DAT405 Assignment 3 – Group 95

Martin Blom - (4,5 hrs)
Jakob Windt - (4,5 hrs)

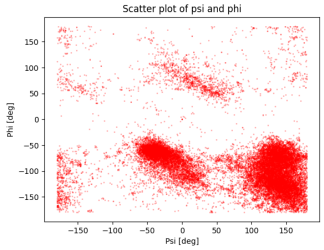
February 2, 2023

Problem 1

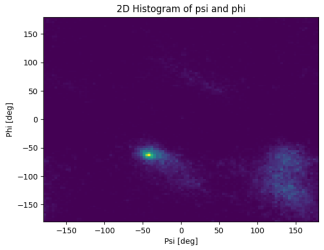
The reasoning behind the need for both a scatter plot and a 2d histogram is that it's hard to see the concentration of points in the scatter plot, which the 2d histogram displays in a clearer way.

```
# Filter data
psi = data['psi']
phi = data['phi']

# Scatter plot
plt.scatter(psi, phi, s=4, alpha=0.25, c="red", marker=".")
plt.xlabel('Psi [deg]')
plt.ylabel('Phi [deg]')
plt.title('Scatter plot of psi and phi')
plt.plot()
plt.figure()

# 2D histogram
plt.hist2d(psi, phi, bins=100)
plt.xlabel('Psi [deg]')
plt.ylabel('Phi [deg]')
plt.title('2D Histogram of psi and phi')
plt.plot()
plt.show()
```

Listing 1: Python code – Problem 1.



(a) Scatter plot



(b) 2D-Histogram

Figure 1: Graphs of prediction accuracy

Problem 2

A

We decided to set the amount of clusters to six. The reason behind this is that if we went lower, we'd lose one of the clusters in the top of the plot. If we went higher it would just add more clusters to the bottom right cluster, which we didn't believe would fit in.

```
    # Filter data
    psi = data['psi']
    phi = data['phi']

    # Insert data into pairs in an array
    data = []
    for x in range(0, min(len(phi), len(psi))):
        data.append([psi[x], phi[x]])

    # K-means clustering
    kmeans = KMeans(n_clusters=6, n_init=10, algorithm= 'elkan')
    label = kmeans.fit_predict(data)
    data = np.array(data)
    label = np.array(label)

    # Getting centroids
    centroids = kmeans.cluster_centers_
    u_clusters = np.unique(label)

    # Plot clusters
    for cluster in u_clusters:
        plt.scatter(data[label == cluster, 0], data[label == cluster, 1], s=6, alpha
    =0.25, marker=".")
    plt.scatter(centroids[:, 0], centroids[:, 1], c='black', s=45)
    plt.legend()
    plt.xlabel('Psi [deg]')
    plt.ylabel('Phi [deg]')
    plt.title('K-means cluster scatter plot of psi and phi')
    plt.show()
```

Listing 2: Python code – Problem 2a.

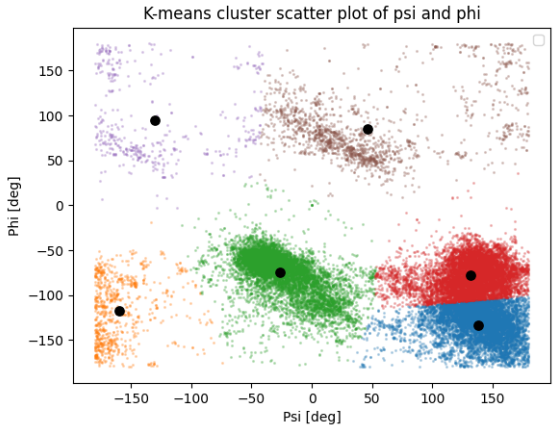


Figure 2: Scatter plot of Psi and Phi using K-means clustering method

B

Yes, the clustering in part A looks reasonable to us. We believe the optimal k-count for this data to be 6. However, the clustering would be if the cluster in the bottom-right corner would be one instead of two. To do this, you'd most likely need to separate the cluster from the rest, cluster the other data individually, and then re-insert the bottom-right cluster.

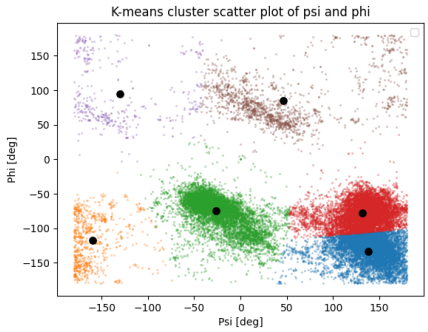
Problem 3

A

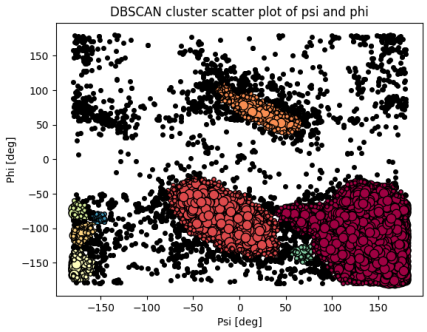
We increased the "eps" value until the amount of noise (represented as the black dots) was reduced to a reasonable level. The value we arrived at was 8.5. For the "min-samples" we chose 52 because it decreased the amount of core points to make the cluster easier to depict. Comparing the DBSCAN to the K-means clustering method, we almost find the same result, except that the DBSCAN separates the data into a few more clusters. This can be seen in the bottom left corner for example, where there is more than one cluster, compared to the K-means plot. However, by modifying the "eps" and "min-samples" parameters, it could be possible to arrive at the same result.

B

When we completed part A, we already highlighted the clusters with larger (core points) and smaller (member points) using more colorful markers, while keeping the outlier markers smaller and colored black.



(a) Scatter plot of Psi and Phi using K-means clustering method



(b) Scatter plot of Psi and Phi using the DBSCAN method

Figure 3: Graphs of two different clustering methods

```
    def dbscanPlot(phi, psi):

    # Insert data into pairs in an array
    data = []
    for x in range(0, min(len(phi), len(psi))):
        data.append([psi[x], phi[x]])

    # DBSCAN clustering
    dbscan = DBSCAN(eps=8.5, min_samples=52).fit(data)
    labels = dbscan.labels_
    data = np.array(data)

    # Plot clusters
    unique_labels = set(labels)
    core_samples_mask = np.zeros_like(labels, dtype=bool)
    core_samples_mask[dbscan.core_sample_indices_] = True

    colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
    for k, col in zip(unique_labels, colors):
        if k == -1:
            # Black used for noise.
            col = [0, 0, 0, 1]

        class_member_mask = labels == k

        xy = data[class_member_mask & core_samples_mask]
        plt.plot(
            xy[:, 0],
            xy[:, 1],
            "o",
            markerfacecolor=tuple(col),
            markeredgecolor="k",
            markersize=9,
        )

        xy = data[class_member_mask & ~core_samples_mask]
        plt.plot(
            xy[:, 0],
            xy[:, 1],
            "o",
            markerfacecolor=tuple(col),
            markeredgecolor="k",
            markersize=4,
        )
    plt.xlabel('Psi [deg]')
    plt.ylabel('Phi [deg]')
    plt.title('DBSCAN cluster scatter plot of psi and phi')
    plt.show()

    return labels
```

Listing 3: Python code – Problem 3a.

C

There are a total of 2185 outliers in the data, with the GLY residue being the most common.

```
outliers = []
# Get outliers
for x in range(0, len(labels)):
    if labels[x] == -1:
        outliers.append(x)
p_type = np.array(p_type)
outliers = np.array(outliers)
outliers = p_type[outliers]


# Plot outliers in bar graph
for i in np.unique(outliers):
    plt.bar(x=i, height=np.count_nonzero(outliers == i))
plt.title(str(len(outliers)) + ' total outliers in bar graph')
plt.xlabel('Residue name')
plt.ylabel('Number of outliers')
plt.show()
```

Listing 4: Python code – Problem 3c.

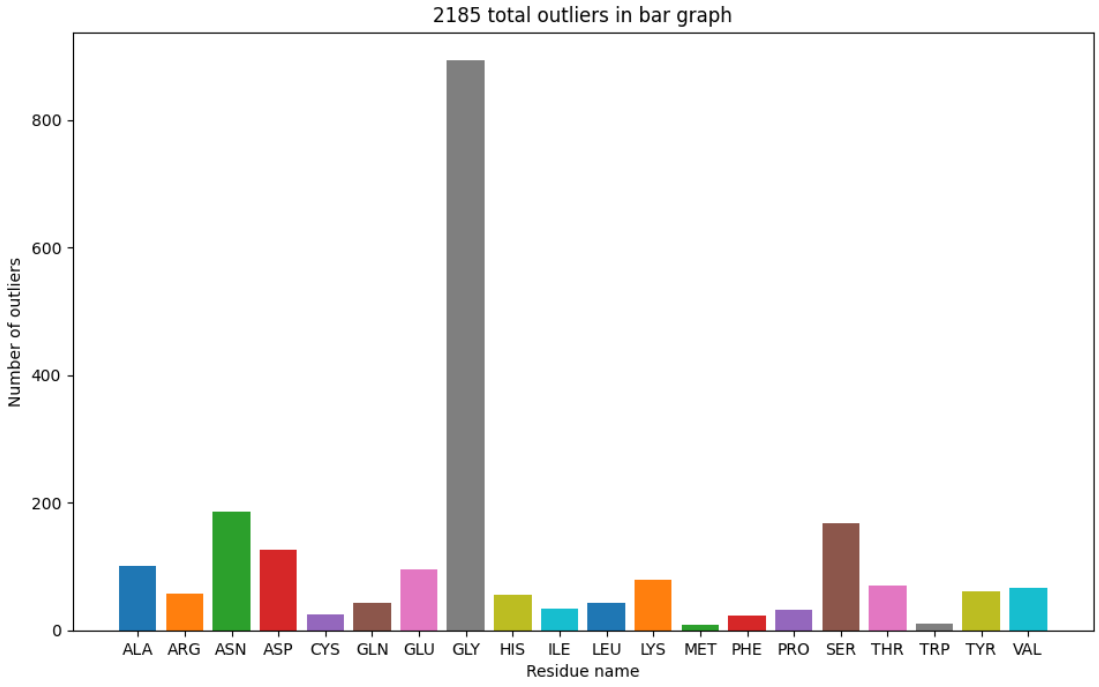


Figure 4: Bar graph of amount of outliers per residue

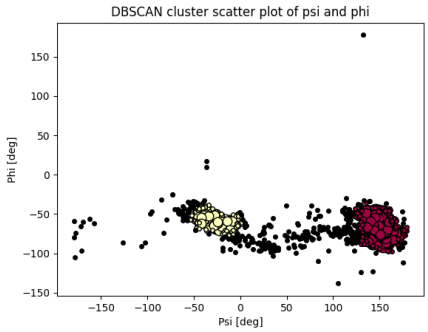
Problem 4

Comparing the general clusters with the PRO residue clusters, we can clearly see that the PRO clusters match on 2 of the clusters from general in the same location. Also, comparing the PRO clusters with the K-means clusters, the right hand PRO cluster matches with the red colored one from K-means.

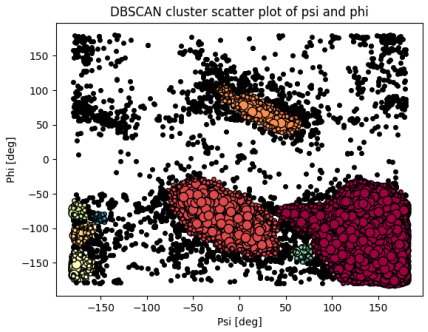
To plot the PRO residue clusters, we use the same method as in Part A, except that we first filter out the data to only include the PRO residue.

```
# Filter data
PROData = data[data["residue name"] == "PRO"]
psi = PROData['psi']
phi = PROData['phi']

# Reset index
phi = np.array(phi)
psi = np.array(psi)

# Plot data using DBSCAN clustering
dbscanPlot(phi, psi)
```

Listing 5: Python code – Problem 4.



(a) Scatter plot of Psi and Phi using DBSCAN method with only PRO residue



(b) Scatter plot of Psi and Phi using the DBSCAN method

Figure 5: Graphs of two different clustering methods