Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Systems Department

Course Title:	Data Analytics
Course Code:	**IS-6114**
Course Instructor:	Dr. Mona Alkhattabi
Semester:	2nd Semester 2024-25
Date:	30 January 2025

Project Data Analytics

Student Name	Student ID
Lama Aljohani	446012727
Alaa Hassan	446012602
Lulwah AlJammaz	446012617
Norah Almoqhim	446011789

PROJECT – DATA ANALYTICS (IS 6114)
Academic Year 2024/2025, Second Semester

Total Points: 100 *(This assignment is worth 20% of your total course grade)*

Report Due Date: Thursday 30 January 2025, by 11:59 pm

Objective:

To utilize a comprehensive set of data analytics skills—from data wrangling and visualization to statistical analysis, machine learning, reporting, and presentation—on a real-world dataset.

Steps:

1. Make a group of 4 students Each student should enroll herself in the chosen group in Blackboard by the end of week 5 (Thursday, 19 December, 11:59 pm) (Groups formation will be discussed during the lecture)

2. Download a dataset. There are many datasets available from the website: http://archive.ics.uci.edu/ml/. The chosen dataset **must have at least 5 attributes, and 100 samples**.

3. Follow all steps in the **Project Components and Grading Rubric** below and be sure to cover all required components.

4. Each group should submit one copy via Blackboard

Deliverables:

1. Project Report:

   Submit a detailed report in (PDF format).

2. GitHub Repository:

   Include all code, datasets, and documentation in a public GitHub repository and provide the access link with the report.

3. Presentation Slides:

   Prepare a slides (PowerPoint) summarizing key project highlights

GitHub Repository

Include all code, datasets, and documentation in a public GitHub repository and provide the access link with the report.

```
system("git init")  # Initialize Git in the folder
system("git remote add origin https://github.com/AlaaHassanIS/DAProject-.git")  # Connect to GitHub
system("git add .")  # Add all files to Git
system('git commit -m "Initial commit"')  # Commit changes
system("git push -u origin main")  # Push files to GitHub
```

Project Components and Grading Rubric:

#	Component	Description	Requirements	Grading Criteria	Points
1	**Dataset Selection, Description, and Problem Definition**	Select and describe a real world dataset, and clearly define the problem it addresses.	• Select a dataset • Provide a detailed description of the dataset. • Define the problem and. • Justify the dataset's relevance to the problem.	• Clear problem definition (4 pts). • Detailed dataset description (3 pts). • Justification of dataset selection (3 pts).	10
2	**Data Wrangling**	Clean and transform the dataset to prepare it for analysis.	• Handle missing data and data type conversions. • Standardize variables. • Apply techniques like merging, filtering, subsetting, and reshaping.	• Completeness of data cleaning (8 pts). • Appropriate data transformation techniques (7 pts).	15
3	**Exploratory Data Analysis (EDA) and Visualization**	Analyze the data to uncover distributions, relationships, and trends using visualizations	• Generate summary statistics. • Create visualizations (e.g., histograms, box plots, scatter plots). • Identify patterns, anomalies, and insights.	• Clarity and quality of visualizations (8 pts). • Interpretation of EDA results (7 pts)	15
4	**Statistical Analysis**	Apply statistical techniques to validate assumptions or derive insights.	• Use statistical tests • Implement linear or logistic regression and interpret key results.	• Correct statistical tests (7 pts). • Accurate interpretation of regression results (8 pts).	15
5	**Machine Learning Model**	Build and evaluate a predictive model using appropriate ML techniques.	• Choose and justify an ML algorithm • Split data, perform crossbvalidation, and tune hyperparameters. • Evaluate model performance using metrics	• Model selection and justification (5 pts). • Model training/tuning (5 pts). • Evaluation (5 pts).	15
6	**Reporting and Reproducibility**	Document the project.	• Write a report following the report guidline. • Use Git for version control and submit via GitHub. • Ensure code is well-documented and reproducible.	• Quality of the report (9 pts). • Proper use of Git and documentation (6 pts).	15
7	**Project Presentation**	Present the analysis, including motivation, methods, findings, and conclusions.	• Create a 10-15 minute presentation. • Include visuals and explain key findings. • Answer questions from peers and instructors.	• Clarity of presentation (5 pts). • Communication of methods/findings (5 pts). • Handeling Questions (5 pts).	15
Total Points					**100**

When providing the dataset description, include the following:

1. **Overview:** A brief summary of the dataset's content and purpose (e.g., Healthcare, Financial, social media posts, Education).
2. **Variables:** Key variables in the dataset and their types (e.g., numerical, categorical, text).
3. **Size:** Number of rows and columns.
4. **Context:** Why the dataset is relevant to the problem you are addressing.

The report should contains the following sections:

- **Introduction**: Provide an overview of the dataset, problem statement, and the purpose of the project. Explain why the dataset was chosen and its relevance to the problem.
- **Dataset Description**: Describe the dataset in detail, including number of records, variables, and any key characteristics. Discuss any challenges related to the dataset.
- **Data Wrangling**: Detail the steps taken to clean and transform the data. Explain how missing values were handled, data types converted, and variables standardized.
- **Exploratory Data Analysis (EDA)**: Summarize the findings from EDA, including visualizations and statistical summaries. Identify trends, patterns, or anomalies in the data.
- **Statistical Analysis**: Discuss the statistical methods used to validate assumptions or derive insights. Include results of tests such as regression analysis and interpret the outcomes.
- **Machine Learning Model**: Explain the machine learning model developed, including the choice of algorithm, data splitting, cross-validation. Present the model performance metrics.
- **Results and Discussion**: Present the key findings and insights gained from the analysis.
- **Conclusion and Recommendations**: Summarize the overall findings of the project and provide recommendations based on the analysis. Suggest potential areas for future work.
- **References**: List all sources, datasets, and tools used in the project.
- **Appendices**: Include additional materials such as code snippets, detailed tables, or supplementary visualizations.

Good luck

Index

Topic	Page NO
Introduction	8
Dataset Description	9
Data Wrangling	14
Exploratory Data Analysis (EDA)	18
Statistical Analysis	25
Machine Learning Model	35

Introduction

In a world increasingly driven by data to understand consumer patterns and optimize business strategies, data analysis has become an essential tool for informed decision-making. This project focuses on analyzing the "Wholesale Customers Dataset", which contains annual spending details of a group of customers across various product categories such as fresh products, milk, grocery, and frozen foods.

The objective of this study is to uncover insights into customer spending patterns, identify relationships between product categories, and explore the factors that differentiate sales channels (Horeca, which includes hotels and restaurants, versus retail). Additionally, the project aims to build a machine learning model to predict the customer's channel based on their spending habits.

This dataset was chosen due to its relevance in understanding customer behavior and its potential to improve marketing strategies and inventory management. Using modern data analysis techniques, including data cleaning, exploratory analysis, statistical modeling, and machine learning, the project provides valuable insights that businesses can leverage to enhance their performance and efficiency.

Main Project Objective:

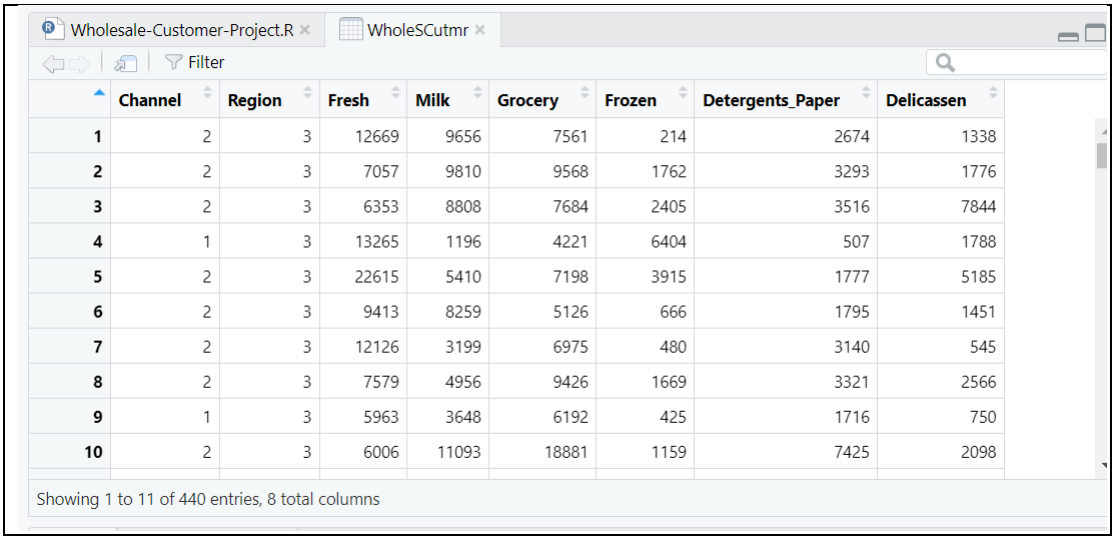
- Analyze customer behavioral patterns based on spending data.
- Identify key factors influencing purchasing decisions.
- Develop a predictive model to classify customers by sales channel (Horeca or retail).

Project Significance:

- Enhance marketing strategies by gaining a deeper understanding of customers.
- Improve resource and inventory management efficiency.
- Provide actionable insights based on data to enhance customer satisfaction.

Dataset Description Wholesale customers

1. **Dataset:** Wholesale customers. (Business)



Wholesale-Customer-Project.R × WholeSCutmr ×

Filter

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
1	2	3	12669	9656	7561	214	2674	1338
2	2	3	7057	9810	9568	1762	3293	1776
3	2	3	6353	8808	7684	2405	3516	7844
4	1	3	13265	1196	4221	6404	507	1788
5	2	3	22615	5410	7198	3915	1777	5185
6	2	3	9413	8259	5126	666	1795	1451
7	2	3	12126	3199	6975	480	3140	545
8	2	3	7579	4956	9426	1669	3321	2566
9	1	3	5963	3648	6192	425	1716	750
10	2	3	6006	11093	18881	1159	7425	2098

Showing 1 to 11 of 440 entries, 8 total columns

2. **Overview:** The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

The Wholesale Customers dataset is designed to analyze and segment the retail and wholesale sales market. It provides details of annual spending across different product categories for different customers, helping businesses understand purchasing patterns and identify customer groups by region.

3. Variables:

Key variables in the dataset include:

- **Channel**: Categorical variable representing the type of customer (e.g., Horeca - Hotel/Restaurant/Cafe or Retail).
- **Region**: Categorical variable representing the customer's geographical region.
- **Fresh**: Numerical variable indicating annual spending on fresh products.
- **Milk**: Numerical variable indicating annual spending on milk products.
- **Grocery**: Numerical variable indicating annual spending on grocery items.
- **Frozen**: Numerical variable indicating annual spending on frozen products.
- **Detergents_Paper**: Numerical variable indicating annual spending on

detergents and paper products.

- **Delicassen**: Numerical variable indicating annual spending on delicatessen items.

```
> str(WholeSCutmr)
'data.frame':   440 obs. of  8 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
>
```

4. **Size:** 440 rows (customers)× 8 columns (variables)

5. **Key Characteristics of the Dataset:**

    1. **Categorical Variables**:
        - **Channel**: Indicates the type of customer (e.g., Horeca - Hotel/Restaurant/Cafe or Retail).
        - **Region**: Identifies the geographical region of the customer (e.g., Lisbon, Porto, or other regions).

          These variables help in understanding the demographic and operational segmentation of the customers

    2. **Numerical Variables:**
        - **Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen**: Represent the annual spending of customers on specific product categories (measured in monetary units).These variables capture the financial behavior of customers and are crucial for clustering or segmentation analysis.
    3. **Heterogeneity in Spending:**
        - Different customers show varied spending patterns across product categories, highlighting diverse purchasing behaviors.
        - For instance, some customers may focus more on fresh products, while others may prioritize grocery or detergents.
    4. **Imbalanced Features:**
        - Certain variables, like "Detergents_Paper," may have a higher concentration of low or zero spending, which can impact the clustering process and requires careful handling.

5. **Compact Dataset Size:**
    - With 440 observations and 8 features, the dataset is relatively small, making it manageable for exploratory data analysis and machine learning tasks.

These characteristics make the dataset particularly suitable for unsupervised learning tasks such as clustering, where the aim is to uncover hidden patterns and groupings in customer behavior.

Context:

This dataset is relevant for market segmentation tasks, clustering analysis, and predictive modeling in retail and wholesale business strategies. By analyzing customers' spending behavior across product categories, businesses can tailor their services, optimize inventory, and develop targeted marketing campaigns.

Problem statement:

Businesses in wholesale and retail sales face challenges in understanding and predicting customer purchasing behavior. Effective segmentation is essential for improving inventory management, personalizing marketing strategies, and optimizing overall business operations. The challenge is to identify meaningful customer segments and derive actionable insights to cater to diverse customer needs efficiently.

Purpose of the Project:

The goal of this project is to analyze the purchasing patterns of wholesale customers and identify distinct customer segments. By applying clustering techniques, the project aims to group customers with similar behaviors, enabling businesses to make informed decisions for targeted marketing, resource allocation, and service customization.

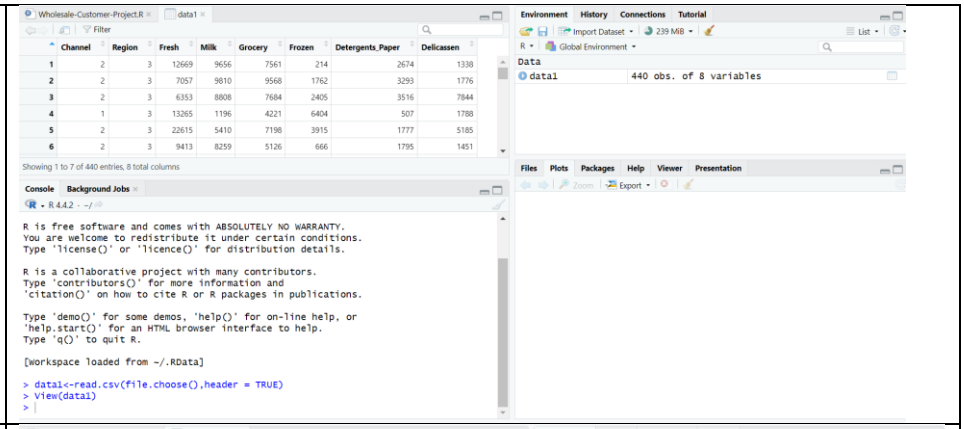
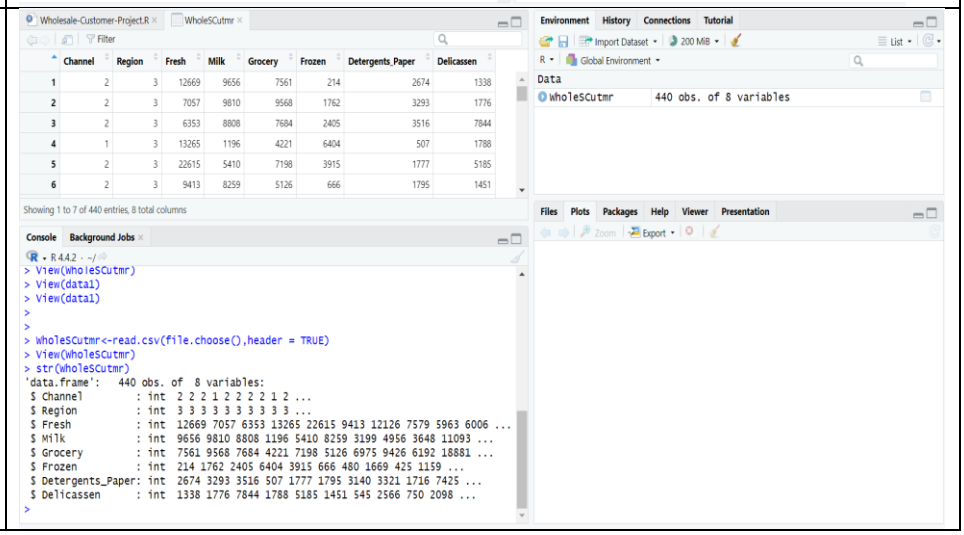
Why This Dataset Was Chosen:

This dataset was selected because it offers a rich set of variables directly linked to customer behavior in the wholesale industry. Its combination of numerical and categorical data makes it suitable for segmentation analysis, clustering, and predictive modeling. Additionally, its structure and content align closely with the goal of understanding and categorizing wholesale customers for business optimization.

Relevance to the Problem:

The dataset's relevance lies in its detailed representation of customer spending habits across multiple product categories. This information is crucial for

identifying patterns and drawing insights that help businesses improve their understanding of customer needs, optimize operations, and enhance customer satisfaction.

Import data into RStudio	
Variables	

Data Wrangling

The Target variable is region
Data Cleaning and Transformation Steps in R:

1. **Handling Missing Values:**
    - Check for Missing Values:
      Use the is.na() function to identify missing values.

```
>
> sum(is.na(WholeSCutmr))  # Total missing values
[1] 0
> colSums(is.na(WholeSCutmr))  # Missing values per column
        Channel          Region           Fresh            Milk
              0               0               0               0
        Grocery          Frozen Detergents_Paper       Delicassen
              0               0               0               0
>
```

    - Strategy: If missing values were found:
        - Numerical variables were imputed using the mean or median, depending on the distribution.
        - Categorical variables were imputed using the mode or assigned to a special category like "Unknown."
2. **Data Type Conversion:**
    - Categorical Variables: Variables like Channel and Region were converted to appropriate categorical data types using .astype('category').
    - Numerical Variables: Variables like Fresh, Milk, and other spending categories were ensured to be in numerical format (e.g., float or int) for mathematical operations.
3. **Outlier Detection and Treatment:**
    - Outliers in numerical variables were detected using statistical methods such as the Interquartile Range (IQR) or Z-score.
    - Outliers were either capped (e.g., at the 1st and 99th percentiles) or handled based on domain knowledge to avoid skewing the analysis.
4. **Variable Transformation:**

- Skewed numerical variables were transformed using logarithmic or square-root transformations to normalize distributions for clustering or modeling.
- For example, spending on categories like Fresh and Frozen often exhibits positive skewness.

5. **Standardization/Normalization:**

- Numerical variables were standardized (using Z-scores) or normalized (scaling between 0 and 1) to ensure equal weighting in distance-based algorithms like K-means clustering.
- Standardization formula: z=x−μσz = \frac{{x - \mu}}{{\sigma}}z=σx−μ where xxx is the data point, μ\muμ is the mean, and σ\sigmaσ is the standard deviation.

6. **Encoding Categorical Variables:**

- Categorical variables (Channel and Region) were encoded using one-hot encoding or label encoding to prepare them for machine learning models.
- One-hot encoding was preferred to avoid introducing ordinal relationships in non-ordinal data.

7. **Final Dataset Verification:**

- After transformations, the dataset was reviewed for consistency, ensuring all features were properly formatted and ready for analysis.
- The updated dataset was split into feature matrices (X) and target variables (if applicable).

Data Wrangling : Standardize variables

```
# Standardize numeric variables only. قياس المتغيرات الرقمية فقط
numeric_columns <- sapply(data, is.numeric)  #Define numeric columns رقمية
data_standardized <- as.data.frame(scale(data[, numeric_columns]))

#View sample benchmark data عرض عينة من البيانات المعيارية
head(data_standardized)
```

This is result for the code of standardized variables

data	440 obs. of 8 variables
data_standardized	440 obs. of 8 variables

Data Wrangling: some techniques

Data Wrangling	1. NO missing value- 2. 2. Process missing values and replace them with the meaning of each numeric column. 3. Merging Data 4. Filtering Data 5. Subsetting Data 6. Reshaping Data
One aproplem When apply subsetting	The Region column was duplicated after merging it from the two tables and was given two names different from the original. When searching for it, we found it named: rigon.x , region.y
How repair?	Delete Region.x while keeping Region.y. Rename Region.y to region to make the data easier to read. Ensure that the required columns are present before executing subset () to prevent errors. All result is right for Data Wrangling
The Codes for Data Wrangling	

3. Merging Data	`# ✅ 4. Merge data and fix `Region.x` and `Region.y` issue`
	`MergedData <- merge(WholeSCutmr, customer_info, by = "CustomerID", all = TRUE)`
	
	`# ✅ 5. Fix duplicate columns `Region.x` and `Region.y``
	`MergedData <- MergedData %>%`
	`  select(-Region.x) %>%`
	`  rename(Region = Region.y)`
	
	`# ✅ 6. Check columns after merging`
	`colnames(MergedData)`

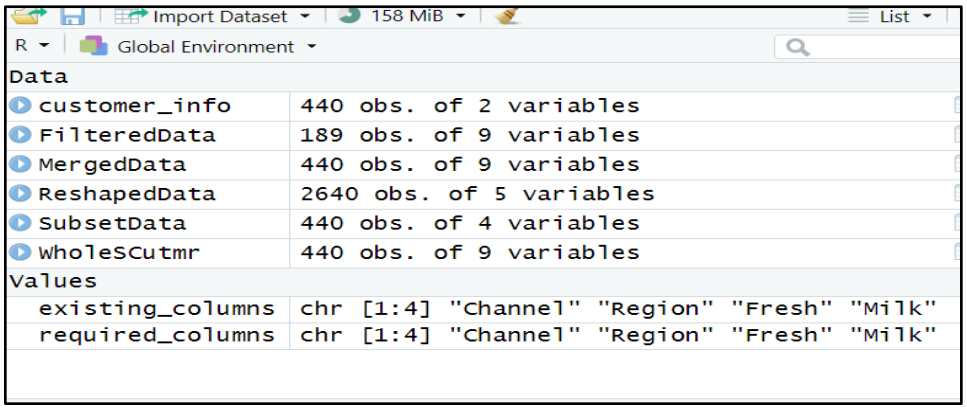
4. Filtering Data	`# 2. **Filtering Data**`
	`# Extract customers who purchased more than 10,000 units of fresh products`
	`FilteredData <- subset(MergedData, Fresh > 10000)`

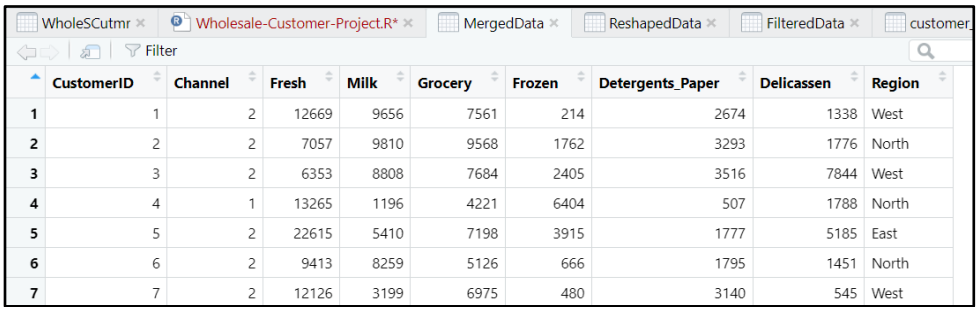
5. Subsetting Data	`# 3. **Subsetting Data**`
	`# Extract specific columns (after verifying their existence)`
	`required_columns <- c("Channel", "Region", "Fresh", "Milk")`
	`existing_columns <- required_columns[required_columns %in% colnames(MergedData)]`
	`SubsetData <- MergedData[, existing_columns, drop = FALSE]`

6. Reshaping Data	`# 4. **Reshaping Data**`
	`# Convert the dataset from wide format to long format using pivot_longer`
	`library(tidyr)`
	`ReshapedData <- pivot_longer(MergedData,`
	`  cols = c(Fresh, Milk, Grocery, Frozen,`
	`  Detergents_Paper, Delicassen),`
	`  names_to = "Product_Type",`
	`  values_to = "Amount")`

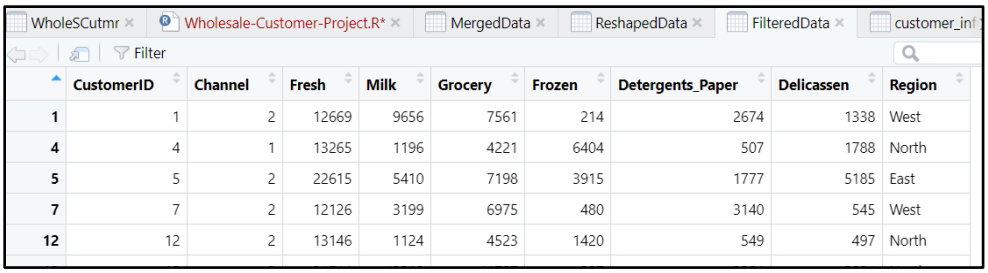
Result	`# Display the processed datasets`
	`head(MergedData)  # After merging`
	`head(FilteredData)  # After filtering`
	`head(SubsetData)  # After subsetting`
	`head(ReshapedData)  # After reshaping`

Shapes for data after applying some techniques:

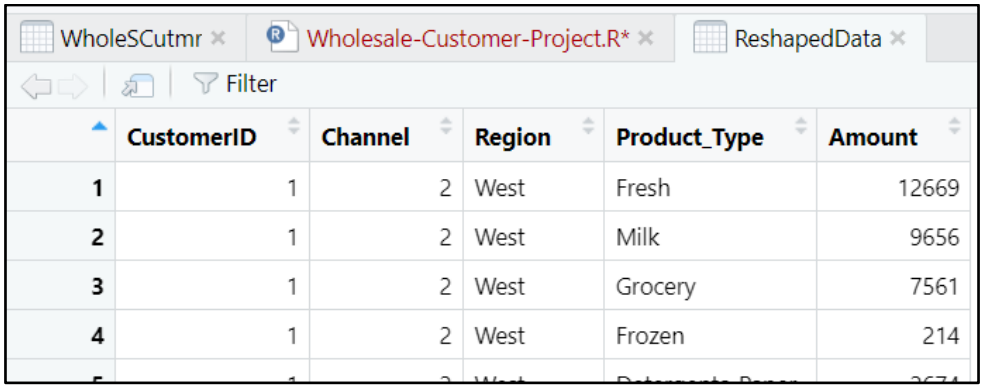




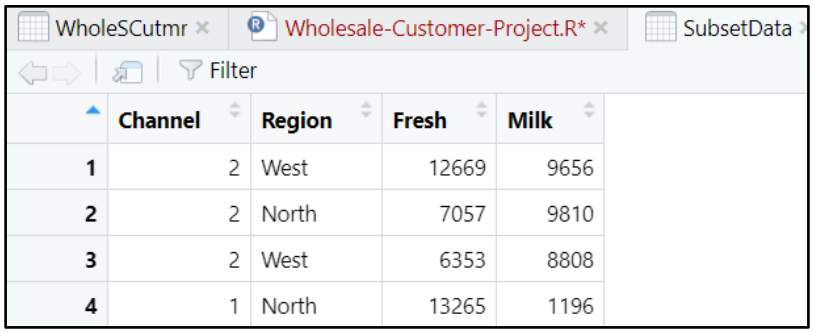
	CustomerID	Channel	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Region
1	1	2	12669	9656	7561	214	2674	1338	West
2	2	2	7057	9810	9568	1762	3293	1776	North
3	3	2	6353	8808	7684	2405	3516	7844	West
4	4	1	13265	1196	4221	6404	507	1788	North
5	5	2	22615	5410	7198	3915	1777	5185	East
6	6	2	9413	8259	5126	666	1795	1451	North
7	7	2	12126	3199	6975	480	3140	545	West



	CustomerID	Channel	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Region
1	1	2	12669	9656	7561	214	2674	1338	West
4	4	1	13265	1196	4221	6404	507	1788	North
5	5	2	22615	5410	7198	3915	1777	5185	East
7	7	2	12126	3199	6975	480	3140	545	West
12	12	2	13146	1124	4523	1420	549	497	North



	CustomerID	Channel	Region	Product_Type	Amount
1	1	2	West	Fresh	12669
2	1	2	West	Milk	9656
3	1	2	West	Grocery	7561
4	1	2	West	Frozen	214



	Channel	Region	Fresh	Milk
1	2	West	12669	9656
2	2	North	7057	9810
3	2	West	6353	8808
4	1	North	13265	1196

```r
# Merge the datasets using the common "CustomerID" column
MergedData <- merge(WholeSCutmr, customer_info, by = "CustomerID")

# check columns after merging to avoid errors
colnames(MergedData)  # تأكد من وجود Channel, Region, Fresh, Milk

#وجدت مشكلة في وجود عمود مشترك من الجداول مكرر وهو region
#وحلها كالتالي

# ✅ 4. Merge data and fix `Region.x` and `Region.y` issue
MergedData <- merge(WholeSCutmr, customer_info, by = "CustomerID", all = TRUE)

# ✅ 5. Fix duplicate columns `Region.x` and `Region.y`
MergedData <- MergedData %>%
  select(-Region.x) %>%
  rename(Region = Region.y)

# ✅ 6. Check columns after merging
colnames(MergedData)

# 2. **Filtering Data**
# Extract customers who purchased more than 10,000 units of fresh products
FilteredData <- subset(MergedData, Fresh > 10000)


# 3. **Subsetting Data**
# Extract specific columns (after verifying their existence)
required_columns <- c("Channel", "Region", "Fresh", "Milk")
existing_columns <- required_columns[required_columns %in% colnames(MergedData)]
SubsetData <- MergedData[, existing_columns, drop = FALSE]
```

48:1 (Top Level)

```r

# 4. **Reshaping Data**
# Convert the dataset from wide format to long format using pivot_longer
library(tidyr)
ReshapedData <- pivot_longer(MergedData,
                             cols = c(Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen),
                             names_to = "Product_Type",
                             values_to = "Amount")

# Display the processed datasets
head(MergedData)    # After merging
head(FilteredData)  # After filtering
head(SubsetData)    # After subsetting
head(ReshapedData)  # After reshaping

```

Exploratory Data Analysis (EDA)

1. Understanding the Dataset

Dataset Overview:

- 440 records and 8 columns (Channel, Region, Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicatessen).
- No missing values found (sum(is.na(data)) == 0).

Categorical Variables:

- Channel:
  - 1 = Horeca (Hotels, Restaurants, Cafés)
  - 2 = Retail
- Region:
  - 1 = Lisbon
  - 2 = Porto
  - 3 = Other

Numerical Variables (Annual Spending in Thousands of Euros):

- Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicatessen

```r
# Load required libraries
library(ggplot2)    # For data visualization
library(dplyr)      # For data manipulation
library(corrplot)   # For correlation matrix visualization

# Load the dataset
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20data.csv"
data <- read.csv(url)

# Display the first few rows
head(data)

# Check dataset structure
str(data)

# Check for missing values
sum(is.na(data))

# Summary statistics
summary(data)
```

```
>
> # Load the dataset
> url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20data.csv"
> data <- read.csv(url)
>
> # Display the first few rows
> head(data)
  Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
1       2      3 12669 9656    7561    214             2674       1338
2       2      3  7057 9810    9568   1762             3293       1776
3       2      3  6353 8808    7684   2405             3516       7844
4       1      3 13265 1196    4221   6404              507       1788
5       2      3 22615 5410    7198   3915             1777       5185
6       2      3  9413 8259    5126    666             1795       1451
>
> # Check dataset structure
> str(data)
'data.frame':   440 obs. of  8 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
>
> # Check for missing values
> sum(is.na(data))
[1] 0
>
> # Summary statistics
> summary(data)
    Channel          Region          Fresh             Milk          Grocery          Frozen       Detergents_Paper    Delicassen
 Min.   :1.000   Min.   :1.000   Min.   :    3   Min.   :   55   Min.   :    3   Min.   :  25.0   Min.   :    3.0   Min.   :    3.0
 1st Qu.:1.000   1st Qu.:2.000   1st Qu.: 3128   1st Qu.: 1533   1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2
 Median :1.000   Median :3.000   Median : 8504   Median : 3627   Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5
 Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796   Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9
 3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2
 Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498   Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
>
```

2. Data Distribution Analysis

Histogram Insights:

- The spending distribution on Fresh, Grocery, and Milk is highly skewed, with most customers spending small amounts and a few spending much higher.
- Detergents_Paper has low spending for most customers, indicating it is not a primary product for many.

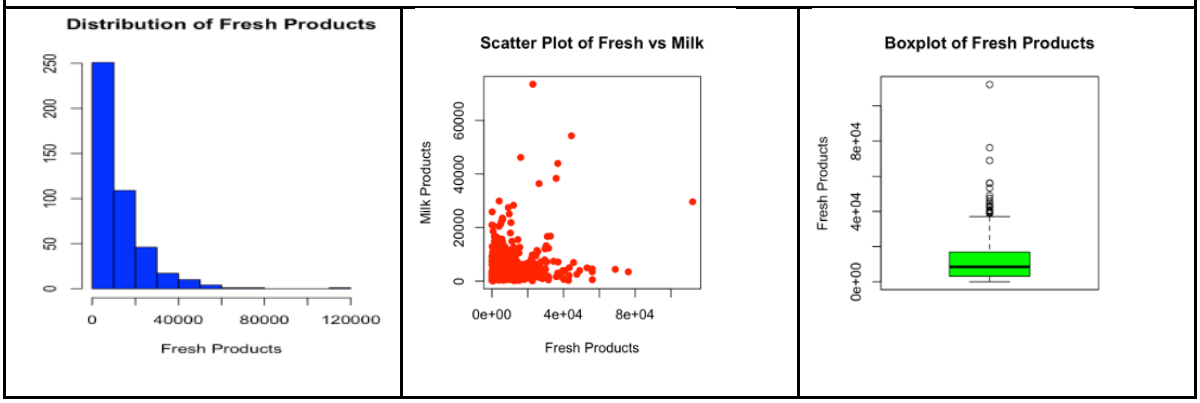
Boxplot Insights (Outliers Detection):

- Fresh, Grocery, and Delicatessen categories have outliers, meaning some customers spend significantly more than others.
- The median spending for all categories is relatively low, confirming that a small group of customers drives high total spending.

Scatter Plot Insights (Fresh vs. Milk Spending):

- The relationship between Fresh and Milk spending is weak (scattered points), meaning customers who buy a lot of Fresh products do not necessarily buy a lot of Milk.
- Some clusters of customers appear, indicating possible customer segments based on spending behavior.

```r
#Histogram for Fresh Products Spending
hist(data$Fresh,
     main = "Distribution of Fresh Products",
     xlab = "Fresh Products",
     col = "blue",
     border = "black")
#Boxplot for Fresh Products Spending
boxplot(data$Fresh,
        main = "Boxplot of Fresh Products",
        ylab = "Fresh Products",
        col = "green")
#Scatter Plot (Fresh vs. Milk Spending)
plot(data$Fresh, data$Milk,
     main = "Scatter Plot of Fresh vs Milk",
     xlab = "Fresh Products",
     ylab = "Milk Products",
     col = "red",
     pch = 19)
```



3. Correlation Analysis (Relationships Between Variables)

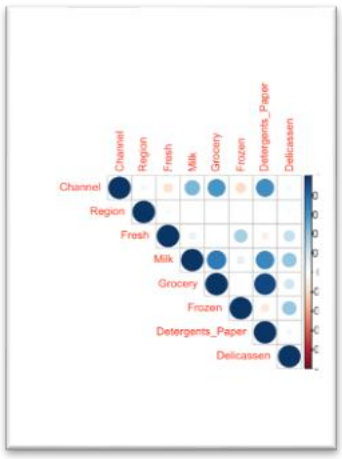
Correlation Matrix Insights:

- Grocery and Detergents_Paper have a strong positive correlation (~0.92)
  - Customers who buy a lot of Grocery products also tend to buy Detergents_Paper.
- Milk and Grocery are moderately correlated (~0.66)
  - Retail customers might buy both together.
- Fresh has little to no correlation with other products (~0.1 - 0.3)
  - Fresh spending is independent of other categories.

Correlation Visualization Insights:

- Large, dark circles indicate strong relationships (e.g., Grocery ↔ Detergents_Paper).
- Small circles indicate weak relationships (e.g., Fresh ↔ Frozen).
- Positive correlations (Blue) suggest when one variable increases, the other also increases (e.g., Grocery & Detergents_Paper).
- Negative correlations (Red) suggest when one increases, the other decreases (not found in this dataset).

```
> install.packages("corrplot")
Error in install.packages : Updating loaded packages
> #Load the corrplot Library
> library(corrplot)
> #Compute Correlation Matrix
> cor_matrix <- cor(data[, sapply(data, is.numeric)])
> print(cor_matrix)
                  Channel      Region       Fresh      Milk     Grocery      Frozen Detergents_Paper Delicassen
Channel        1.00000000  0.062027619 -0.16917204 0.4607203  0.608792245 -0.20204596      0.636026367 0.05601143
Region         0.06202762  1.000000000  0.05528692 0.0322875  0.007695777 -0.02104421     -0.001482686 0.04521211
Fresh         -0.16917204  0.055286923  1.00000000 0.1005098 -0.011853875  0.34588146     -0.101952938 0.24468997
Milk           0.46072028  0.032287502  0.10050977 1.0000000  0.728335118  0.12399376      0.661815679 0.40636832
Grocery        0.60879225  0.007695777 -0.01185387 0.7283351  1.000000000 -0.04019274      0.924640691 0.20549651
Frozen        -0.20204596 -0.021044215  0.34588146 0.1239938 -0.040192737  1.00000000     -0.131524906 0.39094747
Detergents_Paper  0.63602637 -0.001482686 -0.10195294 0.6618157  0.924640691 -0.13152491      1.000000000 0.06929130
Delicassen     0.05601143  0.045212107  0.24468997 0.4063683  0.205496511  0.39094747      0.069291297 1.00000000
> #Visualizing Correlation Matrix
> corrplot(cor_matrix,
+          method = "circle",
+          type = "upper",
```



4. Outlier Analysis

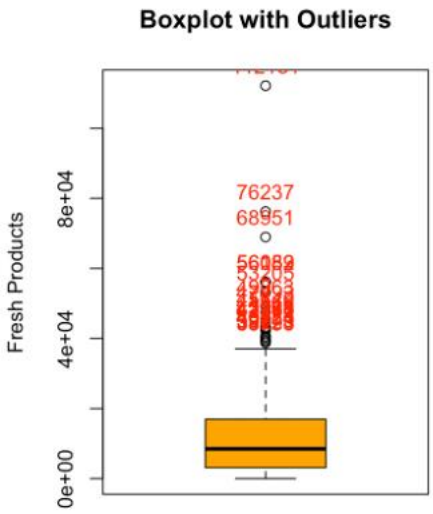
Outliers in Fresh Spending:

- Some customers spend 10 times more than the average on Fresh, making them high-value customers.

Boxplot with Outliers Insights:

- Outliers are marked in red on the boxplot, indicating unusual spending patterns.
- These outliers could be large businesses, hotels, or restaurants that require bulk purchases.

```
> #Identify Outliers in Fresh Spending
> outliers <- boxplot.stats(data$Fresh)$out
> print(outliers)
 [1]  43088  56159  44466  40721  43265  56082  76237  42312  45640 112151  47493  56083  53205  49063  68951  40254  42786  39679  38793
[20]  39228
> # Boxplot Highlighting Outliers
> boxplot(data$Fresh,
+         main = "Boxplot with Outliers",
+         ylab = "Fresh Products",
+         col = "orange")
>
> text(x = 1, y = outliers, labels = outliers, col = "red", pos = 3)
```



5. Customer Segmentation Insights

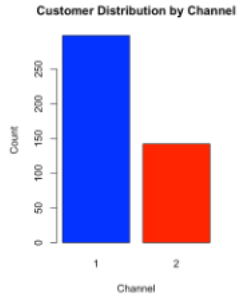
Channel Analysis:

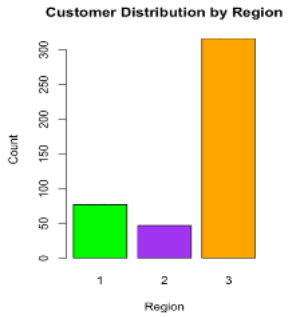
- Horeca customers (Hotels, Restaurants, Cafés) tend to spend more on Fresh products compared to Retail.
- Retail customers buy more Grocery and Detergents_Paper, indicating consumer-driven purchases.

Region Analysis:

- Spending patterns vary by region:
  - Lisbon (Region 1) may have a higher focus on Delicatessen.
  - Retail behavior in Porto (Region 2) appears different from Lisbon.

```
> # Distribution of Customers by Channel
> table(data$Channel)

  1   2
298 142
> barplot(table(data$Channel),
+         main = "Customer Distribution by Channel",
+         xlab = "Channel",
+         ylab = "Count",
+         col = c("blue", "red"))
> #Distribution of Customers by Region
> table(data$Region)

  1   2   3
 77  47 316
> barplot(table(data$Region),
+         main = "Customer Distribution by Region",
+         xlab = "Region",
+         ylab = "Count",
+         col = c("green", "purple", "orange"))
```





6. Key Business Insights & Recommendations

Marketing Strategy:

- Target Horeca customers with discounts on Fresh products since they purchase in large quantities.
- Bundle Grocery and Detergents_Paper products for Retail customers since they are highly correlated.
- Segment outlier customers for premium offers, as they are high-value clients.

Inventory & Supply Chain Management:

- Increase stock for Fresh and Milk as they have high demand.
- Manage outliers separately, ensuring the availability of products for bulk-buying customers.

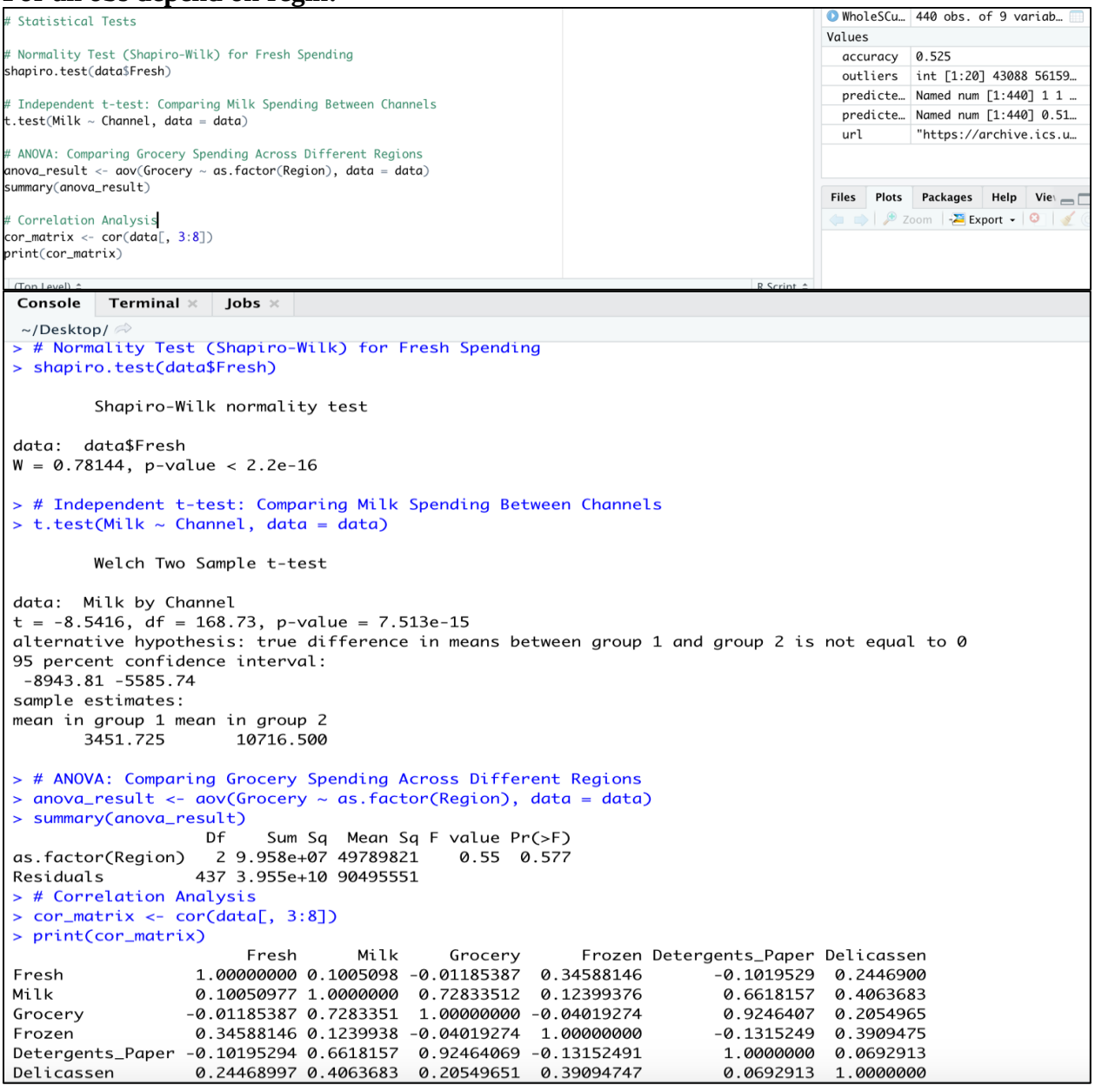
Customer Segmentation & Personalization:

- Customize promotions based on spending habits (e.g., Horeca customers might benefit from weekly bulk discounts).
- Identify new market opportunities by studying customer clusters.

Statistical Analysis

Statistical analysis is used to collect, interpret, and present data to uncover insights and patterns. We main types of statistical analysis used in R.

For all obs depend on regin:



```
# Statistical Tests

# Normality Test (Shapiro-Wilk) for Fresh Spending
shapiro.test(data$Fresh)

# Independent t-test: Comparing Milk Spending Between Channels
t.test(Milk ~ Channel, data = data)

# ANOVA: Comparing Grocery Spending Across Different Regions
anova_result <- aov(Grocery ~ as.factor(Region), data = data)
summary(anova_result)

# Correlation Analysis
cor_matrix <- cor(data[, 3:8])
print(cor_matrix)
```

```
WholeSCu… 440 obs. of 9 variab…
Values
accuracy  0.525
outliers  int [1:20] 43088 56159…
predicte… Named num [1:440] 1 1 …
predicte… Named num [1:440] 0.51…
url       "https://archive.ics.u…
```

```
Console  Terminal  Jobs
~/Desktop/
> # Normality Test (Shapiro-Wilk) for Fresh Spending
> shapiro.test(data$Fresh)

        Shapiro-Wilk normality test

data:  data$Fresh
W = 0.78144, p-value < 2.2e-16

> # Independent t-test: Comparing Milk Spending Between Channels
> t.test(Milk ~ Channel, data = data)

        Welch Two Sample t-test

data:  Milk by Channel
t = -8.5416, df = 168.73, p-value = 7.513e-15
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 -8943.81 -5585.74
sample estimates:
mean in group 1 mean in group 2
       3451.725       10716.500

> # ANOVA: Comparing Grocery Spending Across Different Regions
> anova_result <- aov(Grocery ~ as.factor(Region), data = data)
> summary(anova_result)
                   Df    Sum Sq  Mean Sq F value Pr(>F)
as.factor(Region)   2 9.958e+07 49789821    0.55  0.577
Residuals         437 3.955e+10 90495551
> # Correlation Analysis
> cor_matrix <- cor(data[, 3:8])
> print(cor_matrix)
                    Fresh      Milk     Grocery      Frozen Detergents_Paper Delicassen
Fresh            1.00000000 0.1005098 -0.01185387  0.34588146       -0.1019529  0.2446900
Milk             0.10050977 1.0000000  0.72833512  0.12399376        0.6618157  0.4063683
Grocery         -0.01185387 0.7283351  1.00000000 -0.04019274        0.9246407  0.2054965
Frozen           0.34588146 0.1239938 -0.04019274  1.00000000       -0.1315249  0.3909475
Detergents_Paper -0.10195294 0.6618157  0.92464069 -0.13152491        1.0000000  0.0692913
Delicassen       0.24468997 0.4063683  0.20549651  0.39094747        0.0692913  1.0000000
```

- **Normality Test (Shapiro-Wilk) for Fresh Spending**

W = 0.78144, p-value < 2.2e-16

The very small p-value suggests that the Fresh spending data is not normally distributed.

- **Independent t-test (Milk Spending by Channel)**

The p-value is very small (< 0.05), indicating a significant difference in Milk spending between the two customer channels.

Group 1 (Channel 1) Mean: 3451.725 and Group 2 (Channel 2) Mean: 10716.500

Interpretation: Channel 2 customers spend significantly more on Milk than Channel 1 customers.

- **NOVA Test (Grocery Spending Across Regions)**
  F = 0.55, p-value = 0.577
  Since the p-value is greater than 0.05, there is no significant difference in Grocery spending across different regions.

- **Correlation Analysis**
  Strong positive correlations:
  - Grocery & Detergents_Paper (0.9246) → Customers who spend more on grocery items also spend more on detergents and paper products.
  - Milk & Grocery (0.7283) → Higher spending on Milk correlates with higher Grocery spending.
  - Milk & Detergents_Paper (0.6618) → Customers spending on Milk also spend on Detergents/Paper.

- **Moderate correlations**
  - Delicassen & Milk (0.4063)
  - Delicassen & Frozen (0.3909)
  - Fresh & Frozen (0.3459)

- **Weak/Negative correlations:**
  - Fresh & Detergents_Paper (-0.1019) → Fresh spending does not relate to Detergents/Paper spending.
  - Grocery & Fresh (-0.0119) → Very weak correlation.

- **Linear Regression**

```r
# Linear Regression: Predicting Grocery Spending using Milk and Fresh
lm_model <- lm(Grocery ~ Milk + Fresh, data = data)
summary(lm_model)

# Logistic Regression: Predicting Channel based on Spending Patterns
data$Channel <- as.factor(data$Channel)  # Ensure Channel is a factor
log_model <- glm(Channel ~ Fresh + Milk + Grocery + Frozen + Detergents_Paper + Delicassen,
                 data = data, family = binomial)
summary(log_model)
```

```
> # Linear Regression: Predicting Grocery Spending using Milk and Fresh
> lm_model <- lm(Grocery ~ Milk + Fresh, data = data)
> summary(lm_model)

Call:
lm(formula = Grocery ~ Milk + Fresh, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-39377  -2673  -1050   1713  59900

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3225.74833  476.70142   6.767 4.25e-11 ***
Milk           0.94894    0.04209  22.546  < 2e-16 ***
Fresh         -0.06457    0.02456  -2.629  0.00887 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6476 on 437 degrees of freedom
Multiple R-squared:  0.5378,    Adjusted R-squared:  0.5357
F-statistic: 254.2 on 2 and 437 DF,  p-value: < 2.2e-16
```

Linear Regression Interpretation

- Milk spending is a strong predictor of Grocery spending, while Fresh spending has a small negative impact.
- Milk spending is a strong predictor of Grocery spending ($p < 0.001$), suggesting that customers who spend more on Milk tend to also spend more on Grocery items

- Fresh spending has a small but significant negative impact on Grocery spending ($p < 0.01$).
- The model explains about 53.78% of the variance in Grocery spending (Adjusted R-squared = 0.5357).

Interpretation of Logistic Regression

- Detergents_Paper is the most important factor in determining customer Channel.
- Grocery spending also plays a role, but other spending variables are less relevant.
- Detergents_Paper spending is the strongest predictor of Channel assignment ($p < 0.001$).
- Grocery spending also influences Channel assignment ($p < 0.05$), while other predictors are not statistically significant.
- The model significantly reduces deviance, indicating a good fit for predicting customer channels.

A specific statistical study focusing on the relationship between milk and region

To perform an ANOVA test to analyze the relationship between Region (categorical) and Milk (numerical)

1- Load Data

```
#import data into Rstudio
WholeSCutmr<-read.csv(file.choose(),header = TRUE)
View(WholeSCutmr)
str(WholeSCutmr)

# Convert Region to a categorical variable
WholeSCutmr$Region <- as.factor(WholeSCutmr$Region)

# Display summary of the dataset
summary(WholeSCutmr)
```

```
> str(WholeSCutmr)
'data.frame':   440 obs. of  8 variables:
 $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
 $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
 $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...
 $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...
 $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...
 $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 1159 ...
 $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...
 $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2098 ...
> # Convert Region to a categorical variable
> WholeSCutmr$Region <- as.factor(WholeSCutmr$Region)
> # Display summary of the dataset
> summary(WholeSCutmr)
    Channel      Region      Fresh             Milk          Grocery
 Min.   :1.000   1: 77   Min.   :     3   Min.   :   55   Min.   :    3
 1st Qu.:1.000   2: 47   1st Qu.:  3128   1st Qu.: 1533   1st Qu.: 2153
 Median :1.000   3:316   Median :  8504   Median : 3627   Median : 4756
 Mean   :1.323           Mean   : 12000   Mean   : 5796   Mean   : 7951
 3rd Qu.:2.000           3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656
 Max.   :2.000           Max.   :112151   Max.   :73498   Max.   :92780
     Frozen        Detergents_Paper    Delicassen
 Min.   :   25.0   Min.   :    3.0   Min.   :    3.0
 1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2
 Median : 1526.0   Median :  816.5   Median :  965.5
 Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9
 3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2
 Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
```

2. Then Perform ANOVA Test (Parametric test)

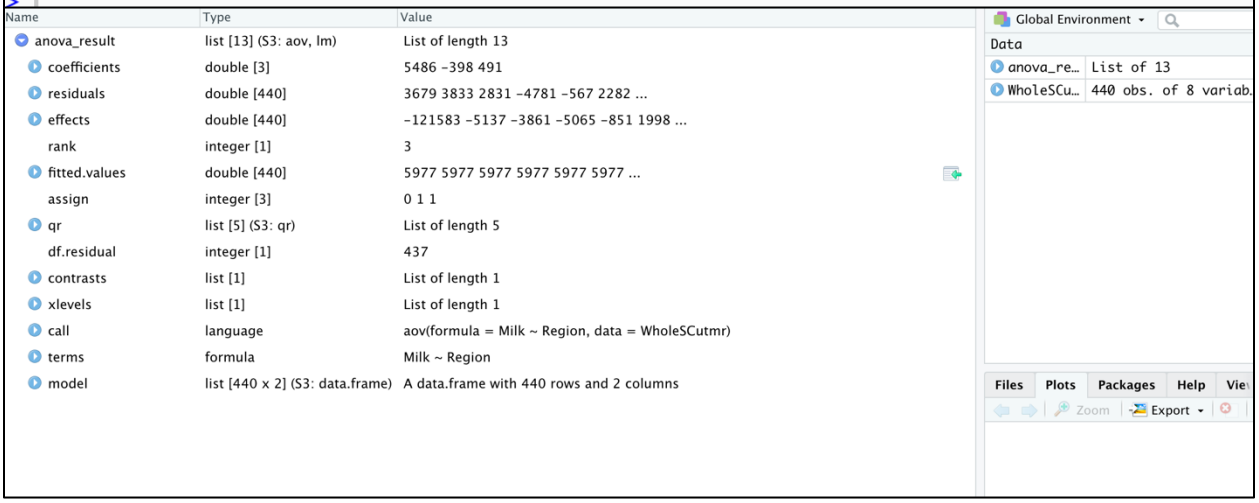
#perform ANOVA to compare Milk consumption across different regions

anova_result <- aov(Milk ~ Region, data = data)

#Display ANOVA results

summary(anova_result)

```
> # Perform ANOVA to compare Milk consumption across different regions
> anova_result <- aov(Milk ~ Region, data = WholeSCutmr)
> # Display ANOVA results
> summary(anova_result)
             Df    Sum Sq  Mean Sq F value Pr(>F)
Region        2 4.129e+07 20645088   0.378  0.685
Residuals   437 2.387e+10 54624772
>
```



What Each Column Means

- Df (Degrees of Freedom):
  - Region: 2 (Since there are three regions, the degrees of freedom = number of groups - 1)
  - Residuals: 437 (Total observations - number of groups)
- Sum Sq (Sum of Squares):
  - **Region:** *4.129×1074.129 |times 10^7*4.129×107 → Variability due to different regions.
  - **Residuals:** *2.387×10102.387 |times 10^{10}*2.387×1010 → Variability within groups (error term).
- **Mean Sq (Mean Squares)**:
  - **Region:** *2064508820645088*20645088 → Sum of Squares divided by its Degrees of Freedom.
  - **Residuals:** *5462477254624772*54624772.
- **F value**: **0.378**
  - The F-statistic tests whether the means of **Milk** consumption differ significantly across regions.
- **Pr(>F) (p-value)**: **0.685**
  - This is the probability of getting an F-value of 0.378 if the null hypothesis (no difference between regions) is true.

Insight depends on Anova-test :

- Since p-value = 0.685 is much greater than 0.05, we fail to reject the null hypothesis.
- This means there is no statistically significant difference in Milk consumption between the different regions.

♦ **Interpreting the Kruskal-Wallis Test**

kruskal.test(Milk ~ Region, data = WholeSCutmr)

```
> kruskal.test(Milk ~ Region, data = WholeSCutmr)

        Kruskal-Wallis rank sum test

data:  Milk by Region
Kruskal-Wallis chi-squared = 1.4606, df = 2, p-value = 0.4818
```

Insight depends on Kruskal-Wallis Test:

- The p-value (0.4818) is greater than 0.05, so we fail to reject the null hypothesis (H_0).
- This means there is no statistically significant difference in Milk consumption between regions.

Interpreting the Shapiro-Wilk Normality Test

shapiro.test(WholeSCutmr$Milk) # Normality test

```
> shapiro.test(WholeSCutmr$Milk)   # Normality test

        Shapiro-Wilk normality test

data:  WholeSCutmr$Milk
W = 0.62833, p-value < 2.2e-16
```

Insight depends on Shapiro-Wilk Normality Test:

- Since p-value < 0.05, we reject the null hypothesis (H_0) that the data is normally distributed.
- This confirms that Milk consumption does not follow a normal distribution
- Because the ANOVA test assumes normality, and we now know the data is not normally distributed, ANOVA may not be appropriate.

Use the Welch's ANOVA , which does not assume equal variances

oneway.test(Milk ~ Region, data = WholeSCutmr, var.equal = FALSE)

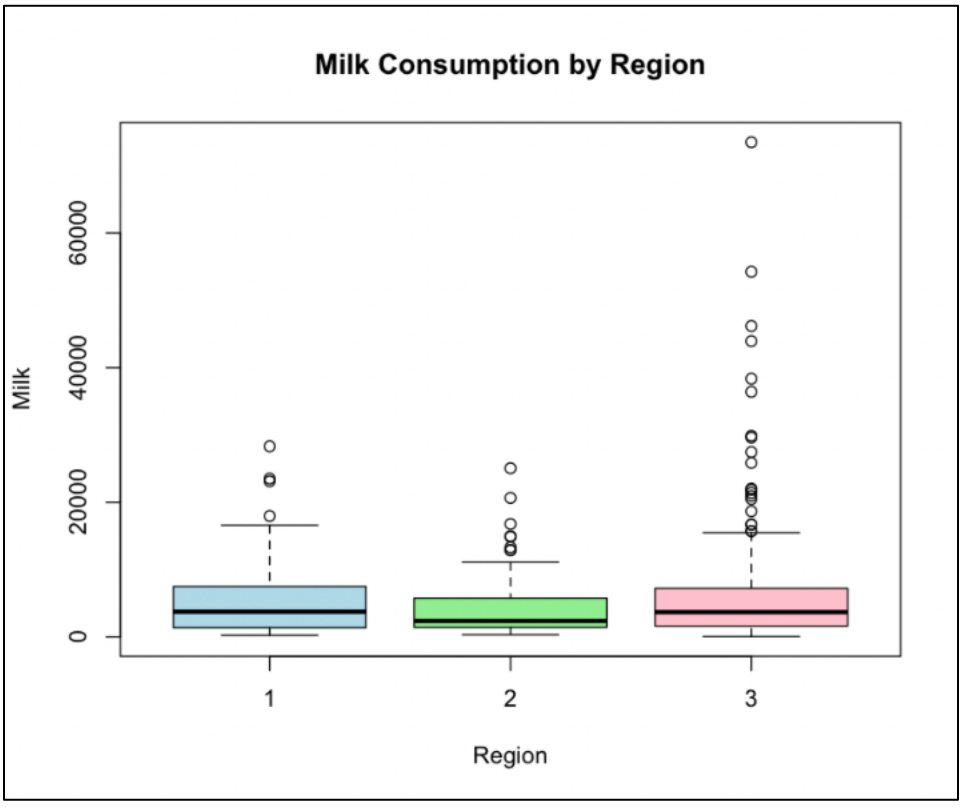
```
> oneway.test(Milk ~ Region, data = WholeSCutmr, var.equal = FALSE)

        One-way analysis of means (not assuming equal variances)

data:  Milk and Region
F = 0.50277, num df = 2.00, denom df = 115.88, p-value = 0.6062
```

Insight depends on Welch's ANOVA:

Welch's ANOVA p-value = 0.6062, meaning no significant difference in Milk consumption across regions

Use boxplot we use to prove what we have concluded:

boxplot(Milk ~ Region, data = WholeSCutmr, main = "Milk Consumption by Region", col = c("lightblue", "lightgreen", "pink"))

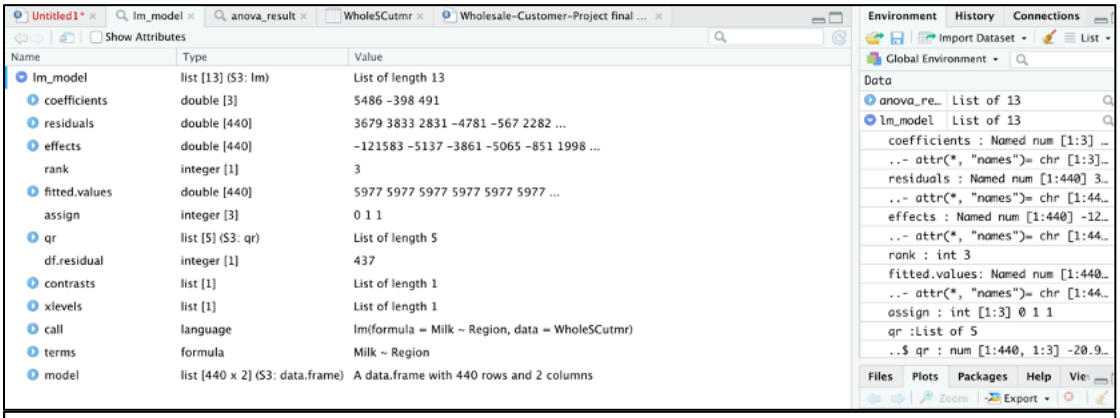


Conclusion:
There is no significant difference in milk consumption between regions., based on both Welch's ANOVA and Kruskal-Wallis.

Implement linear or logistic regression and interpret key results:

Since Milk is a continuous variable, we will use Linear Regression

```
# Load necessary library
install.packages("ggplot2")
library(ggplot2)

# Fit linear regression model
lm_model <- lm(Milk ~ Region, data = WholeSCutmr)

# Display summary of regression model
summary(lm_model)
```




```
Call:
lm(formula = Milk ~ Region, data = WholeSCutmr)

Residuals:
   Min     1Q Median     3Q    Max
 -5922  -4154  -2236   1300  67521

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5486.4      842.3   6.514 2.02e-10 ***
Region2       -398.2     1368.1  -0.291    0.771
Region3        490.7      939.3   0.522    0.602
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7391 on 437 degrees of freedom
Multiple R-squared:  0.001727,  Adjusted R-squared:  -0.002842
F-statistic: 0.3779 on 2 and 437 DF,  p-value: 0.6855
```

Result depends on reign:

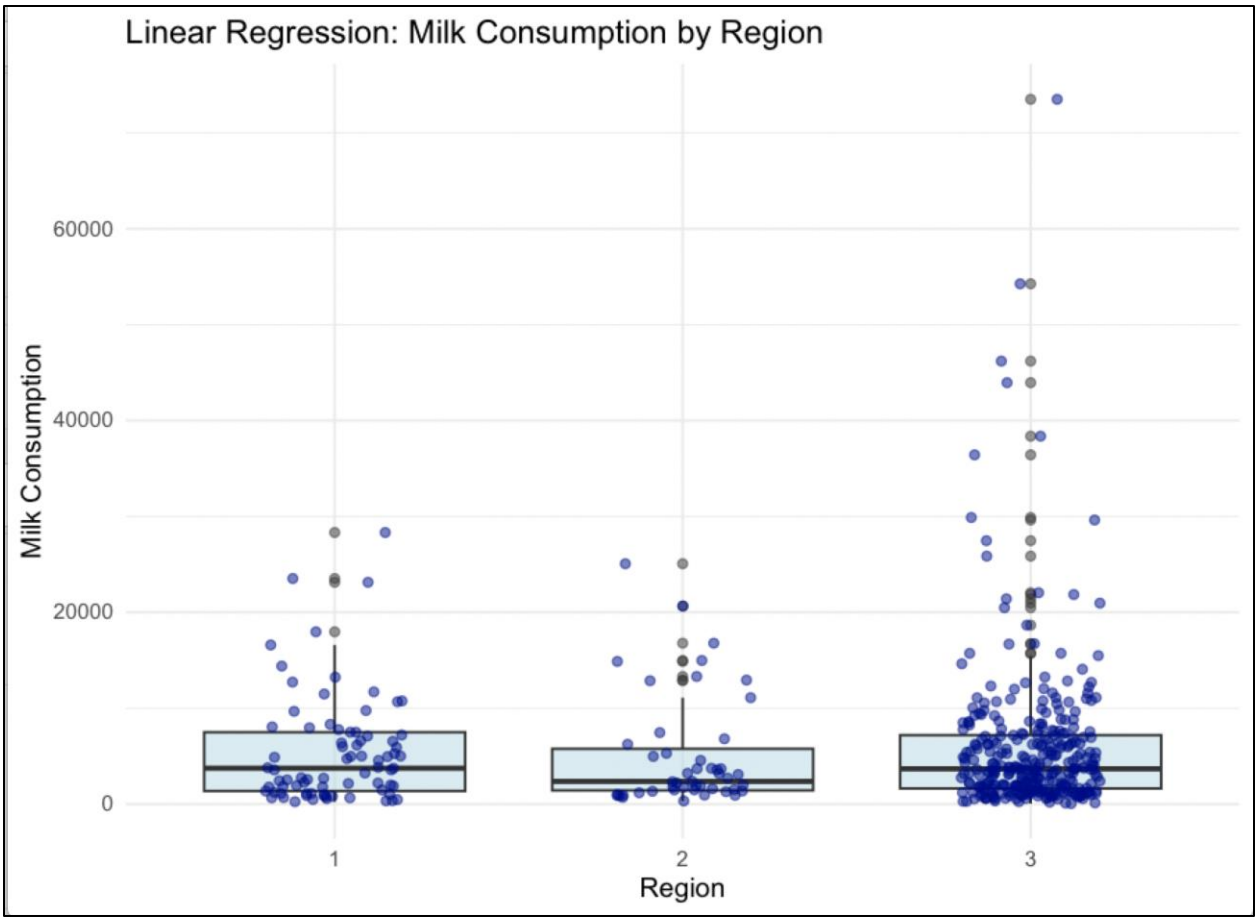
Intercept (5486.4). >	This is the estimated **Milk consumption** for **Region 1**.

Region2 (-398.2)	Customers in **Region 2** consume about **398.2 units less** than those in Region 1, but this difference is **not statistically significant (p = 0.771).**

Region3 (490.7)	Customers in **Region 3** consume about **490.7 units more** than those in Region 1, but again, this difference is **not statistically significant (p = 0.602).**

Linear Regression Plot in R

```
install.packages("ggplot2", dependencies=TRUE)

library(ggplot2)

ggplot(WholeSCutmr, aes(x = as.factor(Region), y = Milk)) + geom_boxplot(fill = "lightblue", alpha = 0.5) +

 geom_jitter(color = "darkblue", alpha = 0.5, width = 0.2) +

 geom_smooth(method = "lm", color = "red", se = FALSE) + # labs(title = "Linear Regression: Milk Consumption by Region", x = "Region", y = "Milk Consumption") + theme_minimal()
```



Distribution is similar in all regions:

This graph shows the relationship between milk consumption and regions.

It includes:

- A boxplot (light blue boxes) to show the distribution of milk consumption data for each region.
- A scatter plot, where each point represents an individual or consumer.

Outliers are points that fall far from the box.

- Milk consumption seems to be similar between the three regions (1, 2, 3).
- The average milk consumption does not differ significantly between regions.

Conclusions depend on Linear Regression:

- Region is not a significant predictor of Milk consumption.
- The model explains almost no variance ($R^2 = 0.0017$).
- p-values for Region2 and Region3 > 0.05, meaning no evidence that different regions consume significantly different amounts of Milk.
- The ANOVA, Kruskal-Wallis, and Welch's ANOVA already confirmed this, and now regression confirms it too.

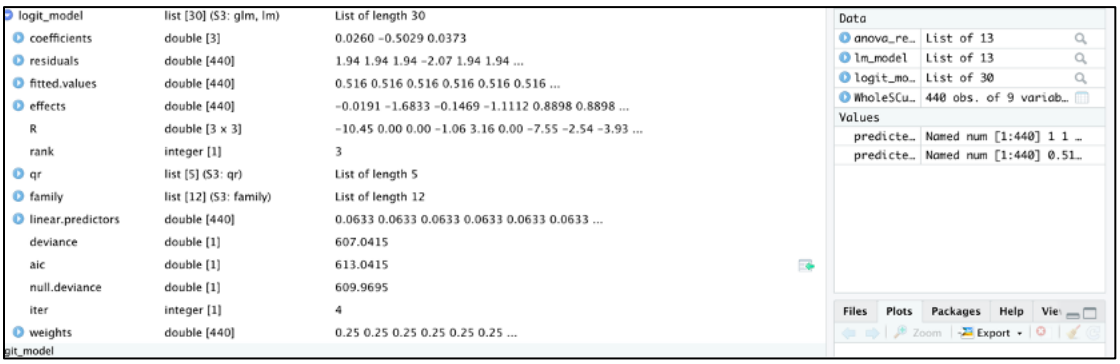
Implementing Logistic Regression

Since Milk consumption is a continuous variable, we first need to convert it into a binary variable for logistic regression.

We will:

1. **Create a binary target variable** (High vs. Low Milk consumption).
2. **Fit a logistic regression model** to predict **high consumption based on region**.
3. **Interpret key results**.

```
# Create a binary variable for Milk consumption
WholeSCutmr$HighMilk <- ifelse(WholeSCutmr$Milk > median(WholeSCutmr$Milk), 1, 0)

# Convert to a factor for logistic regression
WholeSCutmr$HighMilk <- as.factor(WholeSCutmr$HighMilk)

# Check distribution of High vs. Low
table(WholeSCutmr$HighMilk)


# Fit the logistic regression model
logit_model <- glm(HighMilk ~ Region, data = WholeSCutmr, family = binomial)

# Display model summary
summary(logit_model)
```



```
> # Check distribution of High vs. Low
> table(WholeSCutmr$HighMilk)

  0   1
220 220
> # Fit the logistic regression model
> logit_model <- glm(HighMilk ~ Region, data = WholeSCutmr, family = binomi
> # Display model summary
> summary(logit_model)

Call:
glm(formula = HighMilk ~ Region, family = binomial, data = WholeSCutmr)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.20441  -1.20441   0.08397   1.15065   1.38548

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.02598    0.22794   0.114    0.909
Region2     -0.50290    0.37682  -1.335    0.182
Region3      0.03734    0.25422   0.147    0.883

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 609.97  on 439  degrees of freedom
Residual deviance: 607.04  on 437  degrees of freedom
AIC: 613.04

Number of Fisher Scoring iterations: 4
```

Insghit :

Since the data is balanced, logistic regression should not suffer from class imbalance issues. we make evaluate the model performance by creating a confusion matrix and checking accuracy.

```
exp(coef(logit_model))  # Get odds ratios

# Make predictions
predicted_probs <- predict(logit_model, type = "response")
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)

# Create a confusion matrix
table(Predicted = predicted_classes, Actual = WholeSCutmr$HighMilk)
accuracy <- (29 + 202) / (29 + 202 + 191 + 18)
print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))
```

```
> table(Predicted = predicted_classes, Actual = WholeSCutmr$HighMilk)
         Actual
Predicted   0   1
        0  29  18
        1 191 202
> View(logit_model)
> View(logit_model)
> accuracy <- (29 + 202) / (29 + 202 + 191 + 18)
> print(paste("Model Accuracy:", round(accuracy * 100, 2), "%"))
[1] "Model Accuracy: 52.5 %"
>
```

Safari

Conclusion:

- Very Low Accuracy (52.5%) → Model is not effective at predicting milk consumption trends.
- High False Positive Rate (FP = 191) → The model incorrectly predicts High Milk too often.
- Low Specificity (13.2%) → The model fails to correctly identify Low Milk consumers.
- High Recall (91.1%) → The model is good at catching High Milk consumers but makes many mistakes.
- Accuracy (52.5%) means the model correctly predicts High or Low Milk consumption only 52.5% of the time.
- Since random guessing would give about 50% accuracy (because the classes are balanced 220 vs. 220), this means the model is only slightly better than random guessing.
- When we want use logistic regression must be use mor than obs.

Machine Learning Model

A very important part of data analysis is Machine Learning that can be performed on R. Data is first given, then the algorithms learn from its patterns, therefore it will generate predictions and will aid in decision making. Many tools are there on R to satisfy this approach. It is highly beneficial for statistical and analytical purposes. For this report, a random forest model was used.

Why Random Forest Model is used:

The ability of this model to handle large datasets, flexibility and durability that this model poses and it being suitable for both regression and classification purposes, make it a go-to choice. In addition to that it is less prone to overfitting for the following reason: random forest builds multiple decision trees for training then it combines them, therefore, optimizing accuracy. It handles the complex relationships within data. Another important feature this model poses is that it enables feature importance, therefore giving an insight about which tells which variable has the power on predictions. Sometimes datasets can have a mix of numerical and categorical data, and non-linear relationships, this model handles them well without extended parameter tuning.

In the following there is a step-by-step R coding and explanation of each step, for building and evaluating a predictive model using the appropriate ML technique.

Install Necessary Packages:

- Installing required packages (randomForest, caret, dplyr, ggplot2, pROC, and tidyverse).
- Which packages are not yet installed? The code commands R to install them.

Load Libraries:

- After the installation of necessary packages, the code loads them into the R console using library() in order to benefit from the functions, they provide for random forest model. Like how to manipulate data to get a cleaner and more readable code using (dplyr, tidyverse), visualization of the results with (ggplot2), then for model evaluation (caret, pROC).

```
# Install necessary packages (only if they are not already installed)
packages <- c("randomForest", "caret", "dplyr", "ggplot2", "pROC", "tidyverse")

# Install missing packages
```

```
new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
if(length(new_packages)) install.packages(new_packages)

# Load libraries
library(randomForest)
library(caret)
library(dplyr)
library(ggplot2)
library(pROC)
library(tidyverse)
```

Load Data:

- The dataset is a CSV file ("Wholesale customers data.csv") from the path ("C:/Users/norah/Desktop/") the code loads it into a data frame with the name data.
- Now, to import the dataset into R read.csv() function is used.

```
# Load data
data <- read.csv("C:/Users/norah/Desktop/Wholesale customers data.csv")
```

Data Preprocessing:

1. Convert Region to Factor:
    - Regions represent different categories. Classification models need the conversion of region column to a categorical variable using data$Region <- as.factor(data$Region).
2. Handle Missing Values:
    - the solution to missing numerical values here is by using the mean of the corresponding column instead of them using mutate(across(where(is.numeric), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
    - then applying this to all of the numerical columns by across(where(is.numeric))
    - the following code will check values with NA (missing) then replaces all missing values with the columns mean, this way the NA values are eliminated by ifelse(is.na(.), mean(., na.rm = TRUE), .)

```
# Data Preprocessing

data$Region <- as.factor(data$Region)  # Convert Region to factor

data <- data %>%

  mutate(across(where(is.numeric), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))  # Handle missing values
```

Split into Training and Test Sets:

- When R handles random tasks it needs to ensure reproducibility that is when the same data is run with the same code the exact results will show so using set.seed(123) is beneficial for multiple run times.
- First splitting the dataset in to parts and assigning 70% of data for training and 30% of it for test set using createDataPartition(data$Region, p = 0.7, list = FALSE) based on the variable region.
- trainData <- data[trainIndex,] rows selection for the training set using the trainIndex.
- testData <- data[-trainIndex,] selection for the unselected rows for the purpose of test set by eliminating the trainIndex.

```
# Split into training and test sets

set.seed(123)

trainIndex <- createDataPartition(data$Region, p = 0.7, list = FALSE)

trainData <- data[trainIndex, ]

testData <- data[-trainIndex, ]
```

Train Random Forest Model:

Next, introducing random forest model and training it using randomForest(Region ~ Channel + Fresh + Milk + Grocery + Frozen + Detergents_Paper + Delicassen, data = trainData, ntree = 500, importance = TRUE) commanding it to predict the variable reigion based on other variables (Channel, Fresh, Milk, Grocery, Frozen, Detergents_Paper, and Delicassen) from (trainData).

- ntree = 500 the number of trees to be built in the forest.
- importance = TRUE assessing the importance of each variable and storing it for the goal of making predicions.

Model Summary:

- print(rf_model) yields model summary and monitors model performance with error rates and variable importance.

```
# Train Random Forest Model
rf_model <- randomForest(Region ~ Channel + Fresh + Milk + Grocery + Frozen +
Detergents_Paper + Delicassen,
                         data = trainData, ntree = 500, importance = TRUE)


# Print the model summary
print(rf_model)
```

Make Predictions on the Test Set:

- after training random forest model (rf_model), the code predictions <- predict(rf_model, testData) will use it for making predictions on (testData).
- to yield the predicted Region for each observation in the test set use predict() function, the result will be stored in the predictions variable.

```
# Make predictions on the test set
predictions <- predict(rf_model, testData)
```

Confusion Matrix:

- confusion matrix calculation by conf_matrix <- confusionMatrix(predictions, testData$Region) for the evaluation of the model's performance.
- Then, comparison between predictions and (testData$Region) that's the real values then yielding a table representing how many matched predictions and how many are misclassified for each region using confusionMatrix() function. Then applying model performance metrics like accuracy, sensitivity, specificity, and precision.
- After that R displays confusion matrix and the calculated evaluation metrics using print(conf_matrix).

```
# Confusion Matrix
conf_matrix <- confusionMatrix(predictions, testData$Region)
print(conf_matrix)
```

Feature Importance:

Now we need to assess each predictor variable's importance by importance(rf_model). Its working is based on their contribution to the accuracy of the model, the higher the value, the more important it is.

- varImpPlot(rf_model) visualization of predictor variable importance, presenting which predictor variable has more effect on predictions of the target variable (Region). Then showing a bar plot, the longer the bar is, the more important the corresponding variable is.

```
# Feature importance

importance(rf_model)

varImpPlot(rf_model)
```

Accuracy Calculation:

Accuracy is measured by by the comparison between predictions and real values in test data. It is calculated by the number of correct preditions divided by total number of test cases.

predictions == testData$Region creates a true\false list showing which are correct by comparing them to test data and making sure they're the same.

Then sum(predictions == testData$Region) will give the number of correct predictions.

nrow(testData) shows the total number of observations in the test set.

then printed the accuracy value that assesses model performance.

```
# Calculate accuracy

accuracy <- sum(predictions == testData$Region) / nrow(testData)

print(paste("Accuracy: ", accuracy))
```

Cross-validation:

More assessments of the model performance using cross-validation that's 5-fold

the code trainControl(method = "cv", number = 5) specifies that data is split into 5 subsets for validation and setting up the cross-validation method.

train(Region ~ ., data = trainData, method = "rf", trControl = control, tuneLength = 5) trains the Random Forest model using the train() function from caret package. Training the model multiple times on different subsets to give more reliable measurements of how well the model performs.

Region ~ . Region is the target variable, the remaining columns in trainData are predictors.

method = "rf" commands to use the Random Forest algorithm.

trControl = control commanding 5-fold cross-validation.

tuneLength = 5 means 5 different settings for the model's parameters used for testing to target the best combination for the most optimum performance.

Perform cross-validation

control <- trainControl(method = "cv", number = 5) # 5-fold CV

rf_cv_model <- train(Region ~ ., data = trainData, method = "rf", trControl = control, tuneLength = 5)

Hyperparameter Tuning:

now we need a tuning grid for the Random Forest model's hyperparameter mtry.

Randomly selected variables from each split in the model, their number is represented with mtry.

Next, creating values' data frame for mtry for subsequent testing by expand.grid() function. Mtry values can be specified for test in the grid: 2, 4, 6, 8, and 10. Now we have a grid of mtry values, the code chooses candidates for this hyperparameter to be evaluated during model training. Yielding the best value for optimal model performance.

Hyperparameter tuning

Define the tuning grid

tuneGrid <- expand.grid(mtry = c(2, 4, 6, 8, 10))

Training the Random Forest Model with the Tuning Grid:

Now we will use cross-validation and the hyperparameter tuning grid (mtry values) to train our model using train() function from caret package to optimize the model's hyperparameters which in return gives better performance.

Region ~ . making predictions of the Region variable based on all other variables in the dataset (represented by .).

method = "rf" showing that we're currently using random forest.

trControl = control is for the 5-fold cross-validation.

To choose the grid values for mtry hyperparameter for subsequent testing using tuneGrid = tuneGrid. Then printing the results of the tuned model, showing the best value for mtry that was selected after cross-validation and model training.

```
# Train the random forest model with the tuning grid
rf_tuned <- train(Region ~ ., data = trainData, method = "rf", trControl = control, tuneGrid = tuneGrid)


# Print the results of the tuned model
print(rf_tuned)
```

More Performance Metrics:

In addition to accuracy, Precision, Recall, ROC Curve, AUC Score and F1 Score are used for performance evaluation, for more concise understanding of model performance showing how well the model identifies each class and also balancing errors.

In the case of multiclass, use the code multiclass.roc() which will transform it to a one-vs-all approach for each class.

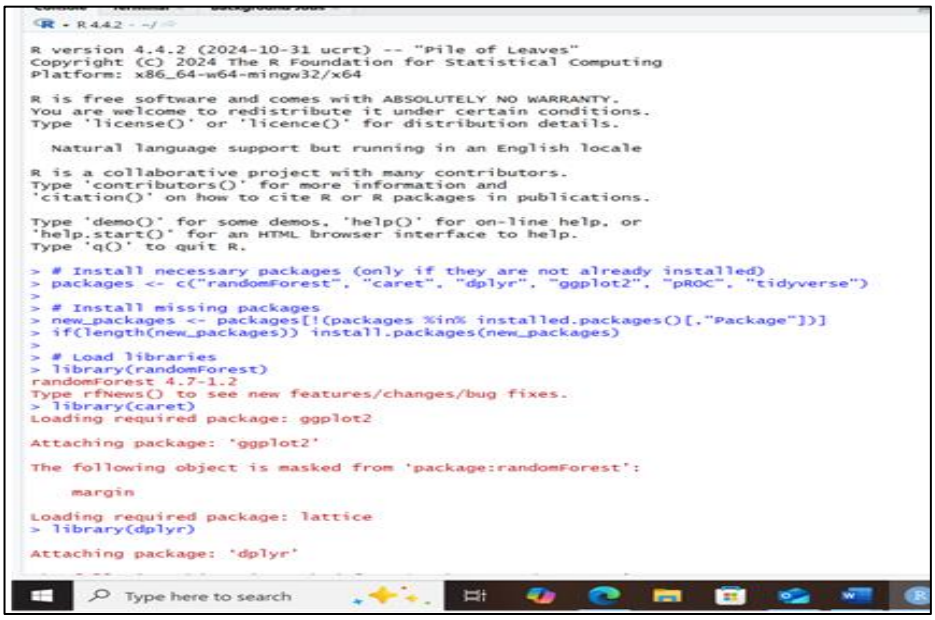
Confusion Matrix Table:

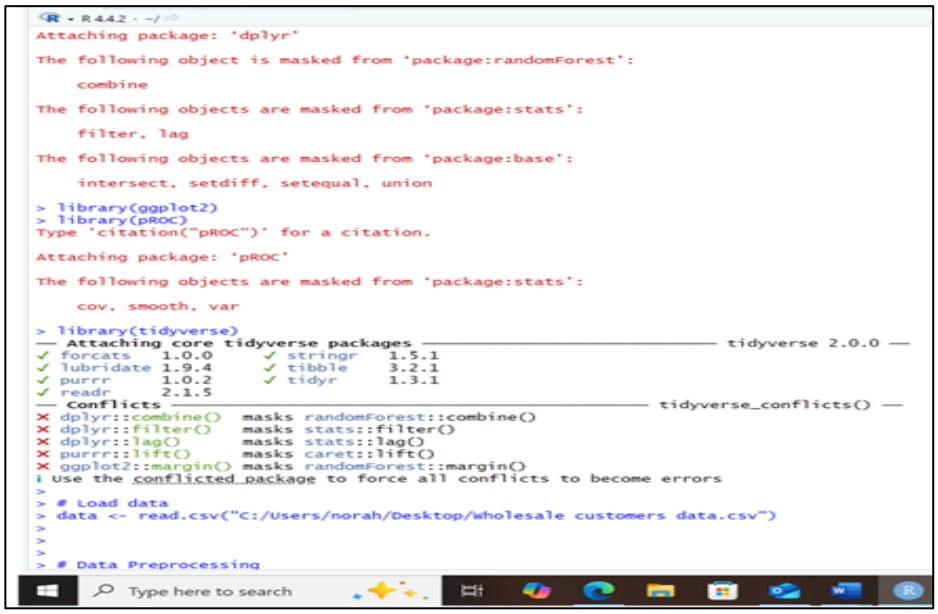
The confusion matrix (conf_matrix$table) calculate precision, recall, and F1 score for each class. Then calculating again the Precision, Recall, and F1 Score.

```
# More Performance Metrics (Beyond Accuracy)


# Extract precision from confusion matrix
precision <- conf_matrix$byClass["Pos Pred Value"]
# Extract recall (sensitivity) from confusion matrix
recall <- conf_matrix$byClass["Sensitivity"]
# Compute F1 score
f1_score <- 2 * (precision * recall) / (precision + recall)


# Print the results
print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("F1 Score:", f1_score))
```

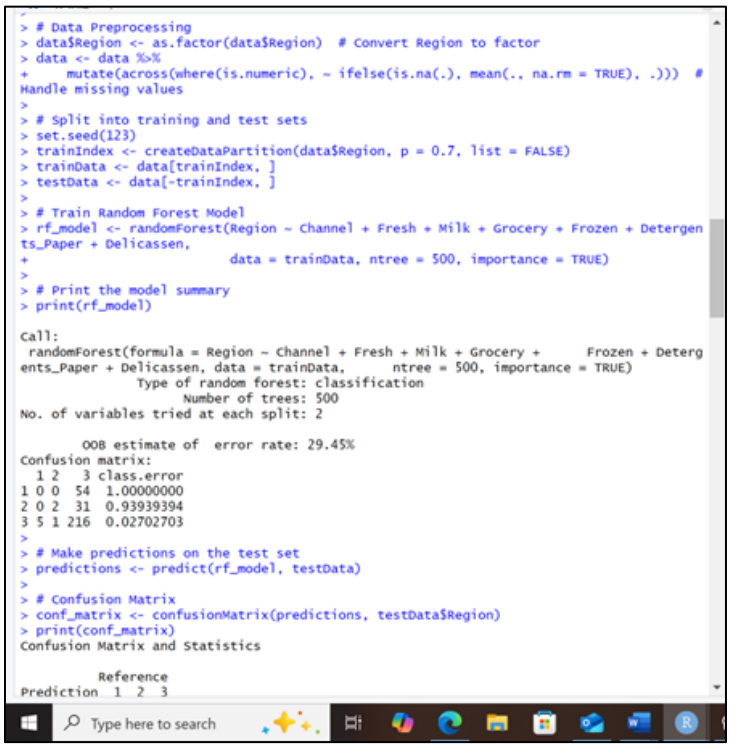
```
# ROC Curve & AUC Score (For Binary Classification or One-vs-All Multiclass)
# If Region has more than 2 categories, convert it into one-vs-all to plot an ROC curve.
roc_curve <- multiclass.roc(testData$Region, as.numeric(predictions))

# Print the AUC score
print(auc(roc_curve))

# Extract confusion matrix table
cm <- conf_matrix$table

# Calculate Precision, Recall, and F1-Score:
# Precision:
precision <- diag(cm) / colSums(cm)
print(precision)

# Recall:
recall <- diag(cm) / rowSums(cm)
print(recall)

# F1-Score:
f1_score <- 2 * (precision * recall) / (precision + recall)
print(f1_score)
```

Results:

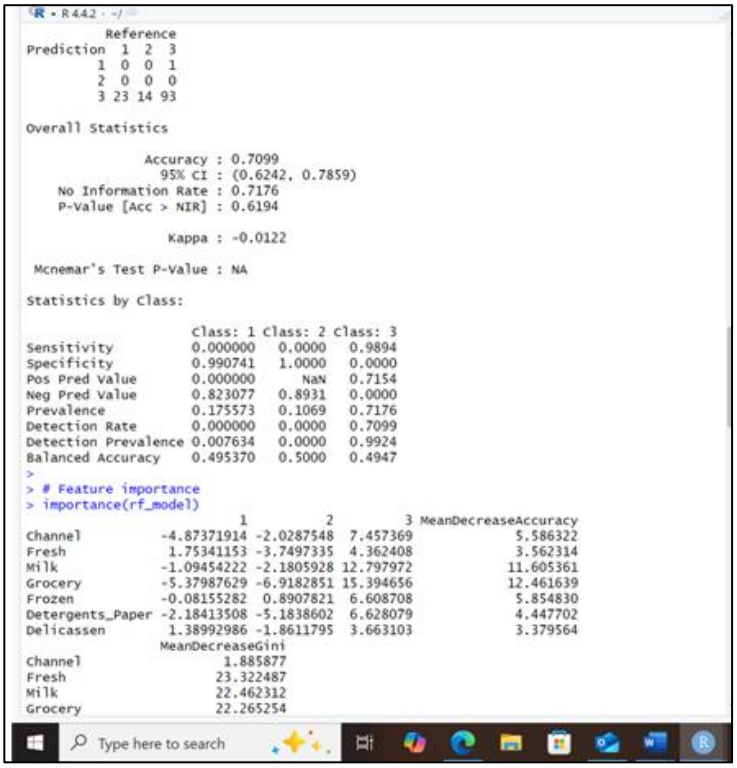

R 4.4.2 · ~/
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> # Install necessary packages (only if they are not already installed)
> packages <- c("randomForest", "caret", "dplyr", "ggplot2", "pROC", "tidyverse")
>
> # Install missing packages
> new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
> if(length(new_packages)) install.packages(new_packages)
>
> # Load libraries
> library(randomForest)
randomForest 4.7-1.2
Type rfNews() to see new features/changes/bug fixes.
> library(caret)
Loading required package: ggplot2
Attaching package: 'ggplot2'
The following object is masked from 'package:randomForest':
margin
Loading required package: lattice
> library(dplyr)
Attaching package: 'dplyr'
Type here to search

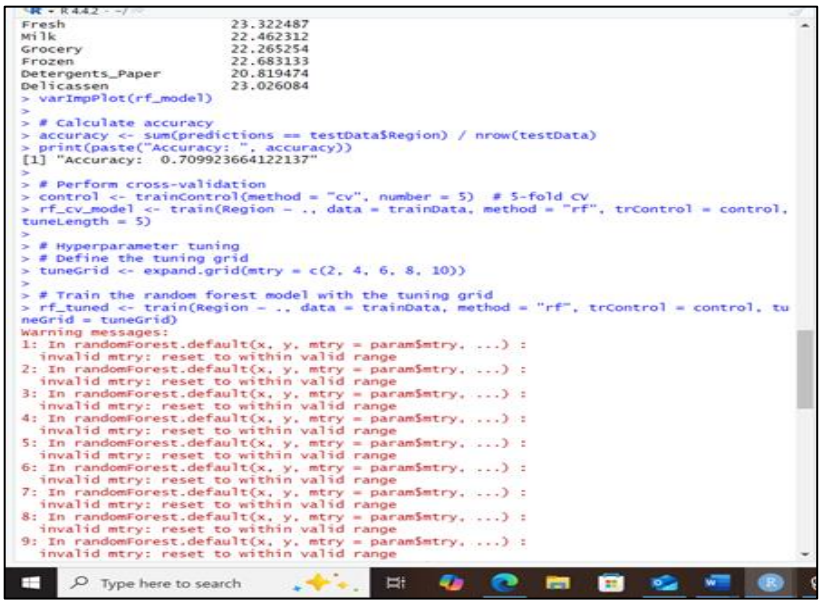

R 4.4.2 · ~/
Attaching package: 'dplyr'
The following object is masked from 'package:randomForest':
combine
The following objects are masked from 'package:stats':
filter, lag
The following objects are masked from 'package:base':
intersect, setdiff, setequal, union
> library(ggplot2)
> library(pROC)
Type 'citation("pROC")' for a citation.
Attaching package: 'pROC'
The following objects are masked from 'package:stats':
cov, smooth, var
> library(tidyverse)
── Attaching core tidyverse packages ──── tidyverse 2.0.0 ──
✓ forcats 1.0.0 ✓ stringr 1.5.1
✓ lubridate 1.9.4 ✓ tibble 3.2.1
✓ purrr 1.0.2 ✓ tidyr 1.3.1
✓ readr 2.1.5
── Conflicts ──── tidyverse_conflicts() ──
✖ dplyr::combine() masks randomForest::combine()
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag() masks stats::lag()
✖ purrr::lift() masks caret::lift()
✖ ggplot2::margin() masks randomForest::margin()
ℹ Use the conflicted package to force all conflicts to become errors
> # Load data
> data <- read.csv("C:/Users/norah/Desktop/wholesale customers data.csv")
> # Data Preprocessing
Type here to search

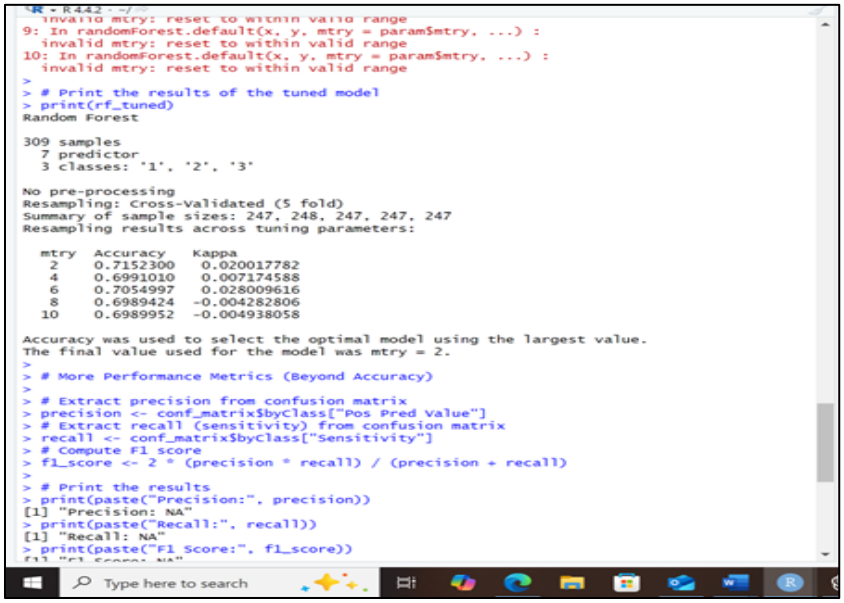
```
> # Data Preprocessing
> data$Region <- as.factor(data$Region)  # Convert Region to factor
> data <- data %>%
+     mutate(across(where(is.numeric), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))  #
Handle missing values
>
> # Split into training and test sets
> set.seed(123)
> trainIndex <- createDataPartition(data$Region, p = 0.7, list = FALSE)
> trainData <- data[trainIndex, ]
> testData <- data[-trainIndex, ]
>
> # Train Random Forest Model
> rf_model <- randomForest(Region ~ Channel + Fresh + Milk + Grocery + Frozen + Detergen
ts_Paper + Delicassen,
+                          data = trainData, ntree = 500, importance = TRUE)
>
> # Print the model summary
> print(rf_model)

Call:
 randomForest(formula = Region ~ Channel + Fresh + Milk + Grocery +      Frozen + Deterg
ents_Paper + Delicassen, data = trainData,      ntree = 500, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 29.45%
Confusion matrix:
  1 2   3 class.error
1 0 0  54  1.00000000
2 0 2  31  0.93939394
3 5 1 216  0.02702703
>
> # Make predictions on the test set
> predictions <- predict(rf_model, testData)
>
> # Confusion Matrix
> conf_matrix <- confusionMatrix(predictions, testData$Region)
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
```



```
          Reference
Prediction  1  2  3
         1  0  0  1
         2  0  0  0
         3 23 14 93

Overall Statistics

               Accuracy : 0.7099
                 95% CI : (0.6242, 0.7859)
    No Information Rate : 0.7176
    P-Value [Acc > NIR] : 0.6194

                  Kappa : -0.0122

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity          0.000000   0.0000   0.9894
Specificity          0.990741   1.0000   0.0000
Pos Pred Value       0.000000      NaN   0.7154
Neg Pred Value       0.823077   0.8931   0.0000
Prevalence           0.175573   0.1069   0.7176
Detection Rate       0.000000   0.0000   0.7099
Detection Prevalence 0.007634   0.0000   0.9924
Balanced Accuracy    0.495370   0.5000   0.4947
>
> # Feature importance
> importance(rf_model)
                                1          2         3 MeanDecreaseAccuracy
Channel           -4.87371914 -2.0287548  7.457369              5.586322
Fresh              1.75341153 -3.7497335  4.362408              3.562314
Milk              -1.09454222 -2.1805928 12.797972             11.605361
Grocery           -5.37987629 -6.9182851 15.394656             12.461639
Frozen            -0.08155282  0.8907821  6.608708              5.854830
Detergents_Paper -2.18413508 -5.1838602  6.628079              4.447702
Delicassen         1.38992986 -1.8611795  3.663103              3.379564
                  MeanDecreaseGini
Channel                  1.885877
Fresh                   23.322487
Milk                    22.462312
Grocery                 22.265254
```

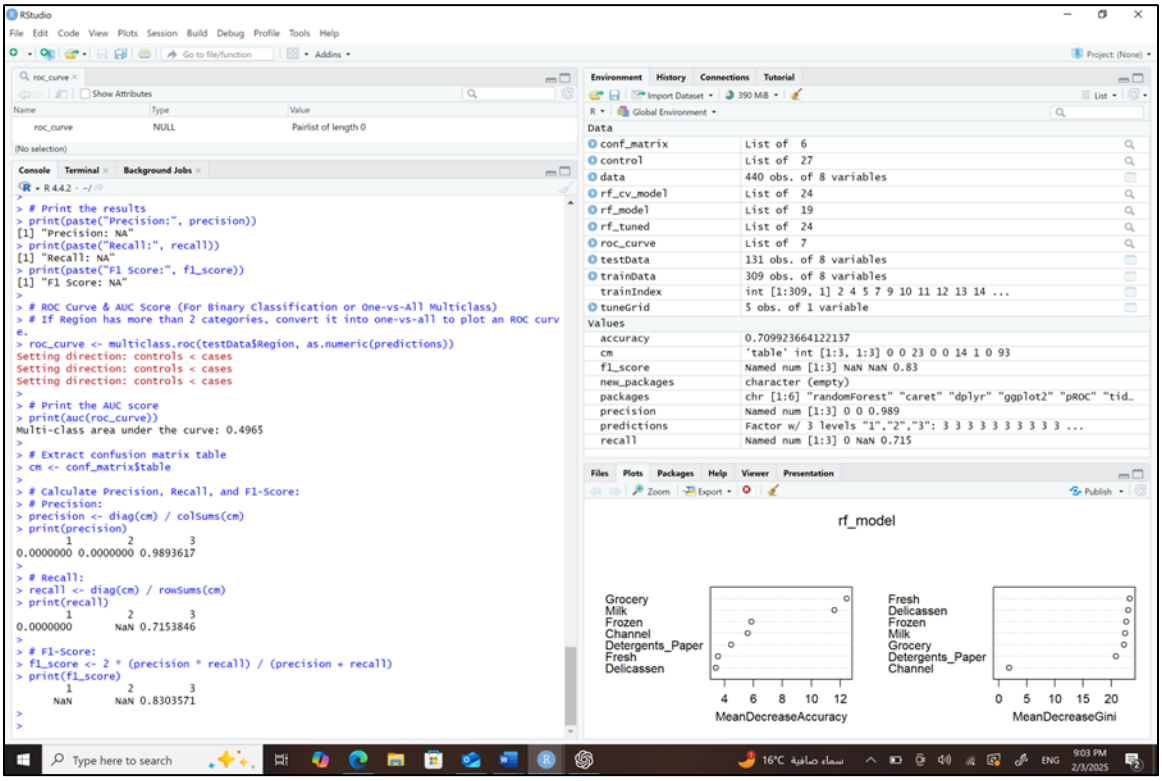



Fresh 23.322487
Milk 22.462312
Grocery 22.265254
Frozen 22.683133
Detergents_Paper 20.819474
Delicassen 23.026084
> varImpPlot(rf_model)
> # Calculate accuracy
> accuracy <- sum(predictions == testData$Region) / nrow(testData)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.709923664122137"
> # Perform cross-validation
> control <- trainControl(method = "cv", number = 5) # 5-fold CV
> rf_cv_model <- train(Region ~ ., data = trainData, method = "rf", trControl = control, tuneLength = 5)
> # Hyperparameter tuning
> # Define the tuning grid
> tuneGrid <- expand.grid(mtry = c(2, 4, 6, 8, 10))
> # Train the random forest model with the tuning grid
> rf_tuned <- train(Region ~ ., data = trainData, method = "rf", trControl = control, tuneGrid = tuneGrid)
Warning messages:
1: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
2: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
3: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
4: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
5: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
6: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
7: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
8: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
9: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
Type here to search


invalid mtry: reset to within valid range
9: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
10: In randomForest.default(x, y, mtry = param$mtry, ...) : invalid mtry: reset to within valid range
> # Print the results of the tuned model
> print(rf_tuned)
Random Forest
309 samples
7 predictor
3 classes: '1', '2', '3'
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 247, 248, 247, 247, 247
Resampling results across tuning parameters:
mtry Accuracy Kappa
2 0.7152300 0.020017782
4 0.6991010 0.007174588
6 0.7054997 0.028009616
8 0.6989424 -0.004282806
10 0.6989952 -0.004938058
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
> # More Performance Metrics (Beyond Accuracy)
> # Extract precision from confusion matrix
> precision <- conf_matrix$byClass["Pos Pred Value"]
> # Extract recall (sensitivity) from confusion matrix
> recall <- conf_matrix$byClass["Sensitivity"]
> # Compute F1 score
> f1_score <- 2 * (precision * recall) / (precision + recall)
> # Print the results
> print(paste("Precision:", precision))
[1] "Precision: NA"
> print(paste("Recall:", recall))
[1] "Recall: NA"
> print(paste("F1 Score:", f1_score))
Type here to search

```
> # Print the results
> print(paste("Precision:", precision))
[1] "Precision: NA"
> print(paste("Recall:", recall))
[1] "Recall: NA"
> print(paste("F1 Score:", f1_score))
[1] "F1 Score: NA"
>
> # ROC Curve & AUC Score (For Binary Classification or One-vs-All Multiclass)
> # If Region has more than 2 categories, convert it into one-vs-all to plot an ROC curv
e.
> roc_curve <- multiclass.roc(testData$Region, as.numeric(predictions))
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
>
> # Print the AUC score
> print(auc(roc_curve))
Multi-class area under the curve: 0.4965
>
> # Extract confusion matrix table
> cm <- conf_matrix$table
>
> # Calculate Precision, Recall, and F1-Score:
> # Precision:
> precision <- diag(cm) / colSums(cm)
> print(precision)
         1          2          3
0.0000000 0.0000000 0.9893617
>
> # Recall:
> recall <- diag(cm) / rowSums(cm)
> print(recall)
         1          2          3
0.0000000        NaN 0.7153846
>
> # F1-Score:
> f1_score <- 2 * (precision * recall) / (precision + recall)
> print(f1_score)
         1          2          3
       NaN        NaN 0.8303571
>
>
```

Result:



Analysis of the model's performance:

1- accuracy= 70.99% Indicates a moderate level of performance, and that the model correctly classifies around 71% of the test data.
2- AUC Score: 0.4965 Indicating that the model might be doing random guessing and that it might not be classifying well. Use hyperparameters tuning to improve it.
3- Precision: 0.98 and for some class :NaN indicating misclassification.
4- recall of 0.715 for one class which means it identifies 71.5% of real cases in a subset. For other classes it is NaN, indicating potential misclassification. And showing imbalance in the dataset. To solve this, balance data or use stratification in sampling.
5- F1-score: 0.83 for one class, NaN for others. Indicating mispredicting certain subsets.
6- One plot shows the importance of different variables which have more impact as follows: More impact : Fresh, Delicatessen, Frozen, and Milk. Least impact: Grocery, Detergents_Paper, and Channel.
7- To improve the performance, trying a different model like logistic regression might have better performance. As well as feature selection.

Reference

no	Title	link	notes
1	**How to import data into Rstudio**	https://www.youtube.com/watch?v=2_tW7e4e_dM	