Mushroom Editability Prediction Project

Kenzie Baker, Alex Hromada

MA 5790

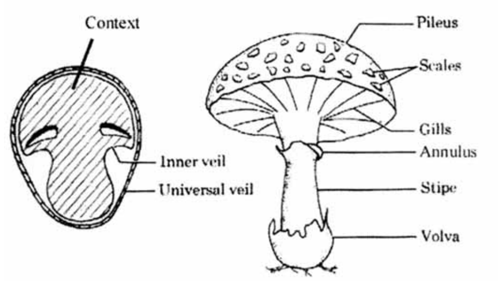
December 2023

Abstract:

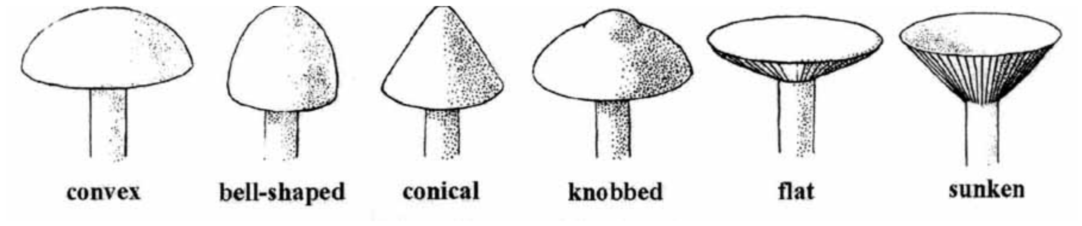
Mushrooms have become more popular than ever in recent years. They are being celebrated for their culinary versatility, nutritional benefits, mental health benefits, and much more. This upsurge of interest in mushrooms has led to many enthusiasts to take foraging into their own hands. This is empowering some consumers by being able to forage and provide for themselves as well as providing a food source for rural populations. This does not come without a risk; the challenge of identifying edible or poisonous mushrooms. There are approximately 100,000 mushroom species across the world, with more than 100 species being poisonous to humans (Seneca, 2023).

This paper aims to leverage a comprehensive set of mushroom traits encompassing cap characteristics, gill properties, stem features, and more, this research endeavors to develop a predictive model that discerns the crucial distinction between 'edible' and 'poisonous' mushrooms. The relationship between these predictors will be explored along with separation of degenerate variables from the set of predictors. Following this, a variety of classification models will be fit to the training data using cross validation training methods. The top models will be used to predict on the test set and the overall best model will then be selected. The next page allows for context of the different features of mushrooms that will be used to predict the edibility.

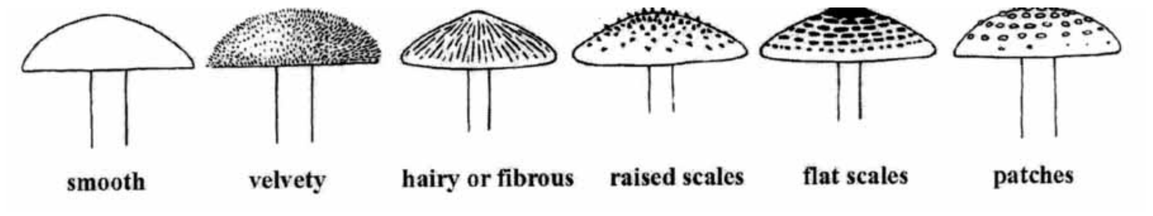
Mushroom Structure:



Mushroom Cap Shape:



Mushroom Cap Surface:



Mushroom Stem Types:

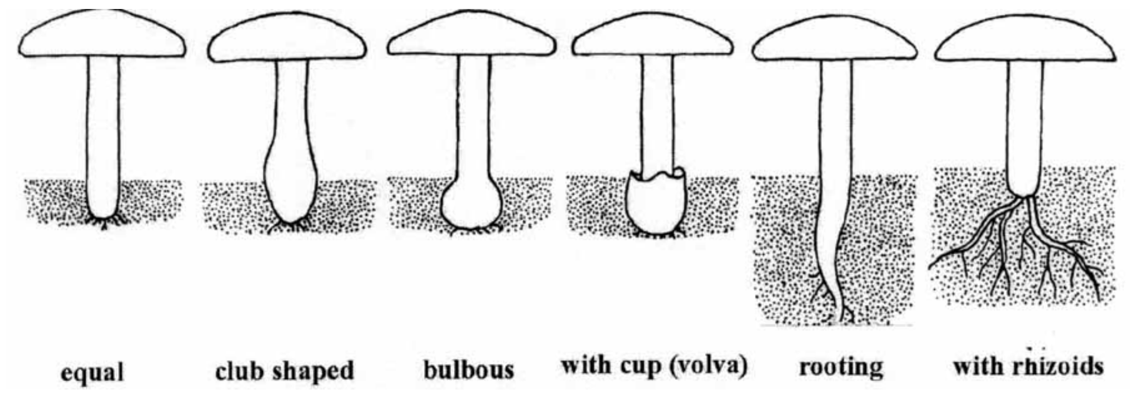


Table of Contents

Background:

Variable Introduction and Definitions:

This dataset includes 61069 hypothetical mushrooms with caps based on 173 species (353 mushrooms per species). Each mushroom is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended (the latter class was combined with the poisonous class). Of the 20 variables, 17 are nominal and 3 are metrical. This different species of mushrooms came from Collins Gem Mushrooms & Toadstools by Patrick Harding in 1999. The original data was donated to UC Irvine in 1987. This data set was then expanded with hypothetical mushroom entries in order to make a data set to use for predictive models.

Variable Names	Variable Description	Variable Values
class	Edible or poisonous	e = Edible, p = poisonous
cap-shape	The shape of the expanded, upper part of the mushroom	b= bell, c = conical, x = convex, f = flat, k = knobbed, s = sunken
cap-diameter	The measured diameter of the upper part of the mushroom	float number in cm
cap-surface	The structure of the upper part of the mushroom	f = fibrous, g = grooves, y = scaly, s = smooth
cap-color	The color of the surface of the upper part of the mushroom	n = brown, b = buff, c = cinnamon, g = gray, r = green, p = pink, u = purple, e = red, w = white, y = yellow
does-bruise-or-bleed	Indicates of there are bruises on the mushroom	t = yes, f = no
gill-attachment	The way the gill is growing on the mushroom	adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?
gill-spacing	The gap of space between each gill	c = close,w = crowded, d = distant
gill-color	The color of the gills	k = black, n = brown, b = buff, h = chocolate, g = gray, r = green, o = orange, p = pink, u = purple, e = red, w = white , y = yellow
stem-height	The height of the root of the mushroom	float number in cm
stem-width	The width of the root of the mushroom	float number in mm

stem-root	The root of the mushroom	bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, unknown=?
stem-surface	The surface of the root of the mushroom	see cap-surface + none=f
stem-color	The color of the root of the mushroom	see cap-color + none=f
veil-type	The type of the mushroom`s veil	p = partial, u = universal
veil-color	The color of the mushroom`s veil	n = brown, o = orange, w = white, y = yellow
has-ring	If there are rings on the mushroom	ring=t, none=f
ring-type	The type of the mushroom`s ring	c = cobwebby, e = evanescent, f = flaring, l = large ,n = none, p = pendant, s = sheathing, z = zone
spore-print-color	The color of the mushrooms spore	k = black, n = brown, b = buff, h = chocolate, r = green, o = orange, u = purple, w = white , y = yellow
season	The time of year the mushroom grows	spring=s, summer=u, autumn=a, winter=w
habitat	The mushroom`s environment	g = grasses, l = leaves, m = meadows, p = paths, u = urban, w = waste, d = woods

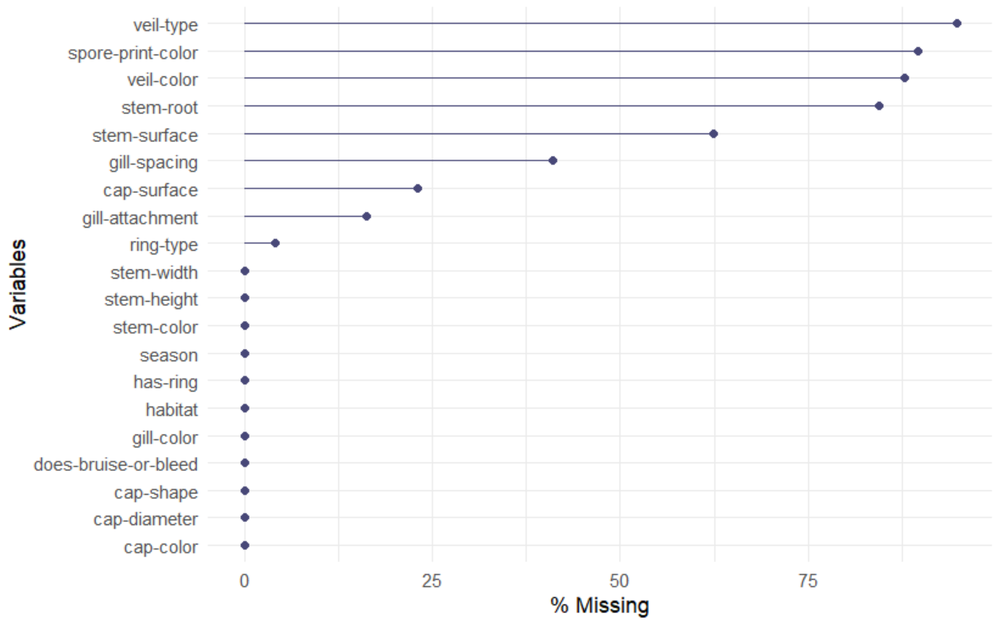
Based on the selection of predictor variables, the relationship between predictors and the editability of given mushroom samples will be analyzed. The data will be preprocessed, including any sort of transformations or elimination of predictors needed. Following this, both linear and nonlinear classification models will be applied attempting to predict the edibility outcome as stated above.

Pre-processing of the predictors:

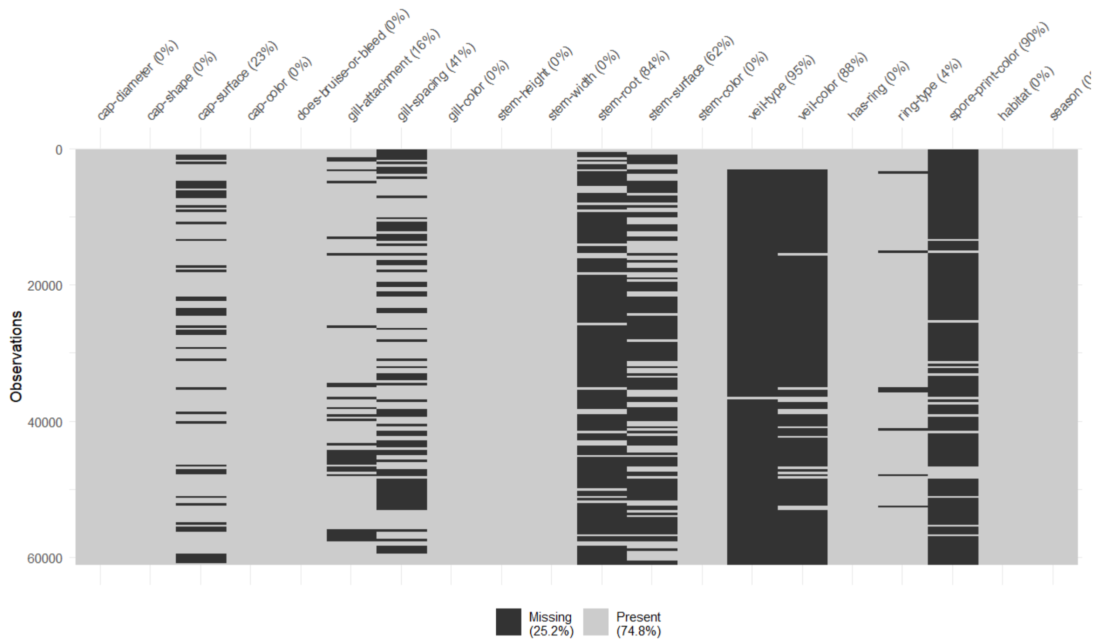
The first step in analyzing this data is to preprocess the data. The first step is to check for any missing values or degenerate distributions. Then dummy variables must be created for the categorical predictors.

Missing Values:

This data set has a significant amount of missing values. Across the entire data set there is 25.17341% of the data missing. Below is each predictor with their corresponding percentage of missing values.Predictors with greater than 50% of the data missing were then removed. It is important to take caution when removing predictors, but in this data set these values would have to be populated with the mode since they are categorical. Populating 50% of the data with the mode of the data could yield unreliable and inaccurate results, so they are removed.



In order to fully understand where these missing values lie, the percent of missing values across all observations can be seen below. This shows that the missing values are correlated to specific predictors and not specific observations. Therefore we can proceed with the removal of these predictors.



Based on the threshold of 0.5, five predictors were removed: "stem-root", "stem-surface","veil-type","veil-color", and "spore-print-color". This reduced the dimensions of the data set from 61069x20 to 61069x15. Their respective percentage of missing values is below.

Stem Root Percent Missing: 84%	Stem Surface Percent Missing: 62%	Veil Type Percent Missing: 95%	Veil Color Percent Missing: 88%	Spore Print Color Percent Missing: 90%

The removal of these predictors reduced the percent missing from 25.17341% to 5.626204%. The remaining missing values are all within the categorical variables, so we proceed with mode imputation.

Mode Imputation:

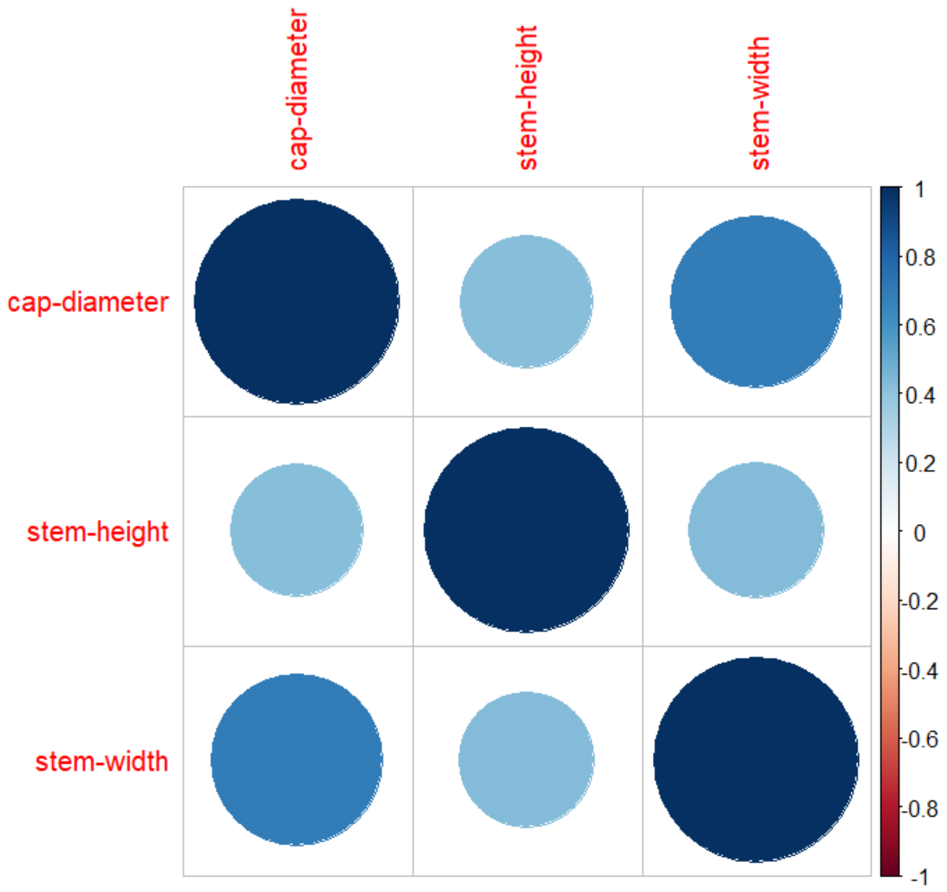
The remaining predictors with missing values were "cap-surface", "gill-attachment", "ring-type", and "gill-spacing". The mode is calculated for each of these predictors and the missing values are populated with their respective modes. After the imputation, all missing values in the data set have been removed or populated.

Degenerate Distributions:

After handling all missing values, we must check for degenerate distributions and handle them accordingly. First we checked for Near Zero Variance and Zero Variance predictors. These predictors must be removed from the data set because the models that will be explored and used cannot support sparse data. The Near Zero Variance Predictors are listed below.

	freqRatio	PercentUnique	zeroVar	nzv
ring-type	19.86078	0.01309994	FALSE	TRUE

There was only one predictor that had near zero variance, with 0.01309994% unique values. This predictor was removed.
The next degenerate distribution that needs to be checked is correlation between predictors. This cannot occur in categorical predictors, so the numeric predictors are separated and checked. Below is the correlation matrix for the 3 numeric predictors.



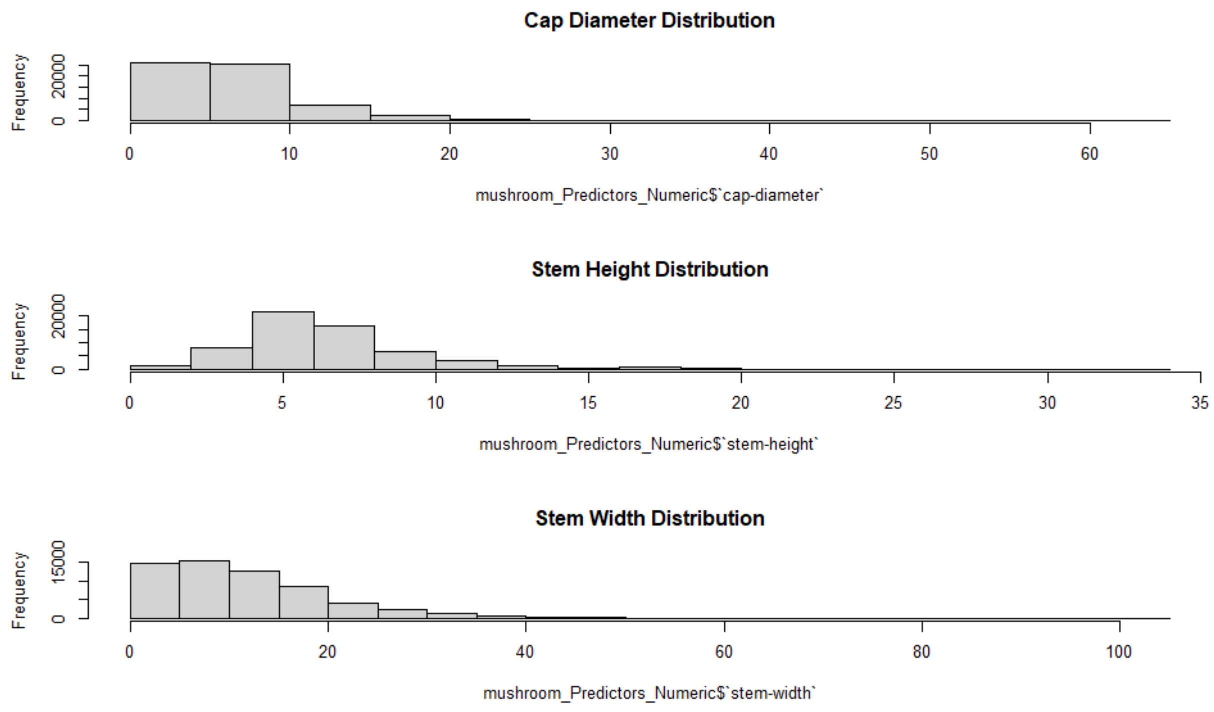
As we can see, there does not appear to be significant correlation between predictors. To further investigate, a threshold of 70% correlation was set and no predictors were above that threshold, so none were removed. Following this dummy variables were created for all remaining categorical predictors.

Transformations:

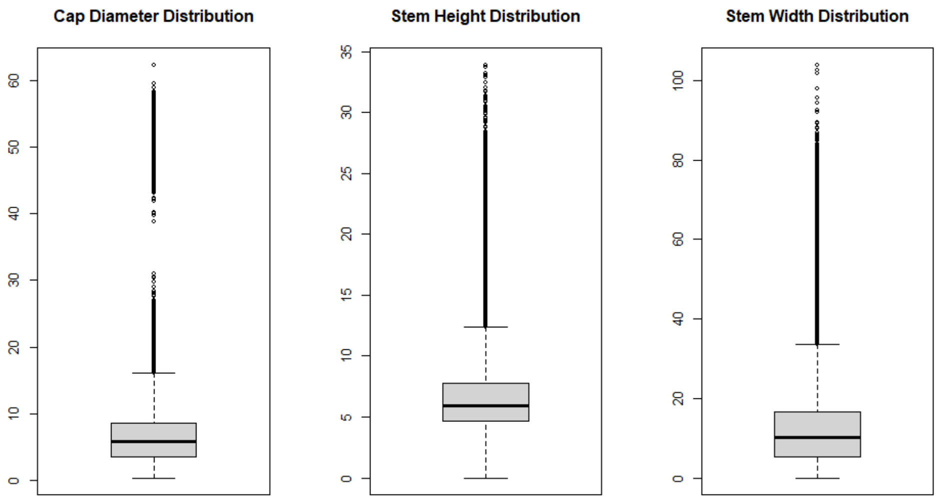
To identify which transformations should be applied, we first investigated the skewness statistic amongst all numeric predictors. This metric was filtered by those predictors whose skewness value was above 1 or below -1. Those predictors and their corresponding skewness values are below.

cap-diameter	stem-height	stem-width
3.822656	2.020804	2.164850

Their corresponding histograms are below to further support the evidence that these are highly skewed predictors.

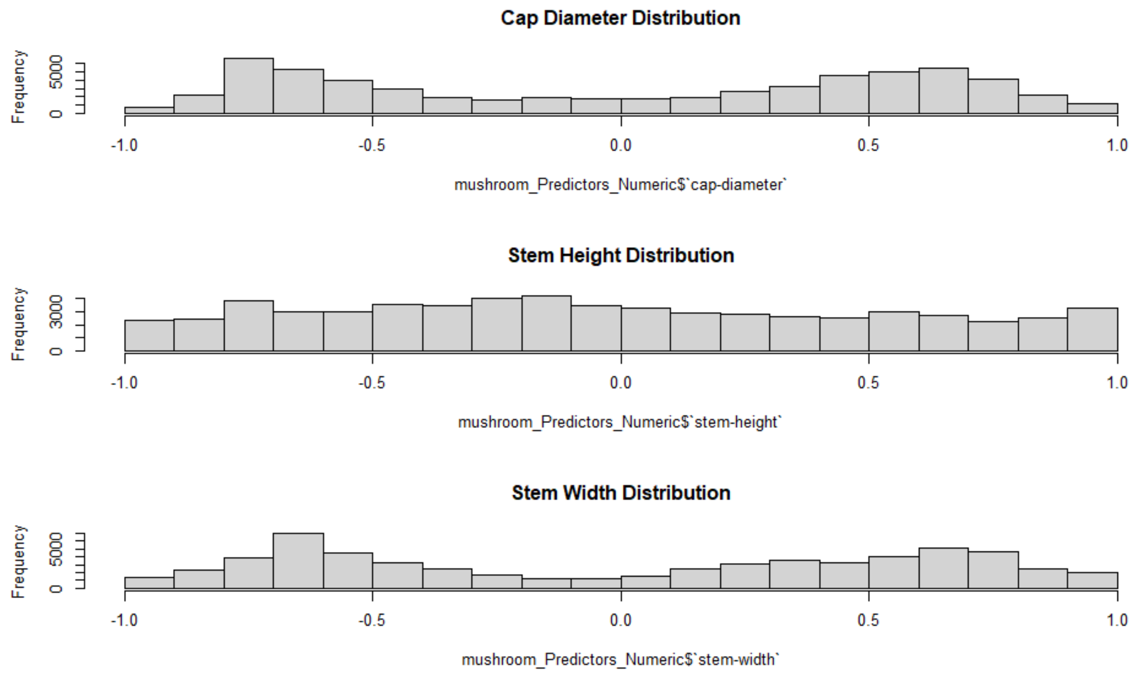


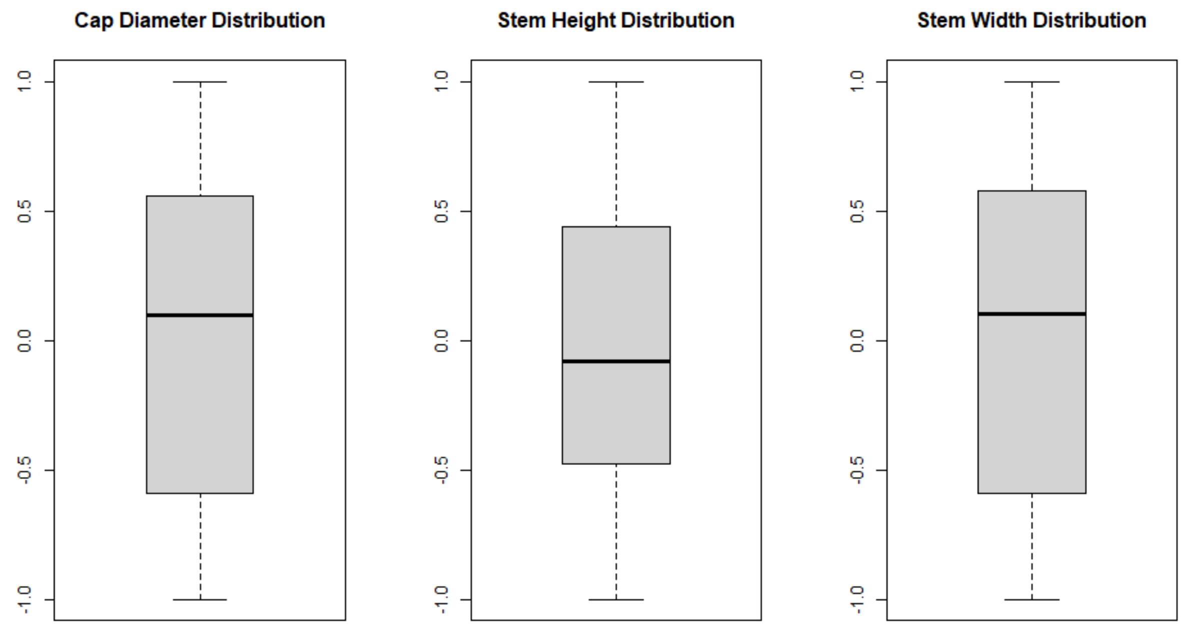
In order to account for this skewness, a box-cox transformation can be applied to each predictor. In order for the box-cox transformation to work, the predictors must not have any zero values. Predictors were checked for any zero values, and a value of 0.1 was added to all observations where this was observed in the predictors. Once identified that a box-cox transformation was necessary, we then checked for any outliers among the numeric predictors. Seen below are the box plots for all numeric predictors.



Each one of the predictors has a significant amount of outliers, in order to account for this a spatial sign transformation will be applied as well.

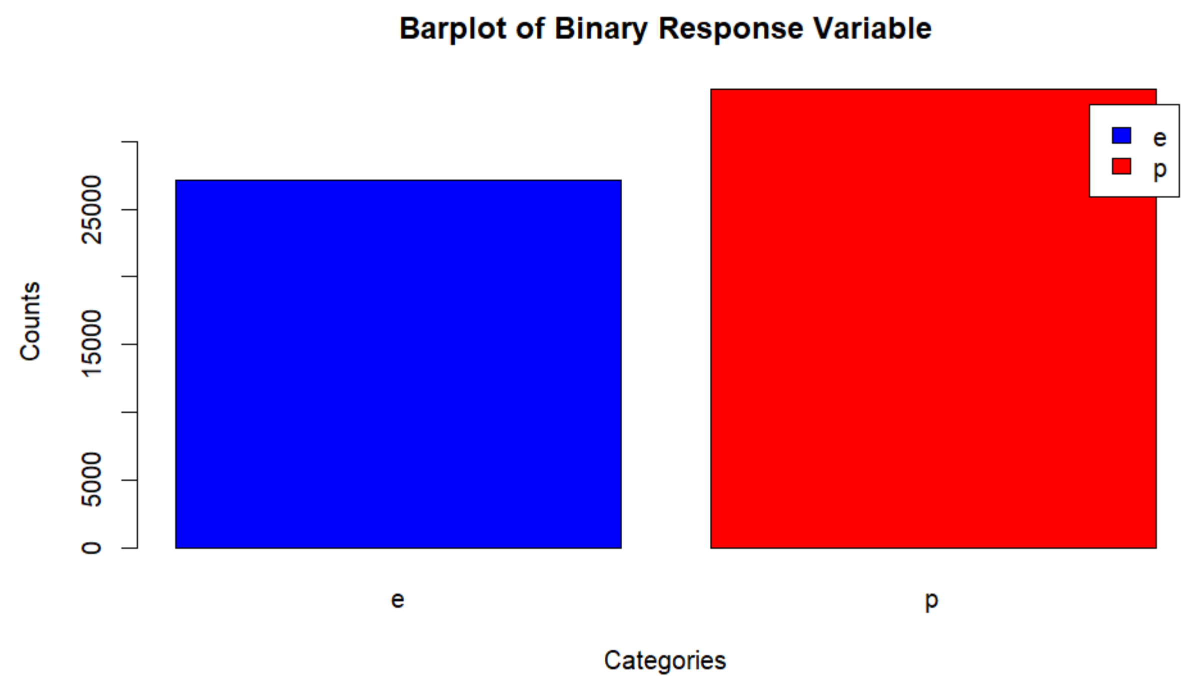
In order to account for the skewness, outliers and differing scales of the continuous variables, the data will first be centered and scaled. Following this, a Box-Cox transformation will be applied followed by a spatial sign transformation to account for outliers. Below are histograms and boxplots of the data after applying the transformations. It is clear that there are fewer outliers and the data is more symmetrical. The lambdas used during the box-cox transformation were 0.2, 0.4, and 0.3





Splitting of the Data:

There is only one possible outcome that we can attempt to predict. Which is whether or not the particular observed mushroom is edible or not which is a binary outcome. The bar plots below show the distribution of this binary outcome. The two classes are relatively balanced with observations that are edible being 44.50867% of the data and observations that are poisonous being 55.49133% of the data.We proceed with a simple random split of 70% in the training data and 30% in the testing data. The test data has 18321 observations across 88 predictors and the training data has 42748 observations across 88 predictors.



Model Fitting:

Model Training:

Below is a table of summaries for all models trained with a binary outcome. All parameters were tuned using 5 fold cross validation. We first attempted running 10-fold cross validation but the run times for many models were greater than 48 hours. We then used ROC as the metric to select the optimal model during training. The resulting ROC value was recorded along with the sensitivity, specificity, accuracy, and kappa values from predicting on the training set. The top five models are highlighted in red and will be further explored. The training set was predicted on once which brought about very optimistic results

Linear Models					
Model	Specificity	Sensitivity	ROC	Accuracy	Kappa
Logistic Regression	0.7826	0.7325	0.8406871	0.7604	0.5149
Linear Discriminant Analysis	0.7801	0.7402	0.8393885	0.7624	0.5195

PLS Discriminant Analysis	0.7708	0.7229	0.8394725	0.7495	0.4933
Penalized Model	0.8693	0.4541	0.8414859	0.6847	0.3362
Nearest Shrunken Centroid	0.7613	0.6047	0.7551841	0.6917	0.3696
Nonlinear Models					
Neural Network	0.9464	0.9588	0.9999838	0.9519	0.9029
Flexible Discriminant Analysis	0.7466	0.5832	0.7932640	0.6739	0.3331
Support Vector Machine	0.9653	0.9615	0.9997658	0.9636	0.9264
KNN	0.9987	0.9957	0.9985157	0.9974	0.9947
Naïve Bayes	0.4066	0.8299	0.8008022	0.5948	0.2237

Predicting Best Models on Test Set:

The top five models selected were then used to predict on the test set for the final model selection. We can mainly look at ROC but also take into account Specificity (True Negative Rate). A high specificity means correctly identifying a high percentage of truly poisonous mushrooms. However, it might increase the chances of misclassifying edible mushrooms as poisonous (false positives) but this is essential to the root of this data. It is far more important to ensure we accurately predict poisonous mushrooms rather than edible ones. It can be seen in the table below that the Neural Network model gave the highest ROC value when predicting on the test set, as well as the second highest specificity. The SVM model does have a specificity 0.001 higher than the NN, but NN beats SVM in accuracy, sensitivity, and kappa. With a ROC of 1 and a very minute difference in the other metrics, The Neural Network model is the model that we have selected and would recommend.

Top 5 Models					
Model	Specificity	Sensitivity	ROC	Accuracy	Kappa
K Nearest Neighbors	0.9987	0.9971	0.9997	0.998	0.9959
Logistic Regression	0.7804	0.7329	0.8333	0.7592	0.513
Neural Network	0.9988	0.9969	1	0.998	0.9959
Penalized Model	0.7873	0.7361	0.8389	0.7644	0.5234

SVM	0.9989	0.9892	0.9998	0.9946	0.9891

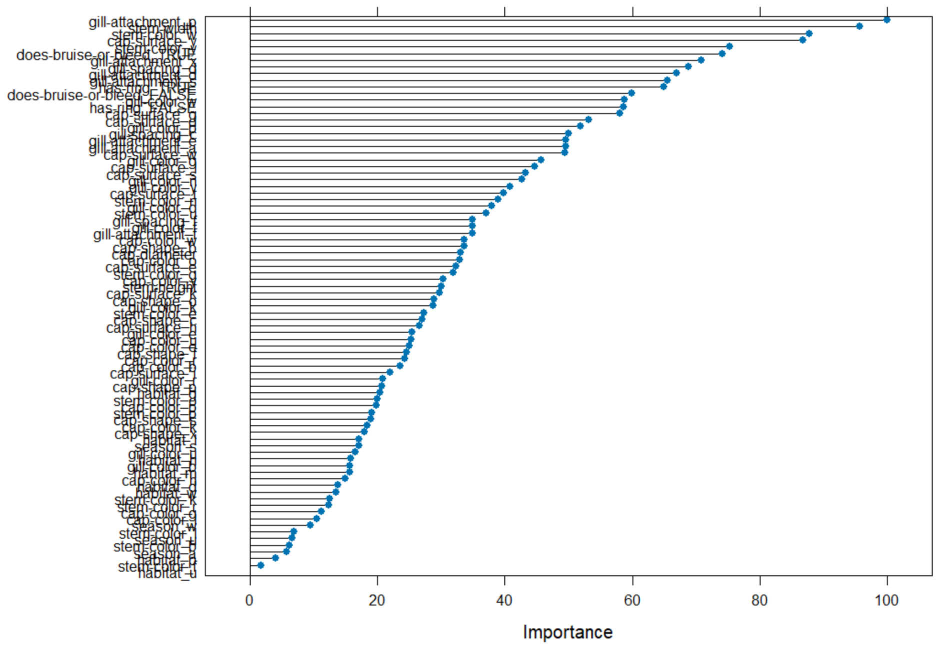
Optimal Model:

Below is the confusion matrix from prediction on the test set for the optimal model selected. Unfortunately still with this highly accurate mode, there were 25 mushrooms that were poisonous and classified as edible.

	Reference	
Predictions	Edible	Poisonous
Edible	8155	12
Poisonous	25	10129

Top Predictors:

It is good practice to look at the top predictors among models to further understand the data. Below is a graphic representation of the top predictors in order for the SVM Model.



```
nnet variable importance

  only 20 most important variables shown (out of 84)

                          Overall
gill-attachment_p          100.00
stem-width                 95.74
stem-color_w                87.86
cap-surface_y              86.86
stem-color_y               75.37
does-bruise-or-bleed_TRUE    74.16
gill-attachment_x           70.89
gill-spacing_d             68.83
gill-attachment_d           67.03
gill-attachment_s           65.50
has-ring_TRUE               64.97
does-bruise-or-bleed_FALSE   59.94
gill-color_w               58.80
has-ring_FALSE              58.64
cap-surface_g              57.98
```

cap-surface_d	53.10
gill-color_p	51.82
gill-spacing_c	49.96
gill-attachment_e	49.63
gill-attachment_a	49.58

The most important 5 predictors for this model are gill-attachment_p, stem-width, stem-color_w, cap-surface_y, and stem-color_y.

Summary:

The Neural Network model, chosen as the final predictive model, exhibited robust performance across multiple evaluation metrics:

Specificity: 0.9988
Sensitivity: 0.9969
ROC: 1
Accuracy: 0.998
Kappa: 0.9891

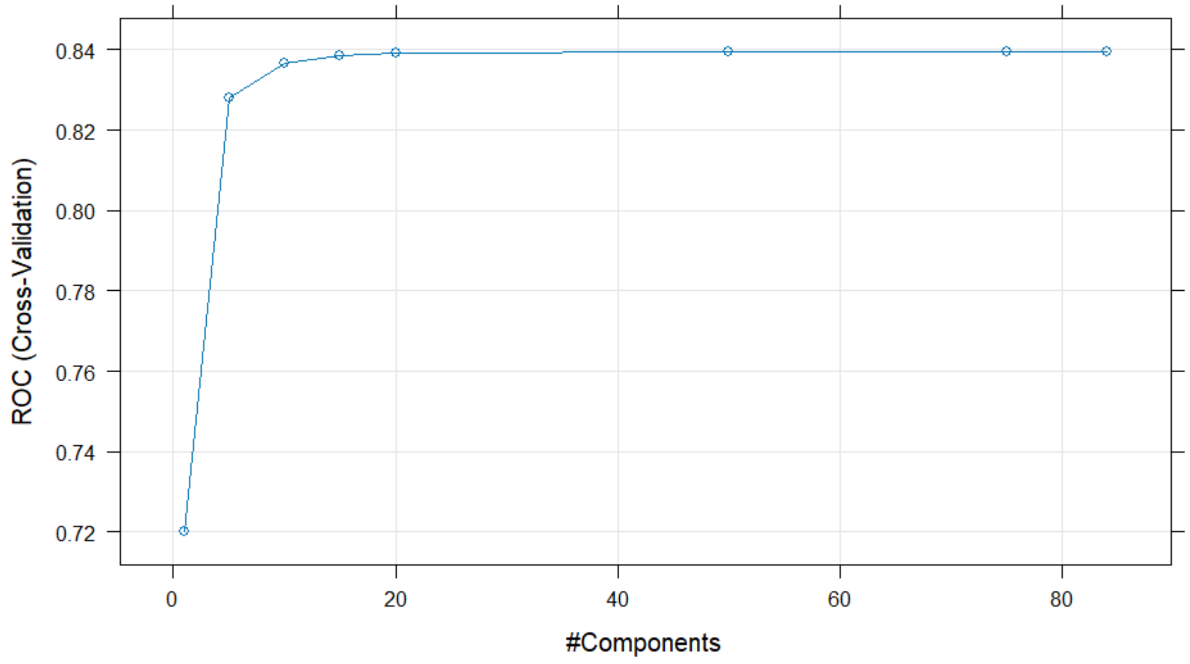
These metrics collectively underscore the Neural Network's robust predictive capability and efficacy in discerning between edible and poisonous mushrooms. Notably high values for specificity and sensitivity, complemented by a near-perfect ROC score, affirm the model's adeptness in accurately identifying both categories of mushrooms. Overall, the Neural Network demonstrates outstanding proficiency in precisely predicting mushroom edibility.

Appendix 1: Supplemental Material for Binary Classification Model

Linear Models:

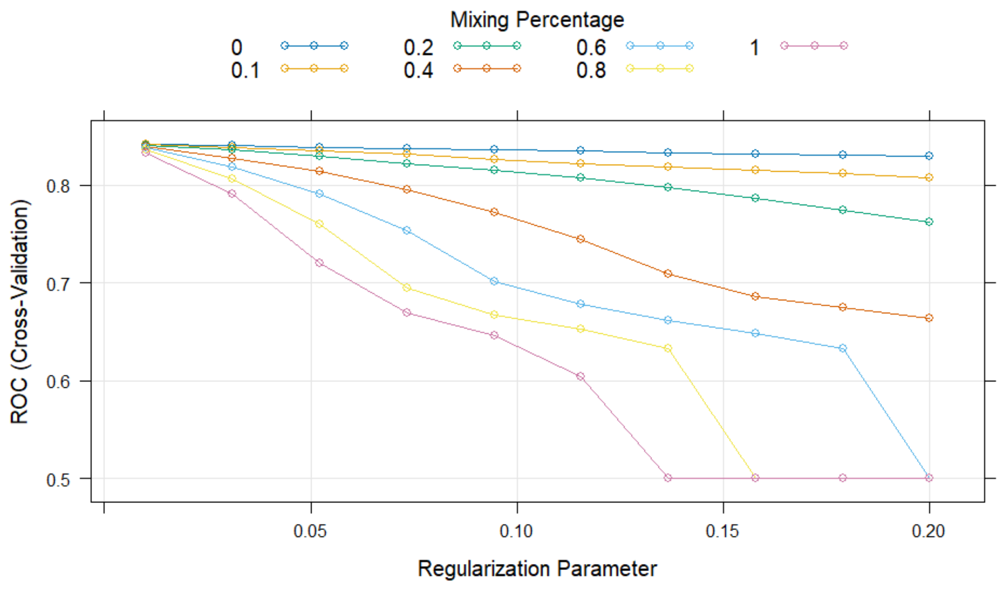
PLS Discriminant Analysis:

ROC was used to select the optimal model using the largest value.
The final value used for the model was ncomp = 75.



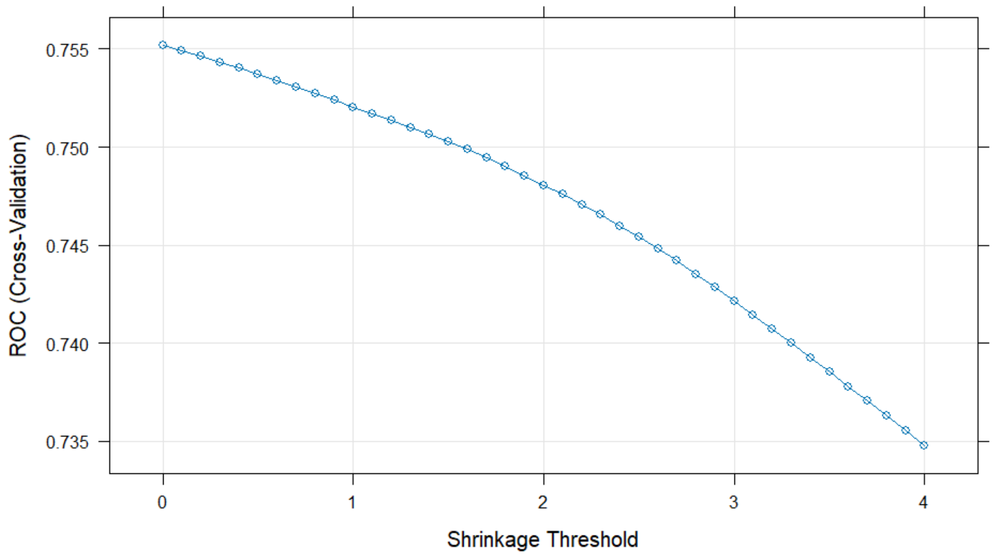
Penalized Model:

ROC was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0 and lambda = 0.01.



Nearest Shrunken Centroid:

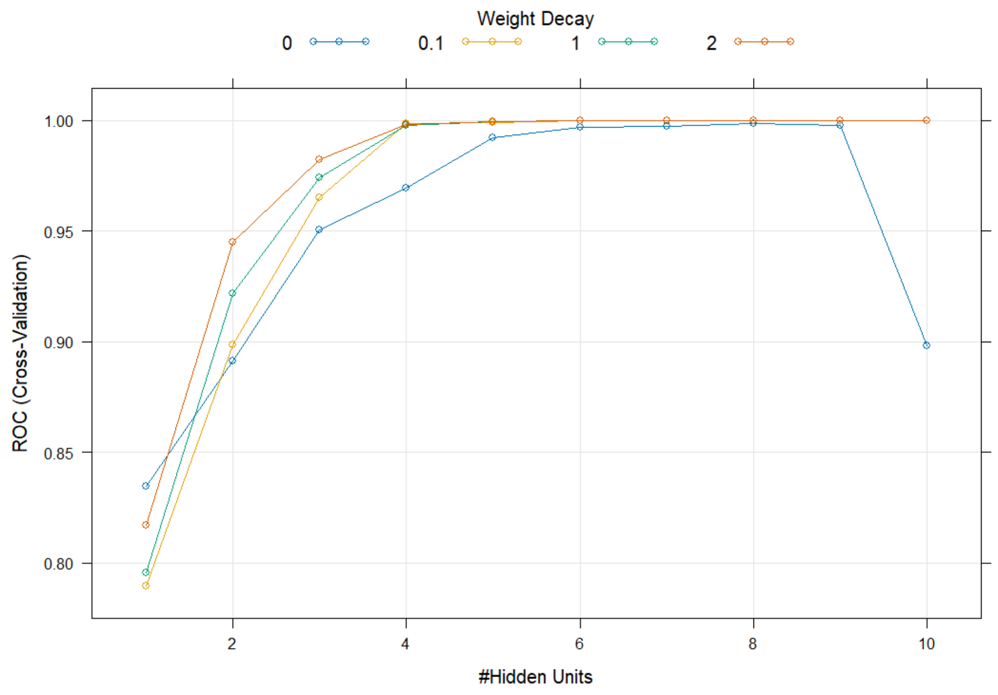
ROC was used to select the optimal model using the largest value.
The final value used for the model was threshold = 0.



Nonlinear Models:

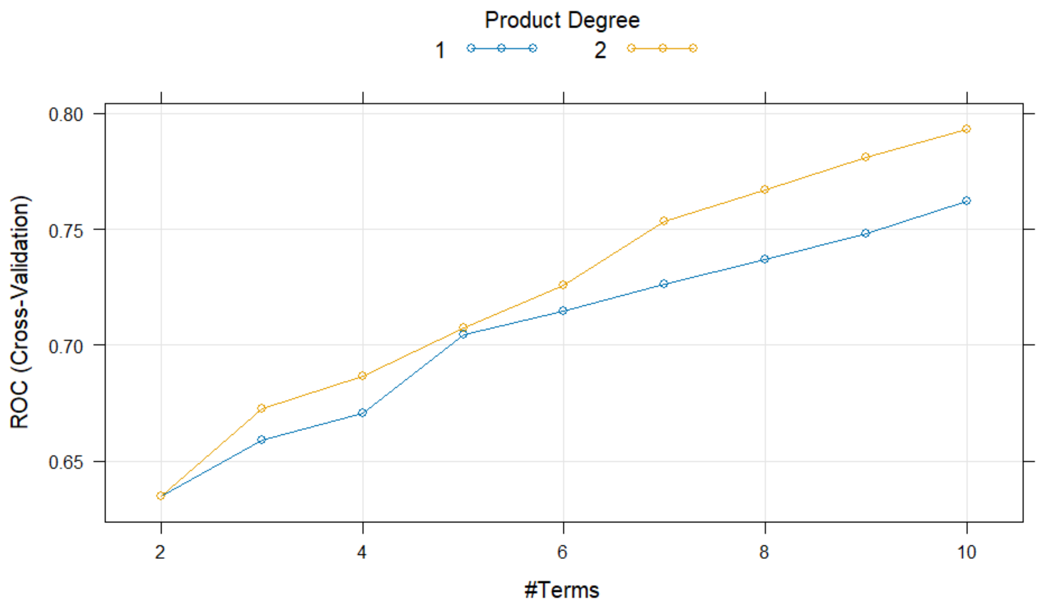
Neural Network:

ROC was used to select the optimal model using the largest value.
The final values used for the model were size = 10 and decay = 0.1



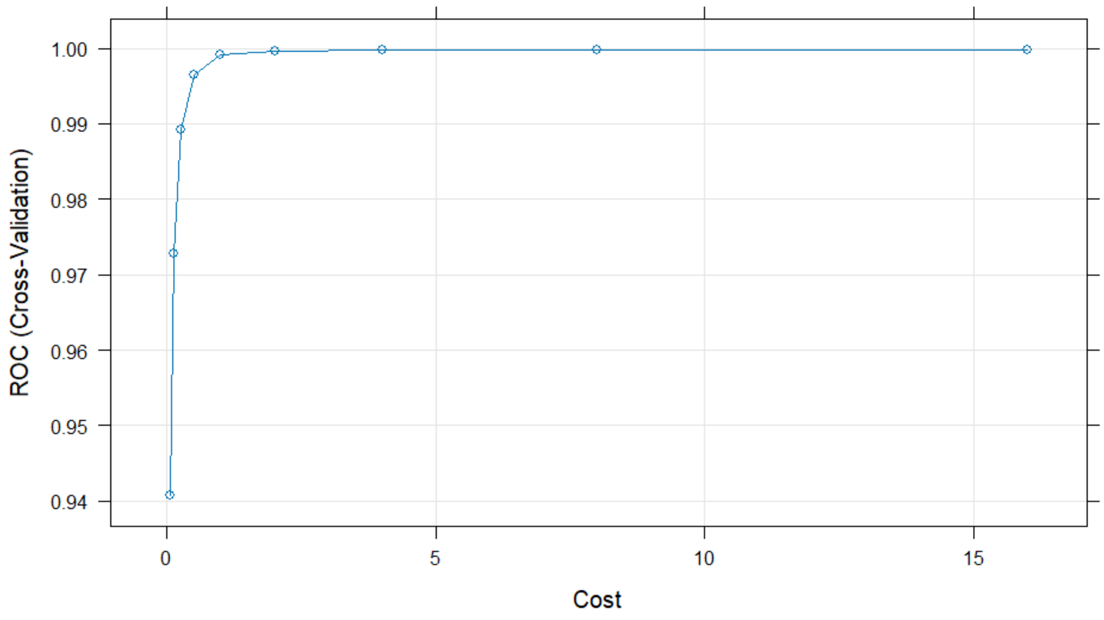
Flexible Discriminant Analysis:

ROC was used to select the optimal model using the largest value.
The final values used for the model were degree = 2 and nprune = 10.



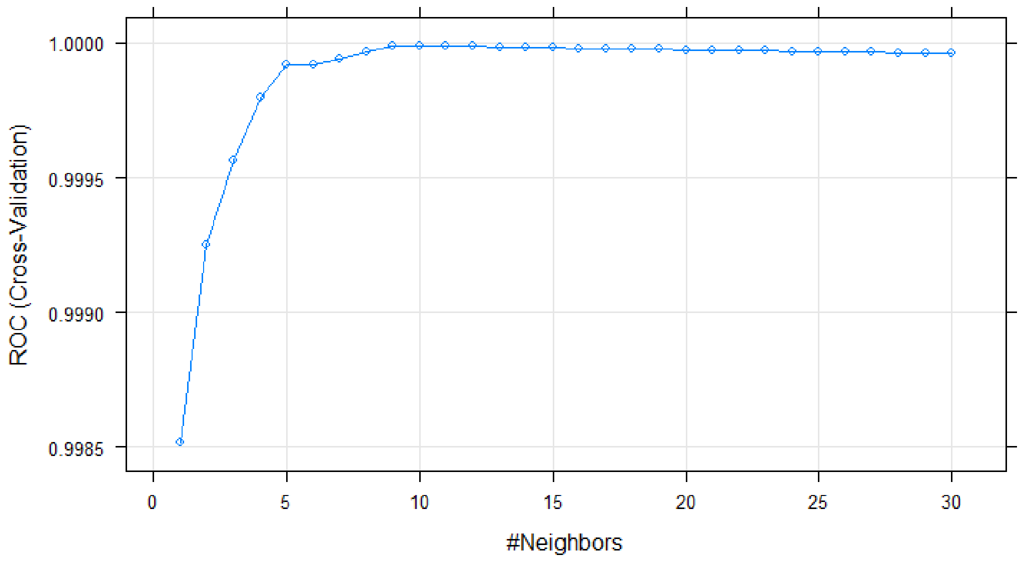
Support Vector Machine:

Tuning parameter 'sigma' was held constant at a value of 0.003946548
ROC was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.003946548 and C = 4.



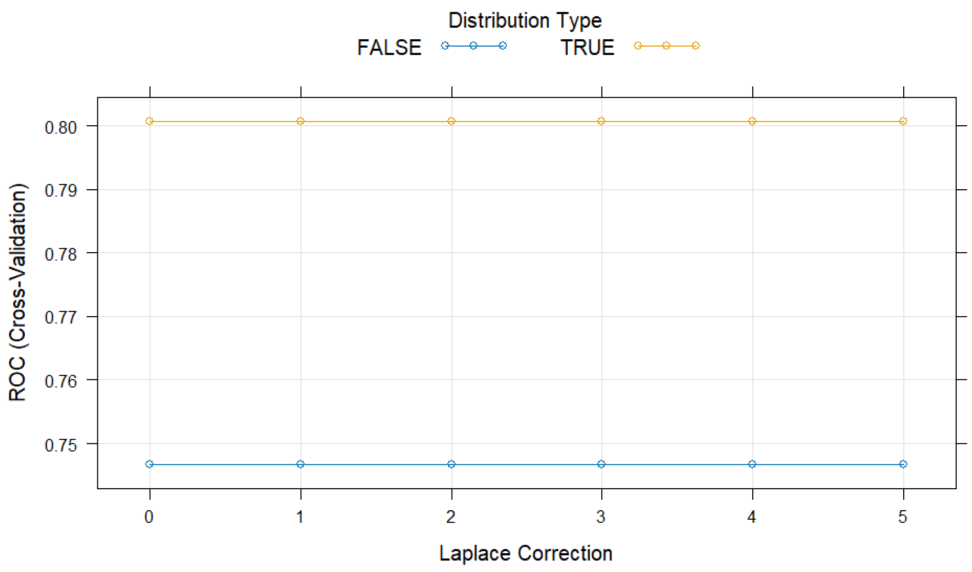
K-Nearest Neighbors:

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 9.



Naive Bayes:

Tuning parameter 'adjust' was held constant at a value of 1
ROC was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.



Citations:

Ranadheera, Senaka. "Mushrooms Are Delicious, but Potentially Deadly." Pursuit, November 28, 2023. https://pursuit.unimelb.edu.au/articles/mushrooms-are-delicious-but-potentially-deadly.

Devzohaib. "Mushroom Edibility Classification." Kaggle, November 11, 2022. https://www.kaggle.com/datasets/devzohaib/mushroom-edibility-classification.

R-Code:

```
JavaScript
library(readr)
mushroom_data <- read_delim("mushroom/secondary_data.csv", delim = ";",
escape_double = FALSE, trim_ws = TRUE)
summary(mushroom_data)
#Histograms of Predictors
```

```r
par(mfrow = c(3, 1))
#for numeric
hist(mushroom_data$`cap-diameter`, main='Cap Diameter Distribution')
hist(mushroom_data$`stem-height`, main='Stem Height Distribution')
hist(mushroom_data$`stem-width`, main='Stem Width Distribution')
#For categorical
par(mfrow = c(3, 5))
barplot(table(mushroom_data$`cap-shape`),
   ylab = "Frequency",
   xlab = "Cap Shape")
barplot(table(mushroom_data$`cap-surface`),
   ylab = "Frequency",
   xlab = "Cap Surface")
barplot(table(mushroom_data$`cap-color`),
   ylab = "Frequency",
   xlab = "Cap Color")
barplot(table(mushroom_data$`does-bruise-or-bleed`),
   ylab = "Frequency",
   xlab = "Does Bruise or Bleed")
barplot(table(mushroom_data$`gill-attachment`),
   ylab = "Frequency",
   xlab = "Gill Attachment")
barplot(table(mushroom_data$`gill-spacing`),
   ylab = "Frequency",
   xlab = "Gill Spacing")
barplot(table(mushroom_data$`gill-color`),
   ylab = "Frequency",
   xlab = "Gill Color")
barplot(table(mushroom_data$`stem-root`),
   ylab = "Frequency",
   xlab = "Stem Root")
barplot(table(mushroom_data$`stem-surface`),
   ylab = "Frequency",
   xlab = "Stem Surface")
barplot(table(mushroom_data$`stem-color`),
   ylab = "Frequency",
   xlab = "Stem Color")
barplot(table(mushroom_data$`veil-type`),
   ylab = "Frequency",
   xlab = "Viel Type")
barplot(table(mushroom_data$`veil-color`),
   ylab = "Frequency",
   xlab = "Viel Color")
barplot(table(mushroom_data$`has-ring`),
    ylab = "Frequency",
    xlab = "Has Ring")
barplot(table(mushroom_data$`ring-type`),
    ylab = "Frequency",
    xlab = "Ring Type")
barplot(table(mushroom_data$`spore-print-color`),
    ylab = "Frequency",
    xlab = "Spore Print Color")
par(mfrow = c(1, 2))
barplot(table(mushroom_data$`habitat`),
    ylab = "Frequency",
    xlab = "Habitat")
barplot(table(mushroom_data$`season`),
    ylab = "Frequency",
    xlab = "Season")

#get data frame with only predictors
mushroom_Predictors <- subset(mushroom_data, select = -class)
mushroom_Predictors_Original <- mushroom_Predictors
mushroom_Class<-subset(mushroom_data, select = class)

#Deal with missing values
library(naniar)
(sum(is.na(mushroom_Predictors))/prod(dim(mushroom_Predictors)))*100
gg_miss_var(mushroom_Predictors, show_pct = TRUE)
vis_miss(mushroom_Predictors, warn_large_data = FALSE)

#Missing percent for those predictors over 50%
colMeans(is.na(mushroom_Predictors))

# get subset of colnames NA proportion is greater than threshold
threshold <- .5
names(mushroom_Predictors)[sapply(mushroom_Predictors, function(x) mean(is.na(x))
> threshold)]

#Remove Columns with > 50% missing data
mushroom_Predictors <- subset(mushroom_Predictors, select = -c(`veil-color`,
`stem-root`, `stem-surface`, `veil-type`, `spore-print-color`))
dim(mushroom_Predictors)
dim(mushroom_Predictors_Original)
colMeans(is.na(mushroom_Predictors))
(sum(is.na(mushroom_Predictors))/prod(dim(mushroom_Predictors)))*100

####PERFORM IMPUTATION#########
##Mode Imputation because only categorical vars have missing values
```

```
colMeans(is.na(mushroom_Pred))
calc_mode <- function(x){
 # List the distinct / unique values
 distinct_values <- unique(x)
 # Count the occurrence of each distinct value
 distinct_tabulate <- tabulate(match(x, distinct_values))
 # Return the value with the highest occurrence
 distinct_values[which.max(distinct_tabulate)]
}

#Replace NA avalues with mode
mushroom_Predictors["cap-surface"][is.na(mushroom_Predictors["cap-surface"])] <-
calc_mode(na.omit(mushroom_Predictors$`cap-surface`))
mushroom_Predictors["gill-attachment"][is.na(mushroom_Predictors["gill-attachme
nt"])] <- calc_mode(na.omit(mushroom_Predictors$`gill-attachment`))
mushroom_Predictors["ring-type"][is.na(mushroom_Predictors["ring-type"])] <-
calc_mode(na.omit(mushroom_Predictors$`ring-type`))
mushroom_Predictors["gill-spacing"][is.na(mushroom_Predictors["gill-spacing"])]
<- calc_mode(na.omit(mushroom_Predictors$`gill-spacing`))
colMeans(is.na(mushroom_Predictors))
dim(mushroom_Predictors)

##Check for NZV before converting
#Check Near Zero Variance for all categorical predictors
library(caret)
nzv <- nearZeroVar(mushroom_Predictors, saveMetrics= TRUE)
library(dplyr)
filter(nzv, zeroVar == TRUE | nzv == TRUE)
# Filter out predictors identified as near-zero variance
zero_var_cols <- which(nzv$nzv == TRUE | nzv$zeroVar == TRUE)
zero_var_cols
mushroom_Predictors_Processed <- mushroom_Predictors[, -zero_var_cols]
#Convert all Categorical Variables to dummy variables
library(fastDummies)
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed, select_columns = "cap-shape")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed, select_columns = "cap-surface")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed, select_columns = "cap-color")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed, select_columns = "does-bruise-or-bleed")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed, select_columns = "gill-attachment")
```

```r
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "gill-spacing")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "gill-color")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "stem-color")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "has-ring")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "habitat")
mushroom_Predictors_Processed <-
dummy_cols(mushroom_Predictors_Processed,select_columns = "season")
colnames(mushroom_Predictors_Processed)

##make seperate data frames##
mushroom_Predictors_Processed<-subset(mushroom_Predictors_Processed, select =
-c(`cap-shape`, `cap-surface`, `cap-color`, `does-bruise-or-bleed`,
`gill-attachment`, `gill-spacing`, `gill-color`, `stem-color`, `has-ring`,
`habitat`, `season`))
mushroom_Predictors_Numeric<-subset(mushroom_Predictors_Processed, select =
c(`cap-diameter`, `stem-height`, `stem-width`))
mushroom_Predictors_Categorical <-subset(mushroom_Predictors_Processed,
select=-c(`cap-diameter`, `stem-height`, `stem-width`))

###look at dimensions###
dim(mushroom_Predictors)
dim(mushroom_Predictors_Processed)
dim(mushroom_Predictors_Categorical)
dim(mushroom_Predictors_Numeric)

##CHECK IF NEED TO DELETE PREDICTORS FOR NUMERIC VARS ONLY
#check out correlation matrix
par(mfrow = c(1, 1))
library(corrplot)
corr_matrix <- cor(mushroom_Predictors_Numeric)
corrplot(corr_matrix)
cor <- cor(mushroom_Predictors_Numeric)
library(caret)
high <- findCorrelation(cor, .7)
high


#taking a look at the skewness values
library(e1071)
skewValues <- apply(mushroom_Predictors_Numeric, 2, skewness)
```

```r
#filtering by those classified as extremely skewed
highSkew<-Filter(function(x) any(x < -1 | x > 1), skewValues)
length(highSkew)
highSkew

par(mfrow = c(3, 1))
#for numeric
hist(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
hist(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height Distribution')
hist(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width Distribution')

par(mfrow = c(1, 3))
boxplot(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width
Distribution')
#Perform transformations on numeric data, these have zero values so need to add
library(caret)
BCStemWidth <- BoxCoxTrans(mushroom_Predictors_Numeric$`stem-width`)
BCStemWidth
BCStemHeight <- BoxCoxTrans(mushroom_Predictors_Numeric$`stem-height`)
BCStemHeight
BCCapDiam <- BoxCoxTrans(mushroom_Predictors_Numeric$`cap-diameter`)
BCCapDiam

#Find samples with zero values in stem-height and stem-width
StemHeightZero <-
mushroom_Predictors_Numeric[mushroom_Predictors_Numeric$`stem-height` == 0,]
StemHeightZero
StemWidthZero <-
mushroom_Predictors_Numeric[mushroom_Predictors_Numeric$`stem-width` == 0,]
StemWidthZero

#Adjust values of all numeric observations
mushroom_Predictors_Numeric$`stem-height` <-
mushroom_Predictors_Numeric$`stem-height` + .1
mushroom_Predictors_Numeric[mushroom_Predictors_Numeric$`stem-height` == .1,]
mushroom_Predictors_Numeric$`stem-width` <-
mushroom_Predictors_Numeric$`stem-width` + .1
mushroom_Predictors_Numeric[mushroom_Predictors_Numeric$`stem-width` == .1,]
mushroom_Predictors_Numeric$`cap-diameter` <-
mushroom_Predictors_Numeric$`cap-diameter` + .1
```

```
##HANDLE Outliers and skewness now that there are no zero values
#Visual Check as well
par(mfrow = c(3, 1))
#for numeric
hist(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
hist(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height Distribution')
hist(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width Distribution')
par(mfrow = c(1, 3))
boxplot(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width
Distribution')
###transformations
#use spatial sign for outliers, boxcox for skew, and center and scale
library(caret)
trans <- preProcess(as.data.frame(mushroom_Predictors_Numeric), method =
c("spatialSign", "BoxCox", "center", "scale"))
mushroom_Predictors_Numeric <- predict(trans,
as.data.frame(mushroom_Predictors_Numeric))
##Skewness check after
skewness(mushroom_Predictors_Numeric$`stem-width`)
skewness(mushroom_Predictors_Numeric$`stem-height`)
skewness(mushroom_Predictors_Numeric$`cap-diameter`)
#Visual Check as well
par(mfrow = c(3, 1))
#for numeric
hist(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
hist(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height Distribution')
hist(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width Distribution')
par(mfrow = c(1, 3))
boxplot(mushroom_Predictors_Numeric$`cap-diameter`, main='Cap Diameter
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-height`, main='Stem Height
Distribution')
boxplot(mushroom_Predictors_Numeric$`stem-width`, main='Stem Width
Distribution')

#add transformed numeric variables back into DF
```

```r
mushroom_Predictors_Processed$`cap-diameter`<-mushroom_Predictors_Numeric$`cap-
diameter`
mushroom_Predictors_Processed$`stem-height`<-mushroom_Predictors_Numeric$`stem-
height`
mushroom_Predictors_Processed$`stem-width`<-mushroom_Predictors_Numeric$`stem-w
idth`
#combine response with processed predictors
mushroom_Processed <- cbind(mushroom_Predictors_Processed, class =
mushroom_data$class)
dim(mushroom_Processed)
#Spend the data

#determine response balance
class_counts <- table(mushroom_Processed$class)
par(mfrow = c(1, 1))
barplot(class_counts,
    main = "Barplot of Binary Response Variable",
    xlab = "Categories",
    ylab = "Counts",
    col = c("blue", "red"), # Color for the bars
    legend = names(class_counts)) # Add a legend with category names
class_counts <- table(mushroom_Processed$class)
class_percentages <- prop.table(class_counts) * 100
class_percentages
# Set seed for reproducibility
set.seed(123)

# Determine number of rows for training set
train_rows <- round(0.7 * nrow(mushroom_Processed))

# Generate random indices for splitting the data
train_indices <- sample(seq_len(nrow(mushroom_Processed)), size = train_rows,
replace = FALSE)

# Create training and testing sets
train_data <- mushroom_Processed[train_indices, ]
test_data <- mushroom_Processed[-train_indices, ]
x_test <- test_data[, -which(names(test_data) == "class")]
y_test <- test_data$class

### models
# Define the trainControl with 5-fold CV and required settings
ctrl <- trainControl(method = "cv",
          number = 5,
          summaryFunction = twoClassSummary,
         classProbs = TRUE,
         savePredictions = TRUE)

## Train the logistic regression model
mushroom_lr_model <- train(x = train_data[, -which(names(train_data) == "class")], #
Select predictors except the response variable,
            y = train_data$class, # Response variable
            method = "glm",
            metric = "ROC",
            trControl = ctrl)
# View the trained model
mushroom_lr_model
#View training confusion matrix
confusionMatrix(data = mushroom_lr_model$pred$pred,
       reference = mushroom_lr_model$pred$obs)
# Predict on test data
mushroom_lr_prob <- predict(mushroom_lr_model,
            newdata = x_test,
            type = "prob")
mushroom_lr_pred <- predict(mushroom_lr_model,
            newdata = x_test,
            type = "raw")
class(mushroom_lr_pred)
class(test_data$class)
test_data$class <- factor(test_data$class)
# View test data confusion matrix
confusionMatrix(data = mushroom_lr_pred,
       reference = test_data$class,
       positive = "e")
# Get ROC curve from test data
mushroom_lr_roc <- roc(response = test_data$class,
          predictor = mushroom_lr_prob$e,
          levels = rev(levels(test_data$class)))
# Plot the ROC curve
par(mfrow = c(1, 1))
plot(mushroom_lr_roc, legacy.axes = TRUE)
# View AUC of ROC curve
auc(mushroom_lr_roc)

## Train the LDA model
mushroom_LDA_model <- train(x = train_data[, -which(names(train_data) == "class")],
            y = train_data$class, # Response variable
            method = "lda",
            metric = "ROC",
            trControl = ctrl)
# View the trained model
mushroom_LDA_model

# View training confusion matrix
confusionMatrix(data = mushroom_LDA_model$pred$pred,
       reference = mushroom_LDA_model$pred$obs)
# Predict on test data
mushroom_lda_prob <- predict(mushroom_LDA_model,
             newdata = x_test,
             type = "prob")
mushroom_lda_pred <- predict(mushroom_LDA_model,
             newdata = x_test,
             type = "raw")
# View test data confusion matrix
confusionMatrix(data = mushroom_lda_pred,
       reference = test_data$class,
       positive = "e")
# Get ROC curve from test data
mushroom_lda_roc <- roc(response = test_data$class,
          predictor = mushroom_lda_prob$e,
          levels = rev(levels(test_data$class)))
# Plot the ROC curve
plot(mushroom_lda_roc, legacy.axes = TRUE)
# View AUC of ROC curve
auc(mushroom_lda_roc)

# Train PLSDA model
plsGrid <- expand.grid(.ncomp = c(1,5,10,15,20,50,75,84))
mushroom_plsda_model <- train(x = train_data[, -which(names(train_data) ==
"class")],
             y = train_data$class,  # Response variable
             method = "pls",
             tuneGrid = plsGrid,
             metric = "ROC",
             trControl = ctrl)

mushroom_plsda_model
plot(mushroom_plsda_model)

confusionMatrix(data = mushroom_plsda_model$pred$pred,
       reference = mushroom_plsda_model$pred$obs)

mushroom_plsda_prob <- predict(mushroom_plsda_model,
              newdata = x_test,
             type = "prob")

mushroom_plsda_pred <- predict(mushroom_plsda_model,
             newdata = x_test,
             type = "raw")

confusionMatrix(data = mushroom_plsda_pred,
       reference = test_data$class,
       positive = "e")

mushroom_plsda_roc <- roc(response = test_data$class,
          predictor = mushroom_plsda_prob$e,
          levels = rev(levels(test_data$class)))
plot(mushroom_plsda_roc, legacy.axes = TRUE)
auc(mushroom_plsda_roc)

#Train GLMNet model
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
          .lambda = seq(.01, .2, length = 10))
mushroom_glmnet_model <- train(x = train_data[, -which(names(train_data) ==
"class")],
             y = train_data$class,  # Response variable
             method = "glmnet",
             tuneGrid = glmnGrid,
             metric = "ROC",
             trControl = ctrl)
mushroom_glmnet_model
plot(mushroom_glmnet_model)
confusionMatrix(data = mushroom_glmnet_model$pred$pred,
       reference = mushroom_glmnet_model$pred$obs)

mushroom_glmnet_prob <- predict(mushroom_glmnet_model,
             newdata = x_test,
             type = "prob")

mushroom_glmnet_pred <- predict(mushroom_glmnet_model,
             newdata = x_test,
             type = "raw")

confusionMatrix(data = mushroom_glmnet_pred,
       reference = test_data$class,
       positive = "e")

mushroom_glmnet_roc <- roc(response = test_data$class,
          predictor = mushroom_glmnet_prob$e,
```

```r
            levels = rev(levels(test_data$class)))
plot(mushroom_glmnet_roc, legacy.axes = TRUE)
auc(mushroom_glmnet_roc)

#Train Nearest Shrunken Centroid model
nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))
mushroom_nsc_model <- train(x = train_data[, -which(names(train_data) == "class")],
            y = train_data$class,  # Response variable
            method = "pam",
            tuneGrid = nscGrid,
            metric = "ROC",
            trControl = ctrl)

mushroom_nsc_model
plot(mushroom_nsc_model)
confusionMatrix(data = mushroom_nsc_model$pred$pred,
       reference = mushroom_nsc_model$pred$obs)

mushroom_nsc_prob <- predict(mushroom_nsc_model,
             newdata = x_test,
             type = "prob")

mushroom_nsc_pred <- predict(mushroom_nsc_model,
             newdata = x_test,
             type = "raw")

confusionMatrix(data = mushroom_nsc_pred,
       reference = test_data$class,
       positive = "e")

mushroom_nsc_roc <- roc(response = test_data$class,
           predictor = mushroom_nsc_prob$e,
           levels = rev(levels(test_data$class)))
plot(mushroom_nsc_roc, legacy.axes = TRUE)
auc(mushroom_nsc_roc)

## Non-linear models
## Train Nonlinear Discriminant Analysis model
library(doParallel)
# Set up parallel processing
cores <- detectCores()
cl <- makeCluster(cores - 1) # Using one less core for other operations
registerDoParallel(cl)

# Your original code with parallel processing
```

```
fdaGrid <- expand.grid(.degree = 1:2, .nprune = 2:10)
mushroom_fda_model <- train(x = train_data[, -which(names(train_data) == "class")],
             y = train_data$class,
             method = "fda",
             tuneGrid = fdaGrid,
             metric = "ROC",
             trControl = ctrl)

# Stop and close the parallel processing cluster
stopCluster(cl)
mushroom_fda_model
plot(mushroom_fda_model)
confusionMatrix(data = mushroom_fda_model$pred$pred,
        reference = mushroom_fda_model$pred$obs)

mushroom_fda_prob <- predict(mushroom_fda_model,
              newdata = x_test,
              type = "prob")

mushroom_fda_pred <- predict(mushroom_fda_model,
              newdata = x_test,
              type = "raw")

confusionMatrix(data = mushroom_fda_pred,
        reference = test_data$class,
        positive = "e")

mushroom_fda_roc <- roc(response = test_data$class,
            predictor = mushroom_fda_prob$e,
            levels = rev(levels(test_data$class)))
plot(mushroom_fda_roc, legacy.axes = TRUE)
auc(mushroom_fda_roc)

## Train Support Vector Machine model
library(kernlab)
sigmaRangeReduced <- sigest(as.matrix(train_data[, -which(names(train_data) ==
"class")]))
svmGrid <- expand.grid(.sigma = sigmaRangeReduced[1],
           .C = 2^(seq(-4, 4)))
mushroom_svm_model <- train(x = train_data[, -which(names(train_data) == "class")],
             y = train_data$class, # Response variable
             method = "svmRadial",
             tuneGrid = svmGrid,
             metric = "ROC",
             trControl = ctrl)
```

```r
mushroom_svm_model
plot(mushroom_svm_model)
confusionMatrix(data = mushroom_svm_model$pred$pred,
       reference = mushroom_svm_model$pred$obs)

mushroom_svm_prob <- predict(mushroom_svm_model,
             newdata = x_test,
             type = "prob")

mushroom_svm_pred <- predict(mushroom_svm_model,
             newdata = x_test,
             type = "raw")
factor(mushroom_svm_pred)
confusionMatrix(data = mushroom_svm_pred,
       reference = test_data$class,
       positive = "e")

mushroom_svm_roc <- roc(response = test_data$class,
      predictor = mushroom_svm_prob[, "p"])

# Plot ROC curve
plot(mushroom_svm_roc, legacy.axes = TRUE)
auc(mushroom_svm_roc)

## Train K-nearest Neighbors model
knnGrid <- data.frame(.k = 1:15)
mushroom_knn_model <- train(x = train_data[, -which(names(train_data) == "class")],
            y = train_data$class,  # Response variable
            method = "knn",
            tuneGrid = knnGrid,
            metric = "ROC",
            trControl = ctrl)

mushroom_knn_model
plot(mushroom_knn_model)
confusionMatrix(data = mushroom_knn_model$pred$pred,
       reference = mushroom_knn_model$pred$obs)

mushroom_knn_prob <- predict(mushroom_knn_model,
             newdata = x_test,
             type = "prob")

mushroom_knn_pred <- predict(mushroom_knn_model,
             newdata = x_test,
            type = "raw")

confusionMatrix(data = mushroom_knn_pred,
       reference = test_data$class,
       positive = "e")

mushroom_knn_roc <- roc(response = test_data$class,
          predictor = mushroom_knn_prob$e,
          levels = rev(levels(test_data$class)))
plot(mushroom_knn_roc, legacy.axes = TRUE)
auc(mushroom_knn_roc)


###NB#############
## Train Naive Bayes model
library(klaR)
library(naivebayes)
nbGrid <- expand.grid(laplace = 0:5, usekernel = c(FALSE, TRUE), adjust = 1) # this is
the equivalent of 'NB' tunegrid
mushroom_nb_model <- train(x = train_data[, -which(names(train_data) == "class")],
            y = train_data$class,  # Response variable
            method = "naive_bayes", # note not 'nb'
            tuneGrid = nbGrid,
            metric = "ROC",
            trControl = ctrl)

mushroom_nb_model
plot(mushroom_nb_model)
confusionMatrix(data = mushroom_nb_model$pred$pred,
       reference = mushroom_nb_model$pred$obs)

mushroom_nb_prob <- predict(mushroom_nb_model,
            newdata = x_test,
            type = "prob")

mushroom_nb_pred <- predict(mushroom_nb_model,
            newdata = x_test,
            type = "raw")

confusionMatrix(data = mushroom_nb_pred,
       reference = test_data$class,
       positive = "e")

mushroom_nb_roc <- roc(response = test_data$class,
          predictor = mushroom_nb_prob$e,
          levels = rev(levels(test_data$class)))
plot(mushroom_nb_roc, legacy.axes = TRUE)
auc(mushroom_nb_roc)

###NEWRAL NET######
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, 1, 2))
maxSize <- max(nnetGrid$.size)
numWts <- (maxSize * (84 + 1) + (maxSize+1)*2) ## 84 is the number of predictors


mushroom_nnet_model <- train(x = train_data[, -which(names(train_data) == "class")],
       y = train_data$class, # Response variable,
        method = "nnet",
        metric = "ROC",
        tuneGrid = nnetGrid,
        trace = FALSE,
        maxit = 2000,
        MaxNWts = numWts,
        trControl = ctrl)
mushroom_nnet_model
plot(mushroom_nnet_model)
confusionMatrix(data = mushroom_nnet_model$pred$pred,
       reference = mushroom_nnet_model$pred$obs)

mushroom_nnet_prob <- predict(mushroom_nnet_model,
            newdata = x_test,
            type = "prob")

mushroom_nnet_pred <- predict(mushroom_nnet_model,
            newdata = x_test,
            type = "raw")

confusionMatrix(data = mushroom_nnet_pred,
       reference = test_data$class,
       positive = "e")

mushroom_nnet_roc <- roc(response = test_data$class,
          predictor = mushroom_nnet_prob$e,
          levels = rev(levels(test_data$class)))
plot(mushroom_nnet_roc, legacy.axes = TRUE)
auc(mushroom_nnet_roc)


########BEST MODEL IS NNet....EXPLORE#############
dev.close()
```

```
varImp(mushroom_nnet_model)
plot(varImp(mushroom_nnet_model))
```