Statistical Learning Accessed Practicle - 6

Venkat Gopinath Polamuri

2023-05-25

Introduction

Import nesscary libraries

1. `library(dplyr)` is used to perform basic data preprocessing.
2. `library(ggplot2)` is used for displaying graphs
3. `library(caret)` is used to create Data partitions, train and test set, confusion Matrix.
4. `library(rpart), library(rpart.plot)` is used for Decision tree model.
5. `library(randomForest)` is used for Random forest model.
6. `library(knitr)` it is used to visualize the outputs in form of Tables.

```
rm(list=ls())
library(rpart)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 4.2.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.2.2
```

Importing Data

To import we areusing the function `read.csv()` and assigning it to a variable Data

```
data = read.csv("mushrooms.csv")
```

Data characterisation

Firstly, let us find number of variables and its types in the Data

```
names(data)
```

```
## [1] "Edible"     "CapShape"   "CapSurface" "CapColor"   "Odor"
## [6] "Height"
```

```
str(data)
```

```
## 'data.frame':    8124 obs. of  6 variables:
##  $ Edible    : chr  "Poisonous" "Edible" "Edible" "Poisonous" ...
##  $ CapShape  : chr  "Convex" "Convex" "Bell" "Convex" ...
##  $ CapSurface: chr  "Smooth" "Smooth" "Smooth" "Scaly" ...
##  $ CapColor  : chr  "Brown" "Yellow" "White" "White" ...
##  $ Odor      : chr  "Pungent" "Almond" "Anise" "Pungent" ...
##  $ Height    : chr  "Tall" "Short" "Tall" "Short" ...
```

From above output we can observe that the Data consists of 6 varaiable and 8124 rows:

- **Edible:** This variable provide that if a Mushroom is edible or not. It is Nominal, categorical data, it consists of Edible or Posisonous for each row.
- **CapShape:** This variable provide the shape of cap of Mushroom. It is Nominal, categorical data,
- **CapSurface:** This variable provide the surface of cap of Mushroom. It is Nominal, categorical data,
- **CapColor:** This variable provide the color of cap of Mushroom. It is Nominal, categorical data,
- **Odor:** This variable provide the odor or smell of Mushroom. It is Nominal, categorical data,
- **Height:** This varaible specifics if the Mushroom is Tall or short. It is Nominal, categorical data,

Exploratory Data Analysis

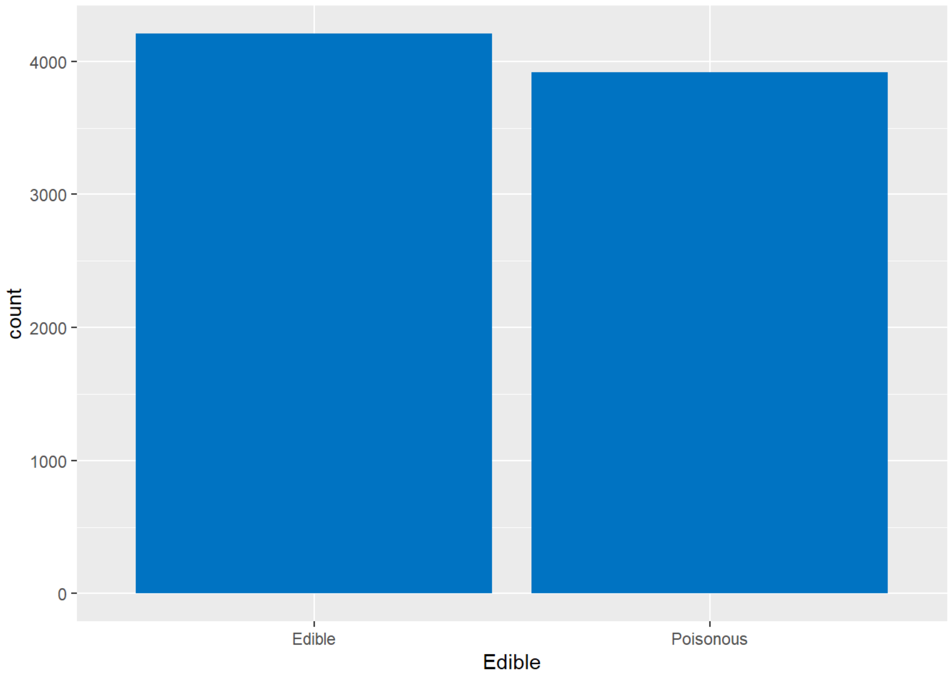
Edible

It is consists of two main categories Edible or Poisonous, with Edible rows as 4208, Poisonous rows as 3916

```
print(table(data$Edible))
```

```
##
##    Edible Poisonous
##      4208      3916
```

```
ggplot(data, aes(Edible))+geom_bar(fill = "#0073C2FF")
```



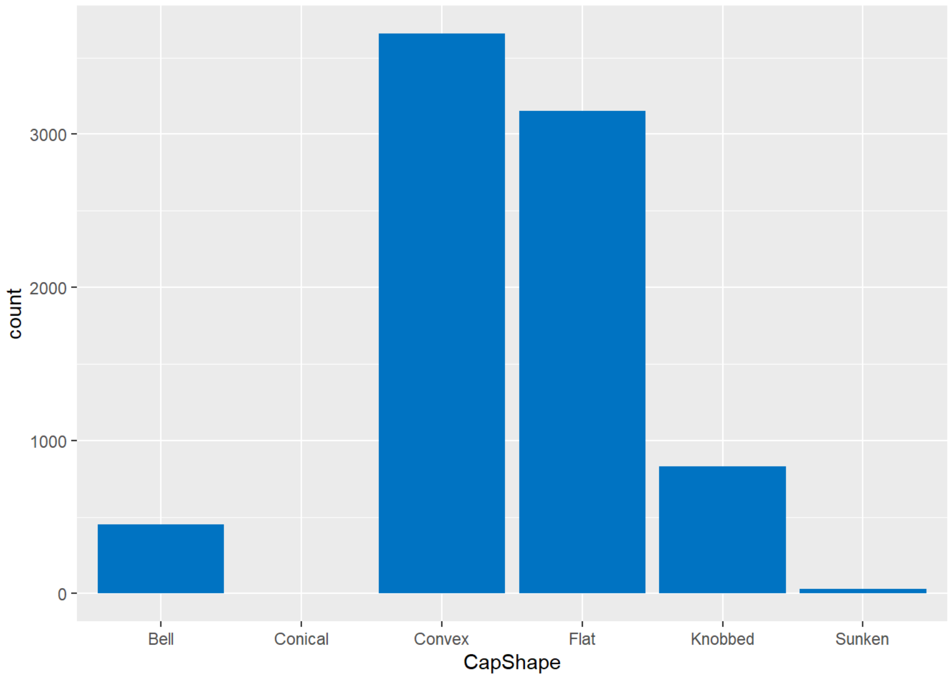
CapShape

It is consists of 6 main categories bell, conical, flat, knobbed, Sunken. out of these Convex, Flat are highest and Conical, Sunken are least occuring Cap shapes

```
table(data$CapShape)
```

```
##
##    Bell Conical  Convex    Flat Knobbed  Sunken
##     452       4    3656    3152     828      32
```

```
ggplot(data, aes(CapShape))+geom_bar(fill = "#0073C2FF")
```



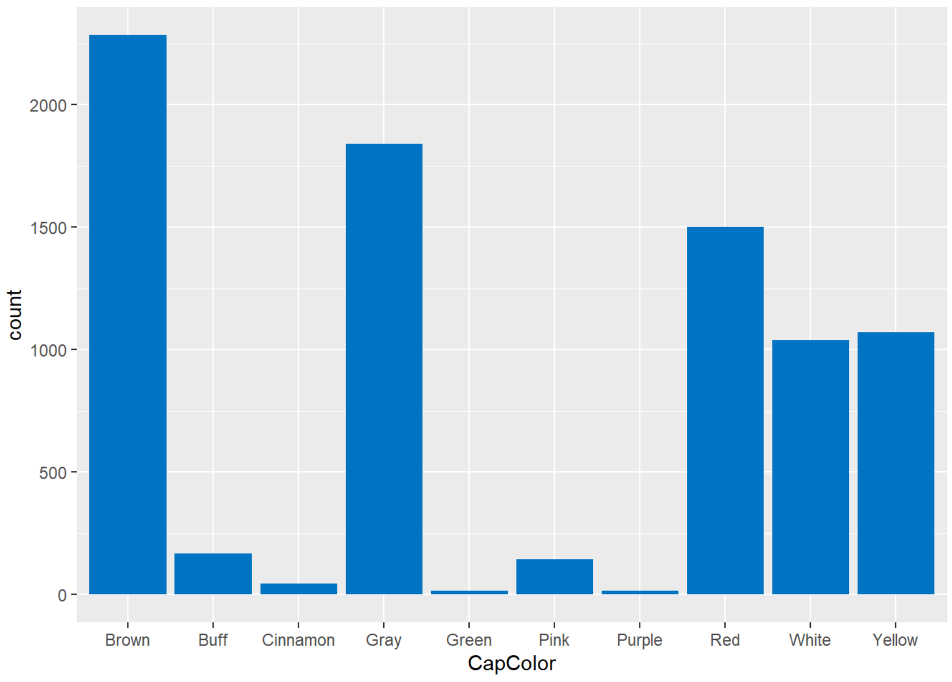
CapColor

It consists of 10 different colors like Brown, Buff, cinnamon etc, Most of the Mushrooms are visible in Brown and Grey colour, Mushrooms with Green and Purple color are very rare to see.

```
table(data$CapColor)
```

```
##
##    Brown      Buff Cinnamon      Gray     Green      Pink    Purple       Red
##     2284       168       44      1840        16       144        16      1500
##    White    Yellow
##     1040      1072
```

```
ggplot(data, aes(CapColor))+geom_bar(fill = "#0073C2FF")
```



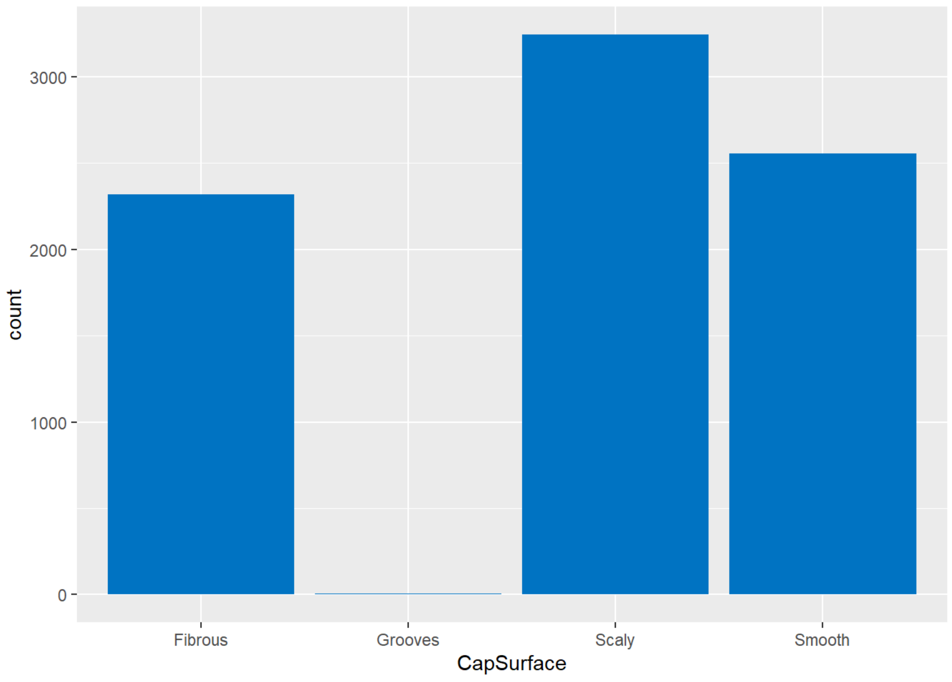
CapSurface

It consists of 4 cap surface namely fibrous, Groves, scally, Smooth. Mushrooms with Scaly surface are more with Grooves as less

```
table(data$CapSurface)
```

```
##
## Fibrous Grooves   Scaly  Smooth
##    2320       4    3244    2556
```

```
ggplot(data, aes(CapSurface))+geom_bar(fill = "#0073C2FF")
```



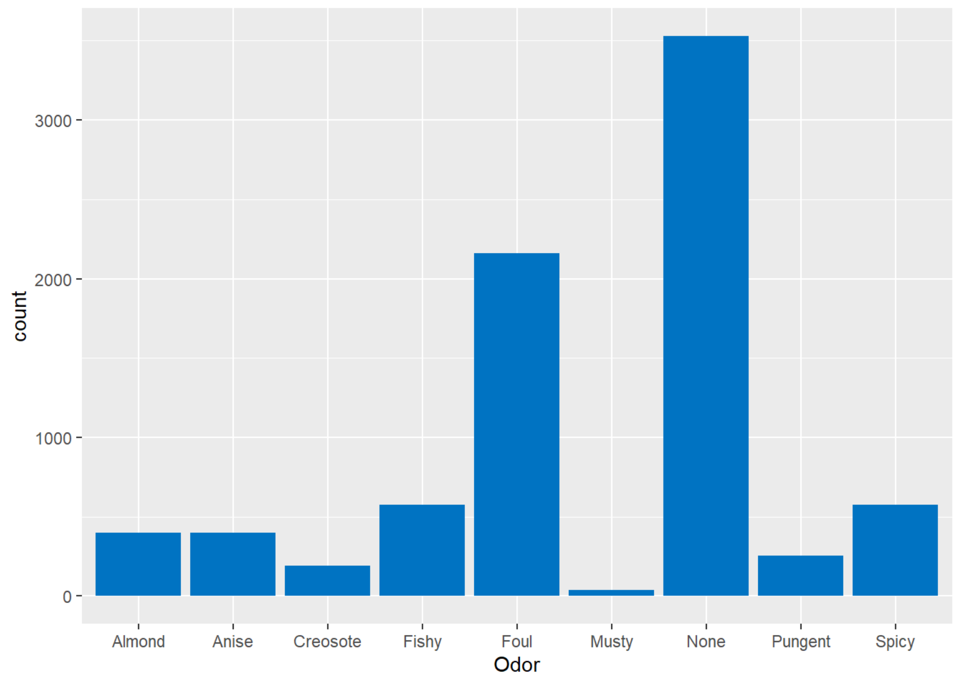
Odor

Our data consists of 9 different Odor, but most of the mushrooms are without Odor, few mushrooms are with Musty ordor.

```
table(data$Odor)
```

```
##
##    Almond    Anise Creosote    Fishy     Foul    Musty     None  Pungent
##       400      400      192      576     2160       36     3528      256
##     Spicy
##       576
```

```
ggplot(data, aes(Odor))+geom_bar(fill = "#0073C2FF")
```



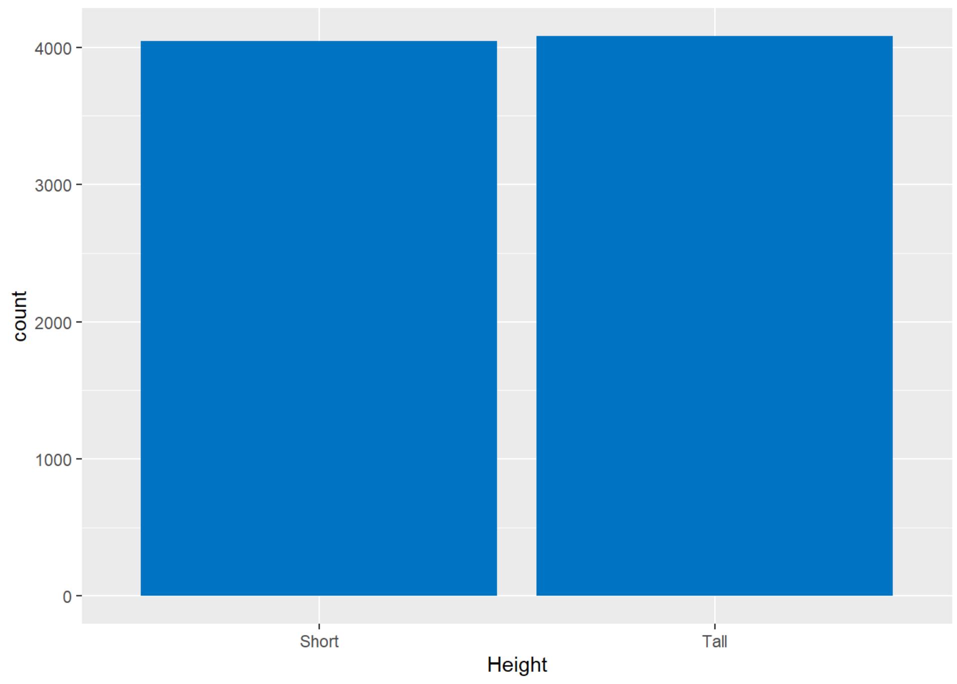
Height

It consists of two items namely short and tall, they both are having nearly equal split in our Data.

```
table(data$Height)
```

```
##
## Short  Tall
##  4043  4081
```

```
ggplot(data, aes(Height))+geom_bar(fill = "#0073C2FF")
```



Data Preprossing

As our data is a categorical we can’t use it for analysis we need to convert the categorical data to factor `as.factor()` will help to convert categorical data to factor

```
data$Edible = as.factor(data$Edible)
data$CapShape = as.factor(data$CapShape)
data$CapSurface = as.factor(data$CapSurface)
data$CapColor = as.factor(data$CapColor)
data$Odor = as.factor(data$Odor)
data$Height = as.factor(data$Height)
str(data)
```

```
## 'data.frame':    8124 obs. of  6 variables:
##  $ Edible    : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 2 1 1 1 1 2 1 ...
##  $ CapShape  : Factor w/ 6 levels "Bell","Conical",..: 3 3 1 3 3 3 1 1 3 1 ...
##  $ CapSurface: Factor w/ 4 levels "Fibrous","Grooves",..: 4 4 4 3 4 3 4 3 3 4 ...
##  $ CapColor  : Factor w/ 10 levels "Brown","Buff",..: 1 10 9 9 4 10 9 9 9 10 ...
##  $ Odor      : Factor w/ 9 levels "Almond","Anise",..: 8 1 2 8 7 1 1 2 8 1 ...
##  $ Height    : Factor w/ 2 levels "Short","Tall": 2 1 2 1 1 1 1 2 2 2 ...
```

After using the factor function we can convert the data to Train dataset and test dataset. we are using **Edible as our traget variable** based on our business problem.

```
train_index <- createDataPartition(data$Edible, p = 0.7, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
print(dim(train_data))
```

```
## [1] 5688    6
```

```
print(dim(test_data))
```

```
## [1] 2436    6
```

from above code we can observe that dataset is split having 70% as training set and 30% is the testing set.

We need to perform modelling for based on using few different columns below code will helps us to give different formulas as a input in here are are considering the Edible as a Traget variables and other columns as input variable.

- Edible ~ CapColor : for this formula we are using Edible as Target and CapColor as Input
- Edible ~ CapSurface : We are using only CapSurface as Input
- Edible ~ CapShape : We are using Capshape as Input
- Edible ~ Odor : We are using Odor as Input
- Edible ~ CapSurface+CapColor+CapShape : We are using CapSurface, CapColor, CapShape as Input variables
- Edible ~ CapSurface+CapColor+CapShape+Height : We are using CapSurface, CapColor, CapShape and Height as Input variables
- Edible ~ CapSurface+CapColor+CapShape+Odor : We are using CapSurface, CapColor, CapShape and Odor as Input variables
- Edible ~ . : We are using All other columns expect Edible as the Input variables.

```
x <- Edible ~ .
x1 <- Edible ~ Odor
x2 <- Edible ~ CapSurface+CapColor+CapShape
x3 <- Edible ~ CapSurface+CapColor+CapShape+Odor
x4 <- Edible ~ CapColor
x5 <- Edible ~ CapSurface
x6 <- Edible ~ CapShape
x7 <- Edible ~ CapSurface+CapColor+CapShape+Height
y <- c(x4,x5,x6,x7,x2,x1,x3,x)
```

Logistic Regression

Algorithm:

1. Initializing for loop to perform model with different formulas
2. Training the model with glm() funtion with Train_Data, as the Target variable is having two values **Edible** or **Poisonous** we are using the family as binomial whioch convert normal model to logistic regression model.
3. Predicting the model on the Test_data, with type as response.
4. Since, the predicted values are in numbers converting it into Edible as Poisonous by keeping threshold as 0.5, prediction values < 0.5 is Edible and > 0.5 is Poisonous as per the method of logistic regression.

5. finding the number of correctly predicted values, accuracy and initializing it to list for further analysis.

```
my_list_log <- c()
original_log <- c()
accuracy_log <- c()
for (i in y){
  model = glm(i, data = train_data, family = binomial)
  predictions <- predict(model, newdata = test_data, type = 'response')
  predictions <- as.factor(ifelse(predictions < 0.5, 'Edible', 'Poisonous'))
  my_list_log <- append(my_list_log,sum(test_data$Edible==predictions))
  original_log <- append(original_log,length(test_data$Edible))
  accuracy_log <- append(accuracy_log,(sum(test_data$Edible == predictions) / length(test_data$Edible)))
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Output:

Below code give the output of logistic regression algorithm in tabular form using Knitr varaiable and Dataframe.

```
x <- "Edible ~ ."
x1 <- "Edible ~ Odor"
x2 <- "Edible ~ CapSurface+CapColor+CapShape"
x3 <- "Edible ~ CapSurface+CapColor+CapShape+Odor"
x4 <- "Edible ~ CapColor"
x5 <- "Edible ~ CapSurface"
x6 <- "Edible ~ CapShape"
x7 <- "Edible ~ CapSurface+CapColor+CapShape+Height"

# Create a data frame with the formulas and predictions
z <- data.frame(Formula = c(x4,x5,x6,x7,x2,x1,x3,x),
                   Test_data = original_log,
                   Predictions = my_list_log,
                   Accuracy = accuracy_log*100)

# Print the table

kable(z, caption = "Table 1: Prediction and Accuracy for respective formulas in logistic Regression")
```

Table 1: Prediction and Accuracy for respective formulas in logistic Regression

Formula	Test_data	Predictions	Accuracy
Edible ~ CapColor	2436	1470	60.34483
Edible ~ CapSurface	2436	1436	58.94910
Edible ~ CapShape	2436	1377	56.52709
Edible ~ CapSurface+CapColor+CapShape+Height	2436	1638	67.24138
Edible ~ CapSurface+CapColor+CapShape	2436	1638	67.24138
Edible ~ Odor	2436	2395	98.31691
Edible ~ CapSurface+CapColor+CapShape+Odor	2436	2415	99.13793
Edible ~ .	2436	2415	99.13793

From above table we can observe that the model with only CapColor, CapSurface, Capshape and all the combined are giving less accuracy when compared with the formula with Order as the input it is giving 98.8 accuracy, Using all the columns as input the Edible column is correctly predicted with 99.17% of accuracy.

Cross validation:

Once we have selected the formula for our model we need to fine tune to get more accurate results and perform cross validation to get justify if our model is good or not. based on the above table I am including only one formula `Edible ~ .` Selecting all the columns as Input except Edible as output.

Algorithm:

1. initialize the Split of train data and test data. I have used 50% TO 90% of splits
2. Running above logistic regression algorithm for single formula `Edible ~ .`
3. finding the number of correctly predicted values and accuracy for each split.

```
train_split = c(0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9)
log_accuracy_split = c()
original_train_log = c()
original_test_log = c()
my_list_split_log = c()
for (i in train_split){
  train_index_split <- createDataPartition(data$Edible, p = i, list = FALSE)
  train_data_split <- data[train_index_split, ]
  test_data_split <- data[-train_index_split, ]
  log_model <- glm(Edible ~ ., data = train_data_split, family = binomial)
  predictions <- predict(model, newdata = test_data_split, type = 'response')
  predictions <- as.factor(ifelse(predictions < 0.5, 'Edible', 'Poisonous'))
  my_list_split_log <- append(my_list_split_log,sum(test_data_split$Edible==predictions))
  original_train_log <- append(original_train_log,length(train_data_split$Edible))
  original_test_log <- append(original_test_log,length(test_data_split$Edible))
  log_accuracy_split <- append(log_accuracy_split,(sum(test_data_split$Edible == predictions) / length(test_data_split$Edibl
e)))

}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Create a data frame with the formulas and predictions
z <- data.frame(train_split = train_split,
                Train_Data = original_train_log,
                   Test_data = original_test_log,
                   Predictions = my_list_split_log,
                   Accuracy = log_accuracy_split*100)

# Print the table
kable(z,caption = "Table 2: Predictions in Logistic regression for different percentage of splits")
```

Table 2: Predictions in Logistic regression for different percentage of splits

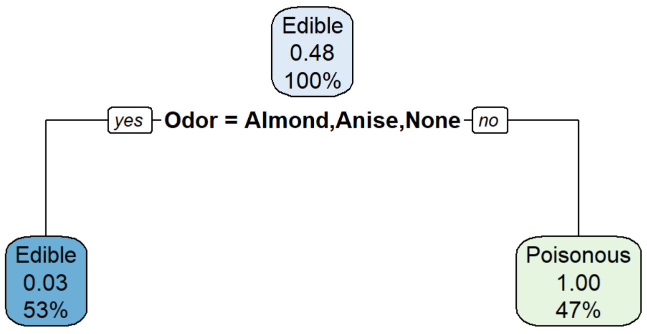
train_split	Train_Data	Test_data	Predictions	Accuracy
0.50	4062	4062	4032	99.26145
0.55	4469	3655	3626	99.20657
0.60	4875	3249	3230	99.41520
0.65	5282	2842	2820	99.22590
0.70	5688	2436	2417	99.22003
0.75	6093	2031	2016	99.26145
0.80	6500	1624	1615	99.44581
0.85	6906	1218	1205	98.93268
0.90	7313	811	806	99.38348

We can observe fromn the above table that the logistic regression is providing the avarage accuracy of 99.3 with train and test split. which is good understanding that the model works good for Mushroom detection.

Decision Tree

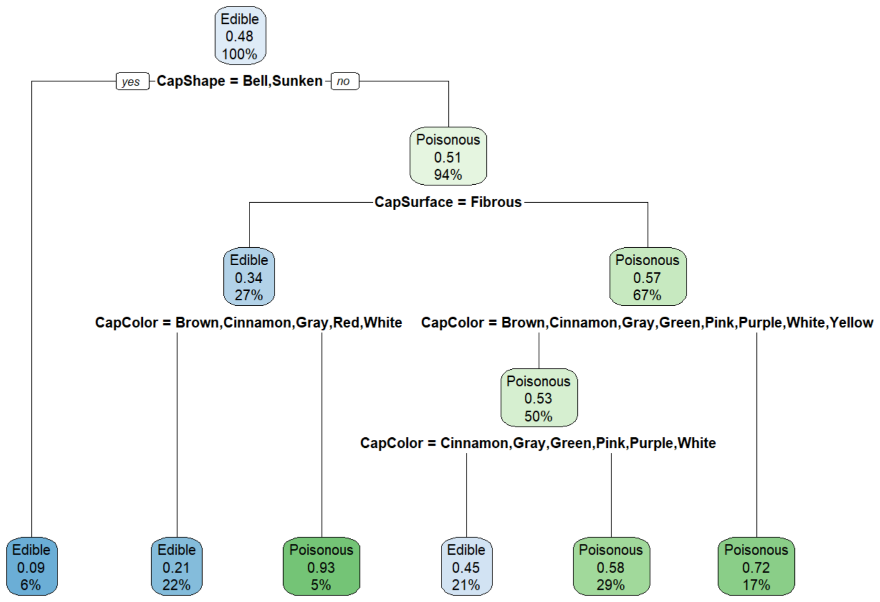
for the decision tree model we need to first plot the tree node splits, below code we helps us to create plot for different formulas and shows there importance:

```
mytree1 = rpart(Edible ~ Odor, data = train_data, method = 'class')
rpart.plot(mytree1)
```



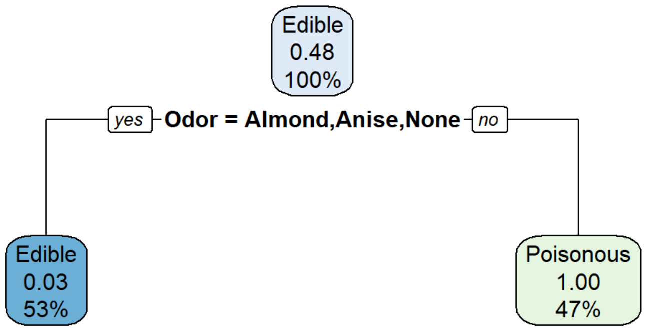
The above plot displays that when we are using the default complexity parameter values and Odor as only input variable we can find the node split is happening once

```
mytree1 = rpart(Edible ~ CapShape+CapColor+CapSurface, data = train_data, method = 'class')
rpart.plot(mytree1)
```



The above plot displays that when we are using the default complexity parameter values and CapShape, CapColor, CapSurface as input variable we can find the node split is using all the input variables present in the Data.

```
mytree1 = rpart(Edible ~ ., data = train_data, method = 'class')
rpart.plot(mytree1)
```

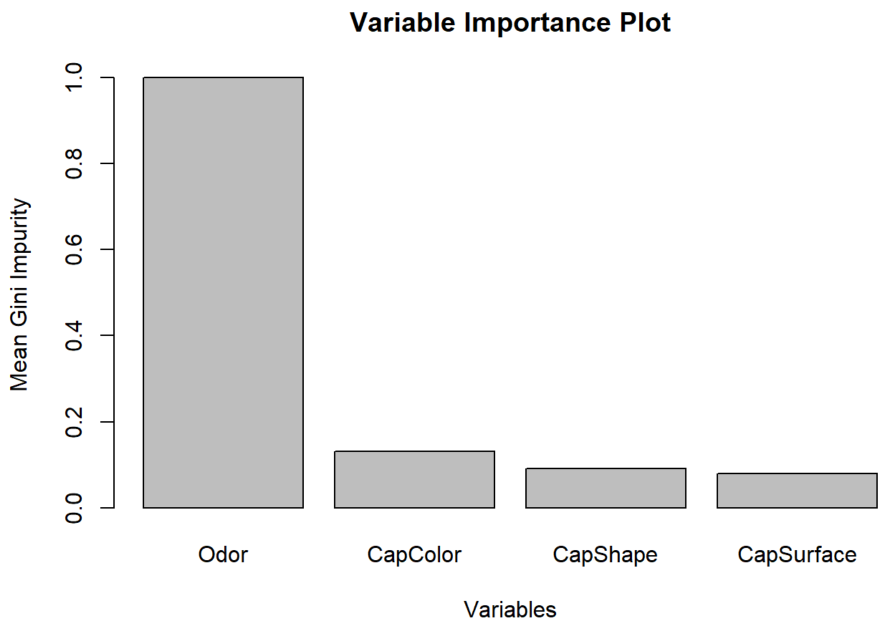


The above plot displays that when we are using the default complexity parameter values and CapShape, CapColor, CapSurface , Odor as input variable we can find the node split is using only single variable, this can be happening because of below reasons:

1. Variable importance of Odor is more compared to other variables
2. Complexity parameter is the default value.

The code give the variable importance of each variable using MeanGini Impurity value for our Data in decreasing order.

```
model <- rpart(Edible ~ ., data = data, method = 'class') #Decision Tree model
variable_importance <- function(model) {
  imp <- model$variable.importance
  imp <- imp / max(imp)  # Normalize importance scores to the maximum value
  imp <- sort(imp, decreasing = TRUE)  # Sort in descending order
  return(imp)
}

# Get variable importance scores based on Gini impurity
importance_scores <- variable_importance(model)

# Plot variable importance based on Gini impurity
barplot(importance_scores, main = "Variable Importance Plot", xlab = "Variables", ylab = "Mean Gini Impurity")
```



From above bar we can observe that the variable importance of Odor is nearly equal to one while others is nearly 0.2, this can be said that Gini impurity of Odor is highest. this is reason why data is only spitted based of Odor variable

Below code displays how complexity parameter value can change use of variables for prediction.

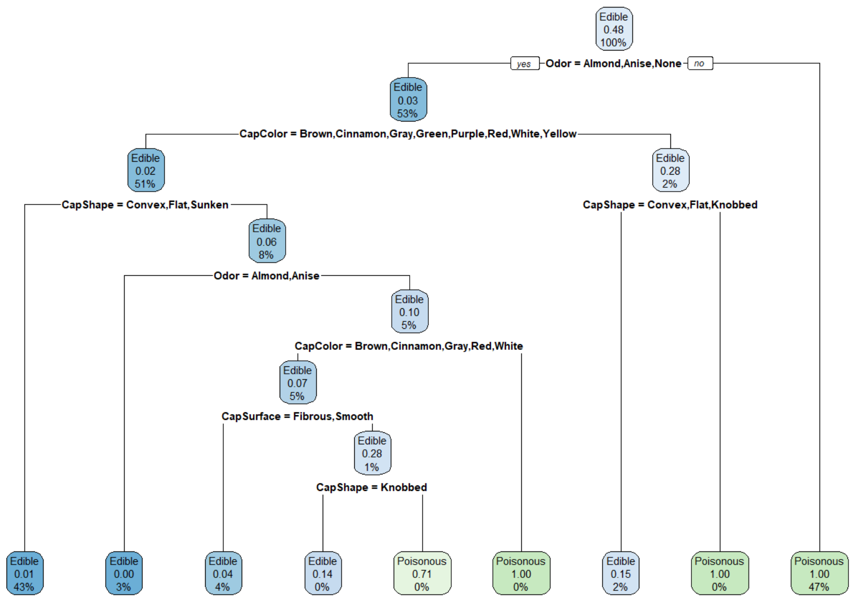
```r
cp_v = c(0.1,0.01,0.000001,0.000002)
for (i in cp_v){
  model <- rpart(Edible ~ ., data = train_data,cp = i)
  predictions <- predict(model, newdata = test_data, type = "class")


# Print the CP table
  cat("\n")
  cat("cp_value used for Decision tree",i,"\n")
  printcp(model)
}
```

```
##
## cp_value used for Decision tree 0.1
##
## Classification tree:
## rpart(formula = Edible ~ ., data = train_data, cp = i)
##
## Variables actually used in tree construction:
## [1] Odor
##
## Root node error: 2742/5688 = 0.48207
##
## n= 5688
##
##        CP nsplit rel error   xerror      xstd
## 1 0.97119      0  1.000000 1.000000 0.0137437
## 2 0.10000      1  0.028811 0.028811 0.0032189
##
## cp_value used for Decision tree 0.01
##
## Classification tree:
## rpart(formula = Edible ~ ., data = train_data, cp = i)
##
## Variables actually used in tree construction:
## [1] Odor
##
## Root node error: 2742/5688 = 0.48207
##
## n= 5688
##
##        CP nsplit rel error   xerror      xstd
## 1 0.97119      0  1.000000 1.000000 0.0137437
## 2 0.01000      1  0.028811 0.028811 0.0032189
##
## cp_value used for Decision tree 1e-06
##
## Classification tree:
## rpart(formula = Edible ~ ., data = train_data, cp = i)
##
## Variables actually used in tree construction:
## [1] CapColor   CapShape   CapSurface Odor
##
## Root node error: 2742/5688 = 0.48207
##
## n= 5688
##
##           CP nsplit rel error   xerror      xstd
## 1 0.97118891      0  1.000000 1.000000 0.0137437
## 2 0.00309993      1  0.028811 0.028811 0.0032189
## 3 0.00133722      3  0.022611 0.022611 0.0028559
## 4 0.00054705      6  0.018600 0.018235 0.0025674
## 5 0.00000100      8  0.017505 0.019694 0.0026672
##
## cp_value used for Decision tree 2e-06
##
## Classification tree:
## rpart(formula = Edible ~ ., data = train_data, cp = i)
##
## Variables actually used in tree construction:
## [1] CapColor   CapShape   CapSurface Odor
##
## Root node error: 2742/5688 = 0.48207
##
## n= 5688
##
##           CP nsplit rel error   xerror      xstd
## 1 0.97118891      0  1.000000 1.000000 0.0137437
## 2 0.00309993      1  0.028811 0.028811 0.0032189
## 3 0.00133722      3  0.022611 0.022247 0.0028331
## 4 0.00054705      6  0.018600 0.018964 0.0026178
## 5 0.00000200      8  0.017505 0.019329 0.0026426
```

From above output we can observe that the complexity parameter for 0.1, 0.01 are using only one variable Odor for the predictions while values 0.0000001, 0.0000002 are using all the values except height for classification of the Data.

This can be visualized by below R plot.

```
mytree1 = rpart(Edible ~ CapShape+CapColor+CapSurface+Odor, data = train_data, method = 'class',cp = 0.0000001)
rpart.plot(mytree1)
```



Compared to all the above plots this graph is using its all the variables in predicting the values, this happens due to the complexity score as 0.0000001

Algorithm:

1. Initializing for loop to perform model with different formulas, and complexity parameter as 0.0000001
2. Training the model with rpart() funtion with Train_Data, and with all the above formulas as specified.
3. Predicting the model on the Test_data, with type as class.
4. finding the number of correctly predicted values, accuracy and initializing it to list for further analysis.

```
my_list_dt <- c()
original_dt <- c()
accuracy_dt <- c()
for (i in y){
  model = rpart(i, data = train_data,cp=0.0000001)
  predictions <- predict(model, newdata = test_data, type = "class")
  my_list_dt <- append(my_list_dt,sum(test_data$Edible==predictions))
  original_dt <- append(original_dt,length(test_data$Edible))
  accuracy_dt <- append(accuracy_dt,(sum(test_data$Edible == predictions) / length(test_data$Edible)))
}
```

Output:

Below code give the output of Decision tree algorithm in tabular form using Knitr varaiable and Dataframe.

```
x <- "Edible ~ ."
x1 <- "Edible ~ Odor"
x2 <- "Edible ~ CapSurface+CapColor+CapShape"
x3 <- "Edible ~ CapSurface+CapColor+CapShape+Odor"
x4 <- "Edible ~ CapColor"
x5 <- "Edible ~ CapSurface"
x6 <- "Edible ~ CapShape"
x7 <- "Edible ~ CapSurface+CapColor+CapShape+Height"

# Create a data frame with the formulas and predictions
z <- data.frame(Formula = c(x4,x5,x6,x7,x2,x1,x3,x),
                Test_data = original_dt,
                Predictions = my_list_dt,
                Accuracy = accuracy_dt*100)

# Print the table
kable(z,caption = "Table 3: Prediction and Accuracy Decision tree with different formulas")
```

Table 3: Prediction and Accuracy Decision tree with different formulas

Formula	Test_data	Predictions	Accuracy

Formula	Test_data	Predictions	Accuracy
Edible ~ CapColor	2436	1470	60.34483
Edible ~ CapSurface	2436	1436	58.94910
Edible ~ CapShape	2436	1377	56.52709
Edible ~ CapSurface+CapColor+CapShape+Height	2436	1711	70.23810
Edible ~ CapSurface+CapColor+CapShape	2436	1718	70.52545
Edible ~ Odor	2436	2395	98.31691
Edible ~ CapSurface+CapColor+CapShape+Odor	2436	2412	99.01478
Edible ~ .	2436	2412	99.01478

From above table we can observe that the model with only CapColor, CapSurface, Capshape and all the combined are giving less accuracy when compared with the formula with Order as the input it is giving 98.8 accuracy, Using all the columns as input the Edible column is correctly predicted with 99.8% of accuracy.

Cross validation:

We are performing the cross validation using different test and train split on the decision tree model from the below code.

```
train_split = c(0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9)
dt_accuracy_split = c()
original_train_dt_split = c()
original_test_dt_split = c()
my_list_split_dt = c()
for (i in train_split){
  train_index_dt <- createDataPartition(data$Edible, p = i, list = FALSE)

  train_data_split <- data[train_index_dt, ]

  test_data_split <- data[-train_index_dt, ]

  model = rpart(Edible ~ CapSurface+CapColor+CapShape+Odor, data = train_data_split)

  predictions <- predict(model, newdata = test_data_split, type = "class")

  my_list_split_dt <- append(my_list_split_dt,sum(test_data_split$Edible==predictions))

  original_train_dt_split <- append(original_train_dt_split,length(train_data_split$Edible))

  original_test_dt_split <- append(original_test_dt_split,length(test_data_split$Edible))

  dt_accuracy_split <- append(dt_accuracy_split,(sum(test_data_split$Edible == predictions) / length(test_data_split$Edibl
e)))

}
```

Output

Output is shown from the below code for respective test and train splits.

```
# Create a data frame with the formulas and predictions
z <- data.frame(train_split = train_split,
                Train_Data = original_train_dt_split,
                   Test_data = original_test_dt_split,
                   Predictions = my_list_split_dt,
                   Accuracy = dt_accuracy_split*100)

# Print the table
kable(z,caption = "Table 4: Prediction and Accuracy Decision tree with different splits")
```

Table 4: Prediction and Accuracy Decision tree with different splits

train_split	Train_Data	Test_data	Predictions	Accuracy
0.50	4062	4062	4013	98.79370
0.55	4469	3655	3600	98.49521
0.60	4875	3249	3203	98.58418
0.65	5282	2842	2794	98.31105
0.70	5688	2436	2401	98.56322

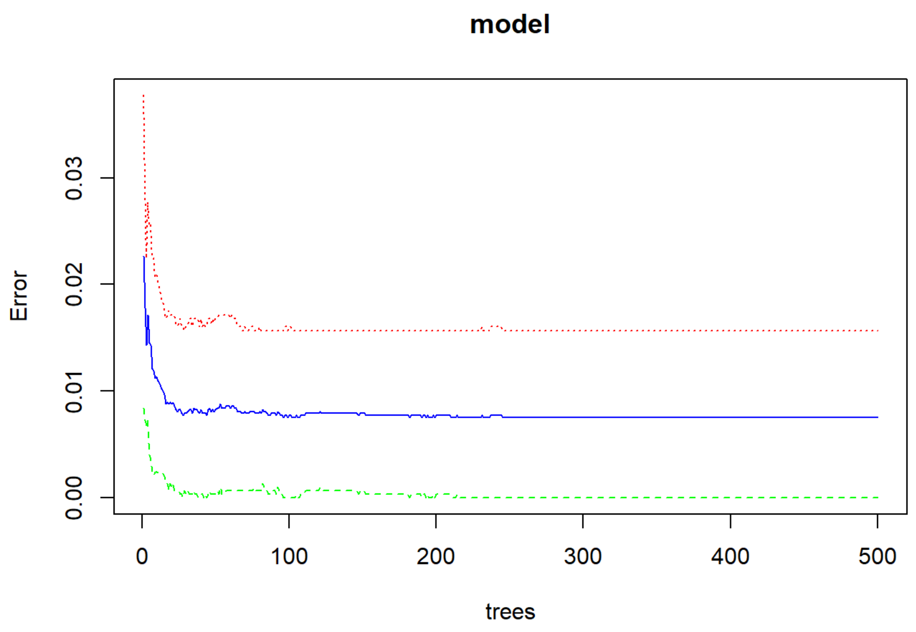
train_split	Train_Data	Test_data	Predictions	Accuracy
0.75	6093	2031	2014	99.16297
0.80	6500	1624	1606	98.89163
0.85	6906	1218	1195	98.11166
0.90	7313	811	799	98.52035

from above output we can observe that the average accuracy for Decision tree is 98.8 for formula `Edible~`.

Random Forest

Initially, we are ploting the Random forest model with our train data with number of trees as 500 to get an assumption and left the mtry as the default.

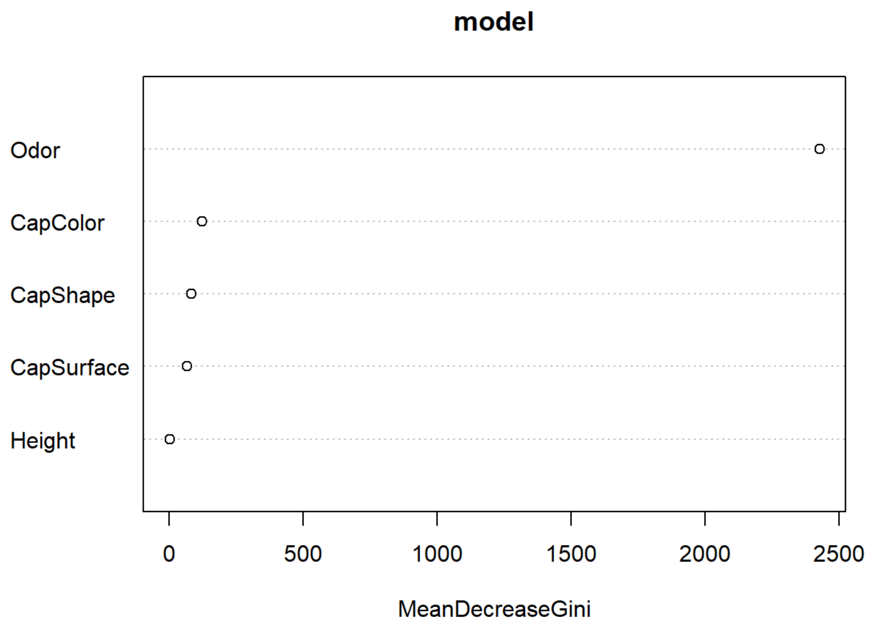
```
model <- randomForest(Edible ~., data = train_data, ntree = 500)
plot(model, col = c("blue", "green", "red"))
```



from above graph we can observe that the error is getting decreased from number of trees as 100 I am using the number of trees as 100 as per the above graph once iot is done we need to consider the variable importance using Gini Impurity

Below code plots the Variable importance using `varImpPlot()` funtion

```
model <- randomForest(Edible ~ ., data = train_data, ntree = 100)

# Create the variable importance plot
varImpPlot(model)
```



The model is also considering the same Gini impurity with Odor as the heighest, Height as the lowest

Algorithm:

1. Initializing for loop to perform model with different formulas
2. Training the model with randomForest() funtion with Train_Data, and with all the above formulas as specified, specifying ntree, mtry.
3. Predicting the model on the Test_data, with type as class.
4. finding the number of correctly predicted values, accuracy and initializing it to list for further analysis.

```
rf_my_list <- c()
rf_original <- c()
rf_accuracy <- c()
for (i in y){
  model = randomForest(i, data = train_data, ntree = 100)
  predictions <- predict(model, newdata = test_data, type = "class")
  rf_my_list <- append(rf_my_list,sum(test_data$Edible==predictions))
  rf_original <- append(rf_original,length(test_data$Edible))
  rf_accuracy <- append(rf_accuracy,(sum(test_data$Edible == predictions) / length(test_data$Edible)))
}
```

Output:

Below code give the output of Random Forest algorithm in tabular form using Knitr varaiable and Dataframe.

```
x <- "Edible ~ ."
x1 <- "Edible ~ Odor"
x2 <- "Edible ~ CapSurface+CapColor+CapShape"
x3 <- "Edible ~ CapSurface+CapColor+CapShape+Odor"
x4 <- "Edible ~ CapColor"
x5 <- "Edible ~ CapSurface"
x6 <- "Edible ~ CapShape"
x7 <- "Edible ~ CapSurface+CapColor+CapShape+Height"

# Create a data frame with the formulas and predictions
z <- data.frame(Formula = c(x4,x5,x6,x7,x2,x1,x3,x),
                   Test_data = rf_original,
                   Predictions = rf_my_list,
                   Accuracy = rf_accuracy*100)

# Print the table
kable(z,format = "simple",caption = "Table 5: Prediction and Accuracy Random forest  with different Formulas")
```

Table 5: Prediction and Accuracy Random forest with different Formulas

Formula	Test_data	Predictions	Accuracy
Edible ~ CapColor	2436	1470	60.34483
Edible ~ CapSurface	2436	1436	58.94910
Edible ~ CapShape	2436	1377	56.52709

Formula	Test_data	Predictions	Accuracy
Edible ~ CapSurface+CapColor+CapShape+Height	2436	1712	70.27915
Edible ~ CapSurface+CapColor+CapShape	2436	1712	70.27915
Edible ~ Odor	2436	2395	98.31691
Edible ~ CapSurface+CapColor+CapShape+Odor	2436	2415	99.13793
Edible ~ .	2436	2415	99.13793

From above table we can observe that the model with only CapColor, CapSurface, Capshape and all the combined are giving less accuracy when compared with the formula with Order as the input it is giving 98.8 accuracy, Using all the columns as input the Edible column is correctly predicted with 98.8% of accuracy.

Cross validation:

Performing crass validation using different test and train splits.

```
train_split = c(0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9)
rf_accuracy_split = c()
original_train_rf = c()
original_test_rf = c()
my_list_rf = c()
for (i in train_split){
  train_index_rf <- createDataPartition(data$Edible, p = i, list = FALSE)
  train_data_rf <- data[train_index_rf, ]
  test_data_rf <- data[-train_index_rf, ]
  model = randomForest(Edible ~ CapSurface+CapColor+CapShape+Odor, data = train_data_rf, ntree = 100)
  predictions <- predict(model, newdata = test_data_rf, type = "class")
  my_list_rf <- append(my_list_rf,sum(test_data_rf$Edible==predictions))
  original_train_rf <- append(original_train_rf,length(train_data_rf$Edible))
  original_test_rf <- append(original_test_rf,length(test_data_rf$Edible))
  rf_accuracy_split <- append(rf_accuracy_split,(sum(test_data_rf$Edible == predictions) / length(test_data_rf$Edible)))
}
```

```
# Create a data frame with the formulas and predictions
z <- data.frame(train_split = train_split,
                Train_Data = original_train_rf,
                   Test_data = original_test_rf,
                   Predictions = my_list_rf,
                   Accuracy = rf_accuracy_split*100)

# Print the table
kable(z,caption = "Table 6: Prediction and Accuracy Random Forest with different splits",format = 'simple')
```

Table 6: Prediction and Accuracy Random Forest with different splits

train_split	Train_Data	Test_data	Predictions	Accuracy
0.50	4062	4062	4026	99.11374
0.55	4469	3655	3626	99.20657
0.60	4875	3249	3229	99.38443
0.65	5282	2842	2813	98.97959
0.70	5688	2436	2422	99.42529
0.75	6093	2031	2011	99.01526
0.80	6500	1624	1619	99.69212
0.85	6906	1218	1207	99.09688
0.90	7313	811	804	99.13687

We can observe from above table that average accuracy of Random Forest in nearly 99.2 which is lowest when compared to logistic regression.

Evaluation

For evalution of the model let us consider the number of miss classified Mushroom that is Poisounous mushrroms are detected as the Edible Mushrooms which is dangerous to health. for this we are using confusion Matrix for each model with Formula as `Edible ~ .`

Confusion Matrix for logistic Regression

```r
train_index <- createDataPartition(data$Edible, p = 0.7, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
for(i in y){
  model = glm(i, data = train_data, family = binomial)
  predictions <- predict(model, newdata = test_data, type = 'response')
  predictions <- as.factor(ifelse(predictions < 0.5, 'Edible', 'Poisonous'))
  cat("\n")
  print(i)
  print(table(Predicted = predictions,actual = test_data$Edible))
}
```

```
##
## Edible ~ CapColor
##            actual
## Predicted   Edible Poisonous
##   Edible       931       636
##   Poisonous    331       538
##
## Edible ~ CapSurface
##            actual
## Predicted   Edible Poisonous
##   Edible       454       233
##   Poisonous    808       941
##
## Edible ~ CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible      1190       994
##   Poisonous     72       180
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Edible ~ CapSurface + CapColor + CapShape + Height
##            actual
## Predicted   Edible Poisonous
##   Edible       823       398
##   Poisonous    439       776
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Edible ~ CapSurface + CapColor + CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible       823       398
##   Poisonous    439       776
##
## Edible ~ Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1262        34
##   Poisonous      0      1140
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Edible ~ CapSurface + CapColor + CapShape + Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1250        12
##   Poisonous     12      1162
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Edible ~ .
##            actual
## Predicted   Edible Poisonous
##   Edible      1250        12
##   Poisonous     12      1162
```

We can observe for the different formulas type 1 error of logistic regression is as follows.

Confusion Matrix for Random Forest

```
train_index <- createDataPartition(data$Edible, p = 0.7, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
for(i in y){
  model = randomForest(i, data = train_data, ntree = 100)
  predictions <- predict(model, newdata = test_data, type = "class")
  cat("\n")
  print(i)
  print(table(Predicted = predictions,actual = test_data$Edible))
}
```

```
##
## Edible ~ CapColor
##            actual
## Predicted   Edible Poisonous
##   Edible       947       661
##   Poisonous    315       513
##
## Edible ~ CapSurface
##            actual
## Predicted   Edible Poisonous
##   Edible       476       244
##   Poisonous    786       930
##
## Edible ~ CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible      1192      1003
##   Poisonous     70       171
##
## Edible ~ CapSurface + CapColor + CapShape + Height
##            actual
## Predicted   Edible Poisonous
##   Edible       909       352
##   Poisonous    353       822
##
## Edible ~ CapSurface + CapColor + CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible       748       229
##   Poisonous    514       945
##
## Edible ~ Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1262        34
##   Poisonous      0      1140
##
## Edible ~ CapSurface + CapColor + CapShape + Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1262        20
##   Poisonous      0      1154
##
## Edible ~ .
##            actual
## Predicted   Edible Poisonous
##   Edible      1262        21
##   Poisonous      0      1153
```

for Random Forest Model the number of Type 1 error(Predicting Mushroom as Edible when it is actually Poisonous) is more when compared to Logistic regression Model.

Confusion Matrix for Descision Tree

```
train_index <- createDataPartition(data$Edible, p = 0.7, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
for(i in y){
  model = rpart(i, data = train_data, cp = 0.0000001)
  predictions <- predict(model, newdata = test_data, type = "class")
  table(predictions,test_data$Edible)
  cat("\n")
  print(i)
  print(table(Predicted = predictions,actual = test_data$Edible))
}
```

```
##
## Edible ~ CapColor
##            actual
## Predicted   Edible Poisonous
##   Edible       909       648
##   Poisonous    353       526
##
## Edible ~ CapSurface
##            actual
## Predicted   Edible Poisonous
##   Edible       451       249
##   Poisonous    811       925
##
## Edible ~ CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible      1176       993
##   Poisonous     86       181
##
## Edible ~ CapSurface + CapColor + CapShape + Height
##            actual
## Predicted   Edible Poisonous
##   Edible       919       350
##   Poisonous    343       824
##
## Edible ~ CapSurface + CapColor + CapShape
##            actual
## Predicted   Edible Poisonous
##   Edible       920       357
##   Poisonous    342       817
##
## Edible ~ Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1262        31
##   Poisonous      0      1143
##
## Edible ~ CapSurface + CapColor + CapShape + Odor
##            actual
## Predicted   Edible Poisonous
##   Edible      1260        17
##   Poisonous      2      1157
##
## Edible ~ .
##            actual
## Predicted   Edible Poisonous
##   Edible      1260        17
##   Poisonous      2      1157
```

for Decision tree Model the number of Type 1 error(Predicting Mushroom as Edible when it is actually Poisonous) is more when compared to Random forest this specifies Decision tree is not a suitable approch.

```
x <- "Edible ~ ."
x1 <- "Edible ~ Odor"
x2 <- "Edible ~ CapSurface+CapColor+CapShape"
x3 <- "Edible ~ CapSurface+CapColor+CapShape+Odor"
x4 <- "Edible ~ CapColor"
x5 <- "Edible ~ CapSurface"
x6 <- "Edible ~ CapShape"
x7 <- "Edible ~ CapSurface+CapColor+CapShape+Height"

# Create a data frame with the formulas and predictions
z <- data.frame(Formula = c(x4,x5,x6,x7,x2,x1,x3,x),
                Test_data = original_dt,
                Log_Prediction = my_list_log,
                Log_accuracy = accuracy_log*100,
                DT_Predictions = my_list_dt,
                DT_accuracy = accuracy_dt*100,
                RF_Predictions = rf_my_list,
                RF_Accuracy = rf_accuracy*100
                   )

# Print the table
kable(z,caption = "Table 7: Prediction and Accuracy with different formulas for all the models",format = "simple")
```

Table 7: Prediction and Accuracy with different formulas for all the models

Formula	Test_data	Log_Prediction	Log_accuracy	DT_Predictions	DT_accuracy	RF_Predictions	RF_Accur
Edible ~ CapColor	2436	1470	60.34483	1470	60.34483	1470	60.34
Edible ~ CapSurface	2436	1436	58.94910	1436	58.94910	1436	58.94
Edible ~ CapShape	2436	1377	56.52709	1377	56.52709	1377	56.52
Edible ~ CapSurface+CapColor+CapShape+Height	2436	1638	67.24138	1711	70.23810	1712	70.27
Edible ~ CapSurface+CapColor+CapShape	2436	1638	67.24138	1718	70.52545	1712	70.27
Edible ~ Odor	2436	2395	98.31691	2395	98.31691	2395	98.31
Edible ~ CapSurface+CapColor+CapShape+Odor	2436	2415	99.13793	2412	99.01478	2415	99.13
Edible ~ .	2436	2415	99.13793	2412	99.01478	2415	99.13

Conclusion:

In this report we have performed the logistic regression, Decision Tree and Random forest Model with selecting different rows as input and output variables, We have fine tunned the Random forest using the number of trees and Decisiion tree using the Complexity parameter value, We further performed cross validation technique to observe how the Test and train split can impact the number of correctly predicted values.

Based, upon the observed analysis I would like to suggest that Logistic regression as the best model since it is giving less Type 1 error which is Predicting Mushrrom as edible when it is actually Poisonous campared to Random forest and Decision tree Models