Classification of Mushroom Dataset using Linear and Non-Linear Models

Predictive Modelling MA5790

Rakesh Chaganam
Krishna Mokkapati

Date: 12/06/2024

Goal of the study

The goal of our study is to support binary classification tasks for identifying mushrooms as edible or poisonous based on its physical characteristics.

The dataset is taken from UCI Machine Learning Repository.

Citation:

D. Wagner, D. Heider, and G. Hattab. "Secondary Mushroom," UCI Machine Learning Repository, 2021. [Online]. Available: https://doi.org/10.24432/C5FP5Q.

Structure of the Dataset

- Sample Size: 61,068 instances
- Response/Target Variable: Class (binary categorical: edible=e, poisonous=p)
- Number of Predictors: 20 features
  - Categorical Predictors: 17
  - Continuous Numerical Predictors: 3

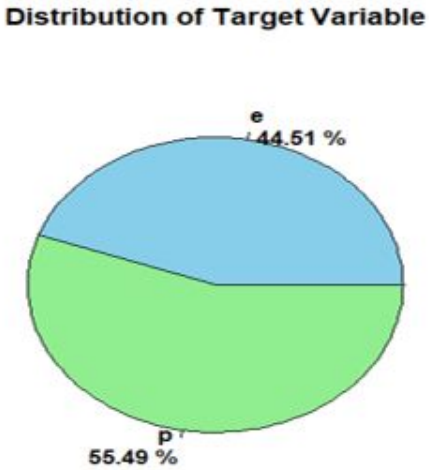
Nominal Categorical Variables (no inherent order)

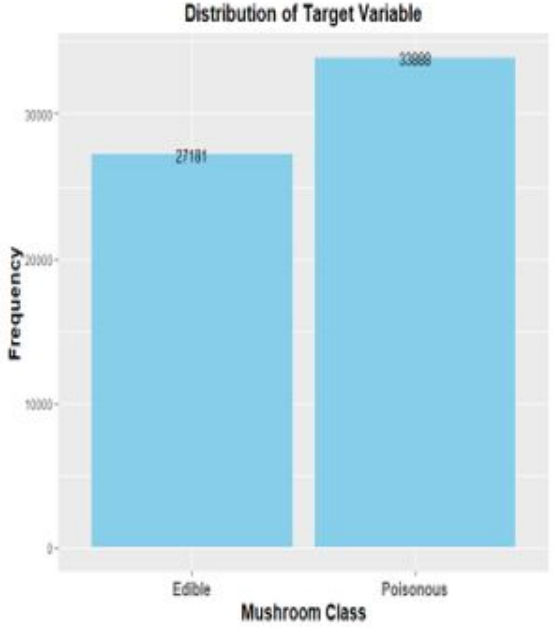
- **cap-shape (n):** bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o
- **cap-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k
- **spore-print-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k
- **gill-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f
- **veil-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f
- **stem-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f
- **habitat (n):** grasses=g, leaves=nl, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d
- **season (n)**: spring=s, summer=u, autumn=a, winter=w
- **cap-surface (n):** fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e4.
- **stem-surface (n):** fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, none=f.
- **gill-attachment (n):** adnate=a, adexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?
- **gill-spacing (n):** close=c, distant=d, none=f
- **stem-root (n):** bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r
- **veil-type (n):** partial=p, universal=u
- **ring-type (n):** cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, zone=z, movable=m, none=f, unknown=?
- **does-bruise-bleed (n):** bruises-or-bleeding=t, no=f
- **has-ring (n):** ring=t, none=f

Continuous Numerical Variable

- **Cap-diameter:** float number in cm
- **stem-height:** float number in cm
- **Stem-width:** float number in mm

Visualization of Target Class Distribution





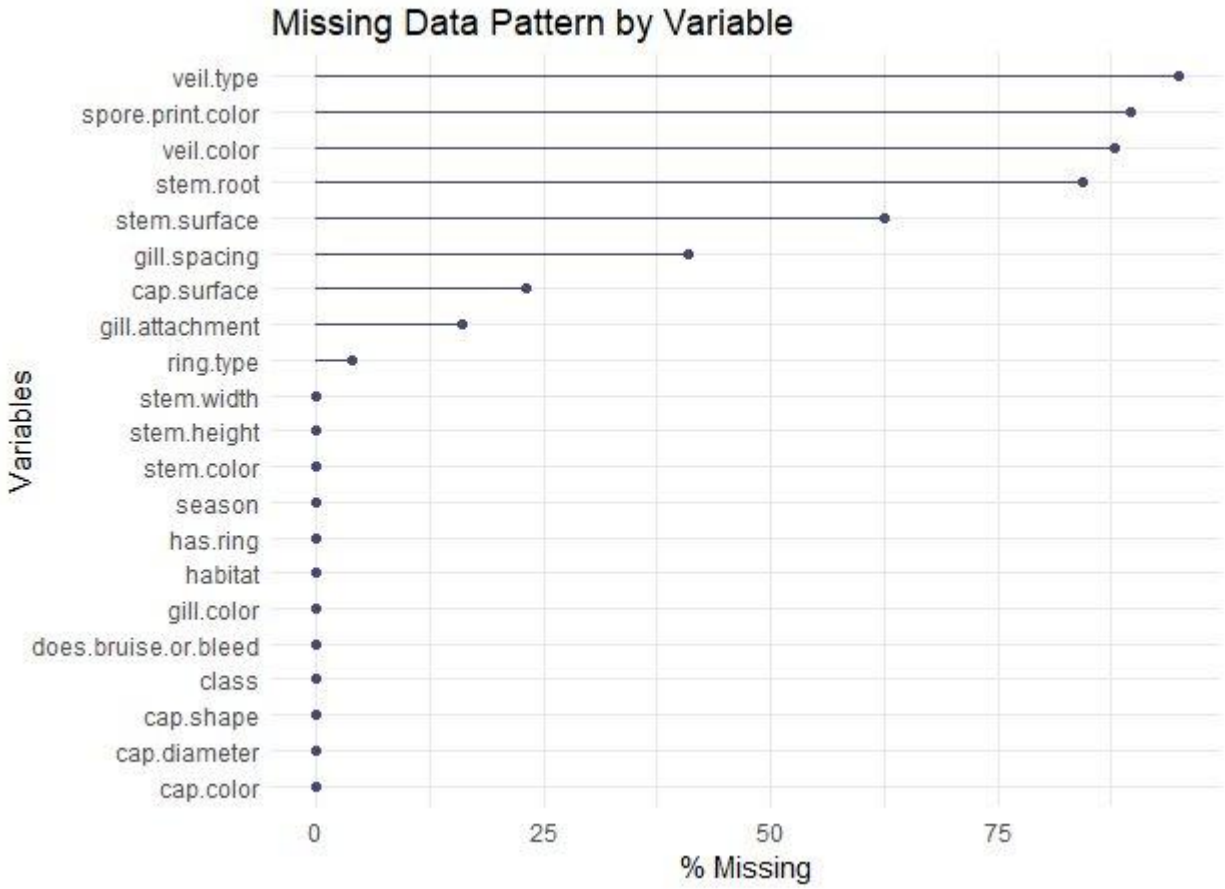
Class Distribution:

- Edible (e): 27,181 (44.6%)
- Poisonous (p): 33,888 (55.4%)
- Majority of the mushrooms are poisonous. Also, the class is slightly imbalanced, it's not extreme enough to be a major concern for modeling.
- With the slight class imbalance (44.6% edible and 55.4% poisonous), it seems manageable

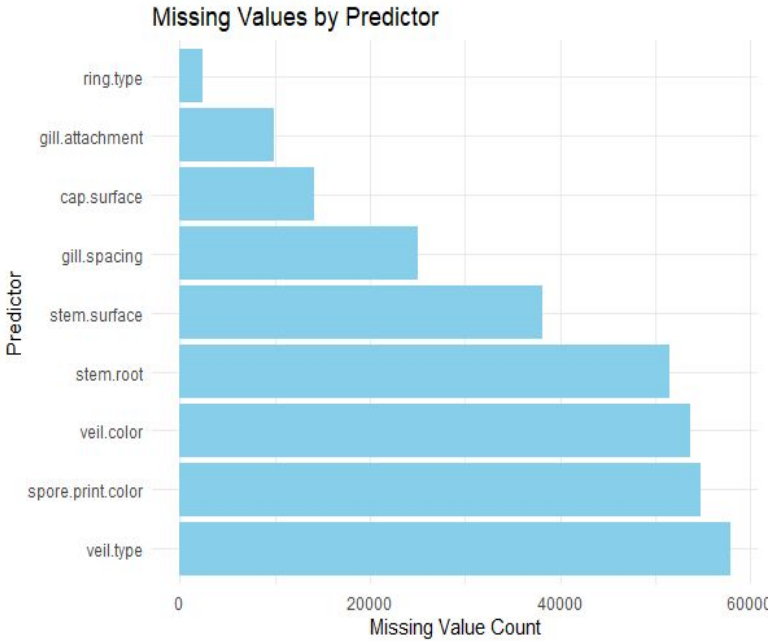
Handling Missing Values

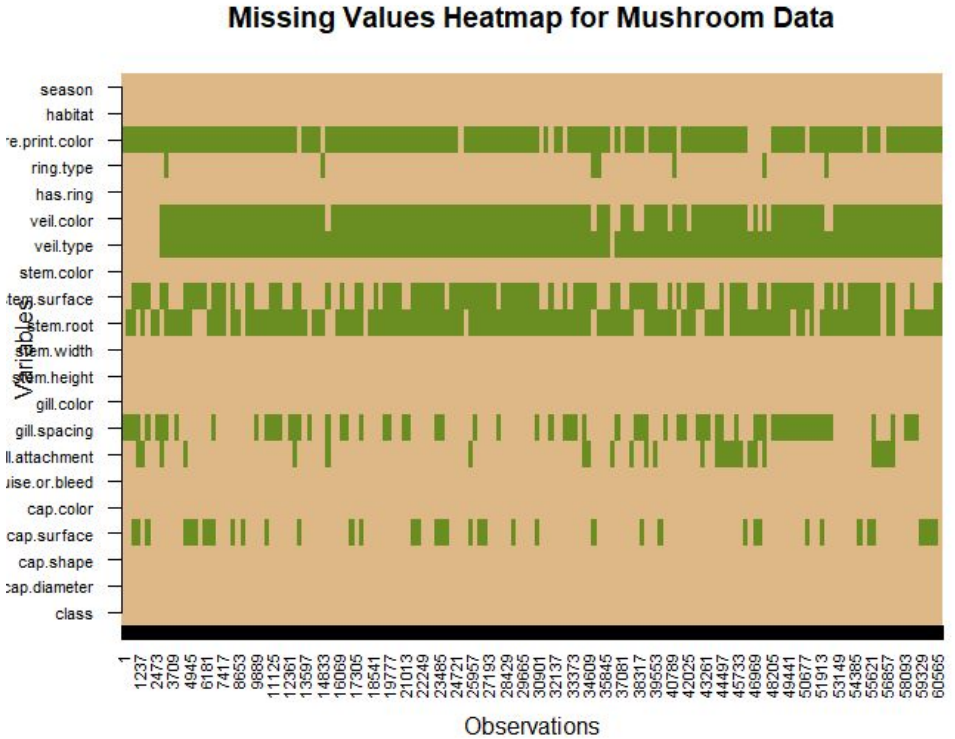
Variables sorted by number of

Variable	Count
veil.type	57892
spore.print.color	54715
veil.color	53656
stem.root	51538
stem.surface	38124
gill.spacing	25063
cap.surface	14120
gill.attachment	9884
ring.type	2471
class	0
cap.diameter	0
cap.shape	0
cap.color	0
does.bruise.or.bleed	0
gill.color	0
stem.height	0
stem.width	0
stem.color	0
has.ring	0
habitat	0
season	0



> Removed the columns which are having missing values above 60%





- After removing the predictors which have missing value more than 60%, we left with **16 observations** including target "Class".

```
> str(mushroom)
'data.frame':	61069 obs. of  16 variables:
 $ class              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ cap.diameter       : num  15.3 16.6 14.1 14.2 14.6 ...
 $ cap.shape          : chr  "x" "x" "x" "f" ...
 $ cap.surface        : chr  "g" "g" "g" "h" ...
 $ cap.color          : chr  "o" "o" "o" "e" ...
 $ does.bruise.or.bleed: chr  "f" "f" "f" "f" ...
 $ gill.attachment    : chr  "e" "e" "e" "e" ...
 $ gill.spacing       : chr  "c" "c" "c" "c" ...
 $ gill.color         : chr  "w" "w" "w" "w" ...
 $ stem.height        : num  16.9 18 17.8 15.8 16.5 ...
 $ stem.width         : num  17.1 18.2 17.7 16 17.2 ...
 $ stem.color         : chr  "w" "w" "w" "w" ...
 $ has.ring           : chr  "t" "t" "t" "t" ...
 $ ring.type          : chr  "g" "g" "g" "p" ...
 $ habitat            : chr  "d" "d" "d" "d" ...
 $ season             : chr  "w" "u" "w" "w" …
```

- After successfully applying **KNN imputation** method on missing variables, we found no missing observations in the data.

```
> sapply(mushroom[, sapply(mushroom, is.character)], function(x) sum(is.na(x)))
            class         cap.shape       cap.surface         cap.color
                0                 0                 0                 0
does.bruise.or.bleed    gill.attachment      gill.spacing        gill.color
                0                 0                 0                 0
       stem.color         has.ring          ring.type           habitat
                0                 0                 0                 0
           season
                0
```

Dealing with categorical Predictors

```
# Create a subset of the predictor variables (exclude 'class')
> mushroom_predictors <- mushroom[, -which(names(mushroom) == "class")]
```

Binary Encoding:

- We applied **label encoding** (0/1) to the two binary variables: `does.bruise.or.bleed` and `has.ring`.

One-Hot Encoding:

- For the remaining categorical variables, We used **one-hot encoding** with `model.matrix()`, converting them into a numerical format where each category gets its own column.

Final predictor data:

- After encoding, all the 15 predictors are now numerical, which results into a total of 81 predictors including continuous numerical variables.

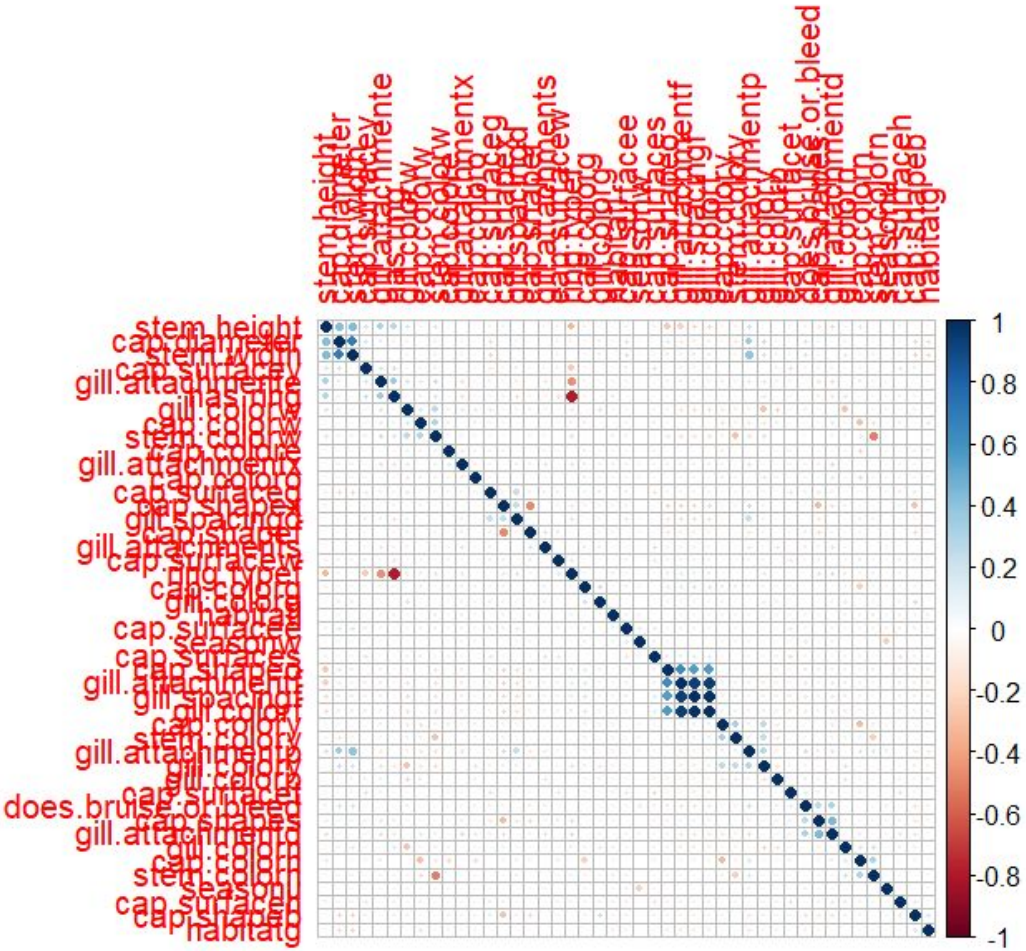
```
> str(mushroom_predictors)
'data.frame':  61069 obs. of  81 variables:
```

Handling Near Zero Variance Predictors and highly Correlated

```
> rownames(nzv[nzv$nzv == TRUE, ])
 [1] "cap.shapec"   "cap.shapep"   "cap.surfacei" "cap.surfacek" "cap.surfacel" "cap.colork"
 [7] "cap.colorl"   "cap.colorp"   "cap.colorr"   "cap.coloru"   "gill.colore"  "gill.colork"
[13] "gill.coloro"  "gill.colorr"  "gill.coloru"  "stem.colore"  "stem.colorf"  "stem.colorg"
[19] "stem.colork"  "stem.colorl"  "stem.coloro"  "stem.colorp"  "stem.colorr"  "stem.coloru"
[25] "ring.typeg"   "ring.typel"   "ring.typem"   "ring.typep"   "ring.typer"   "ring.typez"
[31] "habitath"     "habitatm"     "habitatp"     "habitatu"     "habitatw"     "seasons"

> colnames(mushroom_predictors)[high_corr]
[1] "gill.colorf"      "gill.attachmentf"

> mushroom_predictors <- mushroom_predictors[, !nzv$nzv]
> str(mushroom_predictors)
'data.frame':  61069 obs. of  43 variables:
```

- Before applying transformations, centering and scaling we removed near zero variance. Because these variables don't add much value to the model and might distort the subsequent transformations.
- We had a total of 45 predictors after removing the nZV predictors.
- After removing Highly correlated variables, the predictors dataset reduces dimensionality and removed redundant features, that making our dataset more efficient for processing and modeling.
- Finally we had 43 predictor variables including continuous numerical predictors.

Correlation structure of the predictor data



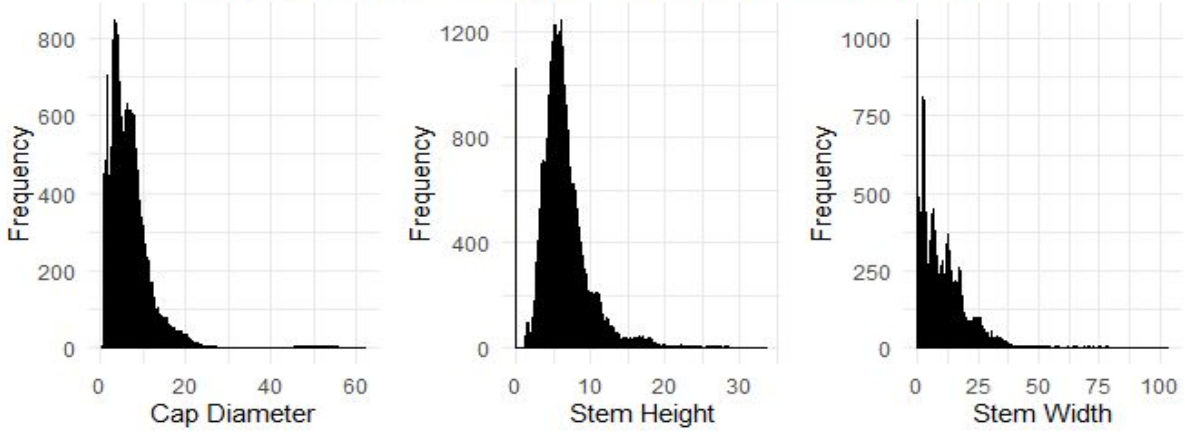
Box-Cox transformation for Skewness

Applied Box-Cox transformation to reduce the skewness. Also applied centering, and scaling to standardize the data using the **preProcess()** func for continuous numerical variables.

- **Box-Cox**: Will transform your continuous variables to reduce skewness.
- **Center**: Subtracts the mean of each variable to center them at 0.
- **Scale**: Divides by the standard deviation to standardize the variables (variance = 1).

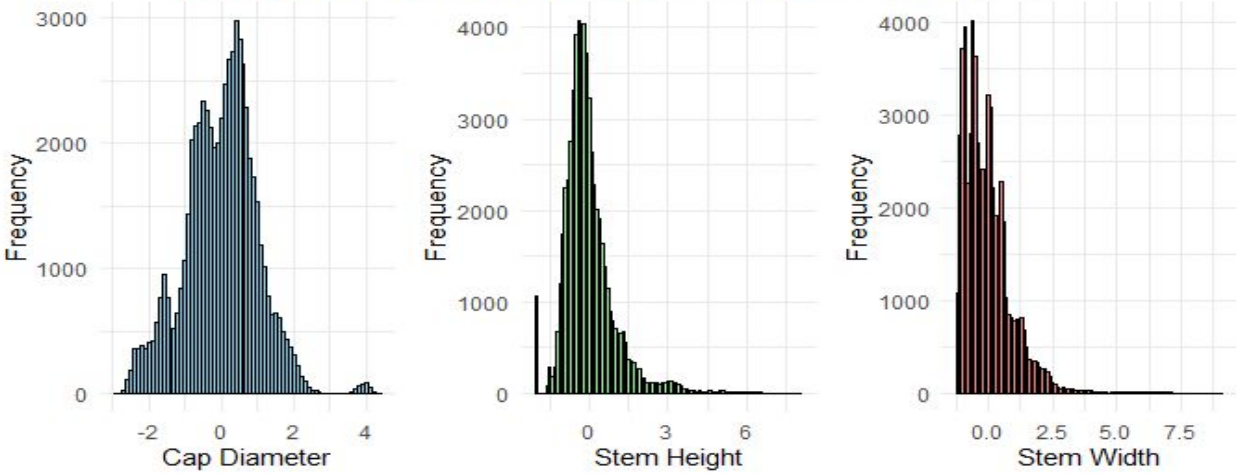
Then, checked the distribution of your continuous variables to verify that the skewness has been reduced and that they are properly centered and scaled.

Histograms of Continuous numerical Predictors



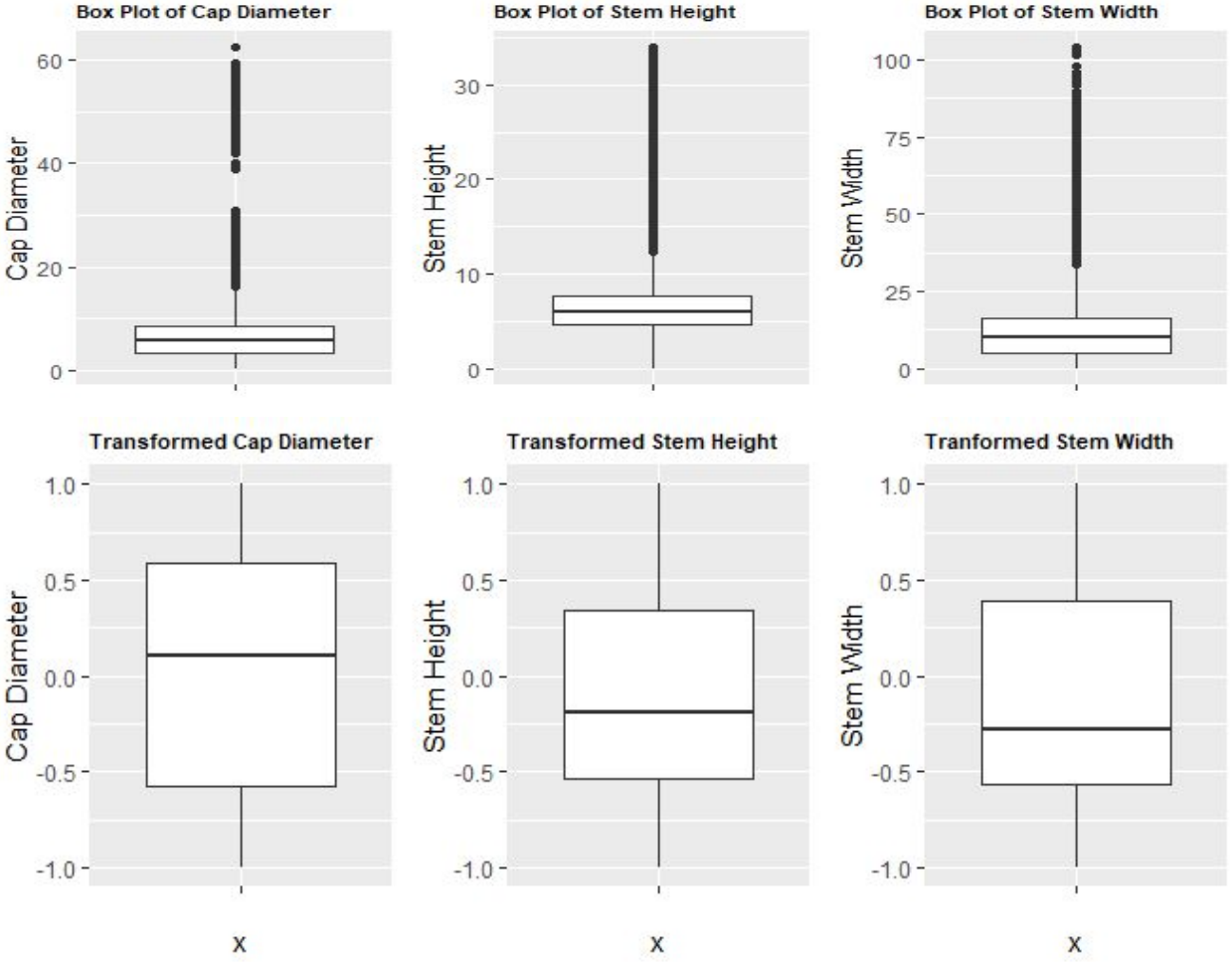

```
> skewness_table
                 Predictor Skewness
cap.diameter cap.diameter 3.822750
stem.height   stem.height 2.020854
stem.width     stem.width 2.164904
```

Histograms of Transformed Continuous Predictors




```
> skewness_table
                 Predictor  Skewness
cap.diameter cap.diameter 0.0485854
stem.width     stem.width 2.1649035
stem.height   stem.height 2.0208540
```

Spatial sign transformation to reduce outliers



Finally, To reduce the impact of outliers on our continuous numerical variables using **Spatial Sign Transformation**, We used the `spatialSign()` function from the **caret** package in R.

Spatial sign transformation rescaled the data to the unit circle and greatly reduced the influence of outliers.

Data Splitting

We split our data into 75% training and 25% testing using **stratified random sampling**, which preserves the class proportions in both sets.

```
> table(train_labels)
train labels
    0     1
23653 19096
> table(test_labels)
test labels
    0     1
10235  8085
```

Resampling

train_control <- trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary, classProbs = TRUE)

- We used **10-fold cross-validation** for model evaluation, which is a robust and appropriate method for this type of binary classification problem.
- Instead of training the model on just one part of the data, cross-validation splits the training data into **10 equal parts**
- The average of the performance across all 10 folds is used as the final performance measure that prevents overfitting and we can achieve robust performance of model.

Linear Models:

Logistic Regression

Linear Discriminant Analysis

Partial Least Square Discriminant Analysis

Penalized Model

Non-Linear Models:

Quadratic Discriminant Analysis

Regularized Discriminant Analysis

Mixture Discriminant Analysis

Neural Networks

Flexible Discriminant Analysis

K- Nearest Neighbors

Support Vector Machine

Naive Bayes

Note: PLSDA, Penalized, FDA, KNN, SVM, Naive Bayes
These models are trained on the data that includes highly correlated predictors.

Logistic Regression

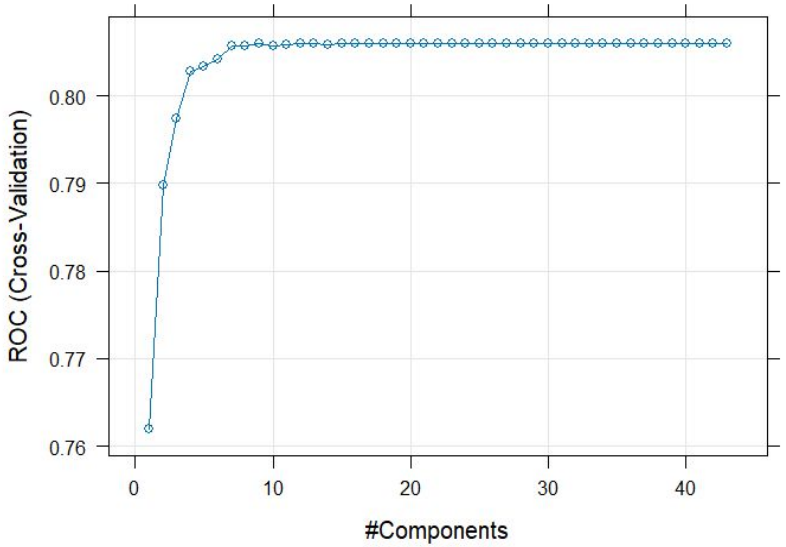
```
> lrFull
Generalized Linear Model

45802 samples
   43 predictor
    2 classes: 'e', 'p'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 41222, 41222, 41222, 41221, 41221, 41221, ...
Resampling results:

  ROC        Sens       Spec
  0.8050311  0.6972236  0.7607728
```

Linear Discriminant Analysis

```
> print(lda_model)
Linear Discriminant Analysis

45802 samples
   43 predictor
    2 classes: 'e', 'p'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 41222, 41221, 41223, 41221, 41221, 41222, ...
Resampling results:

  ROC        Sens       Spec
  0.8052238  0.7041398  0.7574437
```

Partial Least Square Discriminant Analysis

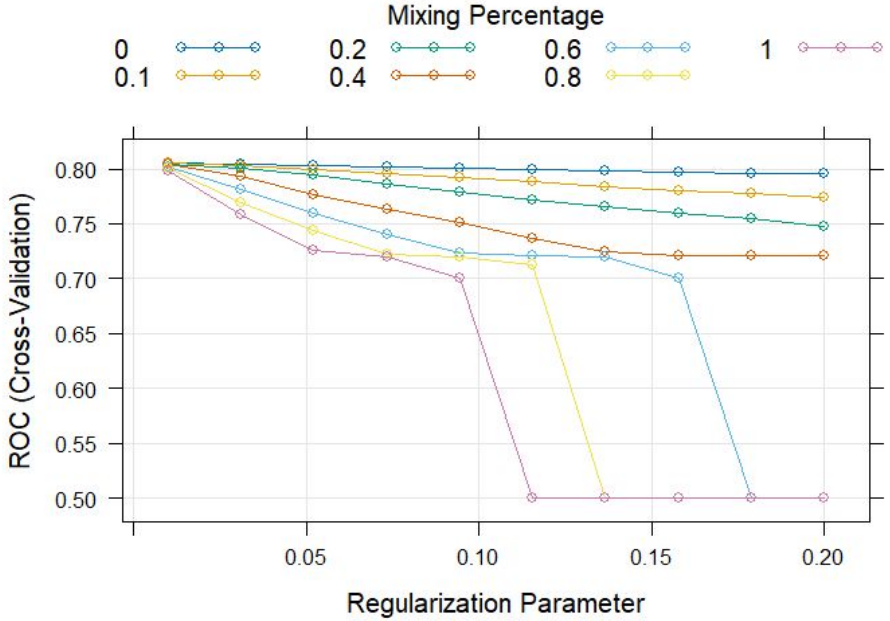
```
> plsda model
Partial Least Squares

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41222, 41221, 41223, 41221,
Resampling results across tuning parameters:

  ncomp  ROC        Sens       Spec
   1     0.7619587  0.6445573  0.7623539
   2     0.7898001  0.6815940  0.7516917
   3     0.7973707  0.6889530  0.7543673
   4     0.8027896  0.7113214  0.7563740
   5     0.8032847  0.7026385  0.7516133
   6     0.8041932  0.7021971  0.7587741
   7     0.8057072  0.7044533  0.7579086
   8     0.8057551  0.7063175  0.7563742
   9     0.8059697  0.7066116  0.7568464
  10     0.8057711  0.7068569  0.7567677
  11     0.8058985  0.7076417  0.7573185
  12     0.8059594  0.7068078  0.7568464
  13     0.8059350  0.7065625  0.7567284
  14     0.8058993  0.7069059  0.7568857
  15     0.8059314  0.7063174  0.7573579

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



Penalized Model

```
> glmnet_model
glmnet

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41221, 41221, 41223, ...
Resampling results across tuning parameters:
```

alpha	lambda	ROC	Sens	Spec
0.0	0.01000000	0.8052286	0.69277923	0.766



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.**

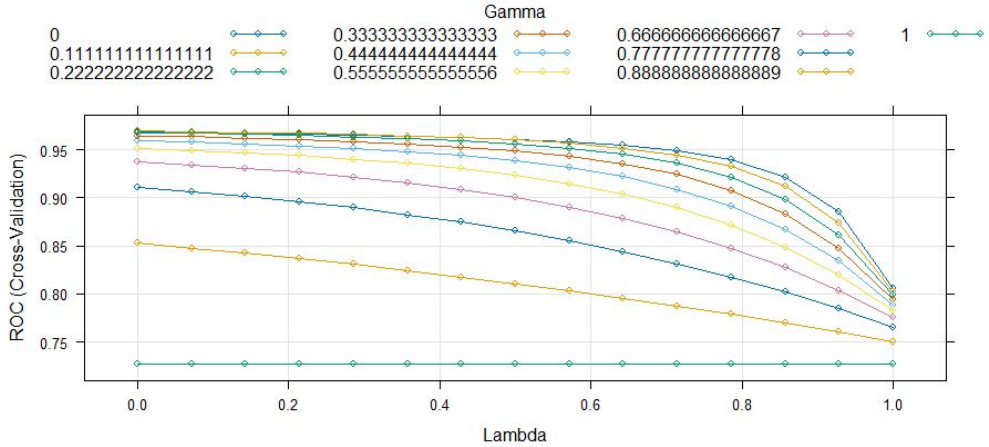
Quadratic Discriminant Analysis

```
> print(qda_model)
Quadratic Discriminant Analysis

45802 samples
   43 predictor
    2 classes: 'e', 'p'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5
times)
Summary of sample sizes: 41221, 41221, 41221 , 41223,
...
Resampling results:
```

ROC	Sens	Spec
0.9686081	0.9055722	0.9089156

Regularized Discriminant Analysis

```
> print(rda_model)
Regularized Discriminant Analysis

45802 samples
   43 predictor
    2 classes: 'e', 'p'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41222, ...
Resampling results across tuning parameters:
```

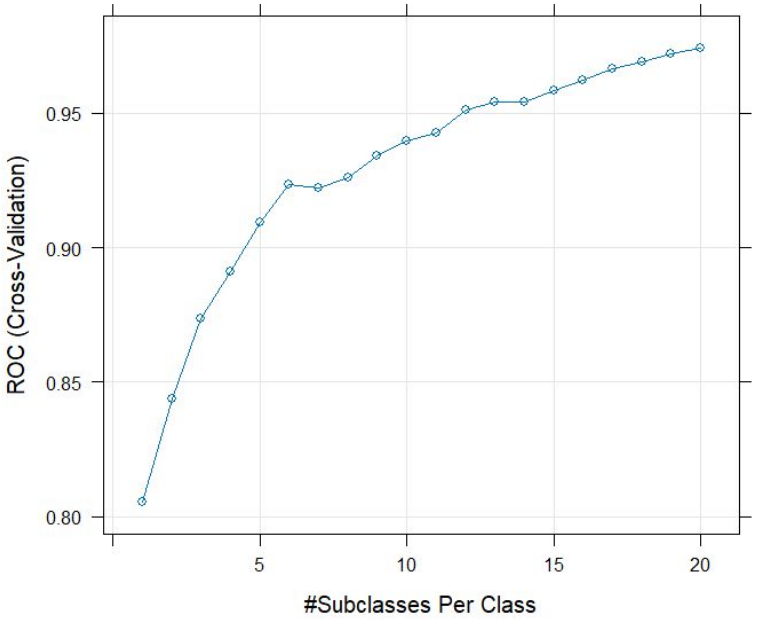


gamma	lambda	ROC	Sens	Spec
0.1111111	0.00000000	0.9695277	0.9141570	0.9097422

```
ROC was used to select the optimal model using the largest value.
The final values used for the model were  gamma = 0.1111111 and lambda = 0.
```

Mixture Discriminant Analysis

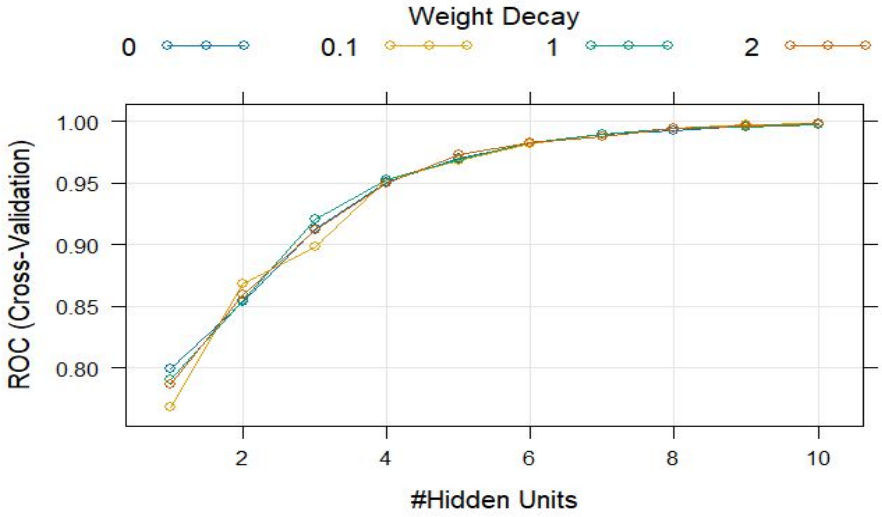
```
> print(mda model)
Mixture Discriminant Analysis

45802 samples
   43 predictor
    2 classes: 'e', 'p'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41223,...
Resampling results across tuning parameters:

  subclasses  ROC        Sens       Spec
   1          0.8052233  0.7041105  0.7578692
   2          0.8438200  0.7488468  0.7917450
   3          0.8735766  0.7610628  0.8358116
   4          0.8911137  0.7912766  0.8468676
   5          0.9095430  0.8126130  0.8644550
   6          0.9235696  0.8286544  0.8791699
   7          0.9221555  0.8332196  0.8764568
   8          0.9259304  0.8308139  0.8858610
   9          0.9340407  0.8440619  0.8972309
  10          0.9396715  0.8566657  0.9047856
  11          0.9428186  0.8633850  0.8947910
  12          0.9512827  0.8742740  0.9090729
  13          0.9542695  0.8823670  0.9052567
  14          0.9540724  0.8834002  0.9071828
  15          0.9584486  0.8819278  0.9185949
  16          0.9623158  0.8938953  0.9191066
  17          0.9667747  0.9037092  0.9241017
  18          0.9690277  0.9100870  0.9228034
  19          0.9719353  0.9065557  0.9258339
  20          0.9744045  0.9149894  0.9321682

ROC was used to select the optimal model using the largest value.
The final value used for the model was  subclasses = 20.
```



Neural Network Model

```
> print(nnet_model)
Neural Network

45802 samples
   43 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (43), scaled (43)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41222,...
Resampling results across tuning parameters:
```

size	decay	ROC	Sens	Spec
10	0.1	0.9982541	0.9826839	0.9882362

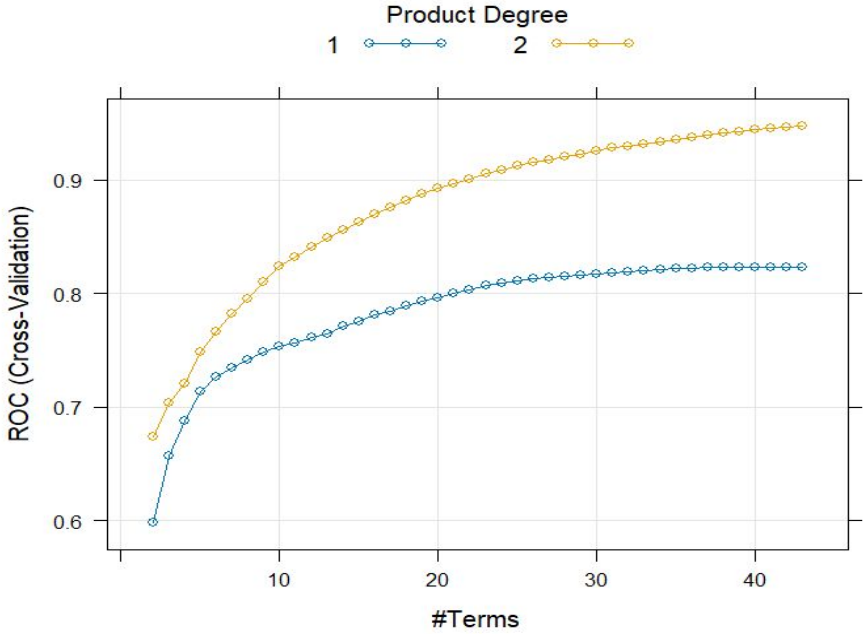


ROC was used to select the optimal model using the largest value.
The final values used for the model were **size = 10** and **decay = 0.1.**

Flexible Discriminant Analysis

```
> print(fda_model)
Flexible Discriminant Analysis

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41222, ...
Resampling results across tuning parameters:
```

degree	nprune	ROC	Sens	Spec
2	43	0.9474637	0.8344440	0.9215838

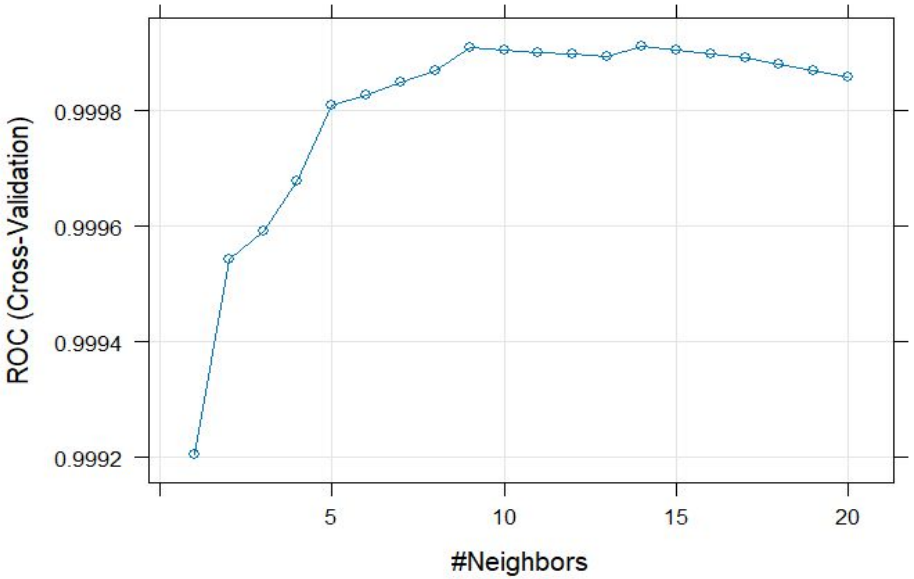


ROC was used to select the optimal model using the largest value.
The final values used for the model were **degree = 2** and **nprune = 43.**

K-Nearest Neighbors

```
> knn_model
k-Nearest Neighbors

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41221, 41222, 41222, ...
Resampling results across tuning parameters:
```

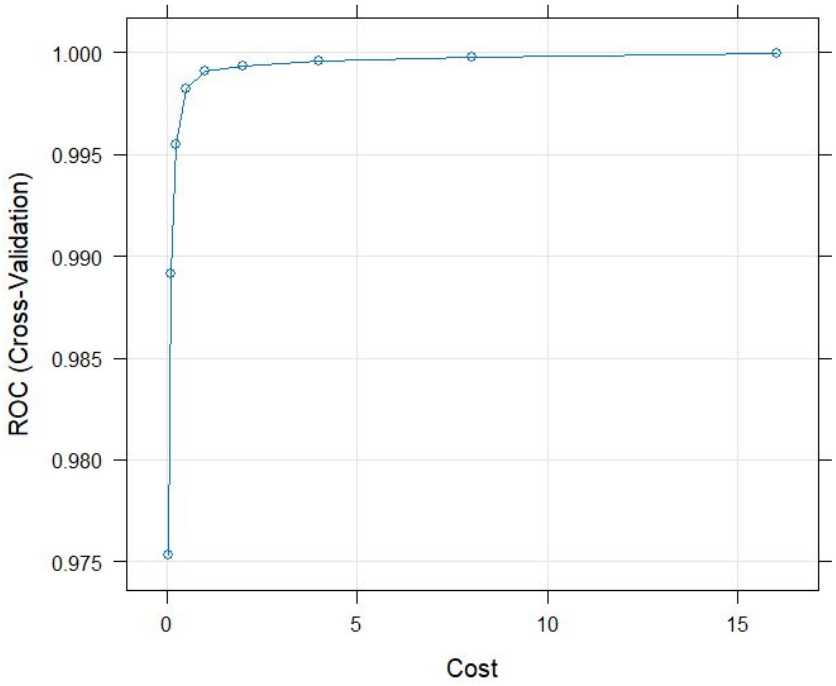
k	ROC	Sens	Spec
14	0.9999114	0.9952910	0.9973245



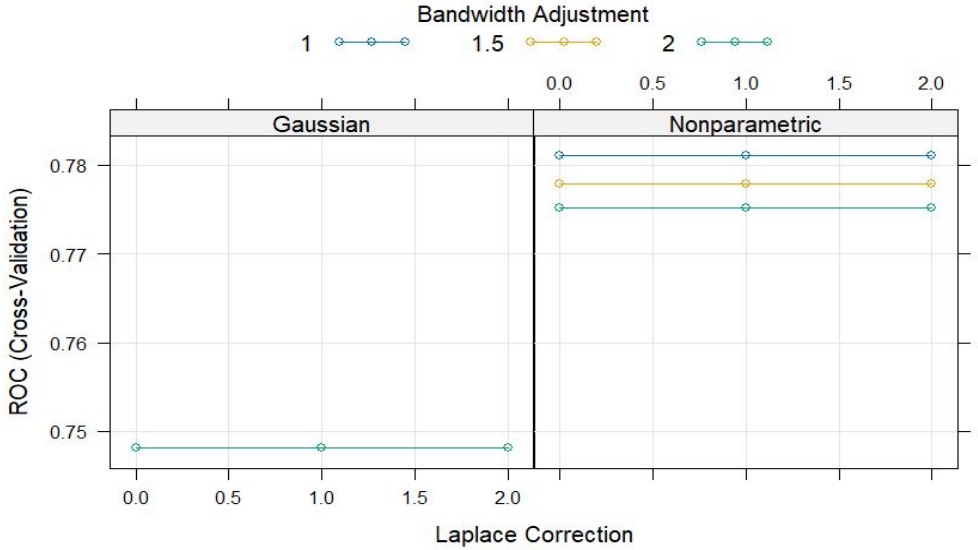
ROC was used to select the optimal model using the largest value.
The final value used for the model was **k = 14.**

Support Vector Machine

```
> svm_model
Support Vector Machines with Radial Basis Function Kernel

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41221, 41221, 41223, ...
Resampling results across tuning parameters:

  C        ROC        Sens       Spec
   0.0625  0.9753128  0.9093500  0.9429885
   0.1250  0.9891212  0.9372615  0.9588842
   0.2500  0.9954955  0.9634063  0.9819012
   0.5000  0.9982293  0.9808692  0.9867014
   1.0000  0.9990738  0.9902876  0.9926425
   2.0000  0.9993332  0.9943099  0.9948852
   4.0000  0.9995605  0.9954871  0.9966557
   8.0000  0.9997691  0.9970077  0.9976786
  16.0000  0.9999773  0.9986264  0.9983081

Tuning parameter 'sigma' was held constant at a value of 0.007983024
ROC was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.007983024 and C = 16.
```



Naive Bayes Model




```
> nb_model
Naive Bayes

45802 samples
   45 predictor
    2 classes: 'e', 'p'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 41222, 41222, 41222, ...
Resampling results across tuning parameters:
```

fL	usekernel	adjust	ROC	Sens	Spec
0	TRUE	1.0	0.7810282	0.3721176	0.8893610

ROC was used to select the optimal model using the largest value.
The final values used for the model were **fL = 0, usekernel = TRUE** and **adjust = 1.**

Model Training Results

Linear Models:

Model	Tuning Parameter	Training ROC
Logistic Regression	N/A	0.8050
LDA	N/A	0.8052
PLSDA	Ncomp = 9	0.8059
Penalized	alpha = 0, lambda = 0.01	0.8052

Non-Linear Models:

Model	Tuning parameter	Training ROC
QDA	N/A	0.9686
RDA	gamma = 0.111 and lambda = 0	0.9695
MDA	subclasses = 20	0.9392
Neural Network	size = 10 and decay = 0.1	0.9982
FDA	degree = 2, nprune = 43	0.9474
K-NN	k = 14	0.9999
SVM	sigma = 0.00798(const), c = 16	0.9999
Naive Bayes	Const at (fL = 0, usekernel = TRUE and adjust = 1)	0.7810

Best Model:

All the models performed well; But based on ROC score we found,both KNN and SVM models performed very well than other models during training.

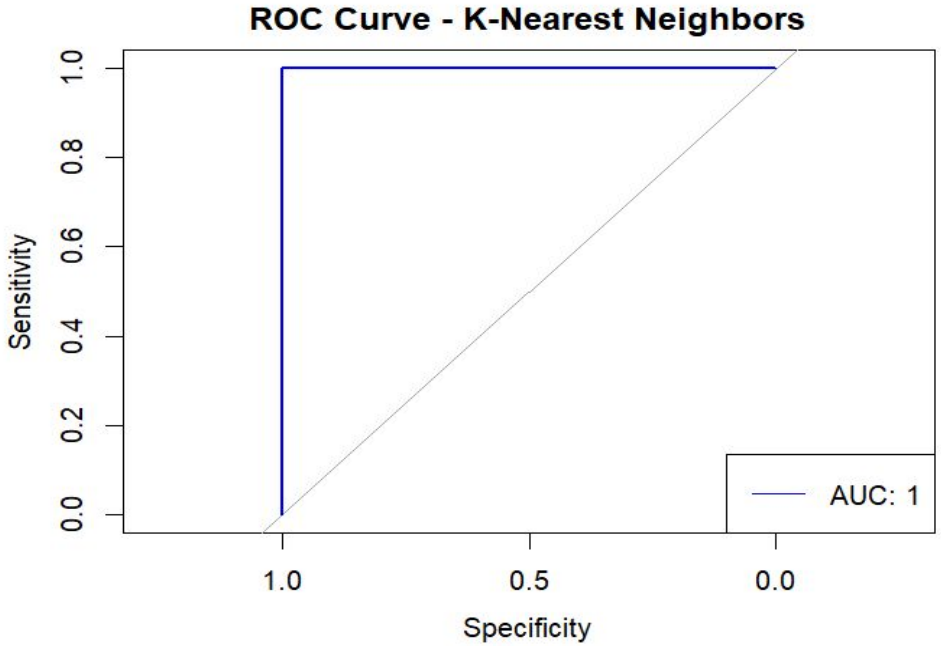
K-NN and **SVM** are tied for the best training performance with **ROC = 0.9999**.

After these, the **Neural Network** model also demonstrated strong performance with an ROC score of **0.9982**.

Test performance on best performed models

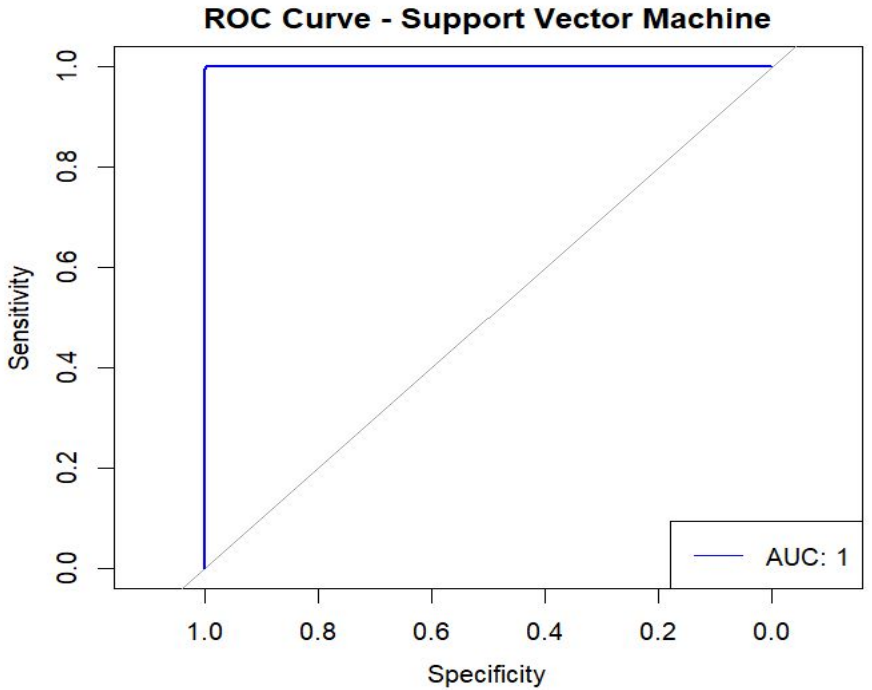
K-NN Predictions

```
> confusionMatrix(knn_predictions,
testing set$class)
Confusion Matrix and Statistics

          Reference
Prediction    e    p
         e 6772   14
         p   23 8458

               Accuracy : 0.9976
                 95% CI : (0.9967, 0.9983)
    No Information Rate : 0.5549
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9951

 Mcnemar's Test P-Value : 0.1884

            Sensitivity : 0.9966
            Specificity : 0.9983
         Pos Pred Value : 0.9979
         Neg Pred Value : 0.9973
             Prevalence : 0.4451
         Detection Rate : 0.4436
   Detection Prevalence : 0.4445
      Balanced Accuracy : 0.9975

       'Positive' Class : e
```



SVM Predictions

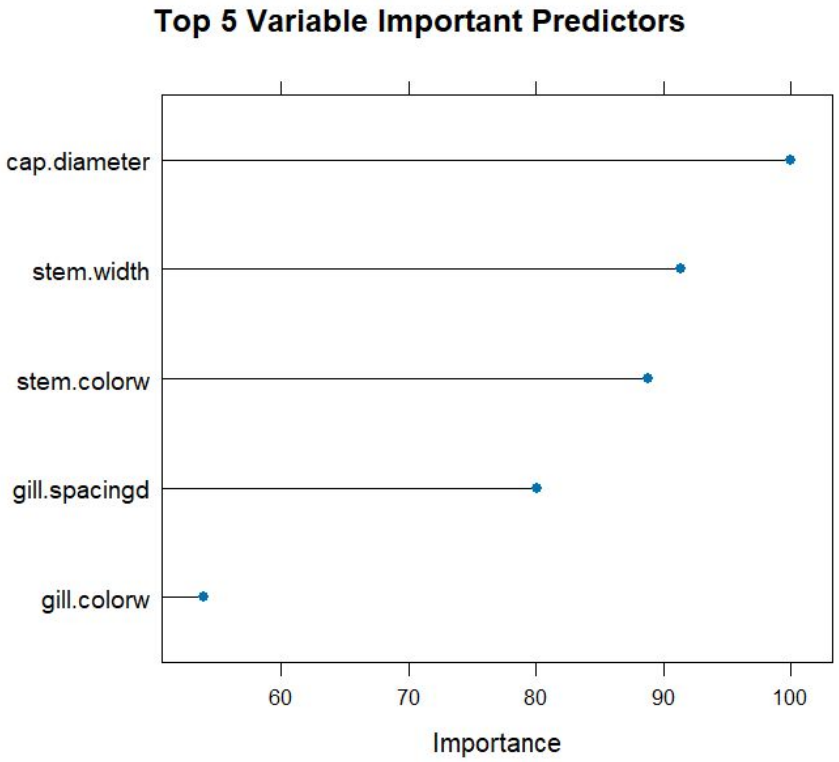
```
> confusionMatrix(svm_predictions, testing_set$class)
Confusion Matrix and Statistics

          Reference
Prediction    e    p
         e 6780   17
         p   15 8455

               Accuracy : 0.9979
                 95% CI : (0.997, 0.9986)
    No Information Rate : 0.5549
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9958

 Mcnemar's Test P-Value : 0.8597

            Sensitivity : 0.9978
            Specificity : 0.9980
         Pos Pred Value : 0.9975
         Neg Pred Value : 0.9982
             Prevalence : 0.4451
         Detection Rate : 0.4441
   Detection Prevalence : 0.4452
      Balanced Accuracy : 0.9979

       'Positive' Class : e
```



Most Important Variables Predictors

> Considering SVM as the best model, the Important Predictors are;

```
> varImp(knn model)
ROC curve variable importance

  only 20 most important variables shown (out of 45)

                  Importance
cap.diameter          100.00
stem.width             91.41
stem.colorw            88.77
gill.spacingd          80.08
gill.colorw            53.91
gill.attachmentp       52.90
cap.colorn             47.13
gill.colorn            38.50
cap.shapeb             36.45
cap.surfaces           30.41
habitatg               29.24
stem.colory            28.40
cap.shapex             27.43
cap.colore             26.02
seasonw                24.38
stem.height            23.67
cap.surfacet           21.84
stem.colorn            21.11
has.ring               20.20
cap.shapeo             20.14
```



Thank You