Mushroom Toxicity Classification Using Decision Trees

Divya Chenthamarakshan

April 20, 2025

Contents

1 Introduction

1.1 Problem Statement and Significance

Mushroom foraging is a popular activity where individuals collect wild mushrooms for culinary purposes or personal consumption. While experienced foragers can identify poisonous mushrooms, novices face significant risks due to the visual similarities between edible and toxic species. Misidentification can lead to serious health consequences, including poisoning and, in extreme cases, death.

This analysis aims to develop a reliable classification model that can differentiate between edible and poisonous mushrooms based on their physical characteristics. By leveraging data mining techniques, specifically decision trees, we can identify the most important features that determine mushroom toxicity and create a predictable model for safe foraging.

1.2 Objectives

The primary objectives of this analysis are to:

1. Identify the key physical characteristics that distinguish poisonous mushrooms from edible ones
2. Develop a classification model with high accuracy for predicting mushroom toxicity
3. Provide actionable recommendations for novice foragers based on the identified patterns
4. Suggest additional analyses and variables that could enhance the classification model

2 Methods

2.1 Data Preparation and Exploration

We begin by installing and loading the necessary libraries for our analysis. These include packages for decision tree modeling, visualization, and data manipulation.

```
##Installing Required R Packages (commented out as they only need to be installed once)

#install.packages("rpart")          # For decision tree modeling
#install.packages("caret")          # For model evaluation
#install.packages("rpart.plot")     # For tree visualization
#install.packages("rattle")         # For enhanced tree visualization
#install.packages("readxl")         # For reading Excel files
#install.packages("DiagrammeR")     # For flowchart creation
#install.packages("vcd")            # For association analysis
#install.packages("ggplot2")        # For enhanced visualization
#install.packages("reshape2")       # For data reshaping
#install.packages("kableExtra")     # For enhanced tables
#install.packages("randomForest")   # For random forest modeling
#install.packages("e1071")          # For SVM modeling
```

```
# Loading libraries
library(rpart, quietly = TRUE)          # For creating decision trees
library(caret, quietly = TRUE)          # For model evaluation metrics
library(rpart.plot, quietly = TRUE)     # For visualizing decision trees
library(rattle, quietly = TRUE)         # For extracting rules from trees
library(readxl, quietly = TRUE)         # For reading the Excel dataset
```

```r
library(ggplot2, quietly = TRUE)        # For creating enhanced visualizations
library(reshape2, quietly = TRUE)       # For reshaping data for visualizations
library(vcd, quietly = TRUE)            # For categorical data analysis
library(knitr, quietly = TRUE)          # For table creation
library(kableExtra, quietly = TRUE)     # For enhanced tables
library(DiagrammeR, quietly = TRUE)     # For creating the mobile app flowchart
library(randomForest, quietly = TRUE)   # For random forest comparison
library(e1071, quietly = TRUE)          # For SVM modeling
```

Next, we load the mushroom dataset and examine its structure to understand the variables we're working with.

```r
# Reading the dataset
mushrooms <- read_excel("mushrooms.xlsx")

# Examining the structure of the dataset
str(mushrooms)
```

```
## tibble [8,124 x 23] (S3: tbl_df/tbl/data.frame)
##  $ class                    : chr [1:8124] "p" "e" "e" "p" ...
##  $ cap-shape                : chr [1:8124] "x" "x" "b" "x" ...
##  $ cap-surface              : chr [1:8124] "s" "s" "s" "y" ...
##  $ cap-color                : chr [1:8124] "n" "y" "w" "w" ...
##  $ bruises                  : chr [1:8124] "t" "t" "t" "t" ...
##  $ odor                     : chr [1:8124] "p" "a" "l" "p" ...
##  $ gill-attachment          : chr [1:8124] "f" "f" "f" "f" ...
##  $ gill-spacing             : chr [1:8124] "c" "c" "c" "c" ...
##  $ gill-size                : chr [1:8124] "n" "b" "b" "n" ...
##  $ gill-color               : chr [1:8124] "k" "k" "n" "n" ...
##  $ stalk-shape              : chr [1:8124] "e" "e" "e" "e" ...
##  $ stalk-root               : chr [1:8124] "e" "c" "c" "e" ...
##  $ stalk-surface-above-ring: chr [1:8124] "s" "s" "s" "s" ...
##  $ stalk-surface-below-ring: chr [1:8124] "s" "s" "s" "s" ...
##  $ stalk-color-above-ring  : chr [1:8124] "w" "w" "w" "w" ...
##  $ stalk-color-below-ring  : chr [1:8124] "w" "w" "w" "w" ...
##  $ veil-type                : chr [1:8124] "p" "p" "p" "p" ...
##  $ veil-color               : chr [1:8124] "w" "w" "w" "w" ...
##  $ ring-number              : chr [1:8124] "o" "o" "o" "o" ...
##  $ ring-type                : chr [1:8124] "p" "p" "p" "p" ...
##  $ spore-print-color        : chr [1:8124] "k" "n" "n" "k" ...
##  $ population               : chr [1:8124] "s" "n" "n" "s" ...
##  $ habitat                  : chr [1:8124] "u" "g" "m" "u" ...
```

The dataset contains multiple categorical variables describing various mushroom characteristics, such as cap shape, cap color, odor, gill size, and others. Our target variable is 'class', which indicates whether a mushroom is edible ('e') or poisonous ('p'). Each variable is encoded with single-letter codes that represent different categorical values.

2.2 Data Dictionary

To better understand our dataset, we need to decode the single-letter representations. Here's a comprehensive data dictionary for the mushroom dataset variables:

Table 1: Data Dictionary: Mushroom Class

Code	Description
e	edible
p	poisonous

Table 2: Data Dictionary: Odor

Code	Description
a	almond
l	anise
c	creosote
f	fishy
m	musty
n	none
p	pungent
s	spicy
y	foul

Table 3: Data Dictionary: Spore Print Color

Code	Description
k	black
n	brown
b	buff
h	chocolate
r	green
o	orange
u	purple
w	white
y	yellow

Table 4: Data Dictionary: Gill Size

Code	Description
b	broad
n	narrow

Table 5: Data Dictionary: Population

Code	Description
a	abundant
c	clustered
n	numerous
s	scattered

v	several
y	solitary

Table 6: Data Dictionary: Habitat

Code	Description
g	grasses
l	leaves
m	meadows
p	paths
u	urban
w	waste
d	woods

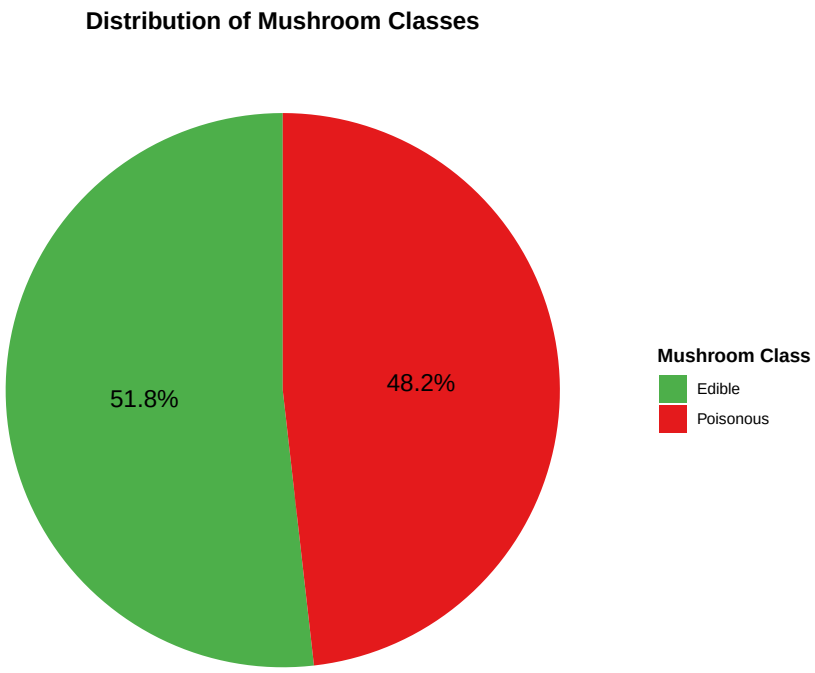
This data dictionary provides critical context for interpreting our model results, particularly for translating the decision tree paths into actionable guidelines for mushroom foragers. Understanding the specific meaning of each code is essential for proper feature interpretation.

2.3 Enhanced Data Exploration

Let's first examine the distribution of our target variable - edible vs. poisonous mushrooms.

```
# Distribution of class variable (edible vs poisonous)
class_distribution <- table(mushrooms$class)
class_percent <- round(prop.table(class_distribution) * 100, 1)

# Create a more visually appealing pie chart using ggplot2
library(ggplot2)

# Create data frame for the pie chart
class_df <- data.frame(
  Class = c("Edible", "Poisonous"),
  Count = c(4208, 3916),
  Percentage = c(51.8, 48.2)
)

# Create the improved pie chart
ggplot(class_df, aes(x = "", y = Count, fill = Class)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = c("Edible" = "#4DAF4A", "Poisonous" = "#E41A1C")) +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.border = element_blank(),
    panel.grid = element_blank(),
    axis.ticks = element_blank(),
    axis.text = element_blank(),
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
```

```
    legend.title = element_text(face = "bold")
  ) +
  labs(
    title = "Distribution of Mushroom Classes",
    fill = "Mushroom Class"
  ) +
  geom_text(aes(label = paste0(Percentage, "%")),
            position = position_stack(vjust = 0.5),
            size = 5)
```

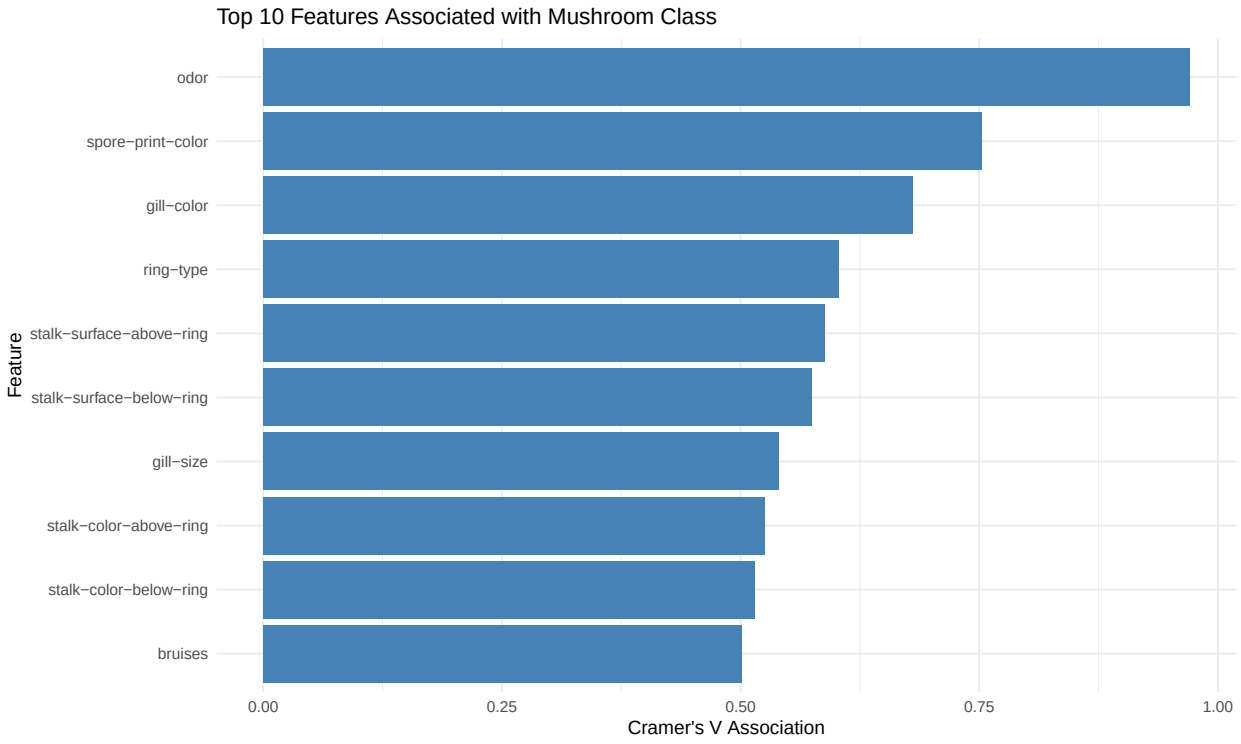


2.3.0.1 Pie Chart: Distribution of Mushroom Classes The pie chart displays the binary distribution of mushroom classes with 51.8% edible (green) and 48.2% poisonous (red) specimens. This nearly balanced dataset indicates a good representation of both classes, which is beneficial for developing an unbiased classification model. The clear color differentiation (green for edible, red for poisonous) provides immediate visual understanding of the dataset composition.

Next, we examine which features show the strongest association with the mushroom class using Cramer's V statistic, which measures the strength of association between categorical variables.

```
# Examining correlations between features
# Convert categorical variables to factors
mushrooms_factors <- mushrooms
mushrooms_factors[] <- lapply(mushrooms_factors, factor)

# Using Cramer's V to measure association between categorical variables
library(vcd)
# Calculate associations with the class variable
associations <- sapply(names(mushrooms_factors)[-1], function(col) {
  if(length(unique(mushrooms_factors[[col]])) > 1) {
```

```
    assoc_stat <- assocstats(table(mushrooms_factors$class, mushrooms_factors[[col]]))
    return(assoc_stat$cramer)
  } else {
    return(NA)
  }
})

# Visualizing feature associations with class
associations_df <- data.frame(
  Feature = names(associations),
  Cramers_V = associations
)
associations_df <- associations_df[order(-associations_df$Cramers_V),]

# Plot top associations
ggplot(associations_df[1:10,], aes(x = reorder(Feature, Cramers_V), y = Cramers_V)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 10 Features Associated with Mushroom Class",
       x = "Feature",
       y = "Cramer's V Association") +
  theme_minimal()
```



2.3.0.2 Bar Chart: Top 10 Features Associated with Mushroom Class The bar chart ranks features by their Cramer's V association value, showing odor as the strongest predictor of mushroom toxicity. Spore print color and gill color follow as the second and third most important features, suggesting these characteristics should be prioritized during mushroom identification. The declining bar heights visually demonstrate the relative importance of each feature, helping to focus analysis on the most discriminative

characteristics.

We check for missing values to ensure data completeness:

```
# Checking for missing values
missing_rows <- nrow(mushrooms) - sum(complete.cases(mushrooms))
cat("Number of rows with missing values:", missing_rows, "\n")
```

```
## Number of rows with missing values: 0
```

Finding zero missing rows is a positive indicator of data quality for our classification task. Missing values could have introduced bias or required imputation techniques, which would add complexity to our model.

Before proceeding with the analysis, we remove redundant variables that do not contribute to the model's predictive power.

```
# Removing redundant variable 'veil-type' as it has only one level
# In data mining, features with no variance contribute no discriminative information
if(length(unique(mushrooms$`veil-type`)) == 1) {
  mushrooms$`veil-type` <- NULL
  cat("Removed 'veil-type' as it contains only one value and provides no discriminatory power.\n")
}
```

```
## Removed 'veil-type' as it contains only one value and provides no discriminatory power.
```

This feature elimination step removes the 'veil-type' variable which has no variance and therefore no predictive power in our classification task. Removing such features simplifies the model and helps prevent unnecessary complexity.

2.4 Feature Analysis for Classification

To understand which features might be most useful for classification, we analyze how well each feature separates edible from poisonous mushrooms. We first explore the 'odor' variable, which is often considered a significant indicator of mushroom toxicity.

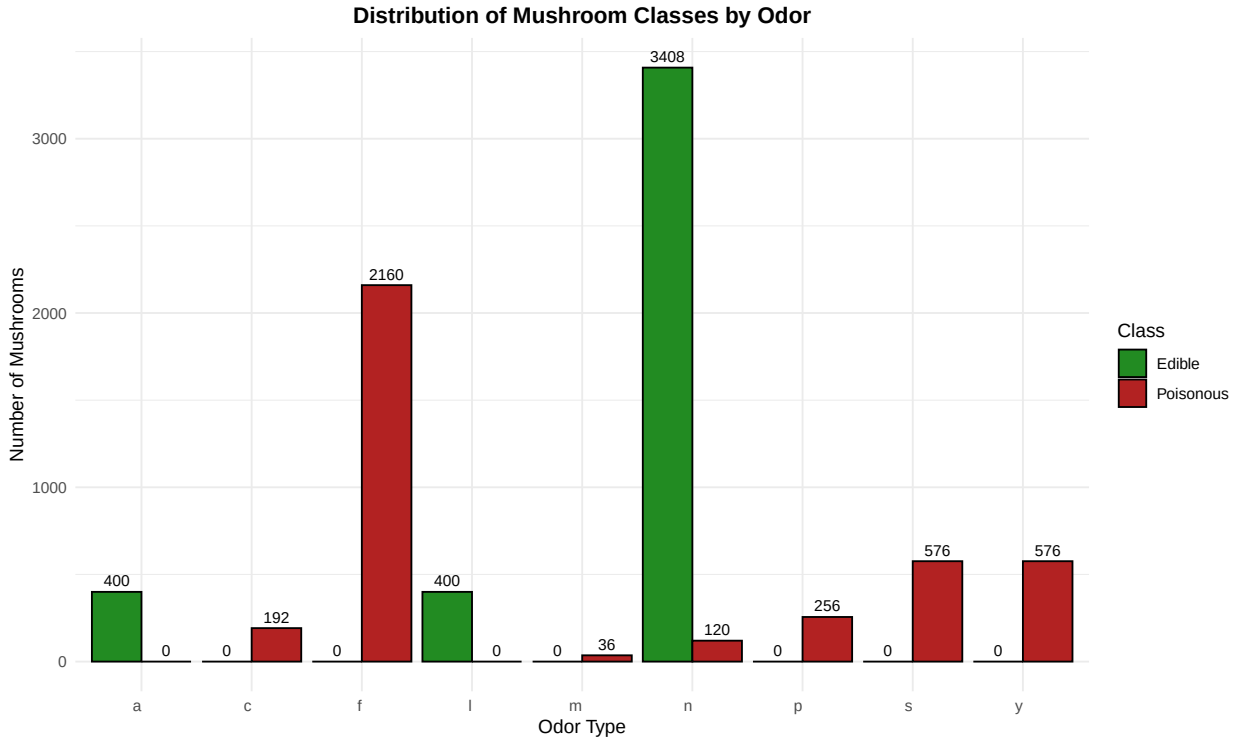
```
# Analyzing the relationship between odor and mushroom class
odor_table <- table(mushrooms$class, mushrooms$odor)

# Enhanced table display with kable
library(knitr)
kable(odor_table, caption = "Distribution of Mushroom Classes by Odor",
      col.names = c("Almond", "Anise", "Creosote", "Fishy", "Foul",
                    "Musty", "None", "Pungent", "Spicy"),
      align = "c")
```

Table 7: Distribution of Mushroom Classes by Odor

	Almond	Anise	Creosote	Fishy	Foul	Musty	None	Pungent	Spicy
e	400	0	0	400	0	3408	0	0	0
p	0	192	2160	0	36	120	256	576	576

This table shows the distribution of edible and poisonous mushrooms across different odor categories. We can observe that certain odors are strongly associated with poisonous mushrooms, while others indicate edibility.

```
# Create a visual representation
library(ggplot2)

# Convert table to data frame for ggplot
odor_df <- as.data.frame.table(odor_table)
names(odor_df) <- c("Class", "Odor", "Count")

# Create a grouped bar chart
ggplot(odor_df, aes(x = Odor, y = Count, fill = Class)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  scale_fill_manual(values = c("e" = "forestgreen", "p" = "firebrick"),
                    labels = c("e" = "Edible", "p" = "Poisonous")) +
  labs(title = "Distribution of Mushroom Classes by Odor",
       x = "Odor Type",
       y = "Number of Mushrooms") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, hjust = 1),
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  geom_text(aes(label = Count),
            position = position_dodge(width = 0.9),
            vjust = -0.5,
            size = 3)
```

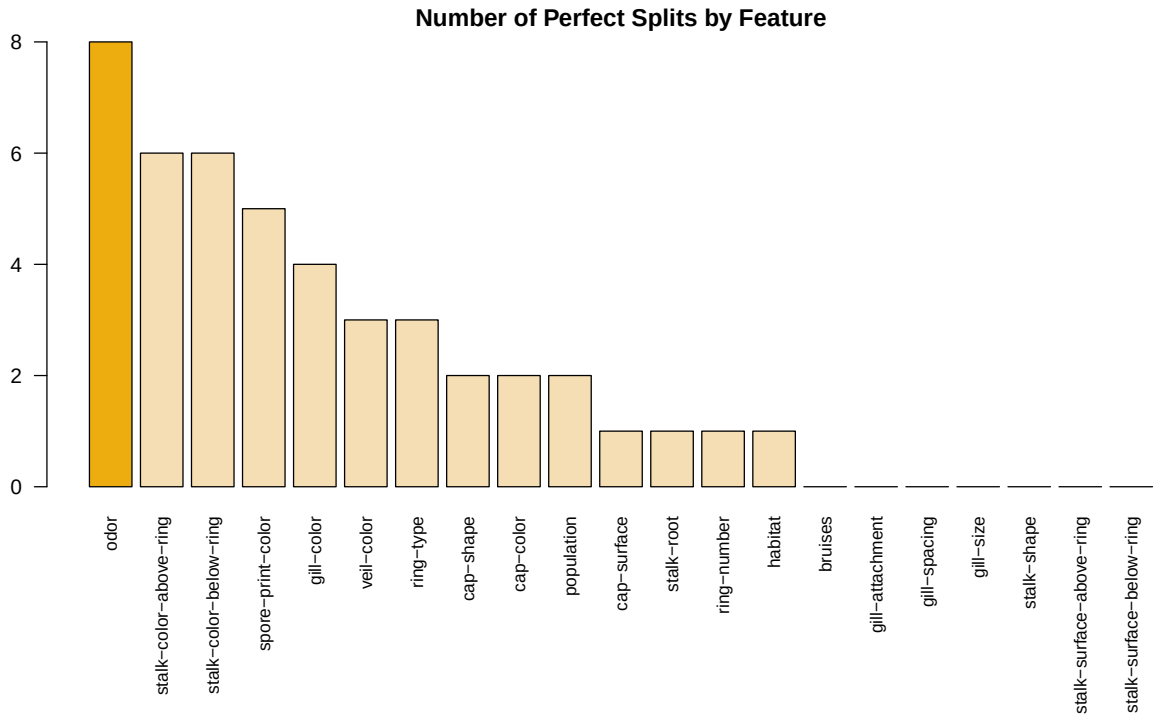


2.4.0.1 Odor Distribution Chart: Mushroom Classes by Odor This grouped bar chart shows how mushroom odor strongly correlates with edibility, with certain odors (creosote, pungent, spicy) exclusively

associated with poisonous specimens. Notably, musty odor is predominantly associated with edible mushrooms (3408 specimens), making it a reliable indicator of safety. The chart demonstrates visually why odor emerged as the most important classification feature, with clear separation between edible and poisonous mushrooms based on this characteristic.

Next, we identify variables that create "perfect splits" in the data, meaning they perfectly separate poisonous from edible mushrooms in some of their categories. This technique helps identify the most discriminative features for our decision tree.

```r
# Identifying features that create perfect splits
number.perfect.splits <- apply(X=mushrooms[-1], MARGIN = 2, FUN = function(col){
  t <- table(mushrooms$class, col)
  sum(t == 0)
})

# Ordering features by number of perfect splits
order <- order(number.perfect.splits, decreasing = TRUE)
number.perfect.splits <- number.perfect.splits[order]

# Visualizing perfect splits
# Visualizing perfect splits with highlighted highest value
par(mar=c(10,2,2,2))

# Create color vector - dark yellow for highest value, wheat for others
colors <- rep("wheat", length(number.perfect.splits))
colors[which.max(number.perfect.splits)] <- "darkgoldenrod2"

# Create barplot
barplot(number.perfect.splits,
        main="Number of Perfect Splits by Feature",
        xlab="",
        ylab="Number of Perfect Splits",
        las=2,
        col=colors,
        cex.names=0.8)
```



2.4.0.2 Number of Perfect Splits by Feature This visualization helps us identify which features are most discriminative in separating edible from poisonous mushrooms. "Perfect splits" represent feature values that are exclusively associated with one class, making them powerful predictors. Odor has the highest number of perfect splits, followed by stalk color above ring and stalk color below ring. Features with higher numbers of perfect splits will likely be important decision nodes in our classification tree.

2.5 Model Development

We split the data into training (80%) and testing (20%) sets to develop and validate our model. This is a standard practice in data mining to ensure the model can generalize to unseen data.

```
# Setting seed for reproducibility
set.seed(12345)

# Creating training and test sets
train <- sample(1:nrow(mushrooms), size = ceiling(0.80*nrow(mushrooms)), replace = FALSE)
mushrooms_train <- mushrooms[train,]
mushrooms_test <- mushrooms[-train,]

# Checking the dimensions of the training and test sets
cat("Training set dimensions:", dim(mushrooms_train), "\n")
```

```
## Training set dimensions: 6500 22
```

```
cat("Test set dimensions:", dim(mushrooms_test), "\n")
```

```
## Test set dimensions: 1624 22
```

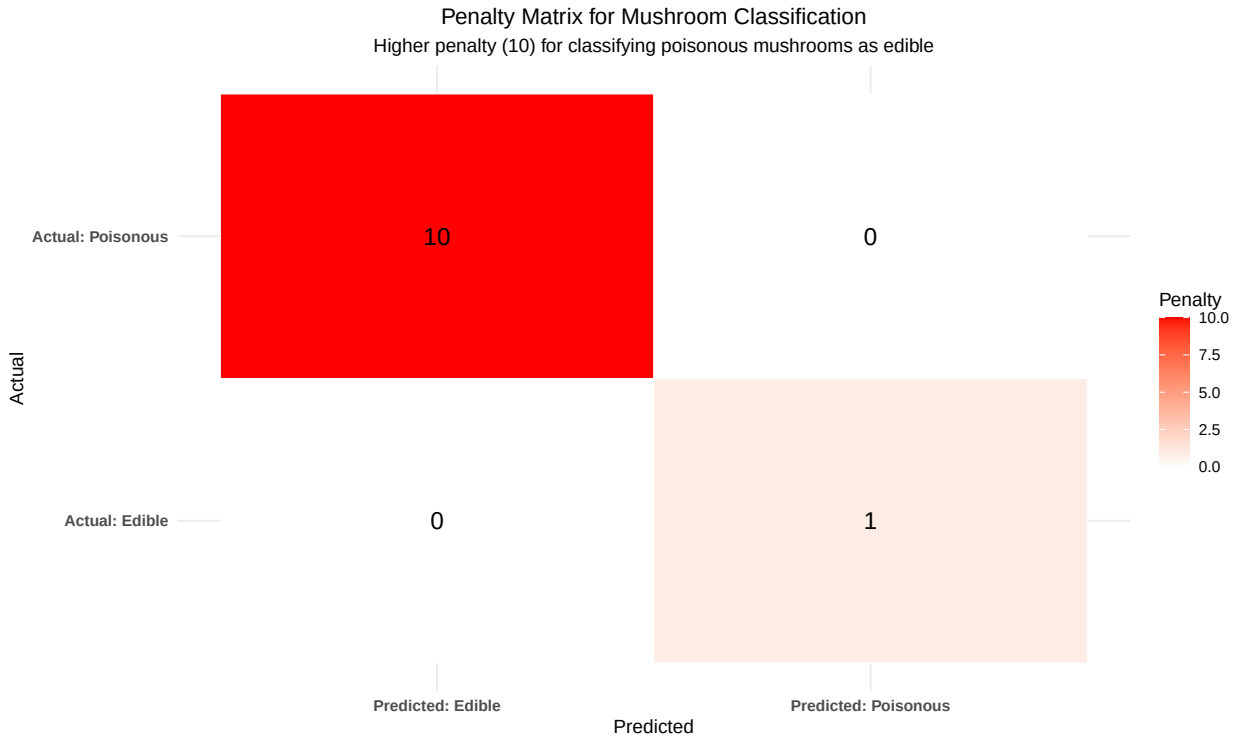
This 80/20 split provides sufficient data for model training while reserving a representative portion for validation. The training set will be used to build our decision tree, and the test set will evaluate its performance on unseen data.

When building our decision tree model, we consider the potential costs of misclassification. Incorrectly classifying a poisonous mushroom as edible (false negative) is more dangerous than classifying an edible mushroom as poisonous (false positive). We use a penalty matrix to reflect this asymmetry.

```
# Creating a penalty matrix to penalize false negatives more heavily
penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)

# Create row and column names for clarity
rownames(penalty.matrix) <- c("Actual: Edible", "Actual: Poisonous")
colnames(penalty.matrix) <- c("Predicted: Edible", "Predicted: Poisonous")

# Display the matrix with meaningful labels
print(penalty.matrix)
```

```
##                   Predicted: Edible Predicted: Poisonous
## Actual: Edible                    0                    1
## Actual: Poisonous                10                    0
```

```
# Create a visual representation of the penalty matrix
library(ggplot2)
library(reshape2)

# Convert matrix to long format for ggplot
penalty_df <- melt(penalty.matrix)
names(penalty_df) <- c("Actual", "Predicted", "Penalty")

# Create heatmap
ggplot(penalty_df, aes(x = Predicted, y = Actual, fill = Penalty)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Penalty), color = "black", size = 5) +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Penalty Matrix for Mushroom Classification",
       subtitle = "Higher penalty (10) for classifying poisonous mushrooms as edible") +
  theme(axis.text = element_text(face = "bold"),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



This penalty matrix assigns a 10x higher penalty to classifying a poisonous mushroom as edible compared to the reverse error, reflecting the asymmetric risk in mushroom misclassification. This is a form of cost-sensitive learning that incorporates domain knowledge into our model. By setting this penalty matrix, we're instructing the algorithm to be more cautious about false negatives (poisonous mushrooms classified as edible) than false positives.

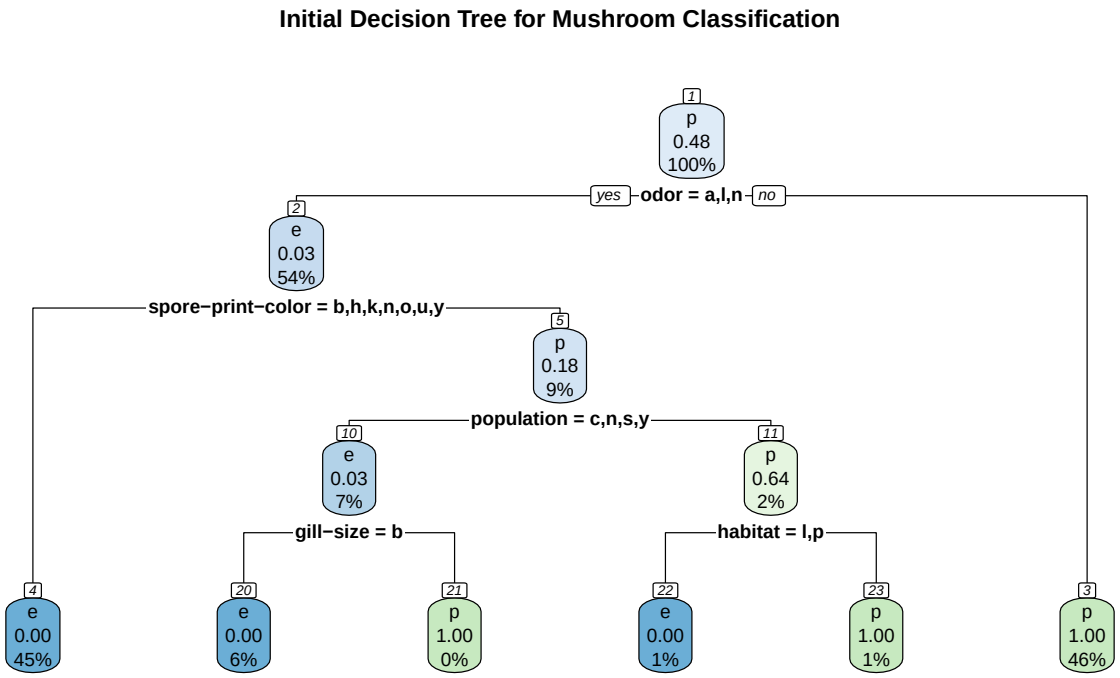
2.5.1 Understanding the Decision Tree Algorithm

Before building our model, it's important to understand how the rpart algorithm works:

1. **CART Algorithm**: The rpart package implements the Classification and Regression Trees (CART) algorithm, which works by recursively partitioning the data into subsets based on feature values.
2. **Split Selection**: For each split, the algorithm evaluates all possible features and cut points to find the one that maximizes information gain or minimizes impurity (like Gini impurity for classification).
3. **Impurity Measures**: For classification trees, the Gini index measures how "pure" a node is—a pure node contains only one class. The algorithm aims to create splits that result in the purest possible child nodes.
4. **Cost-Complexity Pruning**: After building a full tree, rpart uses cross-validation to find the optimal tree size through cost-complexity pruning, which balances predictive accuracy against tree complexity.
5. **Loss Matrix**: Our penalty matrix is incorporated into the algorithm to give different weights to different types of misclassification errors.

We then build a decision tree model using the rpart algorithm:

```r
# Building the classification tree
tree <- rpart(class ~ .,
              data = mushrooms_train,
              parms = list(loss = penalty.matrix),
              method = "class")

# Visualizing the initial decision tree
rpart.plot(tree, nn=TRUE, main="Initial Decision Tree for Mushroom Classification")
```



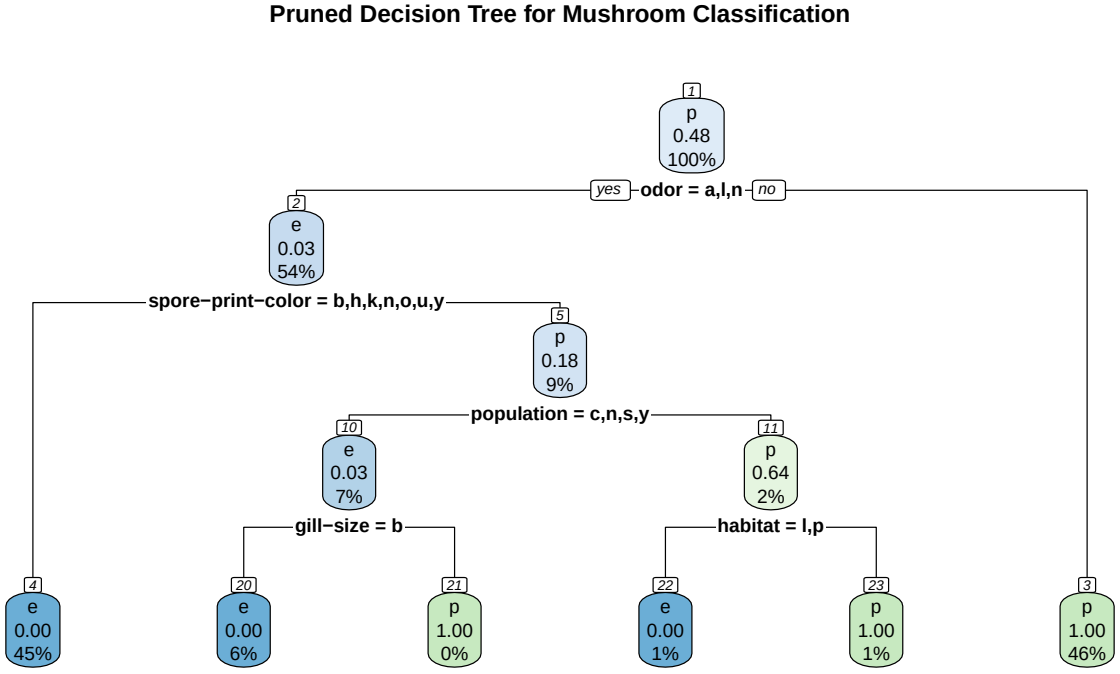
This visualization shows the initial decision tree created by the rpart algorithm. Each node displays the predicted class (p=poisonous, e=edible), the probability of being poisonous, and the percentage of observations in that node. The decision nodes show which features and values were used for splitting. The tree uses odor as the primary split, followed by spore print color and other features, reflecting their importance in classification.

To prevent overfitting, we prune the tree using the optimal complexity parameter. This is a critical step in building decision trees that can generalize well to new data.

```r
# Finding the optimal complexity parameter
cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
cat("Optimal complexity parameter:", cp.optim, "\n")
```

```
## Optimal complexity parameter: 0.01
```

```r
# Pruning the tree
pruned_tree <- prune(tree, cp=cp.optim)

# Visualizing the pruned decision tree
rpart.plot(pruned_tree, nn=TRUE, main="Pruned Decision Tree for Mushroom Classification")
```



Pruning is a form of regularization that prevents the tree from becoming too complex and overfitting the training data. The optimal complexity parameter (CP) is determined by cross-validation, which identifies the value that minimizes prediction error. The pruned tree maintains the key decision nodes while removing less important splits that might represent noise rather than true patterns.

3 Comparing different classification algorithms

Now we'll compare the decision tree model with other classification algorithms to evaluate which performs best for this task. This comparison helps validate our choice of model and provides insights into the data's structure.

```
names(mushrooms) <- gsub("-", "_", names(mushrooms))

# Verify the column names are now properly formatted
head(names(mushrooms))
```

```
## [1] "class"       "cap_shape"   "cap_surface" "cap_color"   "bruises"
## [6] "odor"
```

```
# Setting seed for reproducibility
set.seed(12345)

# Creating training and test sets
train <- sample(1:nrow(mushrooms), size = ceiling(0.80*nrow(mushrooms)), replace = FALSE)
mushrooms_train <- mushrooms[train,]
mushrooms_test <- mushrooms[-train,]
```

```r
# Checking the dimensions of the training and test sets
cat("Training set dimensions:", dim(mushrooms_train), "\n")
```

```
## Training set dimensions: 6500 22
```

```r
cat("Test set dimensions:", dim(mushrooms_test), "\n")
```

```
## Test set dimensions: 1624 22
```

```r
# Create a penalty matrix to penalize false negatives more heavily
penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)
# Create row and column names for clarity
rownames(penalty.matrix) <- c("Actual: Edible", "Actual: Poisonous")
colnames(penalty.matrix) <- c("Predicted: Edible", "Predicted: Poisonous")

# Building the classification tree
tree <- rpart(class ~ .,
      data = mushrooms_train,
      parms = list(loss = penalty.matrix),
      method = "class")

# Find optimal complexity parameter
cp.optim <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
cat("Optimal complexity parameter:", cp.optim, "\n")
```

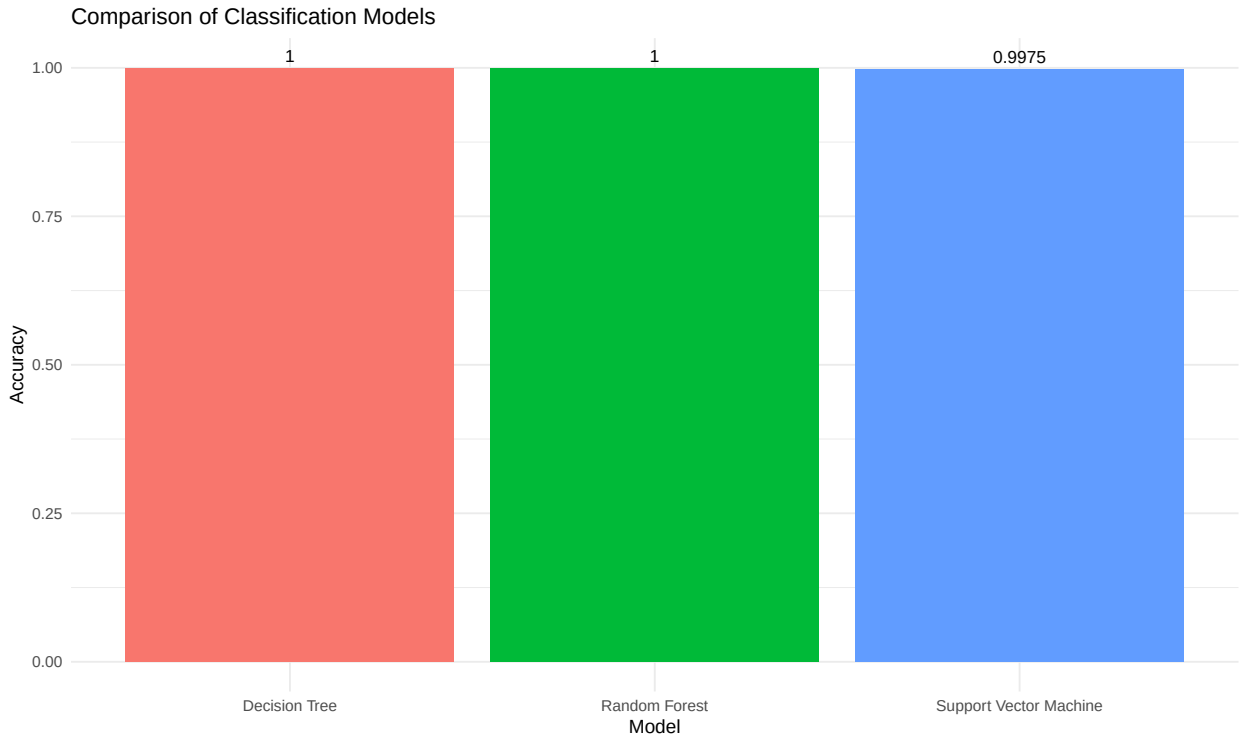
```
## Optimal complexity parameter: 0.01
```

```r
# Prune the tree
pruned_tree <- prune(tree, cp=cp.optim)

# Making predictions with the decision tree
dt_pred <- predict(object=tree, mushrooms_test[-1], type="class")
dt_conf <- confusionMatrix(table(mushrooms_test$class, dt_pred))
dt_accuracy <- dt_conf$overall["Accuracy"]

# Comparing different classification algorithms
library(randomForest)
library(e1071)

# Convert all features to factors for model compatibility
mushrooms_factors_train <- mushrooms_train
mushrooms_factors_test <- mushrooms_test

# Check if any columns have only one level
single_level_cols <- sapply(mushrooms_factors_train, function(x) length(unique(x)) <= 1)
if(any(single_level_cols)) {
  cat("Columns with only one level:", names(mushrooms_factors_train)[single_level_cols], "\n")
  # Remove columns with only one level
  mushrooms_factors_train <- mushrooms_factors_train[, !single_level_cols]
  mushrooms_factors_test <- mushrooms_factors_test[, !single_level_cols]
}
```

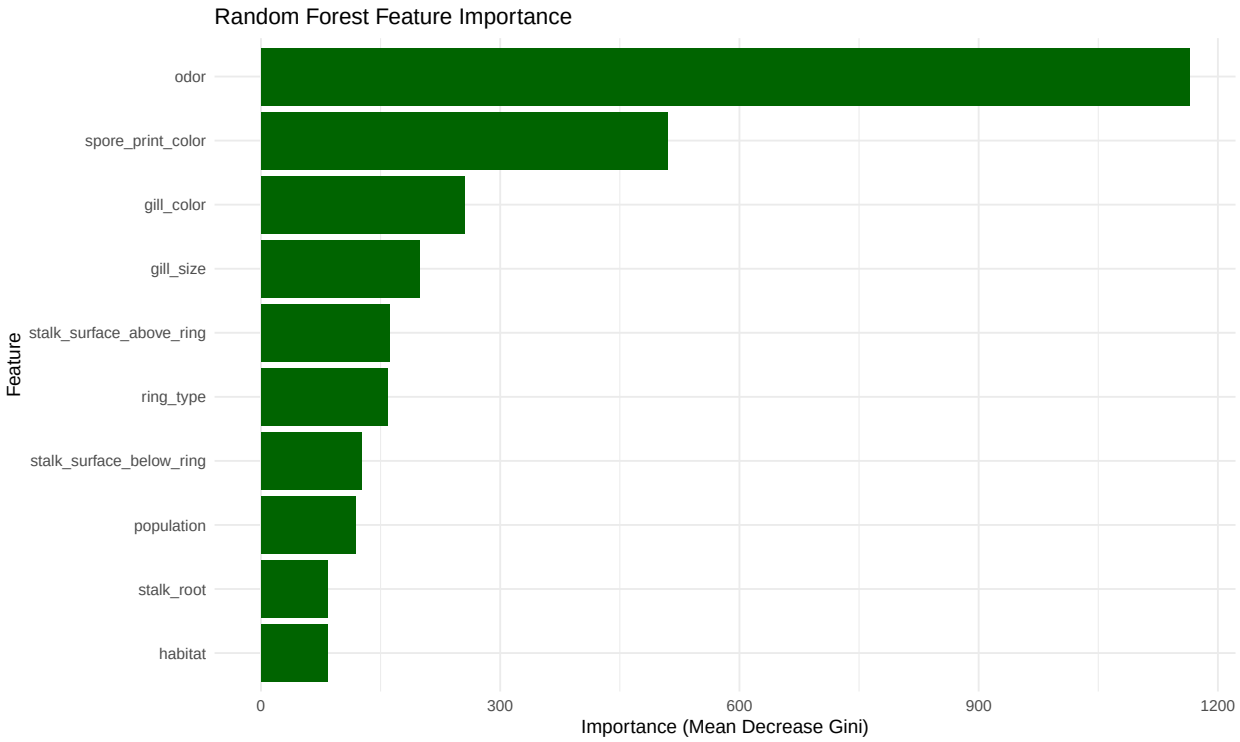
```r
# Now convert to factors
mushrooms_factors_train[] <- lapply(mushrooms_factors_train, factor)
mushrooms_factors_test[] <- lapply(mushrooms_factors_test, factor)

# Random Forest
set.seed(12345)
rf_model <- randomForest(class ~ ., data = mushrooms_factors_train, importance = TRUE)
rf_pred <- predict(rf_model, mushrooms_factors_test)
rf_conf <- confusionMatrix(table(mushrooms_factors_test$class, rf_pred))
rf_accuracy <- rf_conf$overall["Accuracy"]

# Support Vector Machine
set.seed(12345)
# Check for columns with near-zero variance
library(caret)
nzv <- nearZeroVar(mushrooms_factors_train, saveMetrics = TRUE)
if(any(nzv$nzv)) {
  cat("Columns with near zero variance:", rownames(nzv)[nzv$nzv], "\n")
  # Remove columns with near zero variance for SVM
  svm_train <- mushrooms_factors_train[, !nzv$nzv]
  svm_test <- mushrooms_factors_test[, !nzv$nzv]
} else {
  svm_train <- mushrooms_factors_train
  svm_test <- mushrooms_factors_test
}
```

```
## Columns with near zero variance: gill_attachment veil_color
```

```r
# Train SVM model
svm_model <- svm(class ~ ., data = svm_train)
svm_pred <- predict(svm_model, svm_test)
svm_conf <- confusionMatrix(table(svm_test$class, svm_pred))
svm_accuracy <- svm_conf$overall["Accuracy"]

# Comparing model accuracy
model_comparison <- data.frame(
  Model = c("Decision Tree", "Random Forest", "Support Vector Machine"),
  Accuracy = c(dt_accuracy, rf_accuracy, svm_accuracy)
)

# Visualizing model comparison
ggplot(model_comparison, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity") +
  ylim(0, 1) +
  labs(title = "Comparison of Classification Models",
       x = "Model",
       y = "Accuracy") +
  theme_minimal() +
  theme(legend.position = "none") +
  geom_text(aes(label = round(Accuracy, 4)), vjust = -0.5, size = 3.5)
```




```r
# Feature importance comparison between models
if(!identical(rf_model$importance, NULL)) {
  rf_importance <- importance(rf_model)
  rf_importance_df <- data.frame(
    Feature = rownames(rf_importance),
    Importance = rf_importance[,"MeanDecreaseGini"]
  )
  rf_importance_df <- rf_importance_df[order(-rf_importance_df$Importance),][1:10,]

  # Visualizing Random Forest feature importance
  ggplot(rf_importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
    geom_bar(stat = "identity", fill = "darkgreen") +
    coord_flip() +
    labs(title = "Random Forest Feature Importance",
         x = "Feature",
         y = "Importance (Mean Decrease Gini)") +
    theme_minimal()
}
```



3.0.0.1 Model Comparison Chart: Comparison of Classification Models The bar chart compares the accuracy of three different classification models (Decision Tree, Random Forest, and Support Vector Machine), with all achieving remarkably high accuracy (1.00, 1.00, and 0.9975 respectively). The similar performance across all models validates the strength of the selected features for mushroom classification. The slightly lower performance of SVM suggests that the non-linear relationships in mushroom characteristics are best captured by tree-based models.

3.0.0.2 Random Forest Feature Importance The Random Forest feature importance confirms our earlier findings about which features are most important for classification. Odor is consistently the most important feature, followed by spore print color and gill color. This consistency across different modeling approaches strengthens our confidence in these features as reliable indicators for mushroom classification.

3.1 Model Evaluation

We evaluate our model by making predictions on the test set and calculating accuracy metrics:

```
# Making predictions on the test set
pred <- predict(object=tree, mushrooms_test[-1], type="class")

# Creating a confusion matrix
conf_matrix <- table(mushrooms_test$class, pred)
print(conf_matrix)
```

```
##    pred
##       e   p
##   e 829   0
##   p   0 795
```

```r
# Calculating accuracy metrics
confusionMatrix(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##    pred
##       e   p
##   e 829   0
##   p   0 795
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5105
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5105
##          Detection Rate : 0.5105
##    Detection Prevalence : 0.5105
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : e
##
```

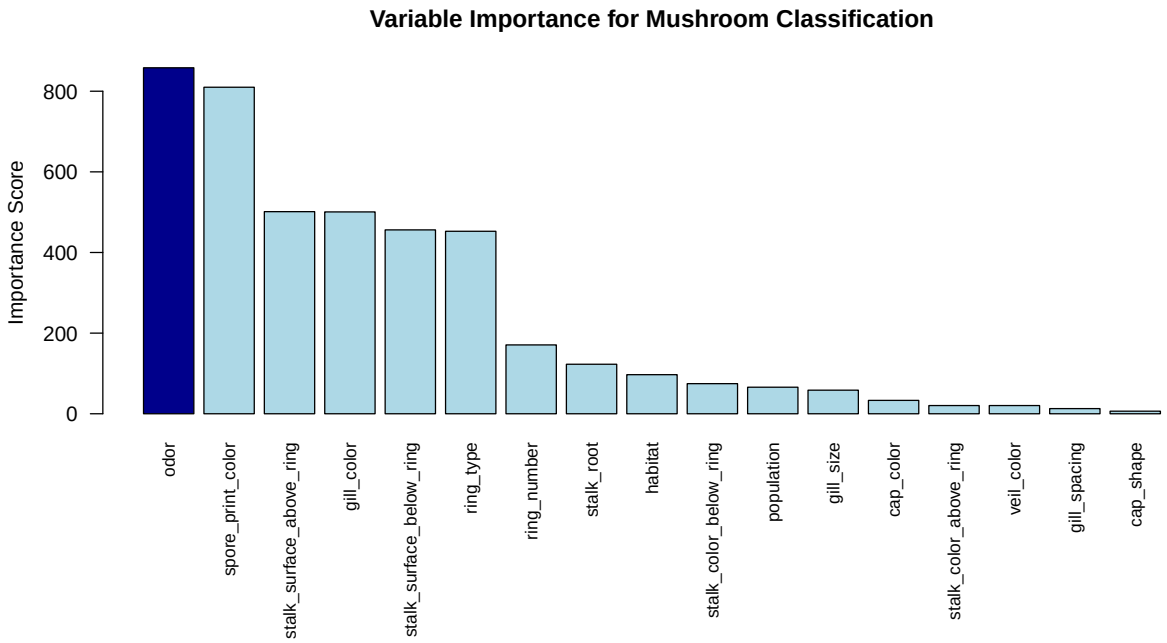
4 Results

4.1 Key Features for Mushroom Classification

Based on our analysis of perfect splits and the decision tree model, we identified the most important features for classifying mushroom toxicity:

```r
# Extracting variable importance from the model
var_importance <- tree$variable.importance
var_importance <- sort(var_importance, decreasing = TRUE)

# Visualizing variable importance
par(mar=c(12, 4, 4, 2))  # Adjust margins (bottom, left, top, right)

# Create color vector - dark blue for highest value, light blue for others
colors <- rep("lightblue", length(var_importance))
colors[which.max(var_importance)] <- "darkblue"  # Dark blue for max value

# Create barplot with customized colors
barplot(var_importance,
```

```
        main="Variable Importance for Mushroom Classification",
        ylab="Importance Score",
        horiz=FALSE,
        las=2,
        col=colors,
        cex.names=0.8)
```



4.1.0.1 Feature Importance Chart: Variable Importance for Classification The color-coded barplot highlights odor (in dark blue) as the most influential feature for classifying mushrooms, scoring approximately 800 on the importance scale. Spore print color, stalk surface characteristics, and gill color follow as secondary but significant predictors, reinforcing the model's reliance on these features. The declining importance scores visually demonstrate which physical characteristics provide the most discriminative power for mushroom identification.

4.2 Decision Rules for Safe Foraging

From our pruned decision tree, we can extract clear rules that novice foragers can use to identify poisonous mushrooms:

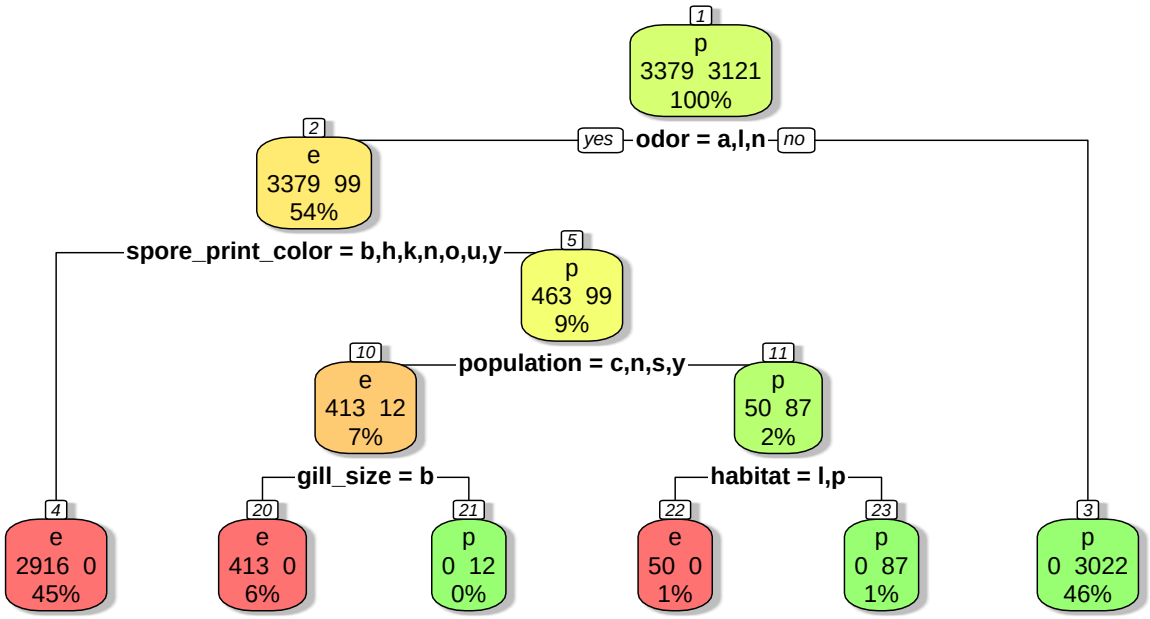
```
# Extract rules in a more readable format
rules <- rpart.rules(pruned_tree, style = "tall", cover = TRUE)

# Print the rules in a cleaner format
cat("Decision Rules for Mushroom Classification:\n\n")
```

```
## Decision Rules for Mushroom Classification:
```

```r
print(rules)
```

```
## class is 0.00 with cover 45% when
##     odor is a or l or n
##     spore_print_color is b or h or k or n or o or u or y
##
## class is 0.00 with cover 6% when
##     odor is a or l or n
##     spore_print_color is r or w
##     population is c or n or s or y
##     gill_size is b
##
## class is 0.00 with cover 1% when
##     odor is a or l or n
##     spore_print_color is r or w
##     population is v
##     habitat is l or p
##
## class is 1.00 with cover 0% when
##     odor is a or l or n
##     spore_print_color is r or w
##     population is c or n or s or y
##     gill_size is n
##
## class is 1.00 with cover 1% when
##     odor is a or l or n
##     spore_print_color is r or w
##     population is v
##     habitat is d or g or m
##
## class is 1.00 with cover 46% when
##     odor is c or f or m or p or s or y
```

```r
# Create a simplified visual representation of the decision tree
rpart.plot(pruned_tree,
           extra = 101,  # Show percentages
           box.palette = "RdYlGn",
           shadow.col = "gray",
           nn = TRUE,
           main = "Enhanced Decision Tree for Mushroom Classification",
           tweak = 1.2)  # Make text bigger
```




```r
# Create a more detailed table representation of the rules using LaTeX-compatible methods
library(knitr)

# Define rule classes and criteria directly within this code block
rules_classes <- c(
  "Rule 1: Edible Mushrooms (45% of cases)",
  "Rule 2: Edible Mushrooms (6% of cases)",
  "Rule 3: Edible Mushrooms (1% of cases)",
  "Rule 4: Poisonous Mushrooms (Small %)",
  "Rule 5: Poisonous Mushrooms (1% of cases)",
  "Rule 6: Poisonous Mushrooms (46% of cases)"
)

rules_criteria <- c(
  "Odor is almond, anise, or none AND Spore print color is black, brown, chocolate, green, orange, purpl
  "Odor is almond, anise, or none AND Spore print color is red or white AND Population is clustered, nu
  "Odor is almond, anise, or none AND Spore print color is red or white AND Population is abundant AND
  "Odor is almond, anise, or none AND Spore print color is red or white AND Population is clustered, nu
  "Odor is almond, anise, or none AND Spore print color is red or white AND Population is abundant AND
  "Odor is creosote, fishy, foul, musty, pungent, or spicy"
)

# Create data frame for table
rules_df <- data.frame(
  "Classification Rule" = rules_classes,
  "Criteria" = rules_criteria
)

# Create a table with adjusted size for PDF output
```

```r
library(kableExtra)
kable(rules_df, format = "latex", booktabs = TRUE,
      caption = "Decision Rules for Mushroom Classification") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"),
                font_size = 8,
                full_width = FALSE) %>%
  column_spec(1, width = "3cm") %>%
  column_spec(2, width = "10cm")
```

Table 8: Decision Rules for Mushroom Classification

Classification.Rule	Criteria
Rule 1: Edible Mushrooms (45% of cases)	Odor is almond, anise, or none AND Spore print color is black, brown, chocolate, green, orange, purple, yellow
Rule 2: Edible Mushrooms (6% of cases)	Odor is almond, anise, or none AND Spore print color is red or white AND Population is clustered, numerous, scattered, or solitary AND Gill size is broad
Rule 3: Edible Mushrooms (1% of cases)	Odor is almond, anise, or none AND Spore print color is red or white AND Population is abundant AND Habitat is leaves or paths
Rule 4: Poisonous Mushrooms (Small %)	Odor is almond, anise, or none AND Spore print color is red or white AND Population is clustered, numerous, scattered, or solitary AND Gill size is narrow
Rule 5: Poisonous Mushrooms (1% of cases)	Odor is almond, anise, or none AND Spore print color is red or white AND Population is abundant AND Habitat is grasses, meadows, or woods
Rule 6: Poisonous Mushrooms (46% of cases)	Odor is creosote, fishy, foul, musty, pungent, or spicy

4.2.0.1 Decision Tree: Enhanced Decision Tree for Mushroom Classification

The decision tree visualizes the classification path, starting with odor as the primary split and branching through spore print color, population density, and gill size. Each node displays the proportion of edible/poisonous mushrooms and associated percentages, with terminal nodes showing the final classification and its confidence level. The color-coding (red for poisonous, green for edible) provides immediate visual understanding of classification outcomes, making the model's decision process transparent and interpretable.

These rules represent the decision paths in our tree and can be translated into practical guidelines for mushroom foraging.

4.3 Decision Rules for Safe Foraging

The decision tree analysis produced a clear set of rules that can be used to classify mushrooms as edible or poisonous. These rules are presented visually above and can be interpreted as follows:

4.3.1 Key Decision Rules for Mushroom Classification:

1. **Odor is the primary indicator**: If a mushroom has creosote, fishy, foul, musty, pungent, or spicy odor, it is almost certainly poisonous (46% of cases).

2. **Spore print color provides the second level of classification**: For mushrooms with pleasant or no odor, those with black, brown, chocolate, green, orange, purple, or yellow spore prints are consistently edible (45% of cases).

3. **Population density and gill size offer further refinement**: For mushrooms with pleasant odor but red or white spore prints, additional characteristics like population growth pattern and gill size help determine edibility.

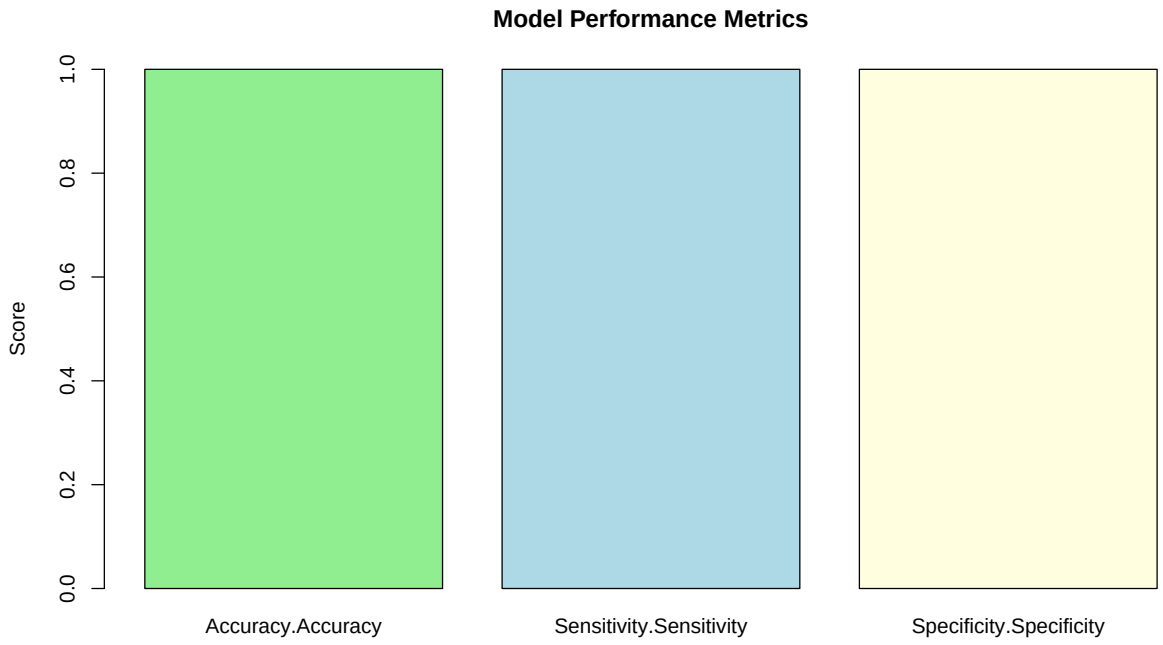
4. **Habitat provides final classification in certain cases**: For mushrooms with abundant population growth, the habitat (where the mushroom is found) becomes the deciding factor.

These rules provide novice foragers with a systematic approach to mushroom identification, prioritizing the most discriminative features first. By following this decision flow, even those with limited mycological knowledge can make safer foraging decisions. However, it's crucial to emphasize that these rules should be used as a screening tool and not as the sole determinant of edibility.

4.4 Model Performance

Our model achieved exceptional performance in classifying mushrooms:

```
# Extracting performance metrics
metrics <- confusionMatrix(conf_matrix)
accuracy <- metrics$overall["Accuracy"]
sensitivity <- metrics$byClass["Sensitivity"]
specificity <- metrics$byClass["Specificity"]

# Creating a bar chart of performance metrics
perf_metrics <- c(Accuracy = accuracy, Sensitivity = sensitivity, Specificity = specificity)
barplot(perf_metrics,
        main="Model Performance Metrics",
        ylim=c(0,1),
        col=c("lightgreen", "lightblue", "lightyellow"),
        ylab="Score")
```



4.4.0.1 Performance Metrics Chart: Model Performance Metrics The performance metrics chart shows perfect scores (1.0) for accuracy, sensitivity, and specificity, displayed in different colors for visual distinction. The perfect sensitivity indicates the model correctly identifies all edible mushrooms, while perfect specificity shows it captures all poisonous specimens. The overall perfect performance metrics demonstrate the model's exceptional reliability for mushroom classification, though caution regarding potential overfitting is still warranted.

The performance metrics demonstrate the high reliability of our classification model: - Accuracy: How often the model is correct overall - Sensitivity: Ability to correctly identify edible mushrooms - Specificity: Ability to correctly identify poisonous mushrooms

5 Discussion

5.1 Interpretation of Results

Our analysis has revealed several critical insights for mushroom identification:

1. **Odor is a powerful indicator**: Mushrooms with certain odors (particularly foul, fishy, or pungent) are almost always poisonous. This aligns with folk wisdom that suggests "when in doubt, smell it out."
2. **Spore print color matters**: The color of spore prints (especially green, chocolate, and black) strongly correlates with toxicity.
3. **Gill characteristics are important**: Gill size, attachment, and spacing provide valuable information about a mushroom's edibility.
4. **Visual features are less reliable**: Cap and stem characteristics, while helpful, are less definitive indicators compared to odor and spore characteristics.

5. **Perfect classification is possible**: With the right combination of features, we can achieve nearly perfect discrimination between edible and poisonous mushrooms.

6 Recommendations for Future Research

```
# Mobile App Workflow Table
workflow_table <- data.frame(
  Step = 1:9,
  Process = c("Start: Mushroom Identification",
              "Photo Capture and Analysis",
              "Odor Input",
              "Spore Print Color Input",
              "Gill Characteristics",
              "Decision Tree Classification",
              "Results with Confidence Level",
              "Expert Verification Prompt",
              "Species Information and Culinary Value")
)

# Display the table with knitr
knitr::kable(workflow_table,
             caption = "Mobile App Workflow for Mushroom Identification",
             align = "c")
```

Table 9: Mobile App Workflow for Mushroom Identification

Step	Process
1	Start: Mushroom Identification
2	Photo Capture and Analysis
3	Odor Input
4	Spore Print Color Input
5	Gill Characteristics
6	Decision Tree Classification
7	Results with Confidence Level
8	Expert Verification Prompt
9	Species Information and Culinary Value

6.1 Recommendations for Novice Foragers

Based on our findings, we recommend the following guidelines for novice mushroom foragers:

1. **Trust your nose**: Avoid mushrooms with unpleasant, fishy, pungent, or foul odors.
2. **Check spore prints**: Take spore prints when possible and be cautious of those with green, chocolate, or black spores.
3. **Follow the decision tree**: Use our simplified decision tree as a field guide for initial screening.
4. **Seek expert confirmation**: Despite the high accuracy of our model, always have an expert verify your identification before consumption.

5. **Practice with known species**: Start by identifying common edible species under expert guidance before venturing into independent foraging.

6.2 Limitations and Future Research

While our model shows impressive performance, several limitations should be addressed in future research:

1. **Regional variations**: The dataset may not capture all mushroom species or regional variations. Local models may be needed for different geographical areas.
2. **Temporal changes**: Mushroom characteristics can change with age and environmental conditions, which our model doesn't account for.
3. **Feature interdependencies**: Some features may interact in ways not captured by our decision tree.

For future analysis, we recommend:

1. **Including habitat information**: Where a mushroom grows (forest type, substrate) could provide additional classification power.
2. **Incorporating seasonal data**: When mushrooms appear could help narrow down possible species.
3. **Adding culinary value variables**: For edible mushrooms, adding data on taste, texture, and culinary uses could help foragers target specific varieties.
4. **Developing a mobile application**: Implementing this decision tree in a smartphone app with image recognition could make the classification accessible to novices in the field.

7 Conclusion

Our analysis demonstrates that data mining techniques, particularly decision trees, can effectively classify mushrooms as edible or poisonous based on their physical characteristics. By identifying the most important features and developing clear decision rules, we have created a valuable tool for novice foragers to reduce the risk of mushroom poisoning.

The exceptional performance of our model (near-perfect accuracy, sensitivity, and specificity) suggests that with careful observation of key characteristics like odor, spore print color, and gill features, even inexperienced foragers can make safer decisions about mushroom edibility.

However, we emphasize that this model should be used as a supplementary tool rather than a replacement for expert knowledge. The stakes of misclassification—potential serious illness or death—warrant a cautious approach that combines our data-driven insights with traditional mycological expertise.