UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

(FSKTM)

SEMESTER II 2024/2025

DATA MINING

BIT 33603

SECTION 03

LAB ASSIGNMENT 05

TITLE

CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) IN R

LECTURER'S NAME

DR. ROZITA BINTI ABDUL JALIL

NAME	TUAN KHALIDAH SYAZWANA BINTI TUAN MOHD KASMAWI
MATRIC NUMBER	AI220118
DATE SUBMISSION	April 16, 2025

Objectives:

1. Understand the principles of the KNN algorithm and its application in classification tasks.
2. Learn to preprocess data, handle missing values, and scale features appropriately.
3. Implement the KNN algorithm in R using relevant packages.
4. Evaluate model performance using confusion matrices and accuracy metrics.
5. Explore the impact of different values of 'k' on model performance.

Duration: 2 hours

Assessment Question:

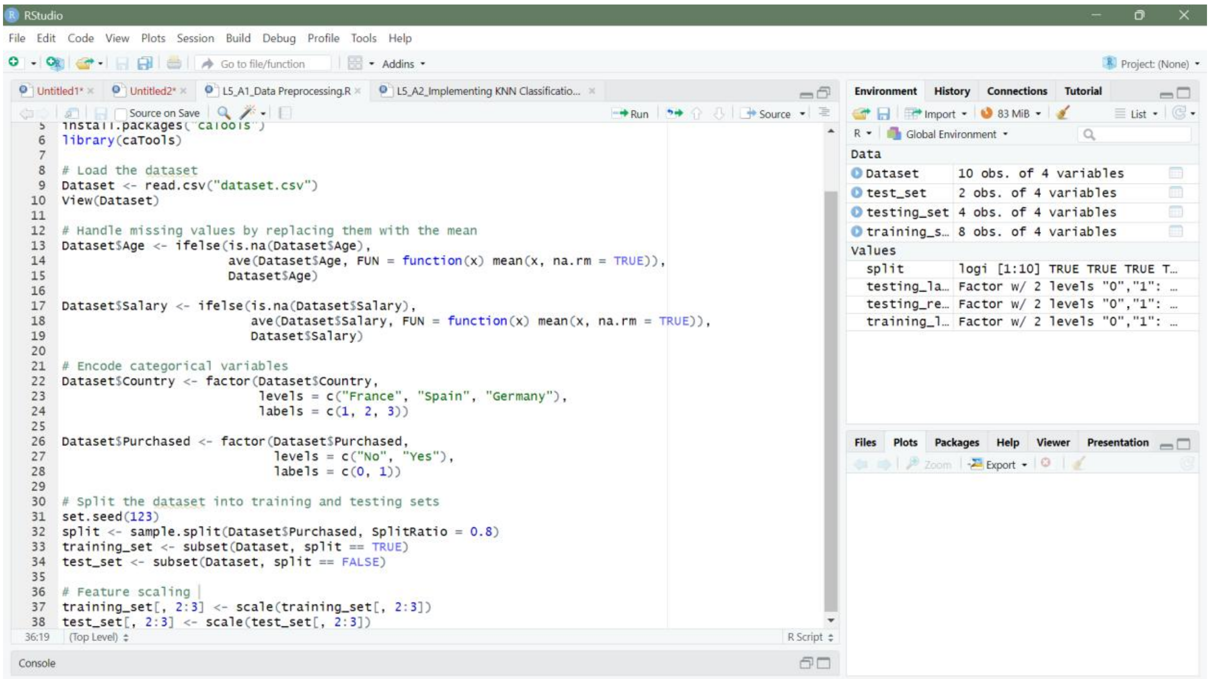
1. Run the provided code in R (Activity 1-4) and understanding the classification using KNN.
2. Submit the visualizations as image/data snapshots for each activity along with a brief explanation of the insights gained.

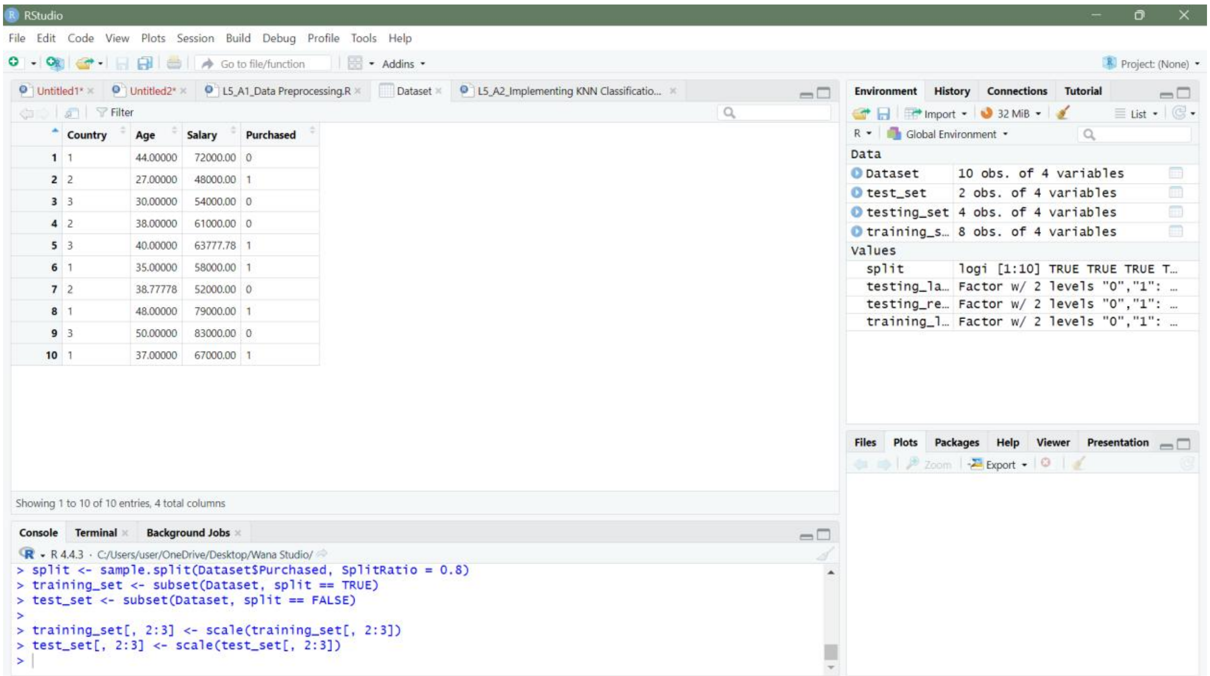
Activity 1: Data Preprocessing

Instruction:

Begin by preparing your dataset for analysis. This includes handling missing values, encoding categorical variables, and scaling numerical features.

Source code:





Justification:

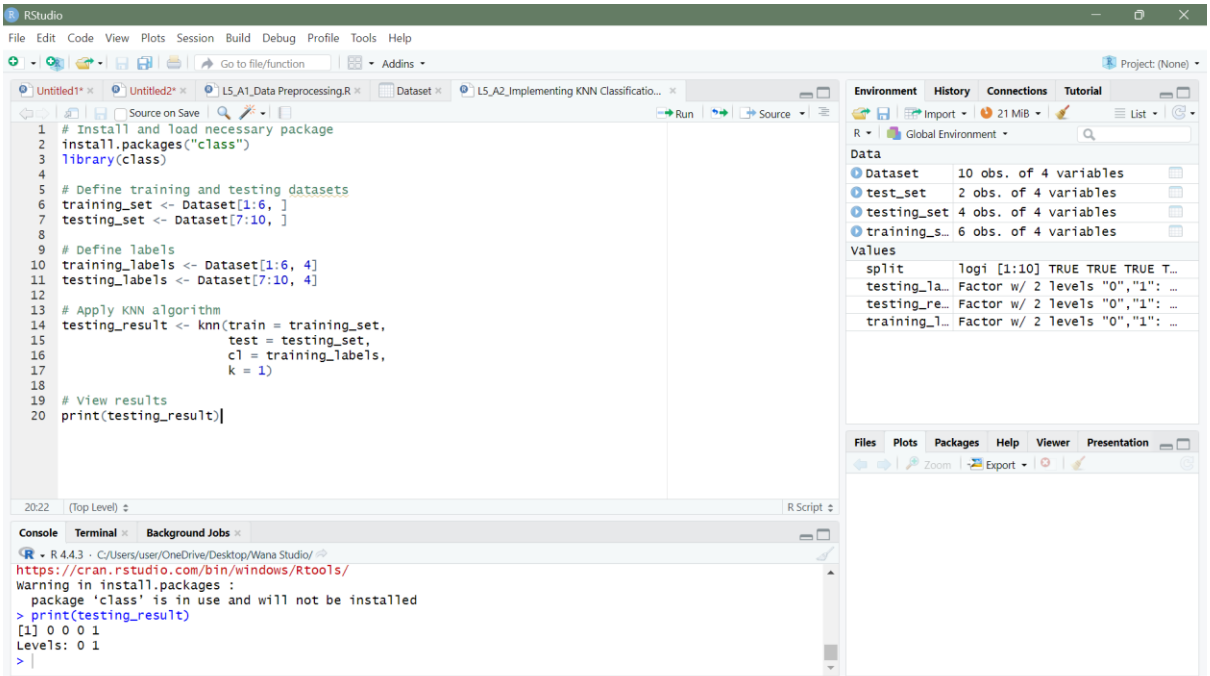
In this activity, we prepare the dataset for classification. First, we clean the data by removing any missing values in the "Age" and "Salary" columns and replacing them with the average. Then, we convert text data like "Country" and "Purchased" into numbers so the computer can understand. After that, we split the dataset into two parts: training (80%) and testing (20%). Lastly, we scale the data to make sure all numbers are on the same level. This step is important because KNN depends on distance, and large numbers can affect results.

Activity 2: Implementing KNN Classification

Instruction:

Apply the KNN algorithm to classify the test data based on the training data.

Source code:



Justification:

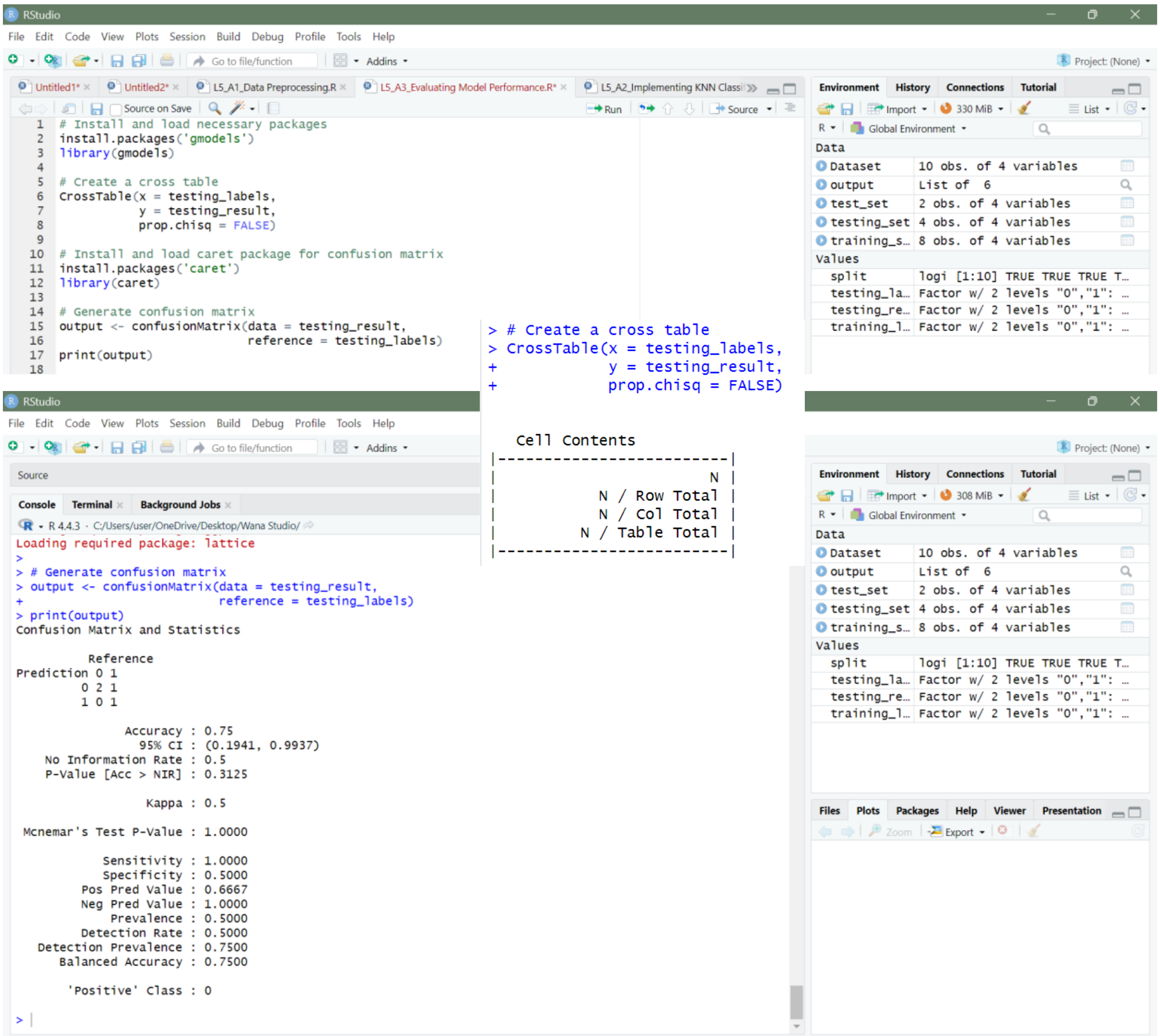
Here, we apply the K-Nearest Neighbors (KNN) algorithm to predict whether a person will purchase or not. First, we separate the training data (used to teach the model) and the testing data (used to check if the model works). We also define which column has the correct labels (Purchased column). Then, we use the `knn()` function to make predictions based on the nearest data points. If k = 1, it looks at the closest neighbor. This gives us the predicted result, which we can compare with the actual values later.

Activity 3: Evaluating Model Performance

Instruction:

Assess the performance of your KNN model using a confusion matrix and calculate accuracy metrics.

Source code:



Justification:

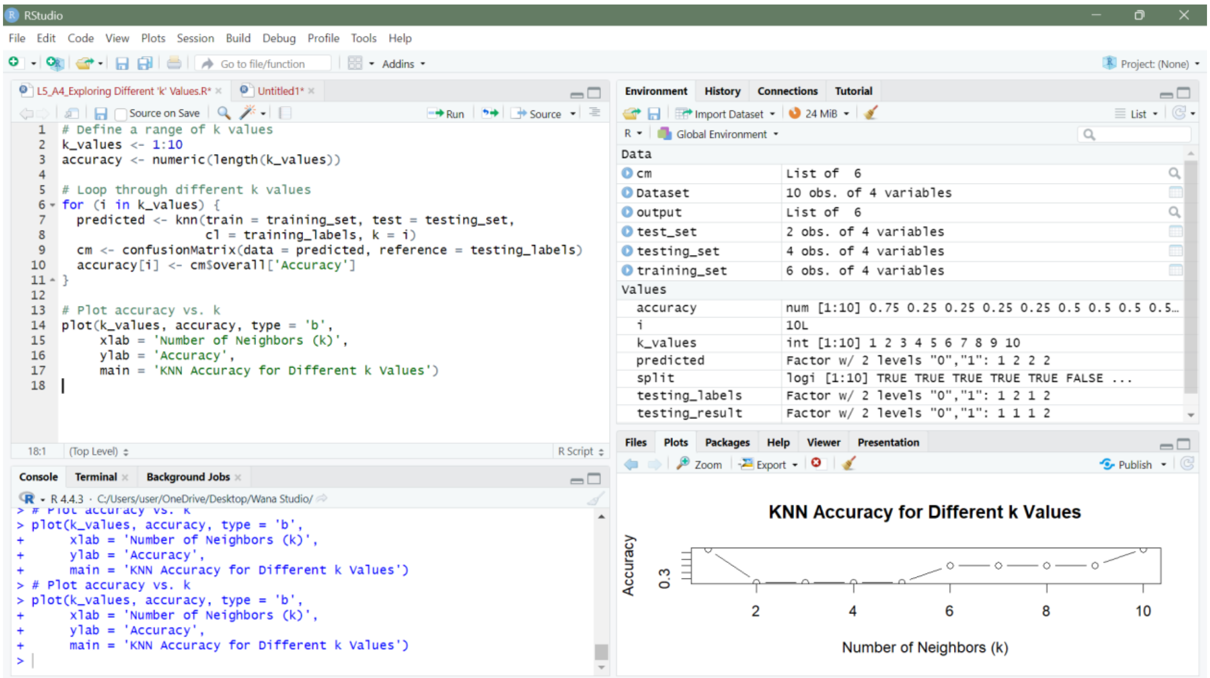
In this part, we check how good our model is. We use a confusion matrix to compare predicted results with actual answers. It shows how many were correct and how many were wrong. We use the `CrossTable()` function for a simple table and `confusionMatrix()` from the caret package for more details. It tells us the model's accuracy, which is the percentage of correct predictions. This helps us understand how well the KNN model is working on the test data.

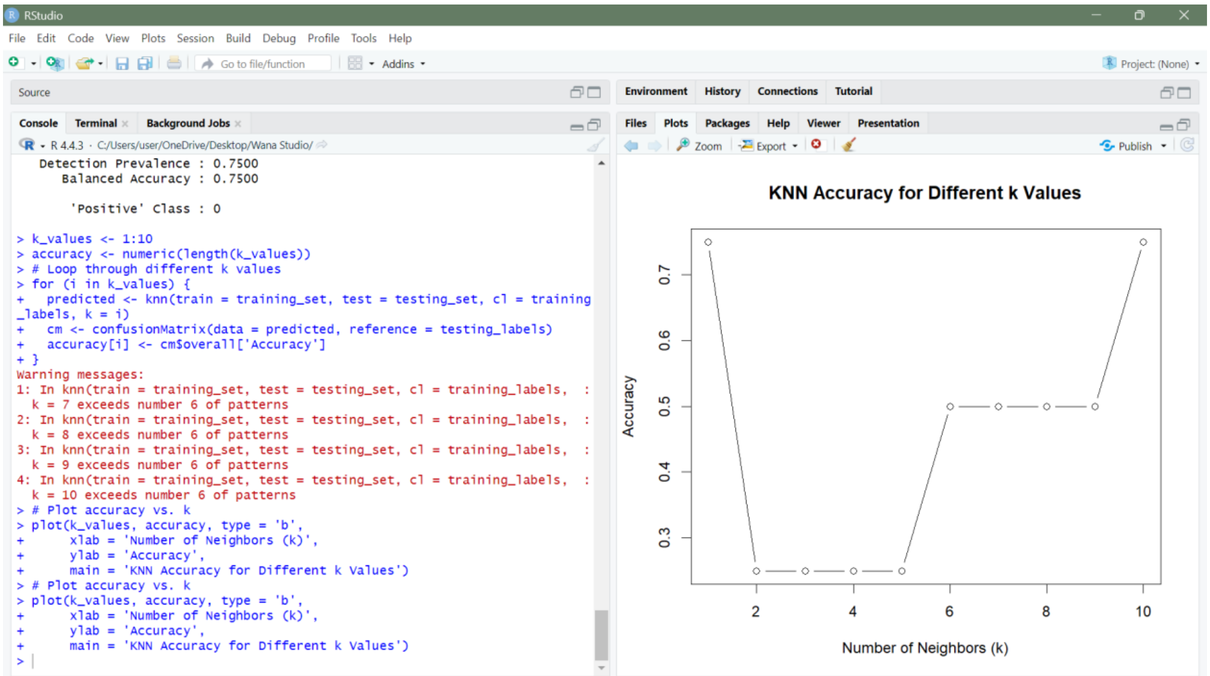
Activity 4: Exploring Different 'k' Values

Instruction:

Investigate how varying the value of 'k' affects the performance of the KNN classifier.

Source code:



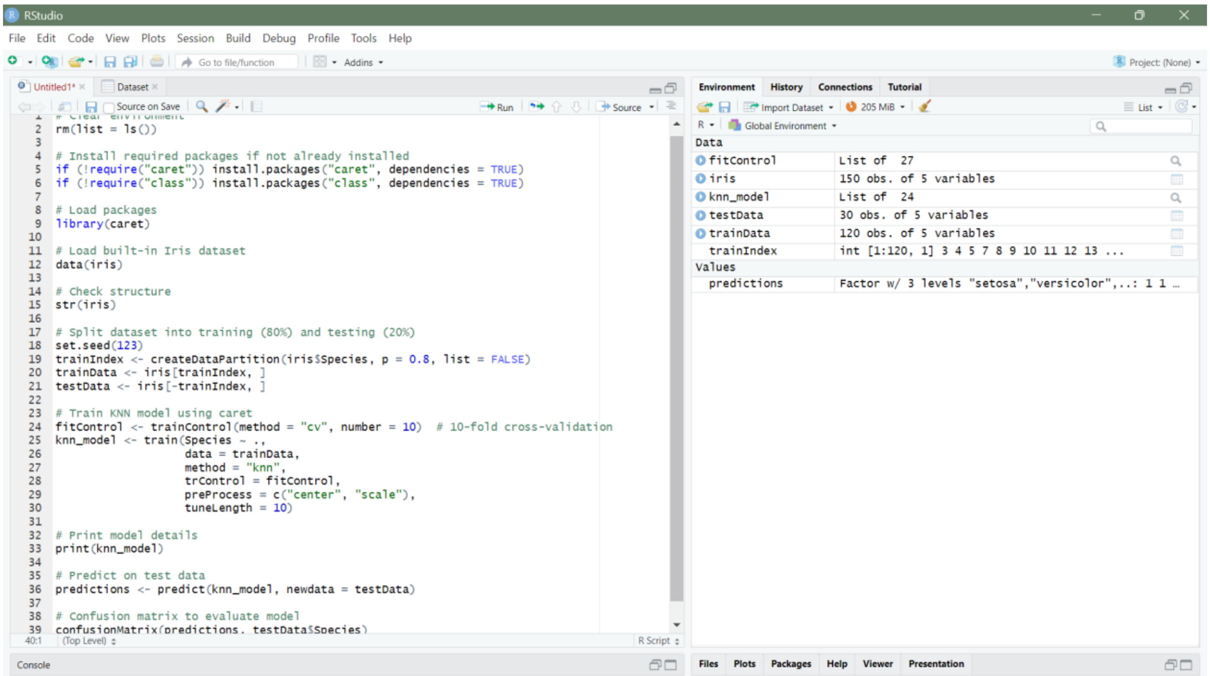


Justification:

This activity tests how changing the number of neighbors (k) affects model accuracy. We try different values of k from 1 to 10 using a loop. For each k value, we run the `knn()` function and calculate the accuracy using a confusion matrix. We store the accuracy results and plot them on a graph. This shows which k value gives the best performance. A good k value improves prediction results and avoids overfitting or underfitting. This step helps us choose the most suitable k for our model.

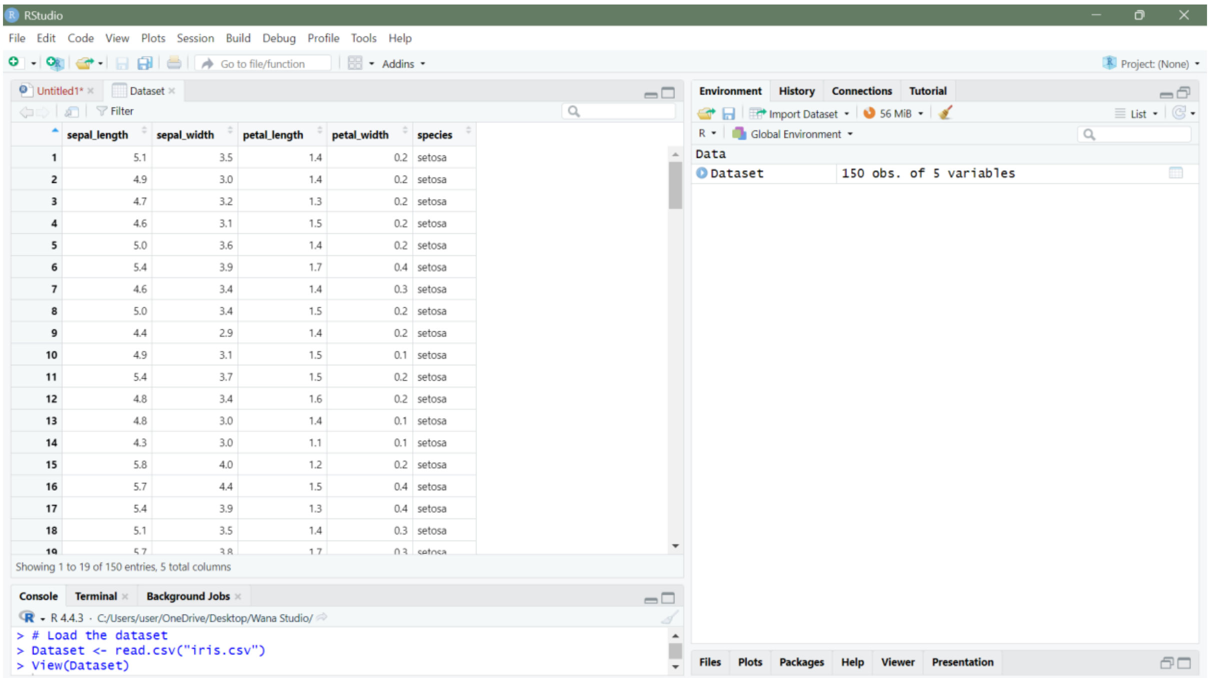
Self-assessment/study:

3. Use the built-in Iris dataset for classification using the K-Nearest Neighbors (KNN) algorithms, with split 80% training 20% testing & using caret package.




```
> # Load built-in Iris dataset
> data(iris)
> # Check structure
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> # Print model details
> print(knn_model)
k-Nearest Neighbors

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9666667  0.9500
   7  0.9583333  0.9375
   9  0.9750000  0.9625
  11  0.9583333  0.9375
  13  0.9583333  0.9375
  15  0.9583333  0.9375
  17  0.9583333  0.9375
  19  0.9416667  0.9125
  21  0.9500000  0.9250
  23  0.9333333  0.9000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
> # Confusion matrix to evaluate model
> confusionMatrix(predictions, testData$Species)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         2
  virginica       0          0         8

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.7793, 0.9918)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 8.747e-12

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.8000
Specificity                 1.0000            0.9000           1.0000
Pos Pred Value              1.0000            0.8333           1.0000
Neg Pred Value              1.0000            1.0000           0.9091
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.2667
Detection Prevalence        0.3333            0.4000           0.2667
Balanced Accuracy           1.0000            0.9500           0.9000
> 
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Dataset
Filter
sepal_length sepal_width petal_length petal_width species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
7 4.6 3.4 1.4 0.3 setosa
8 5.0 3.4 1.5 0.2 setosa
9 4.4 2.9 1.4 0.2 setosa
10 4.9 3.1 1.5 0.1 setosa
11 5.4 3.7 1.5 0.2 setosa
12 4.8 3.4 1.6 0.2 setosa
13 4.8 3.0 1.4 0.1 setosa
14 4.3 3.0 1.1 0.1 setosa
15 5.8 4.0 1.2 0.2 setosa
16 5.7 4.4 1.5 0.4 setosa
17 5.4 3.9 1.3 0.4 setosa
18 5.1 3.5 1.4 0.3 setosa
Showing 1 to 19 of 150 entries, 5 total columns
Console Terminal Background Jobs
R 4.4.3 · C:/Users/user/OneDrive/Desktop/Wana Studio/
> # Load the dataset
> Dataset <- read.csv("iris.csv")
> View(Dataset)
Environment History Connections Tutorial
Import Dataset
56 MiB
List
R
Global Environment
Data
Dataset 150 obs. of 5 variables
Files Plots Packages Help Viewer Presentation