RATHINAM COLLEGE OF ARTS & SCIENCE (AUTONOMOUS)

Coimbatore-641021

DEPARTMENT OF COMPUTER SCIENCE

Core Practical – MACHINE LEARNING Lab Manual

Lab Manual for the Academic Year 2023-24

(in accordance with Computer Science syllabus)

SUBJECT : Machine Learning Lab

STREAM : B.Sc. [CS/CT/IT/AIML]/BCA

Staff Incharge H.O.D

MACHINE LEARNING LAB MANUAL

1. Statistical Analysis of Ozone Level and Weather Factors-File name-Lab1.R
2. Multiple Linear Regression and Diagnostic of Environment Data-File name-Lab2.R
3. Linear Regression analysis of Ozone Level-File name-Lab3.R
4. Linear Regression Analysis of Study, Sleep Hours and Exam scores-File name-linear.R
5. Linear Regression and Model Evaluation with Salary Data-File name- income.R
6. Alcohol Consumption Analysis using Scatter Plot- File name- al.R
7. Alcohol Consumption Analysis using Bar-Graph-File name- al2.R

1. Statistical Analysis of Ozone Levels and Weather Factors

Dataset for 1st 3 programs (Lab1, Lab2, and Lab3):

S. No.	Ozone	Solar R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	27	192	14.3	56	5	5
6	28	193	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	24	194	8.6	69	5	10
11	7	152	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15
16	14	334	11.5	64	5	16
17	34	307	12	66	5	17
18	6	78	18.4	57	5	18
19	30	322	11.5	68	5	19
20	11	44	9.7	62	5	20

Aim:

To understand how weather conditions influence ozone levels through statistical analysis.

Algorithm:

Step 1: Install and Load Required Packages

- Install and load the necessary R packages for data manipulation, visualization, and statistical analysis.

Step 2: Create the Data Frame

- Create a data frame called "Data_Frame" containing data from the given table.

Step 3: Summarize the Data

- Use the `summary` function to summarize the "Data_Frame."

Step 4: Linear Regression

- Fit a linear regression model, taking "Ozone" as the dependent variable and using multiple independent variables from the dataset.

Step 5: Predict Ozone Level for the 21st Day

- Create new data for the 21st day's factors and predict the ozone level using the fitted model.

Step 6: Autocorrelation Analysis

- Find the autocorrelation of the error produced from the fitted line.

Step 7: Multicollinearity Analysis

- Analyze multicollinearity among independent variables and identify suitable solutions to remove multicollinearity.

Step 8: Equal Variance Check

- Find the variance among error terms and comment on the equal variance among error terms in the output.

Step 9: Autocorrelation Test

- Estimate the presence of autocorrelation using the Durbin–Watson test statistic.

Code:

```
# Install and load the required packages

install.packages("dplyr")

install.packages("ggplot2")

install.packages("car")# Load required libraries

library(dplyr)

library(ggplot2)

library(car)# Companion to applied regression

# Create the data frame for the given table

Data_Frame <- data.frame(

  S.No. = 1:20,

  Ozone = c(41, 36, 12, 18, 27, 28, 23, 19, 8, 24, 7, 16, 11, 14, 18, 14, 34, 6, 30, 11),

  Solar_R = c(190, 118, 149, 313, 192, 193, 299, 99, 19, 194, 152, 256, 290, 274, 65, 334,
307, 78, 322, 44),

  Wind = c(7.4, 8, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6, 6.9, 9.7, 9.2, 10.9, 13.2, 11.5, 12,
18.4, 11.5, 9.7),

  Temp = c(67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68, 58, 64, 66, 57, 68, 62),
```

```
  Month = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),
  Day = 1:20
)
```

 # Task 1: Summarize the above table in R

```
summary(Data_Frame)
```

 # Task 2: Find the linear regression line on the given table taking ozone as the dependent variable

```
model <- lm(Ozone ~ Solar_R + Wind + Temp + Month + Day, data = Data_Frame)
summary(model)
```

 # Task 3: Predict the 21st day's ozone level in the air with given factors

```
new_data <- data.frame(
  Solar_R = 100,
  Wind = 15,
  Temp = 70,
  Month = 5,
  Day = 21
)
predicted_ozone <- predict(model, newdata = new_data)
print(predicted_ozone)
```

 # Task 4: Find the autocorrelation of the error produced from the fitted line

```
residuals <- residuals(model)
acf(residuals)
```

 # Task 5: Analyze multicollinearity among independent variables and find a suitable solution to remove multicollinearity

```
alias_table <- alias(model)
print(alias_table)
```

 # Task 6: Find the variance among error terms and comment on the equal variance among error terms in the output

```
plot(model, which = 1)
```

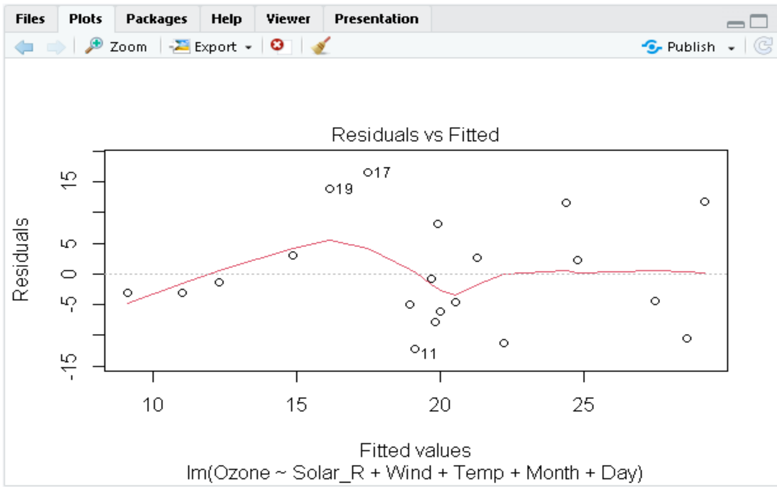
Task 7: Estimate the presence of autocorrelation using the Durbin–Watson test statistic

dw_test <- durbinWatsonTest(model)

dw_statistic <- dw_test$statistic

print(dw_statistic)

Output:



2. Multiple Linear Regression and Diagnostic of Environment Data

Aim:

To assess the impact of multiple environmental factors on a specific environmental variable using multiple linear regression

Algorithm:

Step 1: Load Required Libraries

In this step, we load the necessary libraries: `stats` for statistical functions, `lmtest` for additional linear regression testing, and `car` for regression diagnostics.

Step 2: Create the Data Frame

In this step, a data frame named `Data_Frame` is created, containing various environmental variables such as Ozone, Solar_R, Wind, Temp, Month, and Day.

Step 3: Estimate the Regression Model

Multiple linear regression is performed with Ozone as the dependent variable and Solar_R, Wind, and Temp as predictors. The `lm` function is used to create the regression model.

Step 4: View Model Summary

This step displays a summary of the regression model, including coefficients, standard errors, t-values, p-values, and various statistics for model evaluation.

Step 5: Analyze Significance of Regression Coefficients

The significance of regression coefficients is assessed using analysis of variance (ANOVA). The results are displayed.

Step 6: Evaluate Model Fit

R-squared and adjusted R-squared values are calculated to evaluate the goodness of fit of the regression model.

Step 7: Diagnostic Checking

Cook's Distance and the Press Statistic are computed for diagnostic checking of the regression model.

Step 8: Post-Model Statistical Testing

This step includes various post-model statistical tests to ensure a better model fit and accurate predictions, including tests for heteroscedasticity, autocorrelation, multicollinearity, and residual analysis.

Step 9: Normality Testing on Error Terms

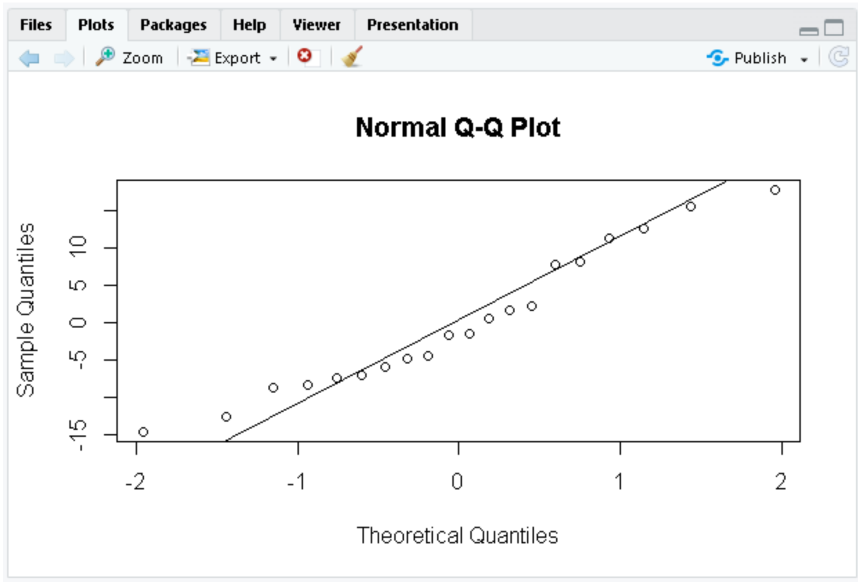
A Shapiro-Wilk test is conducted to assess the normality of the error terms in the fitted model.

Code:

```
install.packages("stats")

install.packages("lmtest")

install.packages("car")# Load required libraries

library(stats)

library(lmtest)

library(car)

# Create the dataframe

Data_Frame <- data.frame(

  S_No = 1:20,

  Ozone = c(41, 36, 12, 18, 27, 28, 23, 19, 8, 24, 7, 16, 11, 14, 18, 14, 34, 6, 30, 11),

  Solar_R = c(190, 118, 149, 313, 192, 193, 299, 99, 19, 194, 152, 256, 290, 274, 65, 334, 307, 78, 322, 44),

  Wind = c(7.4, 8, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6, 6.9, 9.7, 9.2, 10.9, 13.2, 11.5, 12, 18.4, 11.5, 9.7),

  Temp = c(67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68, 58, 64, 66, 57, 68, 62),

  Month = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),

  Day = 1:20

)

# 1. Estimate appropriate regression line with suitable predictors.

# We will perform multiple linear regression using Ozone, Solar_R, Wind, and Temp as predictors.

model <- lm(Ozone ~ Solar_R + Wind + Temp, data = Data_Frame)

summary(model) # View regression coefficients and statistics

# 2. Estimate the significance of regression coefficients using ANOVA and compare with F and partial t test.

anova_result <- anova(model)

anova_result
```

```r
# 3. Model fit using R Square and Adjusted R square values.
rsquared <- summary(model)$r.squared
adj_rsquared <- summary(model)$adj.r.squared
rsquared
adj_rsquared
# 4. Estimate Cook Statistic and Press Statistic for diagnostic checking
cooksd <- cooks.distance(model)
press_statistic <- sum((resid(model)/(1 - cooksd))^2)
cooksd
press_statistic
# 5. Post model statistical testing for better fit and error-free prediction.
# a. Breusch-Pagan Test for Heteroscedasticity
bptest(model)
# b. Durbin-Watson Test for Autocorrelation
dwtest(model)
# c. VIF (Variance Inflation Factor) for Multicollinearity
vif(model)
# d. Residual Analysis and Plots
# Check for linearity and homoscedasticity in residuals
plot(model, which = 1)
# Plot a histogram of residuals and quantile-quantile (Q-Q) plot
hist(resid(model))
qqnorm(resid(model))
qqline(resid(model))
# 6. Normality testing on error terms of the fitted model.
shapiro.test(resid(model))
```

Output:



Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Normal Q-Q Plot
Sample Quantiles
-15
-5
0
5
10
-2
-1
0
1
2
Theoretical Quantiles

3. Linear Regression Analysis of Ozone Level

Aim:

To analyze and model ozone levels using linear regression

Algorithm:

Step 1: Install and Load Required Packages

- Install and load the necessary R packages for data visualization and variance inflation factor (VIF) calculation.

Step 2: Create the Data Frame

- Create a data frame named "Data_Frame" containing information related to ozone levels and weather-related factors.

Step 3: Plot Residuals Versus Fitted Values

- Fit a linear regression model and plot residuals versus fitted values.

Step 4: Plot Residuals Versus Observed Values

- Plot residuals versus observed values.

Step 5: Plot Observed Versus Fitted Values

- Plot observed versus fitted values.

Step 6: Find Maximum Leverage Value

- Calculate and print the maximum leverage value and the corresponding observation index.

Step 7: Interpret Residual Summary

- Output the summary of the linear regression model to interpret the residual statistics.

Step 8: Calculate VIF (Variance Inflation Factor)

- Calculate the VIF values to assess multicollinearity among independent variables.

Code:

```
install.packages("ggplot2")

install.packages("car")
```

 # Load required packages

```
library(ggplot2)  # For plotting
```

library(car)   # For VIF calculation

 # Create the dataframe

Data_Frame <- data.frame(

S_No = 1:20,

Ozone = c(41, 36, 12, 18, 27, 28, 23, 19, 8, 24, 7, 16, 11, 14, 18, 14, 34, 6, 30, 11),

Solar_R = c(190, 118, 149, 313, 192, 193, 299, 99, 19, 194, 152, 256, 290, 274, 65, 334, 307, 78, 322, 44),

Wind = c(7.4, 8, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6, 6.9, 9.7, 9.2, 10.9, 13.2, 11.5, 12, 18.4, 11.5, 9.7),

Temp = c(67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68, 58, 64, 66, 57, 68, 62),

Month = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),

Day = 1:20

)

 # 1. Plot residual versus Fitted values using plot command

model <- lm(Ozone ~ Solar_R + Wind + Temp, data = Data_Frame)

residuals <- residuals(model)

fitted_values <- fitted(model)

plot(fitted_values, residuals, main = "Residuals vs. Fitted Values", xlab = "Fitted Values", ylab = "Residuals")

 # 2. Plot residual versus Observed using Plot command

plot(Data_Frame$Ozone, residuals, main = "Residuals vs. Observed", xlab = "Observed Values", ylab = "Residuals")

 # 3. Plot observed versus and fitted values using plot command

plot(fitted_values, Data_Frame$Ozone, main = "Observed vs. Fitted Values", xlab = "Fitted Values", ylab = "Observed Values")

 # 4. Find out the leverage value in the fitted values using which.max command.

leverage_values <- hatvalues(model)

max_leverage_index <- which.max(leverage_values)

max_leverage_value <- leverage_values[max_leverage_index] cat("Maximum Leverage Value:", max_leverage_value, "\n")

cat("Observation Index with Maximum Leverage:", max_leverage_index, "\n")

 # 5. Interpret the residual summary from the lm( ) command.

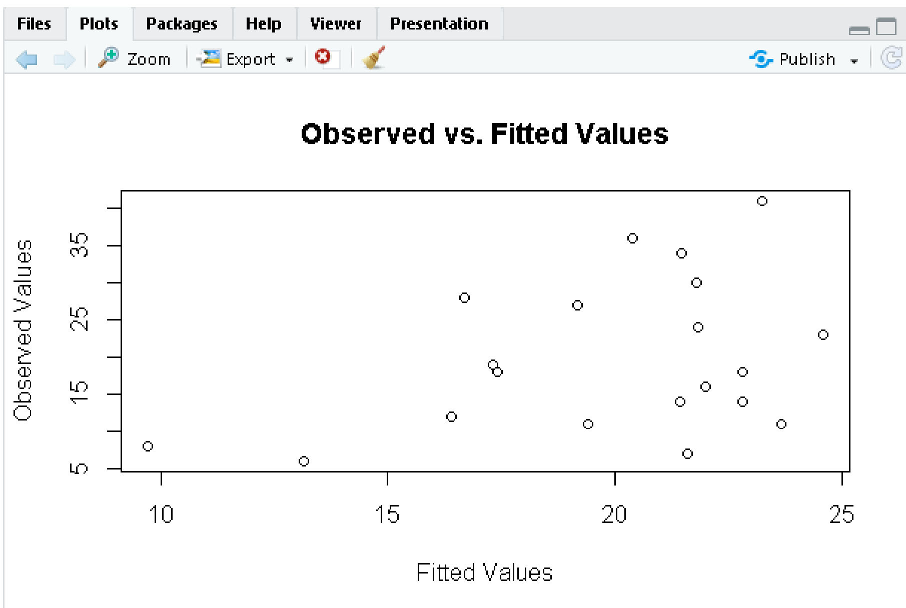
summary(model)

 # 6. Find out the VIF Variance Inflation Factor values using inbuilt function available in R.

vif_values <- vif(model)

print(vif_values)

Output:



4. Linear Regression Analysis of Study Hours, Sleep Hours, and Exam Scores

Dataset:

S.No.	study_hours	sleep_hours	examscore
1	2	7	70
2	3	6	80
3	5	5	90
4	6	6	85
5	4	8	75
6	2	5	65
7	4	5	85
8	8	6	65
9	6	8	55
10	3	4	65

Aim:

To explore and quantify the relationships between study hours, sleep hours, and exam scores through linear regression analysis.

Algorithm:

Step 1: Load Required Libraries

In this step, we load the `ggplot2` library for data visualization.

Step 2: Create a Data Frame

Here, a data frame named `Data_Frame` is created. It contains three variables: `study_hours`, `sleep_hours`, and `examscore`.

Step 3: Print the Data Frame (Optional)

This step displays the contents of the `Data_Frame` data frame.

Step 4: Explore the Data (Optional)

The `summary` function provides basic statistics about the data, such as mean, median, and quartiles. This step is optional but can help you understand the data better.

Step 5: Perform Linear Regression

In this step, a linear regression model is created. The dependent variable is `examscore`, and the independent variables are `study_hours` and `sleep_hours`. The model is stored in the `model` object.

Step 6: Summarize the Model

This step provides a summary of the linear regression model, including coefficients, standard errors, t-values, and p-values. It helps you understand the model's goodness of fit.

Step 7: Predict Student Marks Using the Model

Here, a new data frame named `new_data` is created with values for `study_hours` and `sleep_hours`. The `predict` function is used to make predictions based on the model, and the results are stored in `predicted_marks`.

Step 8: View the Predicted Marks

This step displays the predicted exam scores based on the input data.

Step 9: Plot the Data and the Regression Line

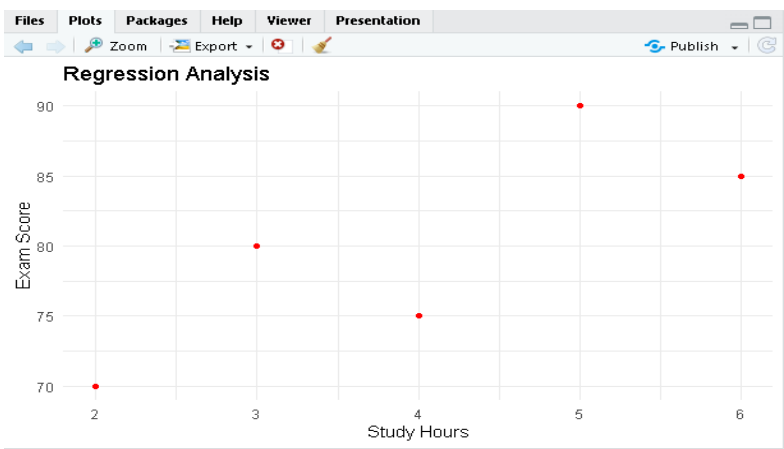
Code:

```
# Load required libraries

library(ggplot2)

# Create a data frame

Data_Frame <- data.frame (
  study_hours = c(2,3,5,6,4),
  sleep_hours = c(7,6,5,6,8),
  examscore = c(70,80,90,85,75)
)

# Print the data frame

Data_Frame

# Explore the data (optional)

summary(Data_Frame)

# Perform linear regression

model <- lm(examscore ~ study_hours + sleep_hours, data = Data_Frame)

# Summarize the model

summary(model)

# Predict student marks using the model

new_data <- data.frame(study_hours = c(4), sleep_hours = c(7))

predicted_marks <- predict(model, newdata = new_data)

# View the predicted marks

print(predicted_marks)
```

Plot the data and the regression line

```
ggplot(Data_Frame, aes(x = study_hours, y = examscore)) +
  geom_point(color = "Red") +
  geom_line(data = new_data, aes(x = study_hours, y = predicted_marks), color = "red") +
  labs(title = "Regression Analysis",
       x = "Study Hours",
       y = "Exam Score") +
  theme_minimal()
```

Output:



5. Linear Regression and Model Evaluation with Salary Data

Dataset:

S.No.	Years of Experienced	Salary
1	1.1	39343
2	1.3	46205
3	1.5	37731
4	2.0	43525
5	2.2	39891
6	2.9	56642
7	3.0	60150
8	3.2	54445
9	3.2	64445
10	3.7	57189

Aim:

To apply linear regression analysis to model salary data and evaluate the effectiveness of the model in predicting salaries based on relevant features.

Algorithm:

Step 1: Install and Load Required Packages

- Install and load necessary R packages for data manipulation, visualization, and machine learning.

Step 2: Load Data

- Load your dataset "income.xlsx" using the "read_excel" function from the "readxl" package.

Step 3: Create a Scatter Plot with Linear Regression Line

- Create a scatter plot with a linear regression line to visualize the relationship between "YearsExperienced" and "Salary."

Step 4: Data Splitting

- Split the data into a training set (70%) and a testing set (30%) based on the "Salary" column.

Step 5: Build a Linear Regression Model

- Fit a linear regression model to predict "Salary" based on "YearsExperienced" using the training data.

Step 6: Make Predictions and Evaluate

- Predict salaries on the test data and evaluate the model's performance.

Step 7: Additional Model Evaluation

- Calculate Mean Squared Error (MSE) and R-squared for further evaluation.

Code:

```
install.packages("readxl")
install.packages("caTools")
install.packages("dplyr")
install.packages("ggplot2")
library(readxl)
library(caTools)
library(readr)
library(dplyr)
library(ggplot2)
dataset1 <- read_excel("Directory of income.xlsx ")
dataset1
View(dataset1)
# Create the scatter plot with linear regression line
ggplot(dataset1, aes(x = YearsExperienced, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm", color = "blue") +
  labs(x = "Years Experienced", y = "Salary", title = "Scatter Plot of Years Experienced vs. Salary with Linear Regression Line")
# Assuming you want to split the data into 70% for training and 30% for testing
split_ratio <- 0.7
# Perform the data split based on the "Salary" column
split <- caTools::sample.split(dataset1$Salary, SplitRatio = split_ratio)
# Extract the training and testing datasets based on the split
training_data <- subset(dataset1, split == TRUE)
testing_data <- subset(dataset1, split == FALSE)
```

```r
# Fit the linear regression model on the training data

lm_model <- lm(Salary ~ YearsExperienced, data = training_data)

# Predict the salaries on the test data

predicted_salaries <- predict(lm_model, newdata = testing_data)

# Convert continuous predictions to discrete classes (High/Low) based on a threshold

threshold <- 40000

predicted_classes <- ifelse(predicted_salaries > threshold, "High", "Low")

# Convert actual salaries to discrete classes (High/Low) based on the same threshold

actual_classes <- ifelse(testing_data$Salary > threshold, "High", "Low")

# Calculate accuracy

accuracy <- mean(predicted_classes == actual_classes)

# Calculate precision

precision <- sum(predicted_classes == "High" & actual_classes == "High") / sum(predicted_classes == "High")

# Calculate recall

recall <- sum(predicted_classes == "High" & actual_classes == "High") / sum(actual_classes == "High")

# Calculate F1 score

f1_score <- 2 * precision * recall / (precision + recall)

# Display the results

cat("Accuracy:", accuracy, "\n")

cat("Precision:", precision, "\n")

cat("Recall:", recall, "\n")

cat("F1 Score:", f1_score, "\n")

# Calculate the Mean Squared Error (MSE)

mse <- mean((testing_data$Salary - predicted_salaries)^2)

# Calculate the R-squared

rsquared <- summary(lm_model)$r.squared
```

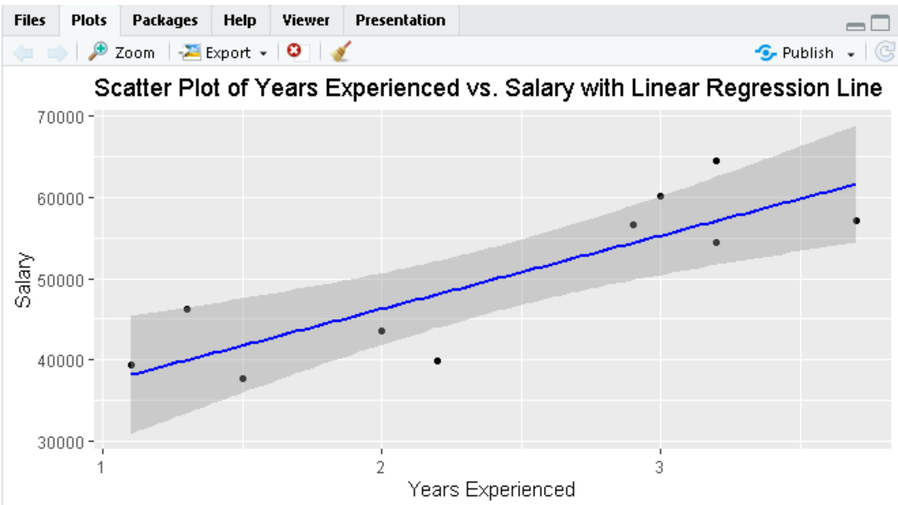
 # Display the results

print(predicted_salaries)

cat("Mean Squared Error (MSE):", mse, "\n")

cat("R-squared:", rsquared, "\n")

Output:



6. Alcohol Consumption Analysis

Dataset:

S. No.	User_ID	First_Name	Last_Name	Age	Gender	Location	Drinks_ per_ Week	Incidents_ per_ Month	Program_ Status
1	1001	John	Doe	28	Male	City A	12	2	Enrolled
2	1002	Jane	Smith	35	Female	City B	5	1	Enrolled
3	1003	Alex	Lee	22	Male	City A	20	3	Not Enrolled
4	1004	Emily	Chen	42	Female	City C	8	0	Enrolled
5	1005	Mark	Brown	31	Male	City B	15	2	Not Enrolled

Aim:

To analyze patterns of alcohol consumption and its determinants, contributing to a better understanding of drinking behaviors and potential factors influencing them.

Procedure:

Step 1: Load Necessary Libraries

- Begin by loading the required library for data visualization:

Step 2: Create Sample Alcohol User Data

- Create a sample data frame named "alcohol_data" containing information about alcohol users. Replace this with your actual data when working with your dataset:

Step 3: Generate Summary Statistics

- Calculate summary statistics for key variables in the "alcohol_data" data frame. For example:

Step 4: Create a Scatter Plot

- Visualize the relationship between variables by creating a scatter plot using ggplot2. This step includes:

  - Setting up data and aesthetics.

  - Adding points to the plot.

  - Labelling the title and axes.

Code:

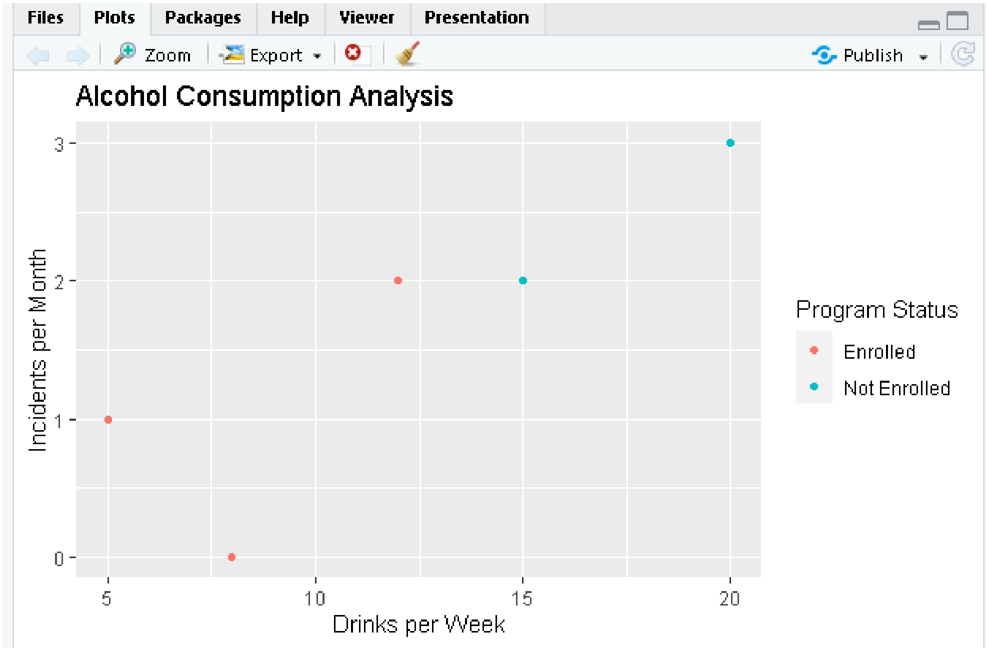
 # Load necessary libraries

library(ggplot2)   # For data visualization

 # Sample alcohol user data (replace this with your actual data)

```r
alcohol_data <- data.frame(
  User_ID = c(1001, 1002, 1003, 1004, 1005),
  First_Name = c("John", "Jane", "Alex", "Emily", "Mark"),
  Last_Name = c("Doe", "Smith", "Lee", "Chen", "Brown"),
  Age = c(28, 35, 22, 42, 31),
  Gender = c("Male", "Female", "Male", "Female", "Male"),
  Location = c("City A", "City B", "City A", "City C", "City B"),
  Drinks_per_Week = c(12, 5, 20, 8, 15),
  Incidents_per_Month = c(2, 1, 3, 0, 2),
  Program_Status = c("Enrolled", "Enrolled", "Not Enrolled", "Enrolled", "Not Enrolled")
)
# Summary statistics
summary(alcohol_data$Drinks_per_Week)
summary(alcohol_data$Incidents_per_Month)
# Create a scatter plot of Drinks per Week vs. Incidents per Month
ggplot(alcohol_data, aes(x = Drinks_per_Week, y = Incidents_per_Month, color =
Program_Status)) +
  geom_point() +
  labs(title = "Alcohol Consumption Analysis",
       x = "Drinks per Week",
       y = "Incidents per Month",
       color = "Program Status")
```

Output:



Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Alcohol Consumption Analysis
Incidents per Month
0
1
2
3
5
10
15
20
Drinks per Week
Program Status
Enrolled
Not Enrolled

7. Alcohol Consumption Analysis (Bar-Chart)

Dataset:

S.No.	User_ID	First_ Name	Last_ Name	Age	Gender	Location	Drinks_ per_ Week	Incidents_ per_ Month	Program_ Status
1	1001	John	Doe	28	Male	City A	12	2	Enrolled
2	1002	Jane	Smith	35	Female	City B	5	1	Enrolled
3	1003	Alex	Lee	22	Male	City A	20	3	Not Enrolled
4	1004	Emily	Chen	42	Female	City C	8	0	Enrolled
5	1005	Mark	Brown	31	Male	City B	15	2	Not Enrolled

Aim:

To visually represent and analyze alcohol consumption data using bar charts, allowing for a clear and insightful depiction of drinking patterns and preferences.

Algorithm:

Step 1: Load Necessary Libraries

- Start by loading the "ggplot2" library, which is used for data visualization.

Step 2: Create Sample Alcohol User Data

- Create a sample data frame named "alcohol_data" with information about alcohol users. Replace this sample data with your actual data when working with real datasets.

Step 3: Calculate Average Drinks per Week by Program Status

- Use the `aggregate` function to calculate the average "Drinks_per_Week" for each "Program_Status."

Step 4: Create a Bar Chart

- Create a bar chart to visualize the average "Drinks_per_Week" for each "Program_Status."

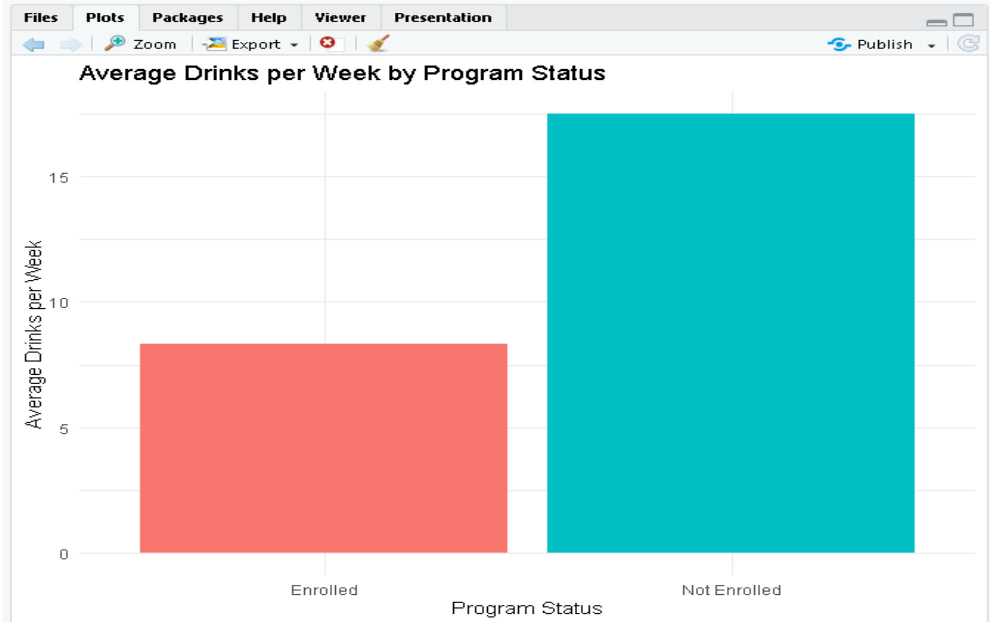
- Use ggplot2 for this purpose.

Code:

```
# Load necessary libraries
library(ggplot2)   # For data visualization
```

```r
# Sample alcohol user data (replace this with your actual data)
alcohol_data <- data.frame(
  User_ID = c(1001, 1002, 1003, 1004, 1005),
  First_Name = c("John", "Jane", "Alex", "Emily", "Mark"),
  Last_Name = c("Doe", "Smith", "Lee", "Chen", "Brown"),
  Age = c(28, 35, 22, 42, 31),
  Gender = c("Male", "Female", "Male", "Female", "Male"),
  Location = c("City A", "City B", "City A", "City C", "City B"),
  Drinks_per_Week = c(12, 5, 20, 8, 15),
  Incidents_per_Month = c(2, 1, 3, 0, 2),
  Program_Status = c("Enrolled", "Enrolled", "Not Enrolled", "Enrolled", "Not Enrolled")
)
# Create a bar chart of average Drinks per Week by Program Status
avg_drinks_by_status <- aggregate(Drinks_per_Week ~ Program_Status, data = alcohol_data, FUN = mean)
ggplot(avg_drinks_by_status, aes(x = Program_Status, y = Drinks_per_Week, fill = Program_Status)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Drinks per Week by Program Status",
       x = "Program Status",
       y = "Average Drinks per Week") +
  theme_minimal() +
  theme(legend.position = "none")
```

Output:



Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Average Drinks per Week by Program Status
Average Drinks per Week
0
5
10
15
Enrolled
Not Enrolled
Program Status