An investigation into the ability of a machine learning algorithm to work out if a mushroom is edible or not, given certain factors.

https://github.com/EyeBall64/MachineLearningAssessment2

Table of contents

Introduction

When I was younger, I used to go walking in the woods around my na’s house all the time, which would have many different types of mushrooms present. My nan always used to tell me to be careful about them all since they could be poisonous. I knew not all of them were, but I wanted to be able to tell which once were for my own curiosity. With this project, i hopefully will.
I will be using many different methods, such as random forests and GLM functions, among others, to get a good idea of whether a mushroom from the data set that I have found is edible.
In the R code, I use a maximum of 80 characters on a line for easy readability, and I am also to of some missing values that I will explore in the data cleaning.

Data cleaning

Lines 82 - 221

Data cleaning is an essential part of the process. It ensures that the algorithms that I am applying to it will actually work. Lots of data will have errors or entries without any information. This can cause problems with some of the algorithms; even if they do run properly, it could cause erroneous results, as it is based on incorrect data.
Luckily, the most glaring problem was highlighted in the information that was given with the data; many entries in the stalk-root feature were NA. I decided that the most logical way of dealing with this was by removing all rows with NAs in; this removed quite a few, but the vast majority still remained, with over 5,000 data points still ready to use.
I wanted to keep an original,I unedited data file in the program, so should if I need to go back after changing something, I could easily do that without having to reload the original data file.
One of the first things that I noticed and decided to change was that all of the data had been copied in as single-character codes. In the information document, the creator of the data file states that this was done to massively reduce the size of the document. However, I thought that it would be a lot easier to handle and observe the data if the original meanings behind the characters were restored. So, quite a lot of the data cleaning segment is devoted to this.
I also noticed that some of the data could be better expressed as logical and numerical variables, so while I was editing the above, I also changed these.
The last thing that I do in this section is check if there are any duplicate rows, which can be a good indicator of any data that has been incorrectly copied. Luckily, there were no duchies.
As it turns out, the original creator of this data set did a very good job of ensuring that all the data was well stored and checked very well, so that there were no errors and only one set of missing variables.

Pie charts

Lines 222 - 429

I decided that there was so much data to deal with, I wanted a way to get a good idea of the data points so that I was able to make more informed decisions about the different machine learning algorithms I were to employ later on. While I was making the pie charts, I used a function that I created and defined in the library section of the code. As well as this, I also created a vector of each of the factors, which contained every variable that they could contain. This proved useful in this section, as I used it when making the different pie charts.
The First set of pie as seen in Figure 1.1 charts that I created all looked at the physical characteristics of the cap. These would be some of the most useful data, as they are the most distinct part of the mushroom. The pie charts showed me that the shape was dominated by a couple of types of shapes, the surface texture was relatively evenly distributed except for grooves, and the colours varied greatly.
The second set of pie charts, as seen in Figure 1.2, shows a variety of characteristics. The bruises are one of the features that I noted in the data cleaning section as being better suited as a logical variable type. The odour shows that about half have no odour, about a quarter have a foul odour, and the rest are distributed relatively evenly. This also shows that the spore print colour is mainly split between 3 colours.
In Figure 1.3, we see information about the gill-related data. Interestingly, most of them are very one-sided, with free, close and broad dominating the pie charts of gill attachment, spacing and size, respectively. The colours, on the other hand, are very well spread out, with a wide variety of each of them.
The next image, as shown in Figure 1.4, has very similar pie charts for both types of above/below data. It may be logical to assume that these are all from the same mushroom, i.e a stalk that is silky above the surface is also silky below the surface, but while this seems logical, these pie charts do not show us significant data to support this. Interesting to note that the stalk shape is very evenly split between the two options.
The next 4 pie charts, seen in Figure 1.5, describe characteristics of the veil and ring, two parts of the mushroom that can be found on the stalk beneath the cap. Again, most of these pie charts are almost completely dominated by 1 variable, and even the 4th is mostly a pendant ring type.
Figure 1.6 shows, among others, the most important data point: whether it is edible or not. We can see that although there are a lot more that are edible, it is not completely dominated by one type, as seen previously. This is very good, as it would be difficult to classify if there were very few data points of one type. This picture also shows the population and habitat, which, like others, have a couple of variable types that are much more prominent and a few that occur a lot less frequently.
Overall, the pie charts have shown us a lot of information about the proportions of data. It is possible that the features that are very dominated by one data type could become a problem in future classification or could be completely ignored. The pie charts do not give us a good idea of how the data points interact with each other, which is important to try to understand.

Probability tables

Lines 430 - 464

I wanted to get a good idea of how some of the features interact with each other so that I was better informed for later algorithms that I could use. Most importantly was to compare them to the edible feature, as that was the main one that I needed to analyse.
The first table, as seen in Figure 2.1, tells us the different amounts of each type of cap shape that are edible or not. Figure 2.2 has used the prob.table function to convert this information into percentages. From both of these, we can see that most of the numbers are not that significant either way unluse there are very few of that specific type of variable, so it is less reliable. For example, in Figure 2.2, we see that if the cap shape is conical, there is a 100% chance of the mushroom not being edible; however, since there are only 4 entries of this type, it's questionable how valid this result is.
A similar thing can be seen in Figures 2.3 and 2.4; if the cap surface is described as ‘grooves’, then there is a 100% chance that the mushroom is not edible, but again, there are only 4 entries. Most of the other types with more entries are closer to 50%, suggesting that they will not be particularly good at suggesting whether a mushroom is edible.
Lastly, in Figures 2.5 and 2.6, we see that the colour of the cap can vary greatly, but is not specifically dominated by one colour, nor are most of them heavily weighted as edible or not. Interestingly, there is the expectation of Goldenrod4, which has over 100 entries, but all of them are inedible, which could suggest it will be important in later analysis.
The probability tables can be good at suggesting what variable may be useful in later algorithms to check for the edibility of mushrooms; however, in these tables, we can only compare edibility to one feature. Ideally, we want to compare if a mushroom is edible or not to as many features as possible to create a good idea of if th mushroom is edible.

Cross-validation and bootstrap

Lines 465 - 483

In this section, I use linear regression using the cap colour as the predictor, which, from the probability tables above, I thought would be a reality good predictor of edibility. First is to split the data into a test and training set, where 50% of the data is in each. I use the set.seed so that the results are random by reproducible. I then create the model and analyse it by looking at the mean squared error, which is very low at 0.1825958.
To ensure that this is not a coincidence, I do the same again with a different set of random training and testing data. Again, I get a very similar and low MSE of 0.1819106. These numbers being this low means that the model has low error, making quite accurate predictions. The fact that they are very similar also implies that there is low variance.
However, I also think there are better functions that could be able to make better type of predictions on this type of data.

Glm function

Lines 484 - 584

I decided to use a GLM function to help make predictions on this set of data. It's probably more suited to data of this type than linear regression and can also easily be converted to a graph to help make visualisation easier.
In Figure 3.1, I make a jitter plot comparing the colour of the caps to their edibility, from the tables above, as well as the linear regression functions, I find hints that these are likely to be correlated. I next use the GLM function and create a summary of the fit that can be seen in Figure 3.2. The significance code shows that almost all are highly significant in the data, with the exception of goldenrod,4, which has no significance; it also has an incredibly high standard error and p-values. This makes sense given what we've seen in the previous table, where although there are some number of goldenrod4 caps, they are not all edible and therefore not very likely to be useful to the data.
In Figure 3.3, I’ve plotted the data above when the function is true so that the points we see are only true. However, this does not show us a great deal as the plot looks the same in Figures 3.1 and 3.3. The summary table for this is also fairly similar, although thought the big outlier of goldenrod4, this is most likely because all values are ot eadbily, so the data for it does not come into play.
I realise that the data would be a lot more useful if all the features could be included, so I attempt to use the GLM function to make a prediction with all features as predictors. Unfortunately, the algorithm did not converge, most likely because it was trying to use all features, even those that were not helpful. Figure 3.5.1 and 3.5.2 show the summary of the function above. There are plenty of NAs since it did not converge. This model can, however, be used to help find the chances of a specific mushroom being edible, although I would question its accuracy. I input various variables for each feature, and it tells me that there is a 2.900715e-12 chance of the mushroom being edible.
Even though I question the validity of this method for this type of data, given the previous results, I want to look at the hnp function to check the accuracy of the model. I use specifically the stalkColorBelowRing and stalkSurfaceBelowRing features, both independently, in Features 3.6 and 3.7 and together in Figure 3.8. Although the graphs stay almost completely within the envelopes, they are not very close to the top right-hand corner, which makes the validity of these graphs questionable.

Leave-one-out Cross-Validation

Lines 585 - 609

Leave-one-out cross-validation looks at a reference variable and then compares the rest of the variables within a specific feature against another feature. In this case, I am comparing the cap shape to the edible to find out if there is much of a correlation between them.
The reference variable in this case is bell-shaped, and almost all of the variables are less likely to be edible than the bell shape, the only exception being sunken. This can be seen in Figure 4.1.
The error estimate of 0.2407 is relatively low; however, not as low as previously seen in other algorithms.
As with previous algorithms that I have used, it would be much more useful if we were able to look at different features at once, so I am not going to go into as much depth with this type of algorithm as I could.

Fitting trees

Lines 610 - 677

The algorithm involved in fitting trees requires all values to be represented as factors so as not to mess up the original data files. I create a new one and convert all features, regardless of the original data type, into factors.
As I have precisely done, I use a seeded random number to create a reproducible random set for training and testing. I then create a tree, in Figure 5.1, I am told the values that are used in the tree, which are quite few, as well as there not being many nodes. In Figures 5.2 and 5.3, we can see the tree made as a test list as well as a diagram. There are very few features that are taken into account, which could be a problem, but it could imply that these features are very decisive for the data set.
To check that there is nothing wrong going on, I look at a table to check the accuracy of the tree that has been created against the test data. Interestingly, the accuracy is 100%; there are no false positives or false negatives.
Despite this, I would still like to check if there is an option to prune the data. We can see in Figure 5.5 that there is not really a good case for pruning the tree; the number of current nodes is sufficient. Indeed, in Figure 5.6, we can see that the tree is identical to the tree in 5.,2 so there is no reason for the trees to be pruned.
There could be a slight concern for over-fitting since the data is 100% correct, but if this is the case, then the whole data file would be the problem, as the test data still gives 100% accuracy as well.

Random forests

Lines 678 - 703

I think that it would be a good idea to also create random forests. These can be more robust prediction methods than just fitting trees, as they can take into account much more of the data and will compare different trees that are created.
In Figure 6.1, we can see that the forest contains 500 trees with 21 variables tried at each fit. These are all the different factors in the data file, which is what we want. Like the previous tree that was made, the forest has a 100% accuracy rating. This could imply that there are, in fact, factors that describe the edibility of a given mushroom very well; this is certainly true within our data set.
I then create a plot, as seen in Figure 6.2, to show the importance of each feature for constructing the above trees. It turns out that the odour is by far the most important feature, with others trailing behind until the least important feature of gill attachment is reached. Interestingly, the values of cap colour are relatively low and are significantly smaller than odour in predicting the edibility of a mushroom, describing the original guesses that I made that assumed the cap colour was fairly useful. This just goes to show the importance of finding the right method to describe the data given, and that methods that only looked at one type of variable were not as useful.

Classification forests

Lines 704 - 756

I split the data into test and training again, using a seeded random number so that the value would be reproducible. I create a classification forest using the rpart function, and then I immediately create a confusion matrix, seen in Figure 7.1, with statistics to look at the forests that have been created. This tree misclassifies 11 mushrooms as poisonous when they were really edible. This is very accurate (and for someone who is trying to use this algorithm, it's good that it's on the safe side). The Kappa value and accuracy show how well this model does, and from a quick observation doesn't look quite as susceptible to overfitting. Most values are close to 1, which is very good.
I create a new forest to compare the confusion matrix with the one above. This forest has a 100% accuracy, no misclassifications unfortunately, even though this looks like a good thing it it could be a result of over-fitting, it could be that the data is very reliant on a few data points, as seen in some of the trees above. This can be seen in Figure 7.2
I then compare different trees using different random subsets of the data file to make sure that the prediction is very robust, as seen in Figure 7.3. It shows that that opposed to what I'd initially thought, the data is unlikely to be overfitting; instead, there are just a few elements that can describe the edibility of the mushrooms in the data set very well. However, there is a chance that this would be specific to the mushrooms within this data set and could not be used on other sets of data.
These show that although it may seem like the models are subject to overfitting, this is unlikely within the data set; if other data sets were to be used, this could be different.

-

Conclusion

Most of the methods that I tried were somewhat useful, although the best ones were GLM, trees and forests. The pie charts and probability tables, while good for getting an initial idea of the data, were not particularly useful for any analysis of it, as they did not show much and could not represent more than 2 features interacting with each other.
Some of the other methods, such as the linear regression, cross-validation and bootstrap were not very useful for various reasons, such as failing to account for more than 1 other factor and not being useful for categorical data.
The GLM functions were definitely helpful, but they had the problem of scale. I wanted to analyse more than one feature against edible, so I tried to do all of them. This just meant that the function did not converge, so it didn’t show me anything.
The trees were very useful in showing the way of categorising edible mushrooms from non-edible mushrooms and but it was always questionable how accurate they were, especially with how they appear to have 100% accuracy.
The forests seemed to alleviate the above problem, and so I think that they where some of the best algorithms that I had for analysing this data.

Images

1. Pie Charts

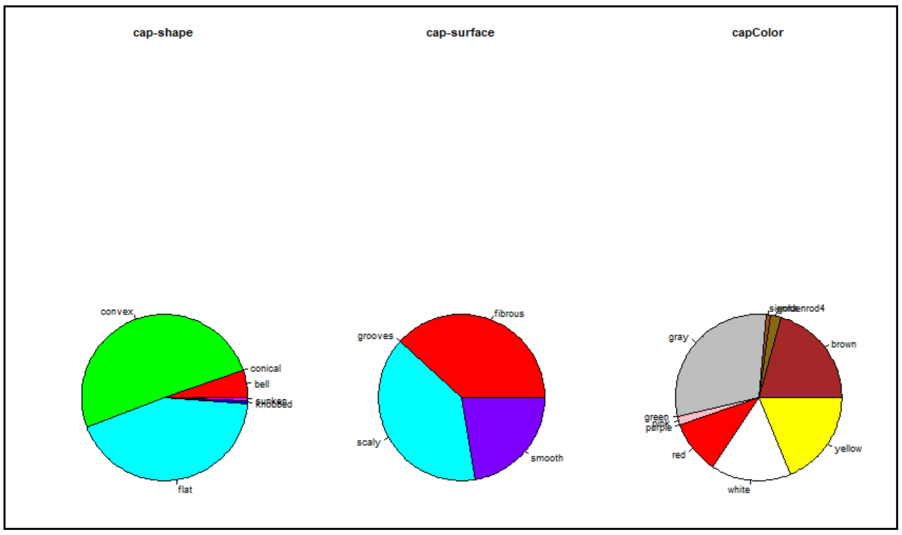


Fig 1.1

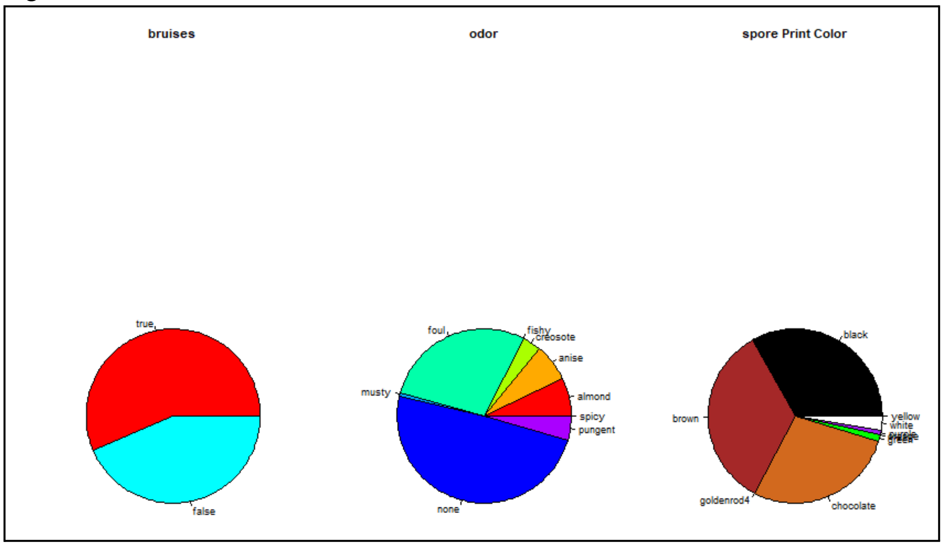
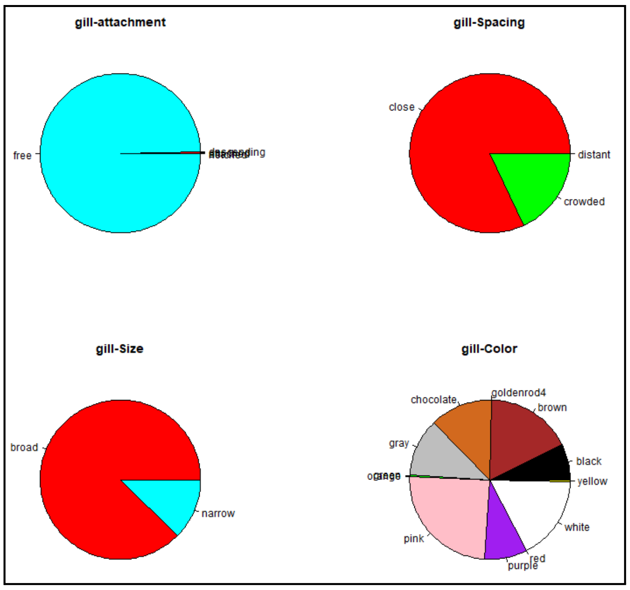
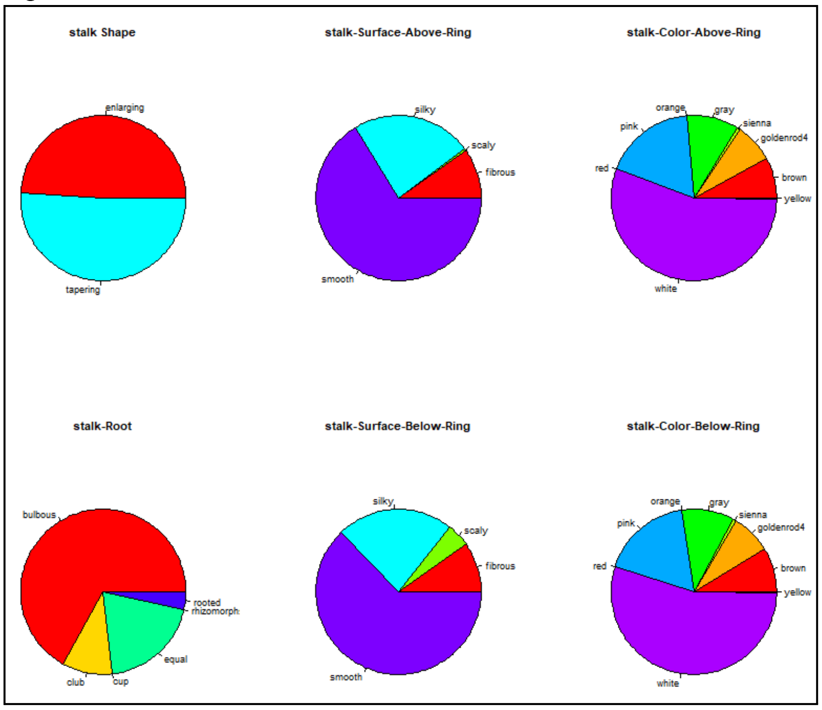


Fig 1.2


gill-attachment
free
descending
attached
gill-Spacing
close
distant
crowded
gill-Size
broad
narrow
gill-Color
chocolate
goldenrod4
brown
gray
black
yellow
orange
green
white
pink
purple
red
Fig 1.3


stalk Shape
enlarging
tapering
stalk-Surface-Above-Ring
silky
scaly
fibrous
smooth
stalk-Color-Above-Ring
orange
gray
pink
sienna
goldenrod4
red
brown
yellow
white
stalk-Root
bulbous
rooted
rhizomorph
equal
club
cup
stalk-Surface-Below-Ring
silky
scaly
fibrous
smooth
stalk-Color-Below-Ring
orange
gray
pink
sienna
goldenrod4
red
brown
yellow
white
Fig 1.4

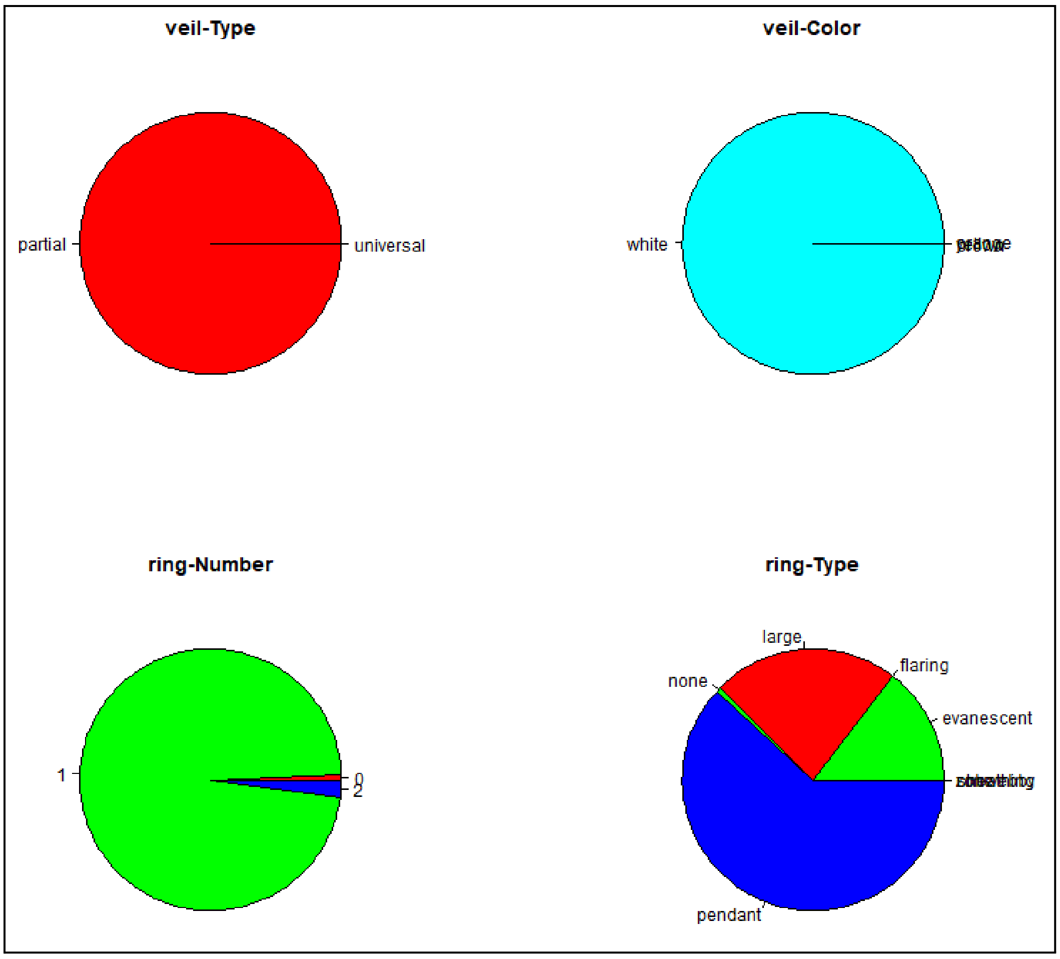

veil-Type
partial
universal
veil-Color
white
brown
orange
yellow
ring-Number
1
0
2
ring-Type
large
flaring
none
evanescent
pendant

Fig 1.5

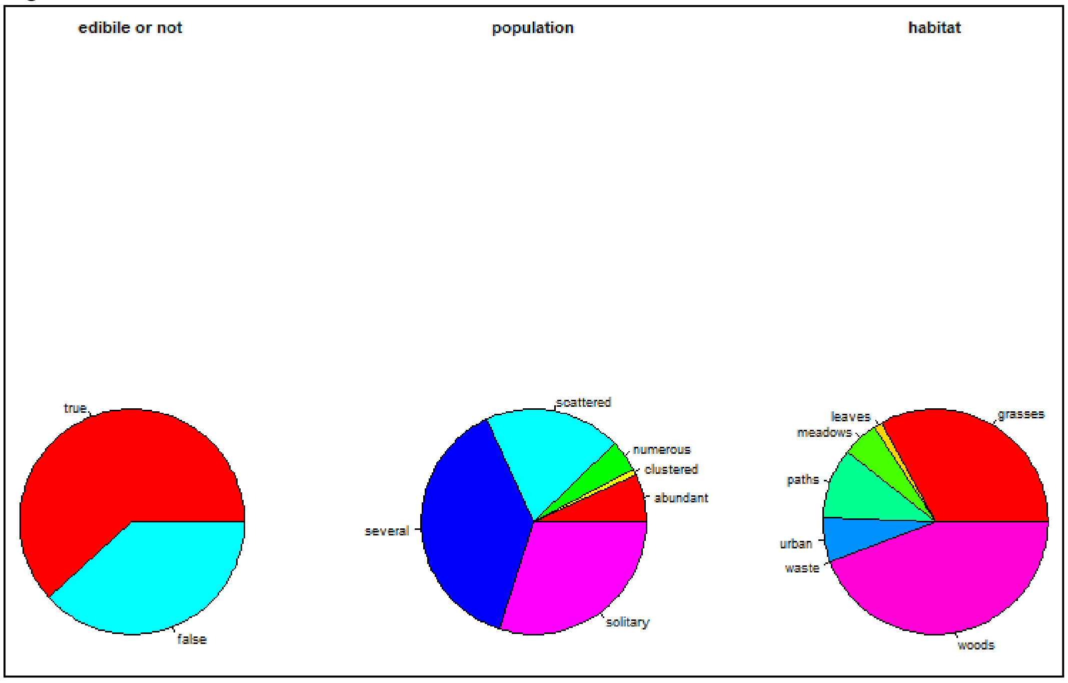
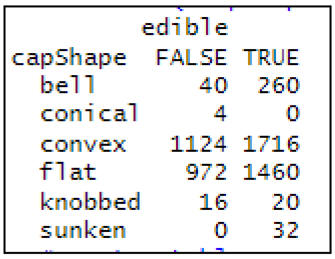

edibile or not
true
false
population
scattered
numerous
clustered
abundant
several
solitary
habitat
leaves
grasses
meadows
paths
urban
waste
woods

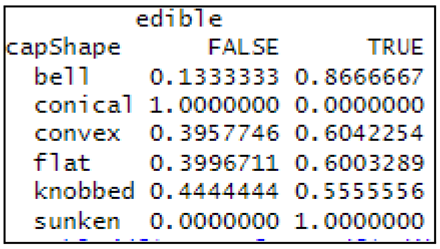
Fig 1.6

2. Probability Tables



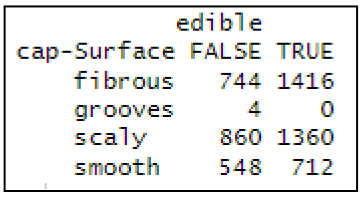
```
           edible
capShape   FALSE TRUE
  bell        40  260
  conical      4    0
  convex    1124 1716
  flat       972 1460
  knobbed     16   20
  sunken       0   32
```

Fig 2.1



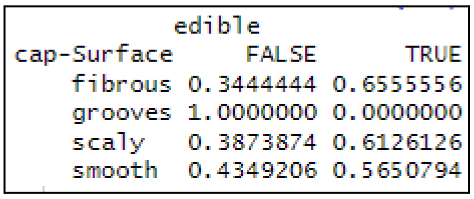
```
           edible
capShape        FALSE      TRUE
  bell      0.1333333 0.8666667
  conical   1.0000000 0.0000000
  convex    0.3957746 0.6042254
  flat      0.3996711 0.6003289
  knobbed   0.4444444 0.5555556
  sunken    0.0000000 1.0000000
```

Fig 2.2



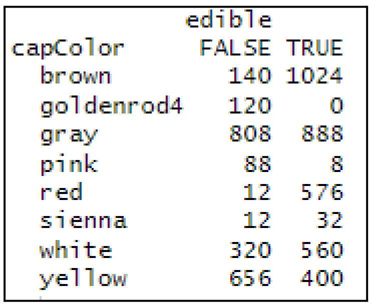
```
            edible
cap-Surface FALSE TRUE
    fibrous   744 1416
    grooves     4    0
    scaly     860 1360
    smooth    548  712
```

Fig 2.3



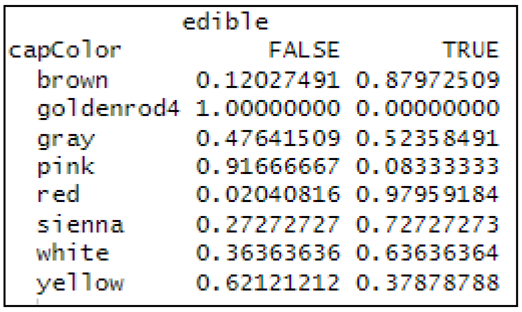
```
            edible
cap-Surface      FALSE      TRUE
    fibrous  0.3444444 0.6555556
    grooves  1.0000000 0.0000000
    scaly    0.3873874 0.6126126
    smooth   0.4349206 0.5650794
```

Fig 2.4



```
             edible
capColor     FALSE TRUE
  brown        140 1024
  goldenrod4   120    0
  gray         808  888
  pink          88    8
  red           12  576
  sienna        12   32
  white        320  560
  yellow       656  400
```

Fig 2.5



```
             edible
capColor          FALSE       TRUE
  brown      0.12027491 0.87972509
  goldenrod4 1.00000000 0.00000000
  gray       0.47641509 0.52358491
  pink       0.91666667 0.08333333
  red        0.02040816 0.97959184
  sienna     0.27272727 0.72727273
  white      0.36363636 0.63636364
  yellow     0.62121212 0.37878788
```

Fig 2.6

3. GLM Functions

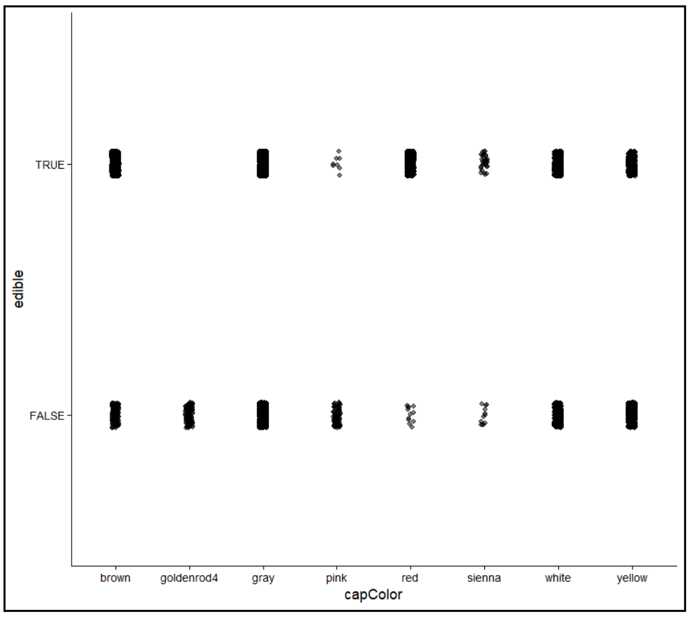
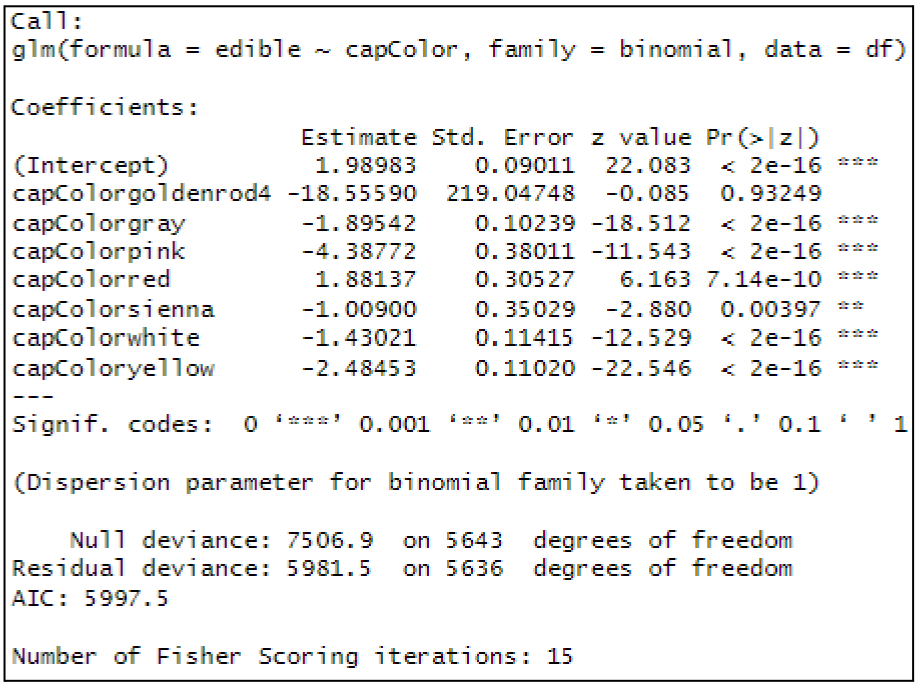


Fig 3.1



```
Call:
glm(formula = edible ~ capColor, family = binomial, data = df)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)    
(Intercept)           1.98983    0.09011  22.083  < 2e-16 ***
capColorgoldenrod4 -18.55590  219.04748  -0.085  0.93249    
capColorgray         -1.89542    0.10239 -18.512  < 2e-16 ***
capColorpink         -4.38772    0.38011 -11.543  < 2e-16 ***
capColorred           1.88137    0.30527   6.163 7.14e-10 ***
capColorsienna       -1.00900    0.35029  -2.880  0.00397 ** 
capColorwhite        -1.43021    0.11415 -12.529  < 2e-16 ***
capColoryellow       -2.48453    0.11020 -22.546  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7506.9  on 5643  degrees of freedom
Residual deviance: 5981.5  on 5636  degrees of freedom
AIC: 5997.5

Number of Fisher Scoring iterations: 15
```

Fig 3.2

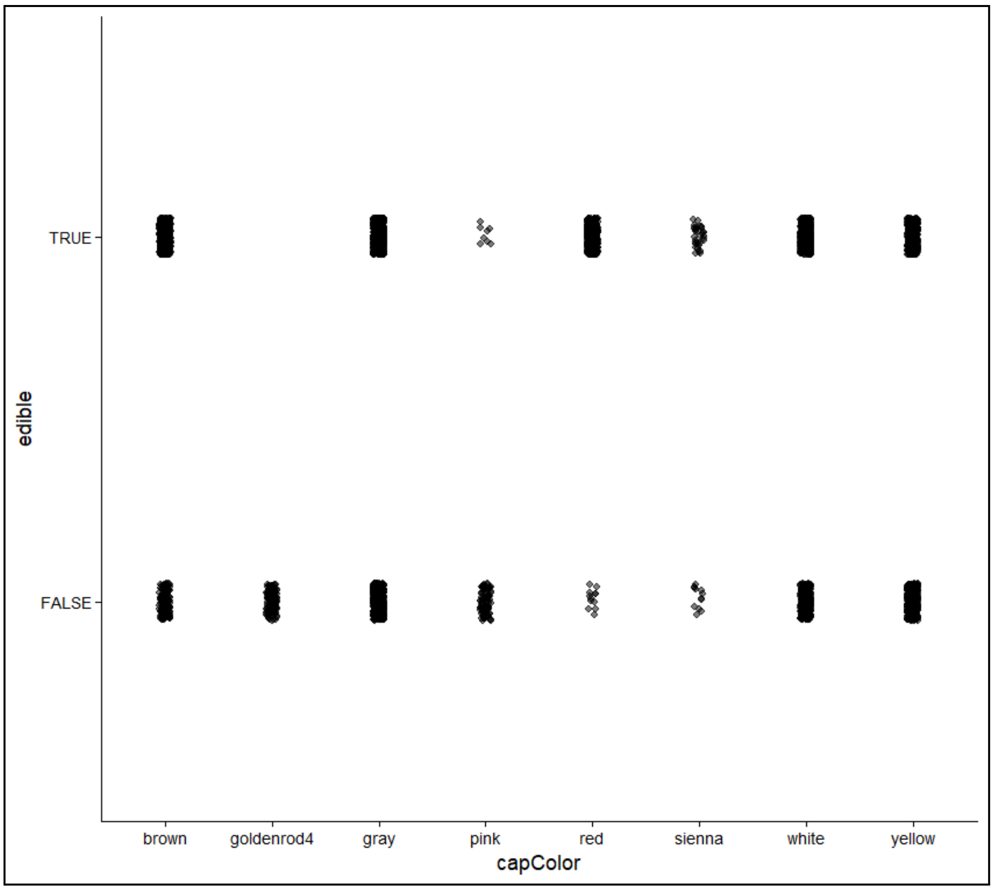
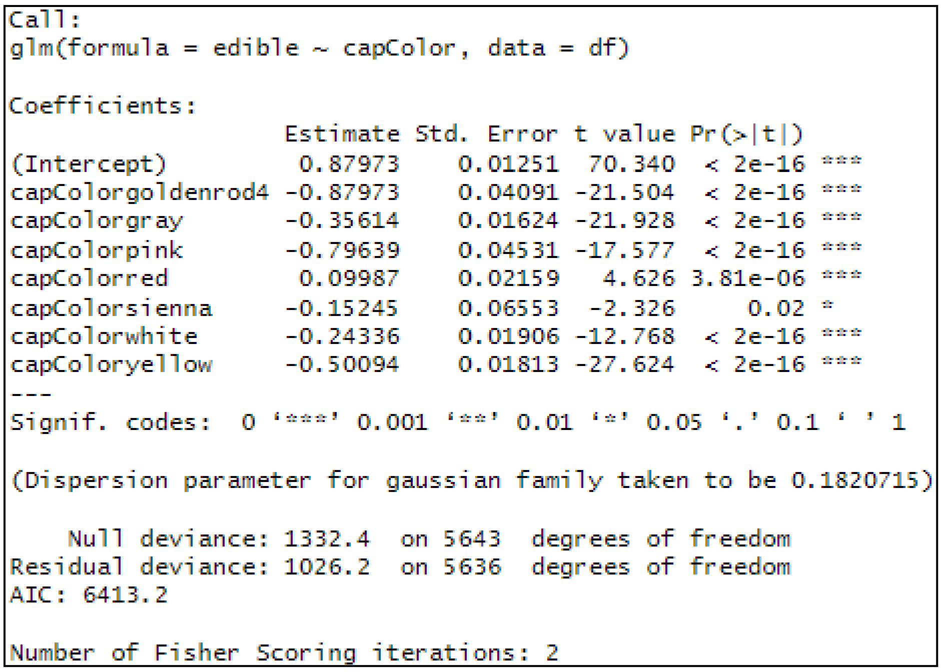
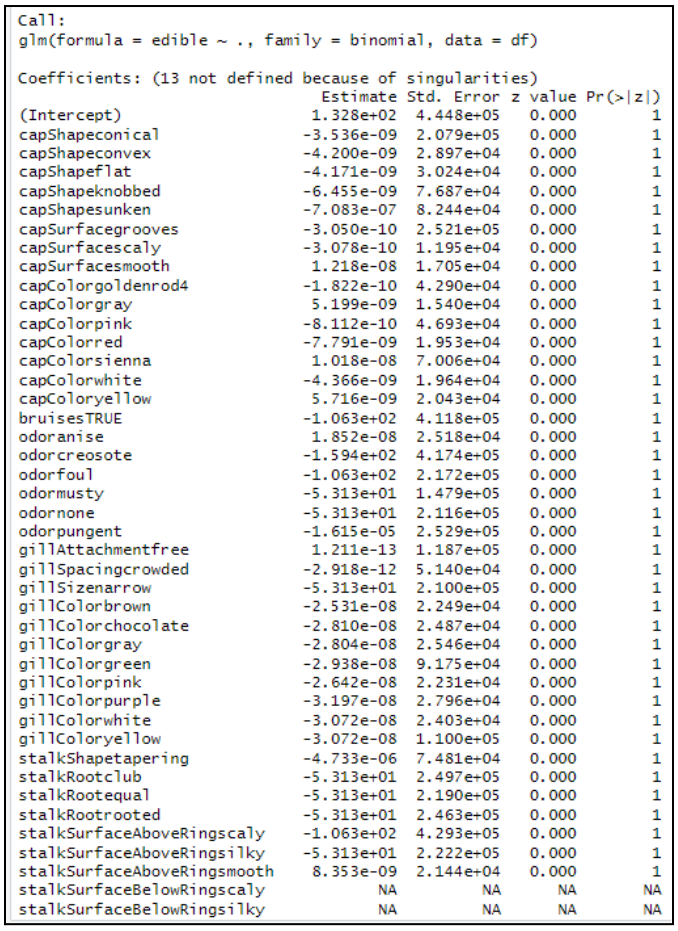


Fig 3.3



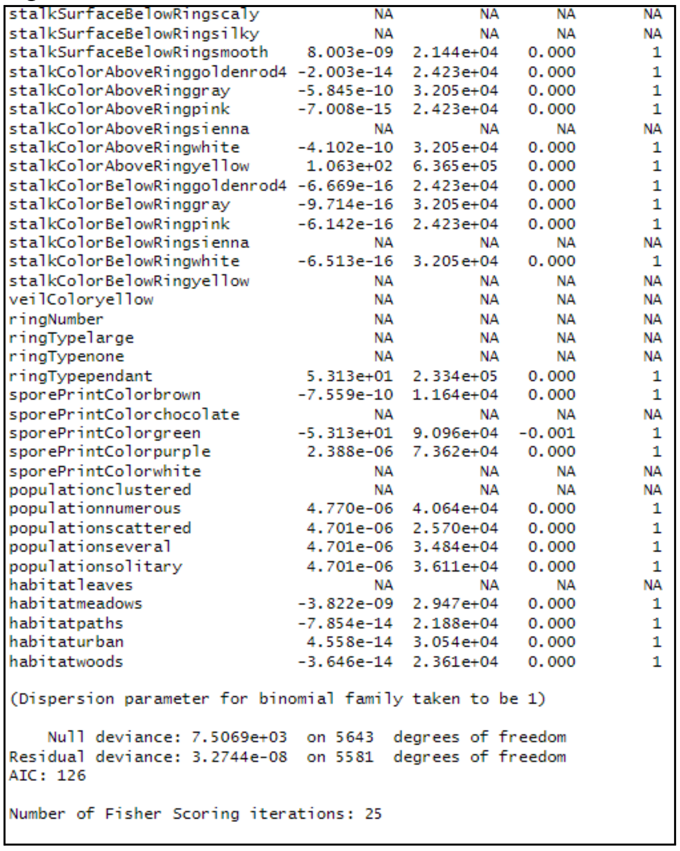
```
Call:
glm(formula = edible ~ capColor, data = df)

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          0.87973    0.01251  70.340  < 2e-16 ***
capColorgoldenrod4 -0.87973    0.04091 -21.504  < 2e-16 ***
capColorgray        -0.35614    0.01624 -21.928  < 2e-16 ***
capColorpink        -0.79639    0.04531 -17.577  < 2e-16 ***
capColorred          0.09987    0.02159   4.626 3.81e-06 ***
capColorsienna      -0.15245    0.06553  -2.326     0.02 *  
capColorwhite       -0.24336    0.01906 -12.768  < 2e-16 ***
capColoryellow      -0.50094    0.01813 -27.624  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1820715)

    Null deviance: 1332.4  on 5643  degrees of freedom
Residual deviance: 1026.2  on 5636  degrees of freedom
AIC: 6413.2

Number of Fisher Scoring iterations: 2
```

Fig 3.4



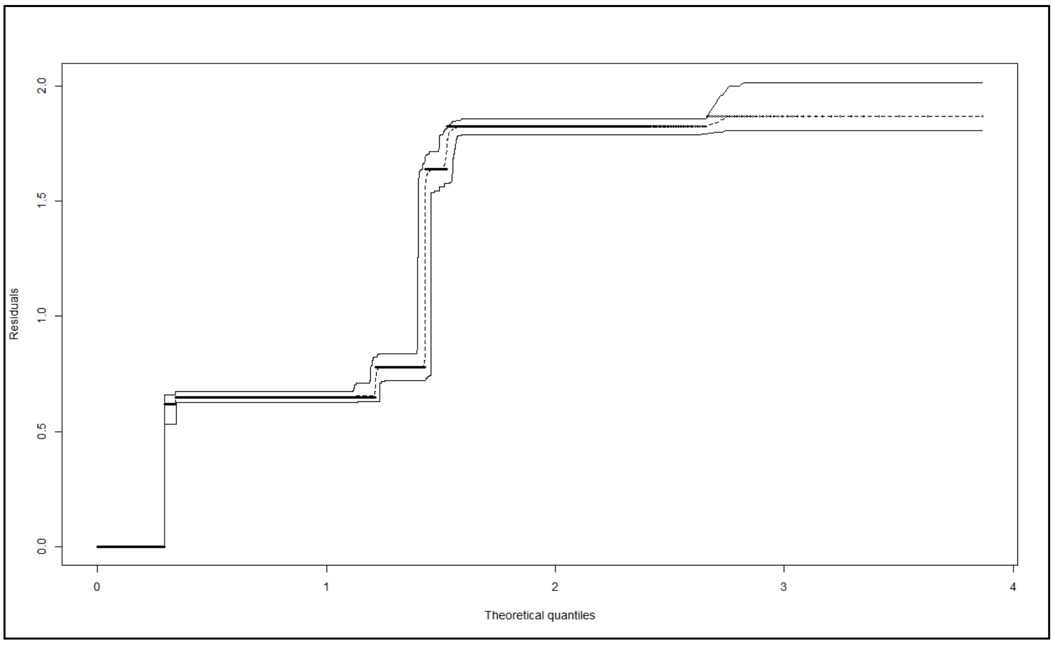
```
Call:
glm(formula = edible ~ ., family = binomial, data = df)

Coefficients: (13 not defined because of singularities)
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                     1.328e+02  4.448e+05   0.000        1
capShapeconical                -3.536e-09  2.079e+05   0.000        1
capShapeconvex                 -4.200e-09  2.897e+04   0.000        1
capShapeflat                   -4.171e-09  3.024e+04   0.000        1
capShapeknobbed                -6.455e-09  7.687e+04   0.000        1
capShapesunken                 -7.083e-07  8.244e+04   0.000        1
capSurfacegrooves              -3.050e-10  2.521e+05   0.000        1
capSurfacescaly                -3.078e-10  1.195e+04   0.000        1
capSurfacesmooth                1.218e-08  1.705e+04   0.000        1
capColorgoldenrod4             -1.822e-10  4.290e+04   0.000        1
capColorgray                    5.199e-09  1.540e+04   0.000        1
capColorpink                   -8.112e-10  4.693e+04   0.000        1
capColorred                    -7.791e-09  1.953e+04   0.000        1
capColorsienna                  1.018e-08  7.006e+04   0.000        1
capColorwhite                  -4.366e-09  1.964e+04   0.000        1
capColoryellow                  5.716e-09  2.043e+04   0.000        1
bruisesTRUE                    -1.063e+02  4.118e+05   0.000        1
odoranise                       1.852e-08  2.518e+04   0.000        1
odorcreosote                   -1.594e+02  4.174e+05   0.000        1
odorfoul                       -1.063e+02  2.172e+05   0.000        1
odormusty                      -5.313e+01  1.479e+05   0.000        1
odornone                       -5.313e+01  2.116e+05   0.000        1
odorpungent                    -1.615e-05  2.529e+05   0.000        1
gillAttachmentfree              1.211e-13  1.187e+05   0.000        1
gillSpacingcrowded             -2.918e-12  5.140e+04   0.000        1
gillSizenarrow                 -5.313e+01  2.100e+05   0.000        1
gillColorbrown                 -2.531e-08  2.249e+04   0.000        1
gillColorchocolate             -2.810e-08  2.487e+04   0.000        1
gillColorgray                  -2.804e-08  2.546e+04   0.000        1
gillColorgreen                 -2.938e-08  9.175e+04   0.000        1
gillColorpink                  -2.642e-08  2.231e+04   0.000        1
gillColorpurple                -3.197e-08  2.796e+04   0.000        1
gillColorwhite                 -3.072e-08  2.403e+04   0.000        1
gillColoryellow                -3.072e-08  1.100e+05   0.000        1
stalkShapetapering             -4.733e-06  7.481e+04   0.000        1
stalkRootclub                  -5.313e+01  2.497e+05   0.000        1
stalkRootequal                 -5.313e+01  2.190e+05   0.000        1
stalkRootrooted                -5.313e+01  2.463e+05   0.000        1
stalkSurfaceAboveRingscaly     -1.063e+02  4.293e+05   0.000        1
stalkSurfaceAboveRingsilky     -5.313e+01  2.222e+05   0.000        1
stalkSurfaceAboveRingsmooth     8.353e-09  2.144e+04   0.000        1
stalkSurfaceBelowRingscaly             NA         NA      NA       NA
stalkSurfaceBelowRingsilky             NA         NA      NA       NA
```

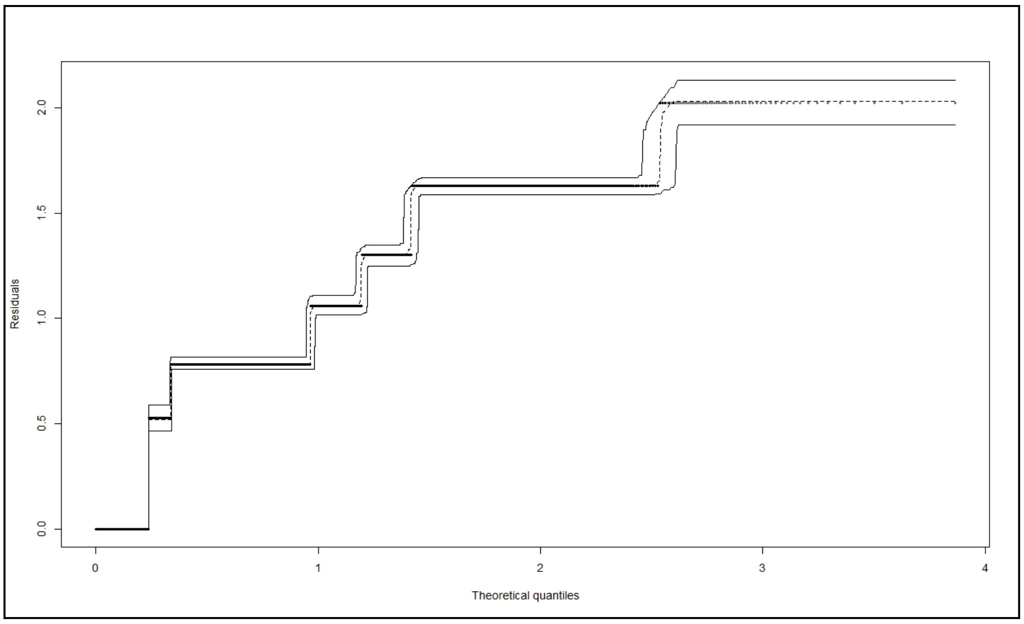
Fig 3.5.1



```
stalkSurfaceBelowRingscaly             NA         NA      NA       NA
stalkSurfaceBelowRingsilky             NA         NA      NA       NA
stalkSurfaceBelowRingsmooth     8.003e-09  2.144e+04   0.000        1
stalkColorAboveRinggoldenrod4 -2.003e-14  2.423e+04   0.000        1
stalkColorAboveRinggray        -5.845e-10  3.205e+04   0.000        1
stalkColorAboveRingpink        -7.008e-15  2.423e+04   0.000        1
stalkColorAboveRingsienna              NA         NA      NA       NA
stalkColorAboveRingwhite       -4.102e-10  3.205e+04   0.000        1
stalkColorAboveRingyellow       1.063e+02  6.365e+05   0.000        1
stalkColorBelowRinggoldenrod4 -6.669e-16  2.423e+04   0.000        1
stalkColorBelowRinggray        -9.714e-16  3.205e+04   0.000        1
stalkColorBelowRingpink        -6.142e-16  2.423e+04   0.000        1
stalkColorBelowRingsienna              NA         NA      NA       NA
stalkColorBelowRingwhite       -6.513e-16  3.205e+04   0.000        1
stalkColorBelowRingyellow              NA         NA      NA       NA
veilColoryellow                        NA         NA      NA       NA
ringNumber                             NA         NA      NA       NA
ringTypelarge                          NA         NA      NA       NA
ringTypenone                           NA         NA      NA       NA
ringTypependant                 5.313e+01  2.334e+05   0.000        1
sporePrintColorbrown           -7.559e-10  1.164e+04   0.000        1
sporePrintColorchocolate               NA         NA      NA       NA
sporePrintColorgreen           -5.313e+01  9.096e+04  -0.001        1
sporePrintColorpurple           2.388e-06  7.362e+04   0.000        1
sporePrintColorwhite                   NA         NA      NA       NA
populationclustered                    NA         NA      NA       NA
populationnumerous              4.770e-06  4.064e+04   0.000        1
populationscattered             4.701e-06  2.570e+04   0.000        1
populationseveral               4.701e-06  3.484e+04   0.000        1
populationsolitary              4.701e-06  3.611e+04   0.000        1
habitatleaves                          NA         NA      NA       NA
habitatmeadows                 -3.822e-09  2.947e+04   0.000        1
habitatpaths                   -7.854e-14  2.188e+04   0.000        1
habitaturban                    4.558e-14  3.054e+04   0.000        1
habitatwoods                   -3.646e-14  2.361e+04   0.000        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7.5069e+03  on 5643  degrees of freedom
Residual deviance: 3.2744e-08  on 5581  degrees of freedom
AIC: 126

Number of Fisher Scoring iterations: 25
```

Fig 3.5.2


Residuals
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
Theoretical quantiles
Fig 3.6


Residuals
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
Theoretical quantiles
Fig 3.7

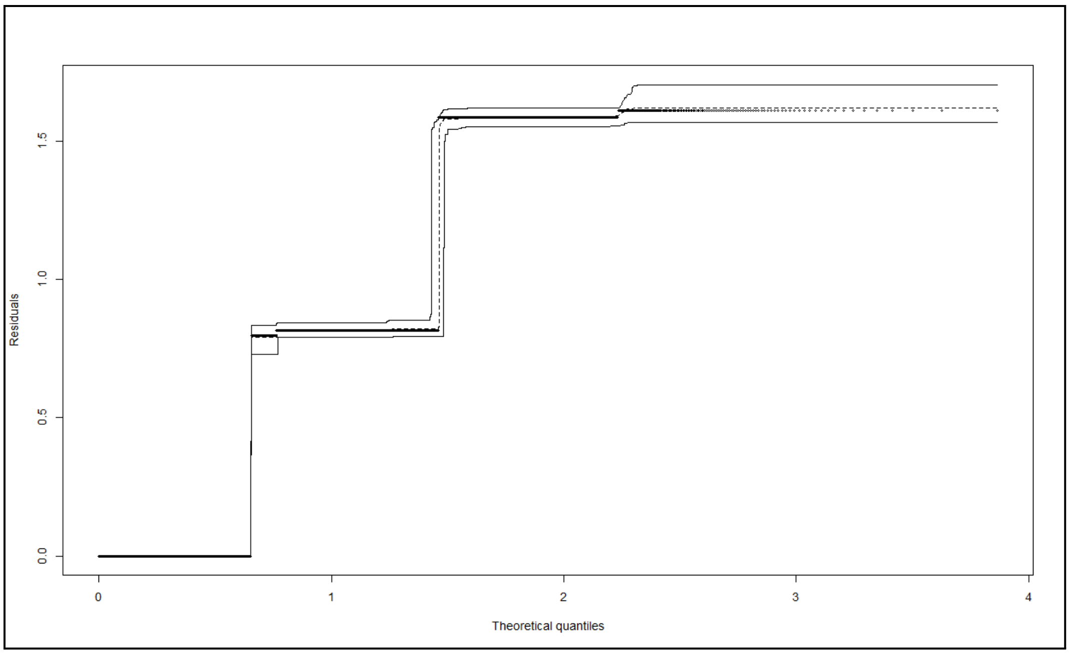
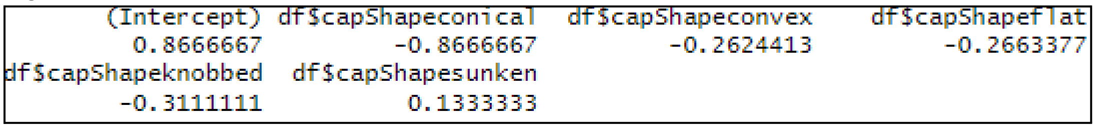


Fig 3.8

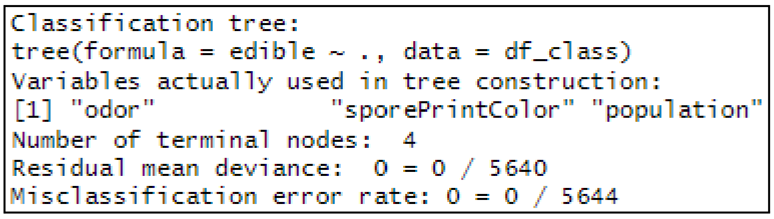
4. Leave-one-out Cross Validation

Fig 4.1



```
       (Intercept) df$capShapeconical  df$capShapeconvex    df$capShapeflat
         0.8666667         -0.8666667         -0.2624413         -0.2663377
df$capShapeknobbed  df$capShapesunken
        -0.3111111          0.1333333
```

5. Fitting Trees



```
Classification tree:
tree(formula = edible ~ ., data = df_class)
Variables actually used in tree construction:
[1] "odor"          "sporePrintColor" "population"
Number of terminal nodes:  4
Residual mean deviance:  0 = 0 / 5640
Misclassification error rate: 0 = 0 / 5644
```

Fig 5.1

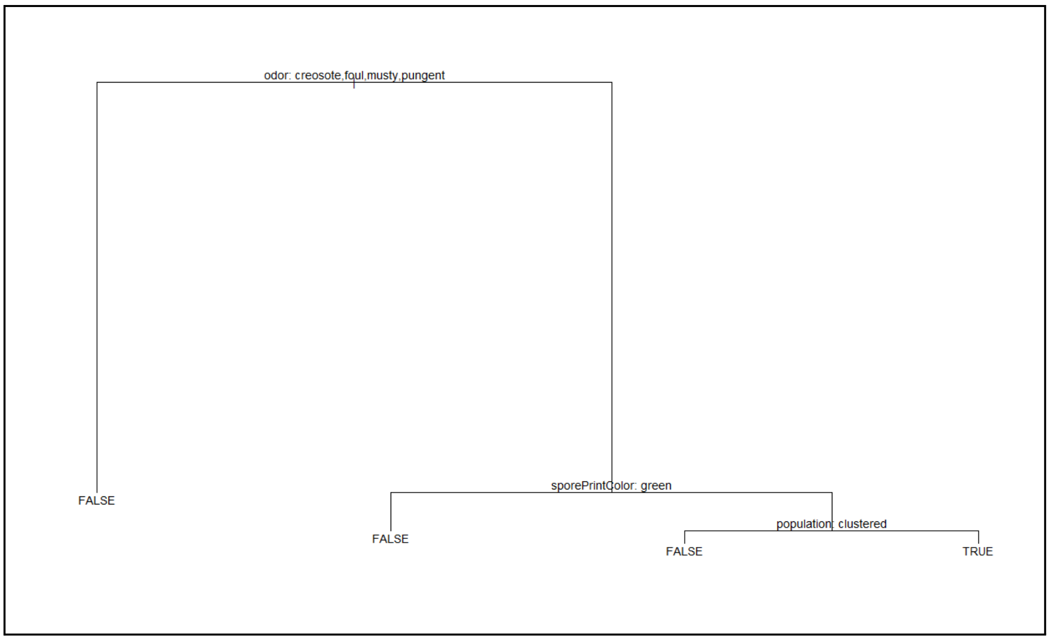
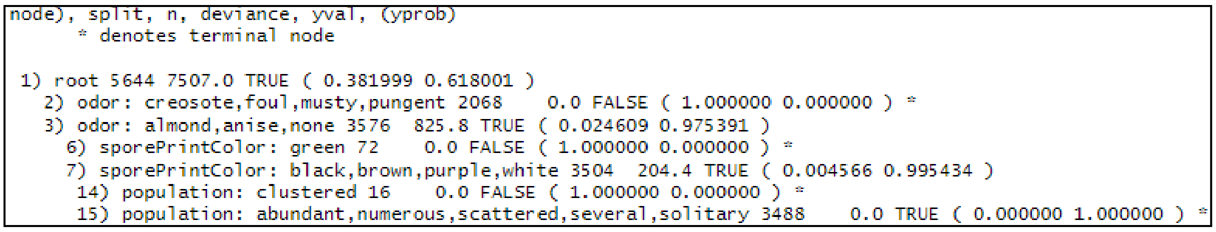
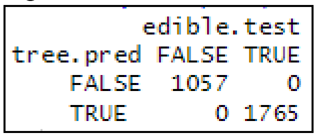


Fig 5.2



```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 5644 7507.0 TRUE ( 0.381999 0.618001 )
   2) odor: creosote,foul,musty,pungent 2068    0.0 FALSE ( 1.000000 0.000000 ) *
   3) odor: almond,anise,none 3576  825.8 TRUE ( 0.024609 0.975391 )
     6) sporePrintColor: green 72    0.0 FALSE ( 1.000000 0.000000 ) *
     7) sporePrintColor: black,brown,purple,white 3504  204.4 TRUE ( 0.004566 0.995434 )
      14) population: clustered 16    0.0 FALSE ( 1.000000 0.000000 ) *
      15) population: abundant,numerous,scattered,several,solitary 3488    0.0 TRUE ( 0.000000 1.000000 ) *
```

Fig 5.3



```
         edible.test
tree.pred FALSE TRUE
    FALSE  1057    0
    TRUE      0 1765
```

Fig 5.4

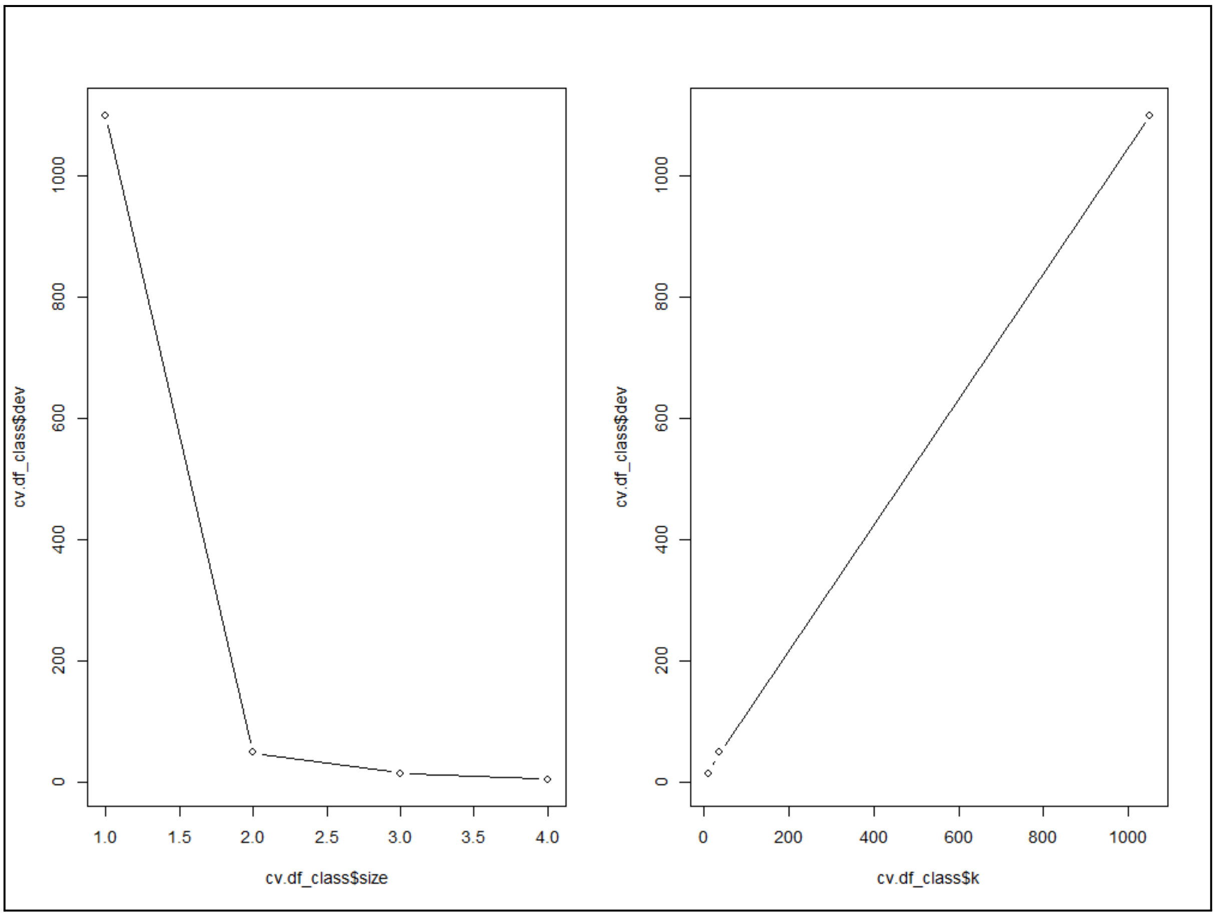

cv.df_class$dev
cv.df_class$size
cv.df_class$dev
cv.df_class$k
0
200
400
600
800
1000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
200
400
600
800
1000

Fig 5.5

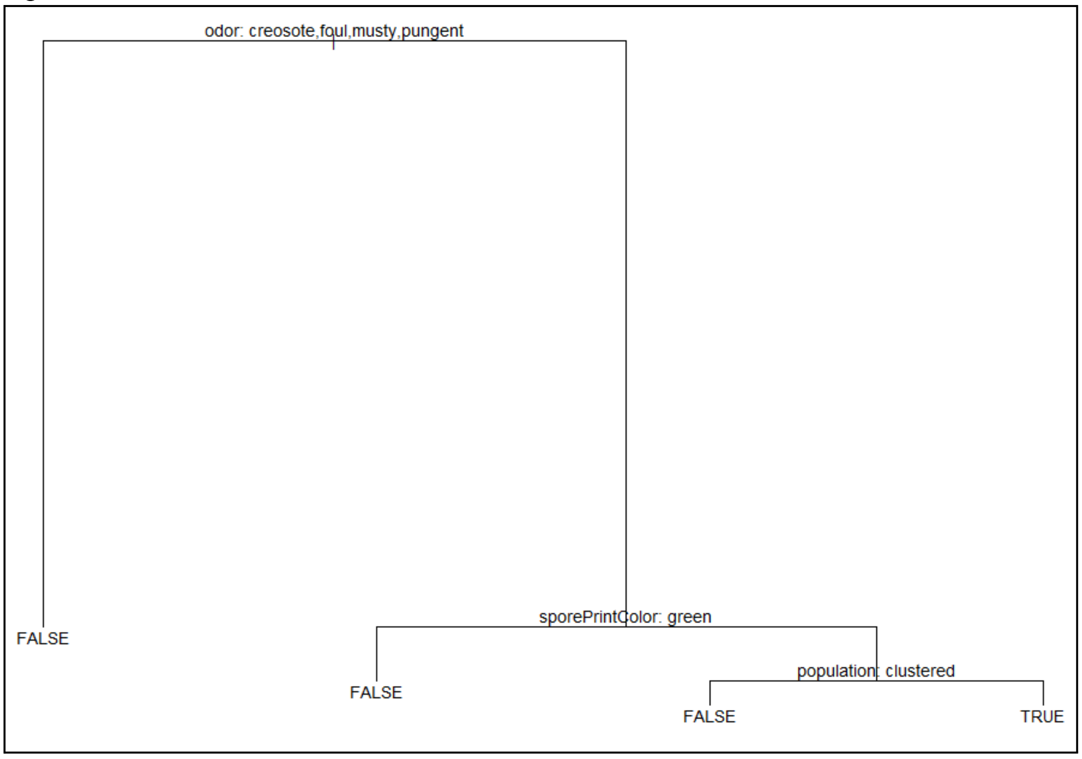
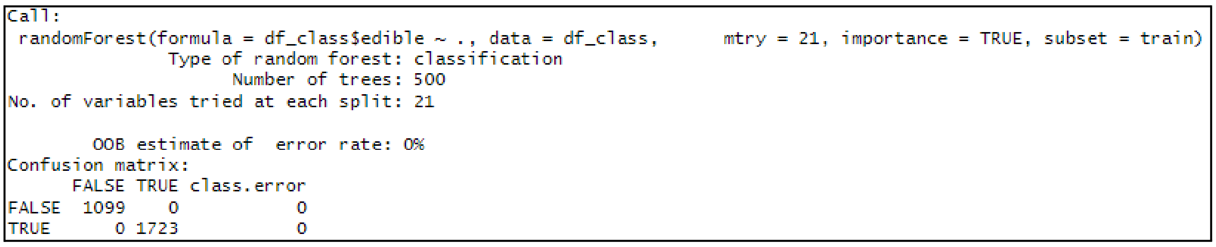

odor: creosote,foul,musty,pungent
FALSE
sporePrintColor: green
FALSE
population: clustered
FALSE
TRUE

Fig 5.6

6. Random Forest



```
Call:
 randomForest(formula = df_class$edible ~ ., data = df_class,      mtry = 21, importance = TRUE, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 21

        OOB estimate of  error rate: 0%
Confusion matrix:
      FALSE TRUE class.error
FALSE  1099    0           0
TRUE      0 1723           0
```

6.1

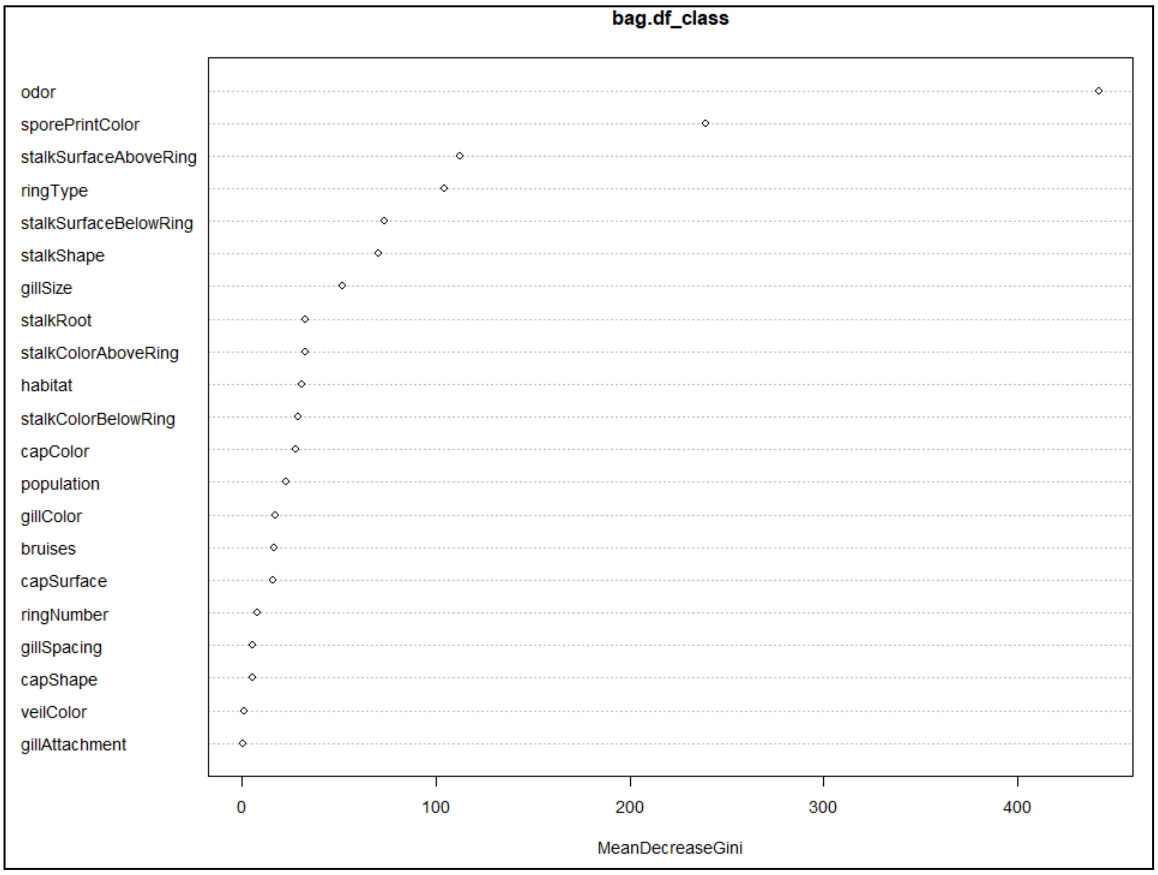
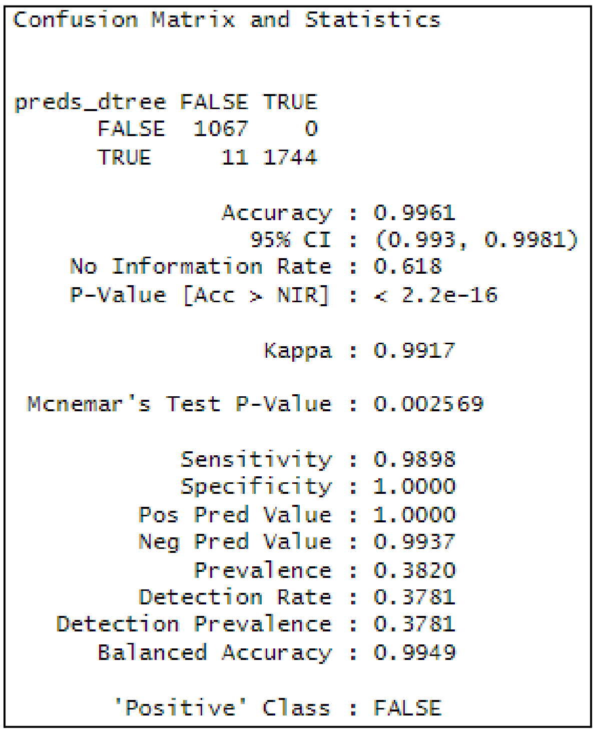


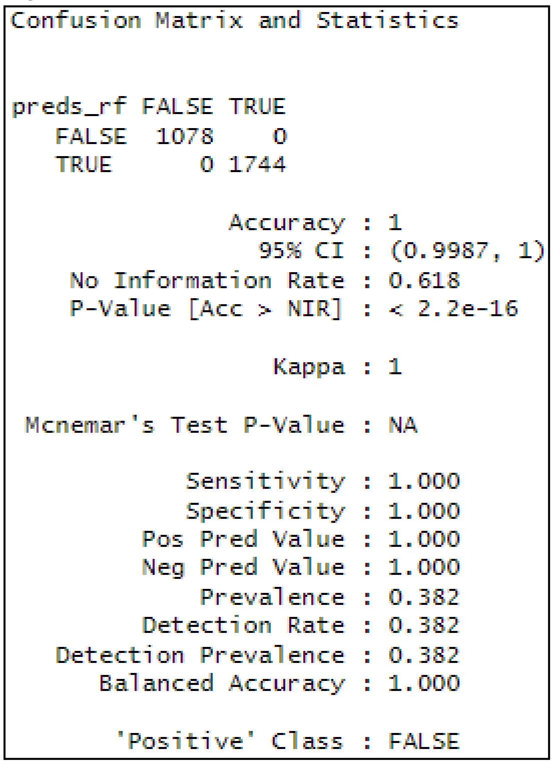
Fig 6.2

7. Classification Forest



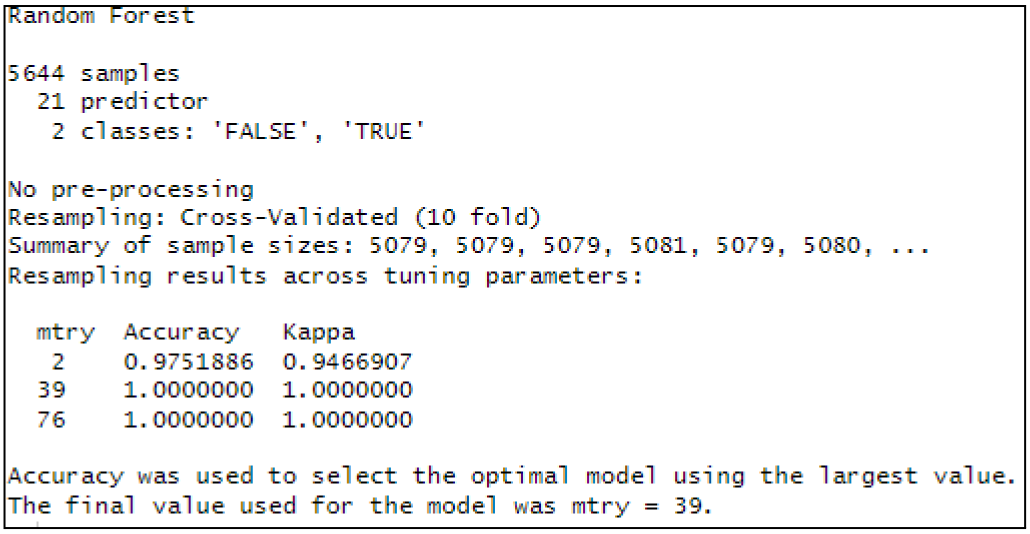
```
Confusion Matrix and Statistics


preds_dtree FALSE TRUE
      FALSE  1067    0
      TRUE     11 1744

               Accuracy : 0.9961
                 95% CI : (0.993, 0.9981)
    No Information Rate : 0.618
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9917

 Mcnemar's Test P-Value : 0.002569

            Sensitivity : 0.9898
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9937
             Prevalence : 0.3820
         Detection Rate : 0.3781
   Detection Prevalence : 0.3781
      Balanced Accuracy : 0.9949

       'Positive' Class : FALSE
```

Fig 7.1



```
Confusion Matrix and Statistics


preds_rf FALSE TRUE
   FALSE  1078    0
   TRUE      0 1744

               Accuracy : 1
                 95% CI : (0.9987, 1)
    No Information Rate : 0.618
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.382
         Detection Rate : 0.382
   Detection Prevalence : 0.382
      Balanced Accuracy : 1.000

       'Positive' Class : FALSE
```

Fig 7.2



```
Random Forest

5644 samples
  21 predictor
   2 classes: 'FALSE', 'TRUE'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 5079, 5079, 5079, 5081, 5079, 5080, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9751886  0.9466907
  39    1.0000000  1.0000000
  76    1.0000000  1.0000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 39.
```

Fig 7.3

Code

```r
########introduction##########################################################
#this will only use a maximium of 80 charcters across for ease of use
#we are told by the data set that there are some missing values from stalk-root
#I have decided to remove these as there are still several thousand lines of
#data remaining.#
#the colours sometimes refer to cinnamon and buff which R does not recognize as
#colours, I have decided to instead refer to these as sienna and goldenrod4
#instead as these are recognized by R as colours and will make whatever algrithm
#I must apply to them easier to use.
######library########
install.packages("tidyverse")
library(tidyverse)
install.packages("hnp")
library(hnp)
install.packages("ISLR2")
library(ISLR2)
install.packages("boot")
library(boot)
install.packages("tree")
library(tree)
install.packages("randomForest")
library(randomForest)
install.packages("caret")
library(caret)
install.packages("dplyr")
library(dplyr)
install.packages("caTools")
library(caTools)
install.packages("rpart")
library(rpart)
colNames <- c("edible", "capShape", "capSurface", "capColor", "bruises", "odor",
              "gillAttachment", "gillSpacing", "gillSize", "gillColor",
              "stalkShape", "stalkRoot", "stalkSurfaceAboveRing",
              "stalkSurfaceBelowRing", "stalkColorAboveRing",
              "stalkColorBelowRing", "veilType", "veilColor", "ringNumber",
              "ringType", "sporePrintColor", "population", "habitat")
                 #creates a string of column titles
capShape_variables <- c("bell", "conical", "convex", "flat",
                        "knobbed", "sunken")
capSurface_variables <- c("fibrous", "grooves", "scaly", "smooth")
capColor_variables <- c("brown", "goldenrod4", "sienna", "gray", "green",
                        "pink", "purple", "red", "white", "yellow")
odor_variables <- c("almond", "anise", "creosote", "fishy", "foul", "musty",
                    "none", "pungent", "spicy")
gillAttachment_variables <- c("attached", "descending", "free", "notched")
gillSpacing_variables <- c("close", "crowded", "distant")
gillSize_variables <- c("broad", "narrow")
gillColor_variables <- c("black", "brown", "goldenrod4", "chocolate",
                         "gray","green", "orange", "pink", "purple","red",
                         "white" ,"yellow")
stalkShape_variables <- c("enlarging", "tapering")
stalkRoot_variables <- c("bulbous", "club", "cup", "equal", "rhizomorphs",
                         "rooted", NA)
stalkSurfaceAboveRing_variables <- c("fibrous", "scaly", "silky", "smooth")
stalkSurfaceBelowRing_variables <- c("fibrous", "scaly", "silky", "smooth")
stalkColorAboveRing_variables <- c("brown", "goldenrod4", "sienna", "gray",
                                   "orange", "pink", "red", "white", "yellow")
stalkColorBelowRing_variables <- c("brown", "goldenrod4", "sienna", "gray",
                                   "orange", "pink", "red", "white", "yellow")
veilType_variables <- c("partial", "universal")
veilColor_variables <- c("brown", "orange","white", "yellow")
ringNumber_variables <- c(0,1,2)
ringType_variables <- c("cobwebby", "evanescent", "flaring", "large", "none",
                        "pendant", "sheathing", "zone")
sporePrintColor_variables <- c("black", "brown", "goldenrod4", "chocolate",
                               "green", "orange", "purple", "white", "yellow")
population_variables <- c("abundant", "clustered", "numerous", "scattered",
                          "several", "solitary")
habitat_variables <- c("grasses", "leaves", "meadows", "paths", "urban",
                       "waste", "woods")

NumberOf <- function(data, check){
  n = 0
  for (i in data){
    if (i == check){
      n = n+1
    }
  }
  return(n)
}

#####data cleaning#######

#loading the data into R using a working directory
df_original <- read.csv("agaricus-lepiotaData.data", header = FALSE)
#the title of the columns as well as the contents are not very useful,
#the agaricus-lepiota.names file (among other things) tells use
#what each of the letters means

df_dirty<- df_original#creates a new data set so the
                    #original will remain untouched
colnames(df_dirty) <- colNames#makes the column
                           #titles what is in the array above

summary(df_dirty) #gives us a summary of the data for each column
#all are characters checking the agaricus-lepiota.names file again most of these
#are correct however there are a couple the could be turned into logical or
#numerical variables and for the ones that are not changing the letter what it
#is actually describing will massivly help readibility

#we are told that the data set has some missing values in the StalkRoot column,
#for now they will be converted to NA's to highlight they are missing
#the following checks that all instance of the columns contains what is expected
#and changes it to a better data type or variable
  #edible
  sum(!df_dirty$edible %in% c("e", "p"))
  df_dirty$edible <- df_dirty$edible == "e"

  #capShape
  sum(!df_dirty$capShape %in% c("b", "c", "x", "f", "k", "s"))
  df_dirty$capShape <- capShape_variables[match(df_dirty$capShape,
                                       c("b", "c", "x","f", "k", "s"))]
  #capSurface
  sum(!df_dirty$capSurface %in% c("f", "g", "y", "s"))
  df_dirty$capSurface <- capSurface_variables[match(df_dirty$capSurface,
                                          c("f", "g","y", "s"))]
  #capColor
  sum(!df_dirty$capColor %in%
       c("n", "b", "c", "g", "r", "p", "u", "e", "w", "y"))
  df_dirty$capColor <- capColor_variables[match(df_dirty$capColor,
                         c("n", "b", "c", "g","r", "p", "u", "e","w", "y"))]
  #bruises
  sum(!df_dirty$bruises %in% c("t", "f"))
  df_dirty$bruises <- df_dirty$bruises == "t"

  #odor
  sum(!df_dirty$odor %in% c("a", "l", "c", "y", "f", "m", "n", "p", "s"))
  df_dirty$odor <- odor_variables[match(df_dirty$odor,
                           c("a", "l", "c", "y", "f", "m", "n","p", "s"))]
  #gillAttachment
  sum(!df_dirty$gillAttachment %in% c("a", "d", "f", "n"))
  df_dirty$gillAttachment <-
    gillAttachment_variables[match(df_dirty$gillAttachment,
                                             c("a","d", "f", "n"))]
  #gillSpacing
  sum(!df_dirty$gillSpacing %in% c("c", "w", "d"))
  df_dirty$gillSpacing <- gillSpacing_variables[match(df_dirty$gillSpacing,
                                       c("c", "w", "d"))]
  #gillSize
  sum(!df_dirty$gillSize %in% c("b", "n"))
  df_dirty$gillSize <- gillSize_variables[match(df_dirty$gillSize,
                                       c("b", "n"))]
  #gillColor
  sum(!df_dirty$gillColor %in%
       c("k", "n", "b", "h", "g", "r", "o", "p", "u", "e", "w", "y"))
  df_dirty$gillColor <- gillColor_variables[match(df_dirty$gillColor,
                                     c("k", "n", "b", "h", "g", "r",
                                      "o", "p", "u", "e", "w", "y"))]
  #stalkShape
  sum(!df_dirty$stalkShape %in% c("e", "t"))
  df_dirty$stalkShape <- stalkShape_variables[match(df_dirty$stalkShape,
                                          c("e", "t"))]
  #stalkRoot
  sum(!df_dirty$stalkRoot %in% c("b", "c", "u", "e", "z", "r", "?"))
  df_dirty$stalkRoot <- stalkRoot_variables[match(df_dirty$stalkRoot,
                                   c("b", "c", "u", "e", "z", "r", "?"))]
  #stalkSurfaceAboveRing
  sum(!df_dirty$stalkSurfaceAboveRing %in% c("f", "y", "k", "s"))
  df_dirty$stalkSurfaceAboveRing <-
    stalkSurfaceAboveRing_variables[match(df_dirty$stalkSurfaceAboveRing,
                                    c("f", "y", "k", "s"))]
  #stalkSurfaceBelowRing
  sum(!df_dirty$stalkSurfaceBelowRing %in% c("f", "y", "k", "s"))
  df_dirty$stalkSurfaceBelowRing <-
```

```
    stalkSurfaceBelowRing_variables[match(df_dirty$stalkSurfaceBelowRing,
                                          c("f", "y", "k", "s"))]
  #stalkColorAboveRing
  sum(!df_dirty$stalkColorAboveRing %in%
        c("n", "b", "c", "g", "o", "p", "e", "w", "y"))
  df_dirty$stalkColorAboveRing <-
    stalkColorAboveRing_variables[match(df_dirty$stalkColorAboveRing,
                                        c("n", "b", "c", "g", "o",
                                          "p", "e", "w", "y"))]
  #stalkColorBelowRing
  sum(!df_dirty$stalkColorBelowRing %in%
        c("n", "b", "c", "g", "o", "p", "e", "w", "y"))
  df_dirty$stalkColorBelowRing <-
    stalkColorBelowRing_variables[match(df_dirty$stalkColorBelowRing,
                              c("n", "b", "c", "g", "o", "p", "e", "w", "y"))]
  #veilType
  sum(!df_dirty$veilType %in% c("p", "u"))
  df_dirty$veilType <- veilType_variables[match(df_dirty$veilType,
                                               c("p", "u"))]
  #veilColor
  sum(!df_dirty$veilColor %in% c("n", "o", "w", "y"))
  df_dirty$veilColor <- veilColor_variables[match(df_dirty$veilColor,
                                            c("n", "o", "w", "y"))]
  #ringNumber
  sum(!df_dirty$ringNumber %in% c("n", "o", "t"))
  df_dirty$ringNumber <- ringNumber_variables[match(df_dirty$ringNumber,
                                                c("n", "o", "t"))]
  #ringType
  sum(!df_dirty$ringType %in% c("c", "e", "f", "l", "n", "p", "s", "z"))
  df_dirty$ringType <- ringType_variables[match(df_dirty$ringType,
                                    c("c", "e", "f", "l", "n", "p", "s", "z"))]
  #sporePrintColor
  sum(!df_dirty$sporePrintColor %in%
        c("k", "n", "b", "h", "r", "o", "u", "w", "y"))
  df_dirty$sporePrintColor<-
    sporePrintColor_variables[match(df_dirty$sporePrintColor,
                              c("k", "n", "b", "h", "r", "o", "u", "w", "y"))]
  #population
  sum(!df_dirty$population %in% c("a", "c", "n", "s", "v", "y"))
  df_dirty$population <- population_variables[match(df_dirty$population,
                                            c("a", "c", "n", "s", "v", "y"))]
  #habitat
  sum(!df_dirty$habitat %in% c("g", "l", "m", "p", "u","w", "d"))
  df_dirty$habitat <- habitat_variables[match(df_dirty$habitat,
                                        c("g", "l", "m", "p", "u","w", "d"))]

sum(is.na(df_dirty)) # this checks the number of NA's
# there are 2,480
df <- na.omit(df_dirty) #removes NA's
sum(is.na(df)) # this checks the number of NA's again
#there are now none

sum(duplicated(df))
#this checks if there are any duplicated rows in the data set
#there are none


####pie charts#####

# the following creates pie charts showing the proportions of
#the variables for each of the cap related features
par(mfrow = c(1, 3))#allows multiple graphs to be displayed at
                    #once to they can be seen and compared easier
pie( c(NumberOf(df$capShape, "bell"),
       NumberOf(df$capShape, "conical"),
       NumberOf(df$capShape, "convex"),
       NumberOf(df$capShape, "flat"),
       NumberOf(df$capShape, "knobbed"),
       NumberOf(df$capShape, "sunken") ) ,
     capShape_variables, main = "cap-shape",
     col = rainbow(length(capShape_variables)))
pie( c(NumberOf(df$capSurface, "fibrous"),
       NumberOf(df$capSurface, "groves"),
       NumberOf(df$capSurface, "scaly"),
       NumberOf(df$capSurface, "smooth")) ,
     capSurface_variables, main = "cap-surface",
     col = rainbow(length(capSurface_variables)))
pie( c(NumberOf(df$capColor, "brown"),
       NumberOf(df$capColor, "goldenrod4"),
       NumberOf(df$capColor, "sienna"),
       NumberOf(df$capColor, "gray"),
       NumberOf(df$capColor, "green"),
       NumberOf(df$capColor, "pink"),
       NumberOf(df$capColor, "purple"),
       NumberOf(df$capColor, "red"),
       NumberOf(df$capColor, "white"),
```

```r
     NumberOf(df$capColor, "yellow")) ,
   capColor_variables, main = "capColor",col = capColor_variables)

# the following creates pie charts showing the proportions
#of the variables for bruises, odor and spore print color
par(mfrow = c(1, 3))#allows multiple graphs to be displayed at
                    #once to they can be seen and compared easier
pie( c(NumberOf(df$bruises, TRUE),NumberOf(df$bruises, FALSE)) ,
   c("true", "false"), main = "bruises",col = rainbow(2))
pie( c(NumberOf(df$odor, "almond"),
     NumberOf(df$odor, "anise"),
     NumberOf(df$odor, "creosote"),
     NumberOf(df$odor, "fishy"),
     NumberOf(df$odor, "foul"),
     NumberOf(df$odor, "musty"),
     NumberOf(df$odor, "none"),
     NumberOf(df$odor, "pungent"),
     NumberOf(df$odor, "spicy")) ,
   odor_variables, main = "odor",col = rainbow(length(odor_variables)))
pie( c(NumberOf(df$sporePrintColor, "black"),
     NumberOf(df$sporePrintColor, "brown"),
     NumberOf(df$sporePrintColor, "goldenrod4"),
     NumberOf(df$sporePrintColor, "chocolate"),
     NumberOf(df$sporePrintColor, "green"),
     NumberOf(df$sporePrintColor, "orange"),
     NumberOf(df$sporePrintColor, "purple"),
     NumberOf(df$sporePrintColor, "white"),
     NumberOf(df$sporePrintColor, "yellow")) ,
   sporePrintColor_variables, main = "spore Print Color",
   col=sporePrintColor_variables)

# the following creates pie charts showing the
#proportions of the variables for gill related features
par(mfrow = c(2, 2))#allows multiple graphs to be displayed
                    #at once to they can be seen and compared easier
pie( c(NumberOf(df$gillAttachment, "attached"),
     NumberOf(df$gillAttachment, "descending"),
     NumberOf(df$gillAttachment, "free"),
     NumberOf(df$gillAttachment, "notched")) ,
   gillAttachment_variables, main = "gill-attachment",
   col = rainbow(length(gillAttachment_variables)))
pie( c(NumberOf(df$gillSpacing, "close"),
     NumberOf(df$gillSpacing, "crowded"),
     NumberOf(df$gillSpacing, "distant")),
   gillSpacing_variables, main = "gill-Spacing",
   col = rainbow(length(gillSpacing_variables)))
pie( c(NumberOf(df$gillSize, "broad"),
     NumberOf(df$gillSize, "narrow")),
   gillSize_variables, main = "gill-Size",
   col = rainbow(length(gillSize_variables)))
pie( c(NumberOf(df$gillColor, "black"),
     NumberOf(df$gillColor, "brown"),
     NumberOf(df$gillColor, "goldenrod4"),
     NumberOf(df$gillColor, "chocolate"),
     NumberOf(df$gillColor, "gray"),
     NumberOf(df$gillColor, "green"),
     NumberOf(df$gillColor, "orange"),
     NumberOf(df$gillColor, "pink"),
     NumberOf(df$gillColor, "purple"),
     NumberOf(df$gillColor, "red"),
     NumberOf(df$gillColor, "white"),
     NumberOf(df$gillColor, "yellow")) ,
   gillColor_variables, main = "gill-Color",col =gillColor_variables)

# the following creates pie charts showing the
#proportions of the variables for the stalk
par(mfrow = c(2, 3))#allows multiple graphs to be displayed
#at once to they can be seen and compared easier
pie( c(NumberOf(df$stalkShape, "enlarging"),
     NumberOf(df$stalkShape, "tapering")),
   stalkShape_variables, main = "stalk Shape",
   col = rainbow(length(stalkShape_variables)))
pie( c(NumberOf(df$stalkSurfaceAboveRing, "fibrous"),
     NumberOf(df$stalkSurfaceAboveRing, "scaly"),
     NumberOf(df$stalkSurfaceAboveRing, "silky"),
     NumberOf(df$stalkSurfaceAboveRing, "smooth")),
   stalkSurfaceAboveRing_variables, main = "stalk-Surface-Above-Ring",
   col = rainbow(length(stalkSurfaceAboveRing_variables)))
pie( c(NumberOf(df$stalkColorAboveRing, "brown"),
     NumberOf(df$stalkColorAboveRing, "goldenrod4"),
     NumberOf(df$stalkColorAboveRing, "sienna"),
     NumberOf(df$stalkColorAboveRing, "gray"),
     NumberOf(df$stalkColorAboveRing, "orange"),
     NumberOf(df$stalkColorAboveRing, "pink"),
     NumberOf(df$stalkColorAboveRing, "red"),
     NumberOf(df$stalkColorAboveRing, "white"),
      NumberOf(df$stalkColorAboveRing, "yellow")) ,
    stalkColorAboveRing_variables, main = "stalk-Color-Above-Ring",
    col = rainbow(length(stalkColorAboveRing_variables)))
pie( c(NumberOf(df$stalkRoot, "bulbous"),
      NumberOf(df$stalkRoot, "club"),
      NumberOf(df$stalkRoot, "cup"),
      NumberOf(df$stalkRoot, "equal"),
      NumberOf(df$stalkRoot, "rhizomorphs"),
      NumberOf(df$stalkRoot, "rooted")),
    stalkRoot_variables, main = "stalk-Root",
    col = rainbow(length(stalkRoot_variables)))
pie( c(NumberOf(df$stalkSurfaceBelowRing, "fibrous"),
      NumberOf(df$stalkSurfaceBelowRing, "scaly"),
      NumberOf(df$stalkSurfaceBelowRing, "silky"),
      NumberOf(df$stalkSurfaceBelowRing, "smooth")),
    stalkSurfaceBelowRing_variables, main = "stalk-Surface-Below-Ring",
    col = rainbow(length(stalkSurfaceBelowRing_variables)))
pie( c(NumberOf(df$stalkColorBelowRing, "brown"),
      NumberOf(df$stalkColorBelowRing, "goldenrod4"),
      NumberOf(df$stalkColorBelowRing, "sienna"),
      NumberOf(df$stalkColorBelowRing, "gray"),
      NumberOf(df$stalkColorBelowRing, "orange"),
      NumberOf(df$stalkColorBelowRing, "pink"),
      NumberOf(df$stalkColorBelowRing, "red"),
      NumberOf(df$stalkColorBelowRing, "white"),
      NumberOf(df$stalkColorBelowRing, "yellow")) ,
    stalkColorBelowRing_variables, main = "stalk-Color-Below-Ring",
    col = rainbow(length(stalkColorBelowRing_variables)))

# the following creates pie charts showing the
#proportions of the variables for veil and ring related features
par(mfrow = c(2, 2))#allows multiple graphs to be displayed
#at once to they can be seen and compared easier
pie( c(NumberOf(df$veilType, "partial"),
      NumberOf(df$veilType, "universal")) ,
    veilType_variables, main = "veil-Type",
    col = rainbow(length(veilType_variables)))
pie( c(NumberOf(df$veilColor, "brown"),
      NumberOf(df$veilColor, "orange"),
      NumberOf(df$veilColor, "white"),
      NumberOf(df$veilColor, "yellow")),
    veilColor_variables, main = "veil-Color",
    col = rainbow(length(veilColor_variables)))
pie( c(NumberOf(df$ringNumber, 0),
      NumberOf(df$ringNumber, 1),
      NumberOf(df$ringNumber, 2)),
    ringNumber_variables, main = "ring-Number",
    col = rainbow(length(ringNumber_variables)))
pie( c(NumberOf(df$ringType, "cobwebby"),
      NumberOf(df$ringType, "evanescent"),
      NumberOf(df$ringType, "flaring"),
      NumberOf(df$ringType, "large"),
      NumberOf(df$ringType, "none"),
      NumberOf(df$ringType, "pendant"),
      NumberOf(df$ringType, "sheathing"),
      NumberOf(df$ringType, "zone")) ,
    ringType_variables, main = "ring-Type",
    col =rainbow(length(ringNumber_variables)))

# the following creates pie charts showing the proportions
#of the variables for bruises, odor and spore print color
par(mfrow = c(1, 3))#allows multiple graphs to be displayed at
#once to they can be seen and compared easier
pie( c(NumberOf(df$edible, TRUE),NumberOf(df$edible, FALSE)) ,
    c("true", "false"), main = "edibile or not",col = rainbow(2))
pie( c(NumberOf(df$population, "abundant"),
      NumberOf(df$population, "clustered"),
      NumberOf(df$population, "numerous"),
      NumberOf(df$population, "scattered"),
      NumberOf(df$population, "several"),
      NumberOf(df$population, "solitary")) ,
    population_variables, main = "population",
    col = rainbow(length(population_variables)))
pie( c(NumberOf(df$habitat, "grasses"),
      NumberOf(df$habitat, "leaves"),
      NumberOf(df$habitat, "meadows"),
      NumberOf(df$habitat, "paths"),
      NumberOf(df$habitat, "urban"),
      NumberOf(df$habitat, "waste"),
      NumberOf(df$habitat, "woods")) ,
    habitat_variables, main = "habitat",
    col=rainbow(length(habitat_variables)))
par(mfrow = c(1, 1))
  #makes it so that only one graph is one each screen for later plots
#one thing to notice from these charts is that there are some values that are
#incredibly one sided and should be investigated more
```

```
NumberOf(df$veilType, "universal") #checks how many "universal" types mushrooms
                                   #there are
#as it turns out there are only partial type so this whole column can
#safely be removed with no consequence
df <- subset(df, select = -veilType )


######probability tables##########

table(df$capShape, df$edible,dnn =
       c("capShape", "edible"))
#creates table comparing the cap shape on whether it is edible
prop.table(table(df$capShape,
                df$edible,
                dnn = c("capShape", "edible")),1)
#probability table of above
#most of the varaibles are not to far from 50/50 so there is unlikely to be
#much we can get from just these specific values

table(df$capSurface, df$edible,dnn =
       c("cap-Surface", "edible"))
#creates table comparing edible with cap surgfce
#with wiether they are edible or not
prop.table(table(df$capSurface,
                df$edible,
                dnn = c("cap-Surface", "edible")),1)#probability table of above
#similar to the table above there isn't much to show  a direct corelation
#for these characteristics  other than  groves of which all are poisonous

table(df$capColor,
     df$edible,
     dnn = c("capColor", "edible"))
#creates table comparing if the mushroom is edible and with its cap colour
prop.table(table(df$capColor,
                df$edible,
                dnn = c("capColor", "edible")),1) #probability table of above
#probability table of above
#this shows some more correlation such as pink and goldenrod4 so these might be
#more defining when doing later algorithms



#######Cross-Validation and the Bootstrap##########

set.seed(1)
train <- sample(length(df$edible), length(df$edible)*0.5)
lm.fit <- lm(df$edible ~ df$capColor , data = df , subset = train)
mean((df$edible - predict(lm.fit, df))[-train]^2)
#only 0.1825958 very low error

#the following checks with a diffrent random set to  make sure that
#it is really that low
set.seed(2)
train <- sample(length(df$edible), length(df$edible)*0.5)
lm.fit <- lm(df$edible ~ df$capColor , data = df , subset = train)
mean((df$edible - predict(lm.fit, df))[-train]^2)
#only 0.1819106 which is very similar and still very low
#however it would be better to use logistic regression as the data is entirely
#categorical


######glm functions########

ggplot(df, aes(y = edible, x = capColor)) +
  geom_jitter(width = 0.05, height = 0.05, alpha = 0.5) +
  theme_classic(base_size = 15)
#creates a jittered plot of cap colour and if the mushroom is poisonous
#the tables above indicated that there are more likely to be a correlation for
#capColors
capColor_edible.glm <- glm(edible ~ capColor,
                           data = df,
                           family = binomial)
summary(capColor_edible.glm)
  #creates a binomial model of the same data as above
plogis(coef(capColor_edible.glm)[1]) #looks at the intercept
plogis(coef(capColor_edible.glm)[2]) #looks at the slope
predicted <- predict(capColor_edible.glm, type = "response", se.fit = TRUE)
df$fit <- predicted$fit
df$se <- predicted$se.fit #creates predictions for what has been plotted above
ggplot(df, aes( y = edible, x = capColor)) +
  geom_jitter(width = 0.05, height = 0.05, alpha = 0.5) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"),
              se = TRUE, color = "blue") +
  theme_classic(base_size = 15)
  #puts the plot on the same graph as above to compare it visually
  #we can see that there is not a specific correlation between these two type
  #most likely because there are purely discrete data (although intetersting
  #to note there are no goldenrod4 edible mushrooms)
df$se <- NULL
df$fit <- NULL  #removes the extra columns that where created since they are
              #no longer needed

edible_model_capColor <- glm(edible ~ capColor, data = df)
  #fits a logistic regression model with edible as our response variable and
  #and capColor as the explanatory variable
  #I chose these values since they seemed the most likely from above to give
  #us useful values
summary(edible_model_capColor) # creates a summary of above
  #the summary shows that all reduce the odds of edibility to varying degrees
  #except for red which slightly increases the odds of edibility

edible_model_all <- glm(edible ~ .,data = df,family = binomial)

                      #fits a logistic regression model with edible as our
#response variable and all other variables as covariates
#we are unfortunately told that the algorithm didn't converge
summary(edible_model_all) #creates a summary of above
#this shows us the chances of each mushroom being edible for each feature
#compared to the first variable of each feature

predict(edible_model_all,
        newdata = data.frame(capShape = "bell",
                             capSurface = "fibrous",
                             capColor = "brown",
                             bruises = TRUE,
                             odor = "almond",
                             gillAttachment = "attached",
                             gillSpacing = "close",
                             gillSize = "broad",
                             gillColor = "black",
                             stalkShape = "enlarging",
                             stalkRoot = "club",
                             stalkSurfaceAboveRing = "smooth",
                             stalkSurfaceBelowRing = "smooth",
                             stalkColorAboveRing = "brown",
                             stalkColorBelowRing = "brown",
                             veilColor = "white",
                             ringNumber = 0,
                             ringType = "evanescent",
                             sporePrintColor = "white",
                             population = "clustered",
                             habitat = "woods"),
        type = "response")
# Compare the estimated probabilities of a mushroom with the above qualities
#it is near zero that this mushroom is edible according to this predictor
#it does give a warning which is most likely because some of the features have
#very little variability however this is just the nature of the data and
#mushrooms themselves.
#This is very useful for when you want to check for a specific mushroom since
#you could enter observed characteristics of the mushroom in the above algorthm
#and it would be able to tell you the probability of it being edible or not.


model_stalkSurfaceBelowRing <- glm(edible ~ stalkSurfaceBelowRing,
                                   data = df,
                                   family = binomial)
hnp(model_stalkSurfaceBelowRing)
model_stalkColorBelowRing <- glm(edible ~ stalkColorBelowRing,
                   data = df,
                   family = binomial)
hnp(model_stalkColorBelowRing)

model_both <- glm(edible ~ stalkSurfaceBelowRing + stalkColorBelowRing,
                  data = df,
                  family = binomial)
hnp(model_both)
#the vast majority of all three of these plots lie within the envelope
#so they are likely a good fit




#####Leave-One-Out Cross-Validation#########

glm.fit <- glm(df$edible ~ df$capShape , data = df)
coef (glm.fit)
#the reference here is bell shaped and almost all are less likely to be edible
#than the reference

glm.fit <- glm(df$edible ~ df$capShape , data = df)
cv.err <- cv.glm (df , glm.fit)
cv.err$delta
```

```r
















######fitting trees#########

#through all the previous analysis I have decided that since almost all the
#data is categorical that decision trees will be a very good way of
#classifying this data

#the classifications only work with factors so all mut be converted to them
df_class <- df
df_class$edible <- as.factor(df$edible)
df_class$capShape <- as.factor(df$capShape)
df_class$capSurface <- as.factor(df$capSurface)
df_class$capColor <- as.factor(df$capColor)
df_class$bruises <- as.factor(df$bruises)
df_class$odor <- as.factor(df$odor)
df_class$gillAttachment <- as.factor(df$gillAttachment)
df_class$gillSpacing <- as.factor(df$gillSpacing)
df_class$gillSize <- as.factor(df$gillSize)
df_class$gillColor <- as.factor(df$gillColor)
df_class$stalkShape <- as.factor(df$stalkShape)
df_class$stalkRoot <- as.factor(df$stalkRoot)
df_class$stalkSurfaceAboveRing <- as.factor(df$stalkSurfaceAboveRing)
df_class$stalkSurfaceBelowRing <- as.factor(df$stalkSurfaceBelowRing)
df_class$stalkColorAboveRing <- as.factor(df$stalkColorAboveRing)
df_class$stalkColorBelowRing <- as.factor(df$stalkColorBelowRing)
df_class$veilColor <- as.factor(df$veilColor)
df_class$ringNumber <- as.factor(df$ringNumber)
df_class$ringType <- as.factor(df$ringType)
df_class$sporePrintColor <- as.factor(df$sporePrintColor)
df_class$population <- as.factor(df$population)
df_class$habitat <- as.factor(df$habitat)

set.seed(45)
train <- sample(length(df$edible), length(df$edible)*0.5)

tree.df_class=tree(edible~. , data = df_class)
summary(tree.df_class)
plot(tree.df_class)
text(tree.df_class, pretty = 0)
tree.df_class
#this has created a decision tree to help identify edible mushrooms which is
#exactly what I wanted, however I should do more checking to make sure it is
#accurate

df_class.test = df_class[-train,]
edible.test=df_class$edible[-train]
tree.df_class=tree(edible~. , data = df_class,subset=train)
tree.pred=predict(tree.df_class,df_class.test,type = "class")
table(tree.pred,edible.test)
#this shows that the accuracy is 100%, there are no false positives or false
#negatives, initial this looks to be a good thing however this could very
#possibly be a result of over fitting

set.seed(3)
cv.df_class=cv.tree(tree.df_class,FUN=prune.misclass)
names(cv.df_class)

par(mfrow=c(1,2))
plot(cv.df_class$size,cv.df_class$dev,type="b")
plot(cv.df_class$k,cv.df_class$dev,type="b")
#the above graphs show that it is unlikely that pruning the tree will help at
#all as it is very accurate and compact as it is
par(mfrow=c(1,1))

prune.df_class=prune.misclass(tree.df_class,best=5)
plot(prune.df_class)
text(prune.df_class,pretty=0)
#the above shows that it the original number of branches is in fact correct

#########random forest##############

set.seed(1)
```

```r
bag.df_class=randomForest(df_class$edible~., data=df_class, subset=train,
                        mtry=21, importance=TRUE)
bag.df_class
# the above creates random forests which shows very similar results to the
#trees above with no false posatives or false negatives

df_class.train = df_class[train,]
for (col in names(df_class.train)) {
  if (is.factor(df_class.train[[col]])) {
    df_class.train[[col]] <- factor(df_class.train[[col]],
                                  levels = levels(df_class[[col]]))
  }
}

df_class.test = df_class[-train,] #creates the test set from the training set
bag.df_class = randomForest(edible ~ ., data = df_class.train)
yhat.bag = predict(bag.df_class, df_class.test)

varImpPlot(bag.df_class)
#this shows how important the odor is specifically for checking the edibility of
#the mushrooms in the data set with sporePrintColor and a few others coming
#just after it

##########classification forest###########

#splits data into training and testing
set.seed(150)
split<-sample.split(df$edible, SplitRatio = 0.5)
training_set<-subset(df_class,split==TRUE)
test_set<-subset(df_class,split==FALSE)
topredict_set <-  dplyr::select(test_set,-edible)

#fits a decision tree
model_dtree<- rpart(edible ~ ., data=training_set)
preds_dtree<-predict(model_dtree,newdata=topredict_set,type="class")

conf_matrix_dtree <- table(preds_dtree, test_set$edible)
confusionMatrix(conf_matrix_dtree)
#this tree misclassifies 11 mushrooms as poisonous when they where really
#edible this is very accurate (and for someone who where to try to use this
#algorithm its good that's its being safe)
#the Kappa value and accuracy show how well this model does, and from a quick
#observation doesn't look quite as susceptible to overfitting

#creates forest
set.seed(1234)
model_rf<-randomForest(edible~.,data=training_set,importance=TRUE, ntree=1000)
preds_rf <- predict(model_rf, topredict_set)
(conf_matrix_forest <- table(preds_rf, test_set$edible))
confusionMatrix(conf_matrix_forest)
#above creates a forest which has a 100% accuracy no there are no
#misclassifcations unfortunately even though this looks like a good thing it
#it could be a result of over fitting, it could be that the data is very
#reliant on a few data points as seen in some of the graphs above

train_control <- trainControl(method = "cv", number = 10)
cv_model <- train(edible ~ ., data = df_class,
                  method = "rf",
                  trControl = train_control)
print(cv_model)
#the above creates and compares different subsets to compare the model
#it shows that that appose to what I'd initially thought the data is unlikely
#to be over fitting instead there are just a few elements that can describe
#edibility of the mushrooms in the data set very well
#However there is a chance that this would be specific to the mushrooms within
#this data set and could not be used on others.
```

Bibliography

Mushroom [Dataset]. (1981). UCI Machine Learning Repository. https://doi.org/10.24432/C5959T.