Analysis of the Wisconsin Dataset using machine learning algorithms.

Maryam Akhtar

Student ID: 239528371

Word count: 1411 (excluding source code listing and references)

1. Introduction

Breast cancer is a critical healthcare concern and the most commonly misdiagnosed cancer in women. It is estimated that one in 5 people worldwide will develop cancer during their lifetime (Romm, A. 2010). This reinforces the need to invest in the fight against cancer. Data mining algorithms applied in the healthcare industry play a significant role due to their high performance in predicting, diagnosis of the diseases, reducing costs of medicine, making real-time decisions to save people's lives.

One of the main objectives is to harness the power of machine learning to construct predictive models that not only classify breast cancer instances but also find out which model is most accurate. We will be using RStudio and adopt a systemic methodology encompassing data loading, pre-processing, and model training. This report encapsulates the journey of unravelling data and insights from the Wisconsin dataset.

2. Data used.

Our analysis comprises of the Wisconsin Breast Cancer dataset and this data includes information taken from FNA biopsies, giving an insight into the characteristics of cell nuclei. Some key variables include radius, texture, and smoothness, which are important in the identification of malignancy. There variables are numerical and give an understanding of cell structures. The Wisconsin dataset has 569 instances (Benign: 357 Malignant: 212), two classes (62.74% benign and 37.26% malignant (Alshayeji, M. *et al.* 2021).

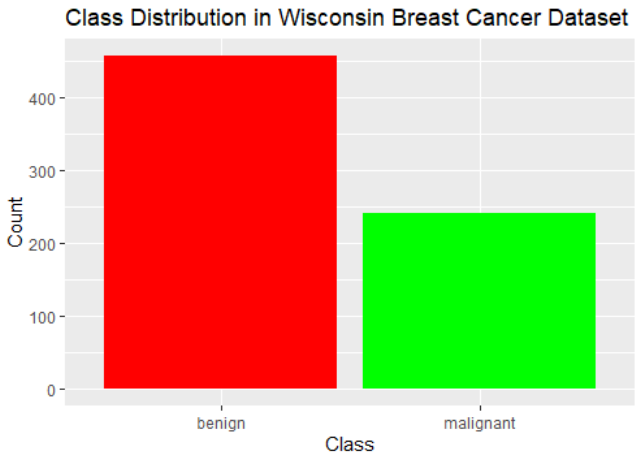


Figure 1: Wisconsin Dataset classes.

For better understanding and relevance of the Wisconsin dataset, we can look into other literature. One study looked at the application of linear programming in both the prognosis and diagnosis of breast cancer. It placed emphasis on the importance of accuracy modelling (Asri, H. *et al.* 2016). Another study explored the efficacy of artificial neural networks when classifying breast cancer, showing the evolution of machine learning methods in medicine (Alshayeji, M. *et al.* 2021). These pieces of information set the foundation for further analysis and classification of this dataset.

3. Machine Learning methods used:

For the Wisconsin dataset, there are various different machine learning methods that can be used for classification. We will be giving a comparison between the performance of 3 classifiers: Logistic Regression, Decision trees and Support Vector Machine (SVM) (Nasser, M. and Yusuf, U.K. 2023). Our objective is to identify the most effective and accurate algorithm for detecting breast cancer using machine-learning algorithms. The diagram (figure 2) shows the proposed process.

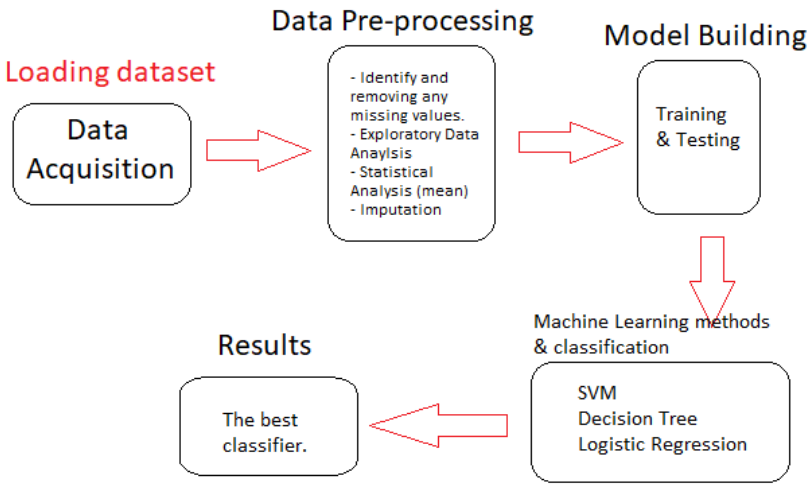


Figure 2: Diagram showing the process.

The first stage in the method begins at data acquisition and pre-processing. The data will essentially be cleaned and ready for building the machine learning algorithms. In this stage, I will complete an exploratory data analysis (EDA) (Asri, H. *et al.* 2016).

For the analysis, I will be using logistic regression because it is simple and can be interpreted well for binary classification. I will use a decision tree because it can make data easy to understand and is good to check for accuracy (Asri, H. *et al.* 2016). In relation to the Wisconsin dataset, the decision tree model is built to predict whether a tumour is malignant (1) or benign (0). Lastly, I used a Support Vector machine (SVM) because it can be used for both linear and non-linear classification. I can use it to check accuracy and it is effective when analysing a large dataset. I used multiple algorithms, comparing their performance using metrics like accuracy, confusion matrices, and precision to identify the best approach for this specific classification task (Nasser, M. and Yusuf, U.K. 2023).

4. Practical: Pre-processing of data:

To begin, I set the folder path and downloaded the file called Wisconsin.csv. For the pre-processing, I utilised R programming to read in the data and implemented the essential pre-processing steps (Enes, A. and Karakoyun, M. 2023). Screenshots of the relevant R code snippets are provided below. First, I important the data using the 'read.csv' function to ensure the data was loaded accurately.

Next, I completed an Exploratory data analysis (EDA). I checked for any missing values. There are 16 missing values.

```
5  # find the location of missing values
6  which(is.na(data))
7
8  # find the count of missing values
9  sum(is.na(data))
10
11
```

11:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.3.2 · C:/Users/user/Desktop/MSc Computer Science/CETM72/Wisconsin Data/

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
> setwd("C:/Users/user/Desktop/MSc Computer Science/CETM72/Wisconsin Data")
> data <- read.csv("wisconsin.csv")
> # Check for missing values and get a summary
> missing_summary <- data.frame
> # find the location of missing values
> which(is.na(data))
 [1] 3519 3536 3635 3641 3654 3660 3731 3745 3771 3788 3790 3793 3811 3817 3907 4113
> # find the count of missing values
> sum(is.na(data))
[1] 16
```

I then accessed the Excel dataset and located the 16 missing values. The missing data was in the 'bare nuclei' column and was represented as 'na'. All the other values in this column had a numerical value between 1 and 10. I used '0' to represent the 'na' value.

1	Cl.thickne	Cell.size	Cell.shap	Marg.adh	Epith.c.si	Bare.nu	Bl.cromat	Normal.n	Mitoses	Class
25	8	4	5	1	2	NA	7	3	1	malignant
42	6	6	6	9	6	NA	7	8	1	benign
141	1	1	1	1	1	NA	2	1	1	benign
147	1	1	3	1	2	NA	2	1	1	benign
160	1	1	2	1	3	NA	1	1	1	benign
166	5	1	1	1	2	NA	3	1	1	benign
237	3	1	4	1	2	NA	3	1	1	benign
251	3	1	1	1	2	NA	3	1	1	benign
277	3	1	3	1	2	NA	2	1	1	benign
294	8	8	8	1	2	NA	6	10	1	malignant
296	1	1	1	1	2	NA	2	1	1	benign
299	5	4	3	1	2	NA	2	3	1	benign
317	4	6	5	6	7	NA	4	9	1	benign
323	3	1	1	1	2	NA	3	1	1	benign
413	1	1	1	1	1	NA	2	1	1	benign
619	1	1	1	1	1	NA	1	1	1	benign
701										
702										

Next, I calculated the mean value in the 'bare nuclei' column and the result was 3.544656.

```
10
11 mean(data$Bare.nuclei, na.rm = TRUE)
12

> mean(data$Bare.nuclei, na.rm = TRUE)
[1] 3.544656
```

For all numerical variables, I computed a mean value (for cl.thickness, cell.size, cell.shape, marg.adhesion, epith.c.size, Bl.cromatin, Normal.nucleoli and Mitoses.

```
13
14  mean(data$Cl.thickness, na.rm = TRUE)
15  mean(data$Cell.size, na.rm = TRUE)
16  mean(data$Cell.shape, na.rm = TRUE)
17  mean(data$Marg.adhesion, na.rm = TRUE)
18  mean(data$Epith.c.size, na.rm = TRUE)
19  mean(data$Bl.cromatin, na.rm = TRUE)
20  mean(data$Normal.nucleoli, na.rm = TRUE)
21  mean(data$Mitoses, na.rm = TRUE)
22
23
22:1  (Top Level)
```

```
Console  Terminal  Background Jobs
R 4.3.2 · C:/Users/user/Desktop/MSc Computer Science/CETM72/Wisconsin Data/
> mean(data$Cl.thickness, na.rm = TRUE)
[1] 4.41774
> mean(data$Cell.size, na.rm = TRUE)
[1] 3.134478
> mean(data$Cell.shape, na.rm = TRUE)
[1] 3.207439
> mean(data$Marg.adhesion, na.rm = TRUE)
[1] 2.806867
> mean(data$Epith.c.size, na.rm = TRUE)
[1] 3.216023
> mean(data$Bl.cromatin, na.rm = TRUE)
[1] 3.437768
> mean(data$Normal.nucleoli, na.rm = TRUE)
[1] 2.866953
> mean(data$Mitoses, na.rm = TRUE)
[1] 1.589413
```

I then replaced the data in the 'class' column to numerical variables. Malignant = 1 and Benign = 0. This is because the numbers 0 and 1 and most commonly used for classification purposes, and I will need this data for binary classification.

```
24  unique(df$Class)
25  df$Class <- ifelse(df$Class == 'malignant', 1, 0)
26  unique(df$Class)
27
```

Once replaced, I used the code unique(df$class) to check.

```
> unique(df$Class)
[1] 0 1
>
```

These pre-processing steps contribute to a cleaner dataset, reducing biases and enhancing the robustness of subsequent analyses.

5. Practical: R Programming content

Logistic Regression:

Step 1: Load the Wisconsin dataset and check the structure:

```
27
28  df <- read.csv("wisconsin.csv")
29  str(df)
30
31
```

Step 2: Ensure Malignant is replaced with 1 and Benign is replaced with 0 and check the updated data.

```
30  df$Class <- ifelse(df$Class == 'malignant', 1, 0)
31  unique(df$Class)
32  head(df)
```

Step 3: I split the data into training and testing sets:

```
33 set.seed(123)  # For reproducibility
34 train_indices <- sample(1:nrow(df), 0.7 * nrow(df))
35 train_data <- df[train_indices, ]
36 test_data <- df[-train_indices, ]
```

Step 4: I used to the logistic regression model and made predictions on a test set:

```
38 model <- glm(Class ~ ., data = train_data, family = "binomial")
39 predictions <- predict(model, newdata = test_data, type = "response")
```

Step 5: Here, I evaluated the model. I created a confusion matrix (results in section 6) and calculated accuracy.

```
41 predicted_classes <- ifelse(predictions > 0.5, 1, 0)
42 conf_matrix <- table(predicted_classes, test_data$Class)
43 print(conf_matrix)
44 accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
45 print(paste("Accuracy:", accuracy))
```

The accuracy obtained was 0.961904761904762.

```
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.961904761904762"
>
```

Decision Tree model:

Initially, I was going to use the rpart.plot package for this but had issues with installation. I tried an alternative approach using the 'partykit' package, which enables to plot decision trees.

Step 1: After installing the package, I split the data into training and testing data:

```
76 set.seed(123)
77 sample_index <- sample(1:nrow(df), 0.7 * nrow(df))
78 train_data <- df[sample_index, ]
79 test_data <- df[-sample_index, ]
```

Step 2: I used this code to train the decision tree model:

```
80 tree_model <- rpart(Class ~ ., data = train_data, method = "class")
```

Step 3: I converted the decision tree to a 'party' object:

```
81 tree_party <- as.party(tree_model)
```

Step 4: I plot the decision tree: (the plot can be seen in section 6).

```
82 plot(tree_party)
```

I also wanted to calculate the accuracy of the Decision tree model.

Step 1: I converted predicted and reference variables to factors with the same level:

```
85 predictions_tree <- as.factor(predictions_tree)
86 test_data$Class <- as.factor(test_data$Class)
```

Step 2: I created a confusion matrix:

```
conf_matrix_tree <- confusionMatrix(data = predictions_tree, reference = test_data$Class)
```

Step 3: I used code to extract accuracy from the confusion matrix. I then print the confusion matrix and accuracy.

```
88  accuracy_tree <- conf_matrix_tree$overall["Accuracy"]
89  print(conf_matrix_tree)
90  cat("Accuracy:", accuracy_tree, "\n")
```

Accuracy: 0.9428571

<u>Support Vector Machine:</u>

Step 1: After the preprocessing, I set the seed and then I split the data into training and testing datasets.

```
install.packages("e1071")
library(e1071)
df$Class <- as.factor(df$Class)
set.seed(123)
index <- sample(1:nrow(df), 0.7 * nrow(df))
train_data <- df[index, ]
test_data <- df[-index, ]
```

Step 2: I built the SVM model:

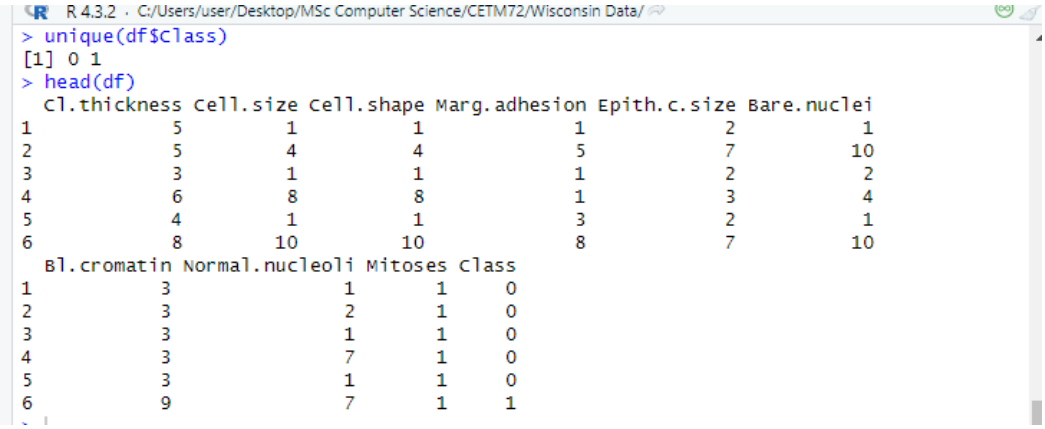
```
125  svm_model <- svm(Class ~ ., data = train_data)
```

Step 3: Checked accuracy, and confusion matrix. Printed accuracy and confusion matrix.

```
190  svm_predictions <- predict(svm_model, newdata = test_data)
191  conf_matrix <- confusionMatrix(svm_predictions, test_data$Class)
192  svm_accuracy <- conf_matrix$overall["Accuracy"]
193  print(paste("SVM Accuracy:", svm_accuracy))
194  print("Confusion Matrix:")
195  print(conf_matrix)
```

6. Practical: Display of data/results:

This was the output in the console when I checked the data updated (step 2) in the data frame.



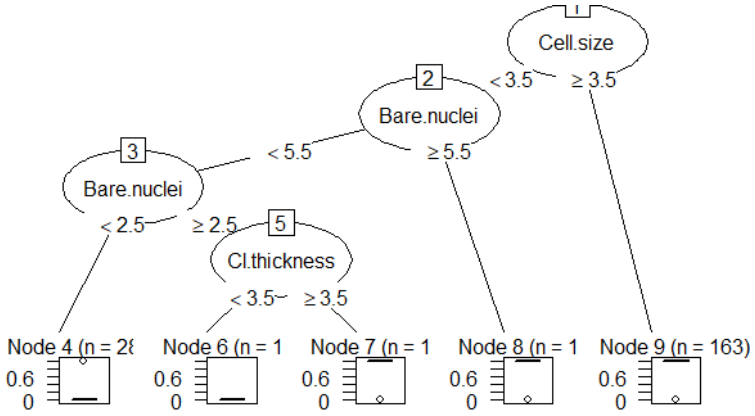
This was the confusion matrix for logistic regression:

```
>   conf_matrix <- table(predicted_classes, test_data$Class)
>   print(conf_matrix)

predicted_classes   0   1
                0 144   2
                1   6  58
> |
```

This was the plot from the Decision Tree Model:

The root node represents the variables and the threshold for splitting the data into groups. Each internal node had branches that lead to the child nodes, showing the different paths it can take (Enes, A. and Karakoyun, M. 2023).



This is the confusion matrix and accuracy from the Decision Tree Model.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 140   2
         1  10  58

               Accuracy : 0.9429
                 95% CI : (0.9023, 0.9701)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8654

 Mcnemar's Test P-Value : 0.04331

            Sensitivity : 0.9333
            Specificity : 0.9667
         Pos Pred Value : 0.9859
         Neg Pred Value : 0.8529
             Prevalence : 0.7143
         Detection Rate : 0.6667
   Detection Prevalence : 0.6762
      Balanced Accuracy : 0.9500

       'Positive' Class : 0

>   cat("Accuracy:", accuracy_tree, "\n")
Accuracy: 0.9428571
>   print(conf_matrix_tree)
```

This was the confusion matrix and accuracy for SVM:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 144   0
         1   6  60

               Accuracy : 0.9714
                 95% CI : (0.9389, 0.9894)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : < 2e-16
```

Figure 3: A comparative line graph between the 3 classifiers.

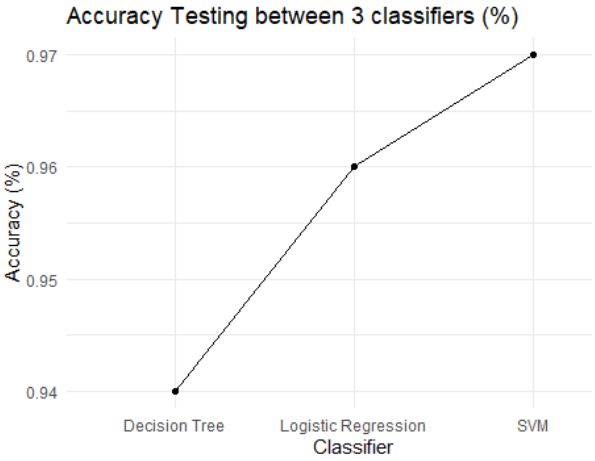


Table 1: Accuracy percentage for Wisconsin Data

Algorithm	Accuracy Testing Set %
Logistic Regression	96.2%
Decision Tree	94.3%
SVM	97.1

From the results, we can see the classifiers have varying accuracies, but SVM has the highest accuracy testing set (97.1%).

Table 2: Table showing the Confusion matrix between the 3 classifiers:

	Benign (0)	Malignant (1)	
Logistic Regression	144 6	2 58	Benign (0) Malignant (1)
Decision Tree	140 10	2 58	Benign (0) Malignant (1)
SVM	144 6	0 60	Benign (0) Malignant (1)

Since confusion matrices are a useful way to assess the classifier, each row in Table 2 represents the rates in an actual class while each column displays the predictions.

7. Conclusions

Three algorithms were applied to the Wisconsin dataset: Logistic Regression, Decision Tree and SVM. After an accurate comparison between the models (figure 3 in results), the Support Vector Machine

achieved a higher efficiency of 97.1%, outperforming the other two algorithms. In conclusion, Support Vector Machine has demonstrated its efficiency in breast cancer prediction and diagnosis and achieves the best performance in terms of accuracy. Due to all the results obtained being related just to the Wisconsin database, it can be considered as a limitation. Therefore, it's necessary to reflect for future works to apply these same algorithms and methods on other databases to confirm the results obtained via this database (Rashed, A et al 2023).

8. **Source code listing:**

Figure 1: Wisconsin Dataset Classes

```
#library ggplot

Library(ggplot)

# Load the Wisconsin dataset

df <- read.csv("wisconsin.csv")

# Assuming the dataset has a column 'Class' containing 'M' for malignant and 'B' for benign

Class_counts <- table(df$Class)

# Create a bar chart

ggplot(data = NULL, aes(x = names(Class_counts), y = Class_counts)) +

  geom_bar(stat = "identity", fill = c('red', 'green')) +

  labs(title = "Class Distribution in Wisconsin Breast Cancer Dataset",

       x = "Class",

       y = "Count")
```

Data loading and any preliminary checks:

```
# working directory

 setwd("C:/Users/user/Desktop/MSc Computer Science/CETM72/Wisconsin Data")

#Reading data

df <- read.csv("wisconsin.csv")

summary(df)
```

Data Pre-processing:

```
# find the location of missing values

which(is.na(data))

# find the count of missing values

sum(is.na(data))

#Calculate mean for all numerical columns

mean(data$Bare.nuclei, na.rm = TRUE)

mean(data$Cl.thickness, na.rm = TRUE)
```

```
mean(data$Cell.size, na.rm = TRUE)
mean(data$Cell.shape, na.rm = TRUE)
mean(data$Marg.adhesion, na.rm = TRUE)
mean(data$Epith.c.size, na.rm = TRUE)
mean(data$Bl.cromatin, na.rm = TRUE)
mean(data$Normal.nucleoli, na.rm = TRUE)
mean(data$Mitoses, na.rm = TRUE)
#Replacement of Malignant with (1) and Benign with (0):
df$Class <- ifelse(df$Class == 'malignant', 1, 0)
#Check for unique values in 'Class' column
unique(df$Class)
```

Logistic regression:

```
#load libraries
library(caret)
library(glmnet)
# Split data into training and testing sets and set seed for reproducibility
set.seed(123)
train_indices <- sample(1:nrow(df) , 0.7 *nrow (df))
train_data <- df[train_indices]
test_data <- df[train_indices]
# Train logistic regression model
model <- glm(Class ~ ., data = train_data, family = "binomial")
# Make predictions on testing data
predictions <- predict(model, newdata = test_data, type = "response")
# Evaluate model performance, print accuracy and confusion matrix.
predicted_classes <- ifelse(predictions  > 0.5, 1, 0)
conf_matrix <- table(predicted_classes, reference = test_data$Class)
print(conf_matrix)
accuracy <- sum(diag(conf_matric)) / sum(conf_matrix)
print (paste("Accuracy:" , accuracy))
```

Decision Tree Model:

```
# Load required libraries
library(caret)
```

```
library(rpart)
library(partykit)
# Split data into training and testing sets
set.seed(123)  # Set seed for reproducibility
split_index <- createDataPartition(wisconsin_data$diagnosis, p = 0.8, list = FALSE)
train_data <- wisconsin_data[split_index, ]
test_data <- wisconsin_data[-split_index, ]
# Train decision tree model
tree_model <- rpart(Class ~ ., data = train_data, method = "class")
# Convert Decision Tree to partykit
Tree_party <- as.party(tree_model)
# Plot decision tree model
Plot(tree_party)
# Make predictions on testing data
predictions_tree <- predict(tree_model, newdata = test_data, type = "class")
#Calculate accuracy of decision tree model
Predictions_tree <- as.factor(predicitons_tree)
Test_data$Class <- as.factor(test_data$Class)
#Create confusion matrix
Conf_matrix_tree <- confusionMatrix(data = predicitions_tree, reference = test_data$Class)
# Print metrics
Accuracy_tree <- conf_matrix_tree$overall[“Accuracy”]
Print(conf_matrix_tree)
Cat(“Accuracy:”, accuracy_tree, “\n”)
```

Support Vector Machine (SVM):

```
# Load required libraries
library(caret)
library(e1071)
# Set seed and split data into training and testing sets
Df$Class <- as.factor(df$Class)
set.seed(123)
index <- sample(1:nrow(df), 0.7 * nrow(df))
train_data <- df[index, ]
```

```
test_data <- df[-index, ]
# Train SVM model:
svm_model <- svm(Class ~ . , data = train_data)
# Check predictions, accuracy, and confusion matrix
svm_predictions <- predict(svm_model, newdata = test_data)
conf_matrix <- confusionMatrix(svm_predictions, test_data$Class)
svm_accuracy <- conf_matrix$overall[“Accuracy”]
#Print Accuracy and confusion matrix
print(paste(“SVM Accuracy:” , svm_accuracy))
print(“confusion Matrix:”)
print(conf_matrix)
```

Figure 3: A comparative line graph between the 3 classifiers

```
# Load library
library(ggplot2)
# Add in classifier names and accuracy values
classifiers <- c("Logistic Regression", "Decision Tree", "SVM")
accuracy <- c(0.96, 0.94, 0.97)
# Creating a data frame
data <- data.frame(Classifier = classifiers, Accuracy = accuracy)
# Plotting the line graph
ggplot(data, aes(x = Classifier, y = Accuracy, group = 1)) +
  geom_line() +
  geom_point() +
  labs(title = "Accuracy Testing between 3 classifiers (%)",
       x = "Classifier",
       y = "Accuracy (%)") +
  theme_minimal()
```

9. References:

Romm, A. (2010) *Chapter 10 - Breast cancer*, *Breast Cancer - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/nursing-and-health-professions/breast-cancer (Accessed: 25 February 2024).

Asri, H. *et al.* (2016) Science Direct: *Using machine learning algorithms for breast cancer risk prediction and diagnosis*. Available at: https://www.sciencedirect.com/science/article/pii/S1877050916302575 (Accessed: 25 February 2024).

Alshayeji, M. *et al.* (2021) *Computer-aided detection of breast cancer on the wisconsin dataset: An Artificial Neural Networks approach*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S1746809421007382 (Accessed: 26 February 2024).

Nasser, M. and Yusuf, U.K. (2023) *Deep Learning Based Methods for Breast Cancer Diagnosis: A Systematic Review and Future Direction*, *Diagnostics (Basel)*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9818155/ (Accessed: 26 February 2024).

Enes, A. and Karakoyun, M. (2023) *Breast Cancer Detection Using Machine Learning Algorithms*, *2nd International Conference on Scientific and Academic Research*. Available at: https://as-proceeding.com/index.php/ijanser/article/view/401 (Accessed: 27 February 2024).

Rashed, A., Elmorsy , A. and Atwa, A. (2023) *Comparative evaluation of automated machine learning techniques for breast cancer diagnosis.*, *Biomedical Signal Processing and Control*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S1746809423004494 (Accessed: 29 February 2024).