CES 632 Project 3:

Data Mining Clustering on Breast Cancer Data

Chrisina Giagnoni

November 24, 2020

Abstract

Unsupervised learning algorithms give information about data without a "teacher". These algorithms are given unlabeled data and use different measures and techniques to extract information about the data. Clustering is a common technique where distance or similarity measures are used to determine if natural groupings occur within the data. We will compare and contrast three different clustering methods: k-means, hierarchical, and density-based spatial clustering and application with noise (DBSCAN). We will explain each method in detail, how parameters are chosen, as well as run each of the algorithms on a real life data set from fine needle aspirates (FNA) of breast tissue. We will use different validation measures to compare parameters within each technique as well as compare the three different techniques against each other. Finally, we offer our conclusions, limitations, and a discussion on possible future work and open questions.

3.1 Problem description

Clustering is an unsupervised learning technique commonly used in data exploration and data description. In unsupervised learning, we assume no predetermined labels and extract information about the data structure from the data itself. There are multiple clustering algorithms, but all have the common goal of "identifying meaningful groups" (Shalev-Shwartz, Ben-David, 2014). This is especially useful when not much is known about the structure of the data. In general, clustering techniques apply a similarity measure to a data set and group similar observations (Kantardzic, 2020). Each clustering algorithm uses different parameters, similarity measures, and ways of employing similarity measures to determine clusters. In this experiment, we will be using three clustering techniques - k-means, hierarchical agglomerative, and DBSCAN - on the Breast Cancer Wisconsin (Diagnostic) Data Set. Detailed explanation on each of the three algorithms will be presented in section 3.4. After running each algorithm on the data, we will compare and contrast the three algorithms.

The data was retrieved from Kaggle as a CSV file made from the the UC Irvine Machine Learning Repository data set; the only difference between the Kaggle data set and the one from UCI was the inclusion of column labels on the Kaggle CSV file. The three creators of the data set are Dr. William H. Wolberg, General Surgery Dept. University of Wisconsin, Clinical Sciences Center, W. Nick Street, Computer Sciences Dept. University of Wisconsin, and Olvi L. Mangasarian, Computer Sciences Dept. University of Wisconsin. The data set has 30 features, all computed from a digital image of cells taken from a fine needle aspirate (FNA). A FNA is when a sample of tissue is taken by inserting a needle into, in this case, a breast mass (UCI). The data set has 32 columns and 569 rows. The first column is an id number, and the second is a diagnosis of M = malignant, B = benign. Then are the 30 numeric variables describing features of the cell from the FNA. The 30 variables describe the mean, SE (standard error), and "worst" (mean of the three largest values) of 10 real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

A full table is given in the appendix section.

Since we are using the data set for unsupervised learning, we will be removing the column for diagnosis, as well as the ID column. This leaves us with 30 numerical variables, on which we will make no prior assumptions. The data set is already clean, with no missing values or columns consisting of a single value.

This data set has been used multiple times in different academic papers; most use the data set for classification problems. However, Bradley, Bennett and Demiriz used the data set in their paper *Constrained k-means clustering*. They presented a new parameter to be used in k-means clustering which would limit the number of empty or nearly empty clusters which tend to appear on higher-dimensional data where larger numbers of clusters are desired. They used 683 observations and 9 dimensions, indicating that the data set has changed, and/or the authors made changes. They compared using regular k-means clustering and their "constrained k-means clustering" using k values of 10, 20, and 50. They showed that using the constrained k-means clustering on the breast cancer data set resulted in every cluster having a minimum size. While our goal in using clustering is different, we will take the same approach and make no assumptions about the number of clusters.

3.2 Software Tools

R is open source statistical software commonly used in academia and industry. The base R comes with many commonly used functions, but "packages" can be installed and used for functions that do not come with the base software. We used R for preprocessing, dimension reduction, and data mining. The following table summarizes all R functions and packages used.

Package	Description	Usage
dplyr package	Tools for working with data frames	Preprocessing
factoextra package	Functions for visualizing multivariate data analyses	Visualization
gridextra package	Functions for visualizing multivariate data analyses	Visualization
prcomp function	PCA analysis	dimension reduction
kmeans function	k-means clustering	descriptive data mining
hclust function	heirarchical aglomerative clustering	descriptive data mining
dbscan package/function	DBSCAN clustering	descriptive data mining
clValid package/function	clustering validation	descriptive data mining
fpc	flexible procedures for clustering	descriptive data mining

Many of the R functions used were built-in, and all packages are cited in the references. Detailed explanation of PCA will be given in section 3.3 and each of the three clustering algorithms will be explained in section 3.4.

3.3 Preprocessing

Preprocessing of data, and dimensionality reduction. Describe the tools used, and the results obtained. Exploratory analysis and preliminary experiments performed with the data: explain iterations performed. Analyze possibilities of data mining with different tools. Apply all preprocessing and data reduction techniques you assume they are necessary and explain why. For every preprocessing technique:

The Breast Cancer Wisconsin (Diagnostic) Data Set had 32 columns. The first thing we did was remove the ID number and diagnosis columns; since we are using the data set for unsupervised learning, neither column will be useful. The data had no missing values, or columns with a single value. The remaining 30 features were all numerical, so we started by looking at the correlations of all of the variables.

Partial correlation matrix for Breast Cancer Data					
	radius mean	texture mean	perimeter mean	area mean	smoothness mean
radius mean	1.0000000	0.32378189	0.9978553	0.9873572	0.17058119
texture mean	0.3237819	1.00000000	0.3295331	0.3210857	-0.02338852
perimeter mean	0.9978553	0.32953306	1.0000000	0.9865068	0.20727816
area mean	0.9873572	0.32108570	0.9865068	1.0000000	0.17702838
smoothness mean	0.1705812	-0.02338852	0.2072782	0.1770284	1.00000000

We can see that there are some highly correlated variables. To illustrate the point, we plotted perimeter mean and radius mean to see a nearly perfect positive correlation.

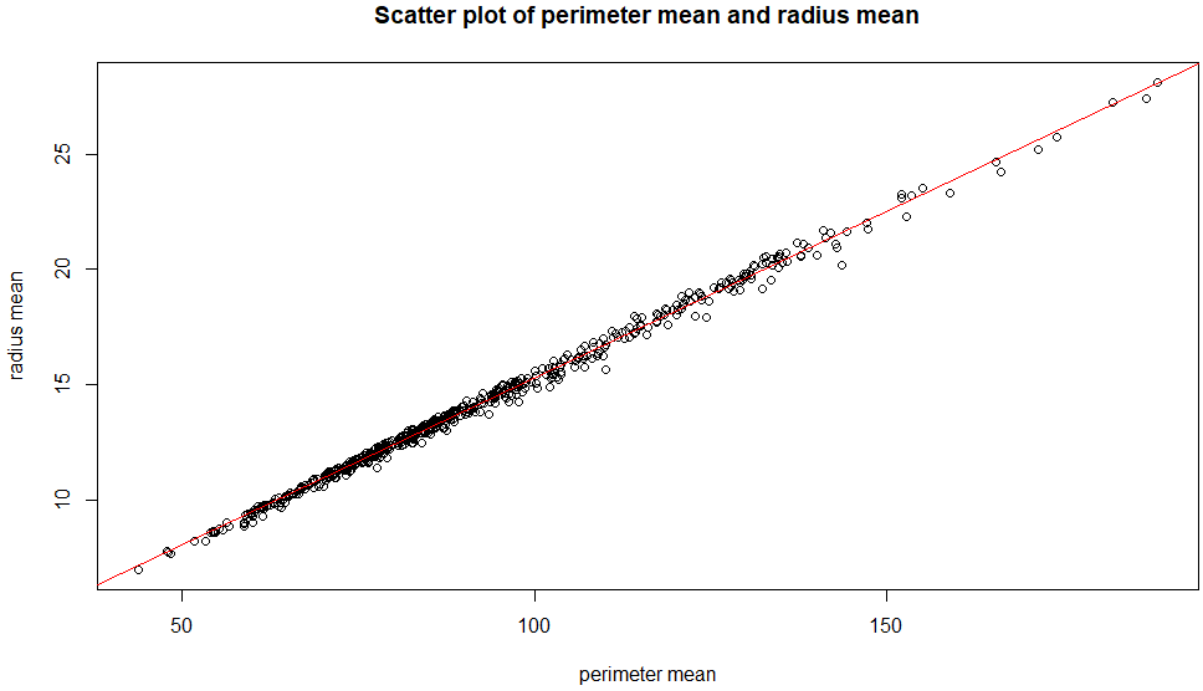

Figure 1: Scatter plot of perimeter mean and radius mean

Highly correlated variables essentially give the same information, so it is not useful to keep pairs of variables with high correlation. In this case, principal component analysis (PCA) would be the best way to reduce dimensionality by compressing the correlated variables. Formally in PCA, we take an n-dimensional vector $(x_1, x_2, ..., x_n)$ and transform it onto an m-dimensional vector $(y_1, y_2, ..., y_m)$ where $m \leq n$. The $m-$dimensional vector stores most of the information in the first few dimensions (Kantardzic, 2020). These first few dimensions in the m-dimensional vector "explain" a certain amount of variance in the data, and we can visualize this with a scree plot. PCA is performed by first calculating the covariance matrix, then finding the eigenvalues and eigenvectors of the matrix. The eigenvalues correspond to eigenvectors which give us the principal axes. Thus we order the eigenvalues from largest to smallest, and that tells us which principal component axis gives us PC1, PC2, ... , PCn. We then decide on the number of principal components to use and transform the original data set by matrix multipilcation as follows

$$Y = \Lambda X$$

Where Y is the transformed data, Λ is the chosen principal components (matrix of eigenvectors), and X is the original data matrix. Then we use Y in place of X for analysis.

We also note that PCA is sensitive to very large or very small values, i.e. bigger values will be more influential on the principal components than smaller values. Thus it is recommended to always scale the data, which we did within the PCA function. In our case, the function scales the data using the formula

$$z_i = \frac{x_i - \overline{x}}{s}$$

where z_i is the new scaled data point, x_i is the original value, $\overline{x}$ is the column mean, and s is the column standard deviation. For example, the radius mean for the first observation is 17.99, the mean and standard deviation for the column are 14.12729 and 3.524049, respectively, so the scaled data point is

$$1.0961 = \frac{17.99 - 14.12729}{3.524049}.$$

For our data set, we used the built in `prcomp` function to scale and then find the principal components, and the `factoextra` package's `fviz_screeplot` function to give us a scree plot. The scree plot shows us the percentage of variance explained by each principal component, so we see that PC1 explains 44% of the variance and PC2 explains 19% of the variance. In general, there are two ways to determine the number of principal components to use. In our case, we choose to pick the number of dimensions that explain at least 90% of the variance (Kantardzic, 2020). For our breast cancer data set we need 7 principal components to explain at least 90% of the variance. Thus, moving forward into data mining, we created a new data frame from the first 7 principal components to use.

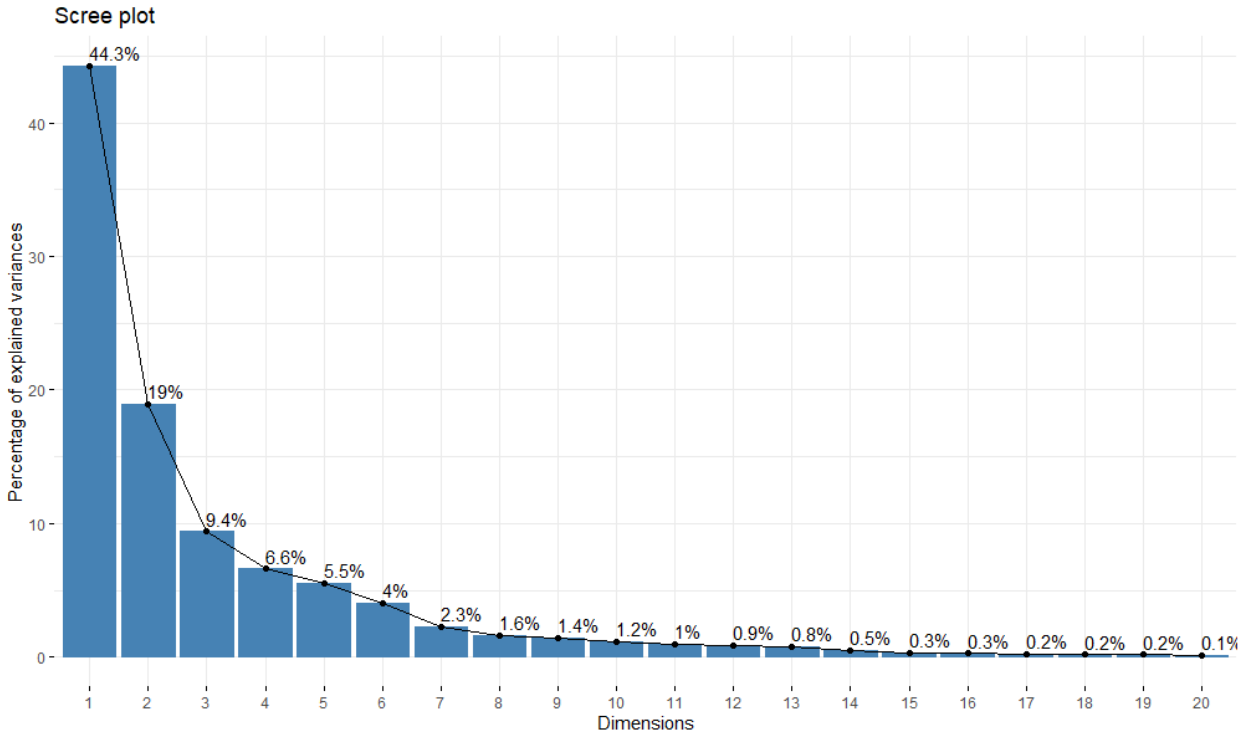

Figure 2: Scree plot for percentage of explained variance per dimension

3.4 Data mining process

Clustering is an unsupervised learning method, meaning that we are using algorithms that employ distance or similarity measures to group data. We will be using three different clustering algorithms - k-means,

hierarchical, and DBSCAN. Each technique has different input parameters and outputs potentially different clusters. Each technique has their own strengths and weaknesses, and in general, for all three techniques one of the biggest challenges is how to choose input parameters. We will discuss each algorithm, how we chose parameters, and also use clustering validation when applicable.

For clustering validation, we used the `clValid` and `fpc` packages in R. The `clValid` package offers three types of clustering validation: internal, stability, and biological. For k-means and hierarchical clustering, we used the internal validation. Unfortunately, the package does not have an input fort DBSCAN so we will evaluate its performance with other measures. Internal validation measures how connected and compact the clusters are, and how separated different clusters are (Brock, et al. 2008). The function `clValid` uses three measurements for internal validation: connectivity, Dunn, and silhouette. The connectivity measure essentially gives a penalty for close points being in different clusters, and we want to minimize the connectivity measure. Silhouette width averages each observation's silhoutte value, which measure the degree of confidence in the cluster label using distance measures. A well-clustered observation has a score of 1 and a poorly clustered observation has a score of -1. Thus, we want to maximize the silhouette width. The Dunn Index is the ratio between the distance of closest points in different clusters and furthest points in the same cluster. We want to maximize this value. (Brock, et al. 2008). Since `clValid` does not allow for evaluation of DBSCAN, we decided to use the `fpc` package to evaluate that algorithm. The three validation measures we used were Dunn index, sillhouette width, and within cluster sum of squares. This allowed for a direct comparison of all three algorithms.

k-means

The first technique we used was k-means clustering, which seeks to form clusters based on a similarity or distance measure by putting points that are similar together and points most dissimilar into different k clusters. The k-means algorithm requires three inputs: k, the number of clusters; the distance measure; and how to choose the initial cluster centroids (center point of the cluster). For our purposes, we used Euclidean distance since all of our data were numerical measurements. In each cluster, the algorithm minimizes the within cluster sum of squares, which is the Euclidean distances between the points in their assigned cluster and the cluster's centroid, given by the following equation

$$w(C_j) = \sum_{x_i \in C_j} (x_i - \mu_j)^2, j = 1, ..., k$$

where C_j is the jth cluster, x_i is a point in C_j, and μ_j is the centroid of the jth cluster. A small $w(C_j)$ tells us that the cluster is compact, and a large $w(C_j)$ tells us that the data points are more spread out from their cluster's centroid. So K-means clustering seeks to minimize the total within cluster sum of squares (TSS), which is simply a sum of within cluster sum of squares over all k clusters (Boehmke). This is given by the equation

$$\sum_{j=1}^{k} w(C_j) = \sum_{j=1}^{k} \sum_{x_i \in C_j} (x_i - \mu_j)^2.$$

There are different ways to choose k, either based on prior knowledge or by comparing the TSS for different k. The latter is the method we used for determining k. The function `fviz_nbclust` gives the following plot which compares the TSS for different values of k. Similar to the scree plot for PCA we look for an "elbow" in the plot. Based on the plot, we decided to run the k-means algorithm for $k = 2, 3, 4$ and 5.

Using the `kmeans` function in R, we input the number of desired clusters, and arbitrarily set the `nstart` parameter to 100. This tells the function to randomly pick k centroids to form clusters 100 times, then output the results that produce the smallest TSS. The default distance measure on the function is Euclidean distance, so we did not have to specify it.

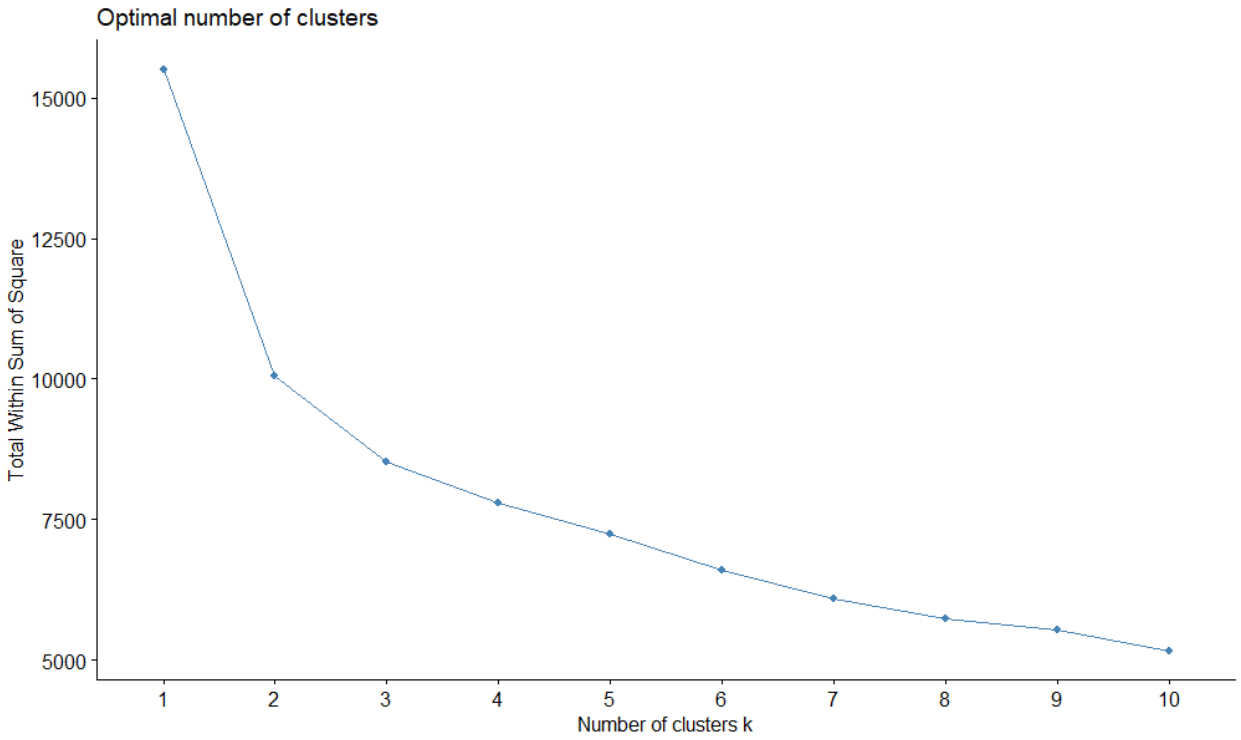

Figure 3: Total within sum of squares and number of clusters

Since our data set is 7-dimensional, we can only visualize the clusters for two factors at a time. We created a plot of the data in the first two principal components (first two dimensions) within their respective clusters for each k. While interesting, it really only gives us a small snapshot of what the clusters actually look like.

After running the k-means algorithm, R outputs the ratio: between ss/ total ss. We hope to make the ratio as close to 1 as possible. We have compiled a table of this output. We can see that as k increases, this ratio does as well. Based on this criteria alone, we would think that out of the four k values, $k = 5$ is the best, however we will also evaluate the results with internal clustering validation.

k	between SS / total SS
2	35.2 %
3	45.1 %
4	50.2 %
5	54.7 %

Using the `clValid` function we found that $k = 2$ had the best connectivity and silhouette scores, and $k = 4$ had the best Dunn score. The results of `clvalid` are summarized in the tables below.

Validation Measures:				
	2	3	4	5
Connectivity	63.2925	108.9353	157.6238	204.2611
Dunn	0.0527	0.0551	0.0565	0.0499
Silhouette	0.3710	0.3376	0.3003	0.2121

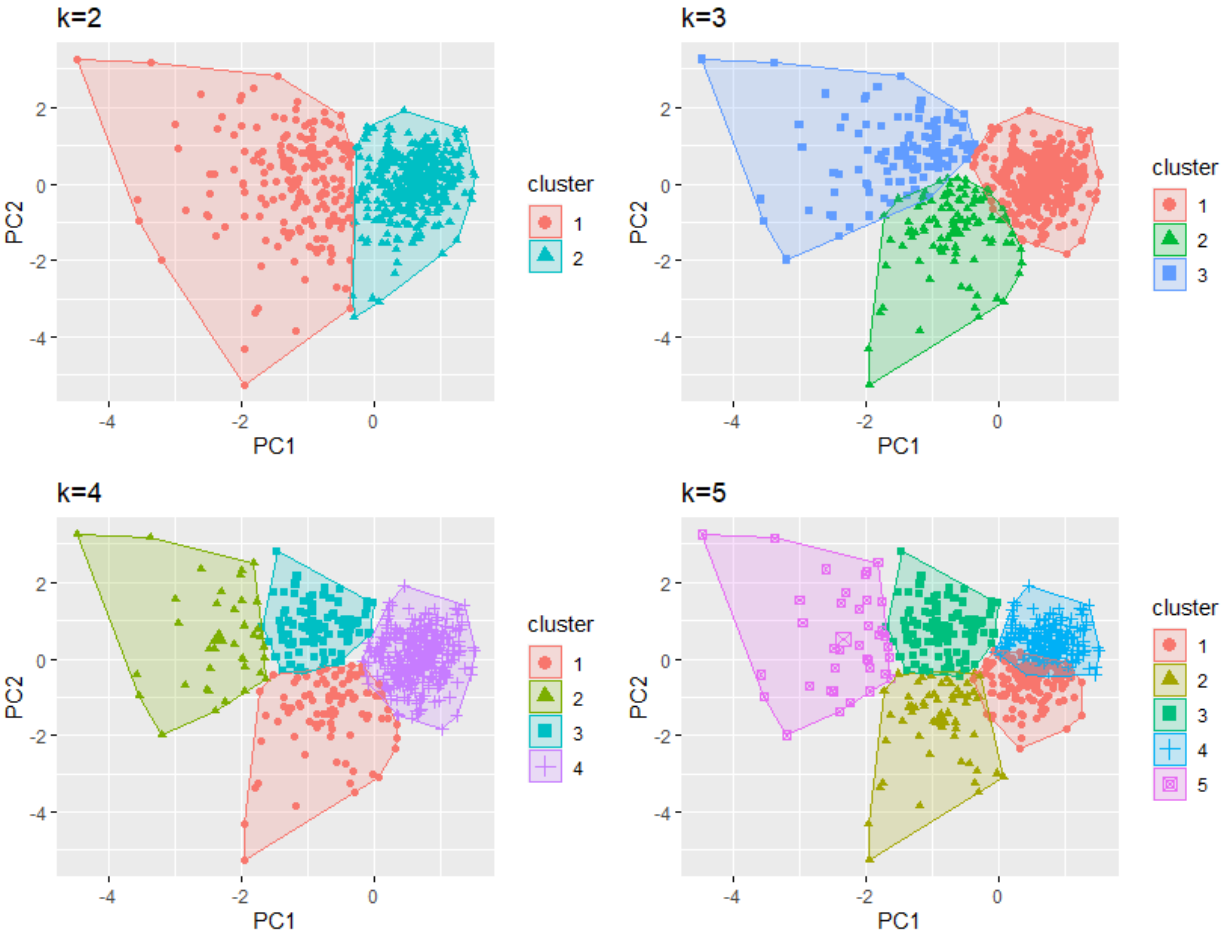

Figure 4: Plot of first two PCs and k-means clusters for k=2,3,4,5

Optimal Scores:			
	Score	Method	Clusters
Connectivity	63.2925	kmeans	2
Dunn	0.0565	kmeans	4
Silhouette	0.3710	kmeans	2

We used four measures of evaluation, the ratio between ss/ total ss, connectivity, Dunn index, and silhouette width. Out of all measures, it appears that $k = 2$ performs the best. K-means clustering could be used as a static model for classification, although that is not the goal of clustering. Since clustering methods describe data, we would re-run the entire algorithm to redefine clusters. However, if we collected new observations of FNA of breast mass, we could use the k-means "model"; this would classify new observations into one of the clusters based on the distance it is from each centroid.

Hierarchical Agglomerative Clustering

Heirarchical agglomerative clustering was the second data mining technique we used. In this algorithm, each observation is in its own cluster, then at each step, using a distance or similarity measure, the algorithm begins to group observations into coarser clusters until all observations are in a single cluster. A threshold value (of distance or similarity measure) determines how the number of clusters (Kantardzic, 2020). We used the `hclust` function and followed Robert Kabaccoff's tutorial on using the function in R. We first created a matrix of distances between all data points using Euclidean distance. Then the function grouped the clusters from fine to coarse. We can plot the dendrogram and visualize all clusters at different thresholds. We have the clusters and thresholds marked on the dendrogram for clusters of size 2, 3, 4, and 5 in red, blue, orange, and green, respectively. So we can see that as we allow for larger distances between points, the number of clusters becomes smaller.

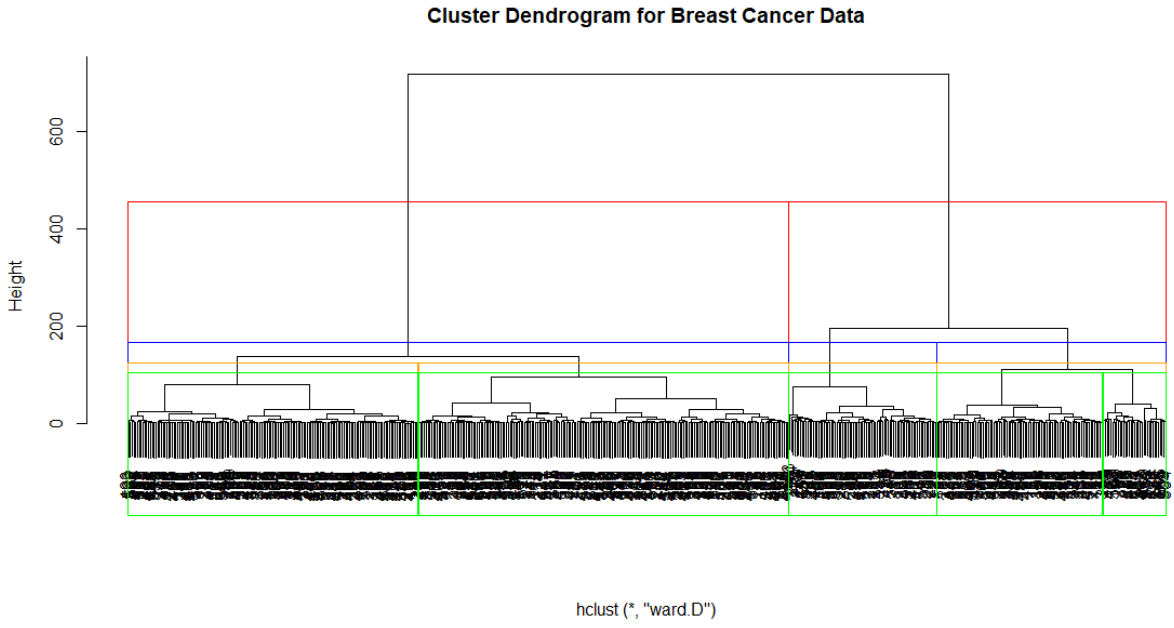

Figure 5: Cluster Dendrogram with different threshold values

Using the `clValid` function we found that two clusters had the best validation scores. The results of `clvalid` are summarized in the tables below.

Validation Measures:				
	2	3	4	5
Connectivity	3.8579	17.6218	18.1968	21.1258
Dunn	0.4979	0.1612	0.1612	0.1612
Silhouette	0.6665	0.5070	0.4593	0.4097

Optimal Scores:			
	Score	Method	Clusters
Connectivity	3.8579	hierarchical	2
Dunn	0.4979	hierarchical	2
Silhouette	0.6665	hierarchical	2

Unlike k-means, in hierarchical agglomerative clustering, we can not misuse it as a model. Given a new observation, we would make an entirely new clustering dendrogram. Since hierarchical clustering is a "bottom up" method of clustering, it is sensitive to a new data point. Thus, a new data point could change the optimal number of clusters.

DBSCAN

The third clustering technique we used was density-based spatial clustering and application with noise (DBSCAN). A density-based approach to clustering means that clusters are dense areas separated by less dense areas. Thus, unlike many other clustering algorithms, DBSCAN has the capability of identifying non-convex clusters. The algorithm also has two input parameters: ε and m where ε is the size of the neighborhood around a point, and m is the minimum number of points within each cluster. While there is no need to predetermine the number of clusters, we must decide how dense the clusters must be. In DBSCAN, using the two parameters, ε and m, the algorithm defines three types of data point: core, border, or noise. A core point has at least m other points in it's ε neighborhood; a border point has less than m other points in it's ε neighbood, but is neighbors with a core point; and a noise point is a point that is neither core nor border point. Then the algorithm outputs the clusters and data point labels (Kantardzic, 2020).

Depending on how we set the parameters can greatly change the shape, size, and number of clusters the algorithm identifies. Thus, one of the biggest challenges in using the DBSCAN method is choosing ε and m. Ideally, m would grow with the size of the data set. We tried to use multiple m values, but any value greater than 5 produced either no clusters or one cluster depending on ε. Thus, we used $m = 5$. To choose ε we created a plot for the number of points within a certain distance of eachother, and chose a value at the "elbow" of the graph (Kassambara). In our case, we decided to compare four ε values: 1, 2, 2.5, and 3.

We can visualize only two components of the seven used in the data at a time, so we plotted the first two principal components and the clusters that DBSCAN identified for each ε. We can see that DBSCAN has found different clusters based on the chosen ε. We summarized the results in a table.

ε	number of clusters	number of noise points
1	2	554
2	3	194
2.5	2	95
3	1	59

Note that the larger ε becomes, the fewer noise points there are, and that for $\varepsilon \geq 3$ the number of clusters stabilizes to 1, and eventually every point is in a single cluster, i.e. there are no noise points. To evaluate the performance of DBSCAN, we used the `cluster.stats` function from the `fpc` package, and looked at three statistics: Dunn Index, average silhouette width, and within cluster ss. The results are summarized in the following table.

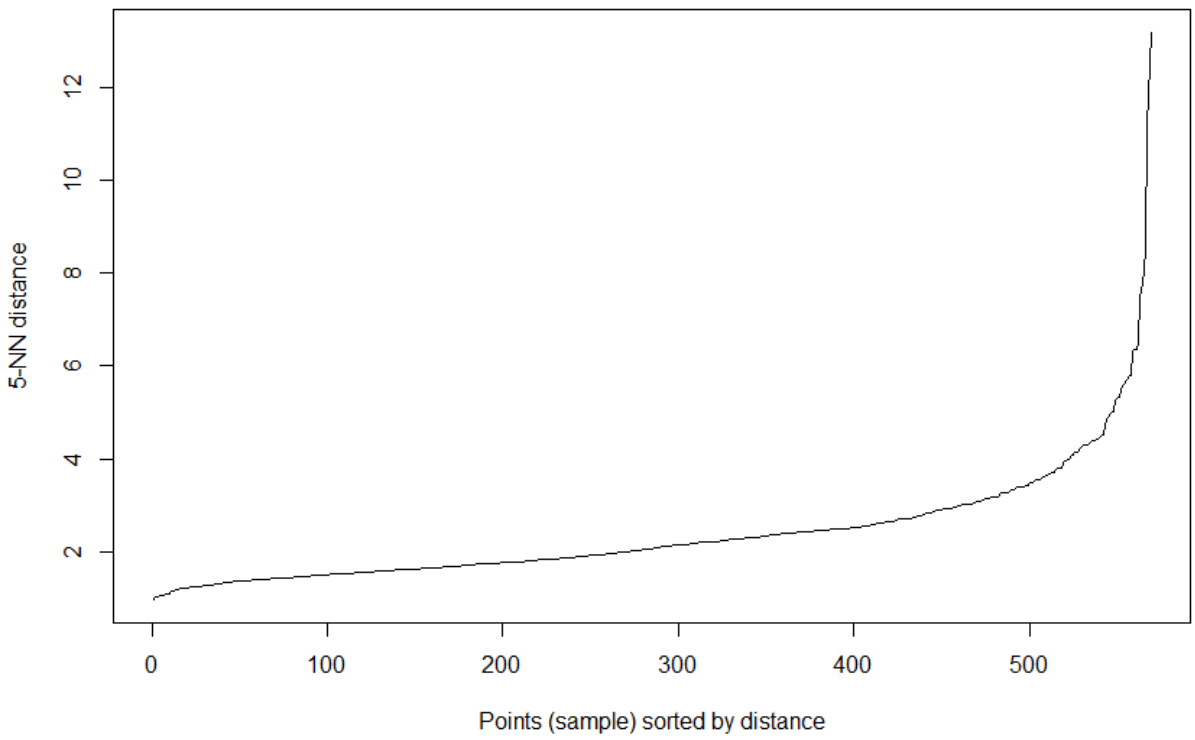

Figure 6: 5-NN plot of distances for different epsilon values

ε	Dunn	Silhoutte width	within cluster ss
1	0.022	-0.278	15381.96
2	0.042	-0.060	13014.62
2.5	0.061	0.273	12841.95
3	0.107	0.392	13417.71

If one cluster was acceptable, then the larger ε values give a better Dunn Index and silhouette width values. However, we see that out of the four values, the within cluster sum of squares has a local minimum at 2.5.

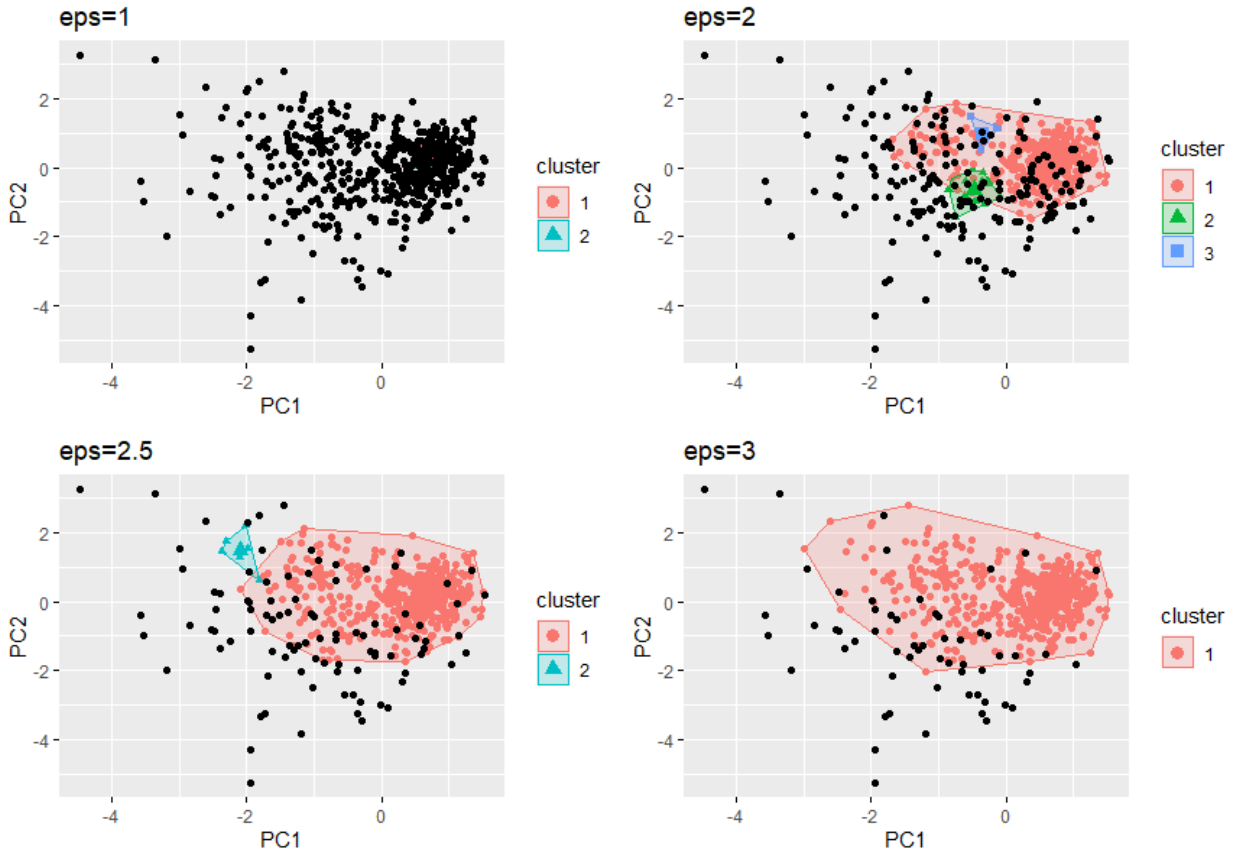

Figure 7: Plot of first two PCs and DBSCAN clusters for epsilon=1,2,2.5,3

Similar to k-means clustering, since DBSCAN uses fixed parameters independent of the data itself, it is possible to use this method as a static model and classify a new data point into a cluster.

Comparison

Below we have summarized some of the validation statistics for each of the techniques according to their "best" parameters. Recall that we want to maximize both the Dunn index and silhouette width. Recall that Silhouette width averages the degree of confidence we have in a point's cluster assignment and the Dunn Index is a ratio that measures how far points are within a cluster versus how close points are in different clusters. In both measures, we aim to maximize the values. We can see that the hierarchical clustering method performs the best on both scores when the threshold value gives two clusters. Out of the tested parameters, k-means and DBSCAN perform similarly.

Method	Parameters	Dunn index	Silhoutte width
k-means	k=2	0.0527	0.3710
Hierarchical	k=2	0.4979	0.6665
DBSCAN	$\varepsilon = 2.5$, m=5	0.061	0.273

While we can not visualize the reduced data set since it is 7-dimensional, we can infer some things from these results. All three methods, with the "best" parameters, i.e. the ones that gave the best validation results, decided there were two clusters. We have included a side-by-side comparison for k-means with $k = 2$ and DBSCAN with $m = 5$, $\varepsilon = 2.5$ with plots of the first two principal components.

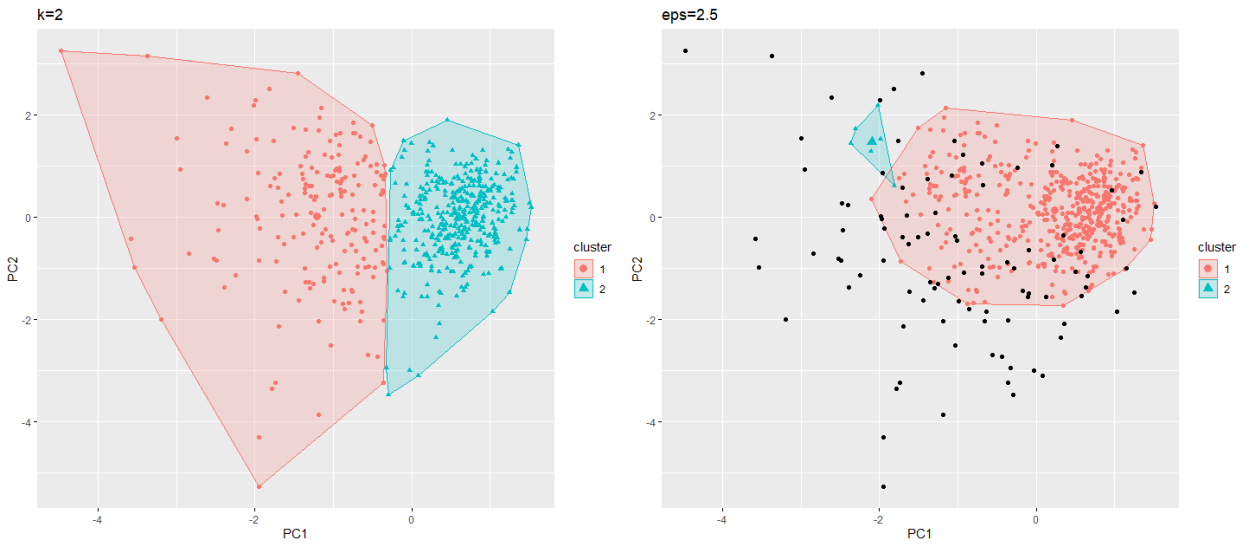

Figure 8: Comparison of k-means with k = 2 and DBSCAN with m = 5, epsilon = 2.5

We can see that both algorithms concluded that the dense region on the right side was its own cluster, but disagreed on what to do with the less densely populated areas of the data. For the given parameters, k-means grouped the dispersed points as its own cluster, whereas DBSCAN classified the points as noise. Further tuning of the DBSCAN parameters may have resulted in similar clusters to k-means; perhaps a better fit since DBSCAN has the potential to find non-convex cluster shapes.

Comparing all three methods, the idea of a noise point only exists for DBSCAN, so if we want to cluster every point, a better choice would be k-means or hierarchical. In this case, hierarchical outperformed k-means in our validation measures. K-means and DBSCAN depend on overall distances of the point - either through distance to a centroid or distances of m points from each other - and in this way the hierarchical algorithm is more flexible. On this data set, that flexibility made the hierarchical algorithm a better fit.

3.5 Conclusions

After preprocessing and performing dimensionality reduction through PCA we performed a cluster analysis on the breast cancer data set. We have run three different clustering algorithms on unlabeled breast cancer data: k-means, hierarchical agglomerative, and DBSCAN. Within each technique, we compared different parameter values, discussed how we chose the values, and then used clustering validation measures to compare the performance of different parameter values. Then we compared the three techniques based on their "best" parameters, and based on our validation measures we concluded that hierarchical clustering with a threshold value that gives two clusters produced the best clusters.

In section 3.3 we noted that many of the variables had high correlations, so we decided that PCA would be the best way to handle the correlated variables and reduce dimensionality. We arbitrarily chose to use 7 principal

components based on the fact that they explain at least 90% of the variability in the data. It is possible that choosing a different number of principal components could have yielded different results. There are also other methods of dimensionality reduction we could have used. We could have removed correlated variables before performing PCA, and may have had a different outcome or even needed fewer principal components. One of the biggest drawbacks in PCA is the loss of interpretability, since the features compress. If we wanted to keep the features intact, we could have set a threshold for correlation and removed correlated variables. However, deciding which variables to eliminate or keep would have been quite subjective, unlike PCA. We also could have used the diagnosis labels and a decision tree or random forest to give insight into which features to keep, but this could impose a two-cluster structure on the data. Deciding how to reduce dimensionality on this data set is an area that could be further explored, and we could see if and how the results of clustering change.

The validation measures we used were only a few of the many that exist. It is possible that if we used different measures that we would have decided that different parameters or different algorithms were better. Both the `clValid` and `fpc` packages have many more validation measures we did not use. In particular, we only used internal validation and did not explore stability or biological validation for these algorithms. This was in part due to ease of interpretability, and because our data set was entirely numerical and we were using Euclidean distance as the distance measure in all three algorithms, it made sense to use internal measures since they are distance based. This is an area that warrants further exploration.

In general, we used different techniques to choose parameters for k-means and DBSCAN, and choosing parameters for these methods is one of the most challenging parts of using these algorithms. We could have explored different methods of choosing or fine tuning parameter selection. In particular, we could have employed more validation measures to decide what parameters were best. Furthermore, ε in DBSCAN can be any positive, real number, so we could have tried different values between 2 and 3, since we noted that 2.5 produced a local minimum in the within cluster sum of squares measure. Perhaps 2.25, or 2.75 would have produced even better results. Since the k parameter in k-means must be a positive integer value, we might have attempted to see if even larger values of k produced better results, but in this case we might be inclined to use the constrained k-means clustering algorithm presented by Bradley et al. to avoid nearly empty clusters produced by large values of k.

We want to note that for most parameter values, DBSCAN either classified most points as either in a single cluster or as noise points. Looking at figure 7, we can see that the data (in the first two principal components) is dense in one area and less dense in the area to the left. We want to propose a change in the algorithm that introduces a new threshold parameter for the number of noise points allowed and a variable ε value. A brief search turned up no such algorithm; however, one may exist upon further investigation. An explanation of the adjusted DBSCAN algorithm is as follows. We set three initial parameters: m the minimum number of points in a cluster, n a maximum allowed number of noise points, and ε_i, $i = 1, 2, ..., k$, is the neighborhood size at iteration i. The ε_i is adjusted at each iteration of the algorithm, until the number of noise points is less than or equal to n. Some amateur psudocode for the proposed algorithm is as follows

```
algorithm: DBSCAN(m, epsilon_i), i in {1,2,3,...}
initialize m, n, epsilon_1
  Repeat DBSCAN(m, \varepsilon_i)
    output clusters_i
    until number of noise points < n
  return clusters_i
end
```

Fine tuning and testing this updated algorithm would be an interesting area for future work (or finding an existing one) since DBSCAN does not have the capability to identify clusters of varying density.

We have seen that there are areas for improvement in clustering procedures. In the initial stages of pre-processing, we had some areas in feature selection and dimensionality reduction that may have been better executed. Each clustering algorithm has parameters, and finding the best parameters is a challenge on its own that warrants its own area of research. We proposed some methods for fine tuning parameters, such as

employing other validation techniques. Finally we also noted that adjustments to the DBSCAN algorithm may make it more flexible, which is one of the strengths of hierarchical clustering. For this data type we found that hierarchical clustering outperformed DBSCAN and k-means clustering methods by our validation methods, and in the future we may find other validation methods draw different conclusions.

4 Appendix

Table of features

Feature	type
id	integer
diagnosis	character
radius mean	numeric
texture mean	numeric
perimeter mean	numeric
area mean	numeric
smoothness mean	numeric
compactness mean	numeric
concavity mean	numeric
concave.points mean	numeric
symmetry mean	numeric
fractal dimension mean	numeric
radius se	numeric
texture se	numeric
perimeter se	numeric
area se	numeric
smoothness se	numeric
compactness se	numeric
concavity se	numeric
concave.points se	numeric
symmetry se	numeric
fractal dimension se	numeric
radius worst	numeric
texture worst	numeric
perimeter worst	numeric
area worst	numeric
smoothness worst	numeric
compactness worst	numeric
concavity worst	numeric
concave.points worst	numeric
symmetry worst	numeric
fractal dimension worst	numeric

References

Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

Christian Hennig (2020). fpc: Flexible Procedures for Clustering. R package version 2.2-8. https://CRAN.R-project.org/package=fpc

Kassambara, A. (2018). "DBSCAN: Density-Based Clustering Essentials." *Data Novia* https://www.datanovia.com/en/lessons/dbscan-density-based-clustering-essentials/

Hahsler M, Piekenbrock M, Doran D (2019). "dbscan: Fast Density-Based Clustering with R." *Journal of Statistical Software*, *91*(1), 1-30. doi: 10.18637/jss.v091.i01 (URL: https://doi.org/10.18637/jss.v091.i01).

"Hierarchical Clustering". *R Documentation*. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/hclust.html

Kabacoff, R. (2017). "Cluster Analysis" *Quick-R* https://www.statmethods.net/advstats/cluster.html

Kantardzic, M. (2020). Data mining: Concepts, models, methods, and algorithms. Hoboken: Wiley.

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2020). dplyr: A Grammar of Data Manipulation. R package version 1.0.2. https://CRAN.R-project.org/package=dplyr

Alboukadel Kassambara and Fabian Mundt (2020). factoextra: Extract and Visualize the Results of Multivariate Data Analyses. R package version 1.0.7. https://CRAN.R-project.org/package=factoextra

Bradley, P., Bennett, K., Xemiriz, A. May 2000 "Constrained K-Means Clustering" *Microsoft Research* microsoft.com/en-us/research/publication/constrained-k-means-clustering/

Boehmke, B. "k-means Cluster Analysis." *UC Business Analytics R Programming Guide.* https://uc-r.github.io/kmeans_clustering

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta (2008). clValid: An R Package for Cluster Validation. Journal of Statistical Software, 25(4), 1-22. URL http://www.jstatsoft.org/v25/i04/.

Shalev-Shwartz, S., & Ben-David, S. (2019). Understanding machine learning: From theory to algorithms. Cambridge: Cambridge University Press.