DAT565 Assignment 3 – Group 79

Ziyuan Wang - (10 hrs)
Yuchuan Dong - (10 hrs)
Meixi Lin - (10 hrs)

September 17, 2023

Problem 1

For problem one, we are required to draw the scatter plot and the histogram of the given data.

As for the scatter plot, axis X indicates angles of phi and axis Y indicates angles of psi. We use different colors to refer to points of different residue names.

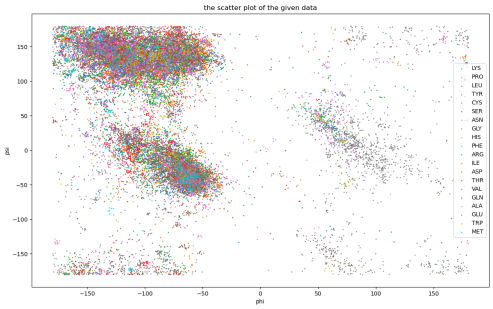

Figure 1: 2D-Histogram of phi and psi

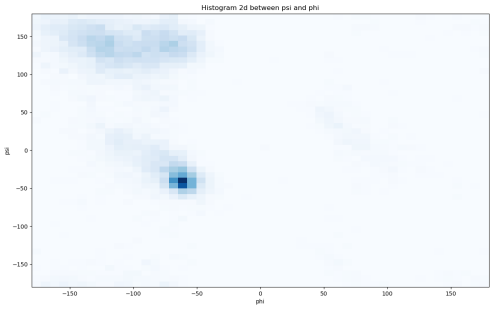

Figure 2: Scatter Plot of phi and psi

For the two-dimensional histogram, the coordinate remains the same as the scatter plot. And the

corresponding graph is shown below. It is clear that angles between -100 and -50 for both psi and phi are the majority, which is the darkest blue in the picture.

Problem 2

.

Problem 2 requires that we need to implement K-Means algorithm to cluster the phi and psi angles in the data file.

Initially, we set the minimum of k is 3 and the maximum of it is 8. We then looped the K-Means algorithm and drew the corresponding scatter plot accordingly, which was listed below. Each cluster is labelled in different color.

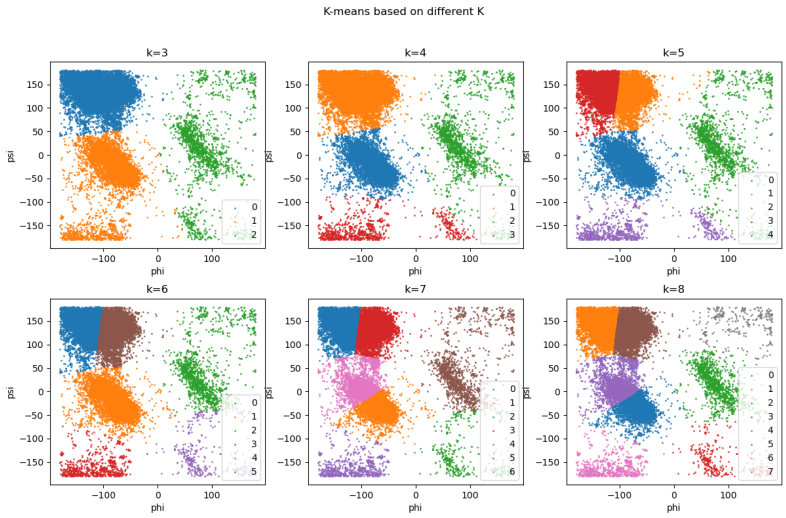

Figure 3: Scatter plot of phi and psi

However, with the parameter k growing larger, the model faced potential risk of overfitting the data. So, then we used the Silhouette Coefficient[1] to approve the performance of our model.

This coefficient is calculated using the mean intra-cluster distance is a and the mean nearest-cluster distance b for each sample point. If the value is close to one, it means that the clustering model performs well and can distinguish the points perfectly. On the contrary, if the value is too close to -1, it means that a sample has been assigned to the wrong cluster, as a different cluster is more closer. And the coefficient is calculated as the following formula:

$$\sum_{i=1}^{n} \frac{b_i - a_i}{max(a_i, b_i)}$$

where n is the number of the sample points.

In order to see the tendency of the Silhouette coefficient with k growing larger, we drew the line chart. And what we found is that, the value keeps decreasing with k increasing which means that the clustering model has a chance to overfit. So the optimal K we suppose is three, because the value of k at three is the largest.

What we present above is the analysis based on principles and metrics. When considering whether k equals to three is reasonable or not, from our perspective, it is reasonable because from the scatter plot where k is 3, we can observe that there is a big gap near the zero angles of phi and data are

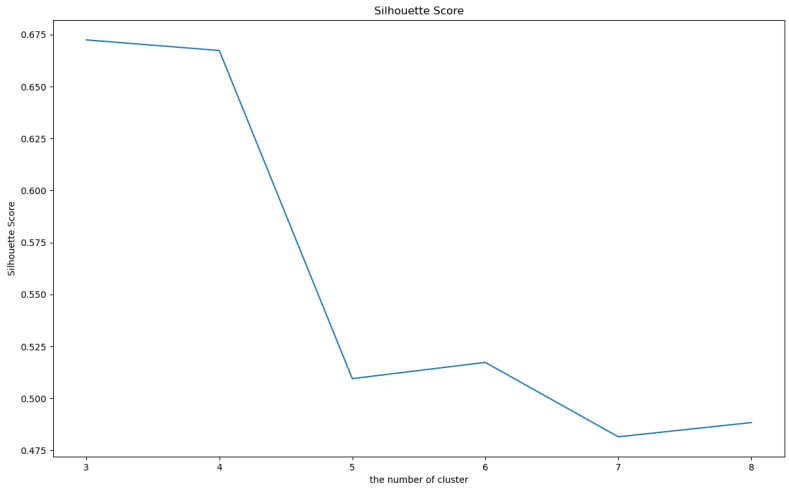

Figure 4: Line Chart of Silhouette Coefficient

distributed mostly around -100 degrees and 100 degrees of phi. Besides that, there is a huge amount of data distributed at the upper right corner, which can be regarded as a cluster. Then, on the same side, the rest of the data are distributed under the blue cluster, which can be considered as another cluster.

Problem 3

As for problem 3, we are required to use a different type of cluster algorithm called DBSCAN. There are two parameters for this algorithm, one of which is called ϵ neighborhood which is for defining the range of the neighborhood. The other is the minimum samples, which defines the minimum samples in the neighborhood.

To specify the two parameters, we first normalize our data using the formula below:

$$X = \frac{X - \mu}{X_{std}}$$

where μ is the mean value of the data and X_{std} is the standard deviation of the data, making our data distribute in a small range and making it easier for us to determine the value of the ϵ and minimum number of samples. We finally set the ϵ to be 0.3 and minimum number of samples to be from 200 to 500.

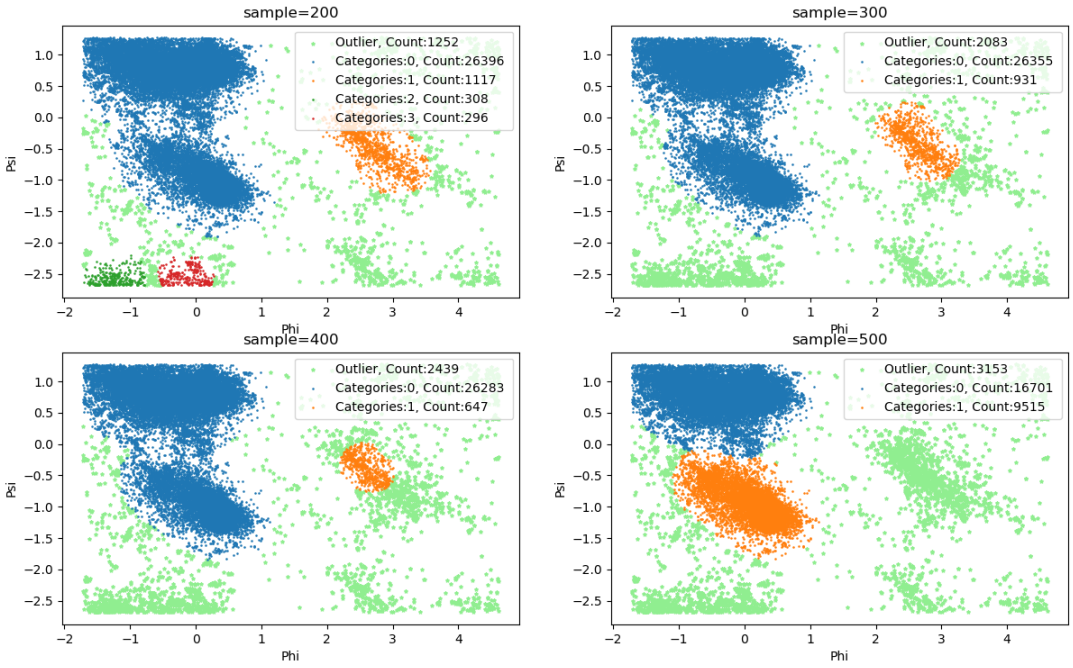

Figure 5: Result of DBSCAN

According to the graph above, when the minimum number of samples are 200, the number of cluster is four. But with the minimum number increasing, the number of cluster stablize at the level of two. We also labelled those points in light green, which are considered as outliers. Then, we take a look at the number of the outliers in the four clustering result. It is obvious that the first model with sample number equaling to 200 has the least number of outlier points.

Next, we plot a bar chart to show how often each of the amino acid residue types are outliers and it is clearly that residue type named GLY has the most outliers among all the other residues.

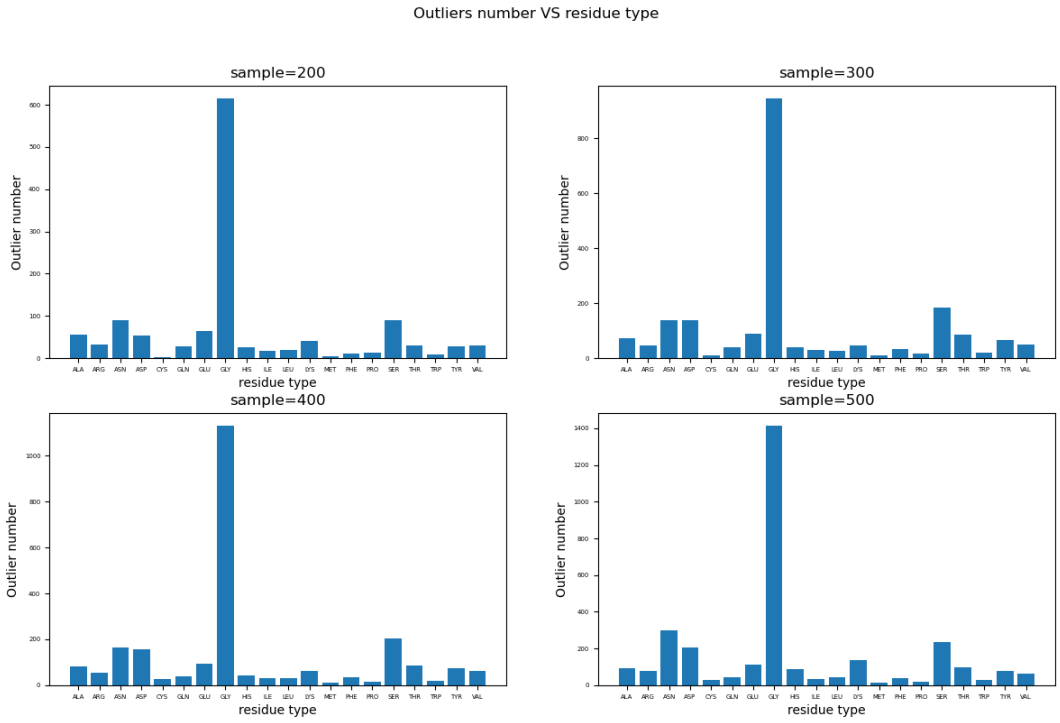

Figure 6: Bar Chart of outliers

Problem 4

In this problem, we utilize the DBSCAN method to cluster amino acid residues of type PRO and compare the clusters with those obtained in Problem 3.

Following the methodology outlined above, we normalize the data and determine the appropriate parameters.

The clusters for amino acid residues of type PRO are presented in Figure 6. We combined the results from Problem 3 and Problem 4 into a single figure with the aim of comparing the clusters of amino acid residues of type PRO with those of the general type. According to figure above, clusters of general type can approximately illustrate different types. The conclusion is that clusters of amino acid residues of type PRO can be categorized in general terms.

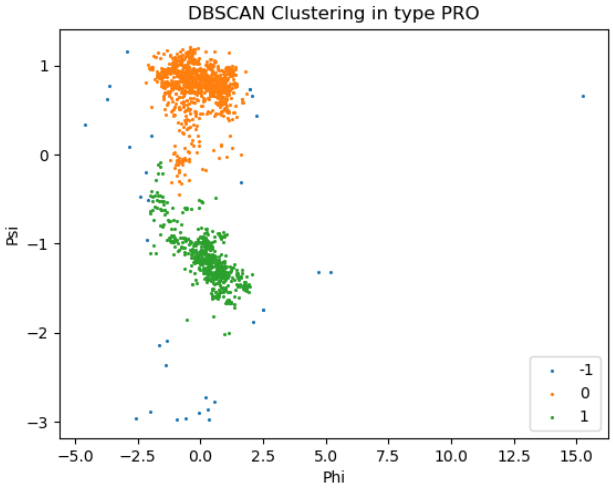

Figure 7: DBSCAN Clustering In TYPE PRO

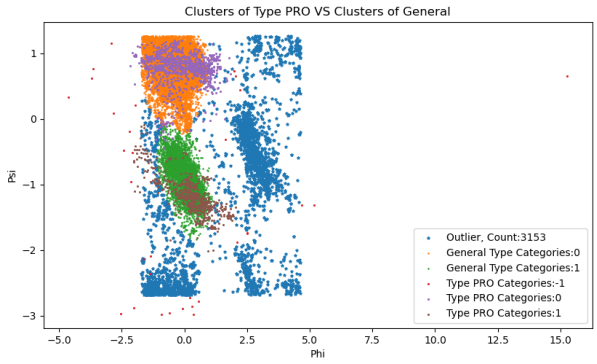

Figure 8: Clusters of Type PRO VS Clusters of General

References

[1] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics 20: 53-65.

Appendixes

Python code

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
import numpy as np
```

```
from sklearn.preprocessing import StandardScaler
"""
Read data, Process data
"""
data = pd.read_csv('data/protein-angle-dataset.csv')
data_array = [[data.iloc[i]['phi'], data.iloc[i]['psi']] for i in range(len(data))]
data_array = np.array(data_array)
# data['phi'] = (data['phi'] - data['phi'].mean()) / data['phi'].std()
# data['psi'] = (data['psi'] - data['psi'].mean()) / data['psi'].std()

def task1():
    residue_name = data['residue name'].unique()
    for name in residue_name:
        phi, psi = data[data['residue name'] == name]['phi'].values, data[data['
    residue name'] == name]['psi'].values
        plt.scatter(phi, psi, label=name, s = 0.9)

    plt.xlabel('phi')
    plt.ylabel('psi')
    plt.title('the scatter plot of the given data')
    plt.legend()
    # plt.show()
    plt.clf()

    plt.hist2d(data['phi'], data['psi'], bins=50, cmap = 'Blues', density=True)
    plt.xlabel('phi')
    plt.ylabel('psi')
    plt.xlim(-180, 180)
    plt.ylim(-180, 180)
    plt.title('Histogram 2d between psi and phi')
    plt.show()

def task2(max_cluster = 8):
    score_list = []
    start_k = 2
    for i in range(start_k, max_cluster+1):
        k = KMeans(n_clusters=i).fit(data_array)
        score = silhouette_score(data_array, k.predict(data_array))
        score_list.append(score)
        labels = np.array(k.labels_)

        plt.subplot(2, 4, i - start_k + 1)
        plt.xlabel('phi')
        plt.ylabel('psi')
        plt.title(f'k={i}')
        for j in range(i):
            index = np.where(labels==j)[0]
            phi, psi = data.iloc[index]['phi'].values, data.iloc[index]['psi'].values
            plt.scatter(phi, psi, label=i, s=0.7)
            # print(index)
            # exit(0)
        plt.legend(labels=[j for j in range(i)], loc='lower right')

    # plt.scatter(data['phi'], data['psi'], label = k.labels_)
    plt.suptitle("K-means based on different K")
    plt.show()
    plt.title('Silhouette Score')
    plt.xlabel('the number of cluster')
    plt.ylabel('Silhouette Score')
    plt.plot([i for i in range(start_k, max_cluster + 1)], score_list)
    plt.show()

def task3():
    # # draw k-distance
```

```
    task3_data = data
    task3_data['phi'] = (data['phi'] - data['phi'].mean()) / data['phi'].std()
    task3_data['psi'] = (data['psi'] - data['psi'].mean()) / data['psi'].std()
    task3_data_array = [[task3_data.iloc[i]['phi'], task3_data.iloc[i]['psi']] for i
    in range(len(data))]

    task3_data_array = np.array(task3_data_array)


    for times, sample in enumerate(range(200, 501, 100)):
        dbscan = DBSCAN(eps=0.3,min_samples=sample).fit(task3_data_array)
        # dbscan.fit(data_array)
        labels = dbscan.labels_
        unique_label, count = np.unique(labels, return_counts=True)
        print(unique_label, count)
        plt.subplot(2,2,times+1)
        for i in range(-1, max(labels) + 1):
            index = np.where(labels==i)[0]
            # print(len(index))
            phi, psi = data.iloc[index]['phi'].values, data.iloc[index]['psi'].values
            if i == -1:
                plt.scatter(phi, psi, label=f'Outlier, Count:{count[i+1]} ', s=7,
    color='lightgreen', marker='*')
            else:

                plt.scatter(phi, psi, label=f'Categories:{i}, Count:{count[i+1]} ', s
    =0.7)
        plt.legend(loc='upper right')
        plt.xlabel('Phi')
        plt.ylabel('Psi')
        plt.title(f'sample={sample}')
    # plt.show()
    plt.clf()
    for times, sample in enumerate(range(200, 501, 100)):
        db = DBSCAN(eps=0.3, min_samples=sample)
        db.fit(task3_data_array)
        # data_new = pd.concat([task3_data,pd.Series({'label': db.labels_})], axis=1)
        task3_data['label'] = db.labels_
        data33 =  (task3_data[task3_data['label']==-1]).groupby('residue name')
        plt.subplot(2,2,times+1)
        plt.bar(data33.size().index, data33.size().values)
        plt.tick_params(labelsize=5)
        plt.xlabel("residue type")
        plt.ylabel("Outlier number")
        plt.title(f"sample={sample}")
    plt.suptitle("Outliers number VS residue type")
    plt.show()
"""
Plot the bar chat between residue and outlier in the case of minsample=200, eps=0.3
"""



def task4():
    pro_data = data[data['residue name'] == 'PRO'].copy()
    pro_data['phi'] = (pro_data['phi'] - pro_data['phi'].mean()) / pro_data['phi'].std
    ()
    pro_data['psi'] = (pro_data['psi'] - pro_data['psi'].mean()) / pro_data['psi'].std
    ()
    data_array = [[pro_data.iloc[i]['phi'], pro_data.iloc[i]['psi']] for i in range(
    len(pro_data))]
    data_array = np.array(data_array)
    # plt.scatter(data_array[:, 0], data_array[:, 1])
    # plt.show()
```

```
    dbscan = DBSCAN(min_samples=50).fit(data_array)
    labels = dbscan.labels_
    for i in range(-1, max(labels) + 1):
        index = np.where(labels==i)[0]
        # print(len(index))
        phi, psi = pro_data.iloc[index]['phi'].values, pro_data.iloc[index]['psi'].
    values
        plt.scatter(phi, psi, label=i, s=1.5)
    plt.legend(labels=[j for j in range(-1, max(labels)+1)], loc='lower right')
    plt.xlabel('Phi')
    plt.ylabel('Psi')
    plt.title('DBSCAN Clustering')
    plt.show()



if __name__ == '__main__':
    task1()
    task2()
    task3()
    task4()
```

Listing 1: Problem 1 Source Code