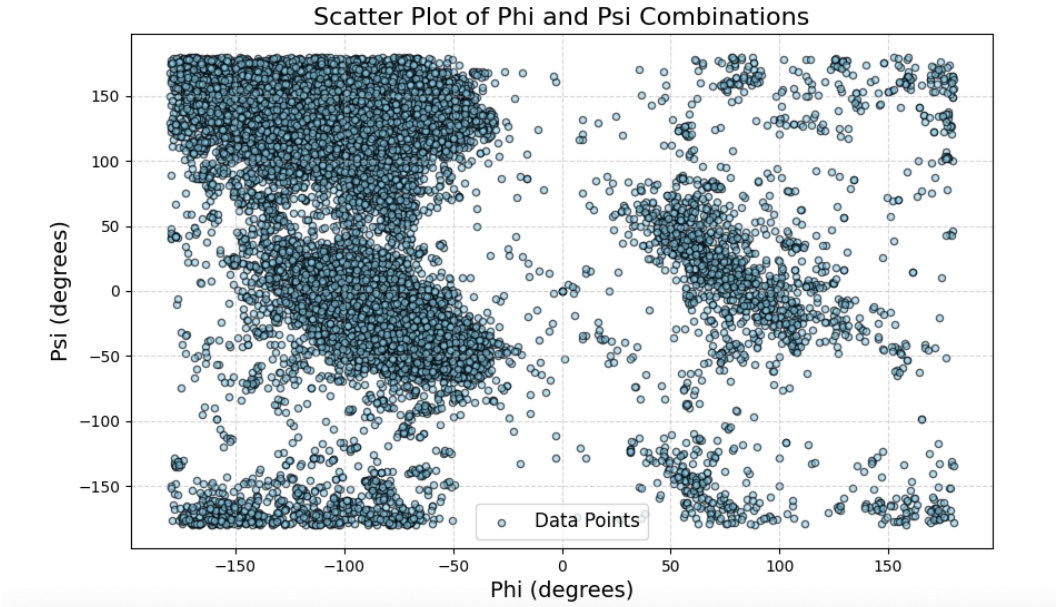
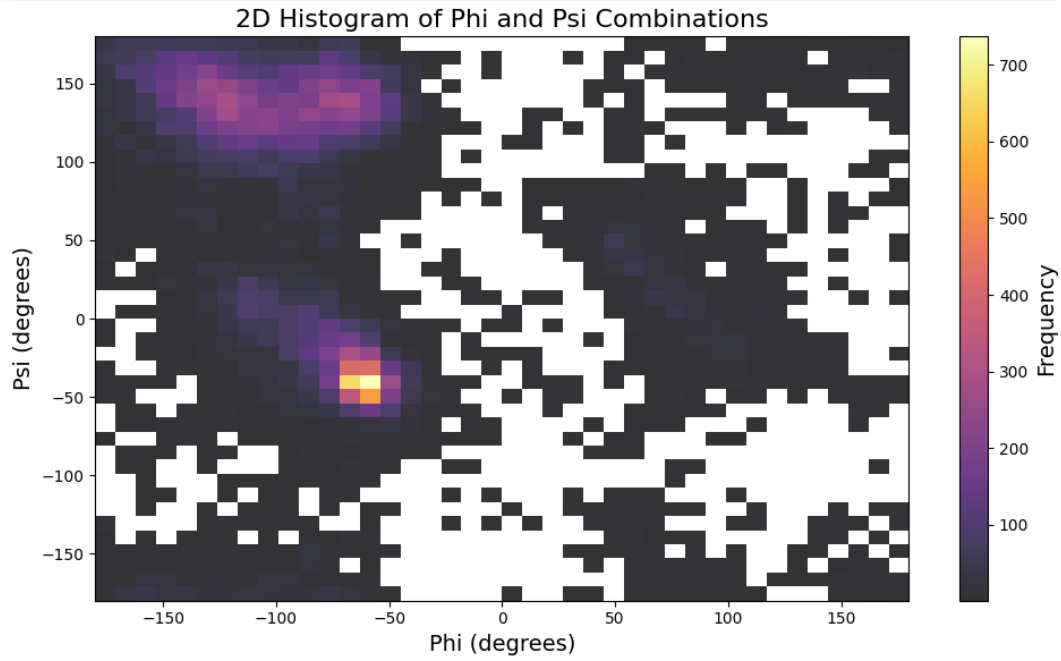
Assignment 3
Mihael Zlatev

The focus of this analysis is the primary conformation of proteins. A protein chain can fold into its native structure through rotations around two key bonds in the main chain, referred to as ϕ (phi) and ψ (psi). Certain combinations of ϕ and ψ angles are not feasible, while others are highly prevalent due to their energetic favorability.

Task 1

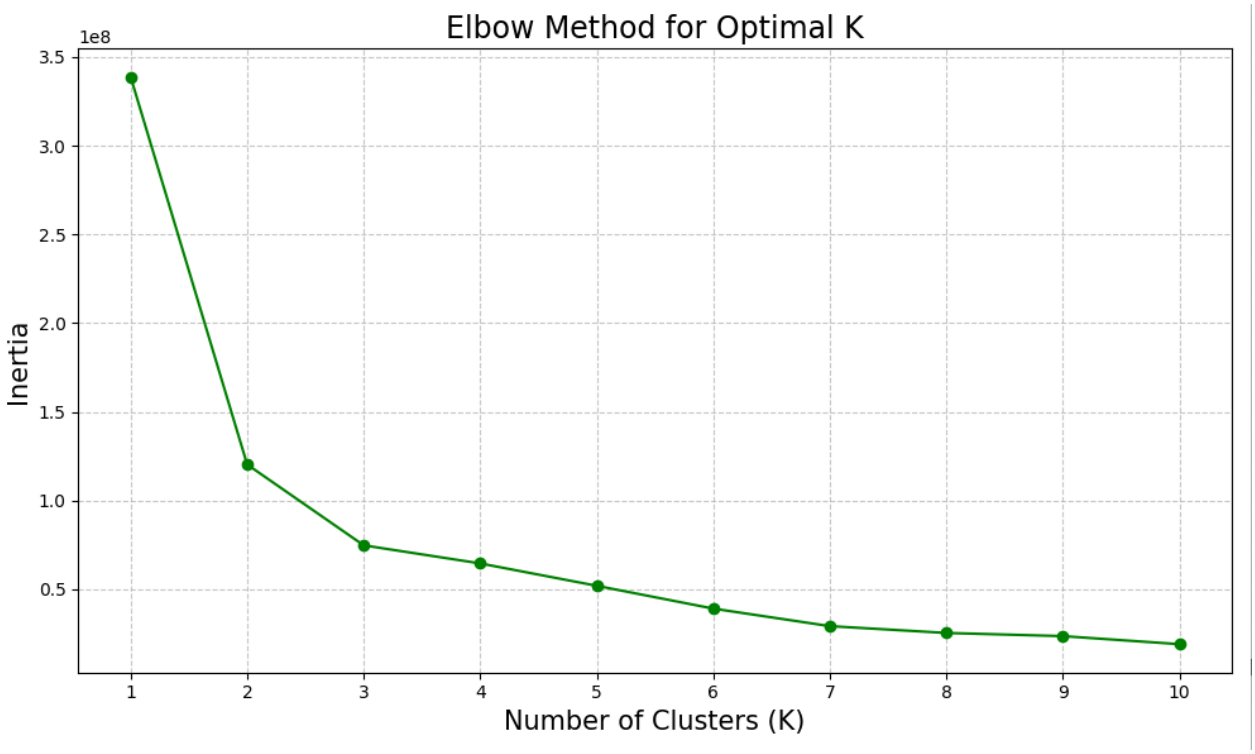




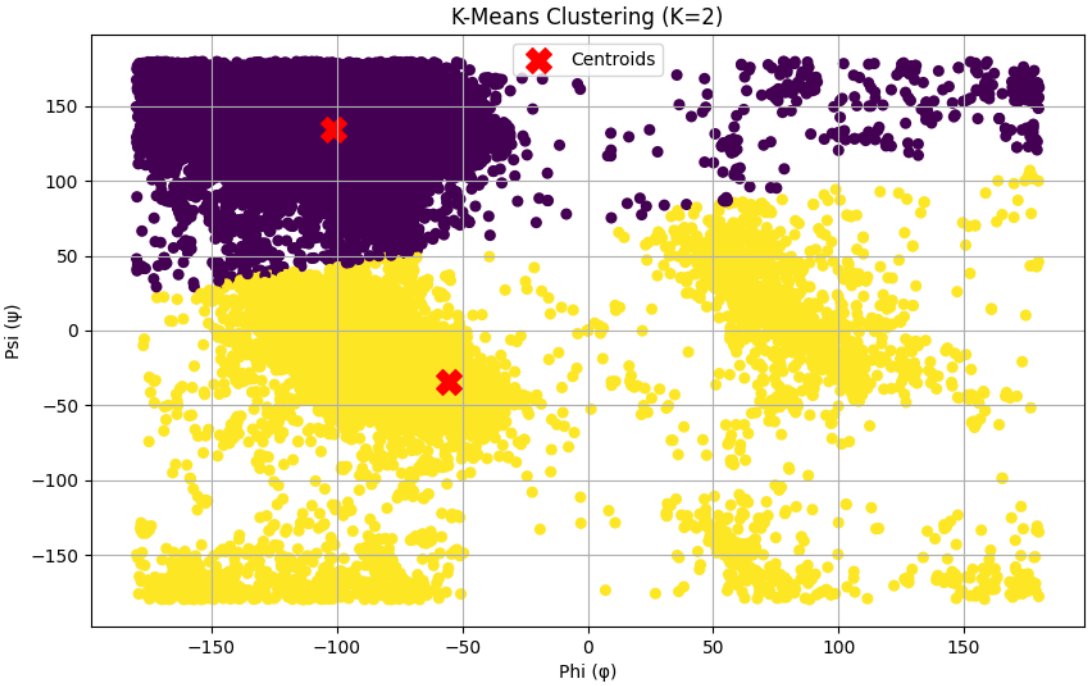
From the plot it can be seen that the cluster are the same using both methods.

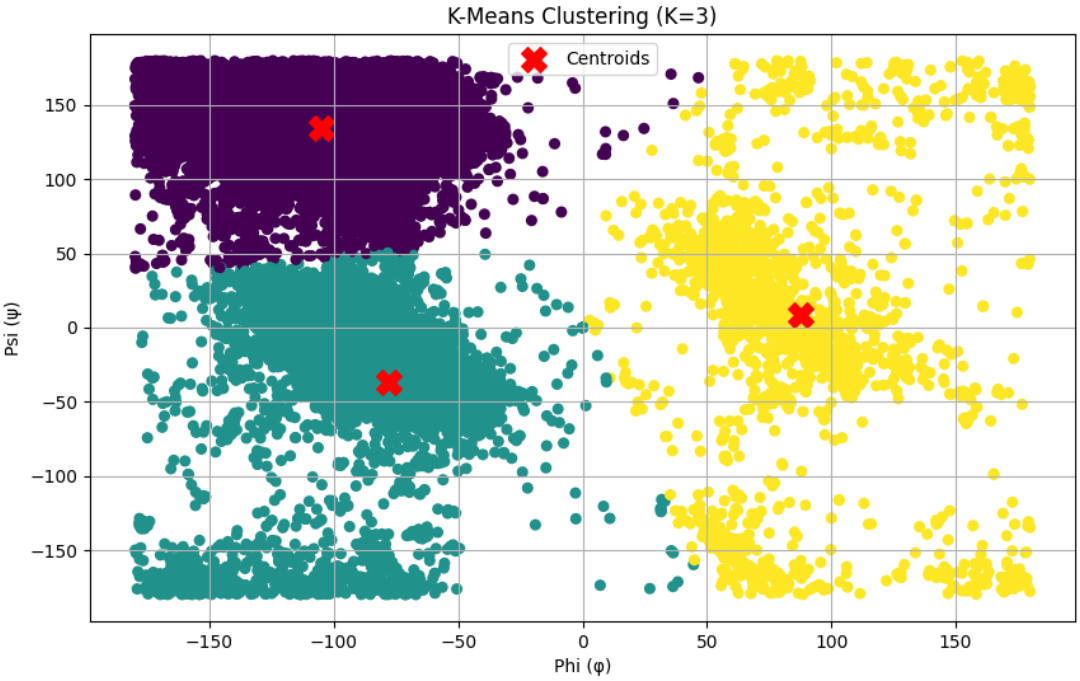
Task 2

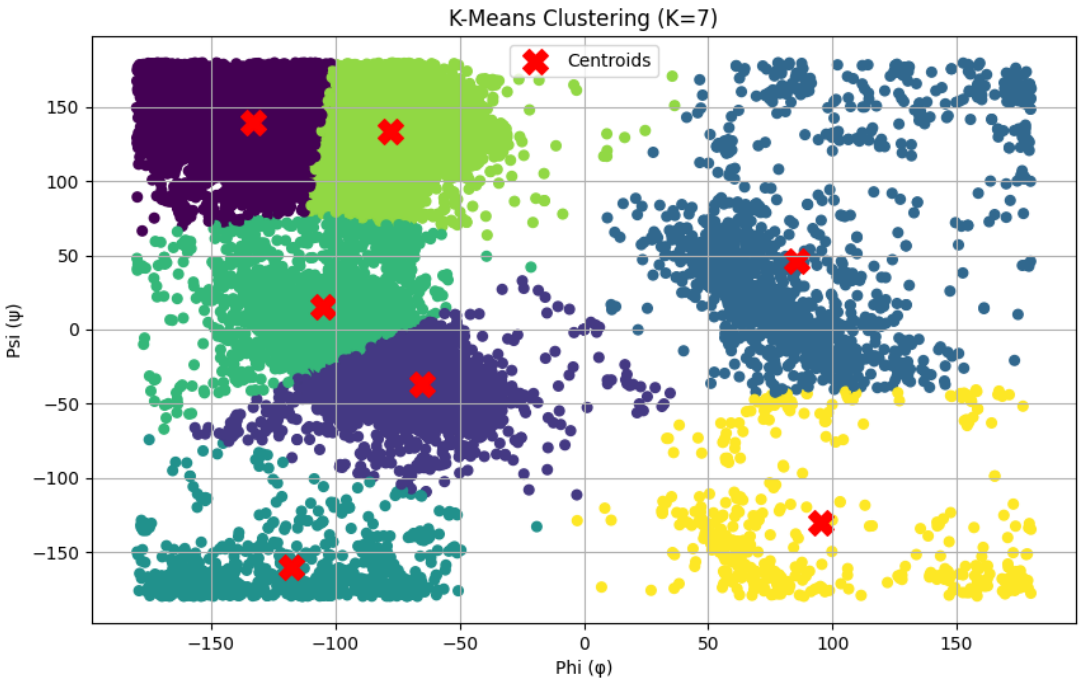
Clustering phi and psi angle combinations using the K-means method



Using the Elbow method, it can be determined that the most appropriate values of K are 2 or 3.

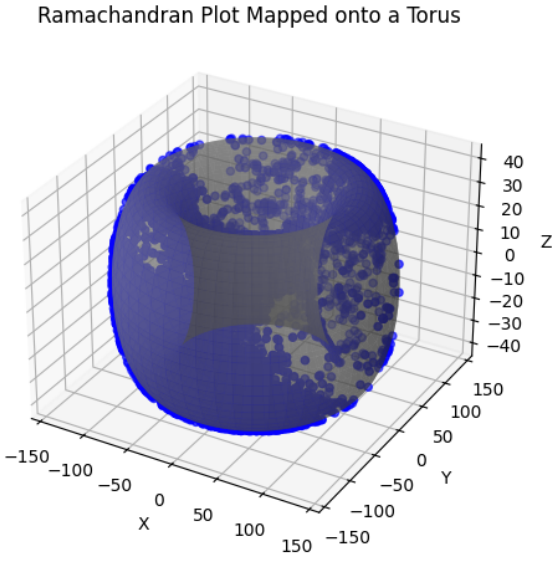

K-Means Clustering (K=2)
Centroids
Psi (ψ)
Phi (φ)
150
100
50
0
−50
−100
−150
−150
−100
−50
0
50
100
150


K-Means Clustering (K=3)
Centroids
Psi (ψ)
Phi (φ)
150
100
50
0
−50
−100
−150
−150
−100
−50
0
50
100
150

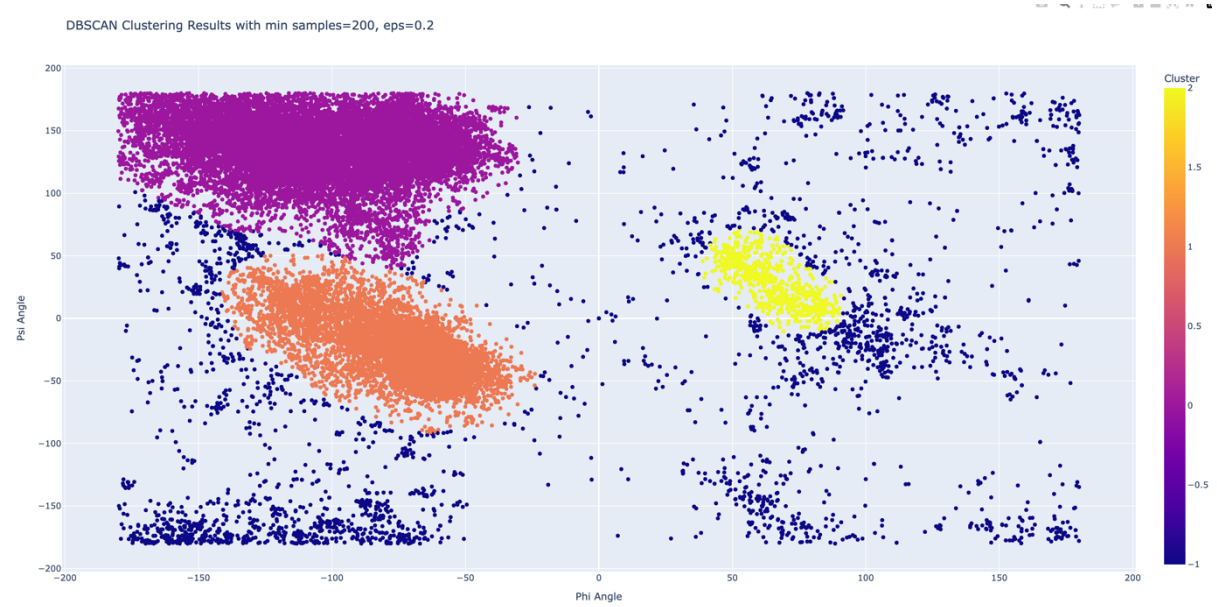


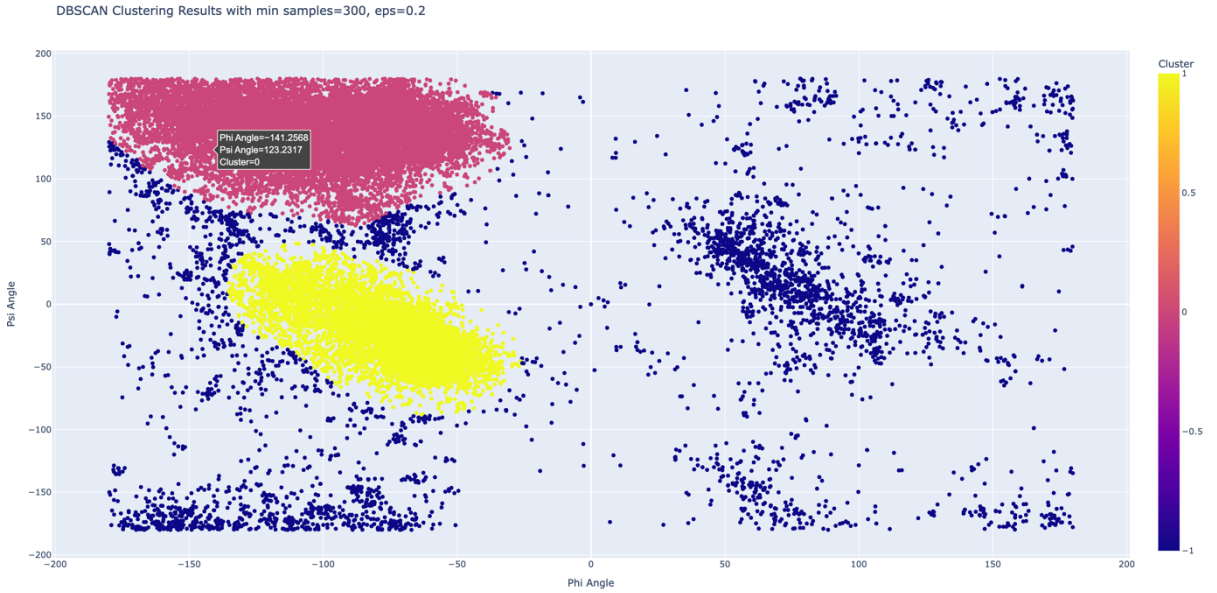
With K values of **2** and 3, the results appear reasonable, and the clusters form as anticipated. However, with **K=7**, the outcomes deviate from expectations. It's important to note that four of the clusters contain significantly fewer data points than the others, making them potentially unreliable for informed decision-making.

Task 2 - Optional Task



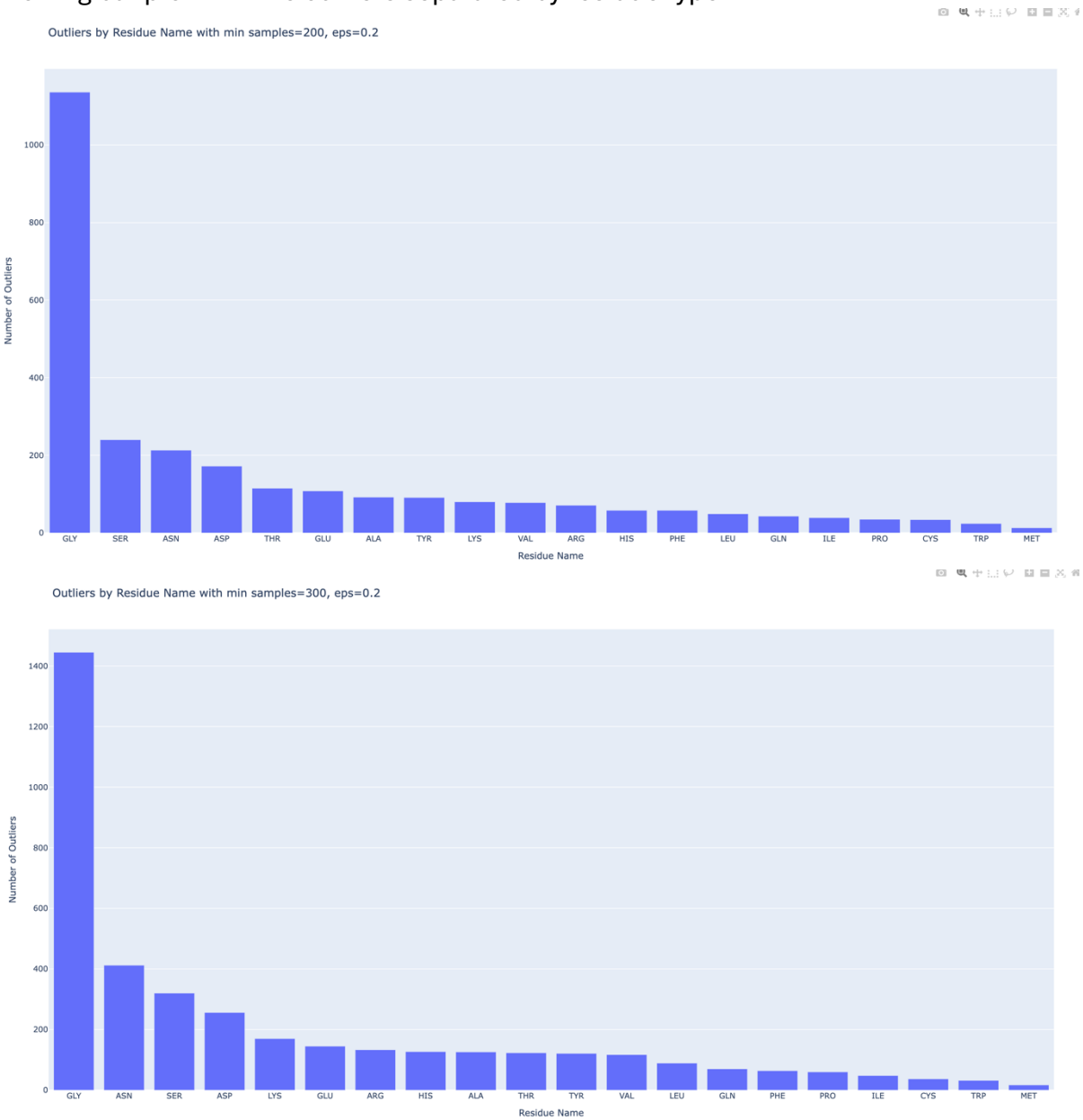
Task 3





The DBSCAN method identifies 2 or 3 clusters, with the remaining points classified as outliers. This indicates that, although K-Means can generate 7 clusters, at least 4 of them are likely insignificant. I experimented with various epsilon and minimum sample values, but the results consistently remained unsatisfactory.

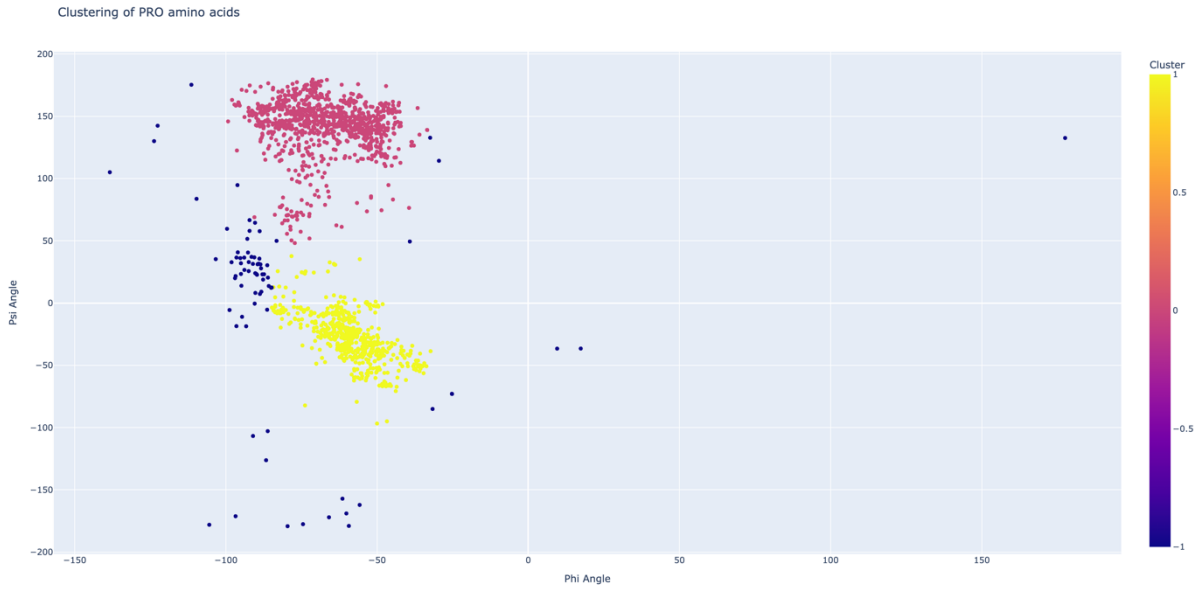
Plotting bar plot with the outliers separated by residue type.



It can be easily seen that there are significantly more outliers of residue type GLY than of any other type. The least outliers are part of the MET amino acid residue type.

Task 4

Using the DBSCAN methods for the PRO amino acid type reveals the following graphic:



It can be observed that the 2 clusters that form strongly correlate with the clusters that are formed when using the DBSCAN method on the whole dataset

PYTHON CODE

```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans

import pandas as pd
import numpy as np
import plotly.express as px
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

DATA_CSV = 'data_assignment3.csv'
PHI = 'phi'
PSI = 'psi'

def init_data():
    data = pd.read_csv(DATA_CSV)
    return data

def plot_distribution_of_phi_psi(phi_data, psi_data):
```

```
    plt.figure(figsize=(10, 6))
    plt.scatter(phi_data, psi_data, c='skyblue', alpha=0.6, s=20, edgecolors='k',
marker='o', label='Data Points')
    plt.title("Scatter Plot of Phi and Psi Combinations", fontsize=16)
    plt.xlabel("Phi (degrees)", fontsize=14)
    plt.ylabel("Psi (degrees)", fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.legend(loc='best', fontsize=12)

    plt.figure(figsize=(10, 6))
    plt.hist2d(phi_data, psi_data, bins=40, cmap='inferno', cmin=1, alpha=0.8)
    plt.title("2D Histogram of Phi and Psi Combinations", fontsize=16)
    plt.xlabel("Phi (degrees)", fontsize=14)
    plt.ylabel("Psi (degrees)", fontsize=14)
    cbar = plt.colorbar()
    cbar.set_label("Frequency", fontsize=14)
    plt.tight_layout()
    plt.show()

def plot_kmean_clustering(data):
    inertia = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=0)
        kmeans.fit(data)
        inertia.append(kmeans.inertia_)

    plt.figure(figsize=(10, 6))
    plt.plot(range(1, 11), inertia, marker='o', linestyle='-', color='g')
    plt.title('Elbow Method for Optimal K', fontsize=17)
    plt.xlabel('Number of Clusters (K)', fontsize=15)
    plt.ylabel('Inertia', fontsize=15)
    plt.xticks(range(1, 11))
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

def plot_romachandran(psi_angles, phi_angles):
    # Create a torus in 3D space
    u = np.linspace(0, 2 * np.pi, 100)
    v = np.linspace(0, 2 * np.pi, 100)
    u, v = np.meshgrid(u, v)
    R, r = 100, 40
    x = (R + r * np.cos(v)) * np.cos(u)
    y = (R + r * np.cos(v)) * np.sin(u)
    z = r * np.sin(v)

    # Create a 3D plot with the torus
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x, y, z, color='gray', alpha=0.5)
```

```
    scaled_phi = phi_angles / 180 * np.pi
    scaled_psi = psi_angles / 180 * np.pi
    x_mapped = (R + r * np.cos(scaled_psi)) * np.cos(scaled_phi)
    y_mapped = (R + r * np.cos(scaled_psi)) * np.sin(scaled_phi)
    z_mapped = r * np.sin(scaled_psi)
    ax.scatter(x_mapped, y_mapped, z_mapped, c='blue', s=20)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.set_title('Ramachandran Plot Mapped onto a Torus')
    plt.show()

def visualise_k_means(K, clustering_data):
    kmeans = KMeans(n_clusters=K, random_state=42)
    kmeans.fit(clustering_data)
    labels = kmeans.labels_
    centroids = kmeans.cluster_centers_
    plt.figure(figsize=(10, 6))
    plt.scatter(clustering_data[:, 0], clustering_data[:, 1], c=labels,
cmap='viridis', s=30)
    plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200,
label='Centroids')
    plt.title(f'K-Means Clustering (K={K})')
    plt.xlabel('Phi (φ)')
    plt.ylabel('Psi (ψ)')
    plt.legend()
    plt.grid(True)
    plt.show()

def plot_DBSCAN_and_outliers_barplot(eps, min_samples, data):
    clustering_data=data[[PHI, PSI]]
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(clustering_data)
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    cluster_assignments = dbscan.fit_predict(X_scaled)
    data['Cluster'] = cluster_assignments
    fig = px.scatter(data, x=PHI, y=PSI, color='Cluster',
                     title=f'DBSCAN Clustering Results with min
samples={min_samples}, eps={eps}', labels={'phi': 'Phi Angle', 'psi': 'Psi Angle'})
    fig.show()

    outliers = data[data['Cluster'] == -1]
    outlier_counts = outliers['residue name'].value_counts().reset_index()
    outlier_counts.columns = ['Residue Name', 'Number of Outliers']
    fig = px.bar(outlier_counts, x='Residue Name', y='Number of Outliers',
                 labels={'Residue Name': 'Residue Name', 'Number of Outliers':
'Number of Outliers'},
                 title=f'Outliers by Residue Name with min samples={min_samples},
eps={eps}')
    fig.show()
```

```
def plot_PRO_amino_acid_type(data, eps, min_samples):
    pro_data = data[data['residue name'] == 'PRO']
    X = pro_data[[PHI, PSI]]
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    cluster_assignments = dbscan.fit_predict(X_scaled)
    pro_data['Cluster'] = cluster_assignments
    fig = px.scatter(pro_data, x='phi', y='psi', color='Cluster',
                     title='Clustering of PRO amino acids', labels={'phi': 'Phi
Angle', 'psi': 'Psi Angle'})
    fig.show()

if __name__ == '__main__':
    data = init_data()
    phi_data = data[PHI]
    psi_data = data[PSI]
    phi_psi_clustering_combinations = data[[PHI, PSI]]
    phi_psi_clustering_combinations_numpy =
phi_psi_clustering_combinations.to_numpy()

    # # Task 1
    plot_distribution_of_phi_psi(phi_data, psi_data)

    # # Task 2
    plot_kmean_clustering(phi_psi_clustering_combinations)
    visualise_k_means(2, phi_psi_clustering_combinations_numpy)
    visualise_k_means(3, phi_psi_clustering_combinations_numpy)
    visualise_k_means(7, phi_psi_clustering_combinations_numpy)
    plot_romachandran(psi_data, phi_data)

    # Task 3
    plot_DBSCAN_and_outliers_barplot(data=data, min_samples=200, eps=0.2)
    plot_DBSCAN_and_outliers_barplot(data=data, min_samples=300, eps=0.2)

    # Task 4
    plot_PRO_amino_acid_type(data, eps=0.6, min_samples=100)
```