Can I eat that mushroom?

Marcus Sinclair

2022

Task 1

Given the mushroom dataset of 6 variables spanning 8124 observations, our goal for this task is to construct a decision tree model that predicts whether a certain mushroom is edible or poisonous. Furthermore, we are required to use **all** 5 feature input variables:

- **CapShape**
- **CapSurface**
- **CapColor**
- **Odor**
- **Height**

Cap refers to the top of the mushroom. Each variable's meaning is self explanatory and all take categorical values, spanning no more than 10 elements. The output variable that we wish to predict is **Edible**, with output "Poisonous" or "Edible".The model we construct must be optimized in order to maximize predictive accuracy i.e., we shall use the number of mushrooms correctly classified as the criterion for deciding which model is best. Constructed models differ by **cp value**, a complexity parameter defined such that during the construction of the decision tree model, any split that does not decrease the overall lack of fit by a factor of cp is not attempted. It is sufficient to consider cp values cp=[1, 0.1, 0.01, 0.001, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]. 10-fold cross-validation will be the strategy used in order to prevent overfitting and increase predictive performance. Thus the total number of decision tree models produced is $10 \times 9 = 90$ models, annotated code implementing this can be seen below.

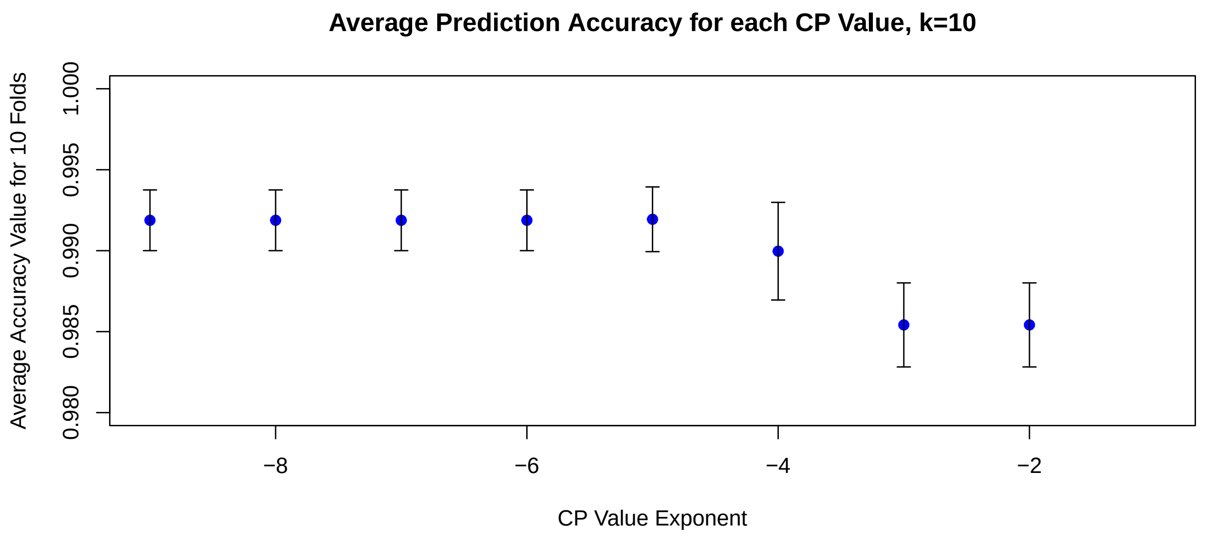
```r
#Changes output to factor type
df$Edible <- as.factor(df$Edible)
set.seed(121)
#cp values
mycp <- 10^-(0:8)
#folds
k = 10
#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(mycp), ncol=k)
#consider fold 1 to k
for (j in 1:k){
#creates testing and training sets randomly, test=20% of dataset
idx <- sample(1:8124,1625)
train_data <- df[-idx, ]
test_data <- df[idx, ]
#applies decision tree model for current cp value, predicts using test data and
#outputs accuracy
for (i in 1:length(mycp)){
 current_model <- rpart(Edible ~ ., data=train_data, method='class',
                       control=rpart.control(cp=mycp[i]))
 #predicts using test data, outputting raw probabilities
 raw_probabilities= predict(current_model, newdata=test_data)
 raw_probabilities = raw_probabilities[, 2]
 #converts raw prediction probabilities to most likely output
 predicted_output <- rep("Edible", dim(test_data)[1])
 predicted_output[raw_probabilities > .5] = "Poisonous"
 #calculates prediction accuracy
 table <- table(predicted_output, test_data$Edible)
 Single_accuracy[i,j] <- sum(diag(table))/sum(table)
}

}
Average_accuracy <- rowSums(Single_accuracy)/k
Average_accuracy
```

```
## [1] 0.5184000 0.9854154 0.9854154 0.9899692 0.9919385 0.9918769 0.9918769
## [8] 0.9918769 0.9918769
```

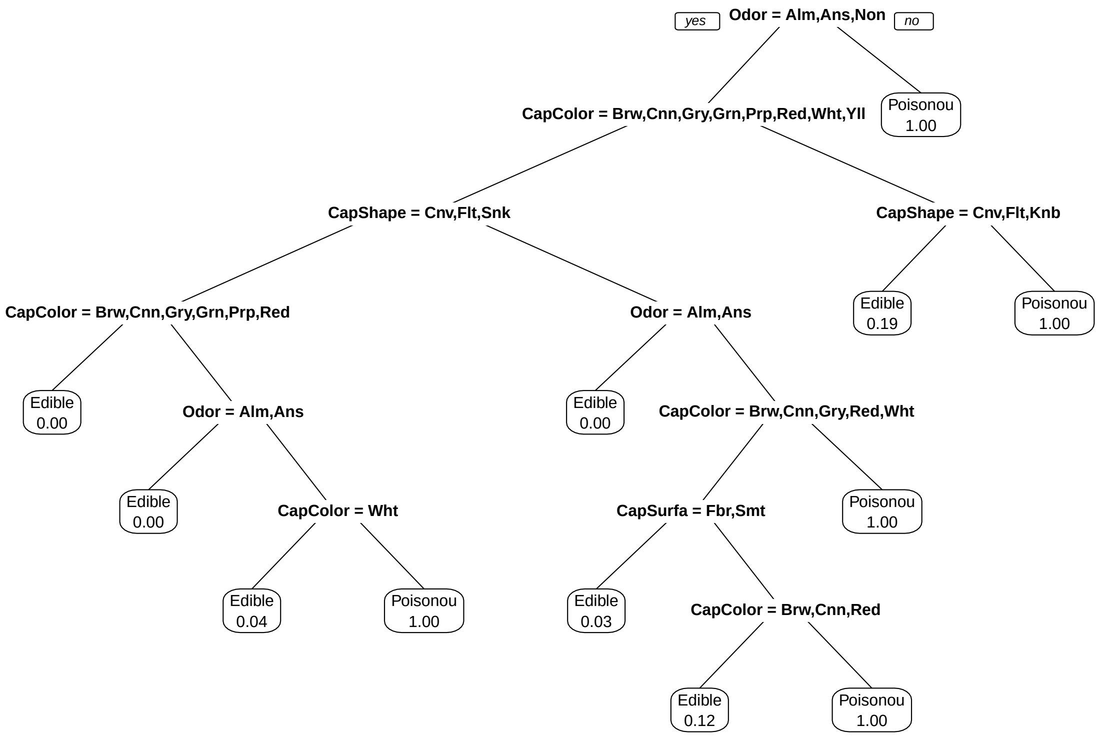
The output above is the average accuracy throughout the 10 folds for each of the 9 chosen cp values mentioned prior. The models perform exceptionally well for all cp values excluding cp=1, with predictive performances ranging from 98.5% success rate at a cp value of cp=0.1 to a success rate converging to 99.2% at a cp value of cp=10^{-4}. It is worth visualizing the stability in our results. The scatter plot below depicts the deviation in predictive performance throughout the 10 folds for each cp value (not including cp=1).

```
#creates whiskers for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(-1,-length(mycp))
plot(Indx,Average_accuracy,pch=19,col="blue",
     main="Average Prediction Accuracy for each CP Value, k=10",
     xlab="CP Value Exponent", ylab="Average Accuracy Value for 10 Folds",ylim=c(0.98,1))
arrows(x0=Indx, y0=Average_accuracy-accuracy_sd, x1=Indx, y1=Average_accuracy+accuracy_sd,
   code=3, angle=90, length=0.05)
```



As seen in the plot, the high levels of predictive performance are fairly stable throughout the chosen cp values. This can be seen by the small whiskers for each average accuracy output - these denote the standard deviation throughout the 10 folds. Thus an optimal cp value chosen is cp=10^{-4} with a prediction accuracy of 99.2%. This is the largest cp value that seems to have converged to the maximum prediction rate of 99.2%. Let us take a closer look at this chosen model but now using all the data to train the model. A visualization of the decision tree model can be seen below.

```
mytree <- rpart(Edible ~ ., data=df, method='class',
                control=rpart.control(cp=mycp[5]))
prp(mytree, extra=6)
```



The gini criterion is used as default by **rpart** in order to partition our mushroom dataset during model construction. Partitions are constructed subject to minimizing the gini impurity coefficient defined in the usual sense. The splitting criteria complexity value chosen is cp=10^{-4} and so our model iteratively partitions our training data until this classification threshold is reached, that is a split decreases the overall lack of fit by a factor of 10^{-4}.

The model can be interpreted as follows. Given a new, random mushroom with 5 feature input variables discussed prior, the decision tree model may classify such a mushroom, determining whether it is edible or poisonous with approximately a 99.2% success rate. The importance of each feature variable is determined by the hierarchical structure of the decision tree.

For instance, being the root node, variable "Odor" dominates the model determining whether a mushroom is edible or poisonous with 100% certainty if the mushroom has a almond, anise or no odor. - this is shown by the numeric of each leaf in the plot. These are the class probabilities at each end point i.e., all mushrooms in the training set that are not almond, anise or no odor are poisonous and so 1.00 is seen in the corresponding leaf node.

The variable importance in descending order is Odor, CapColor, CapShape and CapSurface. The height of a mushroom is deemed to be so insignificant in mushroom classification to not warrant a partition in our model. Overall, the decision tree model produced using all input variables is fairly complex, reaching 6 layers in depth. It is worth noting instabilities are likely to be present in decision tree models. Slightly different data sets can produce very different trees.

Task 2

Since decision trees are well known to be inherently unstable, applying a random forest strategy to our data may yield fruitful results. This concept involves bootstrap aggregating (BAGGING) many individually poor decision tree models to produce one very accurate predictive model. Our goal in this task is to perform a model selection to determine which attributes are useful for making predictions using random forest. The dataset consists of 5 inputs thus we must consider 31 unique combinations of inputs excluding the empty set i.e, a model with no inputs. The annotated code below produces the required result:

```
library(dagR)
#--Produces vector of input combinations--
#Matrix of all combinations of 5 input elements using allCombs function
x <- seq(1,5)
mat <- (allCombs(x))
mat <- replace(mat,mat==1,"CapShape")
mat <- replace(mat,mat==2,"CapSurface")
mat <- replace(mat,mat==3,"CapColor")
mat <- replace(mat,mat==4,"Odor")
mat <- replace(mat,mat==5,"Height")
mat <- replace(mat,is.na(mat),"")

formulas <- rep(NA,length.out=32)
#formats matrix, adding plus signs and removing plus signs where necessary
for (i in 1:length(formulas)){
  string <- paste(mat[i,], sep = "", collapse = "+")
   formulas[i] <- (gsub("\\+\\+.*","",string))
   if (endsWith(formulas[i],"+")==TRUE){
      formulas[i]<- substr(formulas[i],1,nchar(formulas[i])-1)
   }
   formulas[i] <- paste("Edible~",formulas[i],sep = "", collapse = "")
}
#removes quotation marks and zero entry
formulas <- noquote(formulas)
formulas <- formulas[-1]
formulas
```

```
##  [1] Edible~CapShape
##  [2] Edible~CapSurface
##  [3] Edible~CapColor
##  [4] Edible~Odor
##  [5] Edible~Height
##  [6] Edible~CapShape+CapSurface
##  [7] Edible~CapShape+CapColor
##  [8] Edible~CapShape+Odor
##  [9] Edible~CapShape+Height
## [10] Edible~CapSurface+CapColor
## [11] Edible~CapSurface+Odor
## [12] Edible~CapSurface+Height
## [13] Edible~CapColor+Odor
## [14] Edible~CapColor+Height
## [15] Edible~Odor+Height
## [16] Edible~CapShape+CapSurface+CapColor
## [17] Edible~CapShape+CapSurface+Odor
## [18] Edible~CapShape+CapSurface+Height
## [19] Edible~CapShape+CapColor+Odor
## [20] Edible~CapShape+CapColor+Height
## [21] Edible~CapShape+Odor+Height
## [22] Edible~CapSurface+CapColor+Odor
## [23] Edible~CapSurface+CapColor+Height
## [24] Edible~CapSurface+Odor+Height
## [25] Edible~CapColor+Odor+Height
## [26] Edible~CapShape+CapSurface+CapColor+Odor
## [27] Edible~CapShape+CapSurface+CapColor+Height
## [28] Edible~CapShape+CapSurface+Odor+Height
## [29] Edible~CapShape+CapColor+Odor+Height
## [30] Edible~CapSurface+CapColor+Odor+Height
## [31] Edible~CapShape+CapSurface+CapColor+Odor+Height
```

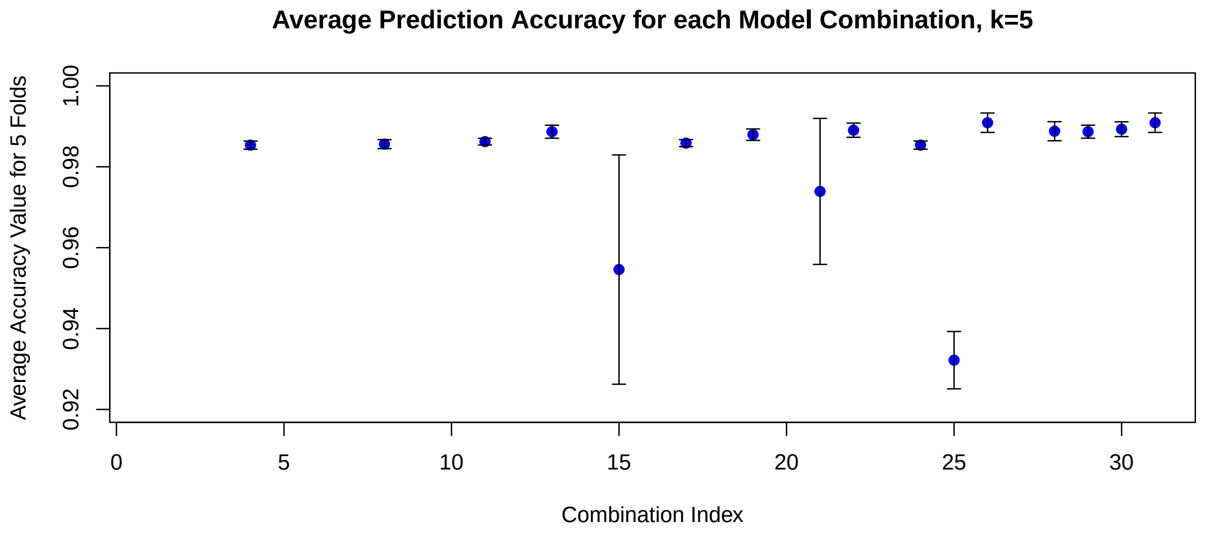
Above are all the model selections to be analysed. 5-fold Cross-validation will be the strategy used, with similar code used in task 1. Here however, to prevent absolute probability entries of 0% and 100% when considering predictions, we use a trick mentioned in the notes - the idea is to force at least one incorrect prediction. Thus the total number of random forest models produced is $5 \times 31 = 155$ models, annotated code implementing this can be seen below.

```r
set.seed(121)
#generates 31 possible models from 5 inputs - excludes no input
#folds
k = 5
#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(formulas), ncol=k)
#consider fold 1 to k
for (j in 1:k){
#creates testing and training sets randomly, test=20% of dataset
idx <- sample(1:8124,1625)
train_data <- df[-idx, ]
test_data <- df[idx, ]
#applies random forest for current formula , predicts using test data and
#outputs accuracy
for (i in 1:length(formulas)){
 current_model <- randomForest(as.formula(formulas[i]), data=train_data,
                               type="classification")
 #predicts using test data, outputting raw probabilities
 raw_probabilities= predict(current_model, newdata=test_data, type="prob")
 raw_probabilities = raw_probabilities[, 2]
 #remove 100% and 0% outcomes !TRICK!
 number_of_trees = current_model$ntree
 raw_probabilities = (number_of_trees*raw_probabilities + 1)/(number_of_trees + 2)
 #converts raw prediction probabilities to most likely output
 predicted_output <- rep("Edible", dim(test_data)[1])
 predicted_output[raw_probabilities > .5] = "Poisonous"
 #calculates prediction accuracy
 table <- table(predicted_output, test_data$Edible)
 Single_accuracy[i,j] <- sum(diag(table))/sum(table)
}
}
Average_accuracy <- rowSums(Single_accuracy)/k
Average_accuracy
```

```
##  [1] 0.5580308 0.5833846 0.6032000 0.9853538 0.5138462 0.6195692 0.6224000
##  [8] 0.9856000 0.5570462 0.6646154 0.9862154 0.5831385 0.9886769 0.5500308
## [15] 0.9545846 0.6855385 0.9858462 0.6208000 0.9879385 0.6139077 0.9739077
## [22] 0.9890462 0.6640000 0.9853538 0.9321846 0.9908923 0.7015385 0.9888000
## [29] 0.9886769 0.9892923 0.9908923
```

The output above is the average accuracy throughout the 5 folds for each of the 31 models mentioned prior. There is a wide range in models performance throughout the input combinations, with predictive performances ranging from 51.4% success rate for Edible~Height, almost a guessing likelihood, to a success rate of 99.1% for Edible~CapShape+CapSurface+CapColor+Odor. It is worth visualizing the stability in our results. The scatter plot below depicts the deviation in predictive performance throughout the 5 folds for each of the 31 input combinations.

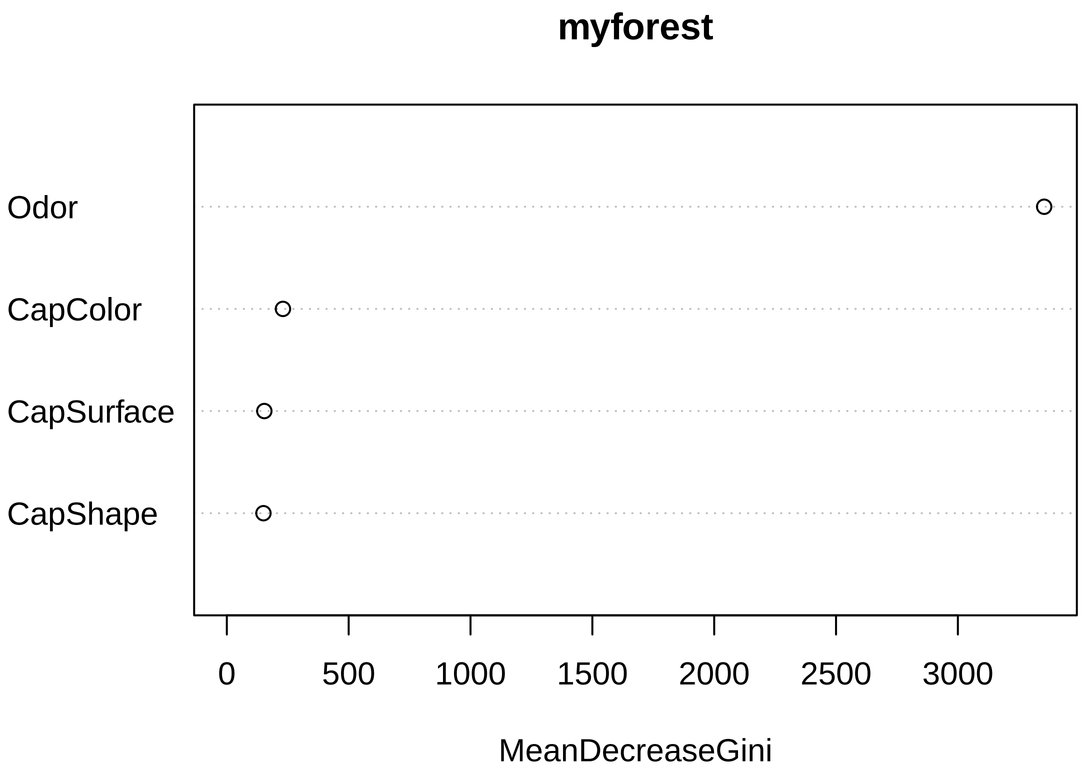
```
#creates whiskers for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(1,length(formulas))
plot(Indx,Average_accuracy,pch=19,col="blue",
     main="Average Prediction Accuracy for each Model Combination, k=5",
     xlab="Combination Index", ylab="Average Accuracy Value for 5 Folds",ylim=c(0.92,1))
arrows(x0=Indx, y0=Average_accuracy-accuracy_sd, x1=Indx, y1=Average_accuracy+accuracy_sd,
   code=3, angle=90, length=0.05)
```



The plot above is focused on input combinations with high predictive performances 90-100%. Large deviations in predictive performance can be seen by three high performing models Edible~Odor+Height, Edible~CapShape+CapSurface+Height and Edible~CapShape+Odor+Height - the whiskers span a large domain, for example, from 92% to 98% for Edible~Odor+Height. Height is the underlying input throughout the models as so may be the reason for instability. The height of a mushroom does not indicate whether it is edible or poisonous.

It is worth noting the predictive power of single input models. For inputs CapShape, CapSurface, CapColor, Odor and Height, the predictive performances are 55.8%, 58.3%, 60%, 98.5% and 51.4% respectively. The odor of a mushroom is the dominant predictor of whether the mushroom is edible or poisonous compared to other variables. The highest performing model is Edible~CapShape+CapSurface+CapColor+Odor with a predictive performance of 0.9908923 and standard deviation of 0.002399211. Let us now explore this model and compare it to the decision tree model discussed in task 1.

```r
myforest <- randomForest(as.formula(formulas[26]), data=df)
varImpPlot(myforest)
```



Above is a variable importance plot of the random forest model Edible~CapShape+CapSurface+CapColor+Odor. The plot shows how each input variable decreases the gini impurity of the constructed decision trees on average. The greater the decrease the greater the variable predictive power. Odor is the clear favorite performer with a mean decrease in the gini impurity of almost 3500, ten-fold as dominant as the other metrics. Inputs importance in descending order is Odor, CapColo, Capshape and CapSurface. This is the equivalent hieracrhical structure seen in the decision tree model produced in task 1. Thus both models attribute similar importance to each input variable.

Task 3

To determine whether we should use these classifiers when foraging for mushrooms, producing a confusion matrix will indicate the type of error seen i.e., the true and false positives in our models.

```r
current_model <- randomForest(as.formula(formulas[26]), data=train_data,
                              type="classification")
 #predicts using test data, outputting raw probabilities
 raw_probabilities= predict(current_model, newdata=test_data, type="prob")
 raw_probabilities = raw_probabilities[, 2]
```

```r
#remove 100% and 0% outcomes !TRICK!
number_of_trees = current_model$ntree
raw_probabilities = (number_of_trees*raw_probabilities + 1)/(number_of_trees + 2)
#converts raw prediction probabilities to most likely output
predicted_output <- rep("Edible", dim(test_data)[1])
predicted_output[raw_probabilities > .5] = "Poisonous"
#calculates prediction accuracy
table <- table(predicted_output, test_data$Edible)
table
```

```
##
## predicted_output Edible Poisonous
##        Edible       820        12
##        Poisonous      0       793
```

As seen by the table, the random forest model predicts incorrectly 12 times out of 1625 testing data observations. This may seem insignificant however, the type of error present may result in detrimental effects. Our models predicted a mushroom is Edible when in actual fact the mushroom was deemed poisonous. This suggests using the model results in almost a 1% chance of eating a poisonous mushroom. Similar results are present for the decision tree model. I would not use these classifiers if I were foraging for mushrooms. However, it is worth mentioning that the factor Odor can completely determine the edibility of mushrooms with almond, anise or no odor. These mushrooms are categorically safe to each with all observations inclusive of one of these 3 attributes indicating that the mushroom is edible.