DAT405 Assignment 3 – Shivneshwar Velayutham

November 22, 2022

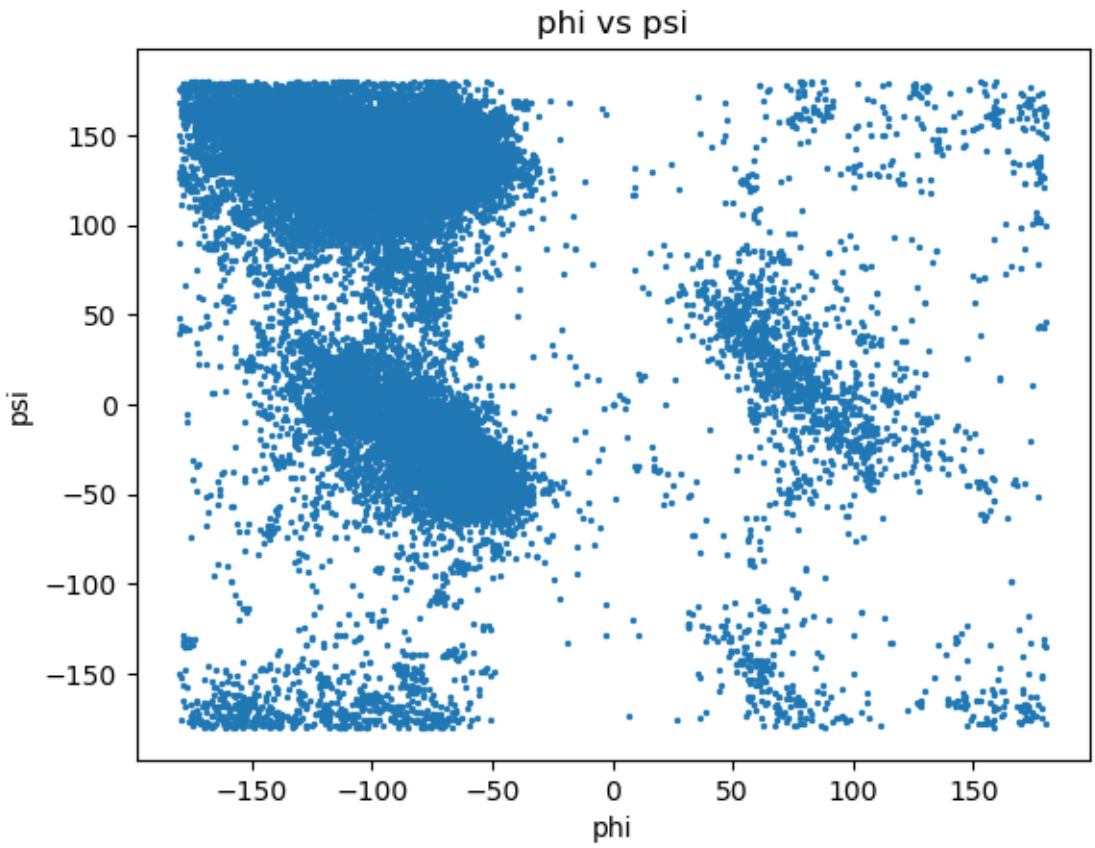
[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_samples, silhouette_score
```

[41]:
```
# Converting input csv to dataframe
df = pd.read_csv("assignment3-data-1.csv")
df2 = df[['phi', 'psi']]
dfpro = df[df['residue name'] == 'PRO'][['phi', 'psi']]
dfgly = df[df['residue name'] == 'GLY'][['phi', 'psi']]
```

1 Problem 1

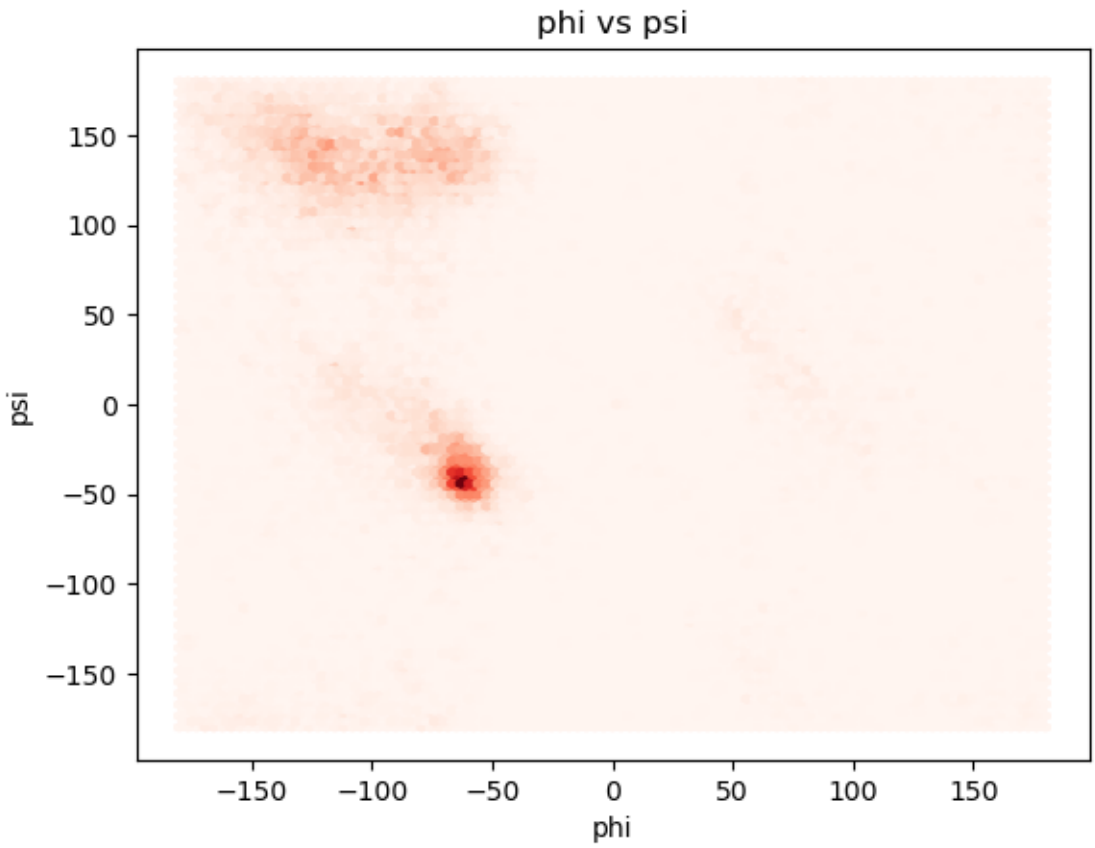
1.1 Problem 1a)

[4]:
```
plt.scatter(df['phi'], df['psi'], s=2)
plt.title('phi vs psi')
plt.xlabel('phi')
plt.ylabel('psi')
plt.show()
```



1.2 Problem 1b)

[5]:
```
plt.hexbin(df['phi'], df['psi'], gridsize = 100, cmap ='Reds')
plt.title('phi vs psi')
plt.xlabel('phi')
plt.ylabel('psi')
plt.show()
```



2 Problem 2

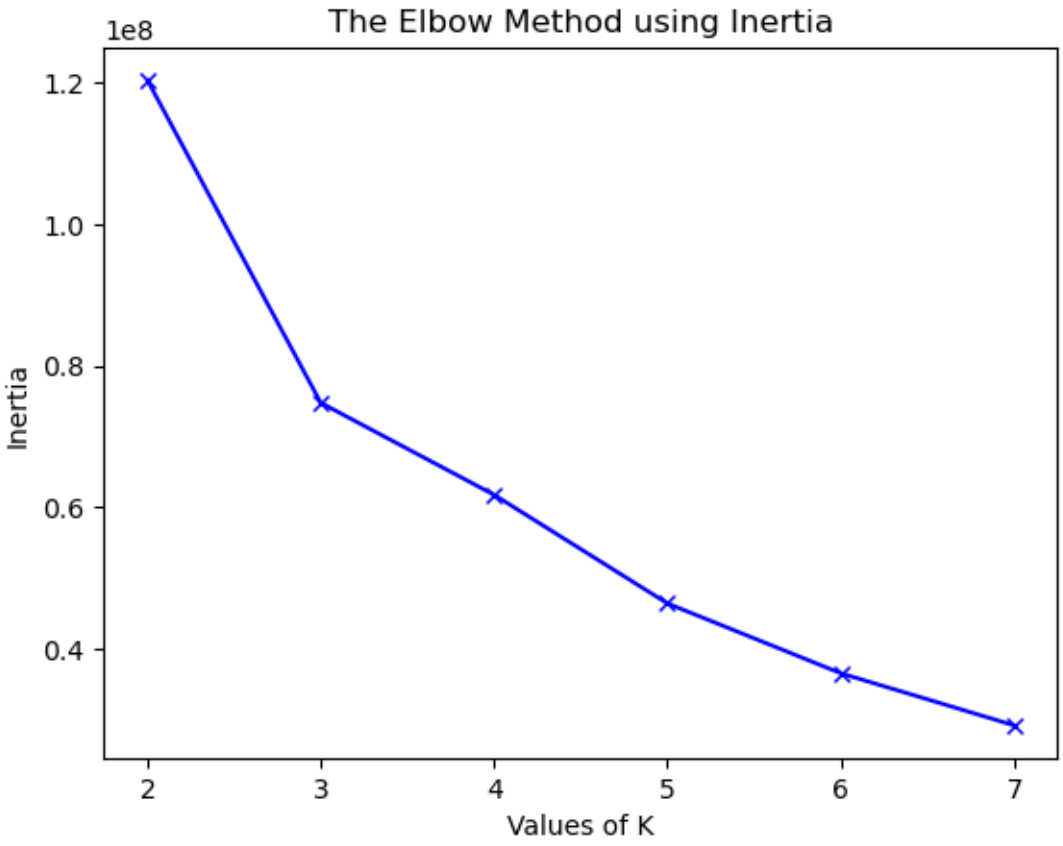
2.1 Problem 2a)

The value of k being experimented starting with 2 to 8. 8 has been selected as with my many trials, 8 is beyond the elbow value and the silhoutte coeffecient is low and reduces and this is bad since it's better when the coeffecient is closer to 1.

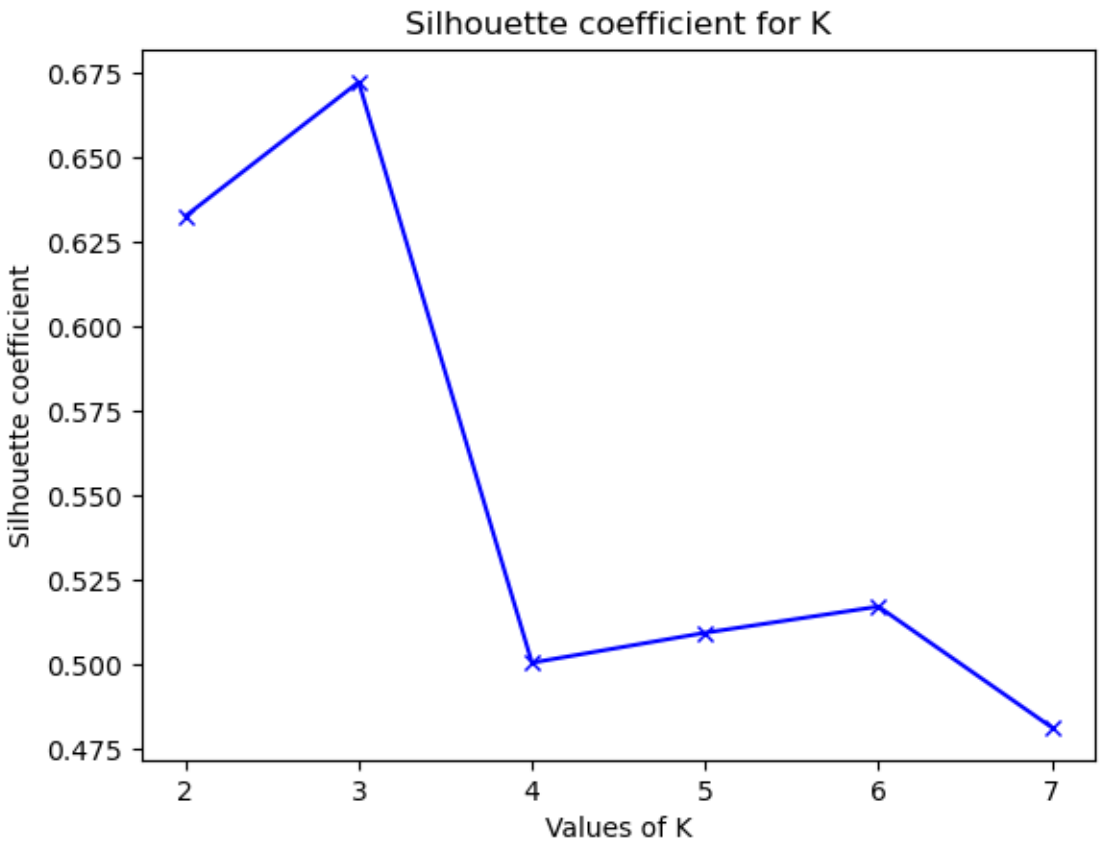
[6]:
```
inertias = []
scores = []
K = range(2, 8)

chosen_k = 3
df3 = []

for k in K:
    # Building and fitting the model
    kmeanModel = KMeans(n_clusters=k)
    dft = kmeanModel.fit_predict(df2)
```

```
    inertias.append(kmeanModel.inertia_)

    scores.append(silhouette_score(df2, dft, metric='euclidean'))

    if k == chosen_k:
        df3 = dft
```

[7]:
```
plt.plot(K, inertias, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Inertia')
plt.title('The Elbow Method using Inertia')
plt.show()
```



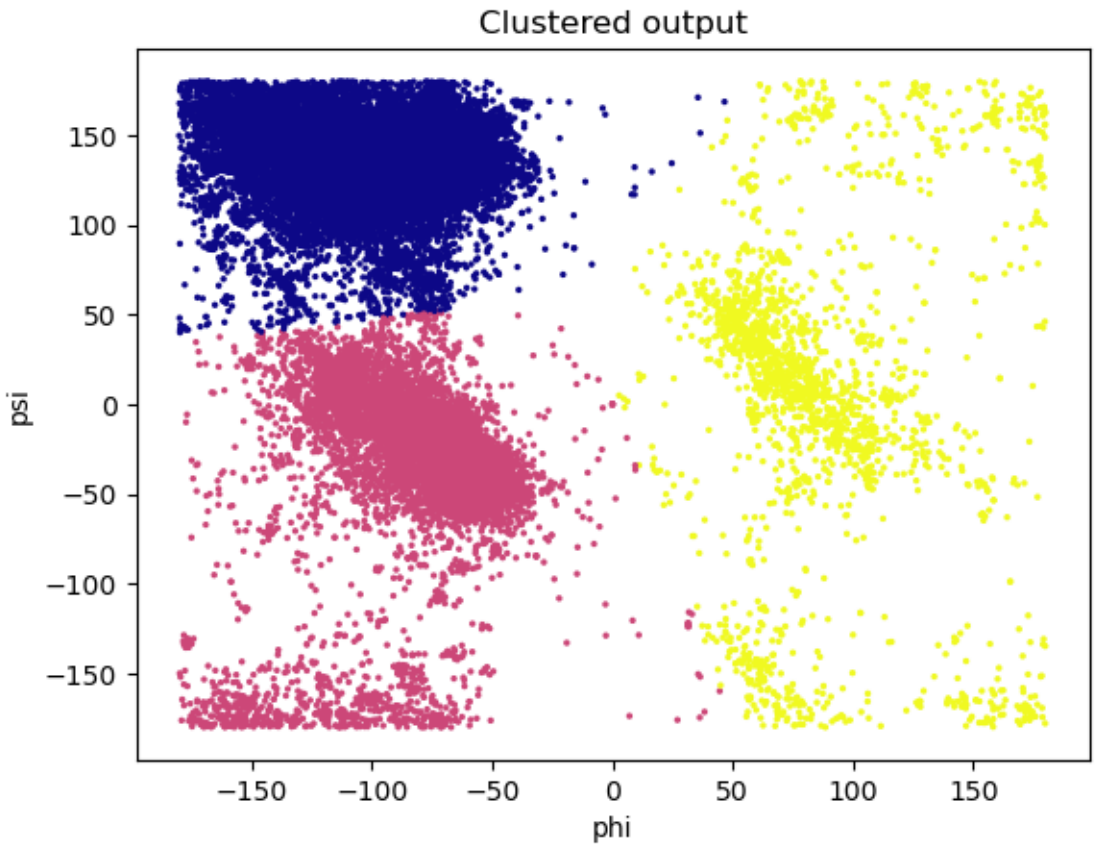
[8]:
```
plt.plot(K, scores, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette coefficient')
plt.title('Silhouette coefficient for K')
plt.show()
```



2.2 Problem 2b)

As we can see from the above graphs the silhouette coeffecient is highest when k equals 3 and the elbow joint occurs at 3 so 3 has been chosen as ideal k value. As mentioned already validation is done with the silhoutte coeffecient.

[87]:
```
plt.scatter(df2['phi'], df2['psi'], c=df3, s=2, cmap='plasma')
plt.title('Clustered output')
plt.xlabel('phi')
plt.ylabel('psi')
plt.show()
```



2.3 Problem 2c)

The clustering does not look reasonable especially the yellow cluster which doesn't look like a proper cluster and all the outliers are taken into consideration. That would be the biggest issue with k means clustering.

2.4 Problem 2d)

Yes, it's possible to augment the data so that we can use the periodic property of the data. The values are between -180 and 180. Thus if we add 180 and then do modulus of 180, then the augmented values of -180 would be 0 and +180 would also be 0 forming a cycle and this is exactly what periodic means.

Without augmentation, -180 and 180 are quite far apart and thus will not be clustered together but with this new augmention, they will be and we are making use of the periodic property.

3 Problem 3

3.1 Problem 3a)

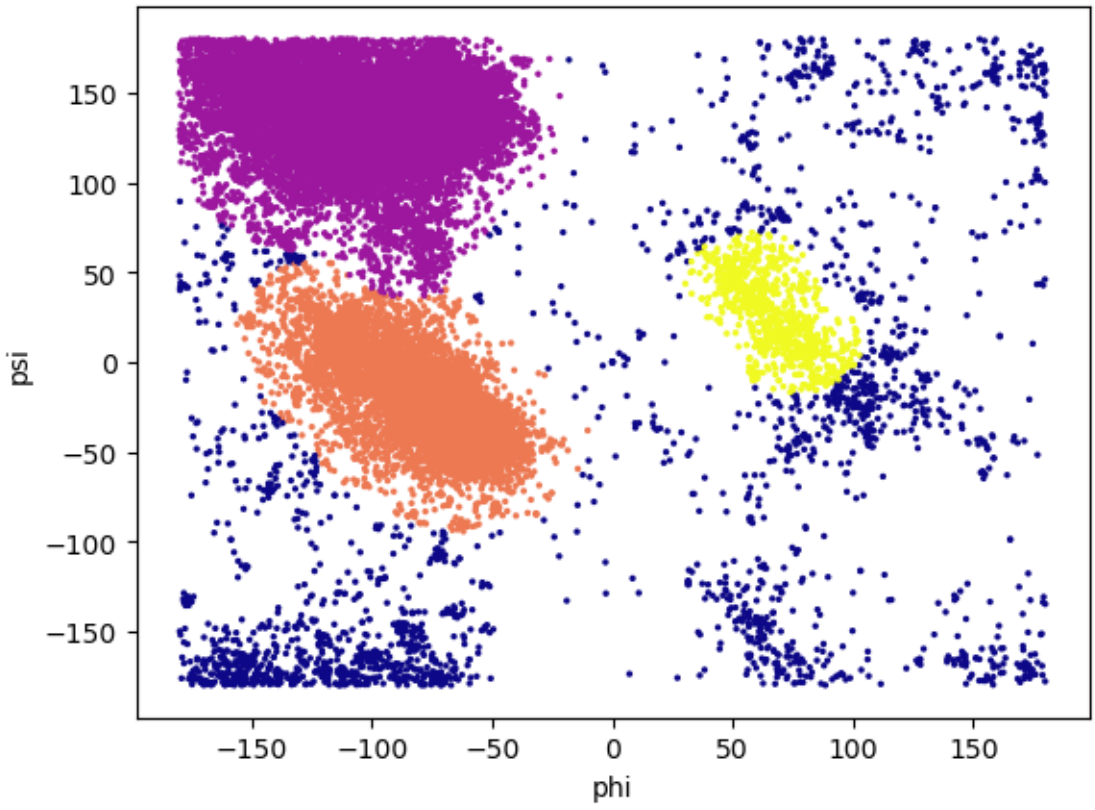
First, the value of min samples was decided. I used the information gained when plotting the heatmap. The gridsize of the heatmap correlates with how hot the hexagon is based on the number of plots in the hexagon. When gridsize was small then the graph was not hot. The graph started getting hotter as I reached 100. Therefore 300 is a good enough value to decide the core plots. Epsilon was selected with an eye test on the scatter plot. 20 was not a big enough distance so that the outliers would be added into the clusters like in k means and that was my main motivation.

3.2 Problem 3b)

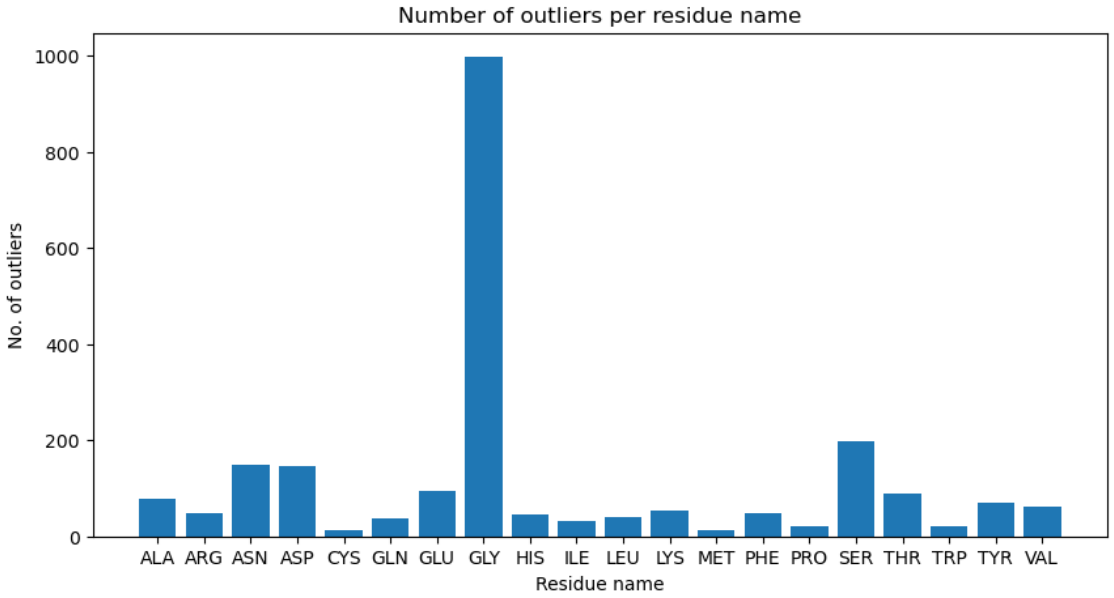
[78]:
```
db = DBSCAN(eps=20, min_samples=300)
df4 = db.fit_predict(df2)

n_clusters_ = len(set(df4)) - (1 if -1 in df4 else 0)
n_noise_ = list(df4).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
```

```
Estimated number of clusters: 3
Estimated number of noise points: 2258
```

[80]:
```
plt.scatter(df2['phi'], df2['psi'], c = df4, cmap= "plasma",  s=2)
plt.xlabel("phi")
plt.ylabel("psi")
plt.show()
```



[38]:
```
op = pd.DataFrame(df4, columns=['cluster_id'])
op['residue name'] = df['residue name']
op = op[(op['cluster_id'] == -1)].groupby('residue name').count().reset_index()

fig = plt.figure(figsize = (10, 5))
plt.bar(op['residue name'], op['cluster_id'])

plt.xlabel("Residue name")
plt.ylabel("No. of outliers")
plt.title("Number of outliers per residue name")
plt.show()
```



3.3 Problem 3c)

The biggest difference is of course that there exist outliers now which didn't exist in k means and more concentrated clusters are chosen.

3.4 Problem 3d)

Yes, small changes do not drastically affect the output of the dbscan. Sometimes maybe a new cluster is formed or in a different place but there do not usually happen when the changes to the parameters are small.

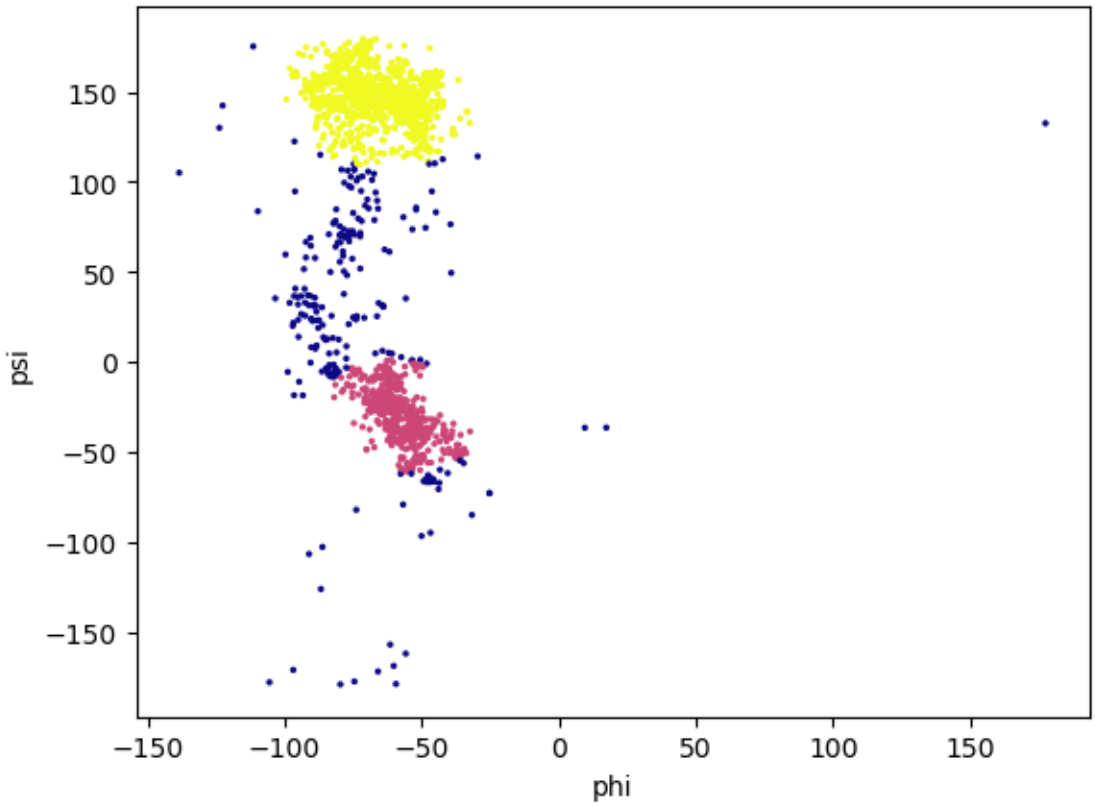
4 Problem 4

The clusters are drastically different since we are only taking into account a subset of the data and that would change the distribution of the data and the clusters formed. The only time when the clustering might look the same is if the subset is very similar to the original data or just one cluster (from the previous predicted clusters) is present in the data.

[81]:
```
db = DBSCAN(eps=20, min_samples=300)
df4 = db.fit_predict(dfpro)

n_clusters_ = len(set(df4)) - (1 if -1 in df4 else 0)
n_noise_ = list(df4).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
```

```
Estimated number of clusters: 2
Estimated number of noise points: 226
```

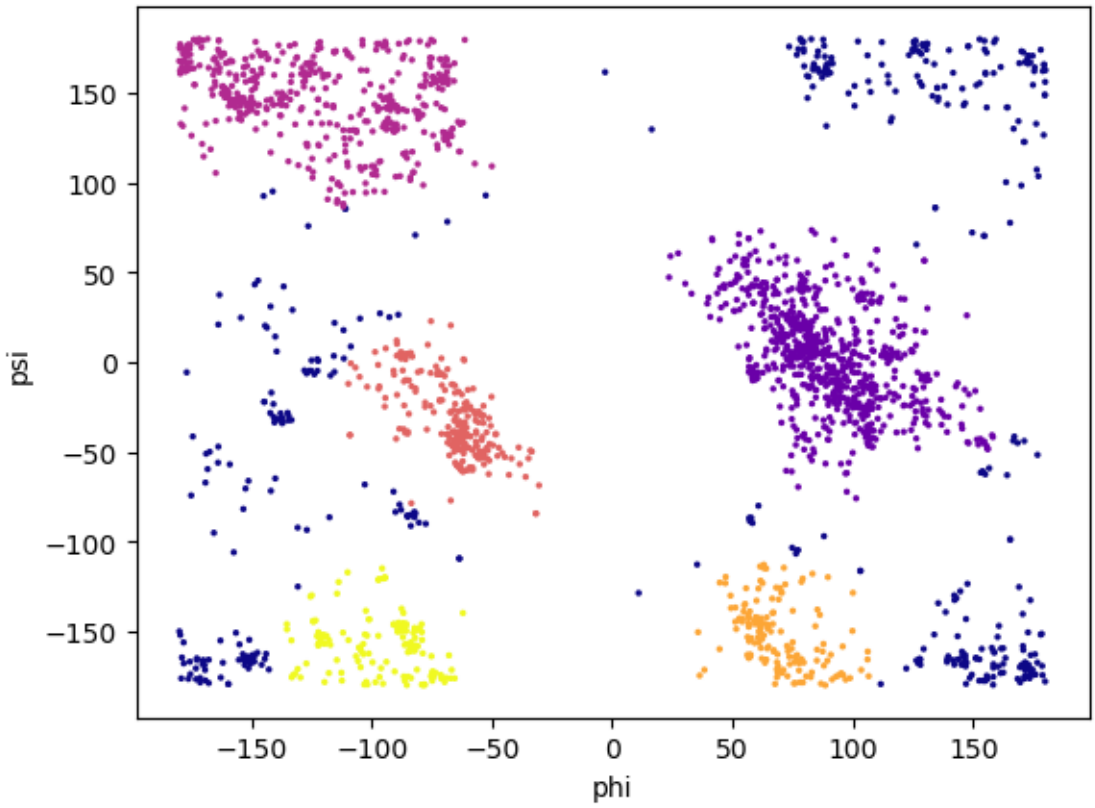
[82]:
```
plt.scatter(dfpro['phi'], dfpro['psi'], c = df4, cmap= "plasma",  s=2)
plt.xlabel("phi")
plt.ylabel("psi")
plt.show()
```



[83]:
```
db = DBSCAN(eps=30, min_samples=100)
df4 = db.fit_predict(dfgly)

n_clusters_ = len(set(df4)) - (1 if -1 in df4 else 0)
n_noise_ = list(df4).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
```

```
Estimated number of clusters: 5
Estimated number of noise points: 414
```

[86]:
```
plt.scatter(dfgly['phi'], dfgly['psi'], c = df4, cmap= "plasma",  s=2)
plt.xlabel("phi")
plt.ylabel("psi")
plt.show()
```


150
100
50
0
−50
−100
−150
psi
−150
−100
−50
0
50
100
150
phi