I conducted a single iteration of the CART-based algorithm on a wine quality dataset (Appendix A), aiming to predict wine quality based on the variables outlined in Figure 3. I have used RStudio (Appendix B). The dataset contains 1,143 observations and 12 variables (Figure 1). I initiated the process by exploring the data (Figure 2), ensuring there were no missing values, and performing feature selection by eliminating unnecessary columns and scaling, as part of my data-pre-processing. Following the steps outlined in Figure 1, during "Phase 1", I partitioned the data into training and test sets. Creating a root node 'N' occurred when I applied the 'Rpart' function to build a decision tree (DT). The function effectively handled checks for tuple homogeneity and emptiness of the attribute list. In "Phase 2", the code at the top of Figure 4 employed a method to select the best attribute for splitting, determining the optimal split criterion. The node was labelled with this criterion, addressing steps vi and vii. 'Rpart' internally handled the update of the attribute list by iteratively selecting the most suitable attribute for splitting each node. During "Phase 3", 'Rpart' divides the tree based on the chosen criterion, creating subtrees for each partition. For each child node, 'Rpart' autonomously decides whether to label it with a class or continue splitting. Figure 5 shows the fully-grown tree. Although not explicitly mentioned, I pruned the tree to prevent overfitting a common practice in DT modelling (Figure 6)

```
> wine <- wine[, !(names(wine) %in% c("Id"))]
> summary(wine)
 fixed.acidity    volatile.acidity  citric.acid
 Min.   : 4.600   Min.   :0.1200   Min.   :0.0000
 1st Qu.: 7.100   1st Qu.:0.3925   1st Qu.:0.0900
 Median : 7.900   Median :0.5200   Median :0.2500
 Mean   : 8.311   Mean   :0.5313   Mean   :0.2684
 3rd Qu.: 9.100   3rd Qu.:0.6400   3rd Qu.:0.4200
 Max.   :15.900   Max.   :1.5800   Max.   :1.0000
 residual.sugar     chlorides       free.sulfur.dioxide
 Min.   : 0.900   Min.   :0.01200   Min.   : 1.00
 1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00
 Median : 2.200   Median :0.07900   Median :13.00
 Mean   : 2.532   Mean   :0.08693   Mean   :15.62
 3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00
 Max.   :15.500   Max.   :0.61100   Max.   :68.00
 total.sulfur.dioxide    density             pH
 Min.   :  6.00       Min.   :0.9901   Min.   :2.740
 1st Qu.: 21.00       1st Qu.:0.9956   1st Qu.:3.205
 Median : 37.00       Median :0.9967   Median :3.310
 Mean   : 45.91       Mean   :0.9967   Mean   :3.311
 3rd Qu.: 61.00       3rd Qu.:0.9978   3rd Qu.:3.400
 Max.   :289.00       Max.   :1.0037   Max.   :4.010
   sulphates         alcohol         quality
 Min.   :0.3300   Min.   : 8.40   Min.   :3.000
 1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
 Median :0.6200   Median :10.20   Median :6.000
 Mean   :0.6577   Mean   :10.44   Mean   :5.657
 3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
 Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

Figure 2: Summary statistics

```
> str(wine)
tibble [1,143 × 12] (S3: tbl_df/tbl/data.frame)
 $ fixed.acidity       : num [1:1143] 7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 6.7 ...
 $ volatile.acidity    : num [1:1143] 0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.58 ...
 $ citric.acid         : num [1:1143] 0 0 0.04 0.56 0 0 0.06 0 0.02 0.08 ...
 $ residual.sugar      : num [1:1143] 1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 1.8 ...
 $ chlorides           : num [1:1143] 0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.097 ...
 $ free.sulfur.dioxide : num [1:1143] 11 25 15 17 11 13 15 15 9 15 ...
 $ total.sulfur.dioxide: num [1:1143] 34 67 54 60 34 40 59 21 18 65 ...
 $ density             : num [1:1143] 0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num [1:1143] 3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.28 ...
 $ sulphates           : num [1:1143] 0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.54 ...
 $ alcohol             : num [1:1143] 9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 9.2 ...
 $ quality             : num [1:1143] 5 5 5 6 5 5 5 7 7 5 ...
```

Figure 3: Dataset structure

```
> fit <- rpart(quality ~ ., data=train_data, method="class")
>
> printcp(fit)

Classification tree:
rpart(formula = quality ~ ., data = train_data, method = "class")

Variables actually used in tree construction:
[1] alcohol          chlorides        sulphates
[4] volatile.acidity

Root node error: 524/916 = 0.57205

n= 916

        CP nsplit rel error  xerror     xstd
1 0.230916      0   1.00000 1.00000 0.028578
2 0.034351      1   0.76908 0.80344 0.028785
3 0.026718      3   0.70038 0.79771 0.028769
4 0.015267      4   0.67366 0.76336 0.028647
5 0.010000      5   0.65840 0.74237 0.028550
```

Figure 4: DT results

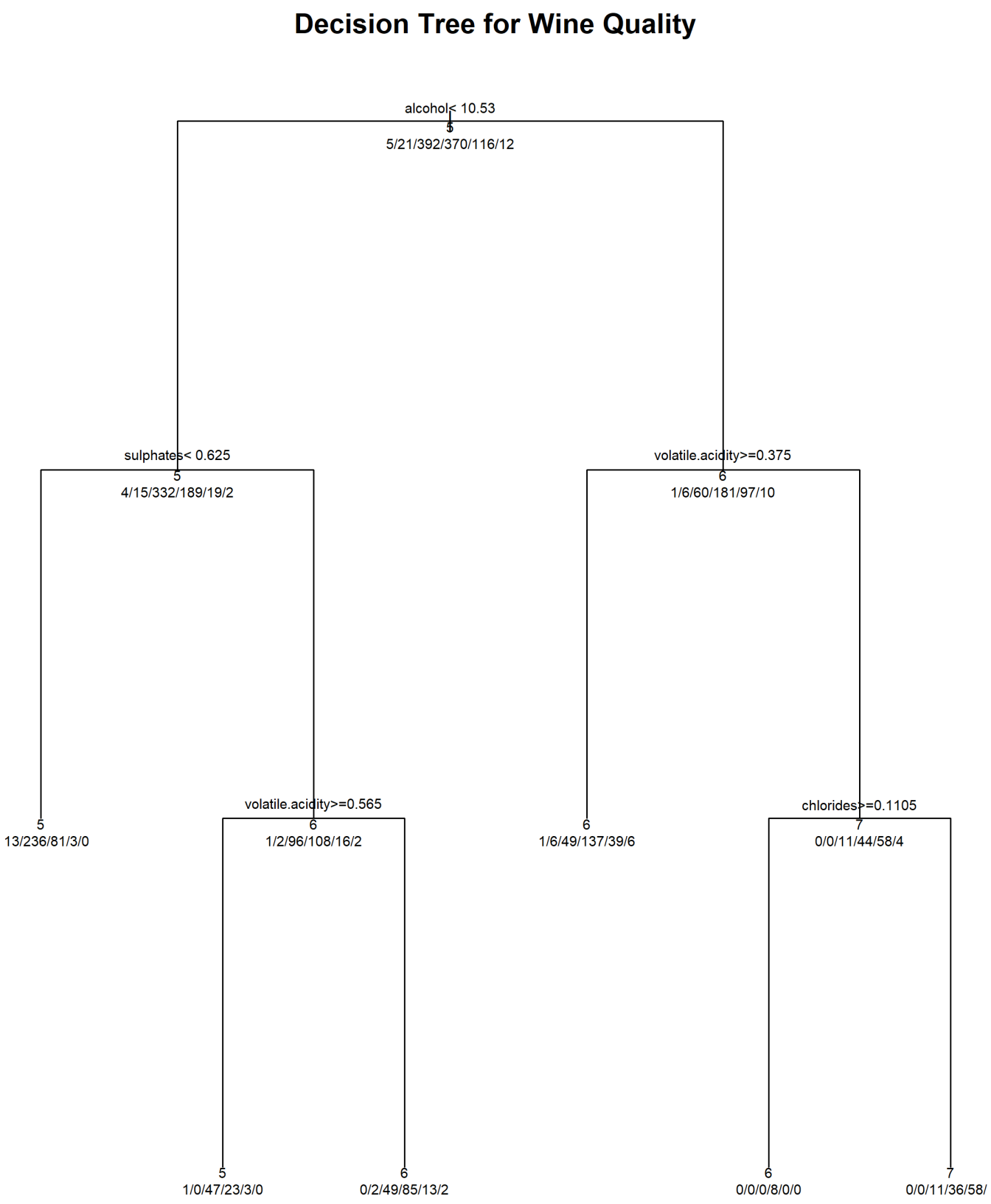

Figure 5: fully-grown DT

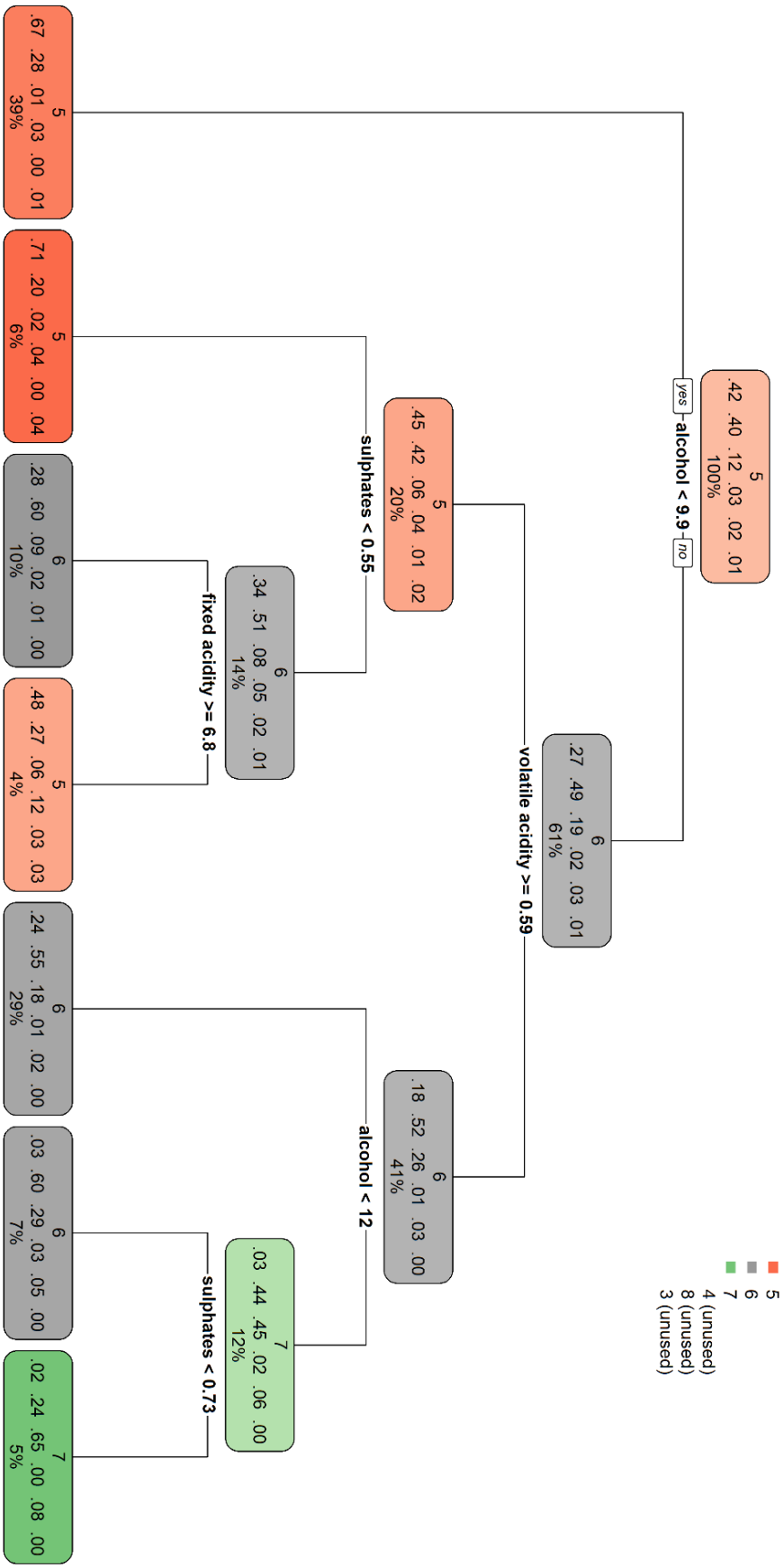

Figure 6: Pruned DT

The evaluation of the DT's performance on test data is conducted using a confusion matrix (Figure 7), as specified in the published paper. The model shows varying effectiveness across classes, excelling in class 5 (74.73% sensitivity), but facing challenges in class 3 and 4. Overall, the model's accuracy is 55.07% with a Kappa coefficient of 28.78%, indicating moderate level agreement between predicted and actual classifications. Specific metrics highlight improvement areas, particularly in classes 3,4 and 8. The model's balanced accuracy, considering both sensitivity and specificity, is 55.25%. Some precision values are unavailable due to zero positive predictions, indicating the need for further analysis; and potential model refinement, especially for classes with low sensitivity or precision, to enhance overall predictive performance.

```
> print(confusion_matrix)
Confusion Matrix and Statistics

          Reference
Prediction  3  4  5  6  7  8
         3  0  0  0  0  0  0
         4  0  0  0  0  0  0
         5  1  5 68 34  1  0
         6  0  7 20 47 16  1
         7  0  0  3 11 10  3
         8  0  0  0  0  0  0

Overall Statistics

               Accuracy : 0.5507
                 95% CI : (0.4834, 0.6165)
    No Information Rate : 0.4053
    P-Value [Acc > NIR] : 6.923e-06

                  Kappa : 0.2878

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5
Sensitivity          0.000000  0.00000   0.7473
Specificity          1.000000  1.00000   0.6985
Pos Pred Value            NaN      NaN   0.6239
Neg Pred Value       0.995595  0.94714   0.8051
Prevalence           0.004405  0.05286   0.4009
Detection Rate       0.000000  0.00000   0.2996
Detection Prevalence 0.000000  0.00000   0.4802
Balanced Accuracy    0.500000  0.50000   0.7229
                     Class: 6 Class: 7 Class: 8
Sensitivity            0.5109  0.37037  0.00000
Specificity            0.6741  0.91500  1.00000
Pos Pred Value         0.5165  0.37037      NaN
Neg Pred Value         0.6691  0.91500  0.98238
Prevalence             0.4053  0.11894  0.01762
Detection Rate         0.2070  0.04405  0.00000
Detection Prevalence   0.4009  0.11894  0.00000
Balanced Accuracy      0.5925  0.64269  0.50000
```

Figure 7: Confusion Matrix

Sondhi, N. and Basu, R. (2022) 'Profiling the online premium brand consumers based on their fashion orientation', *Asia Pacific Journal of Marketing and Logistics*, 35(2), pp. 380–397. Available at: https://doi.org/10.1108/APJML-07-2021-0492.

Stahl, C., Stein, N. and Flath, C.M. (2021) 'Analytics Applications in Fashion Supply Chain Management—A Review of Literature and Practice', *IEEE Transactions on Engineering Management*, 70(4), pp. 1258–1282. Available at: https://doi.org/10.1109/TEM.2021.3075936.

Appendix

Appendix A: Kaggle Dataset

https://www.kaggle.com/datasets/yasserh/wine-quality-dataset?resource=download

Appendix B: CART-algorithm code

```
> selected_features <- wine[, !names(wine) %in% c("Id", "quality")]
>
>
> features <- wine[, -which(names(wine) == "quality")]
> scaled_features <- scale(features)
> selected_features <- wine[, !names(wine) %in% c("Id", "quality")]
```

```
> library(caret)
> set.seed(42)
> splitIndex <- createDataPartition(wine$quality, p = 0.8, list = FALSE)
> train_data <- wine[splitIndex,]
> test_data <- wine[-splitIndex,]
```

```
> library(rpart)
> cart_model <- rpart(quality ~ ., data = train_data, method = "class")
> rpart.plot(cart_model)

> png(file = "large_decision_tree.png", width = 4800, height = 4800, res = 300)
> rpart.plot(cart_model, type = 0, extra = 1, under = TRUE, cex = 1,
+            fallen.leaves = TRUE, box.palette = "RdBu")
> dev.off()
png
  2
```

```
4 0.013233      5   0.67675 0.77316 0.028442
5 0.010000      6   0.66352 0.74858 0.028343
> optimal_cp <- cart_model$cptable[which.min(cart_model$cptable[,"xerror"]),"CP"]
> pruned_cart_model <- prune(cart_model, cp = optimal_cp)
```

```
> train_data$quality <- factor(train_data$quality, levels = unique(wine$quality))
> test_data$quality <- factor(test_data$quality, levels = unique(wine$quality))
> cart_model <- rpart(quality ~ ., data = train_data, method = "class")
>
> predictions <- predict(cart_model, newdata = test_data, type = "class")
> predictions <- factor(predictions, levels = levels(test_data$quality))
> confusionMatrix(predictions, test_data$quality)
```