Hw 2

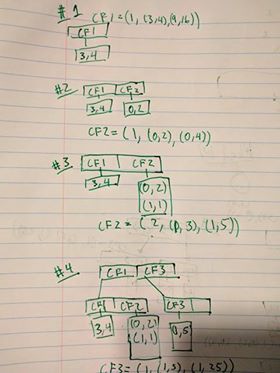
Alex Shung – 504049665

1. K-mean
   1. Distances to the center points

|  |  |  |
| --- | --- | --- |
| Point | Distance | Cluster |
| X1 | 4.123 | 1 |
| X2 | 0 | 1 |
| X3 | 5.916 | 2 |
| X4 | 0 | 2 |
| X5 | 5.744 | 1 |
| X6 | 4.690 | 1 |

* 1. Look above for clustering
  2. Cluster 1 center: (2.75, 2.5, 2.25), Cluster 2 center: (7.5, 7.5, 4.5)
  3. We know the algorithm converges if the new centers did not change the clustering / did not lower the cost

1. Birch

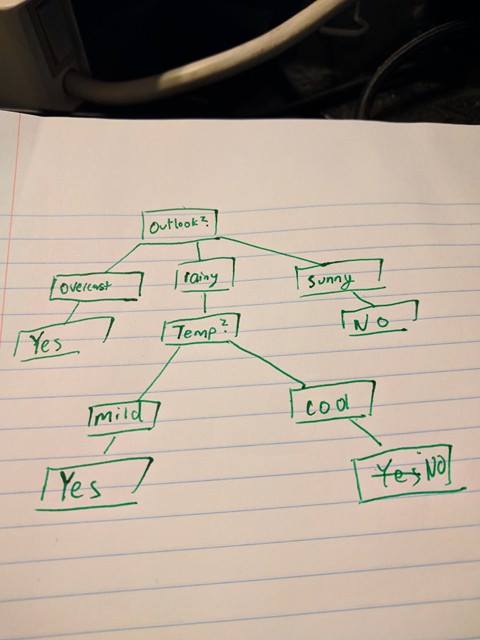


1. DBSCAN

|  |  |  |
| --- | --- | --- |
| ID | Cluster | Type |
| 1 |  | Outlier |
| 2 |  | Outlier |
| 3 |  | Outlier |
| 4 | 1 | Border |
| 5 |  | Outlier |
| 6 | 1 | Core |
| 7 | 1 | Core |
| 8 | 1 | Core |

1. OPTICS
   1. 6,7,8,1,2,3,4,5
   2. OPTICS produces an ordering (instead of a strict partitioning) where as DBSCAN produces a clustering. To do its ordering, OPTICS sorts the points by the reachability -distance of the points present (utilizing a priority heap to do so). OPTICS is not constrained to a single epsilon, and uses the passed epsilon as a maximum (or a very loose threshold) / is optional. Thus it is able to produce information which can show clustering of various hierarchies. It also parses the points in a different order, exhausting a single cluster (depth first) before moving on.
2. Classifications
   1. Split decision tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| First Split |  |  |  |  |
| Attribute | Info(4,2) | Entropy Eq | Entropy | Info Gain |
| Temp | 0.918296 | 5/6 I(4,1) + 1/6 I(1) | 0.601608 | 0.316688 |
| Outlook | 0.918296 | 2/6 I(2) + 1/6 \*0 + 3/6 I(2,1) | 0.45915 | 0.459146 |
| Humidity | 0.918296 | 4/6 I(3,1) + 2/6 (1,1) | 0.8742 | 0.044096 |
| Windy | 0.918296 | 4/6 I(2,2) + 2/6 (2) | 0.666667 | 0.251629 |
|  |  |  |  |  |
| Second Split | |  |  |  |
| Attribute | Info(2,1) | Entroy Eq | Entropy | Info Gain |
| Temp | 0.9183 | 2/3 I(2) + 1/3 I(1) | 0 | 0.9183 |
| Humidity | 0.9183 | 2/3 I(1,1) + 1/3 I(1) | 0.666667 | 0.251633 |
| Windy | 0.9183 | 2/3 I(1,1) + 1/3 I(1) | 0.666667 | 0.251633 |



* 1. Naïve Bayes Classification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Play Golf? | Temp: mild | Humid: high | Outlook: rain | Windy: false | P(X|golf) |
| yes | 4/4 | ¼ | 2/4 | 2/4 | .0625 |
| No | ½ | ½ | ½ | 0/2 | 0 |

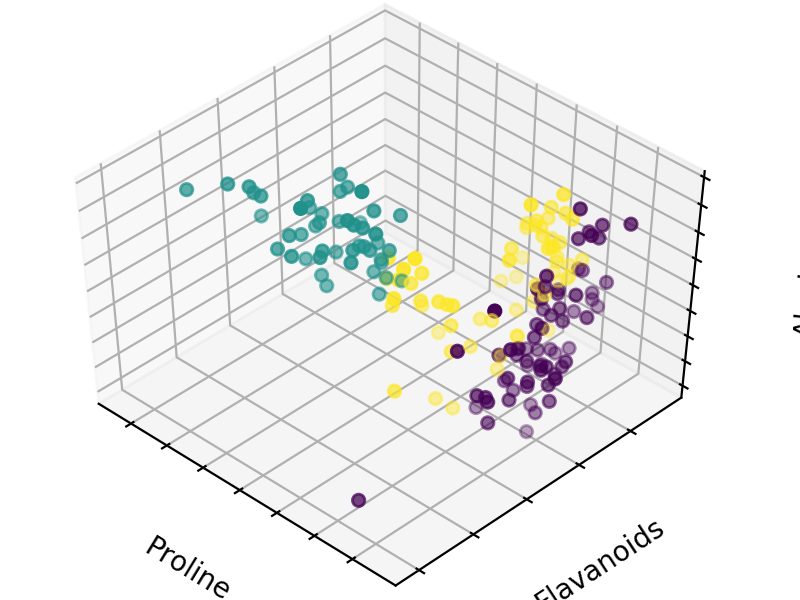
Probability that you play = P(X|golf) \* P(yes) = 0.0625 \* 4/6 = 1/24 = 0.04166

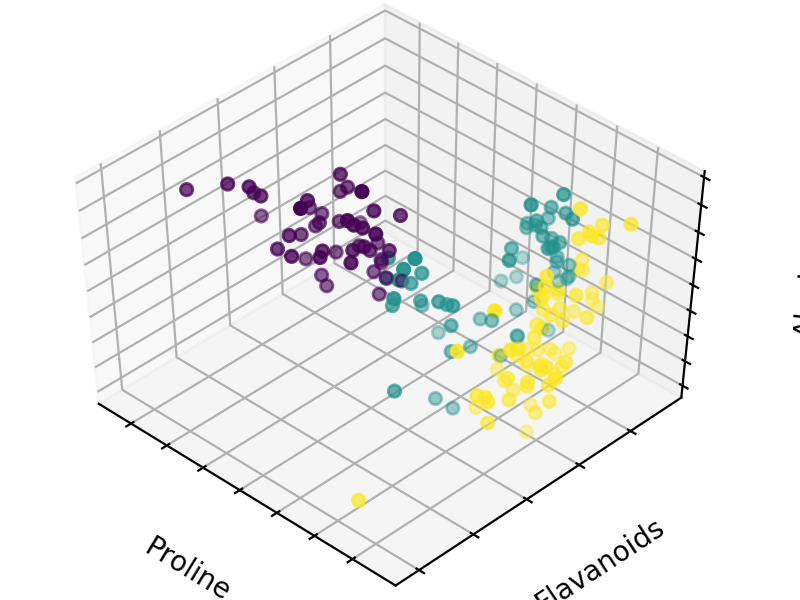
Probability that you don’t play = P(X|golf) \* P(no) = 0

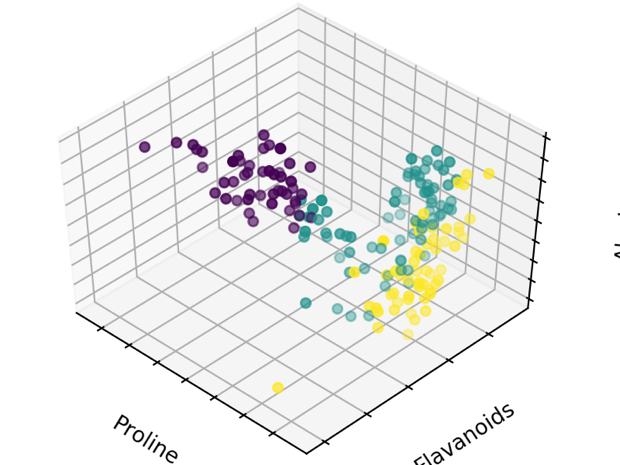
Therefore predict that you will play

* 1. Naïve bayes will need less data than a decision tree as it does not need all enumerations to perform effectively

1. Programming Assignment
   1. Produces the three featuers:
      1. Proline, Flavenoids, Alcohol (Z is alcohol, slightly cut off)
   2. With cluster = 3

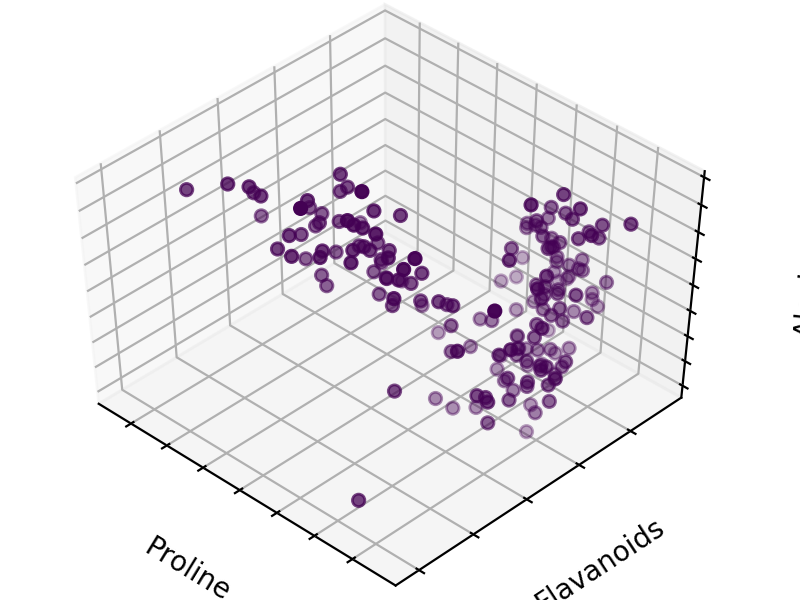
KMeans: Time 13.679ms

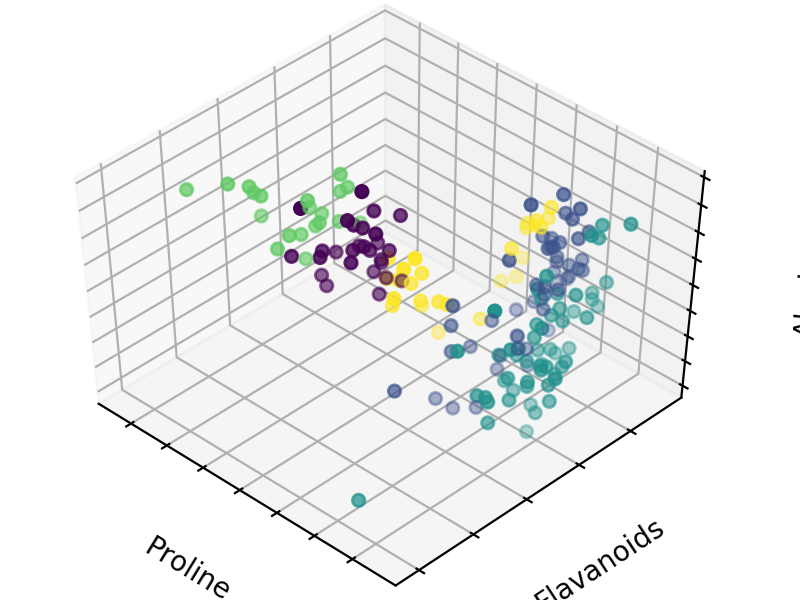
Birch: Time 22.169ms 

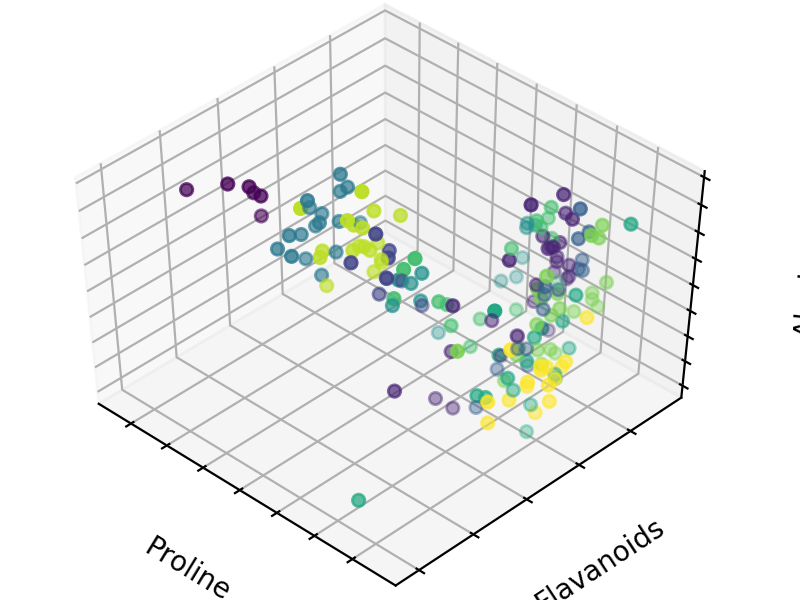
Agglomerative : Time 1.0299ms 

* 1. Differing cluster numbers

KMeans

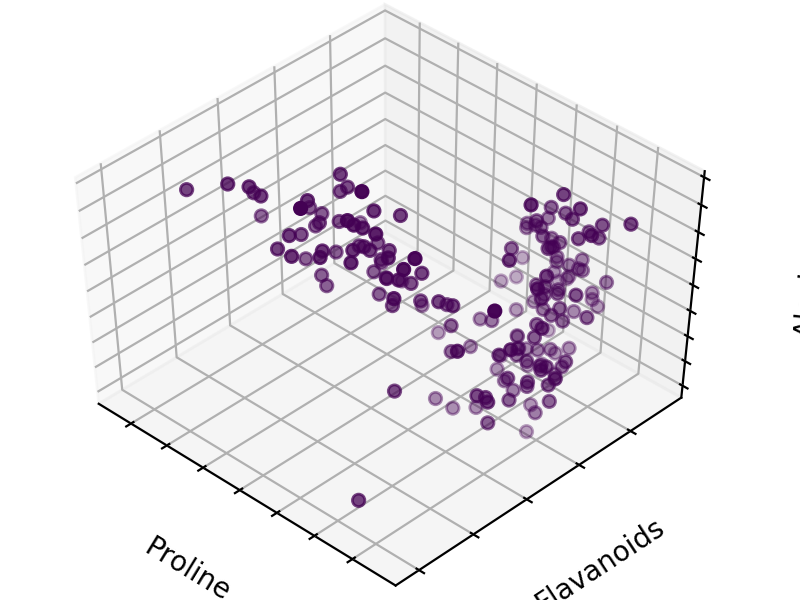
1 cluster: Time 3.54ms 

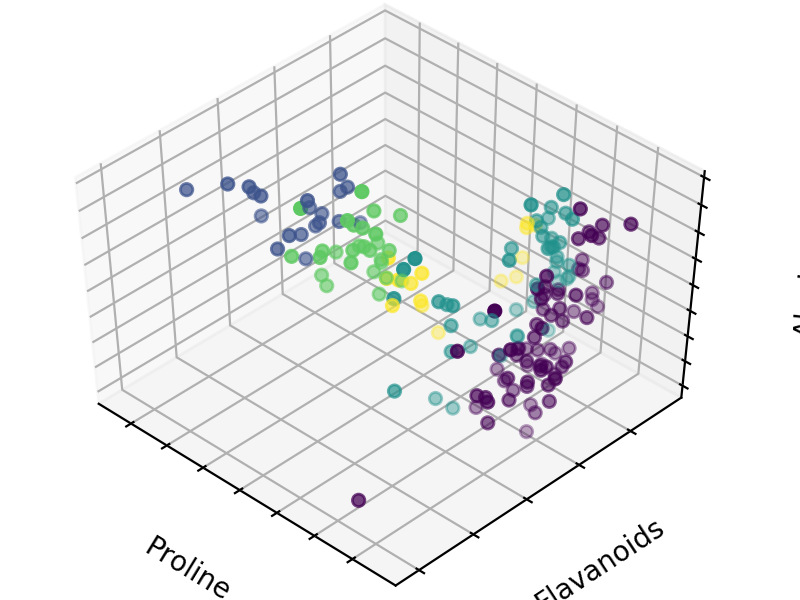
5 cluster: 19.12ms 

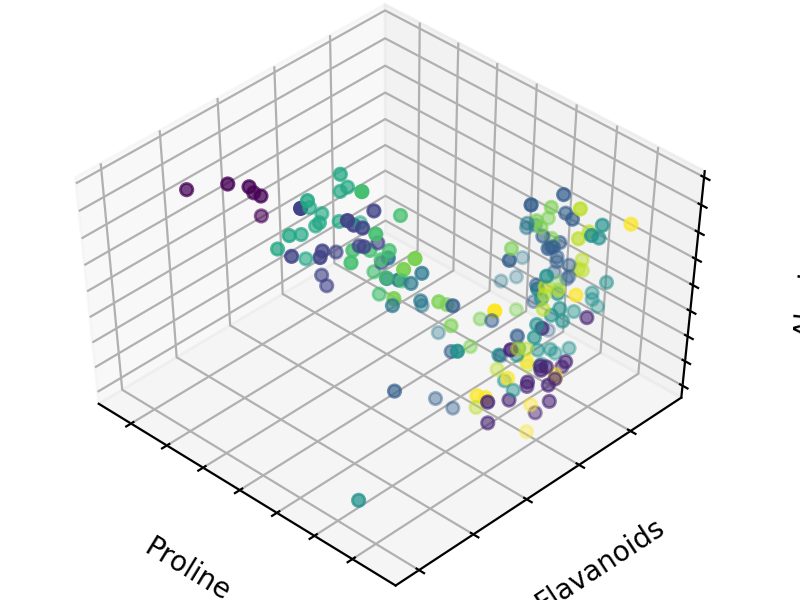
11 cluster: 18.79ms 

Observation: As the number of clusters increased, so too did the processing time (comparing 1 to all others). Also the points become more sporadic with less clearly defined clustering groups

BIRCH

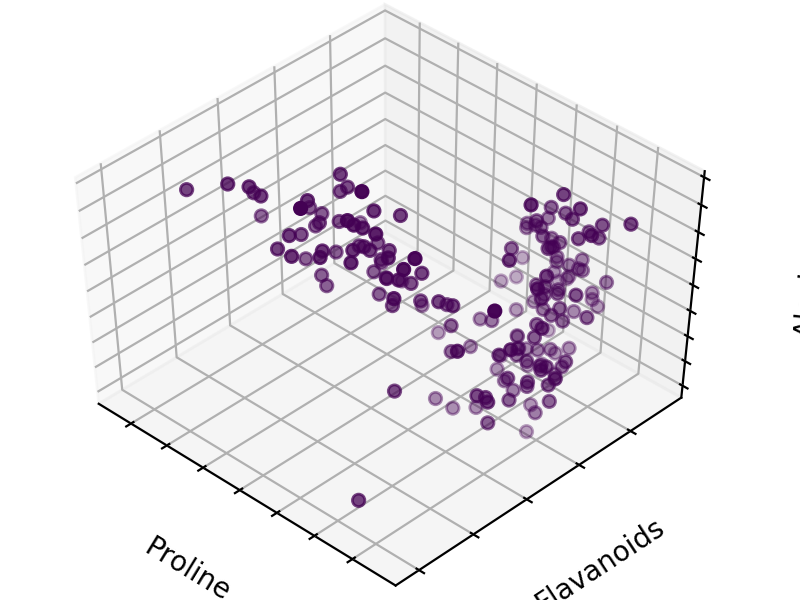
1 cluster: 6.38ms 

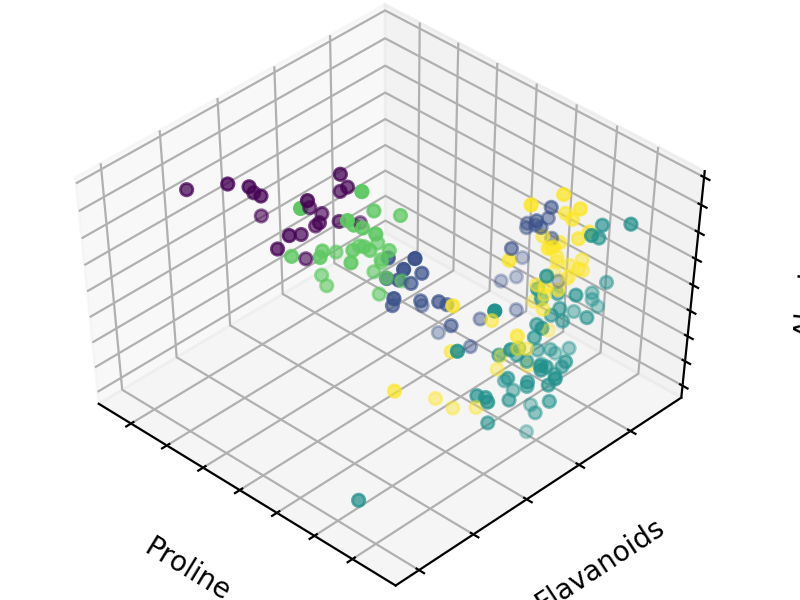
5 clusters: 6.44ms 

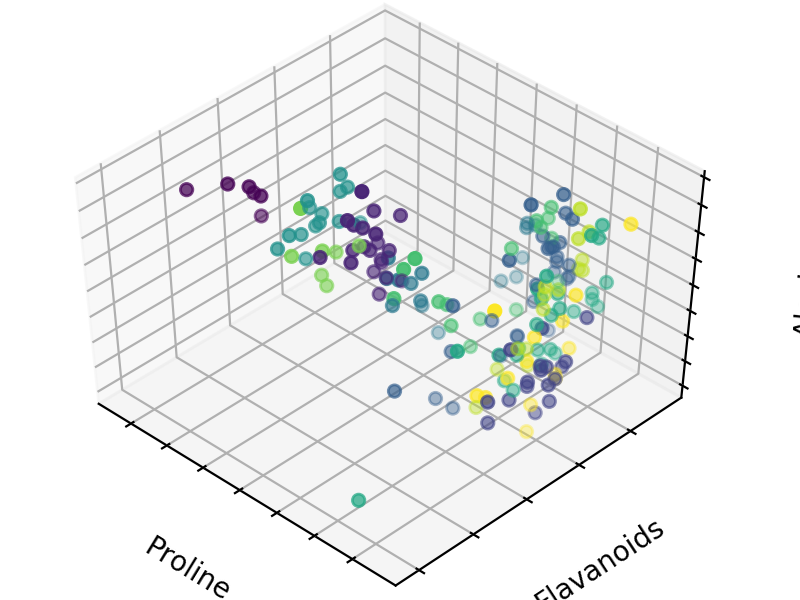
11 clusters: 7.33ms 

Observation: Clustering pattern becomes more scattered at higher clusters, time does increase but at a much slower rate than KMeans

Agglomerative

1 Cluster: 0.99ms 

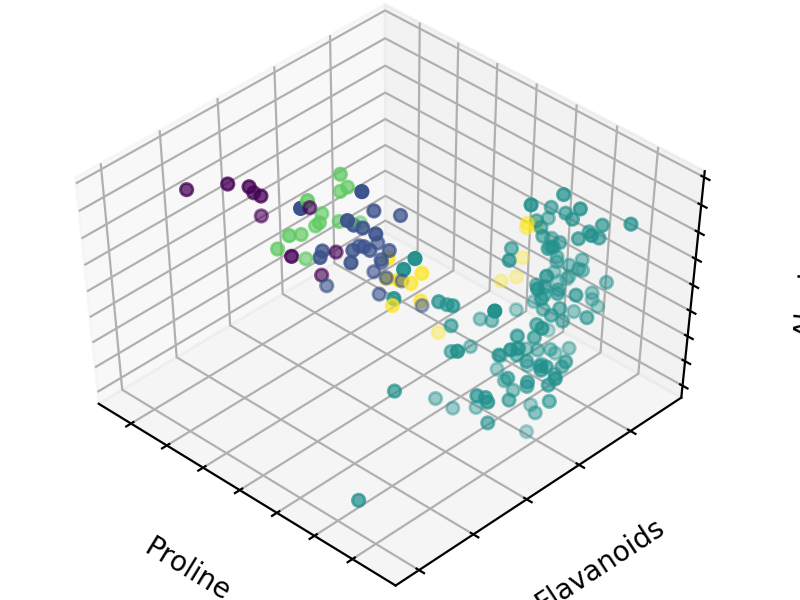
5 clusters: 1.00ms 

11 Clusters:0.95ms 

Observation: Much faster than KMeans and/or BIRCH. Time seems independent from the number of clusters being requested.

Overall Observation: KMeans slower than Birch which is slower than Agglomerative. For this given data set, all of them seem to return relatively similar clusterings.

* 1. DBSCAN time: 1.37ms



* 1. Advantages and disadvantages of each method

|  |  |  |
| --- | --- | --- |
| Method Name | Advantages | Disadvantages |
| KMeans | Simple  Can find clusters decently well even with large cluster number | Cannot handle different size / densities of data  Cannot handle outliers  Could be a local optimum  Need to specify the number of clusters |
| BIRCH | Only requires a single scan of database (can fit in memory)  Can produce outliers  Minimizes I/O | Can be slow as it requires n2logn complexity, where n is the # of points  Can’t handle non-convex shapes  Can’t handle clusters of different sizes |
| Agglomerative | - Fast  Easy to implement | Need to specify number of clusters  Will not scale well as the number of points increases |
| DBSCAN | - Doesn’t require a single threshold (can find hierarchical data / data in different shapes)  - Can detect outliers  - Doesn’t need number of clusters  - Can speed up by selecting good eps, but not required for it to work | - Not deterministic as border points can change clusters based on order  - With different parameters input, can get very different results |