KPMG Data Analysis

Ben Crabtree

18/01/2020

KPMG Data Analysis

Customer demographic and transaction data sets provided Sprocket Central Pty Ltd. They would like to predict high value customers to help optimise their marketing strategy.

Data Quality Assessment

```r
library(tidyverse)
library(readxl)
```

```r
xl_data = 'KPMG_VI_New_raw_data_update_final.xlsx'

# Names of sheets in excel file
excel_sheets(path = xl_data)
```

```
## [1] "Title Sheet"          "Transactions"        "NewCustomerList"
## [4] "CustomerDemographic" "CustomerAddress"
```

Import each sheet as a dataframe, making sure to read in empty strings or variants of NA as R's NA value for consistency:

```r
df_transactions = read_excel(path = xl_data, sheet = "Transactions", skip = 1, na = c("NA",""," ", "n/a
df_new_customer_list = read_excel(path = xl_data, sheet = "NewCustomerList", skip = 1, na = c("NA","",""
```

```
## New names:
## * `` -> ...17
## * `` -> ...18
## * `` -> ...19
## * `` -> ...20
## * `` -> ...21
```

```r
df_customer_demographic = read_excel(path = xl_data, sheet = "CustomerDemographic", skip = 1, na = c("N
df_customer_address = read_excel(path = xl_data, sheet = "CustomerAddress", skip = 1, na = c("NA",""," 
```

Customer Addresses Data Quality

Dataframe with 3999 observations and 6 variables.

```r
names(df_customer_address)
```

```
## [1] "customer_id"        "address"            "postcode"
## [4] "state"              "country"            "property_valuation"
```

```r
dim(df_customer_address)
```

```
## [1] 3999    6
```

Customer ID: A unique identifier for each customer - there are 3999 observations in the data set and the same number of unique IDs so there are no repeats. These numbers are sequential but not complete - there are some missing from the sequence.

```r
length(unique(df_customer_address$customer_id))
```

```
## [1] 3999
```

```r
min(df_customer_address$customer_id)
```

```
## [1] 1
```

```r
max(df_customer_address$customer_id)
```

```
## [1] 4003
```

```r
unique(is.na(df_customer_address$customer_id))
```

```
## [1] FALSE
```

Address: String with the customer's address. Contains 3996 unique values for 3999 observations, indicating that perhaps some of the customers live together. Complete - contains no missing or NA values.

```r
length(unique(df_customer_address$address))
```

```
## [1] 3996
```

```r
unique(is.na(df_customer_address$address))
```

```
## [1] FALSE
```

```r
#anyDuplicated(df_customer_address$address)
#df_customer_address %>% filter(df_customer_address$address == 'NA')
```

Postcode: There are 873 unique post codes ranging between 2000 and 4883 indicating that there are many repeats. Complete - there are no missing values in this column.

```r
length(unique(df_customer_address$postcode))
```

```
## [1] 873
```

```r
min(df_customer_address$postcode)
```

```
## [1] 2000
```

```r
max(df_customer_address$postcode)
```

```
## [1] 4883
```

```r
unique(is.na(df_customer_address$postcode))
```

```
## [1] FALSE
```

State: There are 5 unique values in this column, but there is a uniqueness problem as both 'New South Wales' and 'NSW' refer to the same state. We will deal with this by changing all instances of 'New South Wales' to 'NSW' to keep the format consistent with the other values. There are no missing values in this column.

```r
unique(df_customer_address$state)
```

```
## [1] "New South Wales" "QLD"             "VIC"             "NSW"
## [5] "Victoria"
```

```r
unique(is.na(df_customer_address$state))
```

```
## [1] FALSE
```

Clean:

```r
df_customer_address = df_customer_address %>% mutate(state = replace(state, state == 'New South Wales',
```

Country: Contains only one value - 'Australia'. There are no missing values.

```r
unique(df_customer_address$country)
```

```
## [1] "Australia"
```

```r
unique(is.na(df_customer_address$country))
```

```
## [1] FALSE
```

Property Valuation: Contains integers between 1 and 12 inclusive. There are no missing values.

```r
sort(unique(df_customer_address$property_valuation))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
min(df_customer_address$property_valuation)
```

```
## [1] 1
```

```r
max(df_customer_address$property_valuation)
```

```
## [1] 12
```

```r
unique(is.na(df_customer_address$property_valuation))
```

```
## [1] FALSE
```

Customer Demographic Data Quality

Dataframe with 4000 observations and 13 variables.

```r
names(df_customer_demographic)
```

```
##  [1] "customer_id"
##  [2] "first_name"
##  [3] "last_name"
##  [4] "gender"
##  [5] "past_3_years_bike_related_purchases"
##  [6] "DOB"
##  [7] "job_title"
##  [8] "job_industry_category"
##  [9] "wealth_segment"
## [10] "deceased_indicator"
## [11] "default"
## [12] "owns_car"
## [13] "tenure"
```

```r
dim(df_customer_demographic)
```

```
## [1] 4000   13
```

Customer ID: There are 4000 unique values, between 1 and 4000, matching the number of observations. There are no missing values.

```r
length(unique(df_customer_demographic$customer_id))
```

```
## [1] 4000
```

```r
min(df_customer_demographic$customer_id)
```

```
## [1] 1
```

```
max(df_customer_demographic$customer_id)
```

```
## [1] 4000
```

```
unique(is.na(df_customer_demographic$customer_id))
```

```
## [1] FALSE
```

First Name: There are 3139 unique first names, indicating some repeats as would be expected. There are no missing values.

```
length(unique(df_customer_demographic$first_name))
```

```
## [1] 3139
```

```
unique(is.na(df_customer_demographic$first_name))
```

```
## [1] FALSE
```

Last Name: There are 3726 unique last names - fewer repeats than in the first name column as expected. There are missing values in this column. They have been imported as NA.

```
length(unique(df_customer_demographic$last_name))
```

```
## [1] 3726
```

```
unique(is.na(df_customer_demographic$last_name))
```

```
## [1] FALSE  TRUE
```

Gender: There are 6 unique values for gender. Corresponding to male there are 'Male' and 'M' and corresponding to female there are 'Female', 'Femal' and 'F'. These all need to be consolidated to one standard. There is also 'U', which does not clearly correspond to either and should be changed to NA. We can use the gendercodeR package to recode the values appropriately to 'male', 'female' and NA.

```
unique(df_customer_demographic$gender)
```

```
## [1] "F"      "Male"   "Female" "U"      "Femal"  "M"
```

```
unique(is.na(df_customer_demographic$gender))
```

```
## [1] FALSE
```

```
library(gendercodeR)
df_customer_demographic = df_customer_demographic %>% mutate(gender = gendercodeR::recode_gender(gender
```

Bike Related Purchases for the Past 3 Years: Contains 100 unique integers between 0 and 99. No missing values.

```
length(unique(df_customer_demographic$past_3_years_bike_related_purchases))
```

```
## [1] 100
```

```
min(df_customer_demographic$past_3_years_bike_related_purchases)
```

```
## [1] 0
```

```
max(df_customer_demographic$past_3_years_bike_related_purchases)
```

```
## [1] 99
```

```
unique(is.na(df_customer_demographic$past_3_years_bike_related_purchases))
```

```
## [1] FALSE
```

Date of Birth: Need to convert the values in the date column to the correct format:

```
df_customer_demographic = df_customer_demographic %>% mutate(DOB = as.numeric(DOB)) %>% mutate(DOB = ja
```

```
## Warning: NAs introduced by coercion
```

There are 3448 unique dates in this column, and there are missing values, imported as NA.

```
length(unique(df_customer_demographic$DOB))
```

```
## [1] 3448
```

```
min(df_customer_demographic$DOB, na.rm = TRUE)
```

```
## [1] "1931-10-23"
```

```
max(df_customer_demographic$DOB, na.rm = TRUE)
```

```
## [1] "2002-03-11"
```

```
unique(is.na(df_customer_demographic$DOB))
```

```
## [1] FALSE  TRUE
```

Job Title: There are 196 unique values in this column. There are missing values, imported as NA.

```
length(unique(df_customer_demographic$job_title))
```

```
## [1] 196
```

```r
unique(is.na(df_customer_demographic$job_title))
```

```
## [1] FALSE  TRUE
```

Job Industry Category: There are 10 unique values in this column. There are missing values, imported as NA.

```r
unique(df_customer_demographic$job_industry_category)
```

```
##  [1] "Health"             "Financial Services" "Property"
##  [4] "IT"                 NA                   "Retail"
##  [7] "Argiculture"        "Manufacturing"      "Telecommunications"
## [10] "Entertainment"
```

```r
unique(is.na(df_customer_demographic$job_industry_category))
```

```
## [1] FALSE  TRUE
```

Wealth Segment: There are 3 unique values in this column and no missing values.

```r
sort(unique(df_customer_demographic$wealth_segment))
```

```
## [1] "Affluent Customer" "High Net Worth"    "Mass Customer"
```

```r
unique(is.na(df_customer_demographic$wealth_segment))
```

```
## [1] FALSE
```

Deceased Indicator: There are two values in this column - 'Y' and 'N'. There are no missing values.

```r
sort(unique(df_customer_demographic$deceased_indicator))
```

```
## [1] "N" "Y"
```

```r
unique(is.na(df_customer_demographic$deceased_indicator))
```

```
## [1] FALSE
```

Default: There ae 93 unique values in this column, but they appear to be nonsensical strings of characters Not sure how to clean this data. Contains missing values.

```r
length(unique(df_customer_demographic$default))
```

```
## [1] 93
```

```
unique(is.na(df_customer_demographic$default))
```

```
## [1] FALSE  TRUE
```

Owns Car: Two values - ‘Yes’ and ‘No’. No missing values.

```
unique(df_customer_demographic$owns_car)
```

```
## [1] "Yes" "No"
```

```
unique(is.na(df_customer_demographic$owns_car))
```

```
## [1] FALSE
```

Tenure: Contains integer values between 1 and 22 inclusive. Contains missing values.

```
sort(unique(df_customer_demographic$tenure))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
unique(is.na(df_customer_demographic$tenure))
```

```
## [1] FALSE  TRUE
```

New Customer List Data Quality

Dataframe with 1000 observations and 18 variables.

```
names(df_new_customer_list)
```

```
##  [1] "first_name"
##  [2] "last_name"
##  [3] "gender"
##  [4] "past_3_years_bike_related_purchases"
##  [5] "DOB"
##  [6] "job_title"
##  [7] "job_industry_category"
##  [8] "wealth_segment"
##  [9] "deceased_indicator"
## [10] "owns_car"
## [11] "tenure"
## [12] "address"
## [13] "postcode"
## [14] "state"
## [15] "country"
## [16] "property_valuation"
## [17] "...17"
## [18] "...18"
## [19] "...19"
## [20] "...20"
## [21] "...21"
## [22] "Rank"
## [23] "Value"
```

```r
dim(df_new_customer_list)
```

```
## [1] 1000   23
```

First Name: Contains 940 unique first names. No missing values.

```r
length(unique(df_new_customer_list$first_name))
```

```
## [1] 940
```

```r
unique(is.na(df_new_customer_list$first_name))
```

```
## [1] FALSE
```

Last Name: Contains 962 unique last names. contains missing values, imported as NA.

```r
length(unique(df_new_customer_list$last_name))
```

```
## [1] 962
```

```r
unique(is.na(df_new_customer_list$last_name))
```

```
## [1] FALSE  TRUE
```

Gender There are 3 unique values for gender, 'Male', 'Female' and 'U'. We will use gendercodeR again to recode the values appropriately to 'male', 'female' and NA.

```r
unique(df_new_customer_list$gender)
```

```
## [1] "Male"   "Female" "U"
```

```r
unique(is.na(df_new_customer_list$gender))
```

```
## [1] FALSE
```

```r
library(gendercodeR)
df_new_customer_list = df_new_customer_list %>% mutate(gender = gendercodeR::recode_gender(gender))
```

Bike Related Purchases for the Past 3 Years: Contains 100 unique integers between 0 and 99. No missing values.

```r
length(unique(df_new_customer_list$past_3_years_bike_related_purchases))
```

```
## [1] 100
```

```r
min(df_new_customer_list$past_3_years_bike_related_purchases)
```

```
## [1] "0"
```

```r
max(df_new_customer_list$past_3_years_bike_related_purchases)
```

```
## [1] "99"
```

```r
unique(is.na(df_new_customer_list$past_3_years_bike_related_purchases))
```

```
## [1] FALSE
```

Date of Birth: There are 3448 unique values in this column, and there are missing values, imported as NA.

```r
length(unique(df_new_customer_list$DOB))
```

```
## [1] 962
```

```r
min(df_new_customer_list$DOB, na.rm = TRUE)
```

```
## [1] "1938-06-08"
```

```r
max(df_new_customer_list$DOB, na.rm = TRUE)
```

```
## [1] "28912"
```

```r
unique(is.na(df_new_customer_list$DOB))
```

```
## [1] FALSE  TRUE
```

Job Title: There are 185 unique values in this column. There are missing values, imported as NA.

```r
length(unique(df_new_customer_list$job_title))
```

```
## [1] 185
```

```r
unique(is.na(df_new_customer_list$job_title))
```

```
## [1] FALSE  TRUE
```

Job Industry Category: There are 10 unique values in this column. There are missing values, imported as NA.

```r
unique(df_new_customer_list$job_industry_category)
```

```
##  [1] "Manufacturing"      "Property"           "Financial Services"
##  [4] "Entertainment"      "Retail"             "IT"                
##  [7] "Telecommunications" "Health"             NA                  
## [10] "Argiculture"
```

```r
unique(is.na(df_new_customer_list$job_industry_category))
```

```
## [1] FALSE  TRUE
```

Wealth Segment: There are 3 unique values in this column and no missing values.

```r
sort(unique(df_new_customer_list$wealth_segment))
```

```
## [1] "Affluent Customer" "High Net Worth"    "Mass Customer"
```

```r
unique(is.na(df_new_customer_list$wealth_segment))
```

```
## [1] FALSE
```

Deceased Indicator: There is only one value in this column - ‘N’ as none of the new customers are deceased. There are no missing values.

```r
sort(unique(df_new_customer_list$deceased_indicator))
```

```
## [1] "N"
```

```r
unique(is.na(df_new_customer_list$deceased_indicator))
```

```
## [1] FALSE
```

Owns Car: Two values - ‘Yes’ and ‘No’. No missing values.

```r
unique(df_new_customer_list$owns_car)
```

```
## [1] "Yes" "No"
```

```r
unique(is.na(df_new_customer_list$owns_car))
```

```
## [1] FALSE
```

Tenure: Contains integer values between 1 and 22 inclusive. Contains no missing values.

```r
sort(unique(df_new_customer_list$tenure))
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```r
unique(is.na(df_new_customer_list$tenure))
```

```
## [1] FALSE
```

Address: 1000 unique values, no missing values.

```r
length(unique(df_new_customer_list$address))
```

```
## [1] 1000
```

```r
unique(is.na(df_new_customer_list$address))
```

```
## [1] FALSE
```

Postcode: 522 values, no missing values.

```r
length(unique(df_new_customer_list$postcode))
```

```
## [1] 522
```

```r
unique(is.na(df_new_customer_list$postcode))
```

```
## [1] FALSE
```

State: Three states - 'QLD', 'NSW' and 'VIC'. No missing values.

```r
unique(df_new_customer_list$state)
```

```
## [1] "QLD" "NSW" "VIC"
```

```r
unique(is.na(df_new_customer_list$state))
```

```
## [1] FALSE
```

Country: One value - 'Australia'. No missing values.

```r
unique(df_new_customer_list$country)
```

```
## [1] "Australia"
```

```r
unique(is.na(df_new_customer_list$country))
```

```
## [1] FALSE
```

Property Valuation: 12 values between 1 and 12 inclusive. These have imported as characters, so need to convert to numeric. No missing values.

```r
sort(unique(df_new_customer_list$property_valuation))
```

```
##  [1] "1"  "10" "11" "12" "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"
```

```r
unique(is.na(df_new_customer_list$property_valuation))
```

```
## [1] FALSE
```

```r
df_new_customer_list = df_new_customer_list %>% mutate(property_valuation = as.numeric(property_valuati
```

Rank: 324 unique values between 1 and 1000. No missing values.

```r
length(unique(df_new_customer_list$Rank))
```

```
## [1] 324
```

```r
min(df_new_customer_list$Rank)
```

```
## [1] 1
```

```r
max(df_new_customer_list$Rank)
```

```
## [1] 1000
```

```r
unique(is.na(df_new_customer_list$Rank))
```

```
## [1] FALSE
```

Value: 324 unique values between 0.34 and 1.72. No missing values.

```r
length(unique(df_new_customer_list$Value))
```

```
## [1] 324
```

```r
min(df_new_customer_list$Value)
```

```
## [1] 0.34
```

```
max(df_new_customer_list$Value)
```

```
## [1] 1.71875
```

```
unique(is.na(df_new_customer_list$Value))
```

```
## [1] FALSE
```

Transactions Data Quality

Dataframe with 20000 observations and 13 variables.

```
names(df_transactions)
```

```
##  [1] "transaction_id"          "product_id"
##  [3] "customer_id"             "transaction_date"
##  [5] "online_order"            "order_status"
##  [7] "brand"                   "product_line"
##  [9] "product_class"           "product_size"
## [11] "list_price"              "standard_cost"
## [13] "product_first_sold_date"
```

```
dim(df_transactions)
```

```
## [1] 20000    13
```

Transaction ID: 20000 unique values between 1 and 20000. No missing values.

```
length(unique(df_transactions$transaction_id))
```

```
## [1] 20000
```

```
min(df_transactions$transaction_id)
```

```
## [1] 1
```

```
max(df_transactions$transaction_id)
```

```
## [1] 20000
```

```
unique(is.na(df_transactions$transaction_id))
```

```
## [1] FALSE
```

Product ID: 101 unique values between 0 and 100. No missing values.

```r
length(unique(df_transactions$product_id))
```

```
## [1] 101
```

```r
min(df_transactions$product_id)
```

```
## [1] 0
```

```r
max(df_transactions$product_id)
```

```
## [1] 100
```

```r
unique(is.na(df_transactions$product_id))
```

```
## [1] FALSE
```

Customer ID: 3494 unique values between 1 and 5034. No missing values.

```r
length(unique(df_transactions$customer_id))
```

```
## [1] 3494
```

```r
min(df_transactions$customer_id)
```

```
## [1] 1
```

```r
max(df_transactions$customer_id)
```

```
## [1] 5034
```

```r
unique(is.na(df_transactions$customer_id))
```

```
## [1] FALSE
```

Transaction Date: 364 unique values. No missing values.

```r
length(unique(df_transactions$transaction_date))
```

```
## [1] 364
```

```r
min(df_transactions$transaction_date)
```

```
## [1] "2017-01-01 UTC"
```

```r
max(df_transactions$transaction_date)
```

```
## [1] "2017-12-30 UTC"
```

```r
unique(is.na(df_transactions$transaction_date))
```

```
## [1] FALSE
```

Online Order: Missing values.

```r
unique(df_transactions$online_order)
```

```
## [1] FALSE  TRUE    NA
```

```r
unique(is.na(df_transactions$online_order))
```

```
## [1] FALSE  TRUE
```

Order Status: No missing values.

```r
unique(df_transactions$order_status)
```

```
## [1] "Approved"  "Cancelled"
```

```r
unique(is.na(df_transactions$order_status))
```

```
## [1] FALSE
```

Brand: 7 unique values between. Contains missing values.

```r
unique(df_transactions$brand)
```

```
## [1] "Solex"          "Trek Bicycles"  "OHM Cycles"     "Norco Bicycles"
## [5] "Giant Bicycles" "WeareA2B"       NA
```

```r
unique(is.na(df_transactions$brand))
```

```
## [1] FALSE  TRUE
```

Product Line: 4 unique values. Contains missing values.

```r
unique(df_transactions$product_line)
```

```
## [1] "Standard" "Road"     "Mountain" "Touring"  NA
```

```r
unique(is.na(df_transactions$product_line))
```

```
## [1] FALSE  TRUE
```

Product Class: 3 unique values. Contains missing values.

```r
unique(df_transactions$product_class)
```

```
## [1] "medium" "low"    "high"   NA
```

```r
unique(is.na(df_transactions$product_class))
```

```
## [1] FALSE  TRUE
```

Product Size: 3 unique values. Contians missing values.

```r
unique(df_transactions$product_size)
```

```
## [1] "medium" "large"  "small"  NA
```

```r
unique(is.na(df_transactions$product_size))
```

```
## [1] FALSE  TRUE
```

List Price: 296 unique values between 12.01 and 2091.47. No missing values.

```r
length(unique(df_transactions$list_price))
```

```
## [1] 296
```

```r
min(df_transactions$list_price)
```

```
## [1] 12.01
```

```r
max(df_transactions$list_price)
```

```
## [1] 2091.47
```

```r
unique(is.na(df_transactions$list_price))
```

```
## [1] FALSE
```

Standard Cost: 104 unique values between 7.21 and 1759.85. Contains missing values.

```r
length(unique(df_transactions$standard_cost))
```

```
## [1] 104
```

```r
min(df_transactions$standard_cost, na.rm = TRUE)
```

```
## [1] 7.21
```

```r
max(df_transactions$standard_cost, na.rm = TRUE)
```

```
## [1] 1759.85
```

```r
unique(is.na(df_transactions$standard_cost))
```

```
## [1] FALSE  TRUE
```

Product First Sold Date:

Need to convert the values in the date column to the correct format:

```r
df_transactions = df_transactions %>% mutate(product_first_sold_date = as.numeric(product_first_sold_da
```

101 unique dates. Contains missing values.

```r
length(unique(df_transactions$product_first_sold_date))
```

```
## [1] 101
```

```r
min(df_transactions$product_first_sold_date, na.rm = TRUE)
```

```
## [1] "1991-01-21"
```

```r
max(df_transactions$product_first_sold_date, na.rm = TRUE)
```

```
## [1] "2016-12-06"
```

```r
unique(is.na(df_transactions$product_first_sold_date))
```

```
## [1] FALSE  TRUE
```

Data Insights

Targeting high value customers based on customer demographics and attributes.

Merge Data Frames

Merge data sets on the customer_id column to form the df_address_demo_transactions data frame:

```r
df_address_demo = merge(df_customer_address, df_customer_demographic, by = 'customer_id')

df_address_demo_transactions = merge(df_address_demo, df_transactions, by = 'customer_id')
```

Merged data frame contains 19,968 observations with 30 columns.

```r
dim(df_address_demo_transactions)
```

```
## [1] 19968    30
```

```r
names(df_address_demo_transactions)
```

```
##  [1] "customer_id"
##  [2] "address"
##  [3] "postcode"
##  [4] "state"
##  [5] "country"
##  [6] "property_valuation"
##  [7] "first_name"
##  [8] "last_name"
##  [9] "gender"
## [10] "past_3_years_bike_related_purchases"
## [11] "DOB"
## [12] "job_title"
## [13] "job_industry_category"
## [14] "wealth_segment"
## [15] "deceased_indicator"
## [16] "default"
## [17] "owns_car"
## [18] "tenure"
## [19] "transaction_id"
## [20] "product_id"
## [21] "transaction_date"
## [22] "online_order"
## [23] "order_status"
## [24] "brand"
## [25] "product_line"
## [26] "product_class"
## [27] "product_size"
## [28] "list_price"
## [29] "standard_cost"
## [30] "product_first_sold_date"
```

Proposed Plan

Calculating Customer Profit Per Year

In order to determine high value customers, we would like to be able to predict some metric that measures any particular customer's value to Sprocket over a given time period - for example profit generated in a year.

We can calculate this for each customer by grouping their transactions by their customer ID in the transactions data set, then subtracting the summed standard costs from the summed list prices across all their purchases to find the profit they have generated. Then we can look at the time period over which these transactions were made and normalise to some standard period of time, say a day or a year.

```
profit = df_address_demo_transactions %>% group_by(customer_id) %>% summarize(profit = sum(list_price) 

head(profit)
```

```
## # A tibble: 6 x 5
##   customer_id profit date_range profit_per_day profit_per_year
##         <dbl>  <dbl> <drtn>              <dbl>           <dbl>
## 1           1  3018. 352 days                9            3285
## 2           2  2226. 112 days               20            7300
## 3           4   221.  76 days                3            1095
## 4           5  2395. 286 days                8            2920
## 5           6  3947. 272 days               15            5475
## 6           7   220.  62 days                4            1460
```

Adapting Training Data Set:

We can then add this column of profit per year for each customer to the customer demographic data frame, matching on customer_id.

```
training_set = merge(df_customer_demographic, profit, by = 'customer_id')
```

We will further sculpt the training set by narrowing down the attributes that may be good predictors of profitability.

These are: gender, past_3_years_bike_related_purchases, DOB (which we will convert to age), wealth_segment, owns_car and tenure.

We will also remove the columns profit date_range profit_per_day, as these are redundant. We leave profit_per_year as the dependant variable.

```
training_set = training_set %>% select(customer_id, profit_per_year, gender, past_3_years_bike_related_
```

```
age = training_set %>% group_by(customer_id) %>% summarize(age = round(as.numeric((Sys.Date() - DOB) / 
```

```
training_set = merge(training_set, age, by = 'customer_id')

training_set = training_set %>% select(profit_per_year, gender, past_3_years_bike_related_purchases, ag
```

There are 190 rows in which case profit per year has been coded as NA. Since we are tring to predict profit per year using other attributes in this dataframe, we will ignore the rows where profit per year is NA when fitting a predictive model.

```
training_set = training_set %>% filter(is.na(training_set$profit_per_year) == FALSE)
```

There are also 49 rows where the profit per year is 'Inf'. We will filter out these rows as well.

```
training_set = training_set %>% filter(training_set$profit_per_year != Inf)
```

Data Exploration

We can now examine the pairwise relationship between each predictor variable and the dependent variable, profit per year.

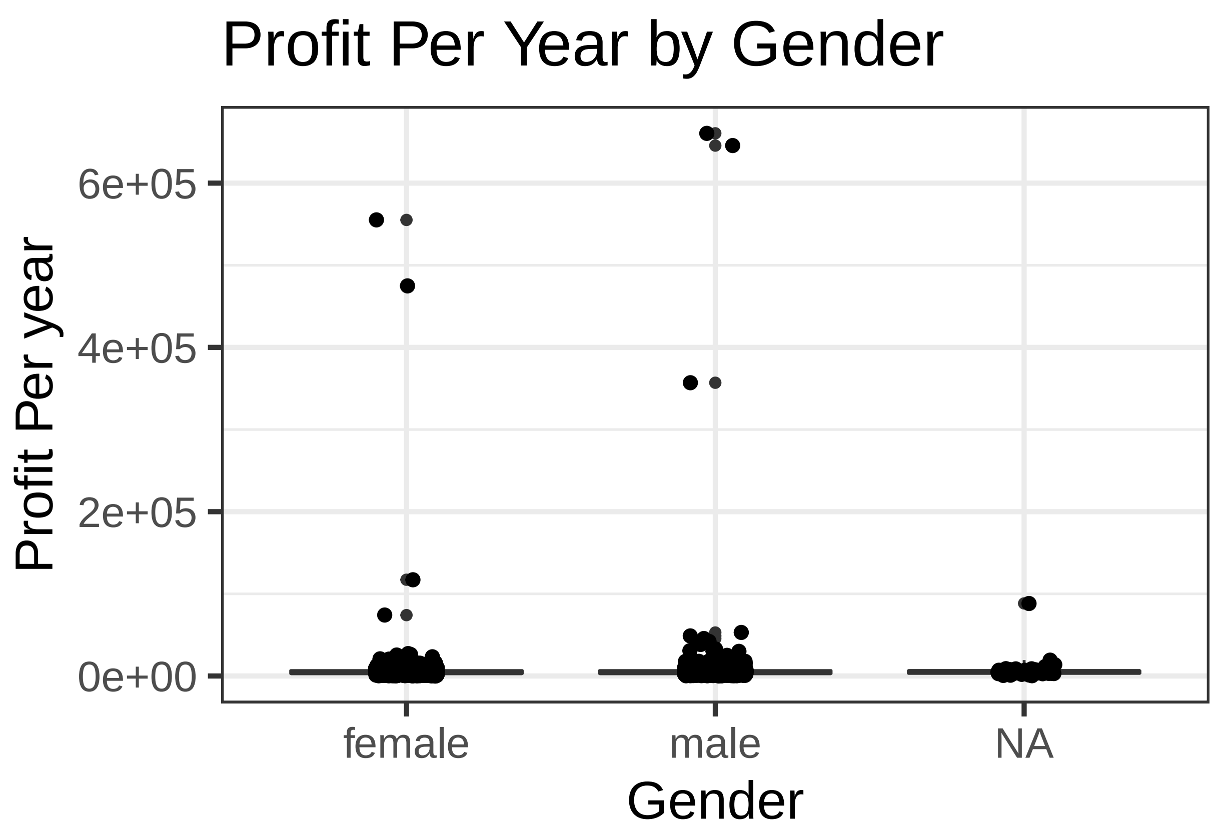
```
head(training_set)
```

```
##   profit_per_year gender past_3_years_bike_related_purchases age
## 1            3285 female                                  93  66
## 2            7300   male                                  81  39
## 3            1095   male                                  33  58
## 4            2920 female                                  56  43
## 5            5475   male                                  35  53
## 6            1460 female                                   6  44
##      wealth_segment owns_car tenure
## 1     Mass Customer      Yes     11
## 2     Mass Customer      Yes     16
## 3     Mass Customer       No      7
## 4 Affluent Customer      Yes      8
## 5    High Net Worth      Yes     13
## 6 Affluent Customer      Yes     11
```

Gender:

There does not appear to be much difference in the profitability of male and female customers. In each case, there are large outliers skewing the results.

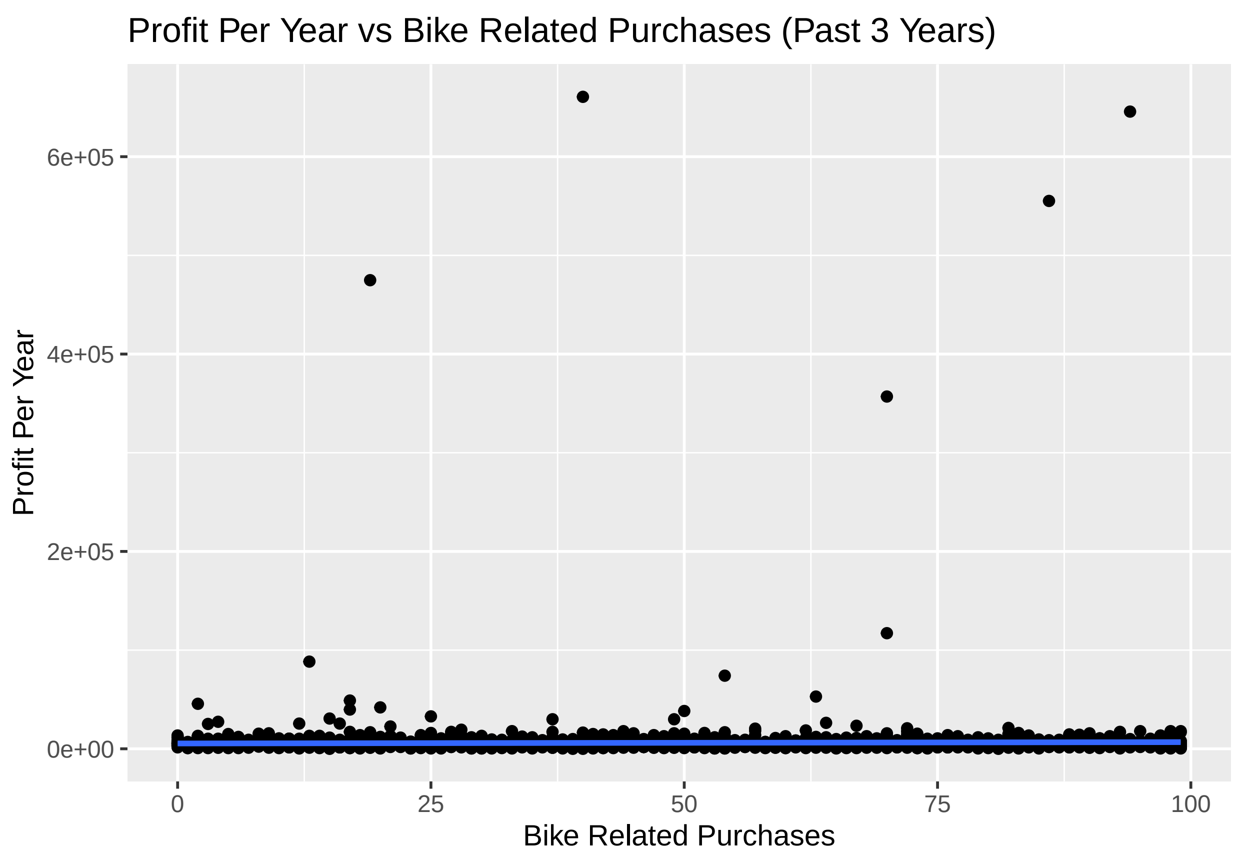
```
training_set %>% ggplot(aes(y = profit_per_year, x = gender)) +
  geom_boxplot(coef = 10) +
  geom_jitter(width=0.1, size = 2) +
  theme_bw(base_size = 20) +
  theme(legend.position = "none") +
  labs(y = "Profit Per year",
       x = "Gender",
       title = 'Profit Per Year by Gender')
```



Bike Related Purchases for the Past 3 Years:

There does not appear to be a linear relationship between purchases over the past 3 years and profit per year.

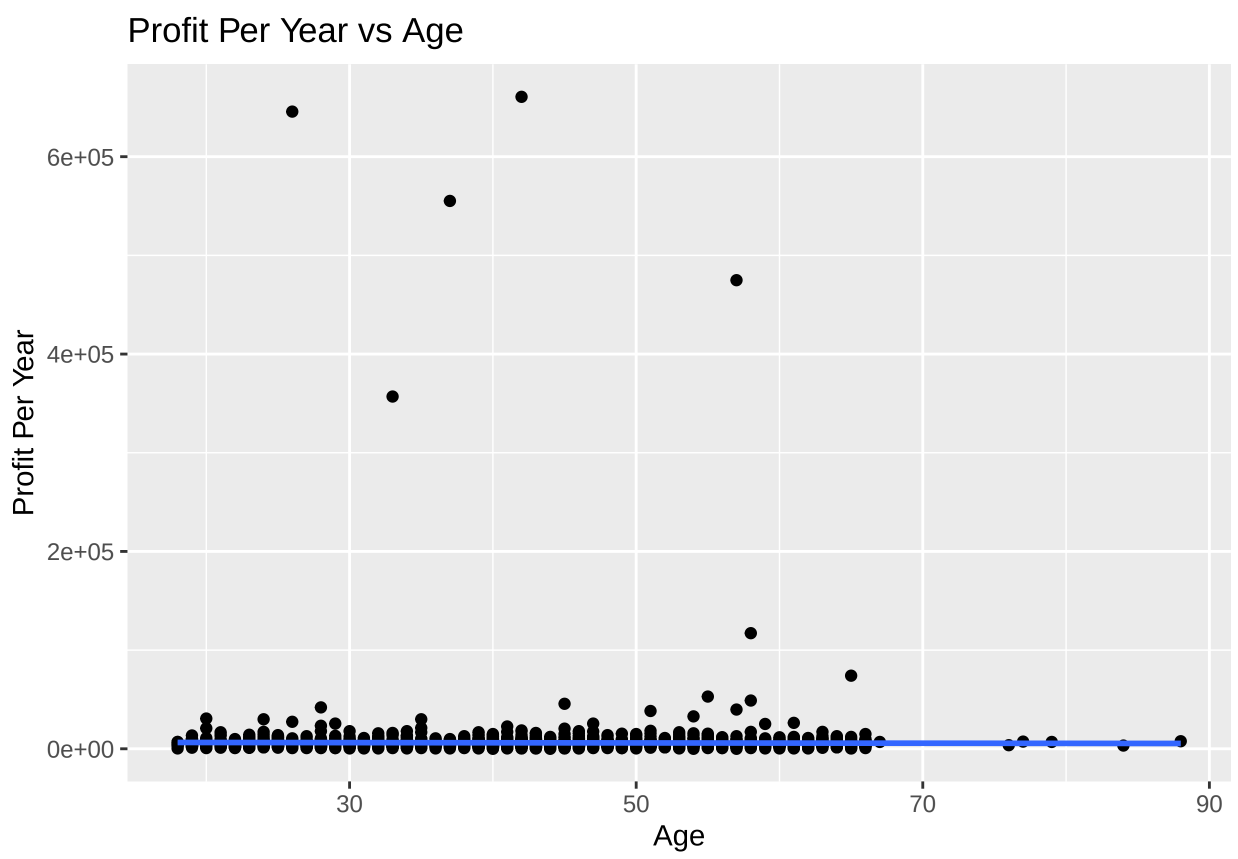
```
training_set %>% ggplot(aes(x = past_3_years_bike_related_purchases, y = profit_per_year)) + geom_point
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Profit Per Year",
       x = "Bike Related Purchases",
       title = 'Profit Per Year vs Bike Related Purchases (Past 3 Years)')
```



Age:

There does not appear to be a linear relationship between age and profit per year.

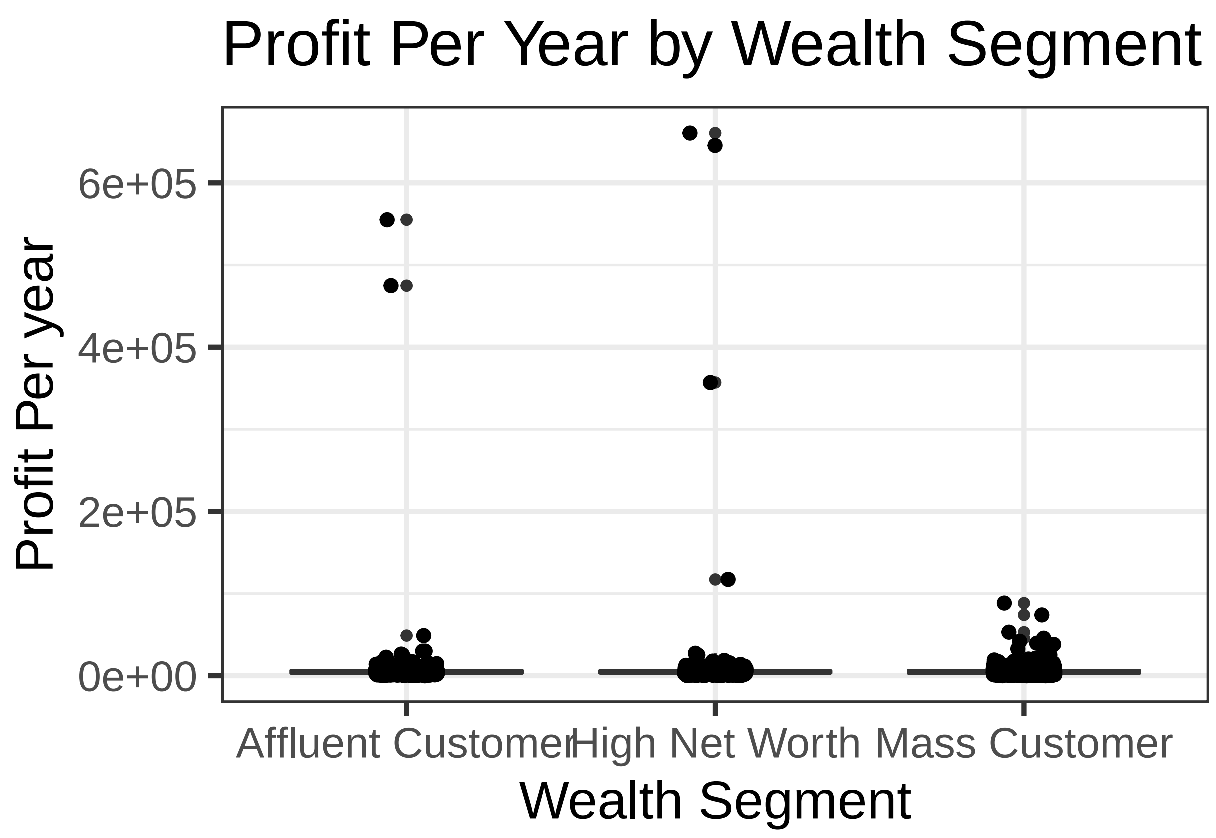
```
training_set %>% ggplot(aes(x = age, y = profit_per_year)) + geom_point() +
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Profit Per Year",
       x = "Age",
       title = 'Profit Per Year vs Age')
```



Wealth Segment:

There does not appear to be much difference in the profitability of customers from different wealth segments. In each case, there are large outliers skewing the results.

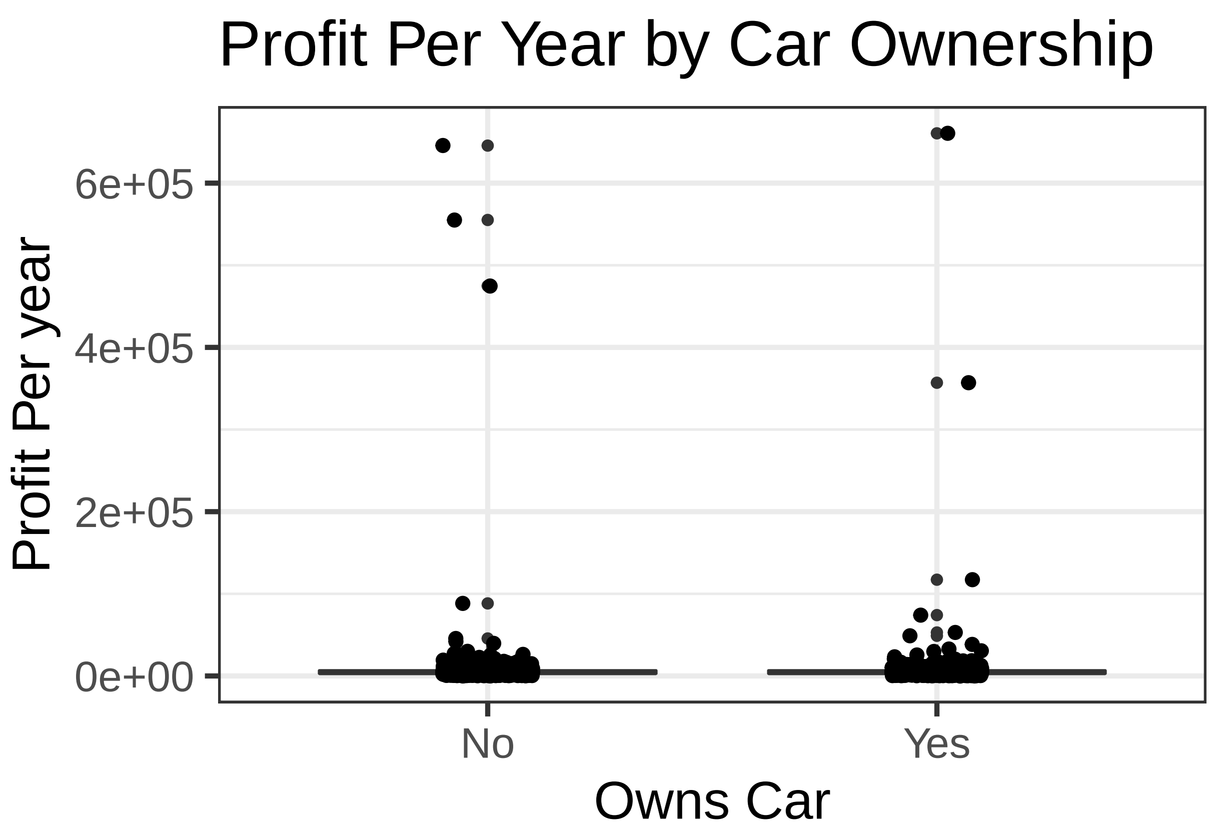
```r
training_set %>% ggplot(aes(y = profit_per_year, x = wealth_segment)) +
  geom_boxplot(coef = 10) +
  geom_jitter(width=0.1, size = 2) +
  theme_bw(base_size = 20) +
  theme(legend.position = "none") +
  labs(y = "Profit Per year",
       x = "Wealth Segment",
       title = 'Profit Per Year by Wealth Segment')
```



Owns Car:

There does not appear to be much difference in the profitability of customers who do and don't own a car. In each case, there are large outliers skewing the results.

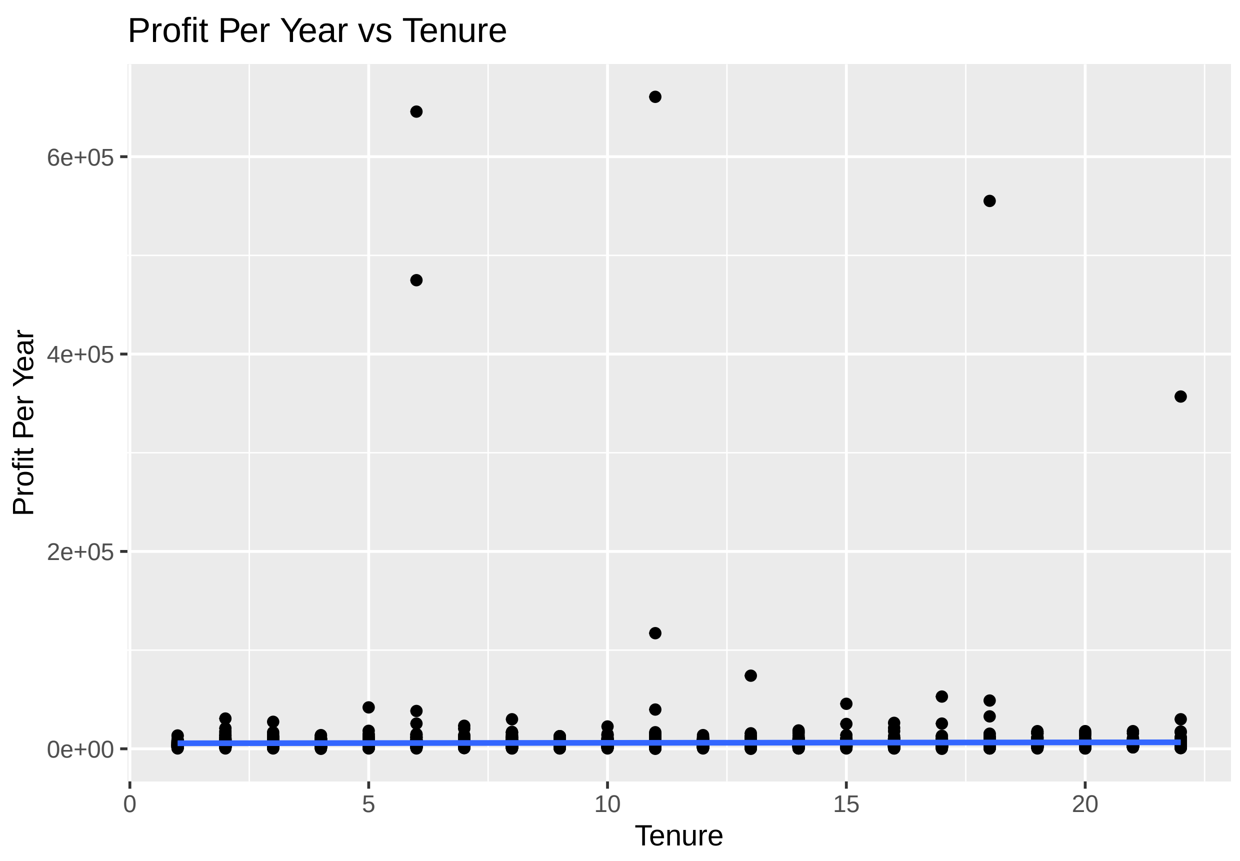
```r
training_set %>% ggplot(aes(y = profit_per_year, x = owns_car)) +
  geom_boxplot(coef = 10) +
  geom_jitter(width=0.1, size = 2) +
  theme_bw(base_size = 20) +
  theme(legend.position = "none") +
  labs(y = "Profit Per year",
       x = "Owns Car",
       title = 'Profit Per Year by Car Ownership')
```



Tenure:

There does not appear to be a linear relationship between tenure and profit per year.

```
training_set %>% ggplot(aes(x = tenure, y = profit_per_year)) + geom_point() +
  geom_smooth(method=lm) +
  theme(legend.position = "none") +
  labs(y = "Profit Per Year",
       x = "Tenure",
       title = 'Profit Per Year vs Tenure')
```



Recoding categorical variables:

We will now recode categorical variables like gender, wealth_segment and owns_car as binary or ordinal numerical variables. We will leave out attributes like job_title and job_industry_category as there is no clear way to recode these as numerical values.

Gender: 'Male' recoded as 1, 'Female' as 0 Owns Car: 'Yes' recoded as 1, 'No' as 0 Wealth Segment: 'Mass Customer' recoded as 0, 'Affluent Customer' as 1, 'High Net Worth' as 2)

```
training_set$gender = ifelse(training_set$gender == "M", 1, 0)
training_set$owns_car = ifelse(training_set$owns_car == "Yes", 1, 0)
training_set$wealth_segment = recode(training_set$wealth_segment, 'Mass Customer' = 0, 'Affluent Custome
```

We also need to drop any rows that contain NA for any of the other variables. This leaves us with 3182 complete observations.

```
training_set = na.omit(training_set)
dim(training_set)
```

```
## [1] 3182    7
```

Fit Predictive Model and Measure Accuracy:

We will now fit a predictive model using the test set to predict customer profit per year from demographic variables. Given that the relationship between each of the predictor variables and the dependent variable does not appear to be linear, a multiple linear regression does not seem to be appropriate.

A better method might be a tree based model like a random forest regression. This has the added benefit of being an ensemble model which prevents overfitting of any single decision tree and balances the bias-variance tradeoff better than a single tree would.

We can measure the accuracy of the model using cross validation and metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

We will fit the model using 60% of the training set. This will allow us to test the predictive accuracy of the model using the remaining 40%, which will give us an idea of how the model performs in and out of sample. This will give us an idea of how confident to be about the predictions made on the unlabelled new customer data set.

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```
set.seed(101)
sample = sample.split(names(training_set), SplitRatio = 0.6)
train = subset(training_set, sample == TRUE)
test  = subset(training_set, sample == FALSE)
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

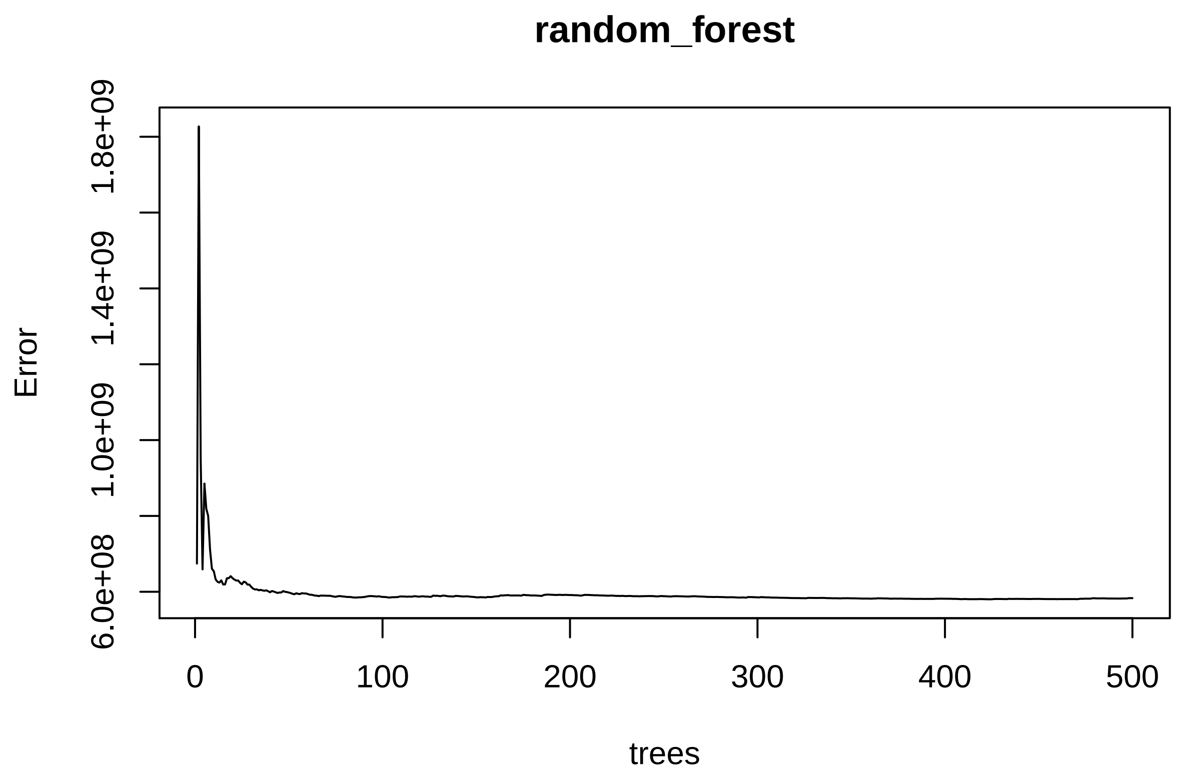
```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(101)

random_forest = randomForest(formula = profit_per_year ~ ., data = train)

random_forest
```

```
##
## Call:
##  randomForest(formula = profit_per_year ~ ., data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 583178183
##                     % Var explained: -6.02
```

We can plot the error rate based on the out of bag sample error as we average across an increasing number of trees and see that it levels out.

```
plot(random_forest)
```



We can also find the number of trees providing the lowest error rate. There are 423 trees providing an average prediction error of $24059.17.

```
which.min(random_forest$mse)
```

```
## [1] 424
```

```
sqrt(random_forest$mse[which.min(random_forest$mse)])
```

```
## [1] 24086.85
```

Testing Predictive Accuracy

Once a predictive model has been fit, we can use the new customer data set as a test set to make predictions about what kind of profit each of these customers will make for Sprocket in a year. Ranking these expected profit, we can determine which customers are expected to be most profitable and treat these customers as high value.

In Sample Accuracy:

```
tree_pred_in = predict(random_forest, newdata = train, interval = "prediction", level = 0.90)

tree_mse_in = mean((train$profit_per_year - tree_pred_in)^2)
sqrt(tree_mse_in)
```

```
## [1] 17802.91
```

```
tree_mae_in = mean(abs(train$profit_per_year - tree_pred_in))
tree_mae_in
```

```
## [1] 3004.436
```

Out of Sample Accuracy:

```
predictions_out = predict(random_forest, newdata = test, interval = "prediction", level = 0.90)

mse_out = mean((test$profit_per_year - predictions_out)^2)
sqrt(mse_out)
```

```
## [1] 20235.15
```

```
mae_out = mean(abs(test$profit_per_year - predictions_out))
mae_out
```

```
## [1] 3679.975
```

The RMSE and MAE are fairly similar both in and out of sample (differing by about $3000 for RMSE and $700 for MAE), which is a good indication our model has not been overfit to the training data and performs well on unseen data.

Predicting Profitability of New Customers

To predict profit per year from unseen demographic data we must adapt the new customer list data into a form identical to that of the training set - that is a data frame with the columns gender, past_3_years_bike_related_purchases, age, wealth_segment, owns_car and tenure.

We will add a customer ID to the new customer list:

```
df_new_customer_list$customer_id = c(1:1000)
```

Now select the appropriate columns for the test set to make predictions on.

```
test_set = df_new_customer_list %>% select(customer_id, gender, past_3_years_bike_related_purchases, DOI
```

```
new_cust_DOB_change = df_new_customer_list %>% mutate(DOB = ifelse(grepl(pattern = '[0-9]{5}', x = DOB)

new_cust_DOB_filter = new_cust_DOB_change %>% filter(DOB != is.na(new_cust_DOB_change$DOB))
```

```
new_cust_DOB_filter$age = round(as.numeric((Sys.Date() - as.Date(new_cust_DOB_filter$DOB)) / 365))
```

```
test_set = new_cust_DOB_filter %>% select(customer_id, gender, past_3_years_bike_related_purchases, age
```

We will drop any rows that contain NA.

```
test_set = na.omit(test_set)
```

We will recode categorical variables in the same way as the training set:

```
test_set$gender = ifelse(test_set$gender == "M", 1, 0)
test_set$owns_car = ifelse(test_set$owns_car == "Yes", 1, 0)
test_set$wealth_segment = recode(test_set$wealth_segment, 'Mass Customer' = 0, 'Affluent Customer' = 1,
```

This leaves us with 929 complete observations.

```
dim(test_set)
```

```
## [1] 929   7
```

New Customer Predictions:

```
new_customer_predictions = predict(random_forest, newdata = test_set, interval = "prediction", level = 
```

```
new_customer_predictions = as.data.frame(new_customer_predictions)
new_customer_predictions$customer_id = test_set$customer_id
```

Merge predictions with new customer dataset:

```
df_new_cust_profit_predictions = merge(df_new_customer_list, new_customer_predictions, by = 'customer_i
```

Customers Ranked by Prediced Yearly Profit

Sort the customers by their predicted profit - highest to lowest. We see there are 53 customers with a predicted yearly profit of $10,000 or above.

According to the model, these are the customers that should be targeted as being high value.

```
head(arrange(df_new_cust_profit_predictions, desc(new_customer_predictions)))
```

```
##   customer_id first_name  last_name gender
## 1          11   Rockwell     Matson   male
## 2         302      Daisy     Pollen female
## 3         610     Dorian      Emery female
## 4         535     Jacobo    Mucklow   male
## 5         403     Gannie      Bargh   male
## 6         949      Nolly Ivanchikov   male
##   past_3_years_bike_related_purchases        DOB
## 1                                  94 1995-01-01
## 2                                  97 1993-08-09
## 3                                  94 1998-08-24
## 4                                  62 1952-12-04
## 5                                  56 1955-02-13
## 6                                  13 1994-02-10
##                  job_title job_industry_category wealth_segment
## 1       Programmer Analyst I                Retail High Net Worth
## 2            Cost Accountant    Financial Services  Mass Customer
## 3                  Professor         Manufacturing  Mass Customer
## 4 Computer Systems Analyst I    Financial Services High Net Worth
## 5         Analyst Programmer                  <NA>  Mass Customer
## 6         Help Desk Operator         Manufacturing High Net Worth
##   deceased_indicator owns_car tenure                address postcode
## 1                  N       No      3      3682 Crowley Point     4573
## 2                  N       No      7      61825 Debs Terrace     3167
## 3                  N      Yes      9       67 Beilfuss Plaza     2168
## 4                  N      Yes     22  5512 Ronald Regan Hill     3122
## 5                  N      Yes     13 1832 Burning Wood Place     3201
## 6                  N      Yes     11          6792 Kropf Hill     2049
##   state   country property_valuation ...17  ...18    ...19     ...20 ...21
## 1   QLD Australia                  6  0.58 0.5800 0.725000 0.7250000    10
## 2   VIC Australia                  9  0.66 0.6600 0.825000 0.7012500   302
## 3   NSW Australia                  8  0.55 0.6875 0.859375 0.7304688   609
## 4   VIC Australia                  8  0.67 0.8375 0.837500 0.8375000   530
## 5   VIC Australia                  7  0.92 1.1500 1.150000 0.9775000   401
## 6   NSW Australia                 11  0.47 0.5875 0.734375 0.7343750   948
##   Rank    Value new_customer_predictions
## 1   10 1.640625                 98264.53
## 2  302 1.030000                 39059.66
## 3  609 0.762500                 31869.18
## 4  530 0.828750                 20523.02
## 5  401 0.935000                 19424.95
## 6  948 0.456875                 18924.01
```

We can export the dataframe with profit predictions to an excel file in order to work with it in other programs as well:

```
library('openxlsx')
```

```
## Warning: package 'openxlsx' was built under R version 3.6.2
```

```
write.xlsx(df_new_cust_profit_predictions, 'New_Customer_List_With_Predicted_Profits.xlsx')
```

Customer Segment Analysis:

We we that mean predicted profit of the high net worth customers is higher than the mean predicted profit of the affluent customers. But in terms of total predicted profit, the mass customers are worth the most.

```r
profit_affluent = df_new_cust_profit_predictions %>% filter(wealth_segment == 'Affluent Customer') %>% s
profit_hnw = df_new_cust_profit_predictions %>% filter(wealth_segment == 'High Net Worth') %>% select(ne
profit_mass = df_new_cust_profit_predictions %>% filter(wealth_segment == 'Mass Customer') %>% select(ne

sum(profit_affluent) / count(profit_affluent)
```

```
##          n
## 1 5832.878
```

```r
sum(profit_hnw) / count(profit_hnw)
```

```
##          n
## 1 7331.831
```

```r
sum(profit_mass) / count(profit_mass)
```

```
##          n
## 1 6296.755
```

Concluding Remarks

One drawback of the model is that we need complete data in each row for all the columns used in the model in order to make predicions. We have seen that this resulted in dropping 71 customers from the new customer list for whom there was incomplete data in the necessary columns. The danger is that the model will not rank these customers and there may be high value customers among those who were dropped from the set used to make predictions.

To mitigate this, in further analysis, perhaps those rows containing missing values could be more closely assessed and a decision made about how to insert dummy values as appropriate so that these customers can be run through the model and ranked.

The model also assumes that a customer's spending will follow a consistent pattern all year as we are taking their spending over a certain period of time (which varies for each customer) and extrapolating it out to a full year. A different metric that could be used to rank customers in future analysis might be a weighted predicted yearly profit, which takes into account peaks and troughs in purchasing activity for each customer or for the 'average' customer, as determined by examination of historical business cycles for Sprocket.