CLASSIFICATION OF SONAR DATASET USING DECISION TREE CLASSIFIER IN R

Aim:

- In this case study, we will be able to build decision tree models using the tree and rpart libraries in R. In addition, we will evaluate the model's performance and learn how to save the trained model on the local system.
- A dataset called "Sonar" has been taken. This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network.
- The data reports the patterns obtained by bouncing sonar signals at various angles and under various conditions.
- There are 208 patterns in all, 111 obtained by bouncing sonar signals off a metal cylinder and 97 obtained by bouncing signals off rocks.
- Each pattern is a set of 60 numbers (variables) taking values between 0 and 1.
- These data have been taken from the UCI Repository Of Machine Learning Databases at

  ftp://ftp.ics.uci.edu/pub/machine-learning-databases

  http://www.ics.uci.edu/~mlearn/MLRepository.html

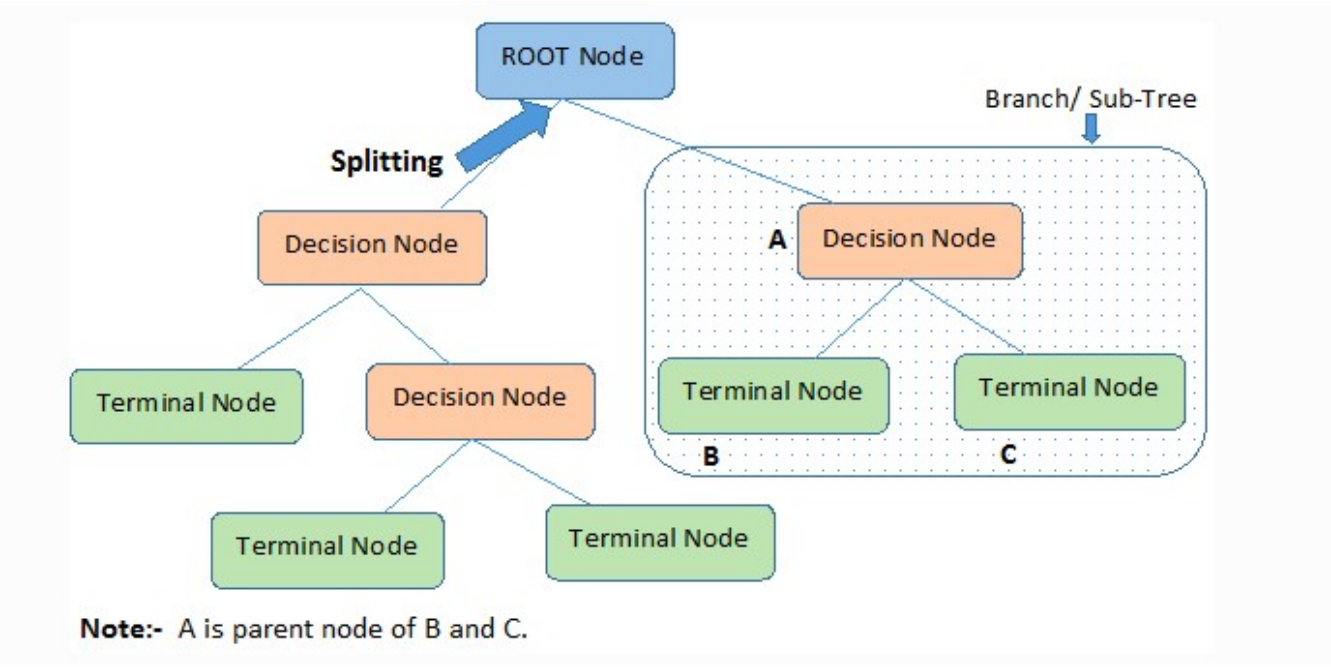
  and were converted to R format by Evgenia Dimitriadou.

Objectives:

1. Understand the concept of the decision tree algorithm
2. Build decision tree models
3. Evaluate the performance of the model

Procedure:

Decision Tree Representation:



Importing the libraries:

```
19 ```{r}
20 ## 2.1: Import the required packages
21
22 library(tree)
23 library(caret)
24 library(rpart)
25 library(rpart.plot)
26 library(mlbench)
27
28 ```
```

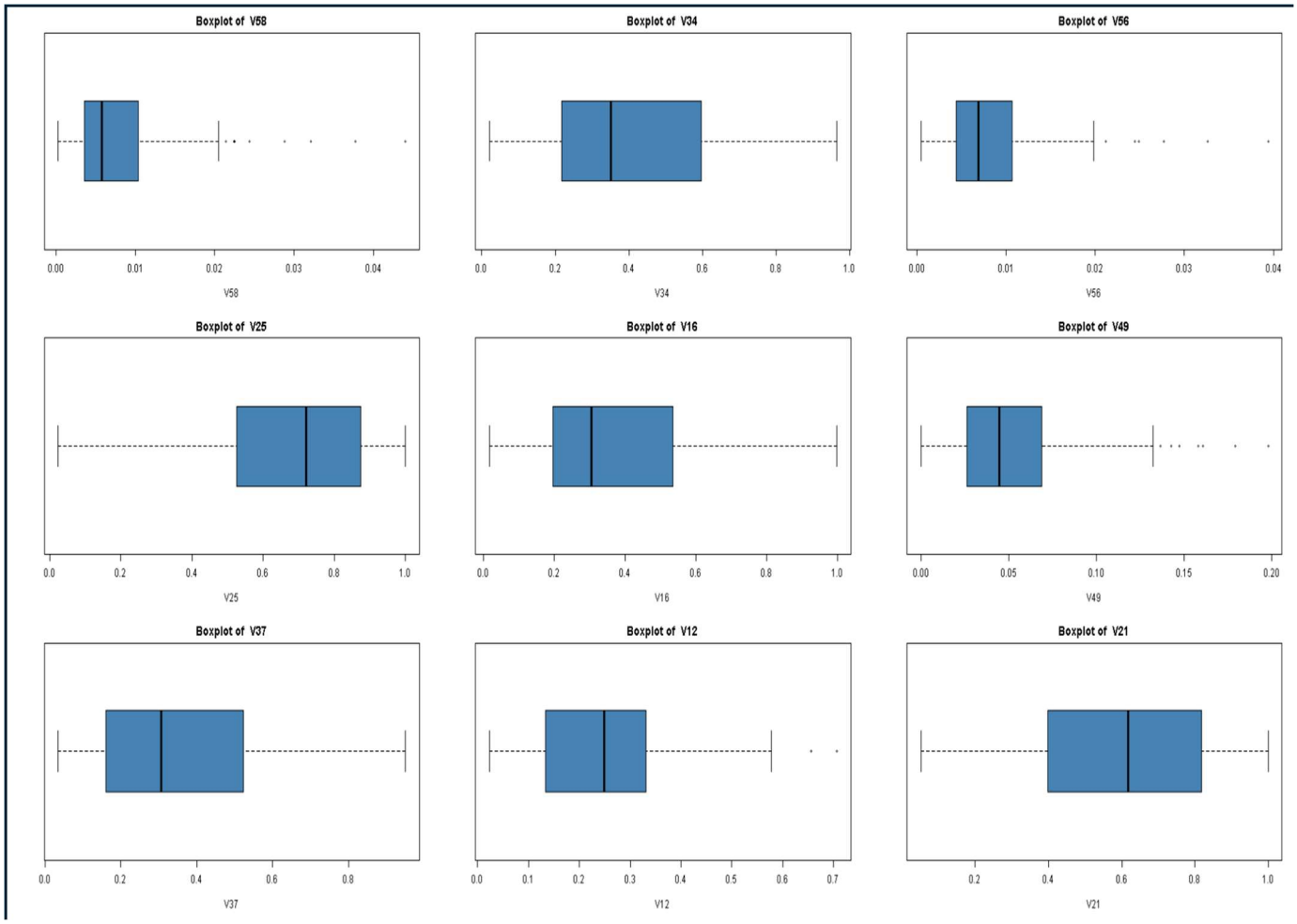
Importing and Exploring the dataset:

```r
## 3.1: Load the dataset in R
data("Sonar")

## 3.2: View the dataset
View(Sonar)

## 3.3: Obtain the frequency of the target variable
table(Sonar$Class)

## 3.4: Check for missing values
sum(is.na(Sonar))

## OR

anyNA(Sonar)

## 3.5: Summarise the data set
summary(Sonar)
## Exercise 3.1: Check the number of rows and columns
nrow(Sonar)
ncol(Sonar)

## Set seed to 11
set.seed(11)

## 3.6: Randomly choose nine features from the data set
columns <- sample(x = 1:60, size = 9)
pre_var <- Sonar[, columns]
par(mfrow = c(3,3))   # This produces a 3 by 3 plot of the box plots

## 3.7: Show box plots of the predictor variables
for (i in 1:ncol(pre_var)){
  boxplot(pre_var[, i], xlab = names(pre_var[i]),
        main = paste("Boxplot of ",names(pre_var[i])),
        horizontal = TRUE, col = "steelblue")
}
```

“Sonar” dataset:

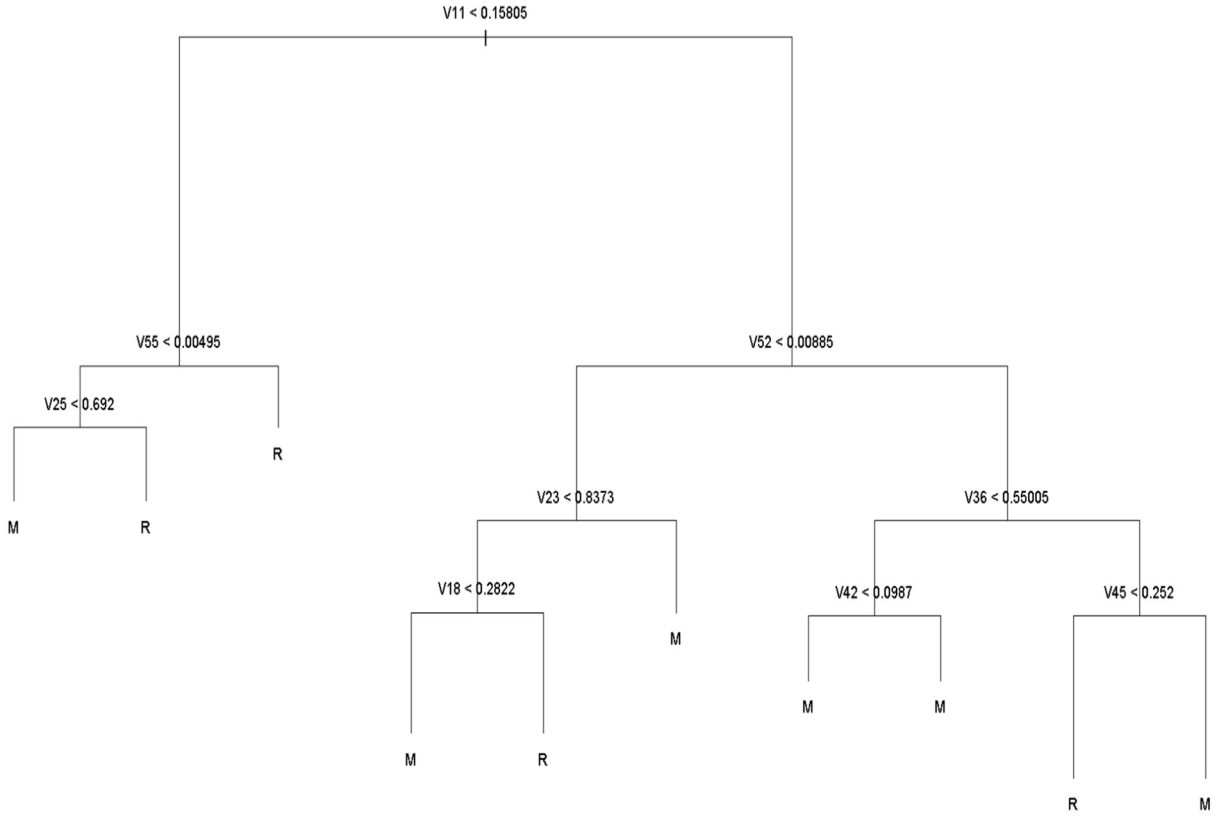
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24
1	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	0.1609	0.1582	0.2238	0.0645	0.0660	0.2273	0.3100	0.2999	0.5078	0.4797	0.5783	0.5071	0.4328	0.5
2	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	0.4918	0.6552	0.6919	0.7797	0.7464	0.9444	1.0000	0.8874	0.8024	0.7818	0.5212	0.4052	0.3957	0.3
3	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	0.6333	0.7060	0.5544	0.5320	0.6479	0.6931	0.6759	0.7551	0.8929	0.8619	0.7974	0.6737	0.4293	0.3
4	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	0.0881	0.1992	0.0184	0.2261	0.1729	0.2131	0.0693	0.2281	0.4060	0.3973	0.2741	0.3690	0.5556	0.4
5	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	0.4152	0.3952	0.4256	0.4135	0.4528	0.5326	0.7306	0.6193	0.2032	0.4636	0.4148	0.4292	0.5730	0.5
6	0.0286	0.0453	0.0277	0.0174	0.0384	0.0990	0.1201	0.1833	0.2105	0.3039	0.2988	0.4250	0.6343	0.8198	1.0000	0.9988	0.9508	0.9025	0.7234	0.5122	0.2074	0.3985	0.5890	0.2
7	0.0317	0.0956	0.1321	0.1408	0.1674	0.1710	0.0731	0.1401	0.2083	0.3513	0.1786	0.0658	0.0513	0.3752	0.5419	0.5440	0.5150	0.4262	0.2024	0.4233	0.7723	0.9735	0.9390	0.5
8	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	0.1465	0.2838	0.2802	0.3086	0.2657	0.3801	0.5626	0.4376	0.2617	0.1199	0.6676	0.9402	0.7832	0.5352	0.6809	0.9
9	0.0223	0.0375	0.0484	0.0475	0.0647	0.0591	0.0753	0.0098	0.0684	0.1487	0.1156	0.1654	0.3833	0.3598	0.1713	0.1136	0.0349	0.3796	0.7401	0.9925	0.9802	0.8890	0.6712	0.4
10	0.0164	0.0173	0.0347	0.0070	0.0187	0.0671	0.1056	0.0697	0.0962	0.0251	0.0801	0.1056	0.1266	0.0890	0.0198	0.1133	0.2826	0.3234	0.3238	0.4333	0.6068	0.7652	0.9203	0.9
11	0.0039	0.0063	0.0152	0.0336	0.0310	0.0284	0.0396	0.0272	0.0323	0.0452	0.0492	0.0996	0.1424	0.1194	0.0628	0.0907	0.1177	0.1429	0.1223	0.1104	0.1847	0.3715	0.4382	0.5
12	0.0123	0.0309	0.0169	0.0313	0.0358	0.0102	0.0182	0.0579	0.1122	0.0835	0.0548	0.0847	0.2026	0.2557	0.1870	0.2032	0.1463	0.2849	0.5824	0.7728	0.7852	0.8515	0.5312	0.3
13	0.0079	0.0086	0.0055	0.0250	0.0344	0.0546	0.0528	0.0958	0.1009	0.1240	0.1097	0.1215	0.1874	0.3383	0.3227	0.2723	0.3943	0.6432	0.7271	0.8673	0.9674	0.9847	0.9480	0.8
14	0.0090	0.0062	0.0253	0.0489	0.1197	0.1589	0.1392	0.0987	0.0955	0.1895	0.1896	0.2547	0.4073	0.2988	0.2901	0.5326	0.4022	0.1571	0.3024	0.3907	0.3542	0.4438	0.6414	0.4
15	0.0124	0.0433	0.0604	0.0449	0.0597	0.0355	0.0531	0.0343	0.1052	0.2120	0.1640	0.1901	0.3026	0.2019	0.0592	0.2390	0.3657	0.3809	0.5929	0.6299	0.5801	0.4574	0.4449	0.3
16	0.0298	0.0615	0.0650	0.0921	0.1615	0.2294	0.2176	0.2033	0.1459	0.0852	0.2476	0.3645	0.2777	0.2826	0.3237	0.4335	0.5638	0.4555	0.4348	0.6433	0.3932	0.1989	0.3540	0.9
17	0.0352	0.0116	0.0191	0.0469	0.0737	0.1185	0.1683	0.1541	0.1466	0.2912	0.2328	0.2237	0.2470	0.1560	0.3491	0.3308	0.2299	0.2203	0.2493	0.4128	0.3158	0.6191	0.5854	0.3
18	0.0192	0.0607	0.0378	0.0774	0.1388	0.0809	0.0568	0.0219	0.1037	0.1186	0.1237	0.1601	0.3520	0.4479	0.3769	0.5761	0.6426	0.6790	0.7157	0.5466	0.5399	0.6362	0.7849	0.7
19	0.0270	0.0092	0.0145	0.0278	0.0412	0.0757	0.1026	0.1138	0.0794	0.1520	0.1675	0.1370	0.1361	0.1345	0.2144	0.5354	0.6830	0.5600	0.3093	0.3226	0.4430	0.5573	0.5782	0.6
20	0.0126	0.0149	0.0641	0.1732	0.2565	0.2559	0.2947	0.4110	0.4983	0.5920	0.5832	0.5419	0.5472	0.5314	0.4981	0.6985	0.8292	0.7839	0.8215	0.9363	1.0000	0.9224	0.7839	0.5
21	0.0473	0.0509	0.0819	0.1252	0.1783	0.3070	0.3008	0.2362	0.3830	0.3759	0.3021	0.2909	0.2301	0.1411	0.1582	0.2430	0.4474	0.5964	0.6744	0.7969	0.8319	0.7813	0.8626	0.7
22	0.0664	0.0575	0.0842	0.0372	0.0458	0.0771	0.0771	0.1130	0.2353	0.1838	0.2869	0.4129	0.3647	0.1984	0.2840	0.4039	0.5837	0.6792	0.6086	0.4858	0.3246	0.2013	0.2082	0.1
23	0.0099	0.0484	0.0299	0.0297	0.0652	0.1077	0.2363	0.2385	0.0075	0.1882	0.1456	0.1892	0.3176	0.1340	0.2169	0.2458	0.2589	0.2786	0.2298	0.0656	0.1441	0.1179	0.1668	0.1
24	0.0115	0.0150	0.0136	0.0076	0.0211	0.1058	0.1023	0.0440	0.0931	0.0734	0.0740	0.0622	0.1055	0.1183	0.1721	0.2584	0.3232	0.3817	0.4243	0.4217	0.4449	0.4075	0.3306	0.4

Box Plot of predictor variables:



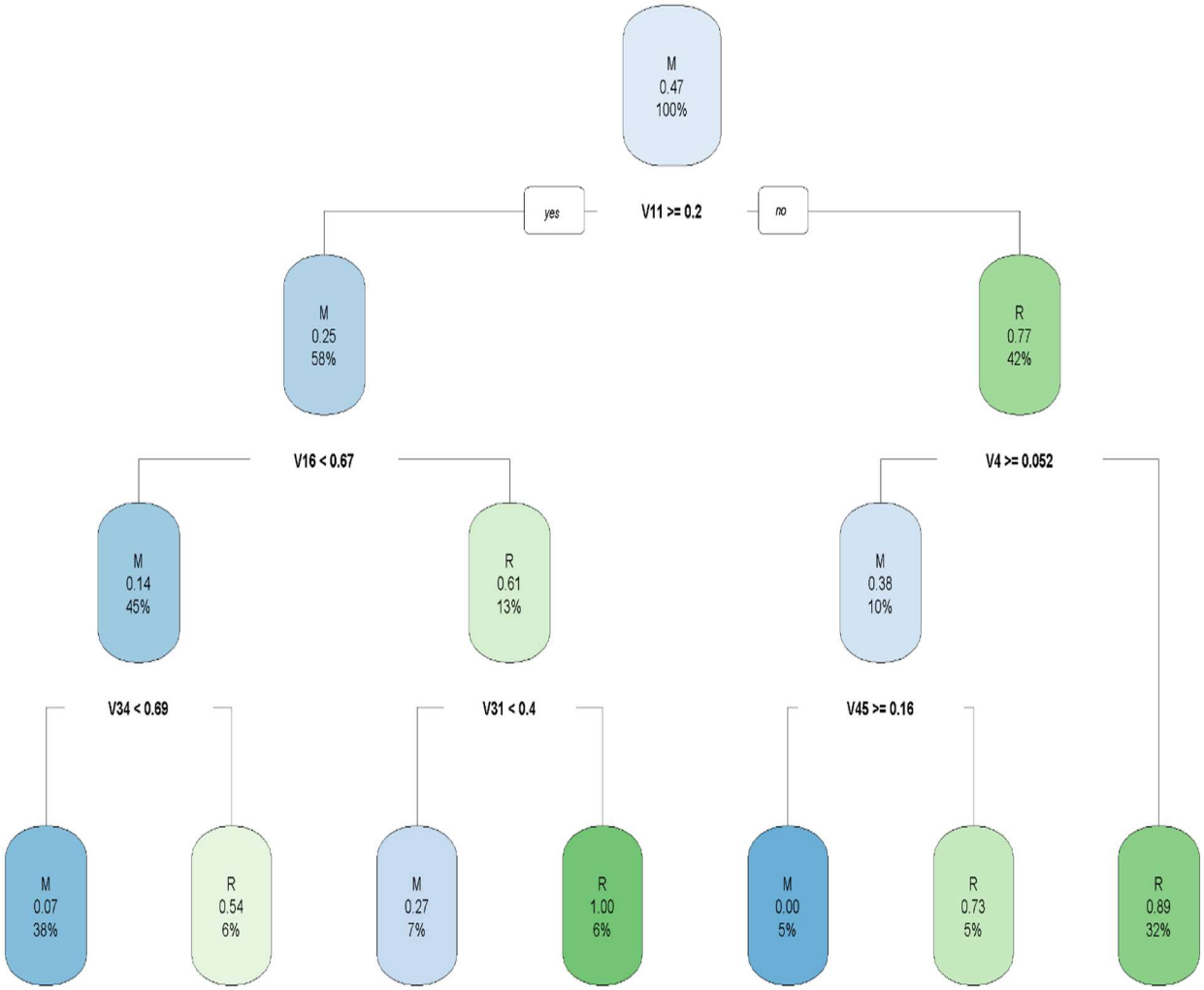
Training and Visualizing the decision tree model:

```
100
101 ## 5.1: Train the model using the tree function
102 model ← tree(
103   formula = Class ~ .,
104   data = train_set
105 )
106
107 ## Exercise 5.1: Check the summary of the model
108 summary(model)
109
110 ## 5.2: Visualize the decision tree model
111 plot(model)
112 text(model)
```



Visualizing the new decision tree model using rpart:

```
115 ## 5.3: Fit the model using the rpart function
116 model1 ← rpart(Class ~ ., data = Sonar,
117               method = "class")
118
119 ## Exercise 5.2: Print the model and the model summary
120 model1
121 summary(model1)
122
123 ## 5.5: Visualize the new decision tree model
124 rpart.plot(model1)
125
```



Evaluating Model Performance:

```
> prediction ← predict(model1, newdata = test_set, type = "class")
> table(x = prediction, y = test_set$Class)
   y
x    M  R
  M 29  3
  R  4 26
> confusionMatrix(data = prediction,
+                 reference = test_set$Class)
Confusion Matrix and Statistics

          Reference
Prediction  M  R
         M 29  3
         R  4 26

               Accuracy : 0.8871
                 95% CI : (0.7811, 0.9534)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 2.418e-09

                  Kappa : 0.7737

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8788
            Specificity : 0.8966
         Pos Pred Value : 0.9062
         Neg Pred Value : 0.8667
             Prevalence : 0.5323
         Detection Rate : 0.4677
   Detection Prevalence : 0.5161
      Balanced Accuracy : 0.8877

       'Positive' Class : M

> F1 ← 2* (0.8788 * 0.8788) / (0.8788 + 0.8788)
> F1
[1] 0.8788
> error_rate ← round(mean(prediction ≠ test_set$Class), 2)
> error_rate
[1] 0.11
>
```

Conclusion:

So, this is how we build, train and visualize the “Decision Tree Classifier”in R programming language.