Analysis on Edibility of Mushroom

Final Project

Qisheng Zhang, Yang Xu,
Xinyue Liu, Yu Yang, Suyang Gao,
Yuting He, Dongwei Fu

Columbia University in the city of New York

supervised by
Prof. Demissie Alemayehu

December, 2017

Contribution

Qisheng Zhang (qz2278):
Exploratory Data Analysis

Xinyue Liu (xl2624):
Decision Tree Model

Yuting He (yh2855):
Random Forest Model

Yu Yang (yy2703):
Linear Discriminant Analysis Model

Yang Xu (yx2378):
Generalized Linear Model

Suyang Gao (sg3393):
Neural Network Model

Dongwei Fu (df2622):
Preparation for PowerPoint and Report

1 Introduction

1.1 Objective

To determine whether mushroom is edible or not based on observation of its features is a realistic problem when it comes to survival in the wild. We tried to select the most relevant features that decide the edibility of mushroom and build different models using these features to predict the edibility of mushroom. In the intermediate study, we noticed that the "odor" variable has dominating influence on the edibility of mushroom, and models including "odor" gave us perfect accuracy. So we also built models with and without "odor" variable and made comparison on the results. The models we used in this analysis are decision tree model, random forest model, linear discriminant analysis model, generalized linear model and neural network model.

2 Material

2.1 Data Source

Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf.

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like “leaflets three, let it be” for Poisonous Oak and Ivy.

2.2 Methods

I Decision Tree

Before analyze the data, we should check the distribution of the class variable in the dataset, which is mushroom type. If the levels distributed imbalanced, there might be some trouble to do the analysis.

```
> table(mushrooms$type)
  edible poisonous
  4208 3916
```

There are around 52 percent mushroom(N = 4,208) are edible, while 48 percent (N = 3,916) are poisonous. Thus, the level of the mushroom is kind of balanced.

The first step is training a model on the data. We will use the 1R implementation in the RWeka package, called OneR(). Applying the 1R classifier, chooses a single factor which is the most predictive among all the factor, and construct the rule.

```
(mushroom_1R <- OneR(class ~ ., data = mushrooms))
## odor:□
##a ->edible
## c -> poisonous□
## f -> poisonous□
##l ->edible□
## m -> poisonous□
##n ->edible□
## p -> poisonous□
## s -> poisonous□
## y -> poisonous□
## (8004/8124 instances correct)
```

From the output, we are able to see that the odor feature was selected for rule generation. The categories of odor specify rules for whether the mushroom is edible or poisonous. And if the mushroom smells fishy, foul, musty, pungent, spicy, or like creosote, the mushroom is poisonous. On the other hand, smells like almond or anise or none, indicate edible mushrooms.

The second step is evaluating model. We get the evaluation from summary the result of oneR. It lists a variety of ways to measure the performance of our oneR classifier.

```
summary(mushroom_1R)
##
```

```
## === Summary ===
##
## Correctly Classified Instances        8004               98.5229 %
## Incorrectly Classified Instances       120
## Kappa statistic                          0.9704
## Mean absolute error                      0.0148
## Root mean squared error                  0.1215
## Relative absolute error                  2.958  %
## Root relative squared error             24.323  %
## Total Number of Instances             8124
##
## === Confusion Matrix ===
##
##    a    b   <--  classified as
## 4208    0    |  a =  edible
##  120 3796    |  b = poisoneous
```

We can see that rules correctly specify 8,004 of the 8,124 mushroom samples, which is about 99 percent. In addition, in the confusion matrix, we found that the columns indicate the true class of the mushroom and the rows indicate the predicted values. Here, a = edible and b = poisonous. There are 120 values are misclassified, they are edible but been classified as poisonous. Furthermore, there are no mushroom, which are poisonous but been misclassified. To sum up, we can use the oneR rule to determine poisonousness, but might miss some mushrooms that are edible.

The last step is to improve model. Using JRip() to get a more complex and accurate learner. From JRip() classifier, which is also included in the RWeka

package, we conclude a nine rules from the mushroom data. The nine rules showed below.

```
(mushroom_JRip <- JRip(class ~ ., data = mushrooms))
## JRIP rules:
## ===========
##
## (odor = f) => class=poisonous (2160.0/0.0)
## (gill.size = n) and (gill.color = b) => class=poisonous (1152.0/0.0)
## (gill.size = n) and (odor = p) => class=poisonous (256.0/0.0)
## (odor = c) => class=poisonous (192.0/0.0)
## (spore.print.color = r) => class=poisonous (72.0/0.0)
## (stalk.surface.below.ring = y) and (stalk.surface.above.ring = k) =>
class=poisonous (68.0/0.0)
## (stalk.color.above.ring = y) => class=poisonous (8.0/0.0)
## (habitat = l) and (cap.color = w) => class=poisonous (8.0/0.0)
##  => class=edible (4208.0/0.0)
##
## Number of Rules : 9
```

II Random Forest

As Shivastava (2014) mentioned, Random Forest builds multiple Decision Trees. For instance, if we have 1000 observations from the complete population and 10 predictor variables, Random Forest will build multiple Decision Tree model by randomly selecting samples and variables. It can randomly take 100 samples from the whole observations, with replacement and 3 randomly chosen variables. It will repeat this process multiple times and then make the final classification on

each observation. Final prediction is a function of each classification, and it can be the mean of each classification or the mode of the classification.

III Linear Discriminant Analysis

Observations into two groups, Linear Discriminant Analysis can be applied. Linear Discriminant Analysis is one of machine learning methods to find a linear combination of data features to characterize two mushroom classes. While LDA is performed, the result is a linear combination of selected features to separate objects from dataset.

Response variables with factor level “edible” and “poisonous” are assigned into 0 and 1 binary data. Even all features in the dataset are category variables, while running the LDA algorithms, all features are assumed as conditional probability $P(X|Y = 0)$ and $P(X|Y = 1)$ to be normally distributed.

Eighty percent of data is trained and left twenty percent of data is test set to validate model generated from training set. First approach is to use all features of mushroom to characterize two classes. Error message of “collinearity error” showing up hint us not each feature necessary to be included. Without applying built-in method, AIC and BIC to select features, manual variable selection provides

more direct view of significance of feature as element of resulting linear combination classifier.

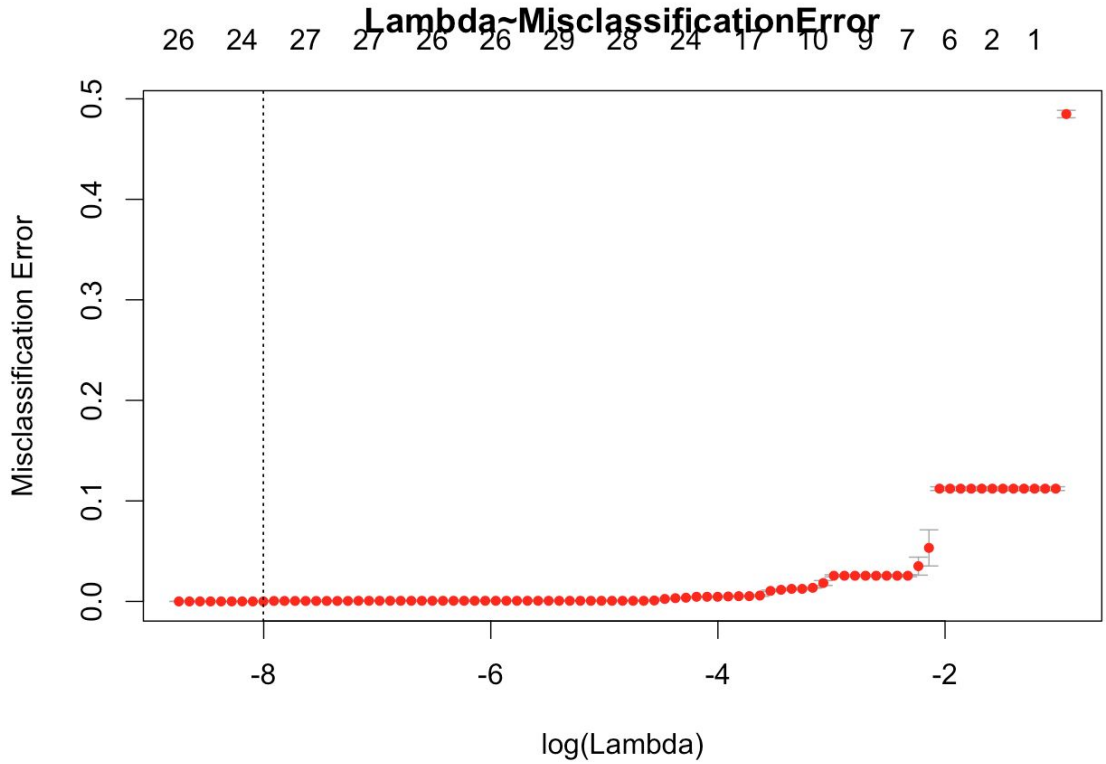
IV Generalized Linear Model

Since the data we use is categorical, we initially generalize a GLM model for the data.

Before we remove "odor" variable, the training error and the test error both equal 0, the accuracy here is 1. However, we find warning message in R shows that the fitted probability occurred numerically 0 or 1, which means it may have overfitting issue.

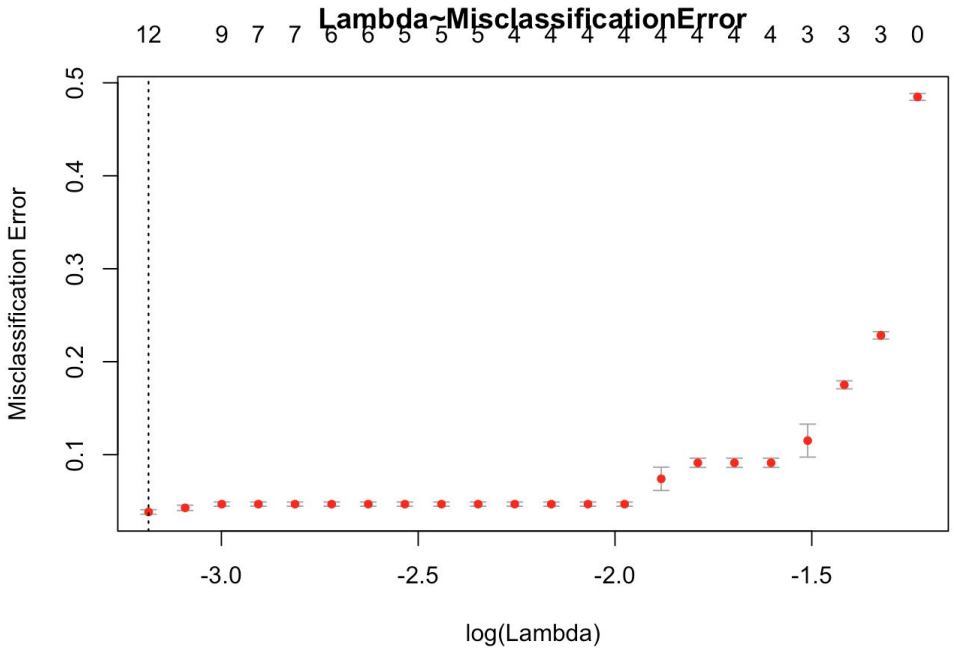
```
> mod1 = glm(class~., data = train, family = binomial, maxit = 30)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

In order to deal with overfitting, we use lasso method to reduce the model complexity. We apply cross validation method to find minimum lambda to minimize the misclassification error, and find the lambda is around 0.033%. To visualize the relation between lambda and misclassification error, we generalize the the corresponding graph.



And after the regularization, we make a prediction to get the test error and accuracy, and get the final model.

After we exclude “odor” variable, similar warning message occurs again, which indicates the overfitting issue still exists. We use lasso method to reduce the model complexity, and apply cross validation method to find minimum lambda to minimize the misclassification error, and find the lambda is around 4.14%.



We use similar method to do regularization, and make a prediction to get the test error and accuracy, as well as the final model.

V Neural Network

The methods are mainly concluded of data preparation, classifier configuration, model training and testing.

Firstly, we created a data frame object by reading the data plus the header in calling pd.read_csv().Then the program replaced the class of poisonous(p) and edible(e) to be 0 and 1. The program later transformed all the independent non-numeric variables into numerical equivalents. For all the 22 variables, we

created 111 features. Later, when we have decided to get rid of the most effective variable ‘odor’, we expanded the 21 variables to 102 features.

Secondly, we configured the classifier called DNNClassified with input of training data and 10, 20 and 10 for the first, second and third hidden layers. Since the input is around 100 features and the output is 2 classes. We assumed this configuration is reasonable.

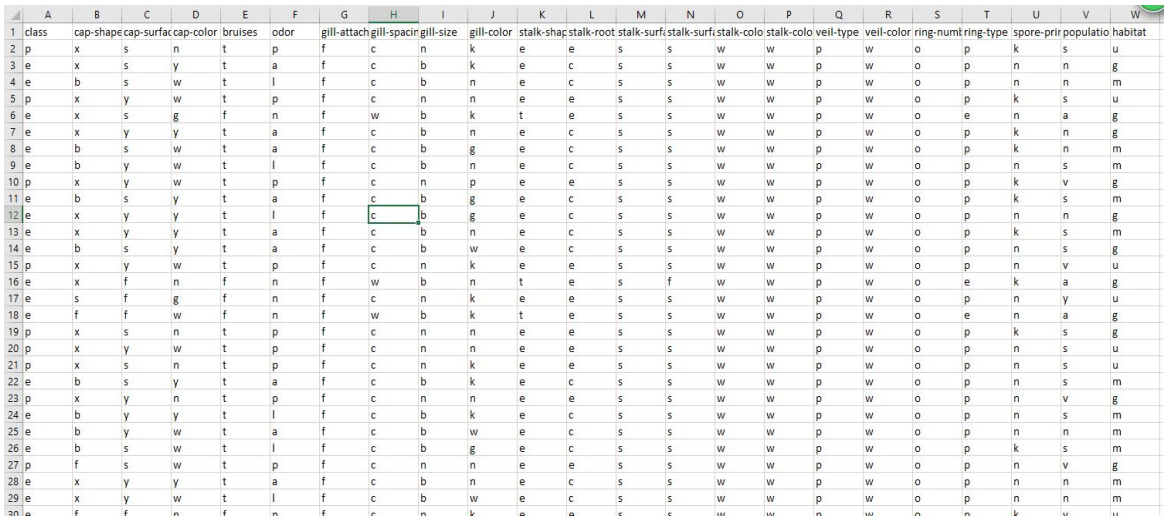
Lastly, we split the data into 80% for training and the rest 20% for test. After the model is trained, we test the accuracy. We did two types of the model with and without the column ‘odor’.

3 Results

3.1 Preliminary

We used EDA to analyze data sets to summarize their main characteristics. Using some tests and graphs to find out interesting facts about our dataset.

First, as shown below, all the variables in our data are categorical, this will affect our methodology to analyze data afterwards.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	class	cap-shape	cap-surfac	cap-color	bruises	odor	gill-attach	gill-spacir	gill-size	gill-color	stalk-shap	stalk-root	stalk-surfi	stalk-surfi	stalk-colo	stalk-colo	veil-type	veil-color	ring-numl	ring-type	spore-prir	populatio	habitat
2	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
3	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
4	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
5	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u
6	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
7	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	k	n	g
8	e	b	s	w	t	a	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	n	m
9	e	b	y	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	s	m
10	p	x	y	w	t	p	f	c	n	p	e	e	s	s	w	w	p	w	o	p	k	v	g
11	e	b	s	y	t	a	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	s	m
12	e	x	y	y	t	l	f	c	b	g	e	c	s	s	w	w	p	w	o	p	n	n	g
13	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	k	s	m
14	e	b	s	y	t	a	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	s	g
15	p	x	y	w	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	v	u
16	e	x	f	n	f	n	f	w	b	n	t	e	s	f	w	w	p	w	o	e	k	a	g
17	e	s	f	g	f	n	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	y	u
18	e	f	f	w	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
19	p	x	s	n	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	g
20	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	s	u
21	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	n	s	u
22	e	b	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	s	m
23	p	x	y	n	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	v	g
24	e	b	y	y	t	l	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	s	m
25	e	b	y	w	t	a	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	n	m
26	e	b	s	w	t	l	f	c	b	g	e	c	s	s	w	w	p	w	o	p	k	s	m
27	p	f	s	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	n	v	g
28	e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
29	e	x	y	w	t	l	f	c	b	w	e	c	s	s	w	w	p	w	o	p	n	n	m
30	e	f	f	n	f	n	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	y	u

There are 8124 records and 23 variables in total in this dataset. However, two of the 23 variables needs to be removed first.

```
>
> dim(data)
[1] 8124   23
>
>
```

The first one is “veil.type”, as shown below, it only has one level “p”, there is no point keeping it in our study.

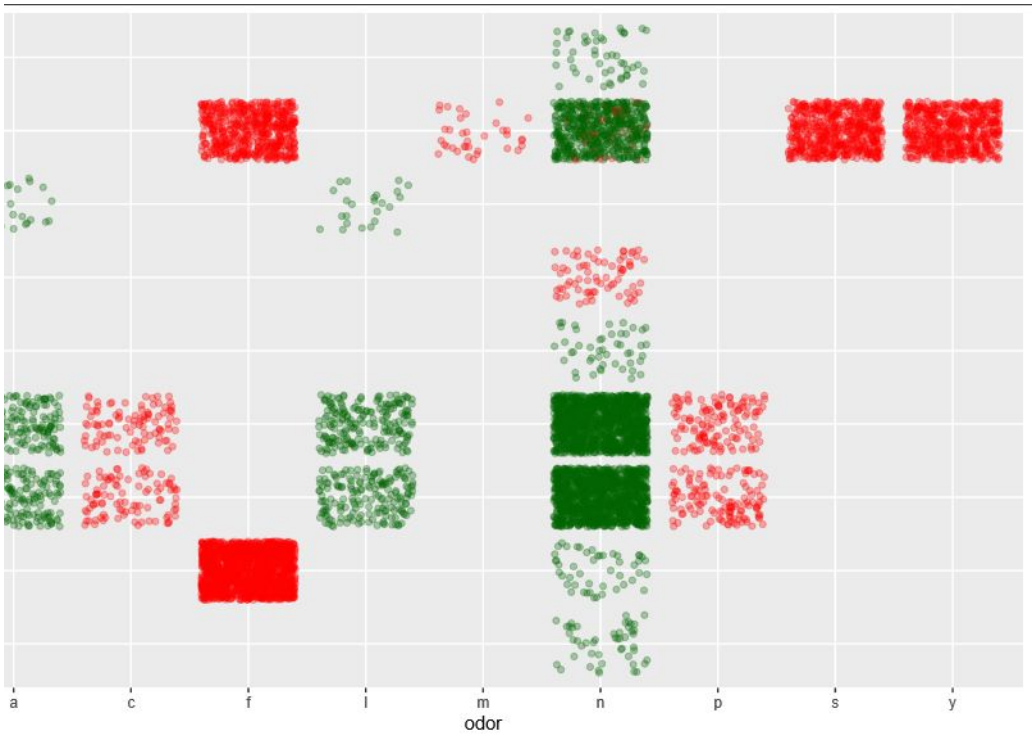
```
>
>
> unique(data$veil.type)
[1] p
Levels: p
>
>
```

The second one is “stalk-root”, because it has more than 4000 missing values, represented by “?”.

- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?

Finally, we have 8124 rows and 21 columns for our data.

Next, we try to find the correlations between each pair of variables. All variables are categorical, so we cannot make a correlation matrix to directly visualize everything. First, we did a Chi-Square Test for every single pair of different variables. The result turns out each pair of variables are correlated with each other, all p-values are 0. This is really weird to us, and not really helping to improve data. Therefore, we then pick some specific variables and plot them to see if we could find something.

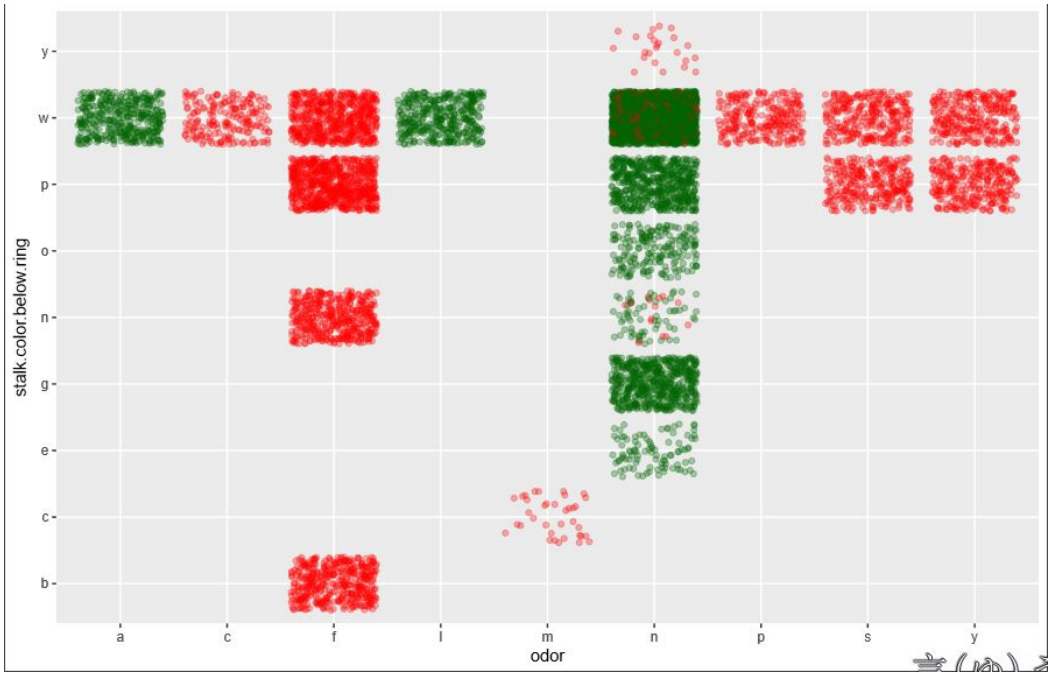


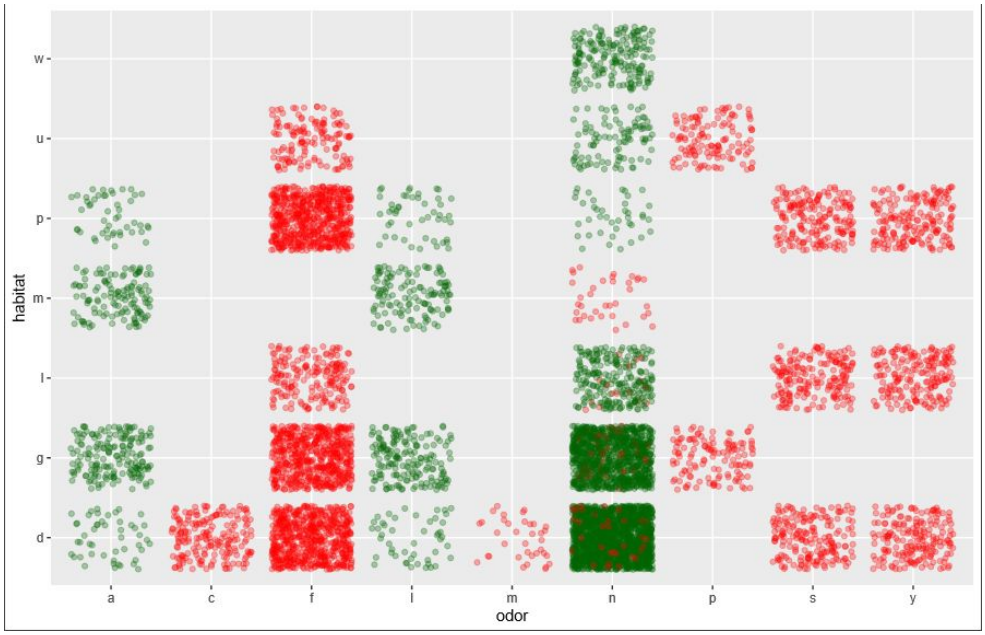
This plot above shows two variables “spore.print.color” and “odor”, for different levels under the two variables, green dots represent that mushrooms are

edible and red ones means poisonous. It is obvious that green area forms three vertical lines, odor a, l and n. But Horizontally, we did not find anything interesting, everything is mixed.

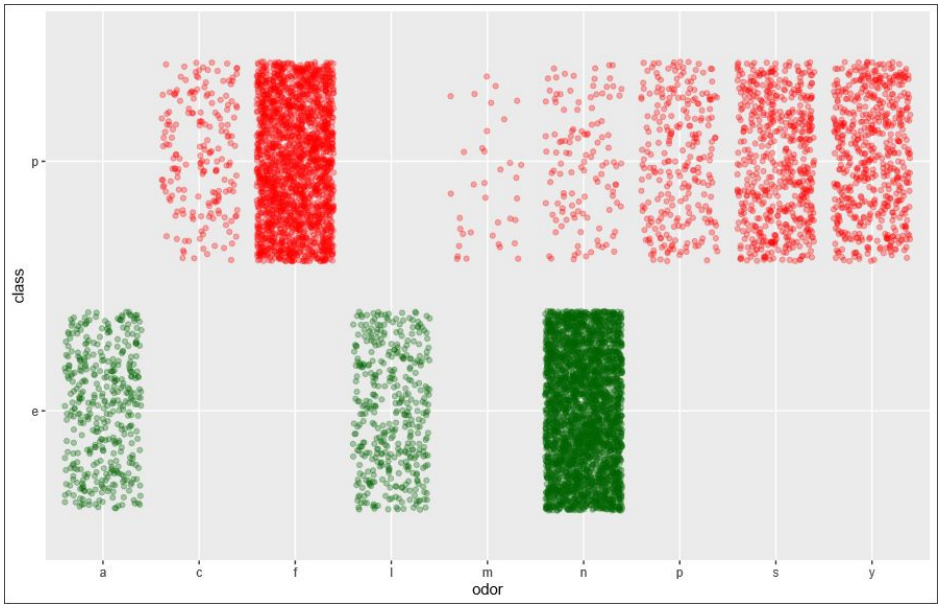
It seems that “odor” is a pretty strong variable do determine whether mushrooms are edible or not, we make some more graphs of “odor” with other variables. The following two plots shows “odor” with “stalk.color.below.ring” and “odor” with “habitat”.

Same for these plots, odor a, l and n are almost all green. No strong pattern





Here is a graph of just “odor” itself. Mushrooms with odor a and l are perfectly safe to eat, and for odor n, most mushrooms are edible. The rest odors are 100% poisonous.



"Odor" is too strong to differentiate whether mushrooms are edible or not, therefore, it might make our models later not reliable. We will try to include "odor" in our model first, and take it out and see if there will be a difference.

3.2 Main Results

I Decision Tree

The number shows the accurate classified, and misclassified. There is no misclassified following the 9 rules. To transfer the R rules:

Features	Class
odor is foul	poisonous
gill size is narrow and the gill color is buff	poisonous
gill size is narrow and the odor is pungent	poisonous

odor is c	poisonous
spore.print.color is green	poisonous
stalk.surface.below.ring is scaly and the stalk.surface.above.ring is silky	poisonous
stalk.color.above.ring is scaly	poisonous
habitat is leaves and cap.color is white	poisonous
Otherwise	edible

Since the factor Odor has great effect at the result, we simple remove this factor from the data set, and do the model again. We found that in oneR model factor spore.print.color feature was selected for rule generation. It has 86.9045% accuracy. From JRip() classifier, we conclude a new nine rules.

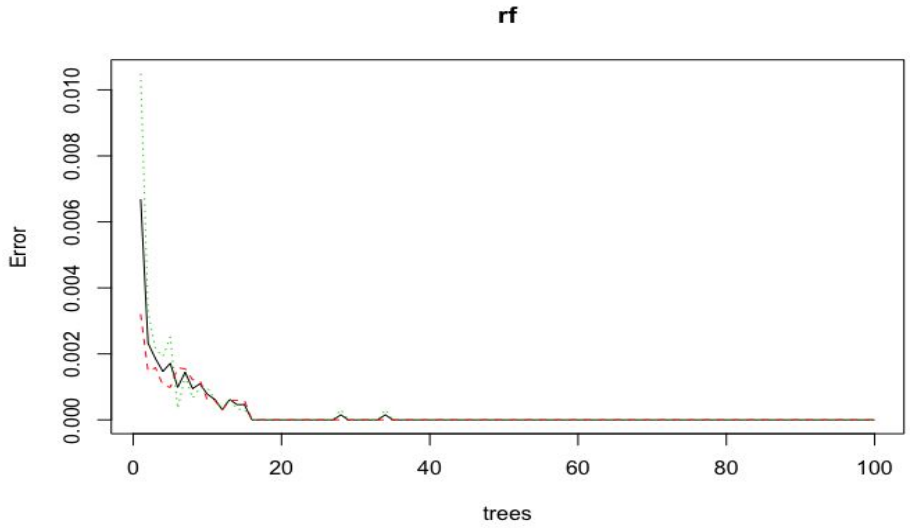
Features	Class
stalk surface above ring is silky and gill spacing is close	poisonous
gill size is narrow and gill color is buff	poisonous
gill size is narrow and population is scattered	poisonous
spore print color is chocolate and cap surface is smooth	poisonous
gill size is narrow and stalk shape is enlarging and bruises is no	poisonous

stalk root is bulbous and stalk color below ring is white	poisonous
gill size is narrow and cap shape is convex	poisonous
stalk color above ring is yellow	poisonous
Otherwise	edible

II Random Forest

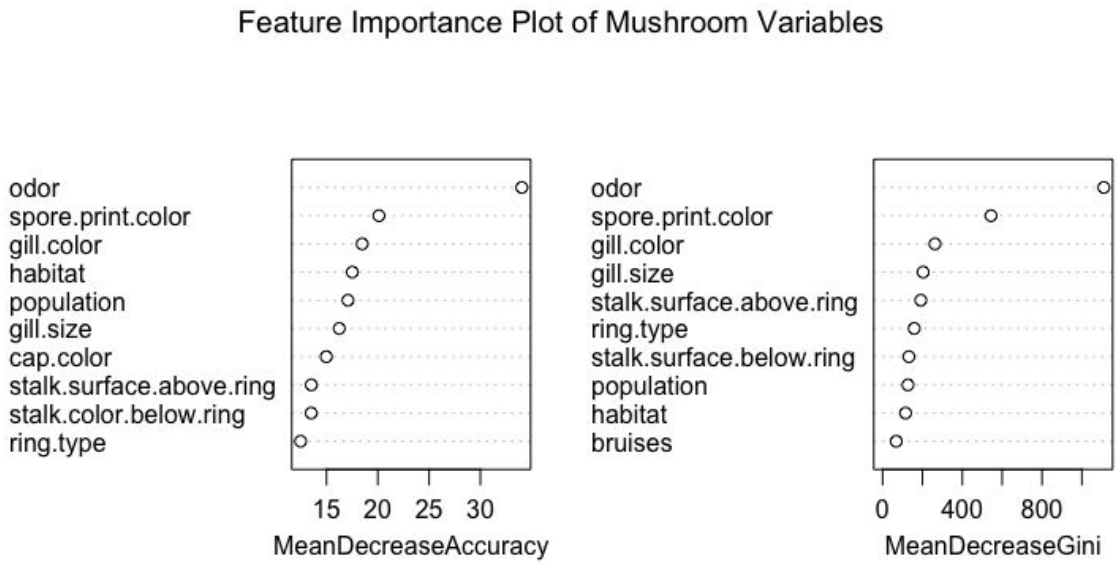
We can use the R package "RandomForest" to conduct the analysis. The package also provides useful functions to select variables. The output from the function "randomForest" will automatically give the error rate of the classification. By taking 80% of the dataset as the training data and the rest 20% as the test data, we have obtained the following result.

```
## Call:
##  randomForest(formula = class ~ ., data = train.data, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##      e    p class.error
## e 3373    0           0
## p    0 3126           0
```



Based on the result, Random Forest is quite accurate that it has an error rate of 0. We can see from the plot that the error decreases significantly after 100 trees, and it reaches 0 using 18 trees or more. Since Random Forest can capture the variance of the predictors at the same time while enabling large number of observations in the classification, it tends to give a much accurate prediction compared to the single Decision Tree model or regression models in many scenarios, especially those with huge sample size and many predictors.

The Variance Importance Plot gives the 10 most significant variables in the classification. The result shows that “odor”, “spore.print.color”, and “gill.color” are the three most important variables in the classification.

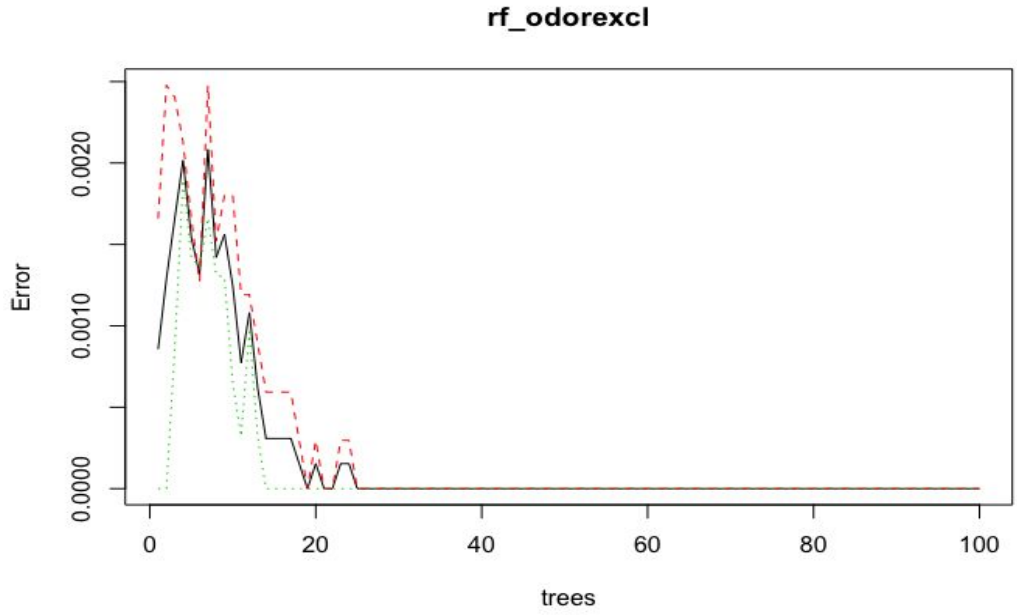


According to Metagenomics. Statistics, the mean decrease in accuracy of a variable is calculated during the out of bag error calculation phase. The more decrease in the accuracy of the random forest because of the exclusion (or permutation) of a single variable, the more important that variable is deemed, and therefore variables with a large mean decrease in accuracy are more important for the classification of the data. On the other hand, as mentioned in Metagenomics. Statistics, the mean decrease in Gini coefficient is a measure of how each variable contributes to the homogeneity of the nodes in the resulting random forest. Each time a particular variable is used to split a node, the Gini coefficient for the child nodes are calculated and compared to that of the original node. The Gini coefficient is a measure of homogeneity from 0 (homogeneous) to 1

(heterogeneous). The changes in Gini are summed for each variable and normalized at the end of the calculation. Variables that result in nodes with higher purity have a higher decrease in Gini coefficient.

Since “odor” is the most significant variable and we have a quite accurate classification using Radom Forest model. It is reasonable to assume that “odor” is a deterministic factor in the classification. After the exclusion of the “odor” variable, the error increases and fluctuates significantly but it still reaches 0 after 22 trees. Therefore, the accuracy of the Random Forest will largely depend on the number of trees we build.

```
## Call:
##  randomForest(formula = class ~ ., data = train.data[, -which(names(mushro
om) ==      "odor")], ntree = 20)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0
## Confusion matrix:
##      e    p class.error
## e 3372    1 0.000296472
## p    0 3125 0.000000000
```



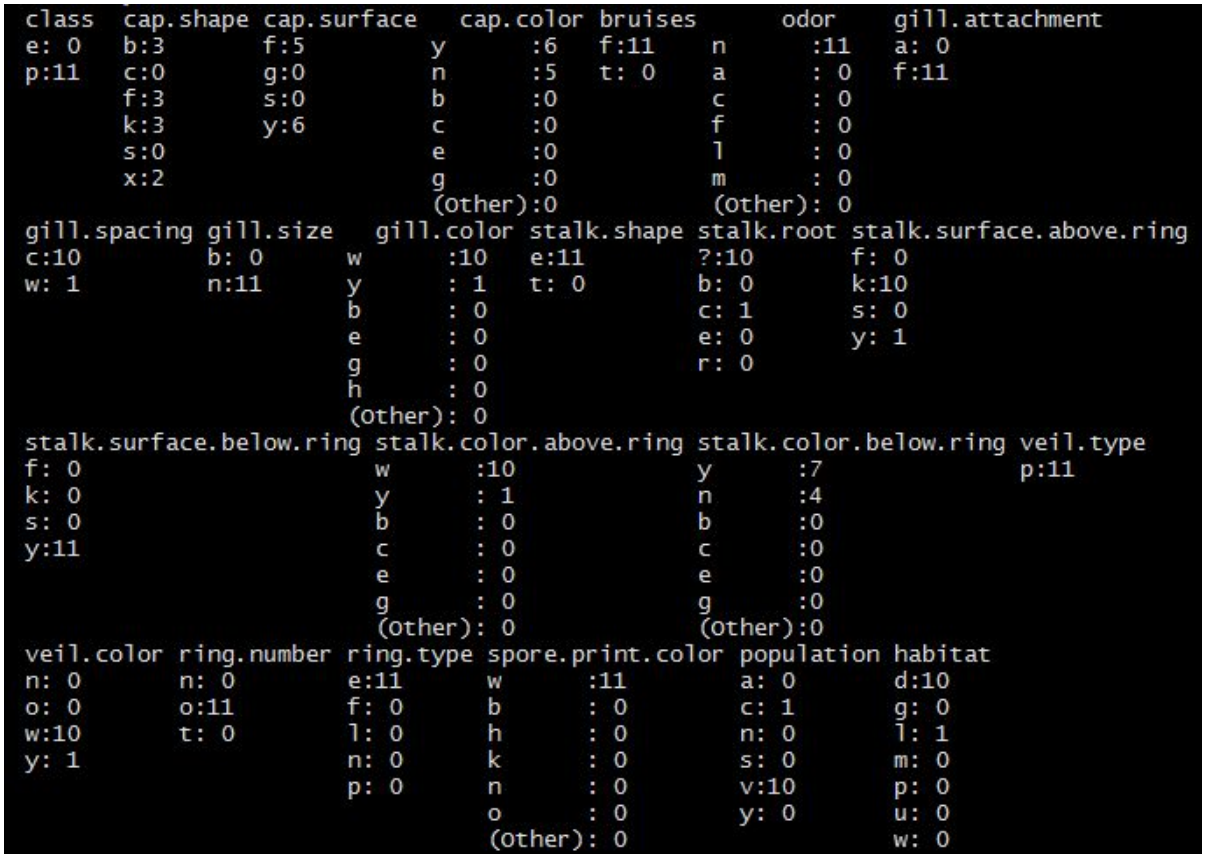
III Linear Discriminant Analysis

Habitat, Odor and spore.print.color are included to generate linear combination classifier. Adding additional feature does not significantly enhance prediction accuracy, only above three variables are considered.

Confusion Matrix is shown as follows,

	Edible Prediction	Poisonous Prediction
True Edible	860	0
True Poisonous	11	754

Testing set prediction error derived from above table is 0.006769231 which is extremely small. 11 observations of mushroom which poisonous are misclassified into edible ones. To understand potential factors leading to misclassification, detail feature of these 11 observations are presented as follows,



All these 11 observation bruises marked as “false”, gill.attachment is “free”, odor marked as “none”, gill.size marked as “narrow”. Adding these features to the LDA to test if these features can improve the prediction accuracy. However, at this stage, none of these features provide significant enhancement on prediction. 11 poisonous mushroom still be categorized into edible ones.

In addition, we find that odor of mushroom has significant influence on classifying mushroom. LDA model with Odor only, the test prediction table is shown as follows,

	Edible Prediction	Poisonous Prediction
True Edible	860	0
True Poisonous	28	737

In the second model, Odor is dropped and other available features are considered to produce a new linear combination classifier. Repeating manually select features, in the new LDA model, Habitat, spore.print.color, population, ring.type, gill.color, gill.size, bruises and cap.coloar are included and test prediction table is as follows,

	Edible Prediction	Poisonous Prediction
True Edible	838	22
True Poisonous	0	765

Unlike previous LDA model, 11 poisonous mushrooms classified as edible ones, the new LDA model predict none of poisonous mushrooms as edible ones. However, 22 edible mushrooms are characterized as poisonous. Testing error is 0.001723077.

Detail features of 22 misclassified mushrooms,

```
class  cap.shape cap.surface   cap.color bruises      odor   gill.attachment
e:22   b: 0      f:8         n      :8   f: 8    l      :9   a: 0
p: 0   c: 0      g:0         y      :8   t:14    n      :8   f:22
       f:10      s:9         w      :6           a      :5
       k: 2      y:5         b      :0           c      :0
       s: 0                  c      :0           f      :0
       x:10                  e      :0           m      :0
                             (other):0           (other):0
gill.spacing gill.size   gill.color stalk.shape stalk.root
c: 0         b: 0      w      :13   e: 8        ?: 0
w:22         n:22      n      : 5   t:14       b:22
                       p      : 4               c: 0
                       b      : 0               e: 0
                       e      : 0               r: 0
                       g      : 0
                       (other): 0
stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
f: 4                     f: 5                     w      :22
k: 0                     k: 0                     b      : 0
s:18                     s:17                     c      : 0
y: 0                     y: 0                     e      : 0
                                                  g      : 0
                                                  n      : 0
                                                  (other): 0
stalk.color.below.ring veil.type veil.color ring.number ring.type
w      :14             p:22      n: 0       n: 0        e: 8
n      : 8                       o: 0       o:22        f: 0
b      : 0                       w:22       t: 0        l: 0
c      : 0                       y: 0                   n: 0
e      : 0                                              p:14
g      : 0
(other): 0
spore.print.color population habitat
n      :14        a: 0       d:14
w      : 8        c: 0       g: 0
b      : 0        n: 0       l: 8
h      : 0        s: 0       m: 0
k      : 0        v:22       p: 0
o      : 0        y: 0       u: 0
(other): 0                   w: 0
```

Attempting to add feature into model where the similar feature all 22 mushrooms share. However, there is no significant improvement on prediction.

IV Generalized Linear Model

Before we exclude the “odor” variable, we make a prediction and get the test error equals 0, with accuracy equals 1, which means we can reliably predict

whether a mushroom is edible based on certain features of the mushroom. In order to analyze the model quantitatively, we use R to get the beta of the model.

```
> final_mod$beta
89 x 1 sparse Matrix of class "dgCMatrix"
                                      s0
cap.shapec                -1.819685928
cap.shapef                 .
cap.shapek                 .
cap.shapes                 0.538916336
cap.shapex                 .
cap.surfaceg              -2.860136310
cap.surfaces               .
cap.surfacey               .
cap.colorc                 1.451887483
cap.colore                 .
cap.colorg                 .
cap.colorn                 .
cap.colorp                 .
cap.colorr                 .
cap.coloru                 .
cap.colorw                 .
cap.colory                 .
bruisest                   .
odorc                     -9.191628797
odorf                    -11.601386713
odorl                      1.036043868
odorm                     -0.008132338
odorn                      4.161026859
odorp                     -6.923110619
odors                     -1.831489013
odory                     -1.866473394
gill.attachmentf           .
gill.spacingw              3.097729291
gill.sizen                -5.270825299
gill.colore                .
gill.colorg                .
gill.colorh                .
gill.colork                .
gill.colorn                .
```

Based on the beta we get, we conclude the final model with “odor”:

Class=-1.82cap.shapec+0.54cap.shapes-2.86cap.surfaceg+1.45cap.colorc-9.19odorc-11.6odorf+1.04odorl-0.01odorm+4.16odorn-6.92odorp-1.83odors-1.87odory+3.10gill.spacingw-5.27gill.sizen-2.29stalk.surface.abov

e.ringk-0.40stalk.surface.below.ringy-0.001stalk.color.above.ringy-2.71stalk.color.below.ringy+0.73ring.typef+0.26spore.print.colorn-15.52spore.print.colorr+1.48spore.print.coloru-4.76spore.print.colorw-4.16populationc+0.18populationn+5.18habitatw

After we exclude the "odor" variable, we apply similar method and find get the test error equals 0.04615, and accuracy equals 0.95385. We apply same method in R to get the beta.

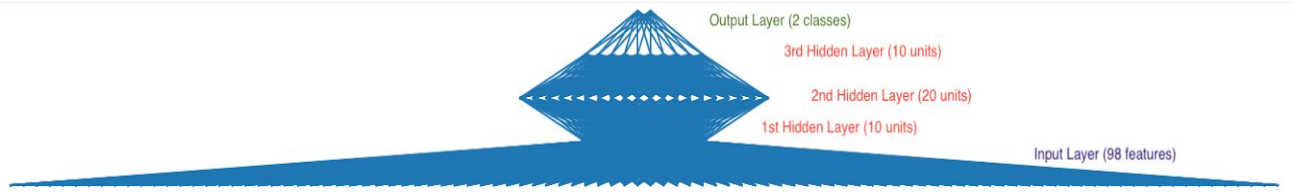
```
> final_mod$beta
79 x 1 sparse Matrix of class "dgCMatrix"
                                      s0
cap.shapec                .
cap.shapef                .
cap.shapek                .
cap.shapes                0.7041149522
cap.shapex                .
cap.surfaceg              .
cap.surfaces              .
cap.surfacey              .
cap.colorc                .
cap.colore                .
cap.colorg                .
cap.colorn                .
cap.colorp                .
cap.colorr                .
cap.coloru                .
cap.colorw                .
cap.colory                .
bruisest                  .
gill.attachmentf          .
gill.spacingw             0.6463881361
gill.sizen               -3.3441066068
gill.colore               .
gill.colorg               .
gill.colorh               .
gill.colork               .
```

Based on the beta we get, we conclude the final model without "odor":

Class=0.70cap.shapes+0.65gill.spacingw-3.34gill.sizen-1.02stalk.surface.above.ringk-0.58stalk.surface.below.ringk-0.48stalk.color.above.ringc-0.00stalk.color.below.ringc-3.05spore.print.colorh+0.11spore.print.colorn-2.31spore.print.colorr+0.49spore.print.coloru-0.03spore.print.colorw

V Neural Network

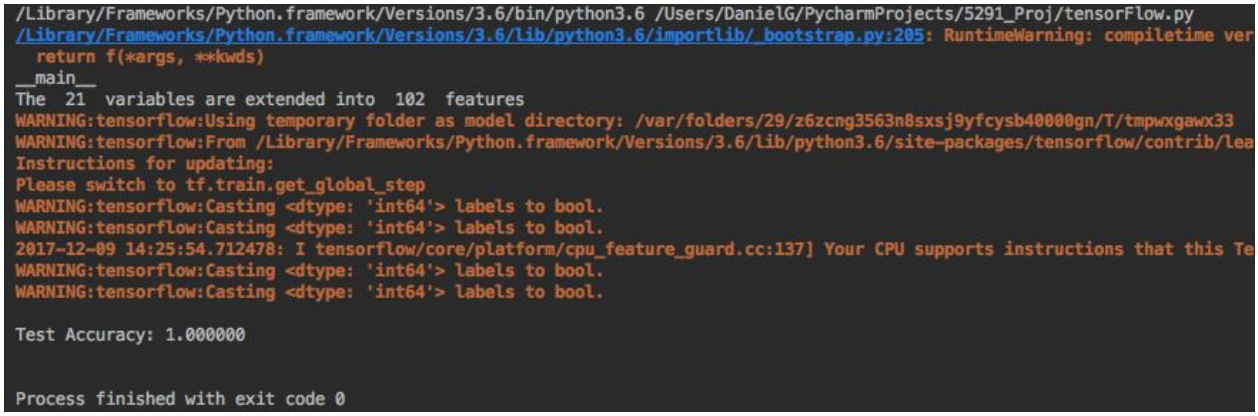
First of all , we made a visualization of how the structure looks like from the input of 20 variables to 98 features to the output after the EDA data cleaning.



As for accuracy, firstly we get the accuracy of 1.00 when using all the variables.

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/DanielG/PycharmProjects/5291_Proj/tensorFlow.py
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime v
  return f(*args, **kwds)
__main__
The  22  variables are extended into  111  features
WARNING:tensorflow:Using temporary folder as model directory: /var/folders/29/z6zcng3563n8sxsj9yfcysb40000gn/T/tmpxlrpvalt
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/tensorflow/contrib/l
Instructions for updating:
Please switch to tf.train.get_global_step
WARNING:tensorflow:Casting <dtype: 'int64'> labels to bool.
WARNING:tensorflow:Casting <dtype: 'int64'> labels to bool.
2017-12-09 14:23:25.538957: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this
WARNING:tensorflow:Casting <dtype: 'int64'> labels to bool.
WARNING:tensorflow:Casting <dtype: 'int64'> labels to bool.

Test Accuracy: 1.000000


Process finished with exit code 0
```

Later, we test again with all the variables but the ‘odor’. We found that the accuracy remains 1.00.



4 Conclusion

4.1 Relevance of Findings

In the previous EDA and Decision Tree analysis, we found that ‘odor’ is a variable with strong determination and effect on the result. Models containing ‘odor’ gives perfect accuracy. After removing, ‘odor’, the LDA model still gives very small test error, and it misclassified edible mushroom into poisonous ones. Combining with real condition, the model is to classify mushrooms, it is safe to say that predicting edible ones to poisonous ones is better than misleading poisonous mushrooms are edible. Therefore, second LDA model without Odor feature is preferred. In neural network, we found the accuracy could remains still pretty high after removing this important variable. We can say the neural network may not be affected by specific variable due to the fact that we converted the all the variables

to features. Decreasing from 111 features to 102 features, therefore, looks like has small impact on the information maintains in each input.

4.2 Plausibility of Assumptions

In the Linear Discriminant Analysis, all features are assumed to be multivariate normal distributed where each variable is continuous ones. Since all features are categorical variables and dummy variable assigned to turn into binary data. In this case, the fundamental normal distribution assumption is violated. Categories data can be transformed into continuous one by applying certain technique, however it leaves difficult explanation on each factor level in each feature of mushroom from data set. To handle binary data, logistic regression is likely to a better alternative than LDA.

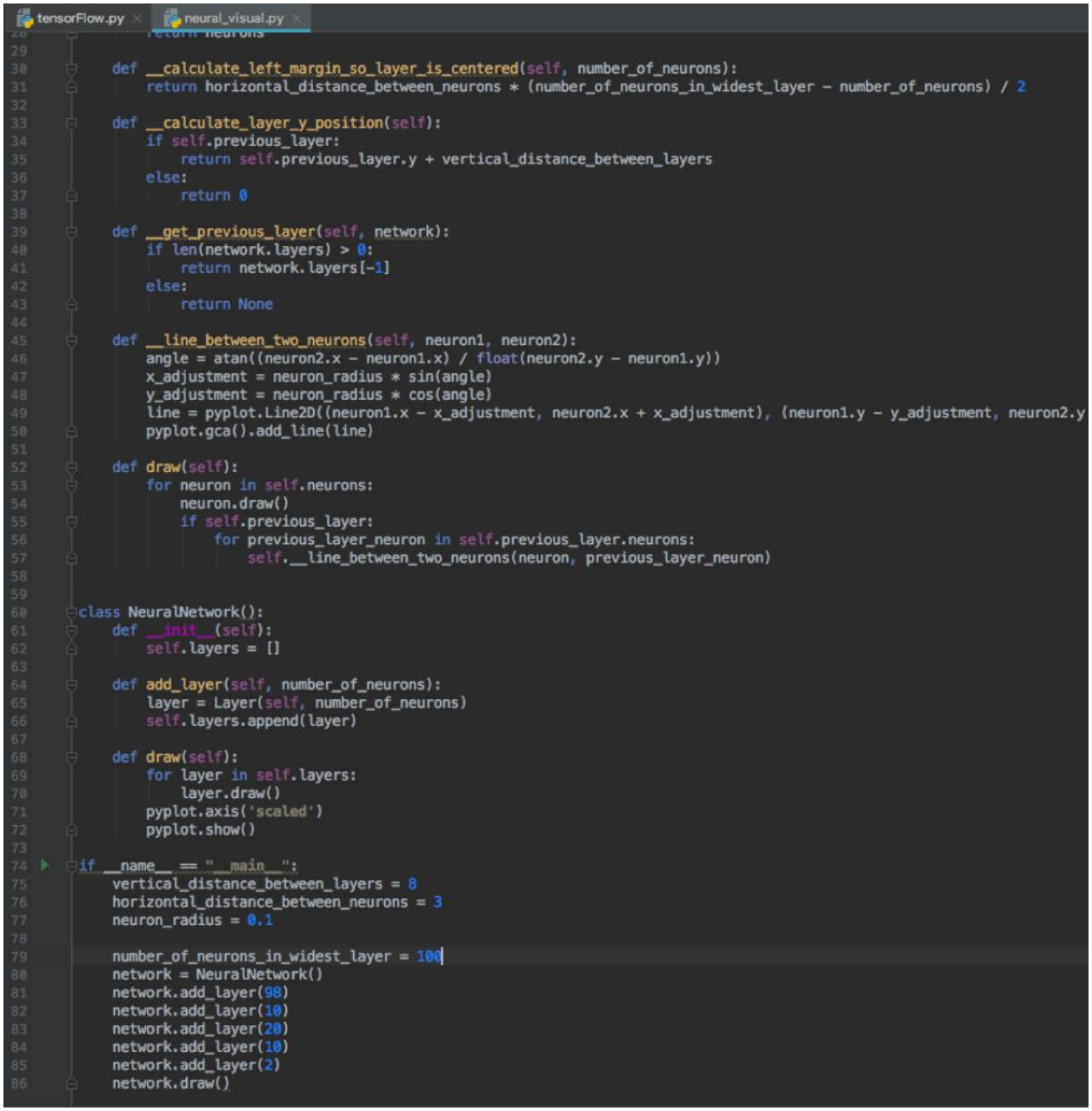
Appendix

```
#RandomForest

mushroom <- read.csv("mushrooms.csv", header=T)
##drop the "veil.type" since it has only 1 level
mushroom <- mushroom[ ,-which(names(mushroom)=="veil.type")]

##create training &test data
###test data contains 20% of the dataset
set.seed(1)
train.ind <- sample(1:nrow(mushroom),size = nrow(mushroom)*0.8, replace=T)
train.data = mushroom[train.ind,]
test.data = mushroom[-train.ind,]
library(randomForest)
##RandomForest model
rf <- randomForest(class~ ., data=train.data, ntree=100)
rf
plot(rf)
##Variable Importance plot
varImpPlot(rf,sort = T, n.var = 10, main="Feature Importance Plot of Mushroom
Variables")
##RandomForest exclude odor variable
rf_odorexcl <- randomForest(class ~ .,  data =
train.data[,-which(names(mushroom)=="odor")], ntree=100)
rf_odorexcl
plot(rf_odorexcl)

#Neural Network
```

tensorFlow.py | neural_visual.py

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split

import os

from tensorflow.contrib.learn import DNNClassifier

dimension = None

def prepare_data(data_file_name):

    #header = ['class', 'cap_shape', 'cap_surface',
     #          'cap_color', 'bruises', 'odor', 'gill_attachment',
      #         'gill_spacing', 'gill_size', 'gill_color', 'stalk_shape',
       #        'stalk_root', 'stalk_surface_above_ring',
        #       'stalk_surface_below_ring', 'stalk_color_above_ring',
         #      'stalk_color_below_ring', 'veil_type', 'veil_color',
          #     'ring_number', 'ring_type', 'spore_print_color',
           #    'population', 'habitat']
    header = ['class', 'cap_shape', 'cap_surface',
              'cap_color', 'bruises','gill_attachment',
              'gill_spacing', 'gill_size', 'gill_color', 'stalk_shape',
              'stalk_root', 'stalk_surface_above_ring',
              'stalk_surface_below_ring', 'stalk_color_above_ring',
              'stalk_color_below_ring', 'veil_type', 'veil_color',
              'ring_number', 'ring_type', 'spore_print_color',
              'population', 'habitat']

    df = pd.read_csv(data_file_name, sep=',', names=header)

    df['class'].replace('p', 0, inplace=True)
    df['class'].replace('e', 1, inplace=True)

    cols_to_transform = header[1:]
    df = pd.get_dummies(df, columns=cols_to_transform)

    df_train, df_test = train_test_split(df, test_size=0.2)

    num_train_entries = df_train.shape[0]
    num_train_features = df_train.shape[1] - 1

    dimension = num_train_features
    print('The ',cols_to_transform.__len__(),' variables are extended into ',dimension,' features')

    num_test_entries = df_test.shape[0]
    num_test_features = df_test.shape[1] - 1

    df_train.to_csv('train_temp.csv', index=False)
    df_test.to_csv('test_temp.csv', index=False)

    open("mushroom_train.csv", "w").write(str(num_train_entries) +
                                          "," + str(num_train_features) +
                                          "," + open("train_temp.csv").read())
```

```
    os.remove("train_temp.csv")
    os.remove("test_temp.csv")

# Define the test inputs
def get_test_inputs():
    x = tf.constant(test_set.data)
    y = tf.constant(test_set.target)

    return x, y

# Define the training inputs
def get_train_inputs():
    x = tf.constant(training_set.data)
    y = tf.constant(training_set.target)

    return x, y

if __name__ == "__main__":

    print('__main__')

    MUSHROOM_DATA_FILE = "mushrooms_data_copy2.csv"

    prepare_data(MUSHROOM_DATA_FILE)

    # Load datasets.
    training_set = tf.contrib.learn.datasets.base.load_csv_with_header(
        filename='mushroom_train.csv',
        target_dtype=np.int,
        features_dtype=np.int,
        target_column=0)

    test_set = tf.contrib.learn.datasets.base.load_csv_with_header(
        filename='mushroom_test.csv',
        target_dtype=np.int,
        features_dtype=np.int,
        target_column=0)

    # Specify that all features have real-value data
    feature_columns = [tf.contrib.layers.real_valued_column("", dimension=111)]

    # Build 3 layer DNN with 10, 20, 10 units respectively.
    classifier = DNNClassifier(
        feature_columns=feature_columns,
        hidden_units=[20, 20, 10],
        n_classes=2,
        )

    # Fit model.
    classifier.fit(input_fn=get_train_inputs, steps=2000)

    # Evaluate accuracy.
    accuracy_score = classifier.evaluate(input_fn=get_test_inputs,
                                         steps=1)["accuracy"]

    print("\nTest Accuracy: {0:f}\n".format(accuracy_score))
```



#Decision Tree

```
library(RWeka)
rm(list=ls())
setwd("~/Desktop")
mushrooms <- read.csv("mushrooms.csv", header = T, stringsAsFactors = TRUE)
mushrooms$class = sapply(mushrooms$class, function(x){ifelse(x=='e' , 'edible',
'poisonous'})
mushrooms$class <- as.factor(mushrooms$class)
```

```
table(mushrooms$class)
mushrooms2 <- mushrooms[,-6]
mushroom_1R <- OneR(class ~ ., data = mushrooms)
mushroom_1R2 <- OneR(class ~ ., data = mushrooms2)
summary(mushroom_1R)
summary(mushroom_1R2)
mushroom_JRip <- JRip(class ~ ., data = mushrooms)
mushroom_JRip2 <- JRip(class ~ ., data = mushrooms2)


#GLM

library(glmnet)
path = "/Users/yangxu/Desktop/mushrooms.csv"
dt = read.csv(path, header = T)
dt$class <- relevel(dt$class, ref = "p")
dt = dt[, -c(17)]
n = dim(dt)[1]
set.seed(1)
train_idx = sample(n, size = floor(0.8*n), replace = FALSE)
train = dt[train_idx, ]
test = dt[-train_idx, ]
mod1 = glm(class~., data = train, family = binomial, maxit = 30)
glm.full <- mod1
glm.null <- glm(class ~ 1, data = train,  family = binomial, maxit = 30)
final_mod = step(glm.null, scope = list(lower = glm.null, upper = glm.full),
                 direction = "both", trace = 10, k = log(nrow(train)))
summary(final_mod)
train_y = predict(final_mod, train, type = "response")
train_y[train_y > 0.5] = "e"
train_y[train_y <= 0.5] = "p"
train_error = mean(train_y != train$class)
train_error
accuracy = 1-train_error
accuracy
test_y = predict(final_mod, test, type = "response")
test_y[test_y > 0.5] = "e"
test_y[test_y <= 0.5] = "p"
test_error = mean(test_y != test$class)
```

```
test_error
accuracy = 1-test_error
accuracy
test_y_prob = predict(final_mod, test, type = "response")
test_y_class = rep('e', times = length(test_y_prob))
test_y_class[test_y_prob > 0.5] = "e"
test_y_class[test_y_prob <= 0.5] = "p"
test_error = mean(test_y_class != test$class)
test_error

train_y_prob = predict(final_mod, train, type = "response")
train_y_class = rep('e', times = length(train_y_prob))
train_y_class[train_y_prob > 0.5] = "e"
train_y_class[train_y_prob <= 0.5] = "p"
train_error = mean(train_y_class != train$class)
train_error

y = dt$class
x = dt[,-1]
x_dummy = model.matrix( ~ ., x)
x_dummy = x_dummy[,2:dim(x_dummy)[2]]
set.seed(1)
train_idx = sample(n, size = floor(0.8*n), replace = FALSE)
train_x_dummy = x_dummy[train_idx, ]
train_y = y[train_idx]
test_x_dummy = x_dummy[-train_idx, ]
test_y = y[-train_idx]
glmnet_mod1 = cv.glmnet(x = train_x_dummy, y = train_y,
                nfolds = 5, family="binomial", type.measure = "class")
plot(glmnet_mod1)
glmnet_mod1$lambda.min
final_mod = glmnet(x = train_x_dummy, y = train_y, family="binomial",
                lambda = glmnet_mod1$lambda.min)
final_mod
final_mod$beta
train_y_pred = predict(final_mod, train_x_dummy, type = "class")
train_error = mean(train_y_pred[,1] != train_y)
train_error
test_y_pred = predict(final_mod, test_x_dummy, type = "class")
```

```
test_error = mean(test_y_pred[,1] != test_y)
test_error
accuracy=1-test_error
accuracy

#LDA

library(MASS)
library(caret)
set.seed(1)
mushroom<-read.csv("E:/ColumbiaUniversity/GR5291/mushrooms.csv",header=T
)
sample.index<-
sample(1:nrow(mushroom),round(nrow(mushroom)*0.8),replace=F)
train <- mushroom[sample.index,]
test <- mushroom[-sample.index,]
model <- lda(class ~ odor, data=train)
pred.class <- predict(model,newdata=test)$class
mean(pred.class != test$class)
table(test$class, pred.class)
model <- lda(class ~ habitat+odor+spore.print.color, data=train)
pred.class <- predict(model,newdata=test)$class
mean(pred.class != test$class)
table(test$class, pred.class)
model<-lda(class~habitat+spore.print.color+population+ring.type+gill.color+gill.si
ze+bruises+cap.color, data=train)
pred.class <- predict(model,newdata=test)$class
mean(pred.class != test$class)
table(test$class, pred.class)


#EDA

library(ggplot2)
library(dplyr)
setwd("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR 5291/PROJECT")
data <- read.csv("mushrooms.csv", header = T)
dim(data)
sum(data$stalk.root == "?") #2480-> Remove variable
```

```
eda0 <- c()
for (i in 1:ncol(data)){
  eda0[i] <- sum(as.numeric(unique(data[,i])))
}
unique(data$veil.type)
eda0
colnames(data)[eda0 == 1] #veil.type only one level - > Remove variable

data <- data[,-c(12,17)]


sum(data$class == "e") # 4208 edible
8124 - 4208  # 3916 poisonous
eda1 <- ggplot(data, aes(class, fill = class))
eda1 + geom_bar() + scale_fill_manual(breaks = c('Edible','Poisonous'),
values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR 5291/PROJECT/3 -
Count.pdf")

#Chi-square Test for correlationship
#H0: a is not associated with b
  #y ~ X's
pvalue <- data.frame(NA)
for (i in 1:21) {
  for (j in 1:21){
          eda2 <- table(data[,i], data[,j])
          pvalue[i,j] <- chisq.test(eda2)$p.value
  }
  }
sum(pvalue > 0.05) #All False....

#correlation graph for two variable each
  #First Method
data %>%
  count(class, odor) %>%
ggplot(mapping = aes(x = class, y = odor)) + geom_tile(mapping = aes(fill = n)) +
  labs(x = "Class", y = "odor")
  #Second Method
ggplot(data = data) +
```

```
  geom_count(mapping = aes(x = data$class, y = data$cap.shape)) +
  labs(x = "Class", y = "cap.shape")
  #Third Method
ggplot(data, aes(x = odor, y = spore.print.color, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))

 #Using the Third Method to create PDF plots
par(ask = FALSE)
ggplot(data, aes(x = odor, y = spore.print.color, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/odor&SporePrintColor.pdf")

ggplot(data, aes(x = odor, y = stalk.color.below.ring, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/odor&StalkColorBelowRing.pdf")

ggplot(data, aes(x = odor, y = habitat, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/odor&habitat.pdf")

ggplot(data, aes(x = odor, y = cap.surface, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/odor&CapSurface.pdf")

ggplot(data, aes(x = spore.print.color, y = stalk.color.below.ring, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/SporePrintColor&StalkColorBelowRing.pdf")
```

```
ggplot(data, aes(x = odor, y = class, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/odor&class.pdf")

ggplot(data, aes(x = spore.print.color, y = class, color = class)) +
  geom_jitter(alpha = 0.3) +
  scale_color_manual(breaks = c('Edible','Poisonous'), values=c('darkgreen','red'))
ggsave("C:/Users/Qisheng/Desktop/Study/GRAD 2017FA/GR
5291/PROJECT/sporePrintColorr&class.pdf")


#Reapeat for all combination between Y and X's
#ColNames <- colnames(data)
#par(mfrow = c(4,6))
#for (i in 1:22){
#  data %>%
#        count(ColNames[1], ColNames[i+1]) %>%
#  ggplot(mapping = aes(x = ColNames[1], y = ColNames[i+1])) +
geom_tile(mapping = aes(fill = n))
#}

  #Method 1
ColNames <- colnames(data)
par(ask = T)
out <- NULL
for (i in 1:22){
  eda3 <- ggplot(data = data) +
          geom_count(mapping = aes_string(x = data[, 1], y = data[, i+1])) +
    labs(x = ColNames[1], y = ColNames[i+1])
  print(eda3)
  out[[i]] <- eda3
}
  #Method 2
ColNames <- colnames(data)
for (i in 1:ncol(data)){
  for (j in 1:ncol(data)){
```

```
        eda4 <- ggplot(data, aes(x = data[, i + 1], y = data[, j + 1], color = data[, 1]))
+
        geom_jitter(alpha = 0.3) +
        scale_color_manual(breaks = c('Edible','Poisonous'),
values=c('darkgreen','red')) +
        labs(x = ColNames[i + 1], y = ColNames[j + 1])
        print(eda4)
  }
}

# Odds Ratio
eda5 <- table(data$class, data$cap.shape)[1:2,3:4]
(eda5[1,1]/eda5[1,2])/(eda5[2,1]/eda5[2,2])


# Check all plots of decision trees
datatest <- data[,c(1,6,13,19,21,3)]
par(ask = TRUE)
ColNames <- colnames(datatest)
for (i in 2:ncol(datatest)){
  for (j in 2:ncol(datatest)){
        eda4 <- ggplot(datatest, aes(x = datatest[, i], y = datatest[, j], color =
datatest[, 1])) +
        geom_jitter(alpha = 0.3) +
        scale_color_manual(breaks = c('Edible','Poisonous'),
values=c('darkgreen','red')) +
        labs(x = ColNames[i], y = ColNames[j])
        print(eda4)
  }
}
```