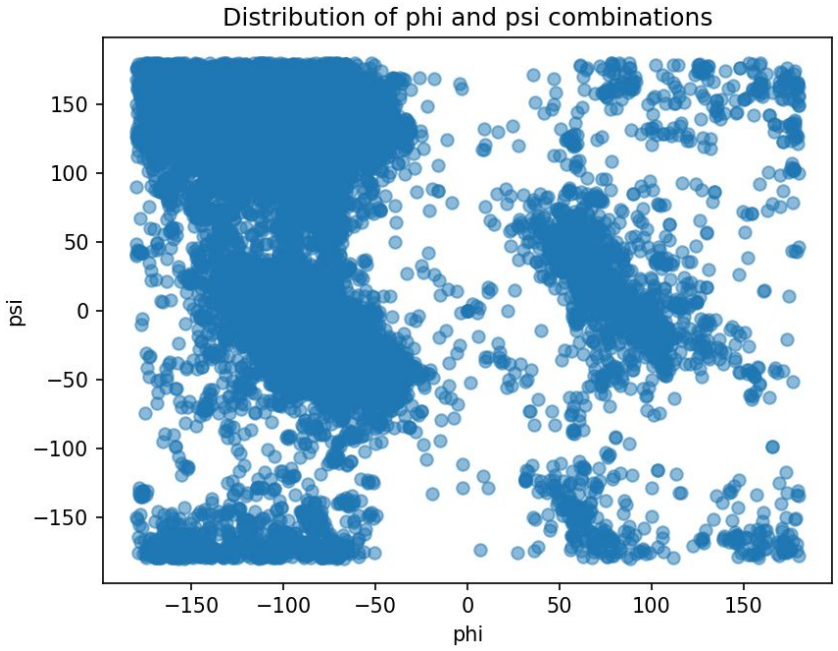
Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

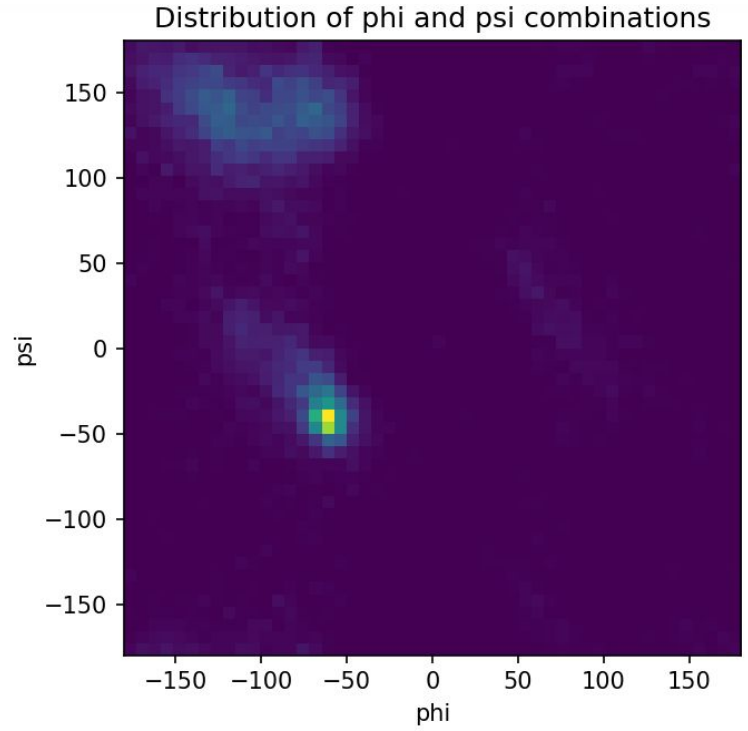
DAT 405 – Lab 3

1a)



Scatter plot of phi and psi combinations.

1b)



Heatmap of phi and psi combinations, lighter color indicating higher density.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

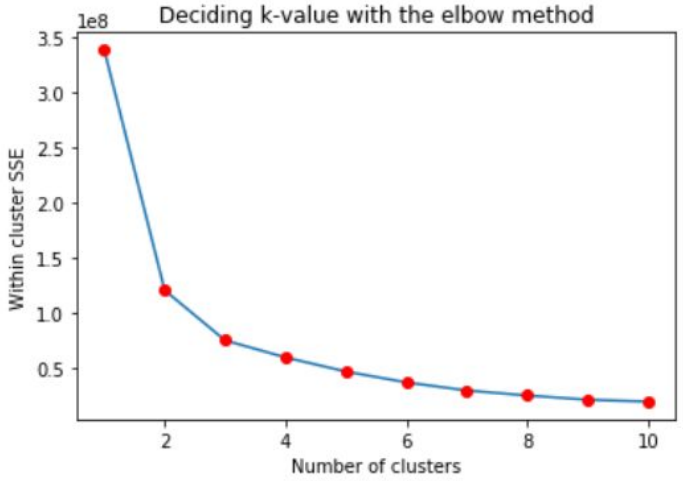
```
# Task 1b - heatmap
heatmap, xedges, yedges = np.histogram2d(xValues, yValues, bins=50)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

plt.clf()
plt.imshow(heatmap.T, extent=extent, origin='lower')
plt.rcParams['figure.figsize'] = [200, 200]
plt.title('Distribution of phi and psi combinations')
plt.xlabel('phi')
plt.ylabel('psi')
plt.show
```

Code for generating heat map

2a) To find the best possible k-value for K-means clustering, we used both the elbow method and the silhouette method.

```
# function returns WSS score for k values from 1 to kmax
def calc_WSS(data,kmax):
    sse = []
    k_values = list(range(1,kmax+1))
    for k in range(1, kmax+1):
        kmeans = KMeans(n_clusters = k).fit(data)
        centroids = kmeans.cluster_centers_
        pred_clusters = kmeans.predict(data)
        curr_sse = 0

    #calculate square of Euclidean distance of each point from its cluster center and add to current WSS
        for i in range(len(data)):
            curr_center = centroids[pred_clusters[i]]
            curr_sse += (data[i, 0] - curr_center[0]) ** 2 + (data[i, 1] - curr_center[1]) ** 2

        sse.append(curr_sse)
    return sse, k_values
```

The elbow method, that can be seen in screenshot above, is a method that runs the k-means clustering for a range of different values for k, and then calculates an average score for the clusters.



By then plotting the graph for the values (y) and number of clusters (x), the optimal number of clusters should be where the 'elbow' of the graph is (where the slope of the graph changes more drastically), which in our case above seems to be for 2 clusters, but could also be for 3 clusters.

```
#func calculating optimal k-value with the silhouette method
def calc_silhouette(data,kmax):
    sil = []
    for k in range(2, kmax+1):
        kmeans = KMeans(n_clusters = k).fit(data)
        labels = kmeans.labels_
        sil.append(silhouette_score(data, labels, metric = 'euclidean'))

    return sil.index(max(sil))+2, max(sil)
```

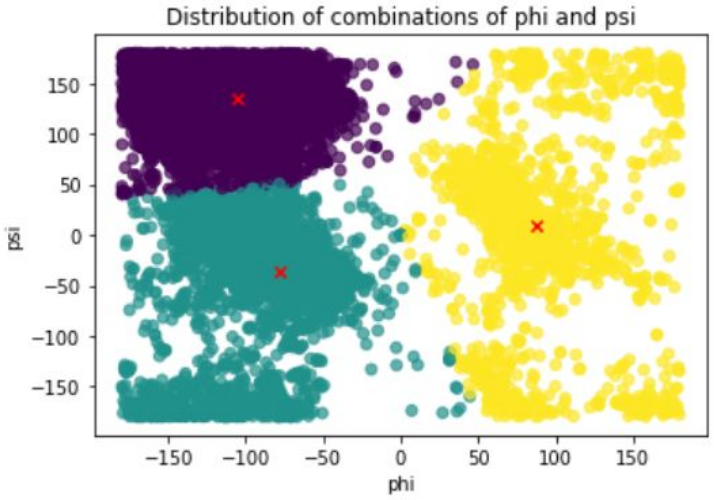
Above is a screenshot of the code for the silhouette method, a method which evaluates how similar each data point is to its own cluster, and with this information it suggests an optimal number of clusters for the data set. Running this code for the phi-psi-dataset gave that the optimal number of clusters are 3. This matches with 3 being a good fit according to the elbow method, although 2 clusters seemed more as the optimal fit according to the elbow method. But, keeping in mind that the elbow method is more of a decision rule/guideline and by just looking at the initial scatterplot, 2 clusters feel like too few for both the spread of the data set as well as the amount of data points and therefore we decided to go for 3 clusters.

2b/c) For validation, as partly mentioned in 2b, we used the silhouette method, stability of subsets and graphical evaluation. The silhouette score ranges between -1 and 1, where -1 indicates a poor match for the data points to their cluster, and 1 indicates a good match. The silhouette score for our plot was ~0.6725, which we believe indicates that our clusters are good but with some variations in density within the clusters.

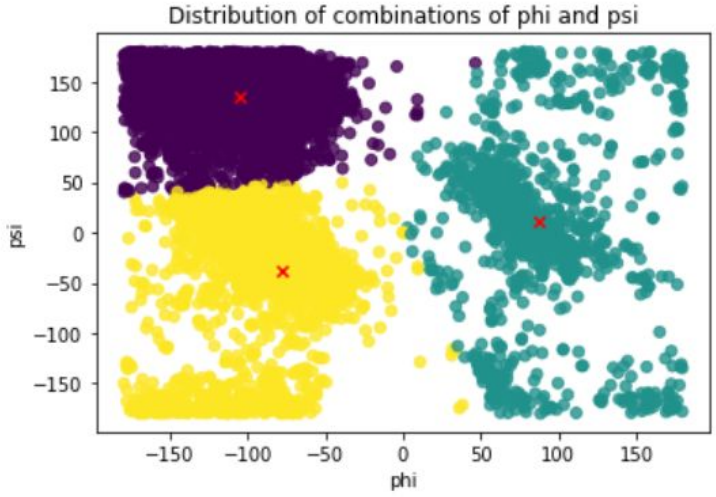
```
#2b - Validation - stability on subsets
np.random.seed(0)

# removing a random 40% av the data
fraction = 0.4
remove_n = int(fraction * int(len(df.index)))
drop_indices = np.random.choice(df.index, remove_n, replace=False)
df_subset = df.drop(drop_indices)

# extracting relevant columns and reshaping the periodic data
phi_psi_subset = df_subset.iloc[:, 3:5].values.reshape(-1,2)

# perform K-Means clustering with same parameters as full sample for comparison
kmeans = KMeans(n_clusters=k_value_1, random_state=0).fit(phi_psi_subset)
```

The stability of subsets method can be seen above, which is a method that plots a k-means clustering with the same parameters as in the full data set version, but only for a subset of the data (60% in this case). By doing this and then visually analyzing if the cluster centers still seem reasonable, and that the silhouette score for the subset is ~ equal to the full data set, we are able to say that this method validates our choice of clusters.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)



Scatter plot with cluster centers for full data set.



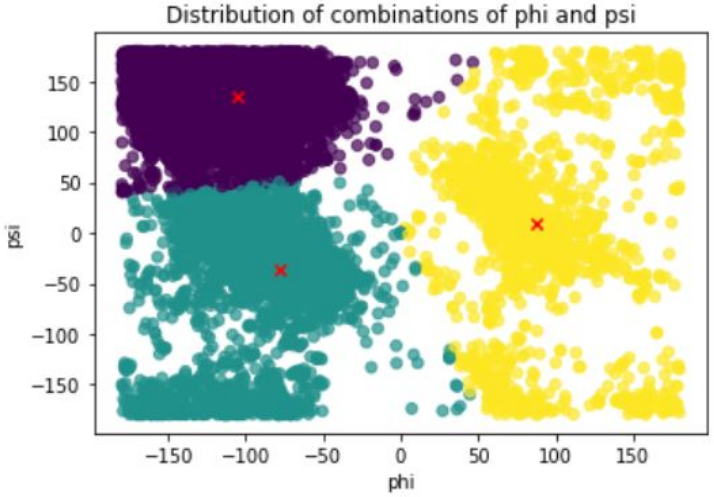
Scatter plot with cluster centers for 60% of the data set.

As can be seen above, the cluster centers still seem reasonable for the 60%-subset. Calculating the silhouette value for the subset, we got a value of ~0.6709, which is very close to the ~0.6725 we got for the full data set. Finally, by just visually looking at the plot, we believe that the 3 clusters represent 3 areas that are more dense than points scattered around them and that the other points surrounding each dense part of the clusters are too widely spread to be clusters of their own.

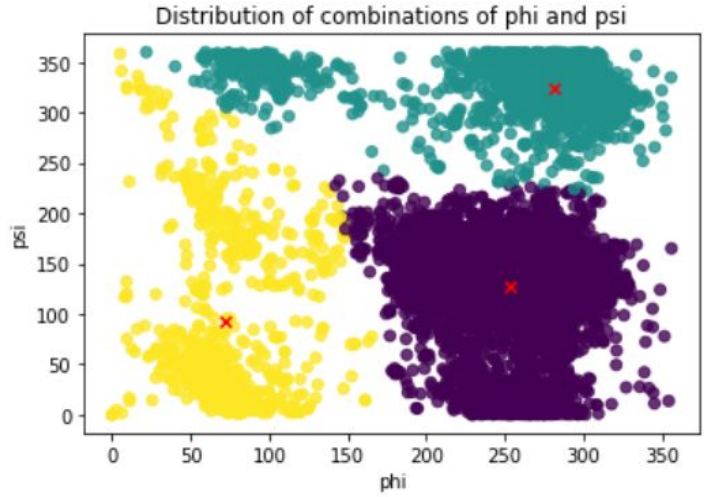
2d) The data given is periodic, which gives that our K-means clustering up until this point has not taken into consideration that i.e. an angle of -170 degrees is close to an angle of ~180 degrees. This ‘unnoticed’ gap could impact the best possible way of clustering the data, and therefore we solved this by converting all angles to its positive resemblance. To get the positive resemblance of the angles, we simply took the array of phi and psi combinations, and used the modulus operator:

```
#translating all angles to its' positive equivalency by using mod func.
phi_psi_pos = phi_psi%360
```

By the code above, all positive angles (angles in the first and second quadrant) keep their value, while the negative angles assume the value of their positive resemblance. By fixing this and then re-plotting the data, we get the second plot below:



Scatter plot of phi and psi combinations with original data.

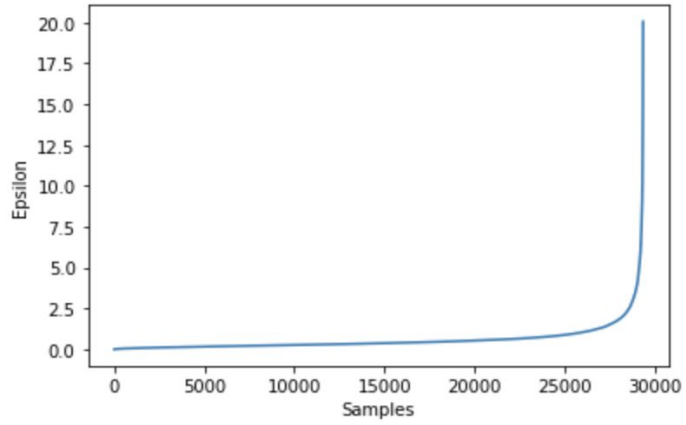


Scatter plot of phi and psi combinations after shifting all angles to positive resemblance.

As can be seen by comparing the graphs above, the shift of the negative angles also shifts what data points belong to which cluster. To get a better idea on whether this data gives better clustering, we compared the silhouette values for before and after. For the original data, we get a silhouette score of ~0.6725, compared to the slightly better silhouette score of ~0.6798, which indicates that our transformation of the periodic data has improved our clustering somewhat.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

3a) The values assigned to the two main parameters deciding the outcome of the DBSCAN, epsilon (the maximum distance between two samples belonging to the same neighbourhood) and min_samples (minimum number of samples in the neighbourhood for a sample to be considered as a core point) proved rather tricky to decide. Regarding epsilon, we decided to calculate the distance two the nearest neighbour for each point, then sorting this distance before plotting it as a graph, Similar to other methods, we then tried to identify the 'elbow' of the graph which according to scientific papers and other sources would yield an optimal value of epsilon. In the case of min_samples, one does best to rely on domain expertise deciding this number. In lack of this, we used the heuristic approach of setting this value to the natural logarithm of the total number of samples. These methods gave us an epsilon between 3 to 5 and a min_samples between 10-11. Since neither of these combinations gave a good result in terms of visibility or silhouette score, we decided to iterate through different epsilons with a fixed min_samples as well as different min_samples with a fixed epsilon and calculating the silhouette score for each relevant combination. We also plotted the DBSCAN clustering to complement the silhouette score, investigating if the result was reasonable. Although proving a rather time consuming method, we ended up with an epsilon of 20 and a min_samples of 40 which both produced a reasonable plot and a good silhouette value.



Graph: Identifying elbow to get epsilon.

```
# calculate optimal value of epsilon with distance to knn
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(phi_psi_positive)
distances, indices = nbrs.kneighbors(phi_psi_positive)

distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.xlabel('Samples')
plt.ylabel('Epsilon')
plt.plot(distances)
```

Code: Deciding initial value of epsilon.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

```
# calculating silhouette score for fixed epsilon and different min_samples
min_samples = [39, 40, 41]
for i in min_samples:
    print("min_samples value is "+ str(i))
    db = DBSCAN(eps=20, min_samples=i).fit(phi_psi_positive)
    core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
    core_samples_mask[db.core_sample_indices_] = True
    labels = db.labels_
    print(set(labels))
    silhouette_avg = silhouette_score(phi_psi_positive, labels)
    print("For min_samples value= "+str(i),
         "The average silhouette score is :", silhouette_avg)
```

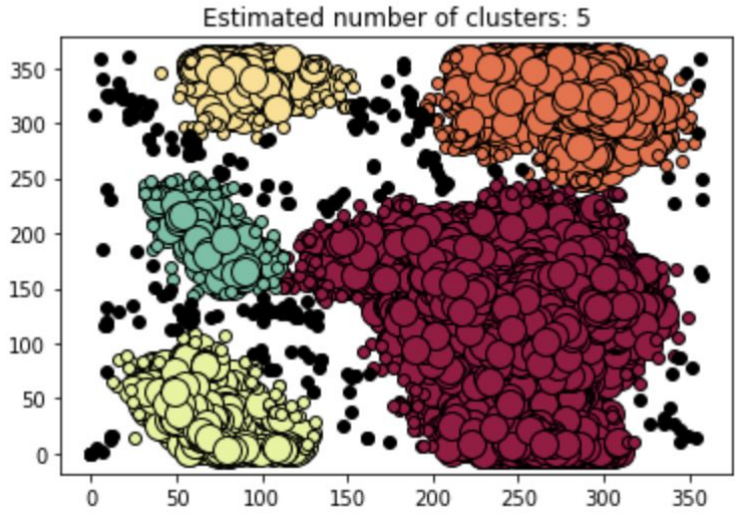
Code: Silhouette score for different min_samples.

```
# calculating silhouette score for different epsilon and fixed min_samples
range_eps = [19, 20, 21, 22]
for i in range_eps:
    print("Epsilon value is "+ str(i))
    db = DBSCAN(eps=i, min_samples=40).fit(phi_psi_positive)
    core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
    core_samples_mask[db.core_sample_indices_] = True
    labels = db.labels_
    print(set(labels))
    silhouette_avg = silhouette_score(phi_psi_positive, labels)
    print("For epsilon value= "+str(i),
         "The silhouette score is :", silhouette_avg)
```

Code: Silhouette score for different epsilon.

3b) Result of DBSCAN and frequency of outliers per amino acid residue type:

```
Estimated number of clusters: 5
Estimated number of noise points: 243
```



Plot: Result of DBSCAN, outliers as black samples.

```
# get dataframe for outliers
df_outlier_values = pd.DataFrame(xy2)
df_temp = df_outlier_values.rename(columns={0:'phi', 1:'psi'})
df_outliers = pd.merge(df_temp, df, how='inner', left_on=['phi', 'psi'], right_on=['phi', 'psi'])

# summing together freq. the different residues
occurence = df_outliers.pivot_table(index=['residue name'], aggfunc='size')

index = np.arange(len(occurence))

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(index, occurence)
plt.xticks(index, occurence.index)
plt.ylabel('Number of outliers')
plt.title('Outliers per amino acid resdue type')
plt.show()
```

Code: Counting the frequency of outliers per residue type and plotting the result.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

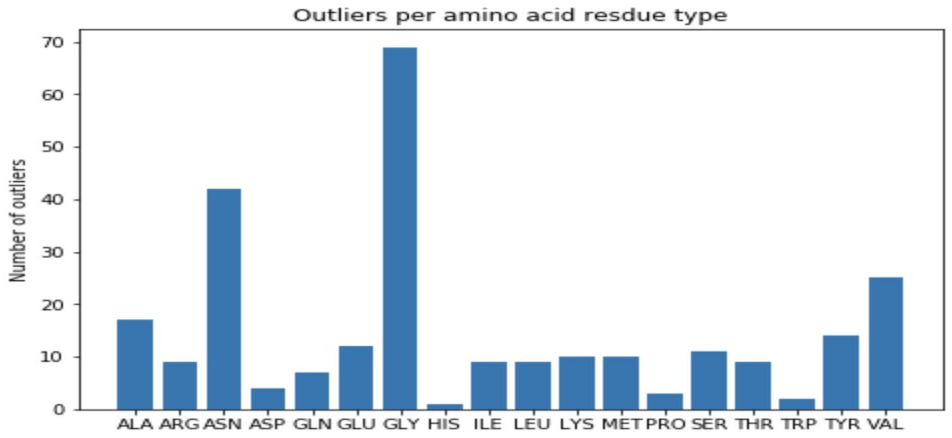


Chart: Number of outliers per amino acid residue type (exclusively those with at least one).

3c) Comparing the clustering from KMeans and DBSCAN there's some obvious differences. For one, the amount of clusters are three for KMeans and five for DBSCAN. The processes used for deciding the number of clusters for KMeans are more proven in comparison to how the equivalent for DBSCAN was decided, something which was more of an advanced trial-and-error approach. Although we did get iterations with three clusters for DBSCAN as well, but weighing in both the visual result and the silhouette score we decided that the epsilon and min_samples values that produced five clusters were the right one to use. Both these clustering methods were validated in the same way using the silhouette score and investigating the stability on subsets. Both produced good results using these methods, which showed a level of robustness to our models. Comparing the silhouette score we see slightly better score for KMeans, although it's very marginal (0.6797 vs. 0.6758) which doesn't imply much in our view. Visually the DBSCAN produces a better result in our view, identifying more and denser clusters that KMeans. For example the leftmost cluster isn't very dense and the upper one seems to consist of two different clusters. Both methods produce reasonable results, but neither is perfect in our view.

3d) As previously described, our method for deciding the values of epsilon and min_samples was an iterative one which allowed us to explore the result of many combinations of different values of these parameters. We could then see that even small changes in either of these parameters could have a significant impact on the result. Both these parameters appeared non-robust to small changes, sometimes less visible seen to the relatively high values we used. For example, changing epsilon to 21 from 20 gave us three clusters and a silhouette score of approximately 0.37, almost dropping to half of its value at epsilon 20. Regarding min_samples changing to directly adjacent numbers such as ± 1 didn't have a significant impact with this given epsilon, but changing to 30 almost halved the silhouette score as well. Moving upwards didn't have the same effect though, with min_samples above 100 producing somewhat similar results with only relatively small changes in silhouette score. Although the number of clusters decreased with higher values and the result lost a lot of quality. The effect of changes in these parameters seems to be interdependent as well, which was illustrated by making small changes to min_samples when epsilon was much smaller. Then a ± 1 change in min_samples changed the result dramatically.

Assignment Group 2
Felix Dunér (25 h)
Sara Hillström (25 h)

4) When stratifying the data with regard to the two amino acid residue types PRO and GLY (this was done separately) we get some interesting results. Here we decided to use KMeans since we've got much more proven methods how to optimize this technique and its parameters. Also, from further research online we found that DBSCAN is not an ideal method to use when your data set has varying densities, which is the case for the phi psi data set. Going through the same process for deciding the value of k as before (identifying elbow, comparing silhouette score) we got two very different plots.
Residue type PRO had a very well defined characteristic with high phi-values. This resulted in the samples creating two rather dense clusters, even though the corresponding psi-values were distributed over the whole spectra. Looking at GLY, we instead see a much larger spread with samples covering most areas of the plot. This led an optimal clustering with five clusters. The lack of one specific characteristic like PRO, led to a much lower silhouette score for the KMeans clustering for amino acid residue types GLY. With approximately 0.60 vs. PRO's 0.81, GLY had a worse silhouette score than the clustering with all the data. We can also see that these two different residue types have somewhat overlapping clusters, which would make it a bit tricky to identify them when clustering the full dataset.