Project 4

Jean Jimenez, Matthew Roland, & Kelly Eng

2023-11-13

Load the required libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Logistic Regression of Mushroom Dataset

Introduction

The original Project 4 consisted of going through and processing spam/ not spam emails and labeling them as so. For this project ,we explored the mushroom dataset from UCI Machine Learning Repo. The classification of mushrooms as edible or poisonous is important to public health, culinary arts, and biological research. For project 4, we will try to classify these mushrooms as poisonous or edible using a logistic regression model.

Data Preprocessing

In the data preprocessing section of the project, several steps were taken to prepare the `mushroom` dataset for logistic regression analysis. First, the dataset was loaded into R for processing. The primary focus was to ensure that the data was 'tidy' and formatted correctly for analysis. This involved converting categorical variables into a suitable numeric format, as many machine learning algorithms, including logistic regression, work better with numerical input.

During this conversion, each category within the variables was assigned a unique numerical identifier. This step is essential to preserve the categorical information in a form that the logistic regression model can utilize effectively. Additionally, the dataset was checked for missing values, and necessary imputations were performed to handle any gaps in the data, ensuring that the model receives a complete dataset for training and testing.

```
# Read the file
url <- "https://raw.githubusercontent.com/Mattr5541/DATA_607_Project-4/main/mushroom/agaricus-lepiota_da
file = readLines(url)

split_data <- lapply(file, function(x) unlist(strsplit(x, ",")))
```

```r
# Convert the data to a dataframe
df <- as.data.frame(do.call(rbind, split_data))

# The names of the 23 columns as detailed in agaricus-lepiota_names.txt
col_names = c('class', 'cap_shape', 'cap_surface', 'cap_color', 'bruises', 'odor', 'gill_attachment', 'g

# Assign the names to each column of the dataframe
names(df) <- col_names

# The type of class for each mushroom is binary, poisonous or edible
# There are no missing values for them, so ifelse() can be used
df$class <- ifelse(df$class == 'p', 'poisonous', 'edible')

# These columns also have binary values
df$bruises <- ifelse(df$bruises == 't', 'bruises', 'no')
df$gill_size <- ifelse(df$gill_size == 'b', 'broad', 'narrow')
df$stalk_shape <- ifelse(df$stalk_shape == 'e', 'enlarging', 'tapering')
df$veil_type <- ifelse(df$veil_type == 'p', 'partial', 'universal')

# Rename all the values in each column to make them more clear as noted in agaricus-lepiota_names.txt
# The same file states the only missing values are in the stalk_root column
df <- df |>
  mutate(cap_shape = case_when(
    cap_shape == 'b' ~ 'bell',
    cap_shape == 'c' ~ 'conical',
    cap_shape == 'x' ~ 'convex',
    cap_shape == 'f' ~ 'flat',
    cap_shape == 'k' ~ 'knobbed',
    TRUE ~ 'sunken'),
    cap_surface = case_when(
      cap_surface == 'f' ~ 'fibrous',
      cap_surface == 'g' ~ 'grooves',
      cap_surface == 's' ~ 'scaly',
      TRUE ~ 'smooth'
    ),
    cap_color = case_when(
      cap_color == 'n' ~ 'brown',
      cap_color == 'b' ~ 'buff',
      cap_color == 'c' ~ 'cinnamon',
      cap_color == 'g' ~ 'gray',
      cap_color == 'r' ~ 'green',
      cap_color == 'p' ~ 'pink',
      cap_color == 'u' ~ 'purple',
      cap_color == 'e' ~ 'red',
      cap_color == 'w' ~ 'white',
      TRUE ~ 'yellow'
    ),
    odor = case_when(
      odor == 'a' ~ 'almond',
      odor == 'l' ~ 'anise',
      odor == 'c' ~ 'creosote',
      odor == 'y' ~ 'fishy',
      odor == 'f' ~ 'foul',
```

```
    odor == 'm' ~ 'musty',
    odor == 'n' ~ 'none',
    odor == 'p' ~ 'pungent',
    TRUE ~ 'spicy'
  ),
  gill_attachment = case_when(
    gill_attachment == 'a' ~ 'attached',
    gill_attachment == 'd' ~ 'descending',
    gill_attachment == 'f' ~ 'free',
    TRUE ~ 'notched'
  ),
  gill_spacing = case_when(
    gill_spacing == 'c' ~ 'close',
    gill_spacing == 'w' ~ 'crowded',
    TRUE ~ 'distant'
  ),
  gill_color = case_when(
    gill_color == 'k' ~ 'black',
    gill_color == 'n' ~ 'brown',
    gill_color == 'b' ~ 'buff',
    gill_color == 'h' ~ 'chocolate',
    gill_color == 'g' ~ 'gray',
    gill_color == 'r' ~ 'green',
    gill_color == 'o' ~ 'orange',
    gill_color == 'p' ~ 'pink',
    gill_color == 'u' ~ 'purple',
    gill_color == 'e' ~ 'red',
    gill_color == 'w' ~ 'white',
    TRUE ~ 'yellow'
  ),
  stalk_root = case_when(
    stalk_root == 'b' ~ 'bulbous',
    stalk_root == 'c' ~ 'club',
    stalk_root == 'u' ~ 'cup',
    stalk_root == 'e' ~ 'equal',
    stalk_root == 'z' ~ 'rhizomorphs',
    stalk_root == 'r' ~ 'rooted',
    TRUE ~ NA
  ),
  stalk_surface_above_ring = case_when(
    stalk_surface_above_ring == 'f' ~ 'fibrous',
    stalk_surface_above_ring == 'y' ~ 'scaly',
    stalk_surface_above_ring == 'k' ~ 'silky',
    TRUE ~ 'smooth'
  ),
  stalk_surface_below_ring = case_when(
    stalk_surface_below_ring == 'f' ~ 'fibrous',
    stalk_surface_below_ring == 'y' ~ 'scaly',
    stalk_surface_below_ring == 'k' ~ 'silky',
    TRUE ~ 'smooth'
    ),
  stalk_color_above_ring = case_when(
    stalk_color_above_ring == 'n' ~ 'brown',
```

```
    stalk_color_above_ring == 'b' ~ 'buff',
    stalk_color_above_ring == 'c' ~ 'cinnamon',
    stalk_color_above_ring == 'g' ~ 'gray',
    stalk_color_above_ring == 'o' ~ 'orange',
    stalk_color_above_ring == 'p' ~ 'pink',
    stalk_color_above_ring == 'e' ~ 'red',
    stalk_color_above_ring == 'w' ~ 'white',
    TRUE ~ 'yellow'
  ),
  stalk_color_below_ring = case_when(
    stalk_color_below_ring == 'n' ~ 'brown',
    stalk_color_below_ring == 'b' ~ 'buff',
    stalk_color_below_ring == 'c' ~ 'cinnamon',
    stalk_color_below_ring == 'g' ~ 'gray',
    stalk_color_below_ring == 'o' ~ 'orange',
    stalk_color_below_ring == 'p' ~ 'pink',
    stalk_color_below_ring == 'e' ~ 'red',
    stalk_color_below_ring == 'w' ~ 'white',
    TRUE ~ 'yellow'
  ),
  veil_color = case_when(
    veil_color == 'n' ~ 'brown',
    veil_color == 'o' ~ 'orange',
    veil_color == 'w' ~ 'white',
    TRUE ~ 'yellow'
  ),
  ring_number = case_when(
    ring_number == 'n' ~ '0',
    ring_number == 'o' ~ '1',
    TRUE ~ '2'
  ),
  ring_type = case_when(
    ring_type == 'c' ~ 'cobwebby',
    ring_type == 'e' ~ 'evanescent',
    ring_type == 'f' ~ 'flaring',
    ring_type == 'l' ~ 'large',
    ring_type == 'n' ~ 'none',
    ring_type == 'p' ~ 'pendant',
    ring_type == 's' ~ 'sheathing',
    TRUE ~ 'zone'
  ),
  spore_print_color = case_when(
    spore_print_color == 'k' ~ 'black',
    spore_print_color == 'n' ~ 'brown',
    spore_print_color == 'b' ~ 'buff',
    spore_print_color == 'h' ~ 'chocolate',
    spore_print_color == 'r' ~ 'green',
    spore_print_color == 'o' ~ 'orange',
    spore_print_color == 'u' ~ 'purple',
    spore_print_color == 'w' ~ 'white',
    TRUE ~ 'yellow'
  ),
  population = case_when(
```

```
      population == 'a' ~ 'abundant',
      population == 'c' ~ 'clustered',
      population == 'n' ~ 'numerous',
      population == 's' ~ 'scattered',
      population == 'v' ~ 'several',
      TRUE ~ 'solitary'
    ),
    habitat = case_when(
      habitat == 'g' ~ 'grasses',
      habitat == 'l' ~ 'leaves',
      habitat == 'm' ~ 'meadows',
      habitat == 'p' ~ 'paths',
      habitat == 'u' ~ 'urban',
      habitat == 'w' ~ 'waste',
      TRUE ~ 'woods'
    ))

# Convert the number of rings from character to numeric
df$ring_number <- as.numeric(df$ring_number)

# According to agaricus-lepopta_names.txt, there are 2480 missing attributes for the stalk_root column
df <- df |>
  filter(!is.na(stalk_root))

mushroom_colors <- c('brown', 'orange', 'white', 'yellow', 'buff', 'gray', 'pink', 'red', 'green', 'pur
surface <- c('fibrous', 'scaly', 'silky', 'smooth')

df_num <- df

df_num$class <- as.numeric(factor(df_num$class, levels=c('poisonous', 'edible')))
df_num$cap_shape <- as.numeric(factor(df_num$cap_shape, levels=c('bell', 'conical', 'convex', 'flat', '
df_num$cap_surface <- as.numeric(factor(df_num$cap_surface, levels=c('fibrous', 'grooves', 'scaly', 'sm
df_num$cap_color <- as.numeric(factor(df_num$cap_color, levels=mushroom_colors))
df_num$bruises <- as.numeric(factor(df_num$bruises, levels=c('bruises', 'no')))
df_num$odor <- as.numeric(factor(df_num$odor, levels=c('almond', 'anise', 'creosote', 'fishy', 'foul',
df_num$gill_attachment <- as.numeric(factor(df_num$gill_attachment, levels=c('attached','descending','f
df_num$gill_spacing <- as.numeric(factor(df_num$gill_spacing, levels=c('close', 'crowded', 'distant')))
df_num$gill_size <- as.numeric(factor(df_num$gill_size, levels=c('broad', 'narrow')))
df_num$gill_color <- as.numeric(factor(df_num$gill_color, levels=mushroom_colors))
df_num$stalk_shape <- as.numeric(factor(df_num$stalk_shape, levels=c('enlarging', 'tapering')))
df_num$stalk_root <- as.numeric(factor(df_num$stalk_root, levels=c('bulbous', 'club', 'cup', 'equal', '
df_num$stalk_surface_above_ring <- as.numeric(factor(df_num$stalk_surface_above_ring, levels=surface))
df_num$stalk_surface_below_ring <- as.numeric(factor(df_num$stalk_surface_below_ring, levels=surface))
df_num$stalk_color_above_ring <- as.numeric(factor(df_num$stalk_color_above_ring, levels=mushroom_color
df_num$stalk_color_below_ring <- as.numeric(factor(df_num$stalk_color_below_ring, levels=mushroom_color
df_num$veil_type <- as.numeric(factor(df_num$veil_type, levels=c('partial', 'universal')))
df_num$veil_color <- as.numeric(factor(df_num$veil_color, levels=mushroom_colors))
df_num$ring_type <- as.numeric(factor(df_num$ring_type, levels=c('cobwebby', 'evanescent', 'flaring', '
df_num$spore_print_color <- as.numeric(factor(df_num$spore_print_color, levels=mushroom_colors))
df_num$population <- as.numeric(factor(df_num$population, levels=c('abundant', 'clustered', 'numerous',
df_num$habitat <- as.numeric(factor(df_num$habitat, levels=c('grasses', 'leaves', 'meadows', 'paths', '

# Convert the binary columns to zeros and ones
```

```r
# Veil type is also binary but it only contains partial for all columns, there's not a single row that
binary_cols <- c(1, 5, 9, 11)
for (col in binary_cols) {
  df_num[[col]] <- df_num[[col]] - 1
}
```

Exploratory Data Analysis

For exploratory data analysis, a thorough examination of the `mushroom` dataset was conducted to gain insights into its characteristics and uncover any underlying patterns. This involved generating summary statistics; including measures like mean, median, and standard deviation for each variable.

Bar plots were created for the various categorical features such as `ring_number`, `bruises`, `gill_size`, and `stalk_shape`. These visualizations are important in showing the frequency distribution of different categories within each feature. The bar plots also help to visualize the relationship between these categorical features and the target variable (edible or poisonous).

A correlation matrix was made to provide insights into how different features are related to each other.

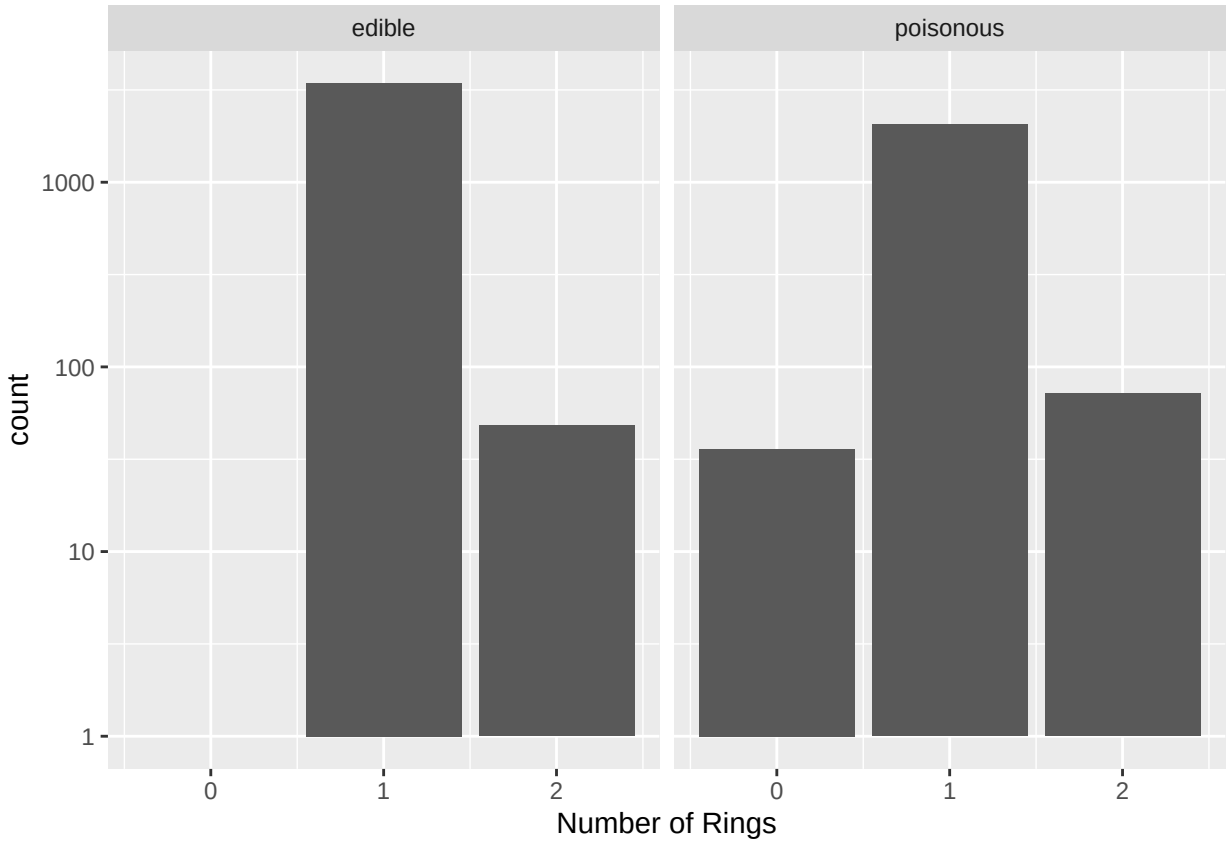
```r
# Summary Statistics
summary(df_num$ring_number)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.015   1.000   2.000
```

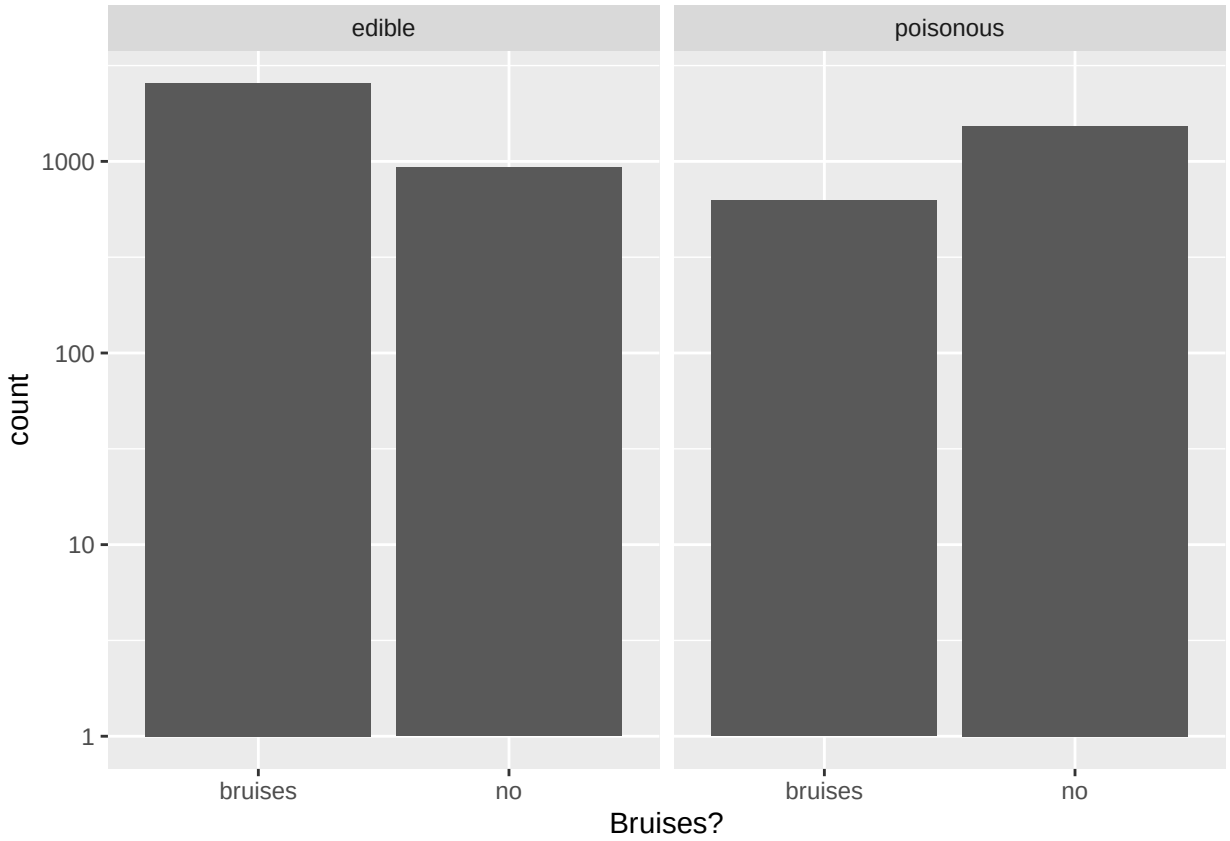
```r
# Gets the mode for all columns
# The type of mushrooms that appear the most in this dataset are poisonous
df_num |>
  summarise(across(everything(), ~as.numeric(names(which.max(table(.))))))
```

```
##   class cap_shape cap_surface cap_color bruises odor gill_attachment
## 1     1         3           4         6       0    7               3
##   gill_spacing gill_size gill_color stalk_shape stalk_root
## 1            1         0          7           1          1
##   stalk_surface_above_ring stalk_surface_below_ring stalk_color_above_ring
## 1                        4                        4                      3
##   stalk_color_below_ring veil_type veil_color ring_number ring_type
## 1                      3         1          3           1         6
##   spore_print_color population habitat
## 1                 1          5       7
```

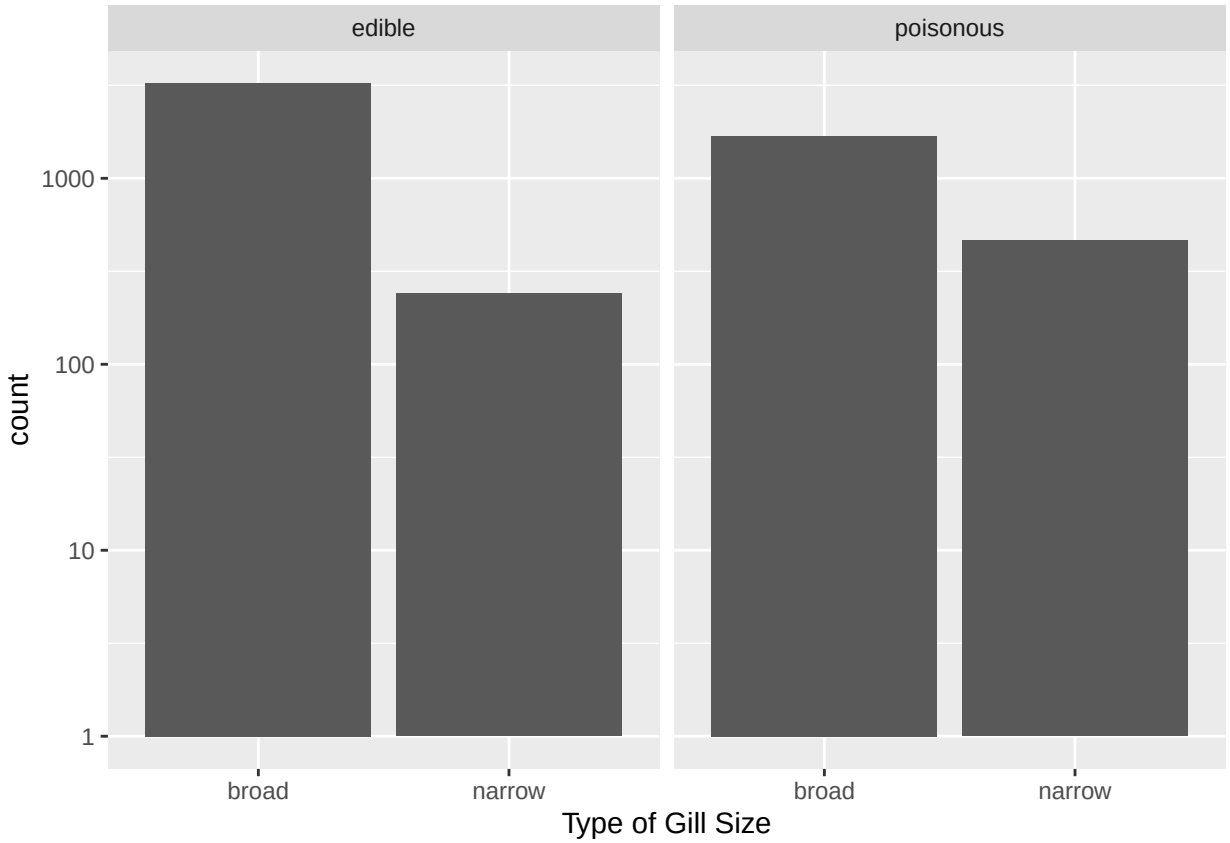
```r
# The amount of mushrooms with 1 or 2 rings are similar for both poisonous and edible types
# There are no mushrooms that are edible with 0 rings in this dataset
df |>
  ggplot(aes(x = ring_number)) +
  geom_bar() +
  facet_grid(. ~ class) +
  labs(x="Number of Rings", "Count") +
  scale_y_log10()
```



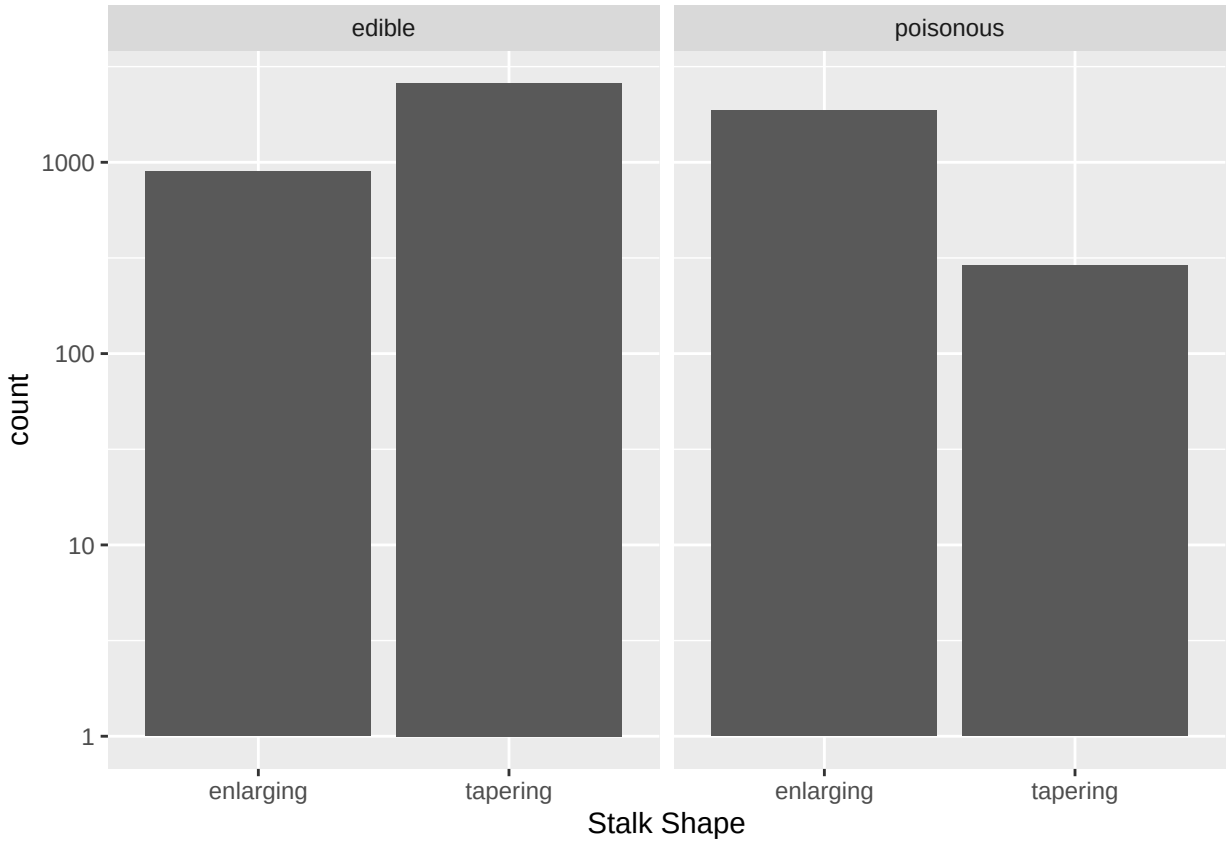

```
# Comparing the binary columns

# Poisonous mushrooms tends not to have bruises. Edible mushrooms tend to have more bruises
df |>
  ggplot(aes(x = bruises)) +
  geom_bar() +
  facet_grid(. ~ class) +
  labs(x="Bruises?", "Count") +
  scale_y_log10()
```




```
# Both poisonous and edible mushrooms have more broad gill sizes than narrow
df |>
  ggplot(aes(x = gill_size)) +
  geom_bar() +
  facet_grid(. ~ class) +
  labs(x="Type of Gill Size", "Count") +
  scale_y_log10()
```




```
# Poisonous mushrooms have more enlarging stalk shape than tapering while the opposite is true for edib
df |>
  ggplot(aes(x = stalk_shape)) +
  geom_bar() +
  facet_grid(. ~ class) +
  labs(x="Stalk Shape", "Count") +
  scale_y_log10()
```



Training Model

Splitting the Data

We will be performing a simple machine learning procedure, so the next step will be to randomly split the dataset into an 70% / 30% ratio for training and testing, respectively

```
print(cor(df_num[, c(1:23)]))
```

```
## Warning in cor(df_num[, c(1:23)]): the standard deviation is zero
```

```
##                              class   cap_shape   cap_surface    cap_color
## class                  1.000000000 -0.07442866 -0.0468588384 -0.094145965
## cap_shape             -0.074428656  1.00000000 -0.1360355590  0.055995891
## cap_surface           -0.046858838 -0.13603556  1.0000000000 -0.052383041
## cap_color             -0.094145965  0.05599589 -0.0523830408  1.000000000
## bruises               -0.435561947  0.17500456 -0.2285895204  0.035063040
## odor                   0.112626833  0.39178575 -0.1865878728  0.063718249
## gill_attachment        0.071944939 -0.04736870 -0.0581044722 -0.056427238
## gill_spacing           0.264159844  0.07020886 -0.2044475297 -0.144347669
## gill_size             -0.215288952  0.07158188 -0.0423561776 -0.122150503
## gill_color            -0.235281902  0.04460456 -0.0752318996  0.012597579
## stalk_shape            0.592445682  0.19355638 -0.1735538666  0.045344886
## stalk_root             0.271684328  0.03249354  0.0546906075 -0.361274640
## stalk_surface_above_ring  0.246214816 -0.12052939  0.0918231084  0.013669579
## stalk_surface_below_ring  0.189169644 -0.13493275  0.0082750463  0.064299207
## stalk_color_above_ring    0.052756291  0.12337621 -0.0245223940  0.176569415
## stalk_color_below_ring    0.045345317  0.11485028 -0.0234544798  0.168681357
```

```
## veil_type                          NA          NA            NA           NA
## veil_color               -0.047920723 -0.01726316  0.0387019345 -0.006054078
## ring_number              -0.008615448 -0.11327544  0.0442890890 -0.014443061
## ring_type                 0.098533305 -0.15075739  0.1973860432  0.061151616
## spore_print_color        -0.437212476  0.04884640 -0.0122651174  0.078088269
## population               -0.203882302  0.16741067 -0.0003903856  0.203156911
## habitat                   0.102083009  0.15257501 -0.0754217326  0.226197897
##                              bruises         odor gill_attachment gill_spacing
## class                    -0.43556195  0.112626833     0.071944939   0.26415984
## cap_shape                 0.17500456  0.391785745    -0.047368698   0.07020886
## cap_surface              -0.22858952 -0.186587873    -0.058104472  -0.20444753
## cap_color                 0.03506304  0.063718249    -0.056427238  -0.14434767
## bruises                   1.00000000  0.025353056    -0.064351010   0.43918287
## odor                      0.02535306  1.000000000    -0.012427493   0.12926755
## gill_attachment          -0.06435101 -0.012427493     1.000000000   0.02662965
## gill_spacing              0.43918287  0.129267549     0.026629645   1.00000000
## gill_size                 0.04017781 -0.007439447     0.021353017   0.17848372
## gill_color                0.32220573  0.034852019     0.042701547   0.08830182
## stalk_shape              -0.34829416  0.540036206     0.057738250   0.31405968
## stalk_root                0.12941254 -0.022209165    -0.005458082   0.46535646
## stalk_surface_above_ring -0.49815038 -0.084534498     0.028452169  -0.34491819
## stalk_surface_below_ring -0.42381521  0.062333938     0.082123661  -0.29486021
## stalk_color_above_ring   -0.11817807  0.228182741    -0.204688855  -0.26037155
## stalk_color_below_ring   -0.12707225  0.219691997    -0.203260804  -0.27660353
## veil_type                         NA           NA              NA           NA
## veil_color                0.04286260  0.027176919     0.002131061   0.08002589
## ring_number              -0.12216179  0.084158509     0.346681618  -0.04231552
## ring_type                -0.81265833 -0.199523522    -0.001909714  -0.69498832
## spore_print_color         0.24598095 -0.093987906     0.054523698  -0.17814453
## population               -0.19629556  0.083650188     0.115499342  -0.59591012
## habitat                  -0.32635512  0.208066175    -0.057171234  -0.38078574
##                             gill_size   gill_color stalk_shape   stalk_root
## class                    -0.215288952 -0.235281902  0.59244568  0.271684328
## cap_shape                 0.071581885  0.044604557  0.19355638  0.032493537
## cap_surface              -0.042356178 -0.075231900 -0.17355387  0.054690608
## cap_color                -0.122150503  0.012597579  0.04534489 -0.361274640
## bruises                   0.040177810  0.322205730 -0.34829416  0.129412536
## odor                     -0.007439447  0.034852019  0.54003621 -0.022209165
## gill_attachment           0.021353017  0.042701547  0.05773825 -0.005458082
## gill_spacing              0.178483716  0.088301818  0.31405968  0.465356460
## gill_size                 1.000000000 -0.107142754 -0.28236762  0.173320541
## gill_color               -0.107142754  1.000000000 -0.07447205 -0.001986807
## stalk_shape              -0.282367616 -0.074472052  1.00000000 -0.046273909
## stalk_root                0.173320541 -0.001986807 -0.04627391  1.000000000
## stalk_surface_above_ring  0.169684968 -0.226252875 -0.01342712 -0.121060618
## stalk_surface_below_ring  0.192483604 -0.161637300  0.06657057 -0.325789753
## stalk_color_above_ring   -0.208079519 -0.052139021  0.17725580 -0.321906677
## stalk_color_below_ring   -0.229419833 -0.043913951  0.18456601 -0.313490282
## veil_type                         NA           NA           NA           NA
## veil_color                0.099801403 -0.028442432 -0.03845800  0.003635492
## ring_number              -0.033930758 -0.026291139 -0.09174828 -0.082381544
## ring_type                 0.187912486 -0.298049579 -0.01184888 -0.356232188
## spore_print_color        -0.152695961  0.178614734 -0.21201900 -0.200204311
## population               -0.008851112 -0.062171035 -0.09959825 -0.520161281
```

```
## habitat                   0.070147202 -0.164714443  0.25984857 -0.539517617
##                         stalk_surface_above_ring stalk_surface_below_ring
## class                                0.24621482              0.189169644
## cap_shape                           -0.12052939             -0.134932745
## cap_surface                          0.09182311              0.008275046
## cap_color                            0.01366958              0.064299207
## bruises                             -0.49815038             -0.423815211
## odor                                -0.08453450              0.062333938
## gill_attachment                      0.02845217              0.082123661
## gill_spacing                        -0.34491819             -0.294860208
## gill_size                            0.16968497              0.192483604
## gill_color                          -0.22625288             -0.161637300
## stalk_shape                         -0.01342712              0.066570573
## stalk_root                          -0.12106062             -0.325789753
## stalk_surface_above_ring             1.00000000              0.421548925
## stalk_surface_below_ring             0.42154893              1.000000000
## stalk_color_above_ring               0.12810174              0.138633257
## stalk_color_below_ring               0.13610770              0.145497456
## veil_type                                   NA                       NA
## veil_color                          -0.05996394             -0.054700515
## ring_number                          0.05729046              0.102365643
## ring_type                            0.57772556              0.504202676
## spore_print_color                   -0.19065556             -0.153964654
## population                           0.24686841              0.235524299
## habitat                              0.35731950              0.385854958
##                         stalk_color_above_ring stalk_color_below_ring
## class                              0.052756291           0.0453453170
## cap_shape                          0.123376210           0.1148502772
## cap_surface                       -0.024522394          -0.0234544798
## cap_color                          0.176569415           0.1686813573
## bruises                           -0.118178072          -0.1270722464
## odor                               0.228182741           0.2196919974
## gill_attachment                   -0.204688855          -0.2032608038
## gill_spacing                      -0.260371547          -0.2766035289
## gill_size                         -0.208079519          -0.2294198325
## gill_color                        -0.052139021          -0.0439139508
## stalk_shape                        0.177255802           0.1845660131
## stalk_root                        -0.321906677          -0.3134902815
## stalk_surface_above_ring           0.128101743           0.1361076996
## stalk_surface_below_ring           0.138633257           0.1454974564
## stalk_color_above_ring             1.000000000           0.3711363006
## stalk_color_below_ring             0.371136301           1.0000000000
## veil_type                                   NA                    NA
## veil_color                        -0.001324077          -0.0009802644
## ring_number                       -0.228806015          -0.2258631047
## ring_type                          0.160300478           0.1764519571
## spore_print_color                 -0.003703535           0.0048091508
## population                         0.256794339           0.2524786093
## habitat                            0.382121322           0.3862637473
##                         veil_type    veil_color  ring_number    ring_type
## class                          NA -0.0479207229 -0.008615448  0.098533305
## cap_shape                      NA -0.0172631588 -0.113275436 -0.150757395
## cap_surface                    NA  0.0387019345  0.044289089  0.197386043
## cap_color                      NA -0.0060540784 -0.014443061  0.061151616
```

```
## bruises                          NA  0.0428625966 -0.122161788 -0.812658330
## odor                             NA  0.0271769190  0.084158509 -0.199523522
## gill_attachment                  NA  0.0021310610  0.346681618 -0.001909714
## gill_spacing                     NA  0.0800258896 -0.042315521 -0.694988316
## gill_size                        NA  0.0998014029 -0.033930758  0.187912486
## gill_color                       NA -0.0284424317 -0.026291139 -0.298049579
## stalk_shape                      NA -0.0384580027 -0.091748279 -0.011848875
## stalk_root                       NA  0.0036354917 -0.082381544 -0.356232188
## stalk_surface_above_ring         NA -0.0599639356  0.057290457  0.577725560
## stalk_surface_below_ring         NA -0.0547005152  0.102365643  0.504202676
## stalk_color_above_ring           NA -0.0013240770 -0.228806015  0.160300478
## stalk_color_below_ring           NA -0.0009802644 -0.225863105  0.176451957
## veil_type                         1            NA           NA           NA
## veil_color                       NA  1.0000000000 -0.003386337 -0.075648421
## ring_number                      NA -0.0033863372  1.000000000  0.090418953
## ring_type                        NA -0.0756484213  0.090418953  1.000000000
## spore_print_color                NA -0.0363168703 -0.001725182 -0.123749162
## population                       NA -0.0769312205  0.132167949  0.443578638
## habitat                          NA -0.0331436154 -0.103343759  0.523897696
##                          spore_print_color    population     habitat
## class                      -0.437212476 -0.2038823018  0.10208301
## cap_shape                   0.048846402  0.1674106717  0.15257501
## cap_surface                -0.012265117 -0.0003903856 -0.07542173
## cap_color                   0.078088269  0.2031569106  0.22619790
## bruises                     0.245980947 -0.1962955617 -0.32635512
## odor                       -0.093987906  0.0836501881  0.20806618
## gill_attachment             0.054523698  0.1154993419 -0.05717123
## gill_spacing               -0.178144526 -0.5959101225 -0.38078574
## gill_size                  -0.152695961 -0.0088511121  0.07014720
## gill_color                  0.178614734 -0.0621710354 -0.16471444
## stalk_shape                -0.212018999 -0.0995982471  0.25984857
## stalk_root                 -0.200204311 -0.5201612807 -0.53951762
## stalk_surface_above_ring   -0.190655556  0.2468684140  0.35731950
## stalk_surface_below_ring   -0.153964654  0.2355242987  0.38585496
## stalk_color_above_ring     -0.003703535  0.2567943390  0.38212132
## stalk_color_below_ring      0.004809151  0.2524786093  0.38626375
## veil_type                            NA            NA          NA
## veil_color                 -0.036316870 -0.0769312205 -0.03314362
## ring_number                -0.001725182  0.1321679488 -0.10334376
## ring_type                  -0.123749162  0.4435786385  0.52389770
## spore_print_color           1.000000000  0.1602302778 -0.07083490
## population                  0.160230278  1.0000000000  0.51657302
## habitat                    -0.070834902  0.5165730219  1.00000000
```

```
#install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
```

```
##     lift
```

```
#install.packages("brglm2")
library(brglm2)
```

```
## Warning: package 'brglm2' was built under R version 4.3.2
```

```
#Setting a seed to keep the outcomes consistent
set.seed(12345)

index <- createDataPartition(df_num$class, p = .70, list = F)

train <- df_num[index,]
test <- df_num[-index,]
```

Training the Model

A logistic regression model was developed to predict whether a mushroom is edible or poisonous based on various features in the dataset. Logistic regression is a popular method for binary classification.

First, we created the logistic regression model, specifying the target variable (edible or poisonous) as a function of the predictor variables. These predictors were `cap_shape`, `cap_color`, `gill_size`, and others. The training dataset, which consisted of 70% of the total data, was used to fit the model. This subset provided a substantial amount of data for the model to learn the patterns and relationships between the features and the target variable.

Once the model was defined and the data was prepared, the next step involved training the model using the `glm()` function in R, specifying the binomial family to denote a logistic regression. During this training phase, the model learned the coefficients for each predictor, adjusting them to best fit the training data.

```
#Function to detect binary columns
# is_bin <- function(train) {
#   binary_col <- sapply(train, function(column) {
#     all(column %in% c(0, 1))
#   })
#   return(binary_col)
# }
#
# binary_col <- is_bin(train)
#
# print(binary_col)

str(train)
```

```
## 'data.frame':    3951 obs. of  23 variables:
##  $ class                 : num  0 1 1 1 0 1 1 0 1 1 ...
##  $ cap_shape             : num  3 3 1 3 3 1 1 3 3 6 ...
##  $ cap_surface           : num  3 3 3 3 4 3 3 4 1 1 ...
##  $ cap_color             : num  1 4 3 6 3 4 4 3 1 6 ...
##  $ bruises               : num  0 0 0 1 0 0 0 0 1 1 ...
##  $ odor                  : num  8 1 2 7 8 1 1 8 7 7 ...
##  $ gill_attachment       : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ gill_spacing          : num  1 1 1 2 1 1 1 1 2 1 ...
##  $ gill_size             : num  1 0 0 0 1 0 0 1 0 1 ...
##  $ gill_color            : num  12 12 1 12 7 6 3 12 1 12 ...
##  $ stalk_shape           : num  0 0 0 1 0 0 0 0 1 0 ...
##  $ stalk_root            : num  4 2 2 4 4 2 2 4 4 4 ...
```

```
##  $ stalk_surface_above_ring: num  4 4 4 4 4 4 4 4 4 4 ...
##  $ stalk_surface_below_ring: num  4 4 4 4 4 4 4 4 1 4 ...
##  $ stalk_color_above_ring  : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk_color_below_ring  : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ veil_type               : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ veil_color              : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ ring_number             : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ring_type               : num  6 6 6 2 6 6 6 6 2 6 ...
##  $ spore_print_color       : num  12 1 1 1 12 12 1 1 12 1 ...
##  $ population              : num  4 3 3 1 5 4 4 5 1 6 ...
##  $ habitat                 : num  5 1 3 1 1 3 1 5 1 5 ...
```

```
#The model properly converges when gill size and veil type are removed; but note that a warning is stil
train_model <- glm(class ~ cap_shape + cap_surface + cap_color + bruises + odor + gill_attachment +
               gill_spacing + gill_color + stalk_shape + stalk_root +
               stalk_surface_above_ring + stalk_surface_below_ring +
               stalk_color_above_ring + stalk_color_below_ring + veil_color + ring_number +
               ring_type + spore_print_color + population + habitat, data = train, family = binomial)
```

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

```
summary(train_model)
```

```
##
## Call:
## glm(formula = class ~ cap_shape + cap_surface + cap_color + bruises +
##     odor + gill_attachment + gill_spacing + gill_color + stalk_shape +
##     stalk_root + stalk_surface_above_ring + stalk_surface_below_ring +
##     stalk_color_above_ring + stalk_color_below_ring + veil_color +
##     ring_number + ring_type + spore_print_color + population +
##     habitat, family = binomial, data = train)
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                65.79078 1826.56649   0.036 0.971267
## cap_shape                  -0.28985    0.10687  -2.712 0.006682 **
## cap_surface                -0.23494    0.07232  -3.248 0.001160 **
## cap_color                  -0.00187    0.03654  -0.051 0.959179
## bruises                    -1.48213    0.45000  -3.294 0.000989 ***
## odor                       -3.31584    0.20108 -16.490  < 2e-16 ***
## gill_attachment            -4.15344  207.75505  -0.020 0.984050
## gill_spacing                2.80895    0.50471   5.565 2.61e-08 ***
## gill_color                 -0.14785    0.02248  -6.578 4.78e-11 ***
## stalk_shape                18.33905    1.06766  17.177  < 2e-16 ***
## stalk_root                  6.86953    0.44158  15.557  < 2e-16 ***
## stalk_surface_above_ring    1.80216    0.15274  11.799  < 2e-16 ***
## stalk_surface_below_ring    1.75452    0.15372  11.413  < 2e-16 ***
## stalk_color_above_ring      0.31952    0.06249   5.113 3.16e-07 ***
## stalk_color_below_ring      0.24207    0.05931   4.082 4.47e-05 ***
## veil_color                -20.16103  572.31316  -0.035 0.971899
## ring_number                19.83821    1.34213  14.781  < 2e-16 ***
## ring_type                  -6.23461    0.39974 -15.597  < 2e-16 ***
## spore_print_color          -0.23299    0.02296 -10.146  < 2e-16 ***
## population                  0.85217    0.16473   5.173 2.30e-07 ***
## habitat                     0.82870    0.08477   9.776  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5260.6  on 3950  degrees of freedom
## Residual deviance: 1133.0  on 3930  degrees of freedom
## AIC: 1175
##
## Number of Fisher Scoring iterations: 14
```

Evaluate Training Data

We can now use the model to make predictions on the training data to evaluate its performance.

The results from the confusion matrix and cross-validation provided insights into how well the model was performing. They offered a detailed look at the model's strengths and weaknesses in classifying the mushrooms as edible or poisonous, based on the training data. This evaluation phase was critical for understanding the efficacy of the logistic regression model before proceeding to test it on unseen data.

```r
#This code will generate predictions
train_pred <- predict(train_model, type = "response", newdata = train)
train_pred_class <- ifelse(train_pred > 0.5, 1, 0)

#And now we can assess the acuracy of those predictions
confusionMatrix(factor(train_pred_class), factor(train$class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1418   60
##          1   97 2376
##
##                Accuracy : 0.9603
##                  95% CI : (0.9537, 0.9661)
##     No Information Rate : 0.6166
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9156
##
##  Mcnemar's Test P-Value : 0.004064
##
##             Sensitivity : 0.9360
##             Specificity : 0.9754
##          Pos Pred Value : 0.9594
##          Neg Pred Value : 0.9608
##              Prevalence : 0.3834
##          Detection Rate : 0.3589
##    Detection Prevalence : 0.3741
##       Balanced Accuracy : 0.9557
##
##        'Positive' Class : 0
##
```

```r
#As we can see, our training data are able to predict edibility outcomes with a 96% accuracy

#And now we can cross-validate the data
cv_results <- train(class ~ cap_shape + cap_surface + cap_color + bruises + odor + gill_attachment +
               gill_spacing + gill_color + stalk_shape + stalk_root +
               stalk_surface_above_ring + stalk_surface_below_ring +
               stalk_color_above_ring + stalk_color_below_ring + veil_color + ring_number +
               ring_type + spore_print_color + population + habitat, data = train, method = "glm",
               trControl = trainControl(method = "cv", number = 10), family = 'binomial')
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.

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

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

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

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

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

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

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

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

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

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

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

```r
print(cv_results)
```

```
## Generalized Linear Model
##
## 3951 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3556, 3555, 3556, 3556, 3556, 3556, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1873762  0.8516147  0.07693206
```

```r
#As we can see, the RMSE value is low, whereas the R^2 is high, indicating that the model is likely a g
```

- **Confusion Matrix**:
  - **True Negatives (1418)**: The model correctly identified 1418 mushrooms as non-poisonous

(edible).
  - **False Positives (60)**: It mistakenly tagged 60 mushrooms as non-poisonous when they were actually poisonous.
  - **False Negatives (97)**: It incorrectly labeled 97 poisonous mushrooms as edible.
  - **True Positives (2376)**: And it got it right with 2376 mushrooms, accurately identifying them as poisonous.

- **Model Performance**:
  - **Accuracy:** 0.9603 or 96.03%
  - **95% Confidence Interval:** 0.9537 to 0.9661
  - **No Information Rate: 0.6166** meaning if we just guessed the most common outcome every time, we'd be right about 61.66% of the time.
  - **Kappa:** 0.9156
  - **Sensitivity:** 0.9360 meaning 93.6% of the time, the model correctly identifies poisonous mushrooms.
  - **Specificity:** 0.9754
  - **Positive Predictive Value :** 0.9594meaning it predicts a mushroom is edible, it's correct about 95.94% of the time.
  - **Negative Predictive Value:** 0.9608 which means that when it predicts a mushroom is poisonous, it's correct about 96.08% of the time.

According to the results, the model is doing a good job at classifying mushrooms as edible or poisonous.

Predictions for Our Test Data

The test dataset, which comprised 30% of the entire dataset, was used for this purpose. It contained the same features as the training set but had not been used during the model training phase. The model made predictions on this test data, estimating whether each mushroom was edible or poisonous based on the learned patterns from the training data.

```
#This code will apply the model to our test dataset
test_pred <- predict(train_model, type = "response", newdata = test)

test_pred_class <- ifelse(test_pred > 0.5, 1,0)

#This code will check for accuracy
confusionMatrix(factor(test_pred_class), factor(test$class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  596   26
##          1   45 1026
##
##                Accuracy : 0.9581
##                  95% CI : (0.9474, 0.9671)
##     No Information Rate : 0.6214
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9104
##
##  Mcnemar's Test P-Value : 0.03266
##
##             Sensitivity : 0.9298
```

```
##             Specificity : 0.9753
##          Pos Pred Value : 0.9582
##          Neg Pred Value : 0.9580
##              Prevalence : 0.3786
##          Detection Rate : 0.3520
##    Detection Prevalence : 0.3674
##       Balanced Accuracy : 0.9525
##
##        'Positive' Class : 0
##
```

```
##As we can see, the results are rather consistent, with a 95.8% accuracy for predicting edibility

library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.2
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

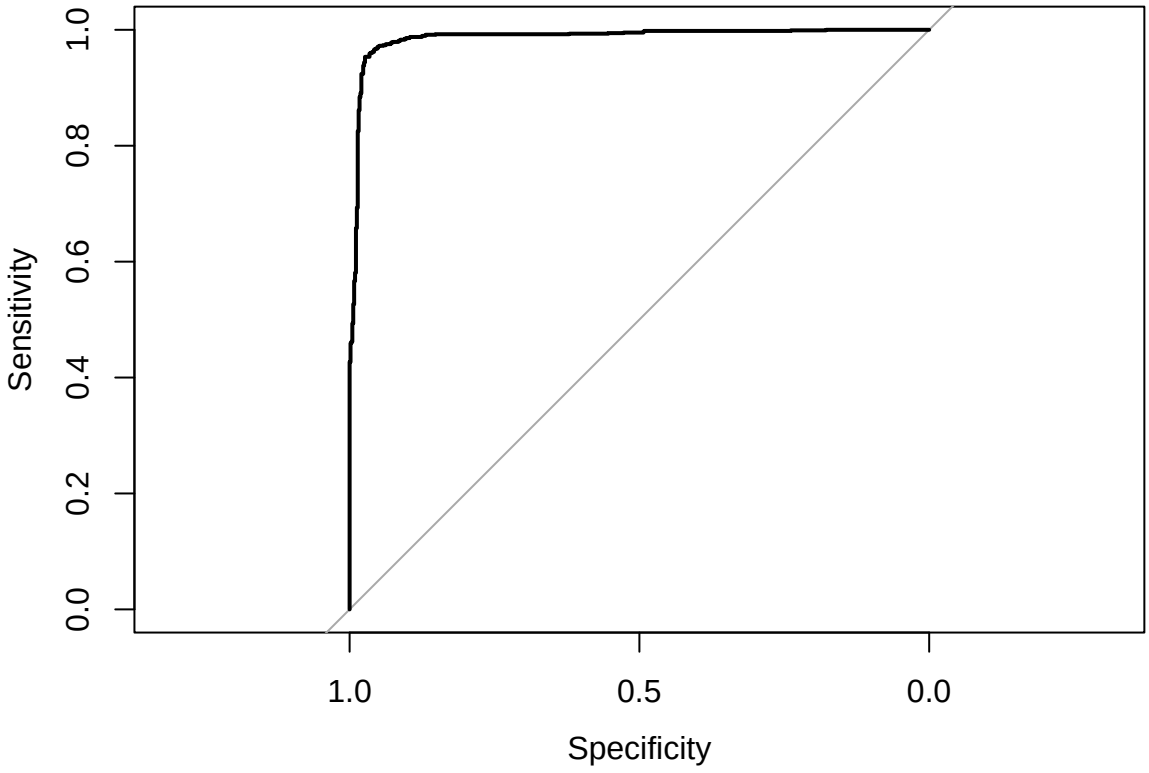
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#The following code will calculate the Area under the Curve
rocCurve <- roc(test$class, test_pred)
```

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

```
## Setting direction: controls < cases
```

```
plot(rocCurve)
```




```
auc(rocCurve)
```

```
## Area under the curve: 0.9867
```

```
#The plot and diagnostics demonstrate the high predictive accuracy of our model
```

- **Confusion Matrix**:
  - `True Negatives (0,0)`: 596 - The model correctly predicted 596 instances where mushrooms were not poisonous (edible).
  - `False Positives (0,1)`: 26 - The model incorrectly predicted 26 instances as non-poisonous (edible) when they were actually poisonous.
  - `False Negatives (1,0)`: 45 - The model incorrectly predicted 45 instances as poisonous when they were actually non-poisonous (edible).
  - `True Positives (1,1)`: 1026 - The model correctly predicted 1026 instances where mushrooms were poisonous.
- **Accuracy**: Our model was correct for 95.81% (0.9581) of the mushrooms in the test set. We are 95% confident that it lies in this range (0.9474, 0.9671).
- **No Information Rate**: 0.6214.This means if we were to guess the most common class for every mushroom, we'd be right about 62.14% of the time.
- **P-Value**: $<$ 2e-16 - This p-value is extremely low. Our model is statistically significant
- **Kappa**: 0.9104 - Since the kappa is close to one, we know that our model is effective.
- **Sensitivity and Specificity**:
  - Sensitivity (True Positive Rate): 0.9298 - About 93% of poisonous mushrooms were correctly identified.

  - Specificity (True Negative Rate): 0.9753 - About 97.53% of non-poisonous mushrooms were correctly identified.

- **Predictive Values**:
  - Positive Predictive Value: 0.9582 - When the model predicts a mushroom is non-poisonous, it is correct about 95.82% of the time.
  - Negative Predictive Value: 0.9580 - When the model predicts a mushroom is poisonous, it is correct about 95.80% of the time.
- **Prevalence**: 0.3786 - 37.86% of the mushrooms in the test set were actually non-poisonous (edible).
- **Detection Rate**: 0.3520 - 35.20% of all mushrooms in the test set were correctly identified as non-poisonous by the model.
- **Detection Prevalence**: 0.3674 - 36.74% of all mushrooms were predicted as non-poisonous by the model.
- **Area Under the Curve (AUC)**: 0.9867 - This is very close to 1, meaning the model has good ability to differentiate between edible and poisonous mushrooms.

Testing

In UCI's Machine Learning Repository, there is a secondary mushroom dataset with more data (but new entries). I will import this new data and test the accuracy of our logistic regression model for identifing whether or not mushrooms are poisonous or edible.

Secondary Mushroom Data Preprocessing

Before we can use our test data, we must prepare it so that it is in the same format of the data used to train our linear model. Since not all columns included in our original model is in the secondary data set, we either have to create columns and estimate the data or train a new model without those columns. We chose to train a new model that will be used to evaluate the new dataset.

```
secondary_dat=read.csv(url("https://raw.githubusercontent.com/Mattr5541/DATA_607_Project-4/main/mushroo

names(secondary_dat)
```

```
## [1] "class.cap.diameter.cap.shape.cap.surface.cap.color.does.bruise.or.bleed.gill.attachment.gill.spa
```

```
col_names_sec = c('class','cap_diameter', 'cap_shape', 'cap_surface', 'cap_color', 'bruises',  'gill_at

split_data = str_split_fixed(secondary_dat$class.cap.diameter.cap.shape.cap.surface.cap.color.does.bruis

test_2_df=as.data.frame(split_data)

names(test_2_df)=col_names_sec



test_2_df$class = as.numeric(factor(test_2_df$class, levels=c('p','e')))

test_2_df$cap_shape = as.numeric(factor(test_2_df$cap_shape, levels=c('b', 'c', 'x', 'f', 'k', 's')))



test_2_df$cap_color = as.numeric(factor(test_2_df$cap_color, levels=c("n","o","w","y","b","g","p","e","
```

```r
test_2_df$bruises = as.numeric(factor(test_2_df$bruises, levels=c('t', 'f')))

test_2_df$gill_attachment = as.numeric(factor(test_2_df$gill_attachment, levels=c('a','d','e', 'f')))


test_2_df$gill_color = as.numeric(factor(test_2_df$gill_color, levels=c("n","o","w","y","b","g","p","e"


test_2_df$veil_type = as.numeric(factor(test_2_df$veil_type, levels=c('p', 'u')))


test_2_df$veil_color = as.numeric(factor(test_2_df$veil_color, levels=c("n","o","w","y","b","g","p","e"


test_2_df$habitat = as.numeric(factor(test_2_df$habitat, levels=c('g', 'l', 'm', 'p', 'u', 'w', 'd')))
```

```r
test_2 = test_2_df %>%
  select(c(class,`cap_shape`,`cap_color`,bruises,`gill_attachment`, `gill_color`, `veil_type`,`veil_colo

train_model_2 = glm(class ~ cap_shape  + cap_color + bruises +  gill_attachment + gill_color +veil_type+
                habitat, data = train, family = binomial)


summary(train_model_2)
```

```
##
## Call:
## glm(formula = class ~ cap_shape + cap_color + bruises + gill_attachment +
##     gill_color + veil_type + veil_color + habitat, family = binomial,
##     data = train)
##
## Coefficients: (1 not defined because of singularities)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      2.454e+01  1.144e+03   0.021   0.9829
## cap_shape        2.748e-02  5.095e-02   0.539   0.5896
## cap_color       -8.603e-02  1.686e-02  -5.102 3.37e-07 ***
## bruises         -1.699e+00  8.255e-02 -20.577  < 2e-16 ***
## gill_attachment  7.090e+00  1.258e+02   0.056   0.9551
## gill_color      -7.867e-02  9.467e-03  -8.309  < 2e-16 ***
## veil_type               NA         NA      NA       NA
## veil_color      -1.452e+01  3.601e+02  -0.040   0.9678
## habitat         -2.901e-02  1.561e-02  -1.859   0.0631 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 5260.6  on 3950  degrees of freedom
## Residual deviance: 4433.2  on 3943  degrees of freedom
## AIC: 4449.2
##
## Number of Fisher Scoring iterations: 13
```

```
#generate predictions
train_pred_2 = predict(train_model_2, type = "response", newdata = train)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
train_pred_class_2 = ifelse(train_pred_2 > 0.5, 1, 0)

#assess the acuracy of those predictions
confusionMatrix(factor(train_pred_class_2), factor(train$class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1049  464
##          1  466 1972
##
##                Accuracy : 0.7646
##                  95% CI : (0.7511, 0.7778)
##     No Information Rate : 0.6166
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5021
##
##  Mcnemar's Test P-Value : 0.9738
##
##             Sensitivity : 0.6924
##             Specificity : 0.8095
##          Pos Pred Value : 0.6933
##          Neg Pred Value : 0.8089
##              Prevalence : 0.3834
##          Detection Rate : 0.2655
##    Detection Prevalence : 0.3829
##       Balanced Accuracy : 0.7510
##
##        'Positive' Class : 0
##
```

```
cv_results_2 = train(class ~ cap_shape + cap_color + bruises + odor + gill_attachment + gill_color + ve
               trControl = trainControl(method = "cv", number = 10), family = 'binomial')
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
print(cv_results_2)
```

```
## Generalized Linear Model
##
## 3951 samples
```

```
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3556, 3556, 3556, 3556, 3556, 3556, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.4179356  0.2624879  0.3584861
```

The second model trained was a less accurate than the first one. But this was expected since the secondary dataset did not the complete set of columns as our original dataset. Lets evaluate this second model on our secondary test data.

confusionMatrix(factor(test_pred_class_2), factor(test_2$class))

```
test_pred_2 = predict(train_model_2, type = "response", newdata = test_2)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
test_pred_class_2 = ifelse(test_pred_2 > 0.5, 2, 1)

test_pred_class_2 = factor(test_pred_class_2, levels = c(1, 2))
test_2$class = factor(test_2$class, levels = c(1, 2))


confusionMatrix(test_pred_class_2, test_2$class)
```

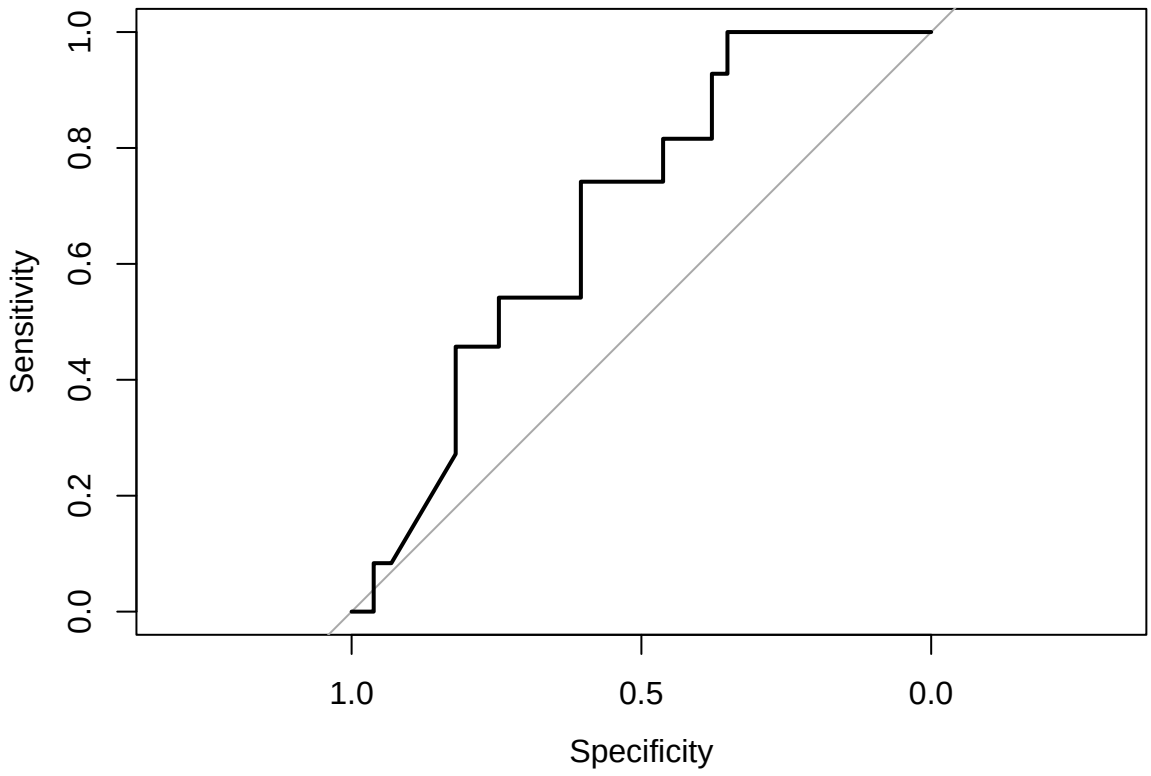
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1 2400  945
##          2   95    0
##
##                Accuracy : 0.6977
##                  95% CI : (0.682, 0.713)
##     No Information Rate : 0.7253
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : -0.0528
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9619
##             Specificity : 0.0000
##          Pos Pred Value : 0.7175
##          Neg Pred Value : 0.0000
##              Prevalence : 0.7253
##          Detection Rate : 0.6977
##    Detection Prevalence : 0.9724
##       Balanced Accuracy : 0.4810
##
##        'Positive' Class : 1
##
```

```
rocCurve_2 = roc(test_2$class, test_pred_2)
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
plot(rocCurve_2)
```




```
auc(rocCurve_2)
```

```
## Area under the curve: 0.6834
```

1. **True Positives (2400)**: The model correctly predicted 2400 instances as 'poisonous' mushrooms.
2. **True Negatives (0)**: The model did not correctly predict any instances of 'edible' mushrooms.
3. **False Positives (945)**: The model incorrectly classified 945 instances as 'poisonous' , which were actually 'edible'.
4. **False Negatives (95)**: The model incorrectly classified 95 instances as 'edible', which were actually 'poisonous'.

Model Performance Metrics

- **Accuracy (69.77%)**: The model correctly predicted about 70% of the instances. However, this is below the No Information Rate, suggesting that the model might not be performing better than random guessing for this dataset.
- **Kappa (-0.0528)**: This means that the model is not suitable for this dataset.

- **Sensitivity (96.19%)**: The model is highly sensitive in predicting poisonous mushrooms but fails significantly in predicting edible mushrooms, as indicated by a specificity of 0%.
- **Specificity (0%)**: The model fails to correctly identify any true negatives (edible mushrooms)
- **Positive Predictive Value (71.75%)**: When the model predicts poisonous mushrooms, it is correct about 72% of the time.

Our secondary model was not as accurate for predicting whether the secondary dataset is poisonous or not. In the future, I would consider using more columns and adding the missing values to my new dataset (either by inserting the mean, or individually mushroom by mushroom getting that data point). The more data a model has to train on, the better the model will get ideally.

Conclusion

In this project, we implemented and evaluated a logistic regression model to classify mushrooms as edible or poisonous,

The logistic regression model, trained on the training set, demonstrated effective learning of the relationships between various mushroom features and their edibility status. The model's performance was initially evaluated on the training data using a confusion matrix and cross-validation, showing promising results.

Our logistic regression model had good results, with high accuracy, sensitivity, and specificity. These outcomes suggest that the model effectively distinguishes between edible and poisonous mushrooms, aligning well with known characteristics of these fungi. Notably, features like odor, gill color, and cap shape have emerged as significant predictors.

There are some results that we should explore in the future. For instance, the model's occasional confusion between certain classes of mushrooms, as evidenced by false positives and negatives, hint at smaller morphological or chemical similarities that aren't captured by the dataset. In the future, we could explore more complex models, like neural networks, to capture complex patterns. Integrating datasets from different geographic and environmental data could enhance the model's applicability to diverse real-world scenarios.