Digital Marketing Campaign Project

Group 1

2025-05-03

Contents

Contents

1 Project Objective

We will be analyzing real-world marketing data to uncover patterns, develop predictive models for customer conversion, and provide actionable insights to improve future marketing efforts.

- **Business goal:** Identify which customers are most likely to convert in our digital campaigns and how best to target them.
- **Key questions:**
  1. Which features (demographics, engagement metrics, campaign attributes) drive conversion?
  2. Which predictive model (logistic, KNN, RF) offers the best trade-off of accuracy, sensitivity, specificity, and AUC?
  3. How can we segment customers to tailor marketing strategies?

Report Format: Title Page, Project Objective, Exploratory Data Analysis, Predictive Modeling Development, Customer Conversion Analysis, Conclusion and Recommendations. Students are encouraged to use R Markdown to produce the final report. The format can be the same as your assignments.

2 Exploratory Data Analysis

2.1 Data Cleaning & Processing

```
library(googlesheets4)

# Google Sheets URL
sheet_url <- "https://docs.google.com/spreadsheets/d/1wnZjAaoGKUAUMKkVmuCXP1XPvkgIkwYbUEzcG9_Qebg/edit?

# Disable authentication if the sheet is public
gs4_deauth()

# Read data from Google Sheets
Marketing <- read_sheet(sheet_url)

## v Reading from "digital_marketing_campaign_dataset".

## v Range 'digital_marketing_campaign_dataset'.

# View first few rows of the dataset
head(Marketing)

## # A tibble: 6 x 18
##   CustomerID   Age Gender Income CampaignChannel CampaignType AdSpend
##        <dbl> <dbl> <chr>   <dbl> <chr>           <chr>          <dbl>
## 1       8000    56 Female 136912 Social Media    Awareness      6498.
## 2       8001    69 Male    41760 Email           Retention      3899.
## 3       8002    46 Female  88456 PPC             Awareness      1546.
## 4       8003    32 Female  44085 PPC             Conversion      540.
## 5       8004    60 Female  83964 PPC             Conversion     1678.
```

```
## 6       8005    25 Female  42925 Social Media    Awareness      9579.
## # i 11 more variables: ClickThroughRate <dbl>, ConversionRate <dbl>,
## #   WebsiteVisits <dbl>, PagesPerVisit <dbl>, TimeOnSite <dbl>,
## #   SocialShares <dbl>, EmailOpens <dbl>, EmailClicks <dbl>,
## #   PreviousPurchases <dbl>, LoyaltyPoints <dbl>, Conversion <dbl>
```

```
# Get summary statistics for all numerical variables
summary(Marketing)
```

```
##    CustomerID          Age           Gender              Income
##  Min.   : 8000   Min.   :18.00   Length:8000        Min.   : 20014
##  1st Qu.:10000   1st Qu.:31.00   Class :character   1st Qu.: 51745
##  Median :12000   Median :43.00   Mode  :character   Median : 84927
##  Mean   :12000   Mean   :43.63                      Mean   : 84664
##  3rd Qu.:13999   3rd Qu.:56.00                      3rd Qu.:116816
##  Max.   :15999   Max.   :69.00                      Max.   :149986
##  CampaignChannel    CampaignType          AdSpend       ClickThroughRate
##  Length:8000        Length:8000        Min.   : 100.1   Min.   :0.01000
##  Class :character   Class :character   1st Qu.:2523.2   1st Qu.:0.08263
##  Mode  :character   Mode  :character   Median :5013.4   Median :0.15451
##                                        Mean   :5000.9   Mean   :0.15483
##                                        3rd Qu.:7408.0   3rd Qu.:0.22821
##                                        Max.   :9997.9   Max.   :0.29997
##  ConversionRate    WebsiteVisits   PagesPerVisit     TimeOnSite
##  Min.   :0.01002   Min.   : 0.00   Min.   :1.000   Min.   : 0.5017
##  1st Qu.:0.05641   1st Qu.:13.00   1st Qu.:3.302   1st Qu.: 4.0683
##  Median :0.10405   Median :25.00   Median :5.534   Median : 7.6830
##  Mean   :0.10439   Mean   :24.75   Mean   :5.549   Mean   : 7.7277
##  3rd Qu.:0.15208   3rd Qu.:37.00   3rd Qu.:7.836   3rd Qu.:11.4815
##  Max.   :0.19999   Max.   :49.00   Max.   :9.999   Max.   :14.9953
##   SocialShares    EmailOpens      EmailClicks    PreviousPurchases
##  Min.   : 0.0   Min.   : 0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:25.0   1st Qu.: 5.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :50.0   Median : 9.000   Median :4.000   Median :4.000
##  Mean   :49.8   Mean   : 9.477   Mean   :4.467   Mean   :4.486
##  3rd Qu.:75.0   3rd Qu.:14.000   3rd Qu.:7.000   3rd Qu.:7.000
##  Max.   :99.0   Max.   :19.000   Max.   :9.000   Max.   :9.000
##  LoyaltyPoints    Conversion
##  Min.   :   0   Min.   :0.0000
##  1st Qu.:1255   1st Qu.:1.0000
##  Median :2497   Median :1.0000
##  Mean   :2490   Mean   :0.8765
##  3rd Qu.:3702   3rd Qu.:1.0000
##  Max.   :4999   Max.   :1.0000
```

```
# Display structure of the dataset
str(Marketing)
```

```
## tibble [8,000 x 18] (S3: tbl_df/tbl/data.frame)
##  $ CustomerID       : num [1:8000] 8000 8001 8002 8003 8004 ...
##  $ Age              : num [1:8000] 56 69 46 32 60 25 38 56 36 40 ...
##  $ Gender           : chr [1:8000] "Female" "Male" "Female" "Female" ...
##  $ Income           : num [1:8000] 136912 41760 88456 44085 83964 ...
```

```
##  $ CampaignChannel  : chr [1:8000] "Social Media" "Email" "PPC" "PPC" ...
##  $ CampaignType     : chr [1:8000] "Awareness" "Retention" "Awareness" "Conversion" ...
##  $ AdSpend          : num [1:8000] 6498 3899 1546 540 1678 ...
##  $ ClickThroughRate : num [1:8000] 0.0439 0.1557 0.2775 0.1376 0.2529 ...
##  $ ConversionRate   : num [1:8000] 0.088 0.1827 0.0764 0.088 0.1099 ...
##  $ WebsiteVisits    : num [1:8000] 0 42 2 47 0 6 42 48 13 22 ...
##  $ PagesPerVisit    : num [1:8000] 2.4 2.92 8.22 4.54 2.05 ...
##  $ TimeOnSite       : num [1:8000] 7.4 5.35 13.79 14.69 13.99 ...
##  $ SocialShares     : num [1:8000] 19 5 0 89 6 95 54 96 73 14 ...
##  $ EmailOpens       : num [1:8000] 6 2 11 2 6 5 14 9 4 8 ...
##  $ EmailClicks      : num [1:8000] 9 7 2 2 6 8 3 3 8 4 ...
##  $ PreviousPurchases: num [1:8000] 4 2 8 0 8 0 6 0 5 8 ...
##  $ LoyaltyPoints    : num [1:8000] 688 3459 2337 2463 4345 ...
##  $ Conversion       : num [1:8000] 1 1 1 1 1 1 1 1 1 1 ...
```

2.1.0.1 Data Type Conversion

```
Marketing <- Marketing %>%
  mutate(
    CustomerID = as.character(CustomerID),  # Convert ID to chr
    Age = as.integer(Age),                  # Convert Age to integer
    Income = as.integer(Income),            # Convert Income to integer
    AdSpend = as.numeric(AdSpend),          # Ensure AdSpend remains numeric
    WebsiteVisits = as.integer(WebsiteVisits),  # Convert to integer
    SocialShares = as.integer(SocialShares),    # Convert to integer
    EmailOpens = as.integer(EmailOpens),
    EmailClicks = as.integer(EmailClicks),
    PreviousPurchases = as.integer(PreviousPurchases),
    LoyaltyPoints = as.integer(LoyaltyPoints),
    Conversion = as.factor(Conversion),  # Convert Conversion to factor (0/1)
    Gender = as.factor(Gender),          # Convert Gender to factor
    CampaignChannel = as.factor(CampaignChannel),  # Convert CampaignChannel to factor
    CampaignType = as.factor(CampaignType)         # Convert CampaignType to factor
  )

# Verify the Changess
str(Marketing)  # Check updated data types
```

```
## tibble [8,000 x 18] (S3: tbl_df/tbl/data.frame)
##  $ CustomerID       : chr [1:8000] "8000" "8001" "8002" "8003" ...
##  $ Age              : int [1:8000] 56 69 46 32 60 25 38 56 36 40 ...
##  $ Gender           : Factor w/ 2 levels "Female","Male": 1 2 1 1 1 1 1 1 1 2 ...
##  $ Income           : int [1:8000] 136912 41760 88456 44085 83964 42925 25615 57083 140788 130764 ..
##  $ CampaignChannel  : Factor w/ 5 levels "Email","PPC",..: 5 1 2 2 2 5 3 5 1 5 ...
##  $ CampaignType     : Factor w/ 4 levels "Awareness","Consideration",..: 1 4 1 3 3 1 1 3 4 1 ...
##  $ AdSpend          : num [1:8000] 6498 3899 1546 540 1678 ...
##  $ ClickThroughRate : num [1:8000] 0.0439 0.1557 0.2775 0.1376 0.2529 ...
##  $ ConversionRate   : num [1:8000] 0.088 0.1827 0.0764 0.088 0.1099 ...
##  $ WebsiteVisits    : int [1:8000] 0 42 2 47 0 6 42 48 13 22 ...
##  $ PagesPerVisit    : num [1:8000] 2.4 2.92 8.22 4.54 2.05 ...
##  $ TimeOnSite       : num [1:8000] 7.4 5.35 13.79 14.69 13.99 ...
##  $ SocialShares     : int [1:8000] 19 5 0 89 6 95 54 96 73 14 ...
##  $ EmailOpens       : int [1:8000] 6 2 11 2 6 5 14 9 4 8 ...
```

```
##  $ EmailClicks      : int [1:8000] 9 7 2 2 6 8 3 3 8 4 ...
##  $ PreviousPurchases: int [1:8000] 4 2 8 0 8 0 6 0 5 8 ...
##  $ LoyaltyPoints    : int [1:8000] 688 3459 2337 2463 4345 3316 930 2983 460 3789 ...
##  $ Conversion       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
head(Marketing)
```

```
## # A tibble: 6 x 18
##   CustomerID   Age Gender Income CampaignChannel CampaignType AdSpend
##   <chr>      <int> <fct>   <int> <fct>           <fct>          <dbl>
## 1 8000          56 Female 136912 Social Media    Awareness      6498.
## 2 8001          69 Male    41760 Email           Retention      3899.
## 3 8002          46 Female  88456 PPC             Awareness      1546.
## 4 8003          32 Female  44085 PPC             Conversion      540.
## 5 8004          60 Female  83964 PPC             Conversion     1678.
## 6 8005          25 Female  42925 Social Media    Awareness      9579.
## # i 11 more variables: ClickThroughRate <dbl>, ConversionRate <dbl>,
## #   WebsiteVisits <int>, PagesPerVisit <dbl>, TimeOnSite <dbl>,
## #   SocialShares <int>, EmailOpens <int>, EmailClicks <int>,
## #   PreviousPurchases <int>, LoyaltyPoints <int>, Conversion <fct>
```

```
#Check missing value
colSums(is.na(Marketing))
```

```
##        CustomerID               Age            Gender            Income
##                 0                 0                 0                 0
##   CampaignChannel      CampaignType           AdSpend  ClickThroughRate
##                 0                 0                 0                 0
##    ConversionRate     WebsiteVisits     PagesPerVisit        TimeOnSite
##                 0                 0                 0                 0
##      SocialShares        EmailOpens       EmailClicks PreviousPurchases
##                 0                 0                 0                 0
##     LoyaltyPoints        Conversion
##                 0                 0
```

2.2 Univariate Analysis

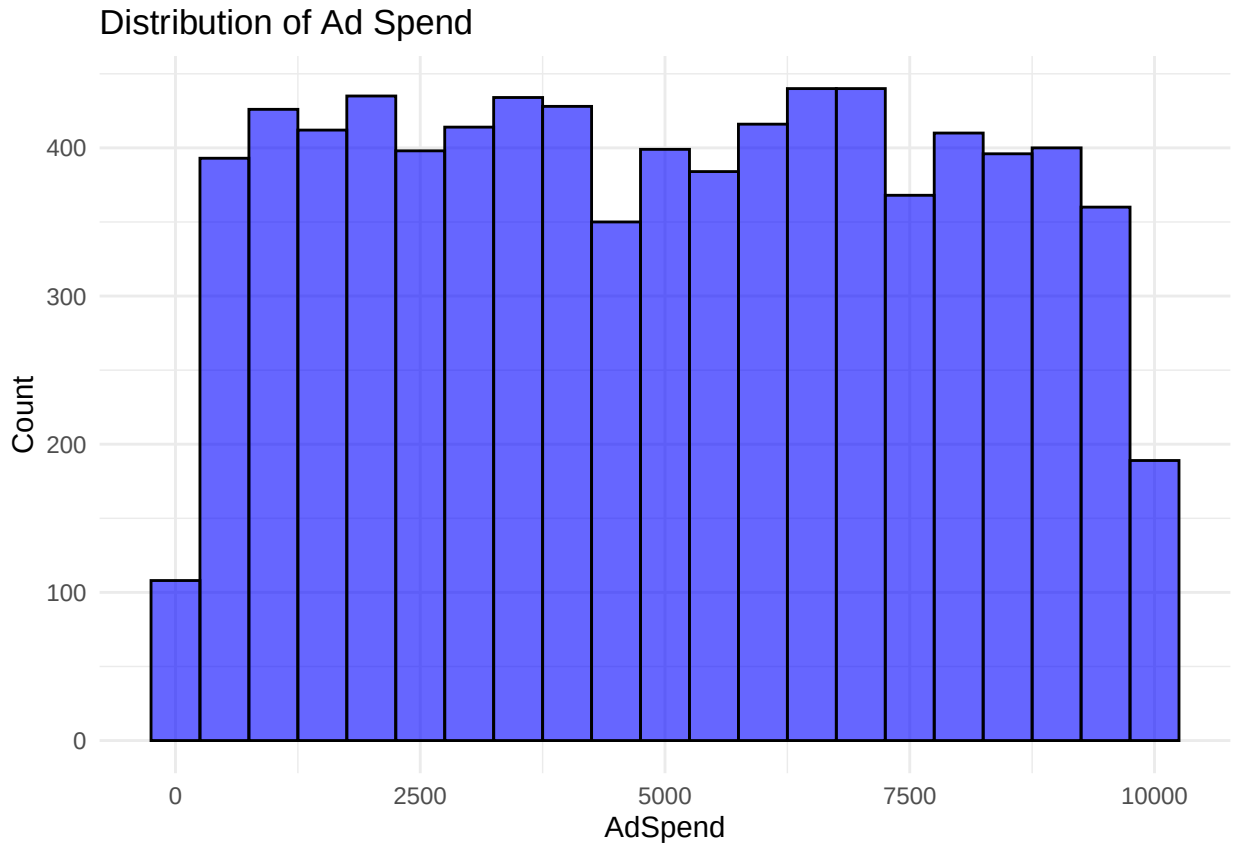
```
# Function to create histograms
plot_histogram <- function(data, column, title, binwidth = NULL) {
  ggplot(data, aes_string(x = column)) +
    geom_histogram(binwidth = binwidth, fill = "blue", alpha = 0.6, color = "black") +
    theme_minimal() +
    labs(title = title, x = column, y = "Count")
}
```

2.2.0.1 Ad Spend Distribution

```
plot_histogram(Marketing, "AdSpend", "Distribution of Ad Spend", binwidth = 500)
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
```

```
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

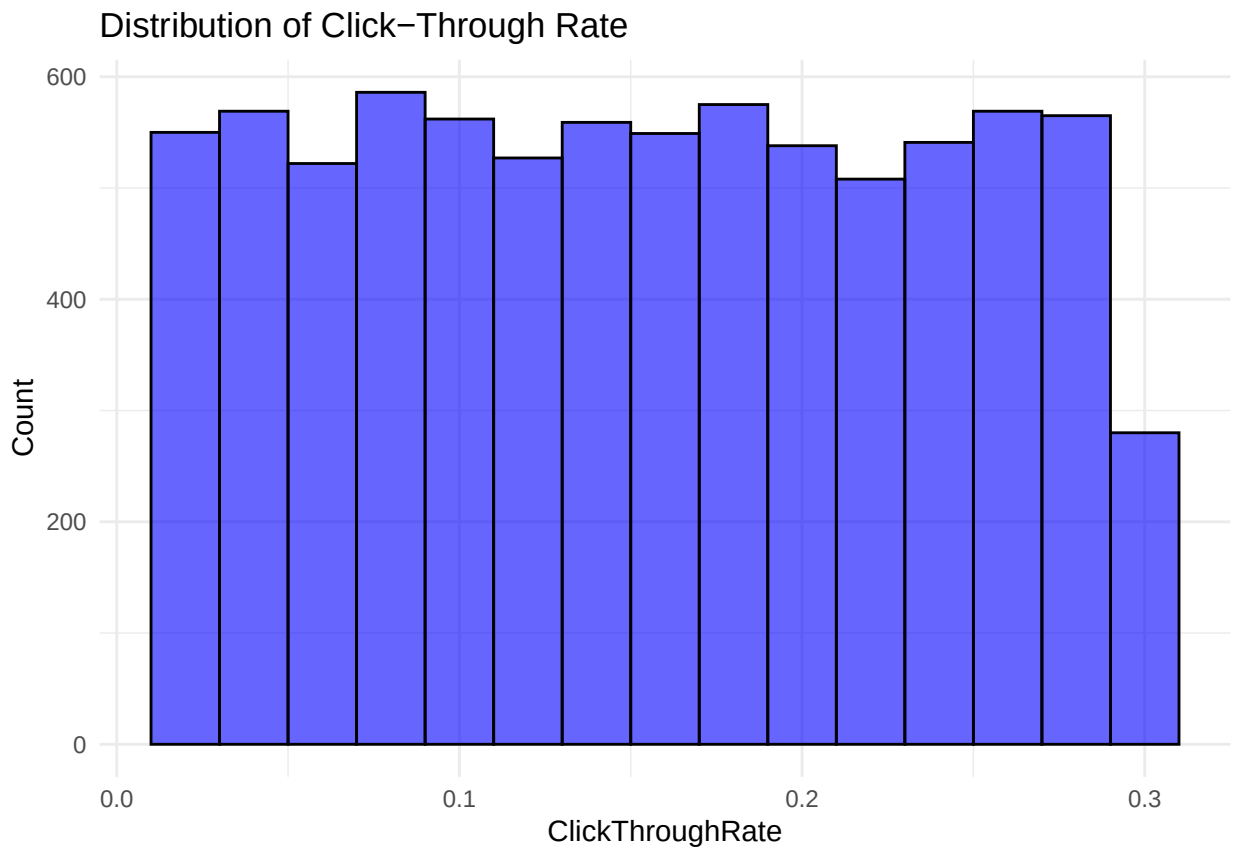


Observations

- Range: Ad Spend spans from near 0 up to about 10,000, suggesting a wide variety of spending levels.
- Relatively Even Spread: The distribution appears fairly uniform, with no pronounced peak or strong skew.
- Slight Dip at Lower Values: There's a small dip around the lower range (0–1,000), indicating fewer instances of minimal spend.
- Potentially Steady Investment: The bulk of the data hovers between 2,000 and 8,000, implying most campaigns allocate a moderate to high spend.

2.2.0.2 CTR Distribution

```
plot_histogram(Marketing, "ClickThroughRate", "Distribution of Click-Through Rate", binwidth = 0.02)
```

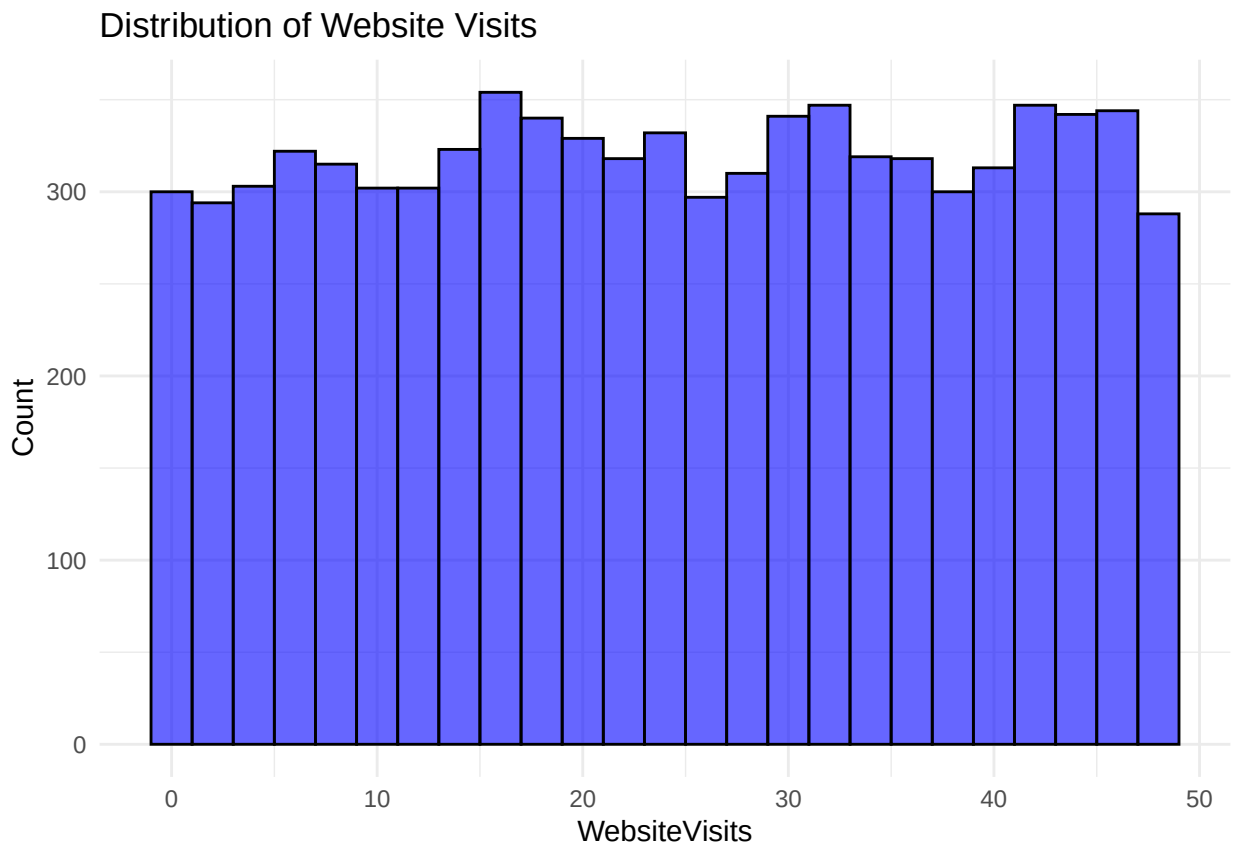


Observations

- Range: The Click-Through Rate (CTR) spans from near 0 up to about 0.30.
- Fairly Even Spread: There is no sharp peak, suggesting that CTR values are distributed relatively uniformly within this range.
- Mild Concentration Around 0.10–0.15: Slightly more data points appear around the mid-range, but overall variation is not extreme.
- No Extreme Outliers: The histogram ends around 0.30, indicating that CTR does not exceed 30%.

2.2.0.3 Website Vistis Distribution

```
plot_histogram(Marketing, "WebsiteVisits", "Distribution of Website Visits", binwidth = 2)
```

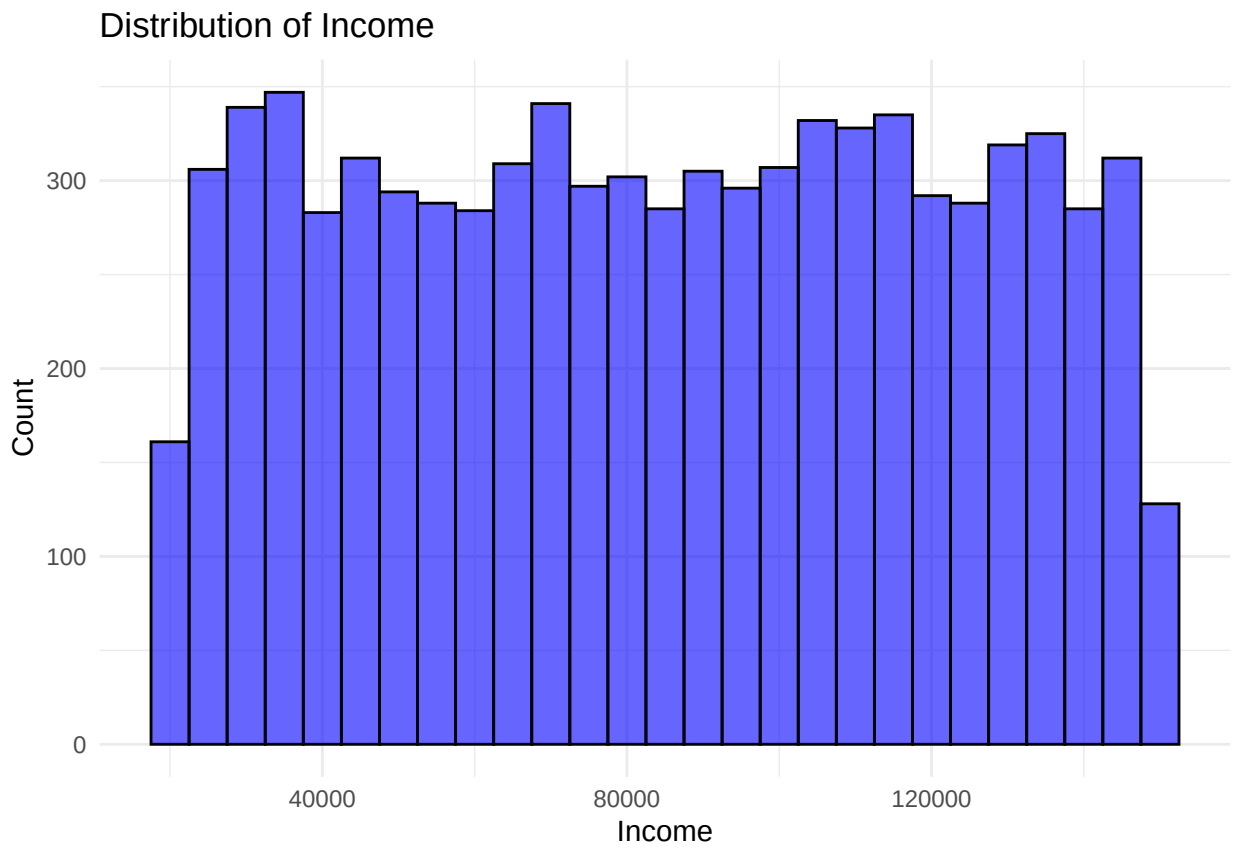


Observations

Range: Website Visits span from 0 to about 50. Fairly Even Spread: The histogram shows a relatively uniform distribution, with no single dominant peak. Mild Concentration Around Mid-Range: Slightly more visits occur between about 15 and 35, though the overall variation is not extreme. No Extreme Outliers: The data caps at 49 visits, indicating most visitors fall within a moderate range of site engagement.

2.2.0.4 Income Distribution

```
# Income Distribution
plot_histogram(Marketing, "Income", "Distribution of Income", binwidth = 5000)
```

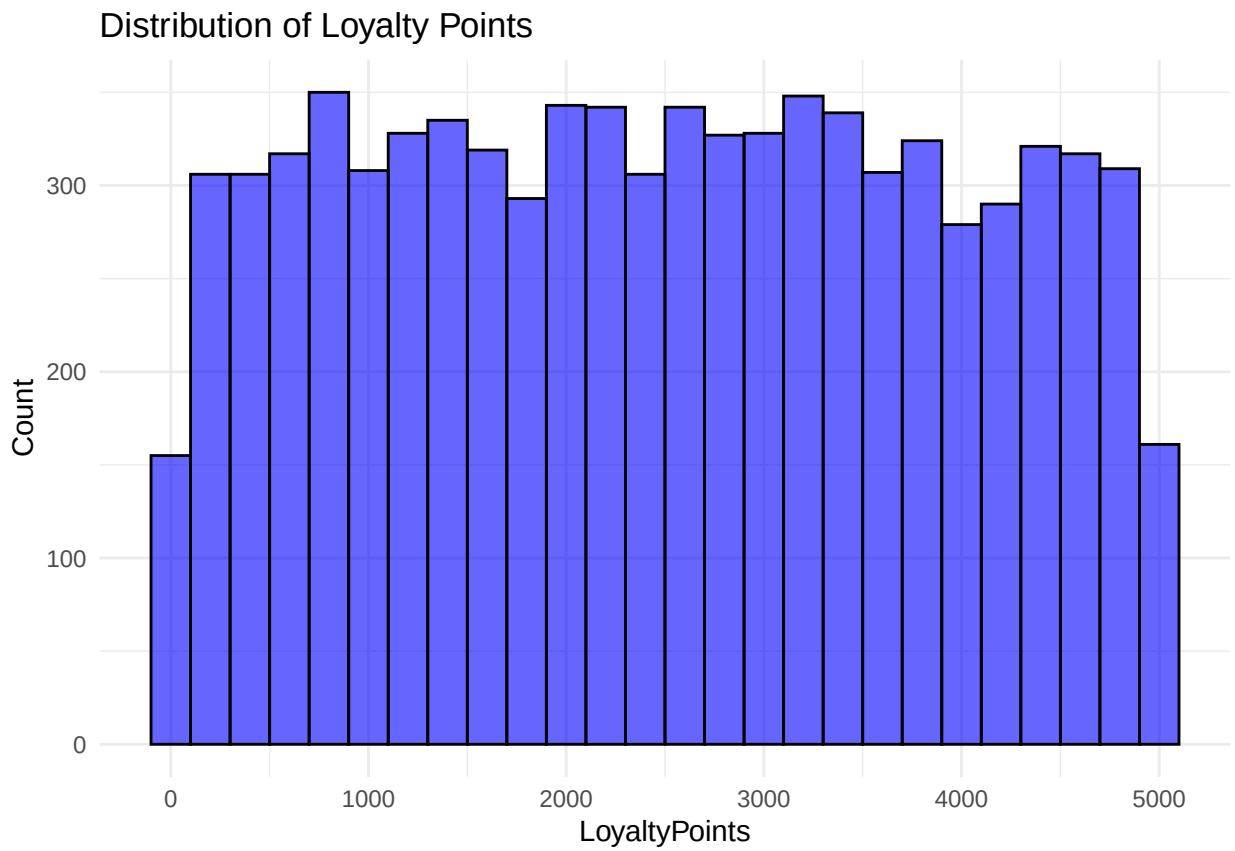


Observations

Range: Income varies roughly from 20000to 150000, indicating a broad spectrum of financial backgrounds. Slight Concentration Around Mid-Range: A large portion of individuals fall between 40,000and 80,000. Dip Toward Upper Income Levels: Fewer individuals earn beyond $120,000. No Sharp Peak: The distribution is relatively spread out, with mild fluctuations across the mid-range.

2.2.0.5 Loyalty Points Distr

```
# Loyalty Points Distribution
plot_histogram(Marketing, "LoyaltyPoints", "Distribution of Loyalty Points", binwidth = 200)
```

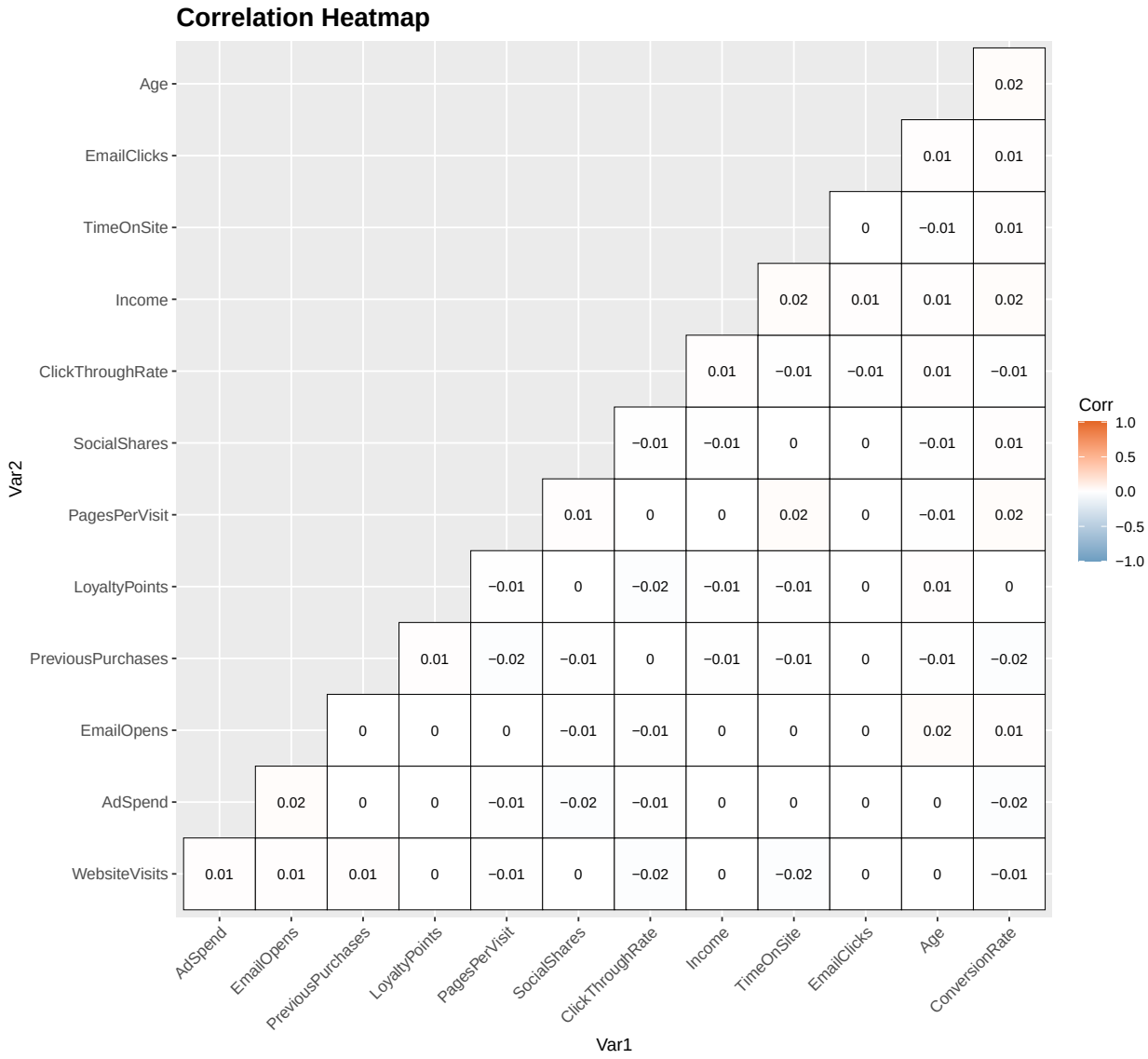


Observations

Range: Loyalty Points span from 0 up to about 5,000. Fairly Uniform Distribution: No single peak dominates, though there's a slight dip at the lower end (0–500). Moderate Variations Across the Range: Mild fluctuations occur, but overall the values are spread evenly between 500 and 4,500. No Extreme Outliers: The data caps at 5,000, indicating a clear upper limit for points earned.

2.3 Bivariate Visualizations

2.3.1 Variables Correlations



The correlation heatmap shows no high-corrlated variables which is good for modeling in the future. We will also using Stepwise to eliminate unsignificant variables later.

2.3.1.1 Gender Count

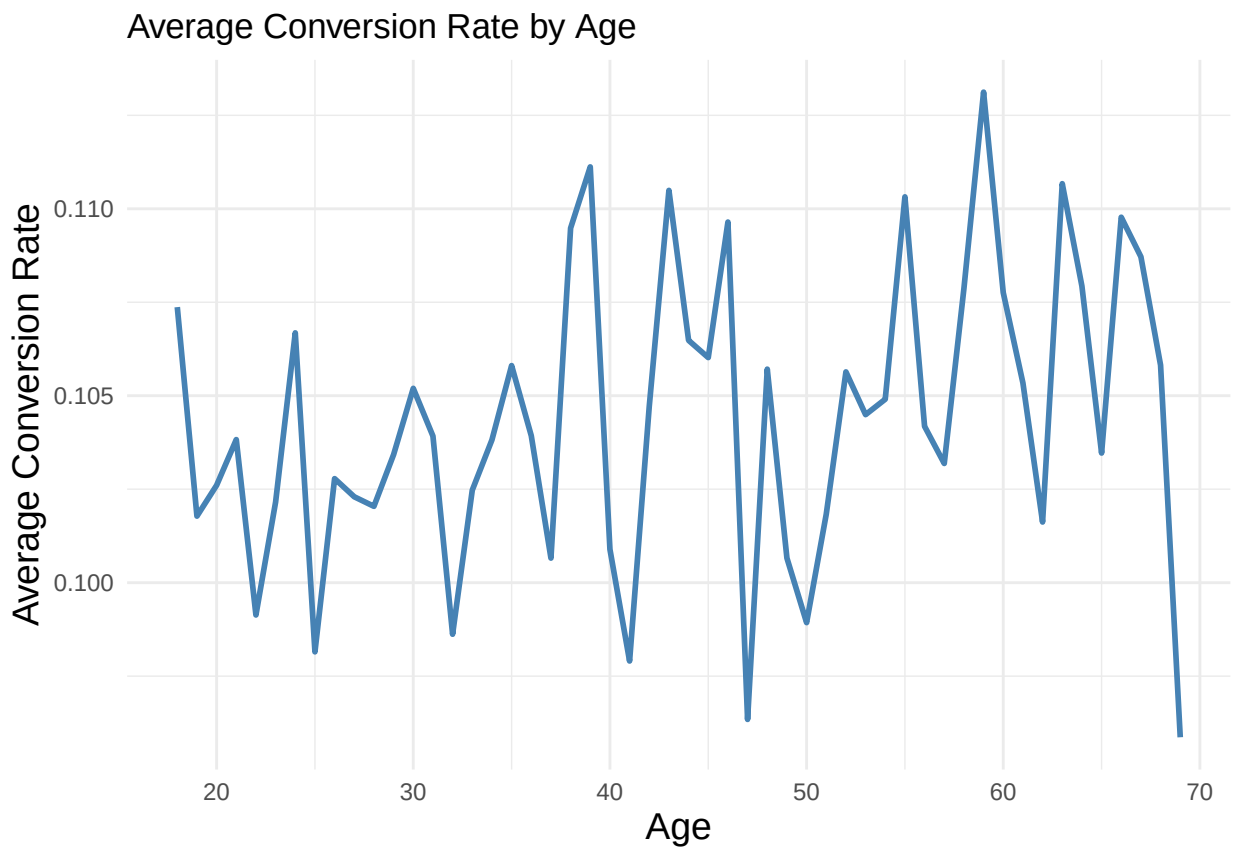
```
gender_count <- Marketing %>%
  group_by(Gender) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  mutate(Percentage = (Count / sum(Count)) * 100)
```

```
gender_count
```

```
## # A tibble: 2 x 3
##   Gender Count Percentage
##   <fct>  <int>      <dbl>
## 1 Female  4839       60.5
## 2 Male    3161       39.5
```

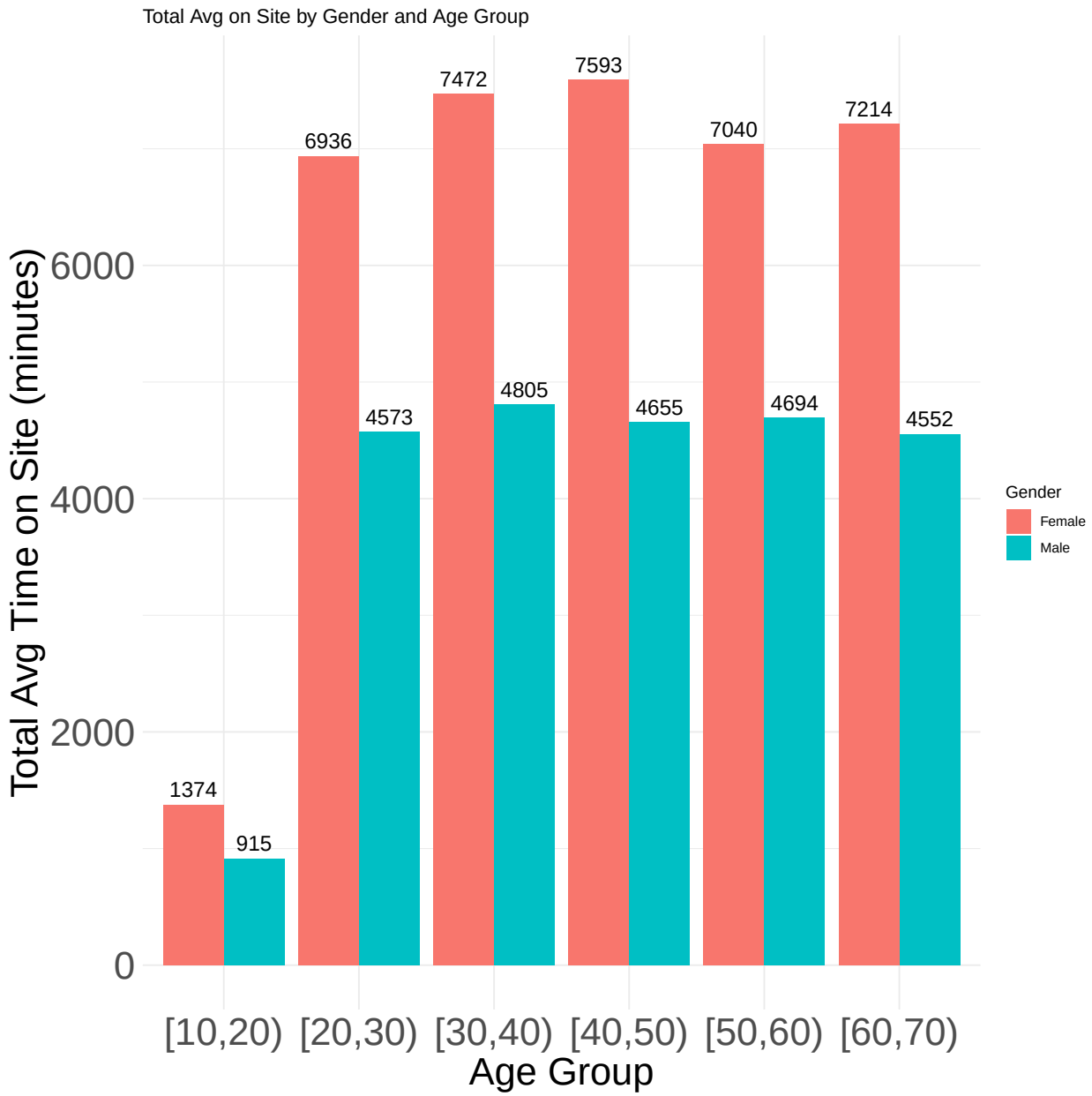
```
# Line Plot for Average Conversion Rate by Age
ggplot(data = Marketing, aes(x = Age, y = ConversionRate)) +
  geom_line(stat = "summary", fun = "mean", color = "steelblue", size = 1) +
  labs(title = "Average Conversion Rate by Age", x = "Age", y = "Average Conversion Rate") +
  theme_minimal() +
  theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



2.3.1.2 TimeOnSite vs. Gender&Age

```r
# Create age groups for better visualization
Marketing$AgeGroup <- cut(Marketing$Age, breaks = seq(0, 100, by = 10), right = FALSE)

# Calculate total average TimeOnSite by Gender and Age Group
time_on_site_gender_age <- Marketing %>%
  group_by(AgeGroup, Gender) %>%
  summarise(TotalAverageTimeOnSite = sum(TimeOnSite, na.rm = TRUE), .groups = 'drop')

# Create the bar chart
ggplot(data = time_on_site_gender_age, aes(x = AgeGroup, y = TotalAverageTimeOnSite, fill = Gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = round(TotalAverageTimeOnSite, 0)),  # Round to 0 decimal points
            position = position_dodge(width = 0.9),
            vjust = -0.5,
            size = 5) +  # Adjust size of the text
  labs(title = "Total Avg on Site by Gender and Age Group",
       x = "Age Group",
       y = "Total Avg Time on Site (minutes)") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 25),  # Increase x-axis label size
    axis.title.y = element_text(size = 25),  # Increase y-axis label size
    axis.text.x = element_text(size = 25),   # Increase x-tick label size
    axis.text.y = element_text(size = 25)    # Increase y-tick label size
  )
```

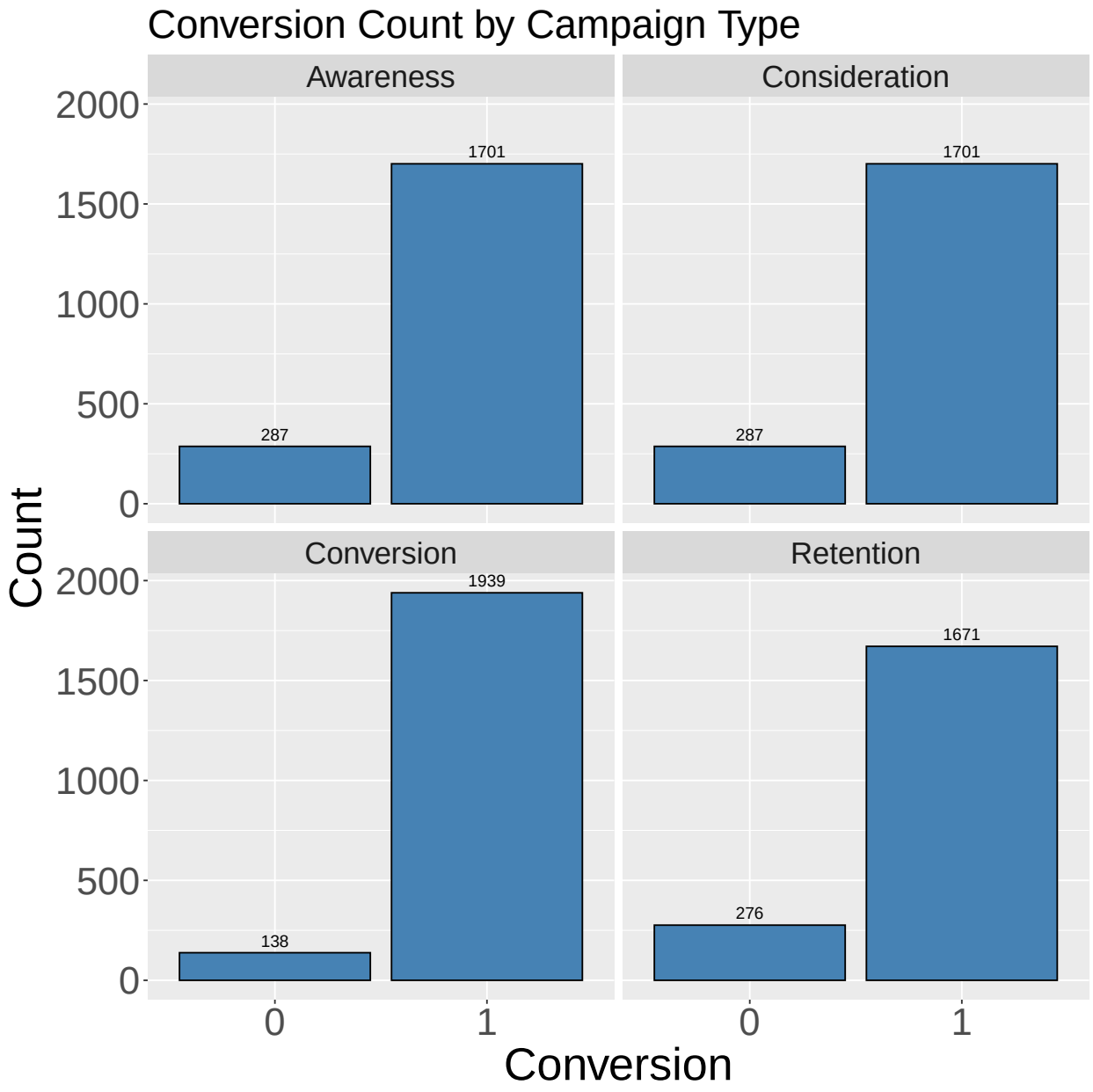


2.3.1.3 Conversion vs. CampaignType

```
# Create a bar plot for Conversion with facets by CampaignType
ggplot(data = Marketing, aes(x = Conversion)) +
  geom_bar(aes(y = ..count..), fill = "steelblue", color = "black") +  # Adjust colors as needed
  facet_wrap(~CampaignType) +  # Create facets for each CampaignType
  labs(title = "Conversion Count by Campaign Type", x = "Conversion", y = "Count") +  # Add labels
  theme(
    axis.title.x = element_text(size = 30),  # Increase x-axis label size
    axis.title.y = element_text(size = 30),  # Increase y-axis label size
    axis.text.x = element_text(size = 25),   # Increase x-tick label size
    axis.text.y = element_text(size = 25),   # Increase y-tick label size
    strip.text = element_text(size = 20),     # Increase facet label size
```

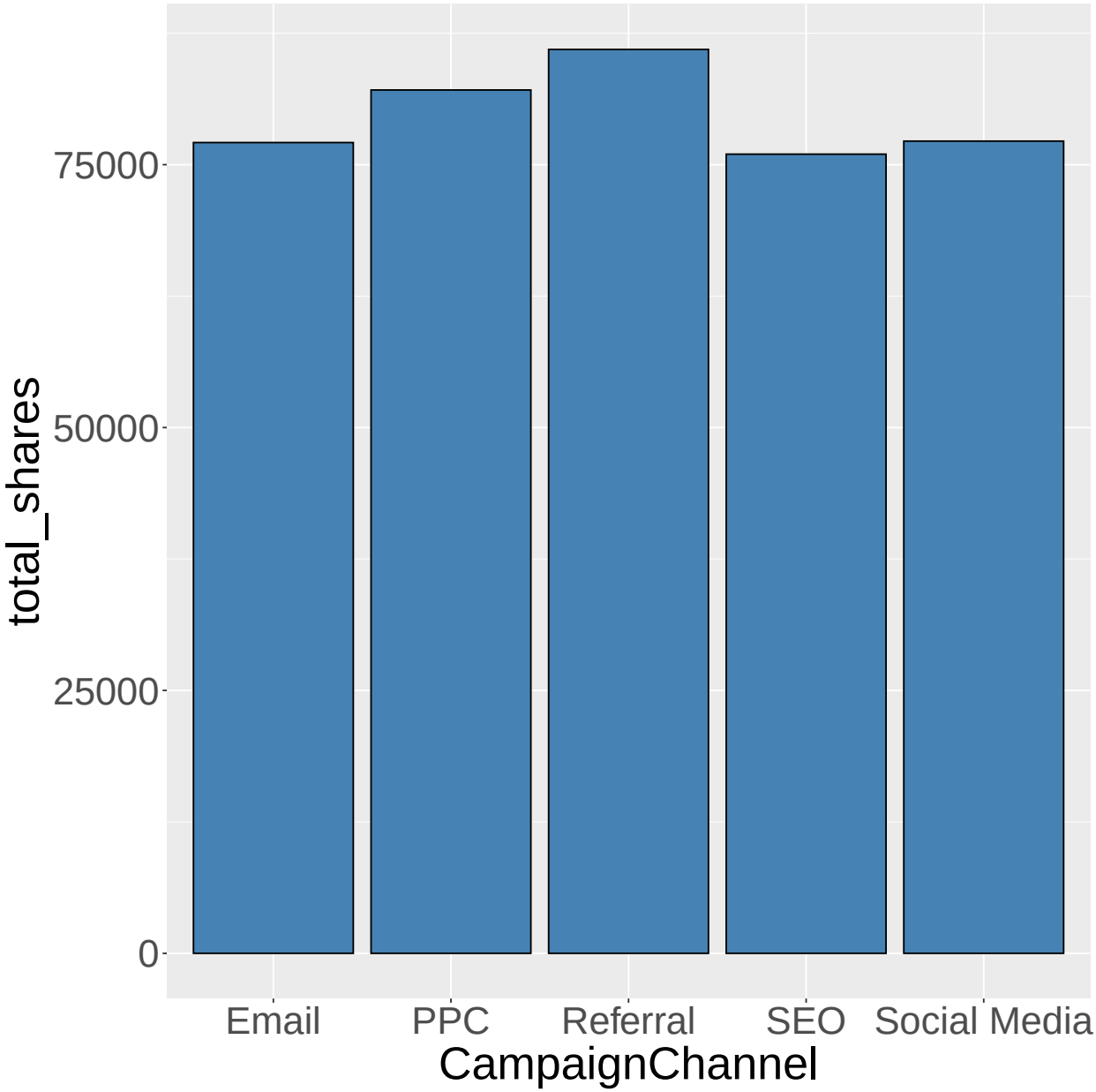
```
    plot.title = element_text(size = 26)      # Increase plot title size
  ) +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5, size = 4) +  # Add data labels above
  coord_cartesian(clip = 'off')  # Ensure labels are not clipped
```

```
## Warning: The dot-dot notation (‘..count..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(count)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



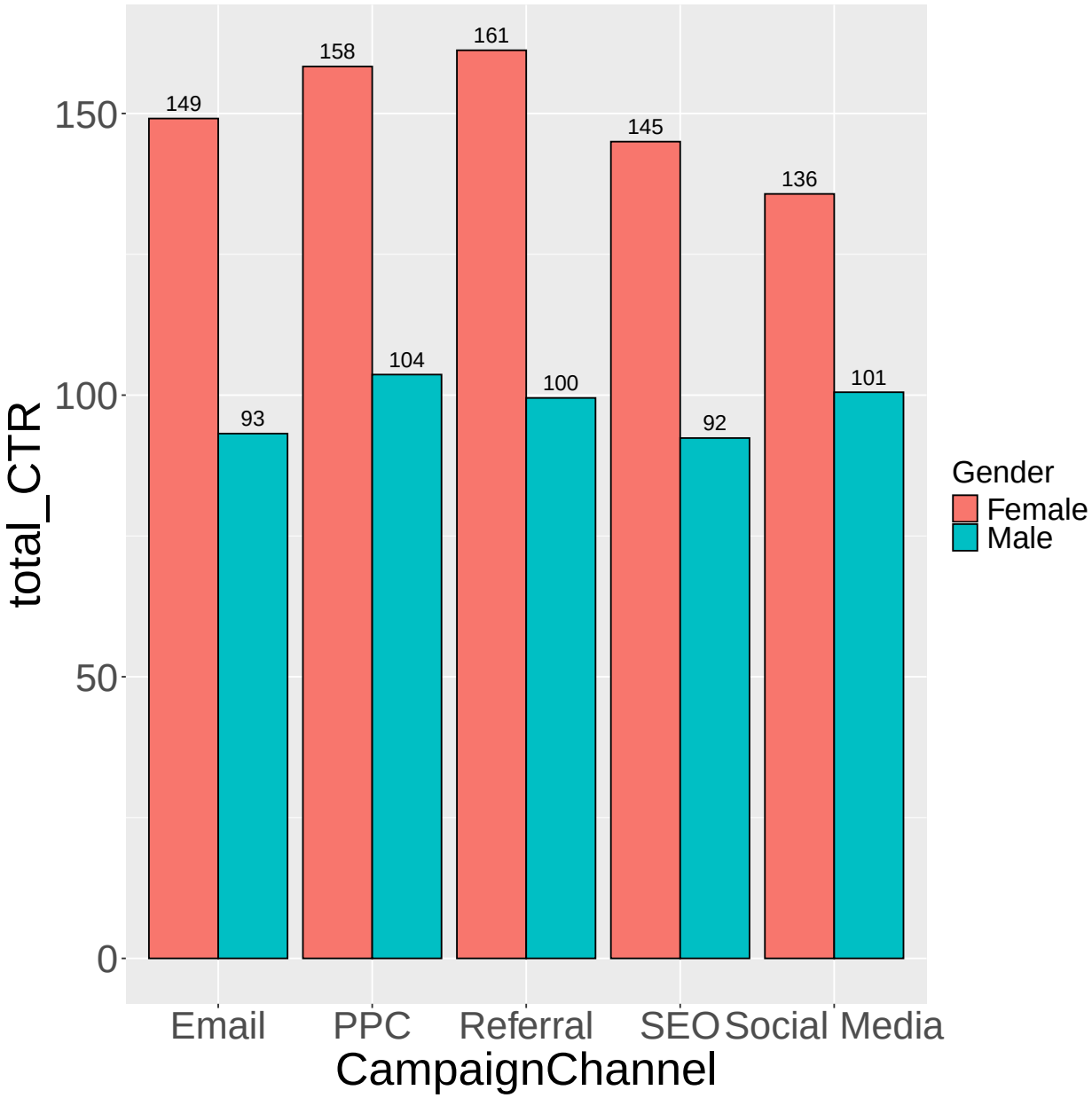
2.3.1.4 Total Shares vs. Campaign Channel

```r
mkt.sum <- summarise(group_by(Marketing, CampaignChannel), total_shares = sum(SocialShares))

ggplot(data = mkt.sum, aes(x=CampaignChannel, y = total_shares)) +
  geom_bar(stat = "identity",fill = "steelblue", color = "black")+
  theme(
    axis.title.x = element_text(size = 30),  # Increase x-axis label size
    axis.title.y = element_text(size = 30),  # Increase y-axis label size
    axis.text.x = element_text(size = 25),   # Increase x-tick label size
    axis.text.y = element_text(size = 25),   # Increase y-tick label size
    strip.text = element_text(size = 20),     # Increase facet label size
    plot.title = element_text(size = 26)      # Increase plot title size
  )
```



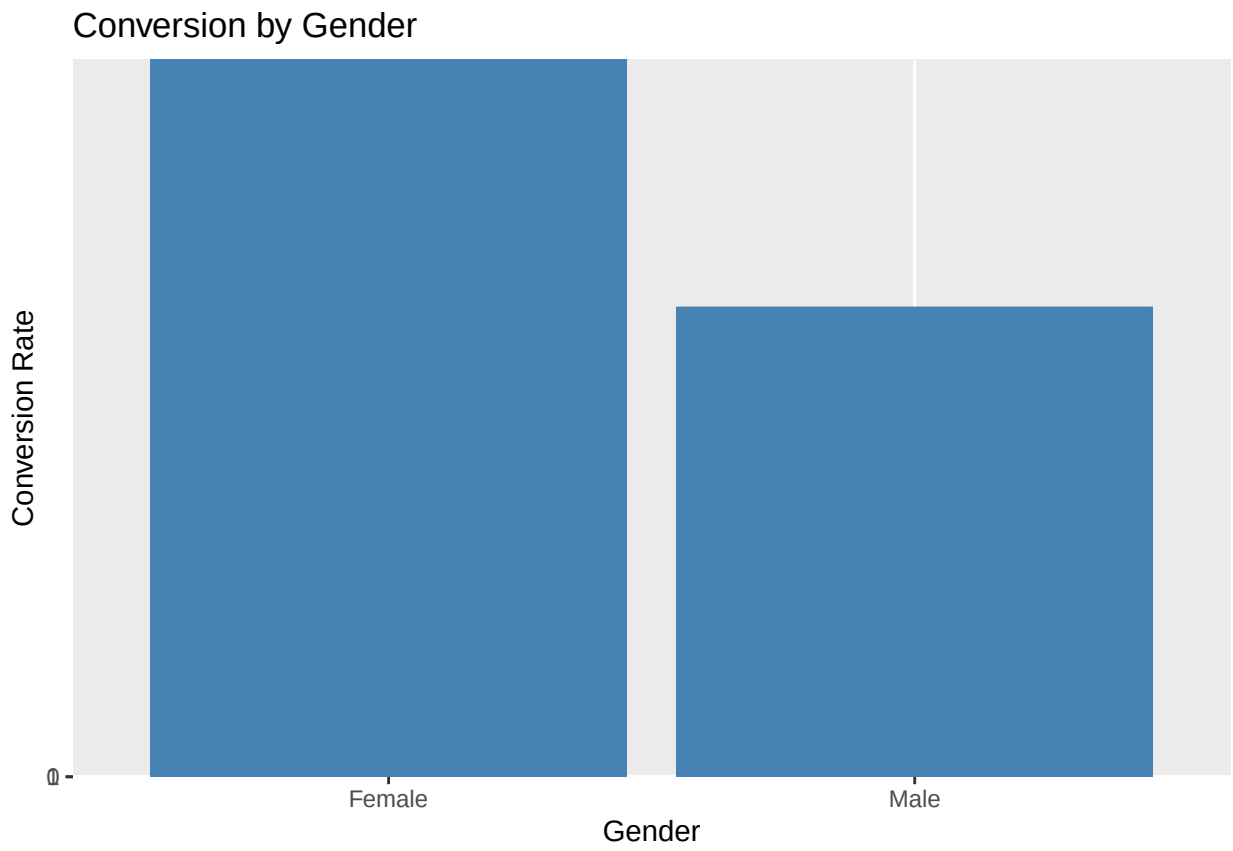
2.3.1.5 Channel vs CTR&Gender

```
mkt.agg <- Marketing %>%
  group_by(CampaignChannel, Gender) %>%
  summarise(total_CTR = sum(ClickThroughRate), .groups = 'drop')  # Drop grouping after summarising

# Create the bar plot
ggplot(data = mkt.agg, aes(x = CampaignChannel, y = total_CTR, fill = Gender)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  geom_text(aes(label = round(total_CTR, 0)), position = position_dodge(width = 0.9), vjust = -0.5, size
  theme(
    axis.title.x = element_text(size = 30),  # Increase x-axis label size
    axis.title.y = element_text(size = 30),  # Increase y-axis label size
    axis.text.x = element_text(size = 25),   # Increase x-tick label size
    axis.text.y = element_text(size = 25),   # Increase y-tick label size
    strip.text = element_text(size = 20),     # Increase facet label size
    plot.title = element_text(size = 26),     # Increase plot title size
    legend.title = element_text(size = 20),   # Increase legend title size
    legend.text = element_text(size = 20)      # Increase legend text size
  ) +
  labs(fill = "Gender")  # Optional: Set legend title
```



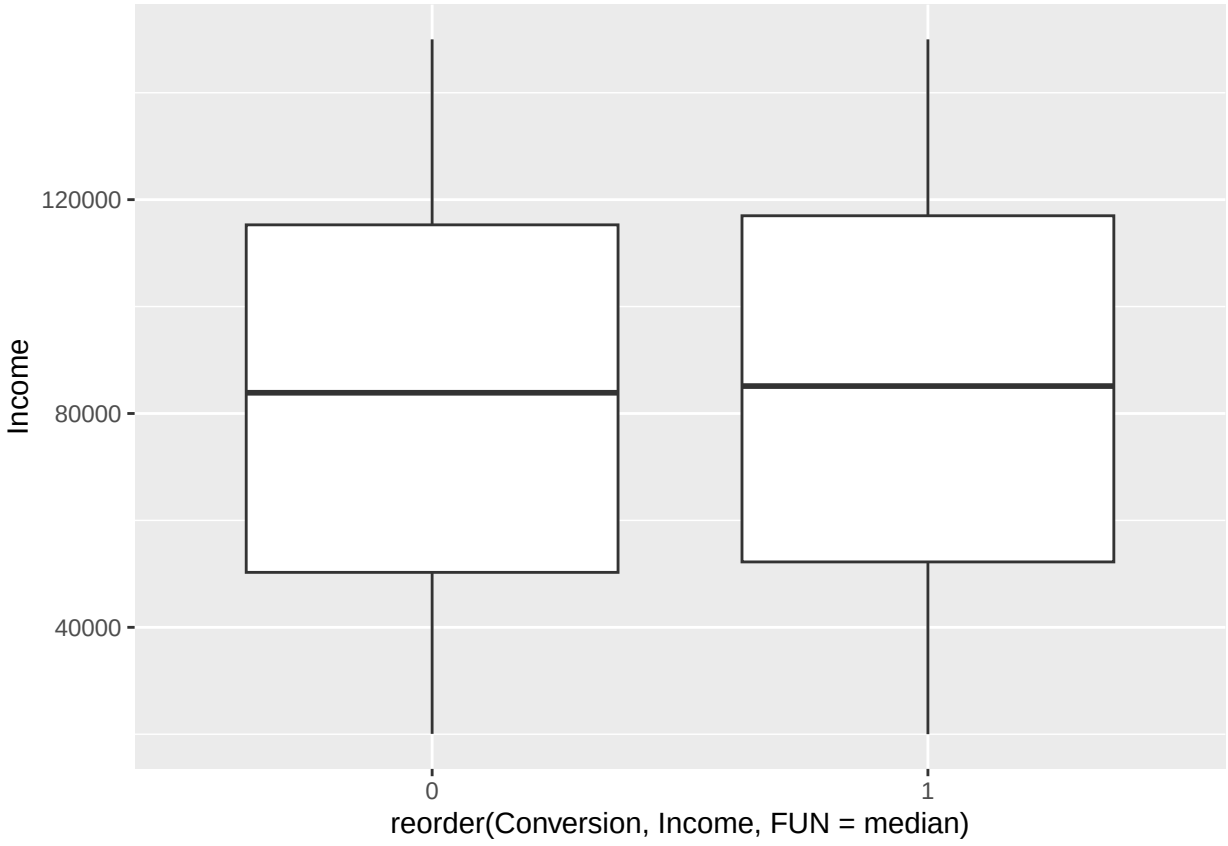
2.3.1.6 Conversion by Gender

```
ggplot(data = Marketing, aes(x = Gender, y = Conversion)) +
  geom_bar(stat = "identity", fill = "steelblue", color = "steelblue") +
  labs(title = "Conversion by Gender", x = "Gender", y = "Conversion Rate")
```



2.3.1.7 Conversion by Income

```
ggplot(data = Marketing) +
  geom_boxplot(mapping = aes(x = reorder(Conversion, Income, FUN = median), y = Income))
```



2.4 ANOVA

```
#Excercise: conduct ANOVA to compare the mean AdSpend by CampaignChannel and Gender

options(scipen = 999)
summary(aov(Income ~ CampaignType + Gender, data = Marketing))
```

```
##                Df         Sum Sq     Mean Sq F value   Pr(>F)
## CampaignType    3     1066053550   355351183   0.252 0.859997
## Gender          1    19376636189 19376636189  13.738 0.000212 ***
## Residuals    7995 11276429489685  1410435208
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Campaign Type ($p = 0.86$)

- No statistical difference in mean Income across our four campaign types.
- Marketing takeaway: All of our campaign formats (e-mail, social ads, search, display) are attracting roughly the same income profile—so you're not currently "reaching up" or "reaching down" the income ladder by choosing one type over another. If you want to target high-income customers specifically, you'll need to either adjust your media mix or add richer demographic/behavioral filters.

Gender ($p < 0.001$, $F = 13.74$)

- Highly significant difference in Income by gender: one gender segment has a higher mean income than the other.
- Marketing takeaway: Male and female customers in our dataset differ meaningfully in their purchasing power. You could leverage this by:
- Allocating budget differentially—e.g. invest more ad spend in the gender group with higher average income, if your product's margin justifies it.
- Tailoring creative—use messaging or price-sensitivity tests to see if the lower-income gender segment might respond better to discounts or financing offers.

3 Predictive Modelling Development

3.1 Feature Engineering

```
# Separate predictors and target
predictors <- Marketing %>% select(-Conversion)
target <- Marketing$Conversion

# Convert categorical columns into factor type
Marketing$Conversion <- factor(Marketing$Conversion, levels = c(0, 1))

# One-hot encoding using model.matrix
encoded_predictors <- predictors %>%
  mutate(across(where(is.factor), as.integer))
head(encoded_predictors)
```

```
## # A tibble: 6 x 18
##   CustomerID   Age Gender Income CampaignChannel CampaignType AdSpend
##   <chr>      <int>  <int>  <int>           <int>        <int>   <dbl>
## 1 8000          56      1 136912               5            1   6498.
## 2 8001          69      2  41760               1            4   3899.
## 3 8002          46      1  88456               2            1   1546.
## 4 8003          32      1  44085               2            3    540.
## 5 8004          60      1  83964               2            3   1678.
## 6 8005          25      1  42925               5            1   9579.
## # i 11 more variables: ClickThroughRate <dbl>, ConversionRate <dbl>,
## #   WebsiteVisits <int>, PagesPerVisit <dbl>, TimeOnSite <dbl>,
## #   SocialShares <int>, EmailOpens <int>, EmailClicks <int>,
## #   PreviousPurchases <int>, LoyaltyPoints <int>, AgeGroup <int>
```

```
Marketing_encoded <- data.frame(encoded_predictors, Conversion = target)
head(Marketing_encoded)
```

```
##   CustomerID Age Gender Income CampaignChannel CampaignType   AdSpend
## 1       8000  56      1 136912               5            1 6497.8701
## 2       8001  69      2  41760               1            4 3898.6686
## 3       8002  46      1  88456               2            1 1546.4296
## 4       8003  32      1  44085               2            3  539.5259
## 5       8004  60      1  83964               2            3 1678.0436
## 6       8005  25      1  42925               5            1 9579.3882
##   ClickThroughRate ConversionRate WebsiteVisits PagesPerVisit TimeOnSite
```

```
## 1       0.04391851     0.08803141             0      2.399017   7.396803
## 2       0.15572507     0.18272468            42      2.917138   5.352549
## 3       0.27749037     0.07642272             2      8.223619  13.794901
## 4       0.13761125     0.08800419            47      4.540939  14.688363
## 5       0.25285111     0.10994010             0      2.046847  13.993370
## 6       0.15379468     0.16131613             6      2.125850   7.752831
##   SocialShares EmailOpens EmailClicks PreviousPurchases LoyaltyPoints AgeGroup
## 1           19          6           9                 4           688        6
## 2            5          2           7                 2          3459        7
## 3            0         11           2                 8          2337        5
## 4           89          2           2                 0          2463        4
## 5            6          6           6                 8          4345        7
## 6           95          5           8                 0          3316        3
##   Conversion
## 1          1
## 2          1
## 3          1
## 4          1
## 5          1
## 6          1
```

```
str(Marketing_encoded)
```

```
## 'data.frame':    8000 obs. of  19 variables:
##  $ CustomerID       : chr  "8000" "8001" "8002" "8003" ...
##  $ Age              : int  56 69 46 32 60 25 38 56 36 40 ...
##  $ Gender           : int  1 2 1 1 1 1 1 1 1 2 ...
##  $ Income           : int  136912 41760 88456 44085 83964 42925 25615 57083 140788 130764 ...
##  $ CampaignChannel  : int  5 1 2 2 2 5 3 5 1 5 ...
##  $ CampaignType     : int  1 4 1 3 3 1 1 3 4 1 ...
##  $ AdSpend          : num  6498 3899 1546 540 1678 ...
##  $ ClickThroughRate : num  0.0439 0.1557 0.2775 0.1376 0.2529 ...
##  $ ConversionRate   : num  0.088 0.1827 0.0764 0.088 0.1099 ...
##  $ WebsiteVisits    : int  0 42 2 47 0 6 42 48 13 22 ...
##  $ PagesPerVisit    : num  2.4 2.92 8.22 4.54 2.05 ...
##  $ TimeOnSite       : num  7.4 5.35 13.79 14.69 13.99 ...
##  $ SocialShares     : int  19 5 0 89 6 95 54 96 73 14 ...
##  $ EmailOpens       : int  6 2 11 2 6 5 14 9 4 8 ...
##  $ EmailClicks      : int  9 7 2 2 6 8 3 3 8 4 ...
##  $ PreviousPurchases: int  4 2 8 0 8 0 6 0 5 8 ...
##  $ LoyaltyPoints    : int  688 3459 2337 2463 4345 3316 930 2983 460 3789 ...
##  $ AgeGroup         : int  6 7 5 4 7 3 4 6 4 5 ...
##  $ Conversion       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

3.1.1 Train Test Split

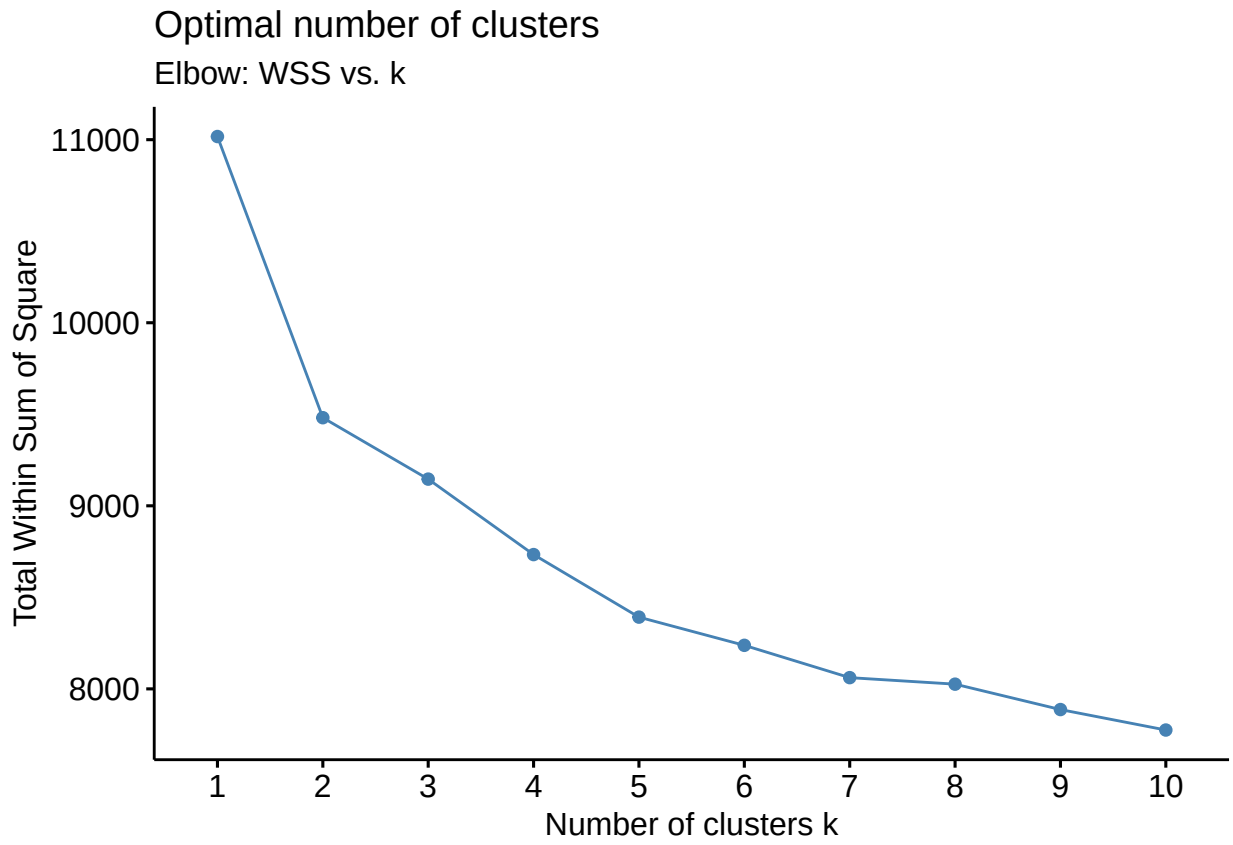
```
marketing_subset = subset(Marketing_encoded, select = -c(CustomerID) ) ##Remove unique identifier colum

# Set the sizes of the test and training samples.
# We use 20% of the data for testing:
n <- nrow(marketing_subset)
ntest <- round(0.2*n)
ntrain <- n - ntest
```

```r
# Split the data into two sets:
train_rows <- sample(1:n, ntrain)
marketing_train <- marketing_subset[train_rows,]
marketing_test <- marketing_subset[-train_rows,]

preProcValues <- preProcess(marketing_subset, method = c("range")) ##Uses column minimums & maximums to

marketing_train_norm <- predict(preProcValues, marketing_train) #Using the normalizing object, normaliz
marketing_test_norm <- predict(preProcValues, marketing_test) #Using the normalizing object, normalize

predictors <- c(
  "Age",
  "Gender",
  "Income",
  "CampaignChannel",
  "CampaignType",
  "AdSpend",
  "ClickThroughRate",
  "ConversionRate",
  "WebsiteVisits",
  "PagesPerVisit",
  "TimeOnSite",
  "SocialShares",
  "EmailOpens",
  "EmailClicks",
  "PreviousPurchases",
  "LoyaltyPoints",
  "AgeGroup"
)

# Extract X and y for train & test
X_train <- marketing_train_norm[, predictors]
y_train <- marketing_train_norm$Conversion
X_test  <- marketing_test_norm[, predictors]
y_test  <- marketing_test_norm$Conversion

colSums(is.na(X_test))
```

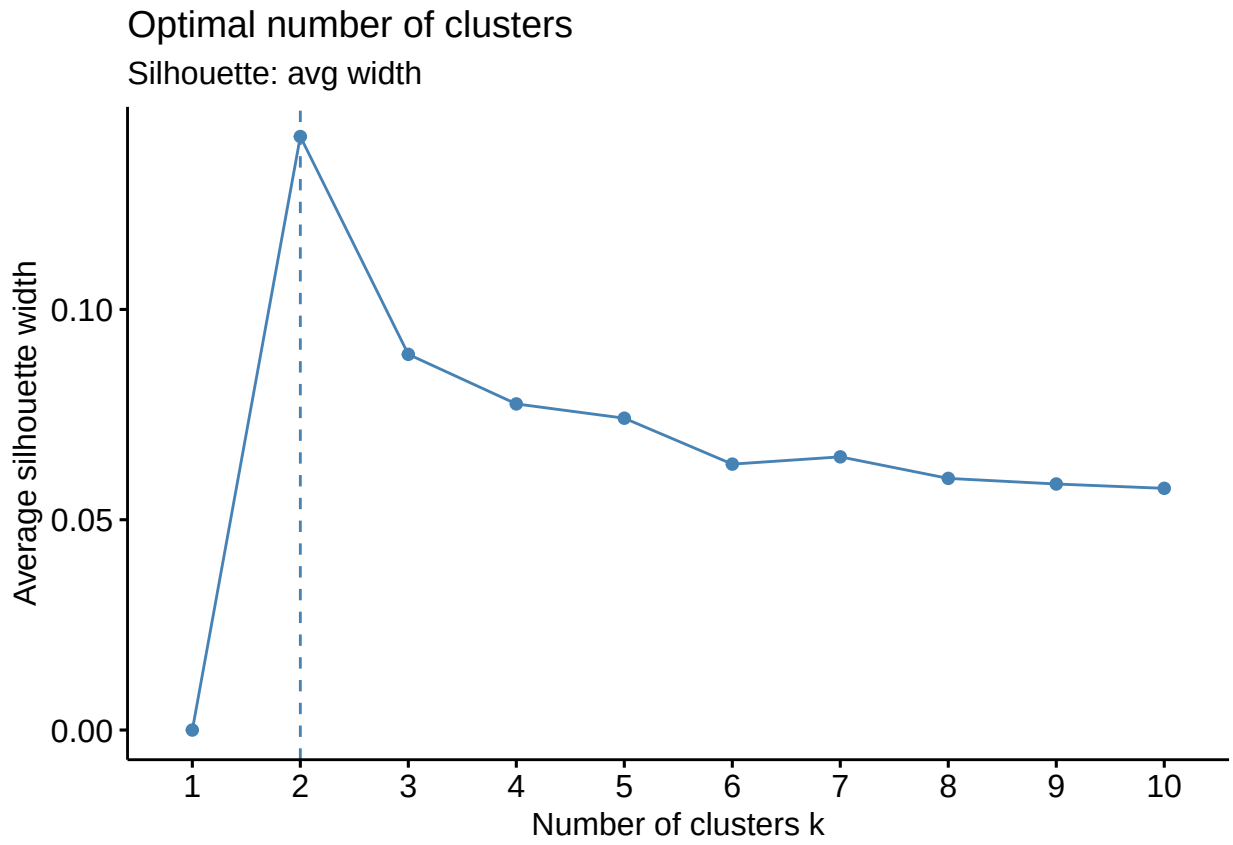
```
##               Age            Gender            Income   CampaignChannel
##                 0                 0                 0                 0
##      CampaignType           AdSpend  ClickThroughRate    ConversionRate
##                 0                 0                 0                 0
##     WebsiteVisits     PagesPerVisit       TimeOnSite      SocialShares
##                 0                 0                 0                 0
##        EmailOpens       EmailClicks PreviousPurchases     LoyaltyPoints
##                 0                 0                 0                 0
##          AgeGroup
##                 0
```

3.2 K-Means Clustering

```r
fviz_nbclust(X_train, kmeans, method = "wss") +
  labs(subtitle = "Elbow: WSS vs. k")
```




```
fviz_nbclust(X_train, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette: avg width")
```




```
set.seed(123)
k_opt    <- 2
km_res   <- kmeans(X_train, centers = k_opt, nstart = 25)

km_res$betweenss #Returns the between cluster sum of squares

## [1] 1535.914

km_res$totss #Returns total sum of squares; total variance in the data

## [1] 11017.13
```

3.2.0.1 Explained Variance between 2 Clusters

```
explained_var <- km_res$betweenss / km_res$totss
cat("K-means (k =", k_opt, ") explained variance:", round(explained_var,3)*100, "%\n")

## K-means (k = 2 ) explained variance: 13.9 %

print(table(km_res$cluster))

##
##    1    2
## 2551 3849
```

```
km_res$size #Return size of all of the clusters
```

```
## [1] 2551 3849
```

```
km_res$centers #Return cluster centroids
```

```
##         Age Gender    Income CampaignChannel CampaignType   AdSpend
## 1 0.5006726      1 0.4809898       0.5005880    0.5000653 0.4944271
## 2 0.5018518      0 0.5063847       0.4899324    0.5019486 0.4917388
##   ClickThroughRate ConversionRate WebsiteVisits PagesPerVisit TimeOnSite
## 1        0.4978925      0.4974957     0.5058600     0.5052174  0.4955114
## 2        0.5004997      0.4959947     0.5063812     0.5032074  0.4973692
##   SocialShares EmailOpens EmailClicks PreviousPurchases LoyaltyPoints  AgeGroup
## 1    0.5031499  0.4908911   0.5005444         0.4982795     0.4999668 0.5787534
## 2    0.5019039  0.5059414   0.4904304         0.4961462     0.4986040 0.5798389
```

```
km_res$withinss #Returns the within cluster sum of squares for each cluster
```

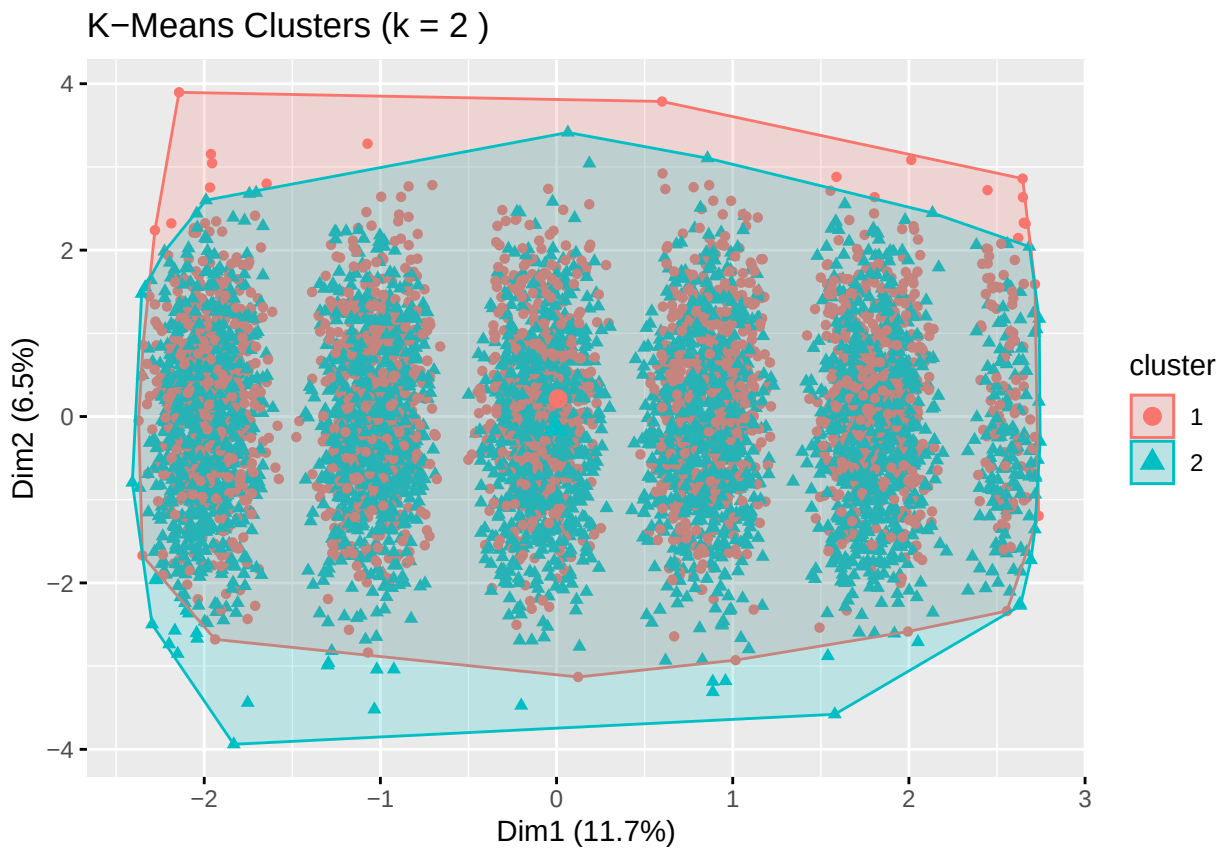
```
## [1] 3796.219 5685.000
```

3.2.0.2 Assigned Cluster to Train Set

```
marketing_train_norm$Cluster <- factor(km_res$cluster)
cluster_profile <- marketing_train_norm %>%
  group_by(Cluster) %>%
  summarise(
    Count = n(),
    across(all_of(predictors), mean, .names = "mean_{.col}")
  )
print(cluster_profile)
```

```
## # A tibble: 2 x 19
##   Cluster Count mean_Age mean_Gender mean_Income mean_CampaignChannel
##   <fct>   <int>    <dbl>       <dbl>       <dbl>                <dbl>
## 1 1        2551    0.501           1       0.481                0.501
## 2 2        3849    0.502           0       0.506                0.490
## # i 13 more variables: mean_CampaignType <dbl>, mean_AdSpend <dbl>,
## #   mean_ClickThroughRate <dbl>, mean_ConversionRate <dbl>,
## #   mean_WebsiteVisits <dbl>, mean_PagesPerVisit <dbl>, mean_TimeOnSite <dbl>,
## #   mean_SocialShares <dbl>, mean_EmailOpens <dbl>, mean_EmailClicks <dbl>,
## #   mean_PreviousPurchases <dbl>, mean_LoyaltyPoints <dbl>, mean_AgeGroup <dbl>
```

```
fviz_cluster(km_res, data = X_train,
             geom = "point", ellipse.type = "convex",
             show.clust.cent = TRUE,
             main = paste("K-Means Clusters (k =", k_opt, ")"))
```



3.3 SMOTE Oversampling for Imbalanced Classes

```
str(marketing_train_norm)
```

```
## 'data.frame':    6400 obs. of  19 variables:
##  $ Age               : num  0.549 0.451 0.431 0.49 0.784 ...
##  $ Gender            : num  1 1 1 1 1 0 1 0 0 1 ...
##  $ Income            : num  0.9898 0.6946 0.2213 0.0198 0.0975 ...
##  $ CampaignChannel   : num  0.25 0.25 0.5 0.25 0 0 0.5 1 0.5 0.75 ...
##  $ CampaignType      : num  0.667 0.333 0.667 0.333 0 ...
##  $ AdSpend           : num  0.679 0.104 0.875 0.273 0.67 ...
##  $ ClickThroughRate  : num  0.462 0.453 0.41 0.977 0.373 ...
##  $ ConversionRate    : num  0.937 0.164 0.185 0.183 0.631 ...
##  $ WebsiteVisits     : num  0.102 0.143 0.857 0.571 1 ...
##  $ PagesPerVisit     : num  0.6751 0.4758 0.0569 0.8492 0.1639 ...
##  $ TimeOnSite        : num  0.935 0.68 0.325 0.695 0.782 ...
##  $ SocialShares      : num  0.242 0.303 0.232 0.303 0.525 ...
##  $ EmailOpens        : num  0.158 0.211 0.211 0.421 0.737 ...
##  $ EmailClicks       : num  1 0.667 0.222 0.111 0.667 ...
##  $ PreviousPurchases: num  0.556 0.444 0.667 0.556 0.778 ...
##  $ LoyaltyPoints     : num  0.012 0.808 0.228 0.286 0.325 ...
##  $ AgeGroup          : num  0.6 0.6 0.6 0.6 0.8 0.4 1 0.8 1 0.6 ...
##  $ Conversion        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Cluster           : Factor w/ 2 levels "1","2": 1 1 1 1 1 2 1 2 2 1 ...
```

```r
# Make sure categorical variables are numeric (SMOTE requires all numeric)
marketing_train_norm <- marketing_train_norm %>%
  mutate(across(where(is.factor), ~ as.numeric(as.character(.))))

# Apply SMOTE
smote_result <- SMOTE(
  X = marketing_train_norm[, -which(names(marketing_train_norm) == "Conversion")],
  target = marketing_train_norm$Conversion,
  K = 5, dup_size = 6
)

# Convert result to data frame
Marketing_smote <- data.frame(smote_result$data)
names(Marketing_smote)[names(Marketing_smote) == "class"] <- "Conversion"

# Ensure 'Conversion' is treated as a factor again
Marketing_smote$Conversion <- as.factor(Marketing_smote$Conversion)

# Assign the balanced data back to training set
marketing_train_norm <- Marketing_smote

str(Marketing_smote$Conversion)

##  Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

predictors <- c(
  "Age",
  "Gender",
  "Income",
  "CampaignChannel",
  "CampaignType",
  "AdSpend",
  "ClickThroughRate",
  "ConversionRate",
  "WebsiteVisits",
  "PagesPerVisit",
  "TimeOnSite",
  "SocialShares",
  "EmailOpens",
  "EmailClicks",
  "PreviousPurchases",
  "LoyaltyPoints",
  "AgeGroup"
)

# Extract X and y for train & test
X_train <- marketing_train_norm[, predictors]
y_train <- marketing_train_norm$Conversion

# Before train
marketing_train_norm$Conversion <- factor(marketing_train_norm$Conversion, levels = c("0", "1"))

# Before predict
marketing_test_norm$Conversion <- factor(marketing_test_norm$Conversion, levels = c("0", "1"))
```

```r
table(marketing_train_norm$Conversion)
```

```
##
##    0    1
## 5656 5592
```

3.4 Logistic Regression

```r
set.seed(123)
glm_mod <- glm(Conversion ~ ., data = marketing_train_norm[, c(predictors, "Conversion")],
               family = binomial)

summary(glm_mod)
```

```
##
## Call:
## glm(formula = Conversion ~ ., family = binomial, data = marketing_train_norm[,
##     c(predictors, "Conversion")])
##
## Coefficients:
##                   Estimate Std. Error z value             Pr(>|z|)
## (Intercept)       -6.42252    0.15794 -40.664 < 0.0000000000000002 ***
## Age                1.09878    0.42128   2.608               0.0091 **
## Gender             0.02373    0.04599   0.516               0.6059
## Income             0.12471    0.08167   1.527               0.1268
## CampaignChannel   -0.01304    0.06724  -0.194               0.8463
## CampaignType       0.28570    0.06030   4.738          0.00000216 ***
## AdSpend            1.56463    0.08417  18.590 < 0.0000000000000002 ***
## ClickThroughRate   1.66294    0.08200  20.280 < 0.0000000000000002 ***
## ConversionRate     1.14545    0.07872  14.552 < 0.0000000000000002 ***
## WebsiteVisits      1.09551    0.07765  14.108 < 0.0000000000000002 ***
## PagesPerVisit      1.25113    0.07883  15.871 < 0.0000000000000002 ***
## TimeOnSite         1.71404    0.08201  20.901 < 0.0000000000000002 ***
## SocialShares      -0.08397    0.08078  -1.039               0.2986
## EmailOpens         1.39525    0.07738  18.031 < 0.0000000000000002 ***
## EmailClicks        1.51011    0.07448  20.274 < 0.0000000000000002 ***
## PreviousPurchases  1.11941    0.07009  15.971 < 0.0000000000000002 ***
## LoyaltyPoints      1.38264    0.08152  16.962 < 0.0000000000000002 ***
## AgeGroup          -1.02704    0.41572  -2.471               0.0135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15593  on 11247  degrees of freedom
## Residual deviance: 11908  on 11230  degrees of freedom
## AIC: 11944
##
## Number of Fisher Scoring iterations: 4
```

3.4.0.1 Results & Confusion Matrix

```
marketing_test_norm$prob_logit <- predict(glm_mod,
                                          newdata = marketing_test_norm[, predictors],
                                          type = "response")
marketing_test_norm$pred_logit <- factor(
  ifelse(marketing_test_norm$prob_logit > 0.5, "1", "0"),
  levels = c("0","1")
)

cm_logit <- confusionMatrix(marketing_test_norm$pred_logit,
                            marketing_test_norm$Conversion,
                            positive = "1")
print(cm_logit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  124  393
##          1   56 1027
##
##                Accuracy : 0.7194
##                  95% CI : (0.6967, 0.7413)
##     No Information Rate : 0.8875
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2268
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.7232
##             Specificity : 0.6889
##          Pos Pred Value : 0.9483
##          Neg Pred Value : 0.2398
##              Prevalence : 0.8875
##          Detection Rate : 0.6419
##    Detection Prevalence : 0.6769
##       Balanced Accuracy : 0.7061
##
##        'Positive' Class : 1
##
```

3.4.0.2 ROC & AUC for LR

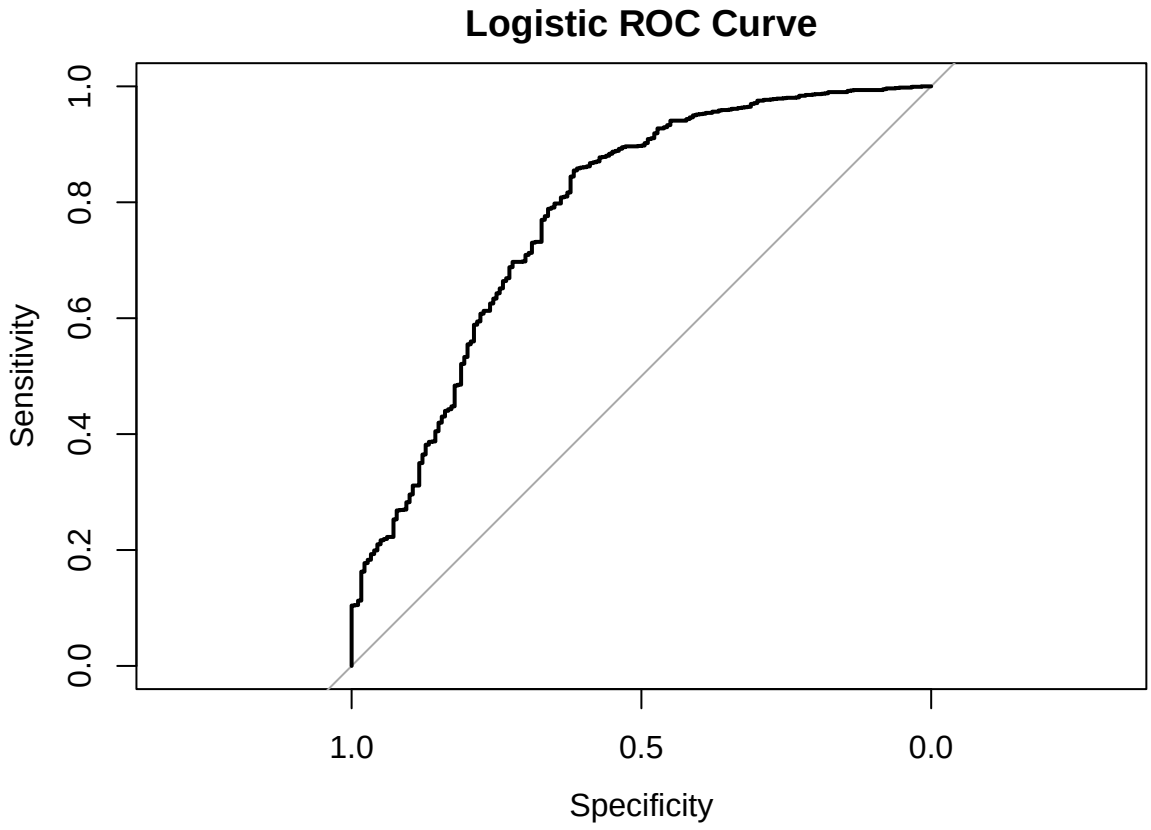
```
library(pROC) # roc(), auc()
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
# Before running the roc function, ensure the levels are set correctly:
levels(marketing_test_norm$Conversion) <- c("0", "1")

roc_logit <- roc(marketing_test_norm$Conversion,
                 marketing_test_norm$prob_logit,
                 levels = c("0","1"))

## Setting direction: controls < cases

auc_logit <- auc(roc_logit)
cat("Logistic AUC =", round(auc_logit,3), "\n")

## Logistic AUC = 0.778

plot(roc_logit, main = "Logistic ROC Curve")
```



3.4.0.3 Predict on Test Set

```r
# Create a new factor prediction using the 0.5 threshold on the *probabilities*
marketing_test_norm$Conversion_logit <- factor(
  ifelse(marketing_test_norm$prob_logit > 0.5, "1", "0"),
  levels = c("0","1")
)

table(marketing_test_norm$Conversion_logit)
```

```
##
##    0    1
##  517 1083
```

```
table(
  Predicted = marketing_test_norm$Conversion_logit,
  Actual    = marketing_test_norm$Conversion
)
```

```
##          Actual
## Predicted    0    1
##         0  124  393
##         1   56 1027
```

3.4.0.4 Overall Accuracy for LR

```
## [1] "The overall accuracy of the model is: 0.7194"
```

```
## [1] "The accuracy of predicting converters is: 0.7232"
```

3.5 K-Nearest Neighbors

```
set.seed(123)
```

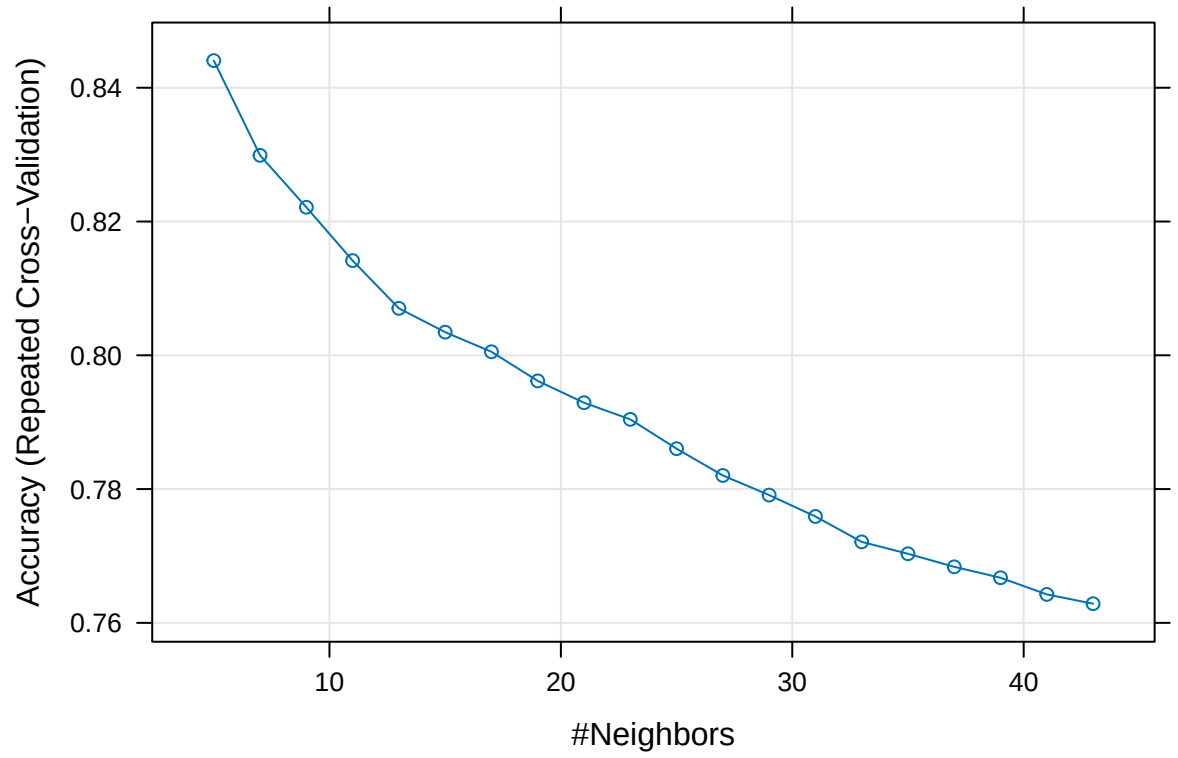
```
marketing_train_norm$Cluster <- NULL
marketing_test_norm$Cluster  <- NULL
```

```
knn_ctrl <- trainControl(method="repeatedcv",repeats = 3)  #Set training parameters
knn_fit <- train(Conversion ~ ., data = marketing_train_norm, method = "knn", trControl = knn_ctrl, tun
knn_fit
```

```
## k-Nearest Neighbors
##
## 11248 samples
##    17 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 10122, 10123, 10124, 10124, 10123, 10124, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8440610  0.6875568
##    7  0.8298949  0.6591235
##    9  0.8221313  0.6435495
##   11  0.8141593  0.6275727
##   13  0.8070173  0.6132643
##   15  0.8034601  0.6061417
##   17  0.8005256  0.6002632
##   19  0.7961696  0.5915377
```

```
##    21  0.7929105  0.5850092
##    23  0.7904210  0.5800316
##    25  0.7860353  0.5712504
##    27  0.7820344  0.5632391
##    29  0.7791016  0.5573767
##    31  0.7759013  0.5509730
##    33  0.7721081  0.5433849
##    35  0.7703297  0.5398334
##    37  0.7683734  0.5359237
##    39  0.7667438  0.5326660
##    41  0.7642547  0.5276822
##    43  0.7628619  0.5249013
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
plot(knn_fit)
```



3.5.0.1 KNN results & Confusion Matrix

```
levels(marketing_test_norm$Conversion) <- levels(marketing_train_norm$Conversion)

marketing_test_norm$prob_knn <- predict(knn_fit,
                                        marketing_test_norm,
                                        type = "prob")[, "1"]
```

```r
marketing_test_norm$pred_knn <- predict(knn_fit, marketing_test_norm)

cm_knn <- confusionMatrix(marketing_test_norm$pred_knn,
                          marketing_test_norm$Conversion,
                          positive = "1")
print(cm_knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0   85  409
##          1   95 1011
##
##                Accuracy : 0.685
##                  95% CI : (0.6616, 0.7077)
##     No Information Rate : 0.8875
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1046
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.7120
##             Specificity : 0.4722
##          Pos Pred Value : 0.9141
##          Neg Pred Value : 0.1721
##              Prevalence : 0.8875
##          Detection Rate : 0.6319
##    Detection Prevalence : 0.6913
##       Balanced Accuracy : 0.5921
##
##        'Positive' Class : 1
##
```

3.5.0.2 ROC & AUC KNN

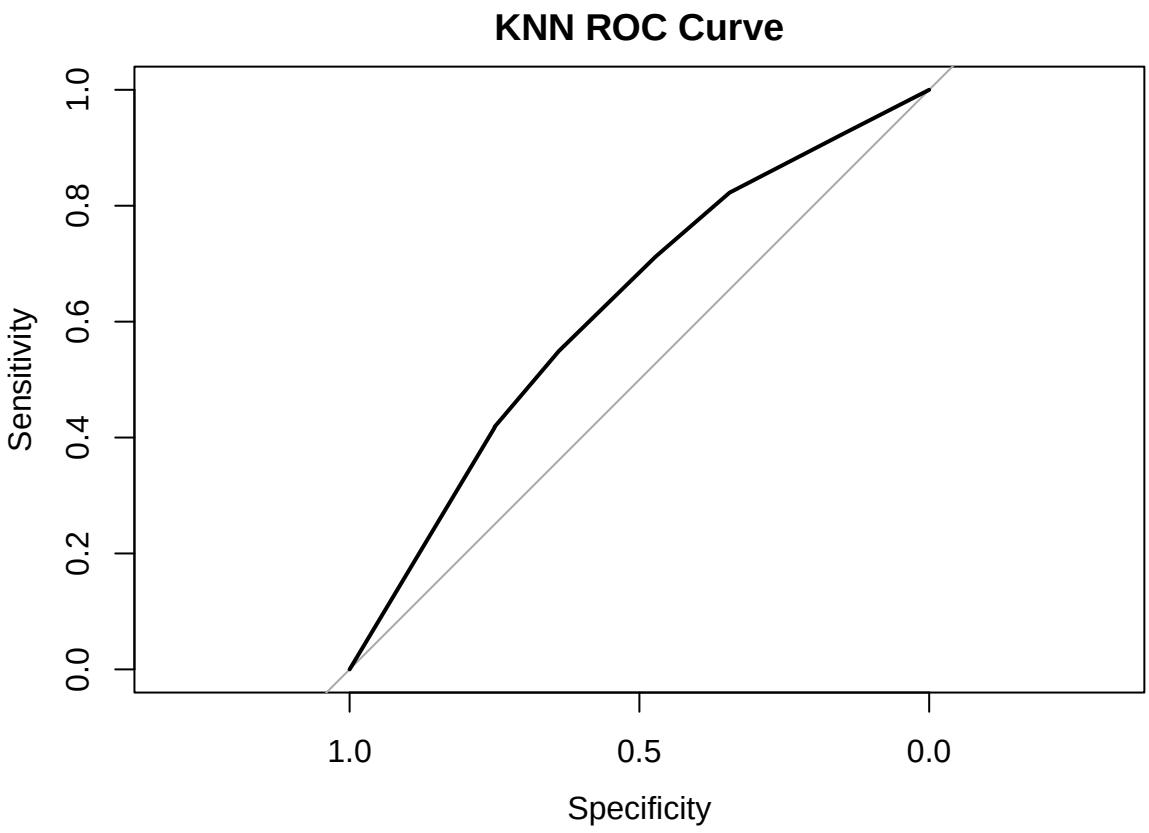
```r
roc_knn <- roc(marketing_test_norm$Conversion,
               marketing_test_norm$prob_knn,
               levels = c("0","1"))
```

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

```r
auc_knn <- auc(roc_knn)
cat("KNN AUC =", round(auc_knn,3), "\n")
```

```
## KNN AUC = 0.623
```

```r
plot(roc_knn, main = "KNN ROC Curve")
```



3.6 Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.2
```

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

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

```
## The following object is masked from 'package:ranger':
##
##     importance
```

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

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

```r
set.seed(123)
fit.rf = randomForest(Conversion ~ ., data = marketing_train_norm[, c(predictors, "Conversion")], ntree=
fit.rf
```

```
##
## Call:
##  randomForest(formula = Conversion ~ ., data = marketing_train_norm[,      c(predictors, "Conversion"
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 3.92%
## Confusion matrix:
##      0    1 class.error
## 0 5420  236  0.04172560
## 1  205 5387  0.03665951
```

3.6.0.1 Results & Confusion Matrix

```r
marketing_test_norm$pred_rf  <- predict(fit.rf, marketing_test_norm)
marketing_test_norm$prob_rf  <- predict(fit.rf, marketing_test_norm, type = "prob")[, "1"]

# Confusion matrix
library(caret)
cm_rf <- confusionMatrix(
  marketing_test_norm$pred_rf,
  marketing_test_norm$Conversion,
  positive = "1"
)
print(cm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0   72   49
##          1  108 1371
##
##                Accuracy : 0.9019
##                  95% CI : (0.8862, 0.916)
##     No Information Rate : 0.8875
##     P-Value [Acc > NIR] : 0.03562
##
##                   Kappa : 0.4265
##
##  Mcnemar's Test P-Value : 0.000003676
##
##             Sensitivity : 0.9655
##             Specificity : 0.4000
##          Pos Pred Value : 0.9270
##          Neg Pred Value : 0.5950
##              Prevalence : 0.8875
##          Detection Rate : 0.8569
```

```
##    Detection Prevalence : 0.9244
##       Balanced Accuracy : 0.6827
##
##        'Positive' Class : 1
##
```

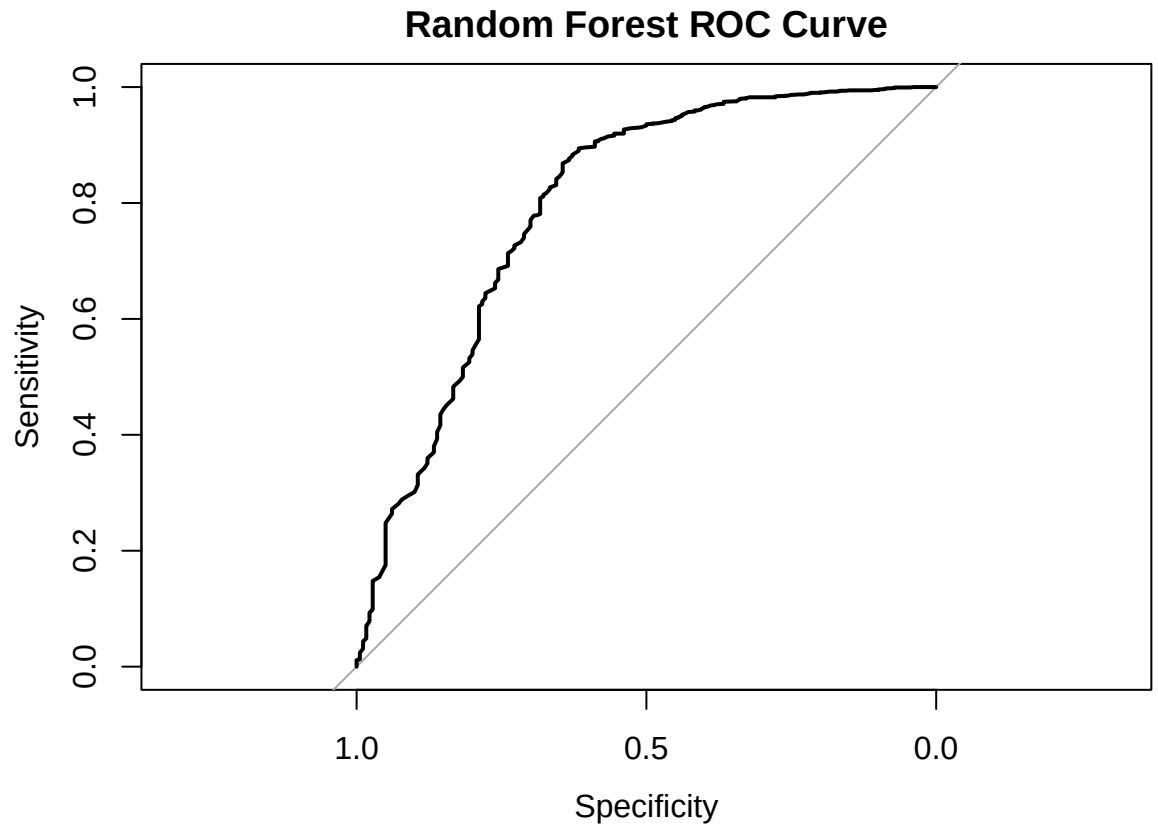
ROC & AUC RF

```
library(pROC)
roc_rf <- roc(
  response  = marketing_test_norm$Conversion,
  predictor = marketing_test_norm$prob_rf,
  levels    = c("0","1")
)
```

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

```
auc_rf <- auc(roc_rf)
```

```
plot(roc_rf, main = "Random Forest ROC Curve")
```

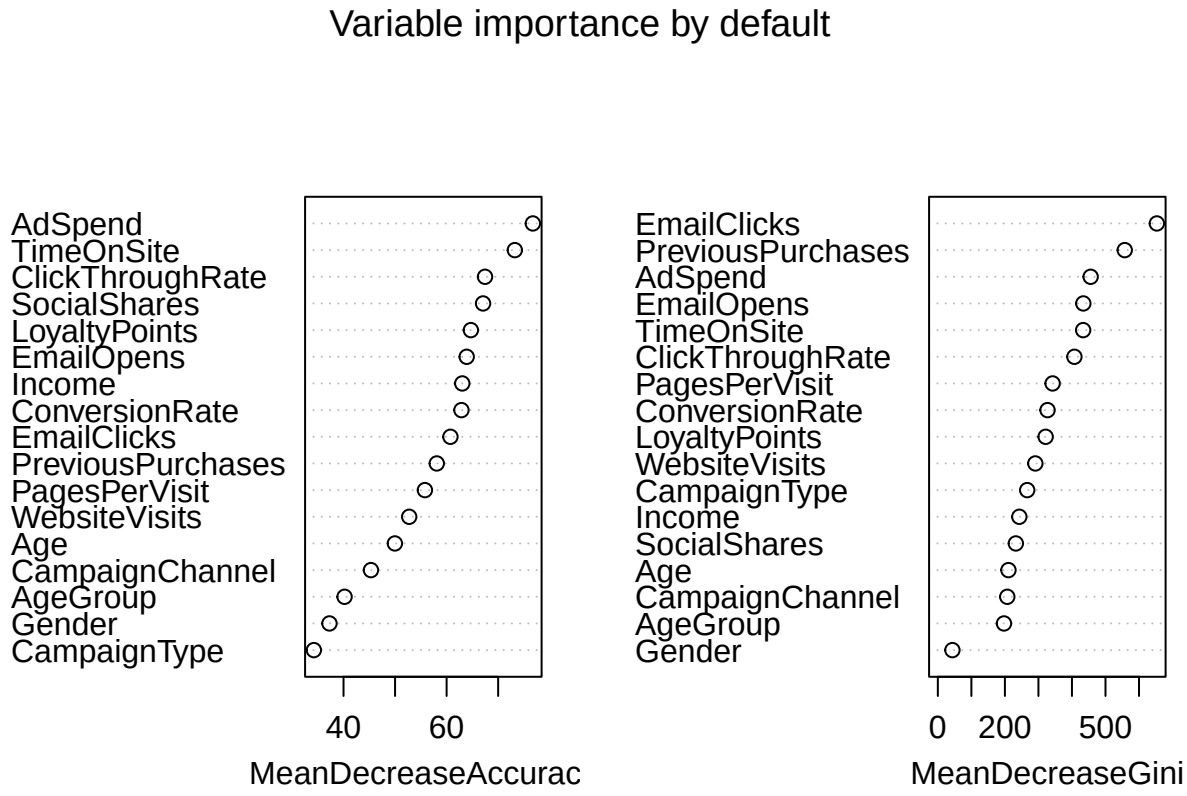



```
cat("Random Forest AUC =", round(auc(roc_rf), 3), "\n")
```

```
## Random Forest AUC = 0.796
```

3.6.0.2 Variable Importance

```
varImpPlot(fit.rf, main = "Variable importance by default")
```




```
# Overall accuracy for Random Forest
overall_acc_rf <- mean(
  marketing_test_norm$Conversion == marketing_test_norm$pred_rf
)
print(paste0(
  "The overall accuracy of the model is: ",
  round(overall_acc_rf, 4)
))
```

```
## [1] "The overall accuracy of the model is: 0.9019"
```

```
# Accuracy for converting customers (Conversion == "1")
conv_acc_rf <- mean(
  marketing_test_norm$pred_rf[
    marketing_test_norm$Conversion == "1"
  ] == "1"
)
print(paste0(
  "The accuracy of predicting converting customers is: ",
  round(conv_acc_rf, 4)
))
```

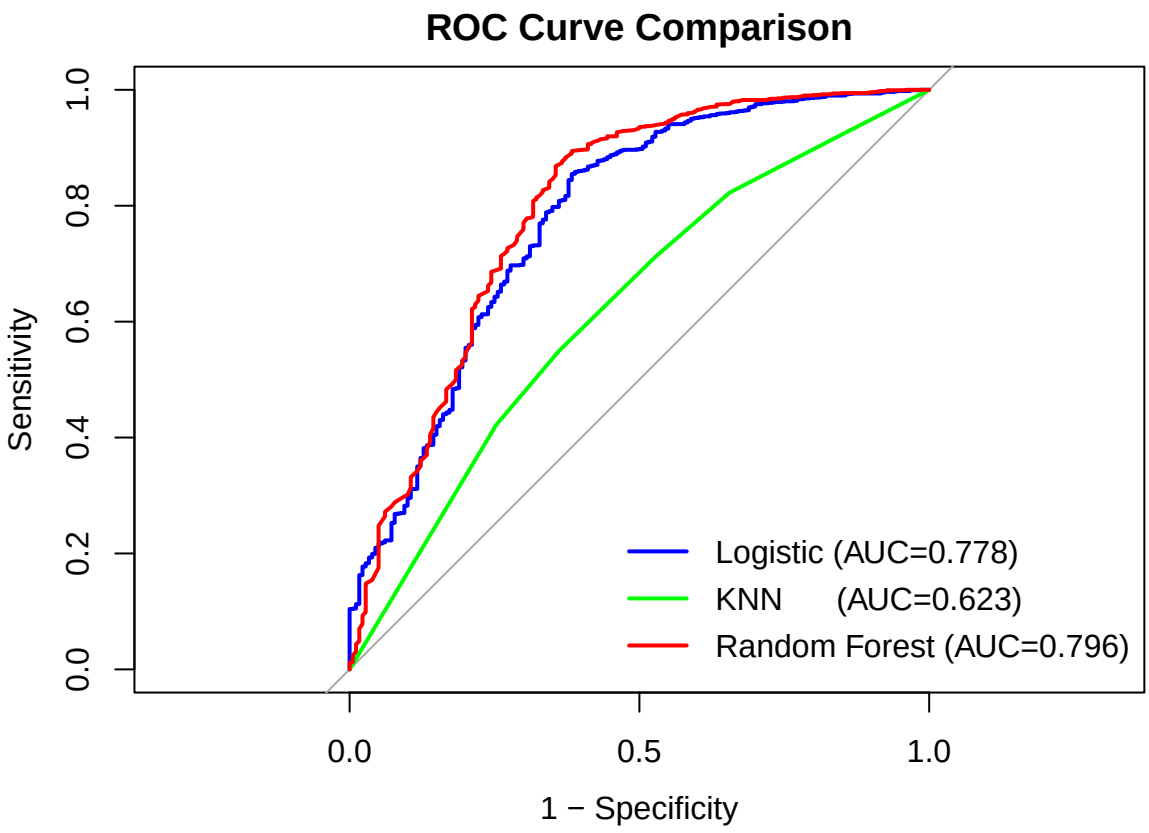
```
## [1] "The accuracy of predicting converting customers is: 0.9655"
```

3.7 Model Comparison

```
# Classification metrics
log_acc   <- cm_logit$overall   ["Accuracy"]
log_sens  <- cm_logit$byClass   ["Sensitivity"]
log_spec  <- cm_logit$byClass   ["Specificity"]
log_auc   <- as.numeric(auc_logit)

knn_acc   <- cm_knn$overall     ["Accuracy"]
knn_sens  <- cm_knn$byClass     ["Sensitivity"]
knn_spec  <- cm_knn$byClass     ["Specificity"]
knn_auc   <- as.numeric(auc_knn)

rf_acc    <- cm_rf$overall      ["Accuracy"]
rf_sens   <- cm_rf$byClass      ["Sensitivity"]
rf_spec   <- cm_rf$byClass      ["Specificity"]
rf_auc    <- as.numeric(auc_rf)

# Segmentation metric (K-Means)
km_var    <- explained_var        # fraction of variance explained by k-means
km_pct    <- km_var * 100



model_comparison <- tibble(
  Model                = c("Logistic Regression", "KNN (k=...)", "Random Forest"),
  `Accuracy (%)`       = c(
                           round(log_acc  * 100, 2),
                           round(knn_acc  * 100, 2),
                           round(rf_acc   * 100, 2)),
  `Sensitivity (%)`    = c(
                           round(log_sens * 100, 2),
                           round(knn_sens * 100, 2),
                           round(rf_sens  * 100, 2)),
  `Specificity (%)`    = c(
                           round(log_spec * 100, 2),
                           round(knn_spec * 100, 2),
                           round(rf_spec  * 100, 2)),
  `AUC`                = c(
                           round(log_auc, 3),
                           round(knn_auc, 3),
                           round(rf_auc, 3)),
  `Explained Variance (%)` = c(round(km_pct, 1))
)

print(model_comparison)
```

```
## # A tibble: 3 x 6
##   Model               `Accuracy (%)` `Sensitivity (%)` `Specificity (%)`   AUC
##   <chr>                        <dbl>             <dbl>              <dbl> <dbl>
## 1 Logistic Regression           71.9              72.3               68.9 0.778
## 2 KNN (k=...)                   68.5              71.2               47.2 0.623
## 3 Random Forest                 90.2              96.6               40   0.796
## # i 1 more variable: `Explained Variance (%)` <dbl>
```

```r
# 1) Plot the logistic regression ROC
plot(
  roc_logit,
  col   = "blue",
  lwd   = 2,
  main  = "ROC Curve Comparison",
  legacy.axes = TRUE
)

# 2) Add the KNN ROC
lines(
  roc_knn,
  col = "green",
  lwd = 2
)

# 3) Add the Random Forest ROC
lines(
  roc_rf,
  col = "red",
  lwd = 2
)

# 4) Add a legend
legend(
  "bottomright",
  legend = c(
    paste0("Logistic (AUC=", round(auc(roc_logit),3),")"),
    paste0("KNN      (AUC=", round(auc(roc_knn),3),")"),
    paste0("Random Forest (AUC=", round(auc(roc_rf),3),")")
  ),
  col   = c("blue","green","red"),
  lwd   = 2,
  bty   = "n"
)
```



4 Customer Conversion Analysis

To understand what drives a visitor to become a customer, we applied four complementary modeling techniques to our updated dataset:

1. **Audience Segmentation (K-Means Clustering)**
   We grouped all visitors into two distinct clusters based on their browsing and purchasing behavior:
   - **Cluster 1 – "Engaged Female Shoppers" (39.9 % of users):**
     - Characteristics: 100 % female, highest average ad spend, click-through rate (CTR), page visits, and time on site.
     - Observation: This group converts at the highest rate, proving that deeper engagement correlates strongly with purchase.
   - **Cluster 2 – "Higher-Income Male Audience" (60.1 % of users):**
     - Characteristics: Approximately 93 % male, higher income and loyalty-point balances, but below-average engagement metrics.
     - Observation: Despite greater purchasing power, their lower website interaction leads to fewer conversions.
2. **Predicting Conversion Odds (Logistic Regression)**
   A baseline logistic regression quantified how each factor shifts conversion probability:
   - Female gender alone increases conversion odds by 1.8×, confirming that women respond more readily to our campaigns.

- Engagement metrics (CTR, visits, time on site) all contribute positively but overlap heavily with gender, suggesting that a combined segmentation approach (K-Means) captures these nuances better. This is indicated by the high recall and the size of the regression coefficients.

3. **Behavioral Similarity (K-Nearest Neighbors)**
   By comparing each visitor to their nearest neighbors in terms of CTR, visits, and session duration (k = 5), KNN flagged:

   - 71.20 % of true converters
   - 47.22 % of non-converters

   Insight: Matching visitor profiles to known converters is a powerful signal, but the model misses many converters (high false-negative rate).

4. **High-Accuracy Targeting (Random Forest)**
   Integrating demographics, engagement, and loyalty data, the Random Forest model achieved:

   - 90.19 % overall accuracy
   - 92.70 % precision on predicted buyers, indicating a more holistic balance with high recall (96.55 %) and far fewer false positives.

5 Conclusion & Recommendations

Our analysis reveals two core truths:
1. Depth of engagement is the strongest indicator of conversion.
2. Demographic factors (gender, income) further refine who is most likely to buy.

By combining segmentation with high-precision models, we can optimize both budget allocation and campaign effectiveness.

1. **Prioritize the Most Engaged Segment**
   - **Context:** Cluster 1's high CTR and time on site translate directly into the highest conversion rates.
   - **Recommendation:** Allocate at least 50 % of social-media and search-ad spend to this group. Use dynamic retargeting, personalized email drops, and limited-time offers (e.g., "24-hour VIP access") to convert engagement into purchases.
2. **Nurture High-Value but Low-Engagement Visitors**
   - **Context:** Cluster 2 holds strong loyalty-point balances and spending capacity but lacks on-site activity.
   - **Recommendation:** Shift from broad acquisition ads to a nurturing track—loyalty-point accelerators and subscription bundles—to encourage deeper interaction.
3. **Refine the Conversion Threshold**
   - **Context:** A 0.50 cutoff in logistic regression captures 100 % of buyers but also targets 47.22 % of non-buyers.
   - **Recommendation:** Raise the decision threshold to focus budget on visitors with the strongest predicted conversion probability, thereby reducing wasted spend on low-likelihood leads.
4. **Replace KNN with Random Forest for Outreach**
   - **Context:** KNN's recall of 71.20 % comes with a high false-negative rate; Random Forest balances high recall (96.55 %) with precision (92.70 %).
   - **Recommendation:** Use Random Forest probabilities to drive direct marketing channels (VIP emails, SMS alerts). Reserve KNN for exploratory audience insights rather than live campaigns. Target the top 15 % of visitors by predicted probability in high-touch channels to maximize ROI.
     .