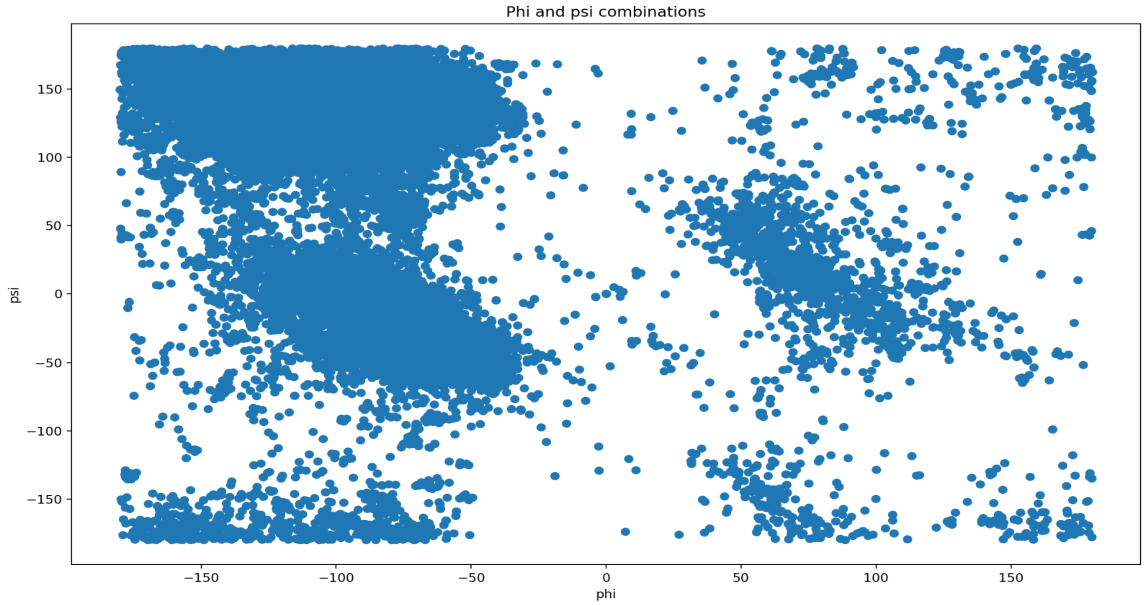
Lab 3 Clustering

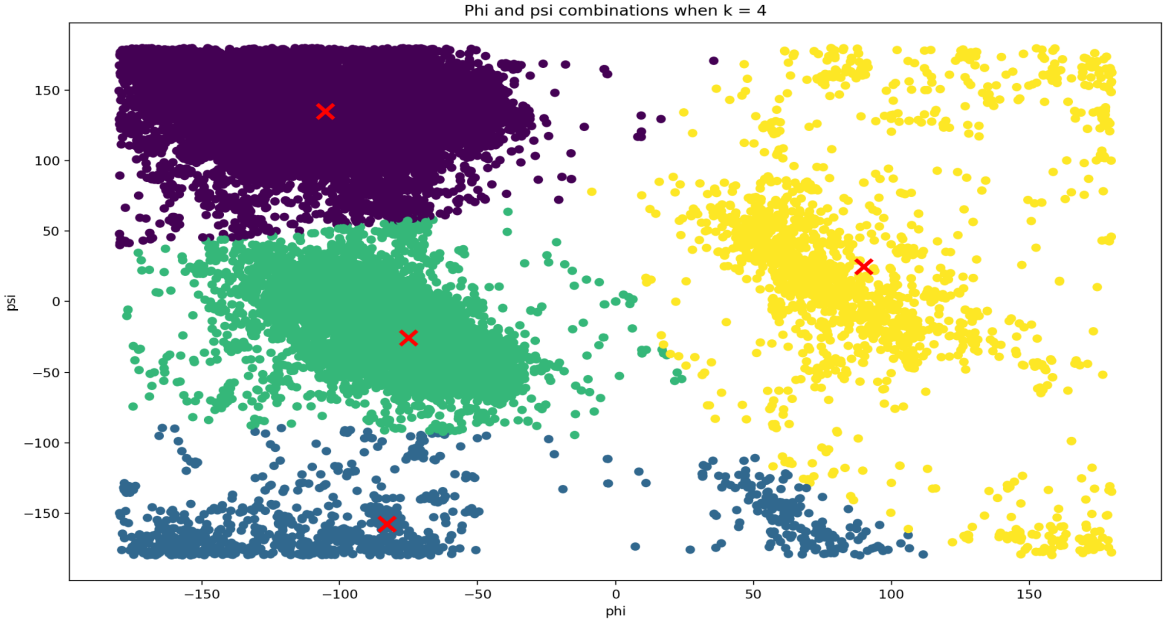
By: Malte Carlstedt & Johan Östling 17h each.

1. Scatter plot of Phi & Psi combinations:

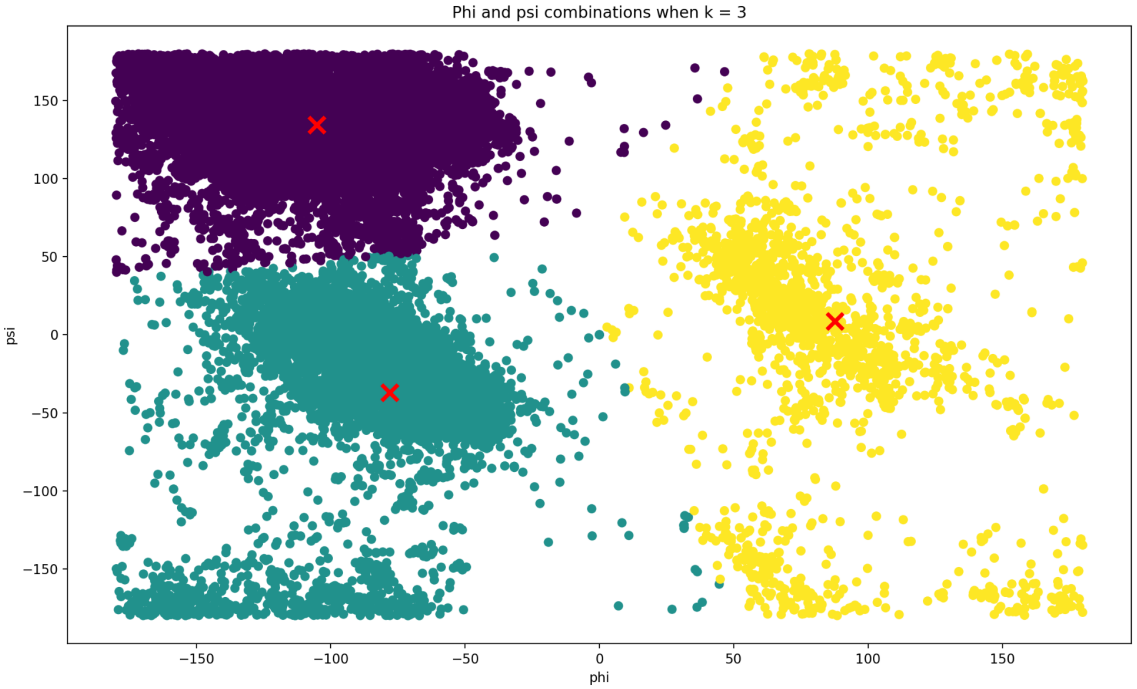


2 A).

From looking at the scatter plot we can see that it naturally forms 4 clusters. Therefore our first estimated value of K, should be 4. With **k=4** we get this plot:

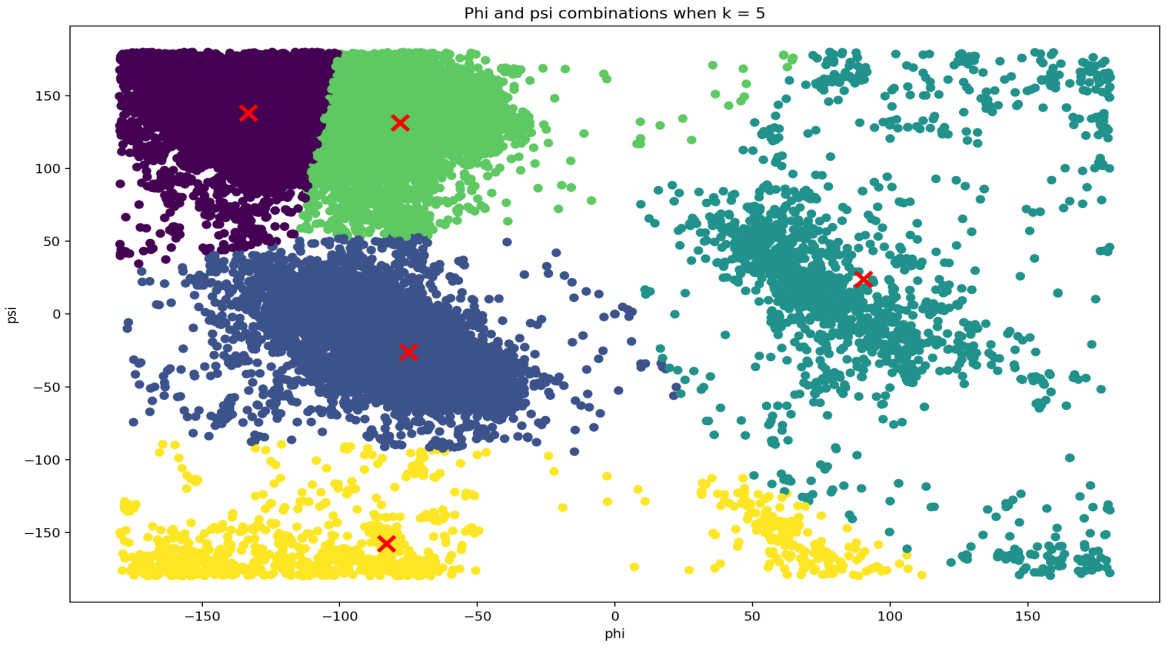


However, the clusters do not divide quite like we wanted to since some part of the right most cluster is also divided into the blue cluster. If we experiment further with other values of K we get this plot for **k=3:**



With k =3 the rightmost cluster is divided into one big spread out cluster. The two clusters to the left are divided into two. Our first implication as to why the two clusters to the left are divided as such are that we believe that most of the data points are in the left half of our scatterplot and therefore are divided into two. As you can see in the top left the individual data points can't even be seen since there are so many.

Experimenting further we also tried **k = 5** we got the following plot. This again divides our left most half into two and also divides our top left clusters into two because of the huge concentration of data points here.



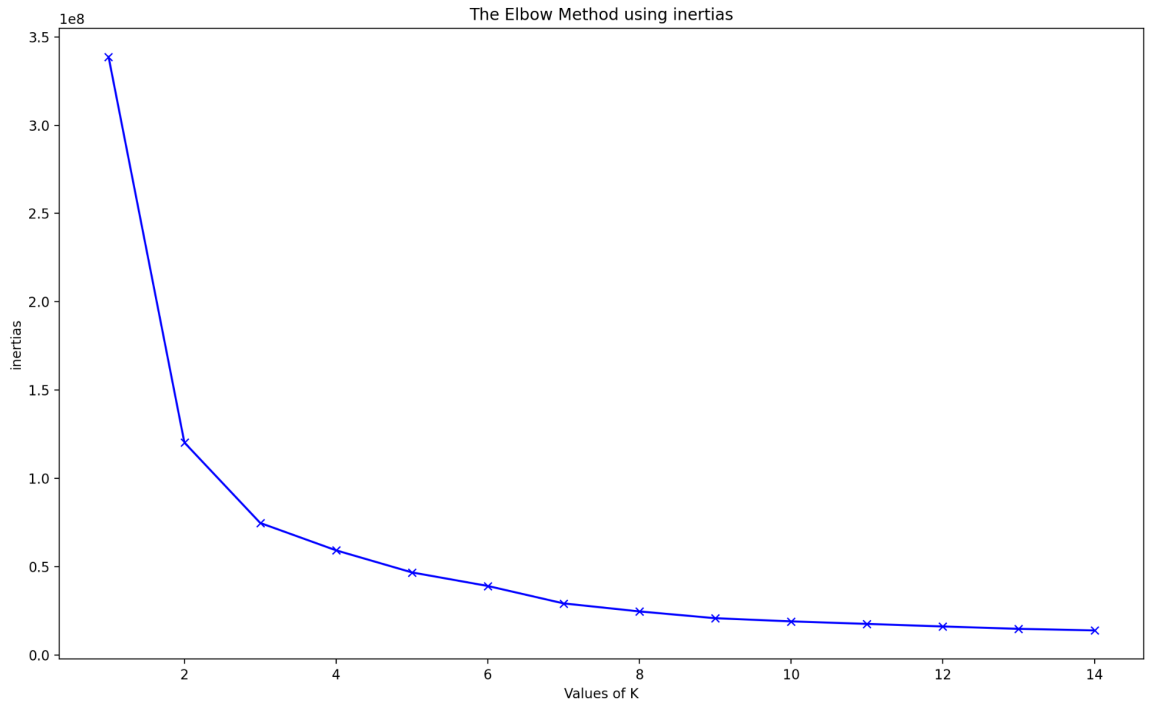
From our visual inspection of the plot we suggest that k=3 is the most appropriate value.

2 C).

Using the elbow method to find optimal value for k

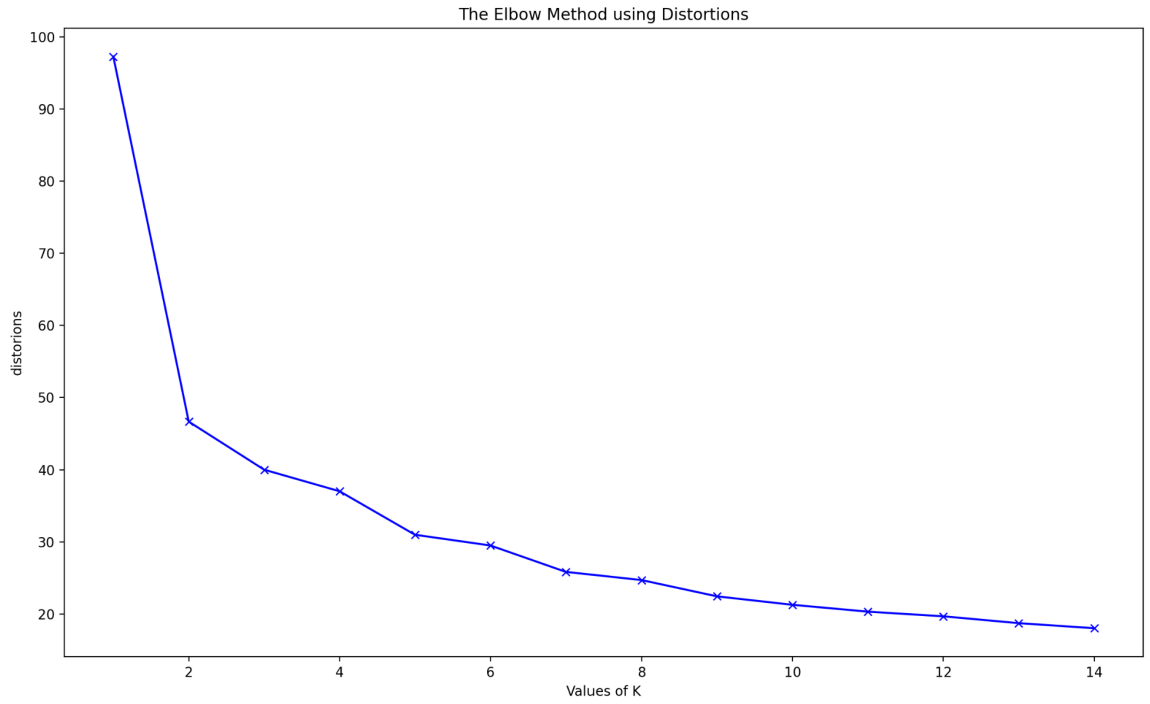
We will use the elbow method to validate this. The elbow method is a way to calculate the optimal k number of clusters. This is done by varying the number of k clusters and then squaring every data point distance to it's cluster center. Hence the drop of when k grows larger, since this means fewer distances and fewer data points in that cluster. How we calculate the distance between each data entry is also very important. There are two different ways of approaching this. Either with inertia or distortion. Inertia is the sum of squared distances of data points to it's clusters center. Distortion on the other hand is defined as: the average of the euclidean squared distance from the centroid of each cluster.

Below you can see from the plot that the sum of the inertias decreases rapidly after k = 3 and after that stays almost horizontal to the x-axis. From that we conclude that the optimal k value is k = 3.



Using elbow method with distortions

Unfortunately using the elbow method with Distortions instead of inertias we found that it was a bit harder to conclude which k value was the clear “elbow” part. That is, where the plot starts to rapidly decrease. It seems that with using distortions the optimal k value is somewhere in between k = 3 to k = 6. We believe that this is because of the way distortions with euclidean distances work and how our data points are located. Since some of our clusters are quite large and concentrated and some clusters have fewer data points and are quite spread out.



Finding k number of clusters using a silhouette score

To then be able to confirm what K number of clusters is the optimal one we calculate a silhouette score for different k numbers of clusters. A silhouette score is a way to measure how good your clustering is. It takes the average distance between clusters minus the average distance between points in a cluster. Then you divide that difference with the maximum value of these to distances. The usual notation for these are that the average distance within a cluster is *a*, and the average distance from each cluster is *b*. The formula then becomes *b-a/max(b,a)*. And therefore the silhouette score will never be bigger than 1 or smaller than -1.

We got the following silhouette score for different k values:

For k-clusters = 2 The average silhouette score is then : 0.6328209708884562
For k-clusters = 3 The average silhouette score is then : 0.6724895253169637
For k-clusters = 4 The average silhouette score is then : 0.6674392423283723
For k-clusters = 5 The average silhouette score is then : 0.5095212375670435
For k-clusters = 6 The average silhouette score is then : 0.4698172916742672

For k-clusters = 7 The average silhouette score is then : 0.48149562265196355
For k-clusters = 8 The average silhouette score is then : 0.487753815552097
For k-clusters = 9 The average silhouette score is then : 0.46636532203390757

From the silhouette score method we can conclude that k=3 is the best in our case.

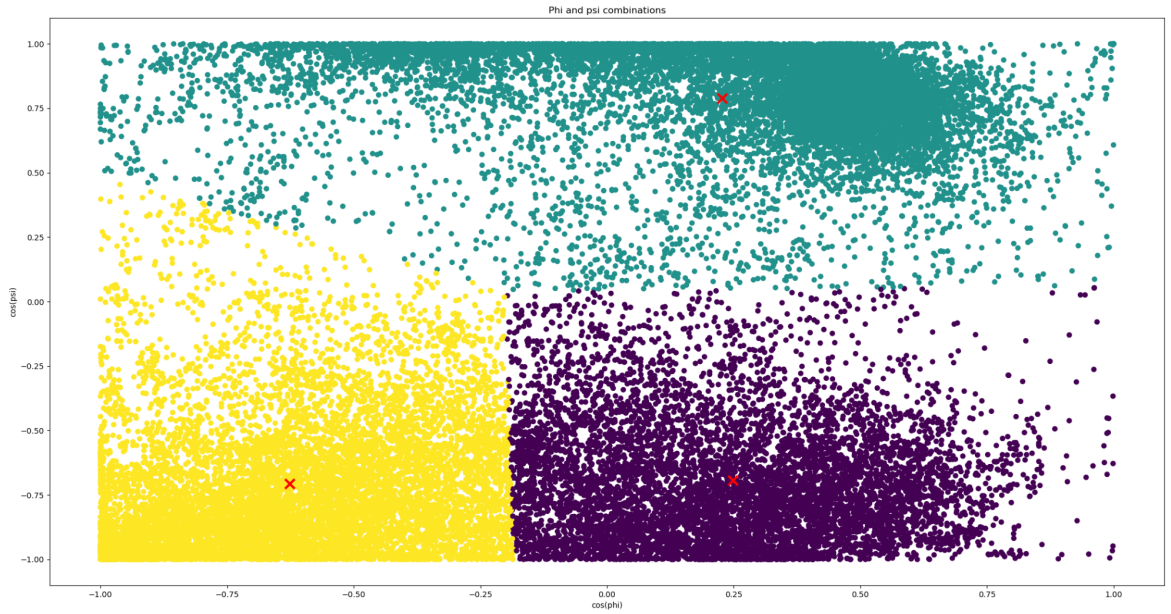
Conclusion

While our elbow method using distortion was undecided, our elbow method using inertia and our silhouette score both suggest k=3 being the best number of clusters. We came to the conclusion that our initial suggestion of k = 3 is therefore the most optimal value for k and it seems fairly reasonable. From an optical inspection of the graph when k = 3. The big yellow cluster is not as clear since the points in this cluster have a larger spread than those in the other two clusters. Since the k-means clustering method takes all data points into account. The large spread of the yellow cluster is therefore heavily influenced by the large spread of data points in the right half. Therefore we do not think that we can do better with the k-means clustering method.

2 D).

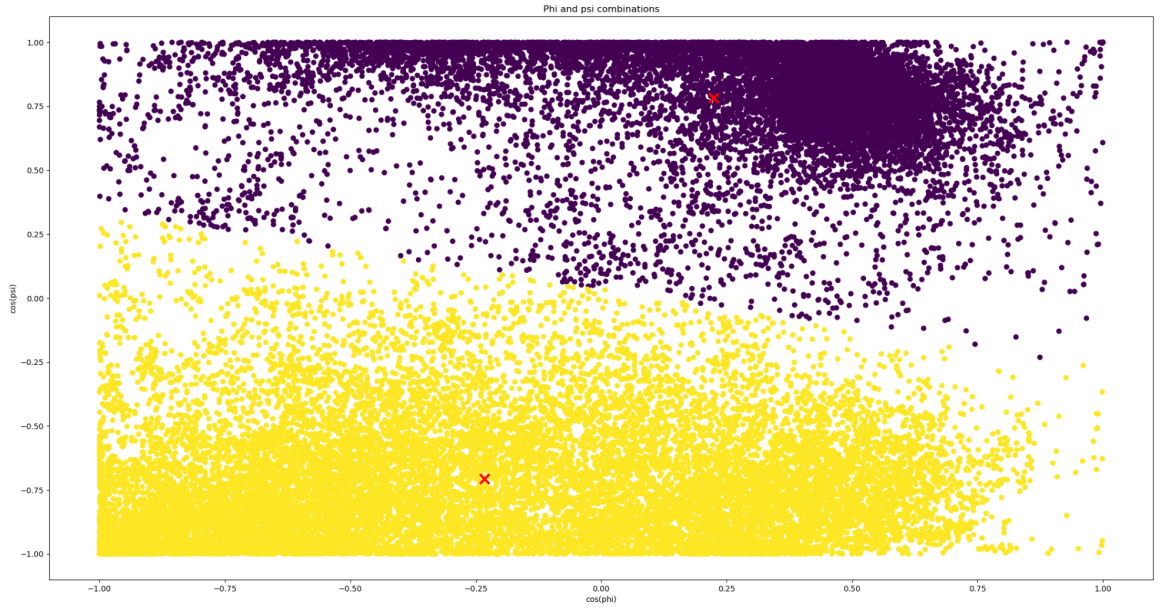
One problem with our plots is that the euclidean distance between points can be wrong due to the data being periodic. The data has periodic attributes since phi and psi are angels. So -180 is the same as 180. So two data points with the same value for phi, but one has -180 for psi, and the other has +180 for psi, is the same point. But when we use the k-means clustering method it thinks that these two points should belong to two different clusters, since they appear to be far apart. How can this problem be solved?

Our solution is to take the cosine function of every data point. Because cos(-180)=cos(180). We got the following scatter plot after applying the cosine function (for k=3):



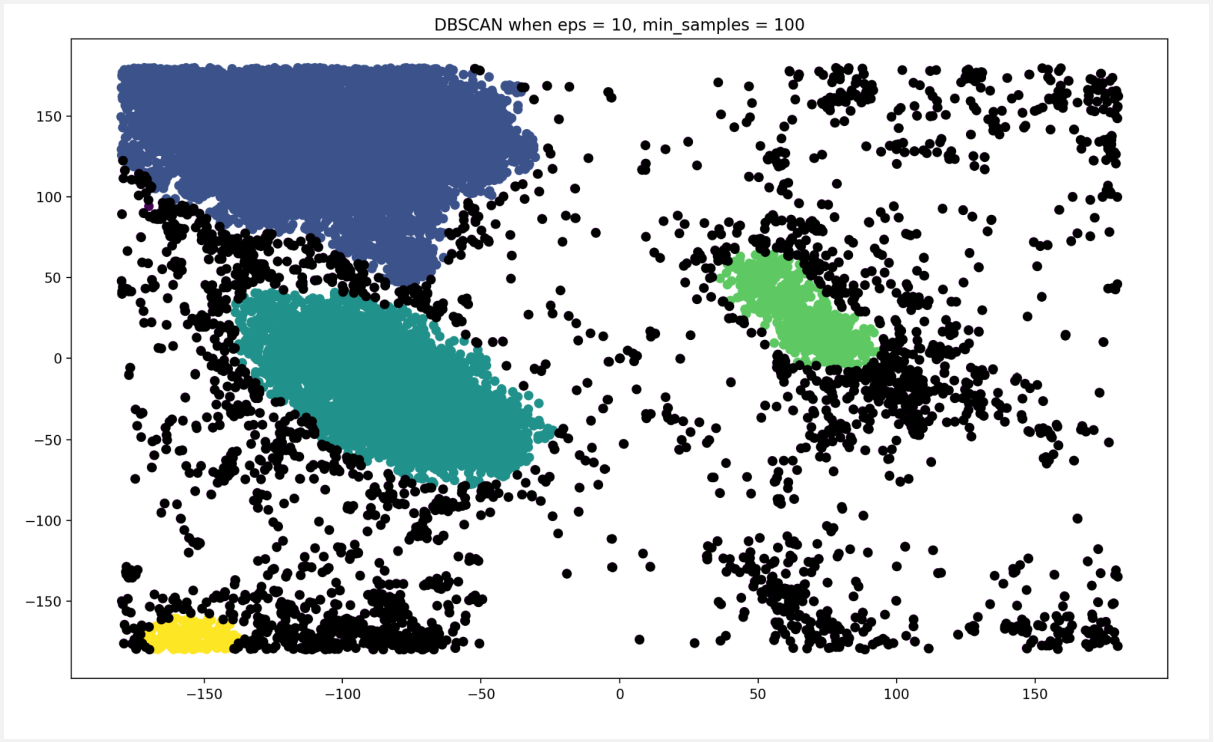
Is k = 3 still the most optimal k for this new plot? We used the silhouette score method for this and got:
For k-clusters = 2 The average silhouette score is then : 0.619739778046773
For k-clusters = 3 The average silhouette score is then : 0.5694175791749613
For k-clusters = 4 The average silhouette score is then : 0.5307806364179385
For k-clusters = 5 The average silhouette score is then : 0.48898000028977345
For k-clusters = 6 The average silhouette score is then : 0.4711689964684355
For k-clusters = 7 The average silhouette score is then : 0.4773523223226649
For k-clusters = 8 The average silhouette score is then : 0.42330064152141444
For k-clusters = 9 The average silhouette score is then : 0.3998469623589122

So after applying the cosine function, k=2 should be the best value for k:
Here is the final plot:

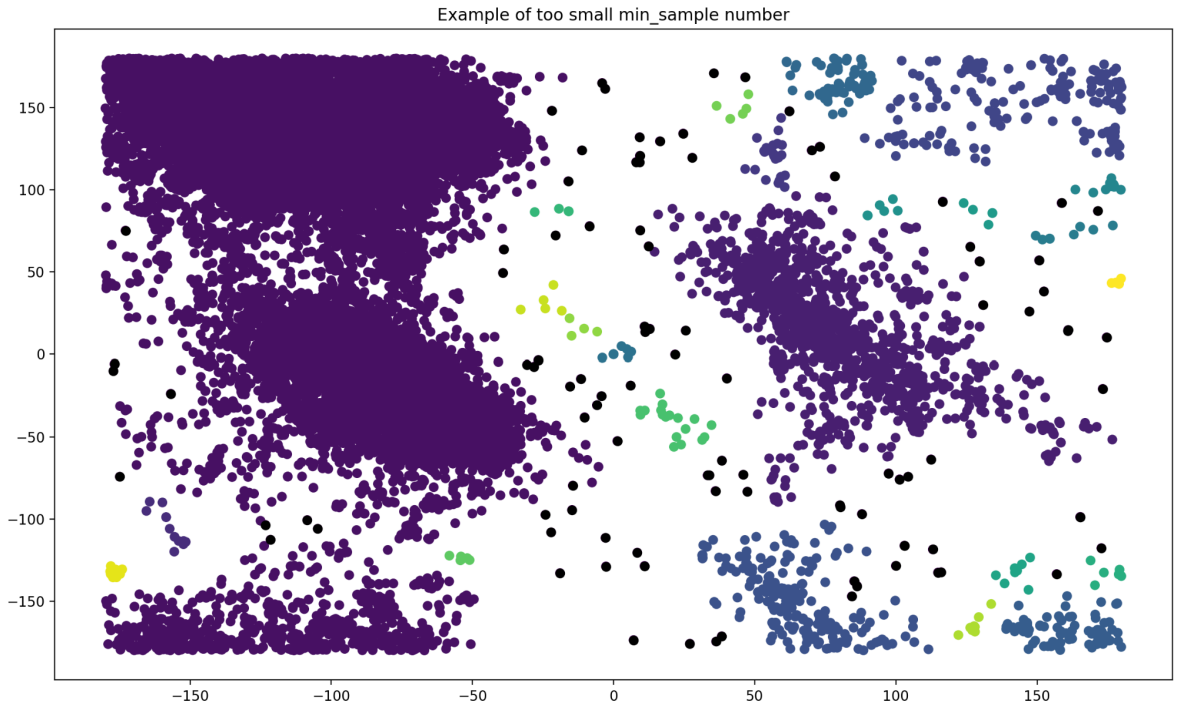


It seems that two clusters are appropriate for this plot. The silhouette score for k=2 was 0.62 which is not very high. This is probably because there is not that good separation between each cluster. One problem with the cosine function method is that the function is not linear. And because the euclidean distance is calculated by the linear distance from two points, and using the euclidean distance after using the cosine function might therefore result in wrongful distances.

3 A)



i + ii. Above you can see one of our first plots with random values for eps and min_sample. The minimum number of samples in the neighborhood is manually set when plotting out our DBSCAN. The number implies how many data points are needed for the data point in question to be considered as a core point. The core point is later on used to grow a cluster. When we chose the minimum number of samples for our DBSCAN we had ultimately boiled it down to two most important factors. The number of minimum samples can not be too few and the number of minimum samples can not be too big. The Swedish word “lagom” (which means not too much, not too little) is perfect in the decision to choose a number for minimum samples. In the case of a minimum number of samples being too small, several small clusters would be able to be created with only a few number of data points in them. If too many small clusters are created the data can become harder to interpret.



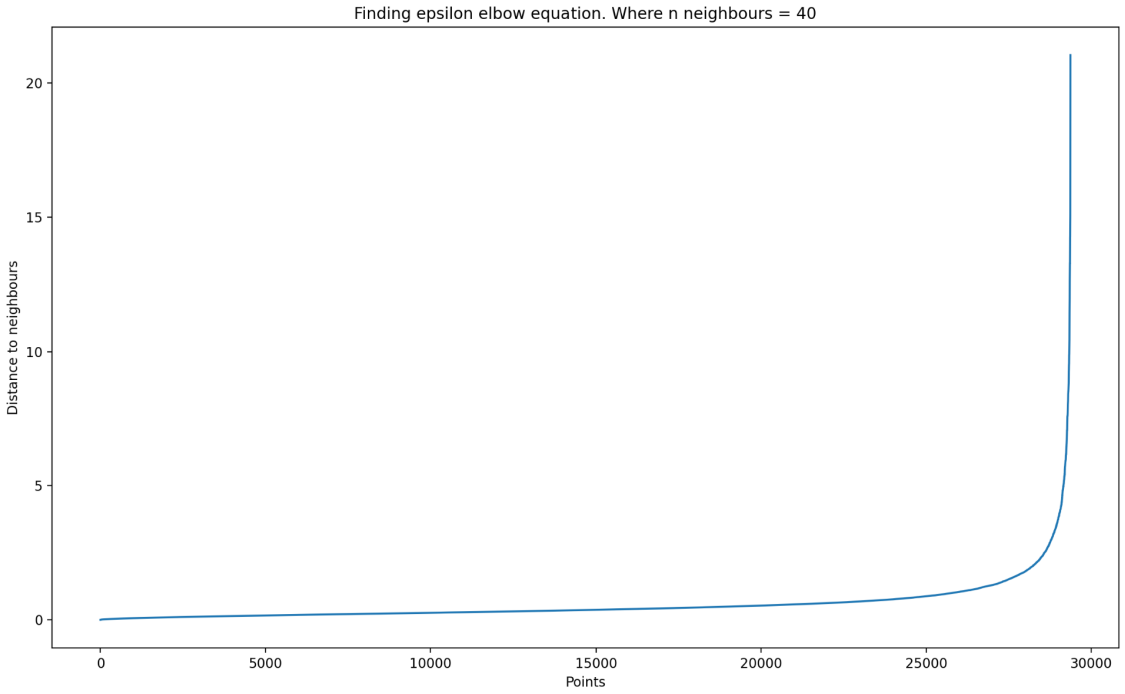
If a too large number for the minimum number of samples is chosen, the clusters created will become large but may miss out on some data points because they were too far away. The size of our data also is to be taken into account. Larger datasets usually relate to larger numbers of minimum points. This is because small clusters are hard to get anything out of when we have thousands of data points.

To find the optimal value for min samples the best way we found was to try some samples until we felt that we found values that would fit our dataset. This was true for both the epsilon value and the min sample value.

To be able to start somewhere we started with an educated guess of the number of minimum samples. By looking at our scatterplot and trying to identify our smallest area that looked like a cluster. By then giving a rough estimate of how many data points the smallest cluster-like area that we could find. And by taking the total number of datapoints in the model (~30.000) into account. We chose to start with 40, as our minimum number of samples and then work upwards from there until we found a value that would fit.

The epsilon value is the value in which the radius from a data point is it's neighbors considered neighbors. If our epsilon value has the value 1.2 and the datapoint that we are currently calculating from has 3 other data points that are within the distance of 1.2 of our original data points. These 3 data points are then considered neighbors. To find an optimal value for epsilon is quite trickier than finding the minimum number of samples. This is because in a densely populated graph we can't tell with our eyes what distance between the data points seems reasonable. To our help we have an elbow method similar to our elbow method used in question 2C. The elbow method for DBSCAN we calculated the average distance between each datapoint. We then plot these results and locate the elbow in the

graph. Or in other words the place where the distance between our data points drastically increases. The number of neighbors we calculate the average distance between for each data point needs to be set manually. Since we are also setting a min number of samples for a cluster, we use the same value for min_samples as for our epsilon equation number of neighbors. Since we chose to start with min_samples set at 40 we do the same for the elbow methods number of neighbors. In the graph below we can estimate that the elbow point is at approximately: 3.



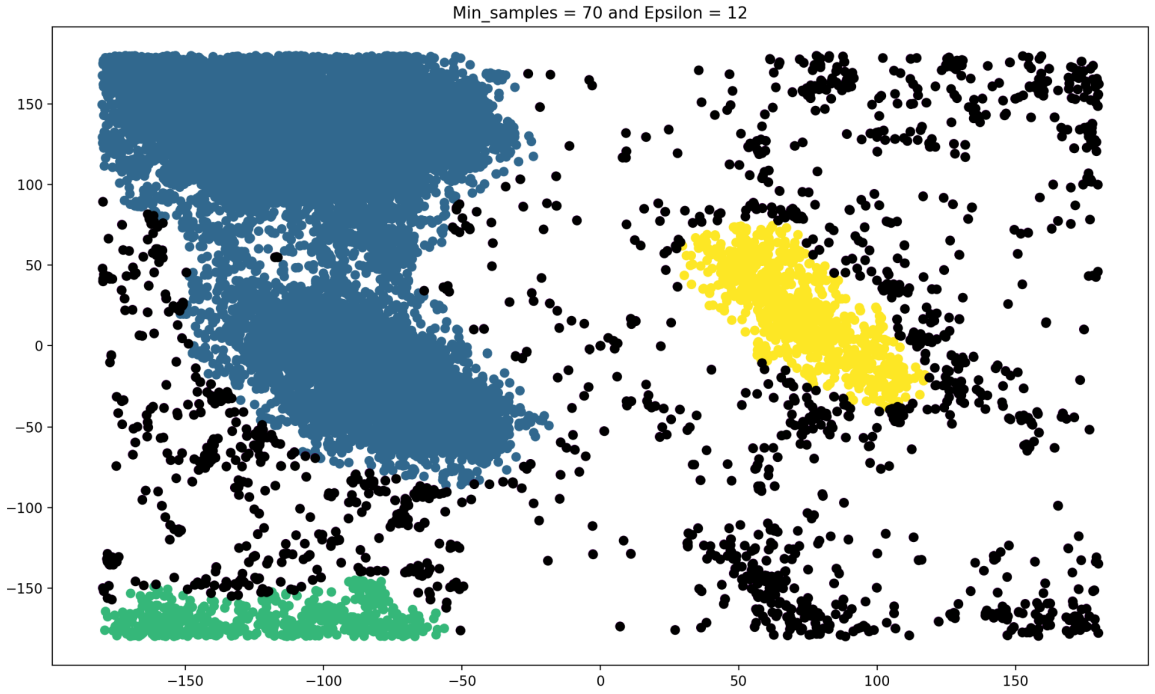
However, the graph does not have a drastic change in curvature as we would've hoped. Instead our elbow method seems to have a very blunt angle and then no barley no increase on the x-axis. This gave us a hard time finding a good epsilon through this graph. Nevertheless we used our initial epsilon value of 3 and gradually increased it along with our min_sample value until we found a suiting pair. Below you can find a table for what combination of min_samples and epsilon value gave what number of clusters and how many noise points.

min_samples	**Epsilon value**	**Number of clusters**	**Number of noise points**
40	3	10	9438
45	4.5	7	5591
50	6	7	3819
55	7.5	5	2954

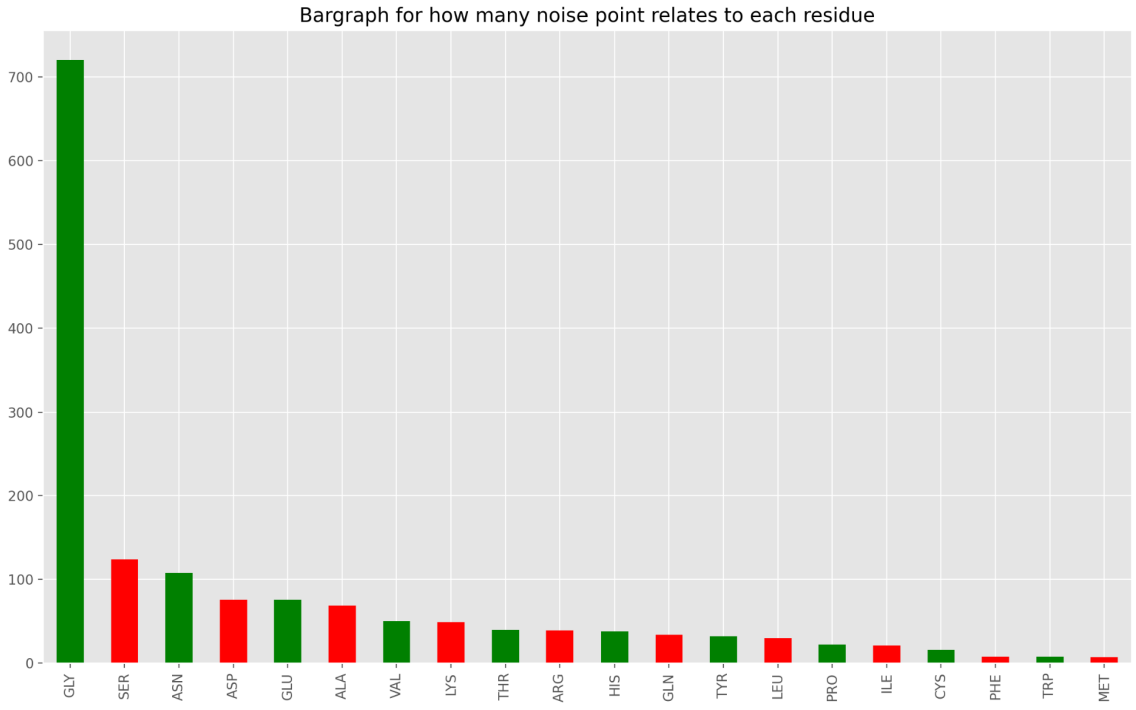
60	9	9	2241
65	10.5	6	1880
70	12	3	1567
75	13.5	3	1434
80	15	3	1276
85	17.5	4	1041
90	19	4	924
95	20.5	4	822
100	22	4	746
105	23.5	4	688
110	25	4	638

The number of noise points kept decreasing with the increasing number for epsilon and number of min points. By then analyzing the table above and the plot given by the values. Our estimate is that the optimal value for min_samples = 70 and epsilon = 12.

3 B) Our graph with outliers in black



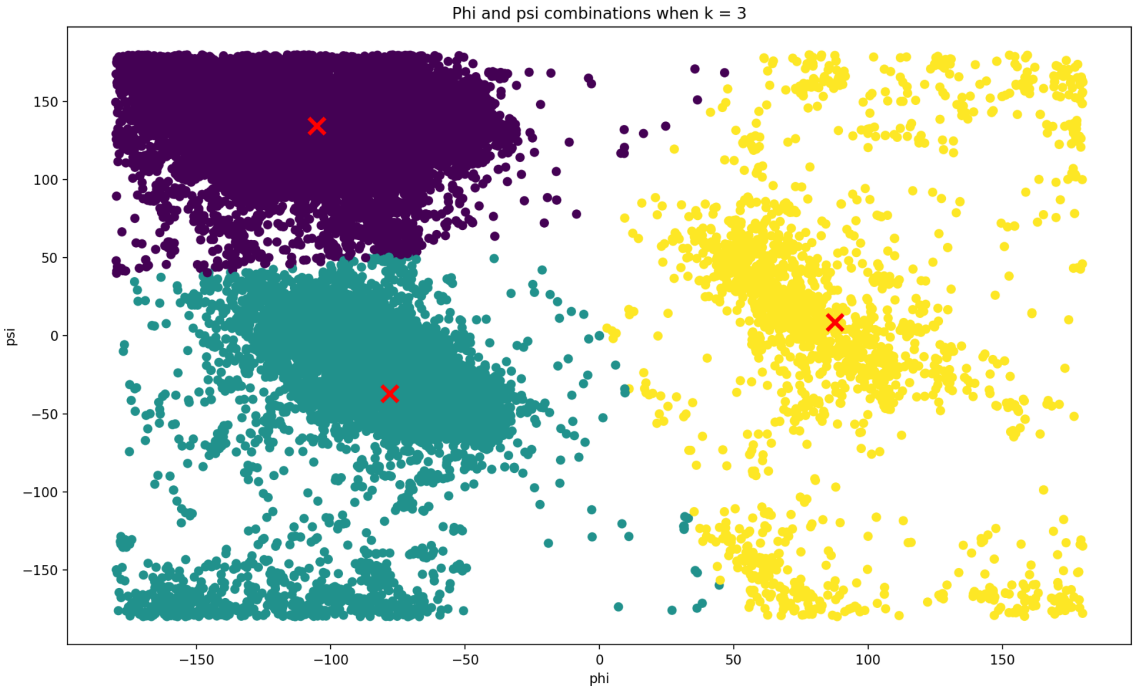
For these values we have 3 clusters and 1567 noise points. These are colored black in the graph above.



Here we can clearly see that GLY is the most frequent outlier.

3 C)

Here was our plot from using k-means clustering:



The first difference between this plot and the one we got using DBSCAN is that this one does not have noise points. This is because k-means clustering takes in all the points the

plot in consideration when clustering. While DBSCAN clusters from a given maximum distance between neighbors and minimum number of points to count as a cluster.

Both methods gave 3 clusters, 2 on the left and 1 on the right. But they are different shapes and sizes. The DBSCAN did not make a cluster out of every point on the right side of the plot, which k-means did. This is because these points are relatively separated and therefore the value chosen for epsilon is too small for DBSCAN to count all these points as a cluster.

On the left the DBSCAN cluster has 2 different clusters, and again for the same reason as why it did not take all points on the left side as one cluster. Because there is not a big seperation between the points in the top left corner.

It is difficult to say which method was most appropriate to cluster correctly because that depends on how you want to cluster. But with DBSCAN it is easier to make the computer cluster the way we want - clustering like a human.

3 D)

We found that DBSCAN were not robust to small changes in either epsilon or minpts. As we wrote in 3 A) if we chose either epsilon or minpts to be too small or too big the clusters formed changed drastically.

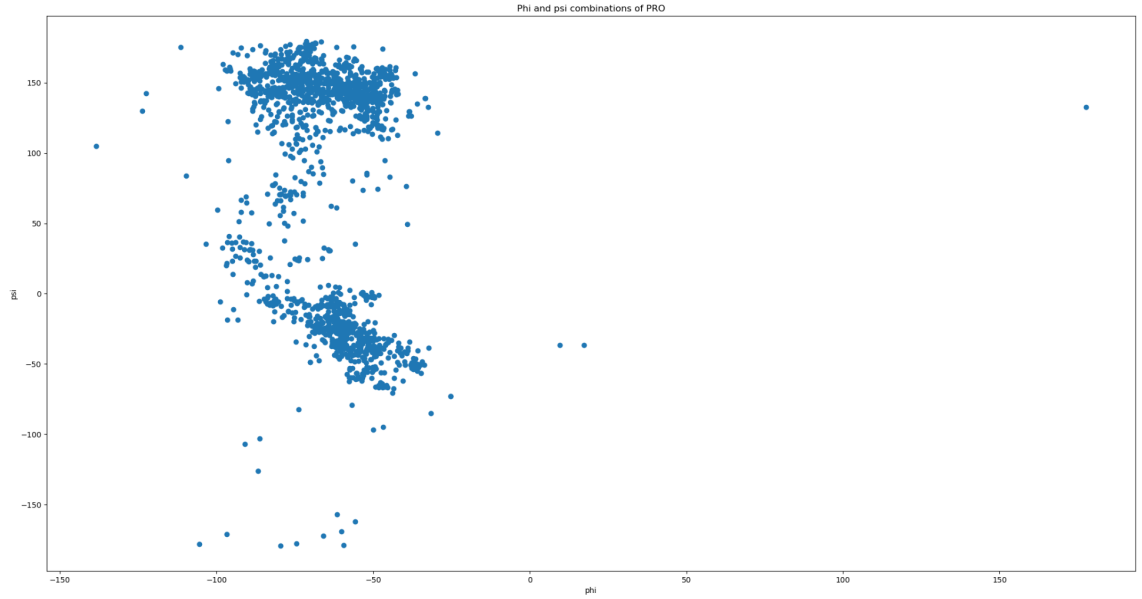
The reason why is because minpts means the minimum number of points close to another point in order for it to count as a core point. And if we chose this number to be too big then the number of noise points increased because the more points in the edges of the clusters no longer counts as core points. And if we choose minpts to be too small then more points accounts for a cluster then we want. For example if we increase our chosen value for minpts from 70 to 80 the number of noise points increased from 1567 to 1667, and if we decrease minpts from 70 to 60 the noise points decreased from 1567 to 1491.

And the reason for why the clustering result by DBSCAN is sensitive for epsilon is because this value decides what distance should be allowed from a point to another point for them to count as neighbors. So if this value gets too big then it may form clusters where we can see that the points should not be in the same cluster. And if epsilon gets too small then we may form too small clusters and we get too many noise points. If we decrease epsilon from 12 to 10 the number of noise points increases from 1567 to 2084 and we get 7 formed clusters that we do not want to form. If we instead increase epsilon from 12 to 14 we get that noise points decrease from 1567 to 1332 and it forms 3 clusters that are larger than we wanted.

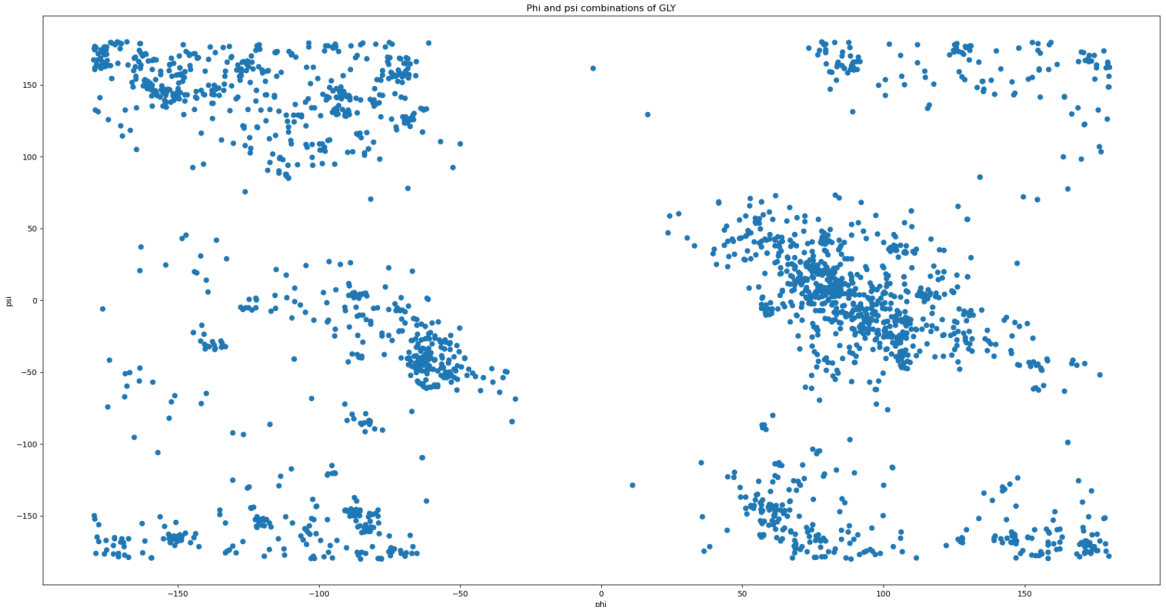
Also our data points have values from -180 to +180 and therefore they are not far away from each other. This increases the sensibility for the values of minpts and epsilon. If the data was spread from -1000 to +1000 then the DBSCAN would be more robust to small changes of these values.

4.

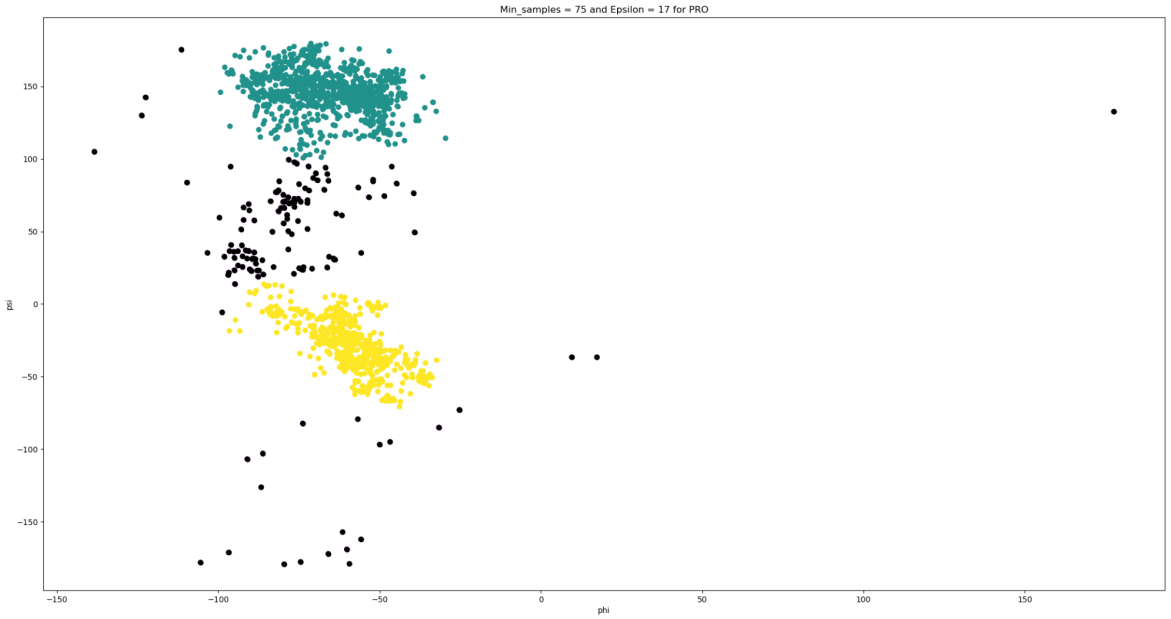
This is the scatter plot for PRO:



And this is for GLY:



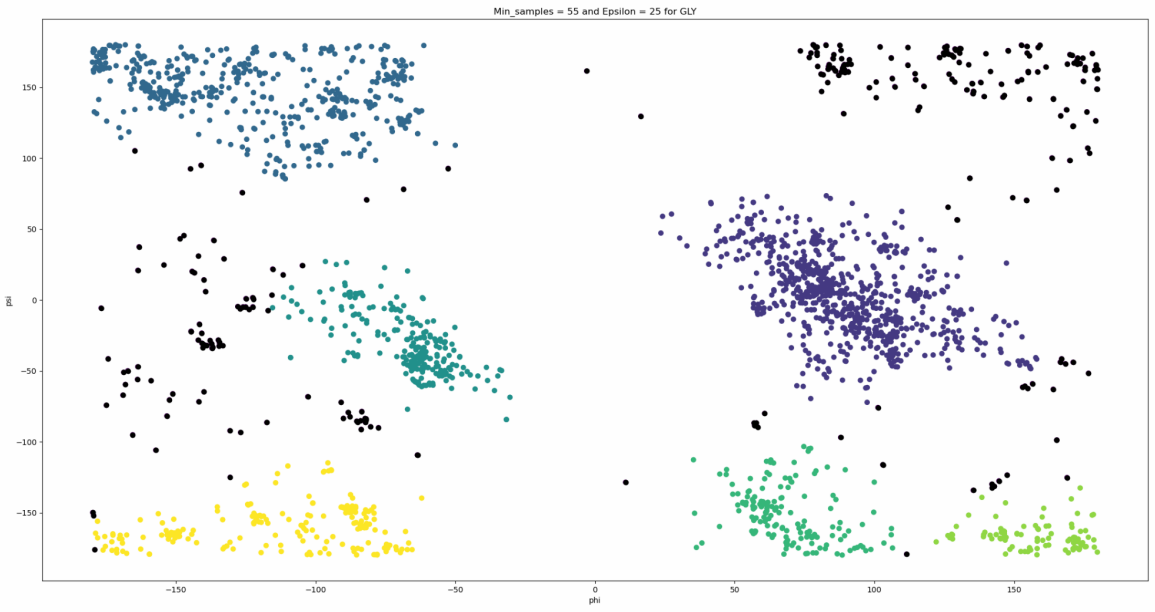
PRO with DBSCAN



This gave us **2** clusters and **139** noise points.

These two clusters seem reasonable since in the first scatter plot these two areas were more dense than the surroundings. If we compare it to the general clusters that all have aminoacids we can see that basically no PRO points were on the right side of the plot. And maybe because all of the PRO points are on the left side was therefore a reason for the general plot being more dense at its left side. From this plot we can see a distinct rotation from the alpha carbon that forms PRO. The difference from the general plot is that PRO's spread for the phi value goes from -180 to 0, while the spread of the value for phi for the general plot goes from -180 to +180.

GLY with DBSCAN



This gave us **6** clusters and **267** noise points.

From the picture above we have only plotted samples with residue = GLY. If we compare the above picture with the general clustering created in assignment 3B. We can clearly see that the GLY plot has several similarities with the general plot. The GLY residue seems to appear all over our plot in contrast to the PRO residue which seems to only appear in some places of our plot. Reason for this might be that the GLY residue has very varying values and therefore appears all over our plot.

Reason behind why our GLY plot has so many more clusters than our general plot is both because we had to increase the epsilon value for the GLY plot because of the density of the plot. The GLY plot has a lot more data points than the PRO plot but the GLY residues are more spread apart. Therefore we used different values for min_samples and epsilon which is one of the reasons why it differs from the general plot.