Assignment III: Can I eat that mushroom?

Sungyeon Lim

Data

The dataset includes six characteristics describing each mushroom: Edible, CapSahpe, CapSurface, CapColor, Odor, and Height. All the variables are categorical, as their values are in classes. In total, the dataset contains 8,124 entries representing 8,124 different mushrooms.

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

```
##      Edible CapShape CapSurface CapColor    Odor Height
## 1 Poisonous   Convex     Smooth    Brown Pungent   Tall
## 2    Edible   Convex     Smooth   Yellow  Almond  Short
## 3    Edible     Bell     Smooth    White   Anise   Tall
## 4 Poisonous   Convex      Scaly    White Pungent  Short
## 5    Edible   Convex     Smooth     Gray    None  Short
## 6    Edible   Convex      Scaly   Yellow  Almond  Short
```

```
nrow(mushrooms)
```

```
## [1] 8124
```

```
ncol(mushrooms)
```

```
## [1] 6
```

Task 1

First, the given dataset needs to be split into training and test datasets to train and evaluate decision tree and random forest models. In this case, the test dataset comprises 1,500 out of 8,124 entries, while the training dataset comprises 6,624 entries.

```
set.seed(22)
test_idx <- sample(dim(mushrooms)[1], 1500)
test_data <- mushrooms[test_idx, ]
train_data <- mushrooms[-test_idx, ]
```

Then, several models are trained to predict the variable Edible, which indicates whether mushrooms are edible. Initially, a decision tree model is trained using all five input variables. Cross-validation and hyperparameter tuning are carried out to find the optimal model. Grid search is performed for the complex parameter (cp), which is one of the decision tree's hyperparameters. After training decision tree models with different values of cp, it is found that the model is optimal when cp is set to 0.001.

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```r
tree_control <- trainControl(method="cv", number=10)
tree_grid <- expand.grid(cp=seq(0.001, 0.1, by=0.001))

tree_model <- train(Edible ~ ., data=train_data, method="rpart", trControl=tree_control, tuneGrid=tree_g
tree_model$bestTune
```

```
##      cp
## 1 0.001
```

Predicting the Edible of the test dataset using the previously trained model shows that 1,481 entries have been correctly predicted.

```r
tree_predictions <- predict(tree_model, newdata=test_data)
tree_accuracy <- sum(tree_predictions == test_data$Edible)
tree_accuracy
```

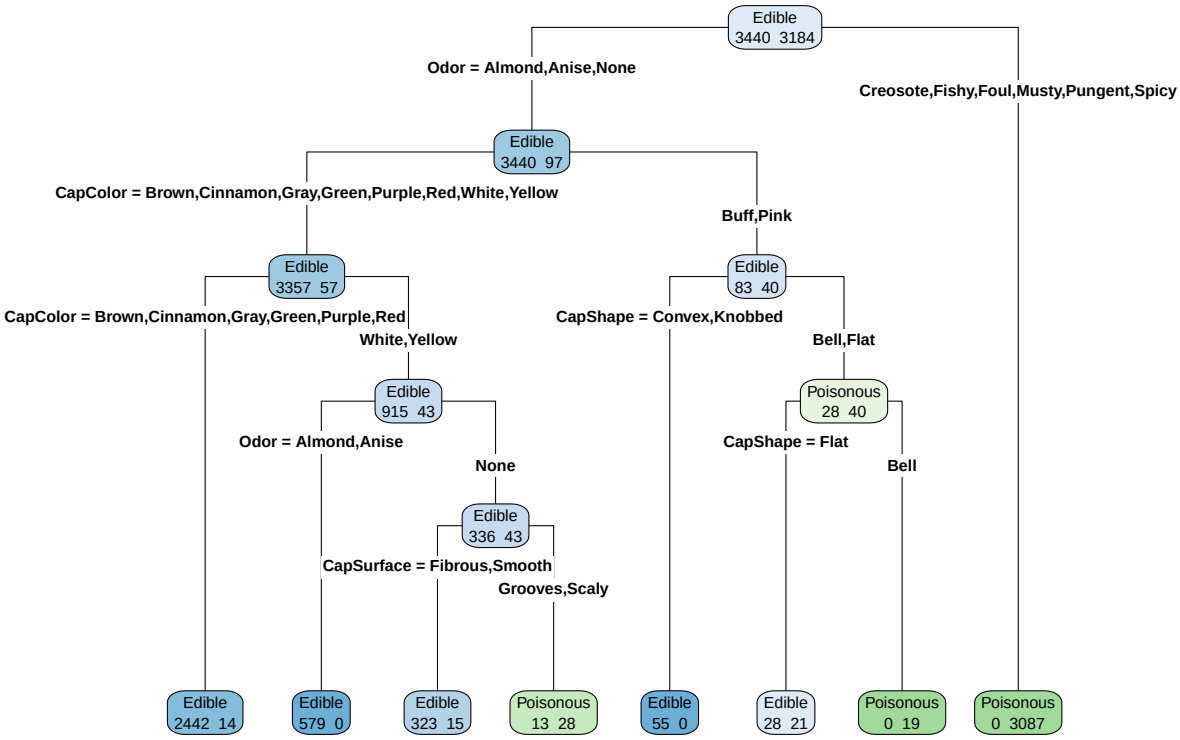
```
## [1] 1481
```

The trained decision tree is visualised, presenting a complex tree.

```r
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```r
tr_final <- rpart(Edible ~ ., data=train_data, control=rpart.control(cp=tree_model$bestTune$cp))
rpart.plot(tr_final, type=4, extra=1)
```



When the decision tree becomes more complex due to an increased number of variables, which can lead to overfitting, it is important to consider variable importance. Variable importance reflects the Gini importance scores for each variable. These scores, obtained from the trained model, show that Odor and CapColor have the greatest impacts on training the model. The model performance can be improved by choosing the two variables with the highest importance scores, reducing overfitting.

```
tr_final$variable.importance
```

```
##       Odor   CapColor   CapShape CapSurface     Height
## 3125.37298  446.96248  314.86229  280.42753    1.91296
```

Another decision tree model is trained using only two selected variables, Odor and CapColor, resulting in an optimal model when cp equals 0.001.

```
tree_control <- trainControl(method="cv", number=10)
tree_grid <- expand.grid(cp=seq(0.001, 0.1, by=0.001))

tree_model <- train(Edible ~ ., data=train_data[c(-2, -3, -6)], method="rpart", trControl=tree_control,

tree_model$bestTune
```

```
##      cp
## 1 0.001
```

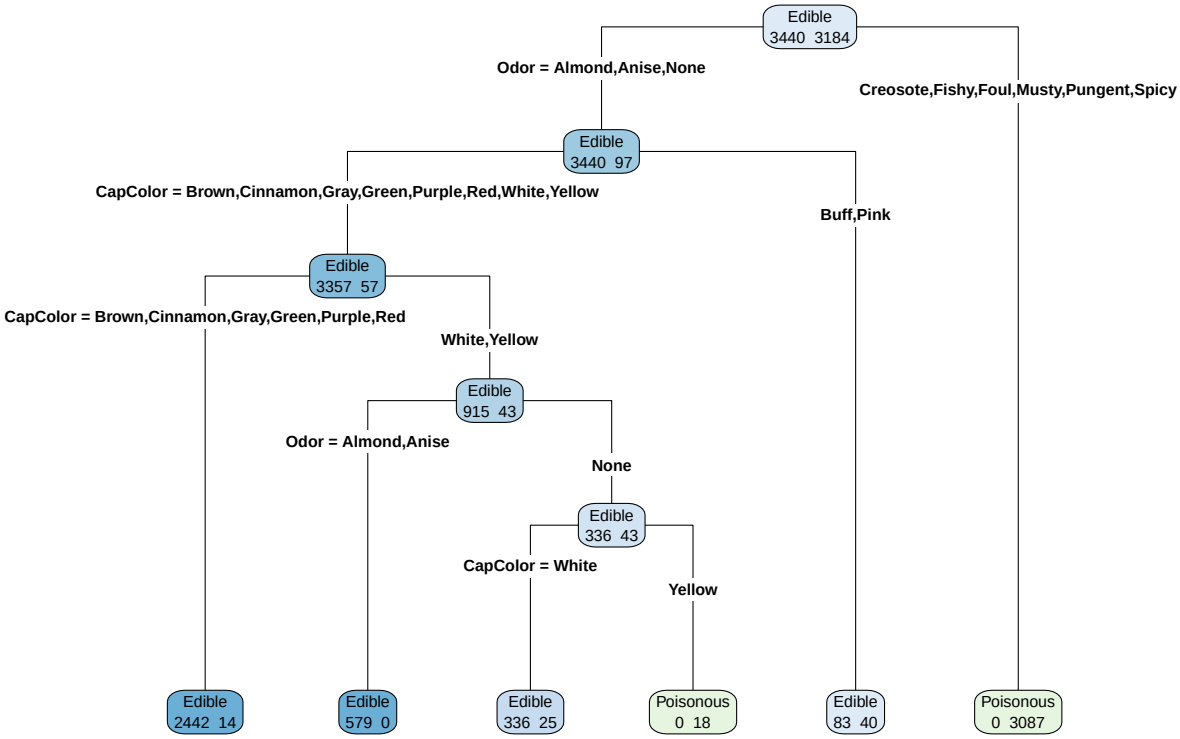
Predicting the Edible of the test dataset using the trained model correctly predicts 1,483 entries. This suggests that the model has been improved and has resolved the overfitting problem.

```
tree_predictions <- predict(tree_model, newdata=test_data[c(-2, -3, -6)])
tree_accuracy <- sum(tree_predictions == test_data$Edible)
tree_accuracy
```

```
## [1] 1483
```

The trained decision tree is visualised, presenting a simpler and less complicated tree.

```
tr_final <- rpart(Edible ~ ., data=train_data[c(-2, -3, -6)], control=rpart.control(cp=tree_model$bestTu
rpart.plot(tr_final, type=4, extra=1)
```



Next, a random forest model is trained using all five input variables. Cross-validation and hyperparameter tuning are performed to find the best model. Grid search is conducted to optimise the mtry parameter, determining the number of randomly selected variables. After training random forest models with different mtry values, it is found that the optimal model is achieved when mtry is set to 5.

```
rf_control <- trainControl(method="cv", number=10)
rf_grid <- expand.grid(mtry=seq(1, ncol(mushrooms)-1, by=1))

rf_model <- train(Edible ~ ., data=train_data, method="rf", trControl=rf_control, tuneGrid=rf_grid)

rf_model$bestTune
```

```
##   mtry
## 5    5
```

The results indicate that as the value of mtry increases, the model's accuracy also increases.

```
rf_model
```

```
## Random Forest
##
## 6624 samples
##    5 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5961, 5962, 5962, 5962, 5961, 5962, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.7821518  0.5559903
##   2     0.9664853  0.9329014
##   3     0.9818863  0.9636883
##   4     0.9861118  0.9721597
##   5     0.9877730  0.9754860
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

Predicting the Edible of the test dataset using the trained model indicates that 1,483 entries are correctly predicted. This is the same number of entries correctly predicted by the decision tree model with the two selected variables. This suggests that both the decision tree and random forest models perform well in this case.

```
rf_predictions <- predict(rf_model, newdata=test_data)
rf_accuracy <- sum(rf_predictions == test_data$Edible)
rf_accuracy
```

```
## [1] 1483
```

Task 2

In order to compare the performance of the decision tree and random forest models, cross-validation is used. The accuracies for the optimal decision tree and random forest models selected from task 1 are determined by averaging 10 accuracy scores from cross-validation. The accuracy of the decision tree model is 0.98807, while the random forest model has an accuracy of 0.98823. The difference in accuracy is slight, differing only at the 4th decimal place.

```
set.seed(22)
train_control <- trainControl(method="cv", number=10, savePredictions=TRUE)

dt_cv <- train(Edible ~ ., data=train_data[c(-2, -3, -6)], method="rpart", trControl=train_control, tun

dt_acc <- dt_cv$results$Accuracy
print(paste("Decision Tree Accuracy: ", dt_acc))
```

```
## [1] "Decision Tree Accuracy:  0.988074895307879"
```

```
set.seed(22)
train_control <- trainControl(method="cv", number=10, savePredictions=TRUE)

rf_cv <- train(Edible ~ ., data=train_data, method="rf", trControl=train_control, tuneGrid=expand.grid(
rf_acc <- rf_cv$results$Accuracy
print(paste("Random Forest Accuracy: ", rf_acc))
```

```
## [1] "Random Forest Accuracy:  0.988225724870473"
```

A paired t-test is conducted to assess whether there is a significant difference in performance between these two models. The mean difference is estimated to be -0.00015, which is too small to conclude that the performance between the models is significant. Additionally, a confidence interval for the mean difference estimate is examined. Since the interval includes 0, it can be concluded that there is no significant difference in performance between the models.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
dt_preds <- dt_cv$pred
rf_preds <- rf_cv$pred

combined_preds <- inner_join(dt_preds, rf_preds, by=c("rowIndex", "obs"))

t_test_result <- t.test(combined_preds$pred.x==combined_preds$obs, combined_preds$pred.y==combined_preds
t_test_result$estimate
```

```
## mean difference
##   -0.0001509662
```

```
t_test_result$conf.int[1:2]
```

```
## [1] -0.0006635792  0.0003616468
```