Terrance Taubes (G36790243) - Noura Azeem (G31005999) - Omar Abdeen (G27205220)

Professor Kaisler

CSCI 3907_80 Big Data and Analytics (Undergraduate)

6 November 2017

Class Project #2 Deliverable

Description of Mushroom Classes

The original classification of the mushroom set classified 4208 mushrooms as edible and 3916 as poisonous. At the time of classification, a third group of mushrooms was unable to be classified and was included in the poisonous cluster. We will be using K-Means, PAM Clustering, and Hierarchical Clustering (hclust) to classify our data. The mushroom data will be reclassified into 3, 5, and 7 clusters, with classifications based on each mushroom's attributes.

R Functions Documentation

Initialization

- Imports
    - ➢ *library(igraph)*
    - ➢ *library(cluster)*
    - ➢ *library(psych)*
    - ➢ *library(e1071)*
    - ➢ *library(rgl)*
    - ➢ *library(RColorBrewer)*
    - ➢ *library(scales)*
    - ➢ *library(kernlab)*
    - ➢ *require(graphics)*

- Load/Handle Data
    - ➢ *mush_data <- read.csv("/path/to/files/", header=FALSE)*
    - ➢ *names(mush_data) <- c("class", "cshape", "csurface", "ccolor", "bruises", "odor", "gattach", "gspace", "gsize", "gcolor", "sshape", "sroot", "ssabove", "ssbelow",*

"scabove", "scbelow", "vtype", "vcolor", "rnumber", "rtype", "spcolor", "popnum", "habitat")

  - Load data to 'mush_data' and add attribute names.

➢ *mush_data.intClass <- mush_data*
➢ *mush_data.intClass[] <- as.numeric(factor(as.matrix(mush_data)))*

  - Create matrix with factor numbers replacing character values from 'mush_data'.
  - 'mush_data.intClass' preserves the 'class' column.

➢ *mush_data.int <- mush_data.intClass*
➢ *mush_data.int$class <- NULL*

  - 'mush_data.int' removes the 'class' column.

Plotting 2D / 3D Data + Filtering Data

- Plot pairs/groups of attributes and compare correlations.

➢ *pairs(class ~ cshape + csurface + ccolor, data=mush_data)*

- 2D plots of attributes and their relation to their classification.

➢ *plot(class ~ cshape + csurface + ccolor, data=mush_data)*

- 3D plots of attributes and their correlations.

➢ *plot3d(mush_data$cshape, mush_data$csurface, mush_data$ccolor)*

Principal Component Analysis

➢ *mush_pca <- princomp(mush_data.int)*

  - Get principal component values for 'mush_data.int'

➢ *plot(mush_pca, main="Mushroom Data Scree Plot")*

  - Scree plot of principal components.

➢ *mush_pca.vec = prcomp(mush_data.int)*

➢ *mush_model <- data.frame(mush_pca.vec$x[,1:3])*

  - PCA Component Vectors (the model)

Statistical Analysis

➢ *summary(mush_data)*

  - Statistical information on Mushroom data.

➢ *describe(mush_data)*

- More statistical information on Mushroom data.

➢ *summary(mush_pca)*

- Statistical information on PCA model.

➢ *loadings(mush_pca)*

- Component vector loadings.

Training and Test Sets

- Create sampling function for splitting data.

➢ *train.50 <- sample(nrow(mush_model), 0.5*nrow(mush_model))*

- Create training and testing data object without classifications.

(*mush_50* : object for 50-50 train/test,

mush_60 : object for 60-40 train/test,

mush_70 : object for 70-30 train/test)

➢ *mush_50 <- list()*
➢ *mush_50$train <- mush_model[train.50,]*
➢ *mush_50$test <- mush_model[-train.50,]*

- Create training and testing data object with classifications.

➢ *mush_50Class <- list()*
➢ *mush_50Class$train <- mush_data.intClass[train.50,]*
➢ *mush_50Class$test <- mush_data.intClass[-train.50,]*

Clustering

<u>K-Means</u>

- K-Means for k=2 (Repeat for k=3, 5, 7)

➢ *km2 <- kmeans(mush_model, 2)*
➢ *km2.50 <- kmeans(mush_50$train, 2)*
➢ *km2.60 <- kmeans(mush_60$train, 2)*
➢ *km2.70 <- kmeans(mush_70$train, 2)*

- Plotting K-Means Clusters for k=2 (Repeat for k=3, 5, 7)

- ➢ *plot(mush_model, col=km2$clust, pch=16)*
- ➢ *plot3d(mush_model$PC1, mush_model$PC2, mush_model$PC3, col=km2$clust)*

PAM Clustering

- Pam Clustering for k=3 (Repeat for k=5, 7)
  - ➢ *pam.3 <- pam(mush_model, 3)*
  - ➢ *mush_50.pam.3 <- pam(mush_50.df, k=3)*
  - ➢ *mush_60.pam.3 <- pam(mush_60.df, k=3)*
  - ➢ *mush_70.pam.3 <- pam(mush_70.df, k=3)*

- Plotting PAM Clusters for k=3 (Repeat for k=5, 7)
  - ➢ *plot(mush_model, col = hclust.3)*
  - ➢ *plot3d(mush_model$PC1, mush_model$PC2, mush_model$PC3, col = hclust.3)*

HClust Clustering

- HClust Clustering for k=[3, 5, 7] (Repeat for 60-40, 70-30 Train/Test sets)
  - ➢ *mush_50.df <- as.data.frame(mush_50$train)*
  - ➢ *mush_50_dist <- dist(mush_50.df, method = "euclidean")*
  - ➢ *mush_50.hc.3 <- cutree(hclust(mush_50_dist), k=3)*
  - ➢ *mush_50.hc.5 <- cutree(hclust(mush_50_dist), k=5)*
  - ➢ *mush_50.hc.7 <- cutree(hclust(mush_50_dist), k=7)*

- Getting Classification Counts for k=3 (Repeat for k=5, 7)
  - ➢ *table(mush_50.hc.3)*
  - ➢ *table(mush_60.hc.3)*
  - ➢ *table(mush_70.hc.3)*

- Plotting HClust Clusters for k=3 (Repeat for k=5, 7)
  - ➢ *plot(mush_model, col = hclust.3)*
  - ➢ *plot3d(mush_model$PC1, mush_model$PC2, mush_model$PC3, col = hclust.3)*
  - ➢ *plot(hclust(mush_50_dist), h=1)*
    - ○ Hierarchical Plot of HClust

LM and GLM Methods

- LM

- ➢ *lm.fit <- lm(mush_data.int$cshape ~ ., data=mush_data.int)*
  - ○ Fit linear model to mushroom cap shape.
- ➢ *summary(lm.fit)*
  - ○ Cap shape explained the least amount of variability.
- ➢ *step(lm.fit)*
- ➢ *lm.fit <- lm(mush_data.int$gattach ~ ., data=mush_data.int)*
  - ○ Fit linear model to mushroom gill attachment.
- ➢ *summary(lm.fit)*
  - ○ Gill attachment explained the most amount of variability.
- ➢ *step(lm.fit)*
- ➢ *plot(lm.fit)*

- GLM

- ➢ *glm(formula = mush_data.int$cshape ~ ccolor + bruises + odor + gattach + gsize + gcolor + sshape + sroot + scabove + scbelow + rtype + spcolor + popnum, family = "binomial", data = mush_data.int)*

Support Vector Machine

- Create SVM models for each train/test set.

(*svm50* : model for 50-50 train/test,

svm60 : model for 60-40 train/test,

svm70 : model for 70-30 train/test)

- ➢ *svm50 <- ksvm(class~., data=mush_50Class$train)*
  - ○ SVM 50-50 Model
- ➢ *svm.50pred <- predict(svm50, mush_50Class$test)*
  - ○ Predictions on 50-50 Test Set with SVM 50-50 Model
- ➢ *svm.50predN <- svm.50pred*
- ➢ *svm.50predN <- apply(svm.50predN, 2, function(x) ifelse(x>12, 15, x))*
- ➢ *svm.50predN <- apply(svm.50predN, 2, function(x) ifelse(x<12, 6, x))*
  - ○ 'svm.50predN' contains rounded values of computed classification, used for accuracy testing.
- ➢ *compare50pred <- data.frame(svm.50predN, ytest)*

- Data frame with first column as predicted values and second column as true values.

➢ *total_predictions50 <- nrow(compare50pred)*
  - Number of predictions made by model.

➢ *correct_count50 <- 0*

➢ *for (i in 1:total_predictions50) {*

```
  if (compare50pred$svm.50predN[i] == compare50pred$class[i]) {
      correct_count50 <- correct_count50 + 1
  }
}
```

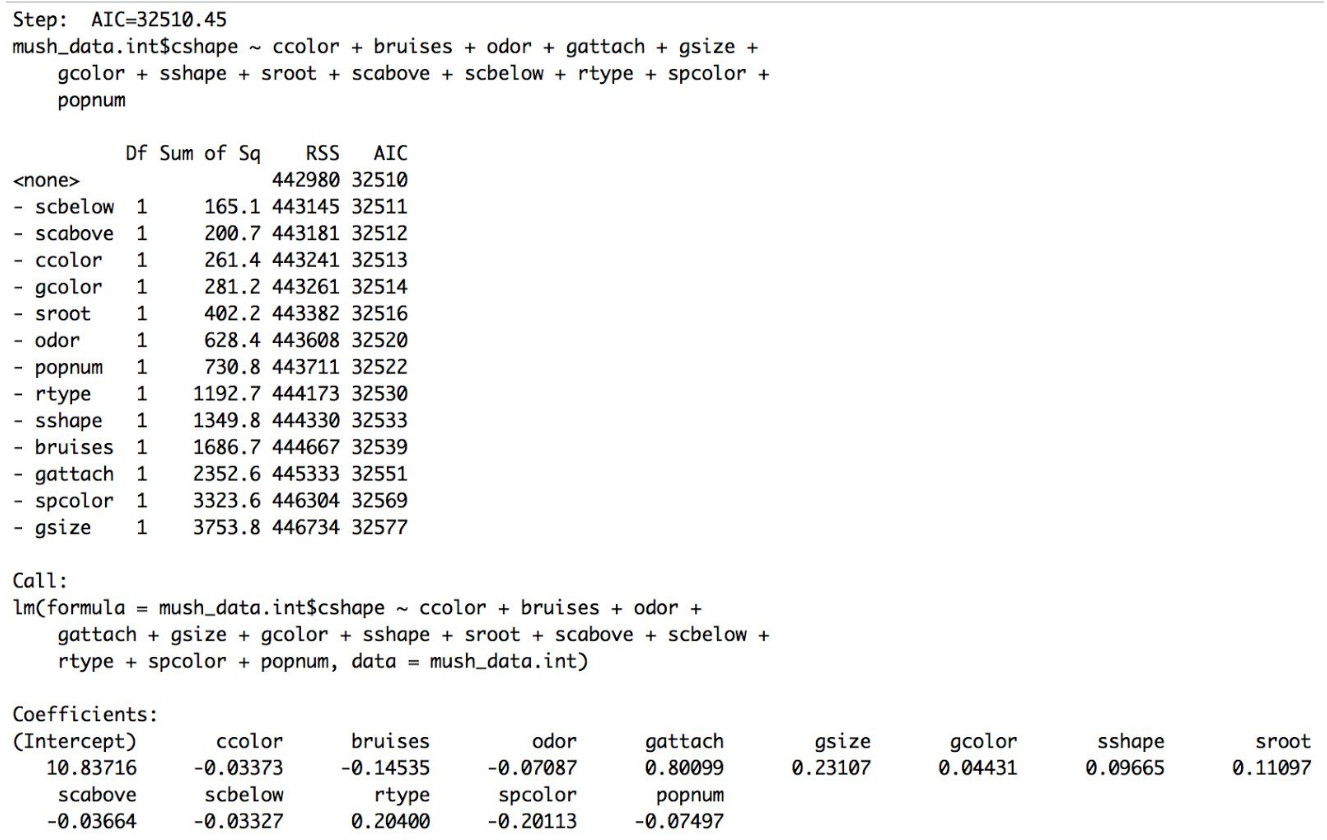
➢ *correct_count50*
  - Number of correct predictions,

➢ *accuracy50 <- (correct_count50 / total_predictions50) * 100*
  - Accuracy for SVM 50-50 Model.

➢ *plot(svm.50pred, data=x)*
  - Plot of SVM separated data points.

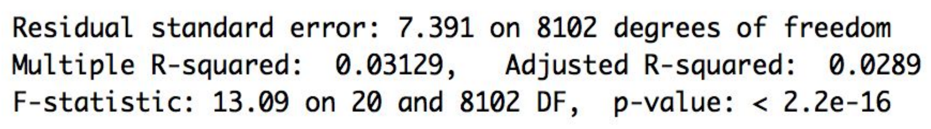
LM and GLM

LM results for Cap Shape (1)



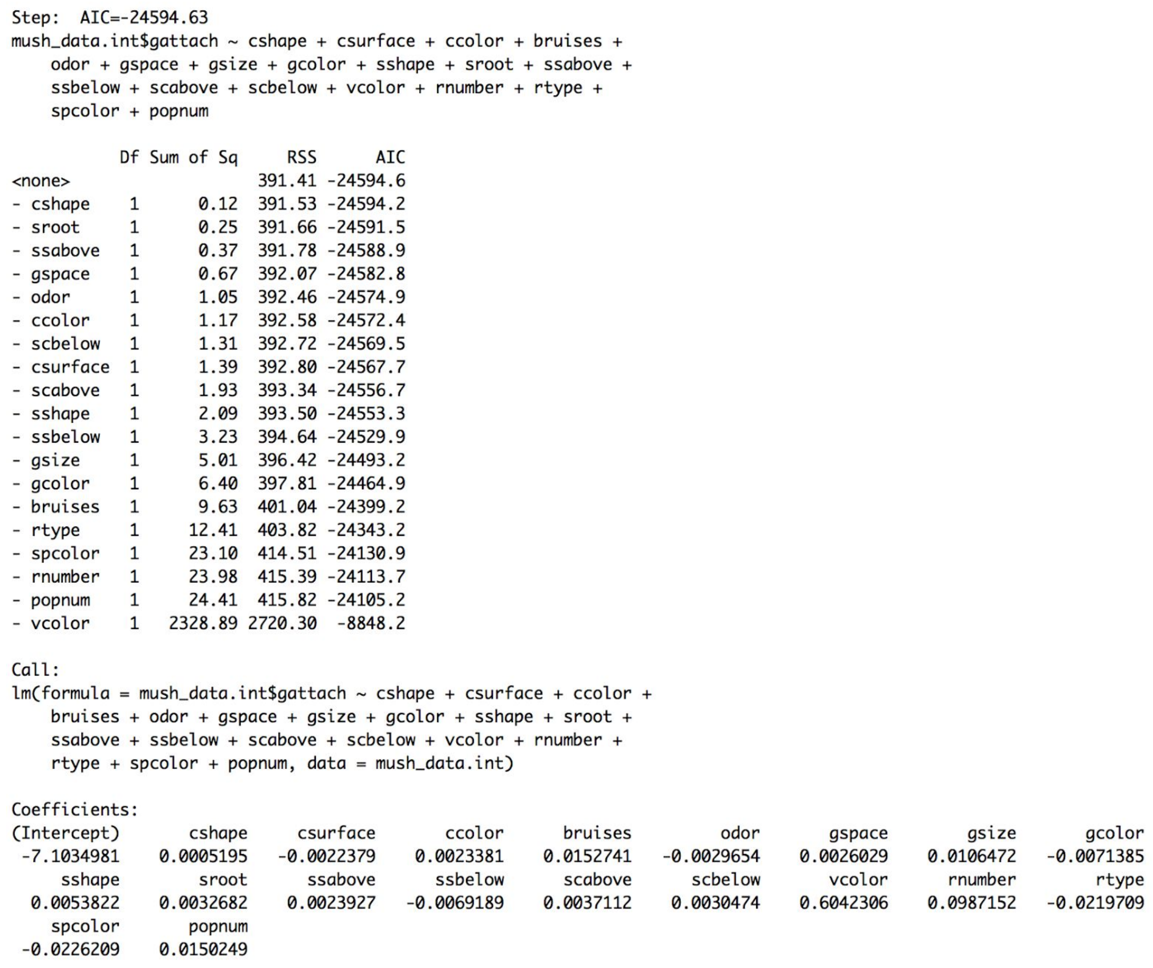
```
Step:  AIC=32510.45
mush_data.int$cshape ~ ccolor + bruises + odor + gattach + gsize +
    gcolor + sshape + sroot + scabove + scbelow + rtype + spcolor +
    popnum

          Df Sum of Sq    RSS   AIC
<none>                 442980 32510
- scbelow  1     165.1 443145 32511
- scabove  1     200.7 443181 32512
- ccolor   1     261.4 443241 32513
- gcolor   1     281.2 443261 32514
- sroot    1     402.2 443382 32516
- odor     1     628.4 443608 32520
- popnum   1     730.8 443711 32522
- rtype    1    1192.7 444173 32530
- sshape   1    1349.8 444330 32533
- bruises  1    1686.7 444667 32539
- gattach  1    2352.6 445333 32551
- spcolor  1    3323.6 446304 32569
- gsize    1    3753.8 446734 32577

Call:
lm(formula = mush_data.int$cshape ~ ccolor + bruises + odor +
    gattach + gsize + gcolor + sshape + sroot + scabove + scbelow +
    rtype + spcolor + popnum, data = mush_data.int)

Coefficients:
(Intercept)       ccolor      bruises         odor      gattach        gsize       gcolor       sshape        sroot
   10.83716     -0.03373     -0.14535     -0.07087      0.80099      0.23107      0.04431      0.09665      0.11097
    scabove      scbelow        rtype      spcolor       popnum
   -0.03664     -0.03327      0.20400     -0.20113     -0.07497
```

LM results for Cap Shape (2)



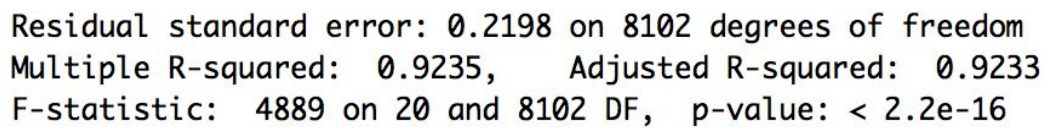
```
Residual standard error: 7.391 on 8102 degrees of freedom
Multiple R-squared:  0.03129,   Adjusted R-squared:  0.0289
F-statistic: 13.09 on 20 and 8102 DF,  p-value: < 2.2e-16
```

LM results for Gill Attachment (1)



```
Step:  AIC=-24594.63
mush_data.int$gattach ~ cshape + csurface + ccolor + bruises +
    odor + gspace + gsize + gcolor + sshape + sroot + ssabove +
    ssbelow + scabove + scbelow + vcolor + rnumber + rtype +
    spcolor + popnum

           Df Sum of Sq     RSS      AIC
<none>                   391.41 -24594.6
- cshape    1      0.12  391.53 -24594.2
- sroot     1      0.25  391.66 -24591.5
- ssabove   1      0.37  391.78 -24588.9
- gspace    1      0.67  392.07 -24582.8
- odor      1      1.05  392.46 -24574.9
- ccolor    1      1.17  392.58 -24572.4
- scbelow   1      1.31  392.72 -24569.5
- csurface  1      1.39  392.80 -24567.7
- scabove   1      1.93  393.34 -24556.7
- sshape    1      2.09  393.50 -24553.3
- ssbelow   1      3.23  394.64 -24529.9
- gsize     1      5.01  396.42 -24493.2
- gcolor    1      6.40  397.81 -24464.9
- bruises   1      9.63  401.04 -24399.2
- rtype     1     12.41  403.82 -24343.2
- spcolor   1     23.10  414.51 -24130.9
- rnumber   1     23.98  415.39 -24113.7
- popnum    1     24.41  415.82 -24105.2
- vcolor    1   2328.89 2720.30  -8848.2

Call:
lm(formula = mush_data.int$gattach ~ cshape + csurface + ccolor +
    bruises + odor + gspace + gsize + gcolor + sshape + sroot +
    ssabove + ssbelow + scabove + scbelow + vcolor + rnumber +
    rtype + spcolor + popnum, data = mush_data.int)

Coefficients:
(Intercept)       cshape     csurface       ccolor      bruises         odor       gspace        gsize       gcolor
 -7.1034981    0.0005195   -0.0022379    0.0023381    0.0152741   -0.0029654    0.0026029    0.0106472   -0.0071385
     sshape        sroot      ssabove      ssbelow      scabove      scbelow       vcolor      rnumber        rtype
  0.0053822    0.0032682    0.0023927   -0.0069189    0.0037112    0.0030474    0.6042306    0.0987152   -0.0219709
    spcolor       popnum
 -0.0226209    0.0150249
```

LM results for Gill Attachment (2)



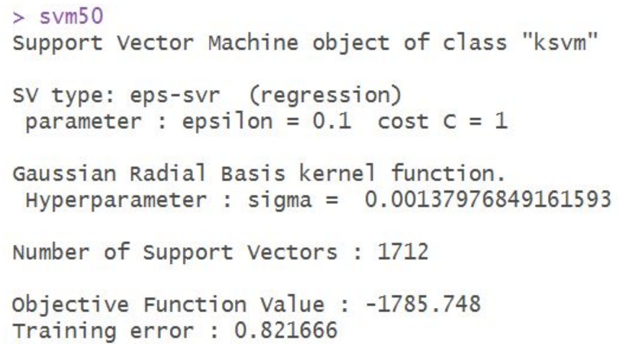
```
Residual standard error: 0.2198 on 8102 degrees of freedom
Multiple R-squared:  0.9235,	Adjusted R-squared:  0.9233
F-statistic:  4889 on 20 and 8102 DF,  p-value: < 2.2e-16
```

Discussion

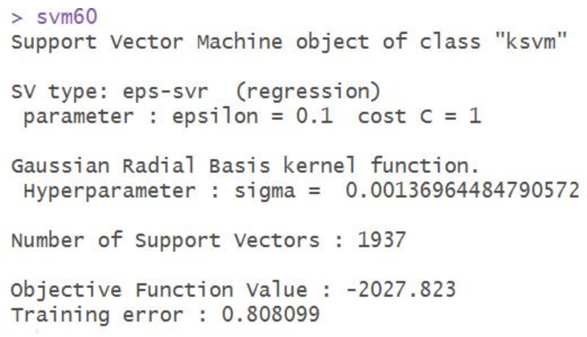
The way that the linear model works is that the model goes through 10 iterations, and gives the best fitting iteration last. The two models above show the difference between the lowest R-squared goodness of fit (LM results for cap size (1) and (2)) and the highest R-squared goodness of fit (LM results for attachment (1) and (2)). The results shown for cap size were only included to show the difference between that and for attachment. The residual standard error (RSE) tells about the quality of a linear regression fit. This tells us more about how much the response will deviate from the regression line, which makes sense as to why there is a lower residual standard error for the LM results for attachment. For the F-Statistic, we are looking for a value that is further from 1. A large F-statistic could also occur because of the number of predictors. Using the coefficients, one could even predict how other attributes of the mushroom could interact with one another.

Support Vector Machine

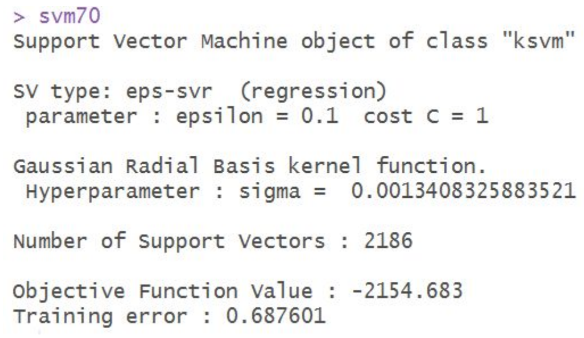
SVM Model 50-50 Train/Test


```
> svm50
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.00137976849161593

Number of Support Vectors : 1712

Objective Function Value : -1785.748
Training error : 0.821666
```

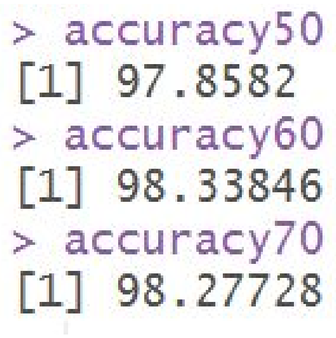
SVM Model 60-40 Train/Test


```
> svm60
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.00136964484790572

Number of Support Vectors : 1937

Objective Function Value : -2027.823
Training error : 0.808099
```

SVM Model 70-30 Train/Test


```
> svm70
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0013408325883521

Number of Support Vectors : 2186

Objective Function Value : -2154.683
Training error : 0.687601
```

Accuracy of SVMs on Test Set


```
> accuracy50
[1] 97.8582
> accuracy60
[1] 98.33846
> accuracy70
[1] 98.27728
```

Discussion

SVM works by trying to find the greatest margin between points considered to be of two different classifications. The line created is used as a decision boundary for classification. SVM has great success as a classifier, with all three trained models having a recall higher than 97.5%.

Tables

Table (3)

	Cluster Size		
Data and Clustering Method	**k=3**	**k=5**	**k=7**
mush_50.pam	2490, 897, 675	675, 764, 896, 1063, 664	648, 358, 880, 534, 564, 647, 431
mush_60.pam	1077, 2996, 801	1068, 1284, 791, 954, 777	1054, 752, 765, 527, 765, 542, 469
mush_70.pam	3543, 1234, 909	1454, 1224, 897, 916, 1195	730, 1208, 883, 880, 906, 489, 590
mush_50.kmeans	1630, 1556, 876	875, 653, 682, 809, 1043	858, 335, 418, 937, 354, 517, 643
mush_60.kmeans	2120, 767, 1987	1268, 1080, 763, 943, 820	577, 627, 750, 481, 802, 1172, 465
mush_70.kmeans	930, 1254, 3502	1518, 1066, 1256, 917, 929	1037, 259, 744, 924, 662, 544, 1516
mush_50.hclust	2532, 878, 652	585, 457, 878, 1490, 652	585, 142, 878, 441, 315, 652, 1049
mush_60.hclust	1053, 3045, 776	1053, 2036, 776, 776, 233	1053, 950, 776, 1086, 686, 233, 90
mush_70.hclust	3435, 1208, 1043	1759, 1208, 1043, 897, 779	1191, 1208, 693, 897, 350, 568, 779

Table (4)

	LM Results for Gill Attachment	LM Results for Cap Shape
Residual Standard Error	.2198 on 8102 degrees of freedom	7.391 on 8102 degrees of freedom
Multiple R-Squared	0.9235	0.03129
Adjusted R-Squared	0.9233	0.0289
F-Statistic	4889 on 20 and 8102 DF	13.09 on 20 and 8102 DF

Clustering Results

K-Means begins by initializing *k* centroid points, located on random values. Each point in the data set is then assigned to the cluster associated with the centroid it is closest to. The centroid for each cluster is then recalculated by finding the median value between all data points within that cluster. The centroids are then repositioned at the newly calculated points. Data points are then reassigned to their nearest centroid's cluster and the process iterates until the centroids stop moving and the clusters have been found. K-Means is a reliable clustering method that requires no building of prediction models; the process is simply algorithmic. However, the issue with this is that K-Means may require many reiterations of the assignment/centroid calculating steps and can be computationally expensive. K-Means was able to successfully cluster the mushrooms into each value of k clusters for each train/test set.

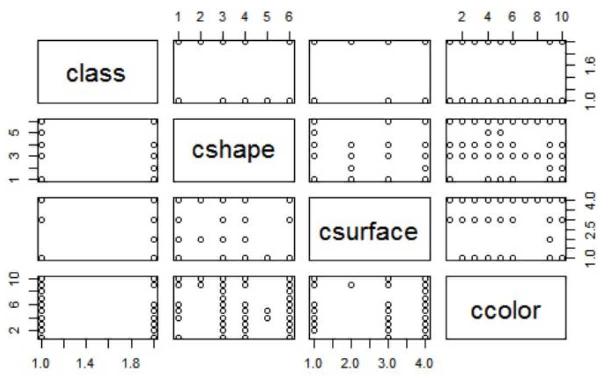
PAM clustering is similar to K-Means clustering, using the same algorithm in addition to the medioshift algorithm, thus explaining the similarity between the results of K-Means and PAM clustering. However, we can see that PAM is more robust to noise and outliers, since it minimizes the sum of pairwise similarities instead of minimizing the sum of squared Euclidean distances. PAM is also known as the K-medoids algorithm, the term medoid referring to the the central object within a cluster. The average dissimilarity between the medoid and all other members of the cluster is minimized. The reason behind PAM's robustness is its use of medoids as its central points instead of the means of the objects in the cluster (centroids).

HClust clustering is a hierarchical clustering method. It defines the cluster distance between two clusters to be the maximum distance between their individual components. Through the clustering process, the two nearest clusters merge to form a new cluster. This whole process is repeated until the whole data set is processed. The cluster sizes for HClust were relatively closer in size than the other clustering methods; however, with all of the clustering methods we have used, there seemed to be one dominant cluster that is strong in correlation and remained throughout the increasing number of clusters (k value).

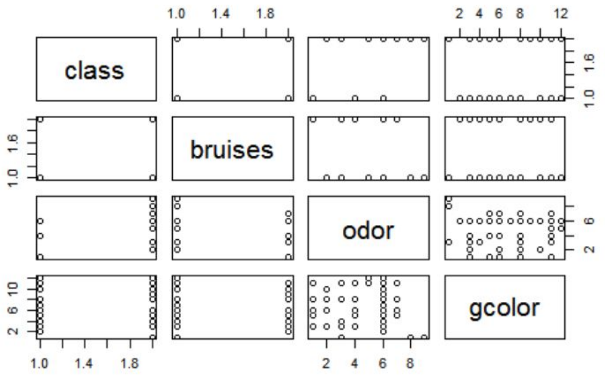
PAM clustering is the best clustering model that we applied. PAM, because of its similarity to K-Means, makes it a widely understood method of classification with easily interpreted results. PAM outperforms K-Means when data is cluttered with noise and outliers. Five clusters is an optimal number as it helps classify data points into separate clusters without reaching the point of confusion. Splitting the clusters into more than five also makes it harder to derive meaning and insights, which is one of the main goals of clustering.

Plots

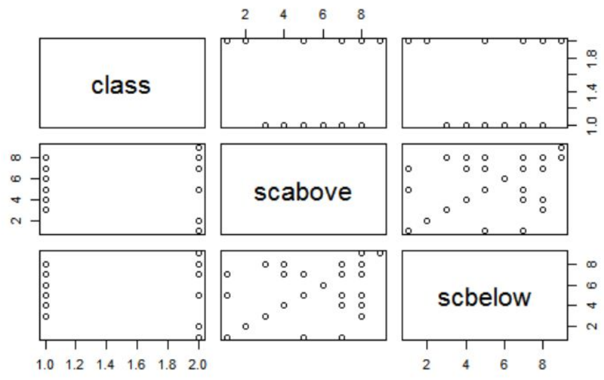
Initial Plotting of Attributes



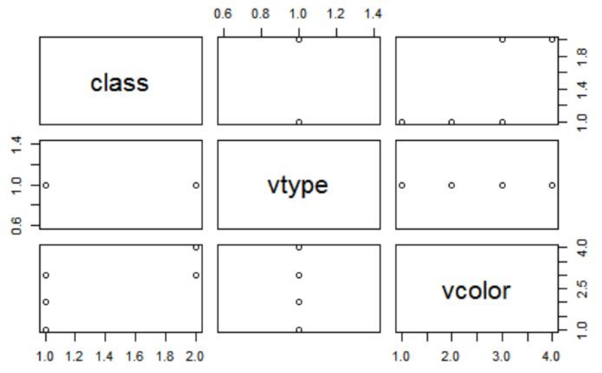
Class ~ Cap Shape, Cap Surface, Cap Color



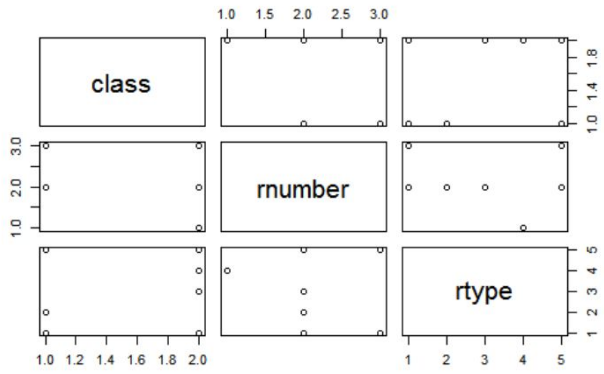
Class ~ Bruises, Odor, Gill Color



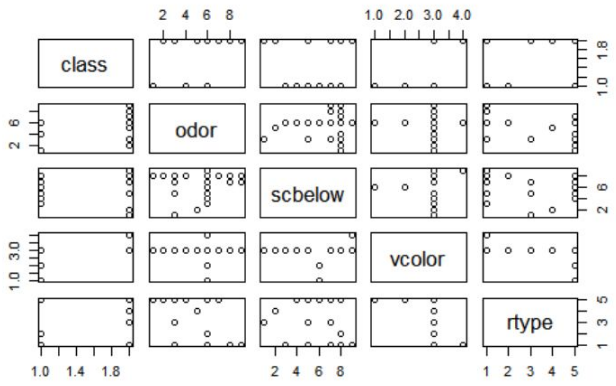
Class ~ Stalk Color Above Ring, Below Ring



Class ~ Veil Type, Veil Color

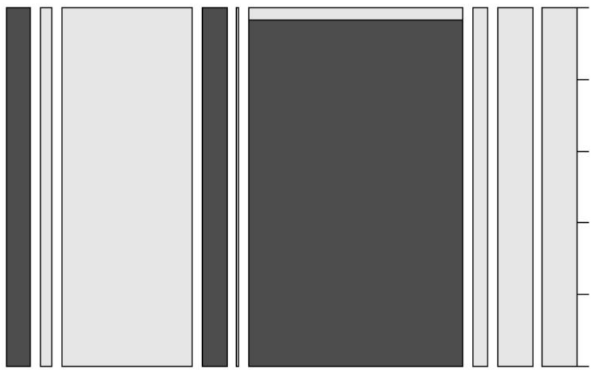


Class ~ Ring Number, Ring Type

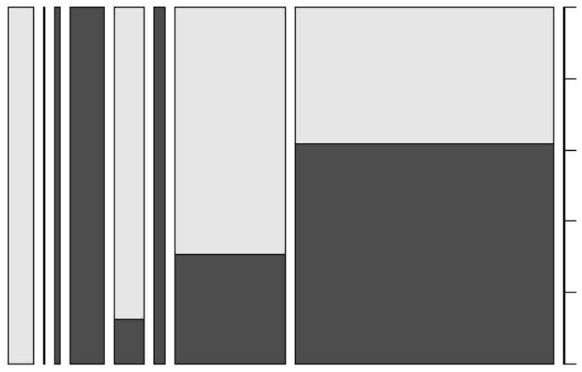


Class ~ Odor, SCBelow, VColor, RType

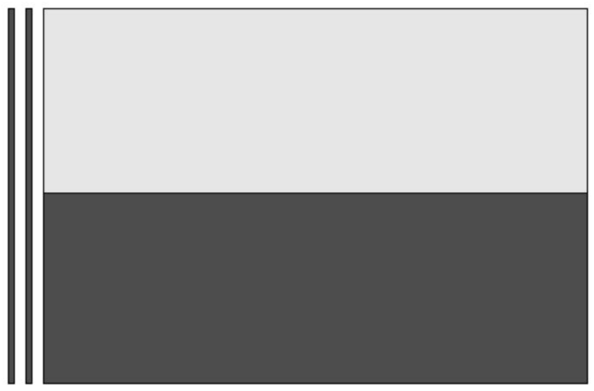
Distribution of Values Plots

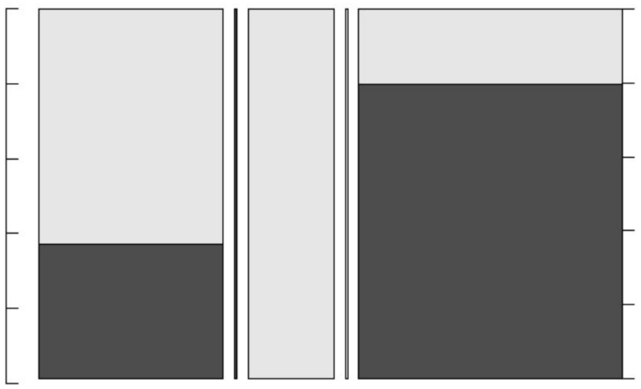


Class vs Odor (a, c, f, l, m, n, p, s, y)

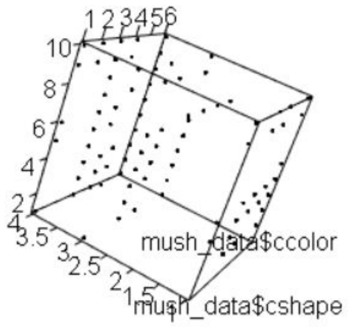


Class vs SCBelow (b, c, e, g, n, o, p, w, y)

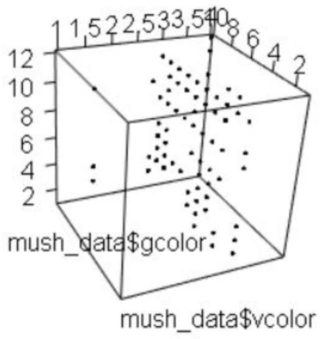

Class vs VColor (b, o, w, y)


Class vs RType (c, e, f, l, n, p, s, z)

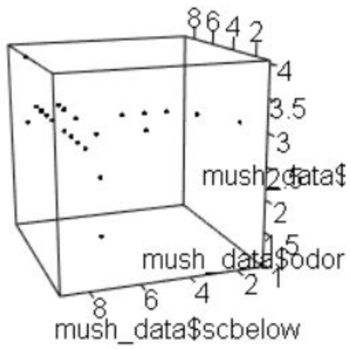
3D Attribute Plots



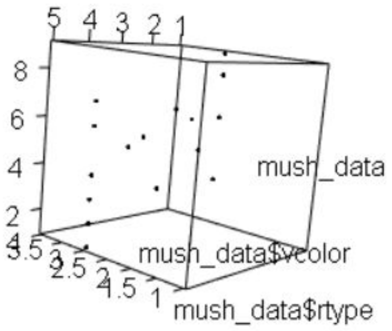
CShape vs CSurface vs CColor



VColor vs CColor vs GColor

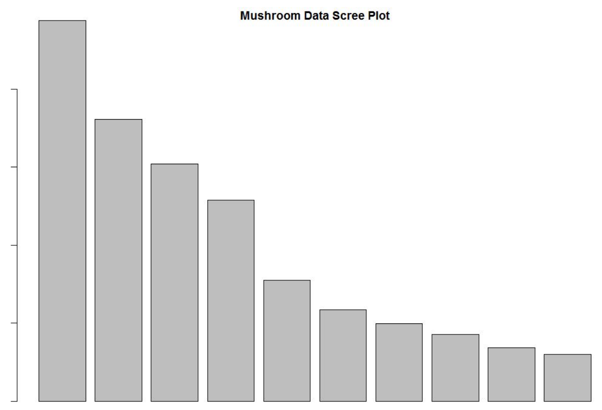


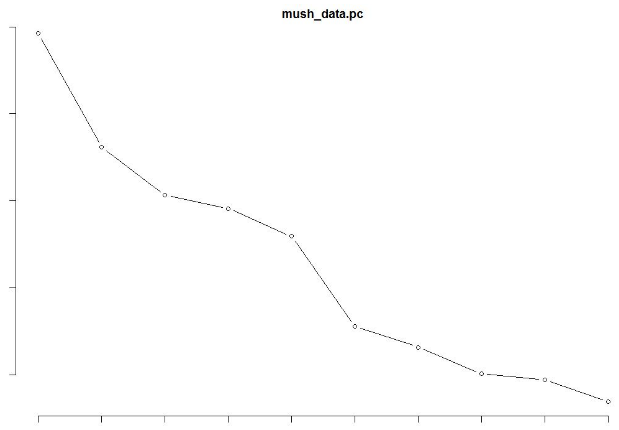
Odor vs SCBelow vs VColor



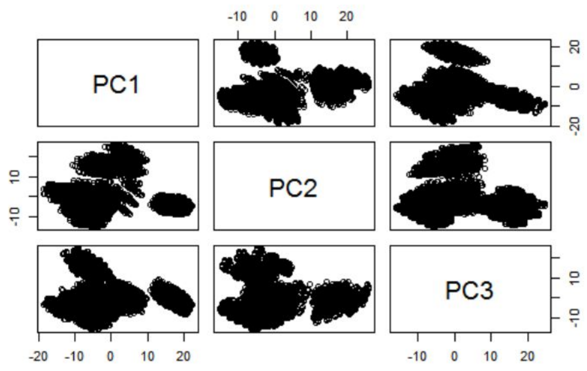
Odor vs VColor vs RType

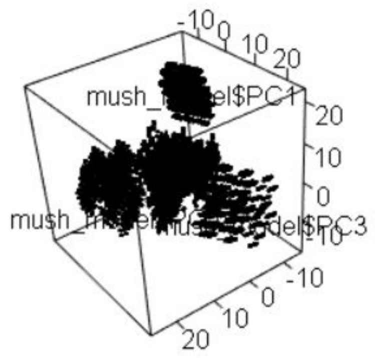
Scree Plots





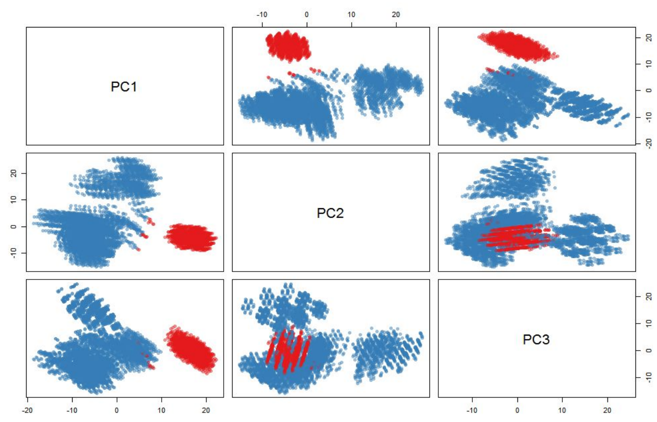
Before Clustering

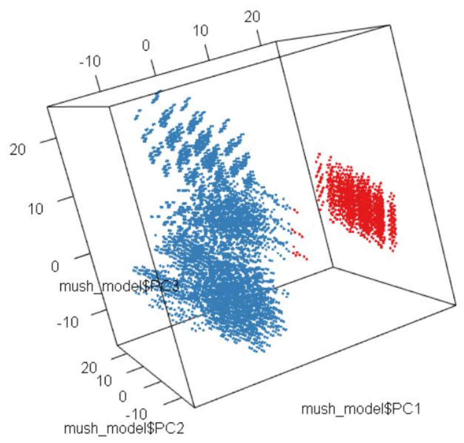




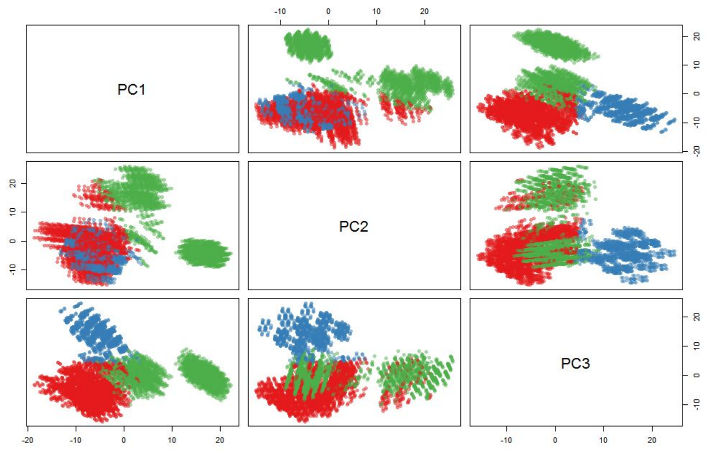
K-Means

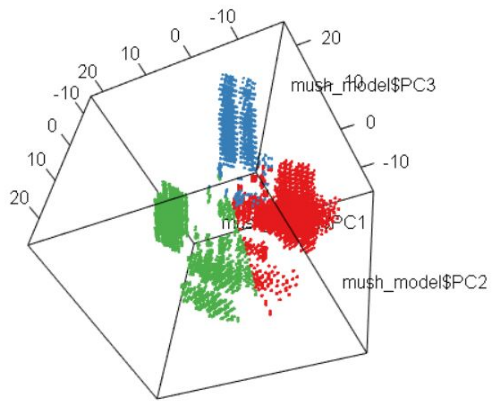
(k=2)



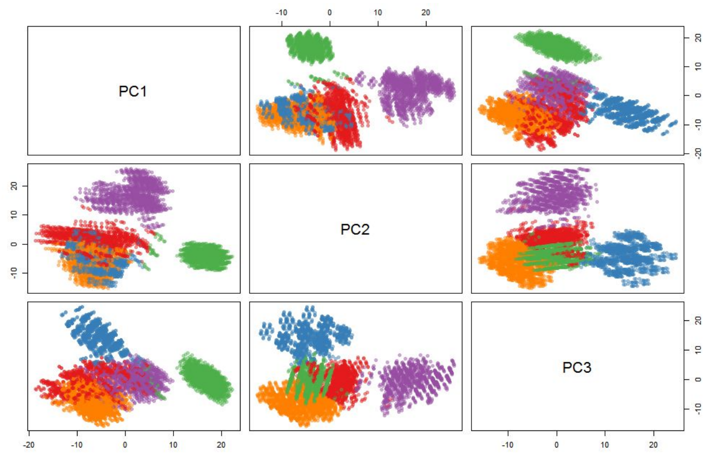


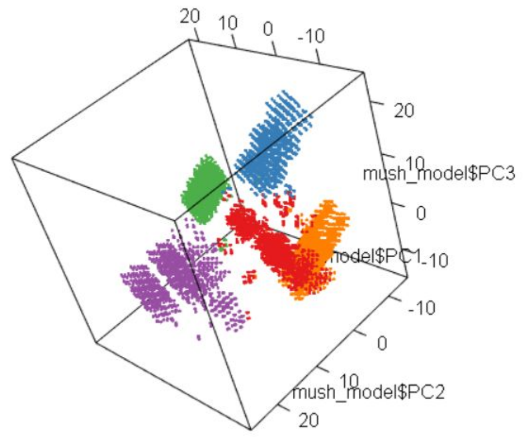
(k=3)


PC1
PC2
PC3

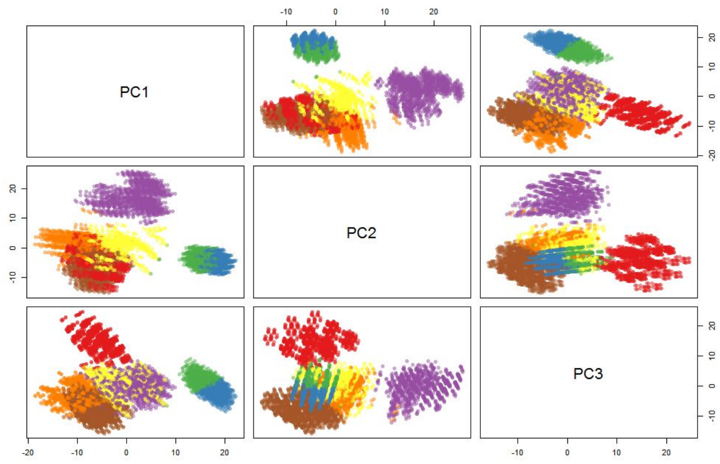

mush_model$PC3
mush_model$PC1
mush_model$PC2

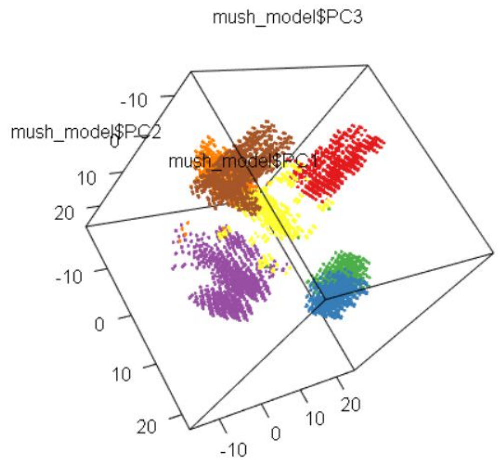
(k=5)


PC1
PC2
PC3


mush_model$PC3
mush_model$PC1
mush_model$PC2

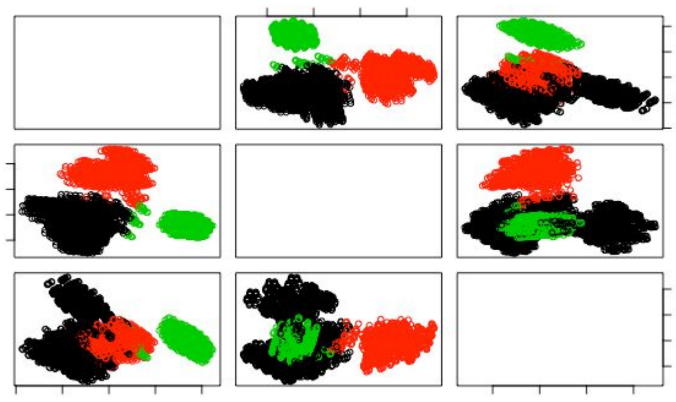
(k=7)

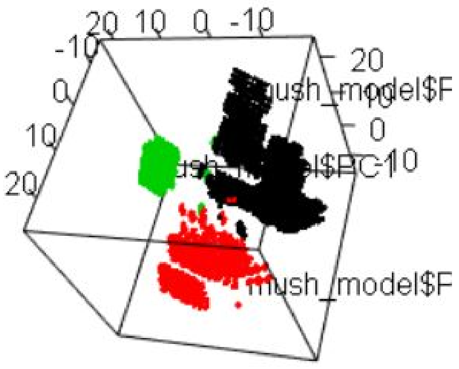

PC1
PC2
PC3

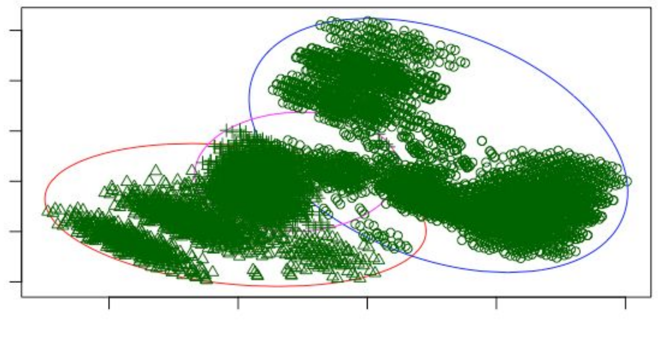

mush_model$PC3
mush_model$PC2
mush_model$PC1

PAM Clustering

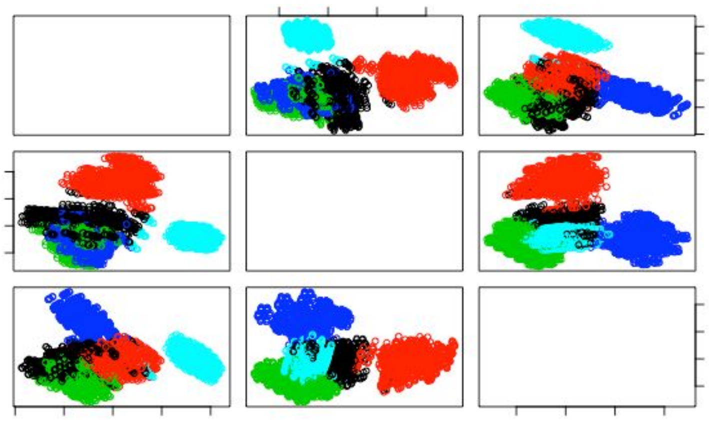
(k=3)

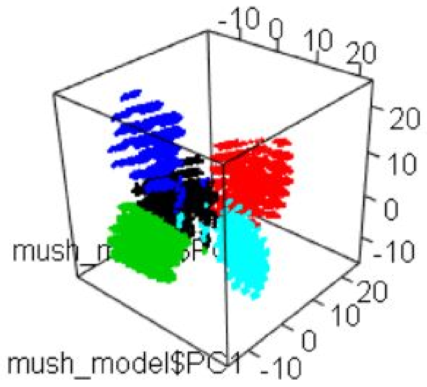


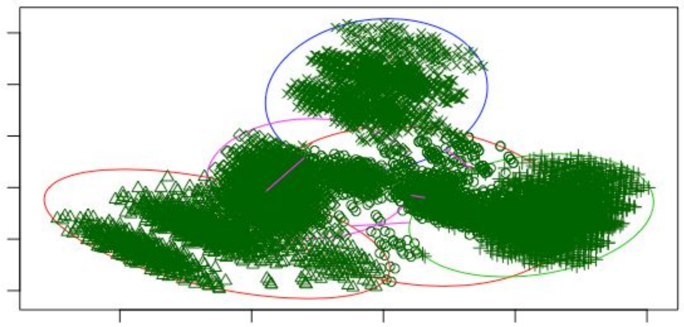




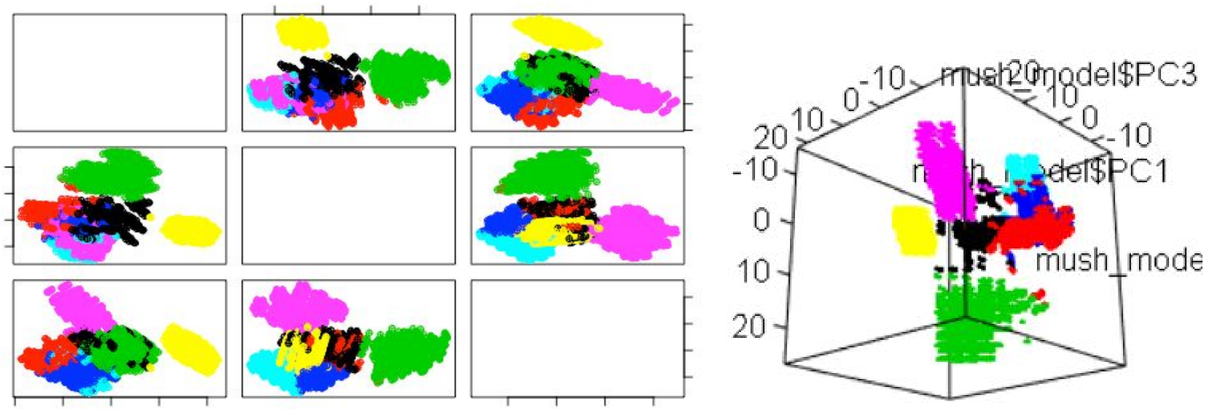
(k=5)

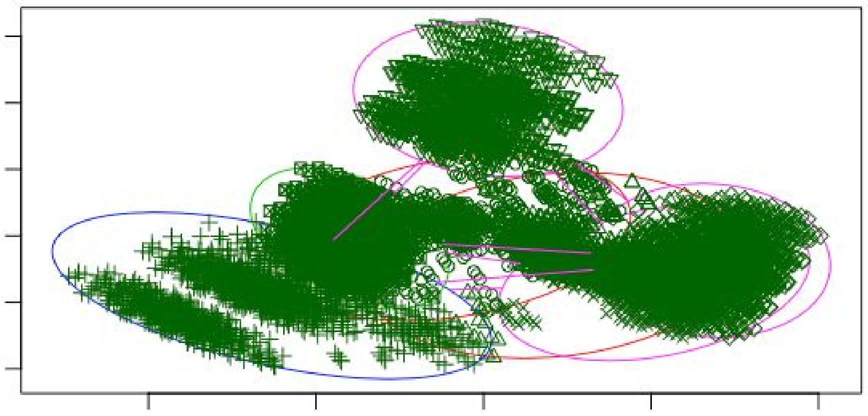






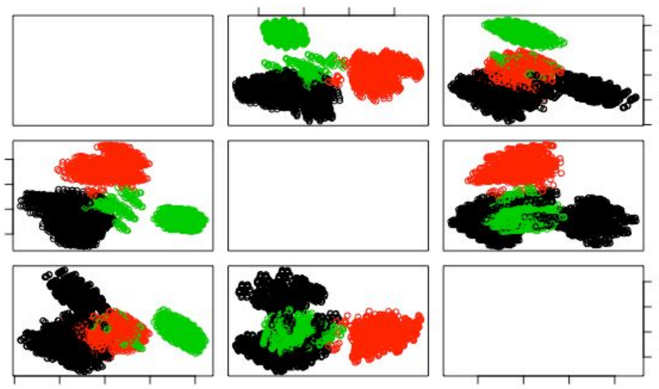
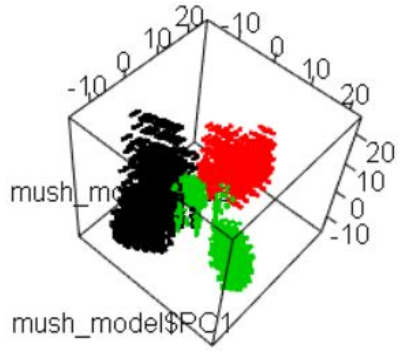
(k=7)



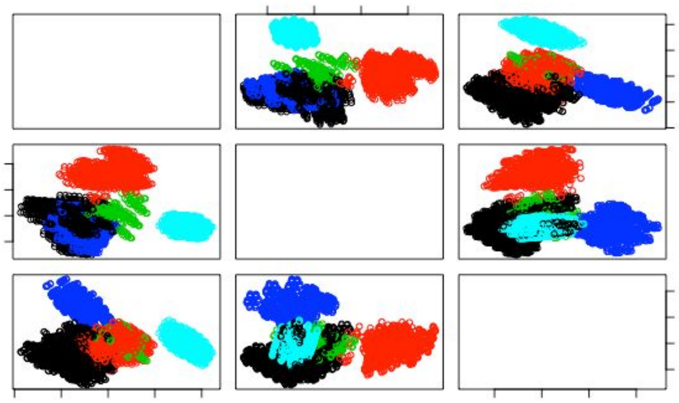
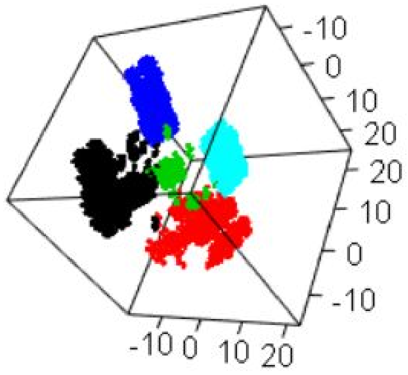


HClust Clustering

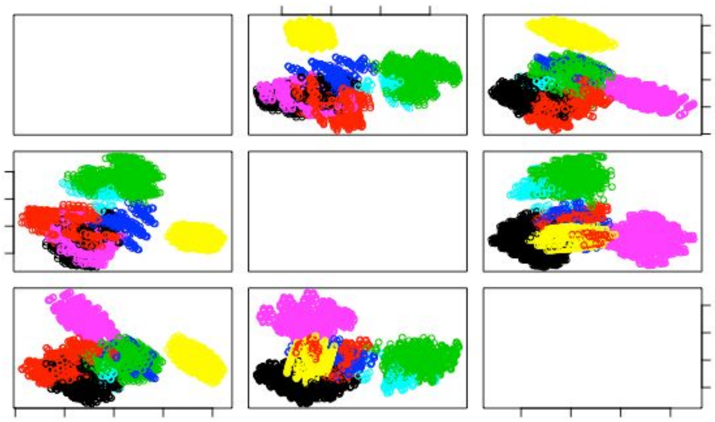
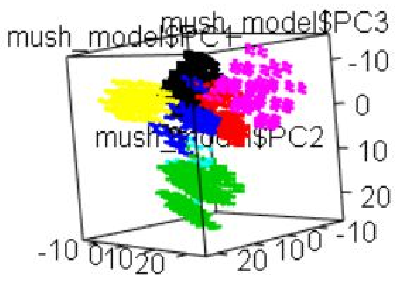
(k=3)

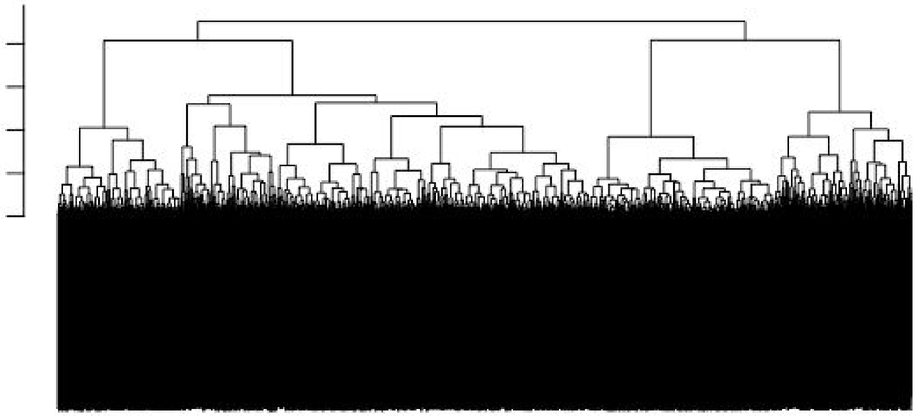
(k=5)

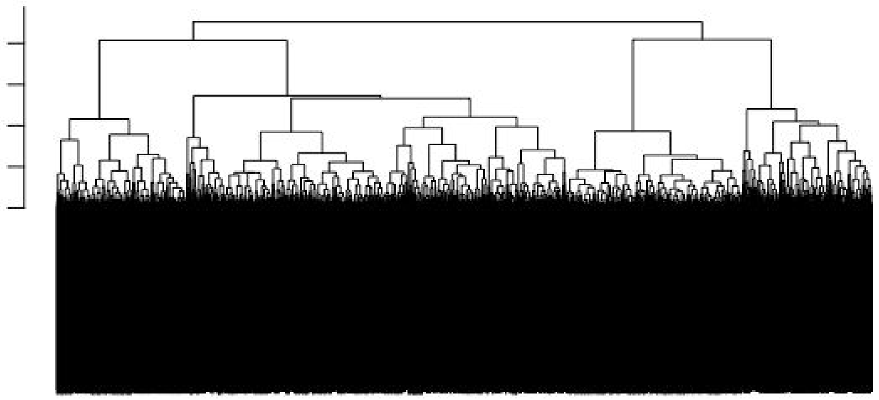
(k=7)

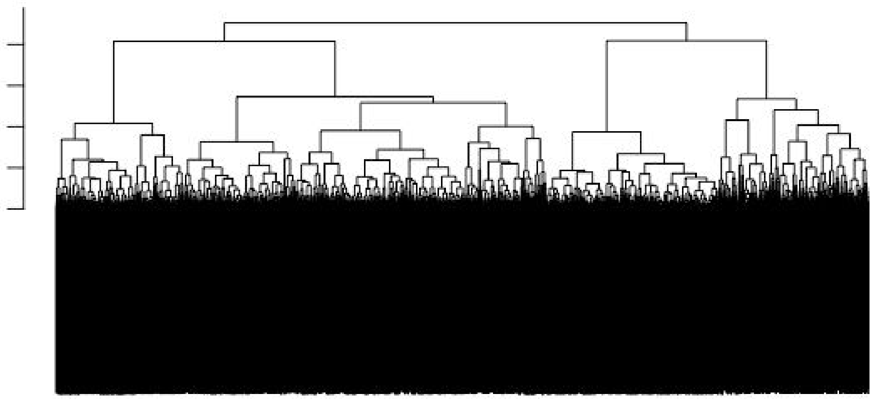
HClust Clustering 50/50



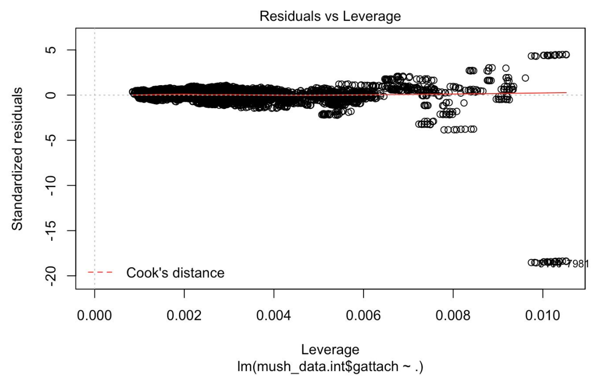
HClust Clustering 60/40



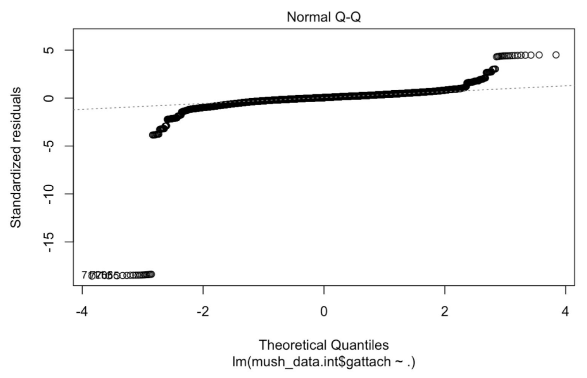
HClust Clusterings 70/30



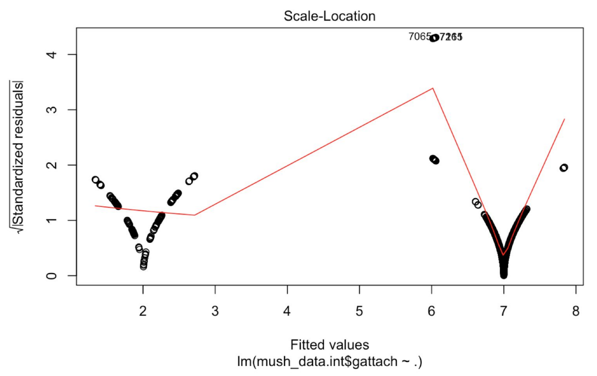
LM and GLM Methods



Residuals vs Leverages

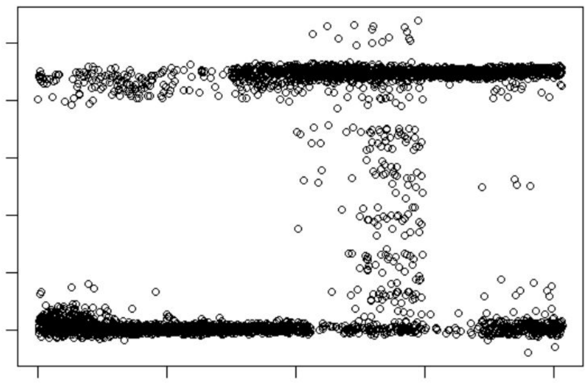


Normal Q-Q

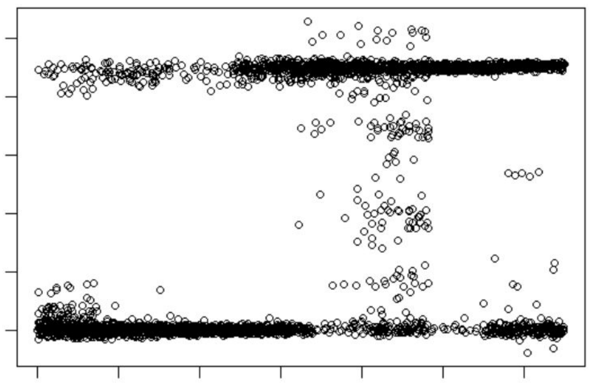


Scale-Location

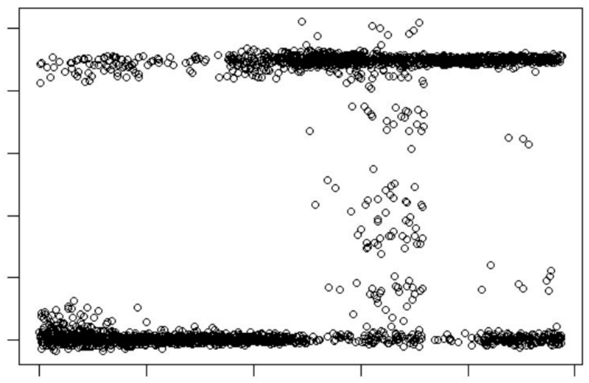
Support Vector Machines



50-50 Train/Test Set



60-40 Train/Test Set



70-30 Train/Test Set

Discussion

Initial Plotting of Attributes

The initial pair plotting of attributes against class is useful because it allowed us to find values of attributes that may only pertain to either edible or poisonous mushrooms. The plots between odor, stalk-color below ring, veil color, and ring type against class show that eight of the nine values for odor (8/9), six of the nine values for stalk-color below ring (6/9), three of the four values for veil color (3/4), and three of the five values for ring type (3/5) indicate the mushroom as being either edible or being poisonous.

Distribution of Values Plots

The plotting of the distribution of values for each attribute was useful in finding the values of an attribute that were completely binary and the percentages of edible and poisonous mushrooms for values that were not. The odor plot shows that even though one of the nine values was not completely binary, the distribution leans almost entirely towards being edible, with less than 10% of examples being poisonous. The stalk-color below ring, veil color, and ring type plots show that some values have much more split classifications.

3D Attribute Plots

The 3D attribute plots were useful in better visualizing relationships between certain variables and their ranges in value combinations.

Scree Plots

The scree plots were useful for performing principal component analysis because we can analyze the plot to find the value at the knee in the line, which represents the optimal number of components to use in PCA. The number of components we chose was three.

K-Means

The 3D plots of K-Means were useful in visualizing the resulting clusters. K-Means was able to form distinct clusters that visually agree with prior observations made on the data set.

PAM Clustering

The PAM clustering is related to the K-Means clustering and the medioshift algorithm. That is why the results of the PAM clustering were similar to the K-means; however, we can see the differences by examining the size of the clusters. We can see how using PAM minimizes the sum of the dissimilarities of the observations to their closest representative object, resulting in different cluster sizes and more robust results than the K-Means classifier. As can be see in the Clustplot, from k=3 to k=7, we can see how the cluster sizes are decreasing as k increases. The pink lines show the links between the different clusters, from their central mediod. The oval cluster that is blue in the bottom left corner of each of the Clustplot plots is the most dominant, as it does not change size while k increases. This means that the points within the cluster are strongly correlated. The ellipses on the Clustplot plots show how separable the data is or is not, and in our case, we can see that the data is not very separable since there is much overlap (especially when k=5 and k=7).

HClust Clustering

The HClust function provides a way to perform a hierarchical cluster analysis by first looking at a set of dissimilarities for n objects. This function starts by assigning objects to their own cluster, and then at each stage joins two of the most similar clusters together until there is only one overarching cluster. This is seen above in the plots section. It starts off looking like a huge conglomerate of clusters at the bottom and then keeps joining different clusters until it reaches the top and there is a single cluster.

LM and GLM Methods

The LM method gives us a more accurate sense of how the points interact with one another. In order to do this, we try the linear model on several different attributes. We tried with the attribute of cshape and saw a low adjusted R-squared value of .0289 which means the model explains a minimal amount of the variability. This could be because of the large amount of variables in the dataset that caused the R-square to be so low. We also did the lm method on the attribute of gattach (Gill attachment) and found a .9233 adjusted r-square which explains much more of the variability. The Residuals vs Leverage plot helps us see that for attachment, most of the values stay around a certain line, with some outliers. These could be influential points. From the Normal QQ plot, we see that the dependent variable is normally distributed because the mean is zero and the points fall on a 45 degree line. The Scale- Location plot shows if the points have a constant variance. The GLM model is a generalization of ordinary linear regression. This allows for points that have more than just a normal distribution.

Support Vector Machines

The plots from the support vector machines display the data points on a 2D plane that can then be separated by finding the largest margin between the two classifications. As we increase the size of the training set from 50% to 70% we see a decreasing number of data points located near the center between the two classifications. This is because as the training set increases, the model becomes more finely-tuned and is able to remove a greater amount of the ambiguity surrounding the points found in the middle, resulting in better classifications and a wider separating margin.

Analysis of Application to Data Science

From working on this project we were exposed to techniques used in the field of data science. Upon looking at the data from first glance, it did not really make sense. We had to go through the KDD- Knowledge Discovery and Data Mining process to be able to ensure we were moving through the project as a data scientist would. We began by performing pre-processing on the data to transfer character values to numerical values, remove any null values, and transform data frames. We discussed which statistical analyses we were to perform, figured out the number of components we would reduce our dataset to with Principal Component Analysis, and which clustering methods we would use for classification. We shared techniques we had learned in previous classes and by finding a common knowledge base, we were better able to work through this project, especially when paired with the lecture notes.

This whole process of extracting data, processing it, building models, and generating useful output has helped us better understand more about what data scientists actually do. We now have more exposure to the processes that take data as input and extract meaningful information. This project has helped us tremendously because now we have more exposure to big data techniques that could be used in future workplaces. Clustering in particular can be used across all types of disciplines such as in marketing, for instance being able to go through a large database of customers to find patterns in customers' behavior and preferences. This helps marketing agencies target certain clusters of people. Knowing these techniques will also help us conduct independent research on interesting datasets that we find on our own.