronics    <int> 15, 130, 70, 137, 61, 98, 139, 39, 15, 142, 54, 134, 24…
$ nonelectronics <int> 98, 288, 266, 49, 119, 54, 297, 281, 253, 223, 199, 64,…
$ home           <int> 0, 0, 2, 2, 2, 1, 2, 0, 0, 3, 1, 2, 0, 3, 2, 0, 0, 1, 0…
$ sports         <int> 1, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 2, 0, 3, 1, 0, 0, 0…
$ clothes        <int> 0, 3, 1, 3, 0, 1, 3, 0, 0, 0, 0, 2, 0, 3, 0, 0, 2, 0, 0…
$ health         <int> 0, 2, 1, 0, 1, 0, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0…
$ books          <int> 0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0…
$ digital        <int> 0, 3, 2, 1, 1, 4, 2, 2, 0, 1, 1, 2, 0, 2, 1, 1, 0, 0, 1…
$ toys           <int> 0, 1, 0, 4, 1, 0, 2, 0, 1, 6, 1, 2, 0, 3, 1, 0, 0, 1, 0…
$ subscribe      <chr> "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes",…
$ city           <chr> "London", "Glasgow", "Glasgow", "London", "London", "Lo…
```

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 is the minimum response level at which the total
profit from responding customers equals the total cost of sending the offers.
# In a targeted marketing campaign, it is used as a financial benchmark to decid
e whether targeting a specific customer segment is worthwhile. Only customers wi
th a predicted response rate higher than the break-even rate should be included.

# Intermediate variables:
cost_per_offer <- 1.5 # Cost of printing and mailing one offer
subscription_fee <- 6.99 # Revenue from 1-month subscription
expected_revenue <- 40 # Expected customer spending on products during the month
- what the firm revenues
COGS <- 0.9
shipping_cost <- 4 # Expected total shipping cost for the one who subscribed: £4
profit_margin <- 1 - COGS

# Gross margin from product purchases
gross_margin_sub <- expected_revenue * profit_margin
```

```r
# Calculating net profit
net_margin_sub <- gross_margin_sub - shipping_cost
# Total profit per subscribed customer
profit_per_customer <-subscription_fee + net_margin_sub

# The minimum subscription rate needed for the firm to break even
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"
```

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

```
[1] "profit_per_customer is 6.99"
```

```r
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**)

```r
# Computing the total mailing cost under blanket marketing as the multiplication
of the cost per offer and number of customers
total_costs_of_mailing_blanket <- cost_per_offer * nrow(data_full)
# Adding a new column with a binary indicator: 1 = subscribed, 0 = not subscribe
d.
data_full <- data_full %>%
  mutate(response = ifelse(subscribe == "yes", 1, 0))
# Sum of the indicator gives the total number of subscribed customers
n_subs <- sum(data_full$response)
# Computing the total profit from blanket marketing
total_profit_blanket <- n_subs * profit_per_customer
# ROI = (Total Profit - Total Cost) / Total Cost
# It measures the net return relative to the initial investment
ROI_blanket <- (total_profit_blanket - total_costs_of_mailing_blanket) / total_c
osts_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"
```

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

```
[1] "total_profit_blanket is  5857.62"
```

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

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

```
# As -0.219 < 0 it is not profitable for Tom to proceed with blanket marketing f
or the remaining customers
# The total mailing cost is higher than the total profit from subscribed custome
rs.
# Since the campaign does not cover its costs, Tom should consider a more target
ed strategy instead.
```

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**)

```
# create the required variables below
pacman::p_load(modelsummary)
data_RFM <- data_full %>%
  mutate(recency = last, # Number of days from the last purchase
         frequency = home + sports + clothes + health + books + digital + toys,
# Total number of all the purchases throughout a year
         monetary_value = electronics + nonelectronics, # Total amount spent
         subscribe_yes = ifelse(subscribe == "yes", 1, 0)) # Same as 'response'
in Q2

# report the summary statistics below
data_RFM %>%
  select(recency, frequency, monetary_value, subscribe_yes) %>%
  datasummary_skim()
```

	Unique	Missing Pct.	Mean	SD	Min	Median	Max	Histogram
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	

```
# Recency varies from 1 to 35 days, with an average of around 12 days, meaning s
ome customers purchased very recently while others have been inactive for longer
.
# Frequency also shows high variation: although the mean is about 4 purchases, m
any customers buy only once or twice, while a smaller group purchases more often
. Monetary value is the most spread-out variable, ranging from £15 to £477, with
a right-skewed distribution driven by high spenders.
# Finally, subscribe_yes has a mean of 0.2, showing that only 20% accepted the o
ffer, which confirms class imbalance in the dataset.
```

```r
# report the correlation table below
cor_RFM <- data_RFM %>%
  select(recency, frequency, monetary_value, subscribe_yes) %>%
  cor()

cor_RFM
```

```
                   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
```

```r
# The correlation table shows clear behavioural links between variables. Frequen
cy and monetary_value have a strong positive correlation: customers who purchase
more frequently also tend to spend more overall. Recency is negatively correlate
d with both of them, meaning that customers who bought something recently are al
so the ones who buy more often and spend more.
# There's also a noticeable pattern between subscription and the main RFM metric
s: subscribe_yes has a positive correlation with frequency and monetary_value, b
ut a negative one with recency. This means subscribers are usually active and va
luable buyers who have purchased recently
# Overall, the correlations confirm that recent, frequent shoppers contribute mo
st to revenue and are more likely to join the subscription plan.
```

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**)

```r
# create the RFM segments below

data_RFM <- data_RFM %>%
  mutate(recency_quintile = ntile(-recency, 5)) %>%
  ungroup() %>%
  # Frequency quintiles within each Recency segment
  group_by(recency_quintile) %>%
  mutate(frequency_quintile = ntile(frequency, 5)) %>%
  ungroup() %>%
  # Monetary quintiles within each RF segment
  group_by(recency_quintile, frequency_quintile) %>%
  mutate(monetary_quintile = ntile(monetary_value, 5)) %>%
  ungroup() %>%
  # Calculating the average response rate for each RFM segment
  group_by(recency_quintile, frequency_quintile, monetary_quintile) %>%
  mutate(avg_response_rate = mean(subscribe_yes, na.rm = TRUE)) %>%
  ungroup()

# report the RFM segment with highest average response rate below
```

```r
highest_avg_response_rate_segment <- data_RFM %>%
  group_by(recency_quintile, frequency_quintile, monetary_quintile) %>%
  summarise(
    avg_response_rate = mean(subscribe_yes, na.rm = TRUE),
    n_customers = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(avg_response_rate)) %>%
  slice(1)

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

```
Rows: 1
Columns: 5
$ recency_quintile   <int> 5
$ frequency_quintile <int> 2
$ monetary_quintile  <int> 5
$ avg_response_rate  <dbl> 0.475
$ n_customers        <int> 40
```

The RFM segment with the highest average response rate is R5–F2–M5, with an average response rate of 47.5%. This means customers who responded recently/often, and spent the most money before- overall are most likely to subscribe

Therefore, this segment can be considered the primary target group for future marketing campaigns to maximise ROI

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**)

```r
# Write your codes below to compute the ROI for the targeted marketing based on
RFM segments
# R5-F2-M5 - this segment had the highest average response rate
#data_RFM <- avg_response_rate = mean(subscribe_yes, na.rm = TRUE) - for this th
e answer was ROI_RFM = 1.21
# As the other segments may also bring profit, so taking into account the break-
even point
target_RFM <- data_RFM %>%
  filter(avg_response_rate > breakeven_response_rate)
    # recency_quintile == 5,
    # frequency_quintile == 2,
    # monetary_quintile == 5

total_costs_of_mailing_RFM <- cost_per_offer * nrow(target_RFM)
# The number of subscribers over profit per customer
total_profit_RFM <- sum(target_RFM$subscribe_yes) * profit_per_customer
# # ROI = (Total Profit – Total Cost) / Total Cost
# It measures the net return relative to the initial investment
```

```
ROI_RFM <- (total_profit_RFM - total_costs_of_mailing_RFM) / total_costs_of_mail
ing_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"
```

```
# Tom should target the R5-F2-M5 segment because it has the highest average resp
onse rate of 47.5%, which is above the break-even response rate of ≈39.8%.
```

These customers are relatively recent buyers, purchase occasionally, and are the highest-spending group. Sending offers to this group results in an ROI of around 39.8%, while blanket marketing had a negative ROI of 21.9%.

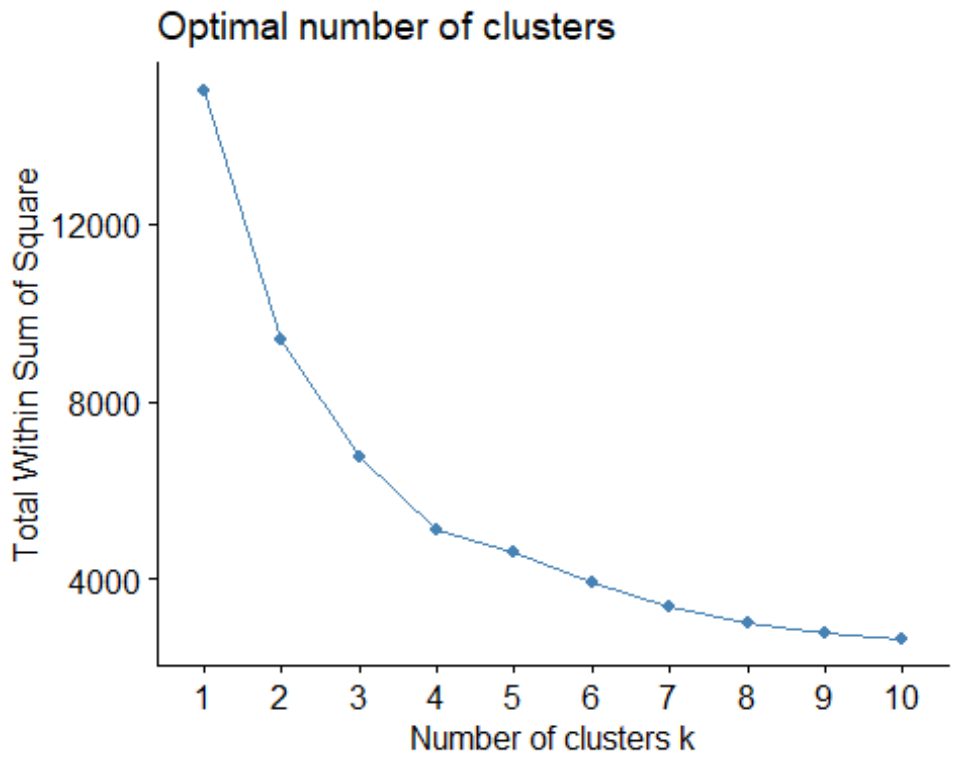
Therefore, focusing on RFM segment will maximise marketing efficiency and return on investment. ROI_RFM > 0

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

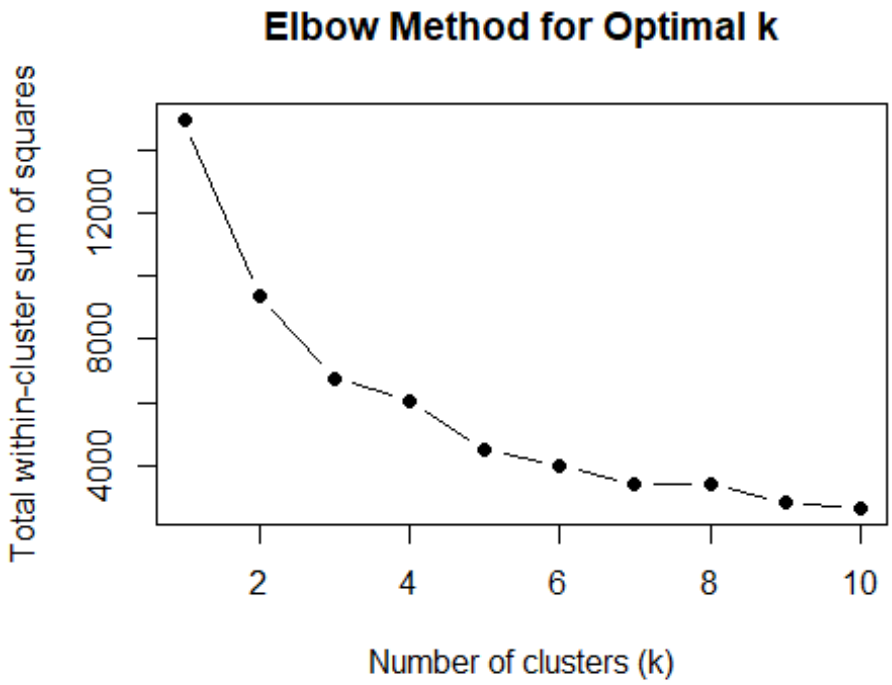
Question 6. Use the three RFM variables to segment customers using the K-means clustering algorithm. Please use `set.seed(888)` for reproducibility. Specify `x` and `centers` in the `kmeans()` function, while using the default values for ALL remaining arguments for simplicity. (**8 pts**)

```
# The RFM variables were scaled to ensure they contribute equally to distance ca
lculations, as K-means uses Euclidean distance.
# Without scaling, variables like monetary value would dominate clustering resul
ts

# create the RFM variables below
pacman::p_load(broom)

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

data_kmeans <- data_RFM %>%
  select(recency, frequency, monetary_value) %>%
  scale() %>%
  as.data.frame()

# Recency, frequency, and monetary_value are on very different scales, so scalin
g ensures that variables contribute equally to distance calculations in K-means.
```

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

# Vector to store total within-cluster sum of squares
wss <- numeric(10)
for (k in 1:10) {
  model <- kmeans(x = data_kmeans, centers = k)
  wss[k] <- model$tot.withinss
}
fviz_nbclust(data_kmeans, kmeans, method = "wss")
```




```r
# Making a plot to visualize
plot(1:10, wss, type = "b", pch = 19,
     xlab = "Number of clusters (k)",
     ylab = "Total within-cluster sum of squares",
     main = "Elbow Method for Optimal k")
```




```
# Before clustering, the three RFM variables were scaled because they have diffe
rent units and ranges.
# The total within-cluster sum of squares (WSS) was plotted for k = 1 1to 0.
# The curve sharply decreases until k = 4 and then flattens and the Elbow method
shows a clear bend at k = 4, indicating that four clusters balance model simplic
ity and within-group similarity

# implement k-means clustering below

# do not modify seeds
set.seed(888) # Using set.seed(888) ensures reproducibility.
result_kmeans <- kmeans(x = data_kmeans, centers = 4)
# The K-means clustering with 4 clusters segments customers based on their recen
cy, frequency, and spending behaviour

# do no modify the code below, this is for TA to check the results
# use broom::tidy() to check the clusters.
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.
data_kmeans_full <- data_RFM %>%
  mutate(cluster = result_kmeans$cluster)

cluster_summary <- data_kmeans_full %>%
  group_by(cluster) %>%
  summarise(
    avg_response_rate = mean(subscribe_yes, na.rm = TRUE),
    n_customers = n()
  ) %>%
  arrange(desc(avg_response_rate))

cluster_summary
```

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

```
# Based on the K-means segmentation, Cluster 2 has the highest average response
rate 26.2%, which is above the break-even response rate of -21%
# Customers in this cluster are likely to be recent subscribers with average pur
chase frequency and relatively high spending.

# Compute the ROI of marketing if Tom conducts k-means targeted marketing to the
segment you selected
target_cluster <- 2 # The highest average response rate
# Filtering customers belonging to the 2nd cluster
target_data <- data_kmeans_full %>%
  filter(cluster == target_cluster)

# Calculating total mailing cost
total_costs_of_mailing_kmeans <- cost_per_offer * nrow(target_data)

# Calculating total profit from all subscribed customers in this cluster
total_profit_kmeans <- sum(target_data$subscribe_yes) * profit_per_customer

# The return on investment of targeting only this cluster
ROI_kmeans <- (total_profit_kmeans - total_costs_of_mailing_kmeans) / total_cost
s_of_mailing_kmeans
```

```
# do not modify the code below

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

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

```
# Targeting only customers in Cluster 2, the ROI of the Amazon campaign is posit
ive and higher than ROI_blanket, confirming that targeted marketing using K-mean
s segmentation is more efficient than blanket marketing.
```

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

Pros:

K-means works reasonably well for our RFM dataset because it groups customers based on how they actually behave, how recently they purchased, how often they shop, and how much they spend. After scaling the variables, the clusters usually form meaningful patterns (like high-frequency/high-spending "loyals" vs. low-value shoppers).

It's fast, easy to run, and gives a quick sense of structure in the customer base without using any complicated models. Once the clusters are formed, we can check each group's actual response rate (using *subscribe_yes*) and target the one that performs best. For this dataset, K-means provides a simple way to focus marketing on segments that look more promising and avoid unnecessary mailings.

Cons:

The big limitation is that K-means doesn't use the outcome (subscribe_yes) at all. It only looks at behaviour, so the clusters might not line up with who is actually likely to respond. A supervised model would rank customers much more precisely.

We also have to pick the number of clusters manually, which is never perfect even with an elbow plot. The method is sensitive to scaling, outliers, and random initialization. And K-means assumes clusters are roughly spherical and balanced under Euclidean distance, which isn't always true for real RFM data. If the structure is uneven or messy, the segments can be misleading and need to be validated.

4. Decision Tree Analysis (24 pts)

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

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

# A binary numeric subscribe_yes was created earlier
# Assign the mutated data into a new dataset
data_tree <- data_RFM %>%
  select(subscribe_yes, recency, frequency, monetary_value)
```

```
# sample the index for training data
training_set_index <- sample(1:nrow(data_tree), 0.7 * nrow(data_tree),
                             replace = FALSE )
# Splitiing data_tree into a training set
data_training <- data_tree %>%
  slice(training_set_index)
# Splitiing data_tree into a test set
data_test <- data_tree %>%
  slice(-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
```

- No, scaling is not required for decision tree models. Decision trees split data by finding thresholds that best separate classes according to impurity measures such as Gini impurity or Information Gain (based on entropy). Since these metrics depend only on the ordering of values, not on the magnitude or units, rescaling does not change the split points. Therefore, changing the scale of a variable, for instance, converting *recency* from days to weeks or *monetary value* from pounds to euros does not affect the splitting points or the resulting tree structure.
A Gini score of 0 indicates a perfectly pure node. Because decision trees focus on reducing impurity rather than measuring distance, they remain robust to variable scaling and do not require standardization or normalization before training

- Splitting data into training and test sets allows us to evaluate how well a model generalises to unseen customers.

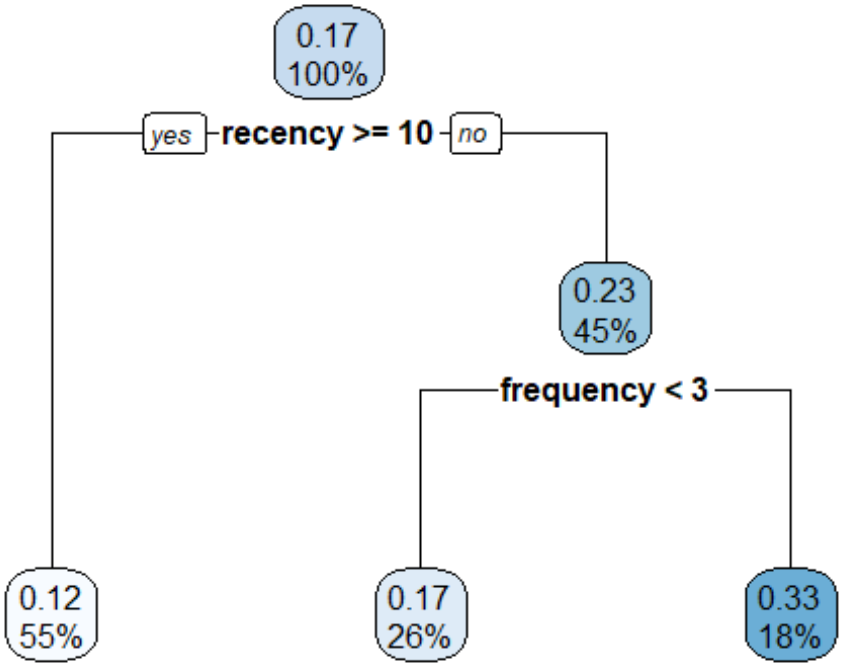
The training set (70%) is used to grow the decision tree and learn the relationships between predictors (*RFM variables*) and the target variable *subscribe_yes*. The test set (30%) is held out and used only for performance evaluation.

The decision tree learns from the training data by recursively splitting nodes to minimise impurity, for example, Gini for classification or Sum of Squared Errors for regression tasks. The test data, which the model has never seen, is used to estimate predictive accuracy and detect overfitting, when the tree memorises training details instead of general patterns. Using separate sets ensures the model's performance reflects its true predictive power and supports reliable marketing decisions for future customers.

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

```
# train model tree1 below
pacman::p_load(rpart, rpart.plot)
# The tree model with RFM classification
tree1 <- rpart(
  formula = subscribe_yes ~ recency + frequency + monetary_value, # Predictors
  data = data_training, # The training set
```

```
  method = "anova" # Sum of Squared Errors is used as
)
# This setup allows the model to predict a continuous outcome between 0 and 1, r
epresenting the probability that a customer subscribes

# visualize tree1 below
rpart.plot(tree1)
```



After training the model, the visualization of tree1 shows the root node and subsequent branches created by the algorithm.

The decision tree searches through all possible predictor variables (recency, frequency, and monetary_value) and all potential cut-off points for each variable. It then selects the variable and threshold that produce the largest reduction in total Sum of Squared Errors between the parent node and the two resulting child nodes. This ensures that each split makes the observations in the child nodes more homogeneous in terms of the outcome variable

The decision tree (tree1) identifies recency as the most important variable for the first split. The root node shows the overall average probability of subscription: around 17% of all customers in the training data subscribed.

The first split divides customers based on recency ≥ 10. Customers with smaller recency values (those who purchased more recently) are sent to the left branch, where the predicted probability of subscription decreases to 0.12.These customers already interacted with the platform recently, so they are less likely to respond to another subscription offer immediately.

Customers with recency < 10 move to the right branch. Within this group, the model performs a second split on frequency < 3, showing that purchase frequency further distinguishes customer behaviour. If frequency < 3, the predicted subscription rate is 0.17, suggesting low engagement and modest conversion potential.If frequency ≥ 3, the probability rises to 0.33, indicating that customers who buy more frequently and have not purchased recently are most likely to subscribe.

In summary, tree1 shows that subscription likelihood increases for less recent but more frequent purchasers: customers who have not interacted very recently yet remain active buyers are the most promising target group.

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

```
# Compute the ROI for tree1 below.
# Make predictions on test set
prediction_tree1 <- predict(tree1, data_test)
# Create a new column 'is_target_decisiontree' to identify which customers to ta
rget.
# A customer is targeted (value 1) if their predicted response probability is gr
eater than the break-even response rate; otherwise, they are not targeted (value
0).
data_test <- data_test %>%
  mutate(
    predicted_prob_decisiontree = prediction_tree1,
    is_target_decisiontree = ifelse(predicted_prob_decisiontree > breakeven_resp
onse_rate, 1, 0)
  )
# Calculate the total cost of the targeted marketing campaign using the decision
tree model.
# This is the cost per offer multiplied by the number of targeted customers.
total_costs_of_mailing_decisiontree1 <- cost_per_offer * sum(data_test$is_target
_decisiontree)

# Identify the customers who were targeted and actually subscribed.
data_test_targeted_customers <- data_test %>%
  filter(is_target_decisiontree == 1) %>% # Keep only targeted customers
  filter(subscribe_yes == 1) # Keep only actual subscribers

# Calculate the total profit from the targeted campaign.
# This is the number of targeted customers who actually subscribed, multiplied b
y the profit per customer
total_profit_decisiontree1 <- sum(data_test_targeted_customers$subscribe_yes) *
profit_per_customer
# ROI = (Total Profit - Total Cost) / Total Cost
ROI_decisiontree1 <- (total_profit_decisiontree1 - total_costs_of_mailing_decisi
ontree1) / total_costs_of_mailing_decisiontree1

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

```
[1] "ROI_decisiontree1 is  0.6776"
```

```
# The ROI from tree1 is much higher compared with the ROI from k-means (0.6776 > 0.2221)
# K-means performs worse because it is an unsupervised method: it creates segments based only on RFM similarity, without using the actual subscription outcomes. As a result, the targeting is broad and includes many low-probability customers.
# The decision tree is a supervised model and learns directly from the response variable. It identifies customers with the highest predicted subscription probability and targets only those above the break-even threshold, which makes the campaign far more precise.
# Therefore, tree1 achieves a higher ROI because it selects a more profitable subset of customers, while k-means cannot optimise targeting in the same way.
```

5. Random Forest (20 pts)

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

[placeholder, delete this before writing your answers]

```
# train the random forest model below
pacman::p_load(ranger)
# set.seed(888) # To make of replication
randomforest <- ranger(
  formula = subscribe_yes ~ recency + frequency + monetary_value,
  data = data_training, # Training dataset for the model
  num.trees = 2000, # 2000 decision trees
  seed = 888, # To make of replication
  probability = TRUE # Set to TRUE to get class probabilities, which we need for targeting
  )
# Compute the RPI for rendom forest:

# Use the trained random forest to make predictions on the test set.
prediction_from_randomforest <- predict(randomforest, data_test)

# Add the predicted probabilities to the test dataset.
# The output 'prediction_from_randomforest$predictions' is a matrix with probabilities for each class (0 and 1).
# We need the second column [, 2], which contains the probability of a positive response (Response = 1).
data_test <- data_test %>%
    mutate(predicted_prob_randomforest = prediction_from_randomforest$predictions[, 2],
           is_target_randomforest =
           ifelse(predicted_prob_randomforest > breakeven_response_rate, 1, 0) # Create a targeting indicator based on the random forest model's predictions
# A customer is targeted if their predicted probability is above the break-even rate
           )

# Calculate the total cost of the random forest–based targeted campaign.
# This equals cost per offer multiplied by the number of targeted customers.
```

```r
total_costs_of_mailing_randomforest <-
  cost_per_offer * sum(data_test$is_target_randomforest)

# Identify the customers who were targeted and actually responded.
data_responding_targeted_customers <- data_test %>%
  filter(is_target_randomforest == 1) %>% # Targeted customers
  filter(subscribe_yes == 1) # Actual responders

# Calculating total profit from customers who actually subscribed
total_profit_randomforest <- nrow(data_responding_targeted_customers) * profit_p
er_customer

# ROI = (Total Profit – Total Cost) / Total Cost
# Compute the ROI for random forest below
ROI_randomforest <- (total_profit_randomforest - total_costs_of_mailing_randomfo
rest) / total_costs_of_mailing_randomforest

# do not modify the code below

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

```
[1] "ROI_randomforest is  0.494045801526717"
```

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)

- There are two trade-offs: the accuracy-interpretability and bias-variance

Accuracy-interpretability: simple models are easy to understand and explain, but they usually offer lower predictive accuracy. More complex models, on the other hand, can capture broader relationships in the data and, therefore, give higher accuracy, but they become harder to interpret. In practice, this means that business users may prefer transparent models, while data scientists may prefer more accurate ones. Finding the right balance depends on the goal of the task and how important transparency is for decision-making.

Bias-variance: a model with high bias is too simple, fails to capture patterns, and performs poorly (underfitting). A model with high variance fits the training data too closely and performs inconsistently on new customers (overfitting). More complex models often reduce bias but increase variance; simpler models do the opposite. The aim in supervised learning is to find a model that generalises well, not too simple, not too sensitive, so that predictions remain stable on unseen data.

- The decision tree (tree1) is highly interpretable: it's seen how it splits customers by recency, frequency, or monetary value. This reflects the interpretability advantage, but the tree misses complex nonlinear interactions. Its predictions are therefore less accurate than more advanced models, showing the accuracy–interpretability tension. Tree1 also has moderate bias and moderate variance, which is why its ROI improves compared with K-means but is still not optimal.

The random forest, which uses 2,000 trees, is much less interpretable but captures far richer patterns. Averaging many trees reduces variance and produces more stable predictions than a single tree. This led to a higher ROI in targeted marketing in your results, showing the benefit of overcoming both bias and variance problems.

Meanwhile, K-means has no response variable at all, and its lower ROI illustrates why supervised models are necessary for this task.

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)

The first recommendation is to tune the random forest hyperparameters, especially those related to tree complexity and randomness. Reducing overfitting helps lower variance and improves performance on unseen data. Even though random forest already reduces variance by averaging many trees, each individual tree can still be too deep and too specialised. Tom can increase performance by limiting the maximum depth of each tree, so the trees focus on large, meaningful patterns instead of small, noisy splits. Another important parameter is mtry, which controls how many predictors are randomly considered at each split. Using a smaller mtry increases diversity between trees and helps the forest discover different relationships in the data. This may improve accuracy and stabilises predictions. Since tuning does not require any new customer information - no need in new data.

The second recommendation is to improve the quality of features using the variables Tom already has. Better-structured predictors often make tree-based models more powerful. Tom can transform existing variables to make patterns easier for the model to learn. For example, recency or monetary value can be grouped into meaningful buckets: low, medium, high; which helps trees split customers more effectively. Tom can also create simple interaction features, such as combining frequency and monetary value to capture “high-spending frequent subscribers.” On the other hand, variables that add noise and don’t correlate with subscription behavior can be removed. Cleaning and reformatting features improves the signal-to-noise ratio and allows the random forest to perform better.