Mushroom Classification

Jyothi, Murali, Vivek, Rajesh

# 1. Introduction

Mushroom classification is a famous Machine learning problem. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom. Each species is labeled with either edible or poisonous.

The mushroom dataset has around 8100 rows with 22 different descriptions of mushrooms and labeled column indicating the class of mushrooms as edible or poisonous. The dataset is simple and pre-processed. We can use the labeled data to build a model with supervised algorithm.

For this problem, we are going build different models with various classification algorithms and validate which models perform better on this dataset in R. We will divide the entire dataset into train data (consisting of 80% random sampled data from original) and test data (remaining 20%). We shall build models training with train data and then check performance on test data. We are also going to build a model with Deep learning.

More details about this dataset can be found at <https://www.kaggle.com/uciml/mushroom-classification>

# 2.Dataset

We are downloading the data from url <https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data> . The dataset is available in CSV format. We can load the and read with *read.csv*

##Importing the data from URL

*url=*[*https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data*](https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data)

*mushrooms\_data <- read.csv(file = url, header = FALSE, sep = ",",strip.white = TRUE)*

The file is sized around 300 kb and contains 8124 rows and 23 columns. All the data in this file is categorical. The first column represents class of the mushroom. ‘e’ as edible and ‘p’ as poisonous and we use this as labeled outcome to train our models.

The dataset we have downloaded from the url mentioned do not have any associated header representing the column names. So we have to add the column names with R explicitly for better knowing about the data. We can achieve this below.

*colnames(mushrooms\_data) <- c("class","cap-shape","cap-surface","cap-color","bruises","odor","gill-attachment","gill-spacing","gill-size","gill-color","stalk-shape","stalk-root","stalk-surface-above-ring","stalk-surface-below-ring","stalk-color-above-ring","stalk-color-below-ring","veil-type","veil-color","ring-number","ring-type","spore-print-color","population","habitat")*

All columns have single character data representing different categories for that factor. For example, the column *cap-shape* has values representing like ‘b’ = bell, ‘c’ = conical, ‘x’ = convex, ‘f’ = flat, ‘k’ = knobbed and ‘s’ = sunken. We can find the description and different categories for all columns at below link.

<https://www.kaggle.com/uciml/mushroom-classification>

**Packages required.**

We shall require below packages to run this problem and build the modles.

*CARET Package*

The [caret](http://cran.r-project.org/web/packages/caret/index.html) package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for data splitting, pre-processing, feature selection, model tuning using resampling, variable importance estimation. There are many different modeling functions in R. Some have different syntax for model training and/or prediction. The package started off as a way to provide a uniform interface the functions themselves, as well as a way to standardize common task

*GGPLOT2 Package*

## The ggplot2 package is used to Create Elegant Data Visualizations using the Grammar of Graphics. ggplot2 take the good parts of base and lattice graphics. with ggplot2 we can create complex multi-layered graphics easily.

*GGTHEMES Package*

ggthemes package can be used to create extra geoms, scales, and themes for [ggplot](http://ggplot2.org/)2

we can install and load all libraries with below code.

*Install.packages(“ggplot2”)*

*Install.packages(“caret”)*

*Install.packages(“ggthemes”)*

*library(ggplot2)*

*library(caret)*

*library(ggthemes)*

# 3. Data loading and segmentation

*Load Data*

We can load data into memory for processing with below command.

*mushrooms\_data <- read.csv(file = url, header = FALSE, sep = ",")*

*Data Segmentation*

After we have loaded entire data set, we have to divide the whole into train set and test set. Train set is used to build the model and test set is used to validate the model accuracy on unseen data.

*set.seed(1029) #for Repoducebality*

*train\_index = sample(nrow(mushrooms\_data), ceiling (0.8\* nrow(mushrooms\_data) ))*

*mushrooms\_train\_dataset <- mushrooms\_data[train\_index, ]*

*mushrooms\_test\_dataset <- mushrooms\_data[-train\_index,]*

#### we have used set.seed with arbitrary integer to produce same random numbers when we rerun this project at later time. Without set.seed the sampling will provide different random number which makes different train set and the final model will be different for each run the project

#### here we are dividing 80% data set into train set and remaining 20% into test data set. Sample() function will pick specified random numbers from given range. sample(300,100) gives 100 random numbers within range 1 to 300.

#### We are generating random index of 6500 from the range of 1 to 8124 . nrow() will return no.of rows in the data frame.

# 4. Exploring the data

We can explore the data by first looking at the dimensions and structure of the dataset and viewing the first few rows of the data set got get basic details

Dimensions

> dim(mushrooms\_data)

[1] 8124 23

#### Structure

> str(mushrooms\_data)

'data.frame': 8124 obs. of 23 variables:

$ class : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1

$ cap-shape : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6

$ cap-surface : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4

$ cap-color : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9

$ bruises : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2

$ odor : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6

$ gill-attachment : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2

$ gill-spacing : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1

$ gill-size : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1

$ gill-color : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6

$ stalk-shape : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1

$ stalk-root : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4

$ stalk-surface-above-ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3

$ stalk-surface-below-ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3

$ stalk-color-above-ring : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8

$ stalk-color-below-ring : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8

$ veil-type : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...

$ veil-color : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3

$ ring-number : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2

$ ring-type : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1

$ spore-print-color : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4

$ population : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1

$ habitat : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2

#### with str(), we can observe that veil-type has only one category and it’s not needed to distinguish the class. So we can remove this with below. This makes our total no.of columns 22.

*mushrooms\_data$`veil-type` <- NULL*

#### Explore all remaining factor with graphs

#### 

#### 5.Modeling the data

## Model I : rpart:

We have trained the model on the trainset , we will also do the 10 fold cross validation and repeat 4 times. After the model is defined we find the predictions using the model for the test set and find out how well the model did. We follow the same process for all the models and thanks to caret package, training the models are more convenient than ever. Trees (also called decision trees, recursive partitioning) are a simple yet powerful tool in predictive statistics. The idea is to split the covariable space into many partitions and to fit a constant model of the response variable in each partition. In case of regression, the mean of the response variable in one node would be assigned to this node. The structure is similar to a real tree (from the bottom up): there is a root, where the first split happens. After each split, two new nodes are created (assuming we only make binary splits). Each node only contains a subset of the observations. The partitions of the data, which are not split any more, are called terminal nodes or leafs. This simple mechanism makes the interpretation of the model pretty easy.

Following is the performance statistics of this model :

Confusion Matrix and Statistics – train set

Reference

Prediction e p

e 3348 37

p 0 3115

As the train set confusion matrix doesn’t look good, which is classifying 37 poisonous mushrooms as edible so we have tried with penalizing the specific case.

*penalty\_matrix = matrix(c(0,1,10,0),2,2,byrow = T)*

*mushrooms\_mdl\_rpart2<- rpart(class ~ . , data=mushrooms\_train\_dataset,*

*parms = list(loss = penalty\_matrix),*

*method = "class")*

After building the model with penalty matrix , the results surprisingly improved.

Confusion Matrix and Statistics

Reference

Prediction e p

e 860 0

p 0 764

Accuracy : 1

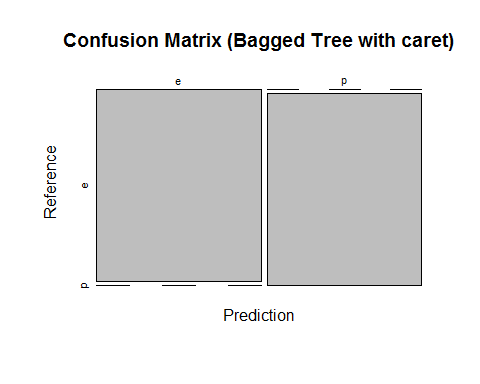
95% CI : (0.9977, 1)

**##Model II : Decision Tree and Bagging:**

Bagging is one of the methods for improving the performance of weak learners such as Trees. Classification trees are adaptive and robust. So we create multiple trees by considering all variables at split and then averaging them for results. Training the model with the same cross validation options like last model and checking the performance :

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction e p  
## e 841 0  
## p 0 783  
##   
## Accuracy : 1   
## 95% CI : (0.9977, 1)  
## No Information Rate : 0.5179   
## 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 : 0.5179   
## Detection Rate : 0.5179   
## Detection Prevalence : 0.5179   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : e   
##

plot(mushroom\_tbl\_crt\_bag\_test$table,main="Confusion Matrix (Bagged Tree with caret)")

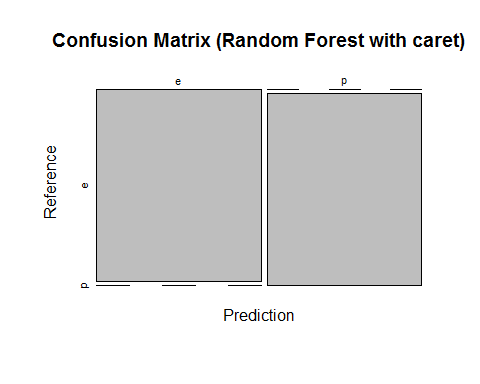


## Model III : RandomForest :

RandomForest is a specific kind of bagging trees , where trees are random and are not every split considers all the variables. Training the model with the same cross validation options as last models. randomforest algo improves the accuracy of the tree based algorithms by creating an ensemble of diverse trees.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction e p  
## e 841 0  
## p 0 783  
##   
## Accuracy : 1   
## 95% CI : (0.9977, 1)  
## No Information Rate : 0.5179   
## 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 : 0.5179   
## Detection Rate : 0.5179   
## Detection Prevalence : 0.5179   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : e   
##

plot(mushroom\_tbl\_crt\_rf\_test$table,main="Confusion Matrix (Random Forest with caret)")

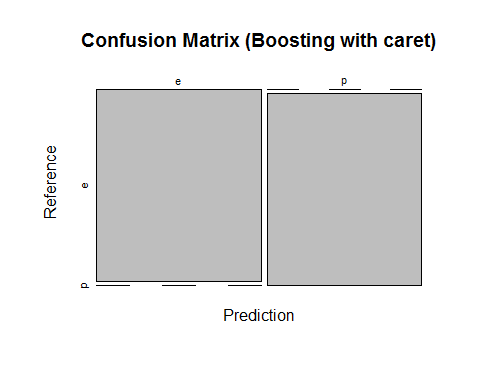


## Model iv : Boosting :

Boosting is classification , in which trees are sequentially grown and every subsequent tree learns from the results of last tree. Training the model with the same cross validation options as last models:

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction e p  
## e 841 0  
## p 0 783  
##   
## Accuracy : 1   
## 95% CI : (0.9977, 1)  
## No Information Rate : 0.5179   
## 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 : 0.5179   
## Detection Rate : 0.5179   
## Detection Prevalence : 0.5179   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : e   
##

plot(mushroom\_tbl\_crt\_boost\_test$table,main="Confusion Matrix (Boosting with caret)")

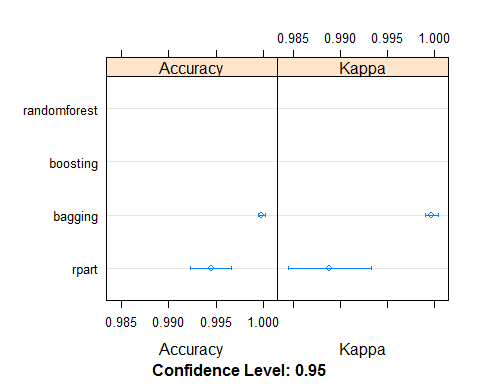


## Model Comparison:

At this stage , we compare the models for their performance to decide which works better on this dataset,

##   
## Call:  
## summary.resamples(object = comparison)  
##   
## Models: rpart, bagging, randomforest, boosting   
## Number of resamples: 10   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.9892 0.9927 0.9946 0.9945 0.9965 0.9985 0  
## bagging 0.9985 1.0000 1.0000 0.9998 1.0000 1.0000 0  
## randomforest 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0  
## boosting 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.9784 0.9853 0.9892 0.9889 0.9931 0.9969 0  
## bagging 0.9969 1.0000 1.0000 0.9997 1.0000 1.0000 0  
## randomforest 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0  
## boosting 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0

dotplot(comparison)



#### 6. Conclusion and Discussion

We have used different models for modeling and almost all models performed well on this dataset. We initially modeled using rpart and the over all accuracy is good but this was distinguishing poisonous as edible which is not useful. Later we could trained model using penalty matrix in rpart with more penalty given to case poison mushrooms classified as edible and this made the model more efficient and achieved 100% accuracy too.

We tried to prune the tree, but this also achieved same results as the original classification tree. The bagging and random forest also achieved 100% results.

We can conclude that tree models are more efficient for this mushroom classification dataset.

#### 7. Appendix

<https://github.com/vivekpruthi1/DSLA1/>

<https://www.kaggle.com/uciml/mushroom-classification>

<https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/>