IE 5390 – Assignment - 09

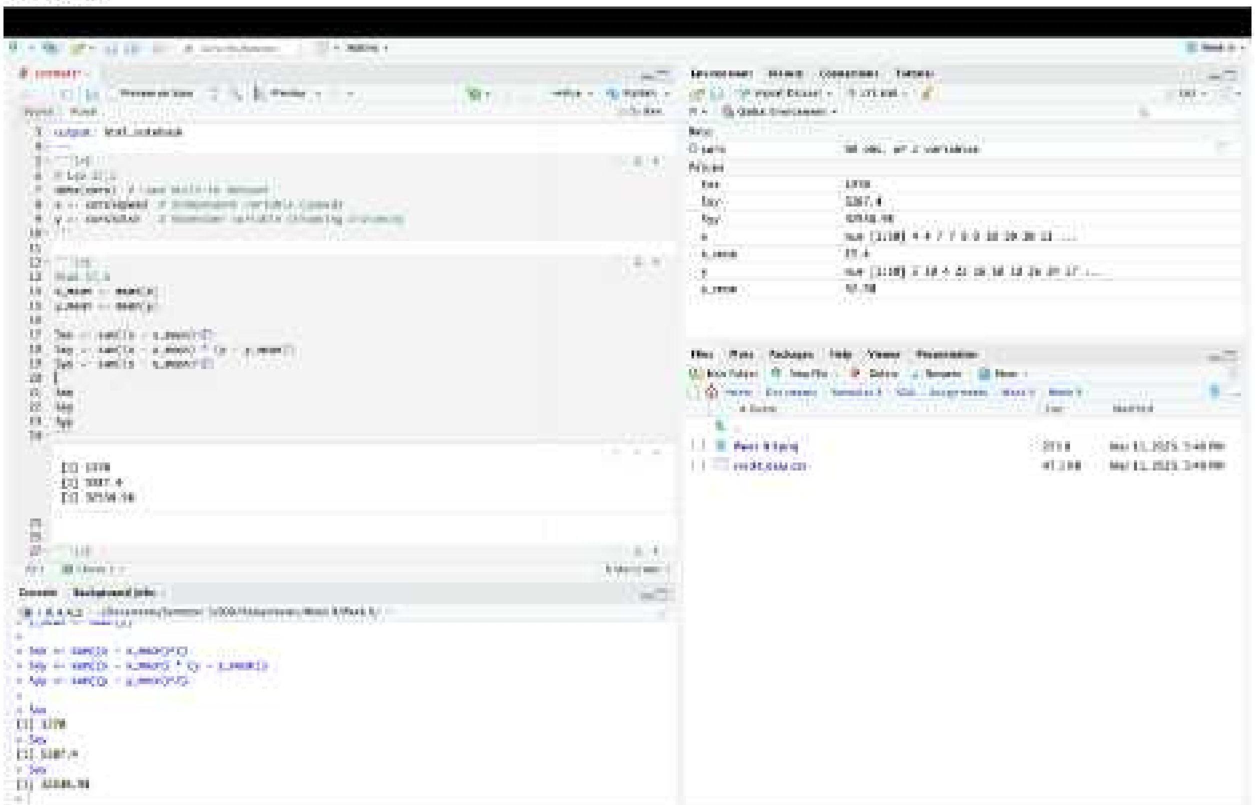
Name: Simran Abhay Sinha

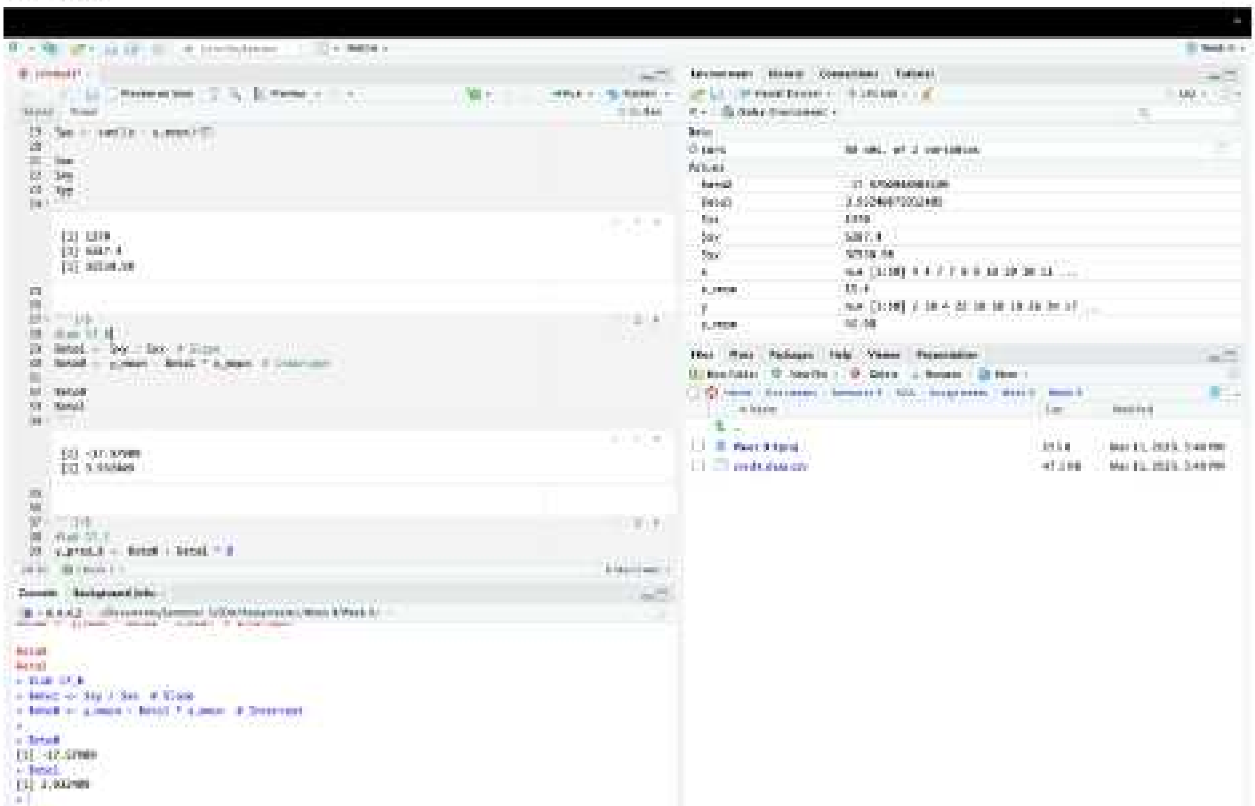
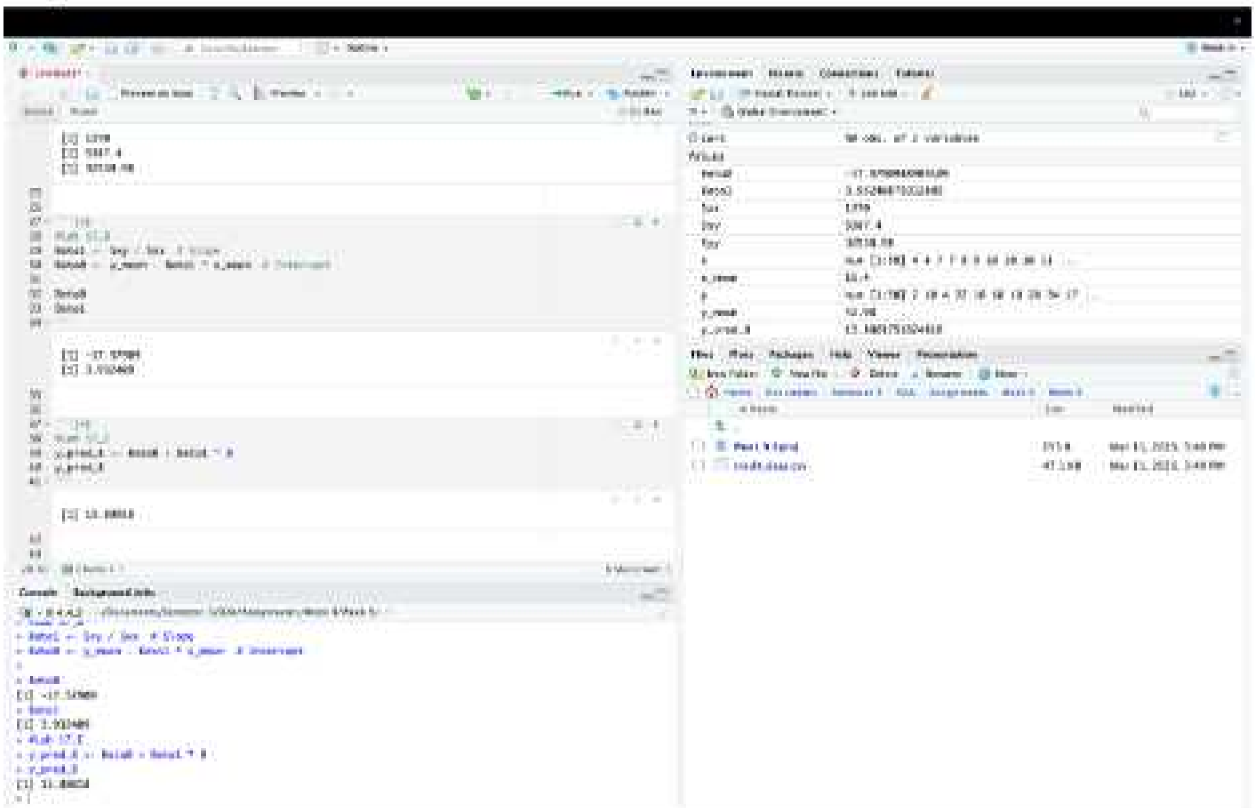
Lab 17

1. Use the cars dataset that we worked with in class. But this time, instead of using the lm() function for linear regression, create the linear regression model from scratch. You have the math preliminaries to help you with the model components. Some hints on what you will need to do:

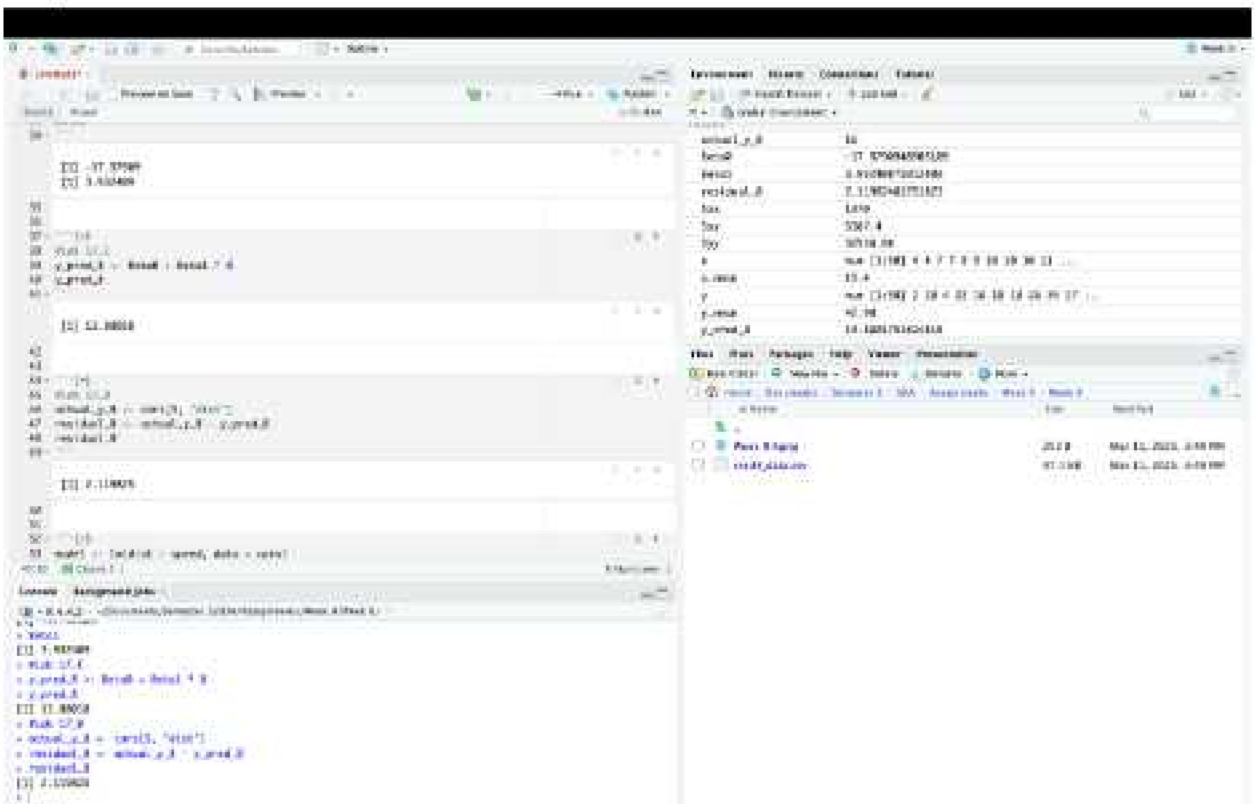
i) Establish what x and y are. In this case, x is speed and y is the stopping distance.
ii) Calculate Sxx, Sxy and Syy
iii) Then calculate Beta0 and Beta1

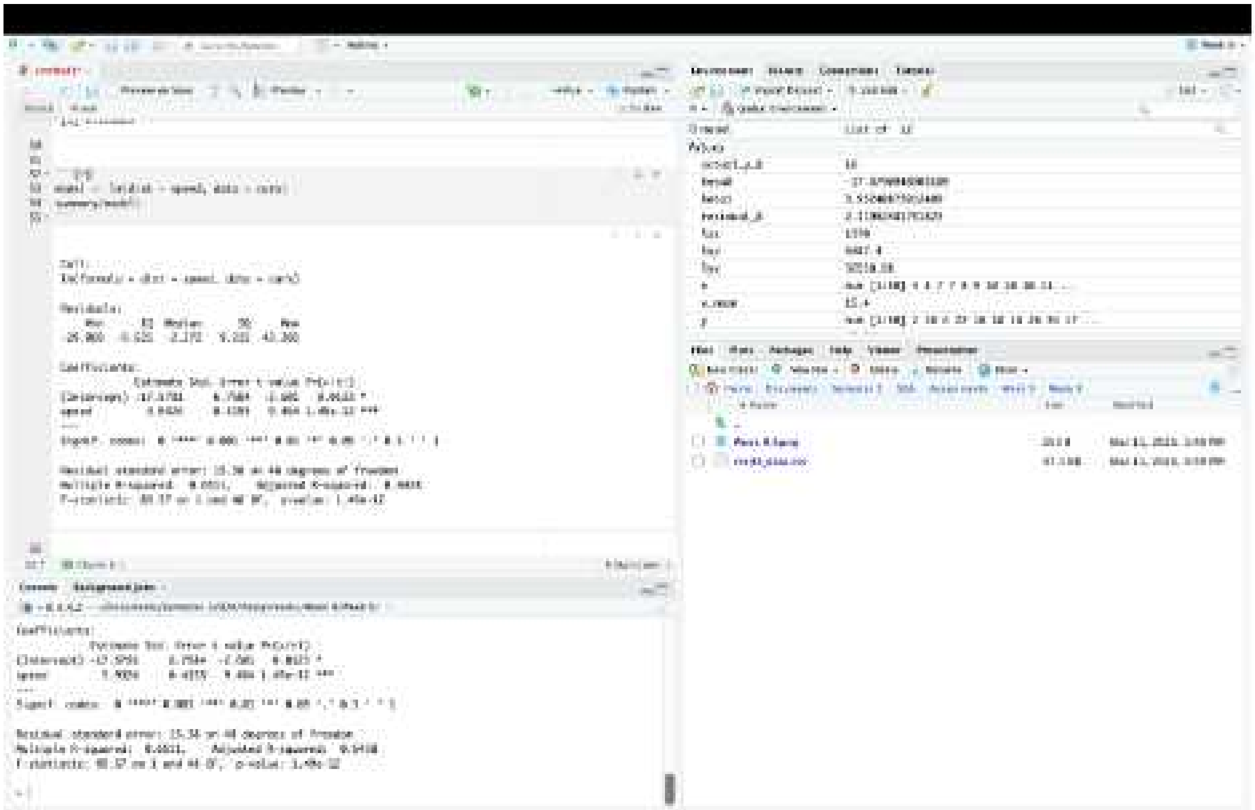
Loading the dataset
Cars is built-in dataset

```
---
```{r}
# Lab 17_1
data(cars)  # Load built-in dataset
x <- cars$speed  # Independent variable (speed)
y <- cars$dist   # Dependent variable (stopping distance)
```
```

Answer the following questions:

a) What are the values for Sxx, Sxy and Syy that you obtained?

Code:

```
```{r}
#Lab 17_A
x_mean <- mean(x)
y_mean <- mean(y)

Sxx <- sum((x - x_mean)^2)
Sxy <- sum((x - x_mean) * (y - y_mean))
Syy <- sum((y - y_mean)^2)

Sxx
Sxy
Syy
```
```

Output:



b) Compare these coefficients you obtained via manual calculate to the ones obtained using the lm() function used in class. Are they same/similar?

Code:

```
```{r}
#Lab 17_B
Beta1 <- Sxy / Sxx  # Slope
Beta0 <- y_mean - Beta1 * x_mean  # Intercept

Beta0
Beta1
```

 [1] -17.57909
```

Output:



Ans: - Intercept (Beta0): -17.57909 and Slope (Beta1): 3.932409

- They are different

c) Predict the stopping distance for a speed of 8 mph. You MUST use the Beta0 and Beta1 you found in this calculation.

Code:

```
```{r}
#Lab 17_C
y_pred_8 <- Beta0 + Beta1 * 8
y_pred_8
```
```

Output:



d) Calculate the residual for a car travelling at 8 mph. What you need to do is obtain the stopping distance for a car travelling at 8 mph using the regression equation manually with Beta0 and Beta1, then compare this with data from row 5 from the cars dataset. What is the residual?

Code:

```
```{r}
#Lab 17_D
actual_y_8 <- cars[5, "dist"]
residual_8 <- actual_y_8 - y_pred_8
residual_8
```
```

Output:





2. Load the faithful dataset by typing data(faithful). Do the following:

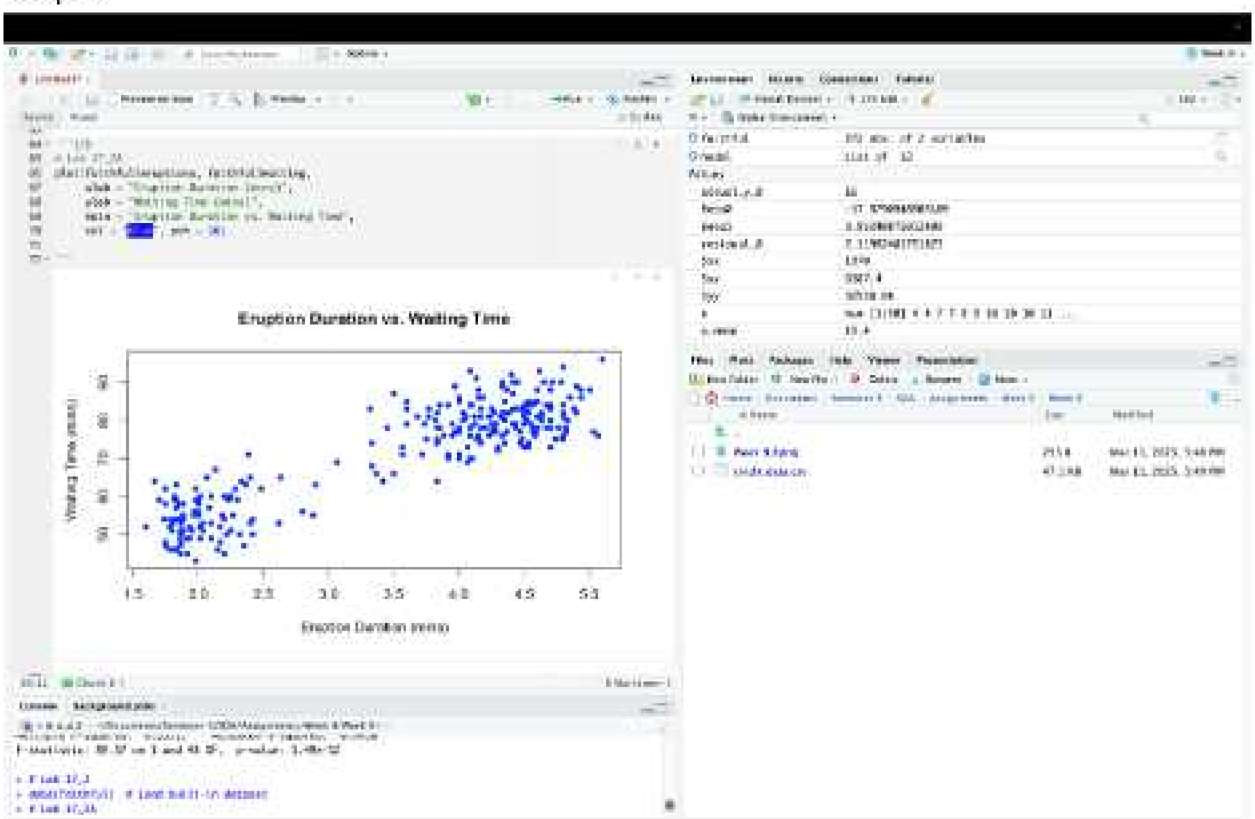
a) Plot the waiting times against the duration of eruption.

Code:

```{r}
# Lab 17_2A
plot(faithful$eruptions, faithful$waiting,
     xlab = "Eruption Duration (mins)",
     ylab = "Waiting Time (mins)",
     main = "Eruption Duration vs. Waiting Time",
     col = "blue", pch = 16)

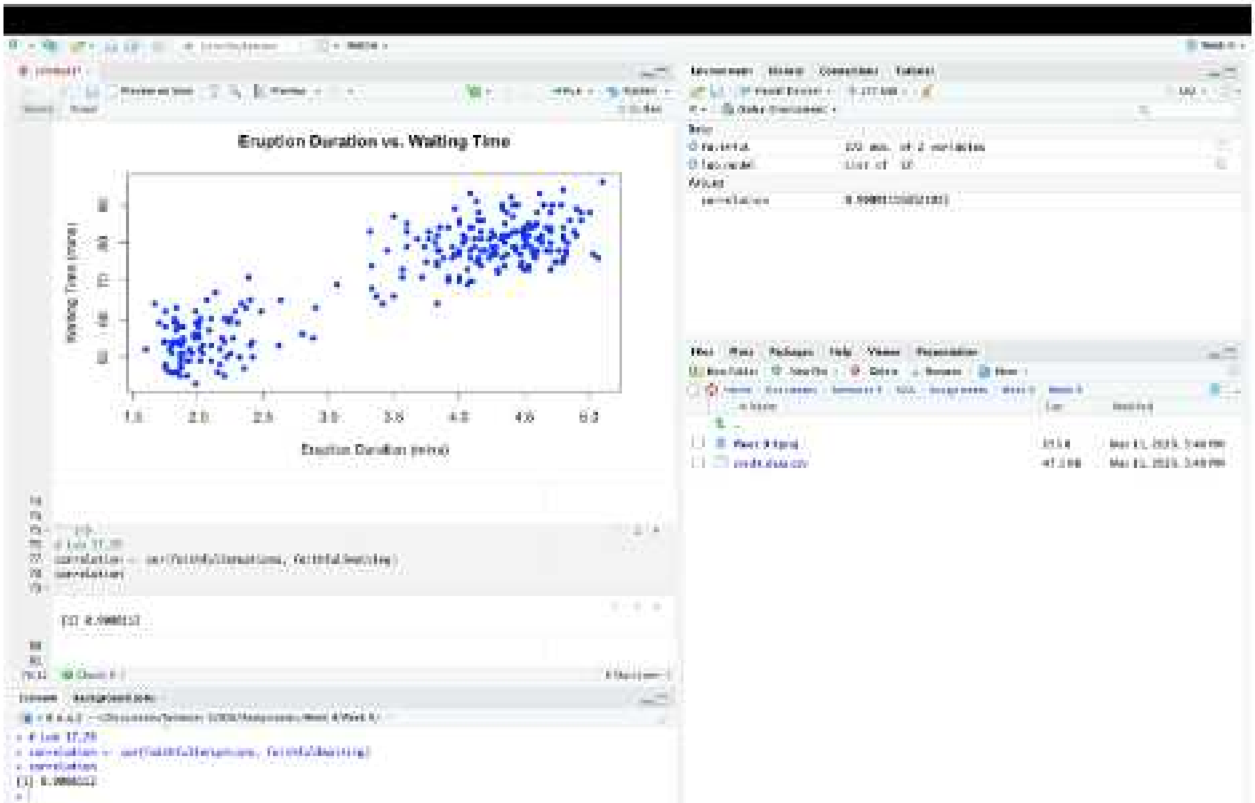
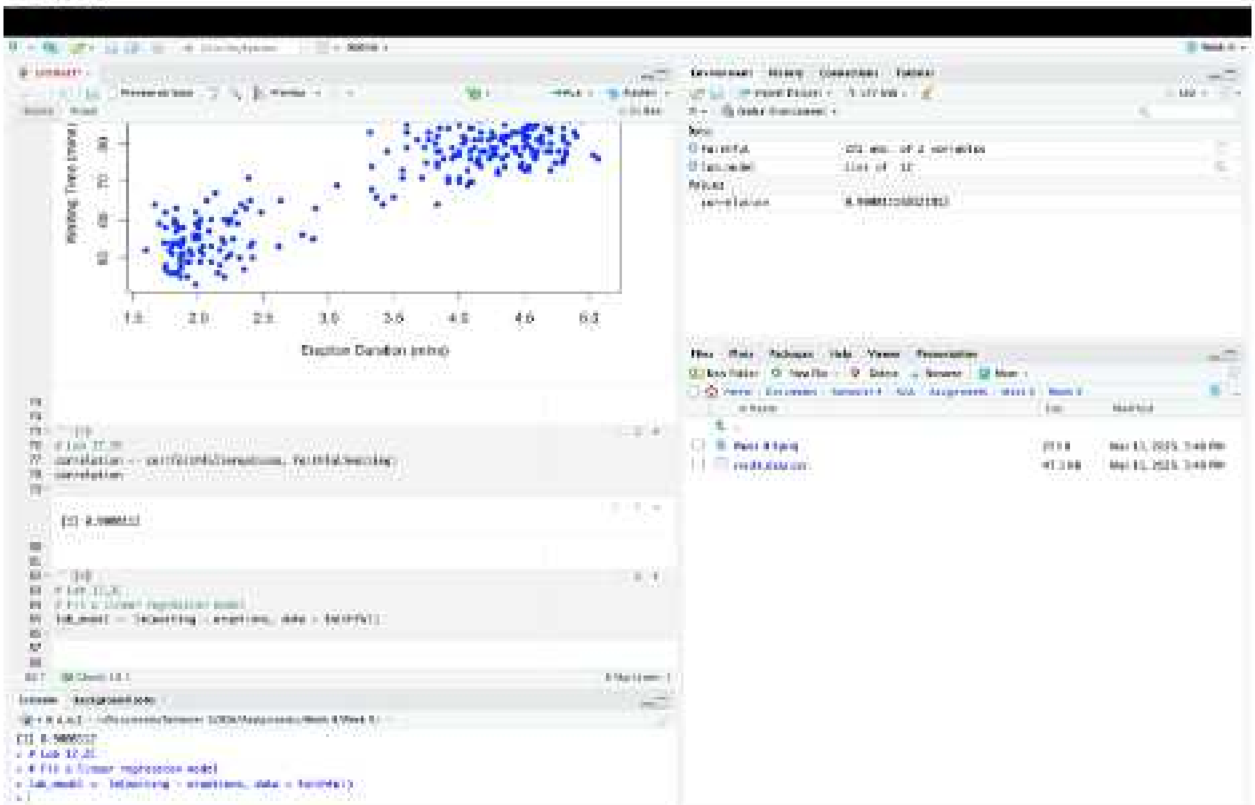
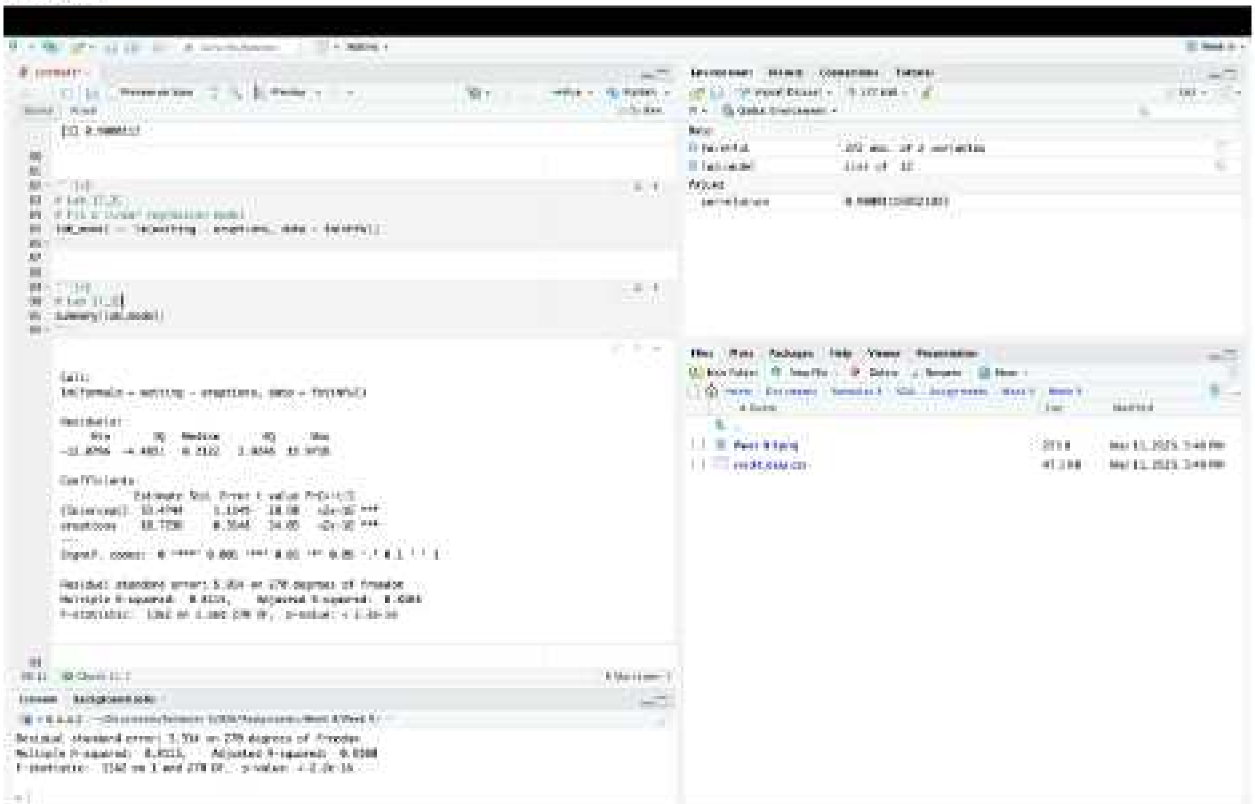
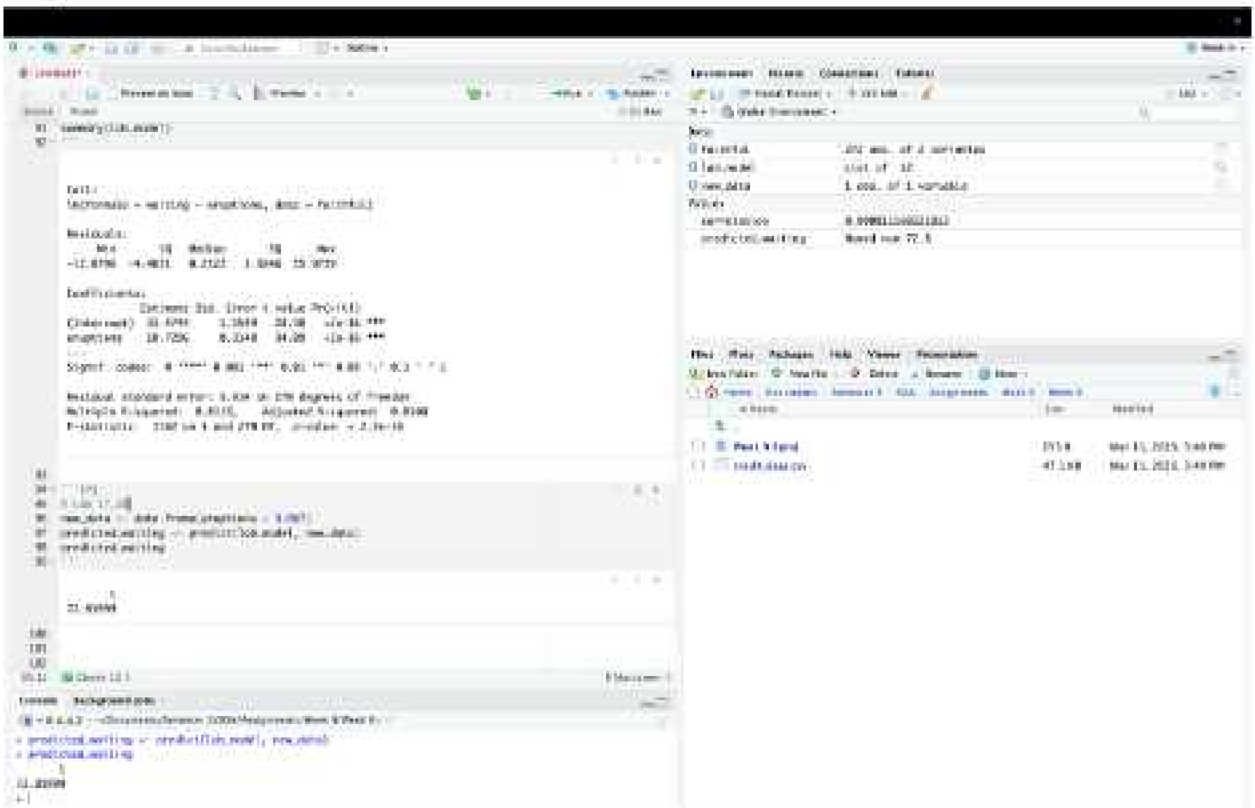
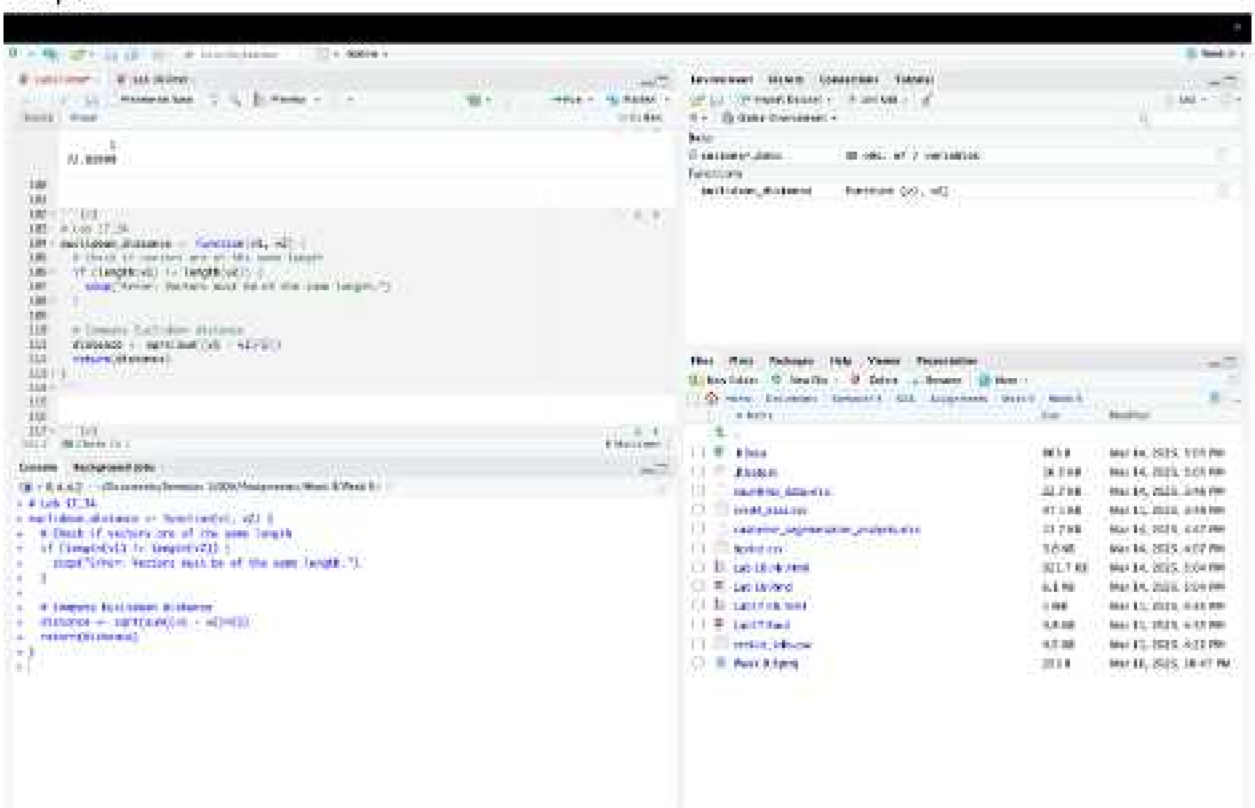
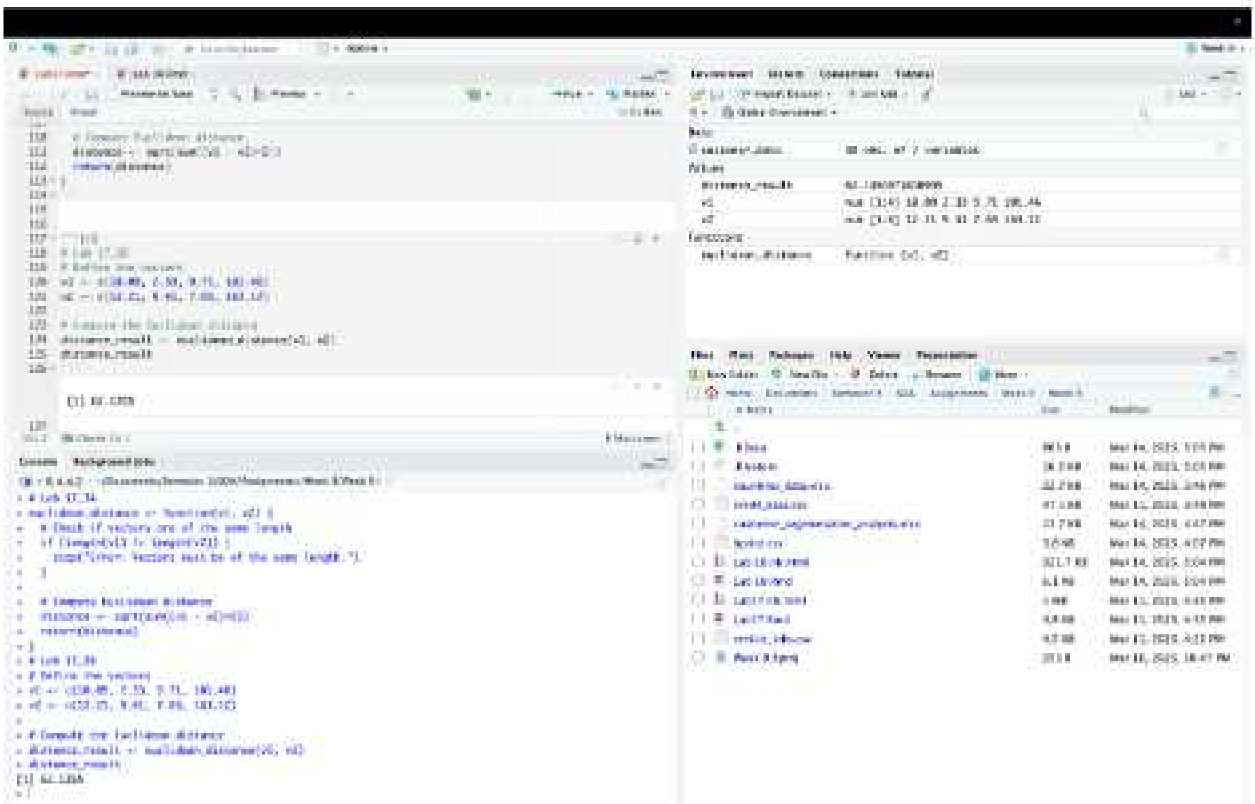
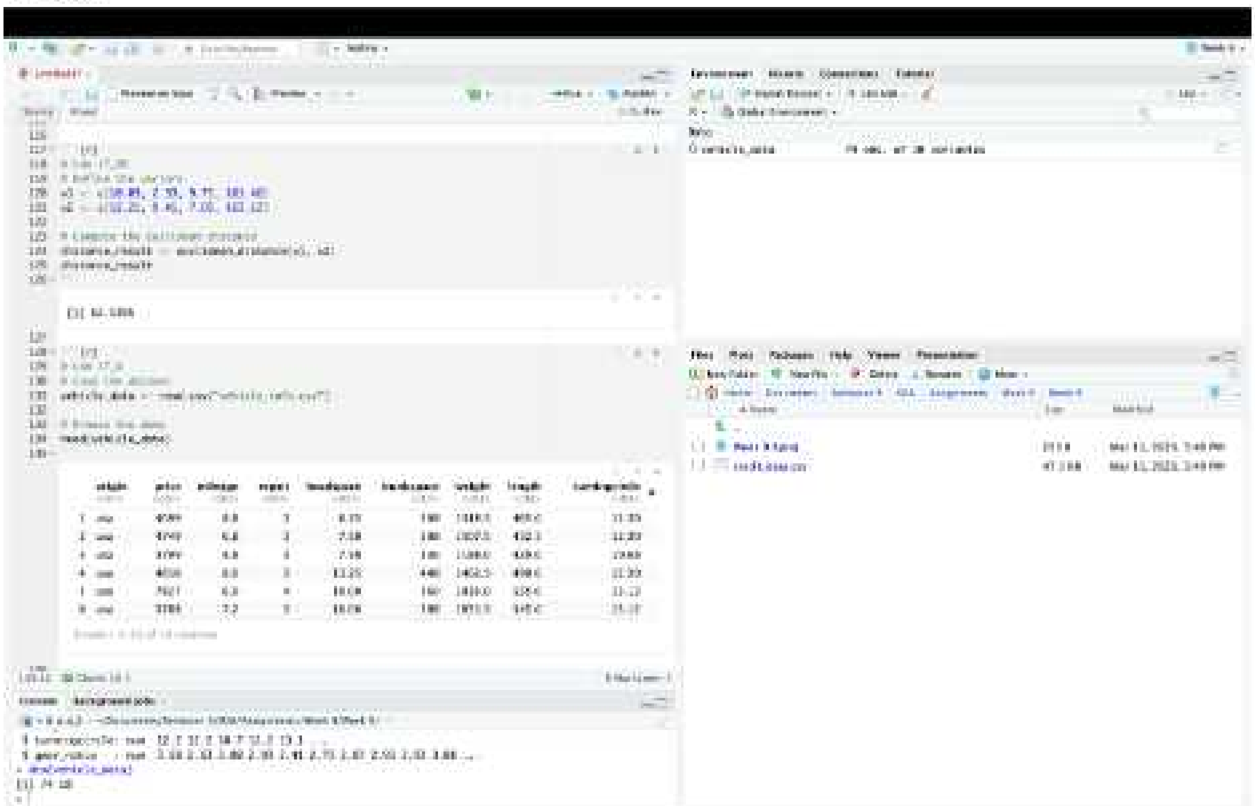
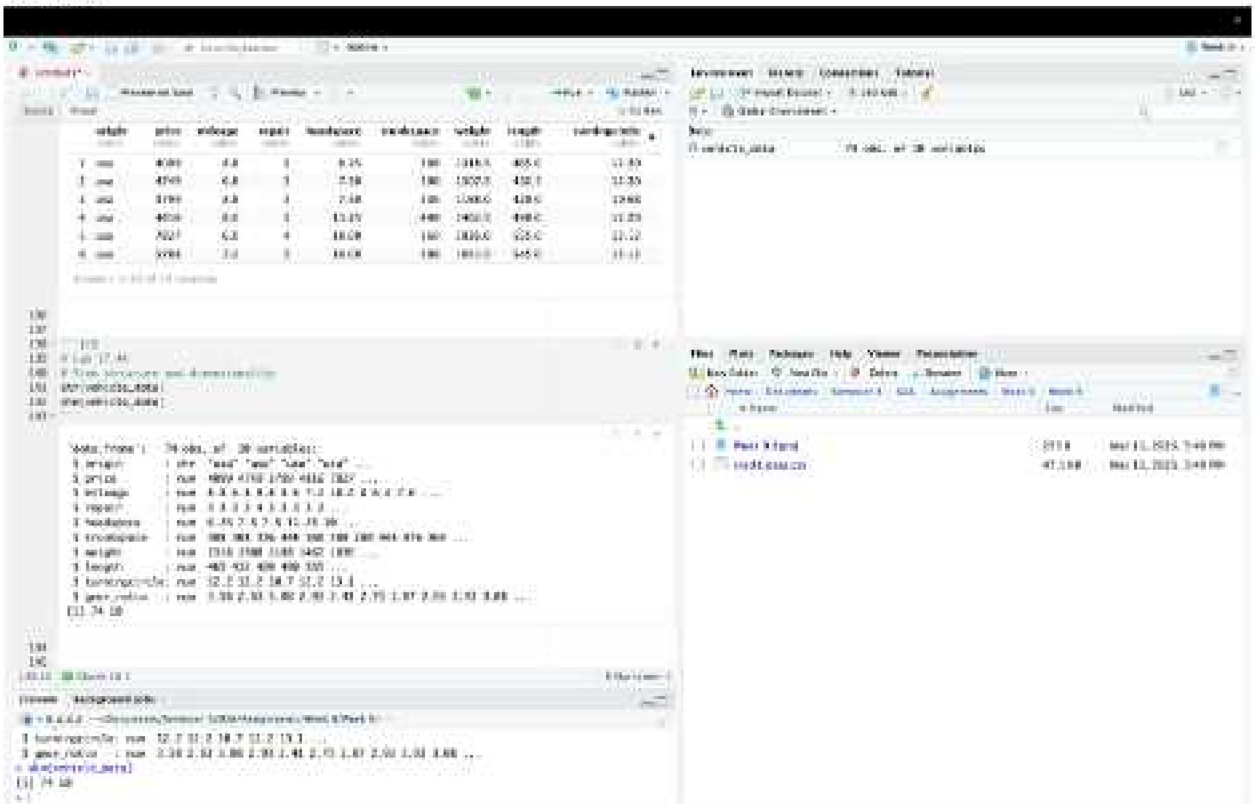
```

Output:

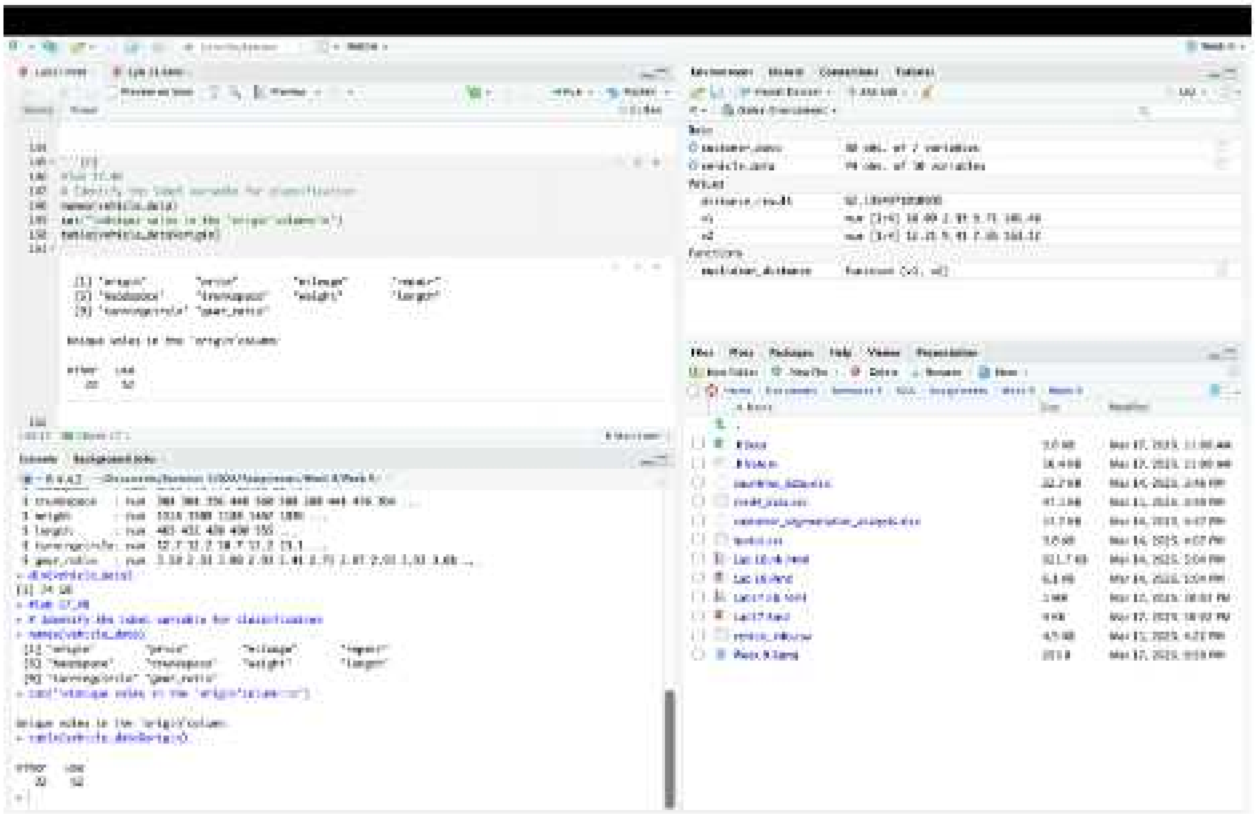


b) Compute the Pearson correlation coefficient between waiting times and duration of eruption. Is there evidence of a linear relationship? Is it: weak, strong, positive, etc.

Code:

```
```{r}
# Lab 17_2B
correlation <- cor(faithful$eruptions, faithful$waiting)
correlation
```
```

Output:



Ans:

c) Use lm to fit waiting times as a linear function of eruption durations and save the result of the regression function to the variable 'lab_model'.

Code:

```
```{r}
# Lab 17_2C
# Fit a linear regression model
lab_model <- lm(waiting ~ eruptions, data = faithful)
```
```

Output:



d) Print the summary of the model. What does the summary tell you about model performance from the metrics shown?

Code:

```
```{r}
# Lab 17_2D
summary(lab_model)
```
```

Output:



Ans:- The model shows a strong relationship between eruption duration and waiting time, with 81.15% of waiting time variation explained by eruption length ($R^2 = 0.8115$).

- The relationship is highly significant ($F = 1162$, $p < 2.2e-16$), with predictions expected to be within ±5.9 minutes of actual values.

e) Using the 'lab_model' to predict the waiting time for an eruption duration of 3.667 mins.

Code:

```
```{r}
# Lab 17_2E
new_data <- data.frame(eruptions = 3.667)
predicted_waiting <- predict(lab_model, new_data)
predicted_waiting
```
```

Output:



3. kNN Preliminaries

a) Write a function that returns the Euclidean distance between two Vectors of the same length. Function must return an error if vectors are of different lengths.

Code:

```
```{r}
# Lab 17_3A
euclidean_distance <- function(v1, v2) {
  # Check if vectors are of the same length
  if (length(v1) != length(v2)) {
    stop("Error: Vectors must be of the same length.")
  }

  # Compute Euclidean distance
  distance <- sqrt(sum((v1 - v2)^2))
  return(distance)
}
```
```

Output:



b) Find the Euclidean distance between the following two vectors:

v1 = (10.09, 2.33, 9.71, 101.46) and v2 = (12.21, 9.41, 7.65, 163.12)

Code:

```
```{r}
# Lab 17_30
# Define the vectors
v1 <- c(10.09, 2.33, 9.71, 101.46)
v2 <- c(12.21, 9.41, 7.65, 163.12)

# Compute the Euclidean distance
distance_result <- euclidean_distance(v1, v2)
distance_result
```
```

Output:



4. Use the vehicle_info.csv dataset provided to you and load it into R.

Importing the vehicle_info.csv

Code:

```
```{r}
# Lab 17_4
# Load the dataset
vehicle_data <- read.csv("vehicle_info.csv")

# Browse the data
head(vehicle_data)
```
```

Output:



a) Browse the data, view its structure and its dimensionality. How many variables does it have? How many rows does it have?

Code:

```
```{r}
# Lab 17_4A
# View structure and dimensionality
str(vehicle_data)
dim(vehicle_data)
```
```

Output:



Ans: It has 74 obs of 10 variables

b) Is there a "label" variable suited for classification purposes? Which variable is that?

Code:

```
```{r}
#lab 17_4B
# Identify the label variable for classification
names(vehicle_data)
cat("\nUnique vales in the 'origin'column:\n")
table(vehicle_data$origin)
```
```

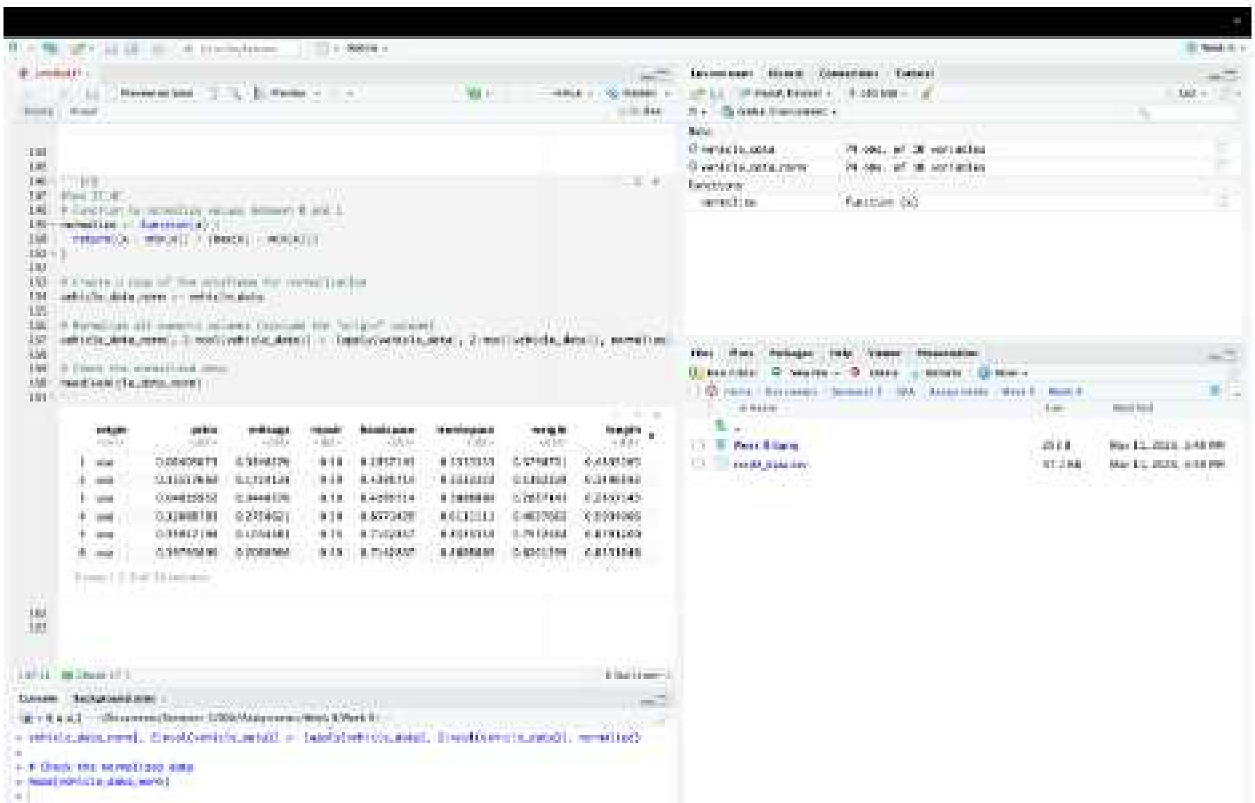
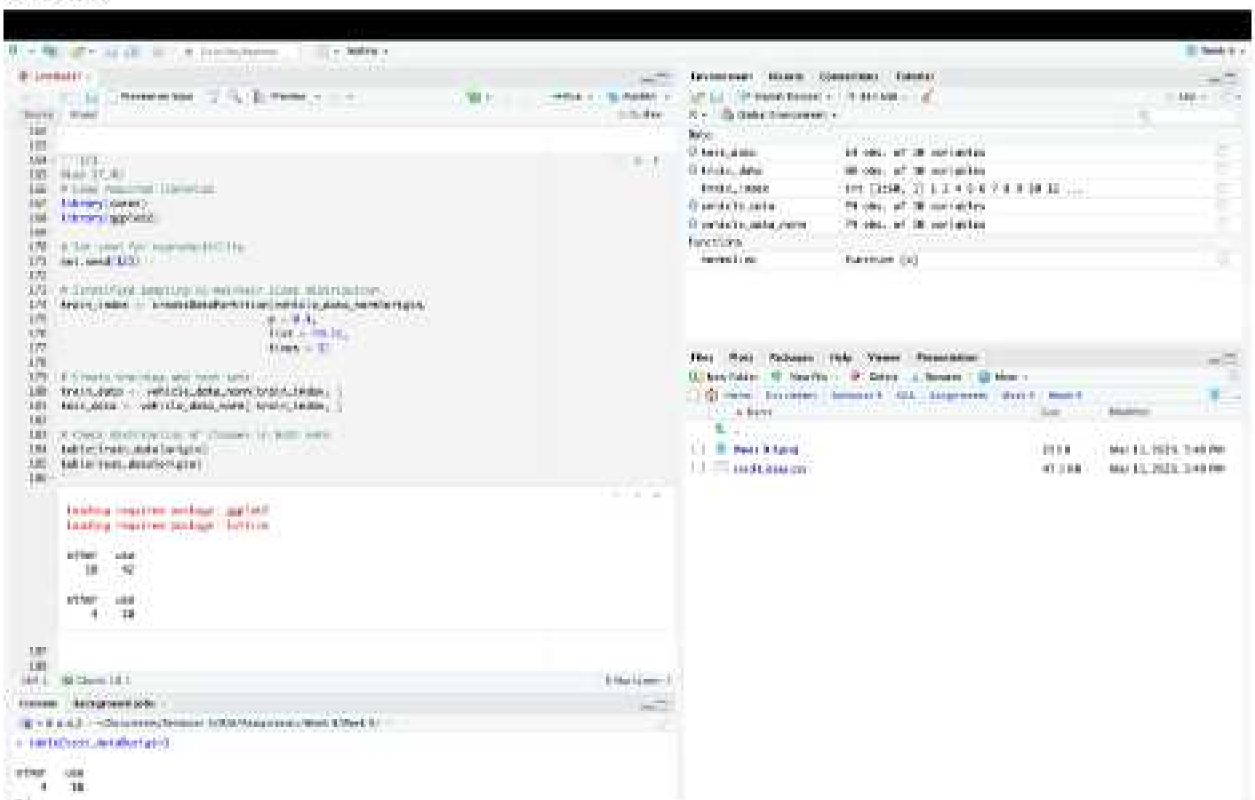
Output:



That is the reason we can use "Origin" column for classification

c) Normalize the data (excluding the "label" variable). The resulting values of all variables should be in the range from 0 to 1.

Code:

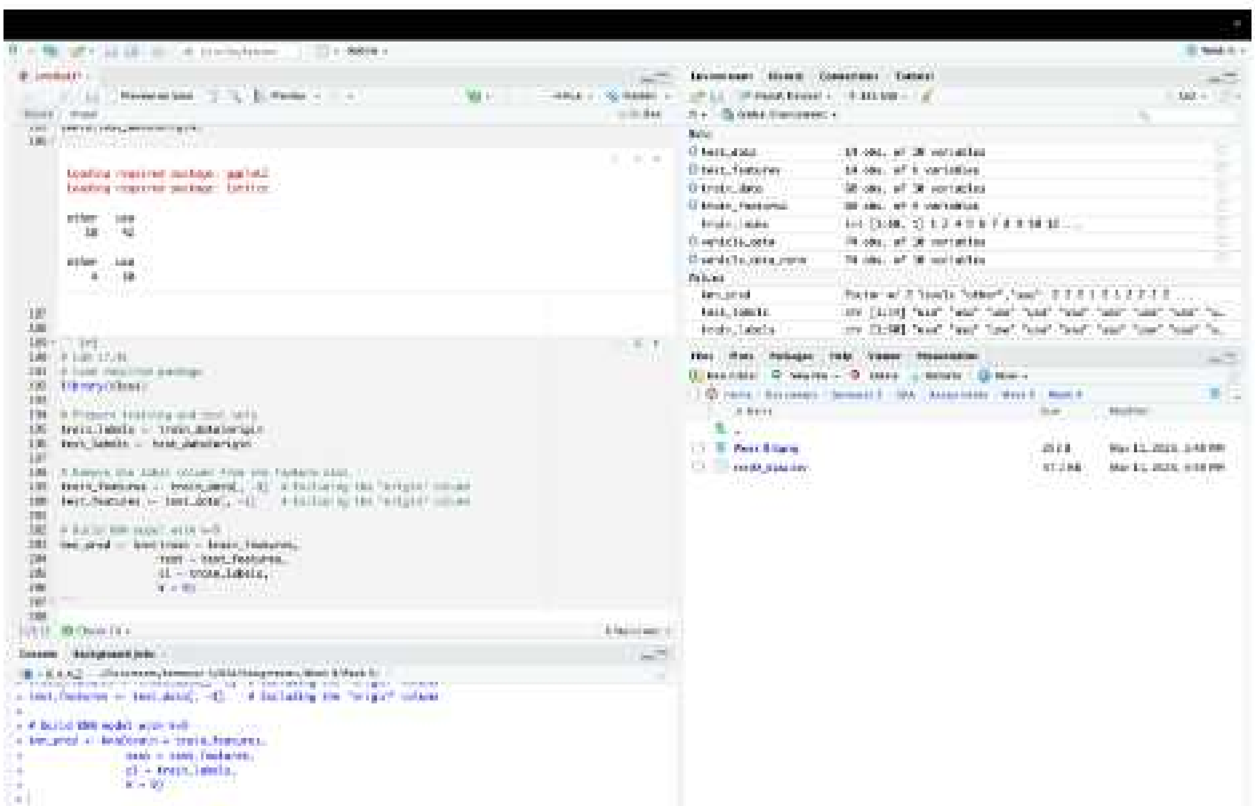
```r
```{r}
#lab 17.4c
# Function to normalize values between 0 and 1
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# Create a copy of the dataframe for normalization
vehicle_data_norm <- vehicle_data

# Normalize all numeric columns (exclude the "origin" column)
vehicle_data_norm[, 2:ncol(vehicle_data)] <- lapply(vehicle_data[, 2:ncol(vehicle_data)], normalize)

# Check the normalized data
head(vehicle_data_norm)
```
```

Output:



d) Create a training and test set. Use 80% of the records for the training set and 20% for test set. Make sure to sample records so that the class labels in the training and testing sets are not skewed.

Code:

```
```{r}
#Lab 17_40
# Load required libraries
library(caret)
library(ggplot2)

# Set seed for reproducibility
set.seed(123)

# Stratified sampling to maintain class distribution
train_index <- createDataPartition(vehicle_data_norm$origin,
                                   p = 0.8,
                                   list = FALSE,
                                   times = 1)

# Create training and test sets
train_data <- vehicle_data_norm[train_index, ]
test_data <- vehicle_data_norm[-train_index, ]

# Check distribution of classes in both sets
table(train_data$origin)
table(test_data$origin)
```
```

Output:



e) Build a knn() classifier using the "class" package. Set k=9.

Code:

```
```{r}
# Lab 17_4E
# Load required package
library(class)

# Prepare training and test sets
train_labels <- train_data$origin
test_labels <- test_data$origin

# Remove the label column from the feature sets
train_features <- train_data[, -1]  # Excluding the "origin" column
test_features <- test_data[, -1]    # Excluding the "origin" column

# Build KNN model with k=9
knn_pred <- knn(train = train_features,
                test = test_features,
                cl = train_labels,
                k = 9)
```
```

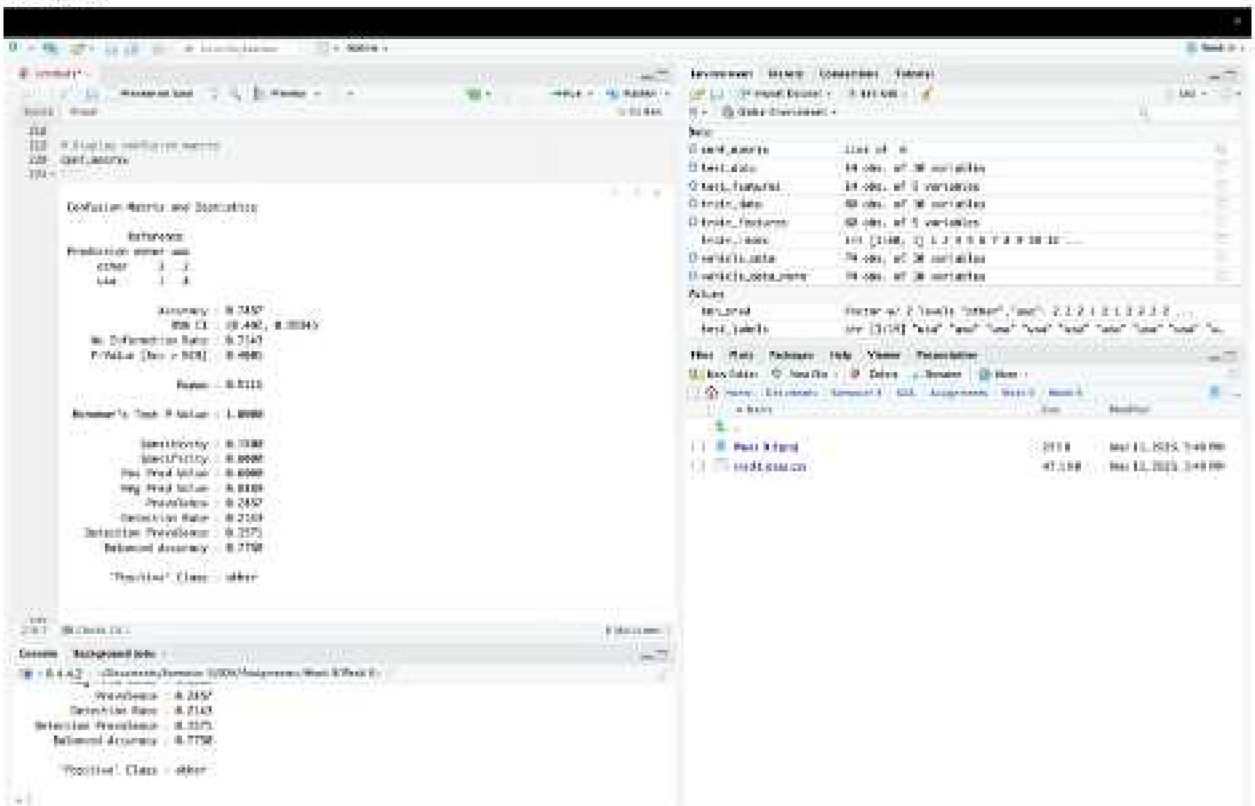
Output:



f) Get the confusionMatrix using the "caret" package. Are there any misclassified labels?

Code:

```r

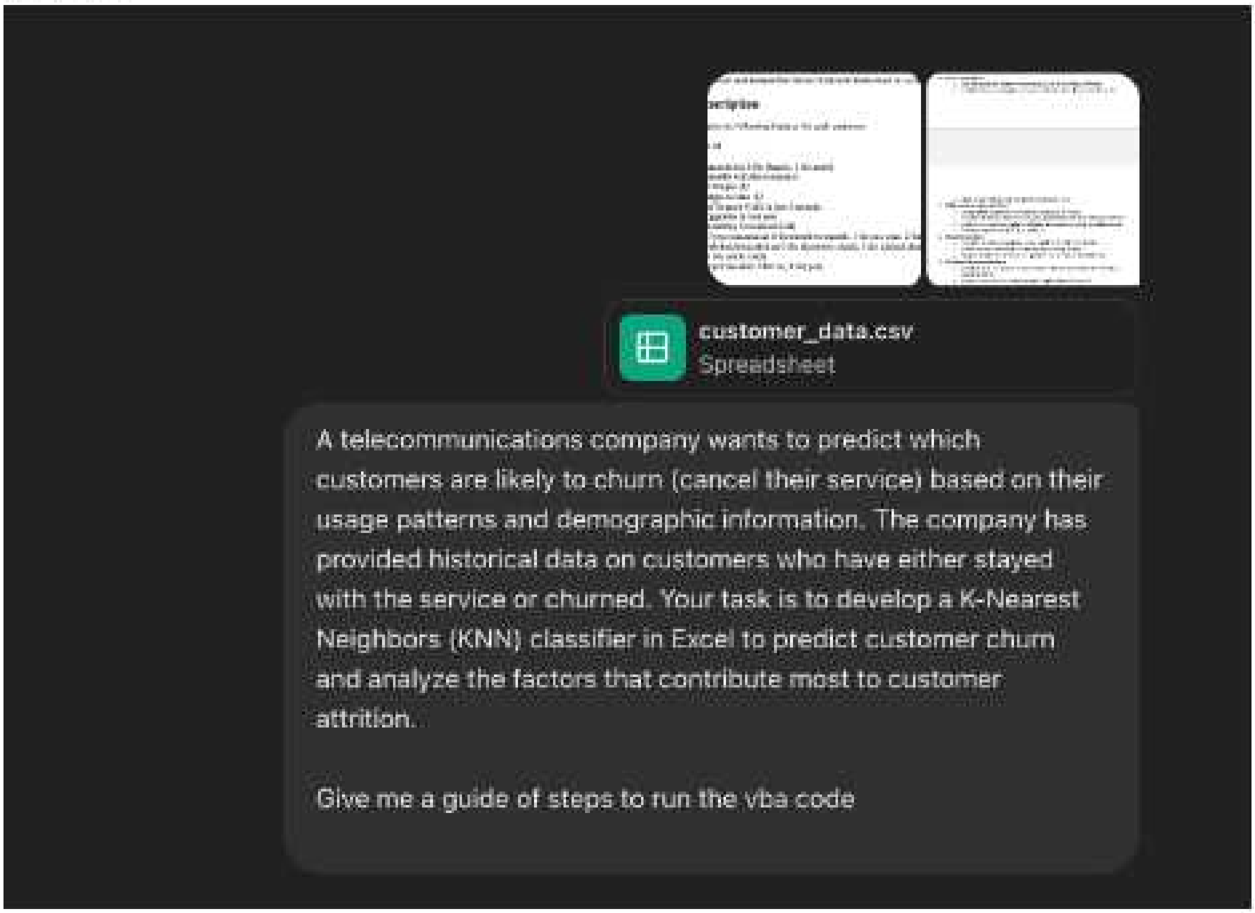
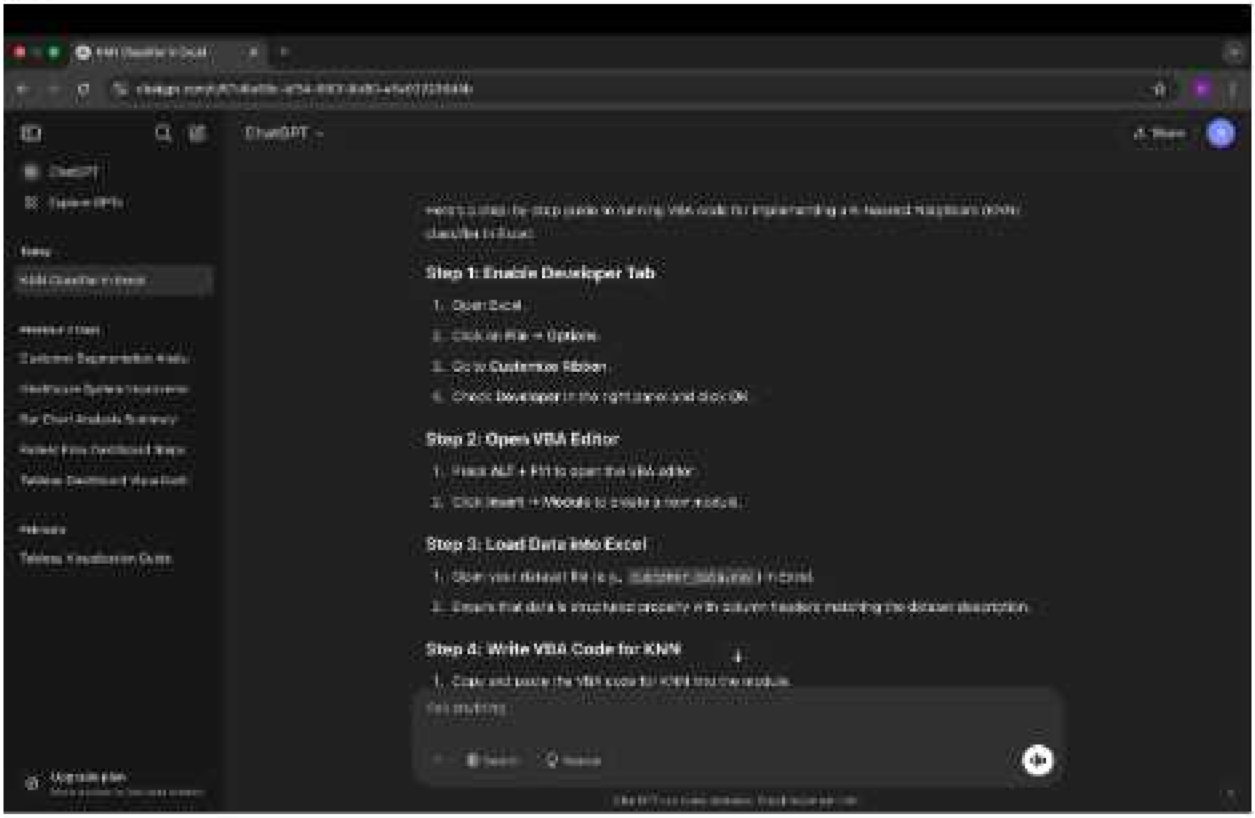
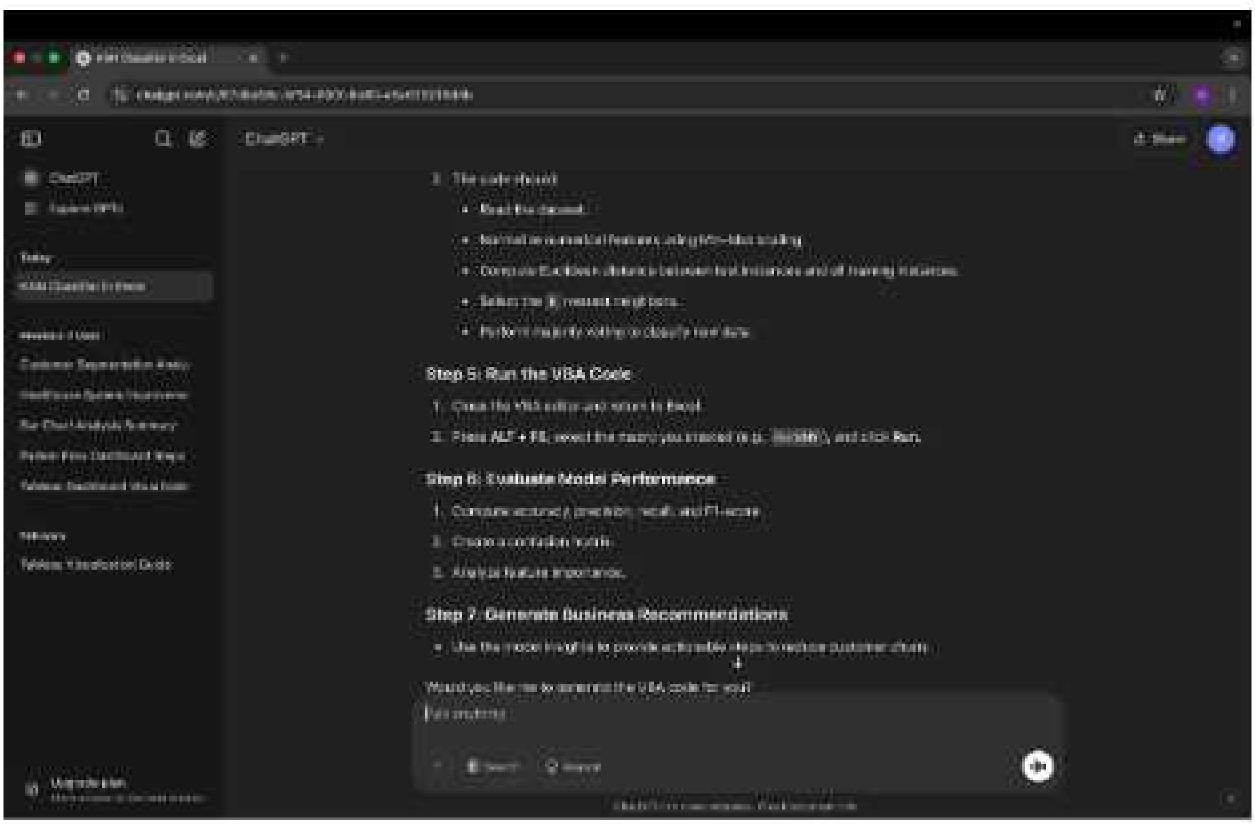
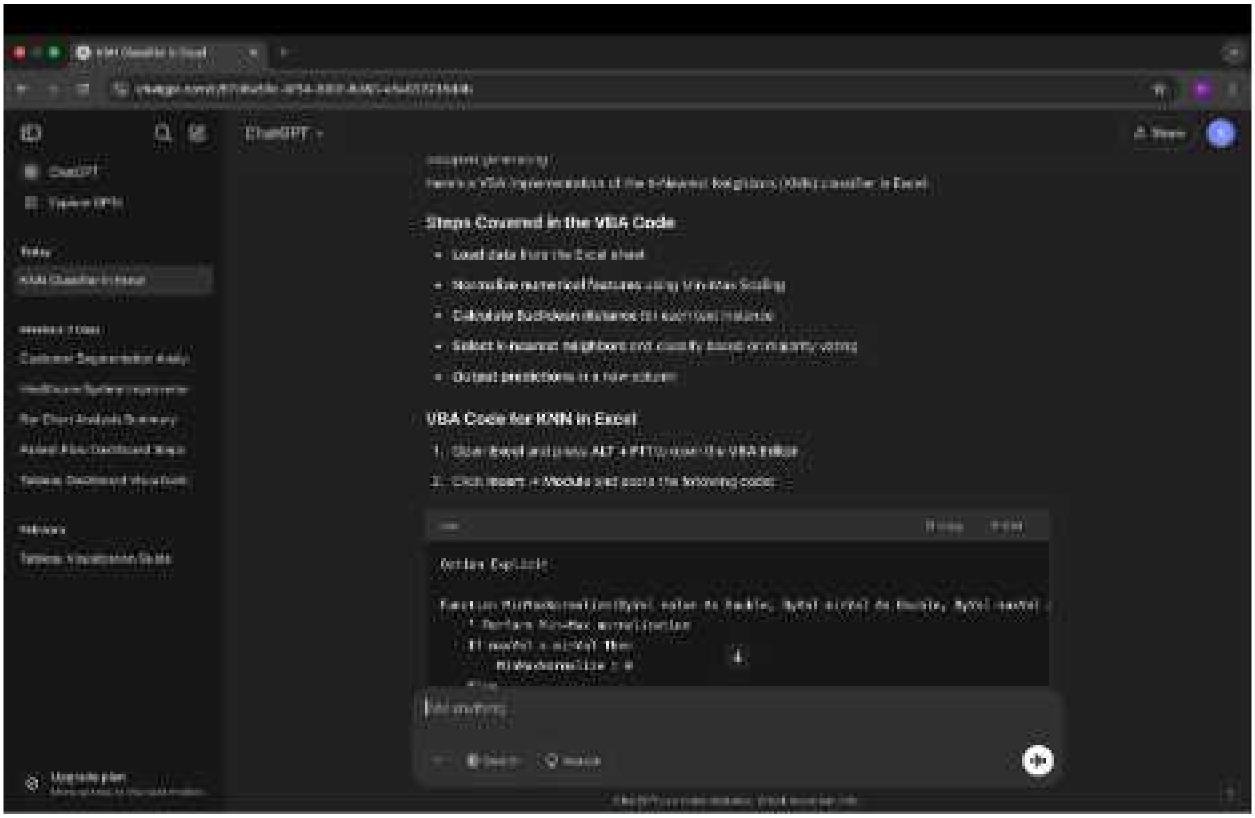
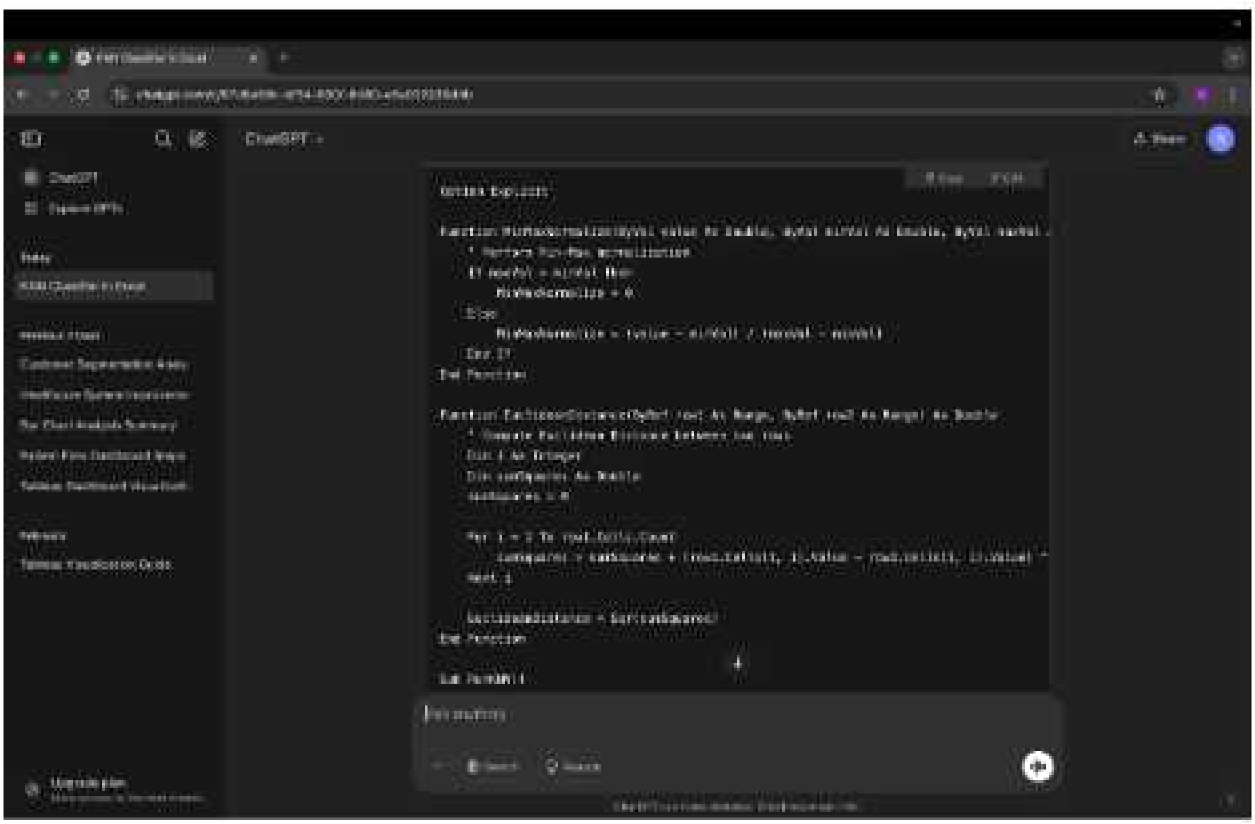
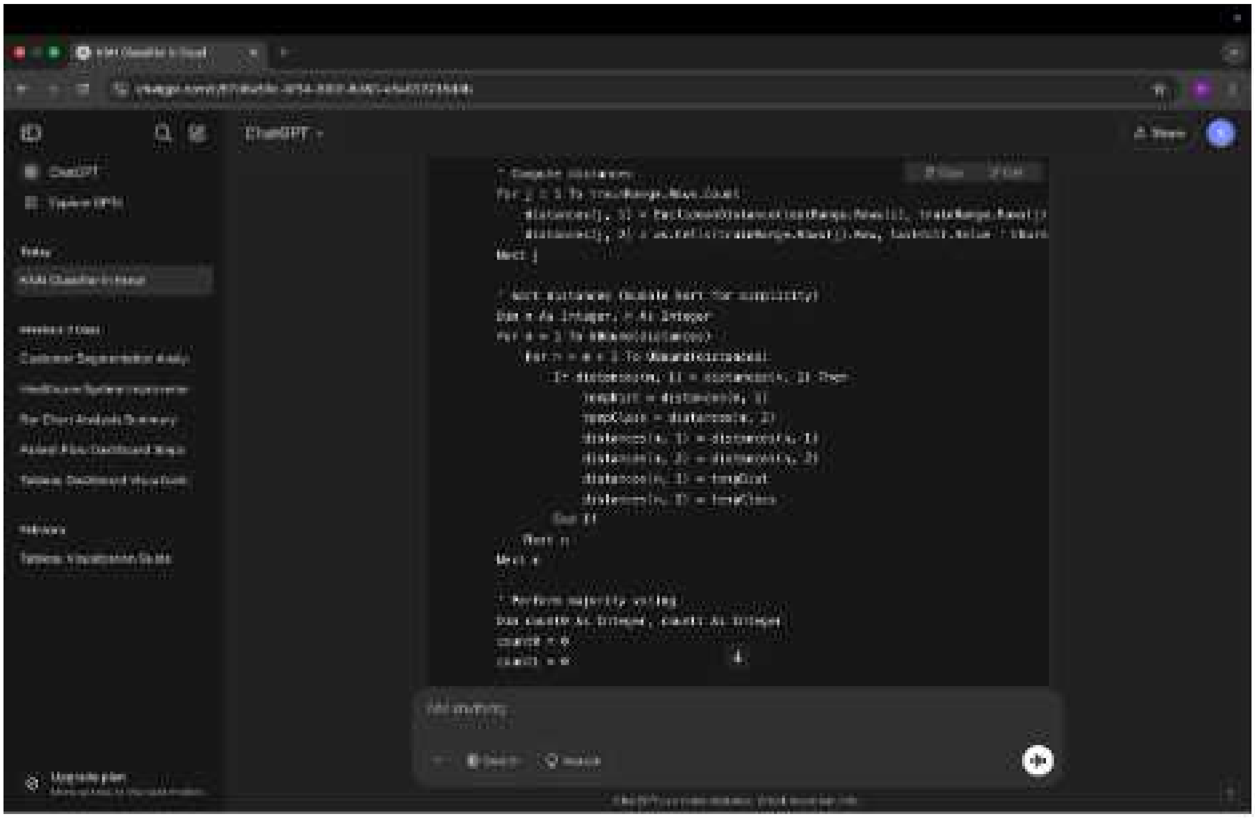
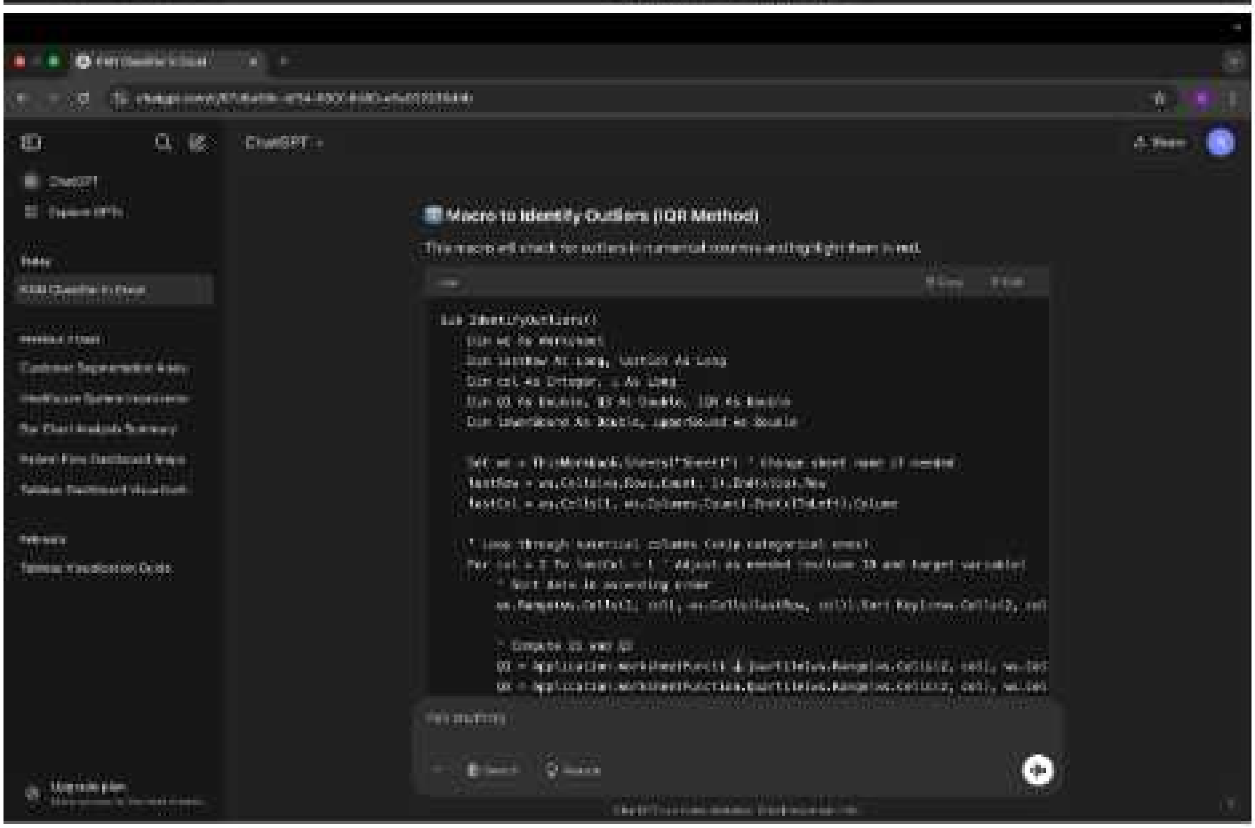
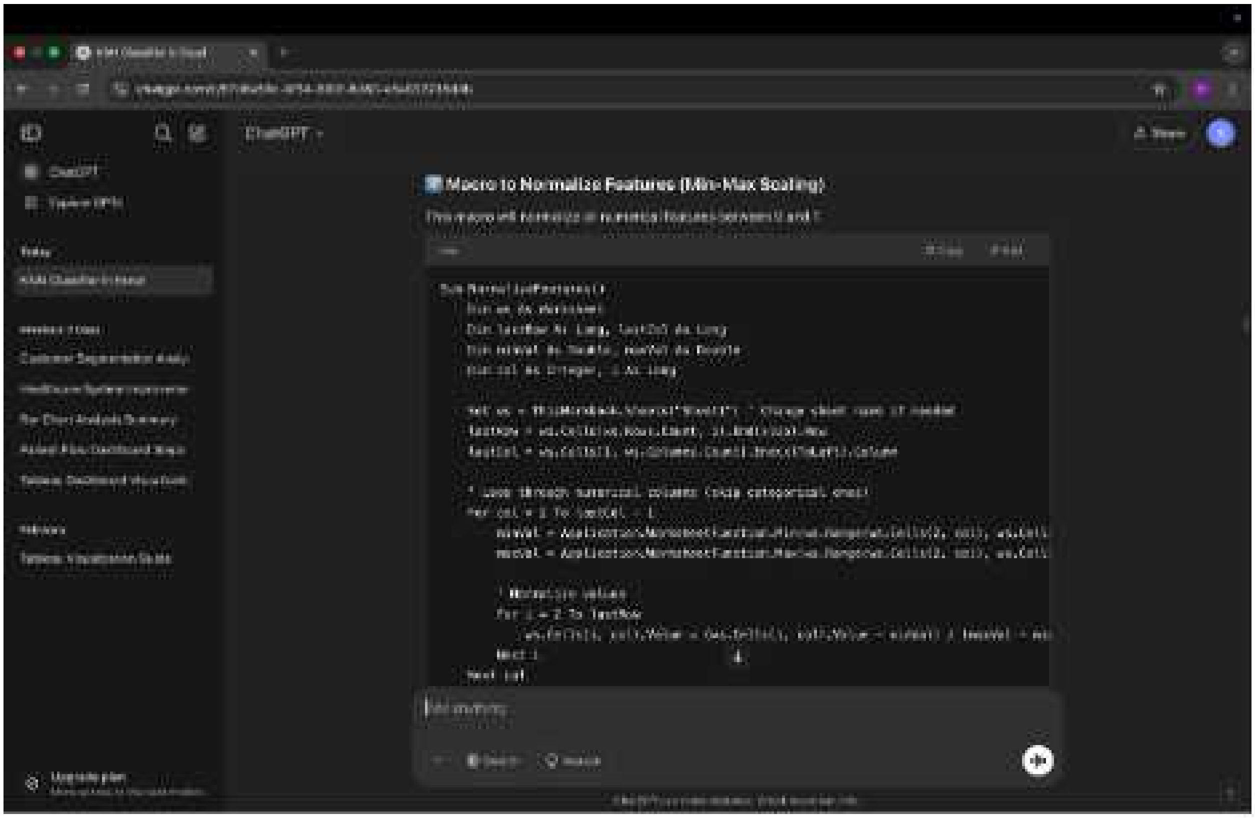
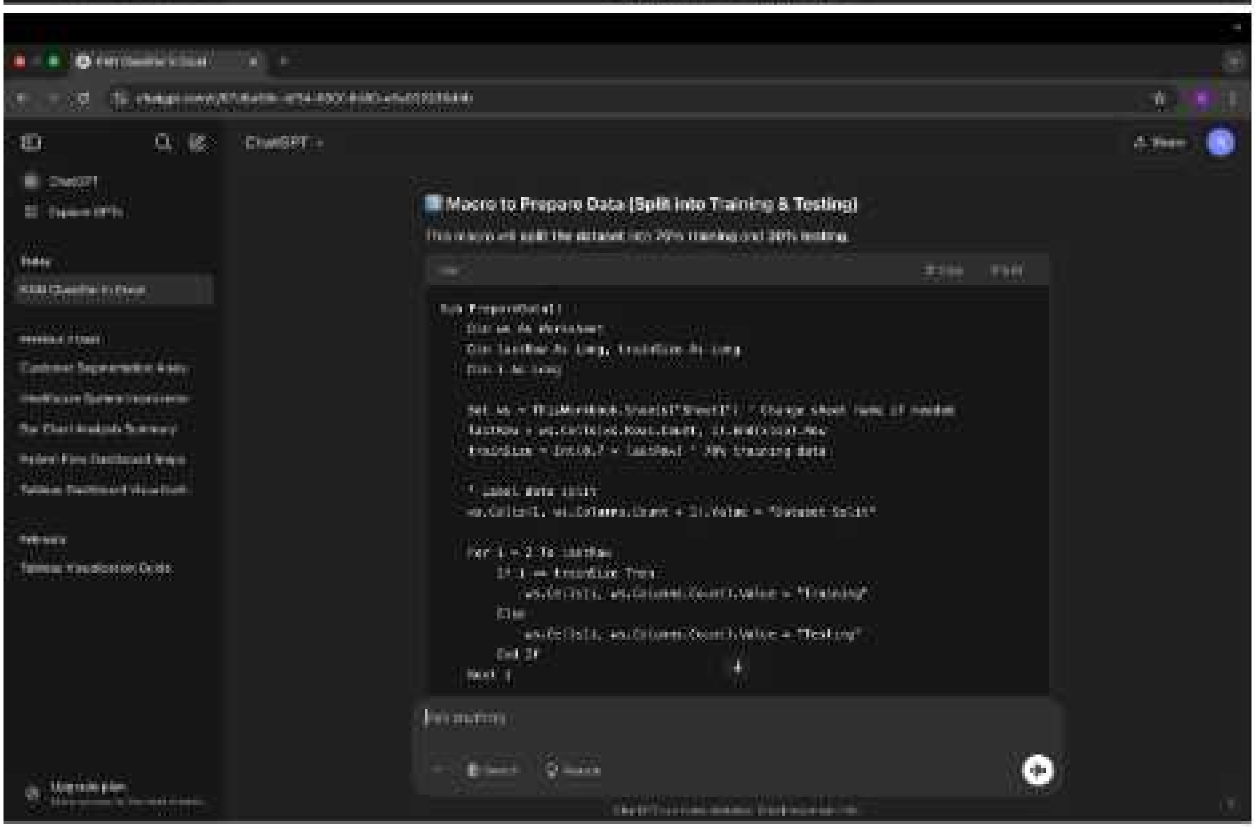
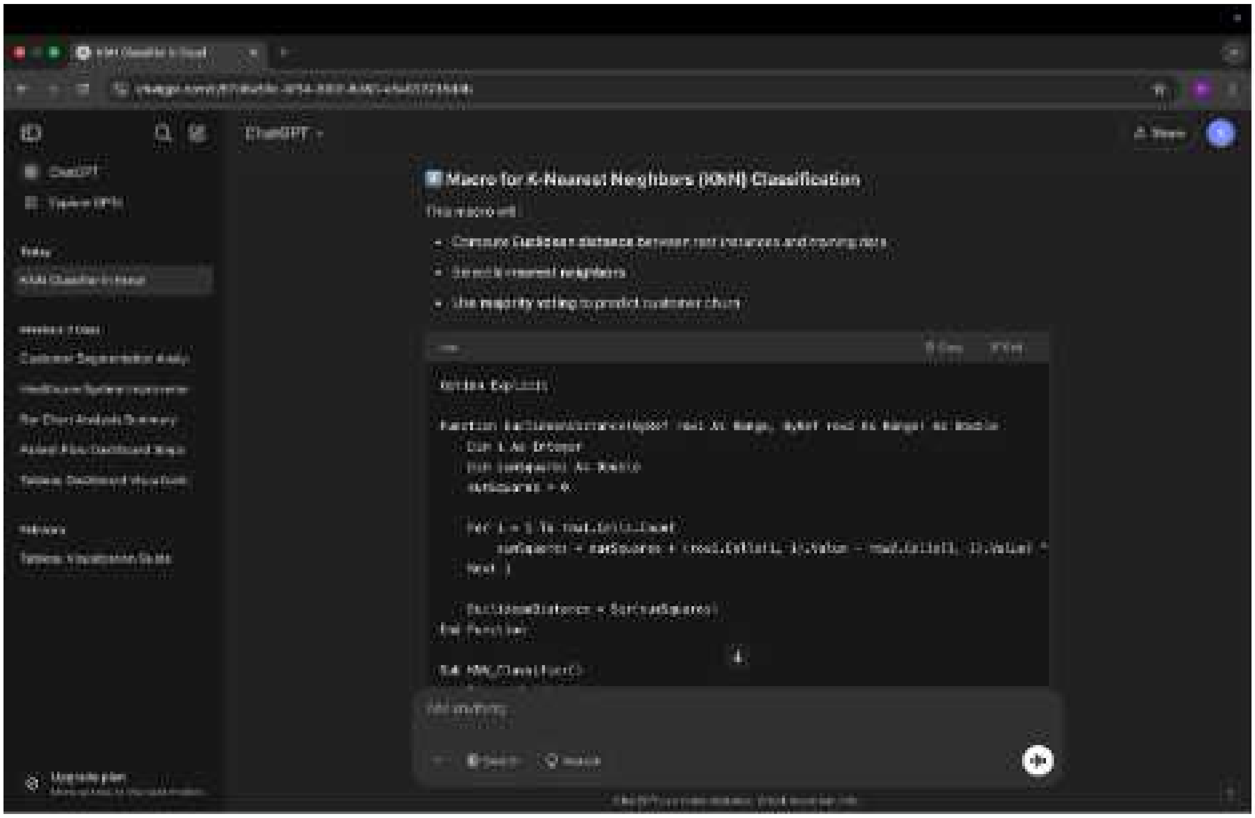
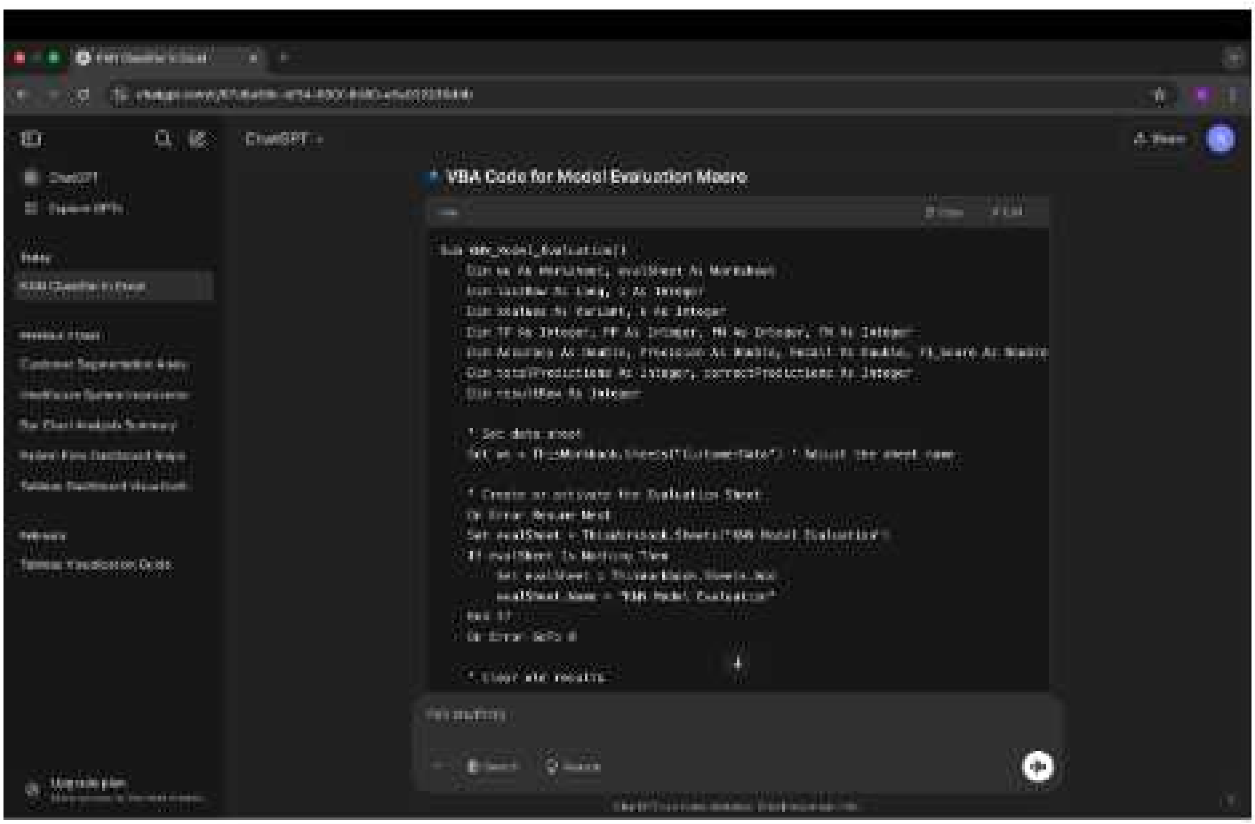
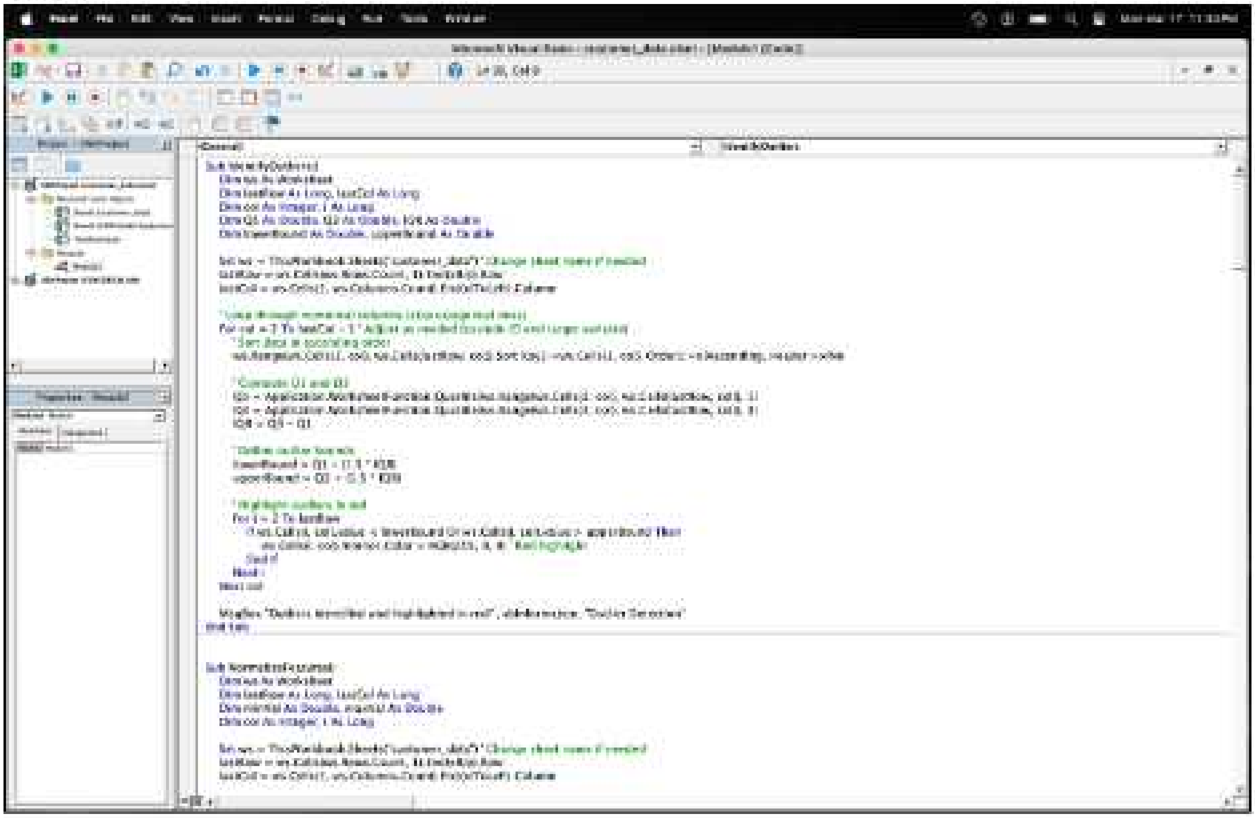
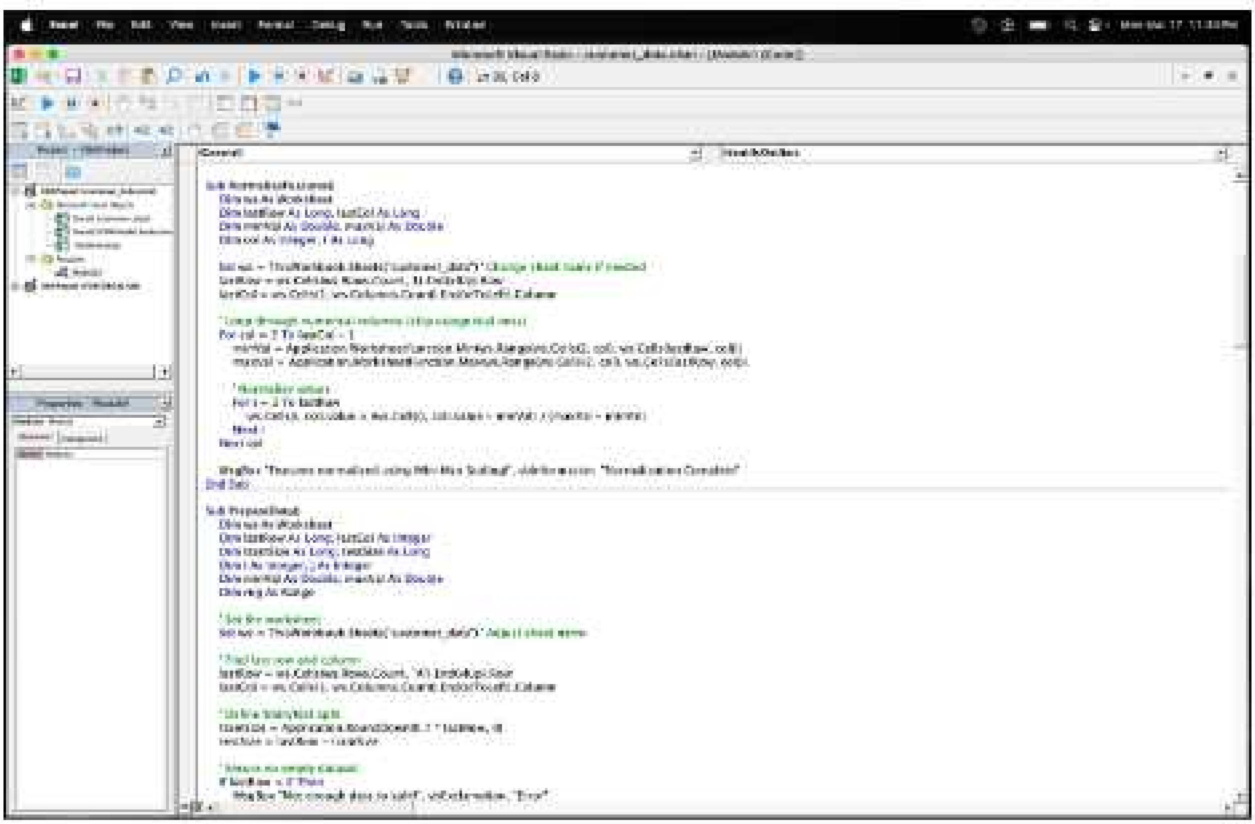
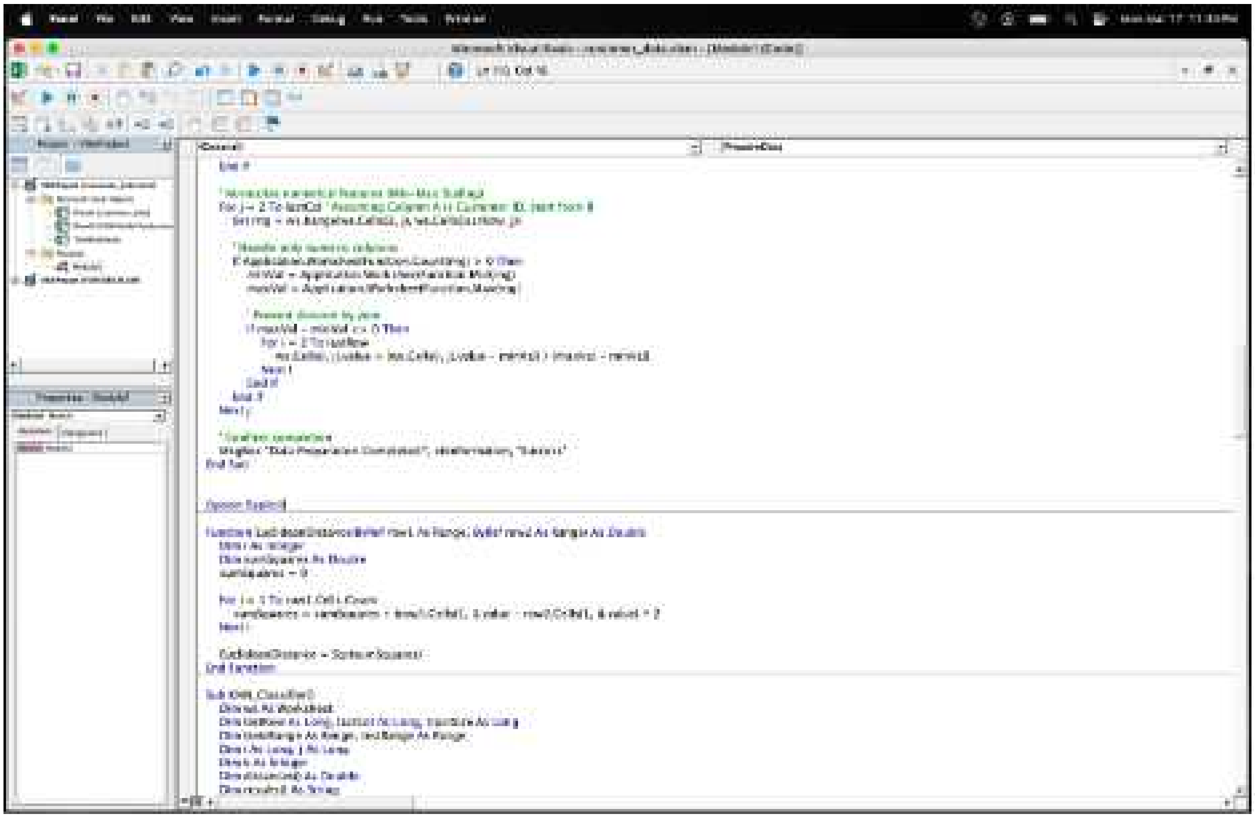
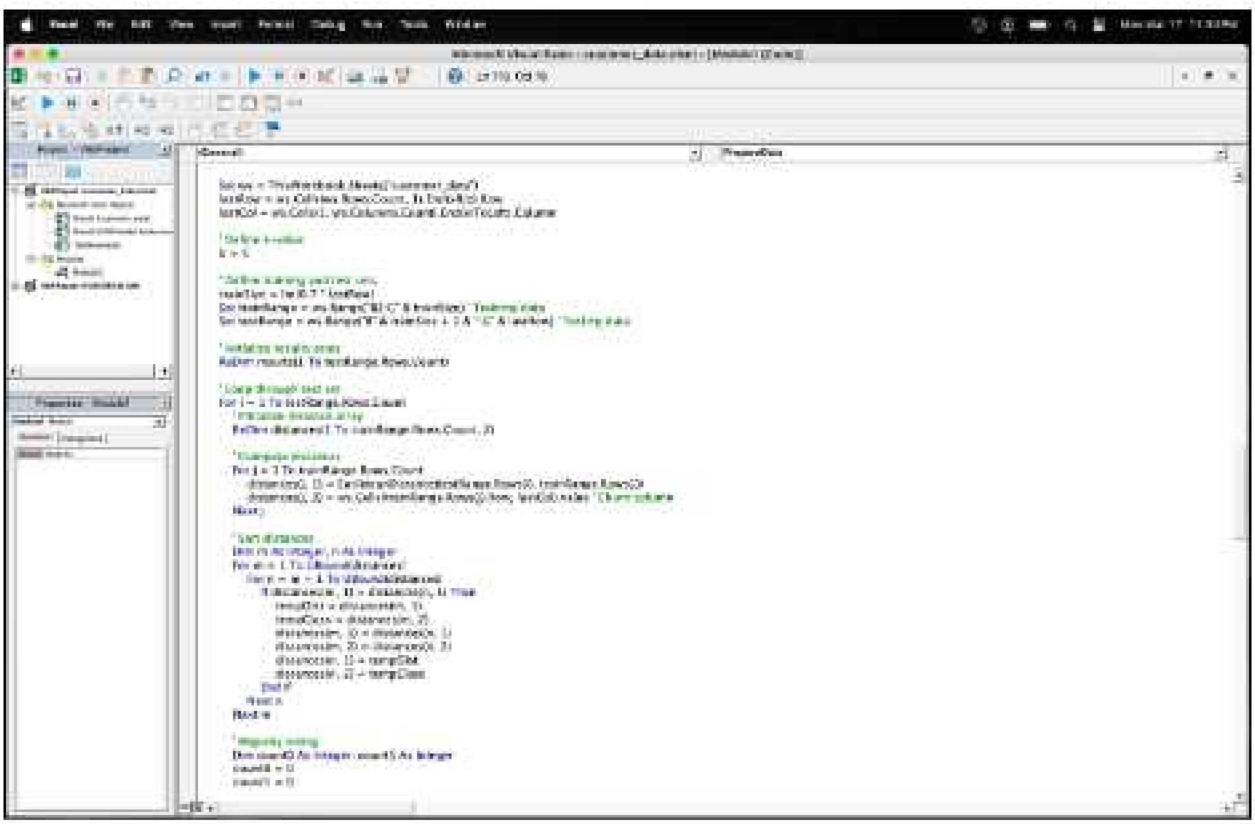
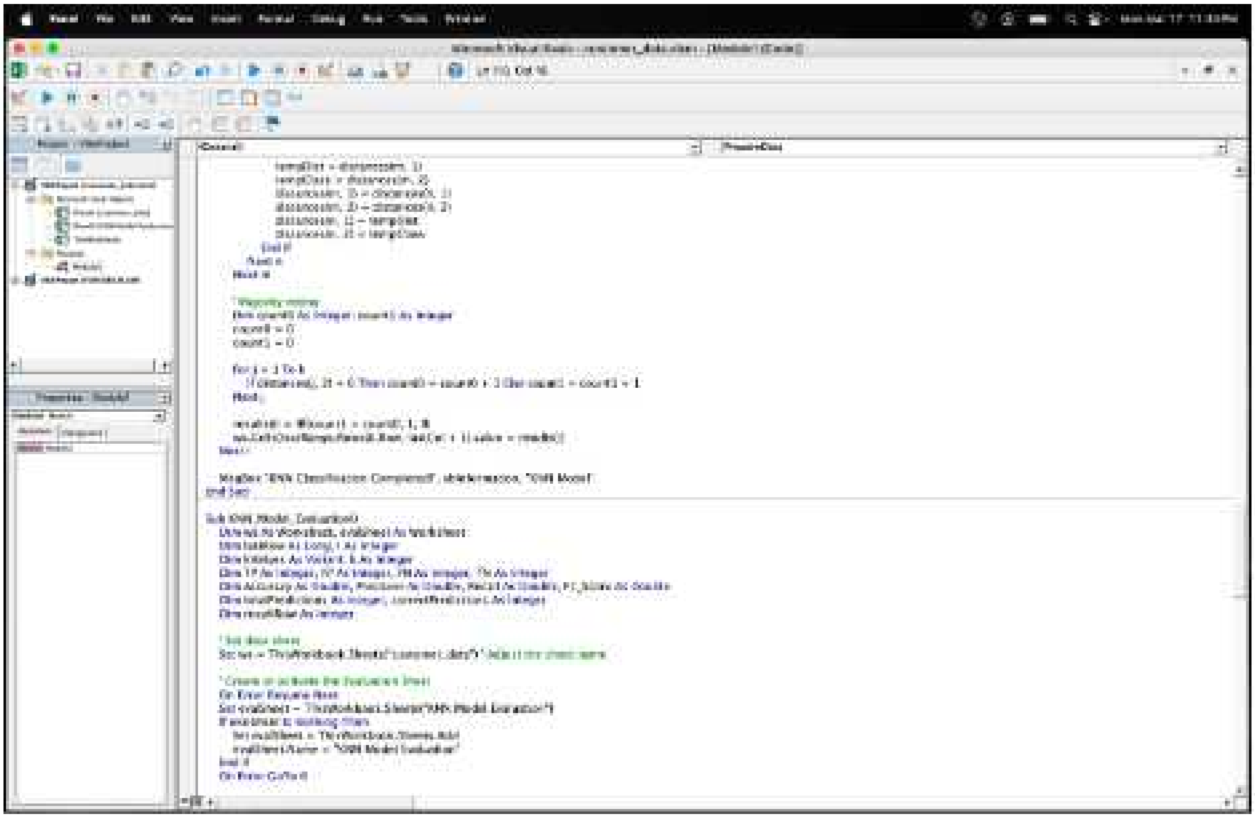
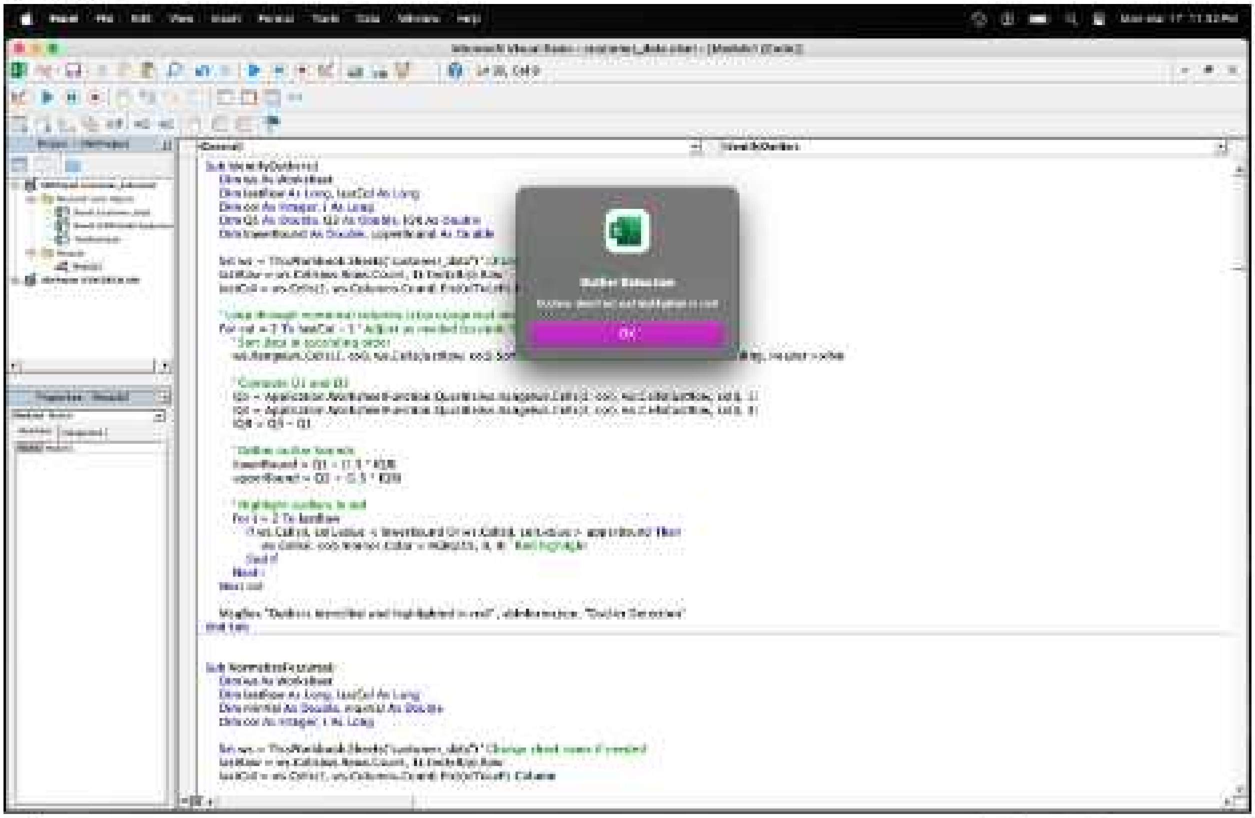
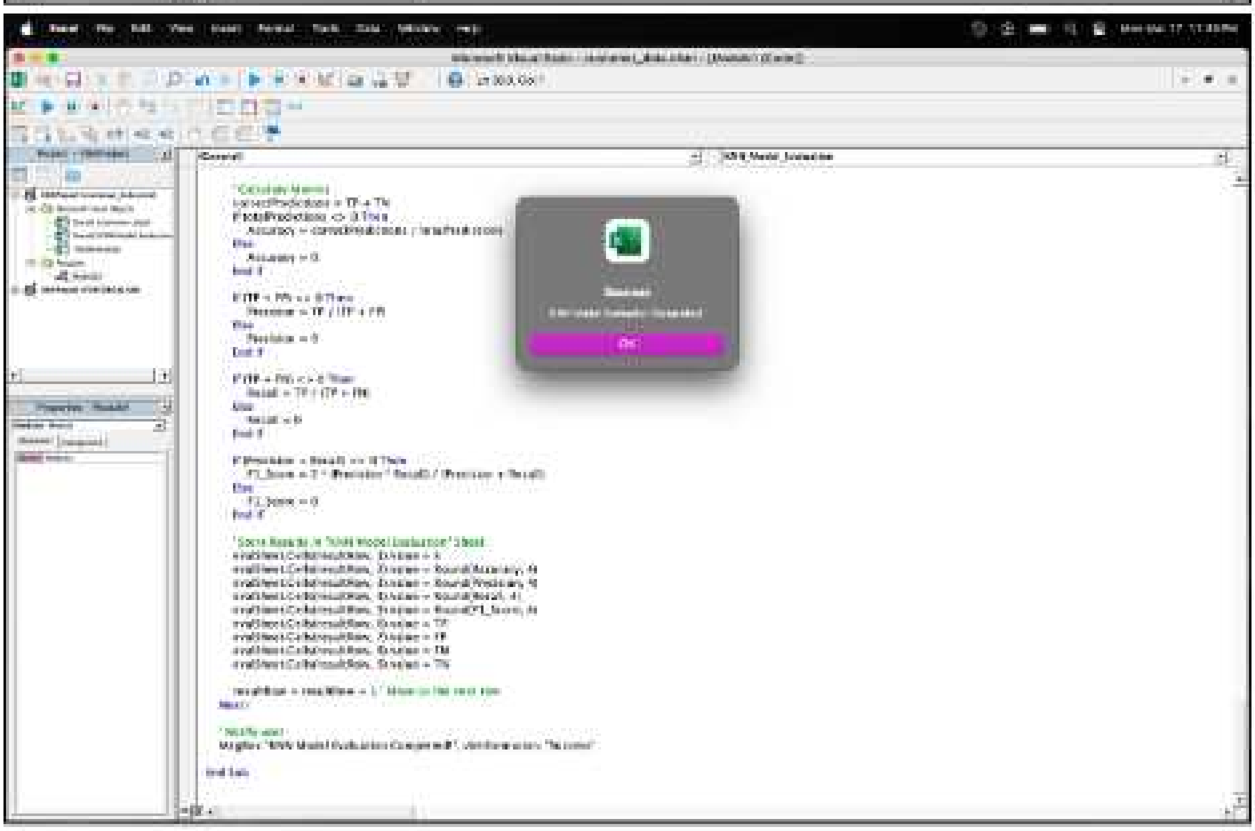
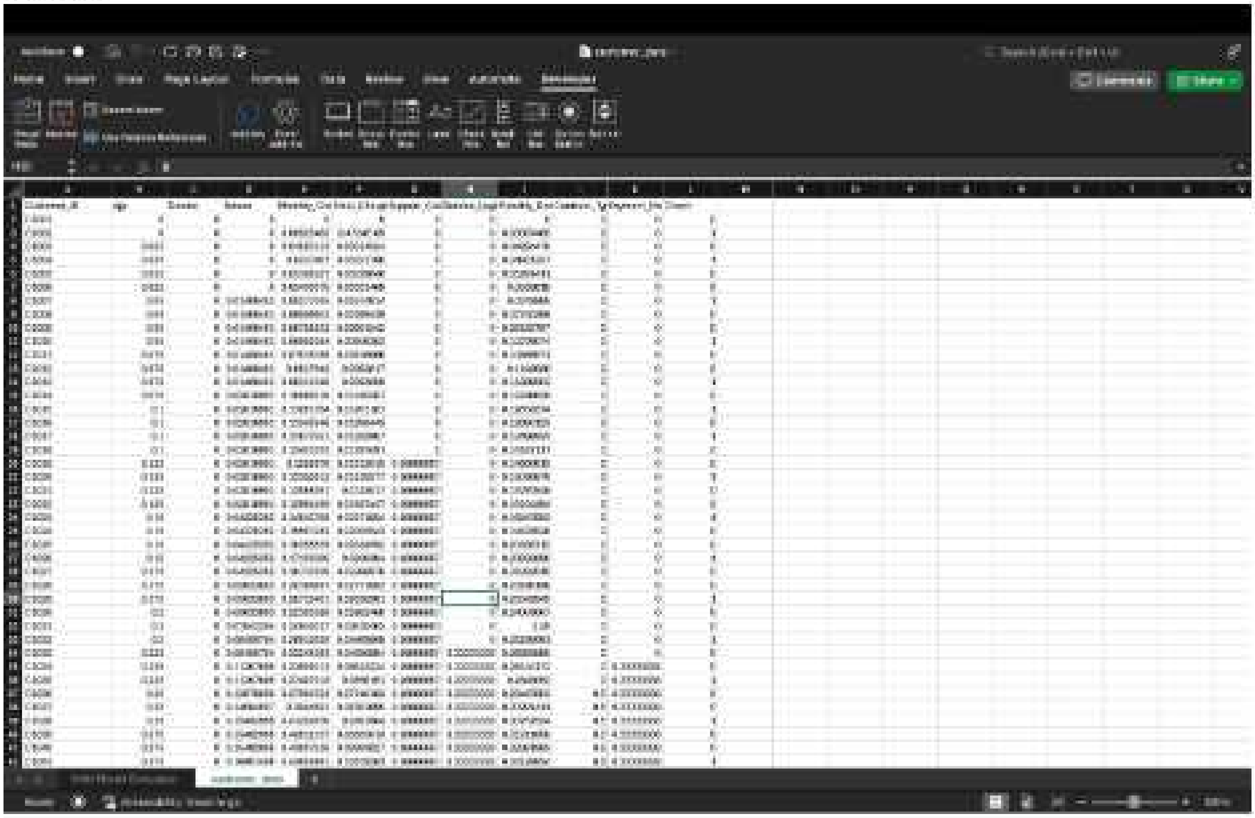
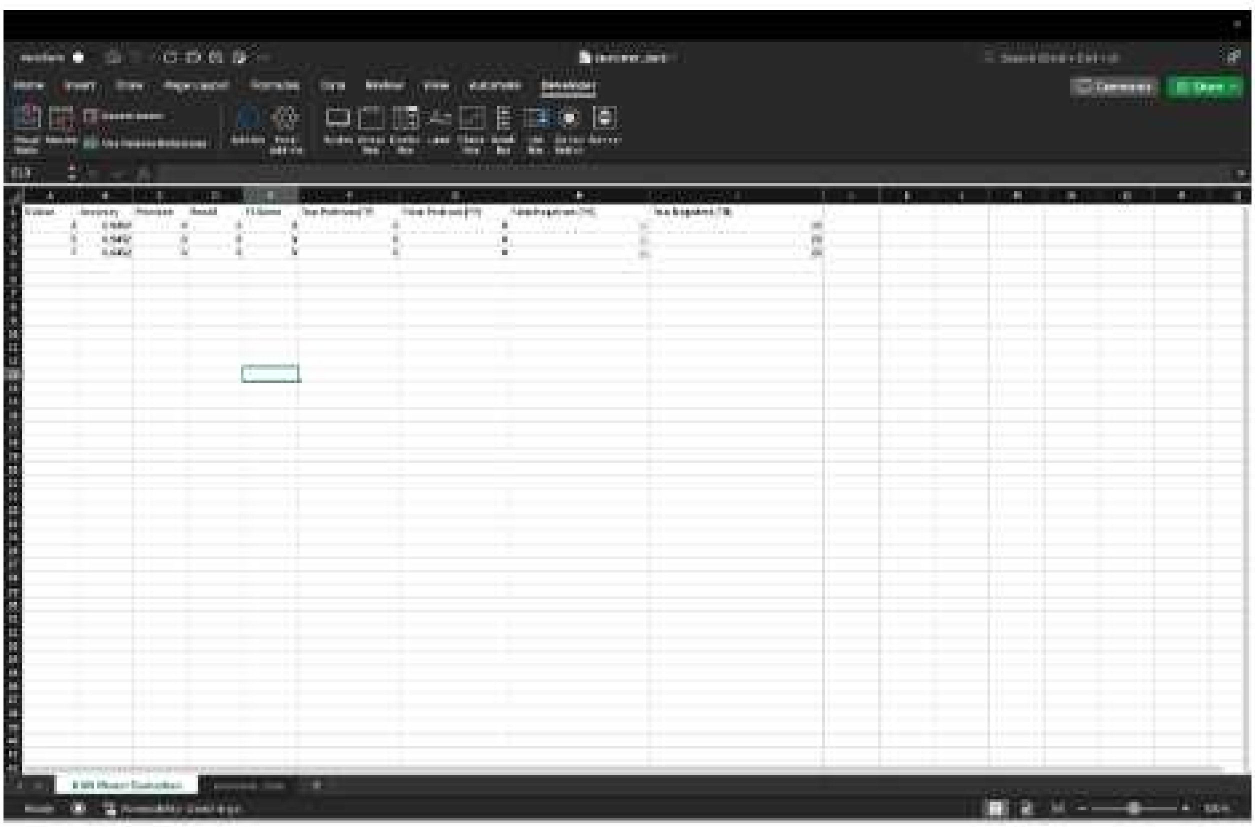
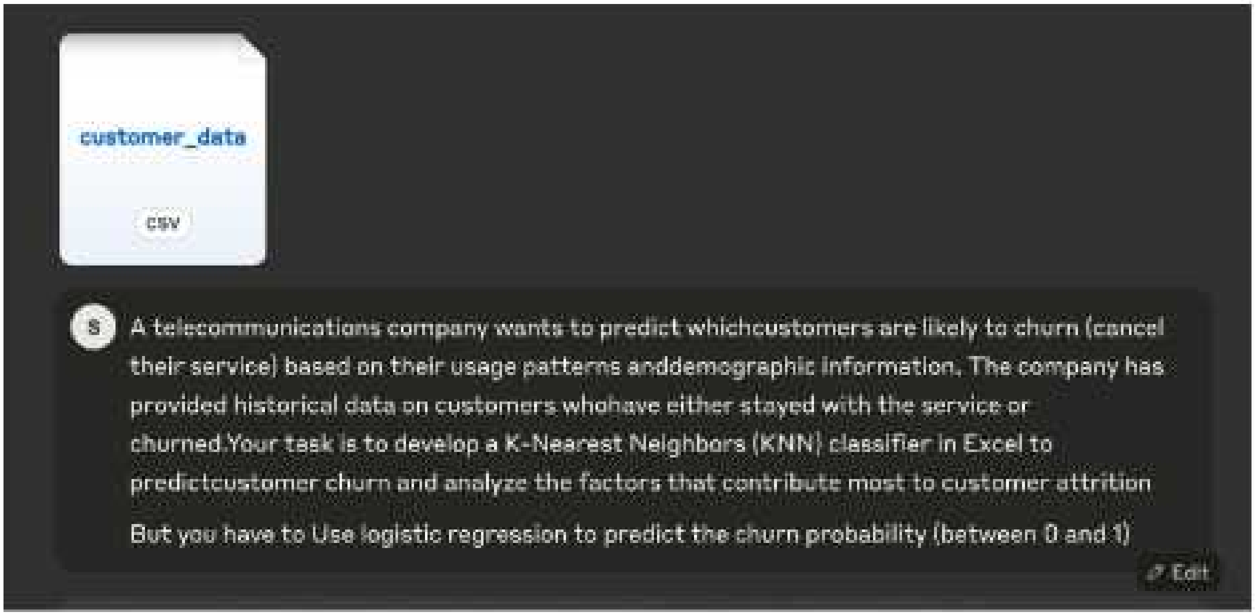
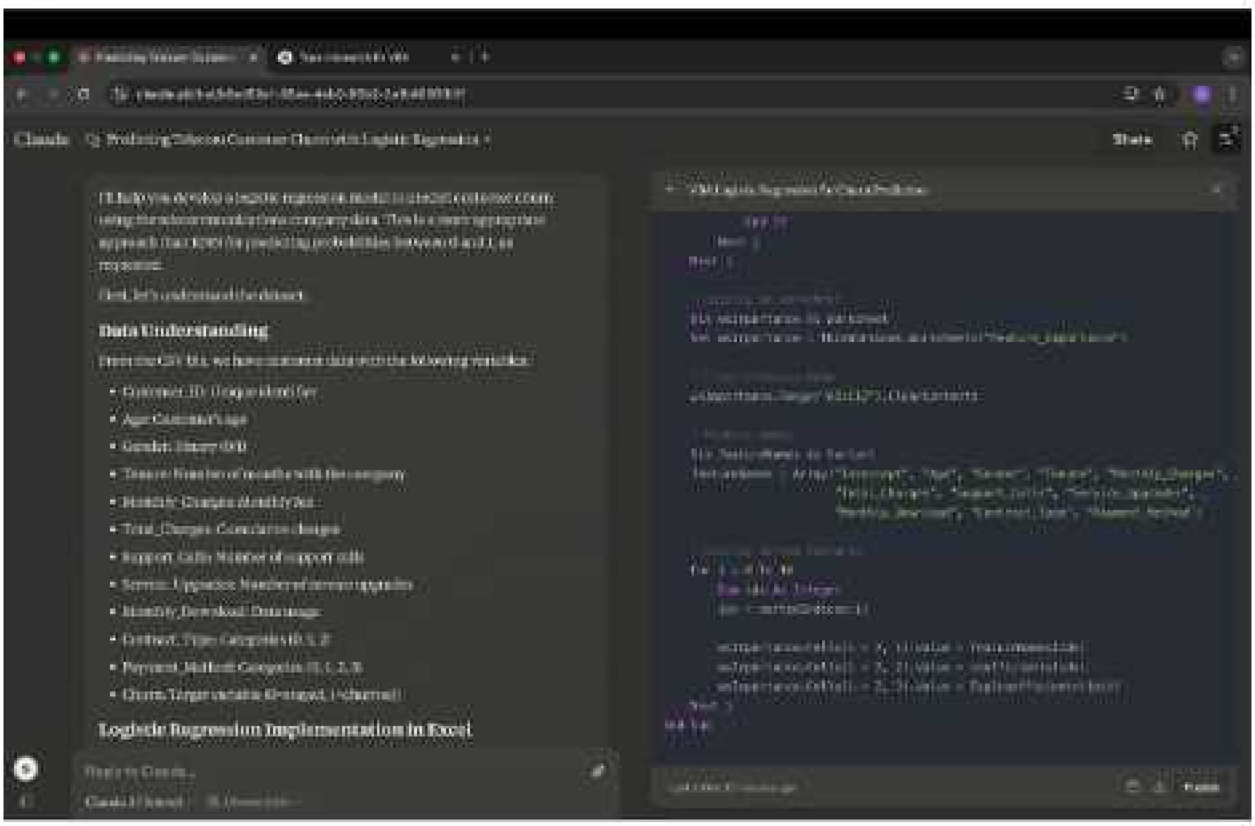
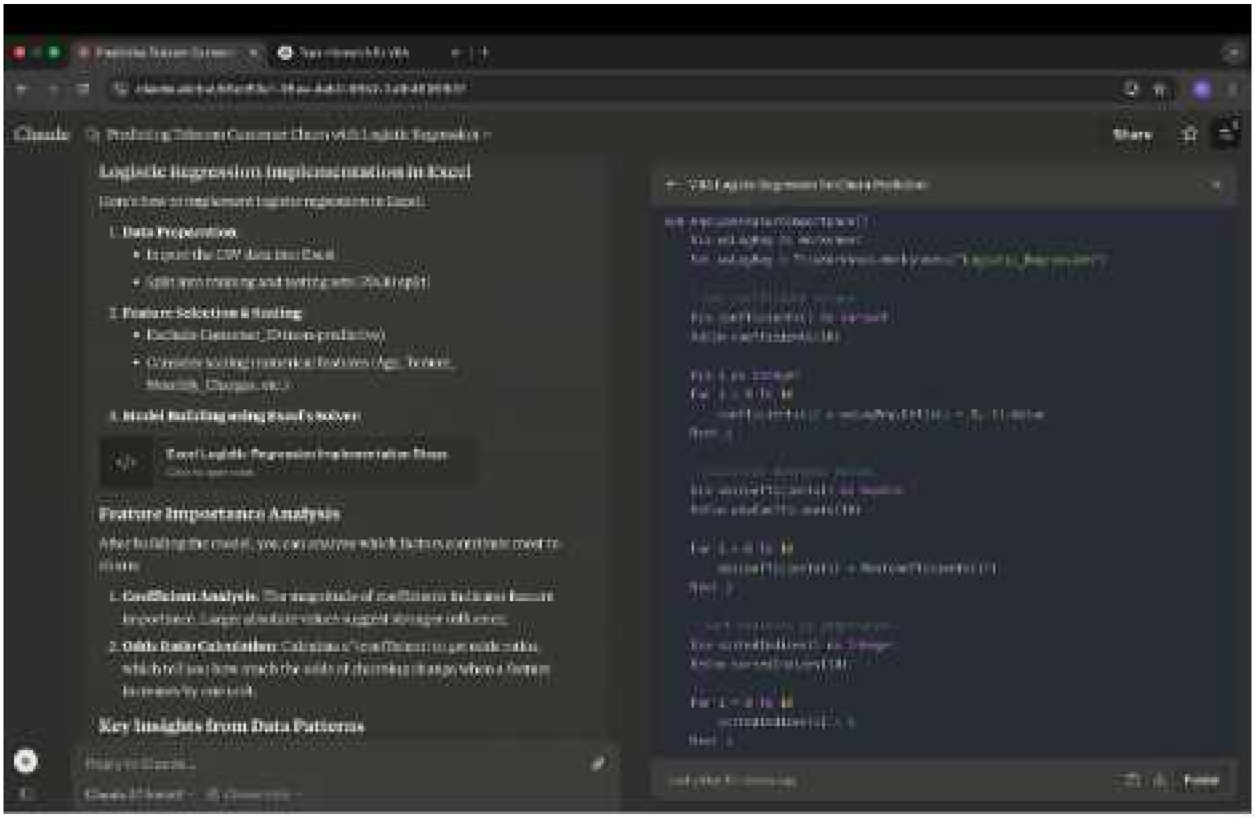
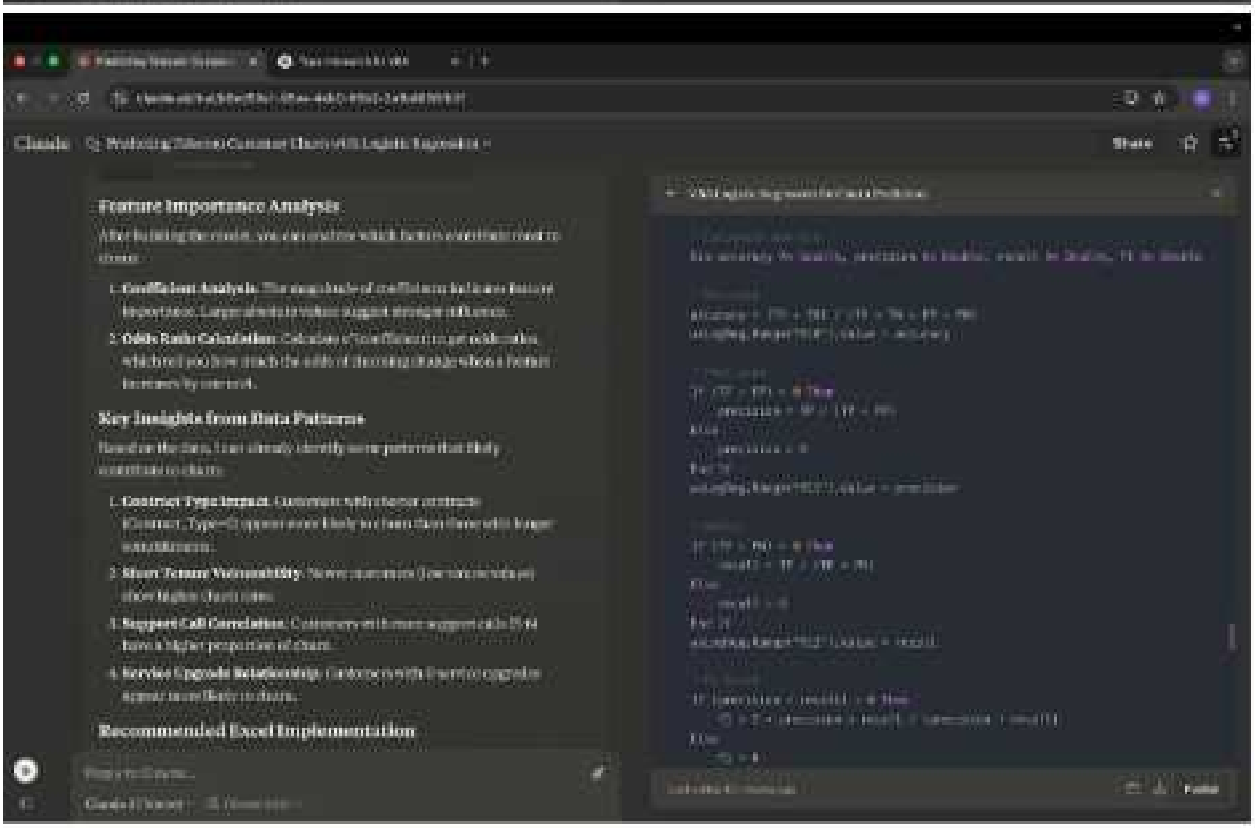
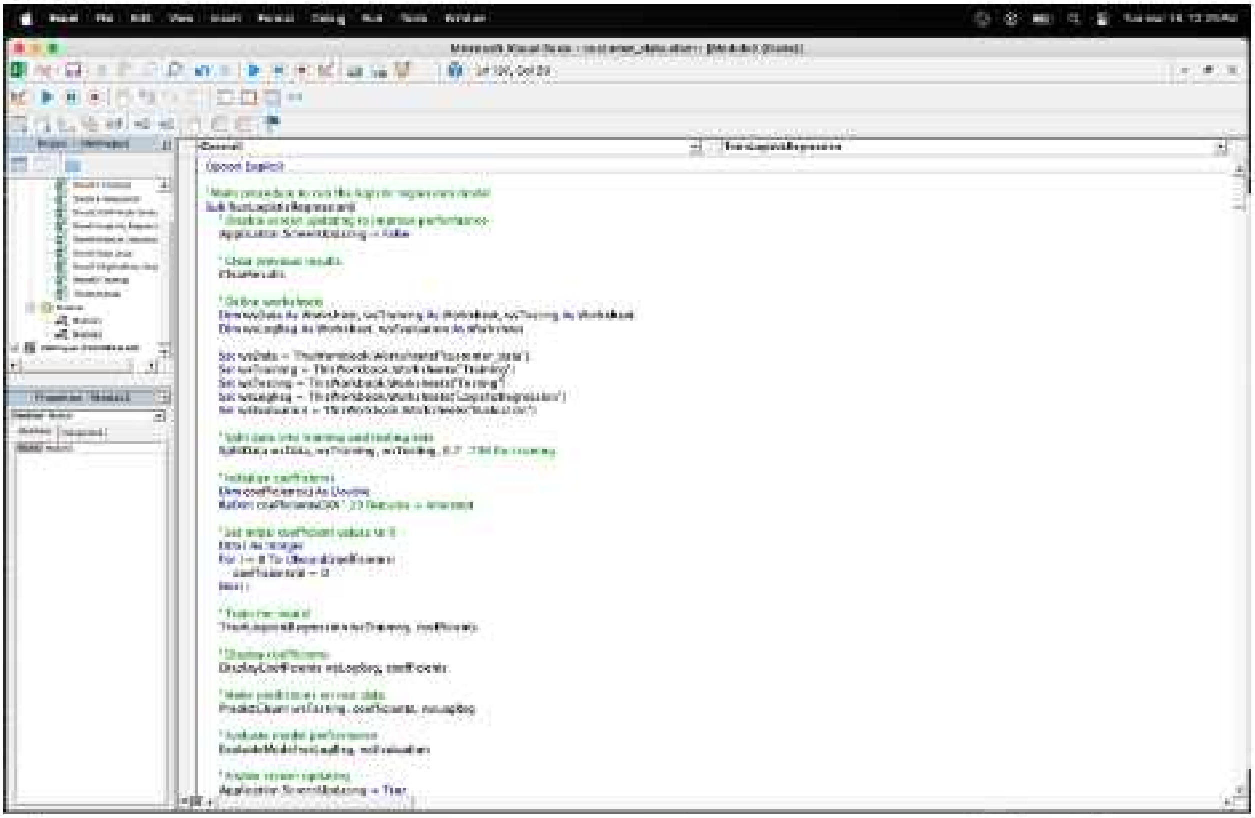
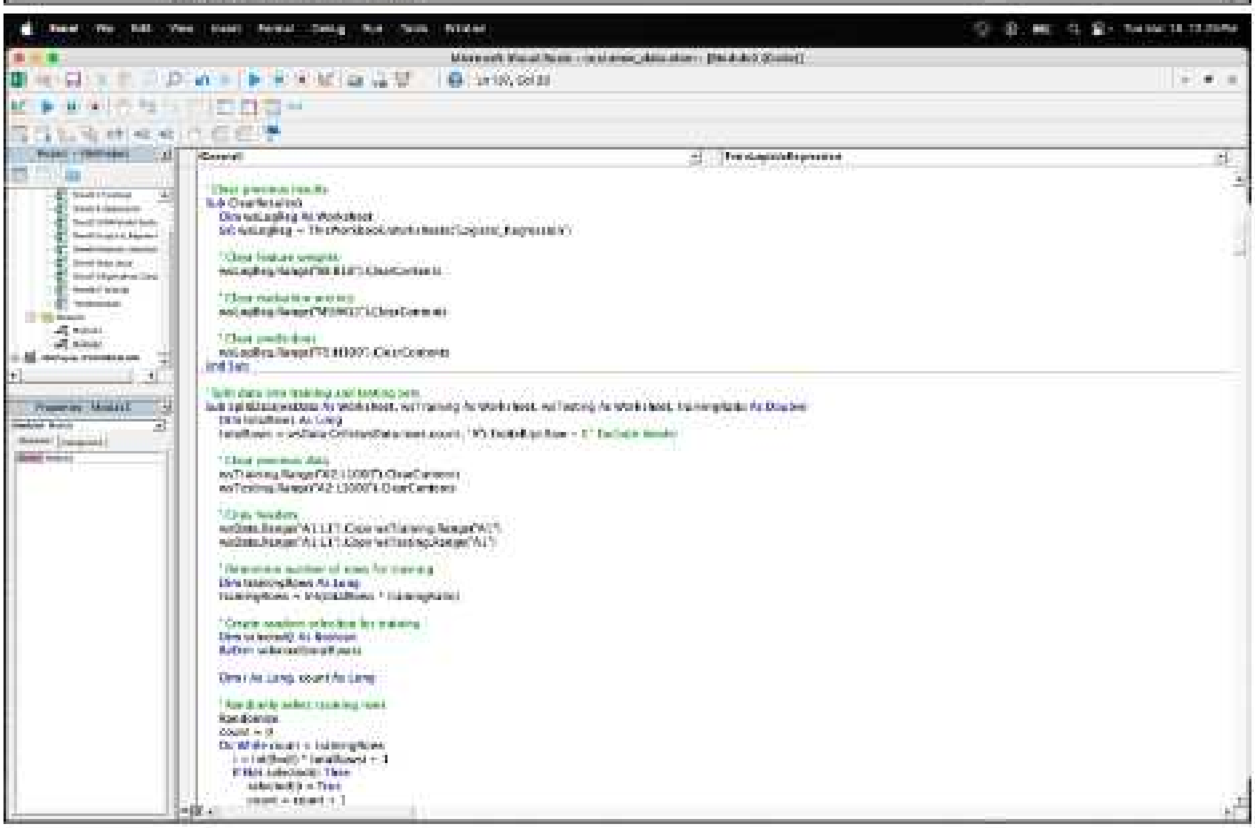
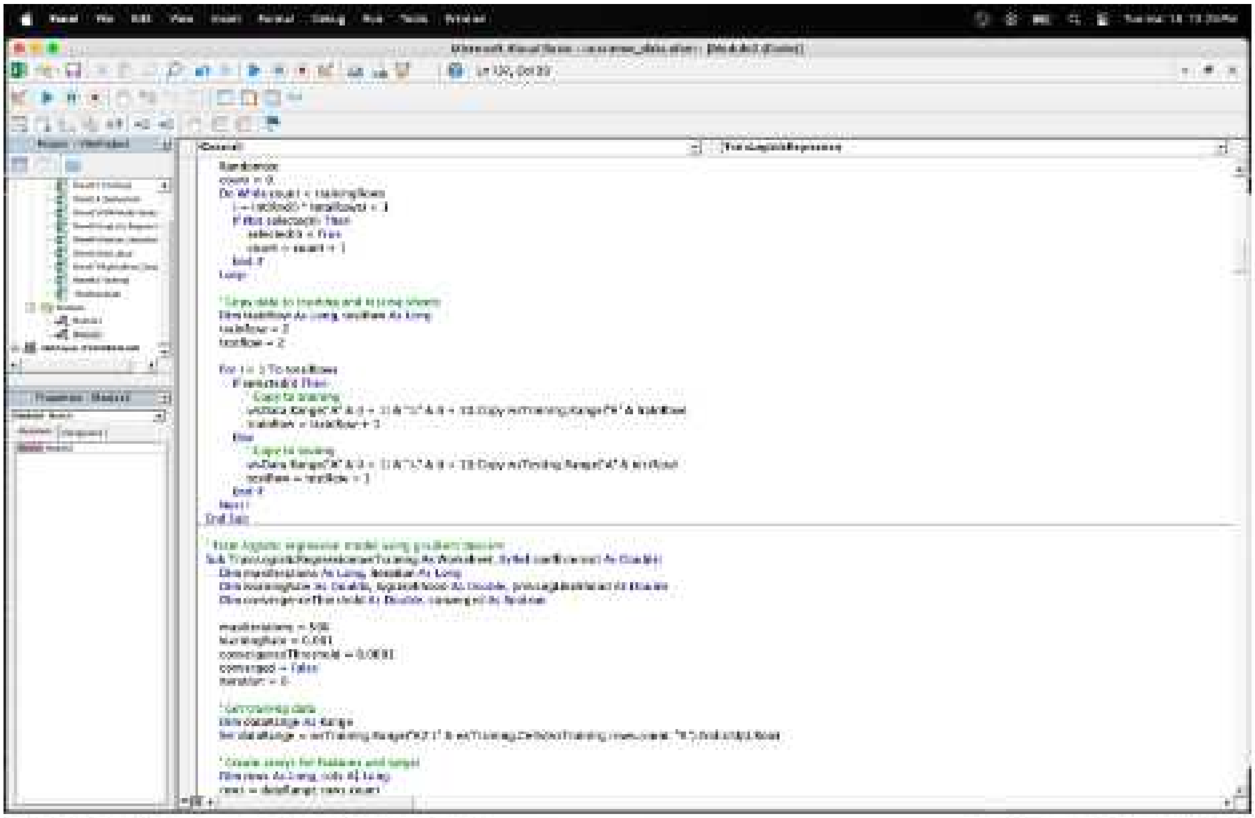
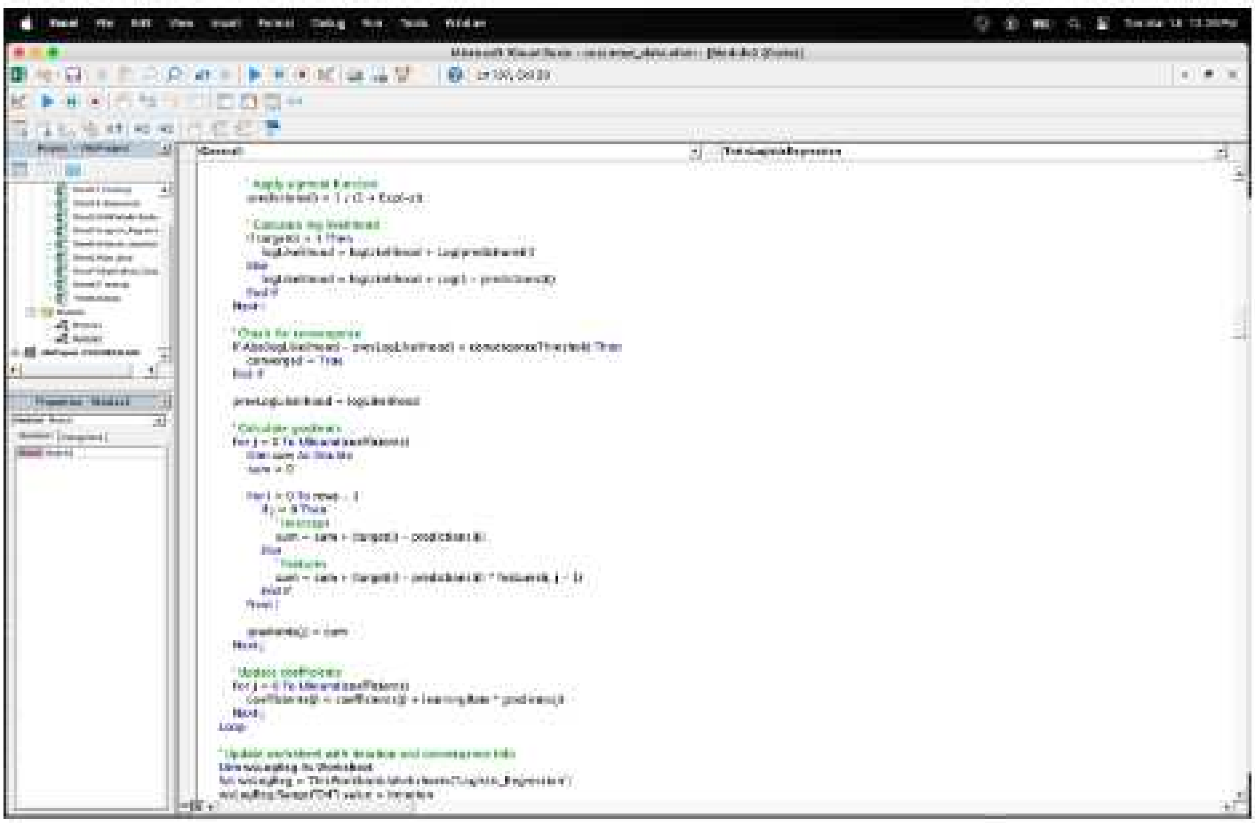
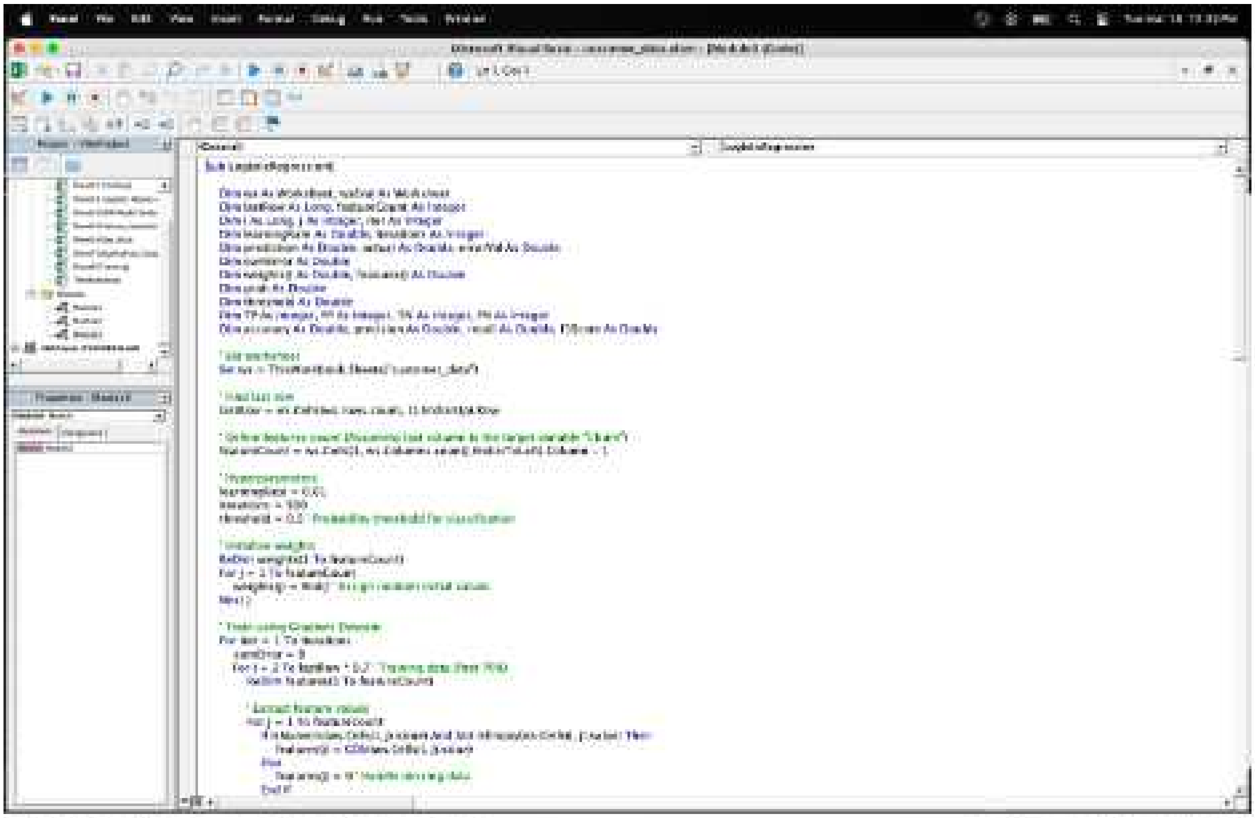
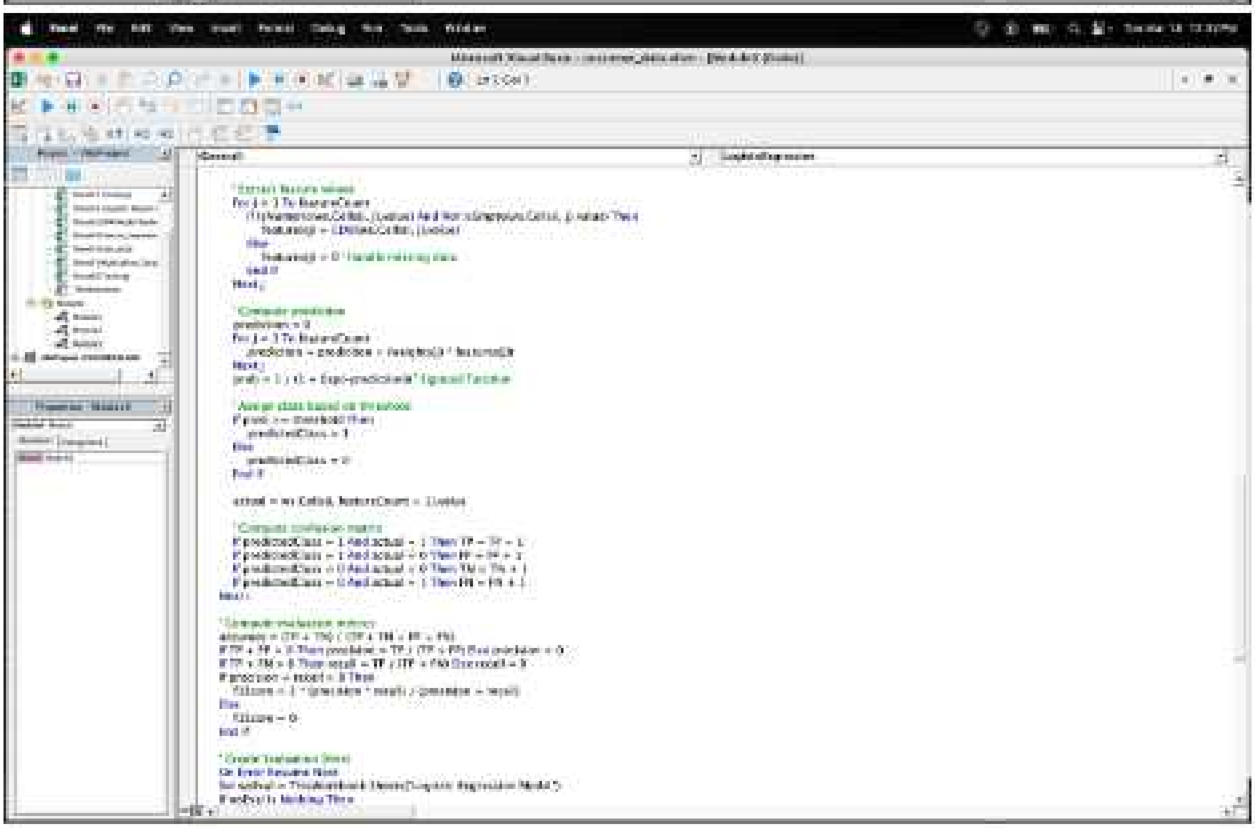
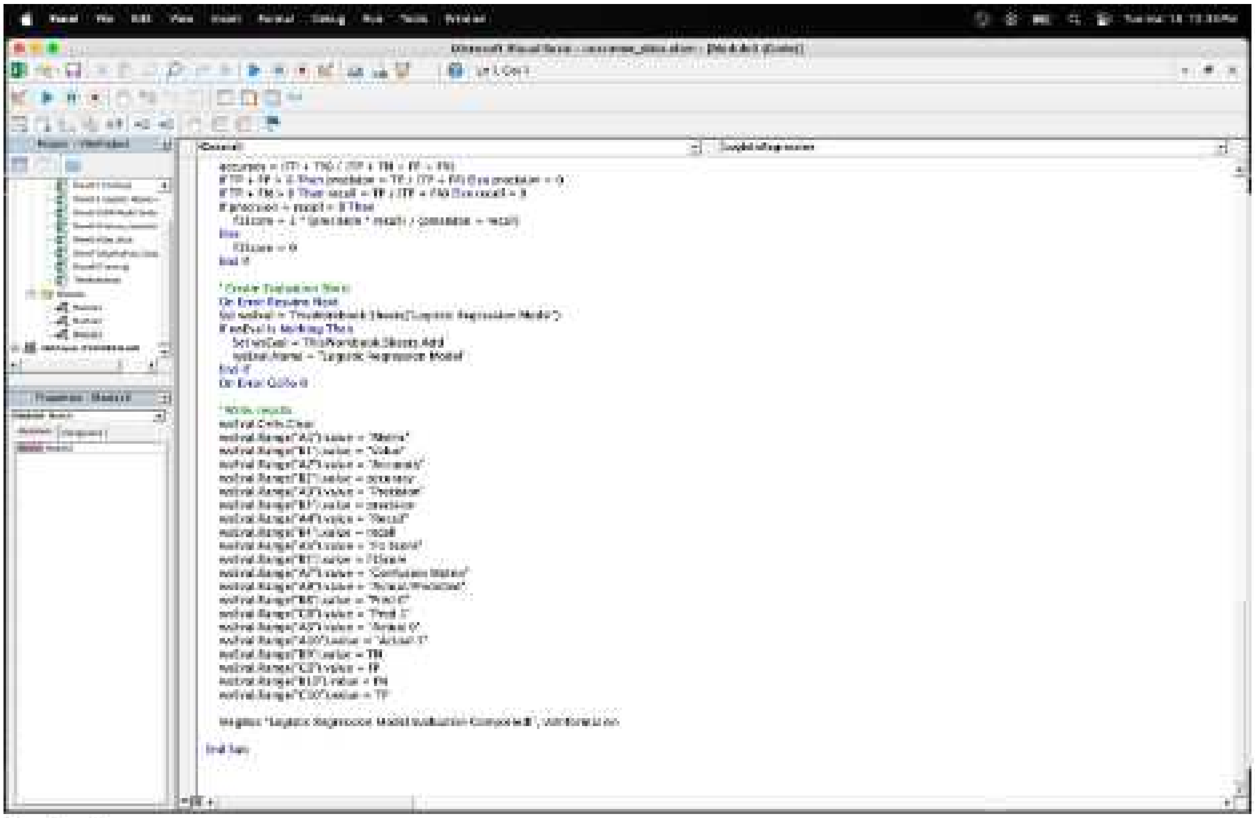
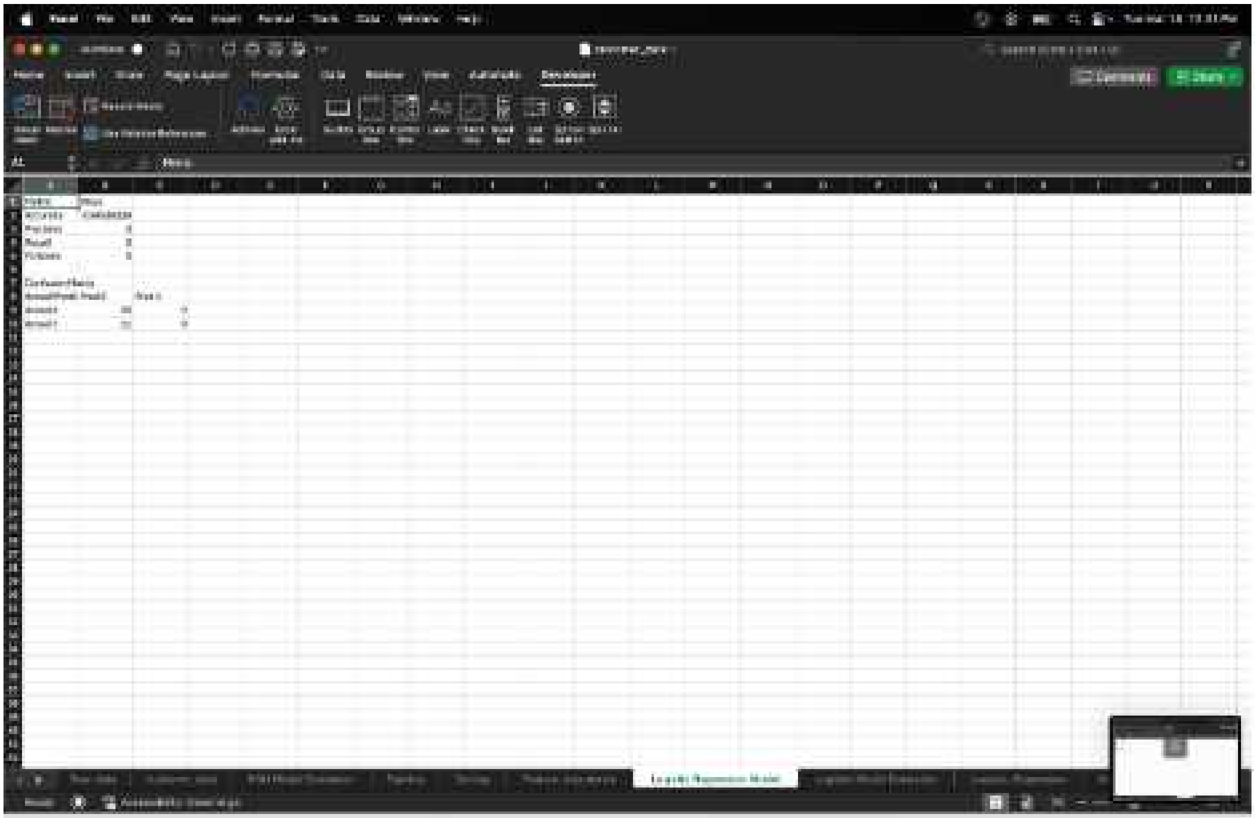
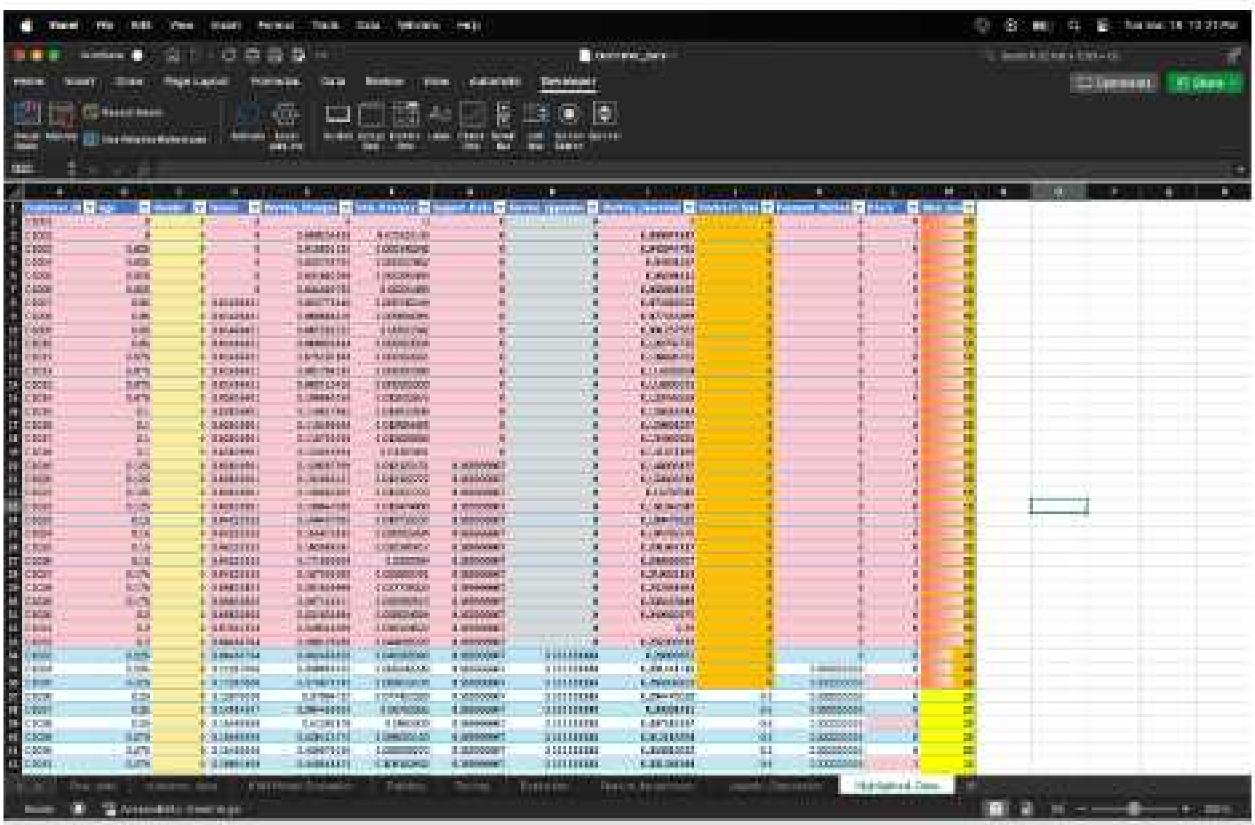
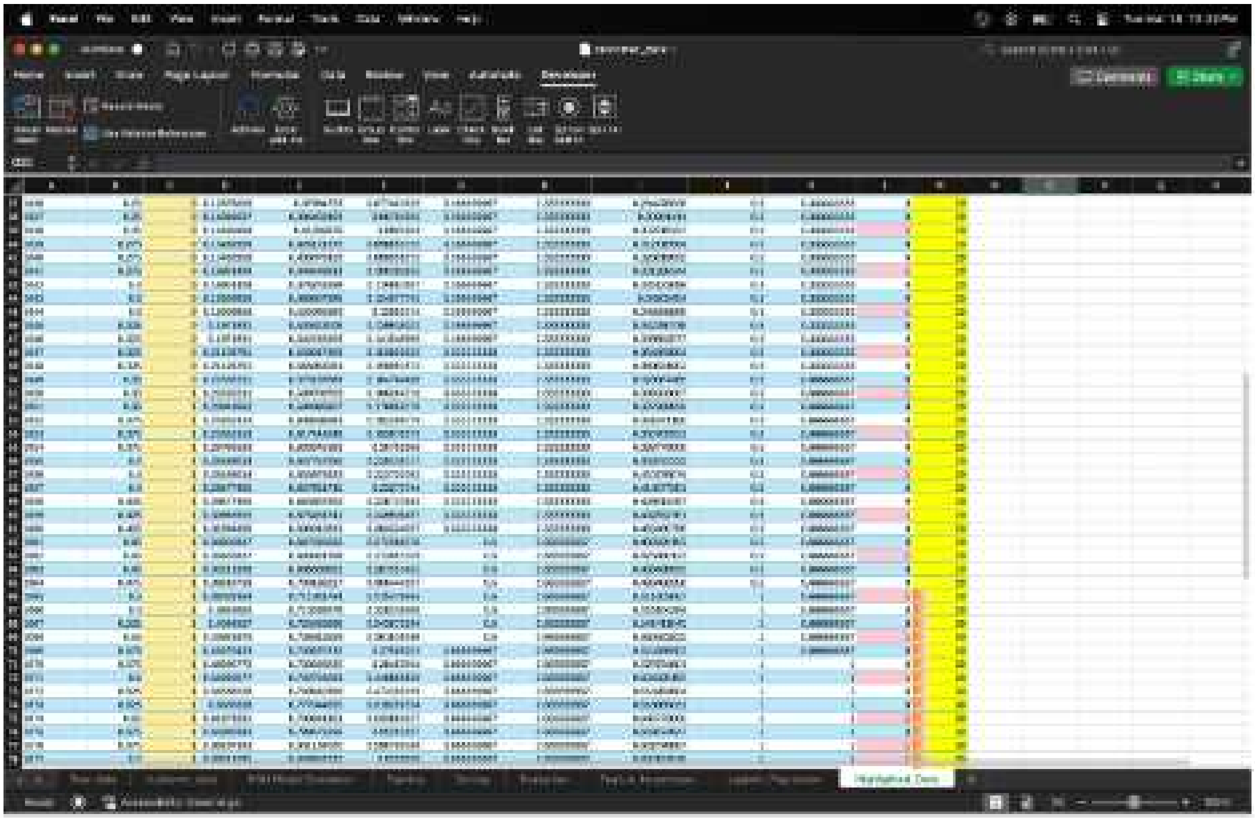
```{r}
# Lab 17_4F
# Load required library
library(caret)

# Create confusion matrix
conf_matrix <- confusionMatrix(data = as.factor(knn_pred),
                               reference = as.factor(test_labels))

# Display confusion matrix
conf_matrix
```
```

Output:



g) What is the accuracy? Is it a good performance?

```
    usa       1   8

               Accuracy : 0.7857
                 95% CI : (0.492, 0.9534)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : 0.4001

                  Kappa : 0.5116

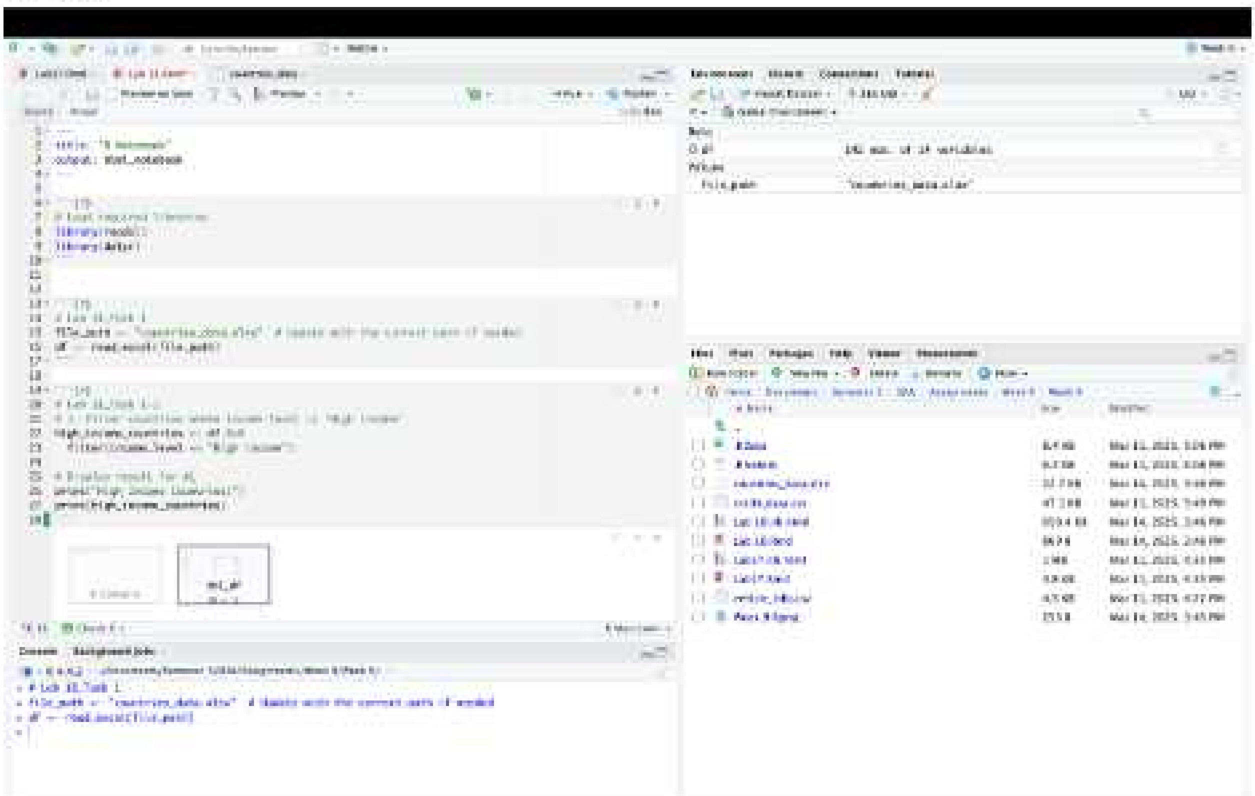
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.7500
            Specificity : 0.8000
         Pos Pred Value : 0.6000
         Neg Pred Value : 0.8889
             Prevalence : 0.2857
         Detection Rate : 0.2143
   Detection Prevalence : 0.3571
      Balanced Accuracy : 0.7750

       'Positive' Class : other
```

Ans: - The accuracy of the model is 0.7857 (78.57%)

- The model performs moderately well but may not be significantly better than random guessing.

- If the application requires high precision or recall, further model tuning or feature engineering might be needed.

5. **Gen AI Prompt Engineering**:
A telecommunications company wants to predict which customers are likely to churn (cancel their service) based on their usage patterns and demographic information. The company has provided historical data on customers who have either stayed with the service or churned. Your task is to develop a K-Nearest Neighbors (KNN) classifier in Excel to predict customer churn and analyze the factors that contribute most to customer attrition.

Bot Used: ChatGPT

Prompt:



Bot:


ChatGPT
Step 1: Enable Developer Tab
Step 2: Open VBA Editor
Step 3: Load Data into Excel
Step 4: Write VBA Code for KNN


ChatGPT
Step 5: Run the VBA Code
Step 6: Evaluate Model Performance
Step 7: Generate Business Recommendations


ChatGPT
Steps Covered in the VBA Code
VBA Code for KNN in Excel
Option Explicit


ChatGPT
Option Explicit




Macro to Identify Outliers (IQR Method)


ChatGPT
Macro to Normalize Features (Min-Max Scaling)


ChatGPT
Macro to Prepare Data (Split into Training & Testing)


ChatGPT
Macro for K-Nearest Neighbors (KNN) Classification
Option Explicit
End Function
Sub KNN_Classifier()


ChatGPT
VBA Code for Model Evaluation Macro

Code:











Msgbox:





Output:





6. **Gen AI Prompt Engineering**: use logistic regression to predict the churn probability (between 0 and 1)

Bot Used: Claude.ai

Prompt:



Bot:




Logistic Regression Implementation in Excel
Data Preparation
Feature Selection & Scaling
Feature Importance Analysis
Coefficient Analysis
Odds Ratio Calculation
Key Insights from Data Patterns


Feature Importance Analysis
Coefficient Analysis
Odds Ratio Calculation
Key Insights from Data Patterns
Contract Type Impact
Short Tenure Vulnerability
Support Call Correlation
Recommended Excel Implementation

Code:















Output:







Lab 18

Task 1:

Use the 'countries_data.xlsx' dataset.

Code:

```
```{r}
# Load required libraries
library(readxl)
library(dplyr)
```

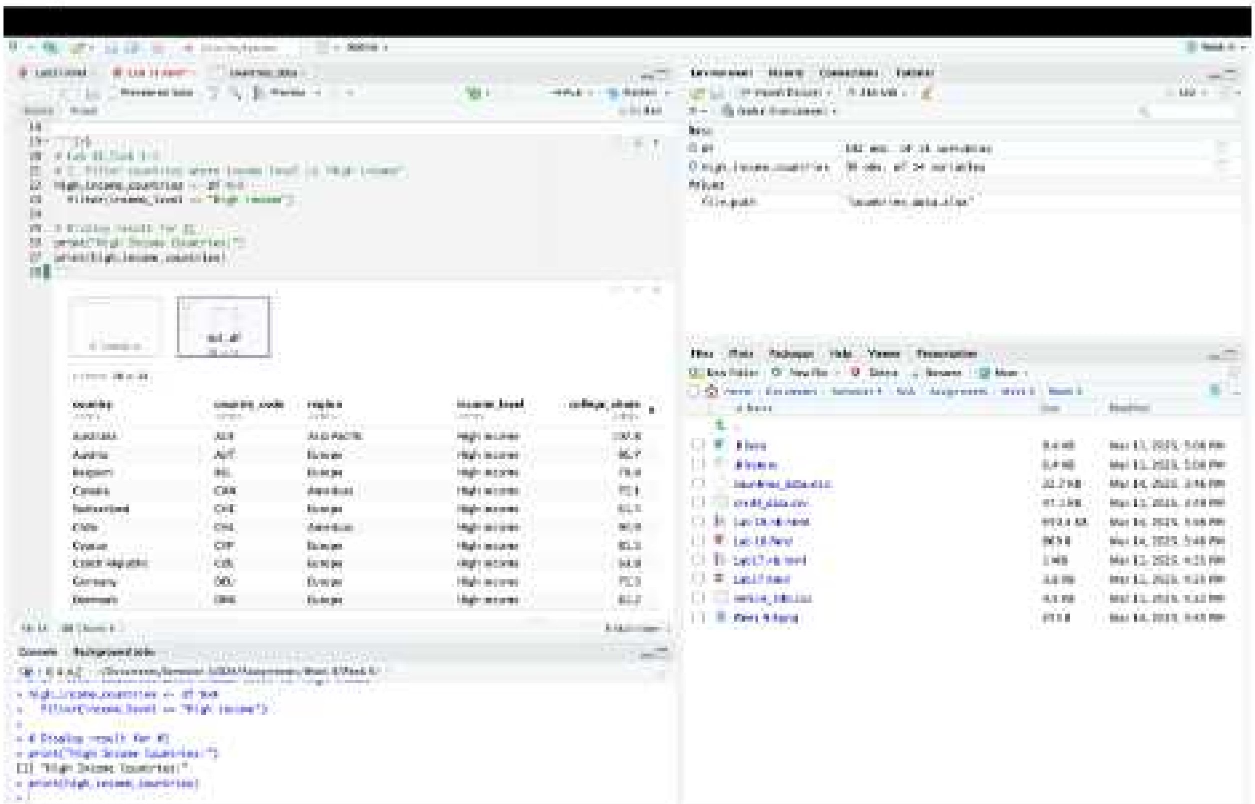
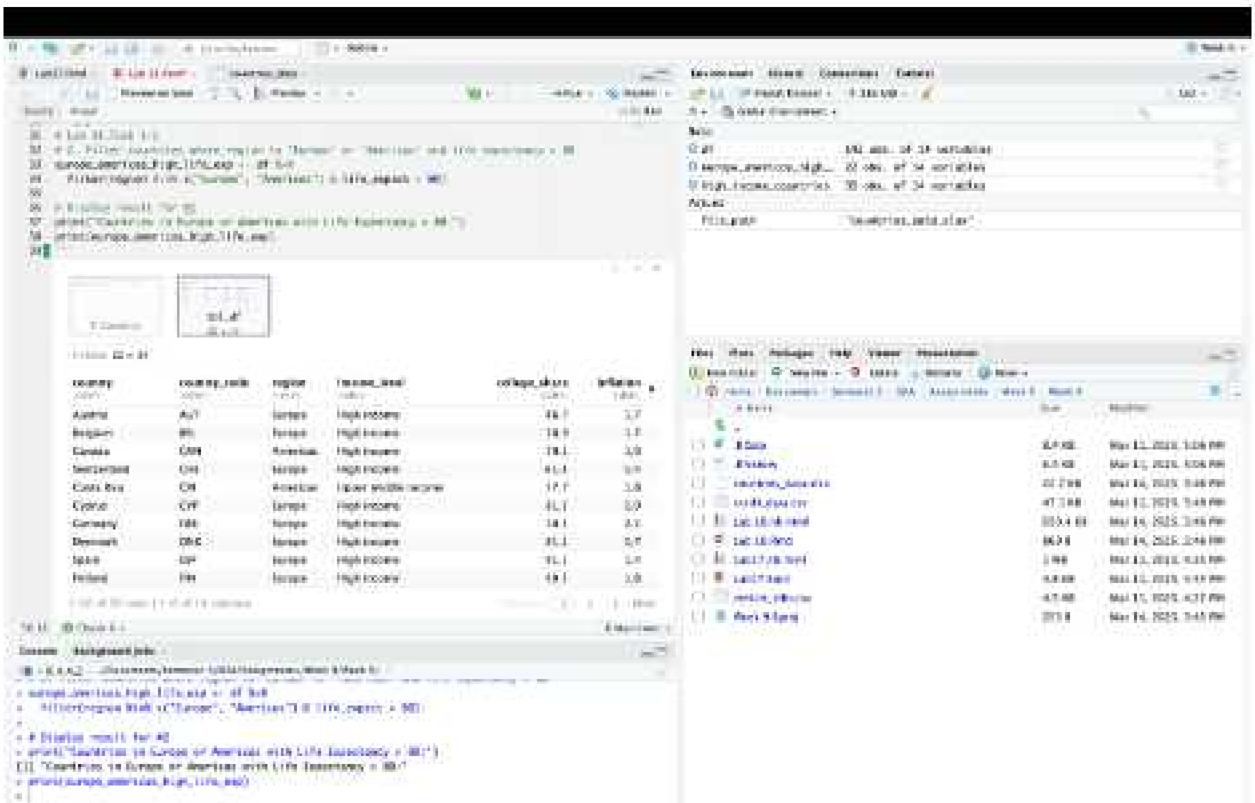
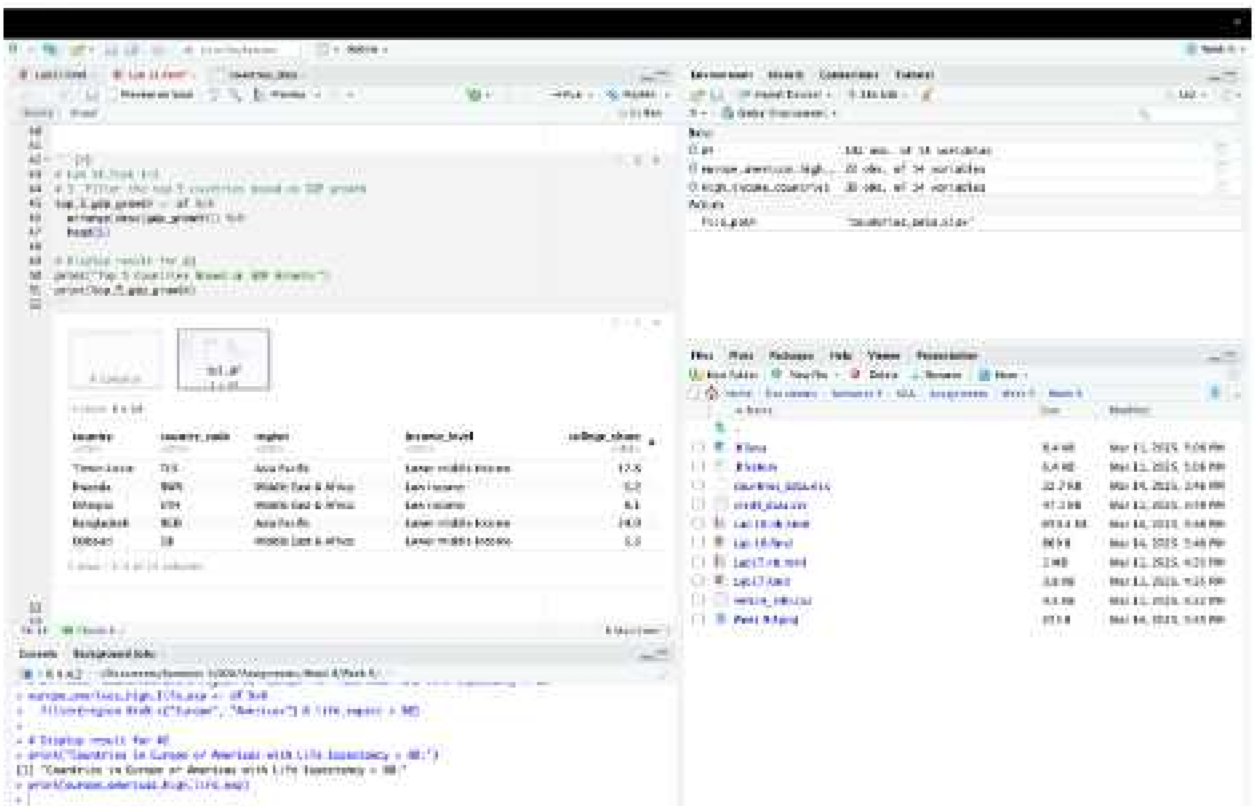
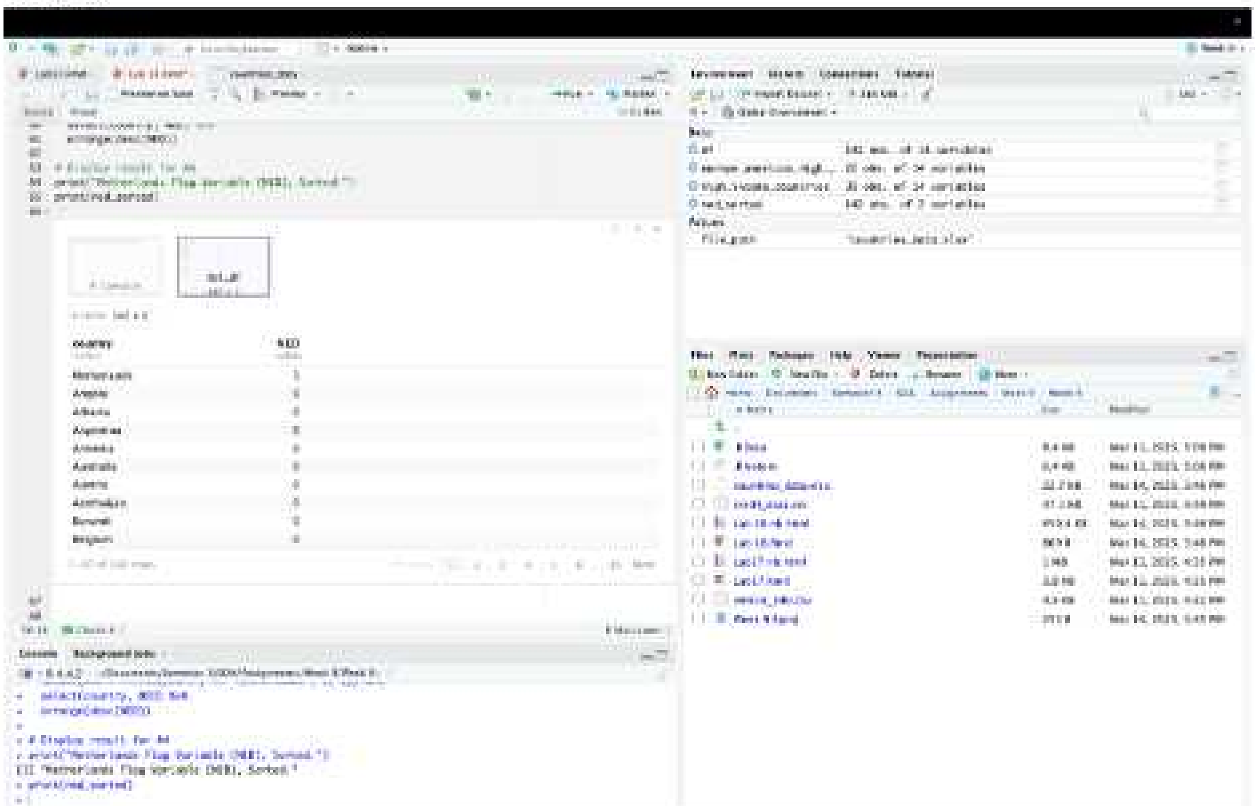

```{r}
# Lab 18_Task 1
file_path <- "countries_data.xlsx"  # Update with the correct path if needed
df <- read_excel(file_path)
```
```

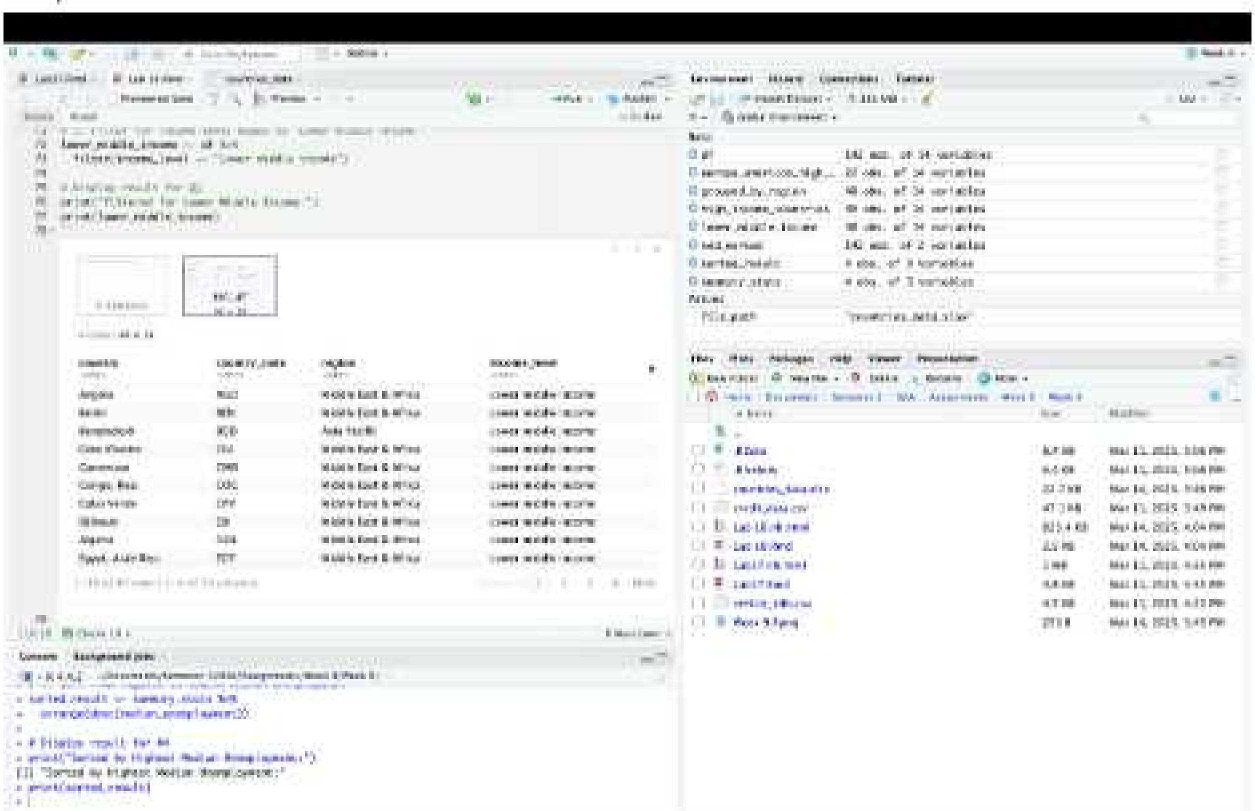
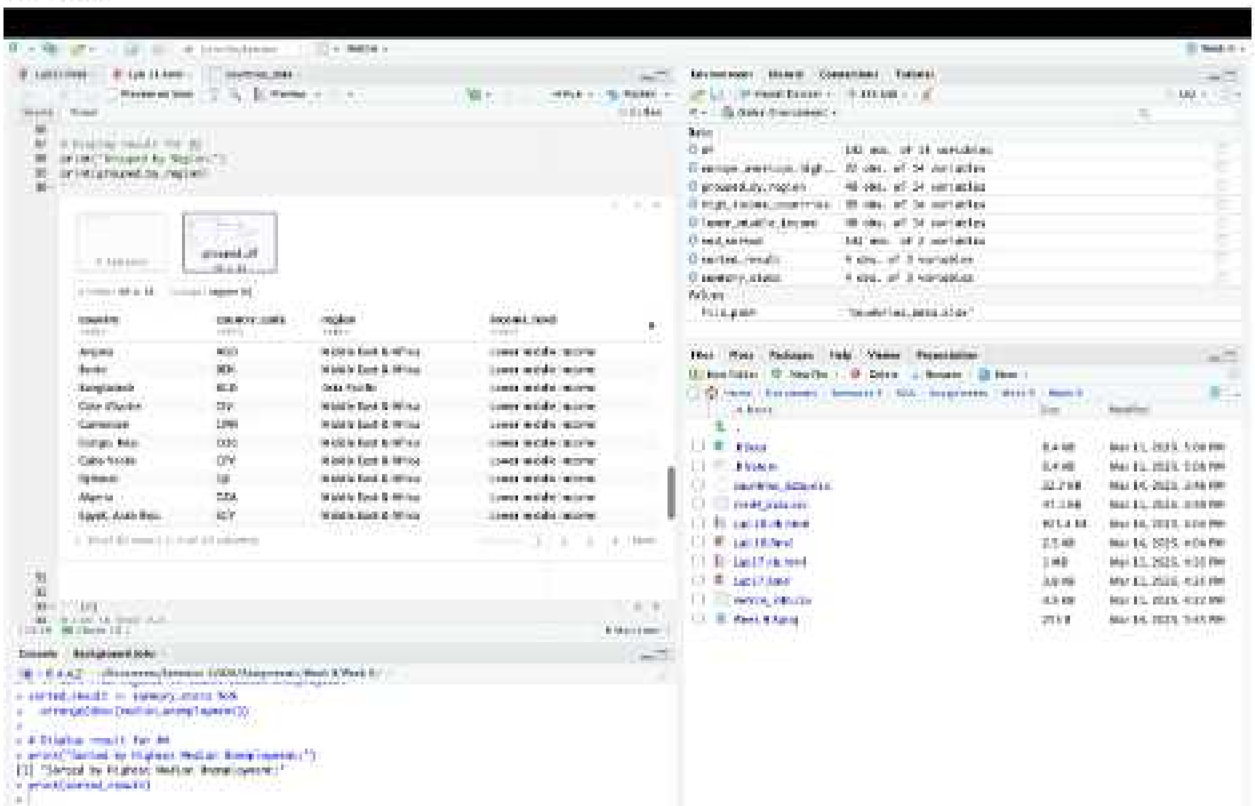
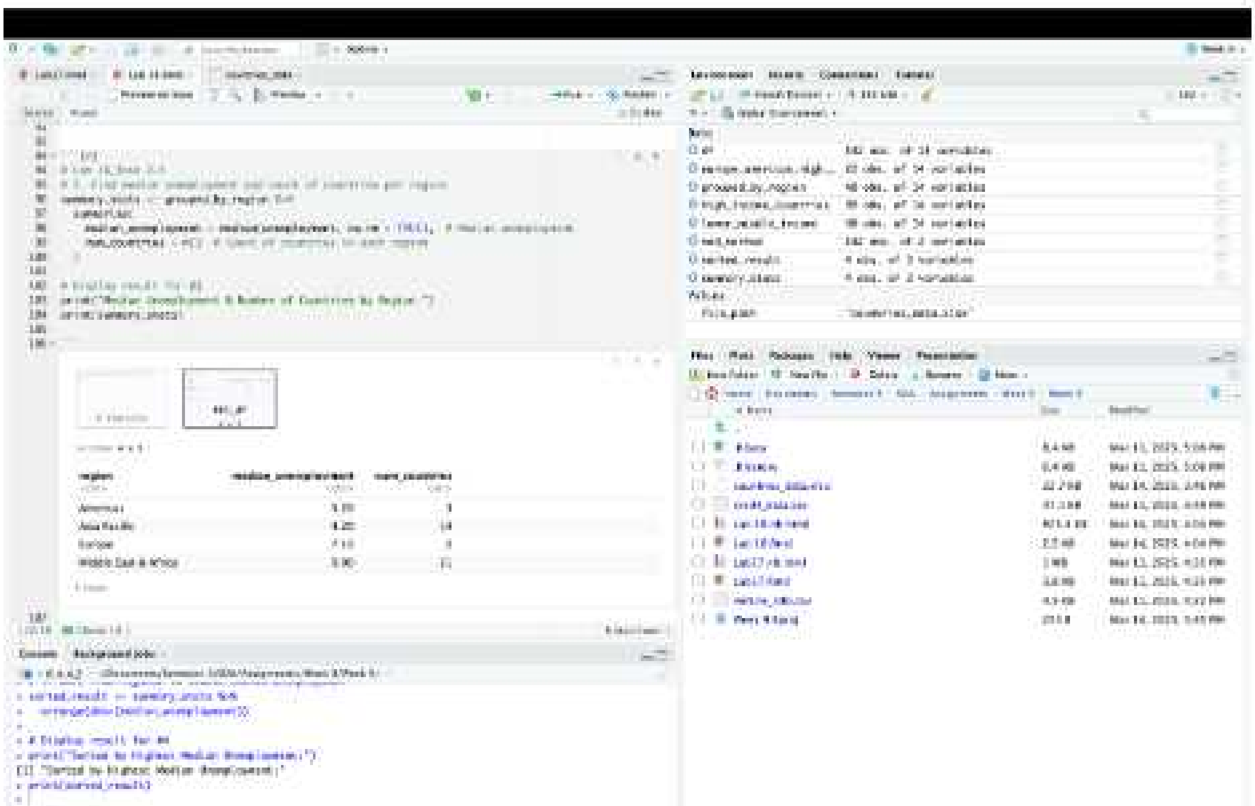
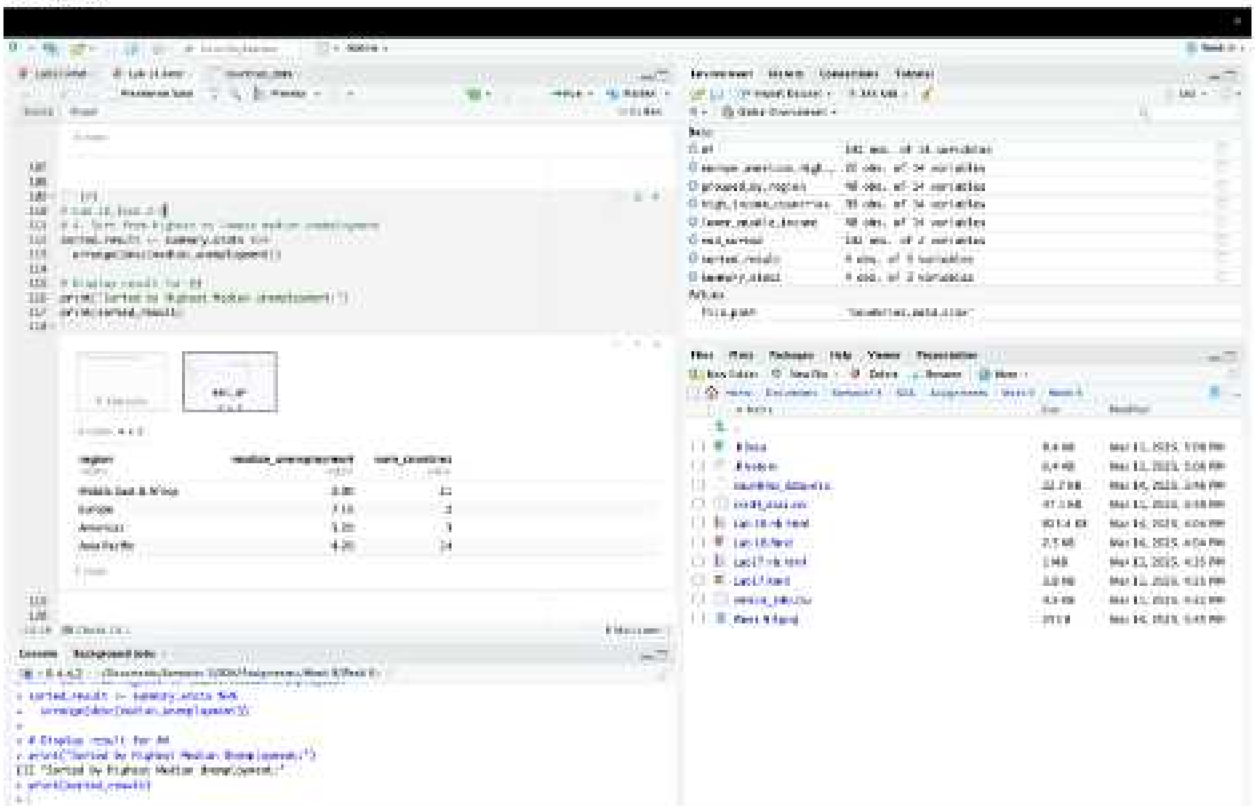
Output:



1. Filter countries where income level is 'High income'.

Code:

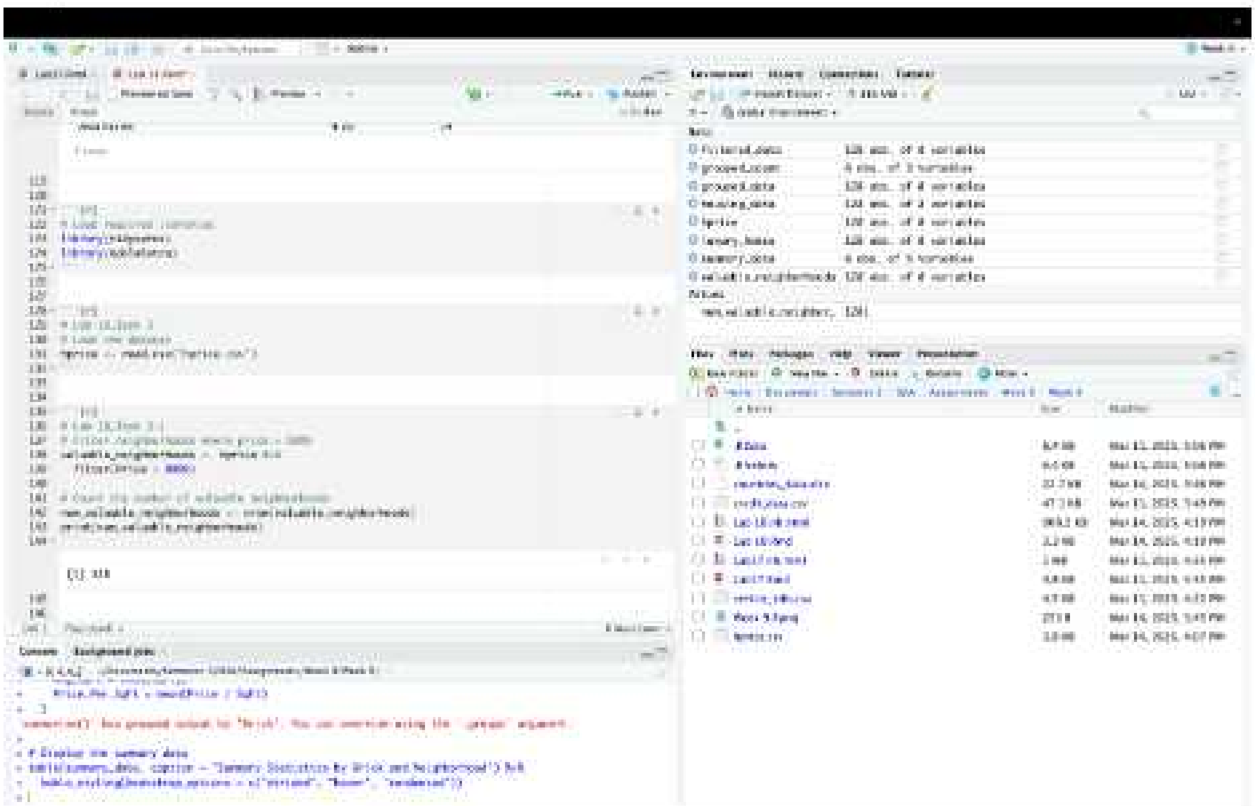
```r
```{r}
# Lab 18_Task 1-1
# 1. Filter countries where income level is 'High income'
high_income_countries <- df %>%
  filter(income_level == "High income")

# Display result for #1
print("High Income Countries:")
print(high_income_countries)
```
```

Output:



2. Filter countries where region is 'Europe' or 'Americas' and life expectancy is greater than 80 years.

Code:

```
```{r}
# Lab 18_Task 1-2
# 2. Filter countries where region is 'Europe' or 'Americas' and life expectancy > 80
europe_americas_high_life_exp <- df %>%
  filter(region %in% c("Europe", "Americas") & life_expect > 80)

# Display result for #2
print("Countries in Europe or Americas with Life Expectancy > 80:")
print(europe_americas_high_life_exp)
```
```

Output:



3. Filter the top 5 countries based on GDP growth.

Code:

```
```{r}
# Lab 18_Task 1-3
# 3. Filter the top 5 countries based on GDP growth
top_5_gdp_growth <- df %>%
  arrange(desc(gdp_growth)) %>%
  head(5)

# Display result for #3
print("Top 5 Countries Based on GDP Growth:")
print(top_5_gdp_growth)
```
```

Output:



4. Create a new variable named NED with a value of 1 when country is equal to Netherlands and 0 when it is not. Select the country name and NED columns side by side and arrange in descending order of NED.

Code:

```
```{r}
# Lab 18_Task 1-4
# 4. Create a new variable NED (1 if country is Netherlands, otherwise 0)
ned_sorted <- df %>%
  mutate(NED = ifelse(country == "Netherlands", 1, 0)) %>%
  select(country, NED) %>%
  arrange(desc(NED))

# Display result for #4
print("Netherlands Flag Variable (NED), Sorted:")
print(ned_sorted)
```
```

Output:



Task 2:

Use the 'countries_data.xlsx' dataset. Write code that does the following:

1. Filter for income level equal to Lower middle income equal to Education

Code:

````


```{r}
# Lab 18_Task 2-1
# 1. Filter for income level equal to 'Lower middle income'
lower_middle_income <- df %>%
  filter(income_level == "Lower middle income")

# Display result for #1
print("Filtered for Lower Middle Income:")
print(lower_middle_income)
```
````

Output:



2. Group by region

Code:

```
```{r}
# Lab 18_Task 2-2
# 2. Group by region
grouped_by_region <- lower_middle_income %>%
  group_by(region)

# Display result for #2
print("Grouped by Region:")
print(grouped_by_region)
```
```

Output:



3. Find (1) the median value for unemployment and (2) the number of countries in each region.

Code:

```r
```{r}
# Lab 18_Task 2-3
# 3. Find median unemployment and count of countries per region
summary_stats <- grouped_by_region %>%
  summarise(
    median_unemployment = median(unemployment, na.rm = TRUE),  # Median unemployment
    num_countries = n()  # Count of countries in each region
  )

# Display result for #3
print("Median Unemployment & Number of Countries by Region:")
print(summary_stats)

```
```

Output:



4. Sort the results from the highest median optimal level downward.

Code:

~~~~

```{r}
# Lab 18_Task 2-4
# 4. Sort from highest to lowest median unemployment
sorted_result <- summary_stats %>%
  arrange(desc(median_unemployment))

# Display result for #4
print("Sorted by Highest Median Unemployment:")
print(sorted_result)
```
~~~~

Output:



Task 3:

Suppose we work for a housing developer like Toll Brothers (NYSE: TOL) and want to allocate resources to marketing and financing the building of luxury homes in major US metropolitan areas. We have data for a test market in the file 'hprice.csv'.

Do the following:

Code:

````
```{r}
# Load required libraries
library(tidyverse)
library(kableExtra)
```


```{r}
# Lab 18_Task 3
# Load the dataset
hprice <- read.csv("hprice.csv")
```
````

1. Create a Pivot table to filter the most valuable (higher price) neighborhoods? (Greater than 9999)

Code:

````
```{r}
# Lab 18_Task 3-1
# Filter neighborhoods where price > 9999
valuable_neighborhoods <- hprice %>%
  filter(Price > 9999)

# Count the number of valuable neighborhoods
num_valuable_neighborhoods <- nrow(valuable_neighborhoods)
print(num_valuable_neighborhoods)
```
````

Output:

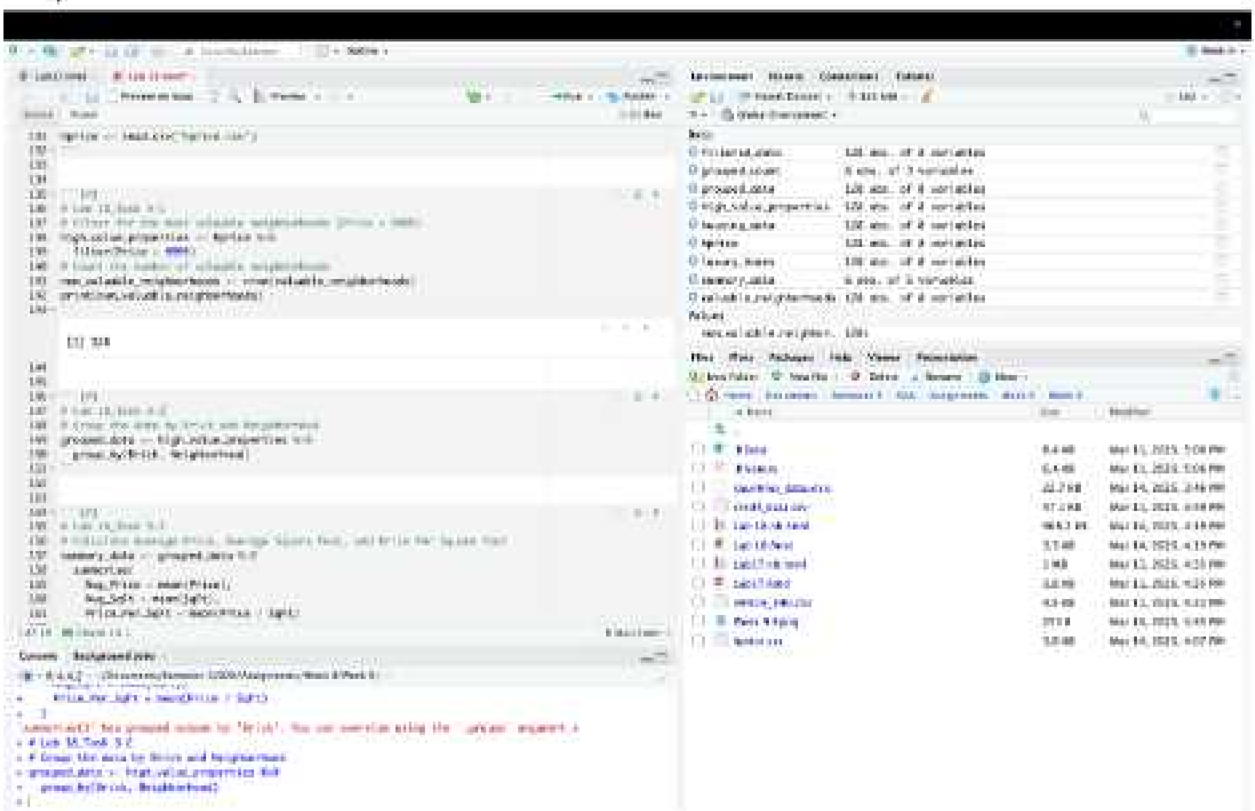


2. Group the Pivot table data by Brick and Neighborhood.

Code:

```
```{r}
# Lab 18_Task 3-2
# Group the data by Brick and Neighborhood
grouped_data <- high_value_properties %>%
  group_by(Brick, Neighborhood)
```
```

Output:



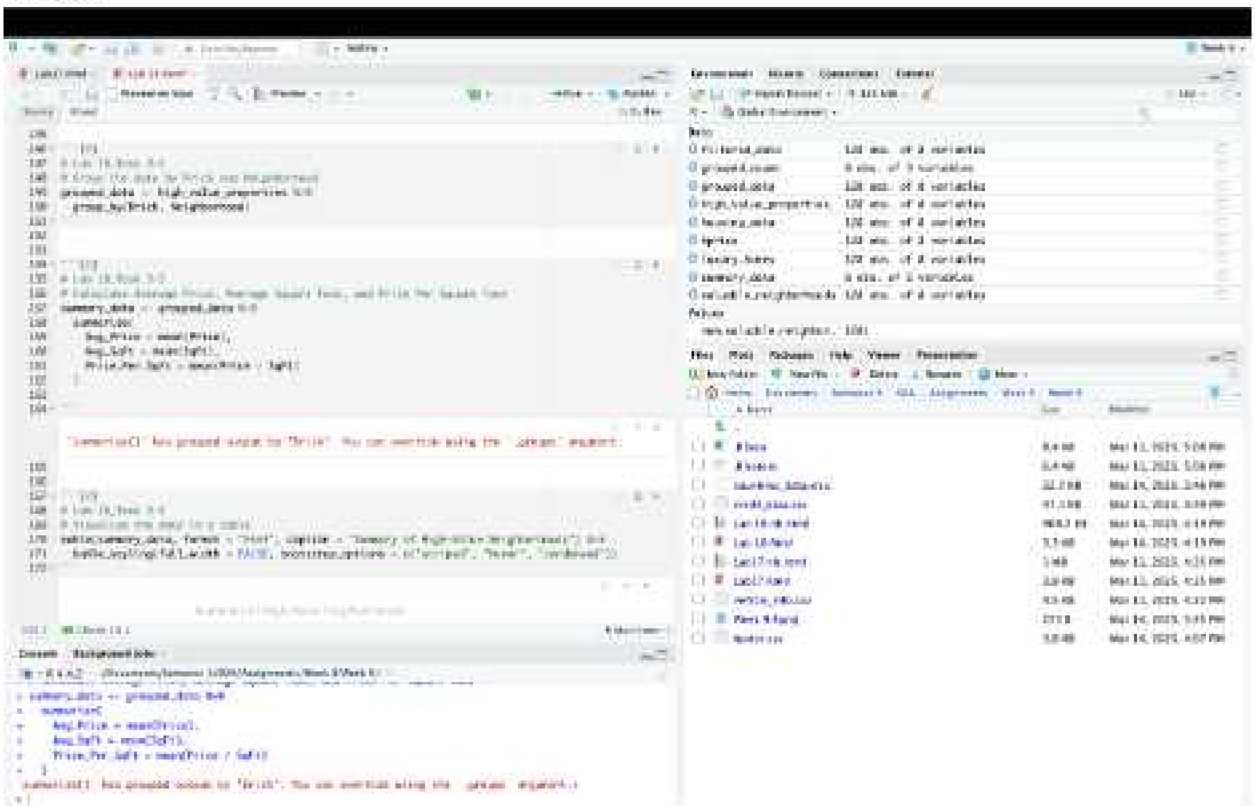
3. Fetch the Average Price, Average Square Foot, and Price Per Square Foot

Code:

```r
```{r}
# Lab 18_Task 3-3
# Calculate Average Price, Average Square Foot, and Price Per Square Foot
summary_data <- grouped_data %>%
  summarise(
    Avg_Price = mean(Price),
    Avg_SqFt = mean(SqFt),
    Price_Per_SqFt = mean(Price / SqFt)
  )

```
```

Output:

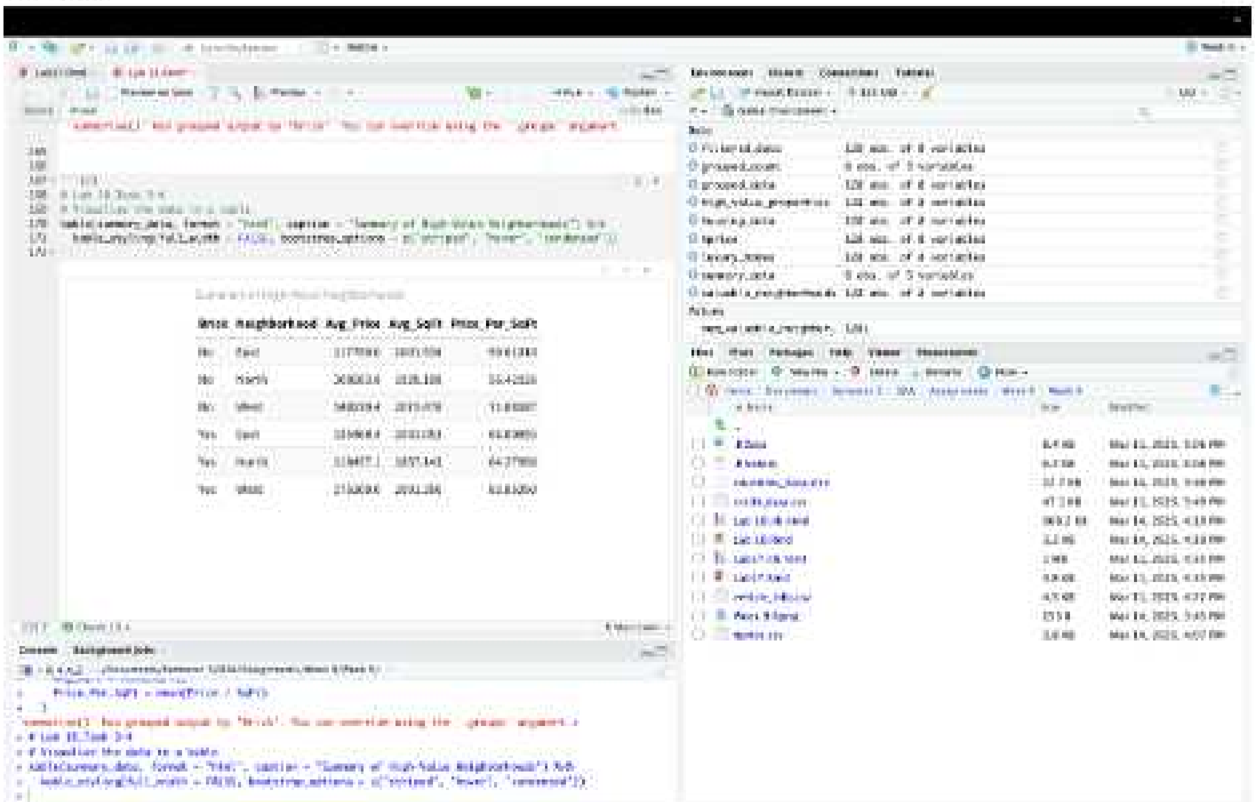


4. Visualize 1-3 in a table (Use the kableExtra package and knitr:: command)

Code:

```
```{r}
# Lab 18_Task 3-4
# Visulize the data in a table
kable(summary_data, format = "html", caption = "Summary of High-Value Neighborhoods") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed"))
```
```

Output:



Task 4:

Gen AI Prompt Engineering:

You work as a data analyst for a retail company. The marketing department has provided you with customer transaction data in an Excel file, customer_sales.csv, and wants you to perform customer segmentation analysis based on how much customers spend and calculate potential impact of targeted marketing campaigns. Use spending-based customer classification system that segments customers into three categories:

"High Spender" - Customers who spent $300 or more

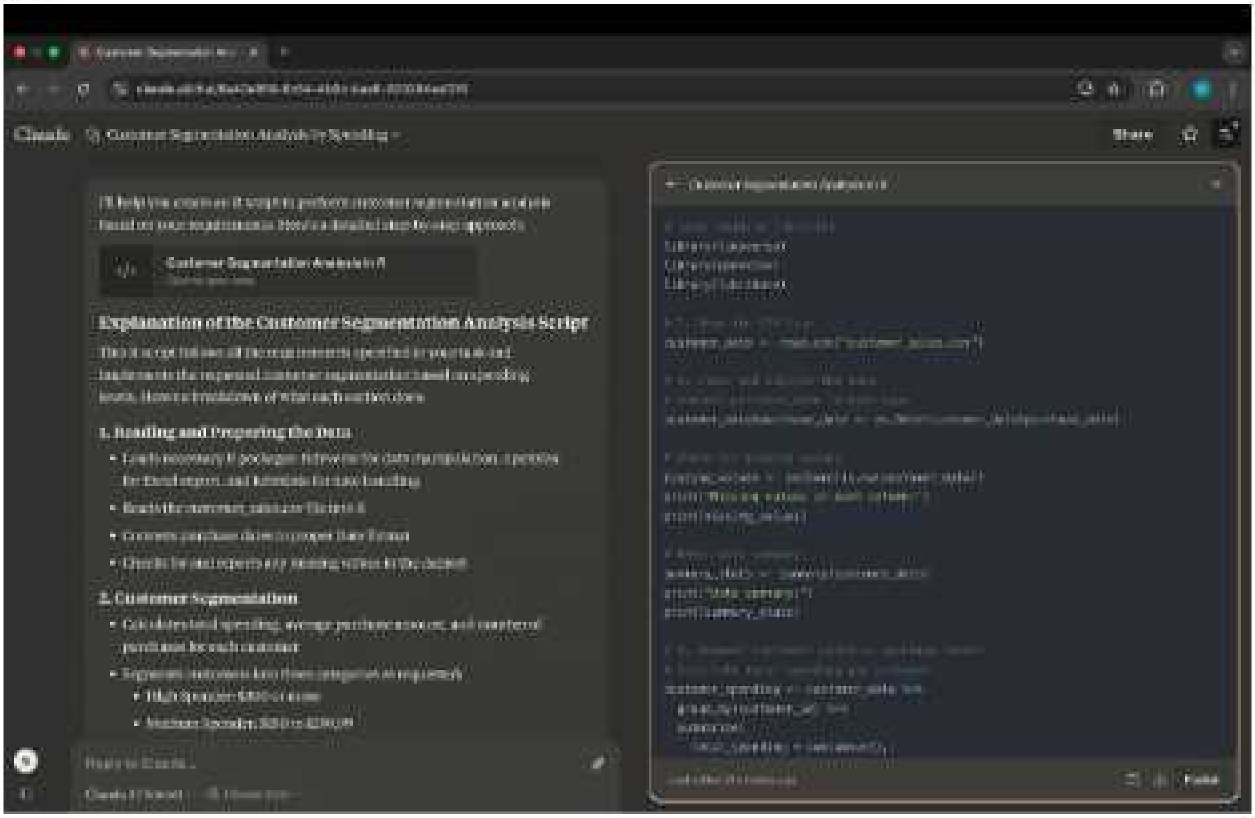
"Medium Spender" - Customers who spent between $150 and $299.99

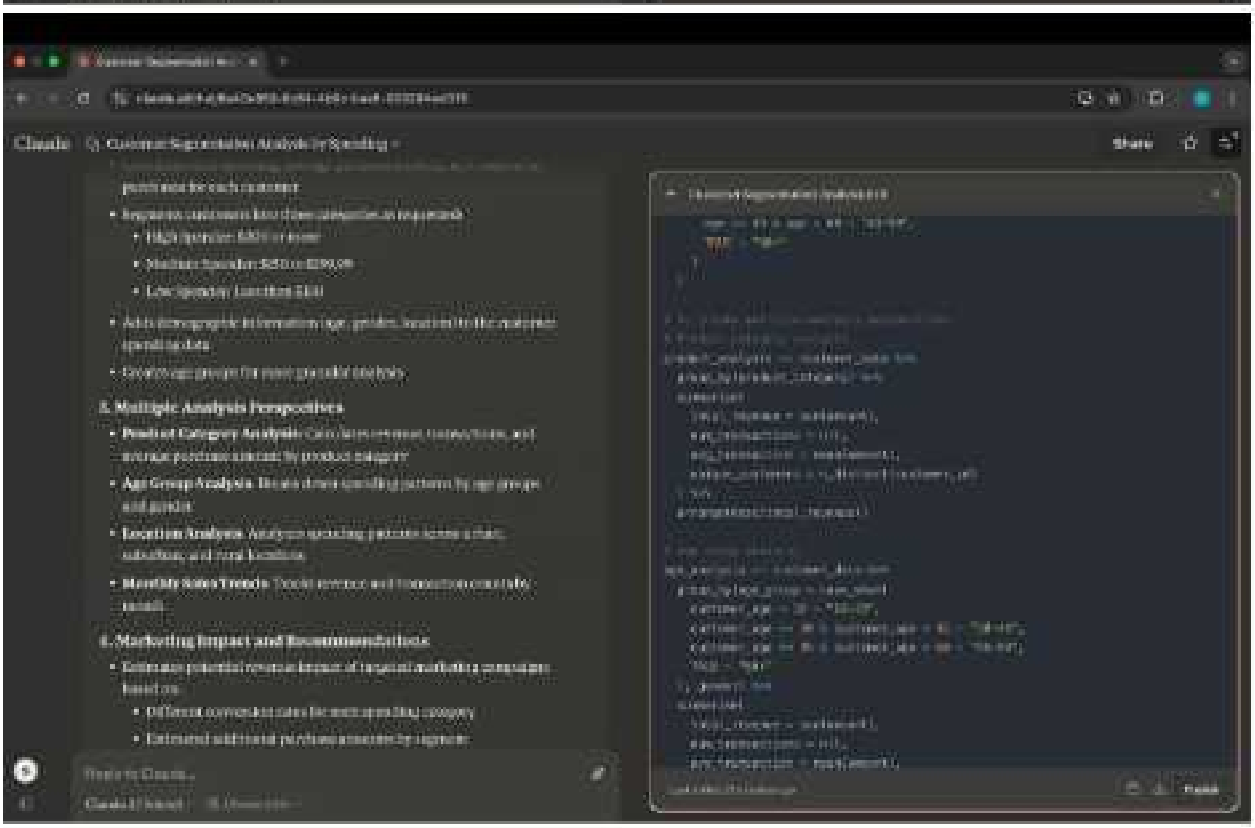
"Low Spender" - Customers who spent less than $150

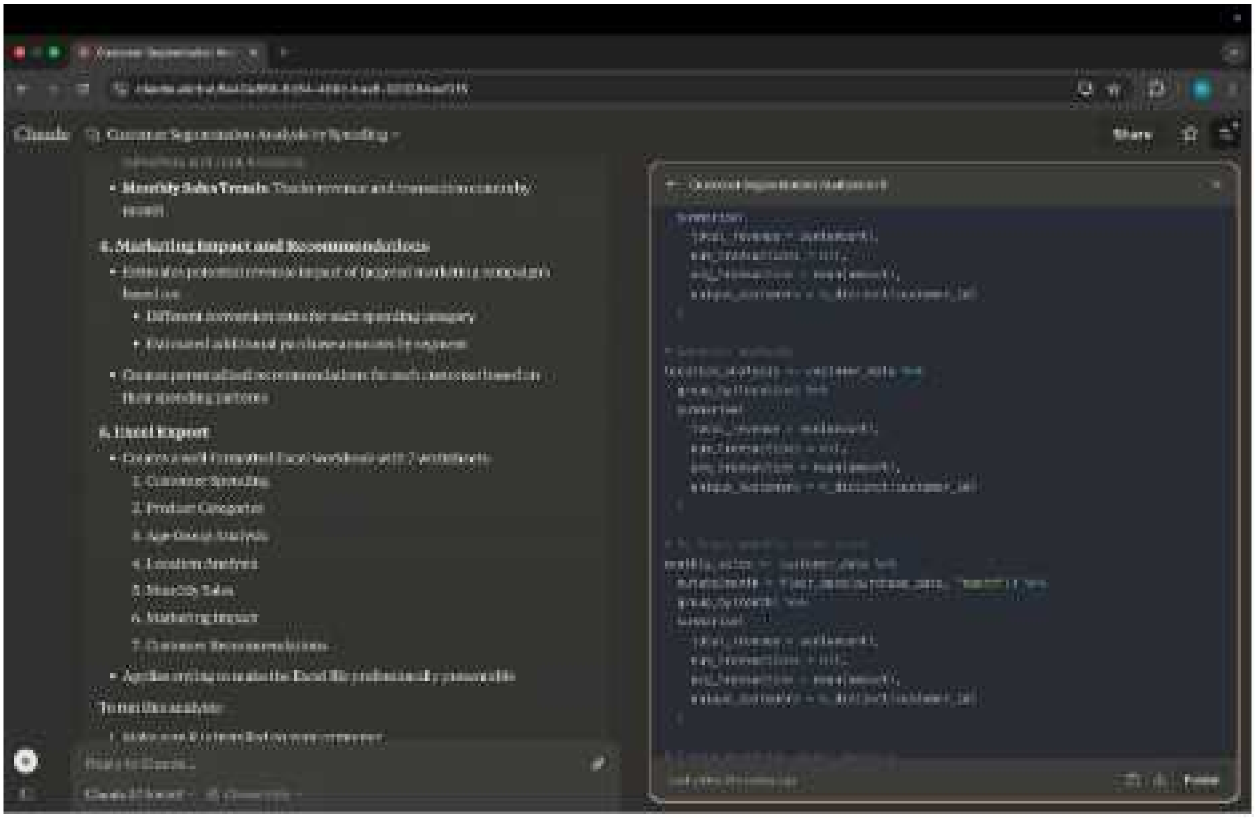
Bot Used: Claude.AI

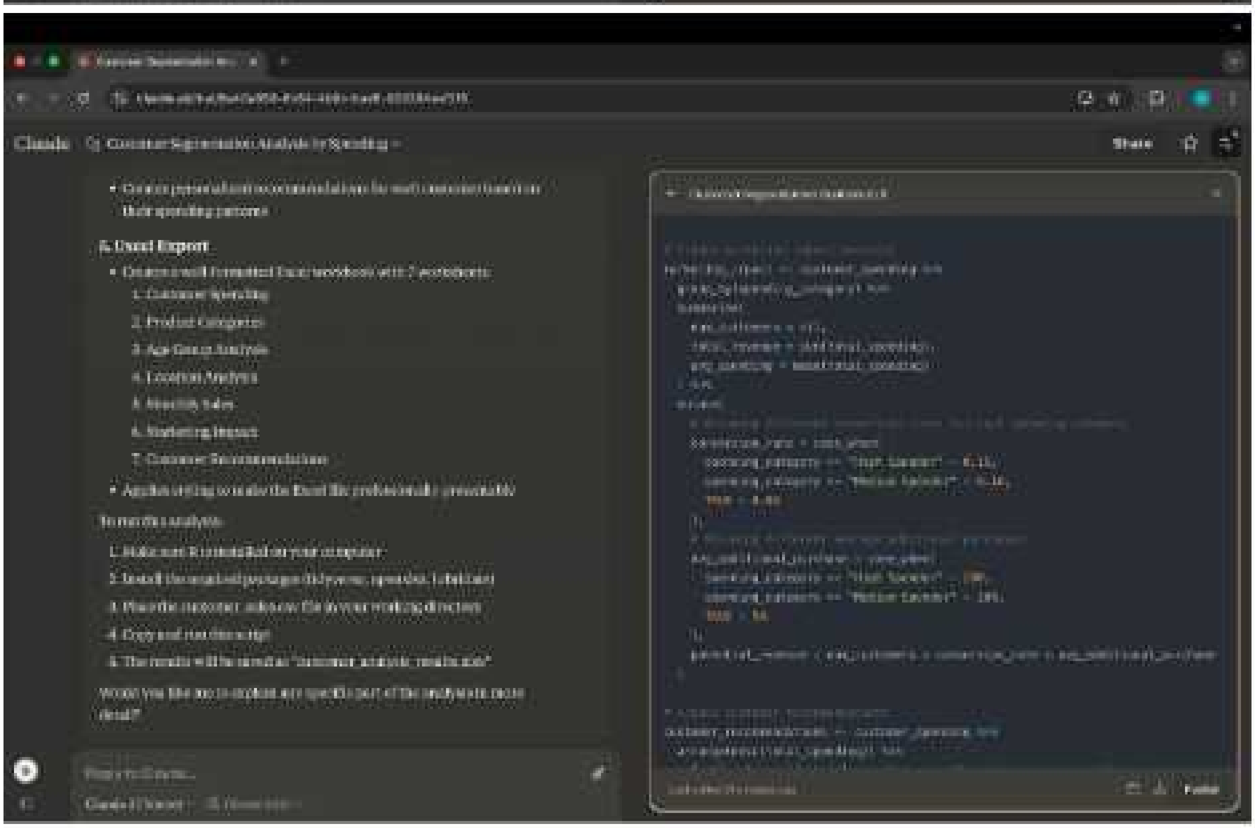
Prompt:

Bot:

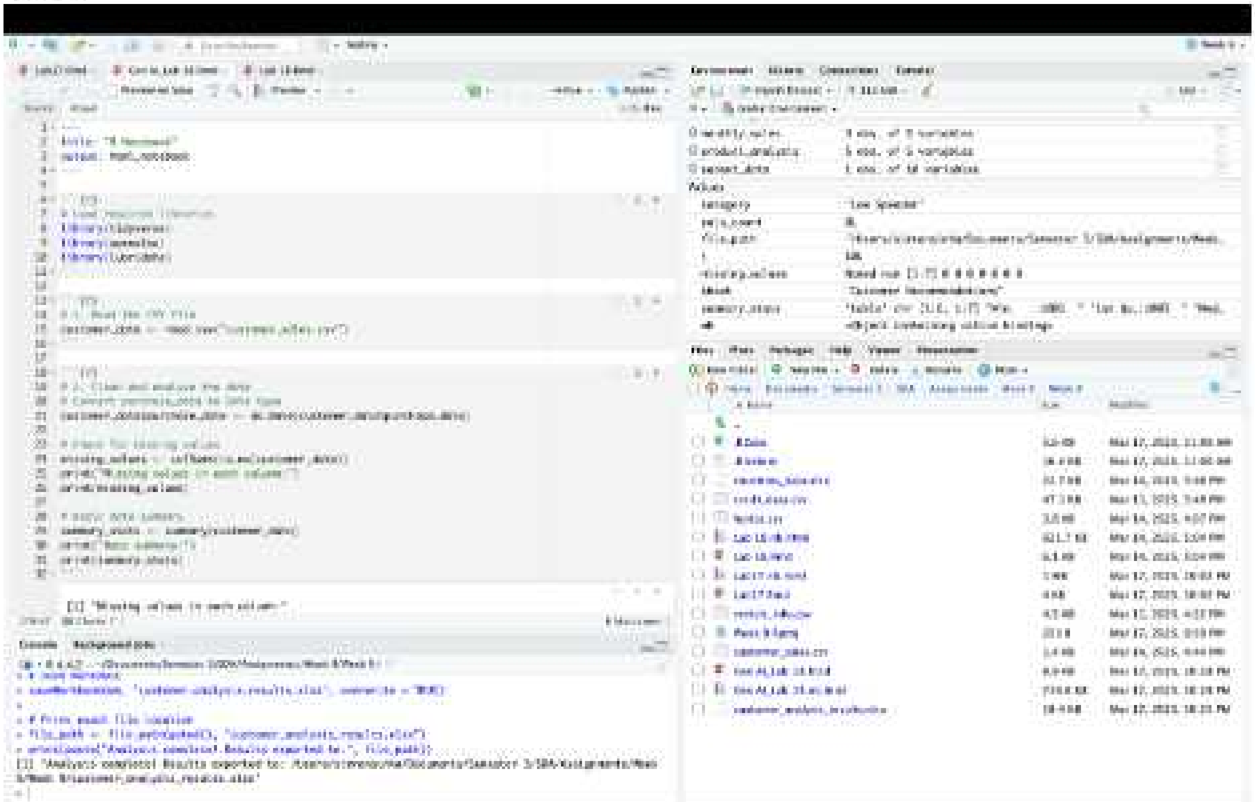

Customer Segmentation Analysis by Spending
Explanation of the Customer Segmentation Analysis Script
1. Reading and Preparing the Data
2. Customer Segmentation

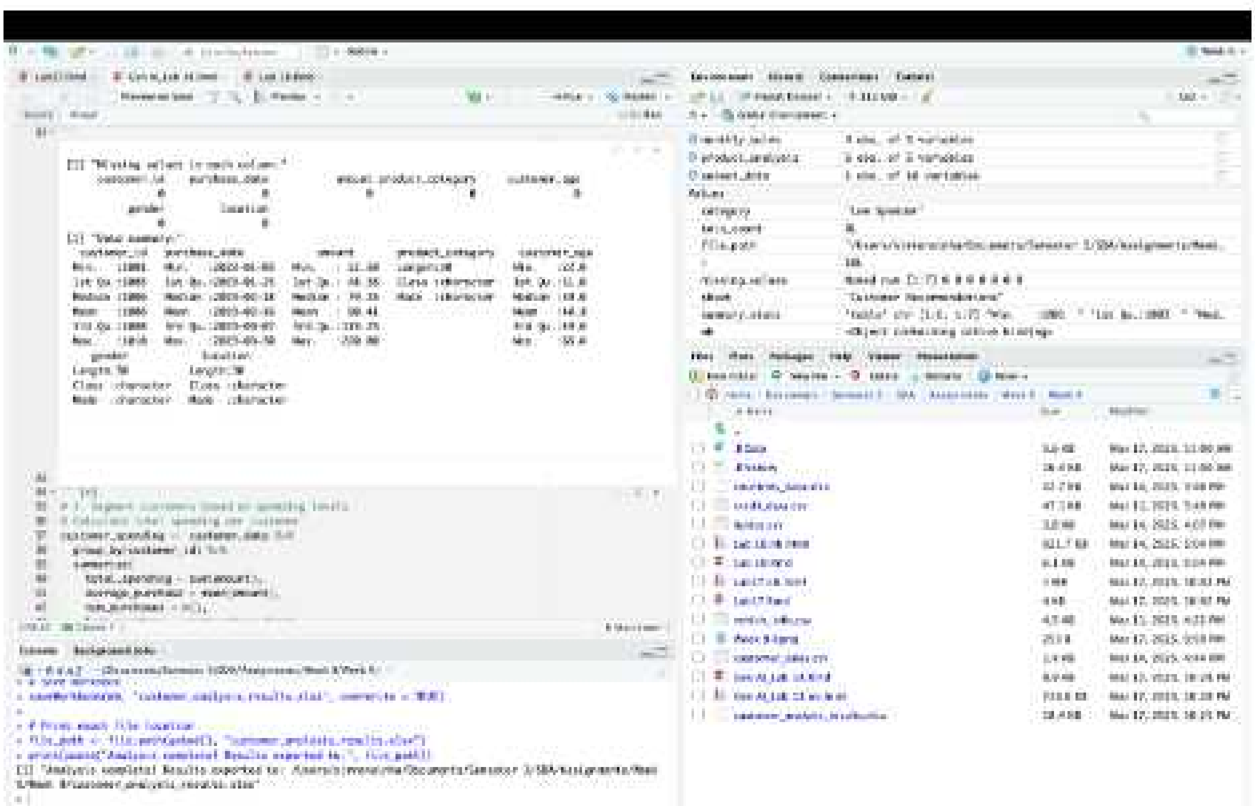

Customer Segmentation Analysis by Spending
3. Multiple Analysis Perspectives
Product Category Analysis
Age Group Analysis
Location Analysis
Monthly Sales Trends
4. Marketing Impact and Recommendations

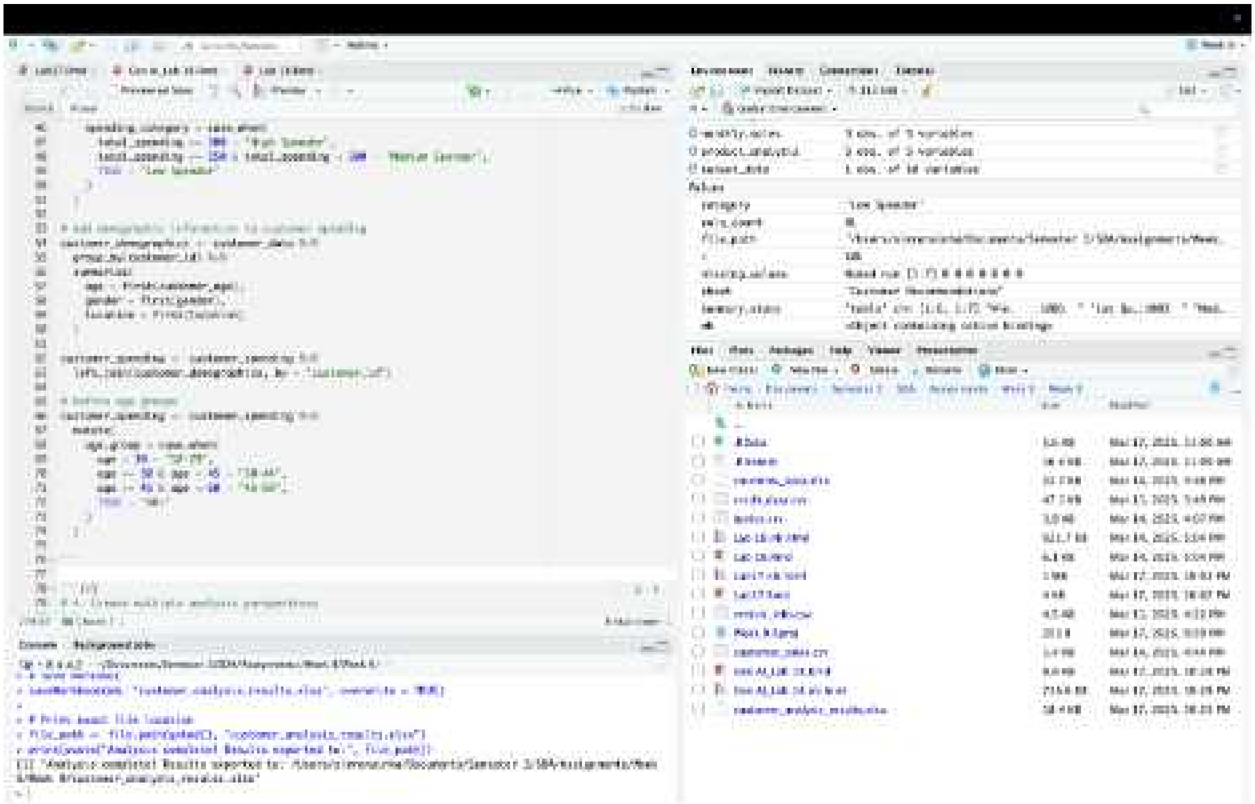

Claude
Share
4. Marketing Impact and Recommendations
5. Excel Export
1. Customer Spending
2. Product Categories
3. Age Group Analysis
4. Location Analysis
5. Monthly Sales
6. Marketing Impact
7. Customer Recommendations

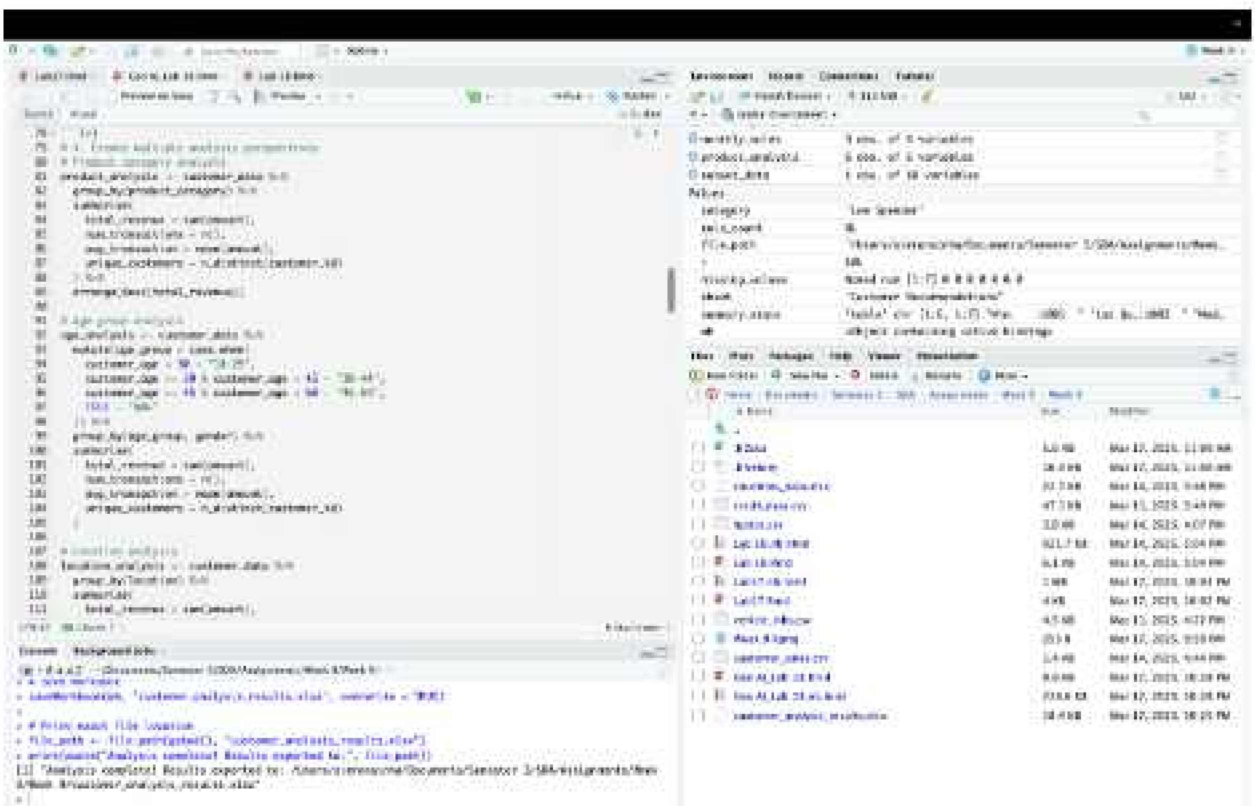

Claude
Share
5. Excel Export
1. Customer Spending
2. Product Categories
3. Age Group Analysis
4. Location Analysis
5. Monthly Sales
6. Marketing Impact
7. Customer Recommendations

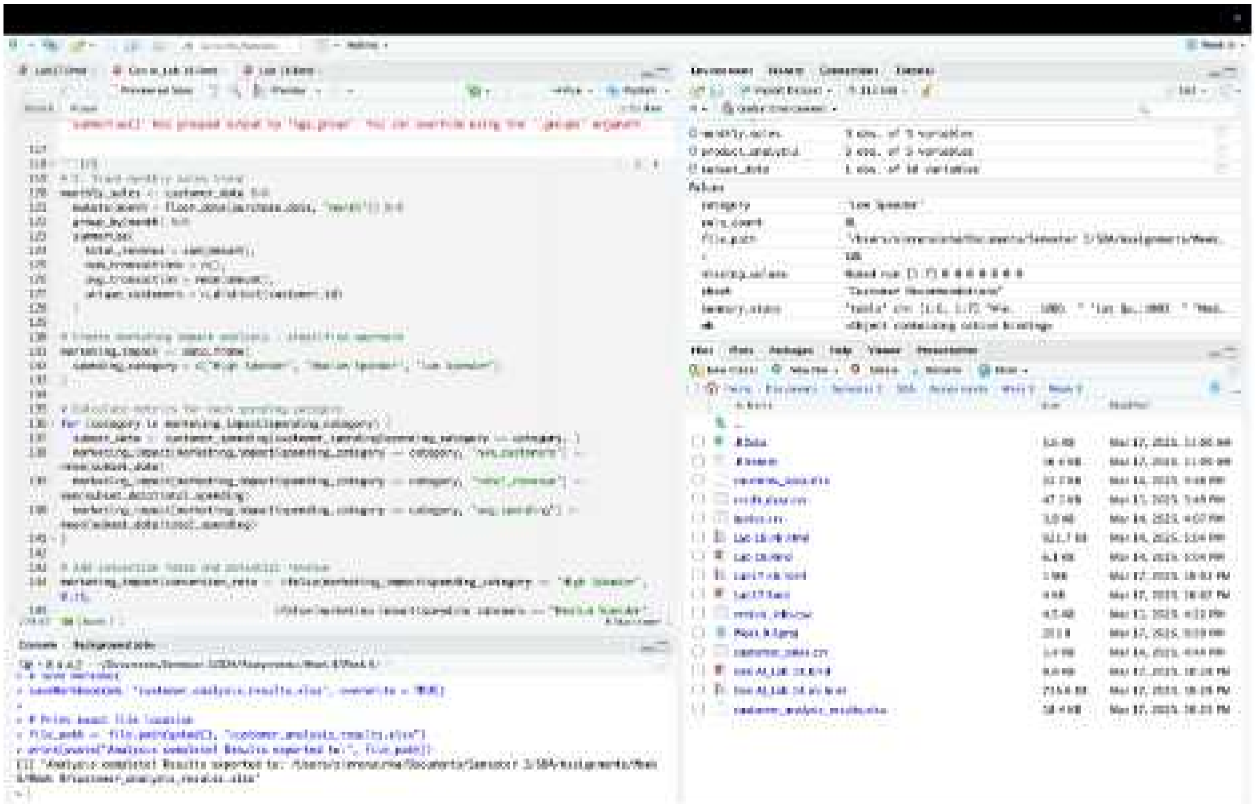
Code:

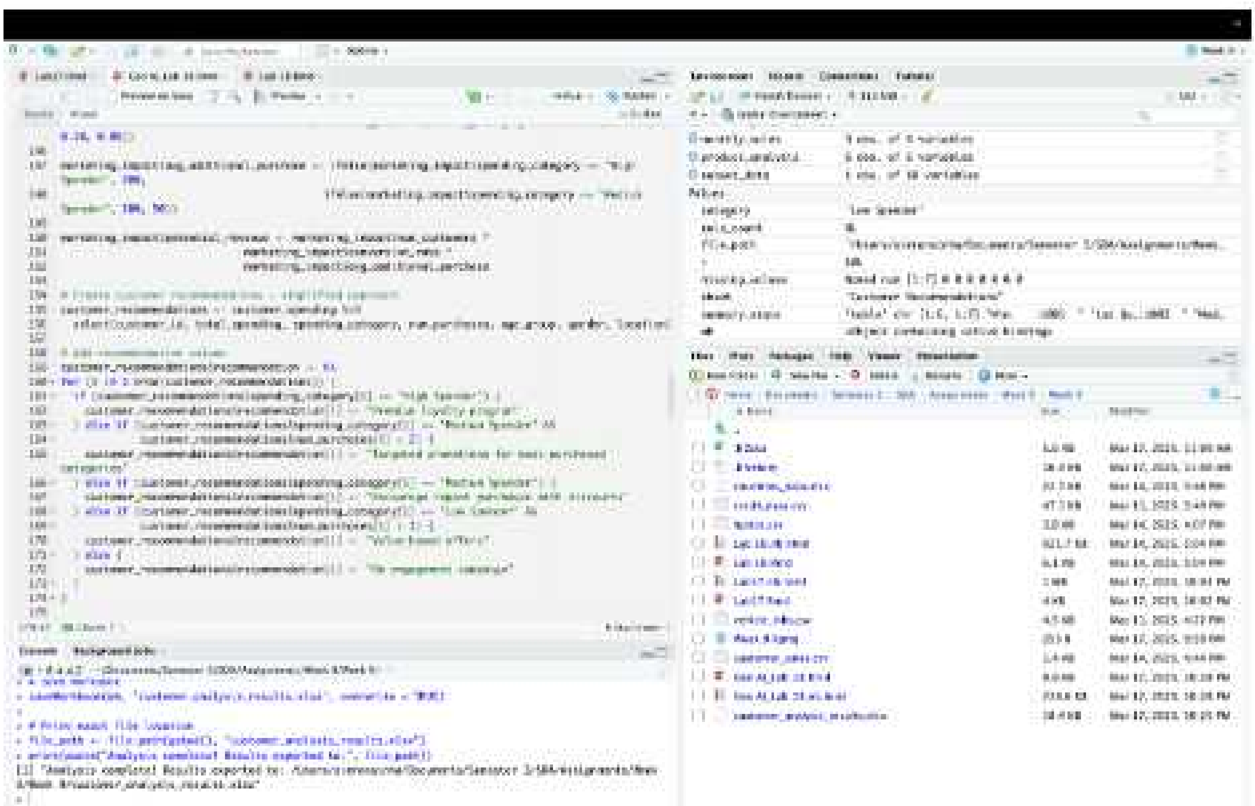


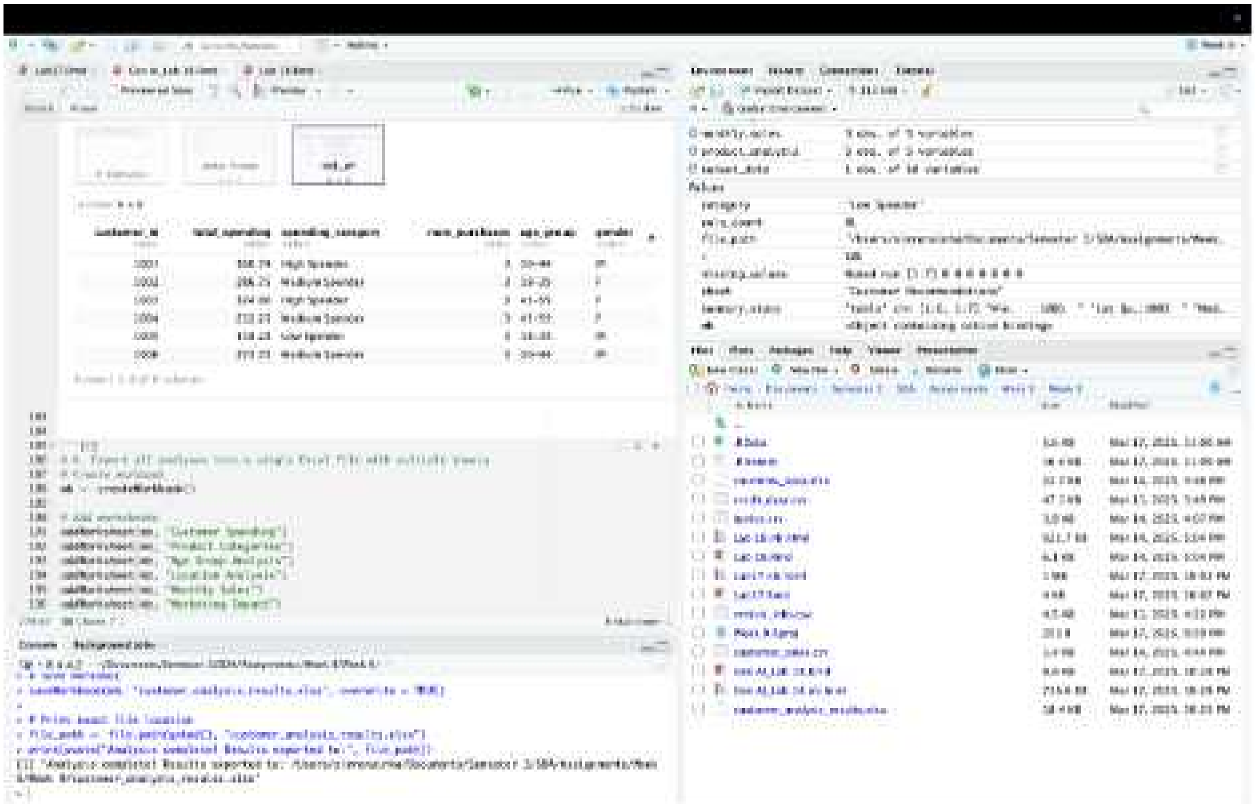


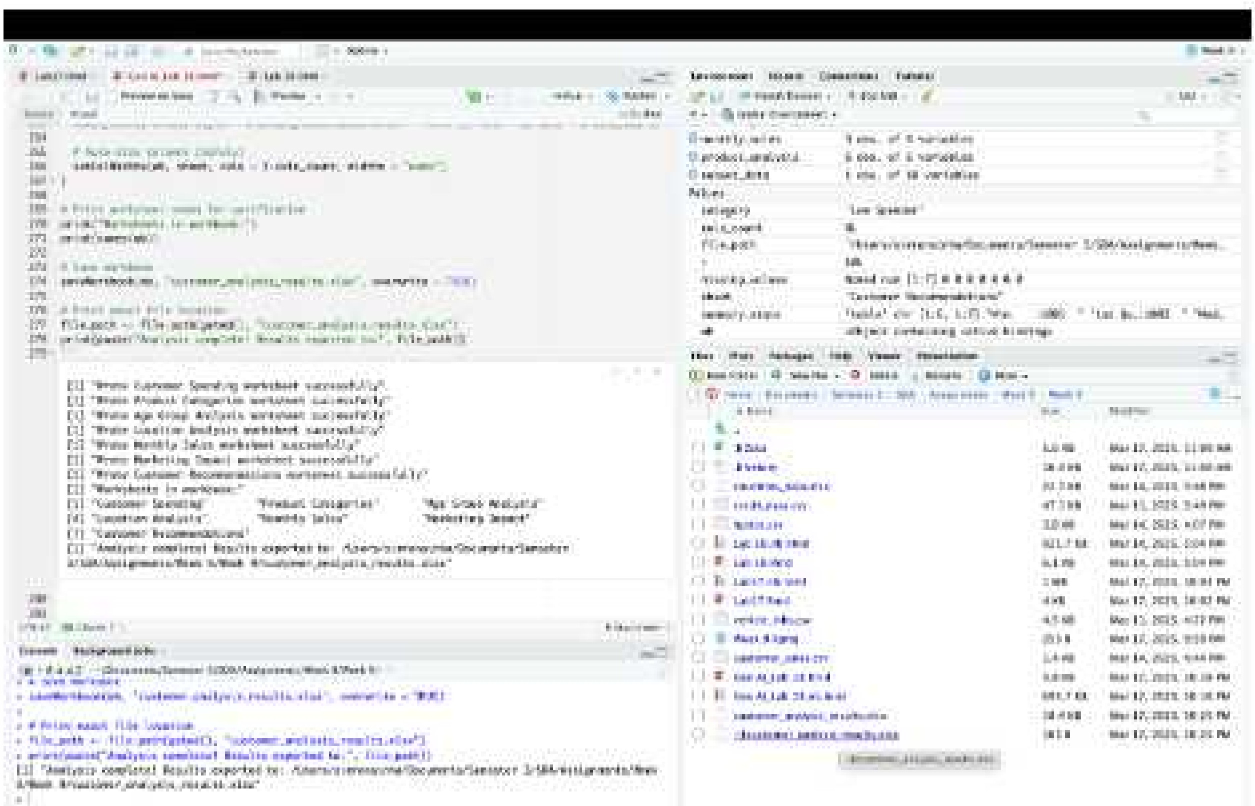




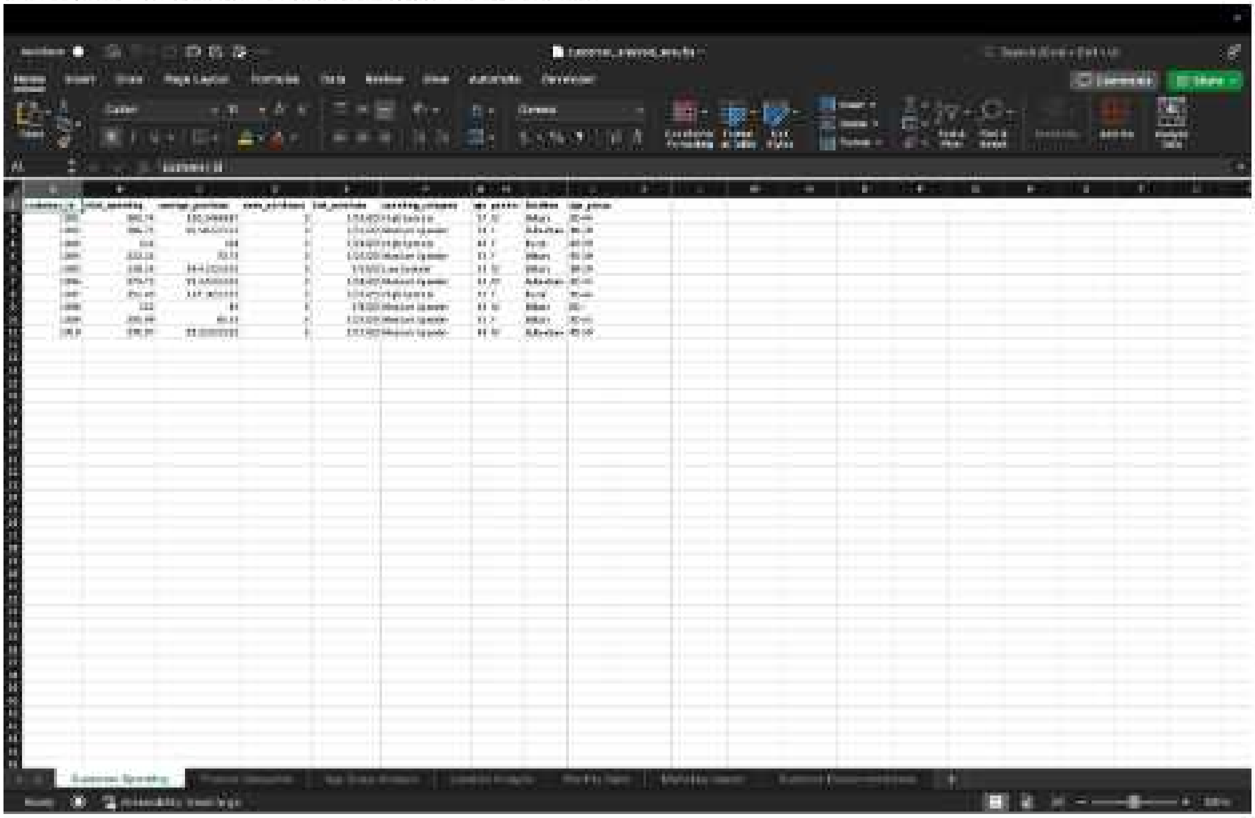


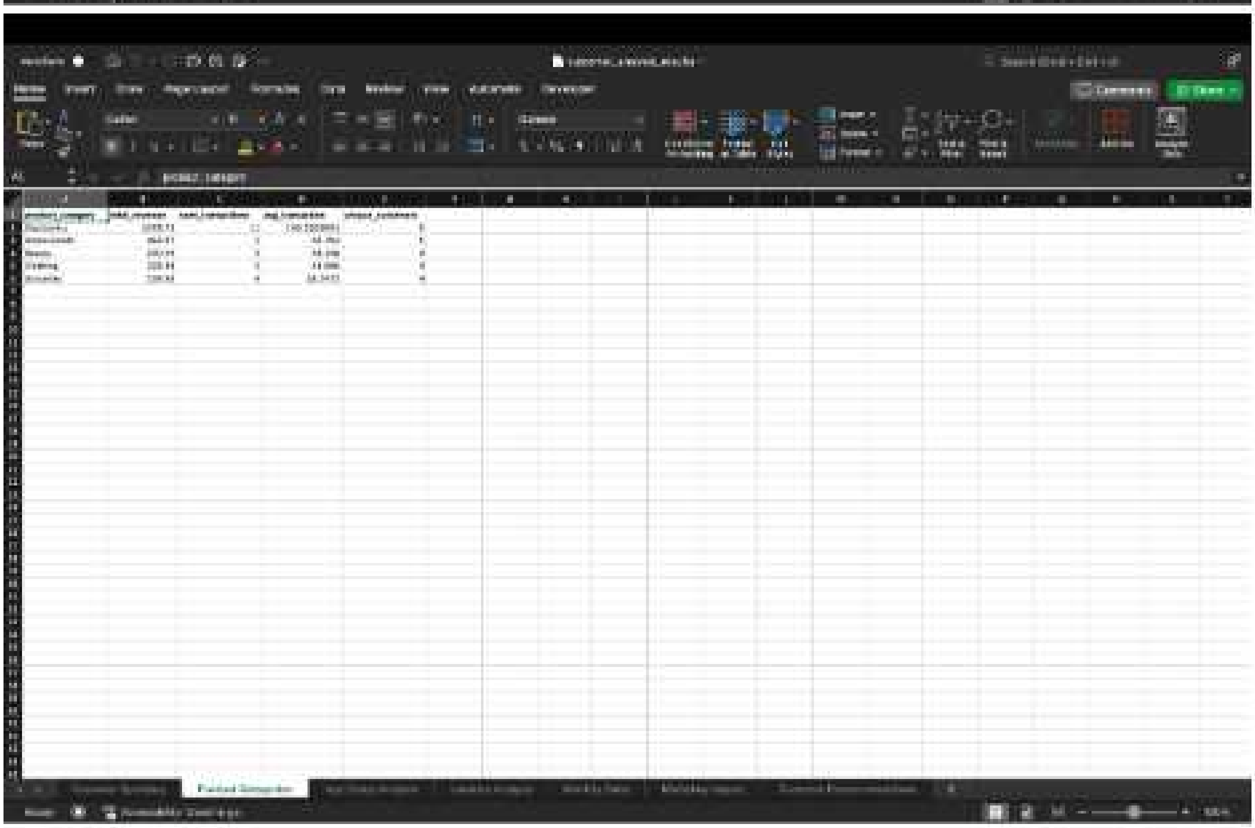


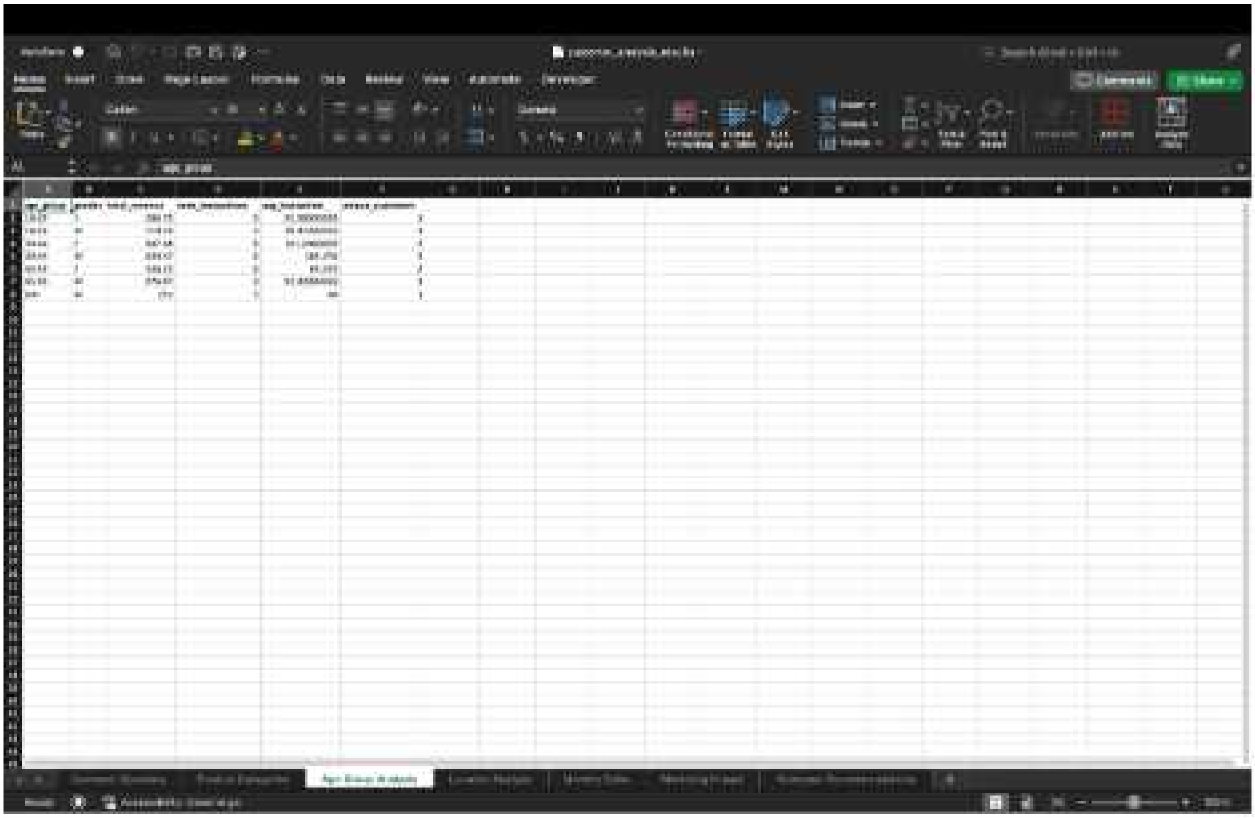


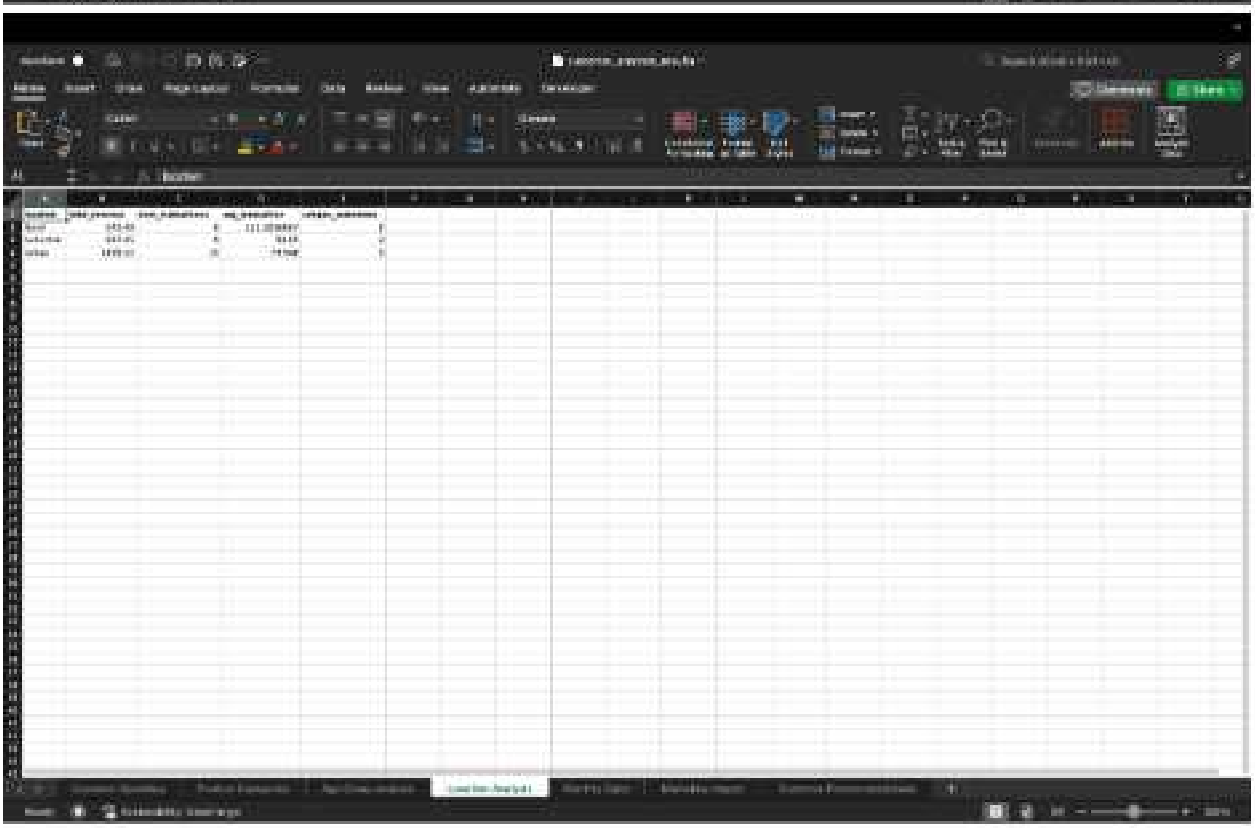


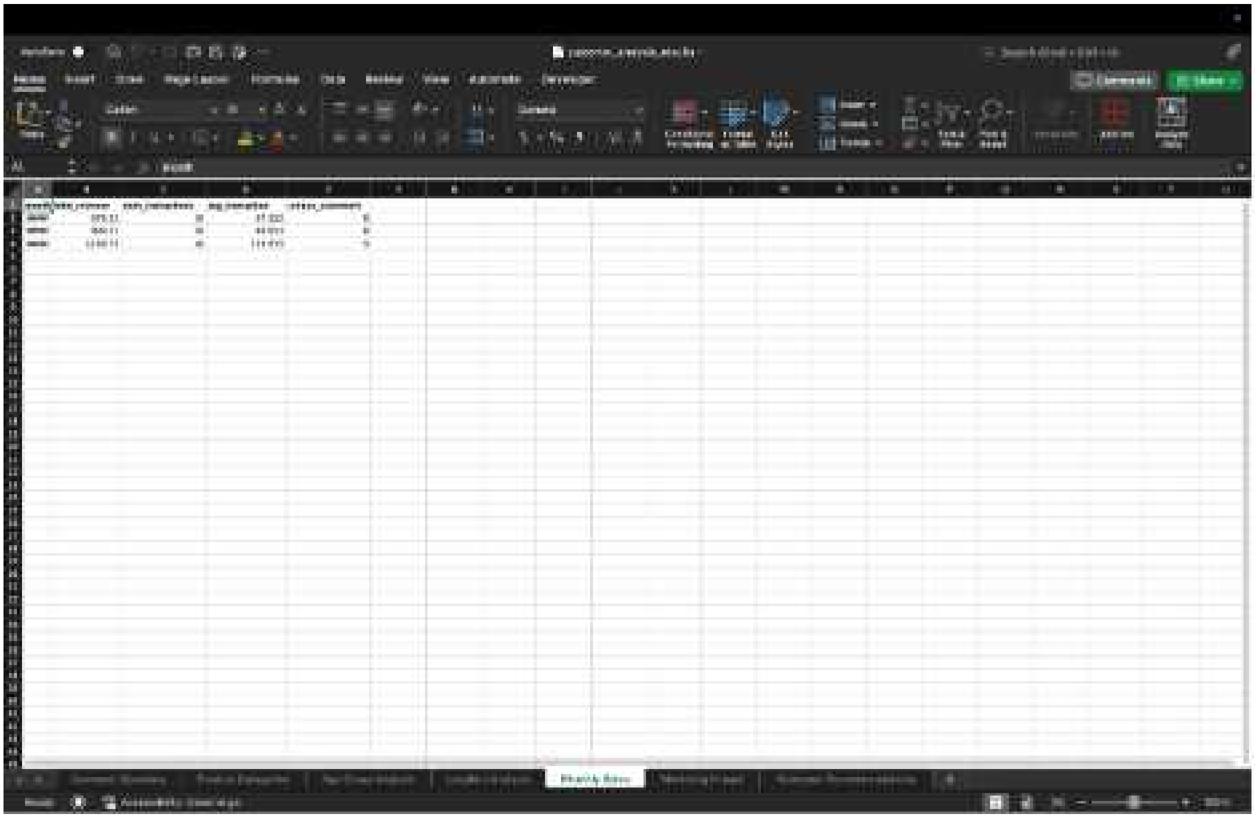
This the excel we got after running R script.

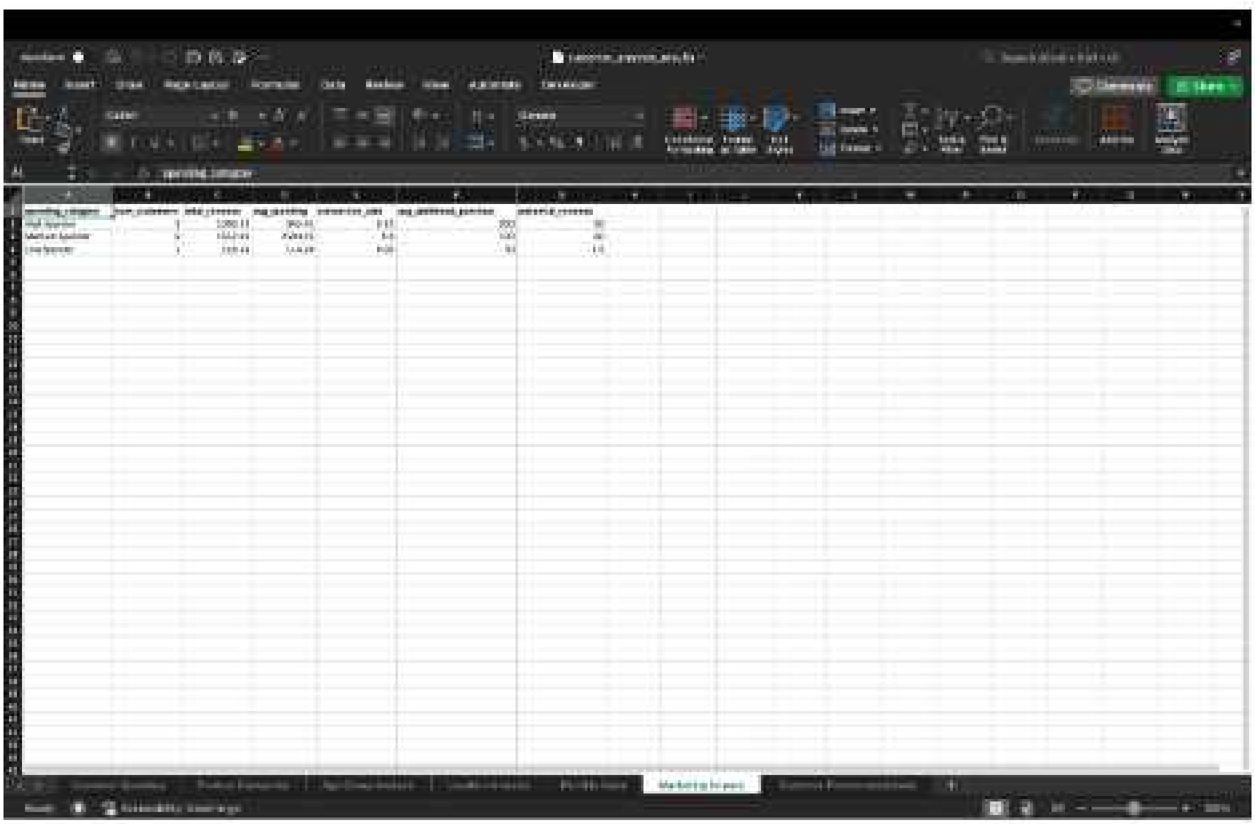


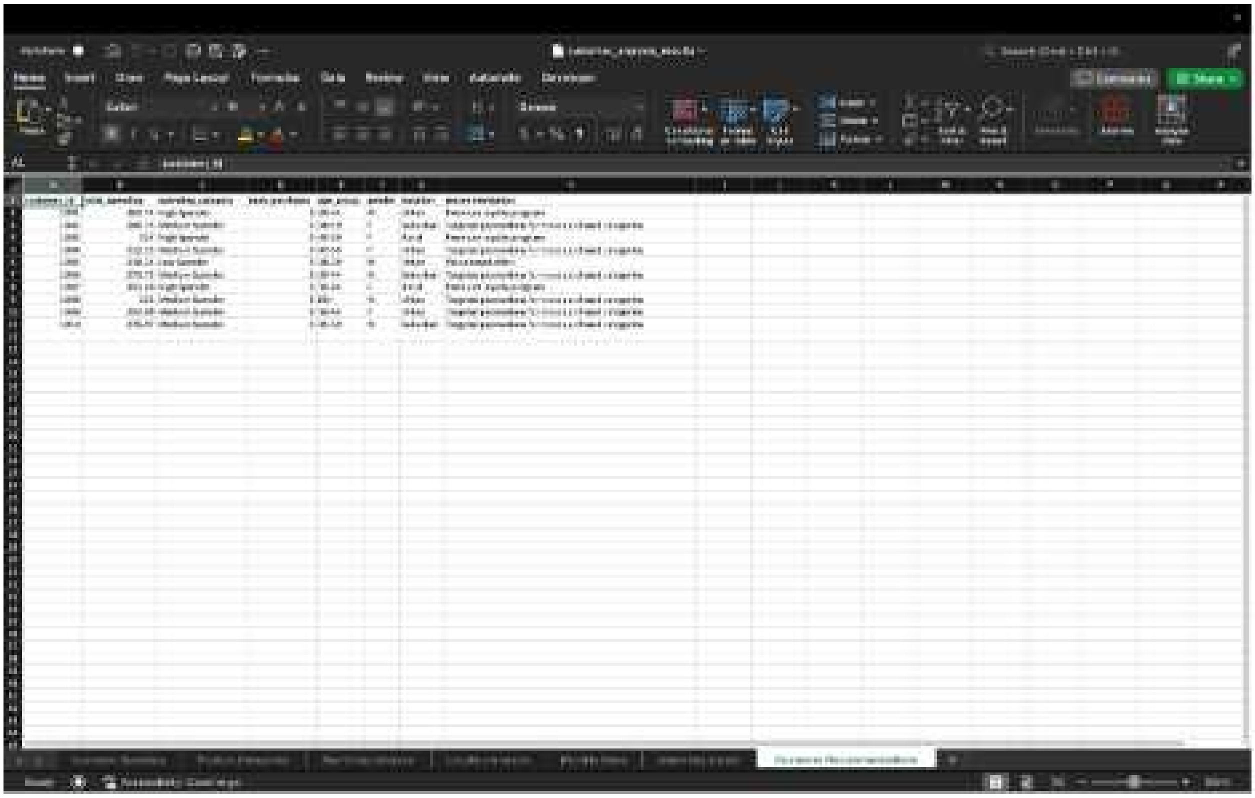












- As we can see the bot's query resulted in a clear response and we have also achieved the required result.