PSTAT131 Final Project

Russell (Senyuan) Liu

2022/11/10

Introduction

You're lost in the forest and the last thing you ate was a pinecone a few hours ago. You come across a mushroom, but you can't tell if it's poisonous or not. Luckily, you have a laptop with a dataset on mushrooms. With only 25% battery left, you enter the descriptions of the mushroom in question as quickly as you can. Do you risk getting poisoned and eat the mushroom… or not and starve?
It is very important for us to know which features in the mushroom feature data set will lead to death and which features are suitable for eating. This paper will establish an appropriate model to predict whether mushrooms are poisonous according to their characteristics.

Loading Data and Packages

The data for this project comes from the kaggle competition platform see https://www.kaggle.com/datasets/uciml/mushroom-classification. The sample size is large, with a total of 8124 pieces of data, and there are 22 independent variables. It includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). 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 variables for the data set are as following:
1.cap_shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2.cap_surface: fibrous=f,grooves=g,scaly=y,smooth=s
3.cap_color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
3.bruises: bruises=t,no=f
5.odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6.gill_attachment: attached=a,descending=d,free=f,notched=n
7.gill_spacing: close=c,crowded=w,distant=d
8.gill_size: broad=b,narrow=n
9.gill_color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10.stalk_shape: enlarging=e,tapering=t
11.stalk_root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12.stalk_surface_above_ring: fibrous=f,scaly=y,silky=k,smooth=s
13.stalk_surface_below_ring: fibrous=f,scaly=y,silky=k,smooth=s
14.stalk_color_above_ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15.stalk_color_below_ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16.veil_type: partial=p,universal=u
17.veil_color: brown=n,orange=o,white=w,yellow=y
18.ring_number: none=n,one=o,two=t
19.ring_type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20.spore_print_color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21.population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22.habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

```r
library(tidyverse)
library(tidymodels)
library(corrplot)
library(janitor)
library(skimr)
library(patchwork)
library(lubridate)
library(ranger)
library(rlang)
library(ggplot2)
library(corrr)
library(klaR)
library(MASS)
library(discrim)
library(installr)
library(kernlab)
library(kknn)
library(vip)
library(conflicted)
conflict_prefer("contr.dummy", "kknn")
tidymodels_prefer()
mushroom <- read_csv("mushrooms.csv")
```

Data Cleaning

Clean names

Convert all "-" of variable name to "_".

```r
colnames(mushroom)
```

```
##  [1] "class"                    "cap-shape"
##  [3] "cap-surface"              "cap-color"
##  [5] "bruises"                  "odor"
##  [7] "gill-attachment"          "gill-spacing"
##  [9] "gill-size"                "gill-color"
## [11] "stalk-shape"              "stalk-root"
## [13] "stalk-surface-above-ring" "stalk-surface-below-ring"
## [15] "stalk-color-above-ring"   "stalk-color-below-ring"
## [17] "veil-type"                "veil-color"
## [19] "ring-number"              "ring-type"
## [21] "spore-print-color"        "population"
## [23] "habitat"
```

```r
mushroom <- mushroom %>%
  clean_names()
```

Missing value checking and filling

We looked at the first 6 lines of the data and looked at each column for the possible value, and we found that data has no NA value, but variable stalk_root has '?' value, as described in the data set, '?' represents a missing stalk root value, so we reassign him a value of 'm'.
In addition, veil_type has only one value 'p'. We have reason to think that this variable has no effect on identifying whether mushrooms are poisonous, so delete.

```
head(mushroom)
```

```
## # A tibble: 6 x 23
##   class cap_shape cap_su~1 cap_c~2 bruises odor  gill_~3 gill_~4 gill_~5 gill_~6
##   <chr> <chr>     <chr>    <chr>   <lgl>   <chr> <chr>   <chr>   <chr>   <chr>
## 1 p     x         s        n       TRUE    p     f       c       n       k
## 2 e     x         s        y       TRUE    a     f       c       b       k
## 3 e     b         s        w       TRUE    l     f       c       b       n
## 4 p     x         y        w       TRUE    p     f       c       n       n
## 5 e     x         s        g       FALSE   n     f       w       b       k
## 6 e     x         y        y       TRUE    a     f       c       b       n
## # ... with 13 more variables: stalk_shape <chr>, stalk_root <chr>,
## #   stalk_surface_above_ring <chr>, stalk_surface_below_ring <chr>,
## #   stalk_color_above_ring <chr>, stalk_color_below_ring <chr>,
## #   veil_type <chr>, veil_color <chr>, ring_number <chr>, ring_type <chr>,
## #   spore_print_color <chr>, population <chr>, habitat <chr>, and abbreviated
## #   variable names 1: cap_surface, 2: cap_color, 3: gill_attachment,
## #   4: gill_spacing, 5: gill_size, 6: gill_color
```

```
lapply(mushroom,unique)
```

```
## $class
## [1] "p" "e"
##
## $cap_shape
## [1] "x" "b" "s" "f" "k" "c"
##
## $cap_surface
## [1] "s" "y" "f" "g"
##
## $cap_color
##  [1] "n" "y" "w" "g" "e" "p" "b" "u" "c" "r"
##
## $bruises
## [1]  TRUE FALSE
##
## $odor
## [1] "p" "a" "l" "n" "f" "c" "y" "s" "m"
##
## $gill_attachment
## [1] "f" "a"
##
## $gill_spacing
## [1] "c" "w"
##
## $gill_size
## [1] "n" "b"
##
## $gill_color
##  [1] "k" "n" "g" "p" "w" "h" "u" "e" "b" "r" "y" "o"
##
## $stalk_shape
## [1] "e" "t"
##
```

```
## $stalk_root
## [1] "e" "c" "b" "r" "?"
##
## $stalk_surface_above_ring
## [1] "s" "f" "k" "y"
##
## $stalk_surface_below_ring
## [1] "s" "f" "y" "k"
##
## $stalk_color_above_ring
## [1] "w" "g" "p" "n" "b" "e" "o" "c" "y"
##
## $stalk_color_below_ring
## [1] "w" "p" "g" "b" "n" "e" "y" "o" "c"
##
## $veil_type
## [1] "p"
##
## $veil_color
## [1] "w" "n" "o" "y"
##
## $ring_number
## [1] "o" "t" "n"
##
## $ring_type
## [1] "p" "e" "l" "f" "n"
##
## $spore_print_color
## [1] "k" "n" "u" "h" "w" "r" "o" "y" "b"
##
## $population
## [1] "s" "n" "a" "v" "y" "c"
##
## $habitat
## [1] "u" "g" "m" "d" "p" "w" "l"
```

```r
#checking missing values
mushroom %>% sapply(function(x) {sum(is.na(x))})
```

```
##                    class                cap_shape              cap_surface
##                        0                        0                        0
##                cap_color                  bruises                     odor
##                        0                        0                        0
##          gill_attachment             gill_spacing                gill_size
##                        0                        0                        0
##               gill_color              stalk_shape               stalk_root
##                        0                        0                        0
## stalk_surface_above_ring stalk_surface_below_ring   stalk_color_above_ring
##                        0                        0                        0
##   stalk_color_below_ring                veil_type               veil_color
##                        0                        0                        0
##              ring_number                ring_type        spore_print_color
##                        0                        0                        0
##               population                  habitat
##                        0                        0
```

```
mushroom$stalk_root[mushroom$stalk_root=='?'] <- 'm'
mushroom<-mushroom %>%
  select(-veil_type)
```

Convert to factor

Because the categorical variables of the data are stored in character type, we need to convert the categorical variables into factor type.

```
mushroom1<-mushroom
#colnames(mushroom1)
mushroom1<-mushroom1 %>%
  mutate(class=                    factor(class),
         cap_shape=                factor(cap_shape),
         cap_surface=              factor(cap_surface),
         cap_color=                factor(cap_color),
         bruises=                  factor(bruises),
         odor=                     factor(odor),
         gill_attachment=          factor(gill_attachment),
         gill_spacing=             factor(gill_spacing),
         gill_size=                factor(gill_size),
         gill_color=               factor(gill_color),
         stalk_shape=              factor(stalk_shape),
         stalk_root=               factor(stalk_root),
         stalk_surface_above_ring= factor(stalk_surface_above_ring),
         stalk_surface_below_ring= factor(stalk_surface_below_ring),
         stalk_color_above_ring=   factor(stalk_color_above_ring),
         stalk_color_below_ring=   factor(stalk_color_below_ring),
         veil_color=               factor(veil_color),
         ring_number=              factor(ring_number),
         ring_type=                factor(ring_type),
         spore_print_color=        factor(spore_print_color),
         population=               factor(population),
         habitat=                  factor(habitat),
         )
```

Convert to numeric

We also convert each variable into numerical type and prepare a numerical data for calculating the correlation coefficient between variables.

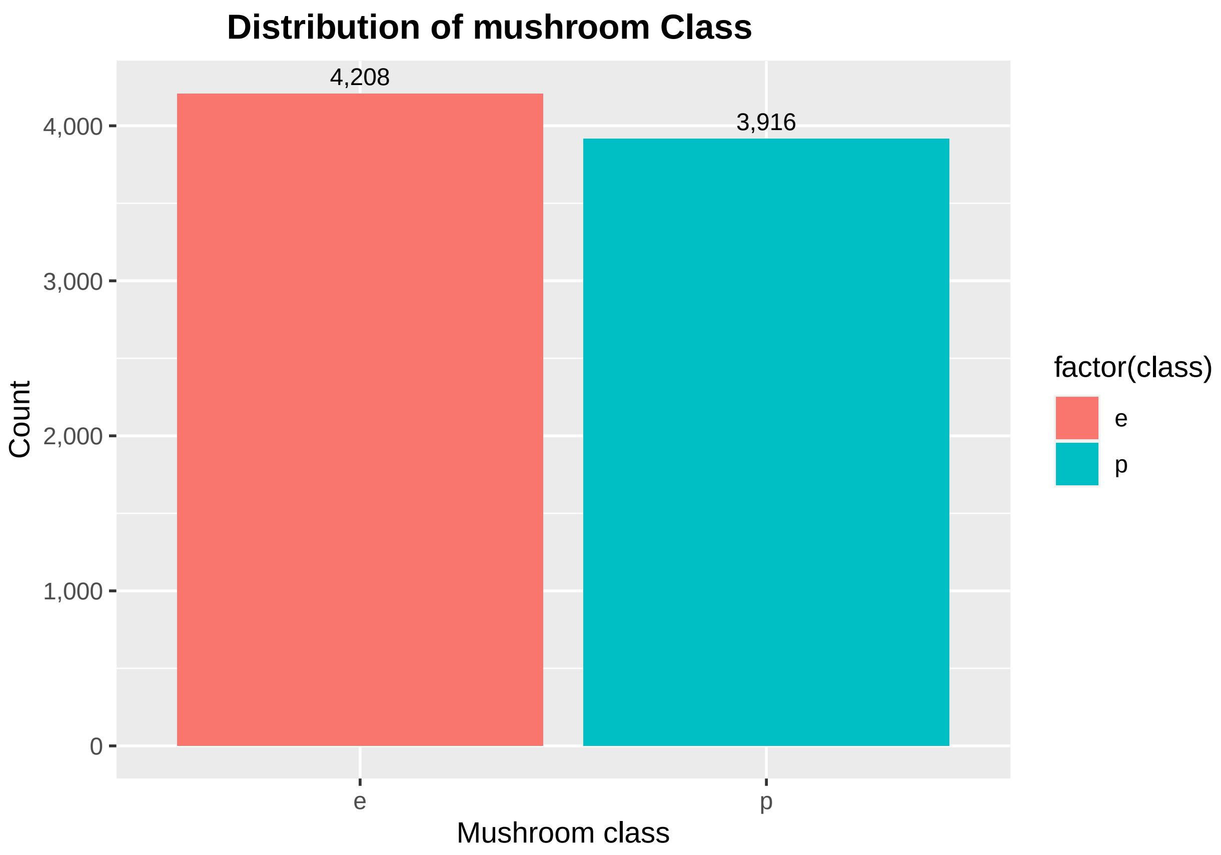
```
mushroom2<-mushroom1
mushroom2<-mushroom2 %>%
  mutate(class=                    as.numeric(class),
         cap_shape=                as.numeric(cap_shape),
         cap_surface=              as.numeric(cap_surface),
         cap_color=                as.numeric(cap_color),
         bruises=                  as.numeric(bruises),
         odor=                     as.numeric(odor),
         gill_attachment=          as.numeric(gill_attachment),
         gill_spacing=             as.numeric(gill_spacing),
         gill_size=                as.numeric(gill_size),
         gill_color=               as.numeric(gill_color),
         stalk_shape=              as.numeric(stalk_shape),
         stalk_root=               as.numeric(stalk_root),
         stalk_surface_above_ring= as.numeric(stalk_surface_above_ring),
         stalk_surface_below_ring= as.numeric(stalk_surface_below_ring),
         stalk_color_above_ring=   as.numeric(stalk_color_above_ring),
         stalk_color_below_ring=   as.numeric(stalk_color_below_ring),
         veil_color=               as.numeric(veil_color),
         ring_number=              as.numeric(ring_number),
         ring_type=                as.numeric(ring_type),
         spore_print_color=        as.numeric(spore_print_color),
         population=               as.numeric(population),
         habitat=                  as.numeric(habitat),
         )
#View(mushroom2)
```

Exploratory Data Analysis

Distribution of mushroom class

Our target variable is the category of mushrooms. From the distribution map, we can see that the number of edible mushrooms in the sample is 4208, and the number of poisonous mushrooms is 3016. The distribution of the two is basically balanced.

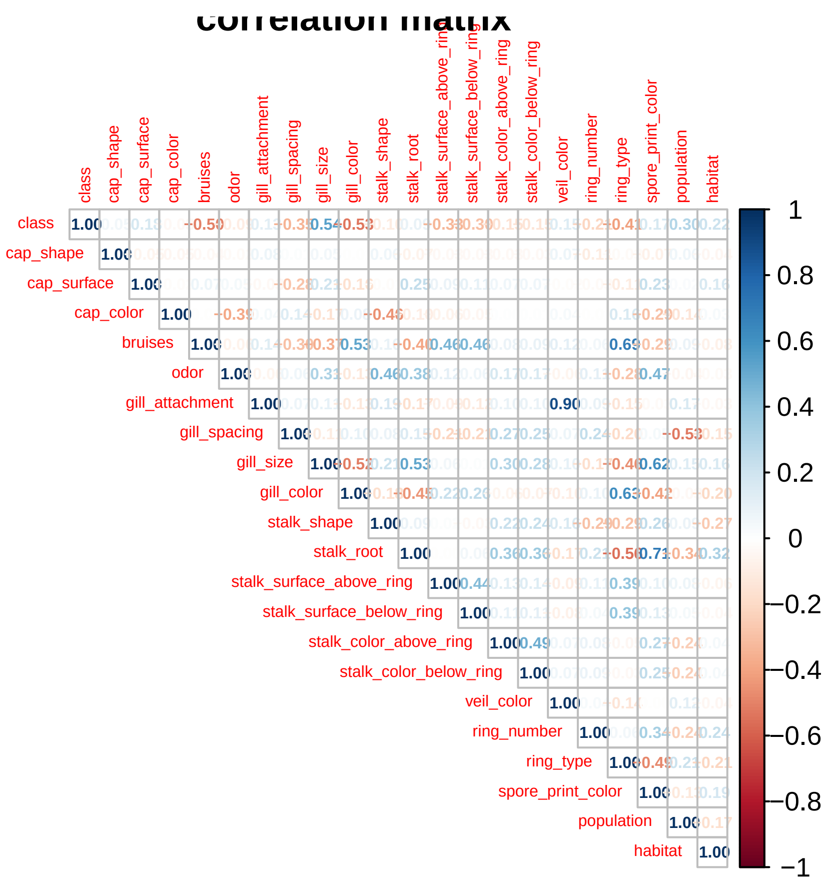
```
theme <- theme(plot.title = element_text(hjust = 0.3, face = "bold"))

mushroom%>%
  ggplot(aes(x = factor(class),
             y = stat(count), fill = factor(class),
             label = scales::comma(stat(count)))) +
  geom_bar(position = "dodge") +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_y_continuous(labels = scales::comma)+
  labs(x = 'Mushroom class', y = 'Count') +
  ggtitle("Distribution of mushroom Class") +
  theme
```



correlation matrix

The mushroom class has the largest positive correlation with the variable gill_size, the correlation coefficient is 0.54. And it has the largest negative correlation with the variable gill_color, the correlation coefficient is -0.53. In addition, variables gill_attachment and veil_colour has a large positive correlation, and the correlation coefficient is 0.9. Various independent variables have different degrees of correlation.

```
correlation <- cor(mushroom2)
corrplot(correlation,tl.cex = 0.5, number.cex = 0.5, method = "number", type = "upper",title ='correlati
```

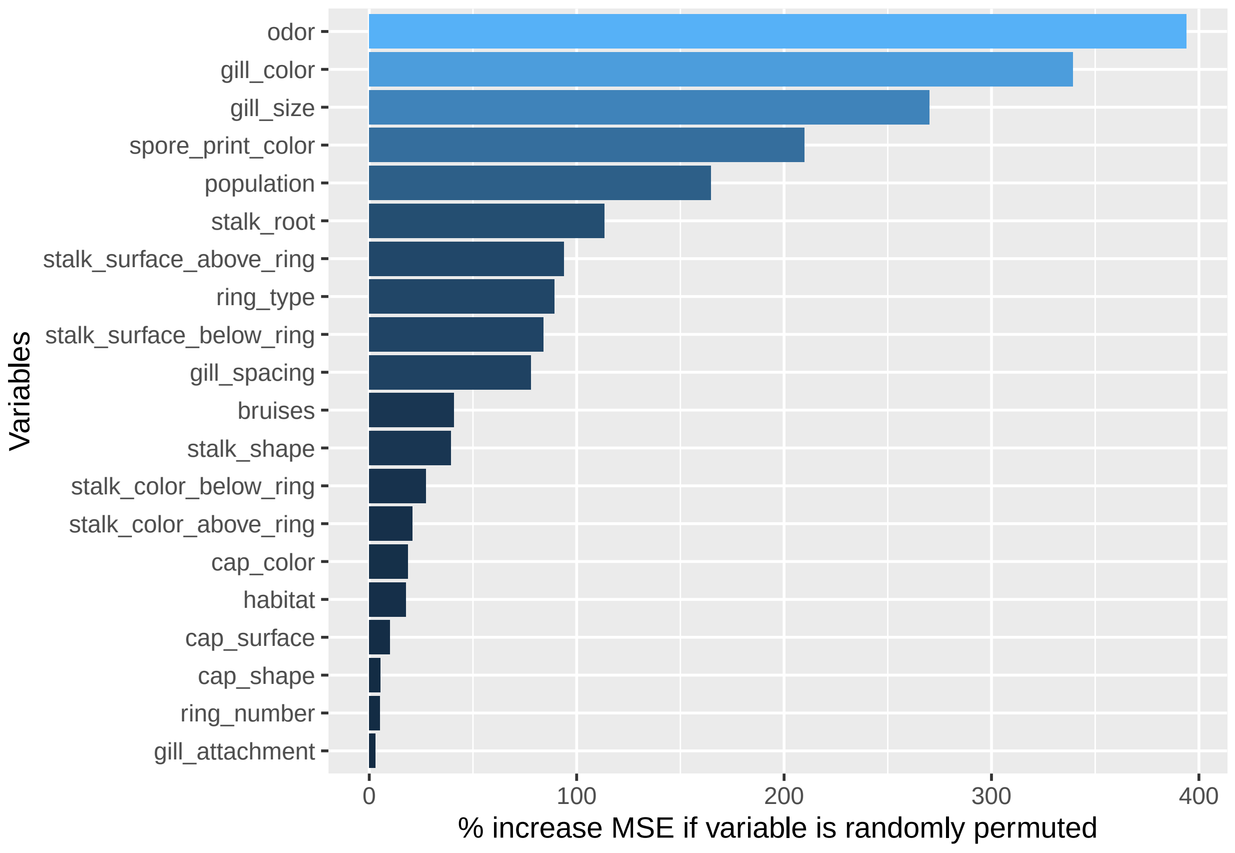


```
#sort(correlation[,1],decreasing=T)
```

Identification of important variables

We calculated the importance of each independent variable through the random forest model. The results showed that odor, gill colour, gill size, spring print color were important variables to predict whether mushrooms were poisonous, while cap shape, ring number, and gill attachment were the three least important variables.

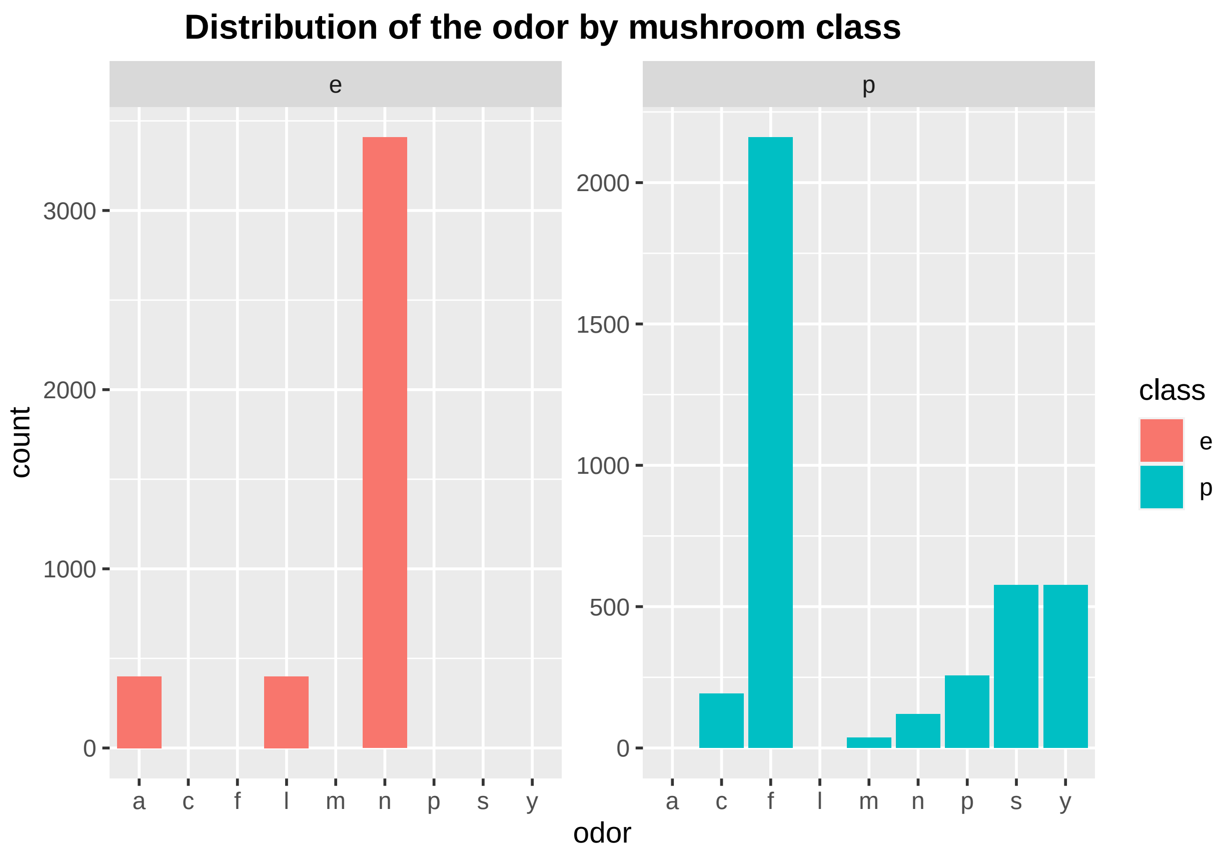
```
library("randomForest")
set.seed(123)
m <- data.frame(mushroom2[, -1])
quick_RF <- randomForest(x = m, y = mushroom2$class, ntree=20,importance=F)
imp_RF <- importance(quick_RF)
imp_DF <- data.frame(Variables = row.names(imp_RF), MSE = imp_RF[,1])
imp_DF <- imp_DF[order(imp_DF$MSE, decreasing = TRUE),]

ggplot(imp_DF[1:20,], aes(x=reorder(Variables, MSE), y=MSE, fill=MSE)) +
  geom_bar(stat = 'identity') +
  labs(x = 'Variables', y= '% increase MSE if variable is randomly permuted') +
  coord_flip() +
  theme(legend.position="none")
```



Distribution of the odor by mushroom class

Most edible mushrooms have no odor(odor=n), while most poisonous mushrooms are foul(odor=f).

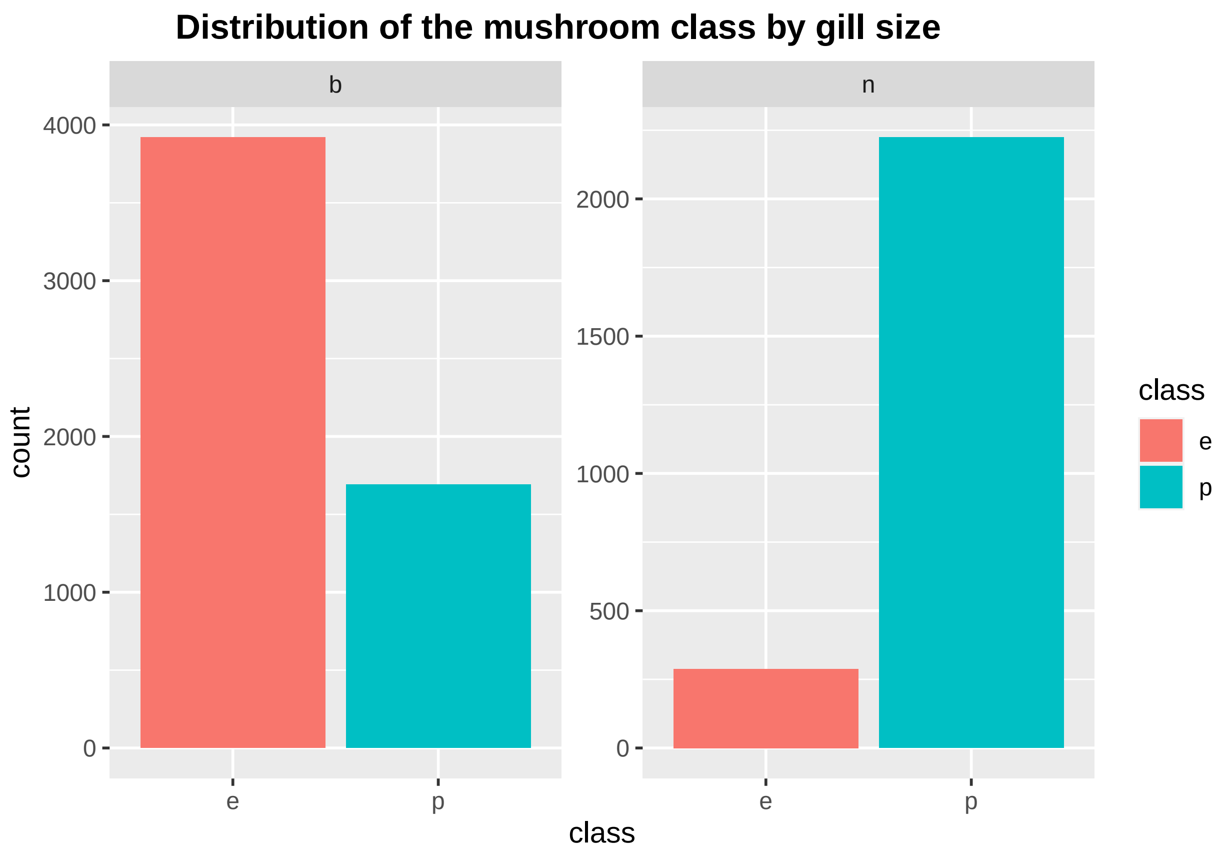
```
mushroom1 %>%
  ggplot(aes(x = odor,
           y = stat(count), fill = class)) +
  geom_bar( ) +
  facet_wrap(~class, scales = "free_y") +
  labs(
    title = "Distribution of the odor by mushroom class  "
  )+
  theme
```



Distribution of the mushroom class by gill size

If the mushroom's gill size is Narrow, the probability of the mushroom being a poisonous mushroom is high, while if the mushroom's gill size is broad, the probability of the mushroom being an edible mushroom is higher.

```
mushroom1 %>%
  ggplot(aes(x = class,
             y = stat(count), fill = class)) +
  geom_bar( ) +
  facet_wrap(~gill_size, scales = "free_y") +
  labs(
    title = "Distribution of the mushroom class by gill size"
  )+
  theme
```



Model Building

Data Split

The dataset contains a total of 8124 samples,of which 80% of the data is used as training data about 6498 and 20% of the data is used as test data about 1626. Stratified sampling was used based on mushroom class.

```
set.seed(123)
dim(mushroom1) #8124   22
```

```
## [1] 8124   22
```

```
mushroom_split <- initial_split(mushroom1,prop = 0.8, strata = "class")
train <- training(mushroom_split)
test <- testing(mushroom_split)
dim(train) # 6498   22
```

```
## [1] 6498   22
```

```
dim(test)  # 1626   22
```

```
## [1] 1626   22
```

Building the Recipe and Tweaking The Data

When establishing recipe, we will remove the four least important variables previously calculated according to the random forest, and code all classification prediction factors. At the same time, we used the 5 fold cross validation repeated three times.

```
train_folds <- vfold_cv(train, v = 5,repeats = 3)

recipe<-mushroom1 %>%
  recipe(class ~ cap_color+spore_print_color + odor + gill_size +
  stalk_root + population + gill_color + habitat + gill_spacing +
  stalk_shape + ring_type + stalk_surface_below_ring + bruises +
  stalk_color_below_ring + stalk_surface_above_ring  +
  stalk_color_above_ring + veil_color ) %>%
 #step_dummy(cap_shape)%>%
 #step_dummy(cap_surface)%>%
  step_dummy(cap_color)%>%
  step_dummy(bruises)%>%
  step_dummy(odor)%>%
 #step_dummy(gill_attachment)%>%
  step_dummy(gill_spacing)%>%
  step_dummy(gill_size)%>%
  step_dummy(gill_color)%>%
  step_dummy(stalk_shape)%>%
  step_dummy(stalk_root)%>%
  step_dummy(stalk_surface_above_ring)%>%
  step_dummy(stalk_surface_below_ring)%>%
  step_dummy(stalk_color_above_ring)%>%
  step_dummy(stalk_color_below_ring)%>%
  step_dummy(veil_color)%>%
 #step_dummy(ring_number)%>%
  step_dummy(ring_type)%>%
  step_dummy(spore_print_color)%>% #
  step_dummy(population)%>%
  step_dummy(habitat)

recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         17
##
## Operations:
##
## Dummy variables from cap_color
## Dummy variables from bruises
## Dummy variables from odor
## Dummy variables from gill_spacing
## Dummy variables from gill_size
## Dummy variables from gill_color
## Dummy variables from stalk_shape
## Dummy variables from stalk_root
## Dummy variables from stalk_surface_above_ring
## Dummy variables from stalk_surface_below_ring
## Dummy variables from stalk_color_above_ring
## Dummy variables from stalk_color_below_ring
## Dummy variables from veil_color
## Dummy variables from ring_type
## Dummy variables from spore_print_color
## Dummy variables from population
## Dummy variables from habitat
```

Model select

I decided to run cross fold validation on the following four models.
1.Boosted Tree Model
2.SVM model
3.Nearest Neighbors model
4.Random Forest model
5.Logistic Regression model

Boosted Tree Model I tuned learning rate, set mode to "classification" (because my outcome is a factor variable), and used the xgboost engine.

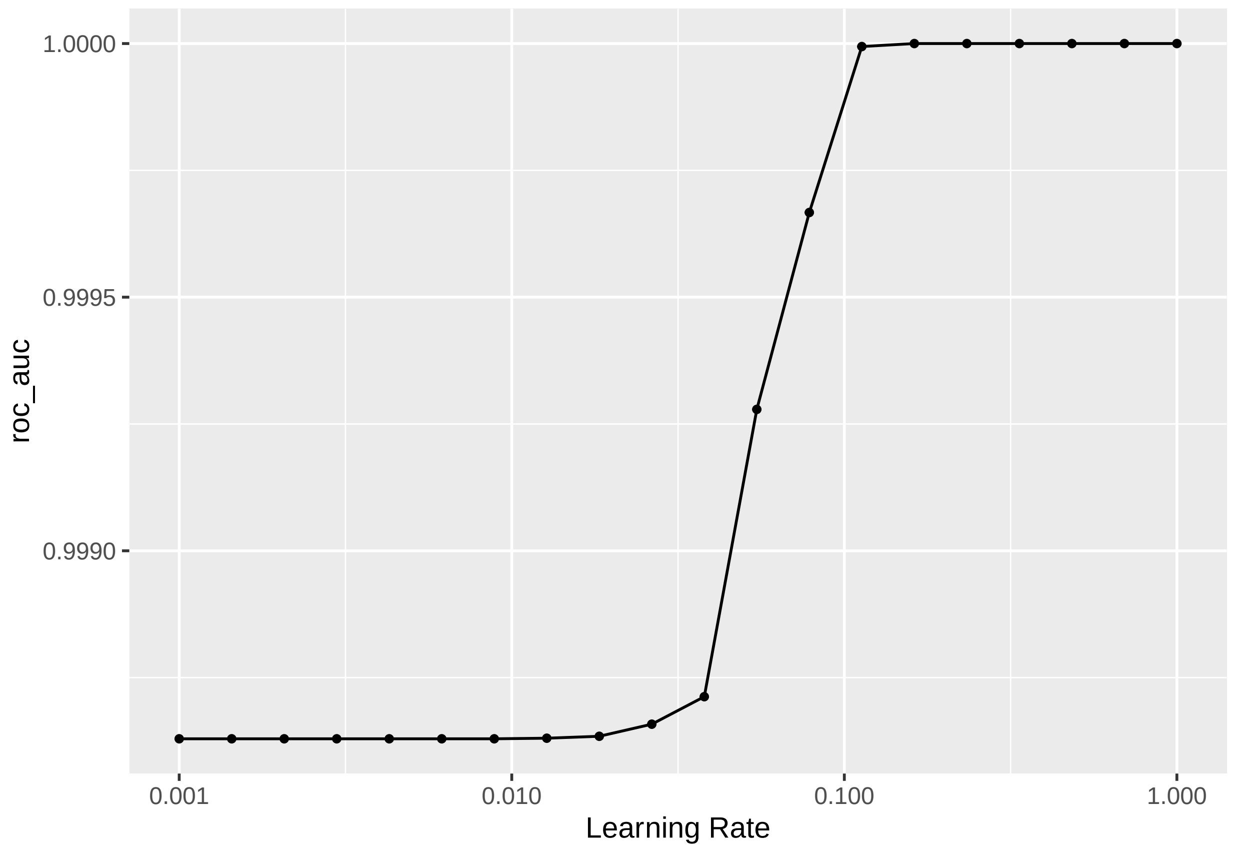
```
bt_model <- boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("classification") %>%
  set_args(learn_rate = tune())

bt_workflow <- workflow() %>%
  add_model(bt_model) %>%
  add_recipe(recipe)

# define grid
bt_grid <- tibble(learn_rate= 10^seq(-3, 0, length.out = 20))
```

Then, I executed my model by tuning learning rate and save model result. This process took 8 minutes, it is quickly.

```
set.seed(123)
bt_res <- bt_workflow %>%
  tune_grid(resamples = train_folds,
            grid = bt_grid)
save(bt_res, bt_workflow, file = "/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/bt_res3.rda")
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric. As the learning rate increases, the auc value increases rapidly and then remains unchanged, its mean auc value is close to 100% when learning rate 0.1623777

```
set.seed(123)
load("/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/bt_res3.rda")
autoplot(bt_res, metric = "roc_auc")
```



```
show_best(bt_res, metric = "roc_auc") %>% select(-.estimator, -.config) #0.1623777
```

```
## # A tibble: 5 x 5
##   learn_rate .metric  mean     n std_err
##        <dbl> <chr>   <dbl> <int>   <dbl>
## 1      0.162 roc_auc     1    15       0
## 2      0.234 roc_auc     1    15       0
## 3      0.336 roc_auc     1    15       0
## 4      0.483 roc_auc     1    15       0
## 5      0.695 roc_auc     1    15       0
```

We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the boosted model using the select_best() function. Finally, the optimal model is used to fit the training set.

```
set.seed(123)
bt_workflow_tuned <- bt_workflow %>%
  finalize_workflow(select_best(bt_res, metric = "roc_auc"))
bt_final <- fit(bt_workflow_tuned, train)
```

SVM Model I tuned cost, set mode to "classification", and used the kernlab engine.

```
svm_model <- svm_rbf( cost  = tune()
                    ) %>%
  set_engine("kernlab")%>%
  set_mode('classification')%>%
  translate()
```

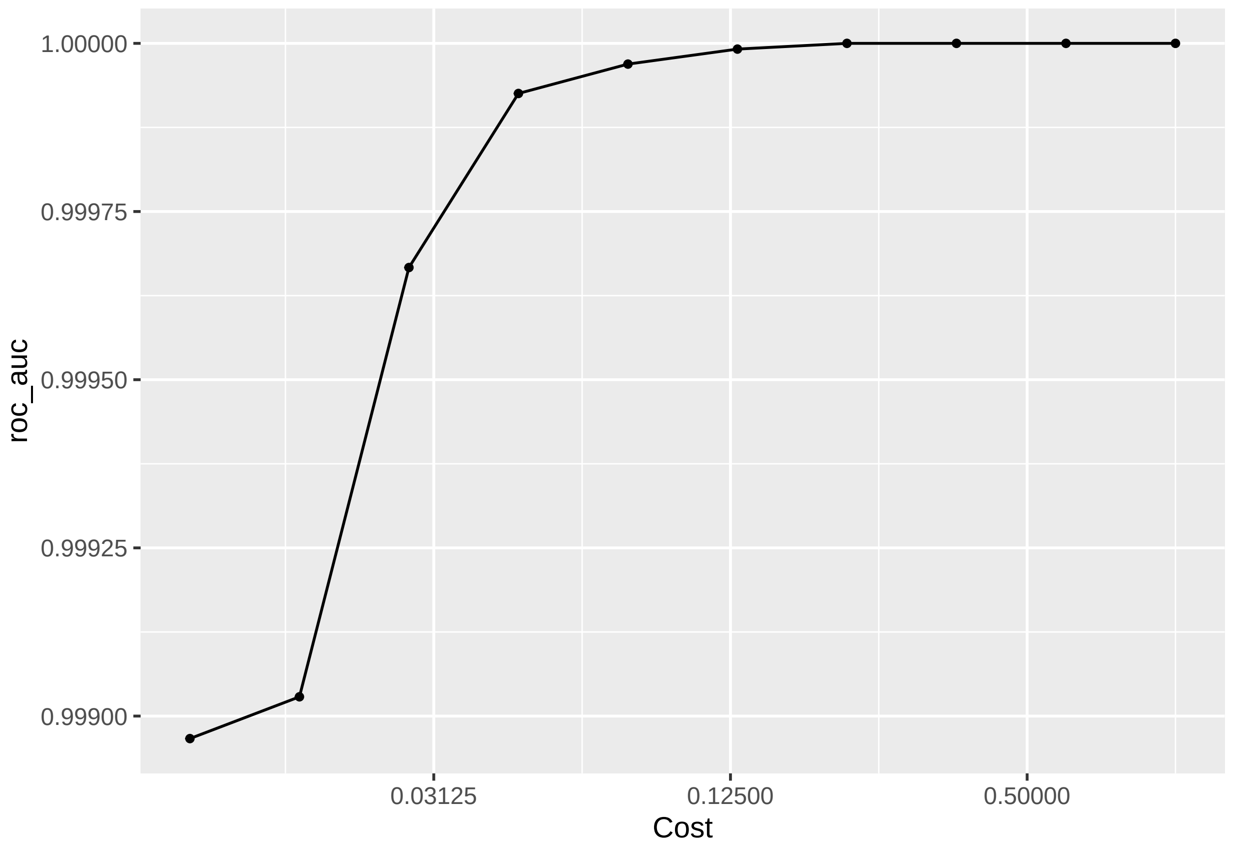
```
svm_workflow <- workflow() %>%
  add_model(svm_model) %>%
  add_recipe(recipe)

svm_grid <- tibble(cost = 10^seq(-2, 0, length.out = 10))
```

Then, I executed my model by tuning cost and save model result. This process takes a little long, about 20 minutes.

```
set.seed(123)
svm_res <- svm_workflow %>%
  tune_grid(resamples =train_folds,
            grid = svm_grid)
save(svm_res, svm_workflow, file = "/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/svm_res3.rd
```

With the increase of cost value, the auc value also increased rapidly, and remained stable when the auc value reached the peak of 100%. We found that when cost=0.2154435 , its mean roc_auc value is close to 100%.

```
load("/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/svm_res3.rda")
autoplot(svm_res, metric = "roc_auc")
```



```
show_best(svm_res, metric = "roc_auc") %>% select(-.estimator, -.config) #cost=0.2154435
```

```
## # A tibble: 5 x 5
##    cost .metric  mean     n    std_err
##   <dbl> <chr>   <dbl> <int>      <dbl>
## 1 0.215 roc_auc  1       15 0
## 2 0.359 roc_auc  1       15 0
## 3 0.599 roc_auc  1       15 0
## 4 1     roc_auc  1       15 0
## 5 0.129 roc_auc  1.00    15 0.00000322
```

Nearest Neighbors Model I tuned neighbors, set mode to "classification" (because my outcome is a factor variable), and used the kknn engine.

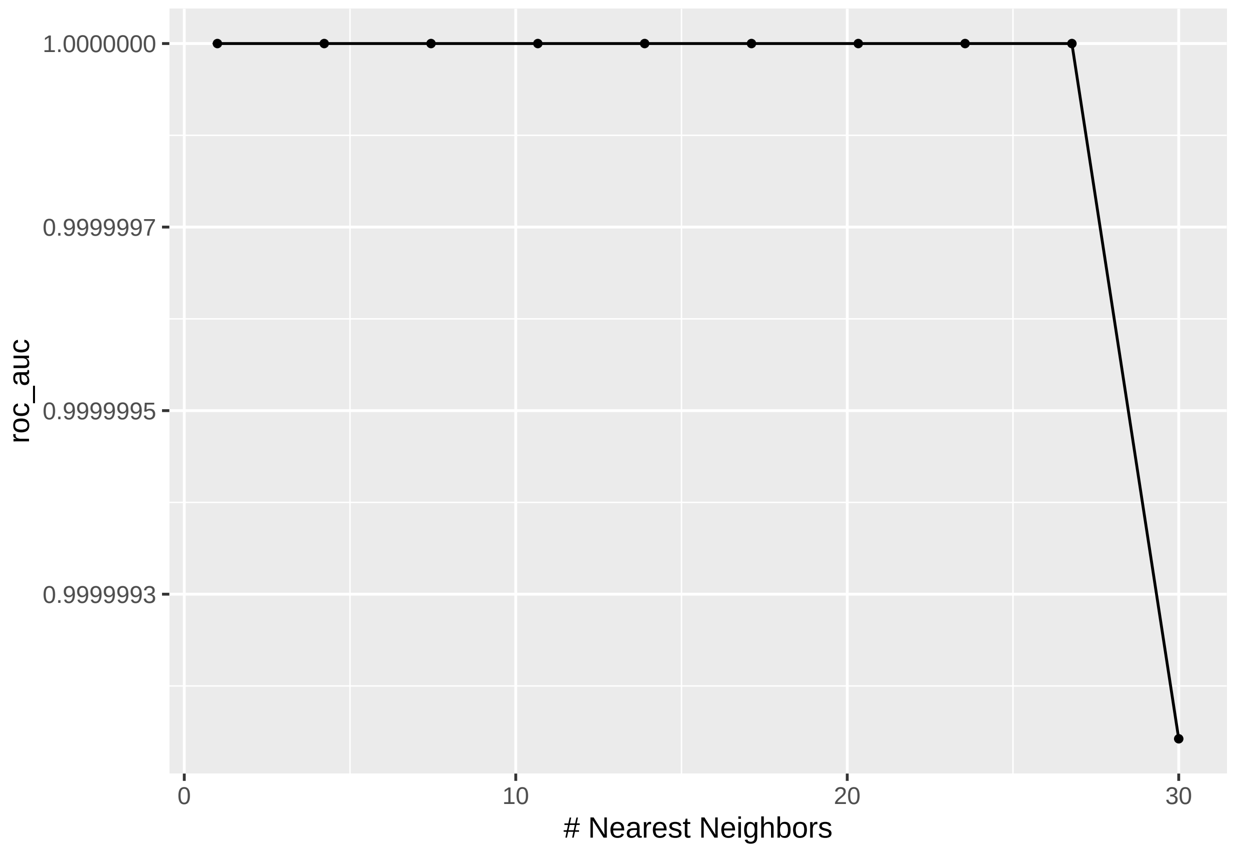
```
knn_model <- nearest_neighbor() %>%
  set_engine("kknn")%>%
  set_mode("classification") %>%
  set_args(neighbors = tune())


knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(recipe)

# define grid
knn_grid <- tibble(neighbors = seq(1, 30,length.out=10))
```

Then, I executed my model by tuning neighbors and save model result. This process took 20 minutes, it's a little slow.

```
set.seed(123)
knn_res <- knn_workflow %>%
  tune_grid(resamples = train_folds,
            grid = knn_grid
       )
save(knn_res, knn_workflow, file = "/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/knn_res3.rda
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric. when neighbors less than 27, its mean roc_auc value is close to 100%.

```
set.seed(123)
load("/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/knn_res3.rda")
autoplot(knn_res, metric = "roc_auc")
```



```
show_best(knn_res, metric = "roc_auc") %>% select(-.estimator, -.config)  #k=1
```

```
## # A tibble: 5 x 5
##   neighbors .metric  mean     n std_err
##       <dbl> <chr>   <dbl> <int>   <dbl>
## 1      1    roc_auc     1    15       0
## 2      4.22 roc_auc     1    15       0
## 3      7.44 roc_auc     1    15       0
## 4     10.7  roc_auc     1    15       0
## 5     13.9  roc_auc     1    15       0
```

Random Forest Model I tuned min_n, set mode to "classification" (because my outcome is a factor variable), and used the ranger engine.

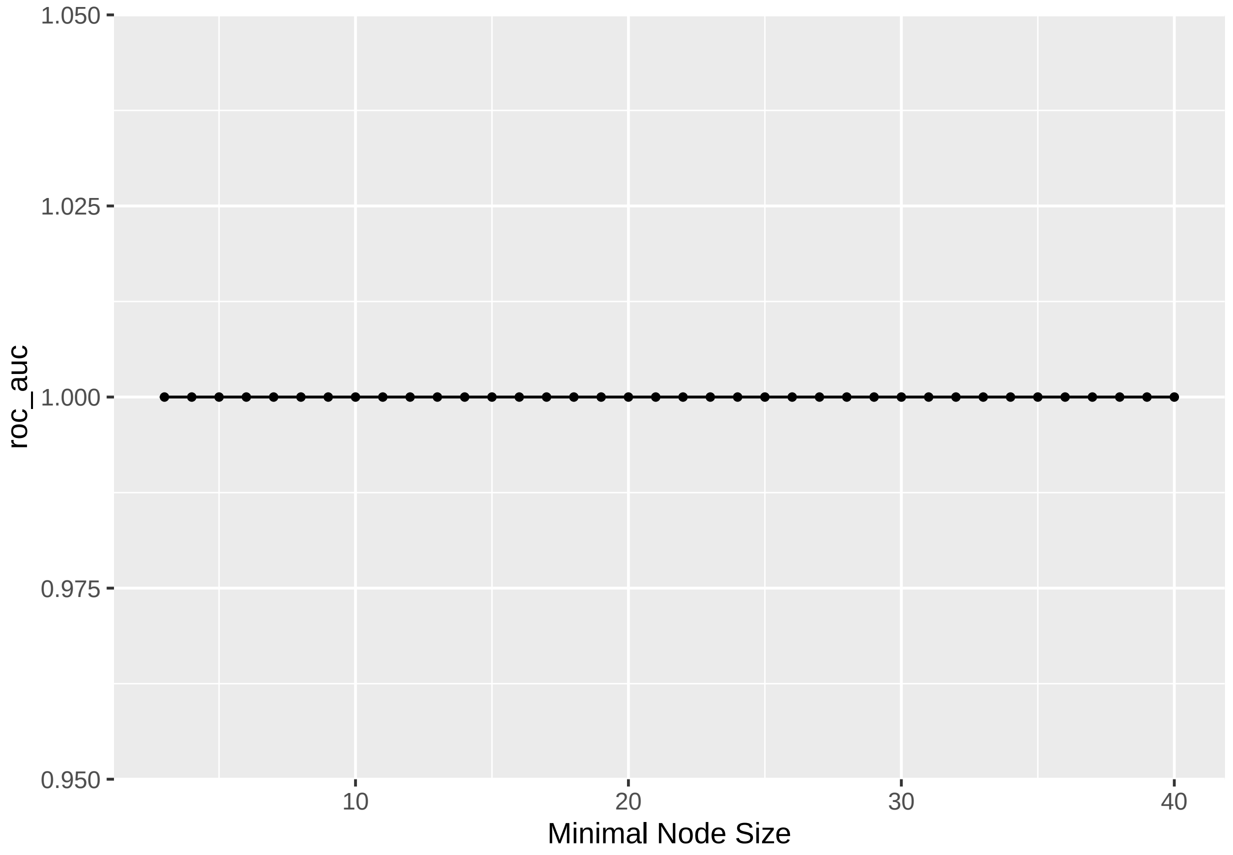
```
rf_model <- rand_forest() %>%
  set_mode("classification") %>%
  set_args(min_n = tune())%>%
  set_engine("ranger", importance = "impurity")


rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_recipe(recipe)

# define grid  data is small,so only tune min_n
rf_grid <- tibble(min_n = seq(3, 40) )
```

Then, I executed my model by tuning neighbors and save model result. This process took 15 minutes, it's a little slow.

```r
set.seed(123)
rf_res <- rf_workflow %>%
  tune_grid(resamples = train_folds,
            grid = rf_grid
           )
save(rf_res, rf_workflow, file = "/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/rf_res3.rda")
```

With the increase of the minimum tree, the auc value is always 100%. The model is very stable.

```r
set.seed(123)
load("/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/rf_res3.rda")
autoplot(rf_res, metric = "roc_auc")  ##
```



```r
show_best(rf_res, metric = "roc_auc") %>% select(-.estimator, -.config)
```

```
## # A tibble: 5 x 5
##   min_n .metric  mean     n std_err
##   <int> <chr>   <dbl> <int>   <dbl>
## 1     3 roc_auc     1    15       0
## 2     4 roc_auc     1    15       0
## 3     5 roc_auc     1    15       0
## 4     6 roc_auc     1    15       0
## 5     7 roc_auc     1    15       0
```

Logistic Regression Model I tuned penalty, set mode to "classification" (because my outcome is a factor variable), and used the LiblineaR engine.

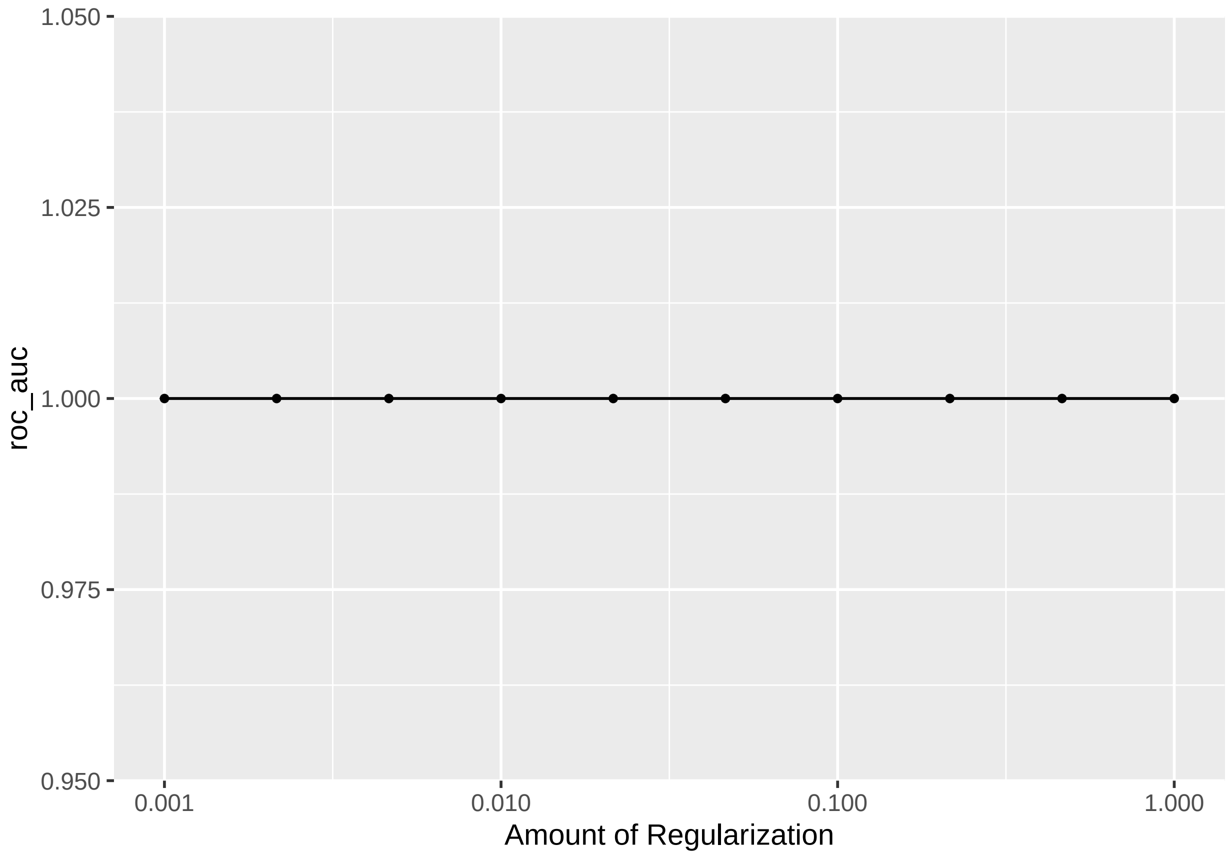
```r
log_model <- logistic_reg() %>%
  set_engine("LiblineaR")%>%
  set_mode('classification')%>%
  set_args(penalty = tune())

log_workflow <- workflow() %>%
  add_model(log_model) %>%
  add_recipe(recipe)

log_grid <- tibble(penalty = 10^seq(-3, 0, length.out = 10))
```

Then, I executed my model by tuning penalty and save model result. This process took 3 minutes, it's very quickly.

```r
library('LiblineaR')
log_res <- log_workflow %>%
  tune_grid(resamples = train_folds,
            grid = log_grid,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc))
save(log_res, log_workflow, file = "/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/log_res3.rda
```

Auc value does not change with the change of pennality, and it is always 100%. The model is very stable.

```r
load("/Users/liusenyuan/Desktop/PSTAT131/FinalProject_Russell/log_res3.rda")
autoplot(log_res, metric = "roc_auc")  ##0.001 best
```



```
show_best(log_res, metric = "roc_auc") %>% dplyr::select(-.estimator, -.config) #100% penalty=0.001
```

```
## # A tibble: 5 x 5
##   penalty .metric  mean     n std_err
##     <dbl> <chr>   <dbl> <int>   <dbl>
## 1 0.001   roc_auc     1    15       0
## 2 0.00215 roc_auc     1    15       0
## 3 0.00464 roc_auc     1    15       0
## 4 0.01    roc_auc     1    15       0
## 5 0.0215  roc_auc     1    15       0
```

Model compare By comparing the average auc value of the above five models after optimization and the duration of model optimization, we finally choose logical regression as the optimal model. Because the auc values of the five models all reach 100%, but the running speed of logical regression is the fastest.

```
mean_roc_auc <- c(1, 1, 1,1,1)
models <- c("Boosted tree model","SVM model","Nearest Neighbors model ","Random forest model","Logistic
time<-c('8min','20min','22min','25min','3min')
results <- tibble(mean_roc_auc = mean_roc_auc, models = models,time=time)

results
```

```
## # A tibble: 5 x 3
##   mean_roc_auc models                     time
##          <dbl> <chr>                      <chr>
## 1            1 "Boosted tree model"       8min
## 2            1 "SVM model"                20min
## 3            1 "Nearest Neighbors model " 22min
## 4            1 "Random forest model"      25min
## 5            1 "Logistic regression model" 3min
```

Final Model Building

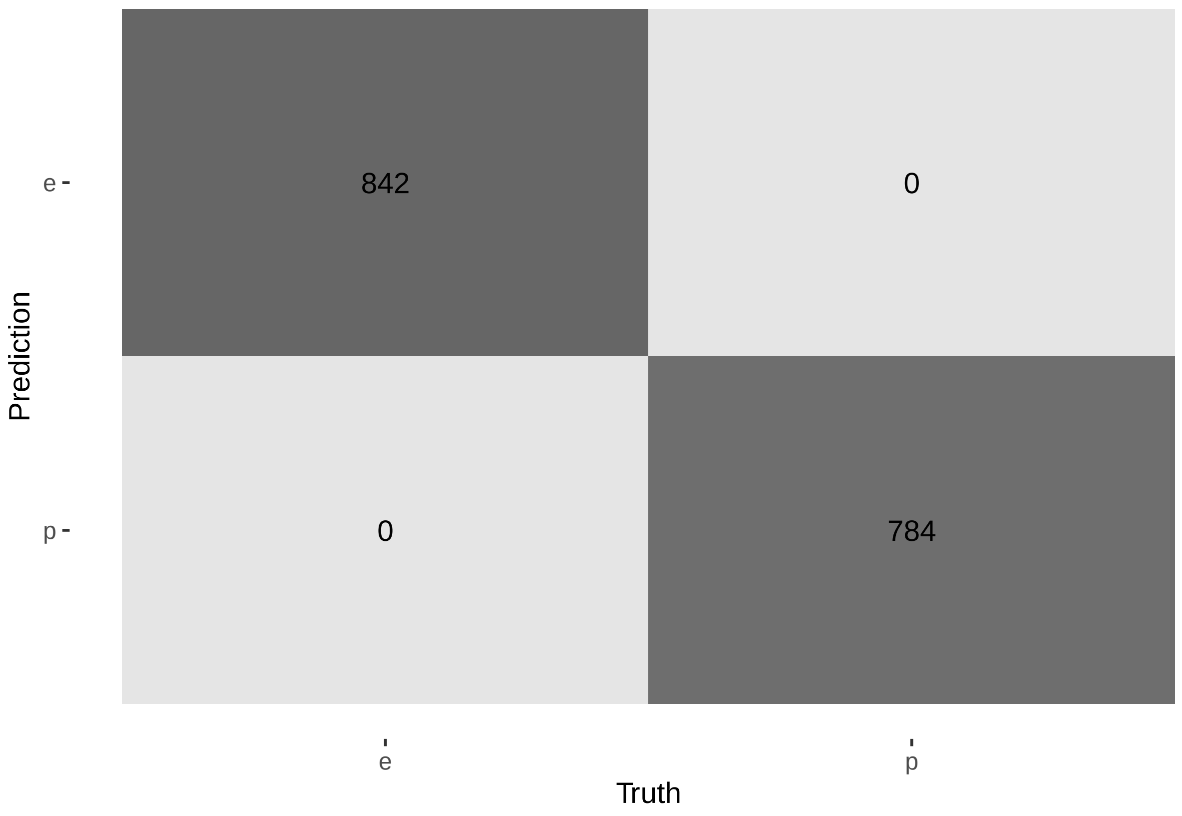
Let's continue with the logistic regression model that performed best. We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the logistic regression model using the select_best() function.
Then we fit optimal logistic regression model to training set.

```
set.seed(123)
log_workflow_tuned <- log_workflow %>%
  finalize_workflow(select_best(log_res, metric = "roc_auc"))
log_final <- fit(log_workflow_tuned, train)
```

Analysis of The Test Set

Lets fit the optimal logistic regression model to the testing data set and plot confusion matrix and calculate accuracy and auc value.The test results show that prediction accuracy rate and auc is 100%,the logistic regression model predicts perfectly.

```
set.seed(123)
#confusion matrix
augment(log_final, new_data = test) %>%
  conf_mat(truth = class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



```
#calculate accuracy
log_acc <- augment(log_final, new_data = test) %>%
  accuracy(truth = class, estimate = .pred_class)
log_acc # 100%
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary             1
```

```
#auc value
log_auc <- augment(log_final, new_data = test) %>%
  roc_auc(class, estimate = .pred_e) %>%
  select(.estimate)  # computing the AUC for the ROC curve
log_auc
```

```
## # A tibble: 1 x 1
##   .estimate
##       <dbl>
## 1         1
```

Conclusion

In order to study which characteristics of mushrooms are related to whether mushrooms are poisonous, we have made relevant exploratory data analysis. Further, we have established five models: Boost, SVM, KNN, Random forest and Logistic region, and tuned them. The auc value of each optimized model is 100%. By comparing the auc value and running time, we finally choose the logical regression model as our final model, which runs fastest. The model also performs very well on the test set, with an accuracy rate of 100%. It can effectively distinguish whether it is a poisonous mushroom according to the characteristics of the mushroom.