Assessed Practical III: Can I eat that mushroom?

Hazel.A.Fernando - 201202015

Last compiled on 07 June, 2024

To assess the performance of new methods, classic machine learning data sets are used. These are data sets that have been used before, and so the predictive accuracy of the new models can be bench marked against existing methods.

One such classical data set is the 'Mushroom data set', a selection of observations about different specimens of gilled mushrooms from The Audubon Society Field Guide to North American Mushrooms (1981).

Within this data set, it includes the following variables:

- Edible
- CapShape
- CapSurface
- CapColor
- Odor
- Height

In this instance, the output is the variable Edible.

Therefore, the aim of this practical is to evaluate the best model/method to determine whether a certain mushroom is poisonous given the other variables provided.

All the variables are non-numeric categorical data points (factors).

A decision tree (DT) model or a random forest (RF) model is considered for this type of outcome variable. As predictive accuracy is being prioritised over interpretability, these methods are more suitable options to explore.

&nbsp:

Read the data in and see what the data is like. All the variables are changed into factors for later packages to work.

```r
mushroom <- read.csv("mushrooms.csv", header = TRUE)
```

&nbsp:

Task 1

Training and testing data sets are created from the main data set to evaluate the predictive power of any models.

```r
set.seed(123)

train_indices <- createDataPartition(mushroom$Edible, p = 0.8, list = FALSE)
train_data <- mushroom[train_indices, ]
test_data <- mushroom[-train_indices, ]
```

A basic or baseline DT model is conducted using the *rpart* function, keeping the default parameter settings the same, and with all the inputs included, to investigate and set an initial accuracy score. This is so later iterations of the DT model can be compared to it.

```r
dt_all_cp1 <- rpart(Edible ~ CapSurface + CapColor + CapShape + Odor + Height,
                                method = "class",
                                data = train_data)

preds_1 <- predict(dt_all_cp1,
                        newdata = test_data,
                        type = "class")
```

```
## [1] "The initial accuracy of the DT model is 0.98892"
```

The predictive performance of the DT model can be improved by tuning the different parameters the function uses.

A series of loops are used to evaluate the best complexity parameter (CP) value, minbucket, maxdepth, and minsplit value that maximises the predictive power of the model. Each parameter is evaluated to find at which values would the DT model's accuracy exceed the initial baseline.

CP values It appears the best accuracy arises when the CP is set between 0.0001 and 0.001.

Cross validation is used to find the best CP value between 0.0001 and 0.001. All other factors have been kept the same.

```
## [1] "Best cp: 0.00029"
```

Maxdepth Values

```
## [1] "Best maxdepth: 7"
```

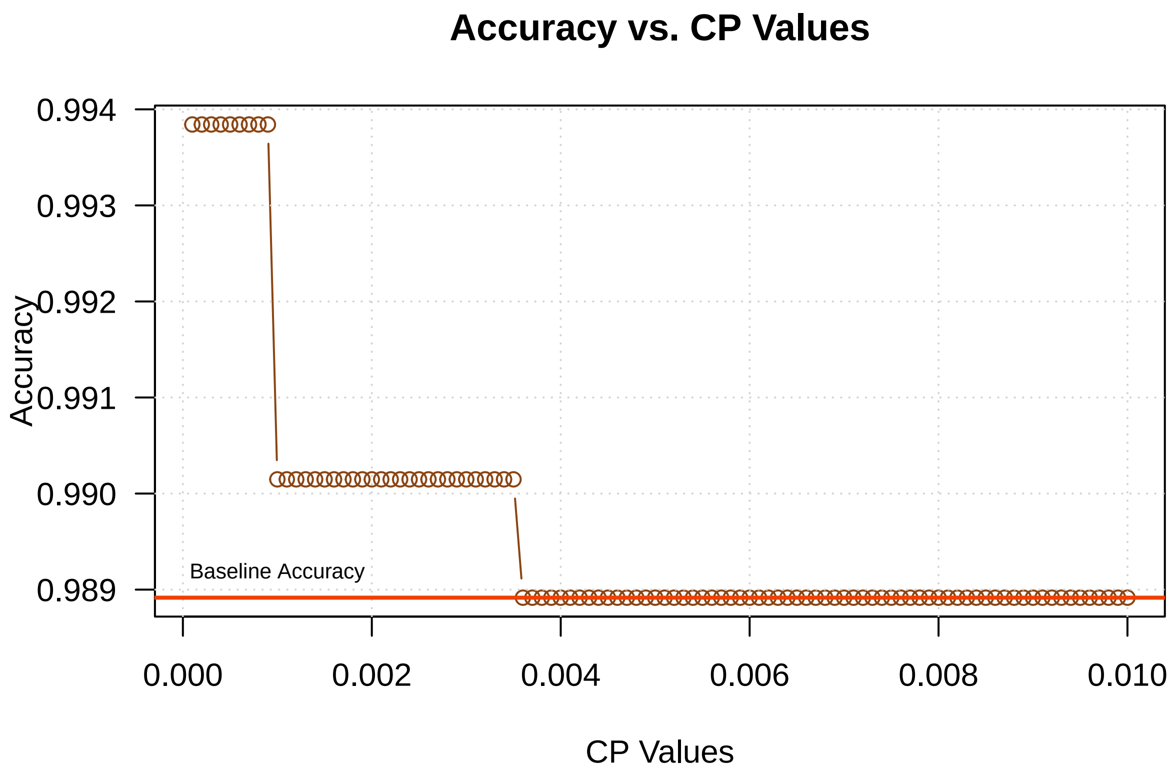

Figure 1: Plot of the prediction accuracy at different values of CP.

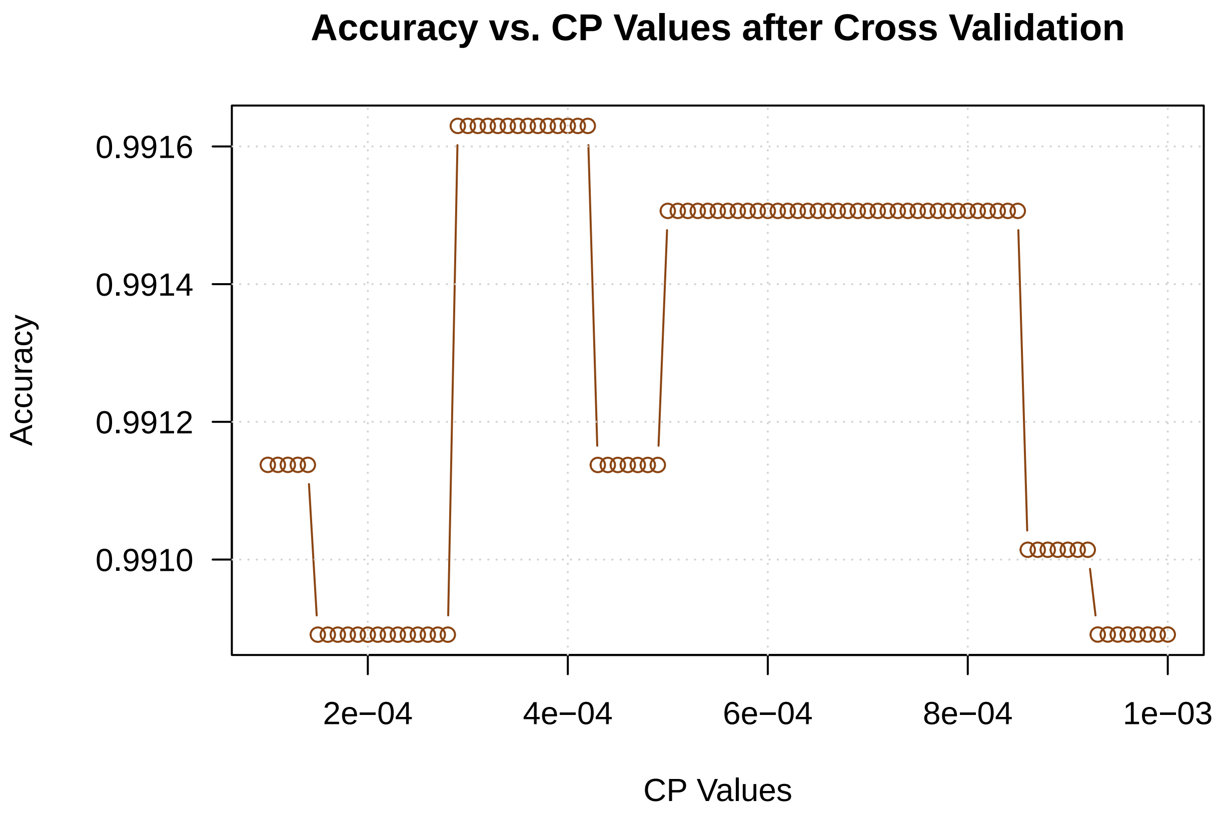

Figure 2: Plot of the prediction accuracy at different values of CP after cross validation.

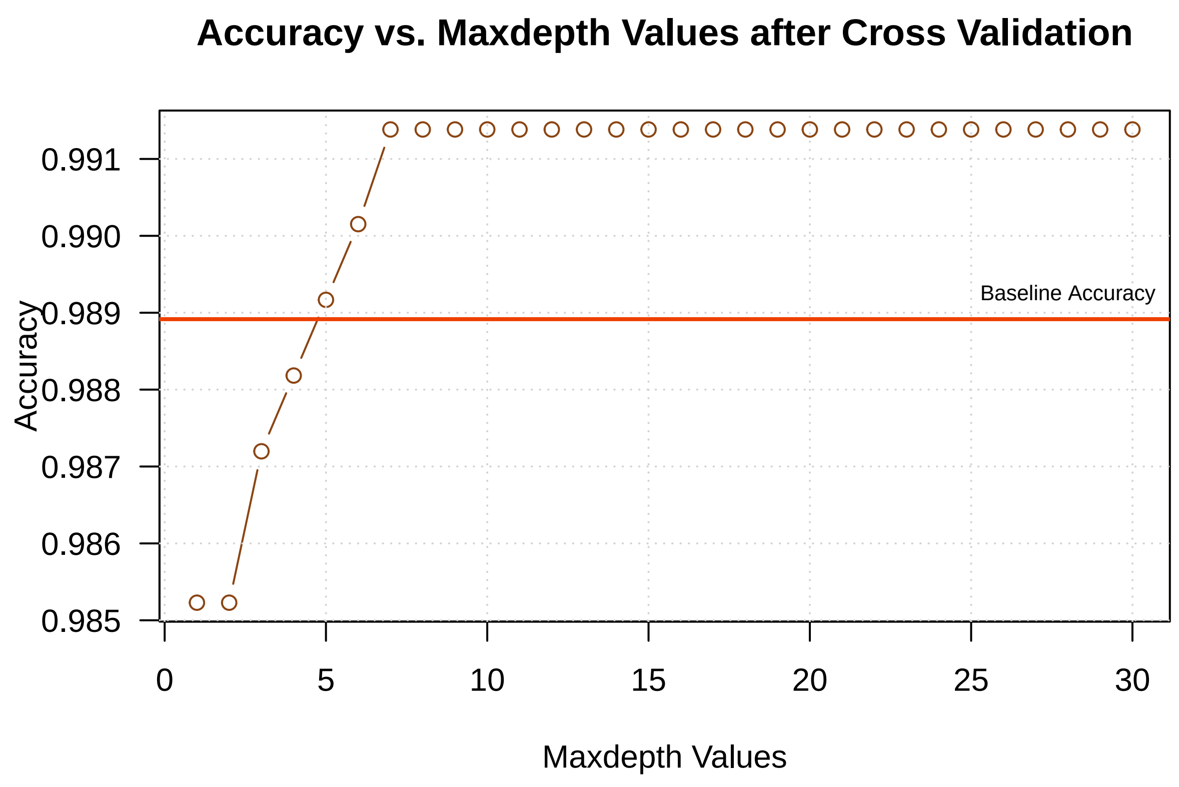

Figure 3: Plot of the prediction accuracy at different values of maxdepth after cross validation.

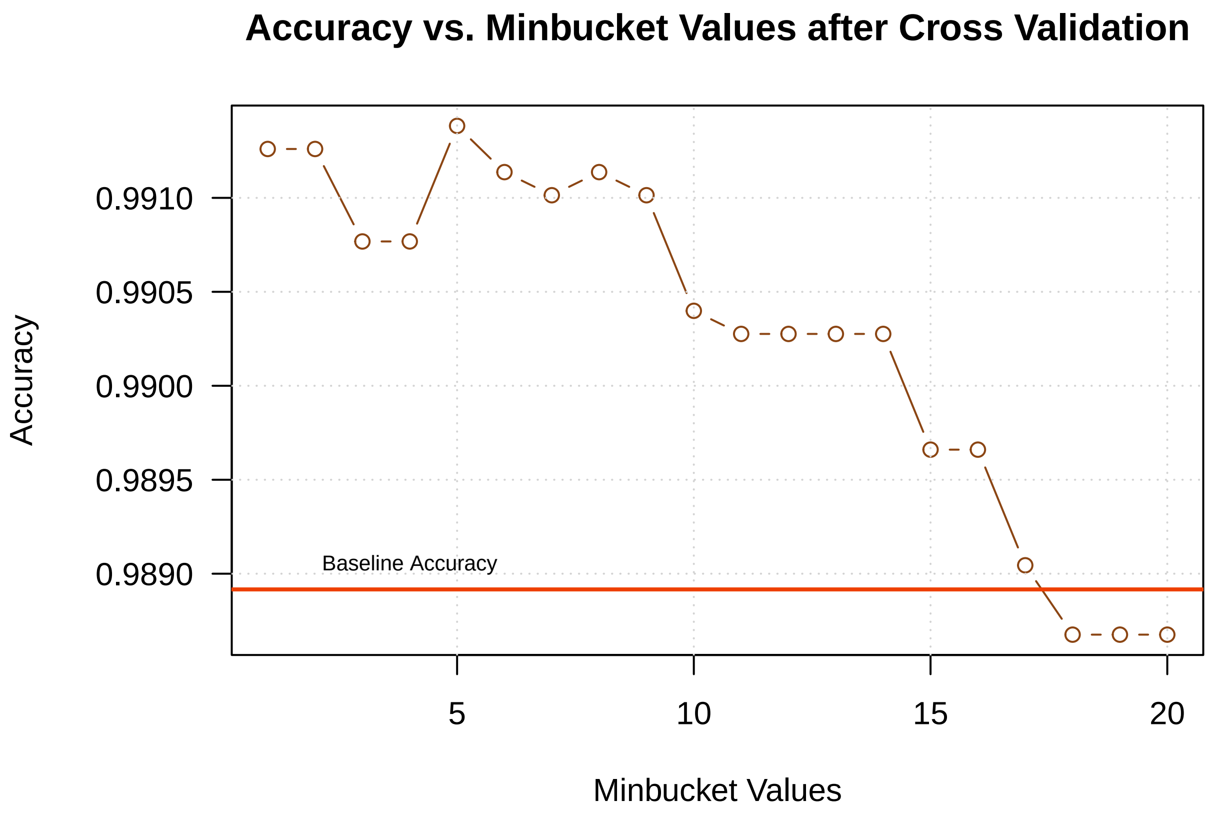

Figure 4: Plot of the prediction accuracy at different values of minbucket after cross validation.

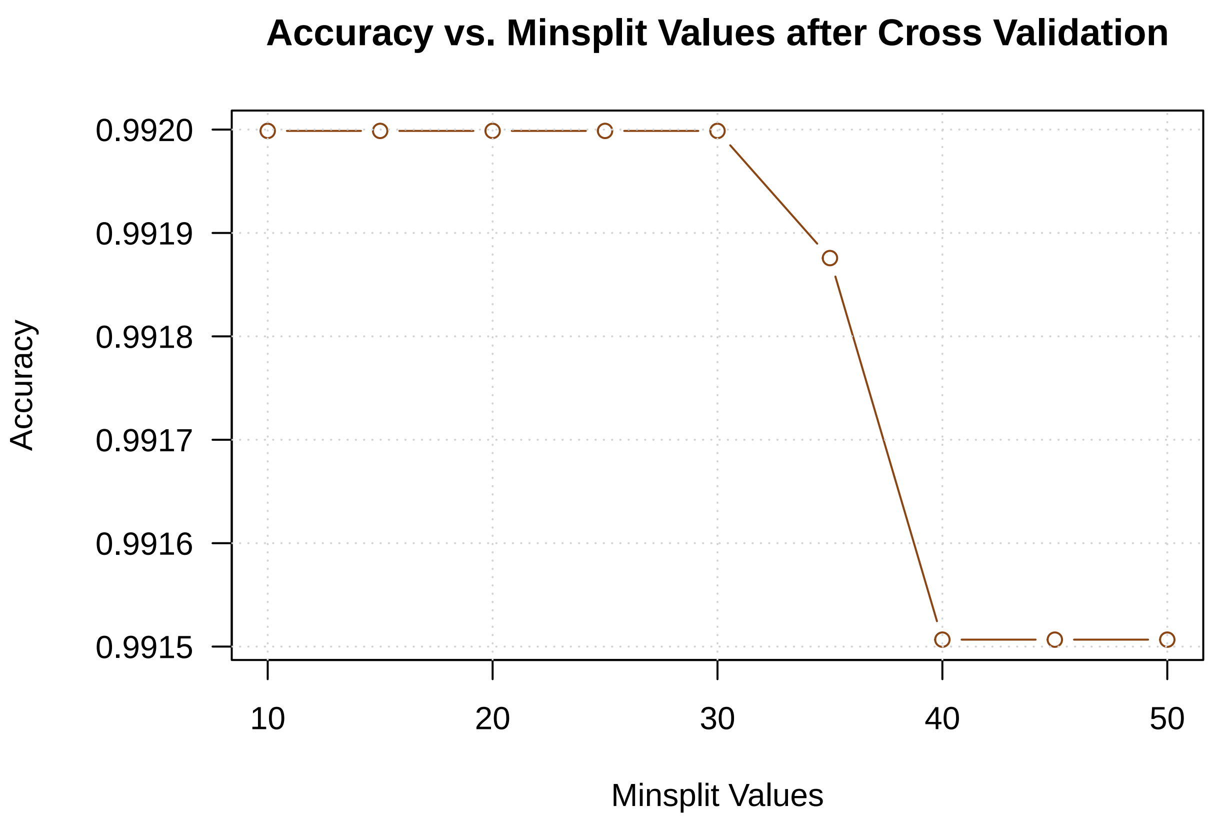

Figure 5: Plot of the prediction accuracy at different values of minsplit after cross validation.

Minbucket

```
## [1] "Best minbucket: 5"
```

```
## [1] "Best minsplit: 10"
```

A CP value between 0 and 0.001 did improve the model the most. When evaluated further with smaller CP values through cross validation, it appeared that CP $= < 0.0009$ is the cause of the increase in the accuracy, therefore a CP value of the best CP will be used in further DT modelling. Sequentially evaluating the maxdepth, minbucket, or minsplit did appear to change the prediction accuracy, therefore the best values of these parameters will be used.

Feature Selection

Feature selection of the inputs can be considered to evaluate whether a certain combination of inputs generate a better predictive DT model.

```r
for (i in seq_along(input_combinations_dt)) {
  input_combinations_dt_i <- input_combinations_dt[[i]]

  formula_dt <- as.formula(paste("Edible ~", paste(input_combinations_dt_i,
                                                   collapse = " + ")))

  combination_labels_dt[i] <- paste(input_combinations_dt_i, collapse = ", ")

  dt_model <- rpart(formula_dt,
                    cp = best_cp,
                    maxdepth = best_maxdepth,
                    minbucket = best_minbucket,
                    minsplit = best_minsplit,
                    method = "class",
                    data = train_data)

  preds_dt <- predict(dt_model, newdata = test_data, type = "class")

  accuracy_dt <- confusionMatrix(factor(preds_dt),
                                 factor(test_data$Edible))$overall[1]

  accuracies_dt[i] <- accuracy_dt
}
```

Table 1: List of all 31 input combinations and the predictive accuracy, in decending order, of the DT model when the combination of inputs were used.

	Combinations	Accuracies
19	CapSurface, CapShape, Odor	0.9919951
22	CapColor, CapShape, Odor	0.9919951
26	CapSurface, CapColor, CapShape, Odor	0.9919951
29	CapSurface, CapShape, Odor, Height	0.9919951
30	CapColor, CapShape, Odor, Height	0.9919951
31	CapSurface, CapColor, CapShape, Odor, Height	0.9919951
11	CapColor, Odor	0.9907635
17	CapSurface, CapColor, Odor	0.9907635
24	CapColor, Odor, Height	0.9907635
28	CapSurface, CapColor, Odor, Height	0.9907635
4	Odor	0.9889163
8	CapSurface, Odor	0.9889163
13	CapShape, Odor	0.9889163
15	Odor, Height	0.9889163
21	CapSurface, Odor, Height	0.9889163
25	CapShape, Odor, Height	0.9889163
6	CapSurface, CapColor	0.6945813
18	CapSurface, CapColor, Height	0.6945813
16	CapSurface, CapColor, CapShape	0.6890394
27	CapSurface, CapColor, CapShape, Height	0.6890394
10	CapColor, CapShape	0.6477833
23	CapColor, CapShape, Height	0.6434729
7	CapSurface, CapShape	0.6305419
20	CapSurface, CapShape, Height	0.6114532
2	CapColor	0.5979064
12	CapColor, Height	0.5979064
1	CapSurface	0.5788177
9	CapSurface, Height	0.5788177
14	CapShape, Height	0.5640394
3	CapShape	0.5609606
5	Height	0.5178571

Odor appears to have the most, if not the only, influence on the outcome.

Every combination that include *Odor* has the highest predictive value, indicating *Odor* has substantially more influence in the output. The importance of each input was accessed (using the *caret* function), and showed that *Odor* has considerably the highest overall importance. The other inputs, *CapColour*, *CapShape* and *CapSurface* all show similar importance or influence on the output.

Height is suggested to have no effect on the output and could be reason to possibly remove *Height* from the final DT model to simplify the interpretability and improve its predictive power. Therefore, *Height* will NOT be included in the final DT model.

The maximised DT model will use all the inputs but *Height*, and the optimised parameters.

```
best_dt <- rpart(formula_dt,
                              cp = best_cp,
                        maxdepth = best_maxdepth,
```

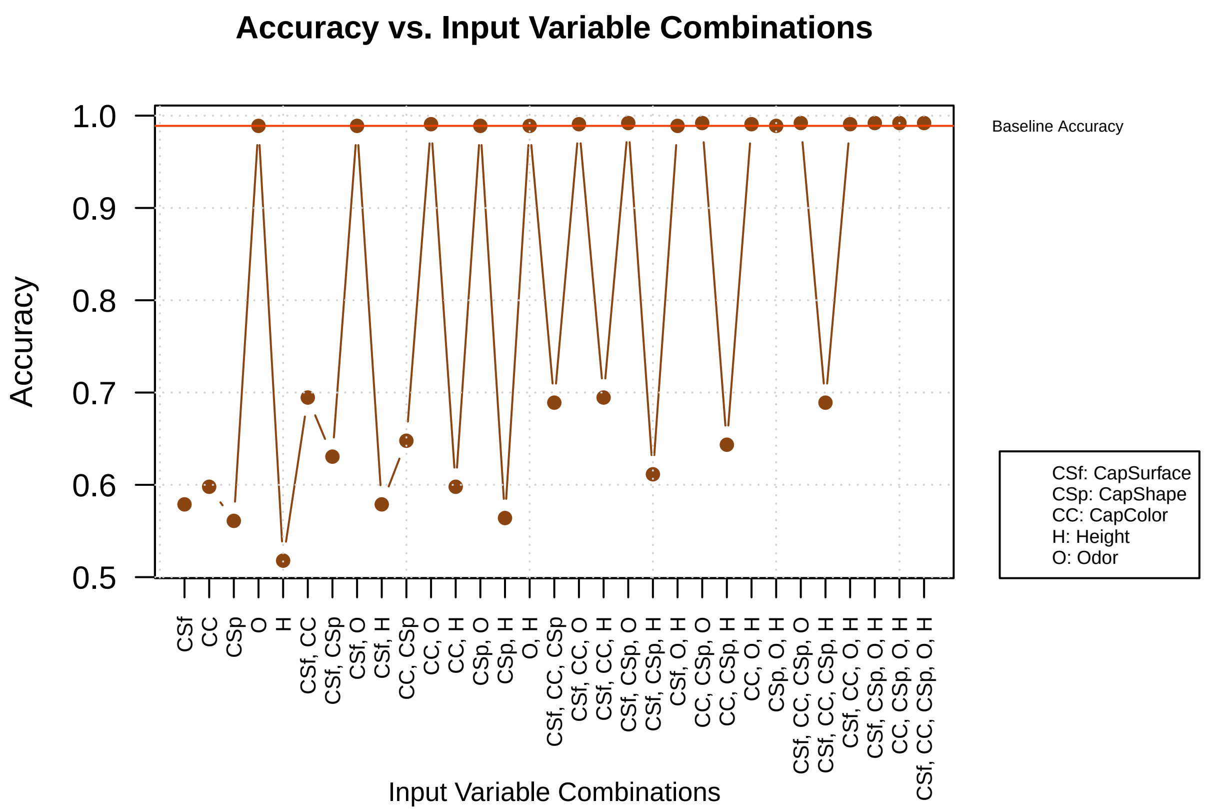

Figure 6: Plot of the predicted accuracy when each combination of inputs were used.

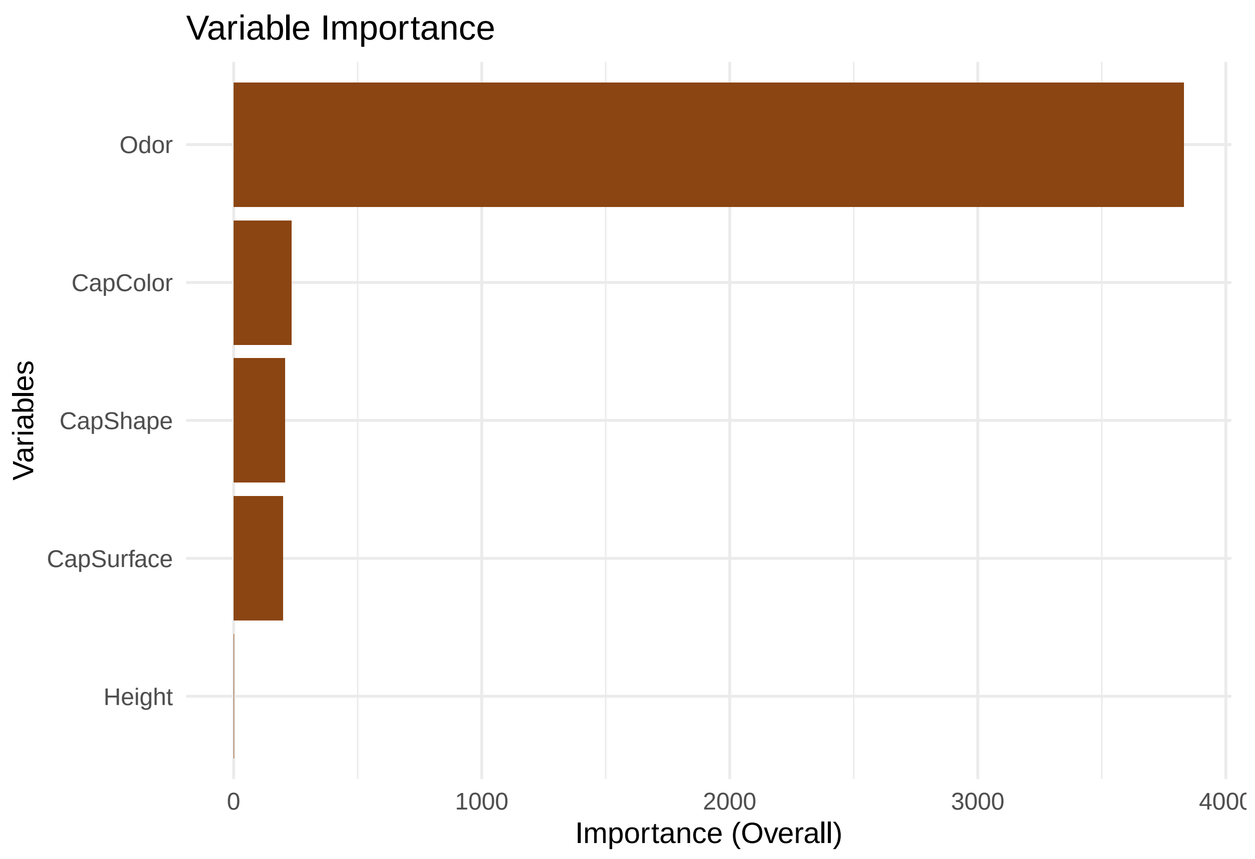

Figure 7: Bar plot of the Mean Decrease Gini value for each input variable. The DT model this was evaluated from is using the optimised parameter values.

```
                    minbucket = best_minbucket,
                    minsplit = best_minsplit,
                    method = "class",
                    data = train_data)

preds_2 <- predict(best_dt,newdata = test_data, type = "class")
```

The predictive accuracy of the DT method using the best combination of inputs and parameters is 0.992

The accuracy of the best DT model is higher than the accuracy of the baseline DT model. Therefore, the tuning of the hyperparameters and feature selection were effective in maximising the DT model.

A preliminary RF model is created using the *randomForest* function, keeping the default parameter settings the same and including all the input variables. The prediction accuracy score will be used as the baseline to help model the best RF model.

```
rf_model <- randomForest(Edible ~ CapShape + CapColor + CapSurface + Height +
                           Odor,
                        data=train_data)

preds_rf <- predict(rf_model, newdata=test_data, type="class")
```

```
## [1] "The initial accuracy of the RF model is 0.99323"
```

The accuracy of the baseline RF model is already higher than the best DT model.

In the same way for the DT model, the predictive performance of the RF model can be improved by tuning the model's parameters.

The mtry, ntree and nodesize parameters are evaluated individually to see the optimal values to use to improve the predictive power of the RF model.

&nbsp:

mtry

```
## [1] "Best mtry: 5"
```

Nodesize

```
## [1] "Best nodesize: 2"
```

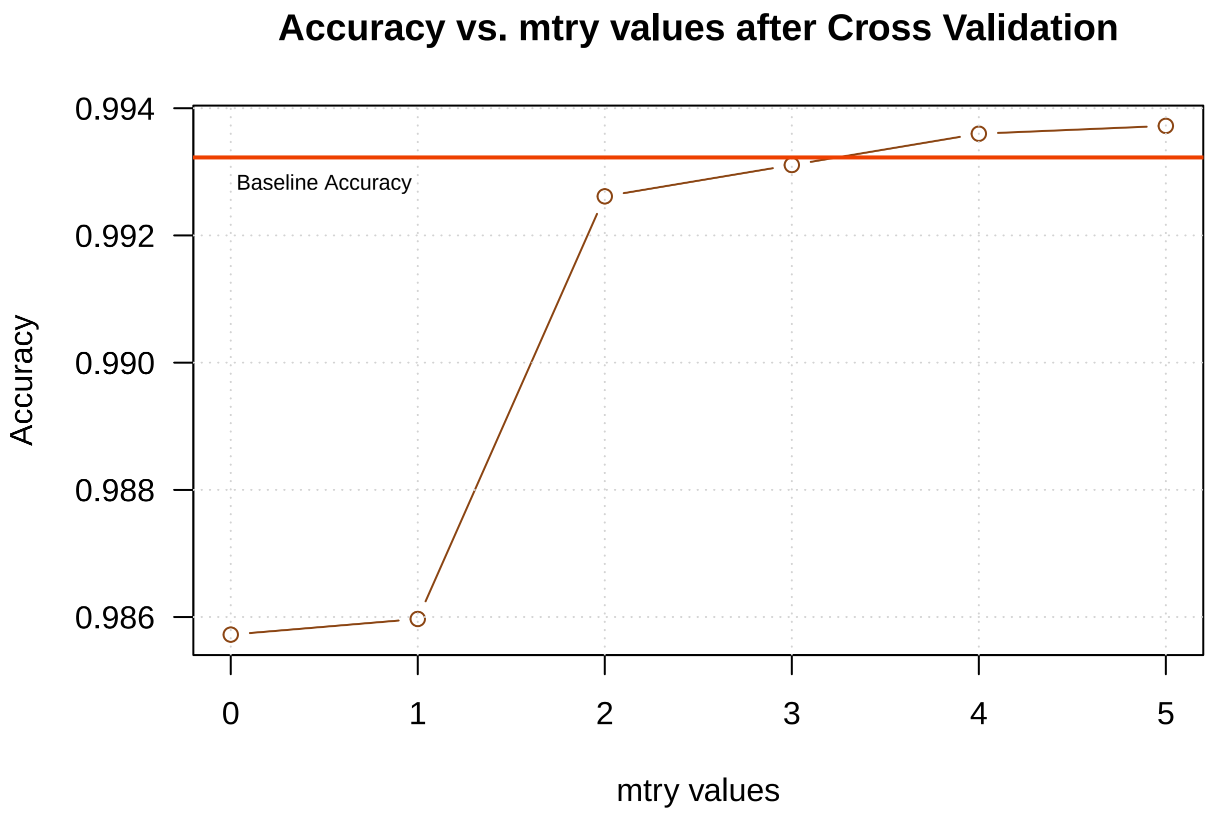

Figure 8: Plot of the accuracy of the RF model at each increasing mtry value after cross validation.

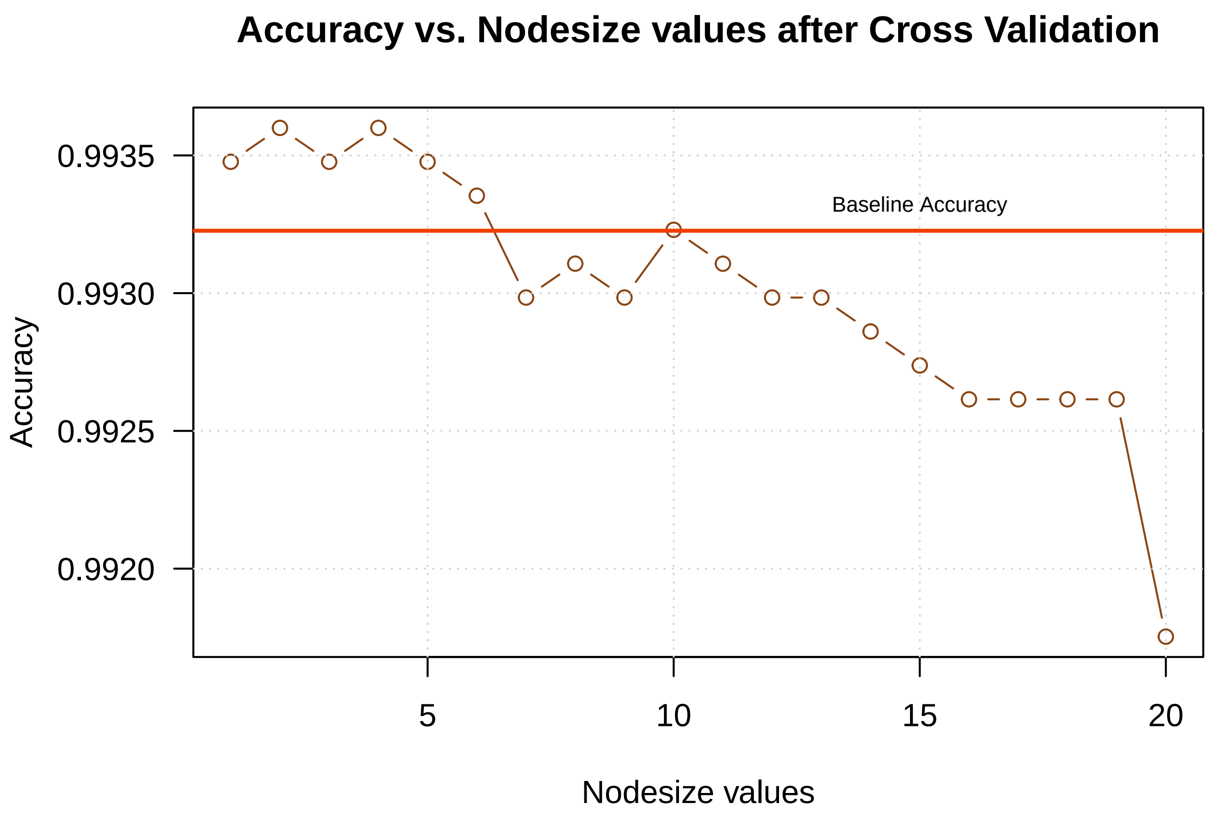

Figure 9: Plot of the accuracy of the RF model at each increasing nodesize value after cross validation.

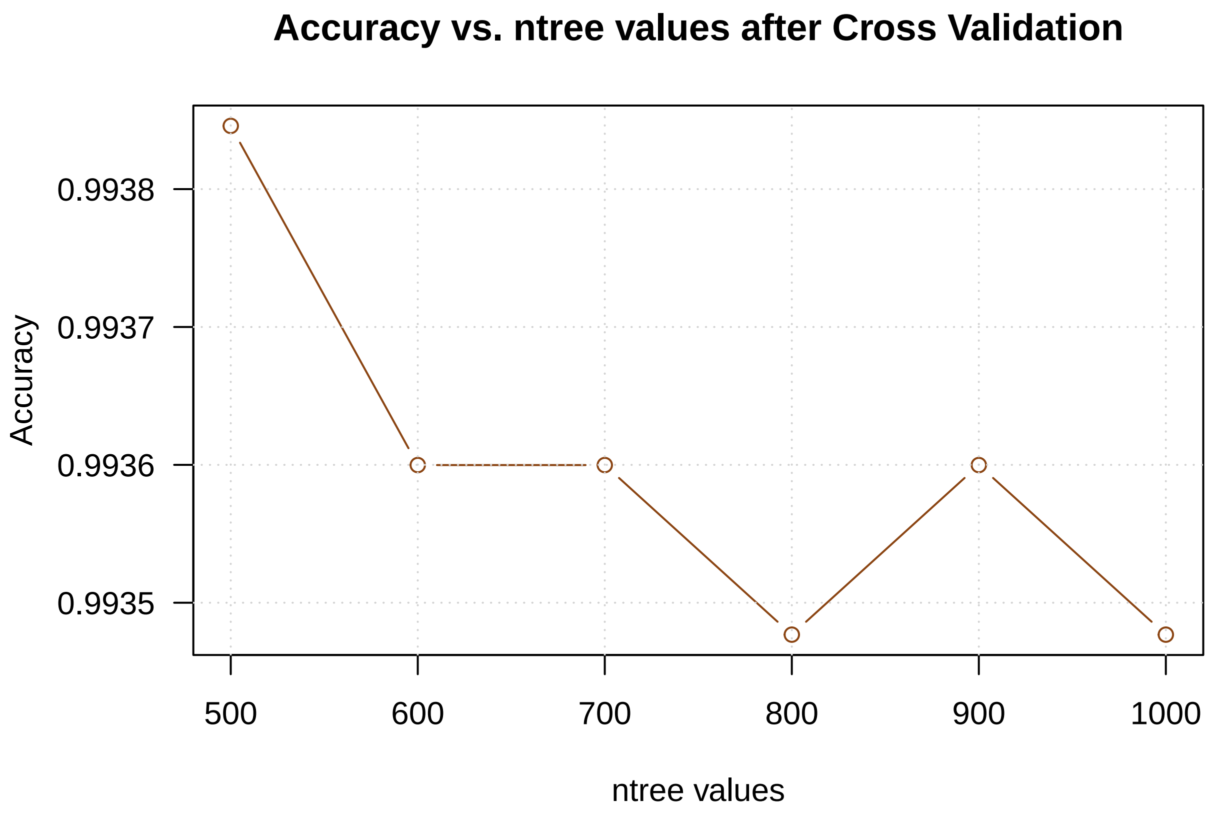

Figure 10: Plot of the accuracy of the RF model at each increasing ntree value after cross validation.

ntree

```
## [1] "Best ntree: 500"
```

The best mtry and nodesize values had given an increase in accuracy as it increased the potential to capture any complex relationships between the inputs. A caution is that this may lead to overfitting, poor generalisation to any unseen data, and more difficulty in interpreting the model. For the moment, these parameters will be used.

At any value between 500 and 1000, the accuracy is still higher than the baseline accuracy, therefore any value of ntree can be used between this range, the best ntree size suggested through the cross validation will be used. ntree > 1000 would show very little improvement, but risk increasing computational time, and so was not considered.

Feature Selection

Feature selection of the inputs are again considered to evaluate whether a certain combination of inputs generate a better predictive RF model.

Table 2: List of all 31 input combinations and the predictive accuracy, in decending order, of the RF model when the combination of inputs were used.

	Combinations	Accuracies
31	CapSurface, CapColor, CapShape, Odor, Height	0.9944581
26	CapSurface, CapColor, CapShape, Odor	0.9938424
19	CapSurface, CapShape, Odor	0.9919951
22	CapColor, CapShape, Odor	0.9919951
29	CapSurface, CapShape, Odor, Height	0.9919951
30	CapColor, CapShape, Odor, Height	0.9919951
17	CapSurface, CapColor, Odor	0.9913793
28	CapSurface, CapColor, Odor, Height	0.9913793
11	CapColor, Odor	0.9907635
24	CapColor, Odor, Height	0.9907635
8	CapSurface, Odor	0.9895320
21	CapSurface, Odor, Height	0.9895320
4	Odor	0.9889163
13	CapShape, Odor	0.9889163
15	Odor, Height	0.9889163
25	CapShape, Odor, Height	0.9889163
16	CapSurface, CapColor, CapShape	0.7087438
27	CapSurface, CapColor, CapShape, Height	0.7081281
18	CapSurface, CapColor, Height	0.6964286
6	CapSurface, CapColor	0.6951970
10	CapColor, CapShape	0.6477833
23	CapColor, CapShape, Height	0.6416256
7	CapSurface, CapShape	0.6305419
20	CapSurface, CapShape, Height	0.6114532
2	CapColor	0.5979064
12	CapColor, Height	0.5979064
1	CapSurface	0.5788177

	Combinations	Accuracies
9	CapSurface, Height	0.5788177
14	CapShape, Height	0.5640394
3	CapShape	0.5609606
5	Height	0.5178571

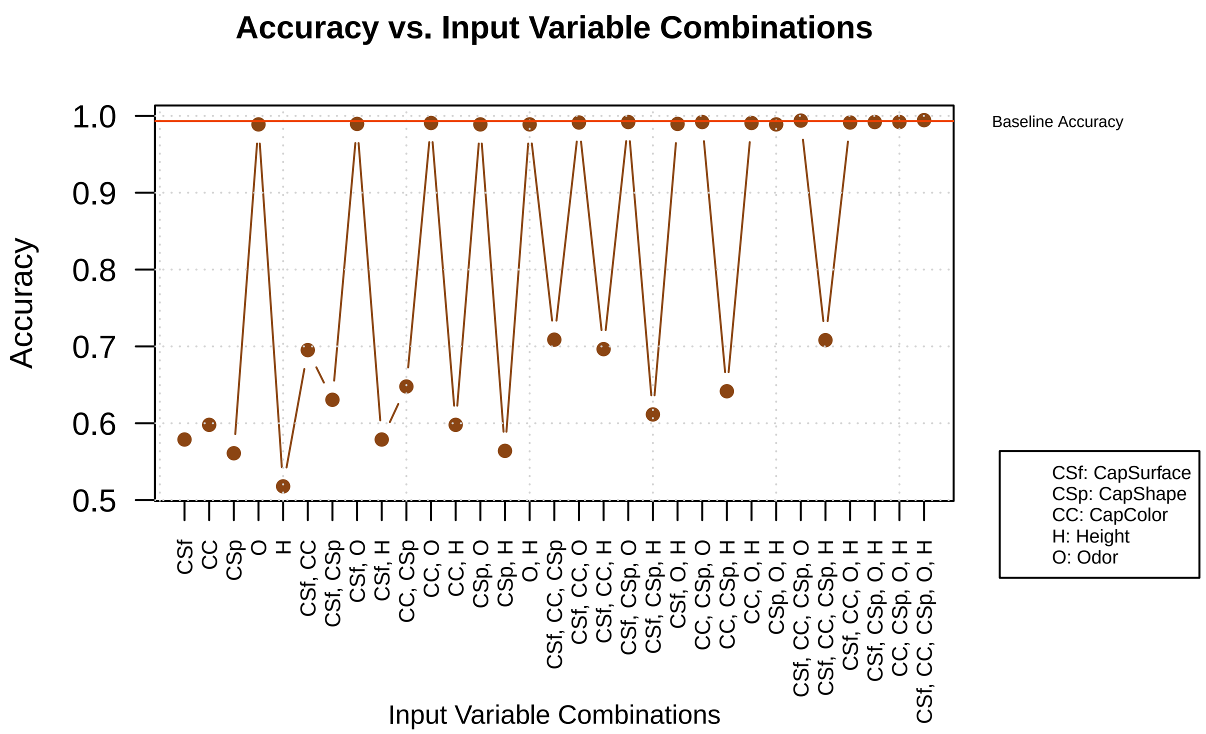

Figure 11: Plot of the predicted accuracy from the RF model when each combination of inputs were used.

Analysis of Feature Importance Using Mean Decrease Gini in Random Forest Models The Mean Decrease Gini, also known as Gini Importance, is a measure of the importance of each variable in a Random Forest (RF) model. This metric indicates how each input contributes to the model's decision-making process, with higher values signifying greater importance to the model's output.

Key Findings on Feature Importance Similar to the Decision Tree (DT) model, the *Odor* variable had the most significant impact on the RF model's predictive accuracy. This conclusion is supported by various metrics, including Table 1, Figure 11, and Figure 12. Other features such as *CapColor*, *CapShape*, and *CapSurface* also demonstrated some influence, as combinations including these inputs yielded high predictive accuracy and the next highest Mean Decrease Gini scores.

Interestingly, when all inputs were included in the model, the overall accuracy was slightly higher than models excluding *Height*. This finding contradicts the Mean Decrease Gini measurement for *Height* (Figure 12), which indicates that *Height* has minimal influence on the model's output. Additionally, models excluding *Height* did not show an accuracy increase compared to those with the same input combinations but

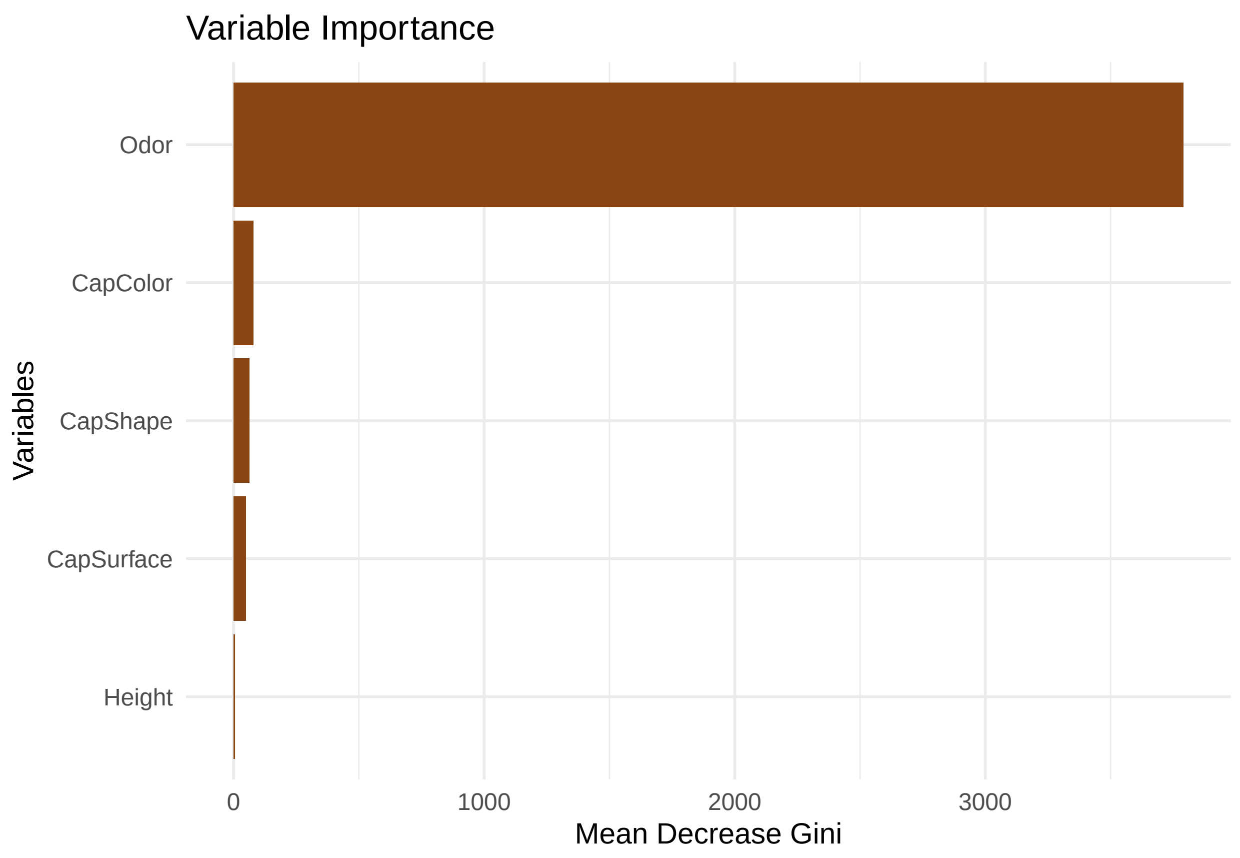

Figure 12: Bar plot of the Mean Decrease Gini value for each input variable. The model this was evaluated from is using the optimised parameter values.

with *Height* included. For instance, models with the inputs *CapSurface, CapColor, Odor* and *CapSurface, CapColor, Odor, Height* both achieved an accuracy of 0.9913793.

These observations suggest that the tuned parameters may be causing the RF model to overfit, thereby inflating the importance of some variables.

Optimal Parameter Selection Through the tuning process, it was determined that setting `mtry` (the number of variables randomly sampled at each split) to 2 appeared to be the best choice, even though `mtry = 4` resulted in a slightly higher accuracy score. Using `mtry = 4` risks overfitting, as indicated by a warning when `mtry = 5` was used. Choosing `mtry = 2` helps ensure no single feature dominates the splits, promoting a more generalised model.

This adjustment aim to create a balanced model that avoids overfitting by ensuring it does not overly rely on any single feature. Consequently, the model is more likely to generalise well to unseen data, maintaining robust predictive performance.

Therefore, the maximised RF model will have a change of its hyperparameters to better suit predicting unseen data, and include all the inputs except Height.

```
best_rf <- randomForest(formula_rf,
                 ntree = best_ntree,
                 mtry = 2,
                 nodesize = best_nodesize,
                 method = "class",
                 data = train_data,
                 importance = TRUE)

preds_3 <- predict(best_rf,
                       newdata = test_data,
                       type = "class")
```

The predictive accuracy of the RF method using the best combination of inputs and parameters is 0.99384

The accuracy of the best RF model is higher than the accuracy of the baseline RF model.

The accuracy of the best RF model is higher than the accuracy of the best DT model. Therefore, the tuning of the hyperparameters and feature selection were effective in maximising the RF model.

The results of this shows that using a RF model with select inputs and parameters out performed the DT model even when the best inputs and parameters for this method were used. This is an expected outcome as the the RF is an ensemble of DT models. By aggregating different DTs, the probability that the majority will predict correctly, or accurately, increases as stated by Condorcet's Jury Theorem.

Analysis of the Decision Tree Model for Mushroom Classification The decision tree model classifies mushrooms as either *Edible* or *Poisonous* based on various features. The tree is constructed using specific parameters designed to control its complexity and depth, striking a balance between underfitting and overfitting.

Key Features and Model Structure The most influential features in the model are *Odor*, *CapColor*, *CapShape*, and *CapSurface*, while *Height* was excluded due to its lack of importance in predicting the mushroom's edibility. This supports the notion that *Height* does not significantly impact the classification outcome.

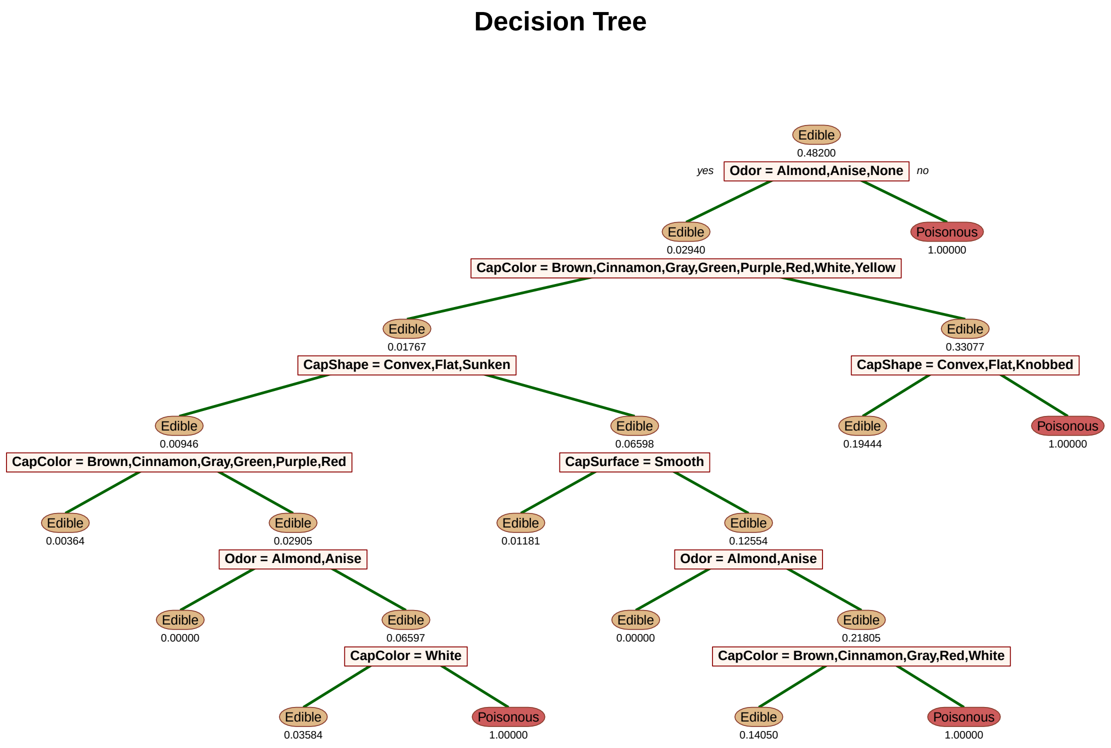

Figure 13: Decision Tree diagram using the DT model with the highest accuracy. As Several models have the same highest accuracy, the model with all the inputs were used.

- Root Node: The root node of the tree is *Odor*, indicating it is the most critical feature for distinguishing between *Edible* and *Poisonous* mushrooms.
- Leaf Nodes: Each leaf node in the tree shows the proportion of correctly classified samples, providing a quantitative measure of the model's confidence. Although some leaf nodes achieve 100% correct classification, indicating clear distinctions between *Poisonous* and *Edible* mushrooms, others still misclassify some samples. In practical terms, this could lead to dangerous misclassifications of *Poisonous* mushrooms as *Edible*, highlighting a potential risk in real-world applications.

Interpretability and Complexity The decision tree is relatively easy to understand and interpret. For instance, a straightforward rule derived from the tree is that if a mushroom has an *Odor* of *Almond*, *Anise*, or *None*, it is classified as *Edible*; otherwise, it is classified as *Poisonous*. This visual representation aids those who may not be familiar with the underlying generation process in comprehending the model's decision-making logic.

However, the model's high complexity is evident from its numerous branches and considerable depth. This intricacy can make the tree challenging to interpret, particularly for non-experts. While the detailed breakdown provided by the tree is thorough, the extensive number of splits and deep nodes may obscure clarity.

Cross-Validation and Model Tuning Determining whether the model has been overfitted or underfitted requires cross-validation for each model configuration. Although the parameters were tuned to optimise predictive performance, they may not necessarily represent the most optimal values. Cross-validation would provide a more reliable assessment of the model's generalisability and help in fine-tuning the parameters to achieve the best balance between accuracy and simplicity.

Task 2

To determine which model, either the optimised DT model or the optimised RF model, has the best predictive accuracy, k-fold cross validation will be performed. This will also help determine whether the models have been over- or under-fitted.

Cross Validation

Functions are created to conduct k-fold cross validation for both a decision tree and random forest. Within the formulas, both will use the tune parameter values settled on from task 1.

```r
#Function for k-fold cross validation for decision tree
cv_decision_tree <- function(data, formula, k = 10) {
  folds <- createFolds(data$Edible, k = k, list = TRUE, returnTrain = TRUE)
  accuracies <- numeric(k)

  for (i in 1:k) {
    train_indices <- folds[[i]]
    train_data_cv <- data[train_indices, ]
    test_data_cv <- data[-train_indices, ]

    model <- rpart(formula, data = train_data_cv, method = "class",
                   cp = best_cp,
                    maxdepth = best_maxdepth,
                    minbucket = best_maxdepth,
                    minsplit = best_minsplit)

    preds <- predict(model, newdata = test_data_cv, type = "class")
    accuracy <- sum(preds == test_data_cv$Edible) / nrow(test_data_cv)
    accuracies[i] <- accuracy
  }

  mean(accuracies)
}


#Function for k-fold cross validation for random forest
cv_random_forest <- function(data, formula, k = 10) {
  folds <- createFolds(data$Edible, k = k, list = TRUE, returnTrain = TRUE)
  accuracies <- numeric(k)

  for (i in 1:k) {
    train_indices <- folds[[i]]
    train_data_cv <- data[train_indices, ]
    test_data_cv <- data[-train_indices, ]

    model <- randomForest(formula, data = train_data_cv,
                          ntree = best_ntree,
                          mtry = 2,
                          nodesize = best_nodesize)

    preds <- predict(model, newdata = test_data_cv)
    accuracy <- sum(preds == test_data_cv$Edible) / nrow(test_data_cv)

    accuracies[i] <- accuracy
  }

  mean(accuracies)
}
```

Running the k fold cross validation.

```r
for (i in seq_along(input_combinations_best)) {
  input_combinations_best_i <- input_combinations_best[[i]]
  formula_best <- as.formula(paste("Edible ~", paste(input_combinations_best_i,
                                                     collapse = " + ")))
  combination_labels_best[i] <- paste(input_combinations_best_i,
                                      collapse = ", ")

  if (i <= 1) {
    # Decision Tree Cross-Validation
    accuracies_best[i] <- cv_decision_tree(mushroom, formula_best, k = 10)
    model_types[i] <- "Decision Tree"
  } else {
    # Random Forest Cross-Validation
    accuracies_best[i] <- cv_random_forest(mushroom, formula_best, k = 10)
    model_types[i] <- "Random Forest"
  }
}
```

Table 3: Table of the accuracy of the models after cross validation, in descending order.

	Combination	Accuracy	ModelType
2	CapSurface, CapColor, CapShape, Odor	0.9923679	Random Forest
1	CapSurface, CapColor, CapShape, Odor	0.9913820	Decision Tree

It appears that the RF model has marginally better accuracy than the DF model. To evaluate whether this difference is statistically significant, a paired t-test will be used.

Statistical Test for Significance

The statistical difference between the unchanged DT and RT models and the optimised models will be considered.

To recap:

- Unchanged/ basic DT Model: All inputs and default parameters
- Unchanged/ basic RF Model: All inputs and default parameters
- Optimised DT Model: All inputs except *Height* and tuned parameters
- Optimised RF Model: All inputs except *Height* and tuned parameters

Each of the four models will be cross validated.

```
result_basic_dt <- cv_decision_tree_basic(mushroom, formulas_best[[1]], k = 10)

result_basic_rf <- cv_random_forest_basic(mushroom, formulas_best[[2]], k = 10)

result_dt <- cv_decision_tree(mushroom, formulas_best[[3]], k = 10)

result_rf <- cv_random_forest(mushroom, formulas_best[[4]], k = 10)
```

Table 4: Averaged accuracy of each model after k-fold cross validation.

Model	Accuracy	Model_type
Basic DT	0.9852299	Decision Tree
Basic RF	0.9922447	Random Forest
Optimised DT	0.9918780	Decision Tree
Optimised RF	0.9921235	Random Forest

Paired t tests across different pairs of the chosen models.

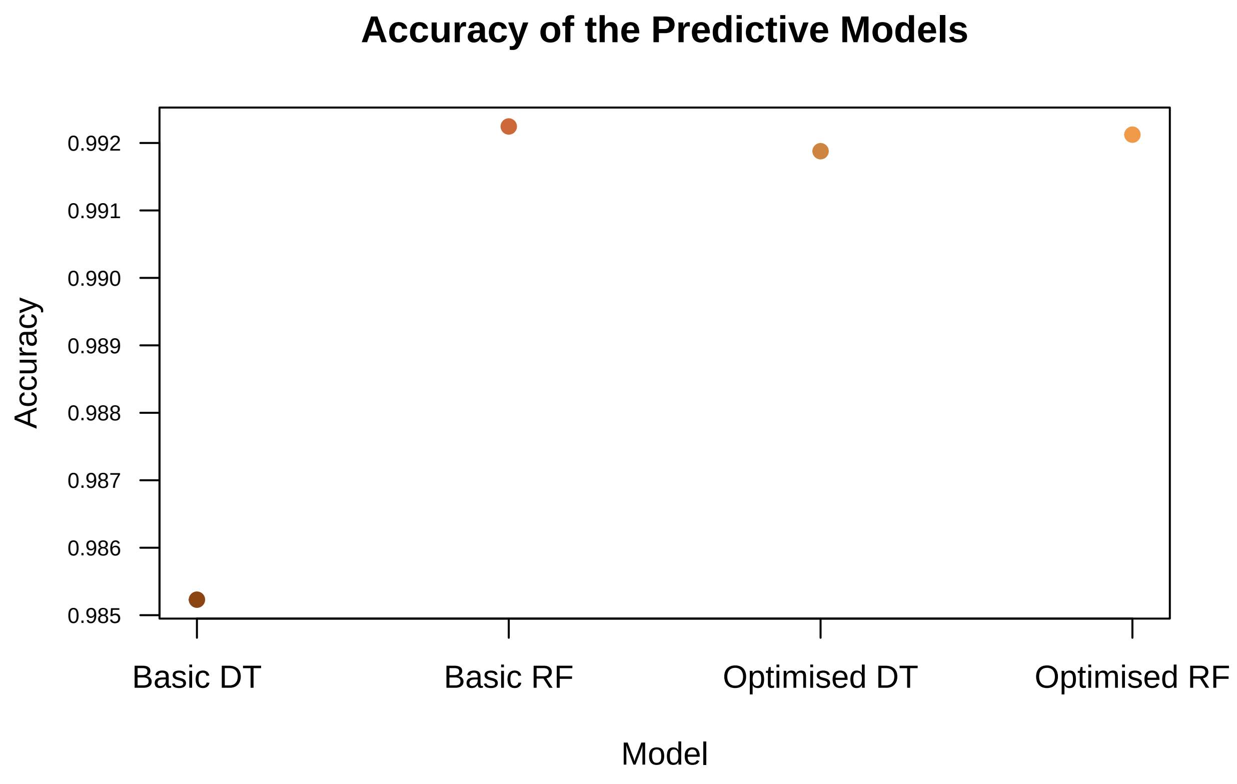

Figure 14: Plot of the averaged accuracy scores of each predictive model after cross validation.

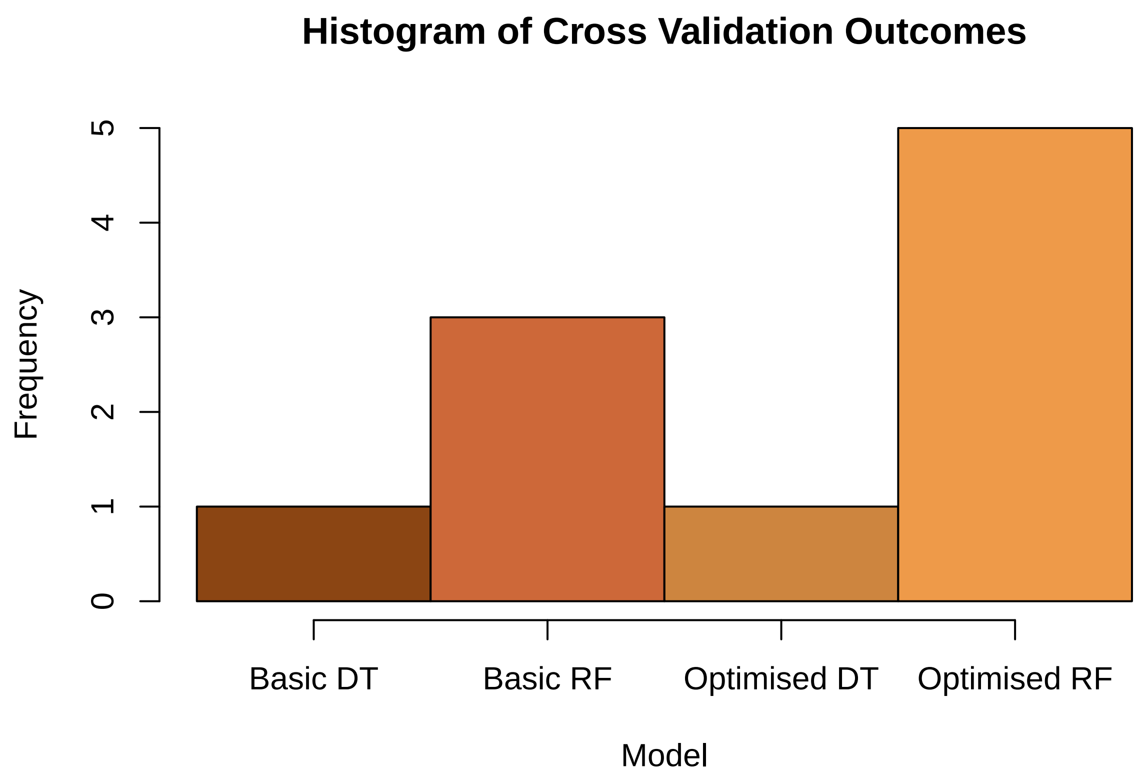

Figure 15: Plot of the frequency at which each model is chosen as the best predictive model, i.e. had the highest accuracy. Each model is iterated the equivalent to 50 times.

```r
#Optimised DT model vs Optimised RF model
t_test_results_best <- t.test(dt_accuracy, rf_accuracy, paired = TRUE)

#Unchanged DT model vs Unchanged RF model
t_test_results_basic <- t.test(dt_basic_accuracy, rf_basic_accuracy,
                               paired = TRUE)

#Optimised DT model vs Unchanged DT model
t_test_results_dt <- t.test(dt_basic_accuracy, dt_accuracy, paired = TRUE)

#Optimised RF model vs Unchanged RF model
t_test_results_rf <- t.test(rf_basic_accuracy, rf_accuracy, paired = TRUE)

#Optimised RF model vs Unchanged DT model
t_test_results_basic_dt_rf <- t.test(dt_basic_accuracy, rf_accuracy,
                                     paired = TRUE)

#Optimised DT model vs Unchanged RF model
t_test_results_basic_rf_dt <- t.test(rf_basic_accuracy, dt_accuracy,
                                     paired = TRUE)
```

Table 5: P-values between the models. Calculated through t tests, and using the accuracy scores from cross validation. [CI: Confidence Interval].

Model_1	Model_2	P_value	Lower_CI	Upper_CI	Significant
Optimised DT	Optimised RF	0.8992300	-0.0045100	0.0040189	No
Basic DT	Basic RF	0.0003456	-0.0098618	-0.0041679	Yes
Optimised DT	Basic DT	0.0003110	-0.0093068	-0.0039895	Yes
Optimised RF	Basic RF	0.9223845	-0.0026149	0.0028572	No
Optimised RF	Basic DT	0.0001215	-0.0093205	-0.0044669	Yes
Optimised DT	Basic RF	0.8312121	-0.0034140	0.0041475	No

Comparison and Optimisation of Decision Tree and Random Forest Models for Predicting Mushroom Edibility To identify the best model for predicting mushroom edibility with the highest accuracy, both basic Decision Tree (DT) and Random Forest (RF) models were optimised by selecting the best parameters and set of inputs. The optimisation process involved cross-validation, iterative cross-validation, and statistical analysis using t-tests. Based on these methods, it was determined that the **optimised Random Forest (RF) model** provided the best accuracy (Figure 15).

Basic Model Performance Both the basic DT and RF models were initially fitted using all available inputs (*CapShape*, *CapColor*, *CapSurface*, *Height*, and *Odor*) with default parameters:

- The basic DT model achieved an accuracy of approximately 98.89%.#
- The basic RF model achieved a higher accuracy of approximately 99.32%.

A paired t-test confirmed that the difference in accuracy between the basic DT and RF models was statistically significant (Table 5).

Decision Tree Optimisation The DT model was optimised by tuning its parameters through cross-validation. Specifically: - Reducing the complexity parameter (CP) to less than 0.0009 improved accuracy. - Feature selection revealed that *Height* did not enhance the model's performance, whereas *Odor* was the most influential feature.

The optimised DT model, which included all inputs except *Height* and utilised the tuned parameters, significantly improved predictive accuracy, reaching approximately 99.20%.

Random Forest Optimisation Similarly, the RF model was optimised by cross-validating its parameters:

- Feature importance analysis showed that *Height* had the least impact, while *Odor* had the most.
- Consequently, the optimised RF model excluded *Height*.

This optimised RF model initially increased accuracy to approximately 99.38%. However, subsequent cross-validation showed a slight decrease in accuracy (Figure 14), indicating possible overfitting. Despite this, iterative cross-validation consistently identified the optimised RF model as the best performer in most iterations.

Comparative Analysis The difference in accuracy between the optimised DT and RF models was marginal and not statistically significant, suggesting that neither model was significantly superior to the other in terms of predictive accuracy.

Feature Importance and Model Interpretation Removing *Height* and tuning parameters did not significantly affect accuracy but helped simplify model interpretation by reducing input variables. Mean Decrease Gini tests (Figure 7 and Figure 12) indicated that *Odor* was the most influential input, followed by *CapColor*, *CapShape*, and *CapSurface*. This simplification can enhance the model's interpretability and reduce the risk of overestimating the importance of influential inputs.

Conclusion

The optimised RF model is considered superior to the optimised DT model due to its robustness to changes in data and slightly higher accuracy. Despite the high accuracy of all models (over 99%), the potential consequences of misclassifying a poisonous mushroom as edible underscore the importance of striving for the highest possible accuracy. Therefore, while both the basic RF, optimised DT and RF models are highly accurate, the optimised RF model is recommended for its better overall performance and robustness.