Data622_Assignment_2

Natalie Kalukeerthie and Anna Moy

2025-09-19

Assignment 1

```
library(tidyverse)
library(ggplot2)
library(corrplot)
library(dplyr)

#import full bank dataset

# Import the CSV file from github
bank <- read_csv2("https://raw.githubusercontent.com/nk014914/Data622/refs/heads/main/bank-full.csv", sh
```

First, we'll look at the structure of the data.

```
## spc_tbl_ [45,211 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age      : num [1:45211] 58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : chr [1:45211] "management" "technician" "entrepreneur" "blue-collar" ...
##  $ marital  : chr [1:45211] "married" "single" "married" "married" ...
##  $ education: chr [1:45211] "tertiary" "secondary" "secondary" "unknown" ...
##  $ default  : chr [1:45211] "no" "no" "no" "no" ...
##  $ balance  : num [1:45211] 2143 29 2 1506 1 ...
##  $ housing  : chr [1:45211] "yes" "yes" "yes" "yes" ...
##  $ loan     : chr [1:45211] "no" "no" "yes" "no" ...
##  $ contact  : chr [1:45211] "unknown" "unknown" "unknown" "unknown" ...
##  $ day      : num [1:45211] 5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : chr [1:45211] "may" "may" "may" "may" ...
##  $ duration : num [1:45211] 261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : num [1:45211] 1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : num [1:45211] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : num [1:45211] 0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : chr [1:45211] "unknown" "unknown" "unknown" "unknown" ...
##  $ y        : chr [1:45211] "no" "no" "no" "no" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   job = col_character(),
##   ..   marital = col_character(),
##   ..   education = col_character(),
##   ..   default = col_character(),
```

```
##   ..   balance = col_double(),
##   ..   housing = col_character(),
##   ..   loan = col_character(),
##   ..   contact = col_character(),
##   ..   day = col_double(),
##   ..   month = col_character(),
##   ..   duration = col_double(),
##   ..   campaign = col_double(),
##   ..   pdays = col_double(),
##   ..   previous = col_double(),
##   ..   poutcome = col_character(),
##   ..   y = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>

##       age            job              marital           education
##  Min.   :18.00   Length:45211       Length:45211       Length:45211
##  1st Qu.:33.00   Class :character   Class :character   Class :character
##  Median :39.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :40.94
##  3rd Qu.:48.00
##  Max.   :95.00
##    default             balance         housing              loan
##  Length:45211       Min.   : -8019   Length:45211       Length:45211
##  Class :character   1st Qu.:    72   Class :character   Class :character
##  Mode  :character   Median :   448   Mode  :character   Mode  :character
##                     Mean   :  1362
##                     3rd Qu.:  1428
##                     Max.   :102127
##    contact              day           month              duration
##  Length:45211       Min.   : 1.00   Length:45211       Min.   :   0.0
##  Class :character   1st Qu.: 8.00   Class :character   1st Qu.: 103.0
##  Mode  :character   Median :16.00   Mode  :character   Median : 180.0
##                     Mean   :15.81                      Mean   : 258.2
##                     3rd Qu.:21.00                      3rd Qu.: 319.0
##                     Max.   :31.00                      Max.   :4918.0
##     campaign          pdays          previous          poutcome
##  Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000   Length:45211
##  1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000   Class :character
##  Median : 2.000   Median : -1.0   Median :  0.0000   Mode  :character
##  Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
##  3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :63.000   Max.   :871.0   Max.   :275.0000
##       y
##  Length:45211
##  Class :character
##  Mode  :character
##
##
##

## [1] 0
```

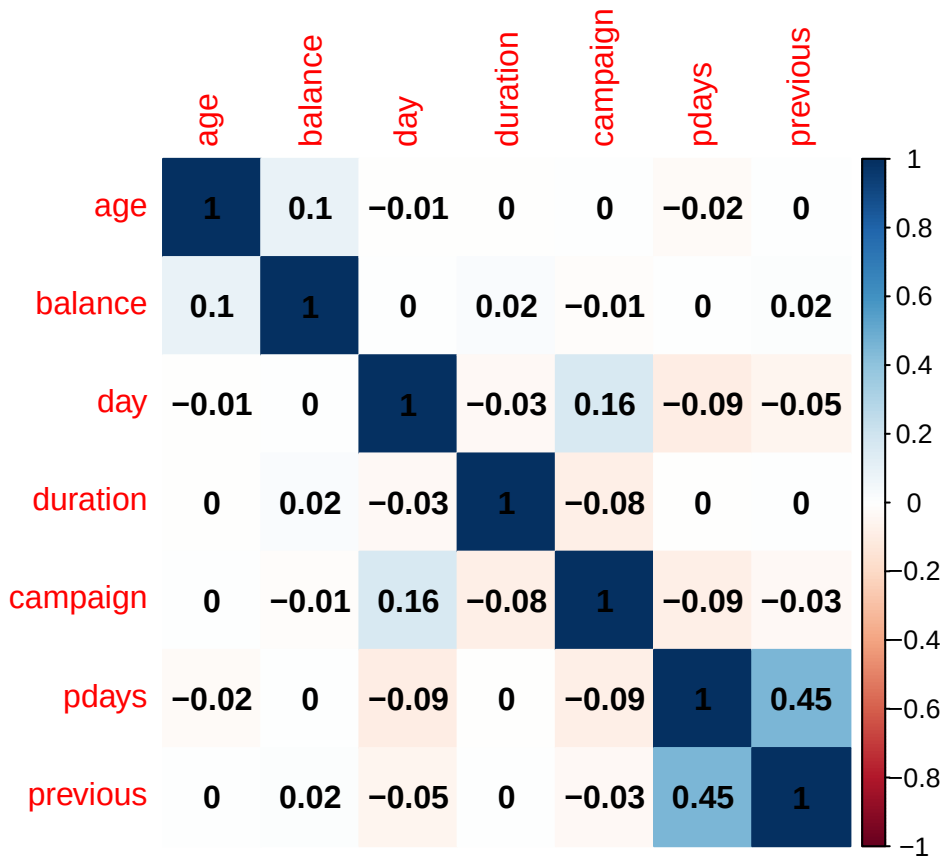
Our chosen dataset,'Bank-Full', has 45,211 rows and 17 columns. It contains both numerical variables (e.g.,

age, balance, duration) and categorical variables (e.g., job, marital status, education). Our target variable is y, which indicates whether a client subscribed to a term deposit. Having a large dataset allows us to run models more effectively.

The summary provides central tendency (mean, median) and spread (quartiles, min, max) for numeric variables, and frequency counts for categorical ones. This helps us quickly spot imbalances or unusual values.

1. Are the features (columns) of your data correlated?

```
numeric_vars <- bank %>% select_if(is.numeric)
corr_matrix <- cor(numeric_vars)
corrplot(corr_matrix, method = "color", addCoef.col = "black")
```

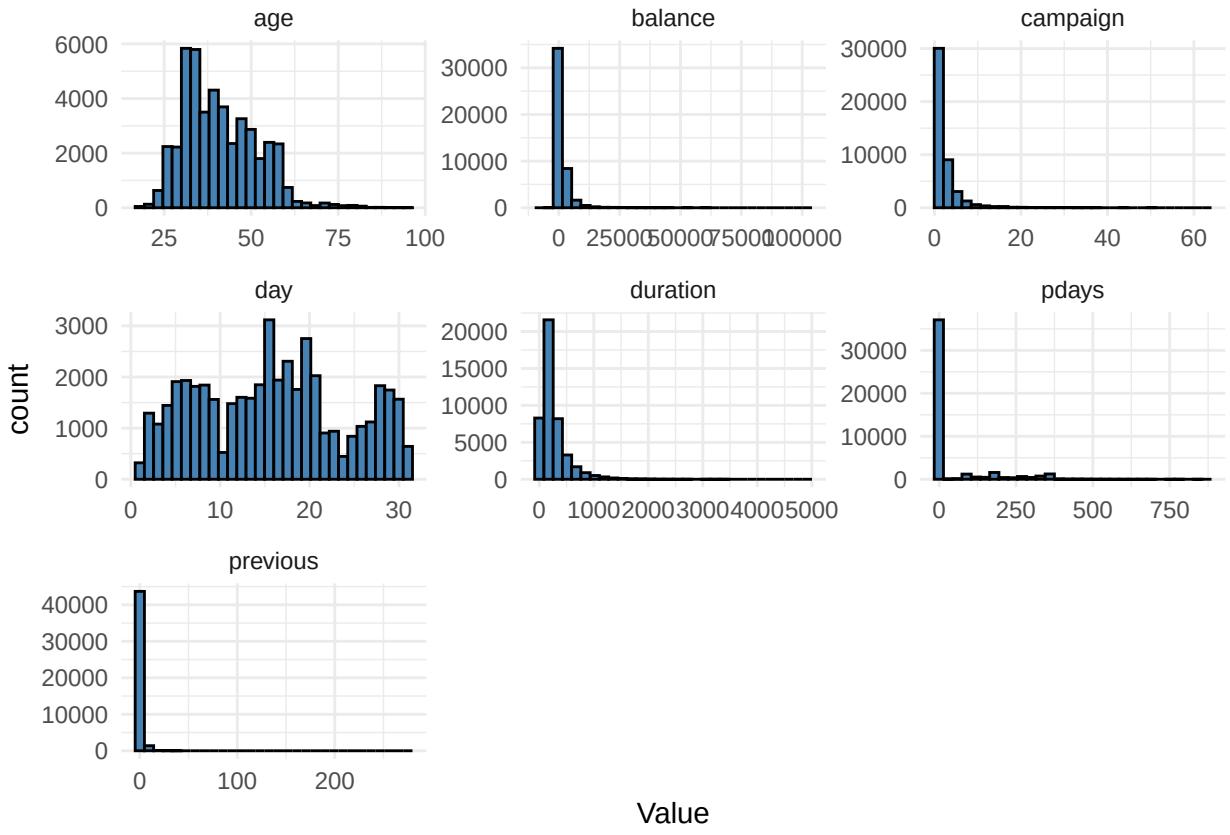


The correlation matrix shows that most numeric features are weakly correlated with each other. For example, duration and campaign have a very low correlation. This indicates that predictors are largely independent, which is good for most modeling approaches. With some indicating zero it means there is no correlation with other features in the data.

2. What is the overall distribution of each variable?

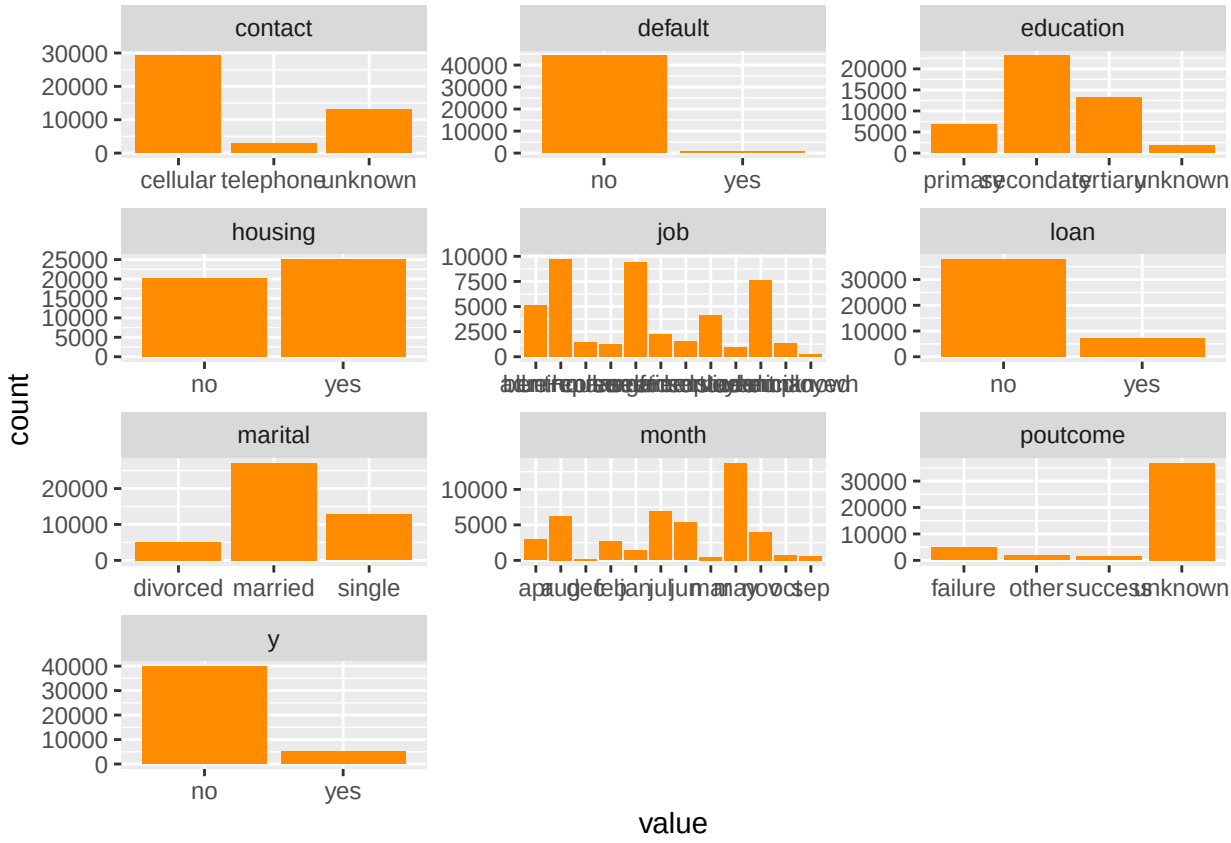
```
#numerical
numeric_vars %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Value)) +
```

```
  facet_wrap(~Variable, scales = "free") +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  theme_minimal()
```




```
#categorical
categorical_data <- select(bank, where(is.character))

categorical_data %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(x = value)) +
  facet_wrap(~name, scales = "free", ncol = 3) +
  geom_bar(fill = "darkorange")
```

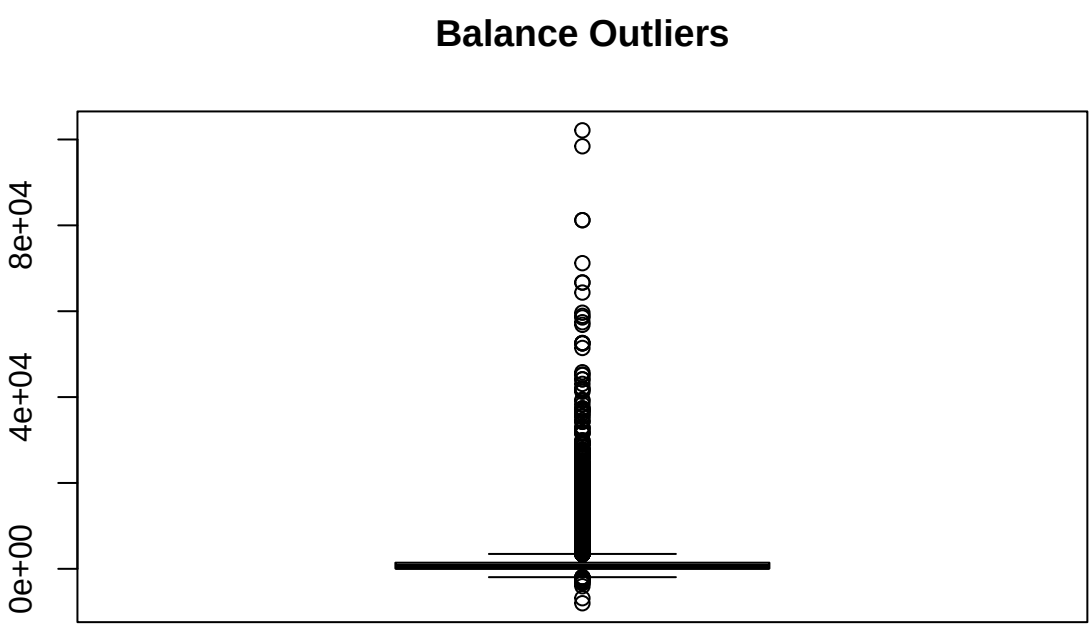


Numerical data: We see that age is right-skewed, with most clients being 25–60. Balance is highly skewed with extreme outliers, along with campaign and duration also being skewed.

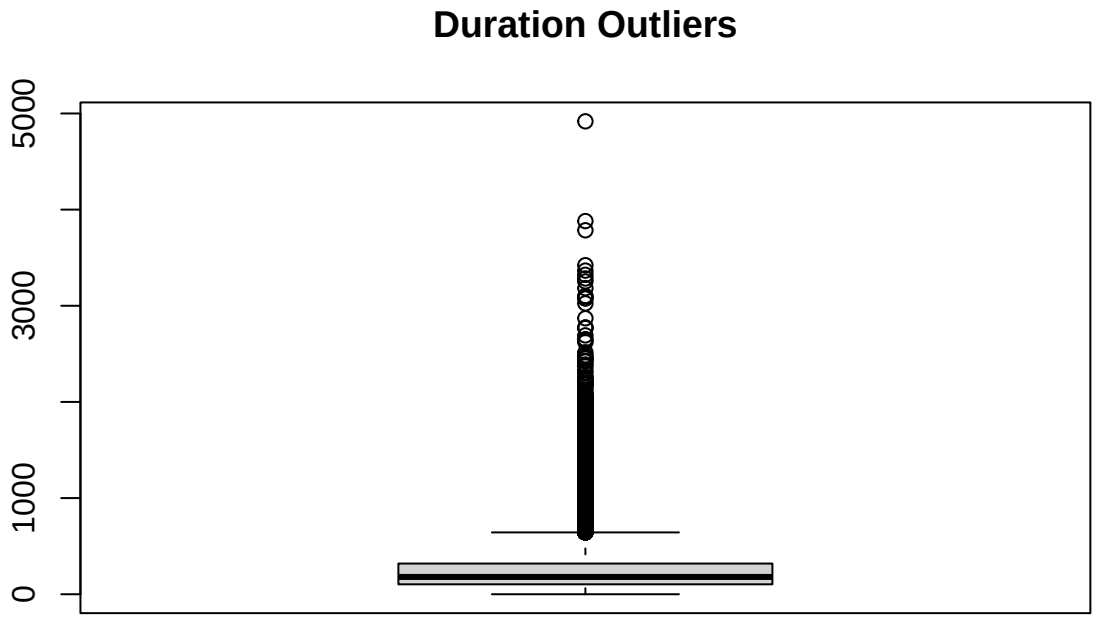
Categorical data: There is imbalance of data as the subscribe a term deposit has higher numbers in no than yes. The distribution of the features are varying in distributions whcih some higher than others.

3. Are there any outliers present?

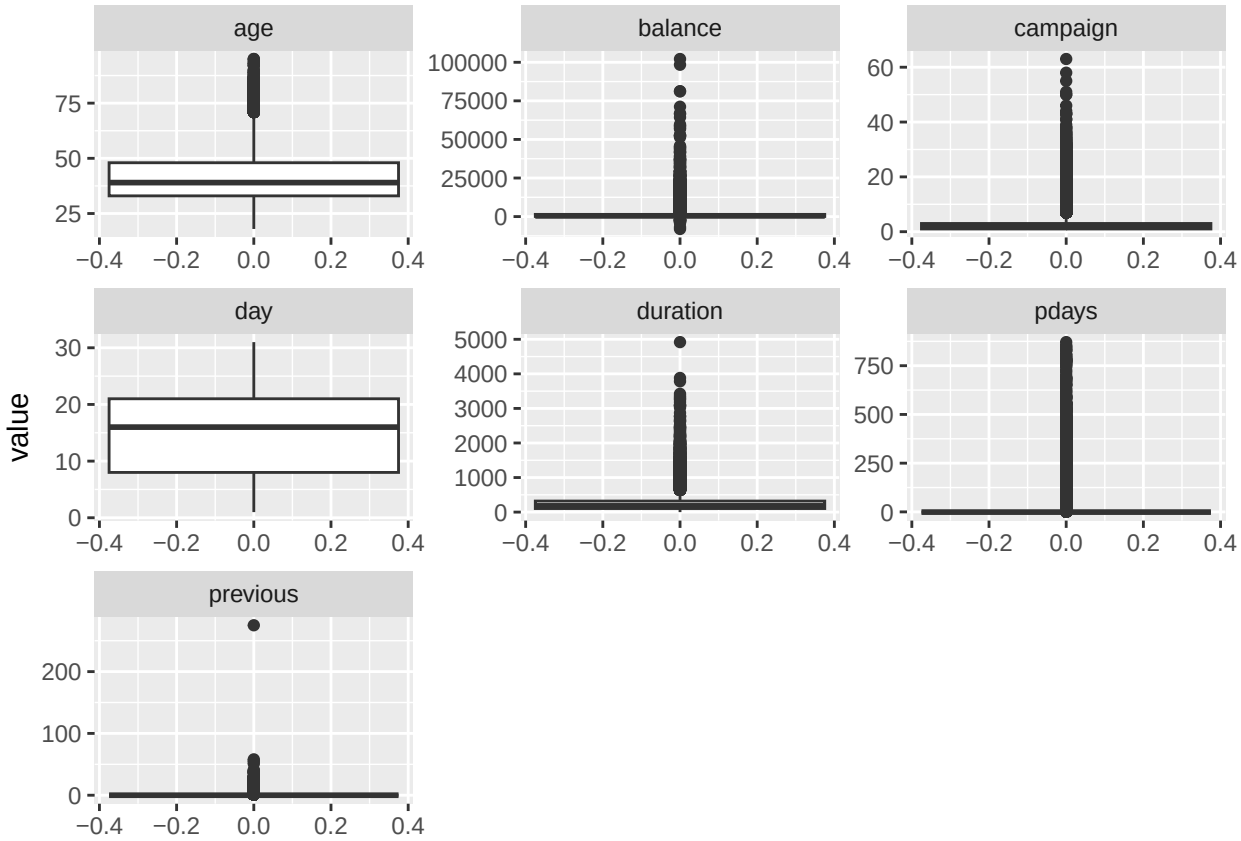
```
boxplot(bank$balance, main = "Balance Outliers")
```




```
boxplot(bank$duration, main = "Duration Outliers")
```



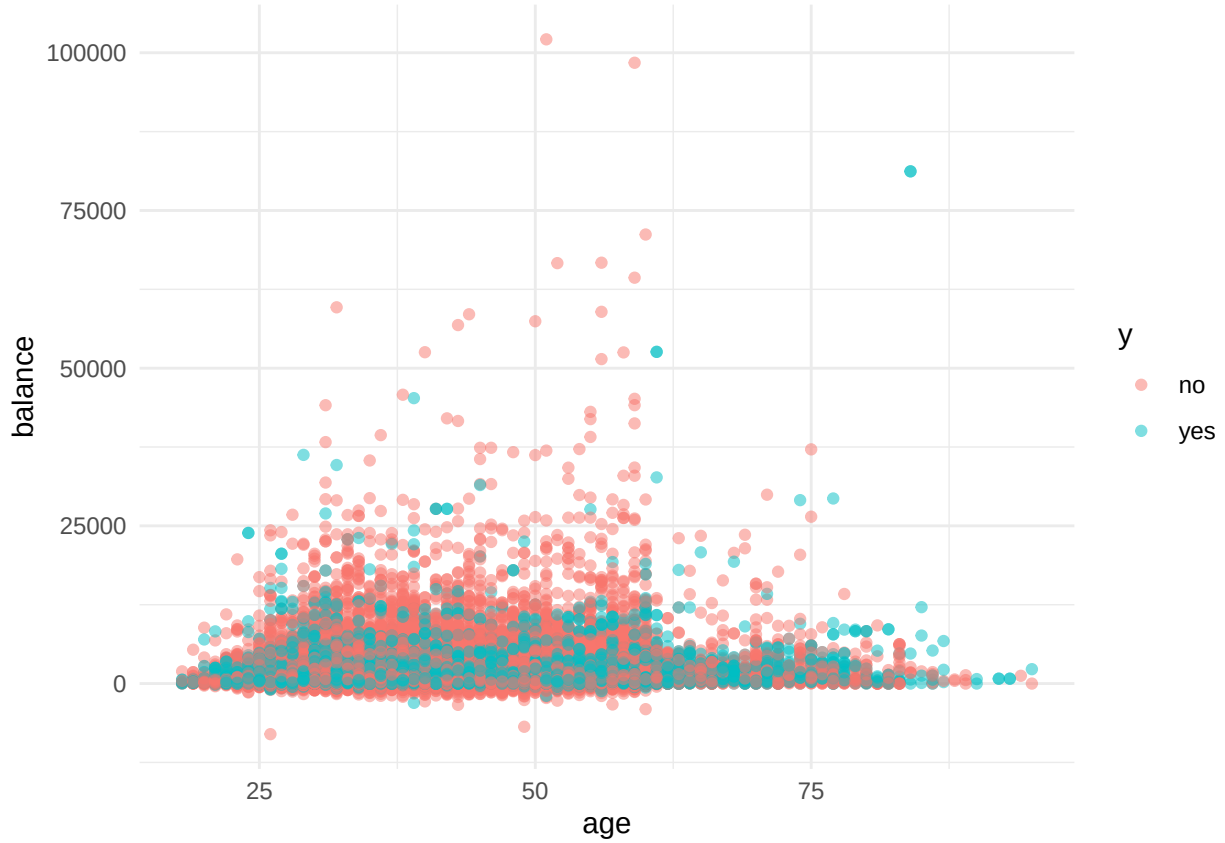

```
numeric_vars %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(y = value)) +
  facet_wrap(~name, scales = "free", ncol = 3) +
  geom_boxplot()
```



There does seem to be some outliers present. Balance and duration show extreme outliers that may affect model training.

4. What are the relationships between different variables?

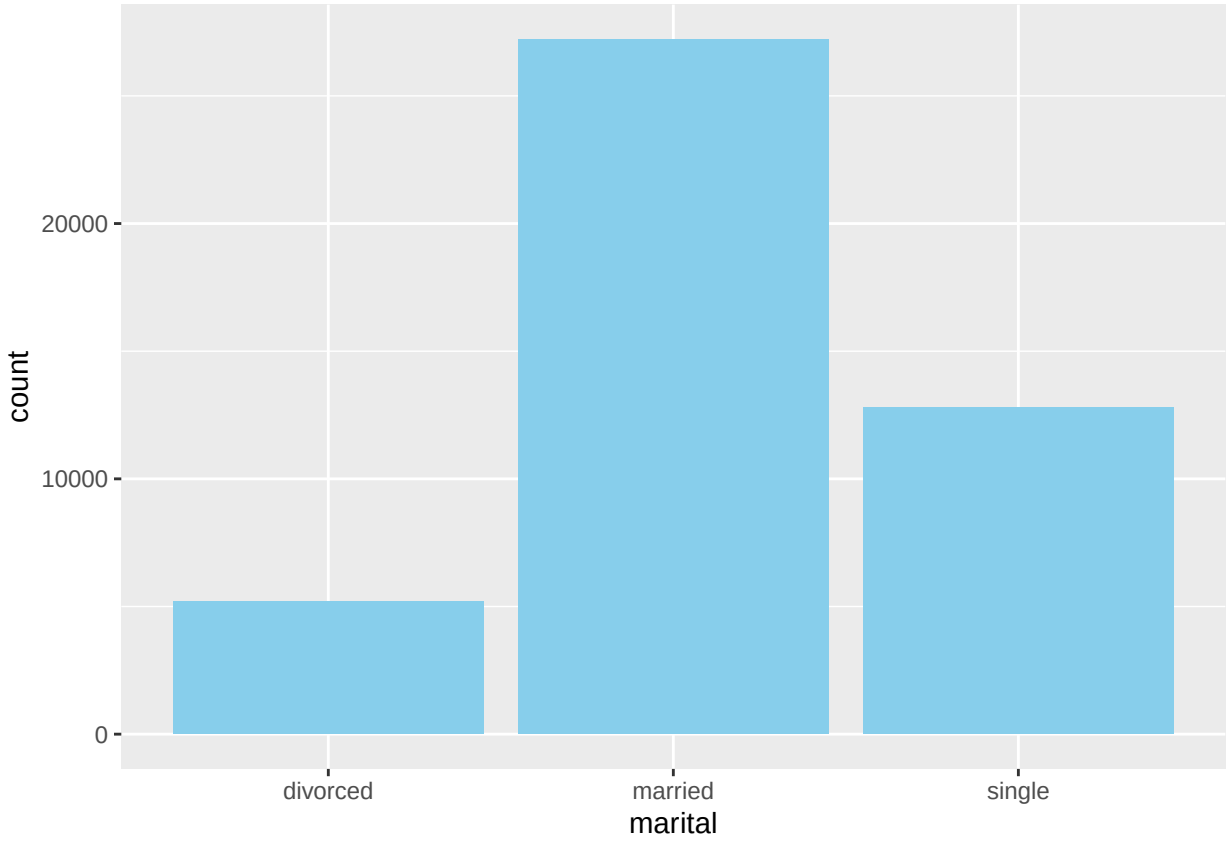
```
ggplot(bank, aes(x = age, y = balance, color = y)) +
  geom_point(alpha = 0.5) +
  theme_minimal()
```



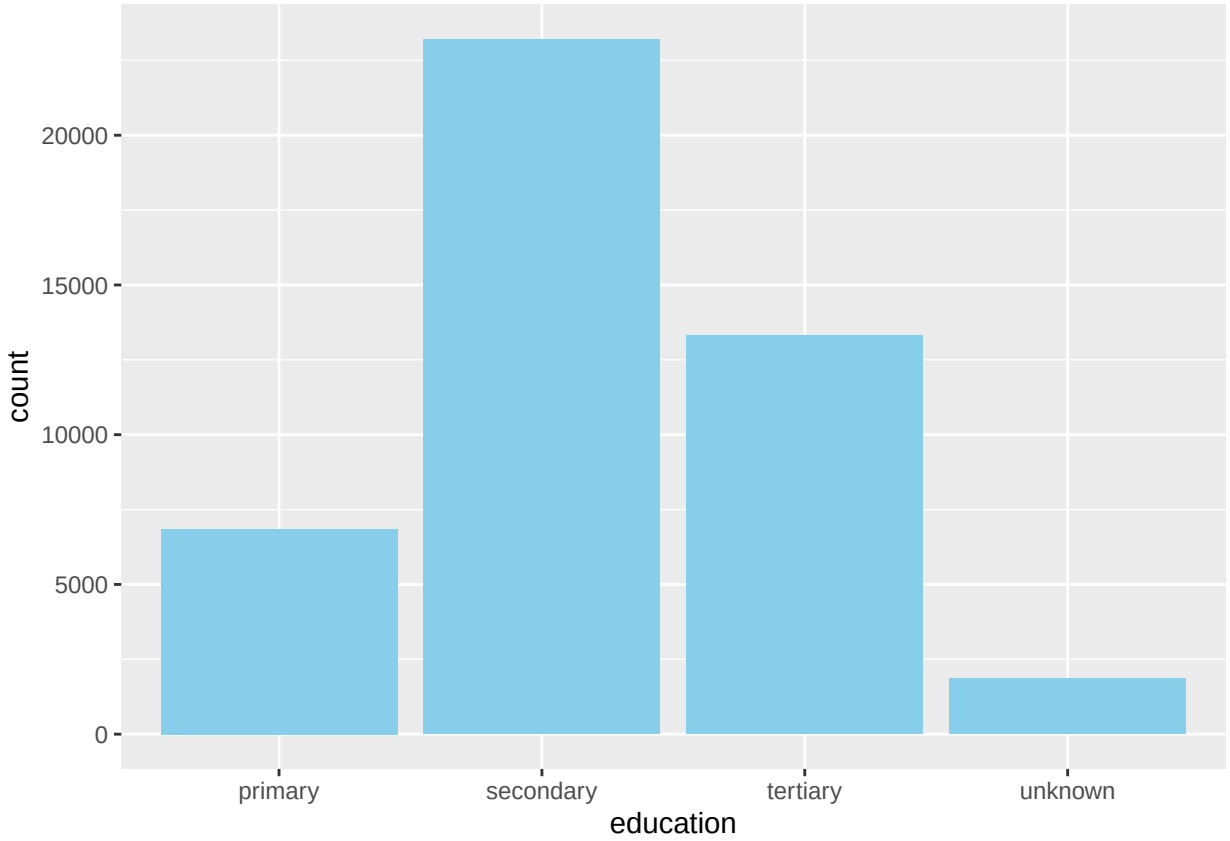
The scatterplot of age vs. balance shows that subscription decisions are not strictly determined by financial status. Younger clients (20s–30s) often have lower balances and mixed responses, while middle-aged clients (30s–50s) with moderate balances appear somewhat more likely to subscribe, possibly due to financial stability and savings goals. Older clients (60+) frequently have higher balances but show fewer "yes" outcomes, suggesting other priorities. Overall, the relationship is noisy (no single trend is distinct) indicating that age and balance alone cannot reliably predict subscription outcomes.

5. How are categorical variables distributed?

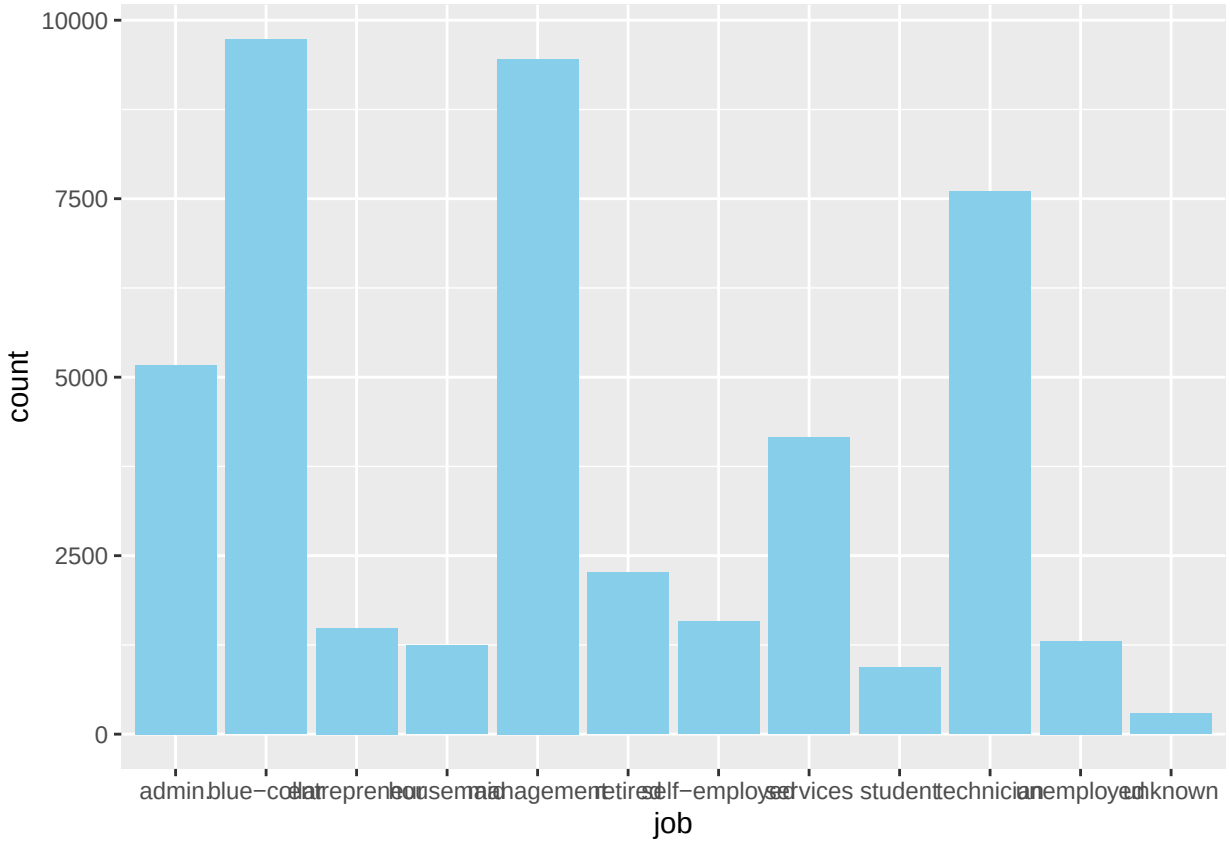
```
ggplot(bank, aes(x = marital)) + geom_bar(fill = "skyblue")
```



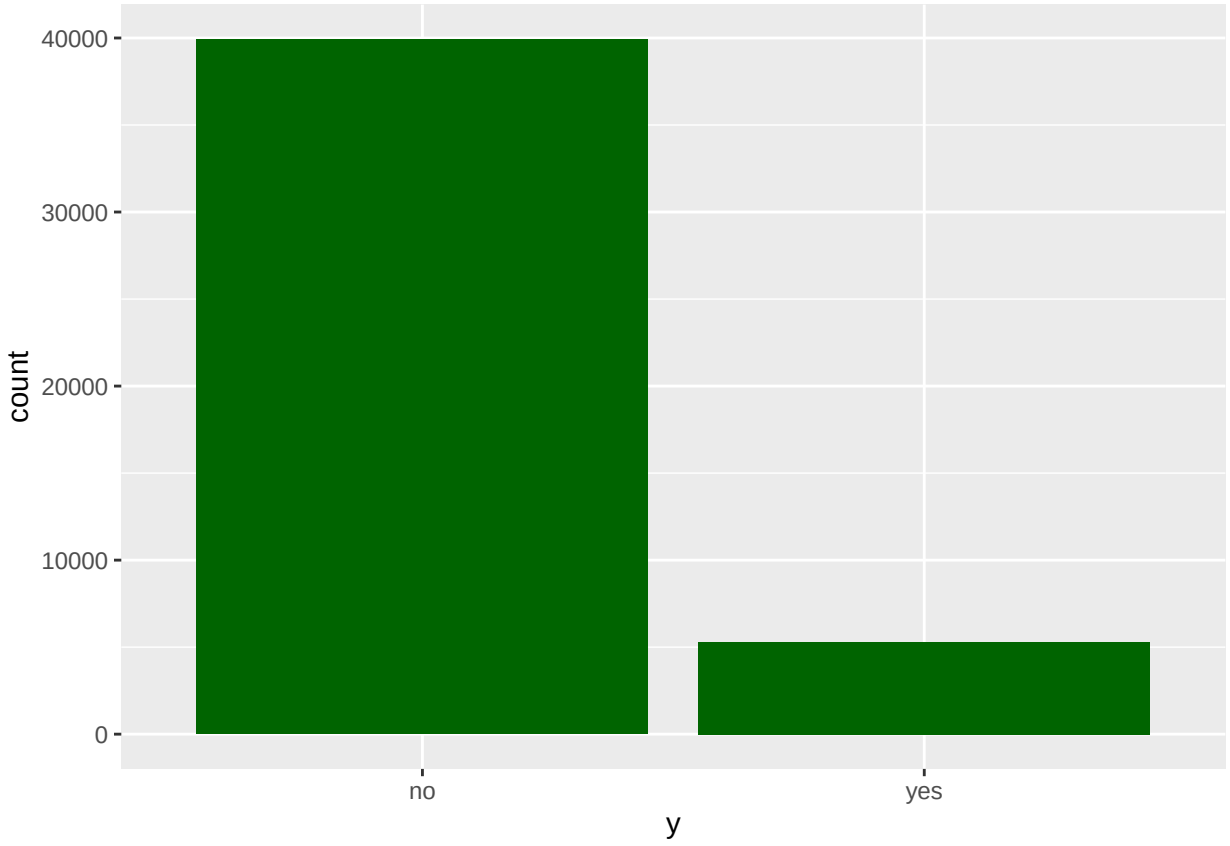

```
ggplot(bank, aes(x = education)) + geom_bar(fill = "skyblue")
```



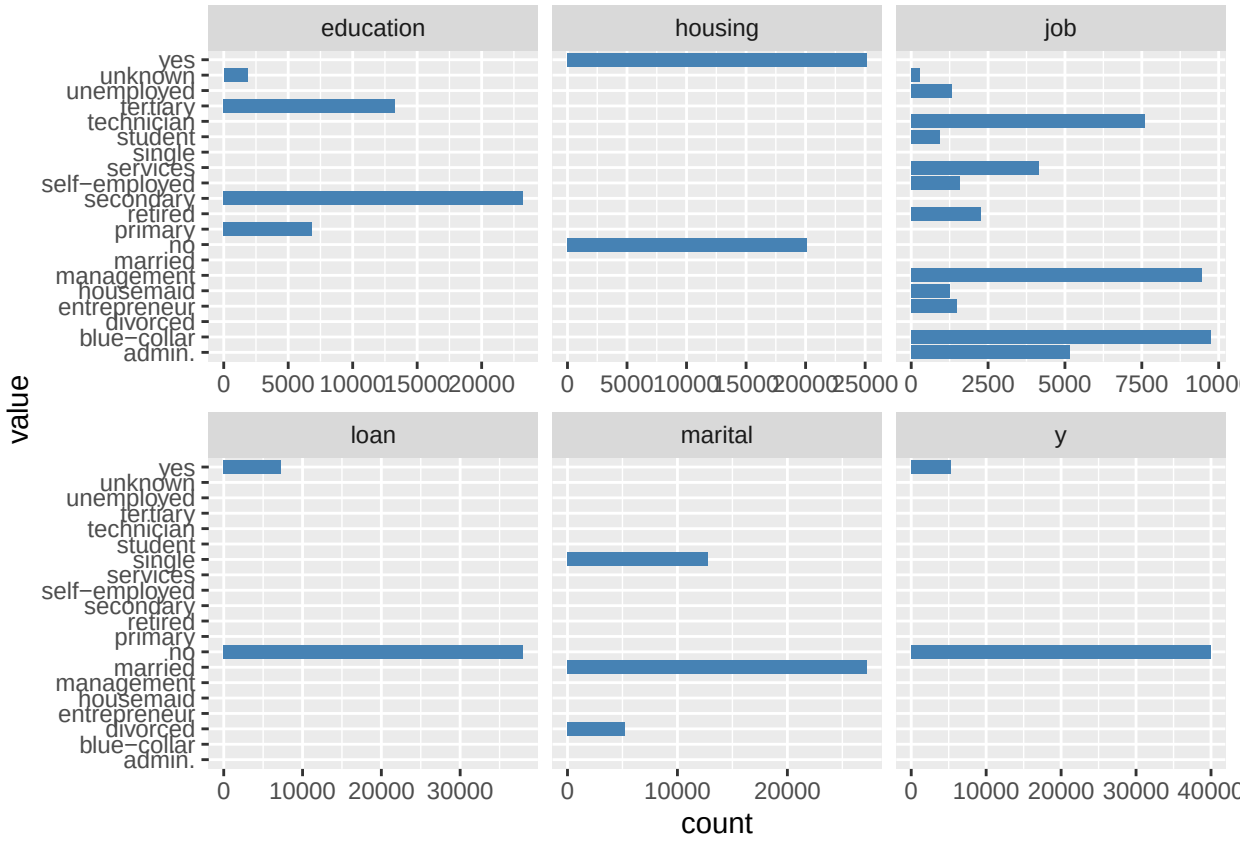

```
ggplot(bank, aes(x = job)) + geom_bar(fill = "skyblue")
```



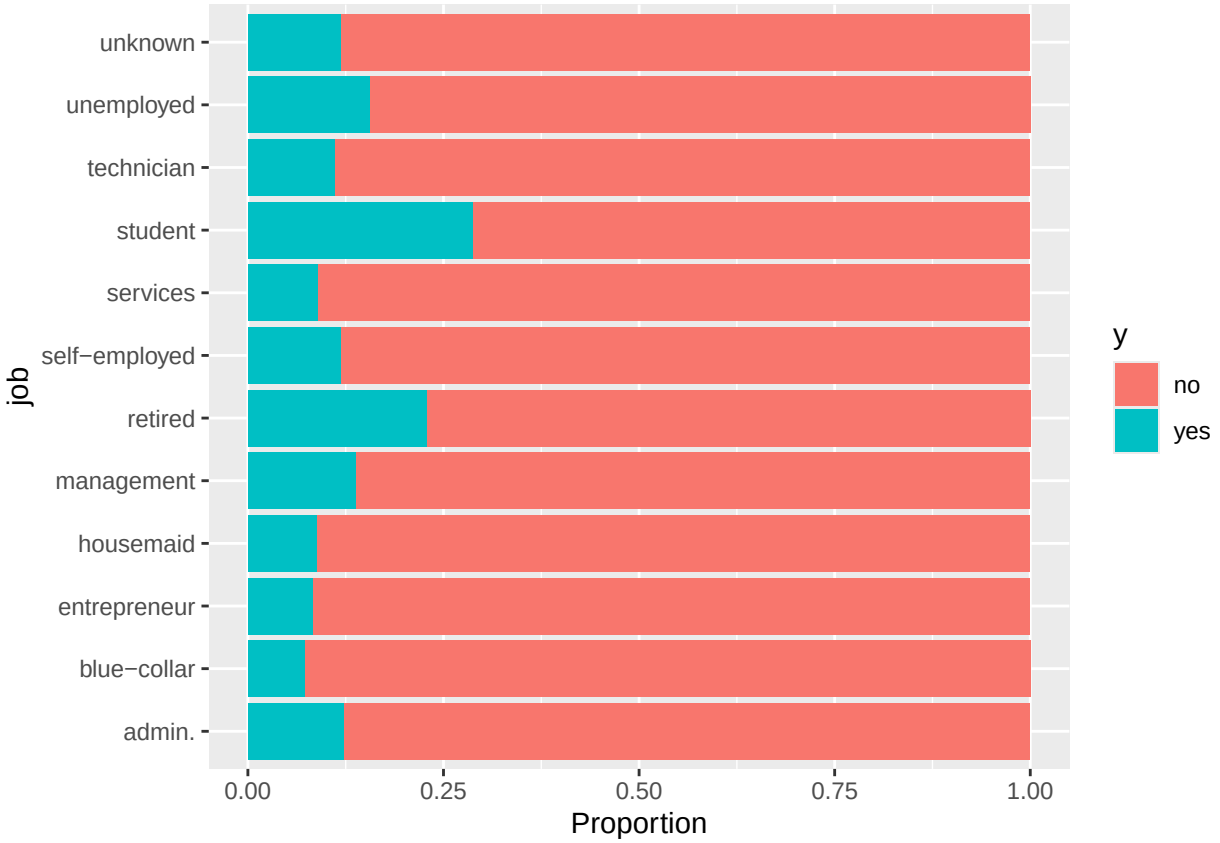

```
ggplot(bank, aes(x = y)) + geom_bar(fill = "darkgreen")
```



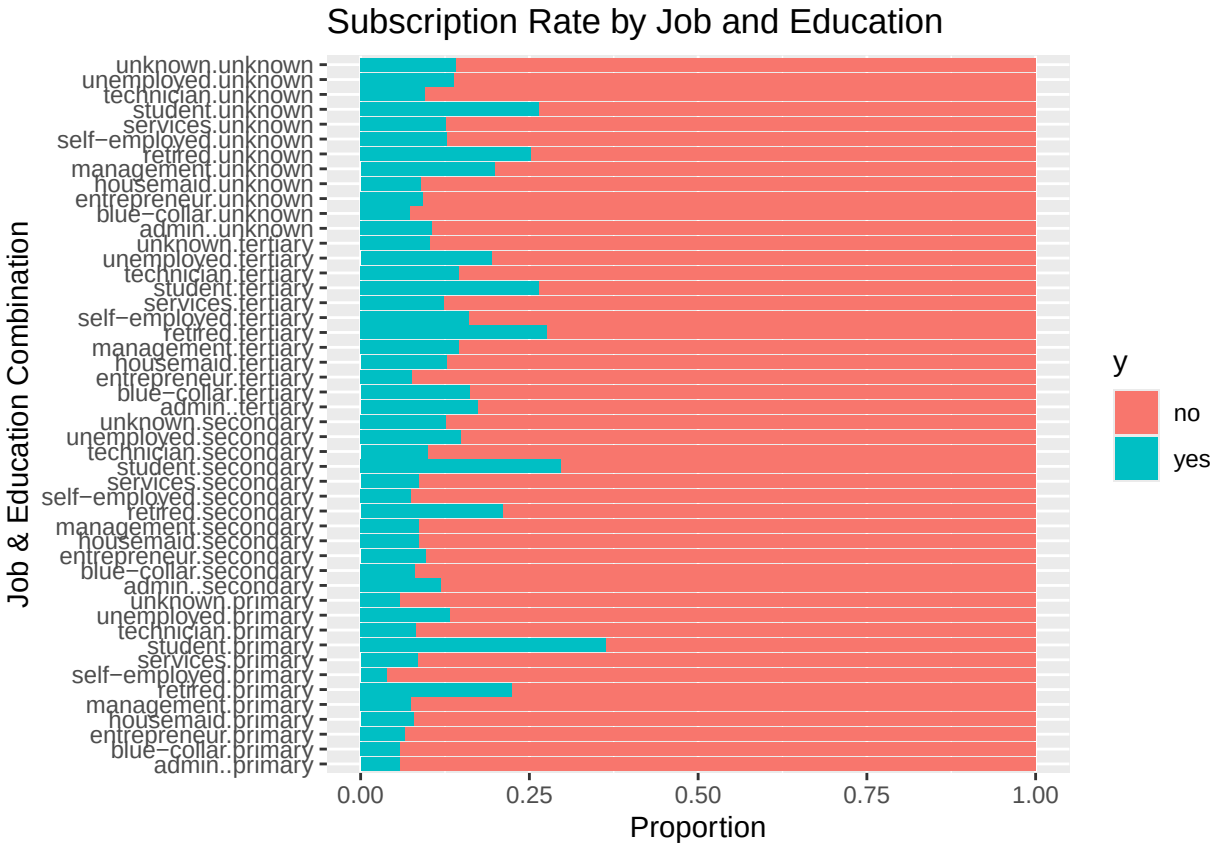

```
bank %>%
  select(job, marital, education, housing, loan, y) %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(x = value)) +
  geom_bar(fill = "steelblue") +
  facet_wrap(~name, scales = "free_x") +
  coord_flip()
```



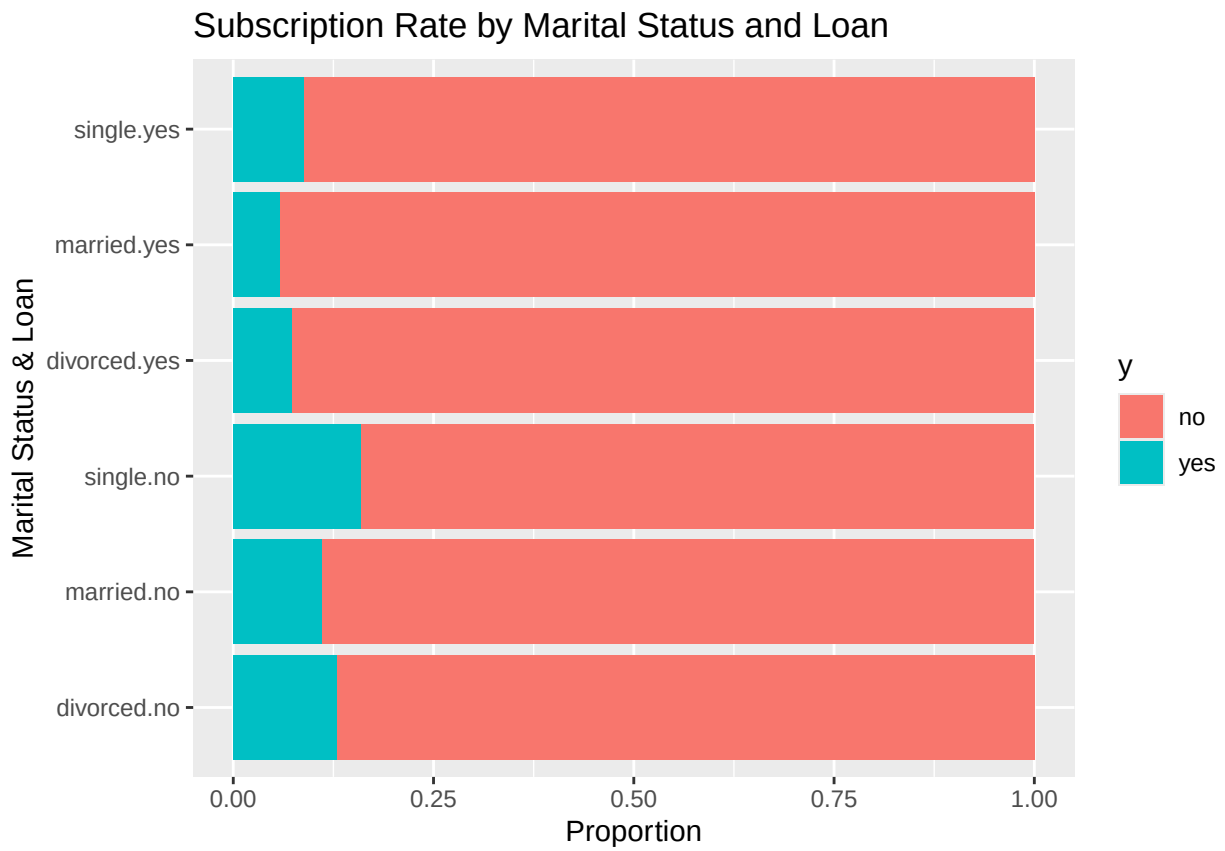

```
ggplot(bank, aes(x = job, fill = y)) +
  geom_bar(position = "fill") +
  ylab("Proportion") +
  coord_flip()
```




```
ggplot(bank, aes(x = interaction(job, education), fill = y)) +
  geom_bar(position = "fill") +
  coord_flip() +
  ylab("Proportion") +
  xlab("Job & Education Combination") +
  ggtitle("Subscription Rate by Job and Education")
```



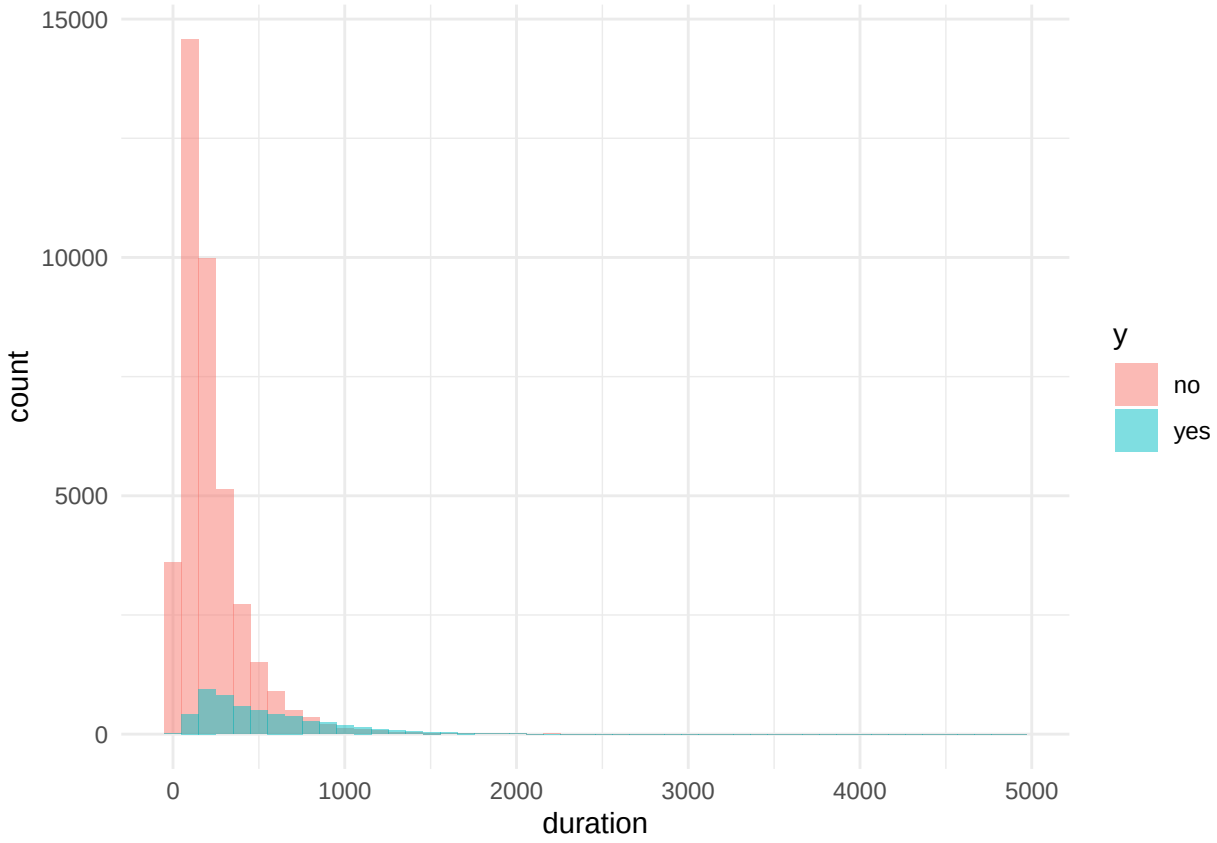

```r
ggplot(bank, aes(x = interaction(marital, loan), fill = y)) +
  geom_bar(position = "fill") +
  coord_flip() +
  ylab("Proportion") +
  xlab("Marital Status & Loan") +
  ggtitle("Subscription Rate by Marital Status and Loan")
```



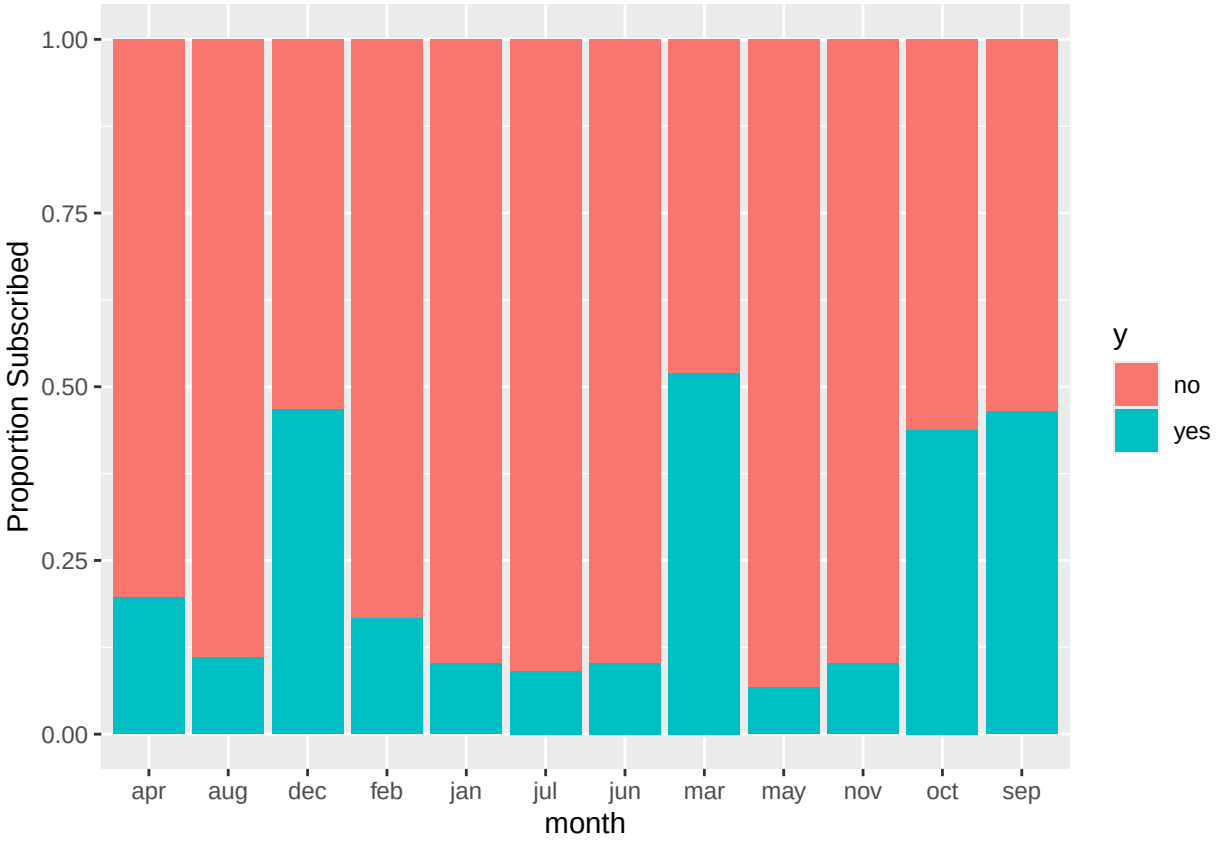
Most clients are married and have secondary education. Blue-collar, management, and technician are the most common jobs. The target variable y shows a heavy imbalance, with many more "no" responses than "yes.". There sems to be a correlation of

6. Do any patterns or trends emerge in the data?

```
ggplot(bank, aes(x = duration, fill = y)) +
  geom_histogram(bins = 50, position = "identity", alpha = 0.5) +
  theme_minimal()
```




```
ggplot(bank, aes(x = month, fill = y)) +
  geom_bar(position = "fill") +
  ylab("Proportion Subscribed")
```



Yes, we're able to see a few patterns from the data, namely:

- Call duration is strongly predictive (longer calls are much more likely to end in a "yes.")
- Middle-aged clients with stable jobs and balances appear somewhat more responsive.
- Campaign outcome is imbalanced, meaning the bank struggled to achieve a high conversion rate overall.
- There were more yes in the month of december, march, october and september than other months.

7. What is the central tendency and spread of each variable?

```
summary(numeric_vars)
```

```
##       age            balance             day            duration
##  Min.   :18.00   Min.   : -8019   Min.   : 1.00   Min.   :   0.0
##  1st Qu.:33.00   1st Qu.:    72   1st Qu.: 8.00   1st Qu.: 103.0
##  Median :39.00   Median :   448   Median :16.00   Median : 180.0
##  Mean   :40.94   Mean   :  1362   Mean   :15.81   Mean   : 258.2
##  3rd Qu.:48.00   3rd Qu.:  1428   3rd Qu.:21.00   3rd Qu.: 319.0
##  Max.   :95.00   Max.   :102127   Max.   :31.00   Max.   :4918.0
##     campaign          pdays          previous
##  Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000
##  1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000
##  Median : 2.000   Median : -1.0   Median :  0.0000
##  Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
##  3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :63.000   Max.   :871.0   Max.   :275.0000
```

The central tendency and spread of the numeric variables provide insights into the typical client profile as well as the variability in the data.

- `age`: mean about 40 years, range 18–95.

The age variable has a mean of approximately 40 years, with clients ranging from 18 to 95 years old, indicating that the dataset includes a wide age spectrum, though most clients fall within the 25–60 range.

- `balance`: median near 448 but very high max (>80,000) showing extreme outliers.

The balance variable is heavily skewed: while the median balance is around 448, some clients have extremely high balances exceeding 80,000, revealing the presence of significant outliers that could influence model performance.

- `duration`: median ~180s, but some calls last over an hour.

Similarly, duration, which measures the length of the last call in seconds, has a median value around 180 seconds, but a few calls exceed an hour, again highlighting extreme cases.

- `campaign`: median 2 contacts, max 63 (also extreme outliers).

The campaign variable, representing the number of contacts performed during this campaign for a client, has a median of 2 and a maximum of 63, showing that some clients were contacted repeatedly. Overall, these statistics indicate that most clients cluster around typical values (middle-aged, moderate balances, short calls, and few campaign contacts), but the presence of extreme outliers and wide spreads suggests that the dataset has heavy tails and variability that should be considered during modeling.

8. Are there any missing values and how significant are they?

```
summary(numeric_vars)
```

```
##       age           balance             day           duration
##  Min.   :18.00   Min.   : -8019   Min.   : 1.00   Min.   :   0.0
##  1st Qu.:33.00   1st Qu.:    72   1st Qu.: 8.00   1st Qu.: 103.0
##  Median :39.00   Median :   448   Median :16.00   Median : 180.0
##  Mean   :40.94   Mean   :  1362   Mean   :15.81   Mean   : 258.2
##  3rd Qu.:48.00   3rd Qu.:  1428   3rd Qu.:21.00   3rd Qu.: 319.0
##  Max.   :95.00   Max.   :102127   Max.   :31.00   Max.   :4918.0
##     campaign          pdays          previous
##  Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000
##  1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000
##  Median : 2.000   Median : -1.0   Median :  0.0000
##  Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
##  3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :63.000   Max.   :871.0   Max.   :275.0000
```

```
print(bank)
```

```
## # A tibble: 45,211 x 17
##      age job       marital education default balance housing loan  contact   day
##    <dbl> <chr>     <chr>   <chr>     <chr>     <dbl> <chr>   <chr> <chr>   <dbl>
##  1    58 manageme~ married tertiary  no         2143 yes     no    unknown     5
##  2    44 technici~ single  secondary no           29 yes     no    unknown     5
##  3    33 entrepre~ married secondary no            2 yes     yes   unknown     5
##  4    47 blue-col~ married unknown   no         1506 yes     no    unknown     5
##  5    33 unknown   single  unknown   no            1 no      no    unknown     5
##  6    35 manageme~ married tertiary  no          231 yes     no    unknown     5
##  7    28 manageme~ single  tertiary  no          447 yes     yes   unknown     5
##  8    42 entrepre~ divorc~ tertiary  yes           2 yes     no    unknown     5
##  9    58 retired   married primary   no          121 yes     no    unknown     5
## 10    43 technici~ single  secondary no          593 yes     no    unknown     5
## # i 45,201 more rows
## # i 7 more variables: month <chr>, duration <dbl>, campaign <dbl>, pdays <dbl>,
## #   previous <dbl>, poutcome <chr>, y <chr>
```

The dataset contains no missing values, so no imputation is required.

Assignment #2

After completing the exploratory data for the banking dataset we need to prepare our data to run through our different models. First was to check and see if there is any imbalance data and it shows there is significant imbalance data.

```
table(bank$y)
```

```
##
##    no   yes
## 39922  5289
```

```
prop.table(table(bank$y))
```

```
##
##        no       yes
## 0.8830152 0.1169848
```

To prepare out model for training we want to remove the duration of the calls the customer had. By removing this it will avoid the model from looking at them to get the result. We also added a column for easy visibility to find out if we were able to contact the client or not. Later on we split the training and test data into a 80 and 20 split.

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.4.1 --

## v broom        1.0.10     v rsample      1.3.1
## v dials        1.4.2      v tailor       0.1.0
## v infer        1.0.9      v tune         2.0.0
## v modeldata    1.5.1      v workflows    1.3.0
## v parsnip      1.3.3      v workflowsets 1.1.1
## v recipes      1.3.1      v yardstick    1.3.2
```

```
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:rsample':
##
##     calibration

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(themis)

# Drop duration (to avoid data leakage on how long the call lasted with the customer)
bank <- bank %>% select(-duration)

# Create new column called pdays_flag to indicate if the pdays are contacted or never contacted
bank <- bank %>%
  mutate(
    pdays_flag = if_else(pdays == -1, "Never Contacted", "Contacted"),
    pdays = if_else(pdays == -1, NA_real_, pdays)  # Optional
  )

# Convert character variables to factors
bank <- bank %>%
  mutate(across(where(is.character), as.factor))


# Train-test split (80/20)
set.seed(123)
split <- initial_split(bank, prop = 0.8, strata = y)
train_data <- training(split)
test_data <- testing(split)
```

We have to conduct some pre-processing for our data in which we filled in NA using the median for variables in pdays. Next we transform the categorical predictors into a dummy variable and apply the SMOTE which helps with imbalance data. Then we check to make sure the training data is balanced to run the models.

```r
#Define recipe with imputation, dummy encoding, and SMOTE for balancing
rec <- recipe(y ~ ., data = train_data) %>%
  step_impute_median(all_numeric_predictors()) %>%  # Impute NAs in numeric predictors (like pdays)
  step_dummy(all_nominal_predictors()) %>%          # Convert categorical predictors to dummy variables
  step_smote(y)                                     # Apply SMOTE to balance classes in training set

# Prep and juice the balanced training data
train_data_balanced <- rec %>%
  prep() %>%
  juice()

# Define a separate recipe for test data (no SMOTE)
rec_test <- recipe(y ~ ., data = test_data) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

test_prepped <- rec_test %>% prep() %>% juice()

# Check class balance in balanced data
print(table(train_data_balanced$y))
```

```
##
##    no   yes
## 31937 31937
```

```r
print(prop.table(table(train_data_balanced$y)))
```

```
##
##  no yes
## 0.5 0.5
```

We are adding performance metrics for us to check later on how each model is performing. The metrics being used are: - Accuracy - Predicts how well our model is performing overall - Precision - For those labeled as positive how many are actually positive - Recall - Of all the actual positives. How many did we find - F1-Score - This is the balance between precision and recalls. We want to have a balance between false positives and false negatives.

```r
library(yardstick)

my_metrics <- metric_set( accuracy, yardstick::precision, yardstick::recall, f_meas )
```

Model 1 : Baseline Decision Tree

Objective: The goal of this experiment was to establish a baseline using a Decision Tree classifier with default settings. Starting with this simple and interpretable model allows us to better understand the initial performance before applying any complexity or tuning.

The results from the baseline model show strong performance. It achieved an accuracy of 89%, indicating that it correctly classified a high proportion of the test data overall. Additionally, the precision, recall, and F1-score are all above 85%, suggesting the model is effective at identifying positive cases, with few false positives and a good balance between precision and recall.

In the banking data there was imbalance data which I needed to clean up before running the models. Without cleaning up the imbalance in the data there would have been a high bias towards the majority which the

results would not truly shows how the data is. With decision trees they are simple and have potential to miss patterns. We

```
#Decision Tree (Baseline)
library(rpart)
```

```
##
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':
##
##     prune
```

```
library(rpart.plot)

# Model specification
dt_model <- decision_tree(mode = "classification") %>%
  set_engine("rpart")

# Fit model
dt_fit <- dt_model %>%
  fit(y ~ ., data = train_data_balanced)

# Predict
dt_preds <- predict(dt_fit, train_data_balanced, type = "prob") %>%
  bind_cols(predict(dt_fit, train_data_balanced)) %>%
  bind_cols(train_data_balanced %>% select(y))

# Evaluate
# Calculate class-based metrics (accuracy, precision, recall, f1-score)
dt_class_metrics <- my_metrics(
  data = dt_preds,
  truth = y,
  estimate = .pred_class
)

# View results
print(dt_class_metrics)
```

```
## # A tibble: 4 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 accuracy  binary         0.899
## 2 precision binary         0.863
## 3 recall    binary         0.949
## 4 f_meas    binary         0.904
```

```
# Baseline DT on training data
dt_preds_train <- predict(dt_fit, new_data = train_data_balanced, type = "prob") %>%
  bind_cols(predict(dt_fit, new_data = train_data_balanced)) %>%
  bind_cols(train_data_balanced %>% select(y))
```

```r
dt_train_metrics <- my_metrics(
  data = dt_preds_train,
  truth = y,
  estimate = .pred_class
)
dt_train_metrics <- dt_train_metrics %>%
  mutate(model = "Baseline_Train")
```

Cross Validation for Baseline Decision tree with 5 fold. Instead of relying on the 80/20 split in the data we used a 5 fold cross validation.

This will allow me to check the performance of my model and detect any overfitting. The results of this shows it has a great performance and at times even higher in some areas compared to the baseline model. There is high accuracy at 88% for the model.

```r
set.seed(123)
cv_folds <- vfold_cv(train_data, v = 5, strata = y)

# Use workflows to bundle model + recipe
dt_workflow <- workflow() %>%
  add_model(dt_model) %>%
  add_recipe(rec)

# Fit with resamples
dt_res <- fit_resamples(
  dt_workflow,
  resamples = cv_folds,
  metrics = my_metrics
)

collect_metrics(dt_res)
```

```
## # A tibble: 4 x 6
##   .metric   .estimator  mean     n  std_err .config
##   <chr>     <chr>      <dbl> <int>    <dbl> <chr>
## 1 accuracy  binary     0.882     5 0.00137  pre0_mod0_post0
## 2 f_meas    binary     0.934     5 0.000774 pre0_mod0_post0
## 3 precision binary     0.920     5 0.000911 pre0_mod0_post0
## 4 recall    binary     0.949     5 0.00122  pre0_mod0_post0
```

Model 2 - Decision Tree, depth 10 and split 4 Objective: The goal for the second decision tree is to increase the depth and split of the model which would increase performance and capture more complex patterns in the data. We increased the depth to 10 and increased the split at 4 which should increase the performance of the decision model.

We evaluated the model using accuracy, precision, recall, and F1-score, comparing the results to those of the baseline decision tree. The metrics showed that the model performed quite well. To check for overfitting, I compared the evaluation metrics on both the training and test data. Since the results were fairly consistent across both sets, it suggests that the model is not overfitting.

In comparison to the default Decision tree the results show the accuracy, precision, recall and f1-score was higher than the baseline Decision tree making it a better model.

```r
#Decision Tree (10 depth and 4 split)
dt_model_deep <- decision_tree(mode = "classification") %>%
  set_engine("rpart", control = rpart.control(maxdepth = 10, minsplit = 4, cp = 0))

# Fit on balanced training data
dt_fit_deep <- dt_model_deep %>%
  fit(y ~ ., data = train_data_balanced)

# Predict on test data (preprocessed without SMOTE)
dt_preds_deep <- predict(dt_fit_deep, new_data = test_prepped, type = "prob") %>%
  bind_cols(predict(dt_fit_deep, new_data = test_prepped)) %>%
  bind_cols(test_prepped %>% select(y))

# Evaluate on test data
dt_metrics_deep <- my_metrics(
  data = dt_preds_deep,
  truth = y,
  estimate = .pred_class
) %>%
  mutate(model = "DT (depth=10, split=4)_Test")

print(dt_metrics_deep)
```

```
## # A tibble: 4 x 4
##   .metric   .estimator .estimate model
##   <chr>     <chr>          <dbl> <chr>
## 1 accuracy  binary         0.881 DT (depth=10, split=4)_Test
## 2 precision binary         0.922 DT (depth=10, split=4)_Test
## 3 recall    binary         0.945 DT (depth=10, split=4)_Test
## 4 f_meas    binary         0.934 DT (depth=10, split=4)_Test
```

```r
# Predict on balanced training data
dt_preds_deep_train <- predict(dt_fit_deep, new_data = train_data_balanced, type = "prob") %>%
  bind_cols(predict(dt_fit_deep, new_data = train_data_balanced)) %>%
  bind_cols(train_data_balanced %>% select(y))

# Evaluate on training data
dt_deep_train_metrics <- my_metrics(
  data = dt_preds_deep_train,
  truth = y,
  estimate = .pred_class
) %>%
  mutate(model = "DT (depth=10, split=4)_Train")

print(dt_deep_train_metrics)
```

```
## # A tibble: 4 x 4
##   .metric   .estimator .estimate model
##   <chr>     <chr>          <dbl> <chr>
## 1 accuracy  binary         0.916 DT (depth=10, split=4)_Train
## 2 precision binary         0.890 DT (depth=10, split=4)_Train
## 3 recall    binary         0.950 DT (depth=10, split=4)_Train
## 4 f_meas    binary         0.919 DT (depth=10, split=4)_Train
```

```r
# Combine everything
all_metrics_full <- bind_rows(
  dt_class_metrics %>% mutate(model = "Baseline_Test"),
  dt_metrics_deep %>% mutate(model = "DT (depth=3, split=2)_Test"),
  dt_train_metrics,
  dt_deep_train_metrics
)

metrics_wide_all <- all_metrics_full %>%
  select(.metric, .estimate, model) %>%
  pivot_wider(names_from = model, values_from = .estimate)

print(metrics_wide_all)
```

```
## # A tibble: 4 x 5
##   .metric   Baseline_Test `DT (depth=3, split=2)_Test` Baseline_Train
##   <chr>             <dbl>                        <dbl>          <dbl>
## 1 accuracy          0.899                        0.881          0.899
## 2 precision         0.863                        0.922          0.863
## 3 recall            0.949                        0.945          0.949
## 4 f_meas            0.904                        0.934          0.904
## # i 1 more variable: `DT (depth=10, split=4)_Train` <dbl>
```

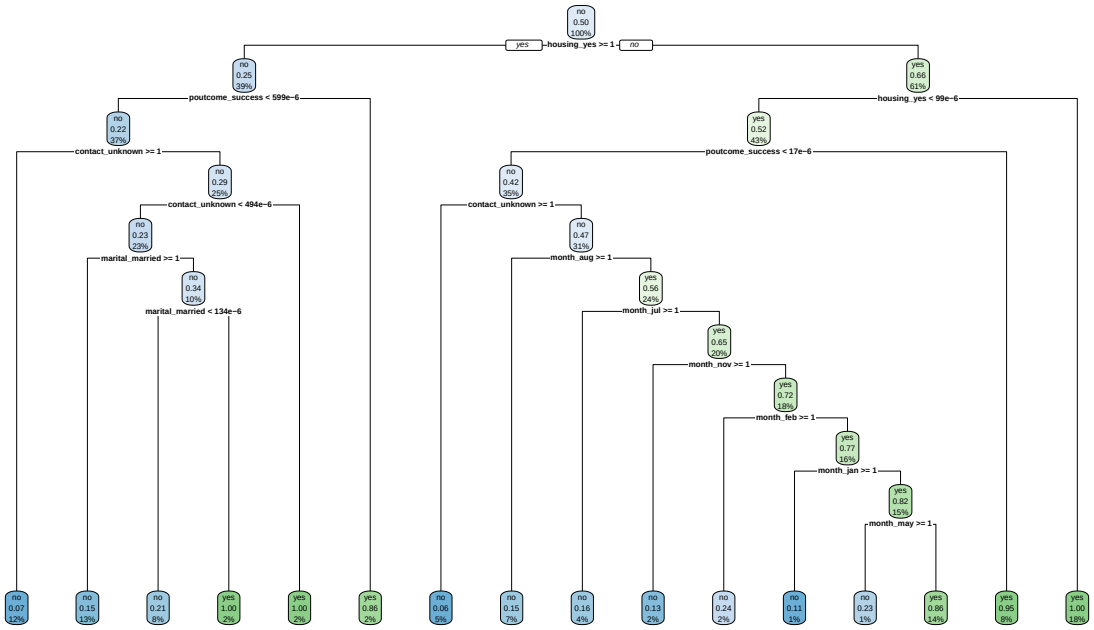
To better understand the change in parameters in the decision tree we visually showed the difference on how the tree would look like.

```r
# Extract rpart fit object from parsnip fit
rpart_base <- dt_fit$fit
rpart_deep <- dt_fit_deep$fit

rpart.plot(rpart_base, main = "Decision Tree (Baseline)")
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```

Decision Tree (Baseline)

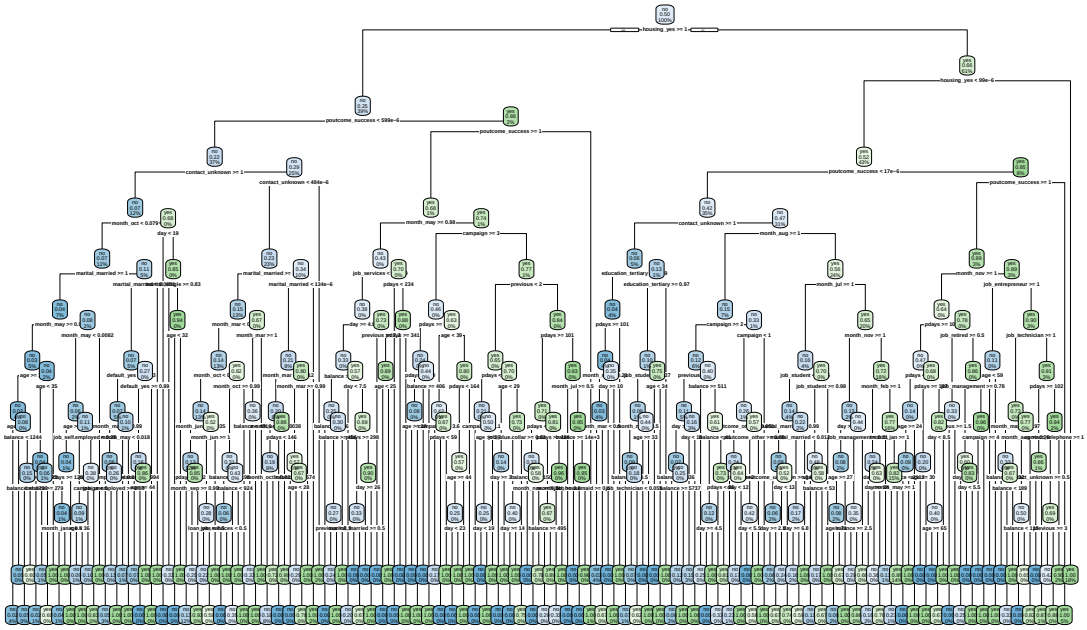



```
rpart.plot(rpart_deep, main = "Decision Tree (10 depth and 4 minisplit)")
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

Decision Tree (10 depth and 4 minisplit)



Model 3 - Random Forest 100 trees Objective: The objective of this experiment was to evaluate the performance of a Random Forest model with a limited number of 100 trees. This serves as a benchmark to see how well a Random Forest performs compared to the Decision Tree baseline, while keeping the number of trees relatively low due to computational constraints.

Overall the performance is great since the results are all 89% or higher which even outperforms the decision tree we conducted in the earlier models. The results for precision, recall and f1-score is high as well.

```
#Random Forest(100 tree)
rf_model <- rand_forest(mode = "classification", trees = 100) %>%
  set_engine("ranger")

# Fit model on balanced training data
rf_fit <- rf_model %>%
  fit(y ~ ., data = train_data_balanced)

#  Predict on test data
rf_preds <- predict(rf_fit, new_data = test_prepped, type = "prob") %>%
  bind_cols(predict(rf_fit, new_data = test_prepped)) %>%
  bind_cols(test_prepped %>% select(y))

# Evaluate on test data
rf_metrics <- my_metrics(
  data = rf_preds,
  truth = y,
  estimate = .pred_class
) %>%
```

```
  mutate(model = "Random Forest_100 tree_Test")

print(rf_metrics)
```

```
## # A tibble: 4 x 4
##   .metric   .estimator .estimate model
##   <chr>     <chr>          <dbl> <chr>
## 1 accuracy  binary         0.893 Random Forest_100 tree_Test
## 2 precision binary         0.905 Random Forest_100 tree_Test
## 3 recall    binary         0.982 Random Forest_100 tree_Test
## 4 f_meas    binary         0.942 Random Forest_100 tree_Test
```

Model 4 - Random Forest , 500 trees and 4 features Objective: The objective is to find out if increasing the number of trees and limiting the number of features considered at each split would lead to better performance. This tests the hypothesis that more trees provide better generalization, and that feature reduction may enhance model efficiency.

The results of this model shows it performed slightly better than the Random Forest model (100 tree) in accuracy, recall and f1-score. The precision score was slightly lower but it is still a pretty high amount. This is the best model so far compared to the results of the two decision tree and with the increase in trees this model was better than the Random Forest -100 trees.

```
# Random Forest (Tuned: 500 trees, mtry = 4)
rf_model_tuned <- rand_forest(mode = "classification", trees = 500, mtry = 4) %>%
  set_engine("ranger")

# Fit model on balanced training data
rf_fit_tuned <- rf_model_tuned %>%
  fit(y ~ ., data = train_data_balanced)

# === Predict on test data ===
rf_preds_tuned <- predict(rf_fit_tuned, new_data = test_prepped, type = "prob") %>%
  bind_cols(predict(rf_fit_tuned, new_data = test_prepped)) %>%
  bind_cols(test_prepped %>% select(y))

# Evaluate on test data
rf_metrics_tuned <- my_metrics(
  data = rf_preds_tuned,
  truth = y,
  estimate = .pred_class
) %>%
  mutate(model = "Random Forest_Tuned_Test")

print(rf_metrics_tuned)
```

```
## # A tibble: 4 x 4
##   .metric   .estimator .estimate model
##   <chr>     <chr>          <dbl> <chr>
## 1 accuracy  binary         0.895 Random Forest_Tuned_Test
## 2 precision binary         0.904 Random Forest_Tuned_Test
## 3 recall    binary         0.986 Random Forest_Tuned_Test
## 4 f_meas    binary         0.943 Random Forest_Tuned_Test
```

Model 5 - Adaboost, 1000 random rows, 50 decision tree Objective: The goal of this experiment was to establish a reference point for AdaBoost performance using a smaller subset of the dataset due to computational limitations. Since we couldn't run AdaBoost on the full dataset, we randomly sampled 1000 rows and trained the model with 50 decision trees. This acts as a starting point for tuning and comparison.

The results shows we have it has low level of accuracy at 66% it is because we are only using a subset of the data which indicates high bias and with only 50 iterations the variance is low since this is reducing the complexity of the data.

```
library(forcats)
library(fastAdaboost)

# Small balanced sample subset for training
set.seed(123)
train_small <- train_data %>%
  group_by(y) %>%
  sample_n(1000) %>%
  ungroup()

# Lump factor levels to reduce complexity (keep top 5 levels)
train_small <- train_small %>%
  mutate(across(where(is.factor), ~ fct_lump(.x, n = 5)))

# Prepare test data by keeping top 5 level
test_small <- test_data %>%
  mutate(across(where(is.factor), ~ fct_lump(.x, n = 5)))

# Align factor levels in test to train
factor_cols <- names(Filter(is.factor, train_small))
for (col in factor_cols) {
  test_small[[col]] <- factor(test_small[[col]], levels = levels(train_small[[col]]))
}

# Remove rows with NA in any factor columns due to unmatched levels
test_small <- test_small %>% filter(if_all(all_of(factor_cols), ~ !is.na(.)))

# Ensure target is factor with same levels
train_small$y <- factor(train_small$y)
test_small$y <- factor(test_small$y, levels = levels(train_small$y))

# Convert to plain data.frames (fastAdaboost sometimes expects this)
train_small <- as.data.frame(train_small)
test_small <- as.data.frame(test_small)

# Train fastAdaboost model
adaboost_model <- fastAdaboost::adaboost(y ~ ., data = train_small, nIter = 50)

# Predict on test data
adaboost_preds <- predict(adaboost_model, newdata = test_small)

# Evaluate
res <- tibble(
  truth = test_small$y,
  estimate = factor(adaboost_preds$class, levels = levels(test_small$y))
```

```
)

metrics <- metric_set(accuracy, yardstick::precision, yardstick::recall, f_meas)
print(metrics(res, truth = truth, estimate = estimate))
```

```
## # A tibble: 4 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 accuracy  binary         0.661
## 2 precision binary         0.937
## 3 recall    binary         0.660
## 4 f_meas    binary         0.775
```

Model 6 - Adaboost, 2000 sample rows, 150 decision tree Objective: The objective is to improve the previous AdaBoost model by increasing both the sample size and the number of decision trees. The hypothesis was that more data and more boosting rounds would help capture more patterns and improve model performance.

The results show the performance increased in accuracy, precision, recall and f1-score when we increased the sample size and the iterations to 150. The performance did not have a significant increase even with increase in sample size and the decision tree numbers. We may have to make more changes in order to get better performance using Adaboost.

```
# Create a larger, balanced training sample
set.seed(123)
train_small2 <- train_data %>%
  group_by(y) %>%
  sample_n(2000) %>%
  ungroup()

# Lump rare factor levels into 'Other' (keep top 5)
train_small2 <- train_small2 %>%
  mutate(across(where(is.factor), ~ fct_lump(.x, n = 5)))

# Prepare test data the same way
test_small2 <- test_data %>%
  mutate(across(where(is.factor), ~ fct_lump(.x, n = 5)))

# Align factor levels in test to match train
factor_cols <- names(Filter(is.factor, train_small2))
for (col in factor_cols) {
  test_small2[[col]] <- factor(test_small2[[col]], levels = levels(train_small2[[col]]))
}

# Remove rows with NA values caused by mismatched levels
test_small2 <- test_small2 %>%
  filter(if_all(all_of(factor_cols), ~ !is.na(.)))

# Make sure the target column 'y' is a factor and aligned
train_small2$y <- factor(train_small2$y)
test_small2$y <- factor(test_small2$y, levels = levels(train_small2$y))

# Convert to plain data.frames (required by fastAdaboost)
train_small2 <- as.data.frame(train_small2)
```

```r
test_small2 <- as.data.frame(test_small2)

# Train fastAdaboost model with increased iterations
adaboost_model2 <- fastAdaboost::adaboost(y ~ ., data = train_small2, nIter = 150)

# Predict on test data
adaboost_preds2 <- predict(adaboost_model2, newdata = test_small2)

# Evaluate
res2 <- tibble(
  truth = test_small2$y,
  estimate = factor(adaboost_preds2$class, levels = levels(test_small2$y))
)


metrics2 <- metric_set(accuracy, yardstick::precision, yardstick::recall, f_meas)
print(metrics2(res, truth = truth, estimate = estimate))
```

```
## # A tibble: 4 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 accuracy  binary         0.661
## 2 precision binary         0.937
## 3 recall    binary         0.660
## 4 f_meas    binary         0.775
```

Based on all six experiments, the Random Forest (500 trees and 4 features) proved to be the most effective. It achieved the best overall metrics, demonstrated strong generalization, and effectively balanced precision and recall. Therefore, I recommend using the optimized Random Forest as the final model. Below the the models and metrics together.

```r
# Combine all model metrics
final_metrics <- bind_rows(
  dt_class_metrics %>% mutate(model = "Baseline_DT_Test"),
  dt_metrics_deep %>% mutate(model = "DT_Depth10_Split4_Test"),
  rf_metrics %>% mutate(model = "RF_100_Trees"),
  rf_metrics_tuned %>% mutate(model = "RF_500_Trees_Mtry4"),
  metrics(res, truth = truth, estimate = estimate) %>% mutate(model = "AdaBoost_50_1000"),
  metrics2(res2, truth = truth, estimate = estimate) %>% mutate(model = "AdaBoost_150_2000")
)

print(final_metrics)
```

```
## # A tibble: 24 x 4
##    .metric   .estimator .estimate model
##    <chr>     <chr>          <dbl> <chr>
##  1 accuracy  binary         0.899 Baseline_DT_Test
##  2 precision binary         0.863 Baseline_DT_Test
##  3 recall    binary         0.949 Baseline_DT_Test
##  4 f_meas    binary         0.904 Baseline_DT_Test
##  5 accuracy  binary         0.881 DT_Depth10_Split4_Test
##  6 precision binary         0.922 DT_Depth10_Split4_Test
```

```
##  7 recall    binary         0.945 DT_Depth10_Split4_Test
##  8 f_meas    binary         0.934 DT_Depth10_Split4_Test
##  9 accuracy  binary         0.893 RF_100_Trees
## 10 precision binary         0.905 RF_100_Trees
## # i 14 more rows
```