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 08

TITLE

CLASSIFICATION SVM, RF, AND NB

LECTURER'S NAME

DR. ROZITA BINTI ABDUL JALIL

NAME	TUAN KHALIDAH SYAZWANA BINTI TUAN MOHD KASMAWI
MATRIC NUMBER	AI220118
DATE SUBMISSION	May 14, 2025

LAB ACTIVITY 8

Lab Objectives:

1. Understand and implement three classification techniques using Support Vector Machine (SVM), Random Forest and Naïve Bayes in R.
2. Compare the classification performance of different algorithms using evaluation metrics such as accuracy, precision, recall, and F-score.

Topic 1: Classification with Support Vector Machine(SVM) in R.

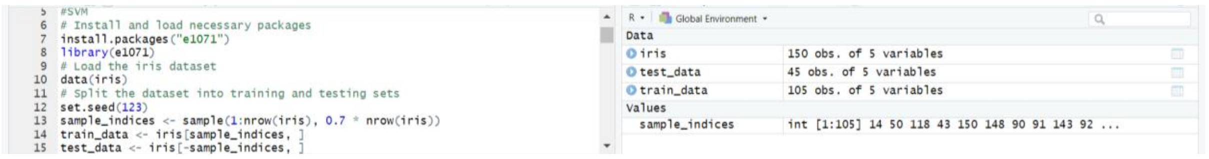
Activity 1: Data Preparation and Package Setup

Instruction:

In this activity, you will install and load the required package, prepare the iris dataset, and split

it into training and testing datasets (70-:20 ratio).

Source code:



```
#SVM
# Install and load necessary packages
install.packages("e1071")
library(e1071)
# Load the iris dataset
data(iris)
# Split the dataset into training and testing sets
set.seed(123)
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
```

```
Data
iris            150 obs. of 5 variables
test_data       45 obs. of 5 variables
train_data      105 obs. of 5 variables
Values
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

This activity prepares the dataset for classification using the Support Vector Machine (SVM) model. First, it clears the environment to avoid any leftover data or errors. Then, it installs and loads the required e1071 package, which contains the SVM functions. Next, it loads the built-in iris dataset and splits it into training (70%) and testing (30%) sets. This split is important so the model can learn from one part of the data and be tested on another to see how accurate it is when classifying new, unseen data.

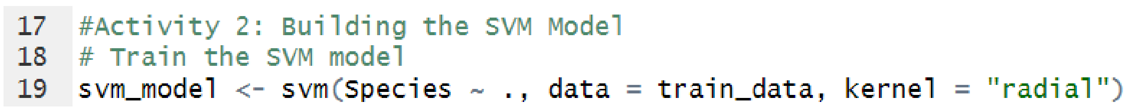
Activity 2: Building the SVM Model

Instruction:

In this activity, you will train an SVM model using the e1071 package with the radial basis

function kernel.

Source code:



```
#Activity 2: Building the SVM Model
# Train the SVM model
svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
```

Justification:

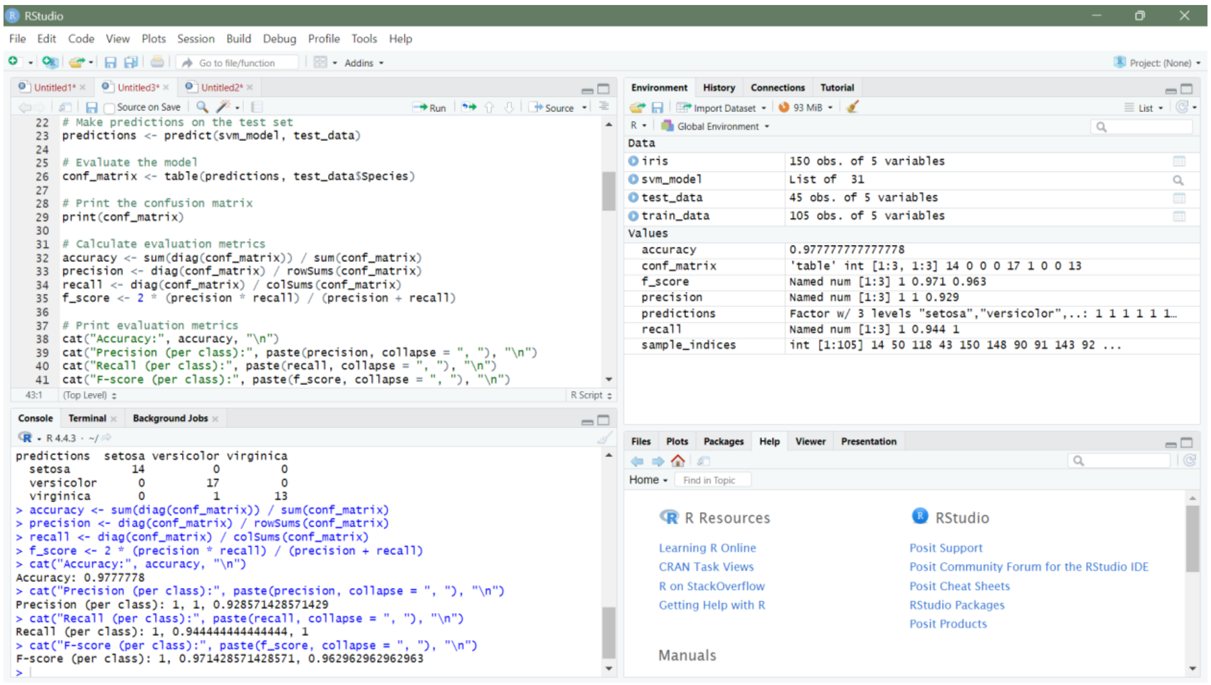
In this activity, we use the training dataset to build an SVM model. The model is created using the svm() function with a radial basis function (RBF) kernel, which helps handle complex data that cannot be separated by a straight line. The model learns the relationship between flower measurements (like petal length and width) and the flower species. Once trained, the model can predict the species of new flowers. This activity is crucial as it builds the "brain" of the classification system.

Activity 3: Evaluating the Model

Instruction:

Here you will test the trained model, generate predictions, and compute evaluation metrics including confusion matrix, accuracy, precision, recall, and F-score.

Source code:



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

# Evaluate the model
conf_matrix <- table(predictions, test_data$Species)

# Print the confusion matrix
print(conf_matrix)

# Calculate evaluation metrics
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
precision <- diag(conf_matrix) / rowSums(conf_matrix)
recall <- diag(conf_matrix) / colSums(conf_matrix)
f_score <- 2 * (precision * recall) / (precision + recall)

# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
```

```
predictions  setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> precision <- diag(conf_matrix) / rowSums(conf_matrix)
> recall <- diag(conf_matrix) / colSums(conf_matrix)
> f_score <- 2 * (precision * recall) / (precision + recall)
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.9777778
> cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
Precision (per class): 1, 1, 0.928571428571429
> cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
Recall (per class): 1, 0.944444444444444, 1
> cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
F-score (per class): 1, 0.971428571428571, 0.962962962962963
> 
```

Data	
iris	150 obs. of 5 variables
svm_model	List of 31
test_data	45 obs. of 5 variables
train_data	105 obs. of 5 variables
Values	
accuracy	0.977777777777778
conf_matrix	'table' int [1:3, 1:3] 14 0 0 0 17 1 0 0 13
f_score	Named num [1:3] 1 0.971 0.963
precision	Named num [1:3] 1 1 0.929
predictions	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1…
recall	Named num [1:3] 1 0.944 1
sample_indices	int [1:105] 14 50 118 43 150 148 90 91 143 92 ...

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1* Untitled3* Untitled2*
Source on Save
Run
Source
Console Terminal Background Jobs
R 4.4.3 · ~/
Environment History Connections Tutorial
Import Dataset
93 MiB
Global Environment
Files Plots Packages Help Viewer Presentation
R Resources
Learning R Online
CRAN Task Views
R on StackOverflow
Getting Help with R
RStudio
Posit Support
Posit Community Forum for the RStudio IDE
Posit Cheat Sheets
RStudio Packages
Posit Products
Manuals

Justification:

This activity checks how well the SVM model performs. It uses the test dataset to make predictions and compares the results with the actual flower species. A confusion matrix is generated to see which predictions were correct or wrong. Then, it calculates important evaluation metrics: accuracy (overall correctness), precision (correctness of positive predictions), recall (how many actual positives were found), and F-score (balance of precision and recall). This helps us understand the strengths and weaknesses of the SVM model.

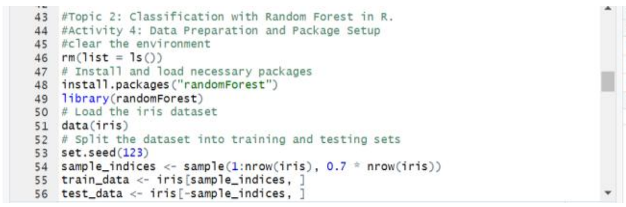
Activity 4: Data Preparation and Package Setup

Instruction:

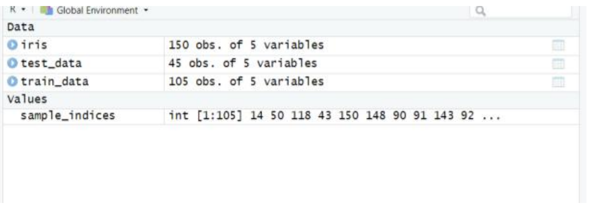
In this activity, you will install and load the required package, prepare the iris dataset, and

split it into training and testing datasets(70-:20 ratio).

Source code:



```
#Topic 2: Classification with Random Forest in R.
#Activity 4: Data Preparation and Package Setup
#clear the environment
rm(list = ls())
# Install and load necessary packages
install.packages("randomForest")
library(randomForest)
# Load the iris dataset
data(iris)
# Split the dataset into training and testing sets
set.seed(123)
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
```



```
Data
iris            150 obs. of 5 variables
test_data       45 obs. of 5 variables
train_data      105 obs. of 5 variables
Values
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

This activity sets up the environment for the Random Forest model. It clears the workspace, installs and loads the randomForest package, and loads the iris dataset. The data is split into 70% training and 30% testing. Splitting the data ensures we can train the model and then evaluate it on new data. This setup is important to ensure the model learns well and can be tested fairly. The package and data preparation must be correct for the model to work properly.

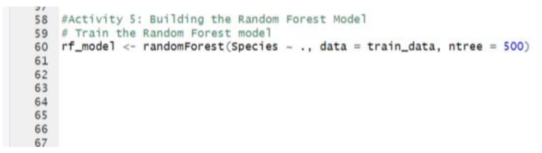
Activity 5: Building the Random Forest Model

Instruction:

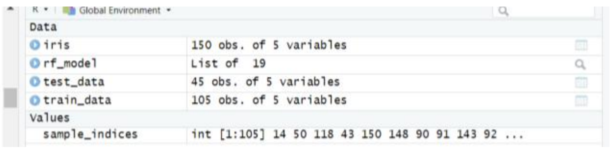
In this activity, you will train a Random Forest model on the training dataset using 500 trees

and apply it to classify iris species.

Source code:



```
#Activity 5: Building the Random Forest Model
# Train the Random Forest model
rf_model <- randomForest(Species ~ ., data = train_data, ntree = 500)
```



```
Data
iris            150 obs. of 5 variables
rf_model        List of  19
test_data       45 obs. of 5 variables
train_data      105 obs. of 5 variables
Values
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

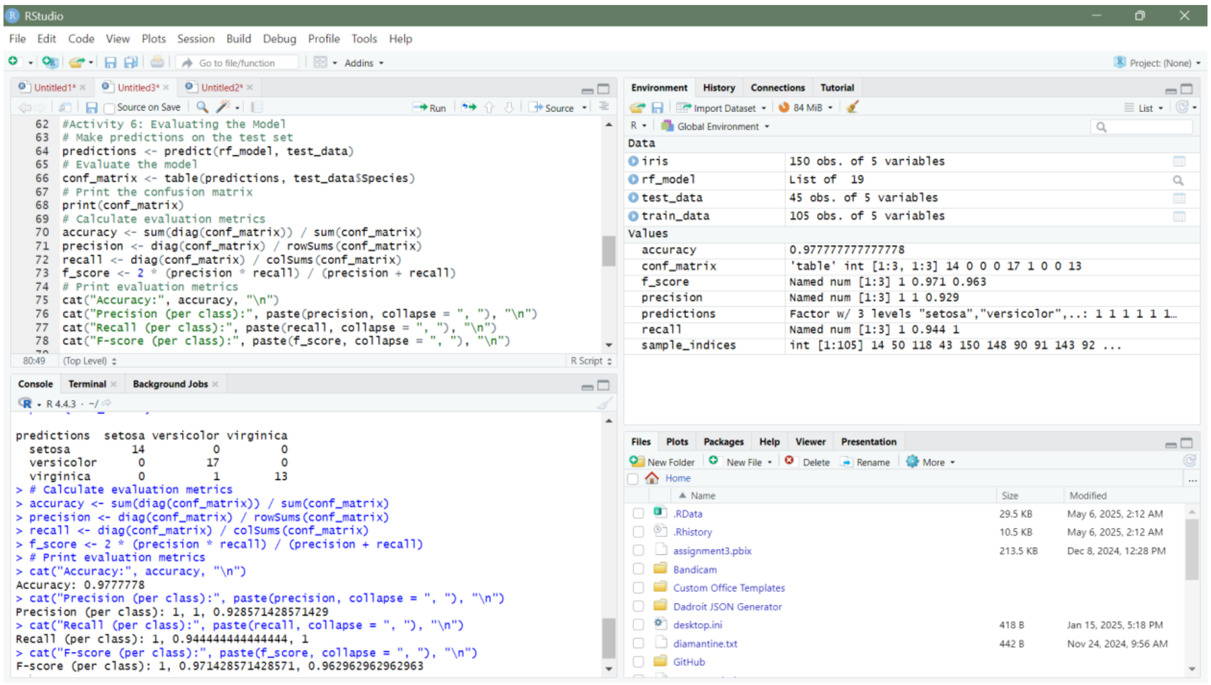
In this activity, a Random Forest model is trained using the training dataset. The model is built using 500 decision trees to make it more powerful and accurate. Each tree learns to classify the flower species based on different combinations of features. The results from all trees are combined to make the final prediction. Random Forest is strong because it reduces the chances of errors that might happen if we use only one decision tree. This activity builds a strong and stable model.

Activity 6: Evaluating the Model

Instruction:

Here you will test the trained model, generate predictions, and compute evaluation metrics including confusion matrix, accuracy, precision, recall, and F-score.

Source code:



```r
#Activity 6: Evaluating the Model
# Make predictions on the test set
predictions <- predict(rf_model, test_data)
# Evaluate the model
conf_matrix <- table(predictions, test_data$Species)
# Print the confusion matrix
print(conf_matrix)
# Calculate evaluation metrics
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
precision <- diag(conf_matrix) / rowSums(conf_matrix)
recall <- diag(conf_matrix) / colSums(conf_matrix)
f_score <- 2 * (precision * recall) / (precision + recall)
# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
```

```
predictions  setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
> # Calculate evaluation metrics
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> precision <- diag(conf_matrix) / rowSums(conf_matrix)
> recall <- diag(conf_matrix) / colSums(conf_matrix)
> f_score <- 2 * (precision * recall) / (precision + recall)
> # Print evaluation metrics
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.9777778
> cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
Precision (per class): 1, 1, 0.928571428571429
> cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
Recall (per class): 1, 0.944444444444444, 1
> cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
F-score (per class): 1, 0.971428571428571, 0.962962962962963
```

Data	
iris	150 obs. of 5 variables
rf_model	List of 19
test_data	45 obs. of 5 variables
train_data	105 obs. of 5 variables
Values	
accuracy	0.977777777777778
conf_matrix	'table' int [1:3, 1:3] 14 0 0 0 17 1 0 0 13
f_score	Named num [1:3] 1 0.971 0.963
precision	Named num [1:3] 1 1 0.929
predictions	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1…
recall	Named num [1:3] 1 0.944 1
sample_indices	int [1:105] 14 50 118 43 150 148 90 91 143 92 ...

Name	Size	Modified
.RData	29.5 KB	May 6, 2025, 2:12 AM
.Rhistory	10.5 KB	May 6, 2025, 2:12 AM
assignment3.pbix	213.5 KB	Dec 8, 2024, 12:28 PM
Bandicam		
Custom Office Templates		
Dadroit JSON Generator		
desktop.ini	418 B	Jan 15, 2025, 5:18 PM
diamantine.txt	442 B	Nov 24, 2024, 9:56 AM
GitHub		

Justification:

Here, we test the Random Forest model using the test data. The model makes predictions, and we compare them with the actual flower types using a confusion matrix. Then, we calculate accuracy, precision, recall, and F-score to measure how well the model did. These metrics show us how often the model was right and whether it treated each flower class fairly. This activity is important because it confirms whether the Random Forest model is reliable and performs well.

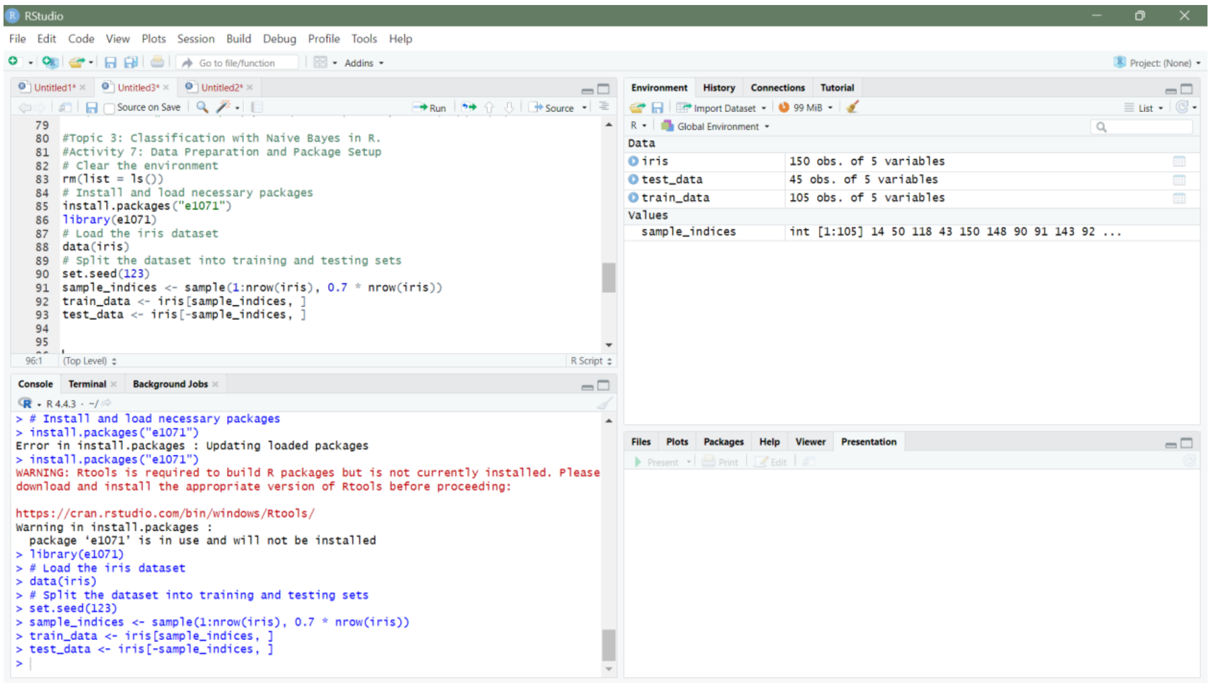
Activity 7: Data Preparation and Package Setup

Instruction:

In this activity, you will install and load the required package, prepare the iris dataset, and

split it into training and testing datasets(70-:20 ratio).

Source code:



```r
#Topic 3: Classification with Naive Bayes in R.
#Activity 7: Data Preparation and Package Setup
# Clear the environment
rm(list = ls())
# Install and load necessary packages
install.packages("e1071")
library(e1071)
# Load the iris dataset
data(iris)
# Split the dataset into training and testing sets
set.seed(123)
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
```

```
> # Install and load necessary packages
> install.packages("e1071")
Error in install.packages : Updating loaded packages
> install.packages("e1071")
WARNING: Rtools is required to build R packages but is not currently installed. Please
download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  package ‘e1071’ is in use and will not be installed
> library(e1071)
> # Load the iris dataset
> data(iris)
> # Split the dataset into training and testing sets
> set.seed(123)
> sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
> train_data <- iris[sample_indices, ]
> test_data <- iris[-sample_indices, ]
> 
```

```
Data
iris            150 obs. of 5 variables
test_data       45 obs. of 5 variables
train_data      105 obs. of 5 variables
Values
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

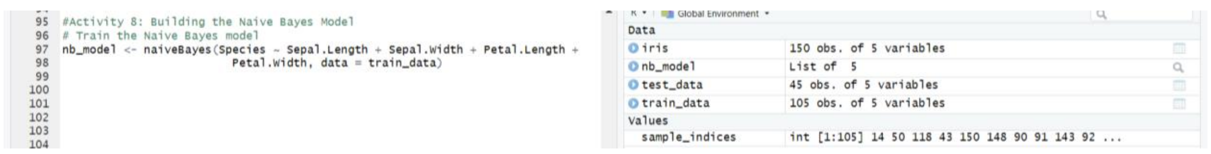
This activity prepares everything needed for building a Naïve Bayes classification model. It clears the R environment, installs and loads the e1071 package, and loads the iris dataset. The data is again split into 70% training and 30% testing sets. Naïve Bayes needs clean and properly formatted data to perform well. By setting up the dataset correctly, we ensure the model can train effectively and give good results during testing. This setup is an important foundation before building the model.

Activity 8: Building the Naive Bayes Model

Instruction:

In this activity, you will build a Naive Bayes classification model using the training data.

Source code:



```r
#Activity 8: Building the Naive Bayes Model
# Train the Naive Bayes model
nb_model <- naiveBayes(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
                       Petal.Width, data = train_data)
```

```
Global Environment
Data
iris              150 obs. of 5 variables
nb_model          List of  5
test_data         45 obs. of 5 variables
train_data        105 obs. of 5 variables
Values
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

In this activity, the Naïve Bayes model is trained using the training dataset. The model is built using the naiveBayes() function, which applies probability rules to predict the flower species. It assumes each feature (like petal and sepal length) is independent. Even though this is a simple model, it often gives good results for classification tasks. This step is essential because it creates a fast and efficient model that can be used to classify new data.

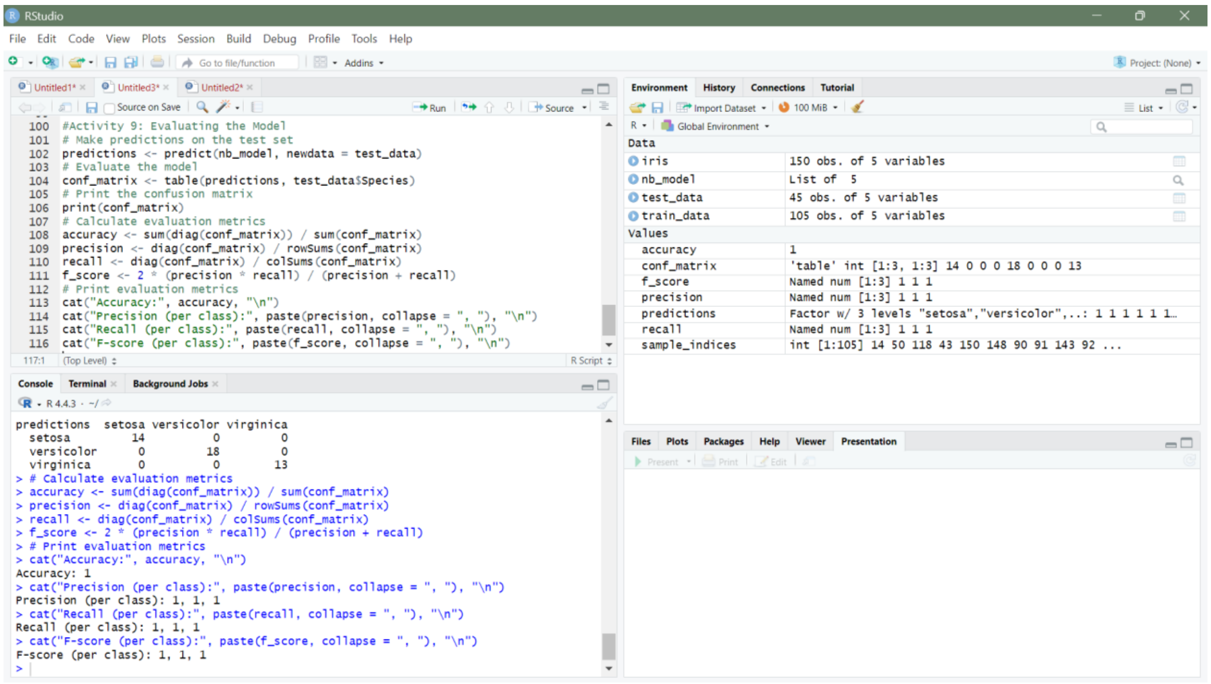
Activity 9: Evaluating the Model

Instruction:

Here you will test the trained model, generate predictions, and compute evaluation metrics

including confusion matrix, accuracy, precision, recall, and F-score.

Source code:



```r
#Activity 9: Evaluating the Model
# Make predictions on the test set
predictions <- predict(nb_model, newdata = test_data)
# Evaluate the model
conf_matrix <- table(predictions, test_data$Species)
# Print the confusion matrix
print(conf_matrix)
# Calculate evaluation metrics
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
precision <- diag(conf_matrix) / rowSums(conf_matrix)
recall <- diag(conf_matrix) / colSums(conf_matrix)
f_score <- 2 * (precision * recall) / (precision + recall)
# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
```

```
predictions  setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         18         0
  virginica       0          0        13
> # Calculate evaluation metrics
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> precision <- diag(conf_matrix) / rowSums(conf_matrix)
> recall <- diag(conf_matrix) / colSums(conf_matrix)
> f_score <- 2 * (precision * recall) / (precision + recall)
> # Print evaluation metrics
> cat("Accuracy:", accuracy, "\n")
Accuracy: 1
> cat("Precision (per class):", paste(precision, collapse = ", "), "\n")
Precision (per class): 1, 1, 1
> cat("Recall (per class):", paste(recall, collapse = ", "), "\n")
Recall (per class): 1, 1, 1
> cat("F-score (per class):", paste(f_score, collapse = ", "), "\n")
F-score (per class): 1, 1, 1
> 
```

```
Data
iris              150 obs. of 5 variables
nb_model          List of  5
test_data         45 obs. of 5 variables
train_data        105 obs. of 5 variables
Values
  accuracy        1
  conf_matrix     'table' int [1:3, 1:3] 14 0 0 0 18 0 0 0 13
  f_score         Named num [1:3] 1 1 1
  precision       Named num [1:3] 1 1 1
  predictions     Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1…
  recall          Named num [1:3] 1 1 1
  sample_indices  int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Justification:

In this activity, we test the Naïve Bayes model using the test dataset. Predictions are made and compared with the actual flower species using a confusion matrix. We then calculate accuracy, precision, recall, and F-score to see how good the model is. These metrics help us check whether the model works well for all flower types. This step is important to evaluate the quality of the predictions and to compare the model fairly with SVM and Random Forest.

Assessment Questions:

Use the iris dataset in R, Apply SVM, Random Forest, and Naive Bayes (Activity 1 - 9) to classify the flower species. Then answer the following questions:

1. For each model (SVM, Random Forest, and Naive Bayes):
    a) Show the confusion matrix output
    b) Report these metrics:

	SVM	**RF**	**NB**
Confusion Matrix Output	predictions setosa vericolor virginica setosa 14 0 0 versicolor 0 17 0 virginica 0 1 13	predictions setosa vericolor virginica setosa 14 0 0 versicolor 0 17 0 virginica 0 1 13	predictions setosa vericolor virginica setosa 14 0 0 versicolor 0 18 0 virginica 0 0 13
Accuracy	0.98	0.98	1
Precision (per class)	1, 1, 0.93	1, 1, 0.93	1, 1, 1
Recall (per class)	1, 0.94, 1	1, 0.94, 1	1, 1, 1
F-score (per class)	1, 0.97, 0.96	1, 0.97, 0.96	1, 1, 1

2. Compare the performance of all three models (SVM, RF, NB):
    a) Which model gave the best accuracy?
    Naive Bayes achieved the highest accuracy, with a perfect score of 1.0(100%).
    Both SVM and Random Forest had an accuracy of 0.98 (98%).

    b) Which model was the most balanced across all classes (based on precision, recall, F-score)?
    Naive Bayes was the most balanced model, as it scored 1.0 in all metrics (precision, recall, and F-score) for all three classes (setosa, versicolor, virginica).
    In contrast, both SVM and Random Forest had slightly lower scores for the versicolor and virginica classes, especially in precision and F-score.

    c) In your opinion, which model is most suitable for this classification task and why?
    Based on the results, Naive Bayes is the most suitable model for this classification task using the Iris dataset, It achieved perfect classification across all three classes, indicating excellent generalization and no misclassification.
    Moreover, Naive Bayes is computationally efficient and easy to interpret, making it a strong choice especially when the dataset is well-behaved and the assumptions of feature independence are not violated.