The Open University of Hong Kong School of Science and Technology

STAT S460F Advanced Topics In Data Mining

Final Project

Mushroom Classification

by

Chau Yuen Ying (12330225)

Data: Jan 2021

**Contents**

[1. Introduction 3](#_heading=h.1fob9te)

[2. Problem description 4](#_heading=h.3znysh7)

[3. Aims and Objectives 5](#_heading=h.2et92p0)

[4. Data description 6](#_heading=h.tyjcwt)

[4.1 Attribute Information 6](#_heading=h.3dy6vkm)

[4.2 Missing Data Treatment 7](#_heading=h.1t3h5sf)

[5. Exploratory and descriptive analysis of data 9](#_heading=h.4d34og8)

[5.1 Bar chart with each variable 9](#_heading=h.2s8eyo1)

[5.2 The dependent variable and independent variables 14](#_heading=h.17dp8vu)

[6. Analysis of data 19](#_heading=h.3rdcrjn)

[6.1 Dummy Data 19](#_heading=h.26in1rg)

[6.3 Feature selection 22](#_heading=h.lnxbz9)

[6.4 Model 25](#_heading=h.35nkun2)

[6.4.1 Support Vector Machine (SVM) 25](#_heading=h.1ksv4uv)

[6.4.2 Logistic Regression 29](#_heading=h.44sinio)

[6.4.3 Native Bayes Classifier 35](#_heading=h.2jxsxqh)

[7. Discussion and interpretation of the analysis 41](#_heading=h.z337ya)

[7.1 SVM Confusion Matrix 41](#_heading=h.3j2qqm3)

[7.2 Logistic Regression Confusion Matrix 42](#_heading=h.1y810tw)

[7.3 Native Bayes Classifier Confusion Matrix 43](#_heading=h.4i7ojhp)

[8. Conclusions, limitations and recommendations 46](#_heading=h.2xcytpi)

[8.1 Conclusions 46](#_heading=h.1ci93xb)

[8.1.1 Feature Selection: 46](#_heading=h.3whwml4)

[8.1.2 Model Compare 46](#_heading=h.2bn6wsx)

[8.2 Recommendations 48](#_heading=h.qsh70q)

[9. References 49](#_heading=h.3as4poj)

# Introduction

In “The Audubon Society Field Guide to North American Mushrooms” (1981), it was drawn 700 full-color identification mushroom photographs by color and shape and described each mushroom species by a detailed physical description, information on edibility, season, habitat, range, etc. (Lincoff, Lincoff, Nehring and Society, 1981). In the Agaricus and Lepiota Family Mushroom from this book, it was descripted 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility. Unknown edibility mushrooms were combined with the poisonous one (*Mushroom Data Set*).

The mushroom dataset is from Kaggle as csv file and spared two class: edible and poisonous. The edible mushroom defined no dangerous for human eating; The poisonous mushrooms defined dangerous for human eating. Based on two type class in mushroom dataset, the model in this project is used supervised learning method.

In this project, the dependent variable is class (type: edible and poisonous) and other 23 variables are independent variables:

| 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 |  |  |

Moreover, all variables are category and nominal variables, so they are converted to dummy variables as binomial.

# Problem description

In this project, it has two problems:

1. Which features are most indicative of a poisonous / edible mushroom?
2. What the model is the best for the mushroom estimation?

For the problem 1, it uses the feature selection method to find the feature for the most indicative of a poisonous or edible mushroom.

For the problem 2, it compares Support Vector Machine (SVM), Logistic Regression model and Native Bayes to estimate mushroom classification.

# Aims and Objectives

The aim / objectives of project includes:

1. Finding which is(are) feature(s) estimated the most accuracy
2. Compare Support Vector Machine (SVM), Logistic Regression model and Native Bayes

# Data description

In the mushroom classification dataset, all variables are categorical variables. The dependent variable is ‘class’. The independent variables include 23 variables. Total data size is 8124.

## 4.1 Attribute Information

| classes: edible=e, poisonous=p  cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s  cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s  cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y  bruises: bruises=t,no=f  odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s  gill-attachment: attached=a, descending=d, free=f, notched=n  gill-spacing: close=c, crowded=w, distant=d  gill-size: broad=b, narrow=n  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  stalk-shape: enlarging=e, tapering=t  stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?  stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s  stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s  stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y  stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y  veil-type: partial=p, universal=u  veil-color: brown=n, orange=o, white=w, yellow=y  ring-number: none=n, one=o, two=t  ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z  spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y  population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y  habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d |
| --- |

## 4.2 Missing Data Treatment

According to above of the attribute Information, ‘?’ value is missing data. So that, ‘?’ is convert to as NA.

mushrooms[mushrooms**==**'?'] <- NA  
**colSums**(**is.na**(mushrooms))

| ## 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 2480 |
| ## 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 |

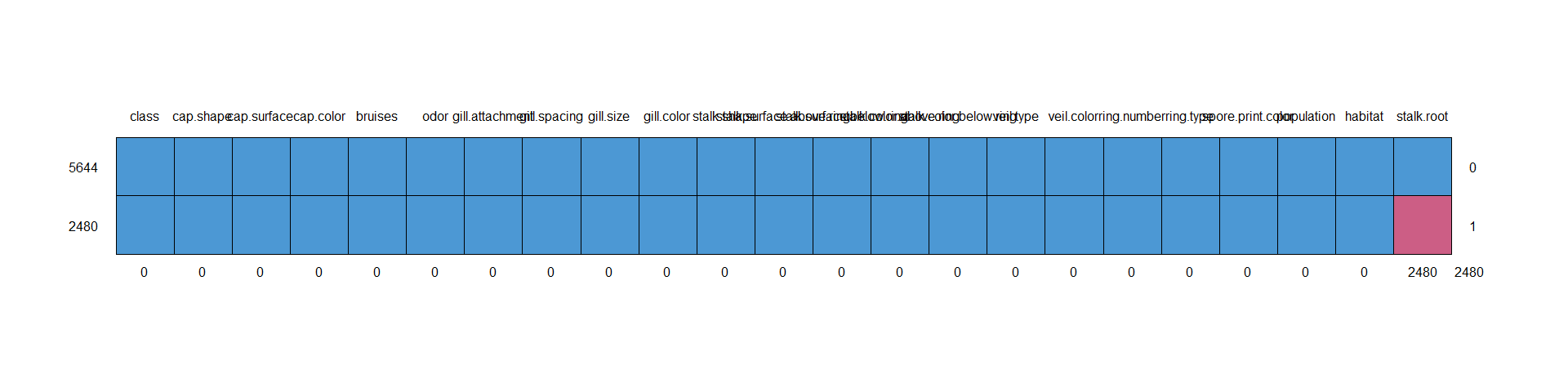
According to the above of result, only stalk.root has missing data and the amount of missing data has 2480.

Checking the percentage of the missing data account of the total data:

(2480**/**8124)**\***100

## [1] 30.52683

Since the missing data account 30.5% of the total data, it will be missing data treatment.



| ## class cap.shape cap.surface cap.color bruises odor gill.attachment | | | | |
| --- | --- | --- | --- | --- |
| ## 5644 1 1 1 1 1 1 1 | | | | |
| ## 2480 1 1 1 1 1 1 1 | | | | |
| ## 0 0 0 0 0 0 0 | | | | |
| ## gill.spacing gill.size gill.color stalk.shape stalk.surface.above.ring | | | | |
| ## 5644 1 1 1 1 1 | | | | |
| ## 2480 1 1 1 1 1 | | | | |
| ## 0 0 0 0 0 | | | | |
| ## stalk.surface.below.ring stalk.color.above.ring stalk.color.below.ring | | | | |
| ## 5644 1 1 1 | | | | |
| ## 2480 1 1 1 | | | | |
| ## 0 0 0 | | | | |
| ## veil.type veil.color ring.number ring.type spore.print.color population | | | | |
| ## 5644 1 1 1 1 1 1 | | | | |
| ## 2480 1 1 1 1 1 1 | | | | |
| ## 0 0 0 0 0 0 | | | | |
| ## habitat stalk.root | | | |  |
| ## 5644 1 1 0 | | | |
| ## 2480 1 0 1 | | | |
| ## 0 2480 2480 |  |  |  |

On the above of pattern, it shows that the missing data has not pattern and random.

Although these missing data may affect class type, the data size enough to provided information for y variable. So, the missing data is deleted.

mushrooms <- **na.omit**(mushrooms) *#Omit empty rows*  
   
**anyNA**(mushrooms)

## [1] FALSE

**dim**(mushrooms)

## [1] 5644 23

After removing the missing data, the new data size is 5644

# Exploratory and descriptive analysis of data

## 5.1 Bar chart with each variable

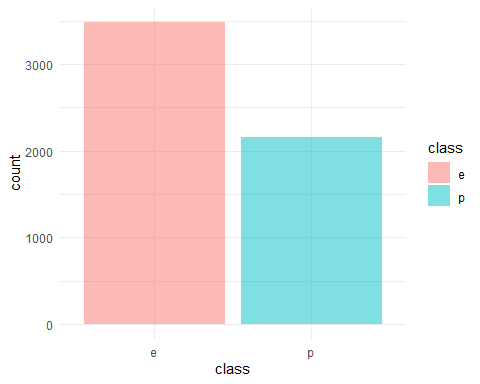
Mushrooms summary

**summary**(mushrooms)

## class cap.shape cap.surface cap.color bruises odor   
## e:3488 b: 300 f:2160 g :1696 f:2460 n :2776   
## p:2156 c: 4 g: 4 n :1164 t:3184 f :1584   
## f:2432 s:1260 y :1056 a : 400   
## k: 36 y:2220 w : 880 l : 400   
## s: 32 e : 588 p : 256   
## x:2840 b : 120 c : 192   
## (Other): 140 (Other): 36   
## gill.attachment gill.spacing gill.size gill.color stalk.shape stalk.root  
## a: 18 c:4620 b:4940 p :1384 e:2764 b:3776   
## f:5626 w:1024 n: 704 n : 984 t:2880 c: 556   
## w : 966 e:1120   
## h : 720 r: 192   
## g : 656   
## u : 480   
## (Other): 454   
## stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring  
## f: 552 f: 552 w :3136   
## k:1332 k:1296 p :1008   
## s:3736 s:3544 g : 576   
## y: 24 y: 252 n : 448   
## b : 432   
## c : 36   
## (Other): 8   
## stalk.color.below.ring veil.type veil.color ring.number ring.type  
## w :3088 p:5644 n: 0 n: 36 e: 824   
## p :1008 o: 0 o:5488 f: 0   
## g : 576 w:5636 t: 120 l:1296   
## n : 496 y: 8 n: 36   
## b : 432 p:3488   
## c : 36   
## (Other): 8   
## spore.print.color population habitat   
## n :1920 a: 384 d:2492   
## k :1872 c: 52 g:1860   
## h :1584 n: 256 l: 64   
## w : 148 s:1104 m: 292   
## r : 72 v:2160 p: 568   
## u : 48 y:1688 u: 368   
## (Other): 0 w: 0

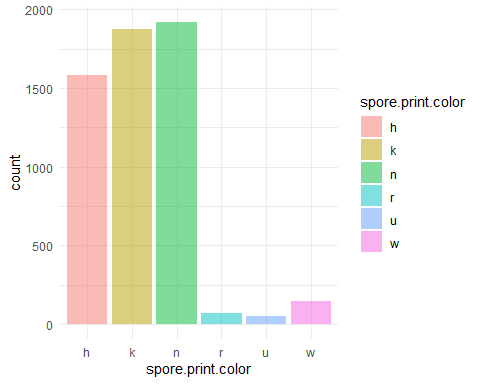
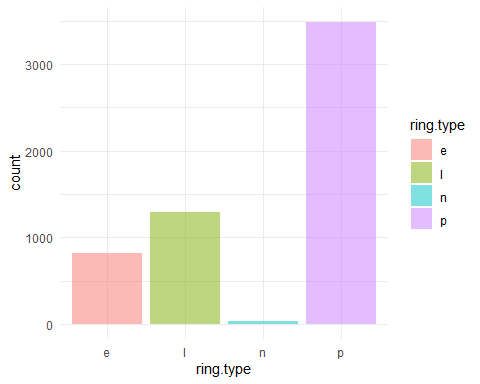
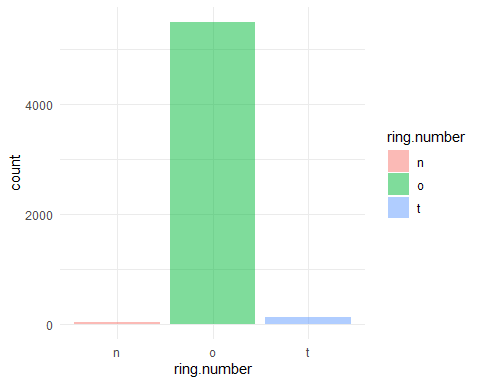
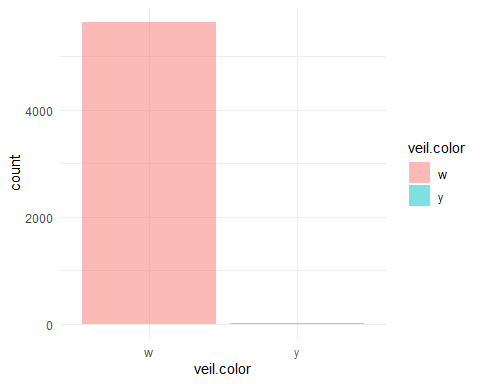
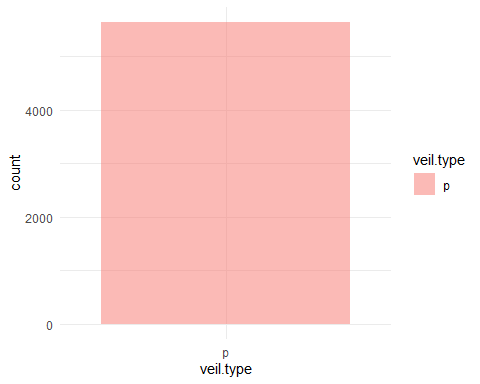
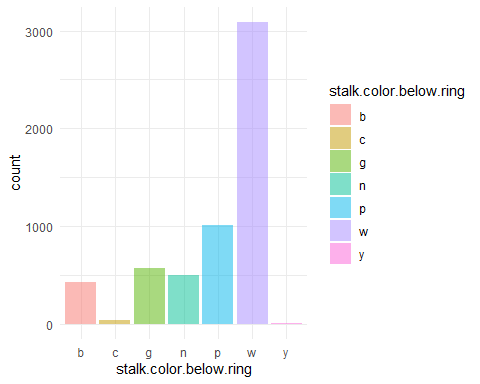
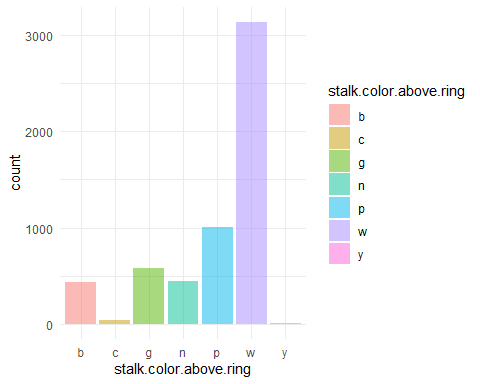
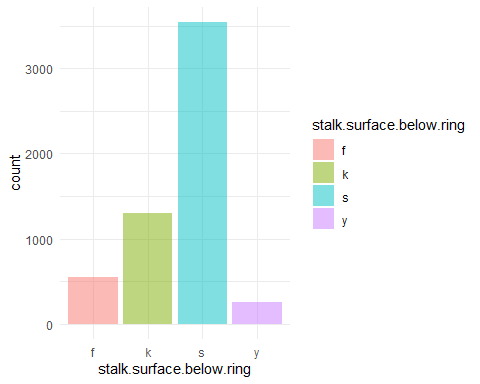
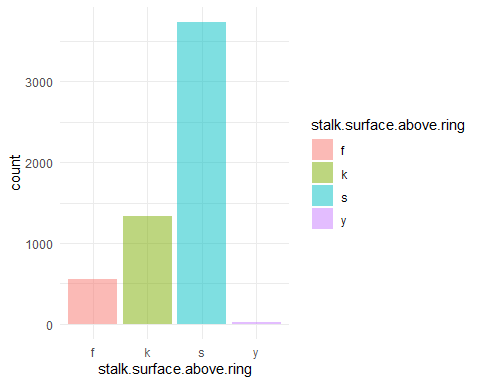
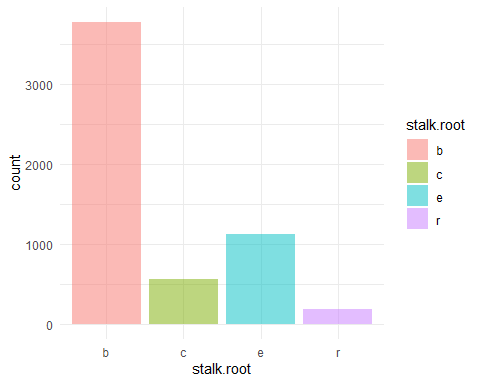
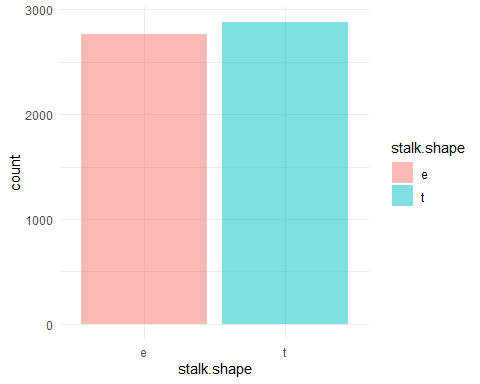
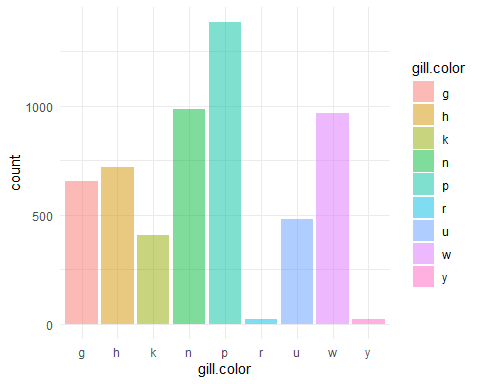
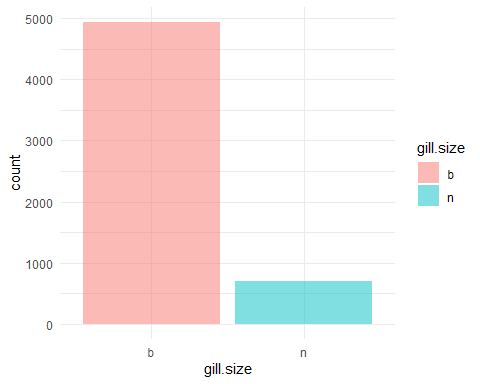
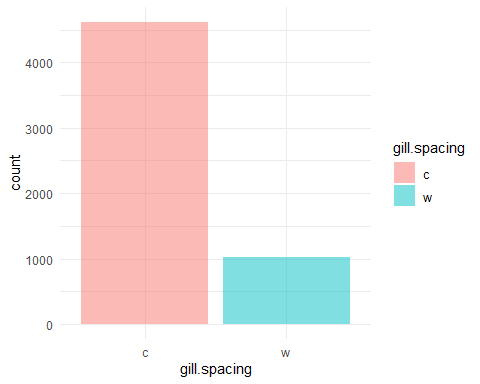
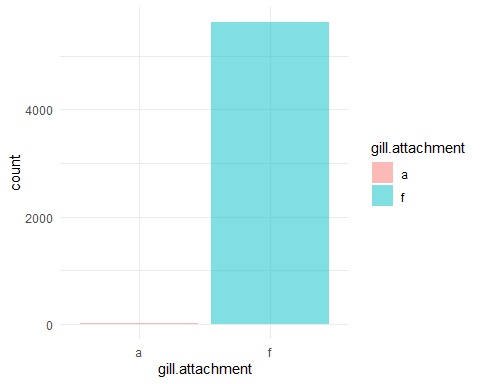
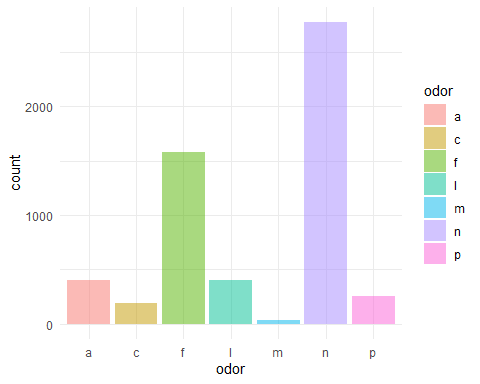
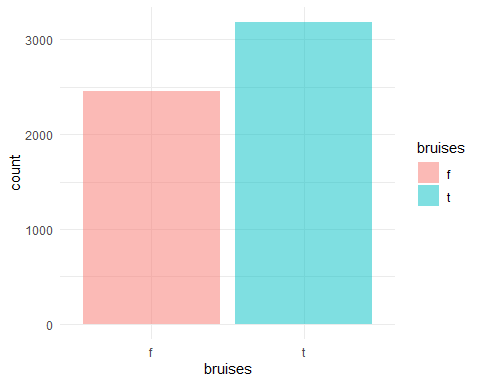
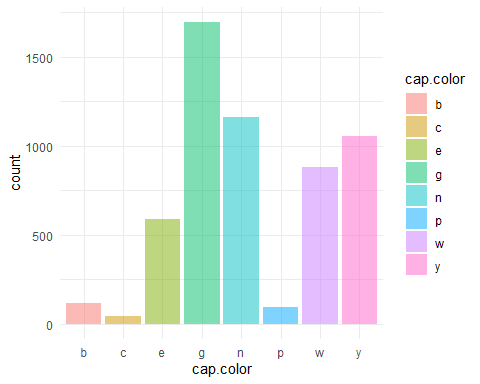
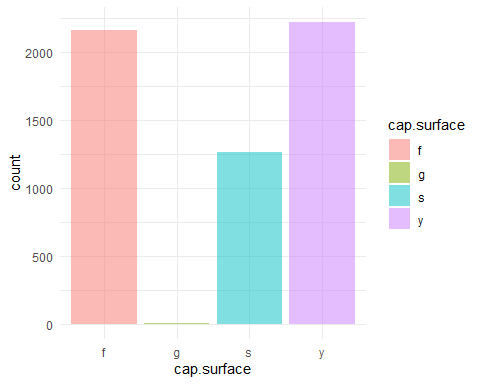
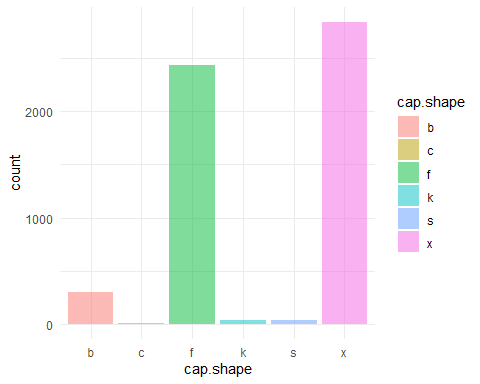
**for** (col **in** **names**(mushrooms[,])){  
 **print**(  
 **ggplot**(data=mushrooms, **aes\_string**(x=col, fill=col), nrow = 4) **+**  
 **geom\_bar**(alpha=0.5, position="dodge") **+**  
 **theme\_minimal**())  
}

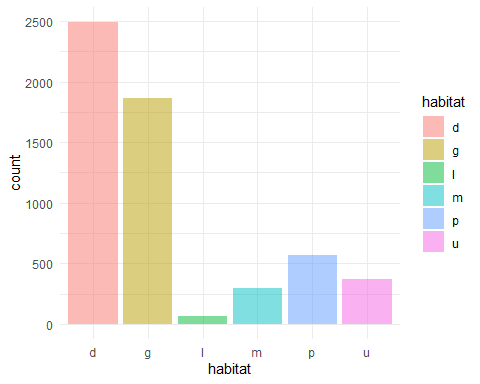
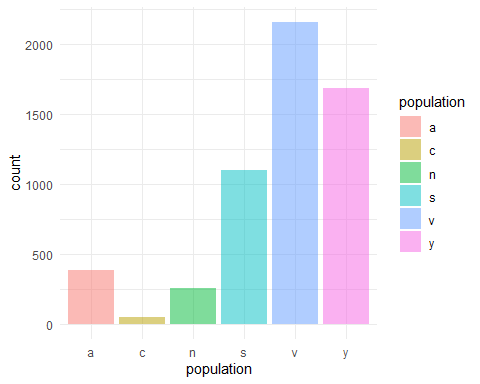
Bar chart:



The above of class bar chart, class type e and type p are not balance. The count amount of type e, 3488, is more than p, 2156.

Independent variables:





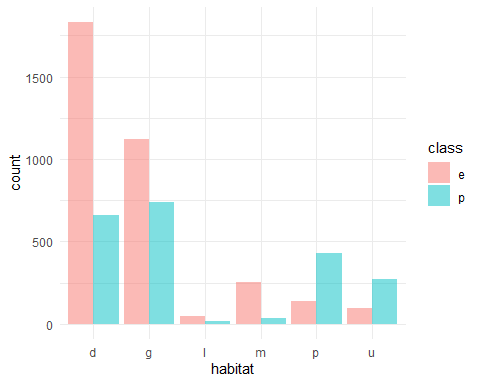
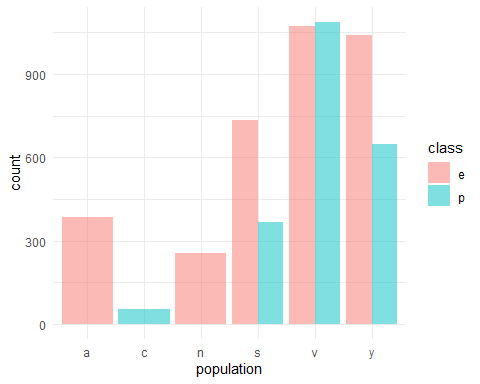
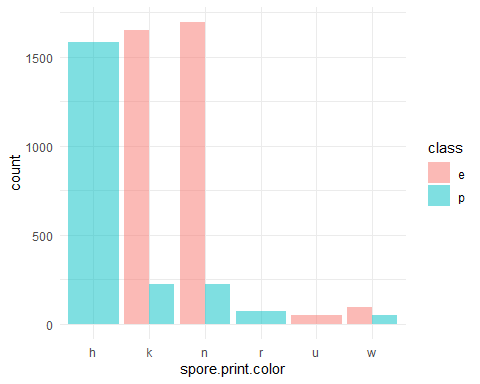
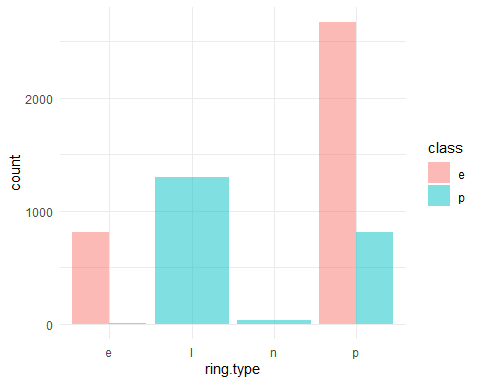
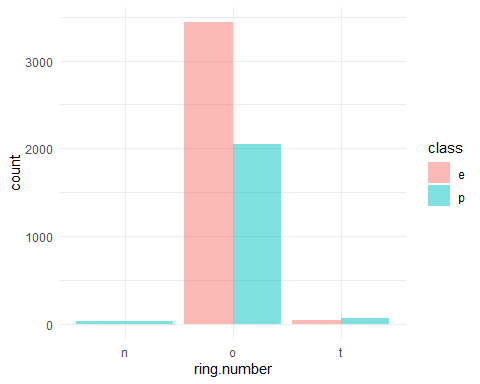
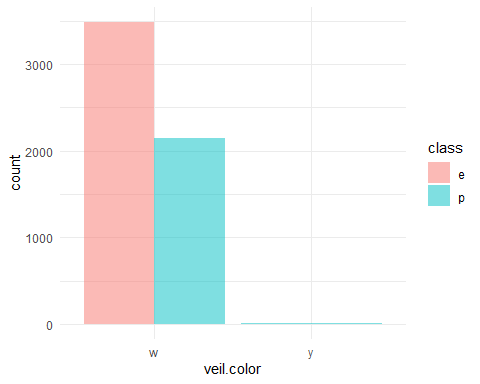
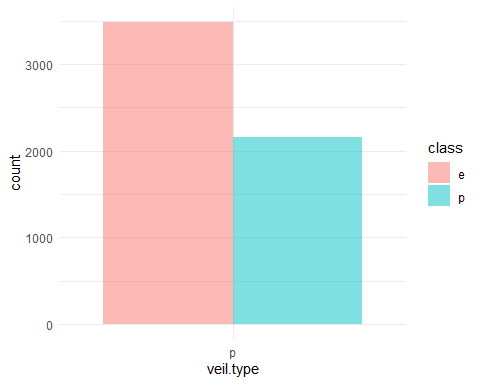
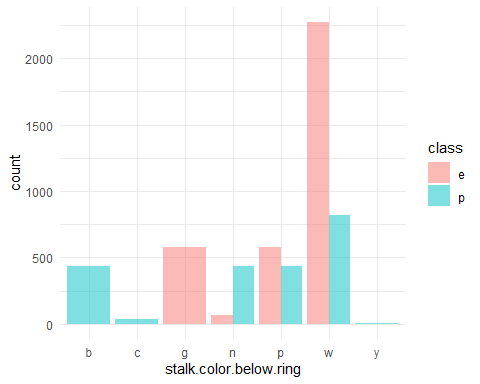
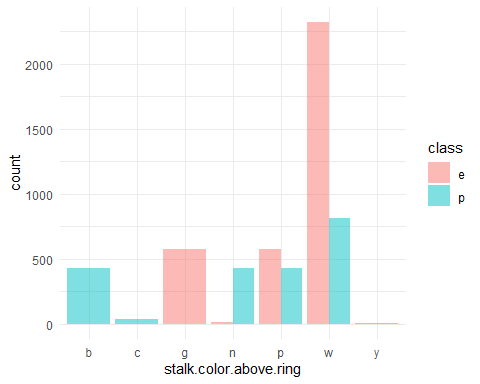
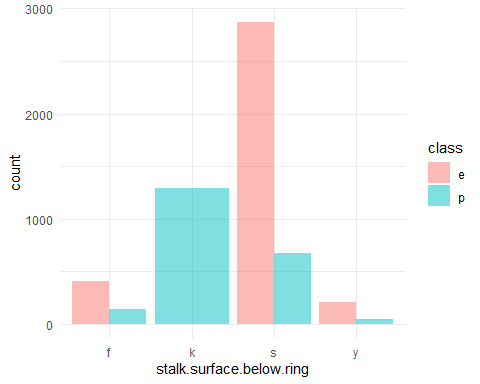
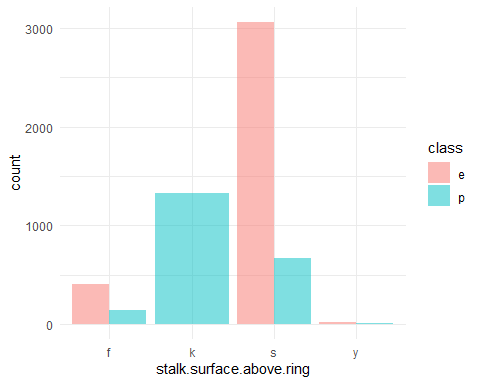
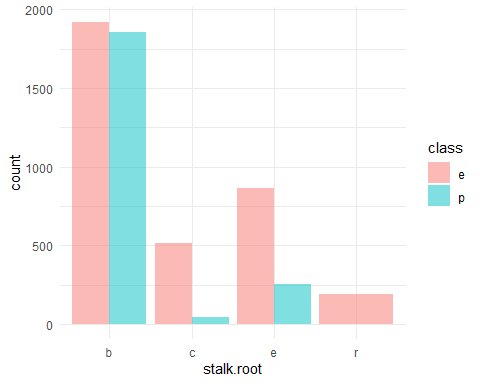
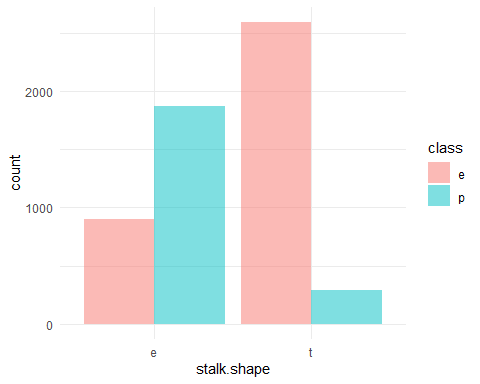
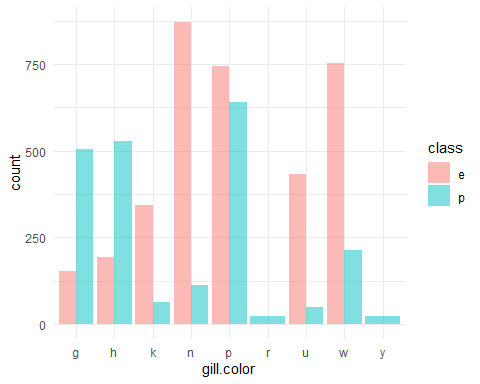
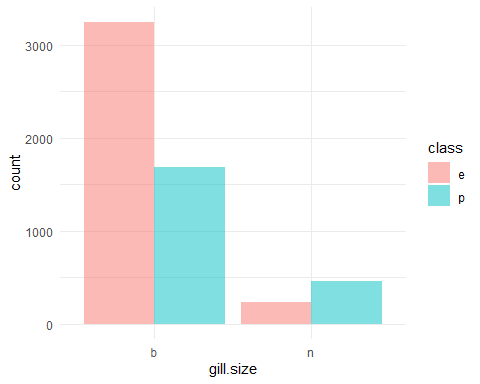
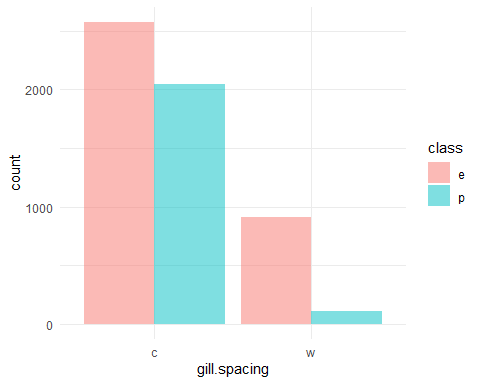
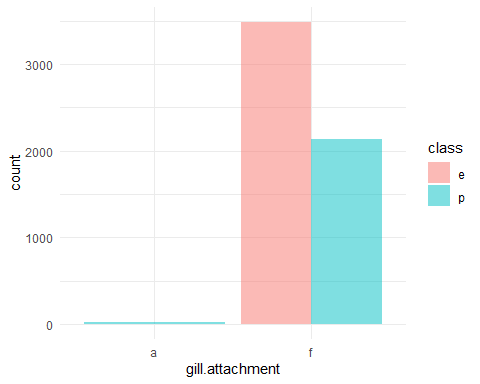
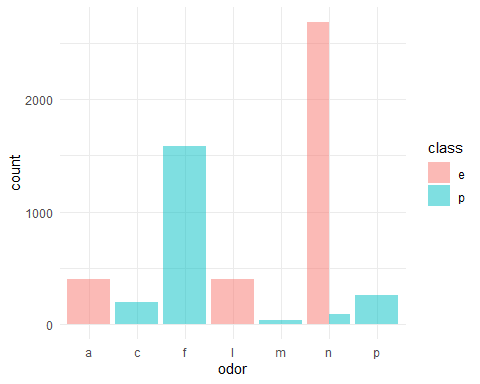
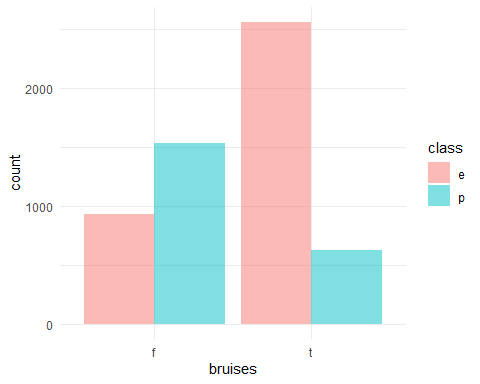
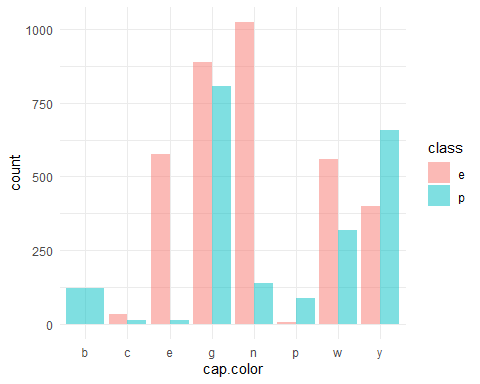
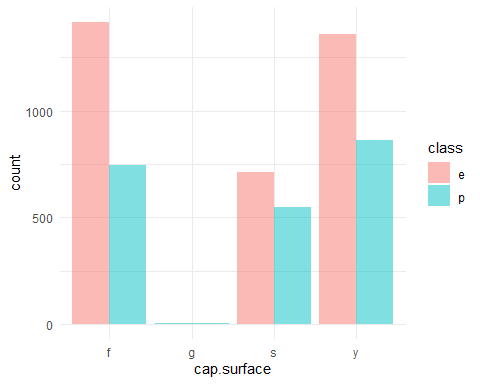
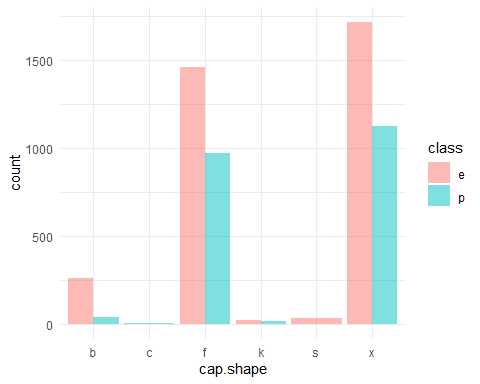
On the above of bar charts form independent variables, some variables extremely focus on once type, including gill.attachment, veil.color, ring.number. Taking veil.color as example, the amount of type w, 5236, is the most large than another type.

In addition, the bar chart of veil.type show as only once type, p.

## 5.2 The dependent variable and independent variables

Since all variables are category variables, the dependent variable and independent variables relationship is show as bar chart.

**for** (col **in** **names**(mushrooms[, -1])){  
 **print**(  
 **ggplot**(data=mushrooms, **aes\_string**(x=col, fill='class')) **+**  
 **geom\_bar**(alpha=0.5, position="dodge") **+**  
 **theme\_minimal**()  
 )  
}



On the above of bar chart, they show that the relationship between dependent variable and independent variables. You can see that some types of dependent variables are zero with the once class, such as, veil.color, cap.surface, odor etc. The reason is someone feature occurred on one class of mushroom.

As veil.type only has one value and the count of veil.type equal to total data count, it will be removed.

mushrooms <- **subset**(mushrooms, select = **-** veil.type)

**dim**(mushrooms)

## [1] 5644 22

Data size: 5644; x variables: 19

# Analysis of data

## 6.1 Dummy Data

Since all variables are category and nominal variables in the mushroom dataset, all variables are converted to dummy variables first.

mushrooms.dummy<-**dummyVars**(**~**.,data = mushrooms, fullRank=T)  
mushrooms.data<-**data.frame**(**predict**(mushrooms.dummy, newdata=mushrooms))

Since some variables have 2 type, they do not separate type with each variable and rename.

mushrooms.data = mushrooms.data **%>%**   
 dplyr**::rename**(  
 class = class.p,  
 bruises = bruises.t,  
 gill.attachment = gill.attachment.f,  
 stalk.shape = stalk.shape.t,  
 gill.size = gill.size.n  
 )  
 **str**(mushrooms.data)

## 'data.frame': 5644 obs. of 95 variables:  
## $ class : num 1 0 0 1 0 0 0 0 1 0 ...  
## $ cap.shape.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.shape.f : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.shape.k : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.shape.s : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.shape.x : num 1 1 0 1 1 1 0 0 1 0 ...  
## $ cap.surface.g : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.surface.s : num 1 1 1 0 1 0 1 0 0 1 ...  
## $ cap.surface.y : num 0 0 0 1 0 1 0 1 1 0 ...  
## $ cap.color.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.e : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.g : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ cap.color.n : num 1 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.p : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.r : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.u : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ cap.color.w : num 0 0 1 1 0 0 1 1 1 0 ...  
## $ cap.color.y : num 0 1 0 0 0 1 0 0 0 1 ...  
## $ bruises : num 1 1 1 1 0 1 1 1 1 1 ...  
## $ odor.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ odor.f : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ odor.l : num 0 0 1 0 0 0 0 1 0 0 ...  
## $ odor.m : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ odor.n : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ odor.p : num 1 0 0 1 0 0 0 0 1 0 ...  
## $ odor.s : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ odor.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.attachment : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ gill.spacing.w : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ gill.size : num 1 0 0 1 0 0 0 0 1 0 ...  
## $ gill.color.e : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.g : num 0 0 0 0 0 0 1 0 0 1 ...  
## $ gill.color.h : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.k : num 1 1 0 0 1 0 0 0 0 0 ...  
## $ gill.color.n : num 0 0 1 1 0 1 0 1 0 0 ...  
## $ gill.color.o : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.p : num 0 0 0 0 0 0 0 0 1 0 ...  
## $ gill.color.r : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.u : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.w : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ gill.color.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.shape : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ stalk.root.c : num 0 1 1 0 0 1 1 1 0 1 ...  
## $ stalk.root.e : num 1 0 0 1 1 0 0 0 1 0 ...  
## $ stalk.root.r : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.surface.above.ring.k: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.surface.above.ring.s: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ stalk.surface.above.ring.y: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.surface.below.ring.k: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.surface.below.ring.s: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ stalk.surface.below.ring.y: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.e : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.g : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.n : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.o : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.p : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.above.ring.w : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ stalk.color.above.ring.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.e : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.g : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.n : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.o : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.p : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ stalk.color.below.ring.w : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ stalk.color.below.ring.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ veil.color.o : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ veil.color.w : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ veil.color.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ring.number.o : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ ring.number.t : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ring.type.f : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ring.type.l : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ring.type.n : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ring.type.p : num 1 1 1 1 0 1 1 1 1 1 ...  
## $ spore.print.color.h : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ spore.print.color.k : num 1 0 0 1 0 1 1 0 1 1 ...  
## $ spore.print.color.n : num 0 1 1 0 1 0 0 1 0 0 ...  
## $ spore.print.color.o : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ spore.print.color.r : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ spore.print.color.u : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ spore.print.color.w : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ spore.print.color.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ population.c : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ population.n : num 0 1 1 0 0 1 1 0 0 0 ...  
## $ population.s : num 1 0 0 1 0 0 0 1 0 1 ...  
## $ population.v : num 0 0 0 0 0 0 0 0 1 0 ...  
## $ population.y : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ habitat.g : num 0 1 0 0 1 1 0 0 1 0 ...  
## $ habitat.l : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ habitat.m : num 0 0 1 0 0 0 1 1 0 1 ...  
## $ habitat.p : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ habitat.u : num 1 0 0 1 0 0 0 0 0 0 ...  
## $ habitat.w : num 0 0 0 0 0 0 0 0 0 0 ...

**dim**(mushrooms.data)

## [1] 5644 95

All variables are converted to dummy data. The new variables are totaled 95.

6.2 Split Data

Since the dependent variables with two class are not balance and decrease the predicted result biased on type e, it needs more training data for training model.

plyr**::count**(mushrooms[ ,1])

## x freq  
## 1 e 3488  
## 2 p 2156

Training data and testing data are 6:4

**set.seed**(1)  
mushrooms.train = mushrooms.data**%>%**  
 **sample\_frac**(0.6)  
mushrooms.test = mushrooms.data**%>%**  
 **setdiff**(mushrooms.train)

## 6.3 Feature selection

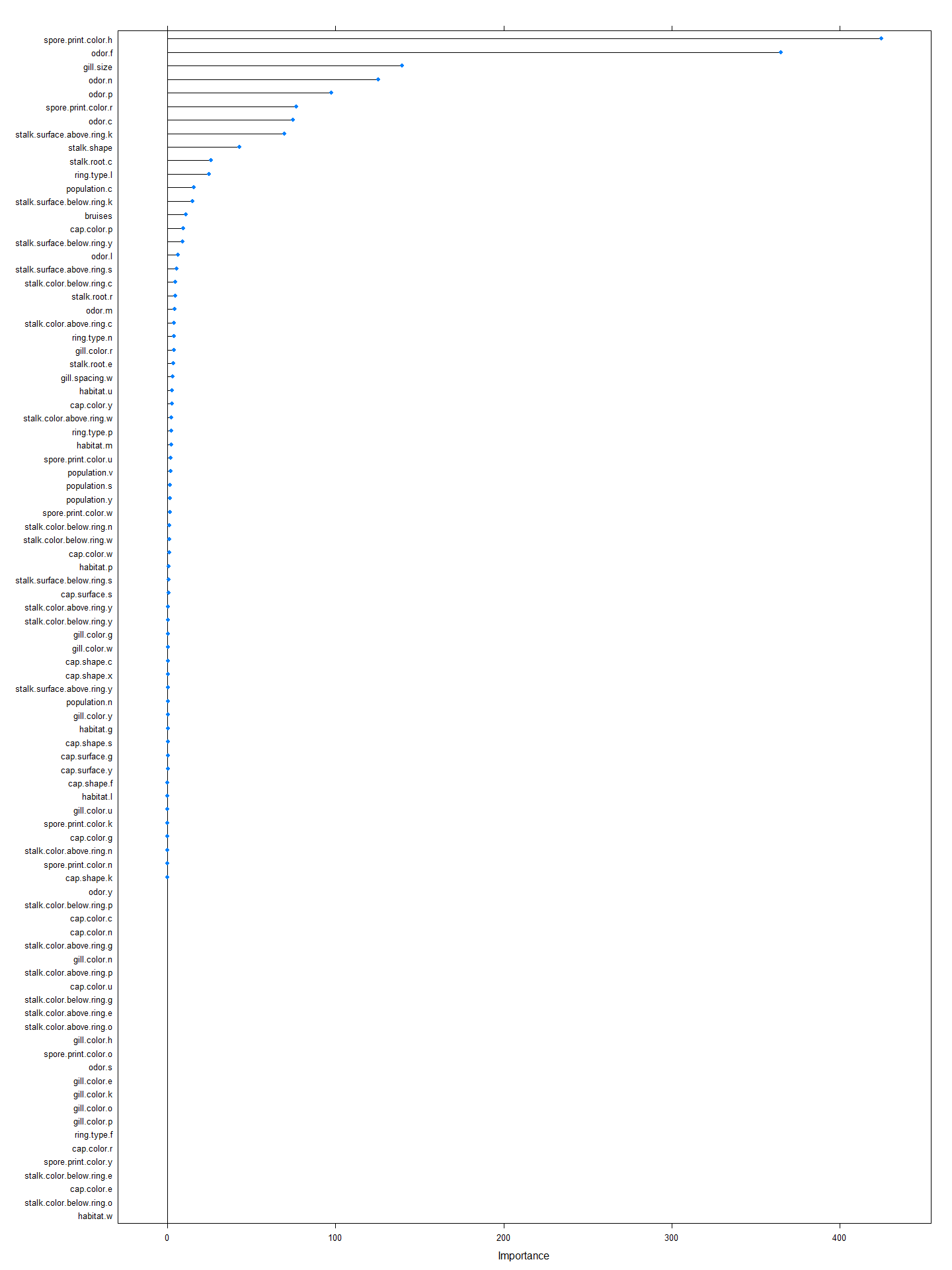
Since the number of independent variables is 95, it processes the feature selection by random forest with cross-valid for reduce the training model time.

control <- **trainControl**(method="repeatedcv", number=10, repeats=5)  
  
feature.select <- **train**(**as.factor**(class)**~**., data=mushrooms.train, method="rf", trControl=control)  
  
feature.importance <- **varImp**(feature.select, scale=FALSE)

*#summarize importance*  
**print**(feature.importance)

## rf variable importance  
##   
## only 20 most important variables shown (out of 88)  
##   
## Overall  
## spore.print.color.h 424.663  
## odor.f 365.010  
## gill.size 139.678  
## odor.n 125.482  
## odor.p 97.546  
## spore.print.color.r 76.801  
## odor.c 74.498  
## stalk.surface.above.ring.k 69.551  
## stalk.shape 42.912  
## stalk.root.c 25.968  
## ring.type.l 24.879  
## population.c 15.646  
## stalk.surface.below.ring.k 14.683  
## bruises 10.913  
## cap.color.p 9.254  
## stalk.surface.below.ring.y 9.052  
## odor.l 6.183  
## stalk.surface.above.ring.s 5.577  
## stalk.color.below.ring.c 4.545  
## stalk.root.r 4.452

*#plot importance*  
**plot**(feature.importance)



According the result, the most important feature have 20, for example, spore.print.color.h, odor.f, gill.size, odor.n, odor.p, stalk.surface.above.ring.k, etc.

In these 20 important feature, it is selected the 5 most important feature, including spore.print.color.h, odor.f, gill.size, odor.n and odor.p.

## 6.4 Model

In the project, it selects the supervised learning for the mushroom classification, so it will use the below of model:

1. Support Vector Machine (SVM)
2. Logistic Regression
3. Native Bayes

### 6.4.1 Support Vector Machine (SVM)

Support Vector Machine is the most population classified method. In there, it is used SVM with radial.

**set.seed**(1)  
svm.fit =**tune**(svm, class **~** odor.f **+** spore.print.color.h **+** gill.size **+** odor.n **+** odor.p,  
 data = mushrooms.train,   
 kernel = "radial",   
 ranges =**list**(cost = **c**(0.1, 1, 5, 10, 100)),  
 gamma = **c**(0.5,1,2,3,4)  
 )

**summary**(svm.fit)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 1  
##   
## - best performance: 0.0362229   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.1 0.03622359 0.007650275  
## 2 1.0 0.03622290 0.007650727  
## 3 5.0 0.03623077 0.007650962  
## 4 10.0 0.03623119 0.007652130  
## 5 100.0 0.03623090 0.007651855

The best cost is 1

svm.best = svm.fit**$**best.model  
**summary**(svm.best)

##   
## Call:  
## best.tune(method = svm, train.x = class ~ odor.f + spore.print.color.h +   
## gill.size + odor.n + odor.p, data = mushrooms.train, ranges = list(cost = c(0.1,   
## 1, 5, 10, 100)), kernel = "radial", gamma = c(0.5, 1, 2, 3, 4))  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.5 1 2 3 4   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 262

SVM Prediction:

*# Make predictions*  
svm.pred <- **predict**(svm.best, mushrooms.test)  
svm.pred <- **ifelse**(svm.pred **>** 0.5, 1, 0)  
*# Model accuracy*  
*# Model accuracy*  
**confusionMatrix**(data = **as.factor**(svm.pred), reference = **as.factor**(mushrooms.test**$**class))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1353 50  
## 1 43 812  
##   
## Accuracy : 0.9588   
## 95% CI : (0.9498, 0.9666)  
## No Information Rate : 0.6182   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9126   
##   
## Mcnemar's Test P-Value : 0.5338   
##   
## Sensitivity : 0.9692   
## Specificity : 0.9420   
## Pos Pred Value : 0.9644   
## Neg Pred Value : 0.9497   
## Prevalence : 0.6182   
## Detection Rate : 0.5992   
## Detection Prevalence : 0.6213   
## Balanced Accuracy : 0.9556   
##   
## 'Positive' Class : 0   
##

The Accuracy rate is 95.88%

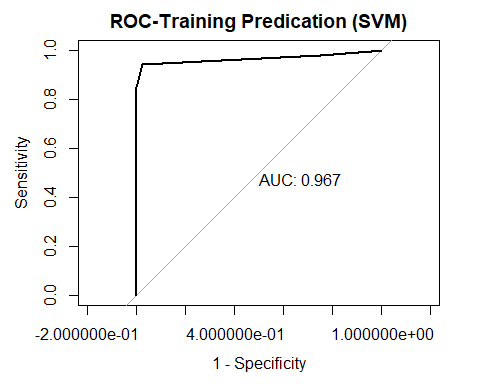
SVM ROC plot:

svm.pred.train = **predict**(svm.best, mushrooms.train, decision.values=TRUE)

**roc**(mushrooms.train[, 1], **as.numeric**(**as.factor**(svm.pred.train)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Training Predication (SVM)')

## Setting levels: control = 0, case = 1

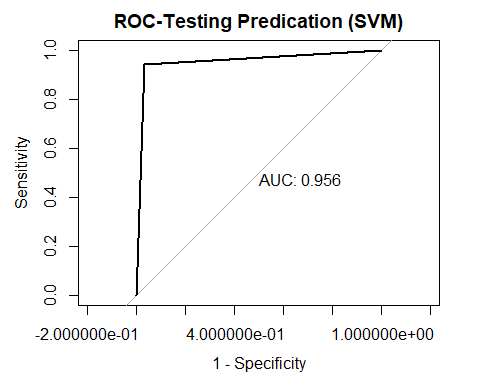
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.train[, 1], predictor = as.numeric(as.factor(svm.pred.train)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Training Predication (SVM)")  
##   
## Data: as.numeric(as.factor(svm.pred.train)) in 2092 controls (mushrooms.train[, 1] 0) < 1294 cases (mushrooms.train[, 1] 1).  
## Area under the curve: 0.9666

**roc**(mushrooms.test[, 1], **as.numeric**(**as.factor**(svm.pred)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Testing Predication (SVM)')

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.test[, 1], predictor = as.numeric(as.factor(svm.pred)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Testing Predication (SVM)")  
##   
## Data: as.numeric(as.factor(svm.pred)) in 1396 controls (mushrooms.test[, 1] 0) < 862 cases (mushrooms.test[, 1] 1).  
## Area under the curve: 0.9556

Testing ROC plot area is 95.6%. Training ROC area is 96.7%. Their gap is little so it does not exist overfitting.

### 6.4.2 Logistic Regression

Logistic Regression model analyze the binary and proportional response data; and the predictors can make the form of categorial, or binary variables (Hilbe, 2017). As the dataset are categorial variables, Logistic Regression model can be used in this project.

glm.fit = **glm**(class **~** odor.f **+** spore.print.color.h **+** gill.size **+** odor.n **+** odor.p,  
 data = mushrooms.train, family = binomial)  
  
**summary**(glm.fit)

##   
## Call:  
## glm(formula = class ~ odor.f + spore.print.color.h + gill.size +   
## odor.n + odor.p, family = binomial, data = mushrooms.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.34414 -0.19857 -0.19857 0.00005 2.80578   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.4344 0.1534 -15.873 < 2e-16 \*\*\*  
## odor.f 23.0005 577.3803 0.040 0.968   
## spore.print.color.h NA NA NA NA   
## gill.size 2.8182 0.1825 15.442 < 2e-16 \*\*\*  
## odor.n -1.4821 0.1870 -7.925 2.28e-15 \*\*\*  
## odor.p 20.1823 1419.5657 0.014 0.989   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4504.14 on 3385 degrees of freedom  
## Residual deviance: 899.47 on 3381 degrees of freedom  
## AIC: 909.47  
##   
## Number of Fisher Scoring iterations: 19

On the above of result, spore.print.color.h need to remove because it is NA. So it need to re-calculate.

glm.fit = **glm**(class **~** odor.f **+** gill.size **+** odor.n **+** odor.p,  
 data = mushrooms.train, family = binomial)  
  
**summary**(glm.fit)

##   
## Call:  
## glm(formula = class ~ odor.f + gill.size + odor.n + odor.p, family = binomial,   
## data = mushrooms.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.34414 -0.19857 -0.19857 0.00005 2.80578   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.4344 0.1534 -15.873 < 2e-16 \*\*\*  
## odor.f 23.0005 577.3803 0.040 0.968   
## gill.size 2.8182 0.1825 15.442 < 2e-16 \*\*\*  
## odor.n -1.4821 0.1870 -7.925 2.28e-15 \*\*\*  
## odor.p 20.1823 1419.5657 0.014 0.989   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4504.14 on 3385 degrees of freedom  
## Residual deviance: 899.47 on 3381 degrees of freedom  
## AIC: 909.47  
##   
## Number of Fisher Scoring iterations: 19

Logistic Regression Prediction

*# Make predictions*  
glm.pred <- **predict**(glm.fit, mushrooms.test)  
glm.pred <- **ifelse**(glm.pred **>** 0.5, 1, 0)

*# Model accuracy*  
**confusionMatrix**(data = **as.factor**(glm.pred), reference = **as.factor**(mushrooms.test**$**class))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1396 121  
## 1 0 741  
##   
## Accuracy : 0.9464   
## 95% CI : (0.9363, 0.9553)  
## No Information Rate : 0.6182   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8833   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8596   
## Pos Pred Value : 0.9202   
## Neg Pred Value : 1.0000   
## Prevalence : 0.6182   
## Detection Rate : 0.6182   
## Detection Prevalence : 0.6718   
## Balanced Accuracy : 0.9298   
##   
## 'Positive' Class : 0   
##

The accuracy rate is 94.64%

Residual and VIF

library(tidyverse)

library(broom)

glm.data <- augment(glm.fit) %>%

mutate(index = 1:n())

glm.data %>% top\_n(3, .cooksd)

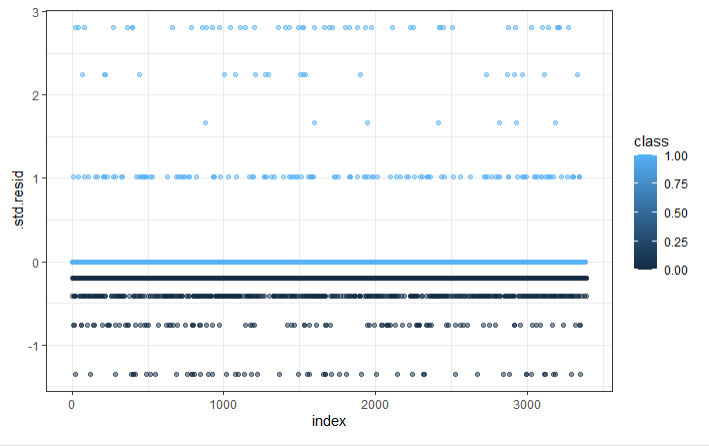
ggplot(glm.data, aes(index, .std.resid)) +

geom\_point(aes(color = class), alpha = .5) +

theme\_bw()

glm.data %>%

filter(abs(.std.resid) > 3)



The residual is no pattern and random. It means the residual is independent.

car::vif(glm.fit)

odor.f gill.size odor.n odor.p

1.000000 1.029954 1.029954 1.000000

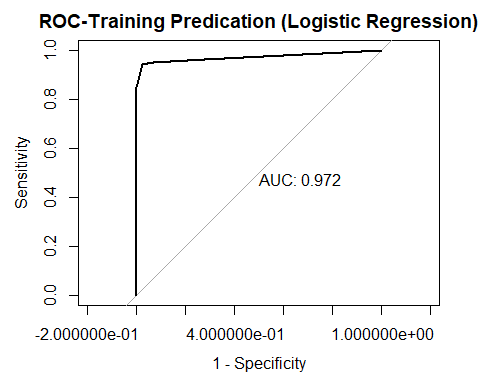
All VIF results are close to 1 so the model is not existing multicollinearity.

Logistic Regression ROC

glm.pred.train = **predict**(glm.fit, mushrooms.train, decision.values=TRUE)  
**roc**(mushrooms.train[, 1], **as.numeric**(**as.factor**(glm.pred.train)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Training Predication (Logistic Regression)')

## Setting levels: control = 0, case = 1

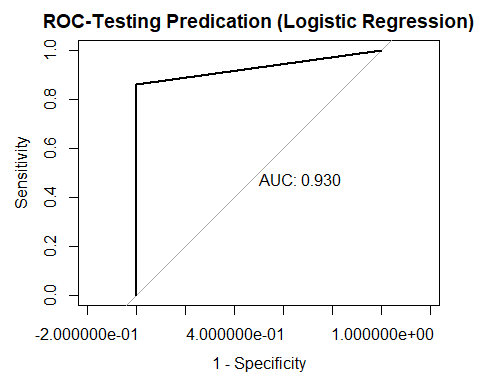
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.train[, 1], predictor = as.numeric(as.factor(glm.pred.train)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Training Predication (Logistic Regression)")  
##   
## Data: as.numeric(as.factor(glm.pred.train)) in 2092 controls (mushrooms.train[, 1] 0) < 1294 cases (mushrooms.train[, 1] 1).  
## Area under the curve: 0.9724

**roc**(mushrooms.test[, 1], **as.numeric**(**as.factor**(glm.pred)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Testing Predication (Logistic Regression)')

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.test[, 1], predictor = as.numeric(as.factor(glm.pred)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Testing Predication (Logistic Regression)")  
##   
## Data: as.numeric(as.factor(glm.pred)) in 1396 controls (mushrooms.test[, 1] 0) < 862 cases (mushrooms.test[, 1] 1).  
## Area under the curve: 0.9298

Testing ROC plot area is 93.0%. Training ROC area is 97.2%. Their gap is small so it may not exist overfitting.

### 6.4.3 Native Bayes Classifier

Native Bayes Classifier is common the classification method. It is a simple probabilistic classifier (Ashish, Rajat, & Sankar, 2007). Since this project is classified the type of mushroom, Native Bayes Classifier is selected method.

nb.fit = **naiveBayes**(**as.factor**(class)**~**odor.f **+** spore.print.color.h **+** gill.size **+** odor.n **+** odor.p,   
 data=mushrooms.train)  
nb.fit

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.6178382 0.3821618   
##   
## Conditional probabilities:  
## odor.f  
## Y [,1] [,2]  
## 0 0.0000000 0.0000000  
## 1 0.7287481 0.4447777  
##   
## spore.print.color.h  
## Y [,1] [,2]  
## 0 0.0000000 0.0000000  
## 1 0.7287481 0.4447777  
##   
## gill.size  
## Y [,1] [,2]  
## 0 0.06883365 0.2532316  
## 1 0.21947450 0.4140506  
##   
## odor.n  
## Y [,1] [,2]  
## 0 0.76720841 0.4227116  
## 1 0.04250386 0.2018137  
##   
## odor.p  
## Y [,1] [,2]  
## 0 0.0000000 0.000000  
## 1 0.1205564 0.325737

Native Bayes Classifier Prediction:

nb.pred <- **predict**(nb.fit, mushrooms.test)  
  
*# Model accuracy*  
**confusionMatrix**(data = **as.factor**(nb.pred), reference = **as.factor**(mushrooms.test**$**class))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1396 221  
## 1 0 641  
##   
## Accuracy : 0.9021   
## 95% CI : (0.8891, 0.9141)  
## No Information Rate : 0.6182   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.782   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.7436   
## Pos Pred Value : 0.8633   
## Neg Pred Value : 1.0000   
## Prevalence : 0.6182   
## Detection Rate : 0.6182   
## Detection Prevalence : 0.7161   
## Balanced Accuracy : 0.8718   
##   
## 'Positive' Class : 0   
##

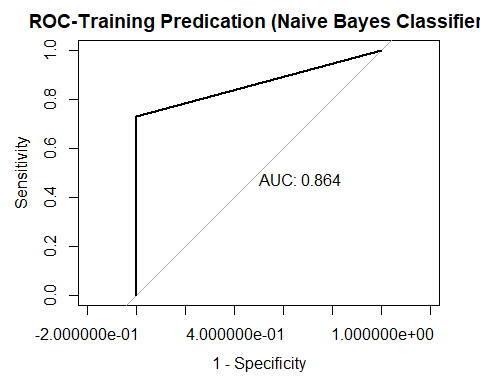
Accuracy is 90.21%

Native Bayes Classifier ROC:

nb.pred.train = **predict**(nb.fit, mushrooms.train, decision.values=TRUE)  
**roc**(mushrooms.train[, 1], **as.numeric**(**as.factor**(nb.pred.train)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Training Predication (Naive Bayes Classifier)')

## Setting levels: control = 0, case = 1

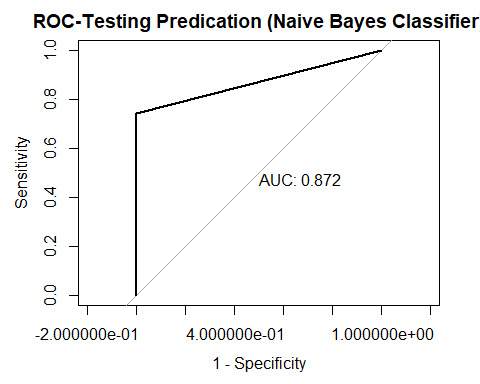
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.train[, 1], predictor = as.numeric(as.factor(nb.pred.train)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Training Predication (Naive Bayes Classifier)")  
##   
## Data: as.numeric(as.factor(nb.pred.train)) in 2092 controls (mushrooms.train[, 1] 0) < 1294 cases (mushrooms.train[, 1] 1).  
## Area under the curve: 0.8644

**roc**(mushrooms.test[, 1], **as.numeric**(**as.factor**(nb.pred)), plot=TRUE, print.auc = TRUE, legacy.axes=TRUE, main='ROC-Testing Predication (Naive Bayes Classifier)')

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases



##   
## Call:  
## roc.default(response = mushrooms.test[, 1], predictor = as.numeric(as.factor(nb.pred)), plot = TRUE, print.auc = TRUE, legacy.axes = TRUE, main = "ROC-Testing Predication (Naive Bayes Classifier)")  
##   
## Data: as.numeric(as.factor(nb.pred)) in 1396 controls (mushrooms.test[, 1] 0) < 862 cases (mushrooms.test[, 1] 1).  
## Area under the curve: 0.8718

Testing ROC plot area is 86.4%. Training ROC area is 87.7%. Their gap is little so it does not exist overfitting.

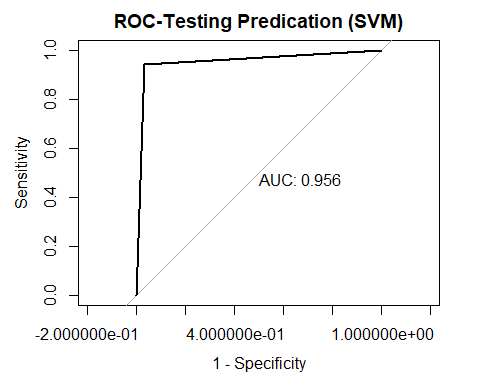
# Discussion and interpretation of the analysis

## 7.1 SVM Confusion Matrix

**confusionMatrix**(data = **as.factor**(svm.pred), reference = **as.factor**(mushrooms.test**$**class))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1353 39  
## 1 43 812  
##   
## Accuracy : 0.9635   
## 95% CI : (0.9549, 0.9709)  
## No Information Rate : 0.6213   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9225   
##   
## Mcnemar's Test P-Value : 0.7404   
##   
## Sensitivity : 0.9692   
## Specificity : 0.9542   
## Pos Pred Value : 0.9720   
## Neg Pred Value : 0.9497   
## Prevalence : 0.6213   
## Detection Rate : 0.6021   
## Detection Prevalence : 0.6195   
## Balanced Accuracy : 0.9617   
##   
## 'Positive' Class : 0   
##

ROC plot



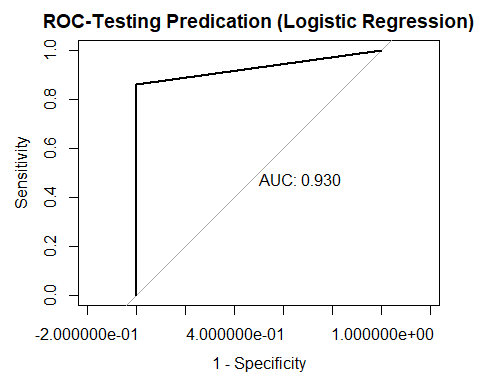
In the confusion matrix, it shows the sensitivity is 96.92%; specificity is 95.42%. Also, the positive prediction value rate is 97.2%; and the negative prediction value rate is 94.97%. Their rate is very closely. The prediction result is accuracy.

On the ROC plot, the roc area is 0.956, the result is good.

## 7.2 Logistic Regression Confusion Matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1396 110  
## 1 0 741  
##   
## Accuracy : 0.951   
## 95% CI : (0.9413, 0.9596)  
## No Information Rate : 0.6213   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8933   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.8707   
## Pos Pred Value : 0.9270   
## Neg Pred Value : 1.0000   
## Prevalence : 0.6213   
## Detection Rate : 0.6213   
## Detection Prevalence : 0.6702   
## Balanced Accuracy : 0.9354   
##   
## 'Positive' Class : 0

ROC



On the confusion matrix, the sensitivity rate is 100% but specificity rate is 87%. Also, the positive prediction value rate is 92.7% but negative prediction value rate is 100%. It means that it exits the biased situation.

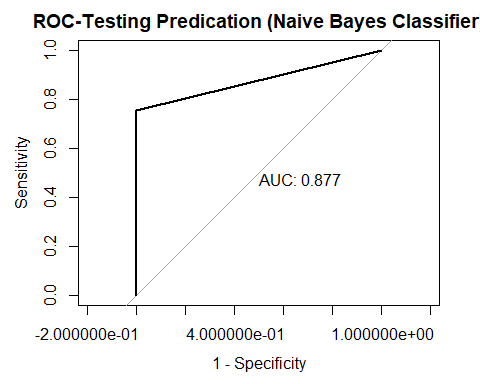
On the ROC plot, the roc area is 0.93, the result is not bad.

## 7.3 Native Bayes Classifier Confusion Matrix

**confusionMatrix**(data = **as.factor**(nb.pred), reference = **as.factor**(mushrooms.test**$**class))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1396 210  
## 1 0 641  
##   
## Accuracy : 0.9065   
## 95% CI : (0.8937, 0.9183)  
## No Information Rate : 0.6213   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7914   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.7532   
## Pos Pred Value : 0.8692   
## Neg Pred Value : 1.0000   
## Prevalence : 0.6213   
## Detection Rate : 0.6213   
## Detection Prevalence : 0.7147   
## Balanced Accuracy : 0.8766   
##   
## 'Positive' Class : 0   
##

ROC Plot



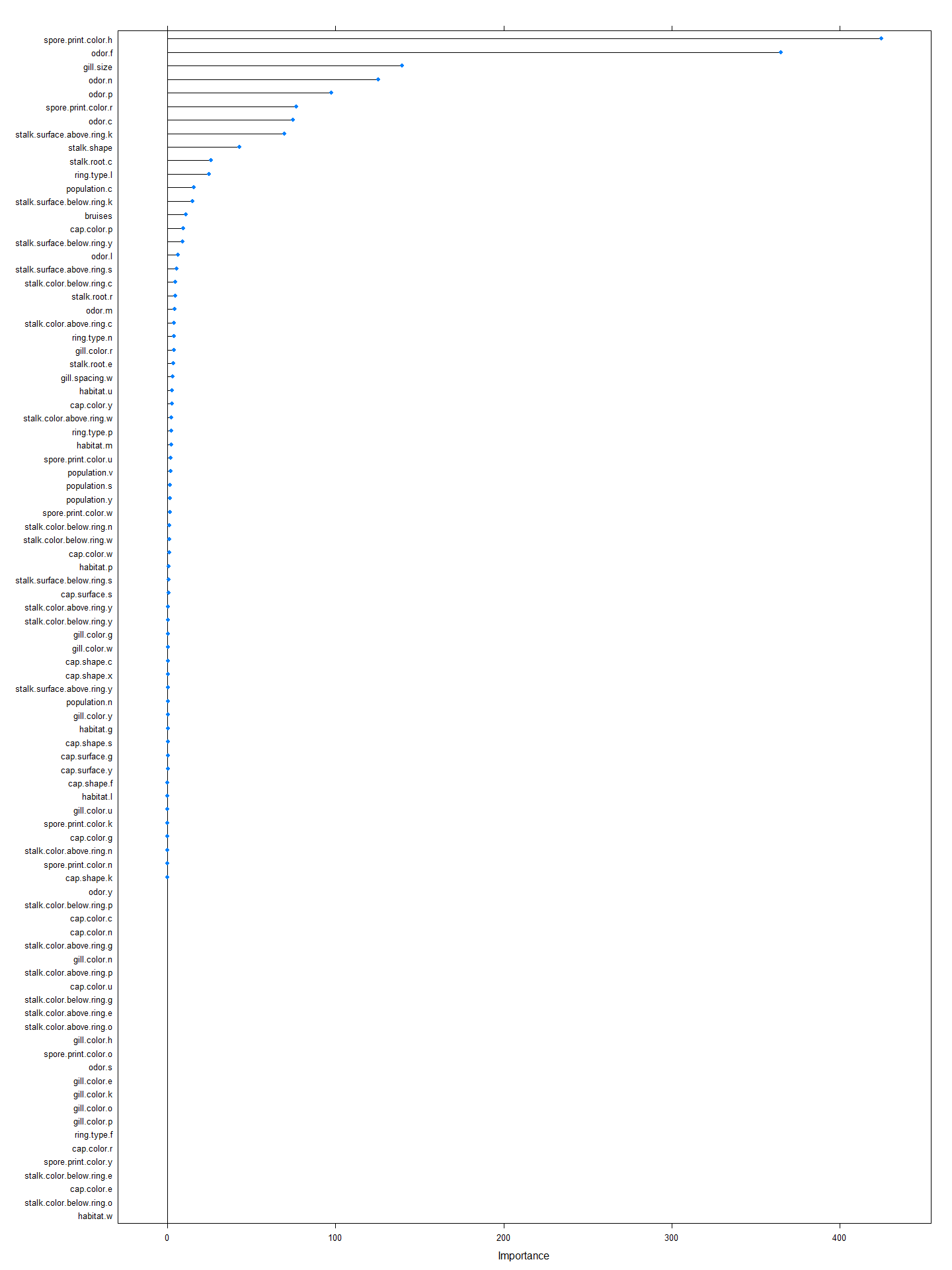
On the confusion matrix, the sensitivity rate is 100% but specificity rate is 75.32%. Also, the positive prediction value rate is 86.92% but negative prediction value rate is 100%. It means that it exits the biased situation.

On the ROC plot, the roc area is 0.877, the result is not bad but it not the most accuracy because the result is under 0.9.

# Conclusions, limitations and recommendations

## 8.1 Conclusions

### 8.1.1 Feature Selection:



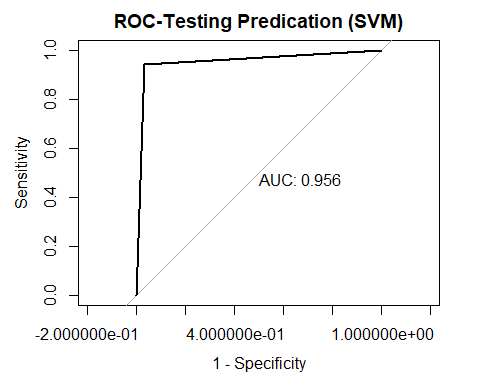
## Overall  
## spore.print.color.h 424.663  
## odor.f 365.010  
## gill.size 139.678  
## odor.n 125.482  
## odor.p 97.546  
## spore.print.color.r 76.801  
## odor.c 74.498  
## stalk.surface.above.ring.k 69.551  
## stalk.shape 42.912  
## stalk.root.c 25.968  
## ring.type.l 24.879  
## population.c 15.646  
## stalk.surface.below.ring.k 14.683  
## bruises 10.913

The most 5 important feature include spore.print.color.h, odor.f, gill.size, odor.n and odor.p.

### 8.1.2 Model Compare

SVM:

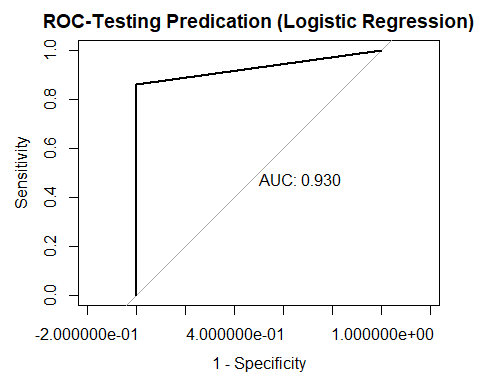
| Accuracy | 95.88% |
| --- | --- |
| Sensitivity | 96.92% |
| Specificity | 94.20% |
| Pos Pred Value | 96.44% |
| Neg Pred Value | 94.97% |



Accuracy, Sensitivity, Specificity, Pos Pred Value and Neg Pred Value are around 94% to 96.44%. ROC area is 0.956. The result is good.

Logistic Regression:

| Accuracy | 94.64% |
| --- | --- |
| Sensitivity | 100% |
| Specificity | 85.96% |
| Pos Pred Value | 92.02% |
| Neg Pred Value | 100% |

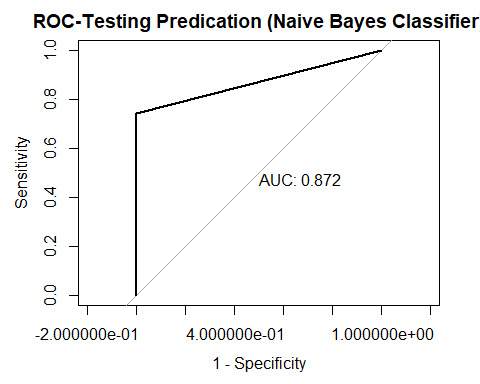


Accuracy, Specificity and Pos Pred Value are between 85% and 94.64%. Their result is not bad. However, Sensitivity and Neg Pred Value are 100%. It means the false positive result never occurred. It maybe exists the bias situation.

ROC area is 0.930. The result is not bad.

Native Bayes Classifier

| Accuracy | 90.21% |
| --- | --- |
| Sensitivity | 100% |
| Specificity | 74.36% |
| Pos Pred Value | 86.33% |
| Neg Pred Value | 100% |



Accuracy, Specificity and Pos Pred Value are between 74% and 90.21%.However, Sensitivity and Neg Pred Value are 100%. It means the false positive result never occurred. It maybe exists the bias situation.

ROC area is 0.872. The result is not bad.

## 8.2 Recommendations

Compare to SVM, Logistic Regression and Native Bayes Classifier result, SVM is recommended because SVM is the most accuracy. Also, the prediction results never occurred the false positive result by Logistic Regression and Native Bayes Classifier. Logistic Regression and Native Bayes Classifier maybe exists the bias situation. Therefore, SVM is the recommend model for the mushrooms classification.

8.3 Limitations

Since the positive responses, edible, are more than the negative responses, poisonous, the result focus on positive responses. Also, 5 mushroom features are used to estimate so it maybe led to the bias situation.

# References

Learning, U. (2016, December 01). Mushroom Classification. Retrieved January 13, 2021, from <https://www.kaggle.com/uciml/mushroom-classification>

Lincoff, G. H., Lincoff, G., Nehring, C., & Society, N. A. (1981). *The Audubon society field guide to North American mushrooms*. New York: Chanticleer Press.

Schlimmer, J. (n.d.). Mushroom Data Set. Retrieved January 23, 2021, from https://archive.ics.uci.edu/ml/datasets/mushroom

Hilbe, J. M. (2017). Preface. In *Logistic Regression Models* (pp. Viii-Vix). London: Taylor & Francis.