Project 2 - Mushroom Dataset

Nisha Iyer, Rachel Jordan, Samuel Dooley

March 20, 2016

1 Intro

The purpose of this write up is to discuss the work levied against discovering the set of 'unsure' mushrooms. The dataset `agaricus-lepiota` was given to us to analyze. This dataset had 22 variables and 8124 observations such that each observation was labeled edible or poisonous. Within the class of poisonous mushrooms, there is a smaller subset of mushrooms whose edibility are unsure, but were labeled poisonous. What follows is a discussion of our team's effort to discover this susbclass.

2 Dataset Familiarization

As stated above, the dataset includes many observations of 22-dimensional data. There is a 23rd dimension that determines the mushroom's class. The variables are listed here:

- `capshape`
- `capsurface`
- `capcolor`
- `bruises`
- `odor`
- `gillattachment`
- `gillspacing`
- `gillsize`
- `gillcolor`
- `stalkshape`
- `stalkroot`
- `stalksurfaceabovering`
- `stalksurfacebelowring`
- `stalkcolorabovering`
- `stalkcolorbelowring`
- `veiltype`
- `veilcolor`
- `ringnumber`
- `ringtype`
- `sporeprintcolor`
- `population`
- `habitat`

So let us first import the data and give the data variables names:

```
mr <- read.table('agaricus-lepiota.data',sep=",",header=FALSE)
names(mr) <- c("class", "capshape", "capsurface", "capcolor","bruises",
               "odor","gillattachment","gillspacing","gillsize","gillcolor",
               "stalkshape","stalkroot","stalksurfaceabovering",
               "stalksurfacebelowring","stalkcolorabovering",
               "stalkcolorbelowring","veiltype", "veilcolor","ringnumber",
               "ringtype","sporeprintcolor","population","habitat")
```

The data in each of these variables above is categorical with the most category each variable can take being twelve in the variable `gillcolor`. So let us convert these data to numerical values, retaining NA values, and scaling to values between 0 and 1:

```
numeric <- mr
numeric[numeric == '?'] <- NA ##Protect the null values
numeric <- as.data.frame(sapply(numeric, as.numeric))

numeric <- numeric-1
numeric <- scale(numeric)
```

We now try to get an understanding of the data themselves. First, we can do that through statistics on the dataset:

```
summary(numeric) #entire dataset

edible <- numeric[which(numeric[,1] == 0),]
poisonous <- numeric[which(numeric[,1] == 1),]
summary(edible) #just edible ones
summary(poisonous) # just poisonous ones
```

which upon examining the values reveals that the following variables have large enough difference between the poisonous and the edible samples to examine further: `capsurface, bruises, odor, gillsize gillcolor, stalkroot, stalksurfacebelowring, stalkcolorabovering, stalkcolorbelowring, ringtype, sporeprintcolor, population, habitat`.

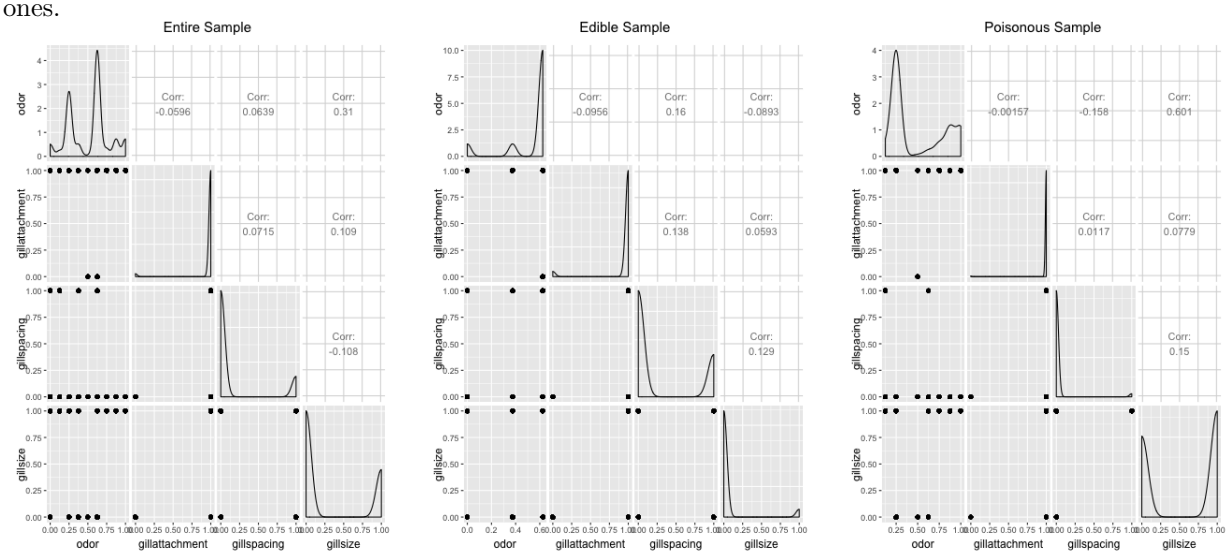
We can also analyze the variance of these variables and ignore those variables that have zero, or near zero variance.

```
nzv <- nearZeroVar(numeric, saveMetrics = TRUE)
## only keep those columns that significant variance
mr_nzv <- numeric[,c(rownames(nzv[nzv$nzv == FALSE,]))]
```

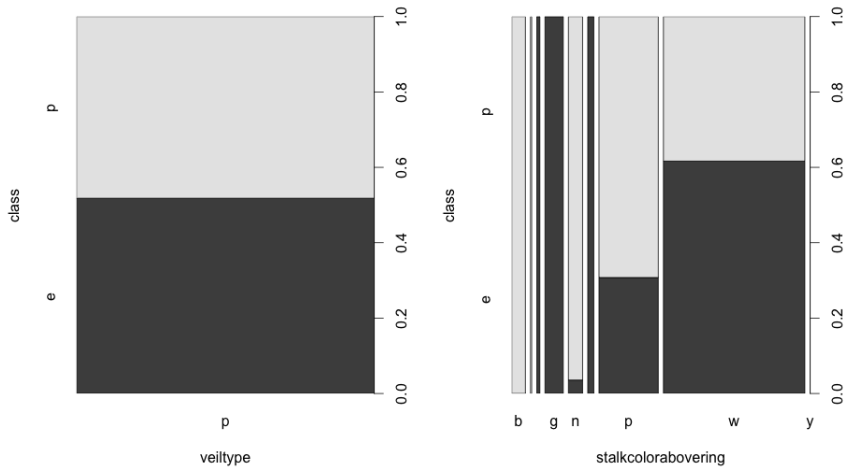
We also want to visualize these data, and so we use some plots to do just that. (In the attached `R` file, we have more plots, but here are just a couple.)

2.1 Plots

Below is an example of univariate plots and correlation of four variables, `odor, gillattachment, gillspacing, and gillsize`. We have plots for each sample, the entire dataset, just the edible ones, and just the poisonous ones.

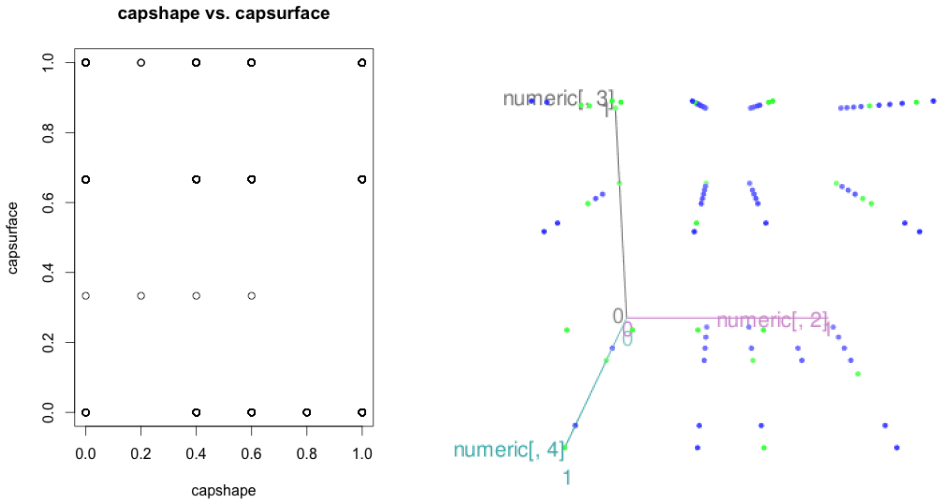


We can also make plots that reveal something about proportions of each category in each class, edible or poisonous. The below show plots for a given variable that display each category that variable can take. The widths of the bars are proportional to the amount of that category in the sample. The bar is then filled in with black representing edible and grey representing poisonous. This fill of each bar is proportional to the break out of that that variable when separated by poisonous and edible.



The left plot reveals that the variable, `veiltype` only has one category and that it is split between the two classes edible and poisonous in relationship to the proportion of each class in the sample. The right plot show that some variable can have many different categories with various break-outs of those variables. This plot also reveals that even some categories are entirely filled by one class. For instance, the category 'g' is only take by the edible mushrooms.

We can even take the regular scatterplot. The left scatterplot is a two-dimensional view of the first two, non-class variables, `capshape` and `capsurface`. The right is a three-dimensional view of the first three, non-class variables, `capshape`, `capsurface`, and `capcolor`. The green corresponds to those that are edible and the green are poisonous.



We can obviously tell that these types of plots are not very useful. Since these data are categorical, the dots 'lie' on top of each other when graphed. Thus, we will stay away from basic scatter plots for categorical data.

3 Dimension Reduction with PCA

We now want to reduce the number of variables that we are analyzing. As stated above, there are 22 variables, not including the one that stipulates whether the mushroom is edible or poisonous. While in the grand picture, 22 is not that many when compared to biological statisticians who commonly work with tens of thousands variables, we still look to reduce the dimensions needed for analysis.

The first and easiest step is to remove variables that have zero or near zero-variance. As discussed above, we have already done this, and the variable that contains the data with only the 'large' variance variables is stored in `mr_nzv`

The next best way to reduce the number of variables is to perform a *Principle Component Analysis*. This type of analysis, in effect, will change the basis of the data frame so that the basis vectors of the space are no longer $\{\vec{e_1}, \ldots, \vec{e_n}\}$ (where $\vec{e_i}$ is the standard spanning vector of the i-th dimension in Euclidian space), but are the vectors of the eigenvalue decomposition of the data matrix.

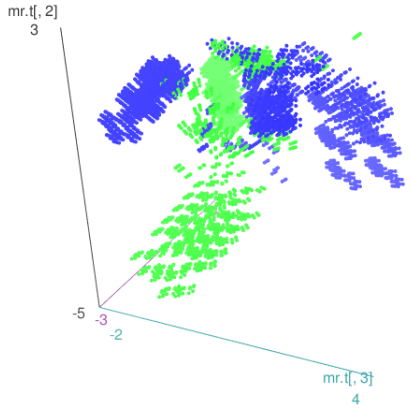
This ensures that the data are linearly transformed in such a way that the spanning vectors of the space are related to the variance that they explain in the data. With PCA, we will get 22 eigenvectors, but we should only consider those eigenvectors whose eigenvalues are greater than 1. So we do that here:

```
pr <- PCA(mr_nzv, ncp = dim(mr_nzv)[2])

## take the first components that have
## eigenvalues greater than 1
index <- max(which(pr$eig$eigenvalue > 1))
mr.test <- pr$ind$coord[,1:index]
mr.test <- as.data.frame(mr.test)
```

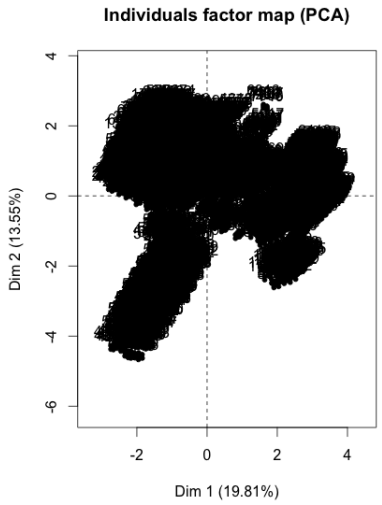
Upon inspection, we see that `index`, which is the number of eigenvalues that are greater than 1, is 5. This means that we should now only have to analyze 5 coordinates, dropping the dimension from 22 to five. We also can see that by these first five vectors can explain 63.19% of the variance which will be sufficient for this problem.

With this basis, we can plot the data in the first three dimensions and see a revealing picture of the data.



We that the first three eigenvectors really do account for the majority of the variance in the data. It also is revealing because it makes it clear that it is much easier to create linear surfaces to separate the classes of mushroom. Again, here we see that green corresponds to poisonous and blue to edible. These two classes of data points are much more evidently separable which will make our jobs much easier when we try to cluster.

From now on, we will use these PCA-transformed data in our analysis.



4 Clusterings

4.1 Preparation

We are now going to look at clustering algorithms to try to separate the poisonous and the edible classes. We will use three different clustering methods: *k-means*, *k-nearest neighbor*, and *hierarchical*.

For each type, we will attempt to break the data into different number of classes: $N = 2$, 3, 5, and 7. As well, we create training and test datasets of various ratios.

4.2 k-means

The k-means algorithm initializes k means and then tries to fit clusters and adjust the means so that the between cluster sum of squares is minimized and the within cluster sum of squares is maximized. We note that k-means does not need a training and test for the algorithm to work, but we have developed a method for using a training and test set. First we train the k means on our initial training data. This produces k centers for the means. We then take those means and fit the test data to the means by assigning a point in the cluster to the class of the closest mean.

The metric that we will use to evaluate how effective the k-means clustering algorithm is on the training and test data will be to rely on how similar the clusters are with respect to their components classes. If the clustering algorithm worked well, we want homogenous groups. That is, we want clusters where all the labels are poisonous and clusters where all labels are edible. It doesn't matter how many clusters. So we analyze the between cluster sum of square/total sum of squares to evaluate the efficacy of the clustering.

Table 1 shows the sum of squares ratio for different training/testing ratios and number of clusters. Table 2 shows the best number of clusters for each method.

Table 1: k-means

	$N=$			
	2	3	5	7
50/50	17.3	41.1	71.8	84.5
60/40	17.4	38.9	70.6	74.3
70/30	27.0	35.2	72.9	84.7

Table 2: k-means Cluster Sizes

		Group						
		1	2	3	4	5	6	7
N=2	50/50	530	3532					
	60/40	1512	1738					
	70/30	917	1521					
N=3	50/50	980	1288	1794				
	60/40	575	698	1977				
	70/30	321	544	1573				
N=5	50/50	326	377	712	902	1745		
	60/40	408	594	699	768	781		
	70/30	311	424	537	567	599		
N=7	50/50	312	377	473	483	669	866	882
	60/40	358	358	367	405	527	548	687
	70/30	253	273	279	306	397	400	530

The most notable thing of this analysis that as N increases, so does the score of the clustering. We do note that this algorithm's metric does not analyze whether the clusters are homogenous, i.e., the ratio of betweenness sum of squares and total sum of squares does not take into account whether the clusters are entirely filled with poisonous or edible mushrooms. That being said, it does approximate it k-means is using the distance metric on the space. The assumption is that the alike mushrooms, e.g. the ones that are edible, are close in distance, and are far from those of dislike mushrooms, the poisonous ones. We have reason to believe this assumption is true, see the 3D figure above to convince yourself of that.

4.3 k-Nearest Neighbors

The idea of k-Nearest Neighbors (knn) is that given a space imbibed with a distance metric will take a training set of data with their a priori labels, along with a number k for which the algorithm will interrogate the k nearest neighbors of a given point to determine what the label of that point should be. After training is completed, the algorithm can attempt to assign labels to a test dataset which does not have labels already attached to it.

We note that this algorithm will not create labels, but will only assign labels that come from the set of possible labels given to it in the training set. In this way, there is no way for us to control the number of clusters the algorithm produces without changing the number of possible labels given to the training data. Now this does not make sense for us to do, given that we are given labels of poisonous or edible. Thus, we will not try to change these and *we understand that we cannot create clusters of size 3, 5, or 7 with knn.* However, we can change k dynamically, so we will produce a matrix where $k = 2$, 3, 5, and 7. This will be sufficient for our purposes.

The *quality metric* that we use on this algorithm is how many of the test dataset does the algorithm get correct? Our given model score will be accuracy of detection which is defined to be

$$\frac{truenegative + truepositive}{truepositive + falsepositive + falsenegative + truenegative.}$$

Table 3: k-Nearest Neighbors Accuracy

	$k=$			
	2	3	5	7
50/50	99.7	99.8	99.6	99.5
60/40	99.2	99.4	99.2	99.1
70/30	99.7	99.6	99.5	99.3

Table 4: k-Nearest Neighbors Cluster Sizes

		Group	
		Edible	Poisonous
k=2	50/50	2098	1964
	60/40	1702	1548
	70/30	1232	1206
k=3	50/50	2096	1966
	60/40	1704	1546
	70/30	1234	1204
k=5	50/50	2104	1958
	60/40	1701	1543
	70/30	1237	1201
k=7	50/50	2101	1961
	60/40	1712	1538
	70/30	1237	1201

With this in mind, we find Table 3 has the accuracy of the algorithm for different k and Table 4 has the group sizes for different k and test/training ratios. We note that Table 4 only has two columns because k-NN will only group the test data into 2 classes because it was only given two classes to train on.

It is very clear from this that we get very good clustering of the dataset into edible and poisonous dataset from this algorithm, $k$nn. We observe that this doesn't reveal much about the third class of unknown edibility, as it will only remain categorizing the data into two classes. A further discussion of this will happen in the Monte Carlo section.

4.4 Hierarchical Clustering

We use Ward's Hierarchical clustering algorithm. Ward's minimum variance method attempts to minimize the variance within each cluster when summed together. The algorithm start with each element of the dataset as its own cluster. Then, the algorithm will try to cluster these together by iteratively finding the pair of clusters that increases this total within variance the least after merging. The increase is defined to be a weighted squared distance between cluster centers.

For Hierarchical Clustering, we are able to define the number of clusters the algorithm for separate the data into. Thus, we are fully able to analyze the clustering algorithm for $N = 2, 3, 5, 7$.

Again, we will use training and test dataset. This algorithm does not naturally have a test and a training component. However, we bootstrap this and approximate a test/training environment. What we will do is to take the test training set and produce a Hierarchical clustering into N clusters. We then take the centers of those clusters and fit the test data to those centers with a simple Euclidian metric. Once that fit as been done, we asses the betweenness sum of squares as a ratio of the total sum of squares, just as we did in k-means.

Table 5 shows the sum of squares ratio for different training/testing ratios and number of clusters. Table 6 shows the best number of clusters for each method.

Table 5: Hierarchical Clustering

	k=			
	2	3	5	7
50/50	20.9	30.8	43.3	47.1
60/40	22.0	32.5	45.9	49.2
70/30	21.8	31.4	43.9	47.2

Table 6: Hierarchical Cluster Sizes

		Group						
		1	2	3	4	5	6	7
N=2	50/50	872	3190					
	60/40	679	2571					
	70/30	561	1877					
N=3	50/50	869	948	2245				
	60/40	674	775	1801				
	70/30	558	558	1322				
N=5	50/50	386	713	880	928	1155		
	60/40	318	577	684	777	894		
	70/30	227	419	551	565	676		
N=7	50/50	310	375	433	507	671	874	892
	60/40	234	307	363	388	543	677	738
	70/30	169	219	287	291	391	522	559

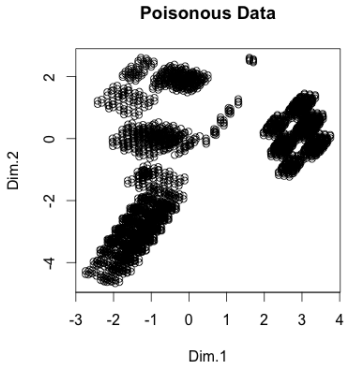
5 Monte Carlo Exploration

Let's perform our analysis on just those mushrooms that are labeled poisonous because we already know that the set of unknown mushrooms fall entirely within the set of mushrooms that are labeled poisonous.

```
mr.poisonous <- mr.test[numeric[,1]==1,]
```

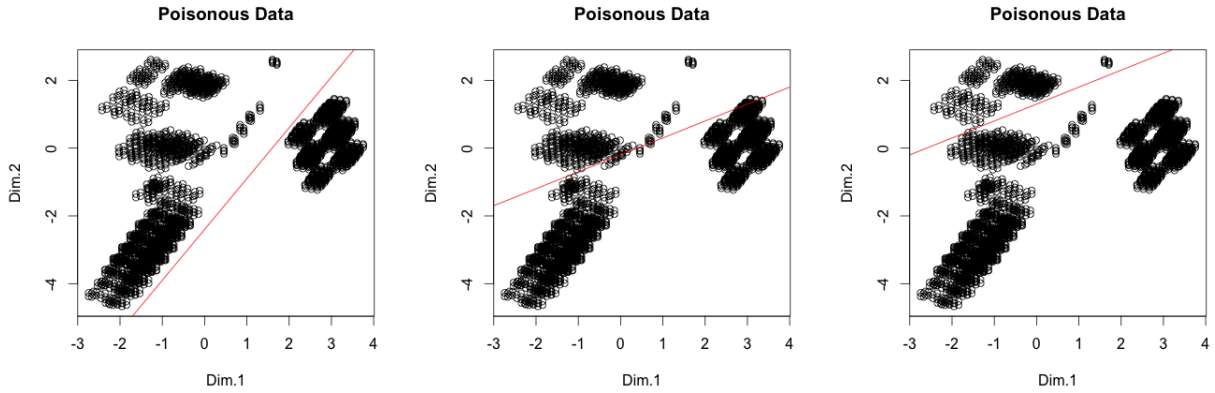
Let us plot these data to understand their properties. We will plot in the first two dimensions only as they explain the most variance of the mushroom dataset.

```
plot(mr.poisonous[,1:2], main="Poisonous Data")
```

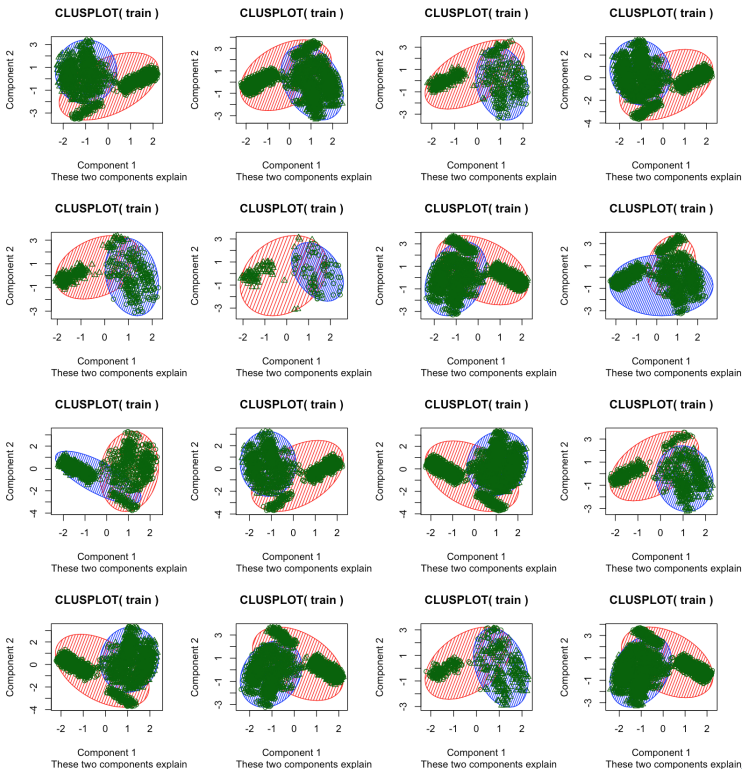


From this plot, we want to find a partition of the data such that the Poisonous classes are linearly separable. This is not an easy task, because the separation line could be draw in a number of ways.

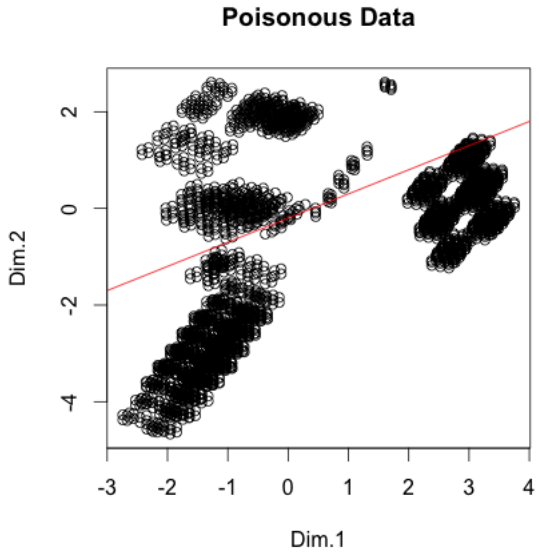
```
abline(-2.4,1.5, col="red")
abline(-0.2,0.5, col="red")
abline(1.3,0.5, col="red")
```



We wonder what the proper clustering is, so we will use a Monte Carlo method to interrogate what line does separate the poisonous data in these two dimensions. We will use Monte Carlo methods to sample from the mushroom dataset and use k-means clustering of the dataset into two portions to determine what cluster is stable under different selections of the seed points for the clustering algorithm. See the code section for specifics on this method, but below is a figure that illustrates the Monte Carlo operation for n=16 samplings:



This illustrates that the most stable clustering of the poisonous dataset into two classes roughly follows this partition:



We further see that the first two principle components contribute the most to the separation of the poisonous and the 'unknown' mushrooms. We note though that this analysis is not perfect as it assumes a roughly even proportion of the data are unknown and poisonous. It further assumes that the two classes found in 2-means clustering are meaningful for this problem. We further state that it may be any one of the obvious clusterings in the above images, but it is hard to know which one is correct because there is such litter a priori knowledge of the problem that was given to us.

6 Discussion

When we look at the three clustering methods, we think that the k-Means method gives us the most insight into the question at hand ? "Which of the mushrooms are actually in the unkown class". Although k-NN is very accurate and is able to predict and classify the test set well, we are unable to train this on 'unknown' so therefore unable to answer the question. Using k-means, we could dive much further to do investigation of the characteristics that stand out within each cluster, whether these characteristics seem to belong to poisonous, edible, or if they seem to stand out as a class of their own. If they do not seem to be part of either 'poisonous' or 'edible' then we can assume that this cluster may be part of the 'unknown' class.

Looking at the data set on a whole, we may think of ways to dive further into the data, to determine the unknown class (given the time and scope of this project these ways would not be feasible) ? or even to verify our conclusions from the clustering in k-means. We could possibly take our 'predicted' 'unknown' cluster observations and label these as 'unknown' in the original data set ? then use k-NN to classify three classes and see how the model does.

We conclude that the best number of clusters for k-Means seems to be 7 clusters on the 70/30 train/test split. For Hierarchical clustering, we would use 7 clusters as well, this time starting with all variables and cutting the algorithm off when it reaches 7 clusters. For k-NN we see that the accuracy is high for mostly all the number of nearest neighbors. For accuracy and performance with the algorithm, we would like to use more neighbors, so we will say k=5 works the best. k=5 is a number of neighbors that is frequently used as a baseline in this methodology and seems to give us 99.5% accuracy on the 70/30 train/test set.