VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

Ho Chi Minh City University of Technology

PROBABILITY & STATISTICS (2023)

Assignment

# *"Project 1: Ad Click Prediction Dataset"*

| | |
|---|---|
| **Instructor**: | Le Xuan Dai |
| **Class**: | CC01 |
| **Group**: | 1 |
| **Students**: | Nguyen Hoang Quoc – 2353027 |

Ho Chi Minh City, April 2025

# Contents

## List of Figures

## List of Tables

# I Introduction

In the digital age, online advertising has emerged as a fundamental strategy for businesses seeking to engage with potential customers across various platforms. Among the numerous challenges faced by digital marketers, one of the most critical is accurately predicting whether a user will interact with a given advertisement—specifically, whether they will click on it. This capability is vital for optimizing advertising strategies, enhancing user engagement, and ultimately maximizing return on investment (ROI).

Ad Click Prediction represents a well-established problem within the realms of data science and machine learning. It involves analyzing patterns in historical user behavior and advertisement data to estimate the likelihood of a user clicking on an ad. Leveraging such predictive models enables marketers to deliver more targeted and effective campaigns, thereby reducing wasted impressions and improving conversion rates.

The focus of this report is to investigate the Ad Click Prediction dataset in detail. It explores the origin and methodology of data collection, the nature and purpose of the data, and the types of features included. Through this analysis, the report aims to uncover key insights that can inform the development of accurate and efficient predictive models for online advertising.

## 1. Dataset overview

The dataset used for analysis in this topic is taken from the "ad_click_dataset.csv" file. It provides insights into user behavior and online advertising, specifically focusing on predicting whether a user will click on an online advertisement. It contains user demographic information, browsing habits, and details related to the display of the advertisement.

Here is some information about this dataset:

- **Author:** Ciobanu Marius
- **Data source:** The dataset on Kaggle was synthetically generated for educational and research purposes. It is not sourced from a real-world advertising platform but was created to simulate realistic user interaction with online advertisements.
- **Sample size:** 10000
- **Number of features:** 9

The original dataset is provided at:
https://www.kaggle.com/datasets/marius2303/ad-click-prediction-dataset

## 2. Selected dataset

Here is the table list of the dataset we chose to analyze:

| Name | Data Type | Non-null count | Null count |
|---|---|---|---|
| id | numerical | 10000 | 0 |
| full_name | categorical | 10000 | 0 |
| age | numerical | 5234 | 4766 |
| gender | categorical | 5307 | 4693 |
| device_type | categorical | 8000 | 2000 |
| ad_position | categorical | 8000 | 2000 |
| browsing_history | categorical | 5218 | 4782 |
| time_of_day | categorical | 8000 | 2000 |
| click | target variable (binary) | 10000 | 0 |

**Table 1:** Dataset overview

### 2.1 id

ID serves as a unique identifier for users in the database. In a system with millions of customers, ID helps to retrieve data quickly, accurately, and efficiently. Eliminating duplication is an important part of analyzing and synthesizing accurate information, thereby improving performance and processing speed.

### 2.2 full_name

The user's first and last name belong to the personal information group. Because this variable does not directly affect user behavior, it is rarely used in analytical or predictive models.

### 2.3 age

An important demographic trait through which we can know what kind of ads a person of that age is interested in. This variable can affect variables such as device_type and browsing_history. Therefore, depending on the different age groups, we will have similar advertising campaigns that help optimize performance and avoid wasting advertising budgets.

### 2.4 gender

It is the user's gender, which consists of 3 values: Male, Female, and non-Binary. Similar to age, it is an important demographic trait. However, this variable may have missing values, as users do not want to disclose, which directly affects the quality of the analysis if there is no processing method.

### 2.5 device_type

Describe the type of device the user is using to access, including Desktop and Mobile. However, there is still a possibility of a system error or the browser not providing information, leading to the need to take measures to deal with this missing value.

### 2.6 ad_position

Reflects ad placement, including options such as Top and Side. Location affects the visibility and engagement with your ads. But like device_type, this variable also has a missing value.

### 2.7 browsing_history

It stores information about the user's browsing behavior, which is categorized into groups such as Shopping, Education, Entertainment, and Social Media. This is a highly valuable behavioral variable in predictive analytics, as it reflects the user's interests and preferences. However, this variable requires an effective handling strategy to take advantage of its value when the user is not filling in information or is in private mode

### 2.8 time_of_day

Represents the time of day that the ad was shown, including the Morning, Afternoon, Evening and Night periods. This is an important time factor, helping to determine when users are more likely to click on the ad. This variable analysis helps optimize ad delivery time to improve campaign performance.

### 2.9 click

Represents the time of day that the ad was shown, including the Morning, Afternoon, Evening and Night periods. This is an important time factor, helping to determine when users are more likely to click on the ad. This variable analysis helps optimize ad delivery time to improve campaign performance.

The primary goal of analyzing variables in this dataset is to predict whether a user will click on an advertisement. By identifying patterns and influential factors behind ad clicks, businesses can optimize their marketing strategies, target the right audience, and increase overall campaign efficiency. This analysis enables data-driven decisions that enhance personalization, improve user engagement, and ultimately maximize return on investment (ROI).

# II Theory

## 1. Exploratory data analysis (EDA)

Exploratory Data Analysis (EDA) is a fundamental process in data science aimed at unveiling the underlying structure of a dataset. It bridges the gap between raw data and advanced statistical modeling by providing insights through intuitive and graphical methods. Unlike traditional confirmatory approaches, which test predefined hypotheses, EDA is inherently open-ended, encouraging exploration and discovery.

### 1.1 EDA Objectives

EDA aims to:

- **Understand the data's story:** Uncover patterns, trends, and hidden relationships; detect anomalies, outliers, and unexpected behaviors.
- **Assess Data Quality & Readiness:** Identify missing values, duplicates, and inconsistencies; also validate data distributions and statistical properties.
- **Guide Effective Decision-Making:** Formulate hypotheses for further testing; determine suitable data preprocessing (cleaning, transformations).
- **Optimize Modeling Strategies:** Select relevant features for machine learning; choose appropriate statistical or ML techniques.
- **Communicate Insights Clearly:** Visualize key findings for stakeholders; summarize data characteristics in an interpretable way.

### 1.2 EDA Tools/Techniques

During Exploratory Data Analysis (EDA), analysts use various visualization techniques and statistical tools to examine data based on its characteristics and research goals. Key methods include:

- **Histograms** – Display how numerical data is distributed, revealing patterns like skewness, multiple peaks, or unusual values.
- **Box Plots** – Summarize numerical data distributions, showing medians, quartiles, and outliers, making them ideal for comparing groups.

- **Scatter Plots** – Explore relationships between two continuous variables, exposing potential correlations or grouped data points.
- **Time Series Plots** – Track data over time to detect trends, repeating patterns, or unexpected fluctuations.
- **QQ Plots** – Compare dataset quantiles with a theoretical distribution (e.g., normal distribution) to check if assumptions about the data's shape hold true.

These tools help uncover insights, validate assumptions, and guide further analysis.

### 1.3 EDA Analysis Approach

In this topic, the analysis will systematically employ foundational visualization tools—including histograms to reveal distributional characteristics, box plots to compare subgroups and detect outliers, and correlation matrices to quantify variable relationships—as critical first steps in our exploratory process. These methods provide an efficient starting point to understand variable distributions, relationships, and overall data structure, laying the groundwork for more sophisticated modeling efforts.

## 2. Histogram

In statistics, a histogram is a type of bar graph that represents the distribution of numerical data. It groups data into bins (also called intervals) and shows how many data points fall into each bin.

The main purposes of a histogram are:

- **Visualizing the distribution of data:** It helps to quickly see patterns such as symmetry, skewness, or the presence of outliers.
- **Understanding frequency:** It shows how frequently data points fall within different ranges or intervals.
- **Detecting data shape:** You can determine if the data is normally distributed, uniform, skewed, etc.
- **Identifying trends or patterns:** For example, it can highlight if most values are concentrated in a certain range.
- **Simplifying large data sets:** A histogram condenses data into a form that's easier to understand at a glance.

However, histograms are distinct from bar charts

| **Bar charts** | **Histogram** |
|---|---|
| Represent one-dimensional data | Represent two-dimensional data |
| The volume of the bars usually does not have any meaning | The area of one bar illustrates the frequency of the data |
| The bars are separated by equal spaces | Because of the continuity of the graph, the bars are adjacent to each other |

## 3. Correlation matrix

A correlogram, also known as a correlation matrix, is a tool to visualize the relationships between pairs of variables in a dataset. It is often displayed as a matrix of scatterplots with symbols such as lines and bubbles to represent correlation coefficients ($r_{xy}$).

Correlation coefficients range: $-1 \leq r_{xy} \leq 1$.

- $-1 \leq r_{xy} < 0$ : negative correlation. $r_{xy}$ is closer to -1, indicating a stronger negative correlation between X and Y, meaning that as one variable increases, the other decreases strongly and consistently.
- $0 \leq r_{xy} < -1$ : positive correlation. $r_{xy}$ is closer to 1, indicating a stronger positive correlation between X and Y, meaning that when one variable increases, the other also increases strongly and consistently.
- $r_{xy}$ **is closer to 0** ,indicating a weak correlation between X and Y.
- $r_{xy} =$ **0:** indicating linear independence between X and Y, meaning X and Y are independent and do not affect each other.

Using a correlogram could help us define

- Whether the data is random or not
- How one observation is related to the next one
- What kind of model might fit the data best (especially in time series analysis)

### 4. Box plot

Box Plot (also called a box-and-whisker plot) is a method for demonstrating graphically the locality, **spread, and skewness** groups of numerical data including minimum, quartiles, and maximum (excluding outliers). The **box** shows the range between Q1 and Q3 (called the interquartile range or IQR), and **whiskers** extend to the minimum and maximum values.

Box plots are particularly useful for:

- Showing the spread of data (how data is distributed)
- Identifying outliers (unusual values that stand out)
- Comparing distributions between different groups or datasets
- Spotting skewness (whether data is more spread on one side
- Summarizing large datasets with just a few key statistics

## III Hypothesis

### 1. Gender influences ad clicking behavior

#### 1.1 Problem Statement and Hypotheses

This section investigates whether gender has a statistically significant influence on consumers' likelihood of clicking on online advertisements. Specifically, the analysis seeks to determine if there is a difference in click behavior between male and female users.

- **Null Hypothesis** ($H_0$): There is no difference in ad click rates between males and females.
- **Alternative Hypothesis** ($H_1$): There is a significant difference in ad click rates between males and females.

This hypothesis is tested using both a bivariate test of independence and a multivariate modeling approach to assess the isolated and combined effects of gender in relation to other user attributes.

## 1.2 Applicable Analytical Methods

**a) Chi-Square Test of Independence**
The Chi-Square test is used to examine the association between two categorical variables: gender (male, female) and click behavior (clicked, not clicked). A contingency table is constructed to observe the frequency distribution of clicks across genders:

| | **Click = 1** | **Click = 0** | **Total** |
|---|---|---|---|
| **Male** | $a$ | $b$ | $a+b$ |
| **Female** | $c$ | $d$ | $c+d$ |
| **Total** | $a+c$ | $b+d$ | $N=a+b+c+d$ |

The Chi-Square statistic is calculated using the formula:

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

- $O_{ij}$: **Observed frequency** in cell $(i, j)$
- $E_{ij}$: **Expected frequency** for cell $(i, j)$, calculated as:

$$E_{ij} = \frac{(\text{Row Total}) \times (\text{Column Total})}{N}$$

The degrees of freedom (df) for a $2 \times 2$ table is calculated as:

$$df = (Number\ of\ rows - 1) \times (Number\ of\ columns - 1)$$
*In this case: df = (2 - 1) × (2 - 1) = 1*

The calculated $\chi^2$ value is compared against the chi-square distribution with $df = 1$ to obtain a p-value.

- If $p \leq \alpha$ (e.g., 0.05), reject the null hypothesis.
- If $p > \alpha$, fail to reject the null hypothesis.

**b) Logistic Regression**
To account for multiple influencing factors simultaneously, a logistic regression model is constructed with the binary response variable clicked (1 = clicked, 0 = not clicked) and gender as one of the predictors.

The logistic regression equation is defined as:

$$\log\left(\frac{P(\text{y}=1)}{1-P(\text{y}=1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

- The left-hand side represents the log-odds of the click event.
- $\beta_i$ represents the coefficients for predictor variables $X_i$ including gender.
- The effect of each variable is assessed using the Wald test, which provides a p-value for the null hypothesis $H_0 : \beta_i = 0$.
- If $p < \alpha$ (0.05), the predictor is considered statistically significant.
- If $p \geq \alpha$, the predictor is not considered statistically significant.

### 1.3 Methodology and Results

**Data Processing:** Missing values in the `gender` variable were addressed using imputation techniques such as assigning a placeholder category ("Unknown").

**Model Implementation in R:**

```r
model <- glm(clicked ~ age + gender + device_type + ad_position +
             browsing_history + time_of_day,
             data = data, family = "binomial")
summary(model)

```

**Findings:** In the logistic regression output, the p-value for the `gender` variable was 0.03, which is below the commonly used threshold of $\alpha = 0.05$.

**Conclusion:** The null hypothesis is rejected. Therefore, the analysis provides statistical evidence that gender has a significant influence on the likelihood of clicking on advertisements.

## 2. The Influence of Device Type on Ad Clicking Behavior

### 2.1 Problem Statement and Hypotheses

This section evaluates whether the type of device used (e.g., mobile, desktop, tablet) has a significant impact on a user's propensity to click on online advertisements. Since device type is a key contextual factor in user interaction, understanding its effect is crucial for optimizing ad delivery strategies.

- **Null Hypothesis** ($H_0$): The probability of clicking on an ad is independent of the device type used.
- **Alternative Hypothesis** ($H_1$): The probability of clicking on an ad differs depending on the device type.

This hypothesis will be examined using both categorical comparison and multivariate modeling methods.

### 2.2 Applicable Analytical Methods

**Logistic Regression:**
Given that the dependent variable (`clicked`) is binary (1 = clicked, 0 = not clicked), logistic regression is the most suitable method to examine how continuous predictors such as age influence the outcome.

The logistic regression equation is formulated as:

$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

- $\beta_i$ represents the coefficients associated with the **age** variable.
- The exponentiated value $e^{\beta_1}$ represents the **odds ratio**: the change in odds of clicking for each one-unit increase in age.

Interpretation Result:

- If $e^{\beta_1} > 1$: An increase in age increases the odds of clicking.
- If $e^{\beta_1} < 1$: An increase in age decreases the odds of clicking.
- A p-value is provided to test the statistical significance of $\beta_1$:
  - If $p < 0.05$, the relationship is considered statistically significant.
  - If $p \geq 0.05$, there is insufficient evidence to conclude an effect.

### 2.3 Methodology and Results

**Data Processing:** Age was retained as a continuous numeric variable. No transformations or binning into age groups were applied at this stage to preserve granularity.

#### Model Implementation in R:

```r
model <- glm(clicked ~ age + gender + device_type + ad_position + browsing_history + time_of_day,
data = data, family = "binomial")
summary(model)

```

**Findings**: The p-value for the age coefficient was observed to be $< 0.001$, indicating strong statistical significance.
The coefficient $\beta_{\text{age}}$ was negative, and the odds ratio $e^{\beta}$ was less than 1, suggesting that as age increases, the likelihood of clicking on an advertisement decreases.

**Conclusion**: The null hypothesis is rejected. The analysis confirms that age has a significant negative effect on ad click behavior. Older users are statistically less likely to click on ads compared to younger users

## 3. Time of Day Influences Ad Clicking Behavior

### 3.1 Problem Statement and Hypotheses

This section investigates whether the time of day influences users' behavior in clicking advertisements. Temporal patterns of internet usage may affect attentiveness or intent, leading to variations in ad engagement across different periods of the day.

- **Null Hypothesis ($H_0$):** The time of day has no effect on the likelihood of clicking an advertisement.
- **Alternative Hypothesis ($H_1$):** The time of day has a significant effect on the likelihood of clicking an advertisement.

### 3.2 Applicable Analytical Methods

**Categorical Logistic Regression with Dummy Variables** The variable `time_of_day` is a categorical feature with multiple levels (*e.g., morning, afternoon, evening, night*). To include it in a logistic regression model, each category (except one reference category) is encoded as a dummy variable.
The logistic regression model is:

$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \sum_{i=1}^{k-1} \beta_i \cdot time\_of\_day_i + \dots$$

Where:

- $time\_of\_day_i$ are binary dummy variables representing different time periods (e.g., morning = 1, otherwise = 0).
- One category (e.g., *morning*) is omitted as the reference level.

Interpreting Results:

- Each $beta_i$ represents the log-odds difference between that time period and the reference category.
- A p-value $< 0.05$ for any $\beta_i$ suggests the corresponding time period significantly differs from the reference in terms of click behavior.

### 3.3 Methodology and Results

**Data Processing**: The `time_of_day` variable was converted into a factor with levels: *morning, afternoon, evening, and night.* The reference category was set to *morning*.
**Model Implementation in R:**

```r
data$time_of_day <- factor(data$time_of_day, levels = c("morning", "afternoon", "
evening", "night"))
model <- glm(clicked ~ age + gender + device_type + ad_position + browsing_history +
time_of_day,
data = data, family = "binomial")
summary(model)
```

**Findings**: From the logistic regression output:

- The coefficients for afternoon and evening time periods had p-values $< 0.05$, suggesting statistically significant differences compared to the *morning*.
- The odds ratio for *evening* was $> 1$, indicating users are more likely to click on ads in the evening than in the morning.
- The *night* period showed a non-significant p-value $p > 0.05$, implying no strong evidence of behavioral difference compared to the *morning*.

**Conclusion**: The null hypothesis is partially rejected. Time of day does influence ad_clicking behavior, particularly during the *afternoon* and *evening*, when users exhibit higher engagement than in the *morning*.

# IV Datasets and model

## 1. Commonly used dataset:

### 1.1 Criteo Display Advertising Challenge Dataset

Released by Criteo Labs, this dataset contains 45 million anonymized ad impressions with both categorical and numerical features. It is widely used for benchmarking click-through rate (CTR) prediction models due to its scale and real-world nature.

### 1.2 Avazu Click-Through Rate Prediction Dataset

Provided for a Kaggle competition, this dataset includes millions of ad impressions with anonymized features like device type, app category, and hour of the click. It is commonly used for evaluating machine learning models in mobile ad CTR prediction.

### 1.3 iPinYou Dataset

Collected from a real Demand-Side Platform (DSP) in China, the iPinYou dataset includes logs of ad impressions, clicks, and bids. It is used in research for real-time bidding (RTB) and CTR prediction, offering a realistic view of online advertising auctions.

### 1.4 Tencent Advertising Dataset (KDD Cup 2017)

Released for the KDD Cup 2017, this dataset contains extensive logs from Tencent's ad platform, including user behavior, ad features, and click labels. It is especially valuable for user-level CTR prediction and large-scale model training.

## 2. Binary Logistic Regression model and random forest comparison

### 2.1 Comparative Analysis of Logistic Regression, Random Forest, and XGBoost for Click-Through Rate Prediction in Digital Advertising

**Authors:** Jiacheng Lou
**Published:** October 2024

**Summary:**

| Aspect | Logistic Regression (LR) | Random Forest (RF) |
|---|---|---|
| **Accuracy** | Baseline | 0.3% higher than LR |
| **Speed** | Very fast (efficient on large data) | **10x slower** than LR |
| **Computational Cost** | Low | High |
| **Model Complexity** | Simple, easy to interpret | Complex, harder to interpret |
| **Performance Gain** | Improves with data cleaning / features | Also improves, but needs more power |
| **Suitability** | Best for fast, interpretable results | Better for slightly higher accuracy |

### 2.2 Random Forest versus Logistic Regression: A Large-Scale Benchmark Experiment

**Authors:** Raphael Couronné, Philipp Probst, Anne-Laure BoulesteixBioMed Central
**Published:** July 2018
**Summary:**

| Criteria | Logistic Regression (LR) | Random Forest (RF) |
|---|---|---|
| **Overall Accuracy** | Lower in most cases | Outperformed LR in 69% of datasets |
| **AUC (ROC Curve)** | Moderate | **H**igher AUC in majority of cases |
| **Data Fit** | Best with linear relationships | Handles complex, non-linear relationships |
| **Interpretability** | High – coefficients are easy to explain | Low – model is more of a black box |
| **Computation Time** | Faster | Slower, especially on large datasets |
| **Robustness to Noise** | Less robust | More robust due to ensemble nature |
| **Best Use Case** | When simplicity, speed, and interpretability are key | When accuracy and handling complexity matter |

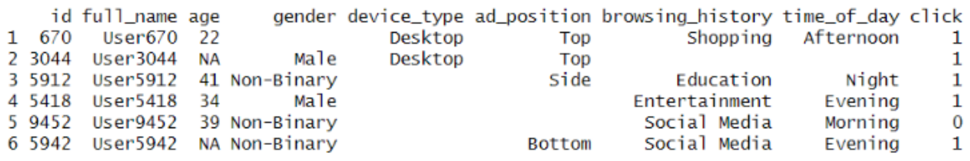
# V Technical Ideas

## 1. Data Preprocessing

### 1.1 Data Reading

The very first step is to extract the information from the ad_click_dataset.csv file.

```
    data <- read.csv("C:/Users/Admin/Desktop/ad_click_dataset.csv")
    head(data)

```

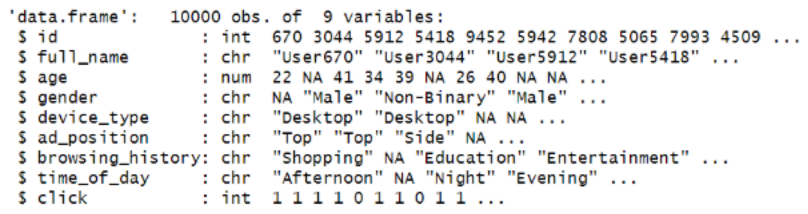


```
    id full_name age     gender device_type ad_position browsing_history time_of_day click
1  670   User670  22              Desktop         Top         Shopping   Afternoon     1
2 3044  User3044  NA       Male   Desktop         Top                                  1
3 5912  User5912  41 Non-Binary                  Side        Education       Night     1
4 5418  User5418  34       Male                          Entertainment     Evening     1
5 9452  User9452  39 Non-Binary                           Social Media     Morning     0
6 5942  User5942  NA Non-Binary                Bottom     Social Media     Evening     1
```

**Figure 1:** *The first 6 rows of the dataset*

Command head(data) is used to display the first few rows of the dataset.
As missing values come in both NA and empty space, we will replace them with NA for convenient processing and coding.

```
# Replace empty strings with NA
data[data== ""] <- NA
str(data)
```



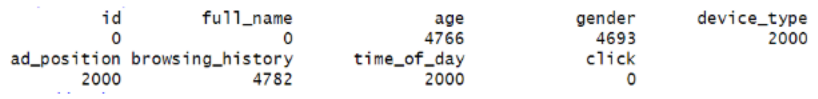
```
'data.frame':   10000 obs. of  9 variables:
 $ id              : int  670 3044 5912 5418 9452 5942 7808 5065 7993 4509 ...
 $ full_name       : chr  "User670" "User3044" "User5912" "User5418" ...
 $ age             : num  22 NA 41 34 39 NA 26 40 NA NA ...
 $ gender          : chr  NA "Male" "Non-Binary" "Male" ...
 $ device_type     : chr  "Desktop" "Desktop" NA NA ...
 $ ad_position     : chr  "Top" "Top" "Side" NA ...
 $ browsing_history: chr  "Shopping" NA "Education" "Entertainment" ...
 $ time_of_day     : chr  "Afternoon" NA "Night" "Evening" ...
 $ click           : int  1 1 1 1 0 1 1 0 1 1 ...
```

**Figure 2:** *Raw data*

## 1.2 Eliminating NA values

When dealing with missing data, dropping rows or columns containing NA values might seem like a clean solution, but it comes at a cost. Removing data can lead to data shrinking, where the overall dataset size diminishes, reducing statistical power, model robustness, and representativeness of different user behaviors. This is especially problematic in behavioral datasets like ad-click logs, where individual histories carry significant predictive value. A model trained on a reduced dataset might miss important patterns or relationships, leading to biased predictions and a drop in real-world performance. By filling in missing values based on the same user's historical data, we can improve the accuracy of the imputation. This approach is more reflective of the user's true behavior compared to using mean or mode imputation, especially when it comes to ad clicks and personal behavioral traits.
Now, our dataset still contains NA and missing values, which need to be dealt with for unnecessary errors. The code R below counts all cases where values are explicitly NA:

```
# Count occurrences of blanks and NA in each column
na_count <- sapply(data, function(x) sum(is.na(x)))
# Print the count of missing values per column
print(na_count)
```



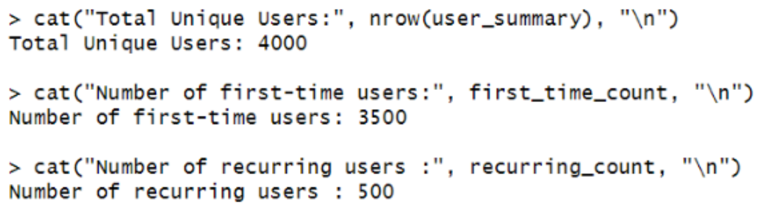
```
             id        full_name              age           gender      device_type
              0                0             4766             4693             2000
    ad_position browsing_history      time_of_day            click
           2000             4782             2000                0
```

**Figure 3:** *Number of NA values in each column*

To better understand the relationship of variables before imputing, we summarize the total records and classify them into first-time and recurring users. It helps identify which users have enough personal history to impute missing values meaningfully, and provides an overview of user relationships.

```
# Summarize each user's total score
user_summary <- cleaned_data %>% group_by(full_name) %>% summarise(total_records = n
())
```

```r
user_summary <- user_summary %>% mutate(user_category = case_when(total_records == 1 ~ "first-time", total_records > 1 ~ "recurring", TRUE ~ "other"))

first_time_count <- sum(user_summary$user_category == "first-time")
recurring_count <- sum(user_summary$user_category == "recurring")
cat("Total Unique Users:", nrow(user_summary), "\n")
cat("Number of first-time users:", first_time_count, "\n")
cat("Number of recurring users :", recurring_count, "\n")

```



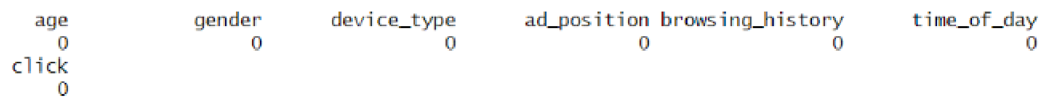
```
> cat("Total Unique Users:", nrow(user_summary), "\n")
Total Unique Users: 4000

> cat("Number of first-time users:", first_time_count, "\n")
Number of first-time users: 3500

> cat("Number of recurring users :", recurring_count, "\n")
Number of recurring users : 500
```

**Figure 4:** *Overview of relationships between users*

The full_name column contains 4,000 unique values, indicating that the dataset of 10,000 records includes recurring users. First-time users consist of 3,500 individuals, each contributing one record, while recurring users include 500 individuals who collectively contributed 6,500 records.

To obtain a clean dataset, the idea is to fill missing values within the context of each user's historical data rather than applying a blanket value across the entire dataset. Grouping by User (group_by (`full_name`)) groups the dataset by each unique user (`full_name`). It ensures that any calculations like median or mode are performed within each user's record cluster, so the imputation respects personal history. For each user, if any age value is missing (is.na(age)), it replaces it with that user's median age (ignoring NAs). This avoids skewing imputations with outliers or irrelevant values from other users. Each categorical column like `gender, device_type, ad_position, browsing_history, and time_of_day` is imputed using the mode (most frequent value) within that user's data, and then we ungroup() to finish. The helper function get_mode handles this by excluding NAs and returning the most frequently occurring value in that user's records. If all values are missing, it returns NA.

```r
#Define a helper function to get the mode of a vector
get_mode <- function(x) {
  x <- x[!is.na(x)]  # Remove NA
  if (length(x) == 0) return(NA)  # Return NA if all values are NA
  uniq_x <- unique(x)
  uniq_x[which.max(tabulate(match(x, uniq_x)))]
}
# Fill NA values in categorical columns with the mode of each user
cleaned_data <- cleaned_data %>%
  group_by(full_name) %>%
  mutate(age = ifelse(is.na(age), median(age, na.rm = TRUE), age), # For numeric column, use the median age of each user
         # For categorical columns, use the per-user mode:
         gender = ifelse(is.na(gender), get_mode(gender), gender),
         device_type = ifelse(is.na(device_type), get_mode(device_type), device_type)
,
         ad_position = ifelse(is.na(ad_position), get_mode(ad_position), ad_position)
,
         browsing_history = ifelse(is.na(browsing_history), get_mode(browsing_history
), browsing_history),
         time_of_day = ifelse(is.na(time_of_day), get_mode(time_of_day), time_of_day)
  ) %>%
  ungroup()
# Fill NA values in age column with -1
cleaned_data$age[is.na(cleaned_data$age)] <- -1
# Fill other NA with unknown
categorical_cols <- c("gender", "device_type", "ad_position", "time_of_day", "
browsing_history")
for (col in categorical_cols) {
  cleaned_data[[col]][is.na(cleaned_data[[col]])] <- "Unknown"
}
# Check for missing values
na_count <- sapply(cleaned_data, function(x) sum(is.na(x)))
print(na_count)
```



```
             age           gender      device_type      ad_position browsing_history      time_of_day 
               0                0                0                0                0                0 
           click 
               0 
```

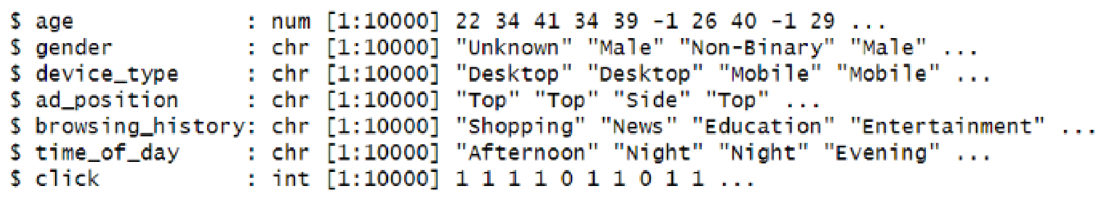
**Figure 5:** *After deleting NA values*

In short, columns ‘‘id’’ and ‘‘full_name’’ remain mostly anonymous through numbers and UserX, therefore, we only use the remaining 7 variables for convenience. First-time users don't have internal historical data to impute missing values. An "Unknown" category and -1 for continuous data like age were introduced to handle missing values if all else fails, preventing over-analyzing the original data while preserving the information carried by missing values. Additionally, it helps mitigate model bias and reduces the risk of overfitting.

### 1.3 Refined data

We can now use the cleaned data to assist further analysis.

```r
# Recheck our data by displaying to the console
str(cleaned_data)
```



```
 $ age             : num [1:10000] 22 34 41 34 39 -1 26 40 -1 29 ...
 $ gender          : chr [1:10000] "Unknown" "Male" "Non-Binary" "Male" ...
 $ device_type     : chr [1:10000] "Desktop" "Desktop" "Mobile" "Mobile" ...
 $ ad_position     : chr [1:10000] "Top" "Top" "Side" "Top" ...
 $ browsing_history: chr [1:10000] "Shopping" "News" "Education" "Entertainment" ...
 $ time_of_day     : chr [1:10000] "Afternoon" "Night" "Night" "Evening" ...
 $ click           : int [1:10000] 1 1 1 1 0 1 1 0 1 1 ...
```

**Figure 6:** *The complete data for use*

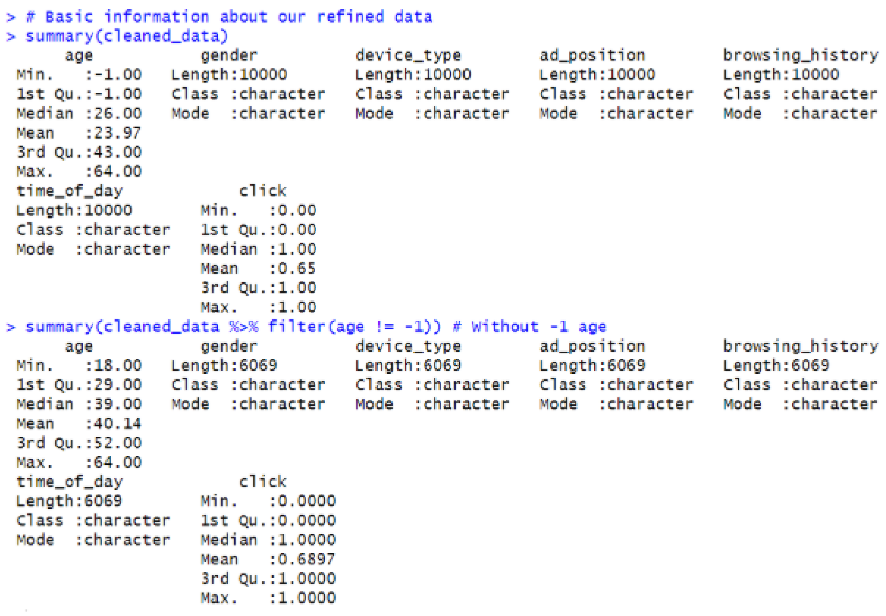
## 2. Descriptive statistics

### 2.1 Data summary

To retrieve the summary log of the data, we use the function summary() in R. Figure 7 displays the summary of 7 variables after the preprocessing step.

```r
# Basic information about our refined data
summary(cleaned_data)
summary(cleaned_data %>% filter(age != -1)) # Without -1 age
```

The dataset consists of both numerical and categorical variables. Among them, "age" is the only continuous variable, while the remaining variables are categorical. Notably, the "click" variable is either binary categorical variable (0 and 1), or numerical variable based on how we are going to deal with the analysis. As shown in Figure 7, the age distribution ranges from 18 to 64 years old, with an average age of approximately 40 years. The median age is 39, indicating that around 50% of the users are younger than 39 years old. This suggests a fairly balanced age distribution, though further analysis will explore whether specific age groups exhibit different ad-clicking behaviors.



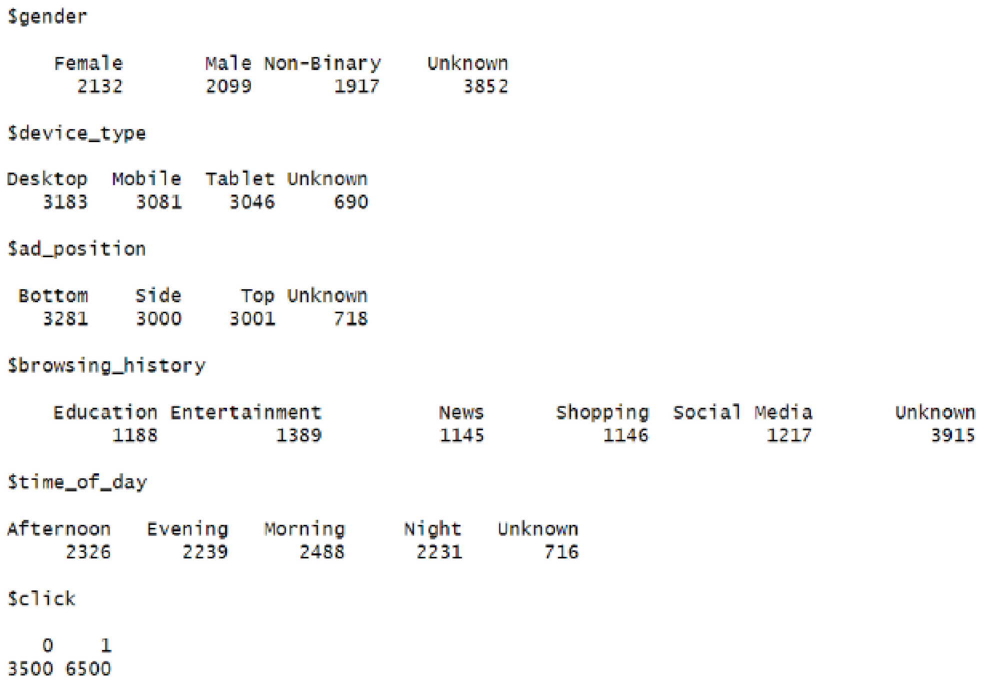
```
> # Basic information about our refined data
> summary(cleaned_data)
      age            gender           device_type         ad_position         browsing_history  
 Min.   :-1.00   Length:10000       Length:10000       Length:10000       Length:10000      
 1st Qu.:-1.00   Class :character   Class :character   Class :character   Class :character  
 Median :26.00   Mode  :character   Mode  :character   Mode  :character   Mode  :character  
 Mean   :23.97                                                                                
 3rd Qu.:43.00                                                                                
 Max.   :64.00                                                                                
 time_of_day            click     
 Length:10000       Min.   :0.00  
 Class :character   1st Qu.:0.00  
 Mode  :character   Median :1.00  
                    Mean   :0.65  
                    3rd Qu.:1.00  
                    Max.   :1.00  
> summary(cleaned_data %>% filter(age != -1)) # Without -1 age
      age            gender           device_type         ad_position         browsing_history  
 Min.   :18.00   Length:6069        Length:6069        Length:6069        Length:6069       
 1st Qu.:29.00   Class :character   Class :character   Class :character   Class :character  
 Median :39.00   Mode  :character   Mode  :character   Mode  :character   Mode  :character  
 Mean   :40.14                                                                                
 3rd Qu.:52.00                                                                                
 Max.   :64.00                                                                                
 time_of_day            click       
 Length:6069        Min.   :0.0000  
 Class :character   1st Qu.:0.0000  
 Mode  :character   Median :1.0000  
                    Mean   :0.6897  
                    3rd Qu.:1.0000  
                    Max.   :1.0000  
```

**Figure 7:** *Summary of 7 variables*

## 2.2 Frequency Table

For the remaining categorical variables, it is better to analyze data using frequency table to count the number of occurrences of each variable. We use the function apply() to create frequency tables for each variable in the dataset, except for "age". The result is presented in Figure 8.

```r
# Frequency Table for categorical variables:
apply(cleaned_data[c("gender","device_type","ad_position","browsing_history","time_of_day","click")],2,table)

```



```
$gender

    Female        Male Non-Binary    Unknown 
      2132        2099       1917       3852 

$device_type

Desktop  Mobile  Tablet Unknown 
   3183    3081    3046     690 

$ad_position

 Bottom    Side     Top Unknown 
   3281    3000    3001     718 

$browsing_history

    Education Entertainment          News      Shopping  Social Media       Unknown 
         1188          1389          1145          1146          1217          3915 

$time_of_day

Afternoon   Evening   Morning     Night   Unknown 
     2326      2239      2488      2231       716 

$click

   0    1 
3500 6500 
```

**Figure 8:** *Frequency table of each categorical variable*

While most variables have acceptable completeness post-imputation, gender and browsing history retain significant unknown values. The statistical table provides a valuable insight into the factors impacting the click rate of users and serves as a great foundation for more advanced analysis.

### 2.3 Histogram chart

To gain clearer insights into the relationships and distribution within the ad_click_prediction dataset, data visualization plays a crucial role. Tools like bar charts and pie charts can effectively highlight key trends and features that may not be immediately apparent in statistical tables. Similarly, histograms are particularly effective for examining the distribution of numerical variables, such as age. A histogram of the **‘‘age’’** variable can help us identify specific patterns in user engagement across different age groups, such as peaks or drops in click activity. By visualizing the data, we can present the findings in a way that is both intuitive and easy to understand.
To start with, we create a histogram for the **‘‘age’’** variable using the ggplot() and geom_histogram() function from the ggplot() library. Figure 8 illustrates a basic histogram of age, generated in R.

```
    #Basic histogram of "age"
    ggplot(cleaned_data %>% filter(age != -1), aes(x = age)) +
      geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
      labs(title = "Histogram of Age", x = "Age", y = "Count") +
      theme_minimal() +
      theme(plot.title = element_text(hjust = 0.5))

```

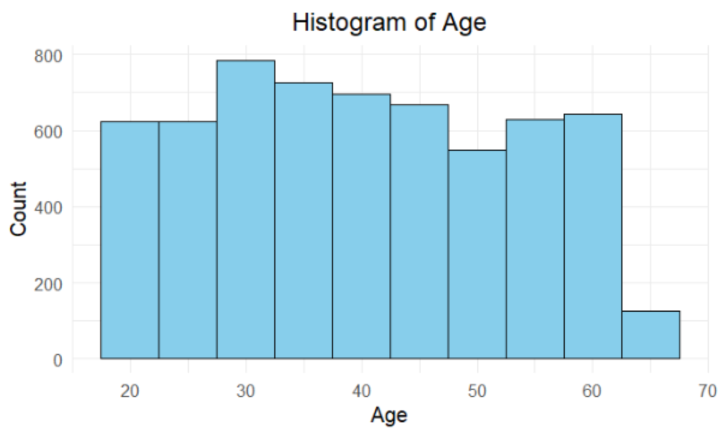


**Figure 9:** *Basic histogram of age*

The distribution spans from around 18 to 64 years. This histogram displays a relatively even spread, with each age interval bar hovering between 550 and 800 counts. Notably, there is a clear peak around age 30-35, where the count approaches 800, with a gradual decline after the peak, suggesting that the middle-aged group dominates the dataset. The histogram also shows a significant drop in the oldest age group (60-70), where counts fall to around 100, significantly lower than all other age groups.
For a more advanced analysis, such as the distribution of **‘‘age’’** in the tendency of users to click or not, we plot the histogram to compare the two categories without age = -1 for convenience. The chart is created using the code below:

```
    # Distribution of age by click
    ggplot(cleaned_data %>% filter(age != -1), aes(x = age, fill = factor(click))) +
        geom_histogram(position="identity", alpha=0.5, binwidth = 5, color = "black") +
        scale_fill_manual(values = c("red", "skyblue"), name = "Click", labels = c("No
    Click", "Click")) +
        labs(title = "Histogram of Age by Click Behavior", x = "Age", y = "Count") +
        theme_minimal() +
        theme(plot.title = element_text(hjust = 0.5))

```

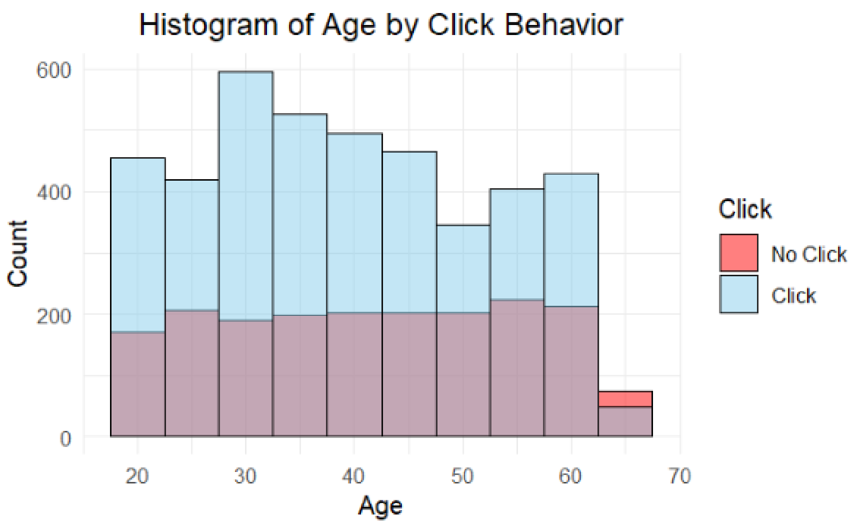


**Figure 10:** *Distribution of age by click*

The histogram of click behavior by age actually reveals a fairly even distribution of click and no-click counts across most age bins from 20 to about 60. While the highest count appears in the 30-35 age bin, this seems consistent with a natural population distribution. From age 40 to around 60, the proportion of clicks remains fairly balanced relative to non-clicks, even though the total number of users begins to decrease. However, beyond age 60, there is a clear decline in click activity. In the 60–70 age range, the No Click class outweighs the blue Click class, both in absolute numbers and as a proportion of total users in those bins. This marks a shift in behavior, with older users increasingly disengaged and less likely to interact. This suggests that age 30-40 dominates the dataset, and genuine engagement patterns reveal higher responsiveness among younger users and growing disengagement with age among older users.

## 2.4 Barplot and pie chart

We plot bar and pie charts to demonstrate information about categorical variables, as shown in Figures 10 and 11.

```r
# Barplot & set up grid for more plots
par(mfrow = c(3, 2), mar = c(2, 2, 2, 2) + 0.1) # Reduced margins
categorical<-c("gender","device_type","ad_position","browsing_history","time_of_day",
"click")
for (var in categorical){
  barplot(table(cleaned_data[[var]]),
          main = paste("Barplot of", var),
          col = "skyblue",
          border = "black",
          cex.names=0.6)
}
# reset plotting layout to default
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)
#Pie chart & set up grid for more plots
par(mfrow = c(3, 2), mar = c(1, 2, 2, 2) + 0.1) # Reduced margins
for (var in categorical){
  counts <- table(cleaned_data[[var]])
  percents <- round(100 * counts / sum(counts), 1)  # Calculate percentages
  labels <- paste0(names(counts), "\n", percents, "%")  # Create labels with percent
  pie(counts,
      labels=labels,
      main = paste("Pie chart of", var),
      col = "skyblue", border = "black",
      radius=1,
      cex=0.7)
}
# reset plotting layout to default
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)

```

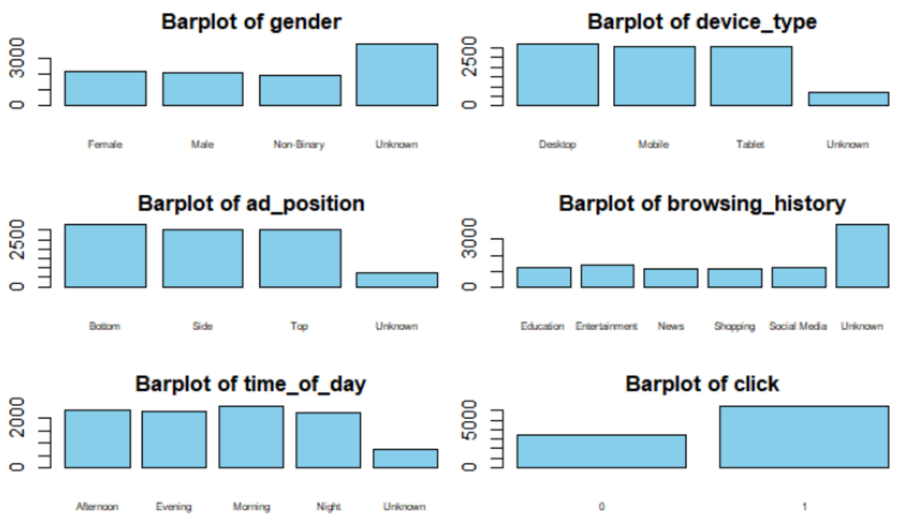


**Figure 11:** *Barplots of 6 categorical variables*

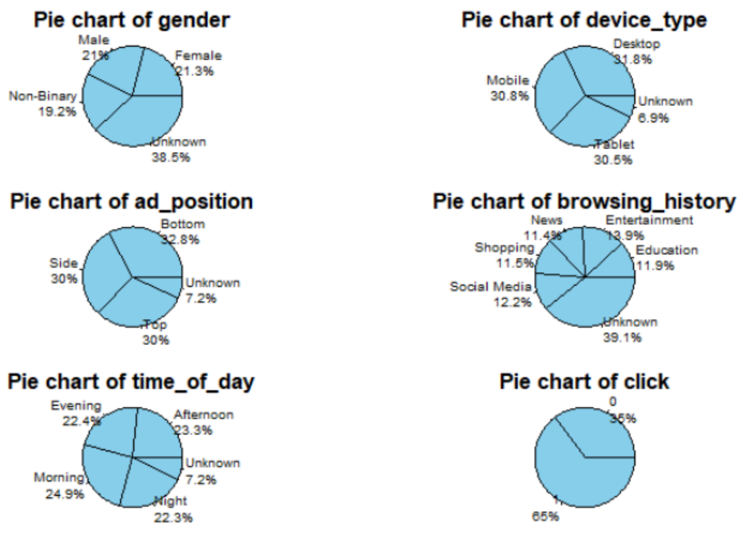


**Figure 12:** *Pie charts of 6 categorical variables*

- **gender:** Female, Male and Non-Binary are somewhat balanced, though Non-Binary is slightly lower and Unknown is the highest.
- **device_type:** Desktop and Mobile are close, with Tablet a bit lower. Unknown contributes relatively minor.
- **ad_position:** Bottom, Side, and Top are evenly spread, while Unknown is trivial.
- **browsing_history:** The five categories (Education, Entertainment, News, Shopping, Social Media) are similarly distributed, though Unknown remains the highest at 39.1%.
- **time_of_day:** Afternoon, Evening, Morning, and Night have close frequencies, with Morning ranked at the top.
- **click:** Binary outcome (1/0) with imbalance, with click roughly doubling non-click.

For a more advanced approach, 3 or more variables are used at the same time to plot a bar chart. The filter() function is used to omit the Unknown columns in `device_type` and `ad_position` for clearer insight. The tapply() function applies a function to subsets of a vector based on factors (groupings) using mean to calculate click rate, and the barplot() function helps to draw the bar chart with x-axis as `device_type` and y-axis as click rate. Figure 12 shows a bar plot using 3 variables concurrently.

```r
# Filter out rows where device_type or ad_position is 'Unknown'
filtered_data <- cleaned_data %>%
  filter(device_type != "Unknown", ad_position != "Unknown")

# Create summary table: click rate by device type and ad position (excluding 'Unknown')
click_table <- tapply(filtered_data$click,
                      list(filtered_data$ad_position, filtered_data$device_type),
                      mean) #Calculates the mean of the click variable grouped by ad_position and device_type

# Print the click rate table
print(click_table)

# Create barplot
barplot(click_table,
        beside = TRUE,
        col = c("skyblue", "lavender", "lightcoral"),  # Adjust number of colors based on number of ad positions
        ylim = c(0, 1),
        main = "Click Rate by Device Type and Ad Position (without Unknown)",
        xlab = "Device Type",
        ylab = "Click Rate",
        cex.main = 0.85,
        cex.lab = 0.8,  # Size of axis labels (x and y labels)
        cex.axis = 0.8,   # Size of axis tick labels
        legend.text = rownames(click_table), # Adds a legend using the row names
        args.legend = list(x = 13.5, y = 1.15, cex = 0.6)) # Specifies the legend position and size

```

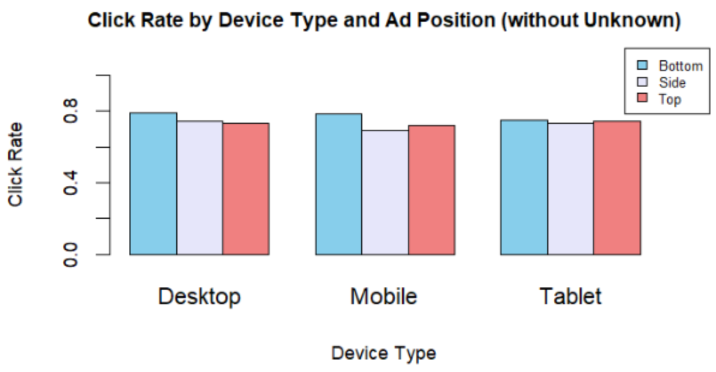


**Figure 13:** *Bar plot of 3 variables*

Mobile has the highest click rates across all ad positions, making it the most engaging platform for ads, while Tablet has relatively lower click rates compared to Mobile and Desktop. The bar plot also indicates that ads placed at Top and Bottom of the page tend to have higher click rates, approaching 0.8, especially for Mobile and Desktop. The Side position shows moderate click rates at around 0.5, suggesting it's less engaging than the others but still significant.

## 2.5 Correlation matrix

Correlation matrix measures the strength and direction of relationships between numerical variables. Since correlation is based on numerical computations, it is only viable for numeric columns. If click is treated as numeric (0/1), we can compute the correlation between "age" and "click", which would indicate whether older users tend to click more or less.

Using the above information, we plot a correlation matrix for a better understanding of relationships between variables, as shown in Figure 13.

```r
# Correlation matrix visualization
correlation_matrix <- cor(cleaned_data[,sapply(cleaned_data, is.numeric)], use = "
complete.obs")
corrplot(correlation_matrix, method = "square", #square tiles
         addCoef.col = "red",                   #coefficient color
         tl.col = "black",                      #text label color
         number.digits = 4)                     #ensure enough space

```

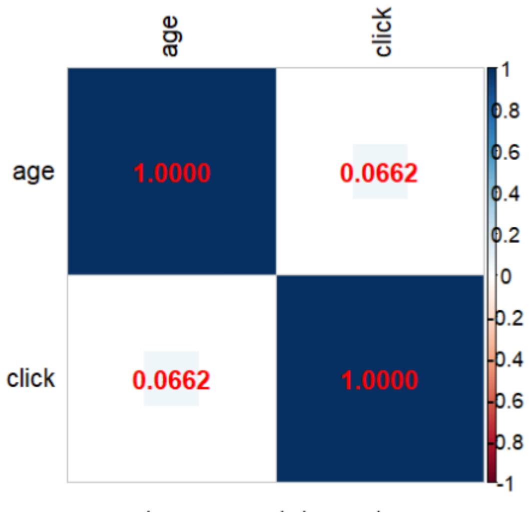


**Figure 14:** *Correlation matrix*

The correlation coefficient between age and click is -0.0662, and the deep blue color represents strong positive correlations, but in this case, the near-white area between "age" and "click" suggests almost no correlation. This means that older and younger users click on ads at approximately the same rate, with no clear trend suggesting that one age group clicks significantly more than the other.
This divergence suggests that other factors might influence click activity, and age alone may not be a decisive predictor. Further analysis can provide deeper insights into how age interacts with other variables to impact click behavior.

## 2.6 Box plot

Box plots are a powerful tool for visualizing the distribution and variability of a numerical variable across different categorical groups. They summarize key statistics, and in this context, creating box plots for age by variables like device_type, time_of_day,... helps to compare distributions of age across categorical groups (e.g., different device_type), and spot trends, variations, or outliers that can be meaningful for analysis.
In R, we can use a for loop to run across categorical variables and put each of them in the function boxplot() to draw multiple box plots at once (given that the window size is adjusted). Because -1 is a placeholder, not a real age, and including it distorts both the visualization and the summary stats like medians and interquartile ranges in yo_oxplots. The presence of -1 artificially lowers the lower whisker and can produce misleading outliers or skewed boxes.

```r
# Box plots and Set layout to display multiple plots
par(mfrow = c(3, 2), mar = c(2, 2, 2, 0) + 0.1) # Reduced margins
categorical_cols<-c("gender","device_type","ad_position","browsing_history","time_of_
day","click")
# Loop through each variable and create box plot
for (var in categorical) {
  formula <- as.formula(paste("age ~", var))
  # Filter out -1 ages and 'Unknown' for clean plots
  filtered_data <- cleaned_data %>% filter(age != -1, !!as.name(var) != "Unknown")
  par(las = 1)  # Adjust axis label size
```

```
      box_stats <- boxplot(formula, data = filtered_data,
                           col = c("skyblue", "lightgreen", "plum", "lightcoral", "gold"), #set colors
                           border = "darkblue",
                           main = paste("Age by", var),
                           xlab = var,
                           xaxt = "n",  # Suppress default x-axis
                           cex.axis = 0.75,  # Size for y-axis tick labels
                           ylab = "Age")
      # Add custom x-axis
      axis(1, at = seq_along(unique(filtered_data[[var]])),
           labels = unique(filtered_data[[var]]),
           cex.axis = 0.58  # Smaller size for x-axis tick labels
           )
    }
    # reset plotting layout to default
    par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1, cex.axis = 1, las = 0)
```

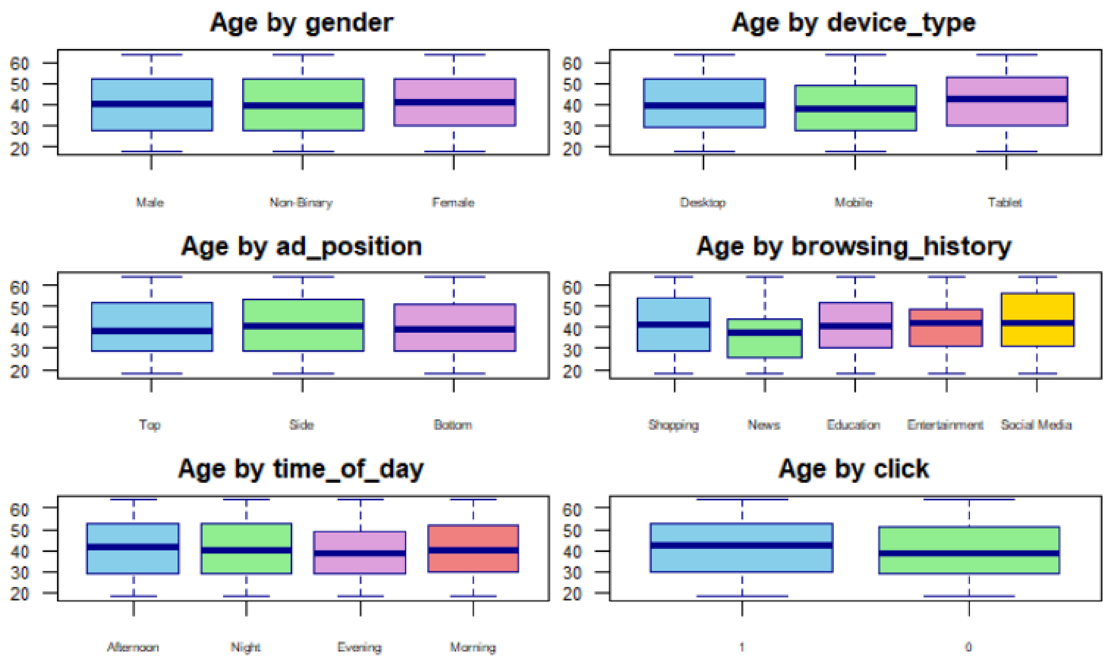


**Figure 15:** *Box plots of variables*

**Age by Gender:** The median age is consistent across genders (Female, Male, and Non-Binary), around the mid-40s, though Female and Male are minorly left-skewed (more younger users). There does not seem to be a noticeable trend or significant variability in the age distribution among the three categories.

**Age by Device Type:** The age distribution is similar for Desktop, Mobile, and Tablet users, with median ages around the mid-40s. For Desktop and Mobile users, the means appear slightly higher than the medians, hinting at a minor right-skew distribution.

**Age by Ad Position:** All three positions, however, have slightly higher means than the medians, indicating a mild right skew.

**Age by Browsing History:** News and Entertainment show a minor left skew with medians higher than their means, while the rest are right-skewed.

**Age by Time of Day:** Evening and Night show higher means than the medians, indicating a notable right skew, perhaps older users are more active in the evening and night. Afternoon and Morning are almost balanced.

**Age by Click:** The no-click shows a mildly right-skewed distribution as the mean is higher than the median. However, the difference is minimal in the click group, although it is a left-skewed distribution.

To add more details and values to our box plots, we add red dots as the means to compare with the medians by points() and text() functions, then put the code below inside the loop.

```r
# Calculate means by group
group_means <- tapply(filtered_data$age, filtered_data[[var]], mean)
# Overlay means (red points)
points(1:length(group_means), group_means, col = "red", pch = 19)  # Red dots for means

# Add text for mean and median values
text(1:length(group_means), group_means + 7,  # Adjust vertical position for clarity
labels = round(group_means, 1), col = "red", cex = 0.8)  # Text for means
text(1:length(box_stats$names), box_stats$stats[3, ] - 2,  # Adjust vertical position for clarity
labels = round(box_stats$stats[3, ], 1), col = "black", cex = 0.7)  # Text for medians

```

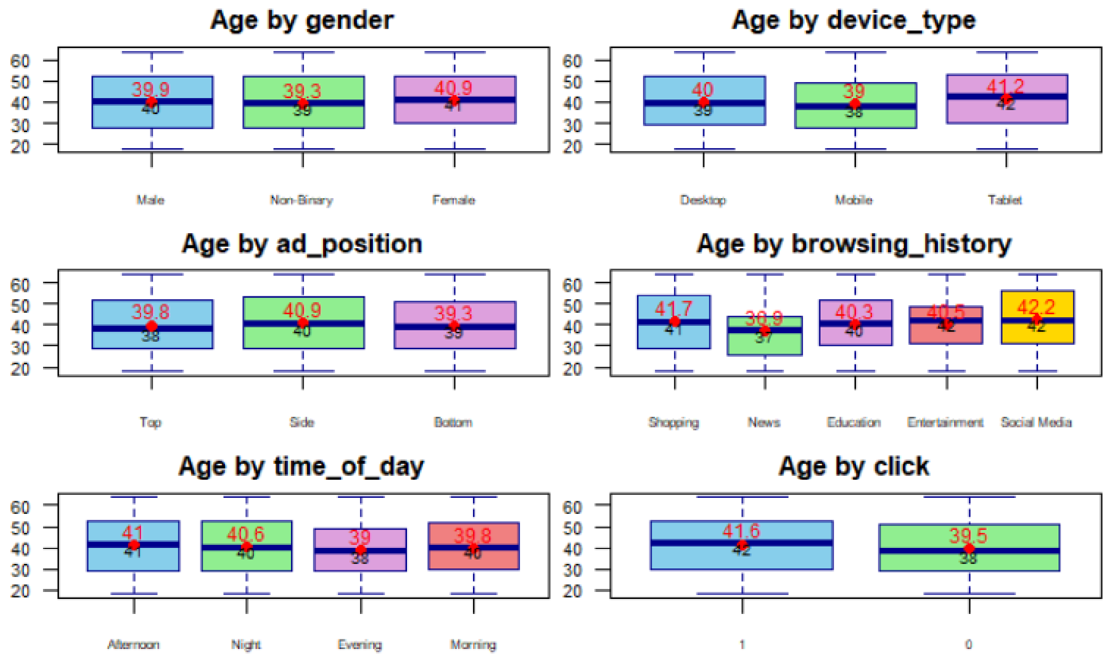


**Figure 16:** *Detailed Box plots of variables with values*

## VI Predictive Modelling

### 1. Binary Logistic Regression model

#### 1.1 Overview

Logistic regression is a statistical method used to predict the likelihood of an event occurring, particularly when the dependent variable is binary (click and no click). It is commonly applied in digital marketing to estimate the probability of user engagement (clicking an ad) based on multiple independent factors such as age, device type, browsing history,. . . The goal of this analysis is to develop a predictive model that can accurately estimate the tendency of users to click or not based on independent variables. By doing so, it offers valuable insights to guide decision-making in areas such as marketing strategies, ad placement, and content optimization.

In comparison to multiple linear regression, logistic regression is more suitable for cases where the dependent variable is binary or categorical. Multiple linear regression is designed for predicting continuous numerical outcomes (e.g., revenue, temperature, or height), but in this dataset, the target variable "click" is binary (0 or 1). If multiple linear regression were used, predicted values might fall outside the range of 0-1, which does not align with the nature of the problem. Logistic regression, on the other hand, constrains predictions within the probability range of 0 to 1, making it more accurate for modeling the likelihood of a user clicking on an ad.
In predictive modeling, especially for supervised learning tasks like logistic regression, the goal is to assess a model's ability to generalize to unseen data. We split the dataset into a training set (typically 80%) and a test set (typically 20%). This 80/20 split, while not mandatory, represents a common balance and is largely accepted as the standard. In R, the function set.seed(100) ensures reproducibility. When sampling data randomly, different runs of the script without a fixed seed will produce different splits. By setting the seed, we guarantee that the same random sampling occurs every time the script runs, allowing for consistent model evaluation, debugging, and reporting.

```r
# ----- Predictive modelling begins -----
# Set up training and testing data:
set.seed(100) # Set seed for reproducibility
# Randomly sample 80% of rows from data for training
train.rows <- sample(rownames(cleaned_data), dim(cleaned_data)[1] * 0.8)
train_data <- cleaned_data[train.rows, ] # Create training set from chosen data

# Use remaining 20% as testing data
test.rows <- setdiff(rownames(cleaned_data), train.rows)
test_data <- cleaned_data[test.rows, ] #Create test set from chosen data

```

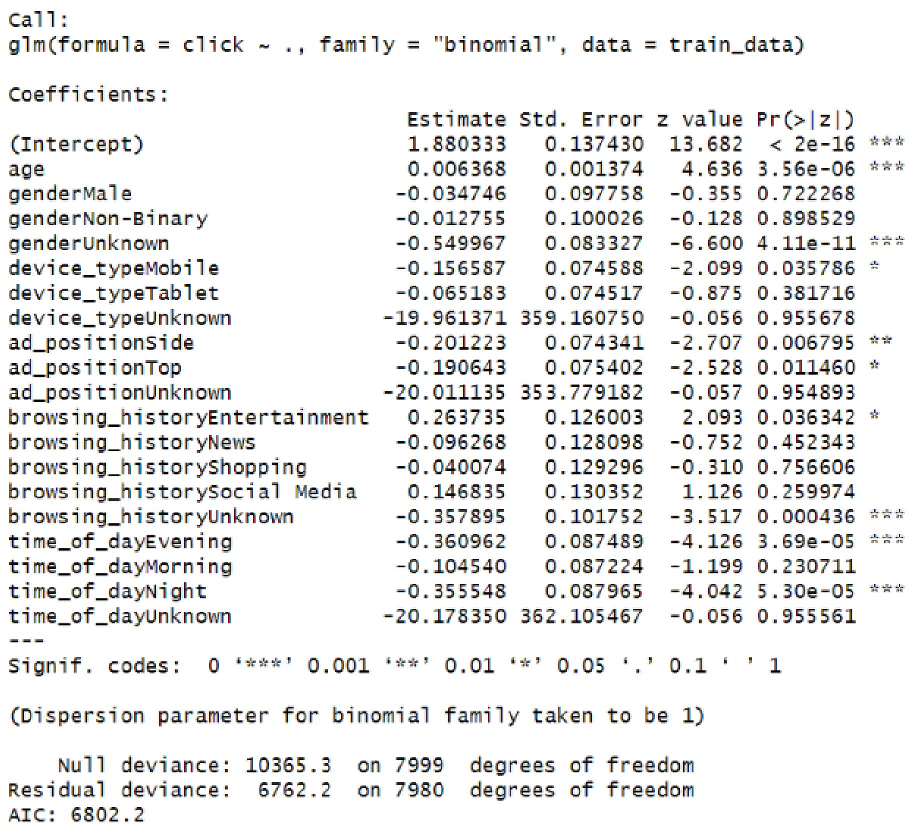
Set Y = ad_click, and p is the probability of users clicking on ads, p = P (Y=1) The model is expressed as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 \text{ age} + \beta_2 \text{ gender} + \beta_3 \text{ device type} + \\ \beta_4 \text{ ad position} + \beta_5 \text{ browsing history} + \beta_6 \text{ time of day} + e$$

### 1.2 Result

We perform the estimation of values by using glm() with family=”binomial” to indicate the logistic model. Furthermore, summary() function shows the results of the model, as in Figure 16.

```r
# 1.Logistic regression model begins
# Fit logistic regression model using all predictors to predict 'click'
# and all other variables as independent variables in the training data. Model follows a binomial distribution.
RM_model <- glm(click ~ ., family = "binomial", data = train_data)

# View detailed information about the trained logistic regression model, including parameters, p-values, and other statistics
summary(RM_model)

```



```
Call:
glm(formula = click ~ ., family = "binomial", data = train_data)

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)    
(Intercept)                     1.880333   0.137430  13.682  < 2e-16 ***
age                             0.006368   0.001374   4.636 3.56e-06 ***
genderMale                     -0.034746   0.097758  -0.355 0.722268    
genderNon-Binary               -0.012755   0.100026  -0.128 0.898529    
genderUnknown                  -0.549967   0.083327  -6.600 4.11e-11 ***
device_typeMobile              -0.156587   0.074588  -2.099 0.035786 *  
device_typeTablet              -0.065183   0.074517  -0.875 0.381716    
device_typeUnknown            -19.961371 359.160750  -0.056 0.955678    
ad_positionSide                -0.201223   0.074341  -2.707 0.006795 ** 
ad_positionTop                 -0.190643   0.075402  -2.528 0.011460 *  
ad_positionUnknown            -20.011135 353.779182  -0.057 0.954893    
browsing_historyEntertainment   0.263735   0.126003   2.093 0.036342 *  
browsing_historyNews           -0.096268   0.128098  -0.752 0.452343    
browsing_historyShopping       -0.040074   0.129296  -0.310 0.756606    
browsing_historySocial Media    0.146835   0.130352   1.126 0.259974    
browsing_historyUnknown        -0.357895   0.101752  -3.517 0.000436 ***
time_of_dayEvening             -0.360962   0.087489  -4.126 3.69e-05 ***
time_of_dayMorning             -0.104540   0.087224  -1.199 0.230711    
time_of_dayNight               -0.355548   0.087965  -4.042 5.30e-05 ***
time_of_dayUnknown            -20.178350 362.105467  -0.056 0.955561    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 10365.3  on 7999  degrees of freedom
Residual deviance:  6762.2  on 7980  degrees of freedom
AIC: 6802.2
```

**Figure 17:** *Summary of the logistic regression model*

From the result, we build the logistic regression equation:

$$\log\left(\frac{p}{1-p}\right) = 1.880333 + 0.0636888 \times \text{age} - 0.034746 \times \text{genderMale} -$$
$$0.012755 \times \text{genderNon} - \text{Binary} + \cdots - 20.178350 \times \text{time\_of\_dayUnknown}$$

Hypothesis testing is performed on $\beta_i$ with the hypotheses:
**Null hypothesis** $H_0$**:** $\beta_i = 0$
**Alternative hypothesis:** $H_1 : \beta_i \neq 0$

Based on the p-value (Pr(>|z|) corresponding to age (2e-16), device_typeMobile (0.035786), ad_positionSide(0.006795) and ad_positonTop(0.011460), browsing_historyEntertainment(0.036342), time_of_dayEvening(3.69e-05), time_of_dayNight(5.3e-05), all values are below the significance level 0.05, thus rejecting the null hypothesis $H_0$. Since $\beta_i \neq 0$, we conclude that the significant predictors are age, device_typeMobile, ad_positionSide and Top, browsing_historyEntertainment, time_of_dayEvening and Night. For convenience, Unknown predictors are not considered in the analysis.

Regarding the other variables, as the p-values are greater than 0.05, we cannot reject the null hypothesis $H_0$. Therefore, these regression coefficients are equal to 0, or these variables are not significant in predicting whether users will click or not.

Besides, the summary gives information about:

- Deviance Residuals: measure the difference between observed values (actual click or no click) and predicted values from the model.
- Null deviance: deviance of the null model (the model based on mean values to predict without predictors).

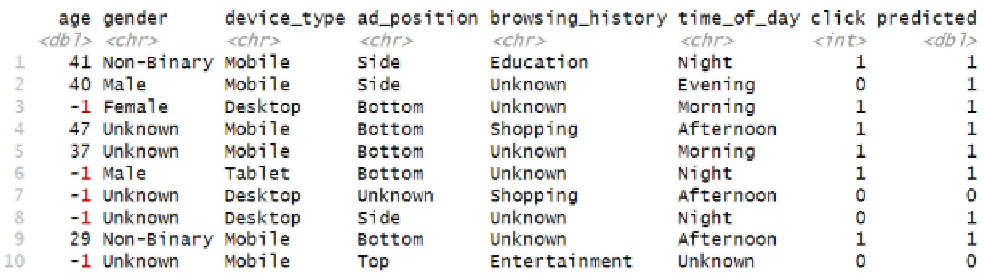
- Residual deviance: deviance of the logistic regression model when predictors are included.
- AIC: is a metric used to evaluate models. It balances goodness of fit and model complexity. A lower AIC value typically suggests a better model, as it indicates a balance between fitting the data well and limiting the number of parameters. While there's no fixed threshold for AIC, it decreases as the model improves, for instance, by adding significant variables or refining the model structure.

In conclusion, the lower value of Residual deviance as compared to the Null deviance suggests that the logistic regression model is improved with added predictors to capture meaningful patterns related to whether users click or not.

## 1.3 Predict

After building the model, we use the function predict() with type="response" to return the probability of the "click = 1", use the function ifelse to round the probability to 0 or 1, expressing whether users click or not.

```r
# 2. Predict "click" probabilities for test data
# Use the trained model to predict the probability of "click" for observations in test_data
# type="response" returns the probability of the "click = 1" class.
predicted <- predict(RM_model, test_data, type = "response")

# Round predicted probabilities to 0 or 1
# If probability >= 0.5, predict "click" (1), otherwise predict "no click" (0).
test_data$predicted <- ifelse(predicted >= 0.5, 1, 0)

# Display the first 10 rows of test data, including the "predicted" column
head(test_data, 10)

```



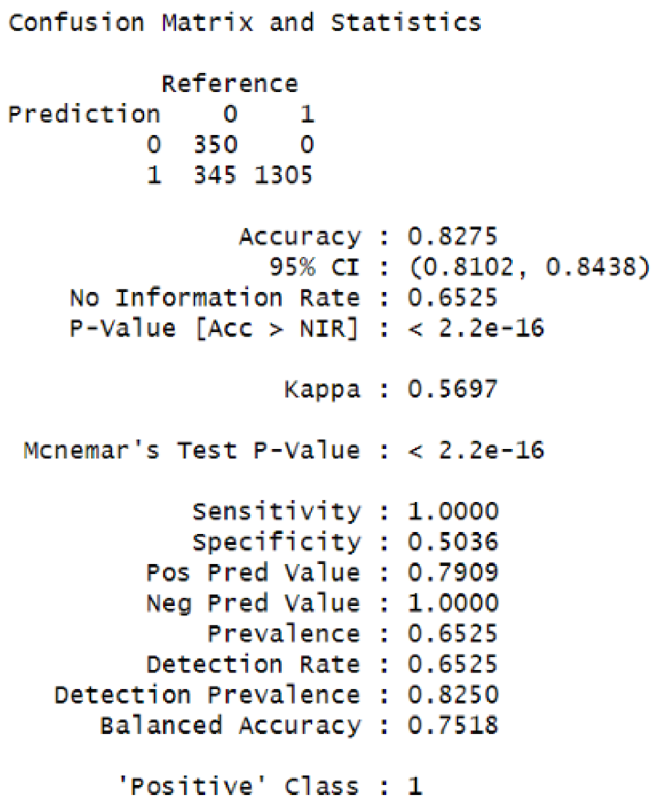
```
     age gender     device_type ad_position browsing_history time_of_day click predicted
   <dbl> <chr>      <chr>       <chr>       <chr>            <chr>       <int>     <dbl>
 1    41 Non-Binary Mobile      Side        Education        Night           1         1
 2    40 Male       Mobile      Side        Unknown          Evening         0         1
 3    -1 Female     Desktop     Bottom      Unknown          Morning         1         1
 4    47 Unknown    Mobile      Bottom      Shopping         Afternoon       1         1
 5    37 Unknown    Mobile      Bottom      Unknown          Morning         1         1
 6    -1 Male       Tablet      Bottom      Unknown          Night           1         1
 7    -1 Unknown    Desktop     Unknown     Shopping         Afternoon       0         0
 8    -1 Unknown    Desktop     Side        Unknown          Night           0         1
 9    29 Non-Binary Mobile      Bottom      Unknown          Afternoon       1         1
10    -1 Unknown    Mobile      Top         Entertainment    Unknown         0         0
```

**Figure 18:** *The first 10 rows with predicted values*

With the help of confusionMatrix() from the library(caret), we build a confusion matrix and calculate the model's performance.

```r
# 3. Evaluate the model with a confusion matrix
# Calculate and display the confusion matrix for the model
# positive = '1' indicates that the positive class is "click = 1"
confusionMatrix(as.factor(test_data$predicted), as.factor(test_data$click), positive
= "1")

```

The results indicate that the model's accuracy is relatively high at 0.8275. Among 2000 samples in test_data, 350 cases are True Negatives (TN), while 0 case is False Negatives (predicted no click, but a click actually occurred), suggesting the model correctly predicted no click. There are 345 False Positives (Predicted click, but no click actually occurred), which slightly affect precision. However, the model is still excellent at predicting actual clicks (True Positive = 1305), showing its strength in identifying clicks correctly. Additionally, sensitivity measures the proportion of actual positive cases that are correctly predicted by the model, so perfect Sensitivity = 1.0 suggests that the model captures all actual clicks. Specificity measures the proportion of actual negative cases that are correctly predicted by the model. Specificity = 0.5036 is moderate, indicating that the model struggles to avoid misclassifying non-clicks as clicks.



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  350    0
         1  345 1305

               Accuracy : 0.8275
                 95% CI : (0.8102, 0.8438)
    No Information Rate : 0.6525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5697

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.5036
         Pos Pred Value : 0.7909
         Neg Pred Value : 1.0000
             Prevalence : 0.6525
         Detection Rate : 0.6525
   Detection Prevalence : 0.8250
      Balanced Accuracy : 0.7518

       'Positive' Class : 1
```

**Figure 19:** *The confusion matrix and performance*

Next, we plot ROC curve - AUC value to evaluate the model accurately. The result is presented in Figure 20 and 21.

```r
# 4. Plot ROC curve and calculate AUC
ROCRpred <- prediction(as.numeric(test_data$predicted), test_data$click)
ROCRperf <- performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf, main = "Logistic Regression ROC Curve")
# Calculate the area under the ROC curve (AUC)
auc <- as.numeric(performance(ROCRpred, "auc")@y.values)
cat("AUC:", auc, "\n")

```

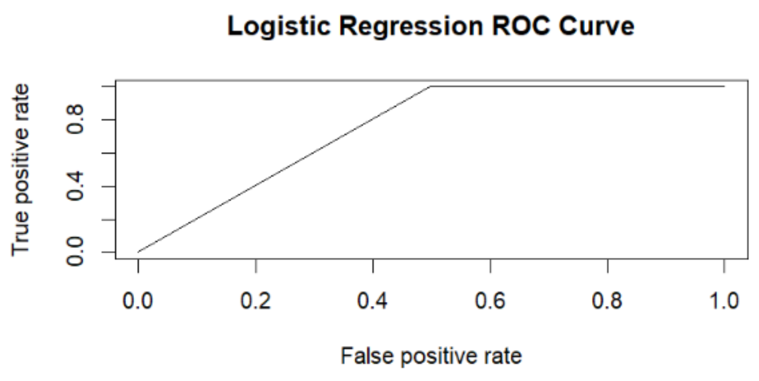
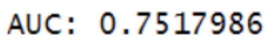


**Figure 20:** *ROC curve*



```
AUC: 0.7517986
```

**Figure 21:** *AUC value*

From the ROC curve and AUC value - the area below the curve (0.7517986), the logistic regression model is performing reasonably well, showing that it captures approximately 75% of the separability between clicks and non-clicks effectively.

### 1.4 Model conditions

It is important to check the conditions of the logistic regression model to ensure its validity and reliability for predictions. Logistic regression relies on several assumptions that should be tested during the modeling process:

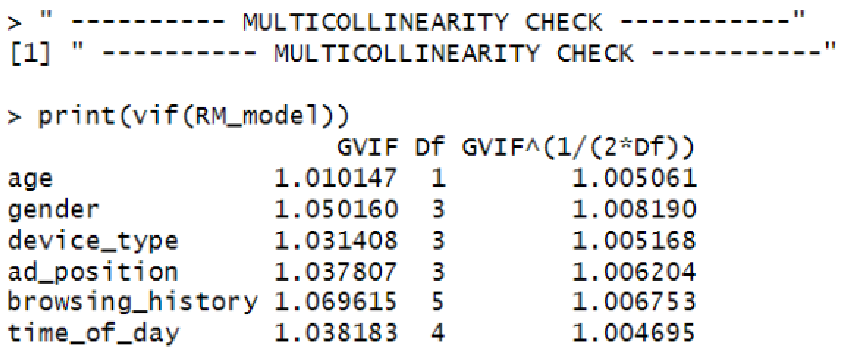
- Dependent variable is binary categorical.
- Linear relationship between the predictors (independent variables) and the log-odds of the outcome is satisfied.
- Absence of Multicollinearity.
- Independence of errors.
- Sample size is large enough.
- Logistic regression uses the logistic function (logit link) to model the probability of an event occurring.
- Data must be processed for missing values and should not have severe outliers.

We perform Multicollinearity check using VIF - statistical measure used to detect multicollinearity in regression models in library(car).

- VIF value of 1 indicates no multicollinearity.
- VIF value greater than 10 often suggests high multicollinearity.

```
" ---------- MULTICOLLINEARITY CHECK -----------"
print(vif(RM_model))

```

The figure shows how much the variance is inflated due to the correlation between predictor variables. VIF values of 1 indicate no multicollinearity.



```
> " ---------- MULTICOLLINEARITY CHECK -----------"
[1] " ---------- MULTICOLLINEARITY CHECK -----------"

> print(vif(RM_model))
                     GVIF Df GVIF^(1/(2*Df))
age              1.010147  1        1.005061
gender           1.050160  3        1.008190
device_type      1.031408  3        1.005168
ad_position      1.037807  3        1.006204
browsing_history 1.069615  5        1.006753
time_of_day      1.038183  4        1.004695
```

**Figure 22:** *VIF result*

In addition, we test severe outliers using Leverage plots ols_plot_resid_lev() in library(olsrr).

```
# ---------- LEVERAGE and OUTLIERS PLOTS -----------"
ols_plot_resid_lev(RM_model)

```

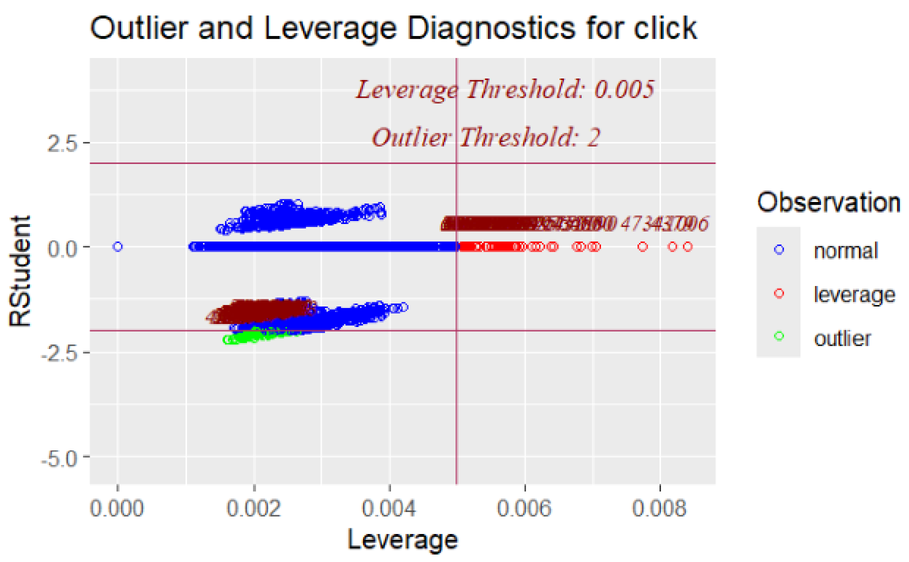
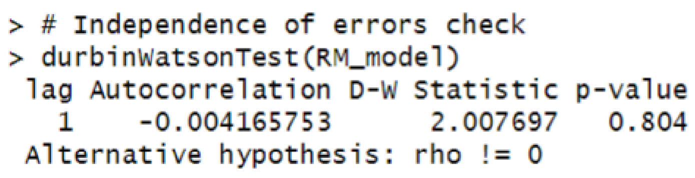


**Figure 23:** *Leverage and outliers plot*

The plot includes a vertical leverage threshold at 0.005. Points exceeding this threshold would be considered high-leverage observations that disproportionately affect the model. While a few points are identified as high leverage (red circles), they do not cross the horizontal outlier threshold (=2). This indicates they may influence the model but are not extreme enough to distort its predictions significantly. The absence of severe outliers is, therefore, satisfied.

Another important condition is independence of errors in a logistic regression model, as it ensures that the residuals (errors) do not exhibit patterns or dependencies, which might bias the model's predictions. Use the Durbin-Watson test to check for autocorrelation, the function is from library(lmtest).

```
#Indepence of errors check
durbinWatsonTest(RM_model)

```



```
> # Independence of errors check
> durbinWatsonTest(RM_model)
 lag Autocorrelation D-W Statistic p-value
   1    -0.004165753      2.007697   0.804
 Alternative hypothesis: rho != 0
```

**Figure 24:** *Durbin-Watson test*

The Durbin Watson test looks for a specific type of serial correlation i.e. first order correlation (the lag is 1 unit). The Hypotheses for the Durbin Watson test are:

- $H_0$ = first order autocorrelation does not exist.
- $H_1$ = first order correlation exists.

DW varies in the range 0-4, as below:

- DW = 2: No autocorrelation. This is the ideal value in a regression model.
- 0 < DW < 1: Positive autocorrelation, where residuals are similar.
- DW > 2: Negative autocorrelation, indicating that residuals tend to be opposite.
- 1.5 < DW < 2.5: This range is considered normal and typically implies no significant autocorrelation

Since our DW = 2.007573 in the range $1.5 < \text{DW} < 2.5$ indicates no significant autocorrelation, we do not reject $H_0$. This suggests that the logistic regression model satisfies the independence of errors.

Overall, the vif() check shows no alarming multicollinearity, the absence of severe outliers is not influencing the model's distortion and performance, and the Durbin-Watson test indicates acceptable residual independence, suggesting no immediate model instability from including age = -1.

## 2. Random Forest model for comparison

### 2.1 Overview

To assess the performance of the logistic regression model in predicting the outcome variable, a Random Forest model was also developed as a comparative classifier. Random Forest is an ensemble learning method that constructs multiple decision trees during training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The Random Forest model was fitted using the same training and test datasets, with the predictor variables identical to those used in the logistic regression model. The model was built with number of trees (ntree) = 100.

The model's performance was evaluated on the test dataset using key metrics:

- Accuracy (Recall)
- Specificity
- Area Under the ROC Curve (AUC)

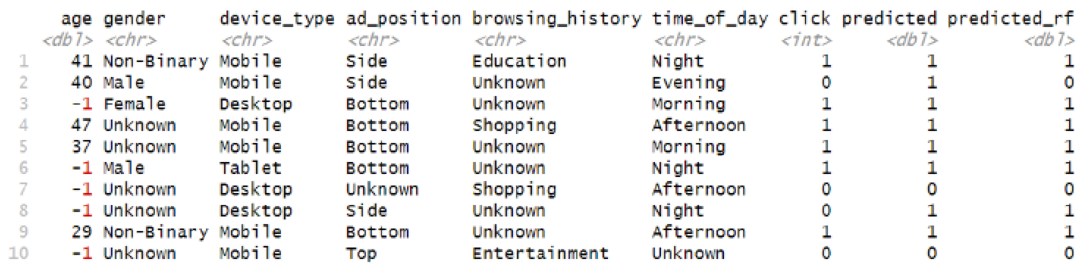
These metrics provide a comprehensive view of the model's predictive ability.

### 2.2 Predict

Random Forest model is constructed using the randomForest package in R. The outcome variable click is first converted to a factor to ensure it's treated as a categorical response variable for classification. The Random Forest model is then fitted using all predictor variables in the train_data dataset, with 100 trees specified in the forest.

Predicted probabilities for the test set are obtained using the fitted Random Forest model. Here, type = "prob" returns predicted probabilities for both classes, and the probability of class 1 (click) is extracted. A threshold is applied to convert probabilities into binary class predictions. In this case, a threshold of 0.5 is used: if the predicted probability  0.5, the outcome is classified as 1 (click), otherwise as 0 (no click). The first 10 records of the test set with predicted outcomes are displayed for inspection using head(10).

```r
# Random Forest model begins
train_data$click <- as.factor(train_data$click)
model_rf <- randomForest(click ~ ., data = train_data, ntree = 100)
print(model_rf)
predicted_rf_prob <- predict(model_rf, test_data, type = "prob")
predicted_rf <- predicted_rf_prob[, 2]

# Round predicted probabilities to 0 or 1
# If probability >= 0.5, predict "click" (1), otherwise predict "no click" (0).
test_data$predicted_rf <- ifelse(predicted_rf >= 0.5, 1, 0)
head(test_data, 10)

```

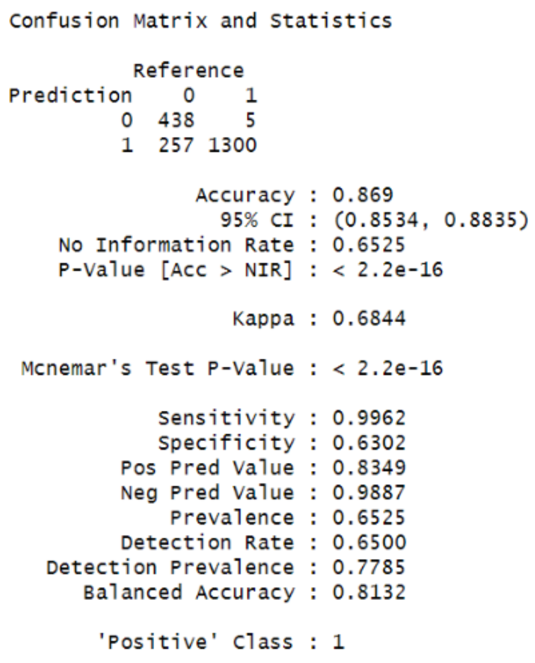


```
     age gender      device_type ad_position browsing_history time_of_day click predicted predicted_rf
   <dbl> <chr>       <chr>       <chr>       <chr>            <chr>       <int>     <dbl>        <dbl>
 1    41 Non-Binary  Mobile      Side        Education        Night           1         1            1
 2    40 Male        Mobile      Side        Unknown          Evening         0         1            0
 3    -1 Female      Desktop     Bottom      Unknown          Morning         1         1            1
 4    47 Unknown     Mobile      Bottom      Shopping         Afternoon       1         1            1
 5    37 Unknown     Mobile      Bottom      Unknown          Morning         1         1            1
 6    -1 Male        Tablet      Bottom      Unknown          Night           1         1            1
 7    -1 Unknown     Desktop     Unknown     Shopping         Afternoon       0         0            0
 8    -1 Unknown     Desktop     Side        Unknown          Night           0         1            1
 9    29 Non-Binary  Mobile      Bottom      Unknown          Afternoon       1         1            1
10    -1 Unknown     Mobile      Top         Entertainment    Unknown         0         0            0
```

**Figure 25:** *The first 10 rows of test$_s$et*

With the help of confusionMatrix() from the library(caret), we build a confusion matrix and calculate the model's performance.

```r
# 3. Evaluate the model with a confusion matrix
# Calculate and display the confusion matrix for the model
# positive = '1' indicates that the positive class is "click = 1"
confusionMatrix(as.factor(test_data$predicted), as.factor(test_data$click), positive
= "1")

```



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  438    5
         1  257 1300

               Accuracy : 0.869
                 95% CI : (0.8534, 0.8835)
    No Information Rate : 0.6525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6844

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9962
            Specificity : 0.6302
         Pos Pred Value : 0.8349
         Neg Pred Value : 0.9887
             Prevalence : 0.6525
         Detection Rate : 0.6500
   Detection Prevalence : 0.7785
      Balanced Accuracy : 0.8132

       'Positive' Class : 1
```

**Figure 26:** *Random Forest confusion matrix*

Random Forest notably outperforms the Logistic Regression model, particularly in accuracy level. The RF model achieves an accuracy of 86.90%, while logistic regression lags behind at 82.75% accuracy. For balanced accuracy, Random Forest is still better at 81.32%, compared to 75.18% of Logistic Regression. Most importantly, sensitivity (recall for the positive class) is 0.9962 for Random Forest and 1.00 for Logisctic Regression, but Random forest has a meaningfully higher specificity at 63.02% compared to 50.36% of Logistic regression, suggesting that Random forest better distinguishes non-click instances while capturing click instances, reducing false positives more effectively. We plot ROC curve and compute AUC value to visualize how strong discriminative ability between the 'click' and 'no click' is.

```r
# 4. Plot ROC curve and calculate AUC
ROCRpred = prediction(as.numeric(test_data$predicted_rf),test_data$click)
ROCRperf = performance(ROCRpred, "tpr","fpr")
plot(ROCRperf, main = "Random Forest ROC Curve")
auc <- as.numeric(performance(ROCRpred,"auc")@y.values)
```

```
  cat("AUC:", auc, "\n")


```

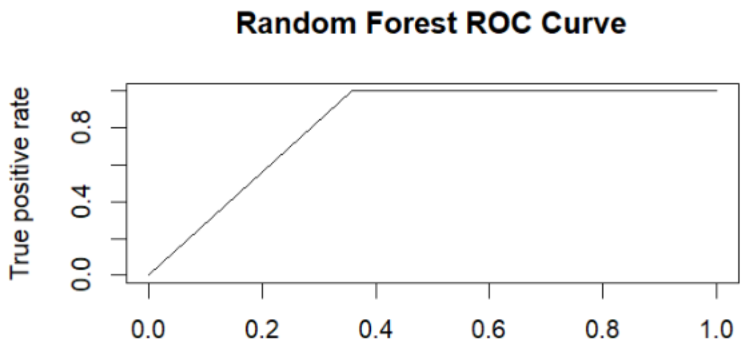
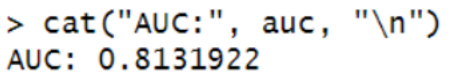


**Figure 27:** *ROC curve of Random Forest model*



```
> cat("AUC:", auc, "\n")
AUC: 0.8131922
```

**Figure 28:** *Random Forest AUC value*

The area below the ROC curve is impressively high at 0.8131922, indicating that Random Forest indeed outperforms Logistic Regression. The ROC curve confirms what the confusion matrix hinted at earlier: improved specificity, and AUC values above 0.8 typically indicate a reliable model for classification problems.
Below is the summary table of the two models for comparison:

| **Model** | **Logistic Regression** | **Random Forest** |
|---|---|---|
| Accuracy | 0.8275 | 0.8690 |
| Sensitivity (Recall) | 1.0000 | 0.9962 |
| Specificity | 0.5036 | 0.6302 |
| Precision (Positive predictive value) | 0.7909 | 0.8349 |
| Negative predictive value | 1.0000 | 0.9887 |
| Balanced accuracy | 0.7518 | 0.8132 |
| AUC (ROC curve area) | 0.7518 | 0.8132 |

**Table 2:** Summary table

## VII Conclusion

In this report, we conducted a comprehensive analysis of the factors influencing users' ad-clicking behavior using a detailed Ad Click Prediction dataset. We began by performing rigorous data preprocessing, including imputation of missing values based on individual user history to ensure dataset integrity and consistency. Through exploratory visualizations such as histograms, bar plots, pie charts, and correlation matrices, we identified key relationships between the likelihood of clicking on an advertisement and several user attributes, including age, gender, device type, ad position, browsing history, and time of day.

Subsequently, we constructed and evaluated a logistic regression model to predict ad clicks based on these attributes. Model fitting and statistical testing revealed that variables such as age, device type (particularly mobile), ad position (top and side positions), browsing history (entertainment), and time of day (evening and night) significantly influenced ad-click probabilities. Validation via the confusion matrix, ROC curves confirmed that our logistic regression model offered robust and accurate predictions. To complement this analysis and explore potential improvements, we also employed a Random Forest model. The Random Forest provided a slightly higher predictive accuracy compared to Logistic Regression, demonstrating its capability to handle nonlinear relationships and interactions among variables effectively. Despite increased computational complexity and reduced interpretability compared to logistic regression, Random Forest proved valuable in capturing subtle and complex patterns within the dataset, thus enhancing our predictive understanding of user click behavior.

Our analysis provides valuable insights into the interactions between user demographics, browsing context, and their impacts on ad engagement. These insights contribute significantly to optimizing digital advertising strategies, enabling marketers and advertisers to deliver highly targeted, effective campaigns.

## VIII Source code

```r
# Import libraries
library(corrplot)
library(ggplot2)
library(dplyr)
library(caret)
library(ROCR)
library(randomForest)
library(car)
library(olsrr)
library(lmtest)
#Import file
data <- read.csv("C:/Users/Admin/Desktop/ad_click_dataset.csv")
head(data)
# Replace empty strings with NA
data[data== ""] <- NA
str(data)
# Count occurrences of blanks and NA in each column
na_count <- sapply(data, function(x) sum(is.na(x)))
# Print the count of missing values per column
print(na_count)
cleaned_data <- data

#Summarize each u s e r s total records and total clicks
user_summary <- cleaned_data %>% group_by(full_name) %>% summarise(total_records = n(),
                                                                  total_clicks = sum(click))
user_summary <- user_summary %>% mutate(user_category = case_when(total_records == 1 ~ "first-time",
                                                                 total_records > 1 ~ "recurring",
                                                                 TRUE ~ "other"))
cat("Total Unique Users:", nrow(user_summary), "\n")

first_time_count <- sum(user_summary$user_category == "first-time")
recurring_count <- sum(user_summary$user_category == "recurring")
cat("Number of first-time users:", first_time_count, "\n")
cat("Number of recurring users :", recurring_count, "\n")

#Define a helper function to get the mode of a vector
get_mode <- function(x) {
  x <- x[!is.na(x)]  # Remove NA
  if (length(x) == 0) return(NA)  # Return NA if all values are NA
  uniq_x <- unique(x)
  uniq_x[which.max(tabulate(match(x, uniq_x)))]
}
# Fill NA values in categorical columns with the mode of each user
cleaned_data <- cleaned_data %>%
  group_by(full_name) %>%
  mutate(age = ifelse(is.na(age), median(age, na.rm = TRUE), age), # For numeric column,
    use the median age of each user
         # For categorical columns, use the per-user mode:
         gender = ifelse(is.na(gender), get_mode(gender), gender),
         device_type = ifelse(is.na(device_type), get_mode(device_type), device_type),
         ad_position = ifelse(is.na(ad_position), get_mode(ad_position), ad_position),
         browsing_history = ifelse(is.na(browsing_history), get_mode(browsing_history),
    browsing_history),
         time_of_day = ifelse(is.na(time_of_day), get_mode(time_of_day), time_of_day)
  ) %>%
  ungroup()
# Fill NA values in age column with -1
cleaned_data$age[is.na(cleaned_data$age)] <- -1
# Fill other NA with unknown
categorical_cols <- c("gender", "device_type", "ad_position", "time_of_day", "browsing_
    history")
for (col in categorical_cols) {
  cleaned_data[[col]][is.na(cleaned_data[[col]])] <- "Unknown"
}
# Check for missing values
na_count <- sapply(cleaned_data, function(x) sum(is.na(x)))
print(na_count)

# drop full_name and id column
cleaned_data <- cleaned_data[, c("age","gender","device_type","ad_position",
                                 "browsing_history","time_of_day","click")]
# Recheck our data by displaying to the console
str(cleaned_data)
# Basic information about our refined data
summary(cleaned_data)
summary(cleaned_data %>% filter(age != -1)) # Without -1 age


# Frequency Table for categorical variables:
apply(cleaned_data[c("gender","device_type","ad_position",
                     "browsing_history","time_of_day","click")],2,table)

#Basic histogram of "age"
ggplot(cleaned_data %>% filter(age != -1), aes(x = age)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Age", x = "Age", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Distribution of age by click
ggplot(cleaned_data %>% filter(age != -1), aes(x = age, fill = factor(click))) +
  geom_histogram(position = "identity", alpha = 0.5, binwidth = 5, color = "black") +
  scale_fill_manual(values = c("red", "skyblue"),
                    name = "Click",
                    labels = c("No Click", "Click")) +
  labs(title = "Histogram of Age by Click Behavior", x = "Age", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Barplot & set up grid for more plots
par(mfrow = c(3, 2), mar = c(2, 2, 2, 2) + 0.1) # Reduced margins
categorical<-c("gender","device_type","ad_position","browsing_history","time_of_day","
    click")
for (var in categorical){
  barplot(table(cleaned_data[[var]]),
          main = paste("Barplot of", var),
          col = "skyblue",
          border = "black",
          cex.names=0.6)
}
# reset plotting layout to default
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)
#Pie chart & set up grid for more plots
par(mfrow = c(3, 2), mar = c(1, 2, 2, 2) + 0.1) # Reduced margins
for (var in categorical){
  counts <- table(cleaned_data[[var]])
  percents <- round(100 * counts / sum(counts), 1)  # Calculate percentages
  labels <- paste0(names(counts), "\n", percents, "%")  # Create labels with percent
  pie(counts,
      labels=labels,
      main = paste("Pie chart of", var),
      col = "skyblue", border = "black",
      radius=1,
      cex=0.7)
}
# reset plotting layout to default
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)

# Filter out rows where device_type or ad_position is 'Unknown'
filtered_data <- cleaned_data %>%
  filter(device_type != "Unknown", ad_position != "Unknown")

# Create summary table: click rate by device type and ad position (excluding 'Unknown')
click_table <- tapply(filtered_data$click,
                      list(filtered_data$ad_position, filtered_data$device_type),
                      mean) #Calculates the mean of the click variable grouped by ad_
    position and device_type

# Print the click rate table
print(click_table)

# Create barplot
barplot(click_table,
        beside = TRUE,
        col = c("skyblue", "lavender", "lightcoral"),  # Adjust number of colors based on
     number of ad positions
        ylim = c(0, 1),
        main = "Click Rate by Device Type and Ad Position (without Unknown)",
        xlab = "Device Type",
        ylab = "Click Rate",
        cex.main = 0.85,
        cex.lab = 0.8, # Size of axis labels (x and y labels)
        cex.axis = 0.8,  # Size of axis tick labels
        legend.text = rownames(click_table), # Adds a legend using the row names
        args.legend = list(x = 13.5, y = 1.15, cex = 0.6)) # Specifies the legend
    position and size

# Correlation matrix visualization
correlation_matrix <- cor(cleaned_data[,sapply(cleaned_data, is.numeric)], use = "
    complete.obs")
corrplot(correlation_matrix, method = "square", #square tiles
         addCoef.col = "red",                   #coefficient color
         tl.col = "black",                      #text label color
         number.digits = 4)                     #ensure enough space

# Box plots and Set layout to display multiple plots
par(mfrow = c(3, 2), mar = c(2, 2, 2, 0) + 0.1) # Reduced margins
categorical_cols<-c("gender","device_type","ad_position","browsing_history","time_of_day"
    ,"click")
# Loop through each variable and create box plot
for (var in categorical) {
  formula <- as.formula(paste("age ~", var))
  # Filter out -1 ages and 'Unknown' for clean plots
  filtered_data <- cleaned_data %>% filter(age != -1, !!as.name(var) != "Unknown")
  par(las = 1)  # Adjust axis label size
  box_stats <- boxplot(formula, data = filtered_data,
                       col = c("skyblue", "lightgreen", "plum", "lightcoral", "gold"), #
    set colors
                       border = "darkblue",
                       main = paste("Age by", var),
                       xlab = var,
                       xaxt = "n",  # Suppress default x-axis
                       cex.axis = 0.75,  # Size for y-axis tick labels
                       ylab = "Age")
  # Add custom x-axis
  axis(1, at = seq_along(unique(filtered_data[[var]])),
       labels = unique(filtered_data[[var]]),
```

```r
        cex.axis = 0.58  # Smaller size for x-axis tick labels
        )
  # Calculate means by group
  group_means <- tapply(filtered_data$age, filtered_data[[var]], mean)
  # Overlay means (red points)
  points(1:length(group_means), group_means, col = "red", pch = 19)  # Red dots for means

  # Add text for mean and median values
  text(1:length(group_means), group_means + 7,  # Adjust vertical position for clarity
        labels = round(group_means, 1), col = "red", cex = 0.8)  # Text for means
  text(1:length(box_stats$names), box_stats$stats[3, ] - 2,  # Adjust vertical position for clarity
        labels = round(box_stats$stats[3, ], 1), col = "black", cex = 0.7)  # Text for medians
}
# reset plotting layout to default
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1, cex.axis = 1, las = 0)


# ----- Predictive modelling begins -----
# Set up training and testing data:
set.seed(100) # Set seed for reproducibility
# Randomly sample 80% of rows from data for training
train.rows <- sample(rownames(cleaned_data), dim(cleaned_data)[1] * 0.8)
train_data <- cleaned_data[train.rows, ] # Create training set from chosen data

# Use remaining 20% as testing data
test.rows <- setdiff(rownames(cleaned_data), train.rows)
test_data <- cleaned_data[test.rows, ] #Create test set from chosen data
# 1.Logistic regression model begins
# Fit logistic regression model using all predictors to predict 'click'
# and all other variables as independent variables in the training data. Model follows a binomial distribution.
RM_model <- glm(click ~ ., family = "binomial", data = train_data)

# View detailed information about the trained logistic regression model, including parameters, p-values, and other statistics
summary(RM_model)

# 2. Predict "click" probabilities for test data
# Use the trained model to predict the probability of "click" for observations in test_data
# type="response" returns the probability of the "click = 1" class.
predicted <- predict(RM_model, test_data, type = "response")

# Round predicted probabilities to 0 or 1
# If probability >= 0.5, predict "click" (1), otherwise predict "no click" (0).
test_data$predicted <- ifelse(predicted >= 0.5, 1, 0)

# Display the first 10 rows of test data, including the "predicted" column
head(test_data, 10)

# 3. Evaluate the model with a confusion matrix
# Calculate and display the confusion matrix for the model
# positive = '1' indicates that the positive class is "click = 1"
confusionMatrix(as.factor(test_data$predicted), as.factor(test_data$click), positive = "1")


# 4. Plot ROC curve and calculate AUC
ROCRpred <- prediction(as.numeric(test_data$predicted), test_data$click)
ROCRperf <- performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf, main = "Logistic Regression ROC Curve")
# Calculate the area under the ROC curve (AUC)
auc <- as.numeric(performance(ROCRpred, "auc")@y.values)
cat("AUC:", auc, "\n")

" ---------- MULTICOLLINEARITY CHECK -----------"
print(vif(RM_model))

# ---------- LEVERAGE PLOTS -----------"
ols_plot_resid_lev(RM_model)
```

```



# Random Forest model begins
train_data$click <- as.factor(train_data$click)
model_rf <- randomForest(click ~ ., data = train_data, ntree = 100)
print(model_rf)
predicted_rf_prob <- predict(model_rf, test_data, type = "prob")
predicted_rf <- predicted_rf_prob[, 2]

# Round predicted probabilities to 0 or 1
# If probability >= 0.5, predict "click" (1), otherwise predict "no click" (0).
test_data$predicted_rf <- ifelse(predicted_rf >= 0.5, 1, 0)
getOption("width") #expand the console window
head(test_data, 10)

# 3. Evaluate the model with a confusion matrix
confusionMatrix(as.factor(test_data$predicted_rf), as.factor(test_data$click), positive =
    "1")

# 4. Plot ROC curve and calculate AUC
ROCRpred = prediction(as.numeric(test_data$predicted_rf),test_data$click)
ROCRperf = performance(ROCRpred, "tpr","fpr")
plot(ROCRperf, main = "Random Forest ROC Curve")
auc <- as.numeric(performance(ROCRpred,"auc")@y.values)
cat("AUC:", auc, "\n")

# Independence of errors check
durbinWatsonTest(RM_model)
```

## References

[1] Sree Vani. (2017). *Forecast of mobile ad click through logistic regression algorithm. International Journal of P2P Network Trends and Technology.*
https://www.ijpttjournal.org/archives/ijptt-v7i5p405

[2] Sitong Zhou. (2022). Analyzing factors of Users' Click Behavior on Ads Based on Logistic Regression and Machine learning. *Atlantis Press.*
https://www.atlantis-press.com/proceedings/isemss-22/125982064

[3] Gilles Louppe. (2014). Understanding Random Forests: From Theory to Practice. *Cornell University.*
https://arxiv.org/abs/1407.7502

[4] Erwan Scornet, Gérard Biau, Jean-Philippe Vert. (2015). Consistency of random forests. *Cornell University.*
https://arxiv.org/abs/1405.2881