Covariance and Correlation :-

```
# Load the dataset
dataset <- read.csv("C:/Users/nadwa/Downloads/Iris.csv")

variance <- apply(dataset[, 1:4], 2, var)
covariance <- cov(dataset[, 1:4])
correlation <- cor(dataset[, 1:4])

# Print the results
cat("Variance:\n")
print(variance)
cat("\nCovariance Matrix:\n")
print(covariance)
cat("\nCorrelation Matrix:\n")
print(correlation)
```

Console Terminal Background Jobs

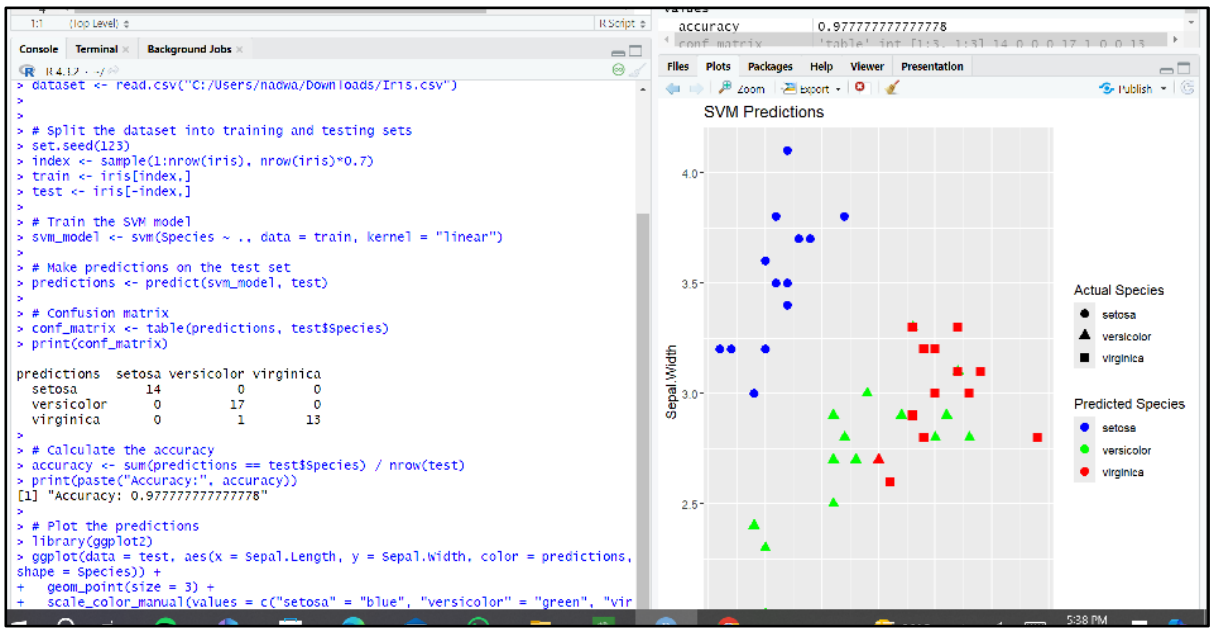
R 4.3.2 · ~/

```
> source("~/.active-rstudio-document")
Variance:
            Id SepalLengthCm  SepalWidthCm PetalLengthCm
 1887.5000000     0.6856935     0.1880040     3.1131794

Covariance Matrix:
                      Id SepalLengthCm SepalWidthCm PetalLengthCm
Id            1887.500000   25.78288591  -7.49228188    67.6677852
SepalLengthCm   25.782886    0.68569351  -0.03926846     1.2736823
SepalWidthCm    -7.492282   -0.03926846   0.18800403    -0.3217128
PetalLengthCm   67.667785    1.27368233  -0.32171275     3.1131794

Correlation Matrix:
                     Id SepalLengthCm SepalWidthCm PetalLengthCm
Id            1.0000000     0.7166763   -0.3977288     0.8827473
SepalLengthCm 0.7166763     1.0000000   -0.1093692     0.8717542
SepalWidthCm -0.3977288    -0.1093692    1.0000000    -0.4205161
PetalLengthCm 0.8827473     0.8717542   -0.4205161     1.0000000
>
```

SVM :-

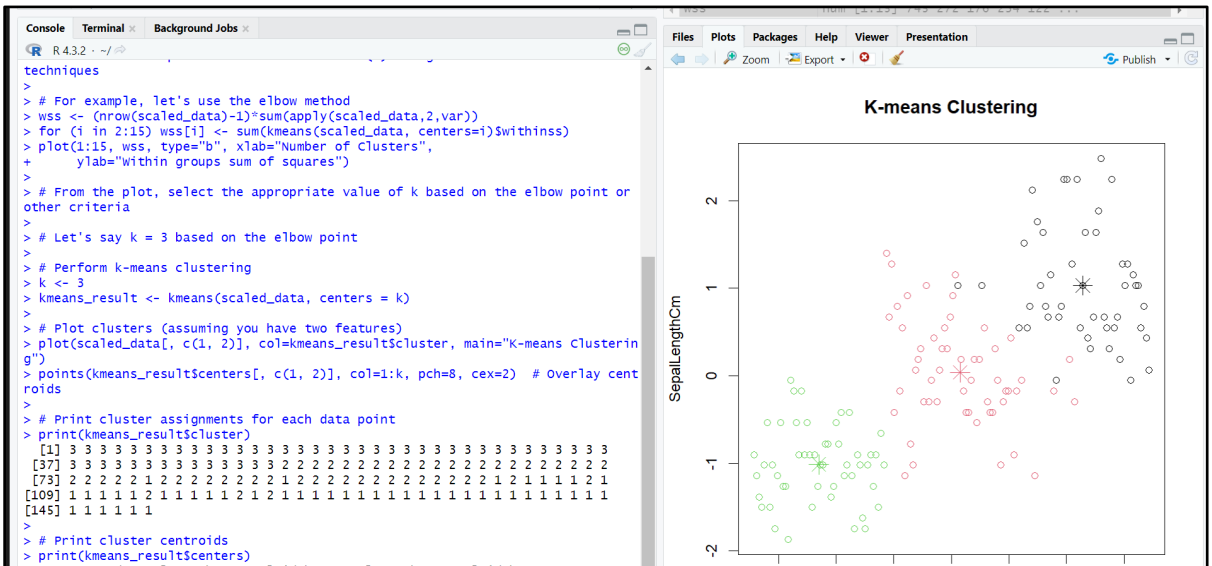
```
# Load the required library
library(e1071)

# Load the dataset
dataset <- read.csv("C:/Users/nadwa/Downloads/Iris.csv")

# Split the dataset into training and testing sets
set.seed(123)
```

```
index <- sample(1:nrow(iris), nrow(iris)*0.7)
train <- iris[index,]
test <- iris[-index,]

# Train the SVM model
svm_model <- svm(Species ~ ., data = train, kernel = "linear")

# Make predictions on the test set
predictions <- predict(svm_model, test)

# Confusion matrix
conf_matrix <- table(predictions, test$Species)
print(conf_matrix)

# Calculate the accuracy
accuracy <- sum(predictions == test$Species) / nrow(test)
print(paste("Accuracy:", accuracy))

# Plot the predictions
library(ggplot2)
ggplot(data = test, aes(x = Sepal.Length, y = Sepal.Width, color = predictions, shape = Species)) +
  geom_point(size = 3) +
  scale_color_manual(values = c("setosa" = "blue", "versicolor" = "green", "virginica" = "red")) +
  labs(title = "SVM Predictions", color = "Predicted Species", shape = "Actual Species")
```



K-Means Clustering:-

```
# Load the dataset
dataset <- read.csv("C:/Users/nadwa/Downloads/Iris.csv")

# Assuming you want to perform clustering on numerical variables only
# Select the numeric columns for clustering
numeric_data <- dataset[, sapply(dataset, is.numeric)]

# Perform scaling (optional but recommended for k-means)
scaled_data <- scale(numeric_data)

# Determine the optimal number of clusters (k) using the elbow method or other techniques

# For example, let's use the elbow method
wss <- (nrow(scaled_data)-1)*sum(apply(scaled_data,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(scaled_data, centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")

# From the plot, select the appropriate value of k based on the elbow point or other criteria

# Let's say k = 3 based on the elbow point

# Perform k-means clustering
k <- 3
kmeans_result <- kmeans(scaled_data, centers = k)

# Plot clusters (assuming you have two features)
plot(scaled_data[, c(1, 2)], col=kmeans_result$cluster, main="K-means Clustering")
points(kmeans_result$centers[, c(1, 2)], col=1:k, pch=8, cex=2)  # Overlay centroids

# Print cluster assignments for each data point
print(kmeans_result$cluster)

# Print cluster centroids
print(kmeans_result$centers)
```



Decision Tree :-

```
# Load the rpart package
library(rpart)

# Load the dataset from your computer's location
dataset <- read.csv("C:/Users/nadwa/Downloads/drug200.csv")

# Check the structure of the dataset
str(dataset)

# Train-test split (optional, depending on your workflow)
# For example, using 70% of the data for training and 30% for testing
set.seed(123)
train_index <- sample(1:nrow(dataset), 0.7 * nrow(dataset))
train_data <- dataset[train_index, ]
test_data <- dataset[-train_index, ]

# Train decision tree model
tree_model <- rpart(Drug ~ ., data = train_data, method = "class")

# Visualize the decision tree (optional)
plot(tree_model)
text(tree_model)

# Make predictions on test data (if split)
predictions <- predict(tree_model, test_data, type = "class")

# Evaluate the model (optional)
# For example, calculating accuracy
accuracy <- sum(predictions == test_data$TargetVariable) / nrow(test_data)
print(paste("Accuracy:", accuracy))
```

```
Console   Terminal   Background Jobs
R 4.3.2 · ~/
> library(rpart)
>
> # Load the dataset from your computer's location
> dataset <- read.csv("C:/Users/nadwa/Downloads/drug200.csv")
>
> # Check the structure of the dataset
> str(dataset)
'data.frame':   200 obs. of  6 variables:
 $ Age        : int  23 47 47 28 61 22 49 41 60 43 ...
 $ Sex        : chr  "F" "M" "M" "F" ...
 $ BP         : chr  "HIGH" "LOW" "LOW" "NORMAL" ...
 $ Cholesterol: chr  "HIGH" "HIGH" "HIGH" "HIGH" ...
 $ Na_to_K    : num  25.4 13.1 10.1 7.8 18 ...
 $ Drug       : chr  "drugY" "drugC" "drugC" "drugX" ...
>
> # Train-test split (optional, depending on your workflow)
> # For example, using 70% of the data for training and 30% for testing
> set.seed(123)
> train_index <- sample(1:nrow(dataset), 0.7 * nrow(dataset))
> train_data <- dataset[train_index, ]
> test_data <- dataset[-train_index, ]
>
> # Train decision tree model
> tree_model <- rpart(Drug ~ ., data = train_data, method = "class")
>
> # Visualize the decision tree (optional)
> plot(tree_model)
> text(tree_model)
>
> # Make predictions on test data (if split)
> predictions <- predict(tree_model, test_data, type = "class")
>
> # Evaluate the model (optional)
> # For example, calculating accuracy
```