Classification Project

Matthew Paz

2024-06-03

```r
setwd('/Users/mpaz/Desktop/capstone')
```

```r
library(readr)
cap_data <- read_csv('superstore_data.csv')
```

```
## Rows: 2240 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Education, Marital_Status, Dt_Customer
## dbl (19): Id, Year_Birth, Income, Kidhome, Teenhome, Recency, MntWines, MntF...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(cap_data)
```

```
## # A tibble: 6 x 22
##      Id Year_Birth Education  Marital_Status Income Kidhome Teenhome Dt_Customer
##   <dbl>      <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl> <chr>
## 1  1826       1970 Graduation Divorced        84835       0        0 6/16/2014
## 2     1       1961 Graduation Single          57091       0        0 6/15/2014
## 3 10476       1958 Graduation Married         67267       0        1 5/13/2014
## 4  1386       1967 Graduation Together        32474       1        1 11/5/2014
## 5  5371       1989 Graduation Single          21474       1        0 8/4/2014
## 6  7348       1958 PhD        Single          71691       0        0 3/17/2014
## # i 14 more variables: Recency <dbl>, MntWines <dbl>, MntFruits <dbl>,
## #   MntMeatProducts <dbl>, MntFishProducts <dbl>, MntSweetProducts <dbl>,
## #   MntGoldProds <dbl>, NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>
```

```r
dim(cap_data)
```

```
## [1] 2240   22
```

```r
class(cap_data)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```r
str(cap_data)
```

```
## spc_tbl_ [2,240 x 22] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id                 : num [1:2240] 1826 1 10476 1386 5371 ...
##  $ Year_Birth         : num [1:2240] 1970 1961 1958 1967 1989 ...
##  $ Education          : chr [1:2240] "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr [1:2240] "Divorced" "Single" "Married" "Together" ...
##  $ Income             : num [1:2240] 84835 57091 67267 32474 21474 ...
##  $ Kidhome            : num [1:2240] 0 0 0 1 1 0 0 0 0 0 ...
##  $ Teenhome           : num [1:2240] 0 0 1 1 0 0 0 1 1 1 ...
##  $ Dt_Customer        : chr [1:2240] "6/16/2014" "6/15/2014" "5/13/2014" "11/5/2014" ...
##  $ Recency            : num [1:2240] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MntWines           : num [1:2240] 189 464 134 10 6 336 769 78 384 384 ...
##  $ MntFruits          : num [1:2240] 104 5 11 0 16 130 80 0 0 0 ...
##  $ MntMeatProducts    : num [1:2240] 379 64 59 1 24 411 252 11 102 102 ...
##  $ MntFishProducts    : num [1:2240] 111 7 15 0 11 240 15 0 21 21 ...
##  $ MntSweetProducts   : num [1:2240] 189 0 2 0 0 32 34 0 32 32 ...
##  $ MntGoldProds       : num [1:2240] 218 37 30 0 34 43 65 7 5 5 ...
##  $ NumDealsPurchases  : num [1:2240] 1 1 1 1 2 1 1 1 3 3 ...
##  $ NumWebPurchases    : num [1:2240] 4 7 3 1 3 4 10 2 6 6 ...
##  $ NumCatalogPurchases: num [1:2240] 4 3 2 0 1 7 10 1 2 2 ...
##  $ NumStorePurchases  : num [1:2240] 6 7 5 2 2 5 7 3 9 9 ...
##  $ NumWebVisitsMonth  : num [1:2240] 1 5 2 7 7 2 6 5 4 4 ...
##  $ Response           : num [1:2240] 1 1 0 0 1 1 1 0 0 0 ...
##  $ Complain           : num [1:2240] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   Year_Birth = col_double(),
##   ..   Education = col_character(),
##   ..   Marital_Status = col_character(),
##   ..   Income = col_double(),
##   ..   Kidhome = col_double(),
##   ..   Teenhome = col_double(),
##   ..   Dt_Customer = col_character(),
##   ..   Recency = col_double(),
##   ..   MntWines = col_double(),
##   ..   MntFruits = col_double(),
##   ..   MntMeatProducts = col_double(),
##   ..   MntFishProducts = col_double(),
##   ..   MntSweetProducts = col_double(),
##   ..   MntGoldProds = col_double(),
##   ..   NumDealsPurchases = col_double(),
##   ..   NumWebPurchases = col_double(),
##   ..   NumCatalogPurchases = col_double(),
##   ..   NumStorePurchases = col_double(),
##   ..   NumWebVisitsMonth = col_double(),
##   ..   Response = col_double(),
##   ..   Complain = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
colSums(is.na(cap_data))
```

```
##                  Id          Year_Birth           Education      Marital_Status 
##                   0                   0                   0                   0 
##              Income             Kidhome            Teenhome         Dt_Customer 
##                  24                   0                   0                   0 
##             Recency            MntWines          MntFruits     MntMeatProducts 
##                   0                   0                   0                   0 
##     MntFishProducts    MntSweetProducts        MntGoldProds   NumDealsPurchases 
##                   0                   0                   0                   0 
##     NumWebPurchases NumCatalogPurchases   NumStorePurchases   NumWebVisitsMonth 
##                   0                   0                   0                   0 
##            Response            Complain 
##                   0                   0
```

```r
# Calculate the total number of rows
totalRows <- nrow(cap_data)

# Calculate the number of missing values in each column
missing <- colSums(is.na(cap_data))

# Calculate the percentage of missing values relative to total rows for each column
(percentage_missing <- (colSums(is.na(cap_data)) / nrow(cap_data)) * 100)
```

```
##                  Id          Year_Birth           Education      Marital_Status 
##            0.000000            0.000000            0.000000            0.000000 
##              Income             Kidhome            Teenhome         Dt_Customer 
##            1.071429            0.000000            0.000000            0.000000 
##             Recency            MntWines          MntFruits     MntMeatProducts 
##            0.000000            0.000000            0.000000            0.000000 
##     MntFishProducts    MntSweetProducts        MntGoldProds   NumDealsPurchases 
##            0.000000            0.000000            0.000000            0.000000 
##     NumWebPurchases NumCatalogPurchases   NumStorePurchases   NumWebVisitsMonth 
##            0.000000            0.000000            0.000000            0.000000 
##            Response            Complain 
##            0.000000            0.000000
```

```r
library(dplyr)
```

```
## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union
```

```r
cap_data <- cap_data %>%
  mutate(NumOfDependents = Kidhome + Teenhome)
```

```r
#Dropping Columns
cap_data <- subset(cap_data, select = -c(Id, Complain, Dt_Customer))
```

```r
#Removing missing values
cap_data <- na.omit(cap_data)
```

```r
# Find duplicated rows
(duplicated_rows <- sum(duplicated(cap_data)))
```

```
## [1] 182
```

```r
#remove duplicate rows across entire data frame
cap_data %>%
  distinct(.keep_all = TRUE)
```

```
## # A tibble: 2,034 x 20
##    Year_Birth Education  Marital_Status Income Kidhome Teenhome Recency MntWines
##         <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl>   <dbl>    <dbl>
##  1       1970 Graduation Divorced        84835       0        0       0      189
##  2       1961 Graduation Single          57091       0        0       0      464
##  3       1958 Graduation Married         67267       0        1       0      134
##  4       1967 Graduation Together        32474       1        1       0       10
##  5       1989 Graduation Single          21474       1        0       0        6
##  6       1958 PhD        Single          71691       0        0       0      336
##  7       1954 2n Cycle   Married         63564       0        0       0      769
##  8       1967 Graduation Together        44931       0        1       0       78
##  9       1954 PhD        Married         65324       0        1       0      384
## 10       1947 2n Cycle   Married         81044       0        0       0      450
## # i 2,024 more rows
## # i 12 more variables: MntFruits <dbl>, MntMeatProducts <dbl>,
## #   MntFishProducts <dbl>, MntSweetProducts <dbl>, MntGoldProds <dbl>,
## #   NumDealsPurchases <dbl>, NumWebPurchases <dbl>, NumCatalogPurchases <dbl>,
## #   NumStorePurchases <dbl>, NumWebVisitsMonth <dbl>, Response <dbl>,
## #   NumOfDependents <dbl>
```

```r
(data_types <- table(sapply(cap_data, class)))
```

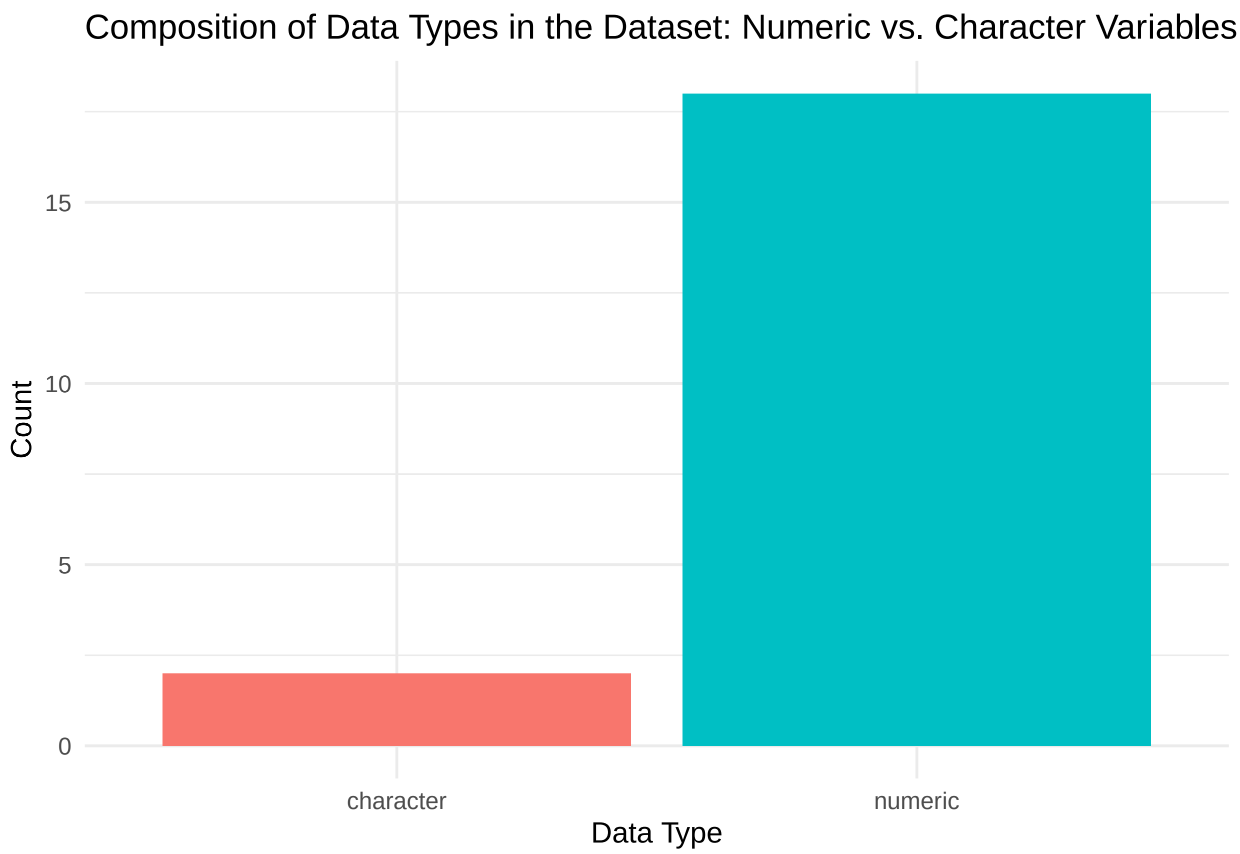
```
##
## character   numeric
##         2        18
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

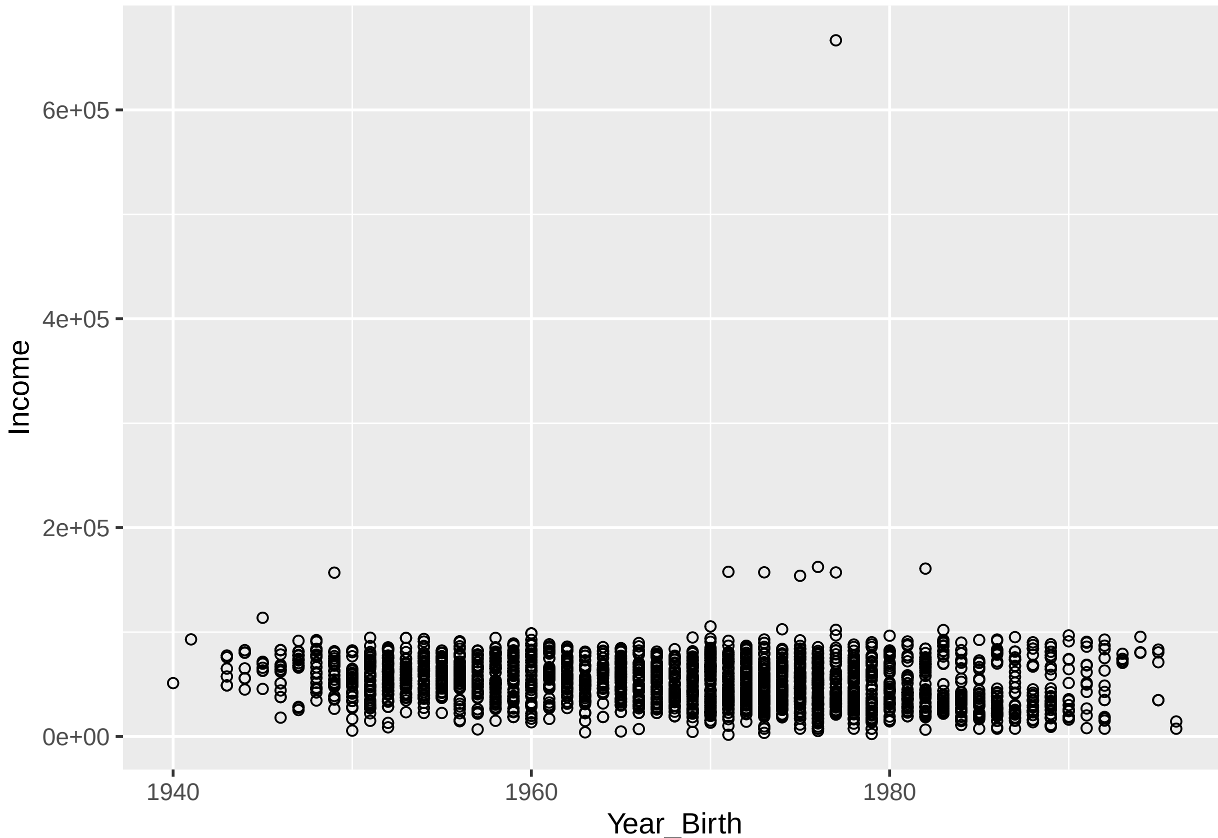
```r
# Convert the table to a data frame
data_types_df <- as.data.frame(data_types)
names(data_types_df) <- c("Data_Type", "Count")

# Create a bar plot
ggplot(data_types_df, aes(x = Data_Type, y = Count, fill = Data_Type)) +
  geom_bar(stat = "identity") +
  labs(title = "Composition of Data Types in the Dataset: Numeric vs. Character Variables", x = "Data Ty
  theme_minimal() +
  guides(fill = FALSE)
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
#Removing consumers who were born before 1900
cap_data <- cap_data %>%
  filter(Year_Birth > 1900)
```

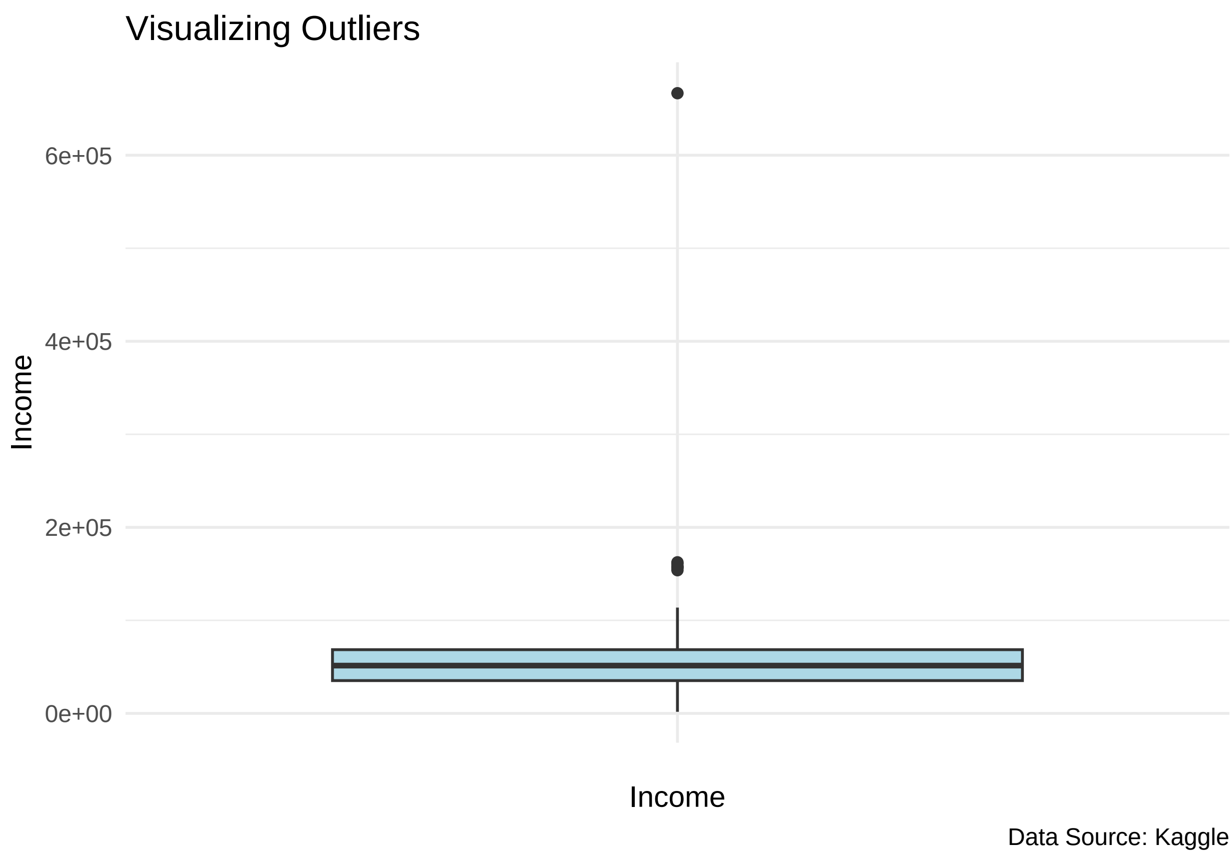
```r
ggplot(cap_data, aes(Year_Birth, Income)) +
   geom_point(shape=1)
```




```r
#Checking outliers
boxplot.stats(cap_data$Income)$out
```

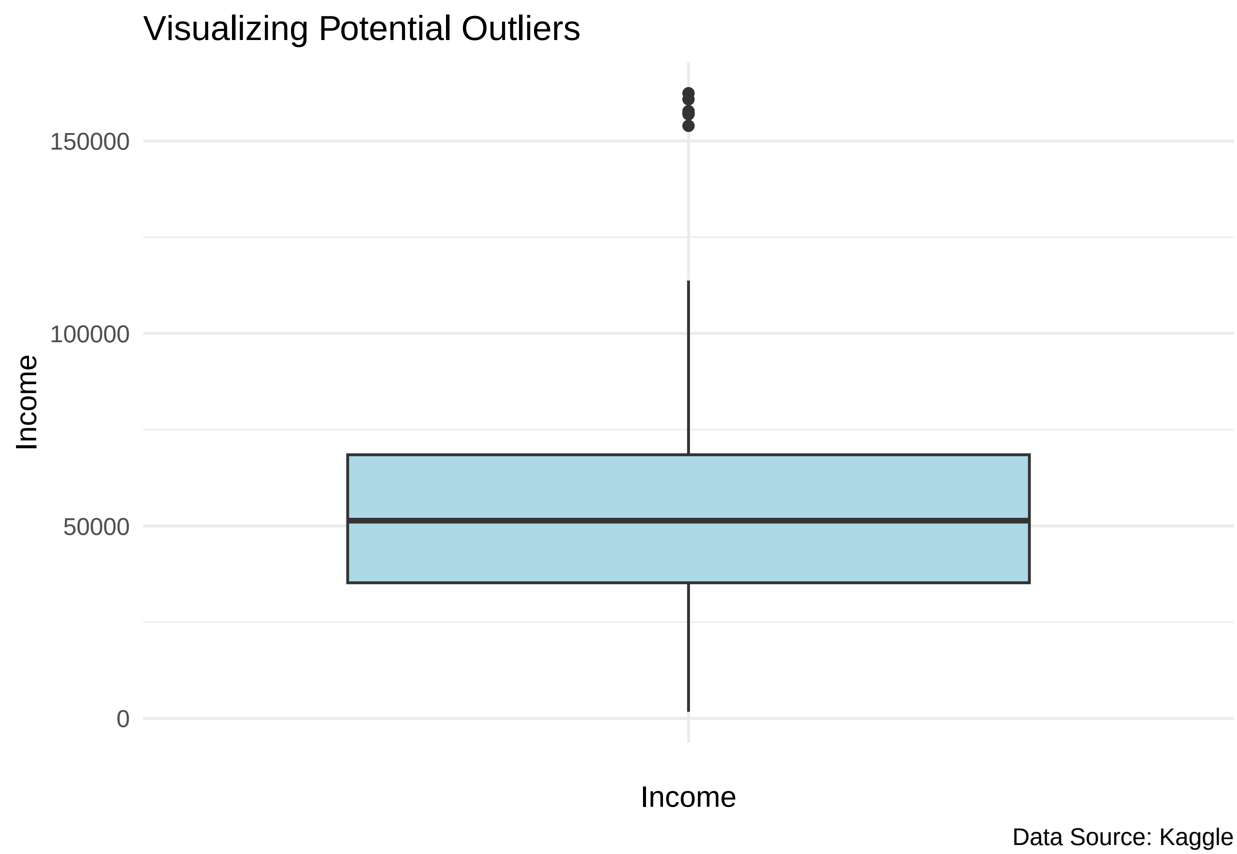
```
## [1] 157146 160803 666666 162397 157733 153924 156924 157243
```

```r
#Visualizing Outliers
ggplot(cap_data, aes("",Income)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Visualizing Outliers", x = "Income", caption = "Data Source: Kaggle") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_minimal()
```




```
max_income <- max(cap_data$Income)

# Remove row with maximum income
cap_data <- cap_data[cap_data$Income != max_income, ]
```

```
ggplot(cap_data, aes("",Income)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Visualizing Potential Outliers", x = "Income", caption = "Data Source: Kaggle") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_minimal()
```




```r
unique(cap_data$Marital_Status)
```

```
## [1] "Divorced" "Single"   "Married"  "Together" "Widow"    "YOLO"     "Alone"
## [8] "Absurd"
```

```r
unique(cap_data$Education)
```

```
## [1] "Graduation" "PhD"        "2n Cycle"   "Master"     "Basic"
```

```r
#Removing the values YOLO and Absurd from the Marital Status
cap_data <- filter(cap_data, !(Marital_Status %in% c("YOLO", "Absurd")))
```

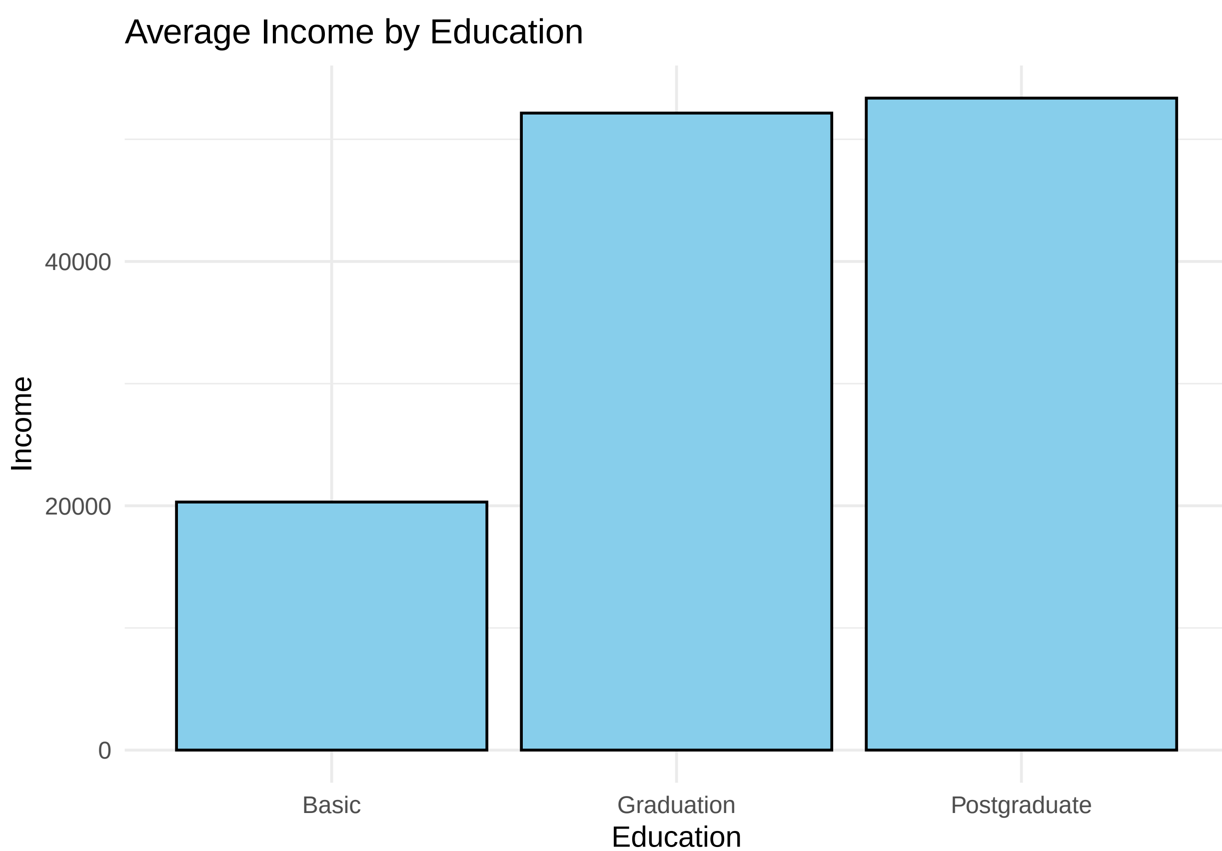
```r
cap_data <- cap_data %>%
  mutate(Marital_Status = case_when(
    Marital_Status %in% c("Alone", "Divorced", "Widow") ~ "Single",
    TRUE ~ Marital_Status  # Keep other values unchanged
  ))
```

```r
#Updating Education
cap_data <- cap_data %>%
  mutate(Education = case_when(
    Education %in% c("PhD", "2n Cycle", "Master") ~ "Postgraduate",
    TRUE ~ as.character(Education)
  ))
```

```r
#Visualizing Average Salary based on education
(avg_salary <- cap_data %>%
  select(Education, Income) %>%
   group_by(Education) %>%
  summarize(Avg_Income = mean(Income)))
```

```
## # A tibble: 3 x 2
##   Education   Avg_Income
##   <chr>            <dbl>
## 1 Basic           20306.
## 2 Graduation      52145.
## 3 Postgraduate    53370.
```

```r
ggplot(avg_salary, aes(Education, Avg_Income)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Average Income by Education", x = "Education", y = "Income") +
  theme_minimal()
```



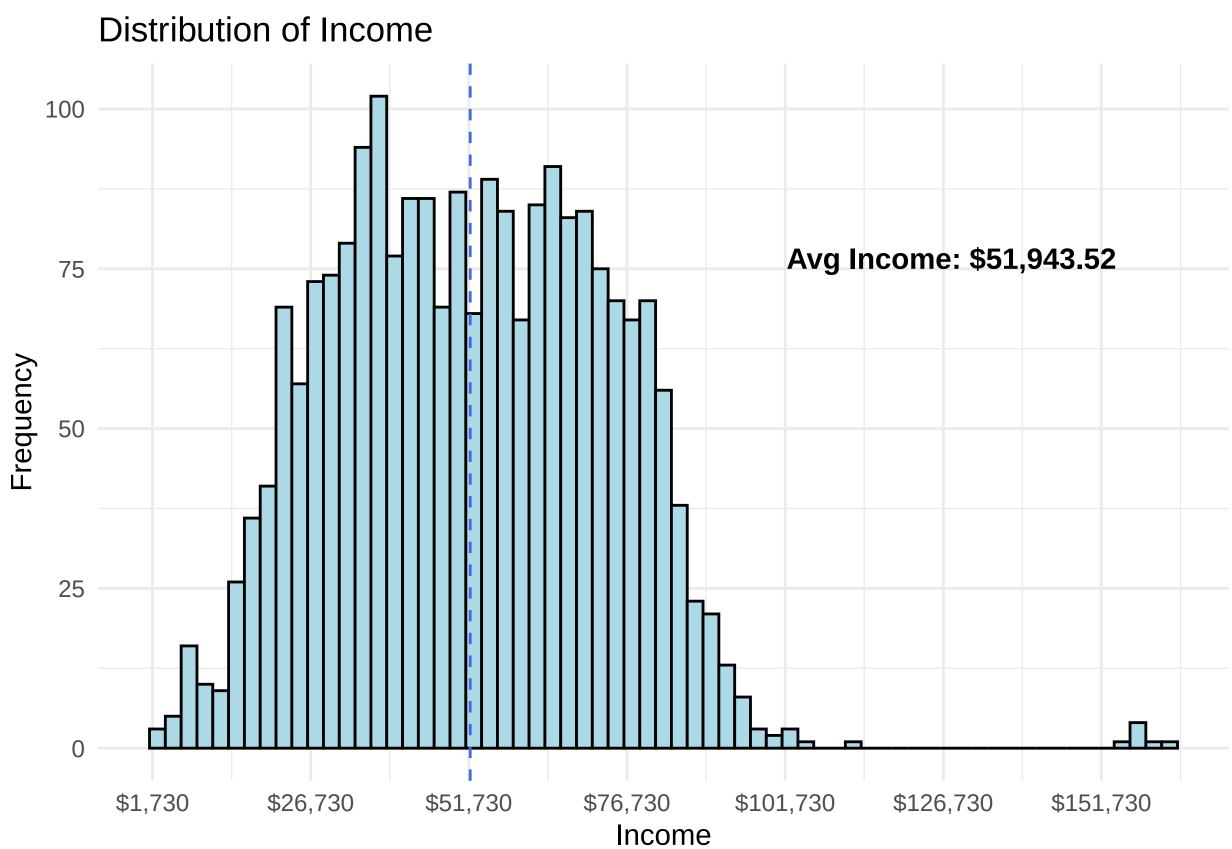

```r
summary(cap_data$Income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1730   35196   51371   51944   68487  162397
```

```r
#Measuring skewness
library(moments)


skewness(cap_data$Income)
```

```
## [1] 0.349063
```

```r
options(scipen=999)

mean_income <- mean(cap_data$Income)

# Create a histogram with customized x-axis labels
ggplot(cap_data, aes(x = Income)) +
  geom_histogram(binwidth=2500, color = "black", fill= "lightblue") +
  scale_x_continuous(breaks = seq(1730, 200000, by = 25000),
                     labels = scales::dollar_format(prefix = "$")) +
  labs(title = "Distribution of Income", x = "Income", y = "Frequency") +
  theme_minimal() +
   geom_vline(xintercept = mean_income, color = "royalblue", linetype = "dashed") +
  annotate("text", x = mean_income + 50000, y = 75, label = paste("Avg Income:", scales::dollar(mean_in
           color = "black", hjust = 0, vjust = 0, fontface = 'bold')
```



##

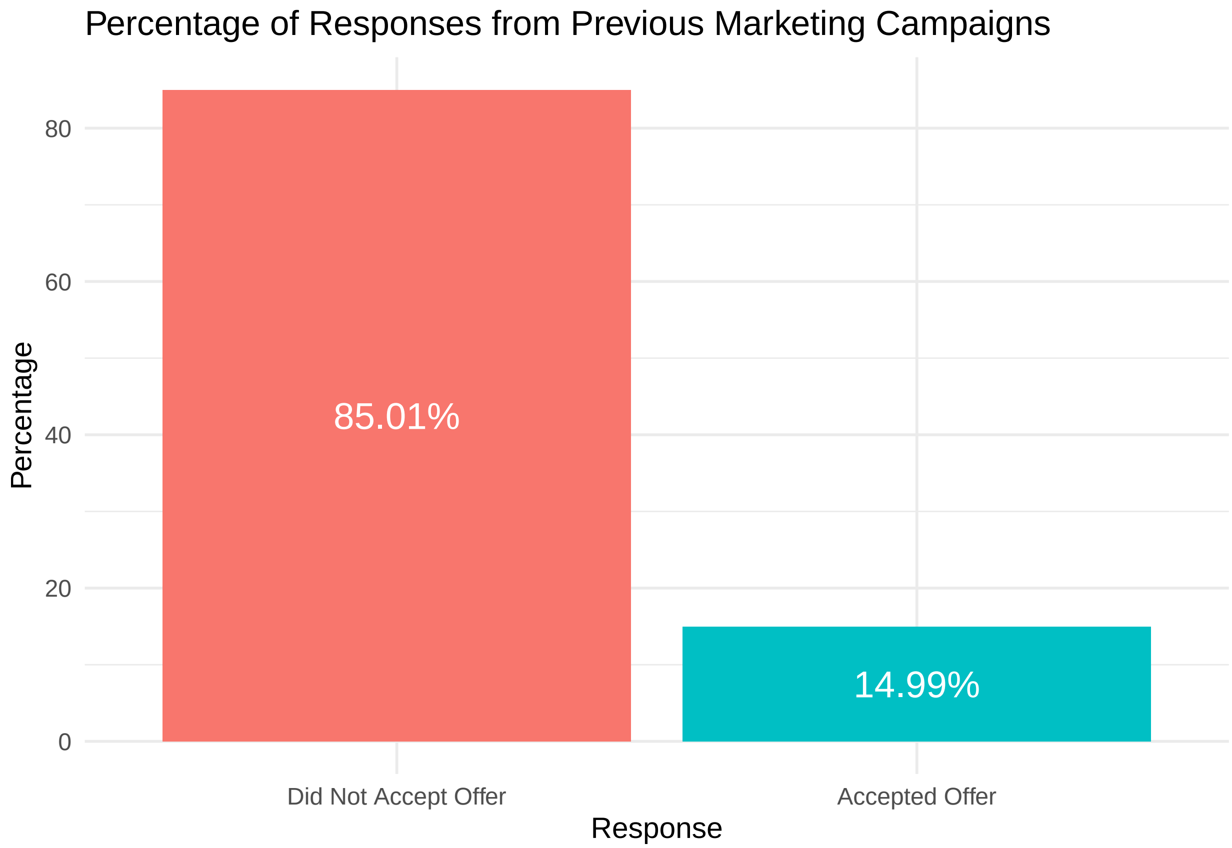
```r
table(cap_data$Response)
```

```
##
##    0    1
## 1877  331
```

```r
original_table <- data.frame(Response = c(0, 1),
                             Count = c(1877, 331))

# Calculate total count
total_count <- sum(original_table$Count)

# Calculate percentages
original_table$Percentage <- round((original_table$Count / total_count) * 100, 2)

# New table with percentages
percentage_table <- original_table
percentage_table$Count <- NULL  # Remove the count column

# Print the percentage table
print(percentage_table)
```

```
##   Response Percentage
## 1        0      85.01
## 2        1      14.99
```

```r
# Bar plot
ggplot(percentage_table, aes(x = factor(Response), y = Percentage, fill = factor(Response))) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(Percentage, "%")),
            position = position_stack(vjust = 0.5),
            size = 5,
            color = "white") +
  labs(title = "Percentage of Responses from Previous Marketing Campaigns",
       x = "Response",
       y = "Percentage", ) +
  theme_minimal() +
  guides(fill = FALSE) +
  scale_x_discrete(labels = c("0" = "Did Not Accept Offer", "1" = "Accepted Offer"))
```




```
(avg_income_marital <- cap_data %>%
  group_by(Marital_Status, Education) %>%
  summarise(avg_income = mean(Income)))
```

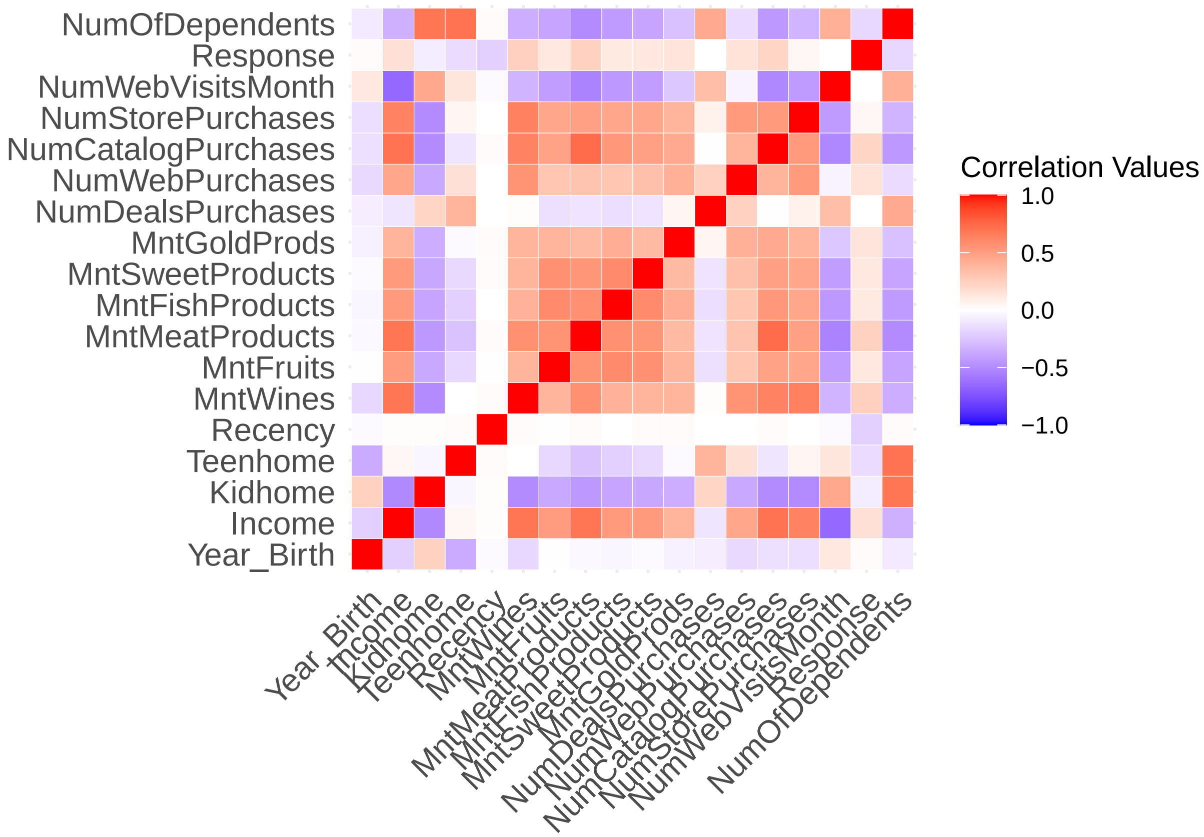
```
## `summarise()` has grouped output by 'Marital_Status'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 9 x 3
## # Groups:   Marital_Status [3]
##   Marital_Status Education    avg_income
##   <chr>          <chr>             <dbl>
## 1 Married        Basic            21960.
## 2 Married        Graduation       50800.
## 3 Married        Postgraduate     54156.
## 4 Single         Basic            17998.
## 5 Single         Graduation       52549.
## 6 Single         Postgraduate     53402.
## 7 Together       Basic            21240.
## 8 Together       Graduation       53607.
## 9 Together       Postgraduate     52152.
```

```
#Selecting numeric datatypes only
num_vals <- cap_data %>%
  select_if(is.numeric)
```

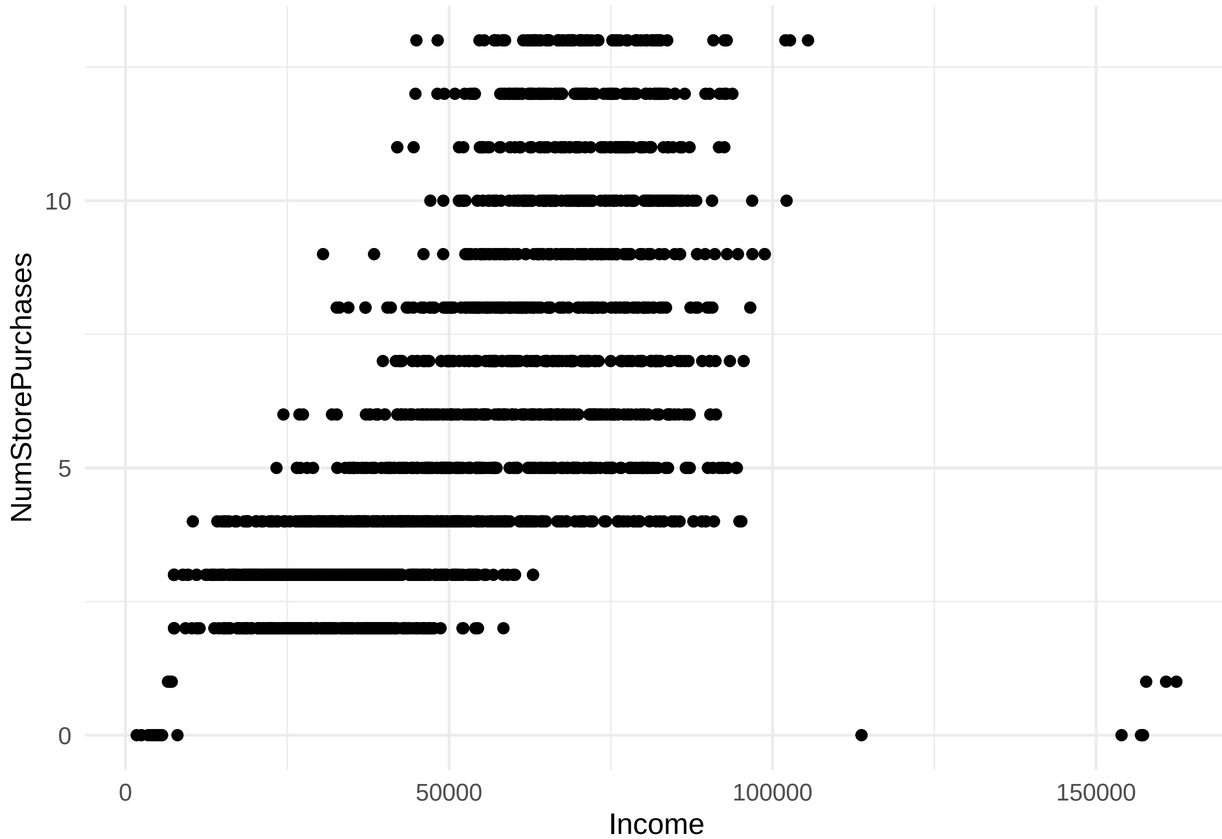
```r
corr <- cor(num_vals)
```

```r
library(ggcorrplot)

ggcorrplot(corr, outline.col = "white", legend.title = "Correlation Values",  ggtheme = ggplot2::theme_
```

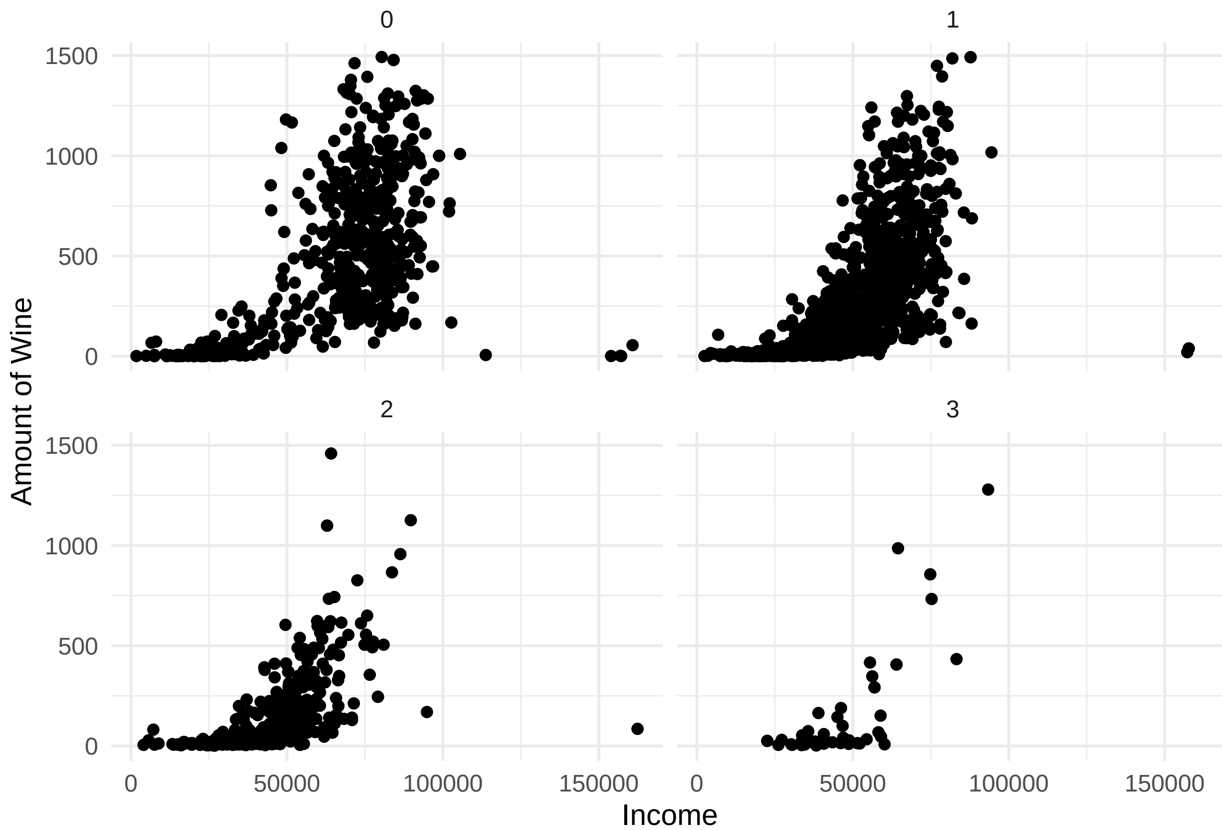


Bi and Multivariate Analysis

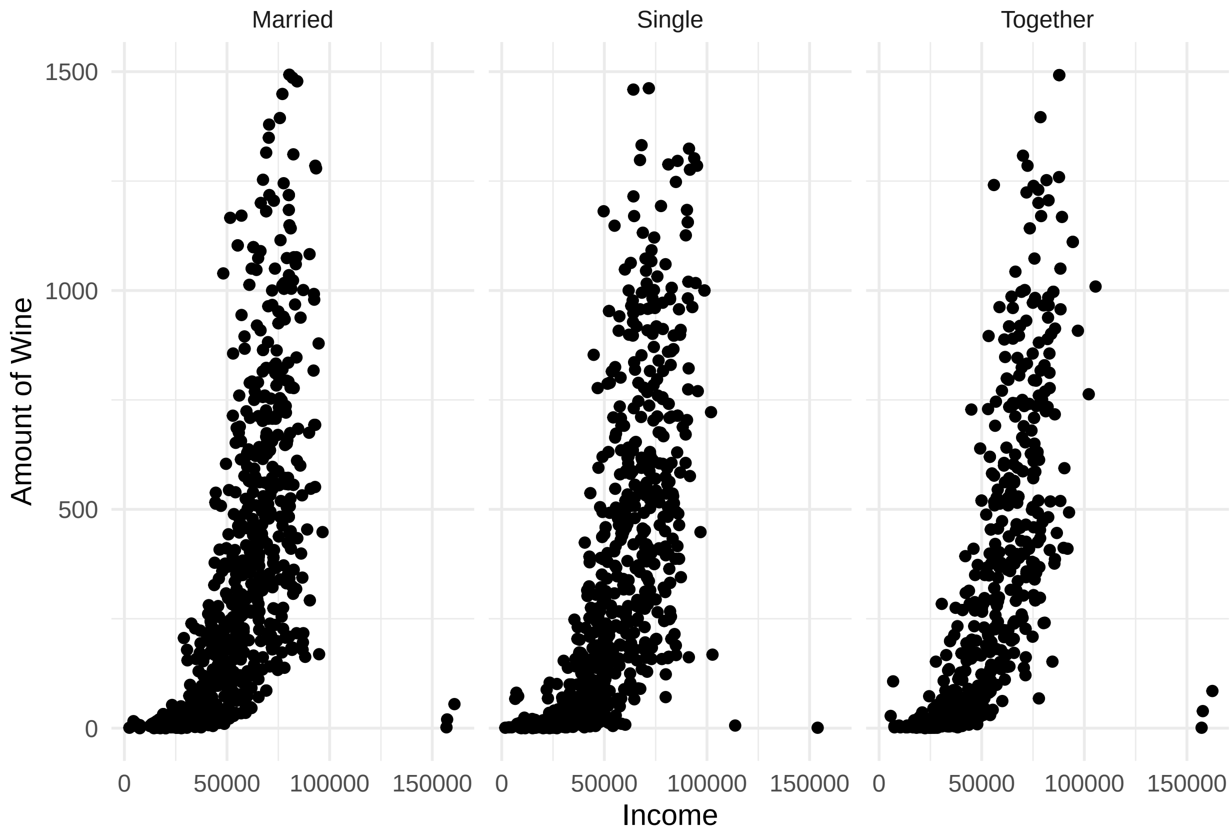
```r
ggplot(cap_data, aes(Income, NumStorePurchases)) +
  geom_point() +
  theme_minimal()
```



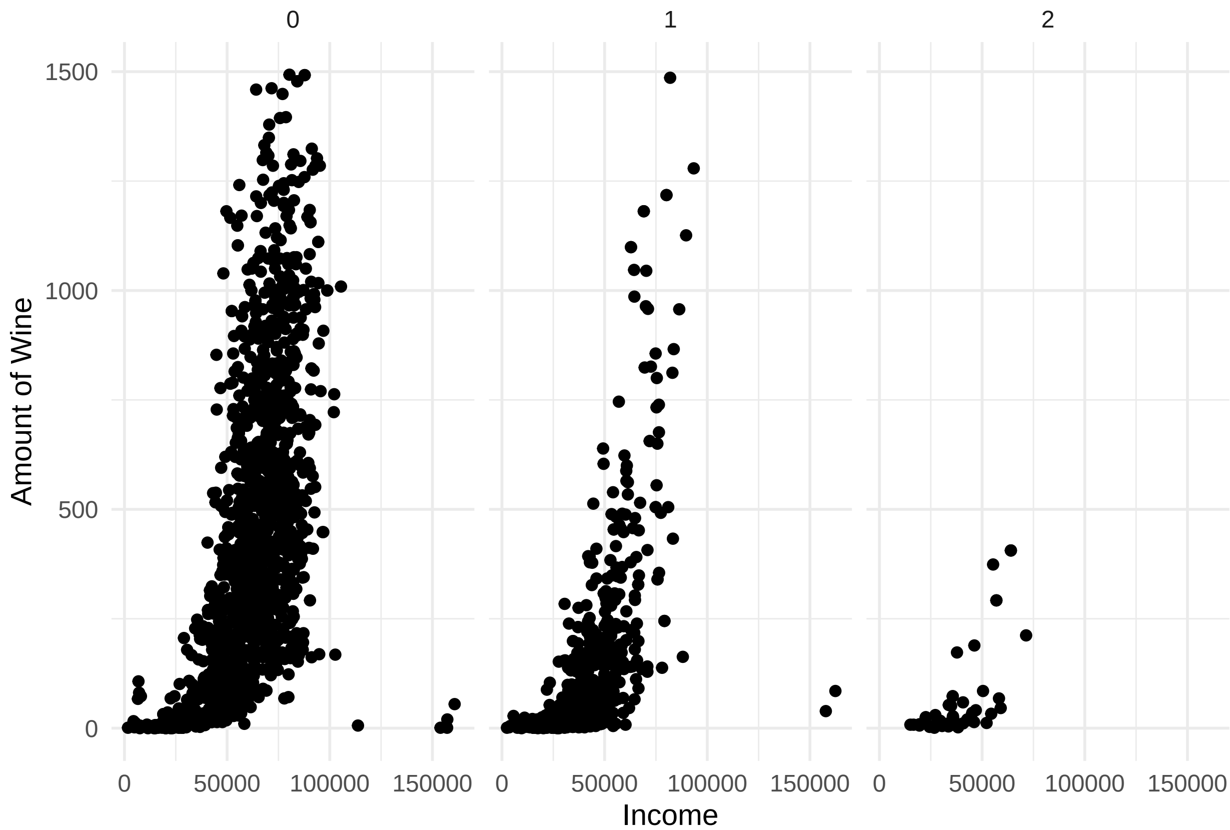

```r
ggplot(cap_data, aes(Income, MntWines)) +
  geom_point() +
  labs(y = "Amount of Wine") +
  facet_wrap(~NumOfDependents) +
  theme_minimal()
```



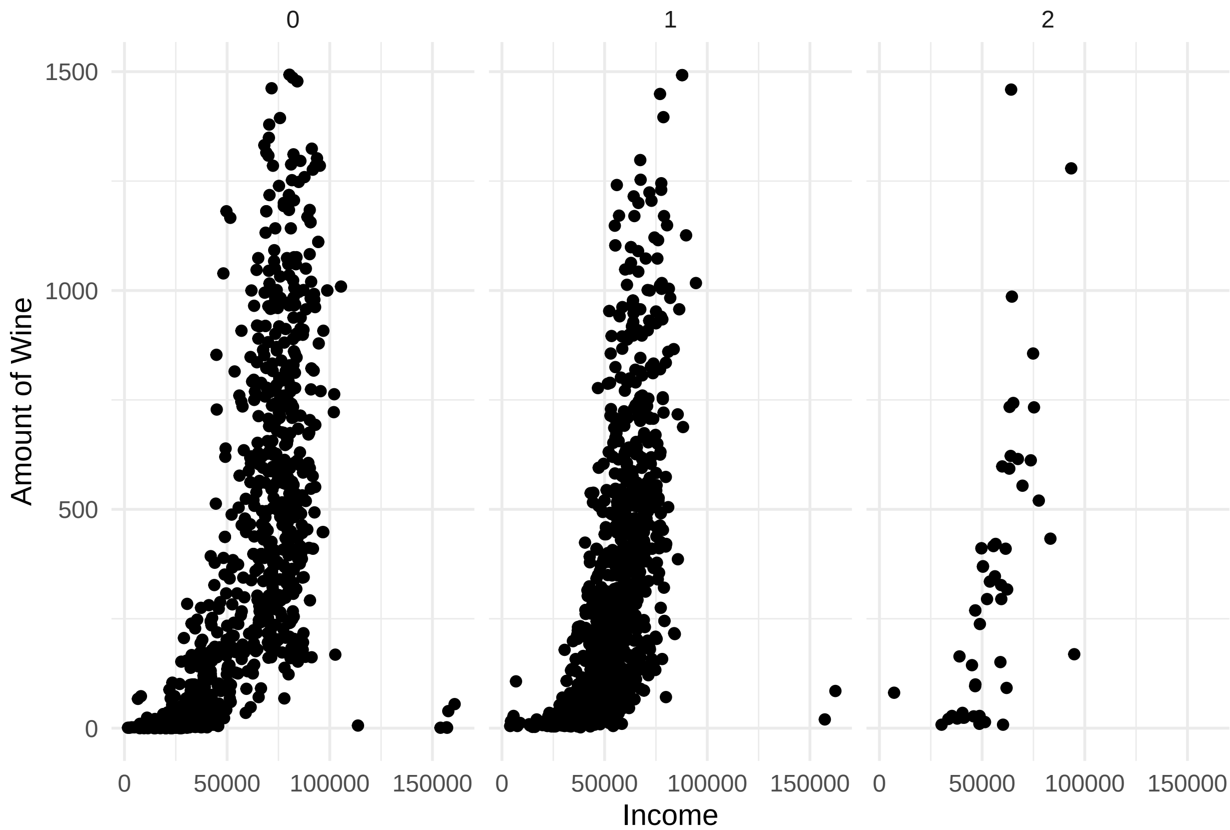

```r
ggplot(cap_data, aes(Income, MntWines)) +
  geom_point() +
  labs(y = "Amount of Wine") +
  facet_wrap(~Marital_Status) +
  theme_minimal()
```




```r
ggplot(cap_data, aes(Income, MntWines)) +
  geom_point() +
  labs(y = "Amount of Wine") +
  facet_wrap(~Kidhome) +
  theme_minimal()
```

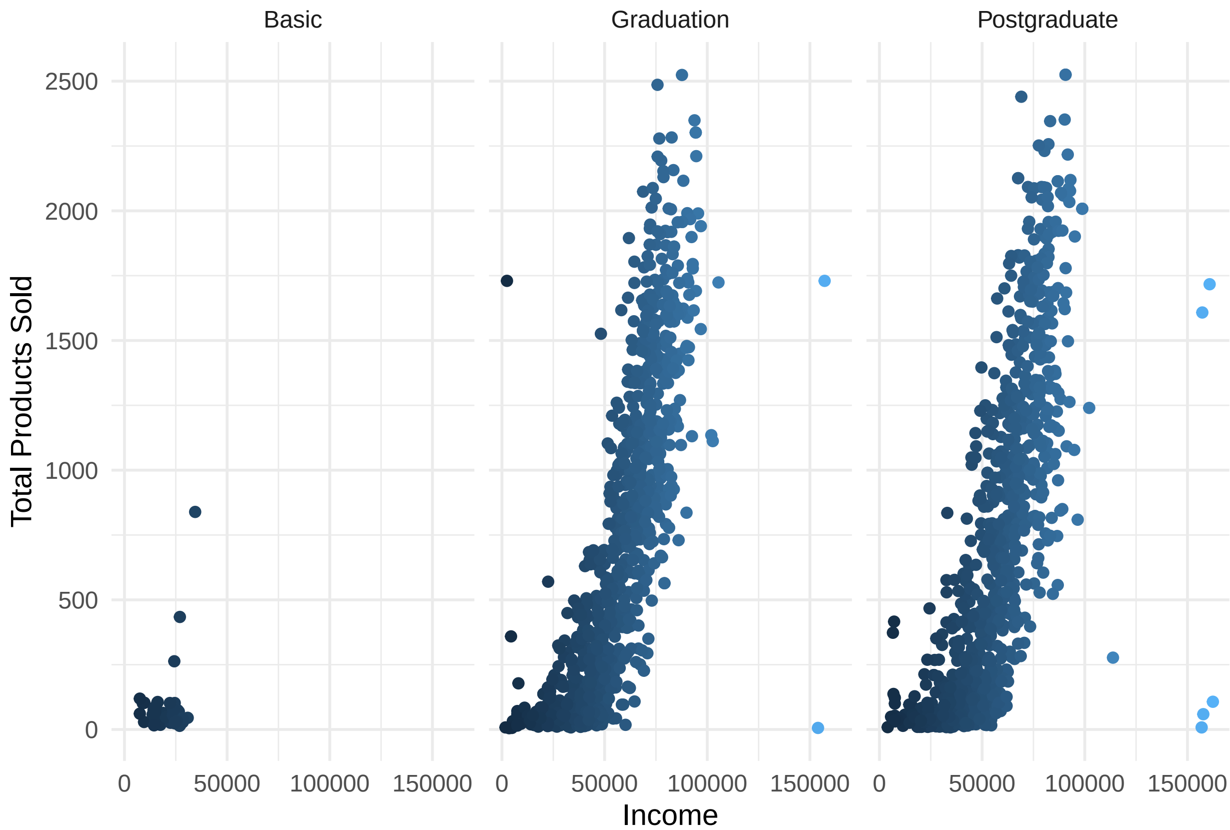



```
ggplot(cap_data, aes(Income, MntWines)) +
  geom_point() +
  labs(y = "Amount of Wine") +
  facet_wrap(~Teenhome) +
  theme_minimal()
```




```
cap_data$Total_Products <- rowSums(cap_data[, c("MntWines", "MntFruits", "MntMeatProducts",
                                                "MntFishProducts", "MntSweetProducts", "MntGoldProds")])
```

```
ggplot(cap_data, aes(Income, Total_Products, color = Income)) +
  geom_point() +
  labs(y = "Total Products Sold") +
  facet_wrap(~Education) +
  theme_minimal() +
  guides(color = FALSE)
```

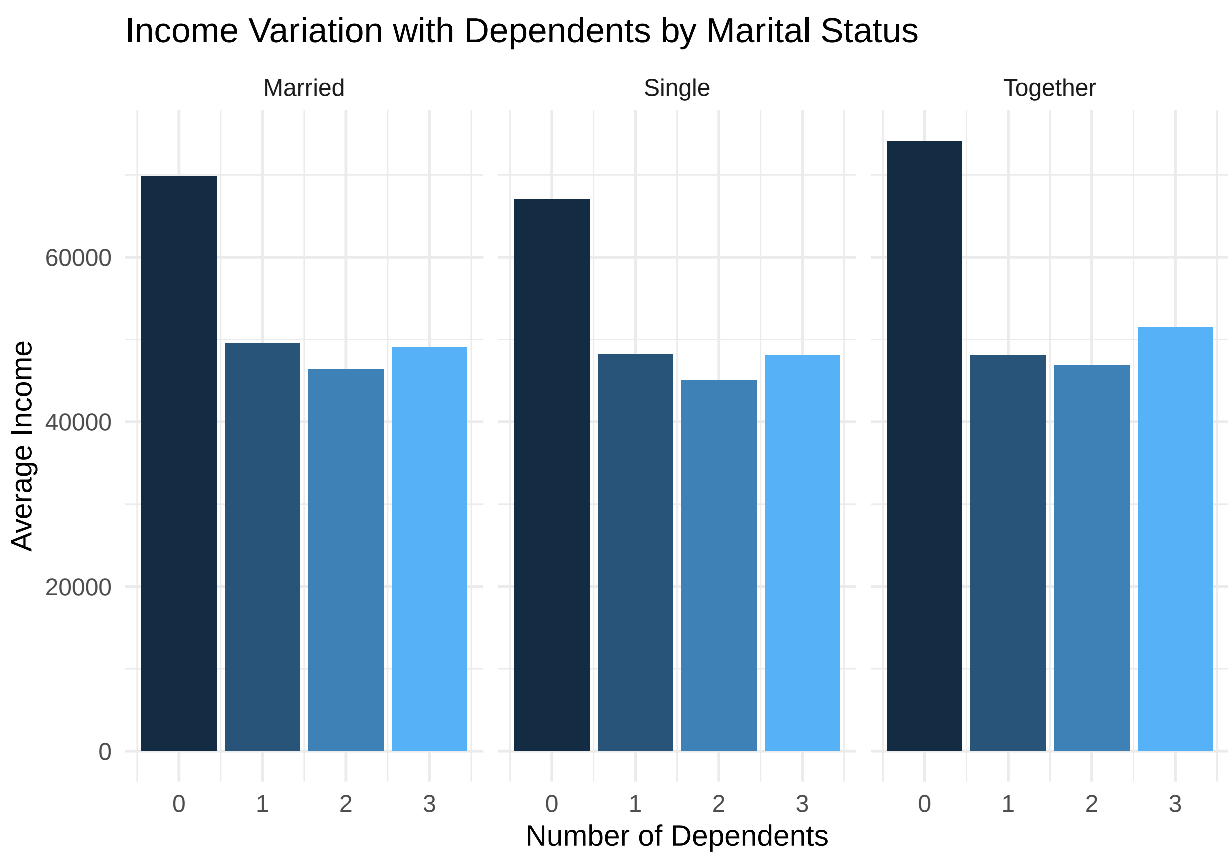



```
(education_agg <- cap_data %>%
  group_by(Education, NumOfDependents, Marital_Status) %>%
  summarize(total_recency = sum(Recency),
            mean_income = mean(Income),
            mean_recency = mean(Recency),
            totalGldPurchases = sum(MntGoldProds),
            totalWebPurchases = sum(NumWebPurchases),
            totalStorePurchases = sum(NumStorePurchases),
            totalDiscountPurchases = sum(NumDealsPurchases),
            .groups = 'drop') %>%
  arrange((mean_income)))
```

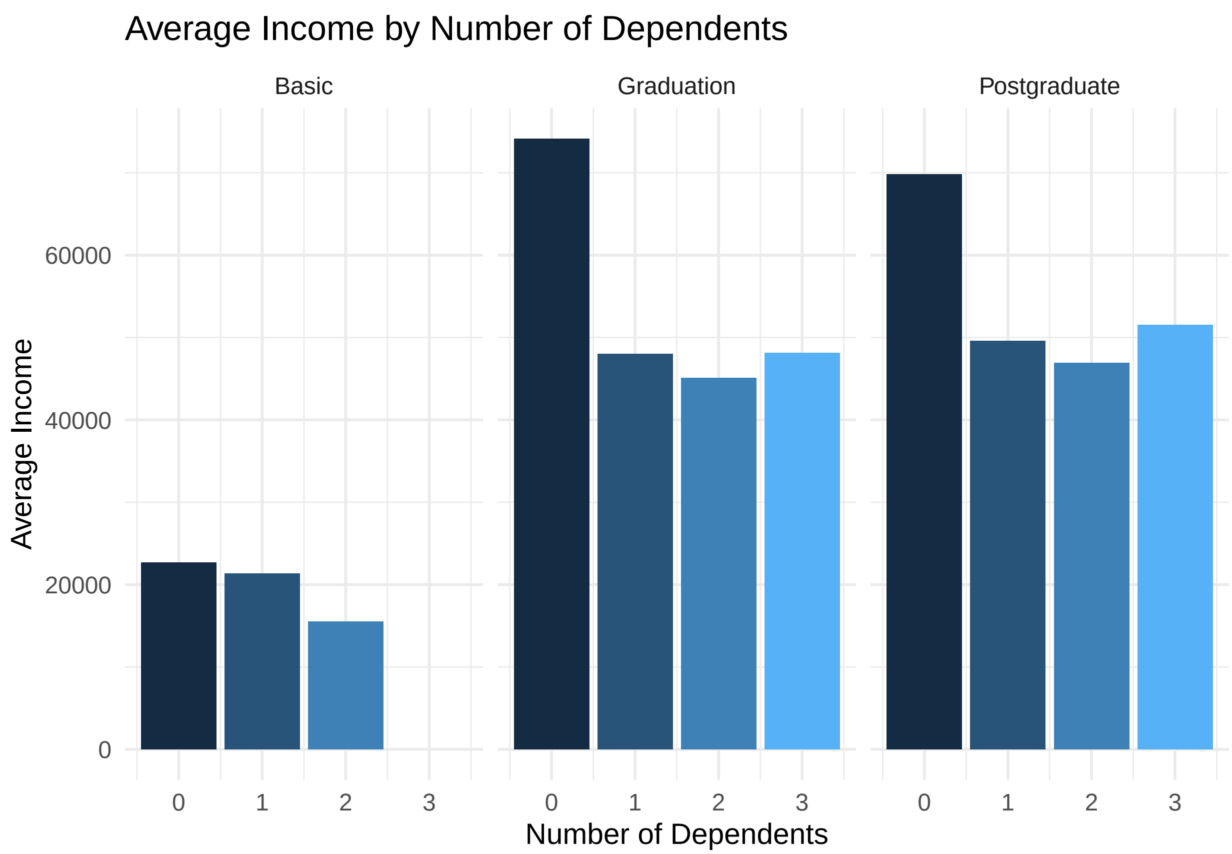
```
## # A tibble: 31 x 10
##    Education    NumOfDependents Marital_Status total_recency mean_income
##    <chr>                  <dbl> <chr>                  <dbl>       <dbl>
##  1 Basic                      2 Single                   118      15535
##  2 Basic                      0 Single                   214      17870
##  3 Basic                      1 Single                   679      18352.
##  4 Basic                      1 Together                 472      20914.
##  5 Basic                      1 Married                  356      21356.
##  6 Basic                      0 Together                 221      21826.
##  7 Basic                      0 Married                  556      22699.
##  8 Graduation                 2 Together                2700      41643.
##  9 Graduation                 3 Married                  362      43293.
## 10 Postgraduate               2 Single                  3463      43582.
```

```
## # i 21 more rows
## # i 5 more variables: mean_recency <dbl>, totalGldPurchases <dbl>,
## #   totalWebPurchases <dbl>, totalStorePurchases <dbl>,
## #   totalDiscountPurchases <dbl>
```

```r
plot <- ggplot(education_agg, aes(x = NumOfDependents, y = mean_income, fill= NumOfDependents)) +
  geom_bar(stat = 'identity', position = "dodge") +
  labs(title = "Income Variation with Dependents by Marital Status",
       x = "Number of Dependents",
       y = "Average Income") +
  facet_wrap(~Marital_Status) +
  theme_minimal() +
  theme(legend.position="none")

# Display the plot
print(plot)
```




```r
ggplot(education_agg, aes(x = NumOfDependents, y = mean_income, fill= NumOfDependents)) +
  geom_bar(stat = 'identity', position = "dodge") +
  labs(title = "Average Income by Number of Dependents",
       x = "Number of Dependents",
       y = "Average Income") +
  facet_wrap(~Education) +
  theme_minimal() +
  theme(legend.position="none")
```




```r
# Convert 'Education' to factor
cap_data$Education <- factor(cap_data$Education, levels = c("Basic", "Graduation", "Postgraduate"))
```

```r
# Convert 'Marital_Status' to factor
cap_data$Marital_Status <- factor(cap_data$Marital_Status, levels = c("Single","Married","Together"))
```

```r
# Convert 'Response' to factor
cap_data$Response <- as.factor(cap_data$Response)
```

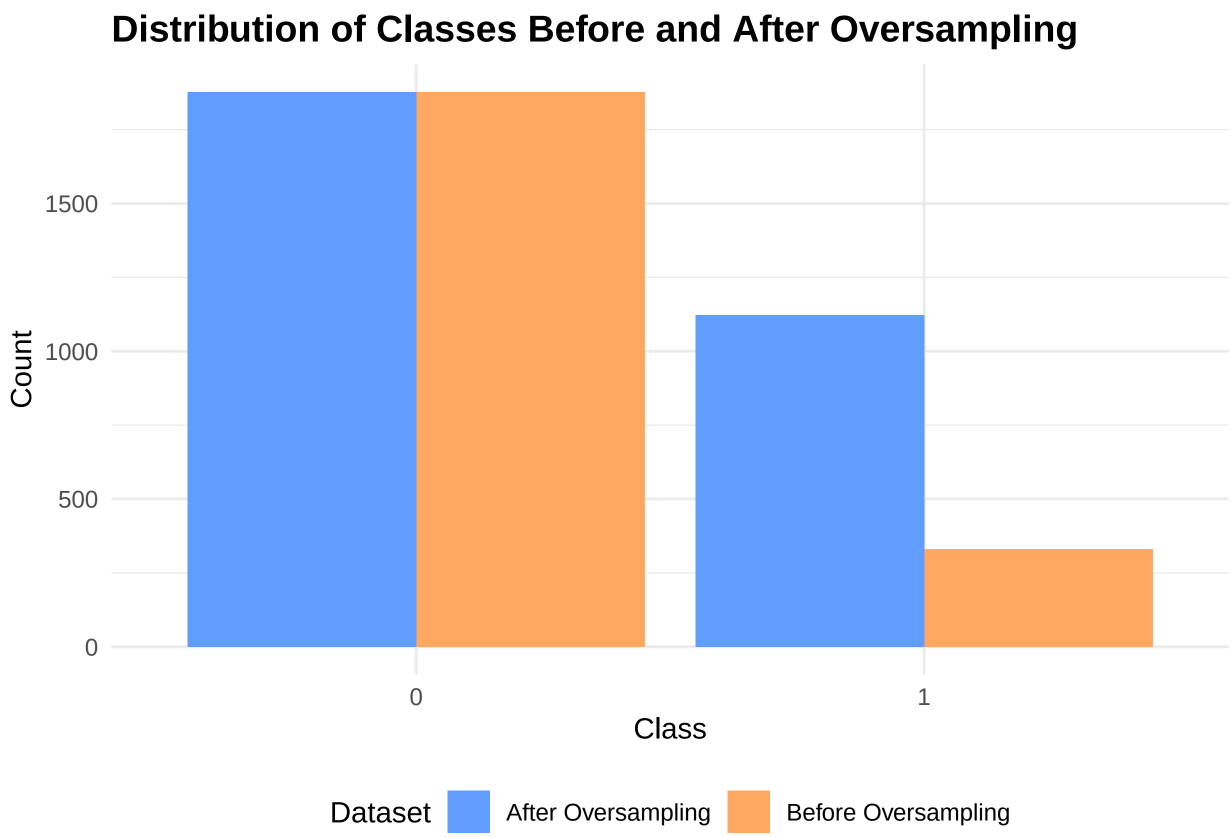
```r
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
# Perform oversampling
oversampled_data <- ovun.sample(Response ~ ., data = cap_data, method = "over", N = 3000)$data
```

```r
# Before oversampling
class_distribution_before <- table(cap_data$Response)
df_before <- data.frame(Class = names(class_distribution_before), Count = as.vector(class_distribution_b

# After oversampling
class_distribution_after <- table(oversampled_data$Response)
df_after <- data.frame(Class = names(class_distribution_after), Count = as.vector(class_distribution_aft
```

```r
# Combine dataframes
combined_df <- rbind(df_before, df_after)

# Plot
ggplot(combined_df, aes(x = Class, y = Count, fill = Dataset)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Distribution of Classes Before and After Oversampling",
       x = "Class",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14)) +
  theme(legend.position = "bottom") +
  scale_fill_manual(values = c("#619CFF", "#FFA861"))
```



Modeling and Evaluation

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:moments':
##
##     kurtosis, moment, skewness
```

```r
set.seed(123)  # for reproducibility
train_index_two <- createDataPartition(oversampled_data$Response, p = 0.8, list = FALSE)
train_data_two <- oversampled_data[train_index_two, ]
test_data_two <- oversampled_data[-train_index_two, ]

# Train Naive Bayes model
nb_model_two <- naiveBayes(Response ~ ., data = train_data_two)

# Train Logistic Regression model
logit_model_two <- glm(Response ~ ., data = train_data_two, family = "binomial")

# Train SVM model
svm_model_two <- svm(Response ~ ., data = train_data_two)


# Make predictions
nb_predictions_two <- predict(nb_model_two, newdata = test_data_two )
#logit_predictions_two <- predict(logit_model_two , newdata = test_data_two , type = "response")
svm_predictions_two <- predict(svm_model_two, newdata = test_data_two )

# Evaluate performance
nb_performance_two <- confusionMatrix(nb_predictions_two, test_data_two$Response)
#logit_performance_two <- confusionMatrix(logit_predictions_two, test_data_two$Response)
svm_performance_two <- confusionMatrix(svm_predictions_two, test_data_two$Response)
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
logit_predictions_two <- predict(logit_model_two , newdata = test_data_two, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```r
confusion_matrix_logit_two <- table(test_data_two$Response, logit_predictions_two > 0.5)
accuracy_logit_two <- sum(diag(confusion_matrix_logit_two)) / sum(confusion_matrix_logit_two)
precision_logit_two <- confusion_matrix_logit_two[2, 2] / sum(confusion_matrix_logit_two[, 2])
recall_logit_two <- confusion_matrix_logit_two[2, 2] / sum(confusion_matrix_logit_two[2, ])
f1_score_logit_two <- 2 * (precision_logit_two * recall_logit_two ) / (precision_logit_two + recall_log
roc_auc_logit_two <- roc(test_data_two$Response, logit_predictions_two)$auc
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Print performance metrics
print(confusion_matrix_logit_two )
```

```
##
##      FALSE TRUE
##   0    311   64
##   1     89  135
```

```r
print(paste("Accuracy:", accuracy_logit_two))
```

```
## [1] "Accuracy: 0.74457429048414"
```

```r
print(paste("Precision:", precision_logit_two))
```

```
## [1] "Precision: 0.678391959798995"
```

```r
print(paste("Recall:", recall_logit_two))
```

```
## [1] "Recall: 0.602678571428571"
```

```r
print(paste("F1-Score:", f1_score_logit_two))
```

```
## [1] "F1-Score: 0.638297872340425"
```

```r
print(paste("ROC-AUC:", roc_auc_logit_two))
```

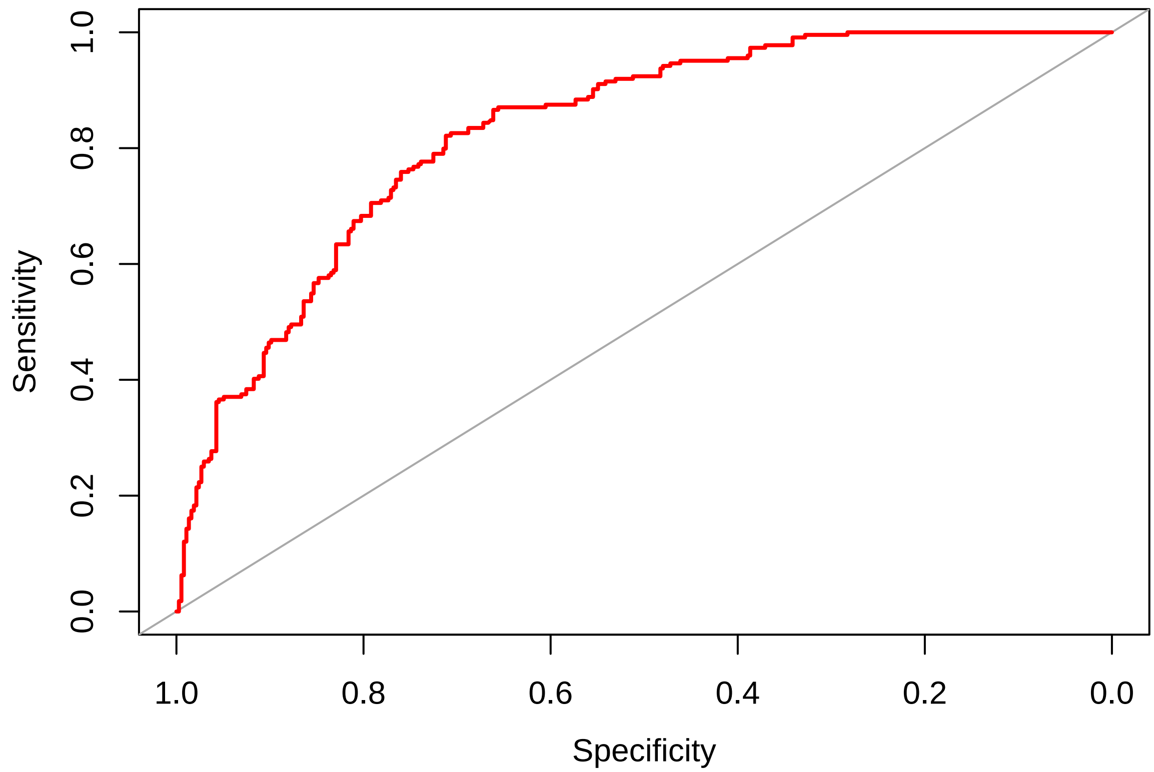
```
## [1] "ROC-AUC: 0.829779761904762"
```

```r
library(pROC)
roc_curve_logit_two <- roc(test_data_two$Response, logit_predictions_two)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc_curve_logit_two, col = "red", lwd = 2, asp = NA)
```




```r
print(nb_performance_two)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 286 101
##          1  89 123
##
##                Accuracy : 0.6828
##                  95% CI : (0.6439, 0.7199)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : 0.002136
##
##                   Kappa : 0.3152
##
##  Mcnemar's Test P-Value : 0.424857
##
##             Sensitivity : 0.7627
##             Specificity : 0.5491
##          Pos Pred Value : 0.7390
##          Neg Pred Value : 0.5802
##              Prevalence : 0.6260
##          Detection Rate : 0.4775
##    Detection Prevalence : 0.6461
```

```
##       Balanced Accuracy : 0.6559
##
##        'Positive' Class : 0
##
```

```r
accuracy_nb_two <- nb_performance_two$overall["Accuracy"]
```

```r
print(svm_performance_two)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 329  54
##          1  46 170
##
##                Accuracy : 0.8331
##                  95% CI : (0.8007, 0.8621)
##     No Information Rate : 0.626
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.6409
##
##  Mcnemar's Test P-Value : 0.4839
##
##             Sensitivity : 0.8773
##             Specificity : 0.7589
##          Pos Pred Value : 0.8590
##          Neg Pred Value : 0.7870
##              Prevalence : 0.6260
##          Detection Rate : 0.5492
##    Detection Prevalence : 0.6394
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : 0
##
```

```r
(svm_accuracy_two <- svm_performance_two$overall["Accuracy"])
```

```
##  Accuracy
## 0.8330551
```

```r
# Calculate ROC curve and AUC for Naive Bayes model
roc_nb_two <- roc(test_data_two$Response, as.numeric(nb_predictions_two))
```

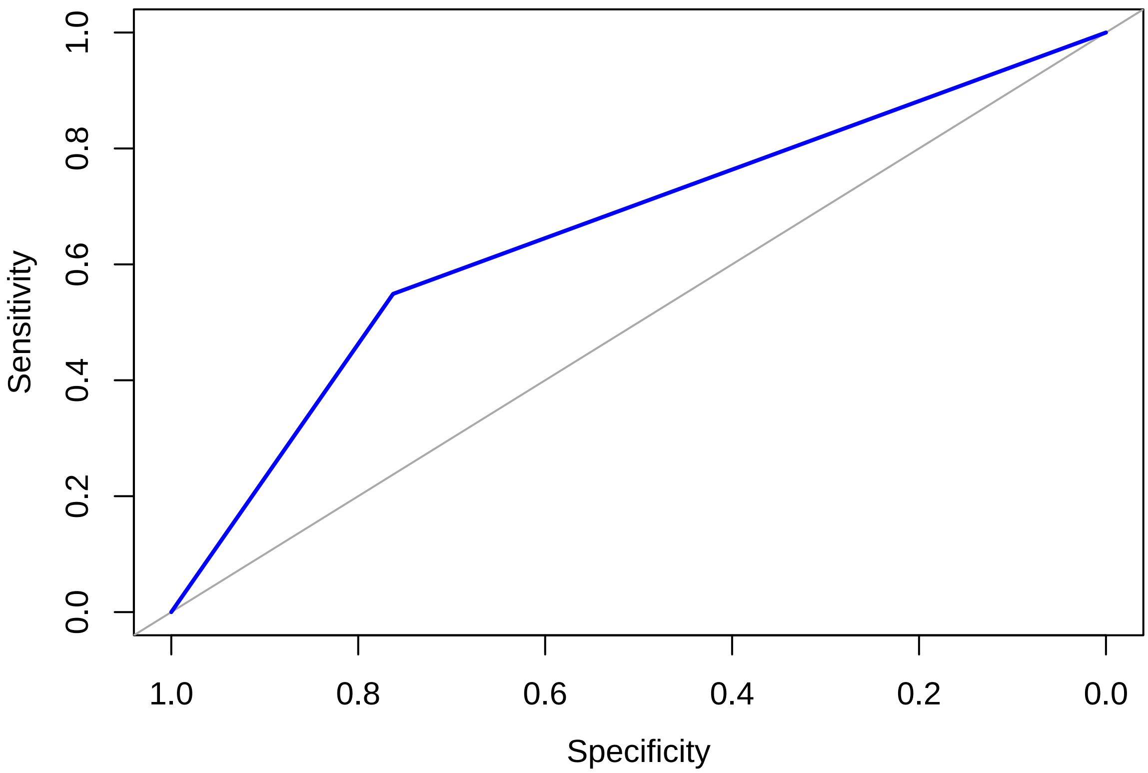
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc_nb_two <- auc(roc_nb_two)

print(paste("Naive Bayes AUC:", auc_nb_two))
```

```
## [1] "Naive Bayes AUC: 0.655886904761905"
```

```r
plot(roc_nb_two, col = "blue", lwd = 2, asp = NA)
```



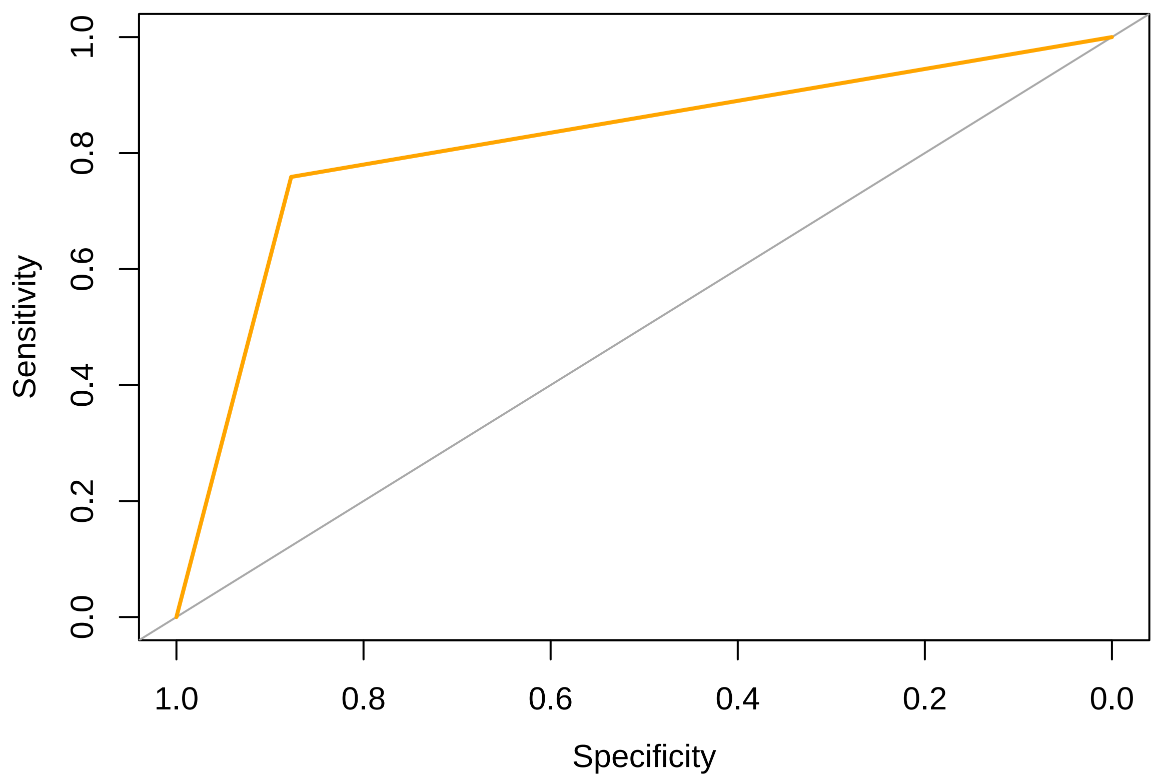

```r
# Calculate ROC curve and AUC for SVM model
roc_svm_two <- roc(test_data_two$Response, as.numeric(svm_predictions_two))
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

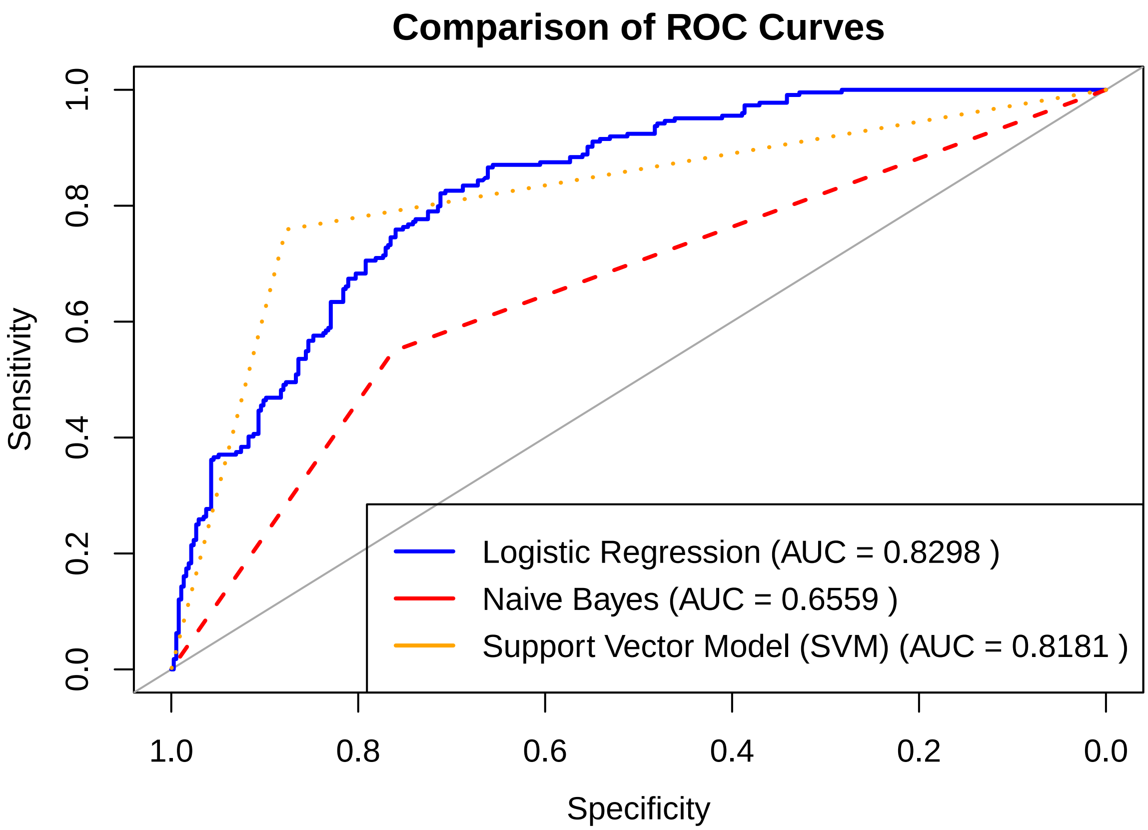
```r
auc_svm_two <- auc(roc_svm_two)

print(paste("Support Vector Model  AUC:", auc_svm_two))
```

```
## [1] "Support Vector Model  AUC: 0.818130952380952"
```

```r
plot(roc_svm_two, col = "orange", lwd = 2, asp = NA)
```




```r
# Plot ROC curves
plot(roc_curve_logit_two, col = "blue", lwd = 2, main = "Comparison of ROC Curves", cex.main = 1.2, asp
lines(roc_nb_two, col = "red", lwd = 2, lty = "dashed")
lines(roc_svm_two, col = "orange", lwd = 2, lty = "dotted")

# Adding AUC values to legend
legend("bottomright", legend = c(
  paste("Logistic Regression (AUC =", round(roc_auc_logit_two, 4), ")"),
  paste("Naive Bayes (AUC =", round(auc_nb_two, 4), ")"),
  paste("Support Vector Model (SVM) (AUC =", round(auc_svm_two, 4), ")")
), col = c("blue", "red", "orange"), lwd = 2)
```



Creating Basline Model

```
# Determine majority class in training data
majority_class <- names(sort(table(train_data_two$Response), decreasing = TRUE))[1]

# Create predictions based on majority class
majority_predictions <- rep(majority_class, nrow(train_data_two))

# Evaluate accuracy of majority classifier
(accuracy_majority <- mean(majority_predictions == oversampled_data$Response))
```

```
## Warning in `==.default`(majority_predictions, oversampled_data$Response):
## longer object length is not a multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

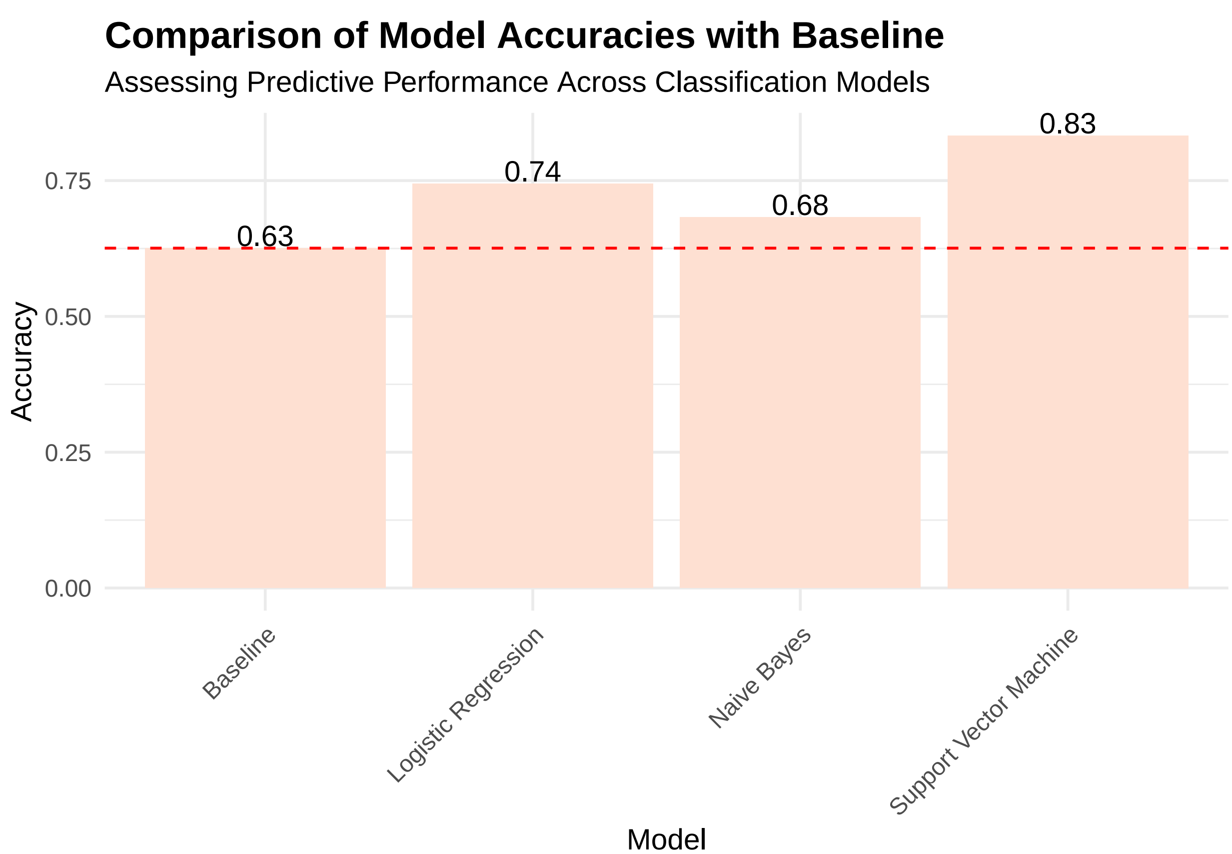
```
## [1] 0.6256667
```

```
model_names <- c("Baseline", "Logistic Regression", "Naive Bayes", "Support Vector Machine")
accuracies <- c(accuracy_majority, accuracy_logit_two, accuracy_nb_two, svm_accuracy_two)


# Combine data into a dataframe
df2 <- data.frame(Model = model_names, Accuracy = accuracies)
```

```r
head(df2)
```

```
##                    Model  Accuracy
## 1               Baseline 0.6256667
## 2    Logistic Regression 0.7445743
## 3            Naive Bayes 0.6828047
## 4 Support Vector Machine 0.8330551
```

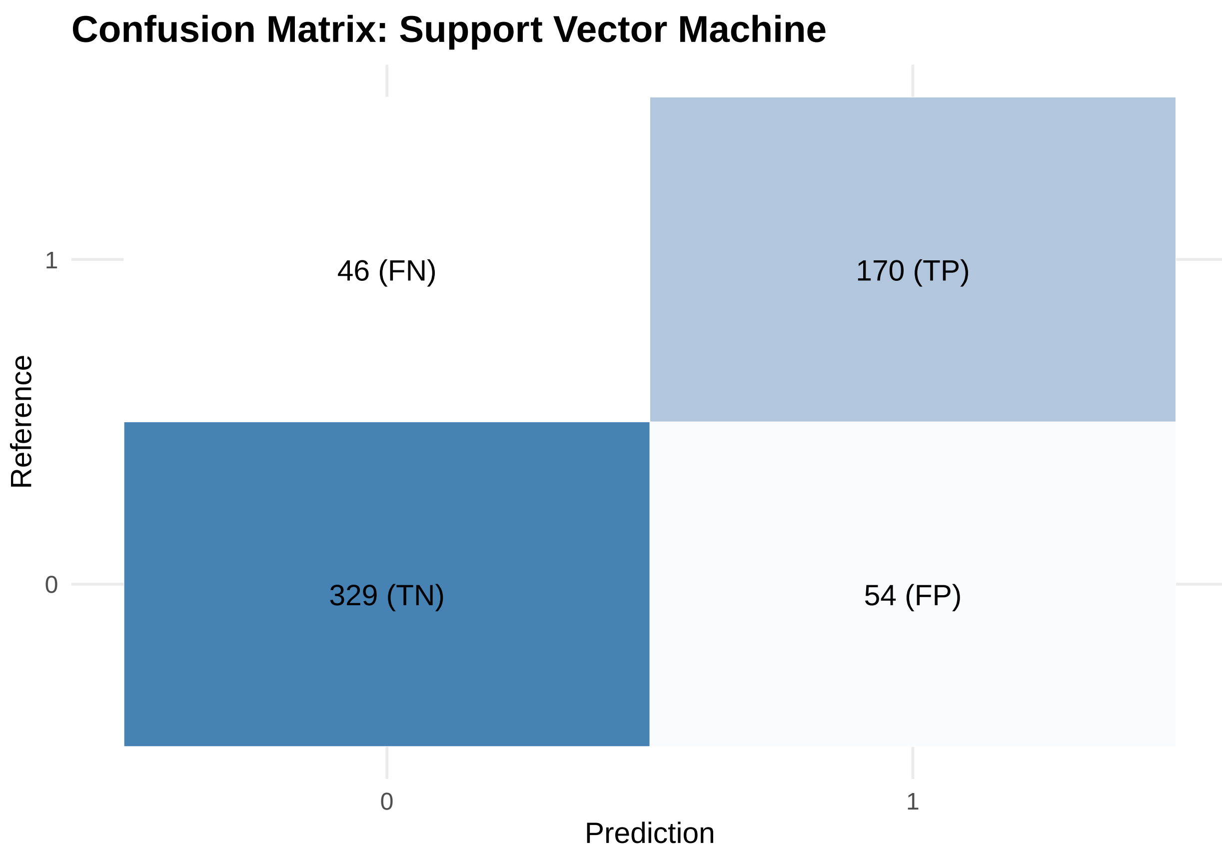
```r
library(RColorBrewer)

# Define a color palette
my_palette <- brewer.pal(3, "Reds")

# Plot
ggplot(df2, aes(x = Model, y = Accuracy)) +
  geom_bar(stat = "identity", fill = my_palette[1]) +
  geom_hline(yintercept = accuracy_majority, linetype = "dashed", color = "red") +
  geom_text(aes(label = round(Accuracy, 2)), vjust = -0.1) +
    labs(title = "Comparison of Model Accuracies with Baseline",
       subtitle = "Assessing Predictive Performance Across Classification Models",
       y = "Accuracy", x = "Model") +
  scale_y_continuous(breaks = seq(0, 1, by = 0.25)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(face = "bold", size = 14))
```



```r
svm_performance_two$table
```

```
##           Reference
## Prediction   0   1
##          0 329  54
##          1  46 170
```

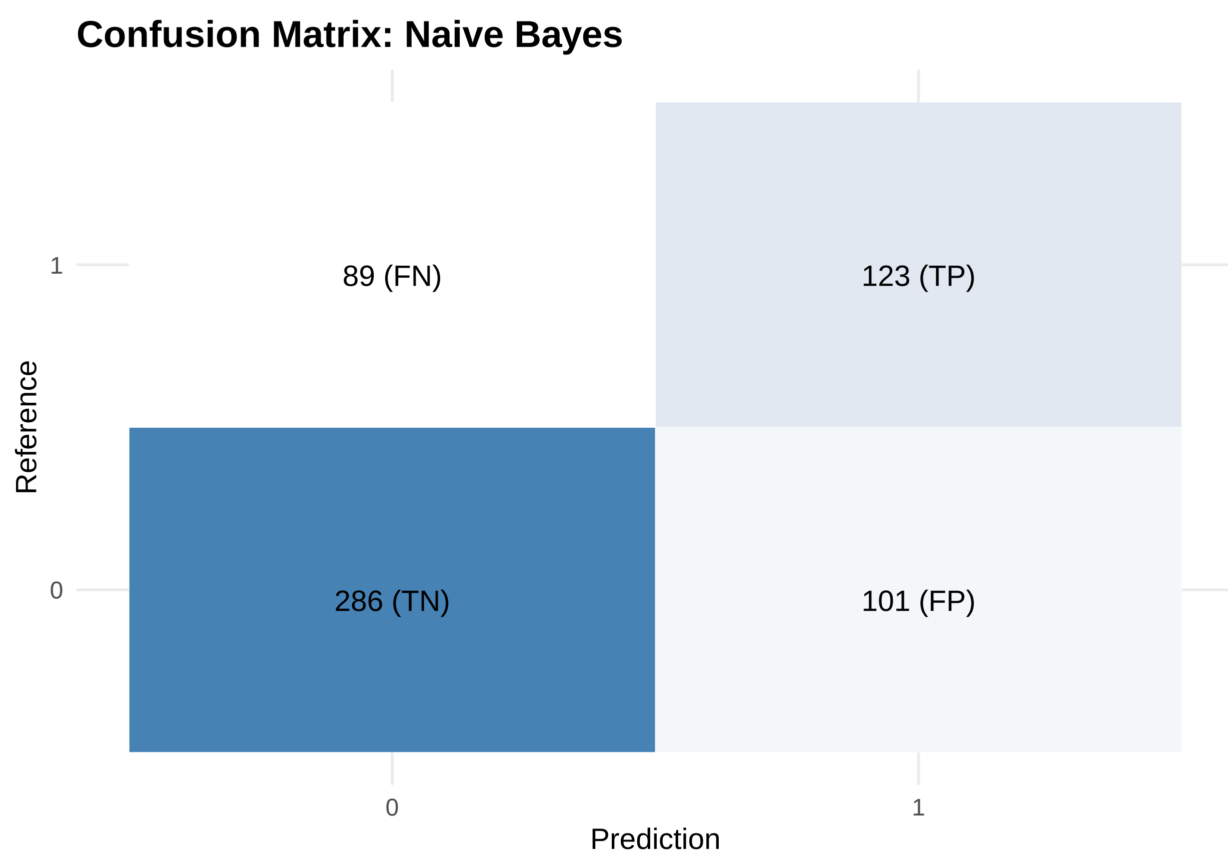
```r
# Create a dataframe from the confusion matrix table
conf_matrix_df_svm <- as.data.frame(svm_performance_two$table)


conf_matrix_df_svm$Label <- ifelse(conf_matrix_df_svm$Prediction == conf_matrix_df_svm$Reference,
                               ifelse(conf_matrix_df_svm$Prediction == "1", "TP", "TN"),
                               ifelse(conf_matrix_df_svm$Prediction == "1", "FN", "FP"))

# Plot heatmap
ggplot(conf_matrix_df_svm, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = paste(Freq, " (", Label, ")", sep = "")), vjust = 1) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix: Support Vector Machine",
       x = "Prediction",
       y = "Reference") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14)) +
  guides(fill = FALSE)
```




```
nb_performance_two$table
```

```
##           Reference
## Prediction   0   1
##          0 286 101
##          1  89 123
```

```
# Create a dataframe from the confusion matrix table
conf_matrix_df_nb <- as.data.frame(nb_performance_two$table)

conf_matrix_df_nb$Label <- ifelse(conf_matrix_df_nb$Prediction == conf_matrix_df_nb$Reference,
                               ifelse(conf_matrix_df_nb$Prediction == "1", "TP", "TN"),
                               ifelse(conf_matrix_df_nb$Prediction == "1", "FN", "FP"))

# Plot heatmap
ggplot(conf_matrix_df_nb, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
 geom_text(aes(label = paste(Freq, " (", Label, ")", sep = "")), vjust = 1) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix: Naive Bayes",
       x = "Prediction",
       y = "Reference") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14)) +
  guides(fill = FALSE)
```



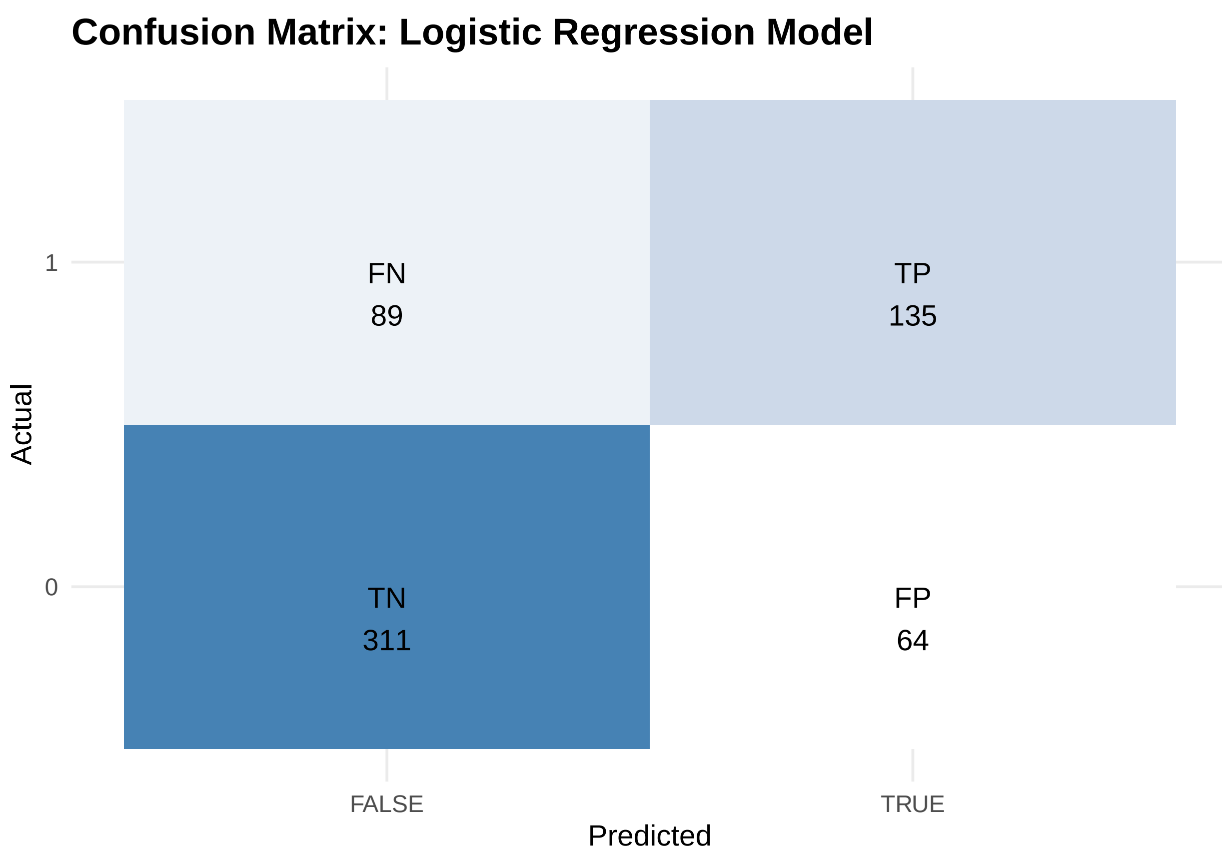

```r
#install.packages("data.table")
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.2

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
confusion_matrix_logit_two <- table(test_data_two$Response, logit_predictions_two > 0.5)

# Convert the table to a data frame for ggplot2
confusion_matrix_df <- as.data.frame(confusion_matrix_logit_two)

# Rename the columns for clarity
colnames(confusion_matrix_df) <- c("Actual", "Predicted", "Count")

# Convert the Actual and Predicted columns to factors if they are not already
confusion_matrix_df$Actual <- as.factor(confusion_matrix_df$Actual)
confusion_matrix_df$Predicted <- as.factor(confusion_matrix_df$Predicted)
```

```r
# Label the confusion matrix
confusion_matrix_df$Label <- ifelse(confusion_matrix_df$Actual == "1" & confusion_matrix_df$Predicted ==
                              ifelse(confusion_matrix_df$Actual == "0" & confusion_matrix_df$Predicted =
                              ifelse(confusion_matrix_df$Actual == "1" & confusion_matrix_df$Predicted =
```

```r
# Plot the confusion matrix
ggplot(data = confusion_matrix_df, aes(x = Predicted, y = Actual, fill = Count)) +
    geom_tile() +
    geom_text(aes(label = paste0(Label, "\n", Count)), vjust = 1) +
    scale_fill_gradient(low = "white", high = "steelblue") +
    labs(x = "Predicted", y = "Actual", fill = "Count") +
    ggtitle("Confusion Matrix: Logistic Regression Model") +
    theme_minimal() +
   theme(plot.title = element_text(face = "bold", size = 14)) +
    guides(fill = FALSE)
```




```r
(confusion_matrix_logit_two <- table(test_data_two$Response, logit_predictions_two > 0.5))
```

```
##
##     FALSE TRUE
##   0   311   64
##   1    89  135
```

```r
nb_performance_two$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value 
##            0.7626667            0.5491071            0.7390181 
##       Neg Pred Value            Precision               Recall 
##            0.5801887            0.7390181            0.7626667 
##                   F1           Prevalence       Detection Rate 
##            0.7506562            0.6260434            0.4774624 
## Detection Prevalence    Balanced Accuracy 
##            0.6460768            0.6558869
```

```r
svm_performance_two$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value 
##            0.8773333            0.7589286            0.8590078 
##       Neg Pred Value            Precision               Recall 
##            0.7870370            0.8590078            0.8773333 
##                   F1           Prevalence       Detection Rate 
##            0.8680739            0.6260434            0.5492487 
## Detection Prevalence    Balanced Accuracy 
##            0.6393990            0.8181310
```

```r
(precision_logit_two)
```

```
## [1] 0.678392
```

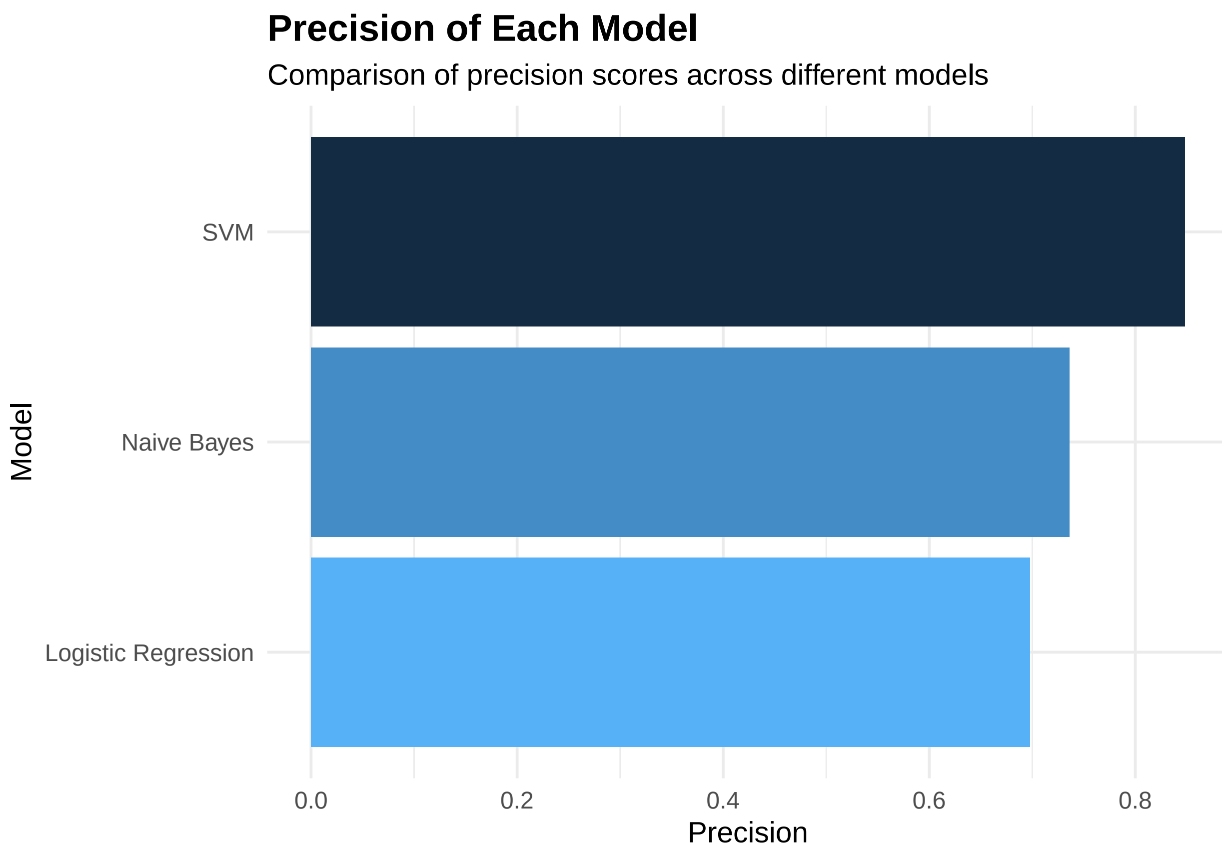
```r
precision_scores <- c(0.6980198, 0.8479381, 0.7361478)
models <- c("Logistic Regression", "SVM", "Naive Bayes")

# Create a data frame
precision_df <- data.frame(Model = models, Precision = precision_scores)
```

```r
(ordered_precision <- precision_df %>%
  group_by(Model)  %>%
  arrange(desc(Precision)))
```

```
## # A tibble: 3 x 2
## # Groups:   Model [3]
##   Model               Precision
##   <chr>                   <dbl>
## 1 SVM                     0.848
## 2 Naive Bayes             0.736
## 3 Logistic Regression     0.698
```

```r
ordered_precision %>%
  ggplot(aes(reorder(Model, Precision), Precision, fill = -Precision)) +
  geom_col() +
   labs(title = "Precision of Each Model",
       subtitle = "Comparison of precision scores across different models",
       x = "Model",
```

```
      y = "Precision") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14)) +
  coord_flip() +
  scale_colour_gradient2() +
  guides(fill="none")
```



Conclusion:

Maximizing precision ensures our model accurately identifies true positives (responders) while minimizing false positives (non-responders). This is crucial in marketing, where targeting the right customers impacts campaign success and cost-effectiveness.

In this project, I evaluated several models for classifying customer responses to marketing campaigns. The Support Vector Machine (SVM) model achieved the highest precision score of 0.85, correctly identifying likely responders 85% of the time.

This high precision makes the SVM model highly effective for this classification task. By using the SVM model, marketers can better target potential customers, leading to more successful and efficient campaigns.

Summary:

- Importance of Precision: High precision minimizes false positives, crucial for effective marketing.

- Model Performance: The SVM model achieved the highest precision score of 0.85 among the evaluated models.
- Implications: Implementing the SVM model can enhance the accuracy of marketing campaigns, improving their efficiency and effectiveness.