assignment9

April 15, 2024

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/home/oneautumleaf/.local/lib/python3.10/site-
packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import
Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g.
as a system package and as a pip package). As a result, the 3D projection is not
available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions
of "
```

[ ]:
```
df = pd.read_csv('datasets/penguins.csv')
df
```

[ ]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
..	…	…	…	…	…	…	…
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0		
343	Gentoo	Biscoe	49.9	16.1	213.0		

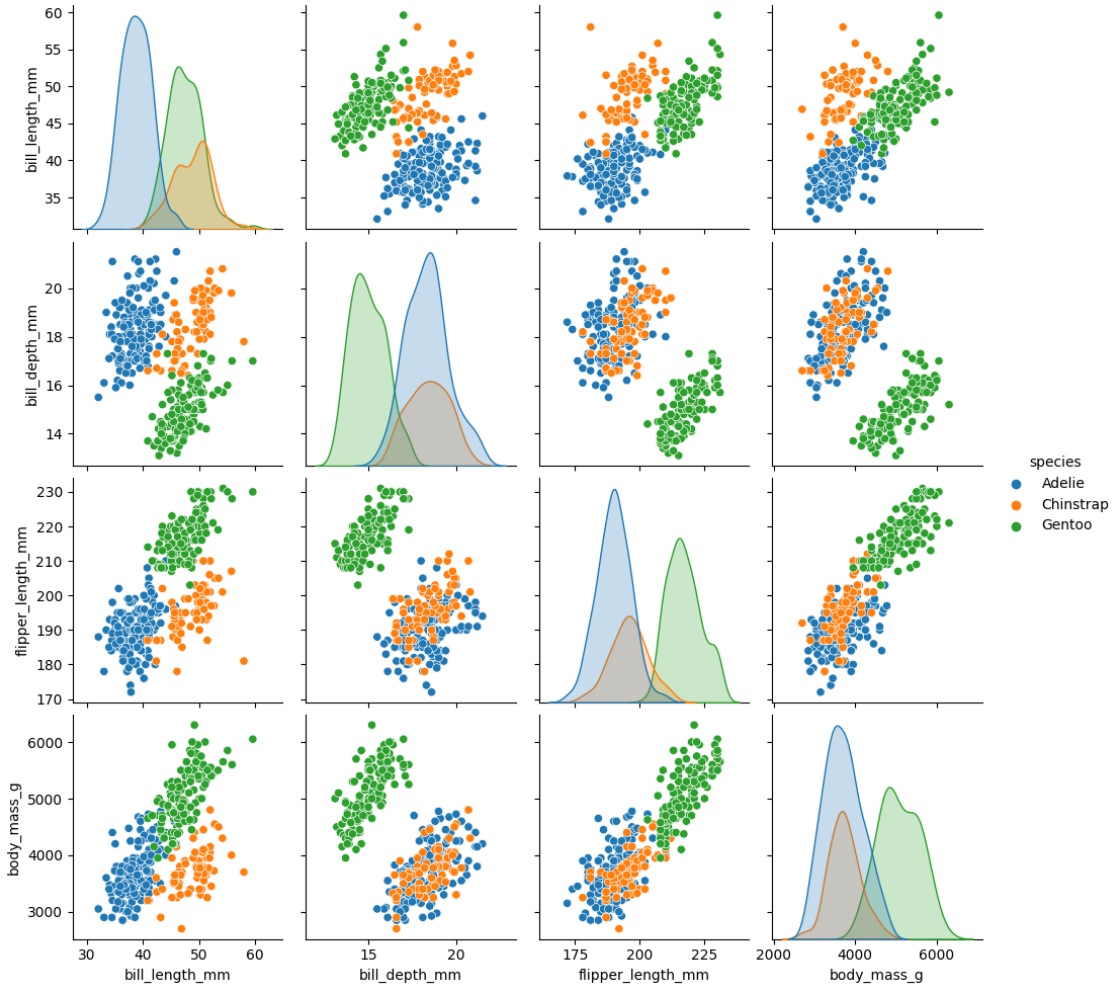
```
342       5200.0  FEMALE
343       5400.0    MALE

[344 rows x 7 columns]
```

[ ]:
```
df.shape
```

[ ]:
```
(344, 7)
```

[ ]:
```
sns.pairplot(df, hue='species')
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f89b2825180>
```



Data Preprocessing

[ ]:
```
df.isna().sum()
```

[ ]:
```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

[ ]:
```
df.dtypes
```

[ ]:
```
species              object
island               object
bill_length_mm      float64
bill_depth_mm       float64
flipper_length_mm   float64
body_mass_g         float64
sex                  object
dtype: object
```

[ ]:
```
df.species.unique()
```

[ ]:
```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

[ ]:
```
df.island.unique()
```

[ ]:
```
array(['Torgersen', 'Biscoe', 'Dream'], dtype=object)
```

[ ]:
```
df.sex.unique()
```

[ ]:
```
array(['MALE', 'FEMALE', nan], dtype=object)
```

[ ]:
```
X = df.drop('species', axis=1)
y = df['species']
```

[ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

categorical_features = [col for col in X.columns if X[col].dtype == 'object']
numeric_features = [col for col in X.columns if X[col].dtype != 'object']
numeric_transformer = Pipeline(
    [
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
```

```
    ]
)

categorical_transformer = Pipeline(
    [
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('encoder', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
    ]
)

preprocessor = ColumnTransformer(
    [
        ('numeric', numeric_transformer, numeric_features),
        ('categorical', categorical_transformer, categorical_features)
    ],
)
```

[ ]:
```
X_t = pd.DataFrame(preprocessor.fit_transform(X), columns=preprocessor.
↪get_feature_names_out())
X_t
```

[ ]:

	numeric__bill_length_mm	numeric__bill_depth_mm	\
0	-0.887622	0.787289	
1	-0.814037	0.126114	
2	-0.666866	0.431272	
3	0.096581	0.075255	
4	-1.329133	1.092447	
..	…	…	
339	0.096581	0.075255	
340	0.528894	-1.450534	
341	1.191161	-0.738500	
342	0.234553	-1.196236	
343	1.099179	-0.535061	

	numeric__flipper_length_mm	numeric__body_mass_g	\
0	-1.420541	-0.564625	
1	-1.063485	-0.502010	
2	-0.420786	-1.190773	
3	-0.277964	-0.188936	
4	-0.563608	-0.940314	
..	…	…	
339	-0.277964	-0.188936	
340	1.007436	0.812900	
341	1.507314	1.939966	
342	0.793203	1.251204	
343	0.864614	1.501663	

```
     categorical__island_Biscoe  categorical__island_Dream  \
0                            0.0                        0.0
1                            0.0                        0.0
2                            0.0                        0.0
3                            0.0                        0.0
4                            0.0                        0.0
..                           …                          …
339                          1.0                        0.0
340                          1.0                        0.0
341                          1.0                        0.0
342                          1.0                        0.0
343                          1.0                        0.0

     categorical__island_Torgersen  categorical__sex_FEMALE  \
0                               1.0                      0.0
1                               1.0                      1.0
2                               1.0                      1.0
3                               1.0                      0.0
4                               1.0                      1.0
..                              …                        …
339                             0.0                      0.0
340                             0.0                      1.0
341                             0.0                      0.0
342                             0.0                      1.0
343                             0.0                      0.0

     categorical__sex_MALE
0                      1.0
1                      0.0
2                      0.0
3                      1.0
4                      0.0
..                     …
339                    1.0
340                    0.0
341                    1.0
342                    0.0
343                    1.0

[344 rows x 9 columns]
```

[ ]:
```
X_t.shape
```

[ ]:
```
(344, 9)
```

[ ]:
```
X.shape
```

[ ]: (344, 6)

[ ]:
```
preprocessor.get_feature_names_out()
```

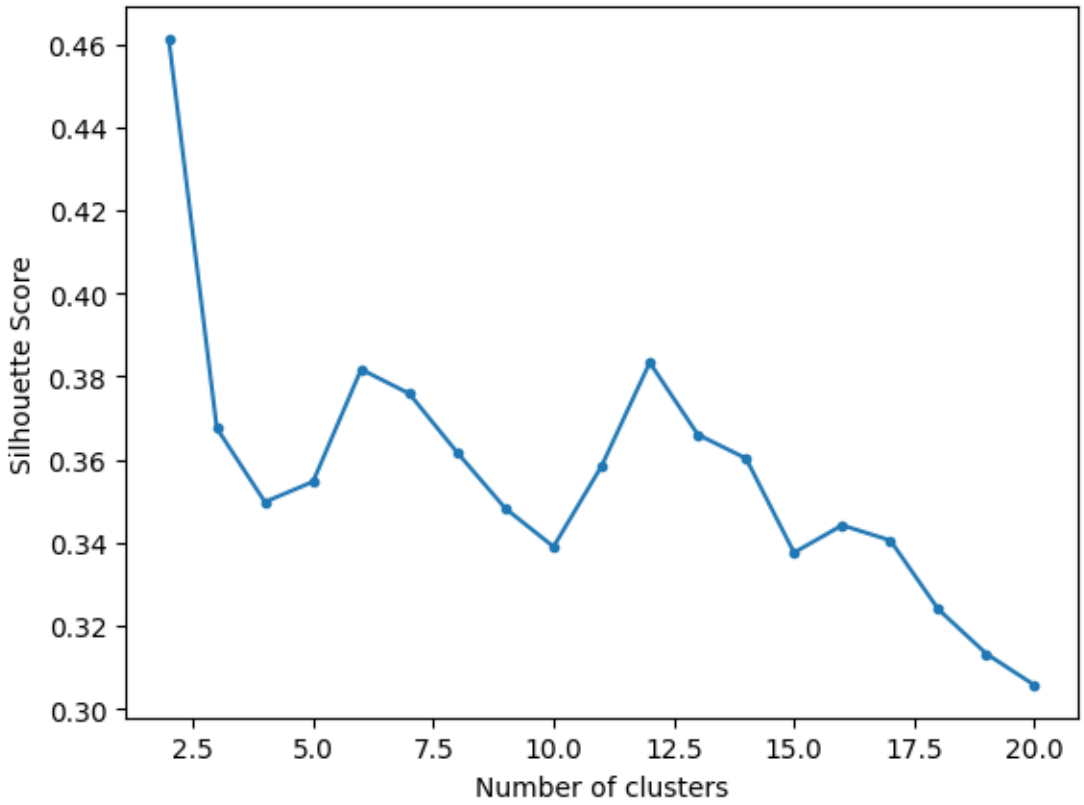
[ ]:
```
array(['numeric__bill_length_mm', 'numeric__bill_depth_mm',
       'numeric__flipper_length_mm', 'numeric__body_mass_g',
       'categorical__island_Biscoe', 'categorical__island_Dream',
       'categorical__island_Torgersen', 'categorical__sex_FEMALE',
       'categorical__sex_MALE'], dtype=object)
```

[ ]:
```
X = pd.DataFrame(preprocessor.fit_transform(X), columns=preprocessor.
  ↪get_feature_names_out())
```

KMeans

[ ]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

[ ]:
```
max_num_clusters = 20
scores = []
for i in range(2, max_num_clusters + 1):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init='auto')
    kmeans_clusters = kmeans.fit_predict(X)
    scores.append(silhouette_score(X, kmeans_clusters))

plt.plot(range(2, max_num_clusters + 1), scores, marker='.')
plt.xlabel("Number of clusters")
plt.ylabel("Silhouette Score")
plt.show()
```

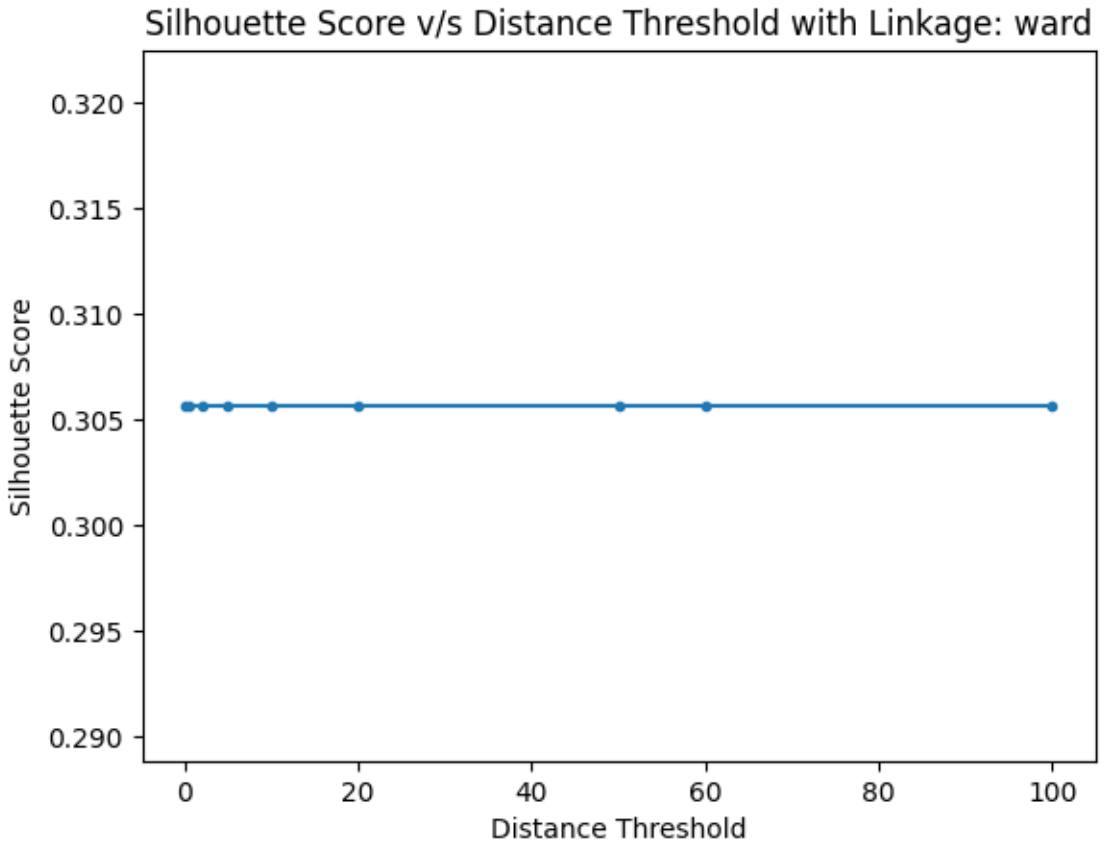


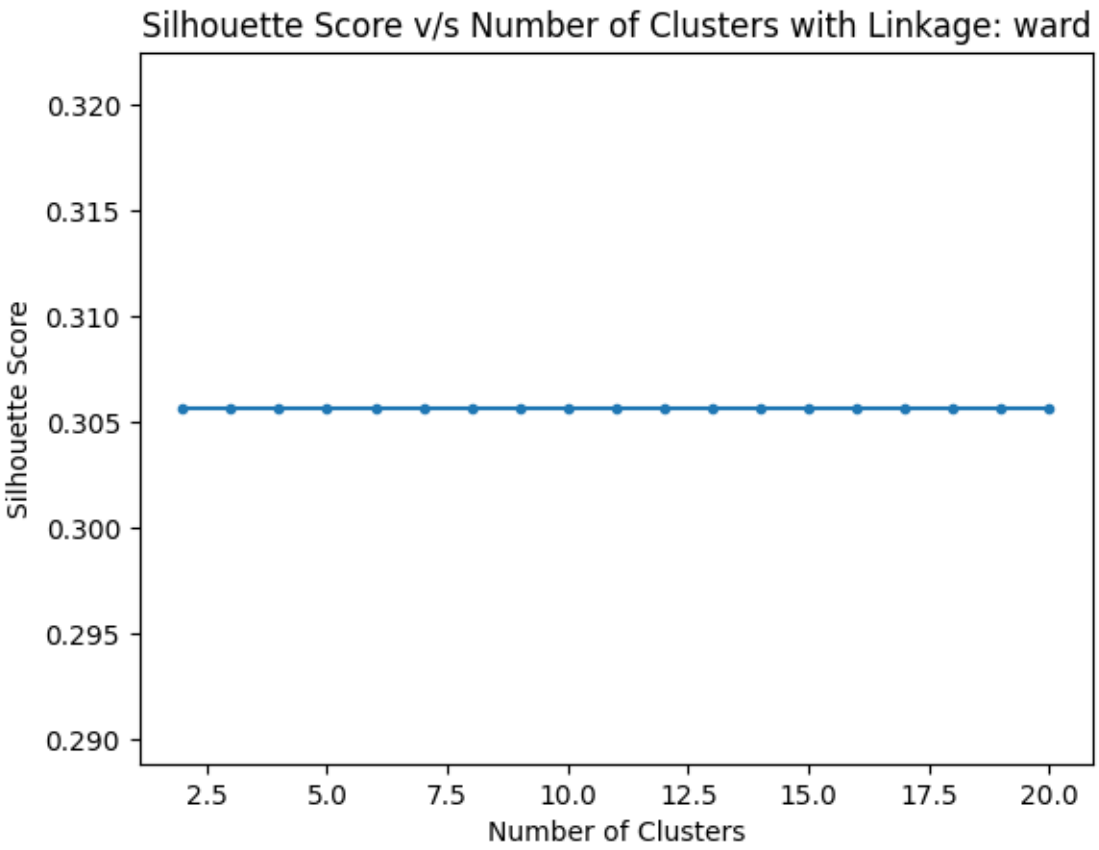
Agglomerative Clustering

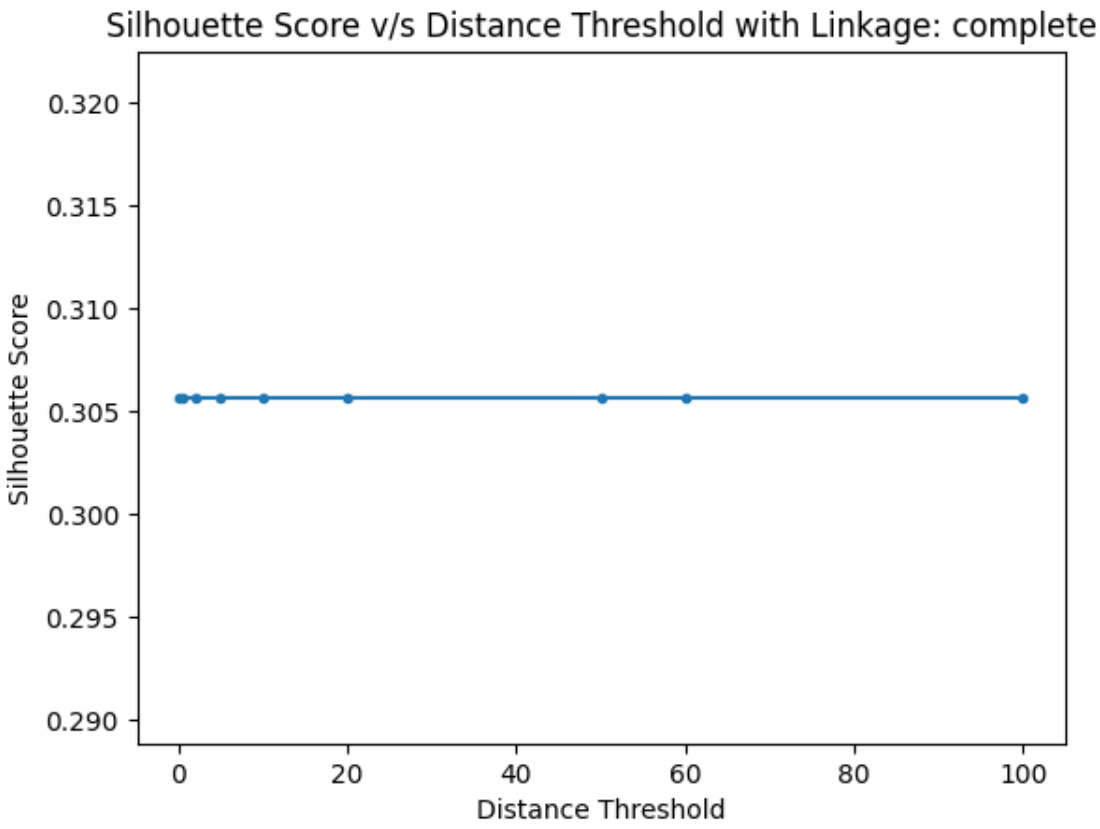
[ ]:
```
from sklearn.cluster import AgglomerativeClustering
```

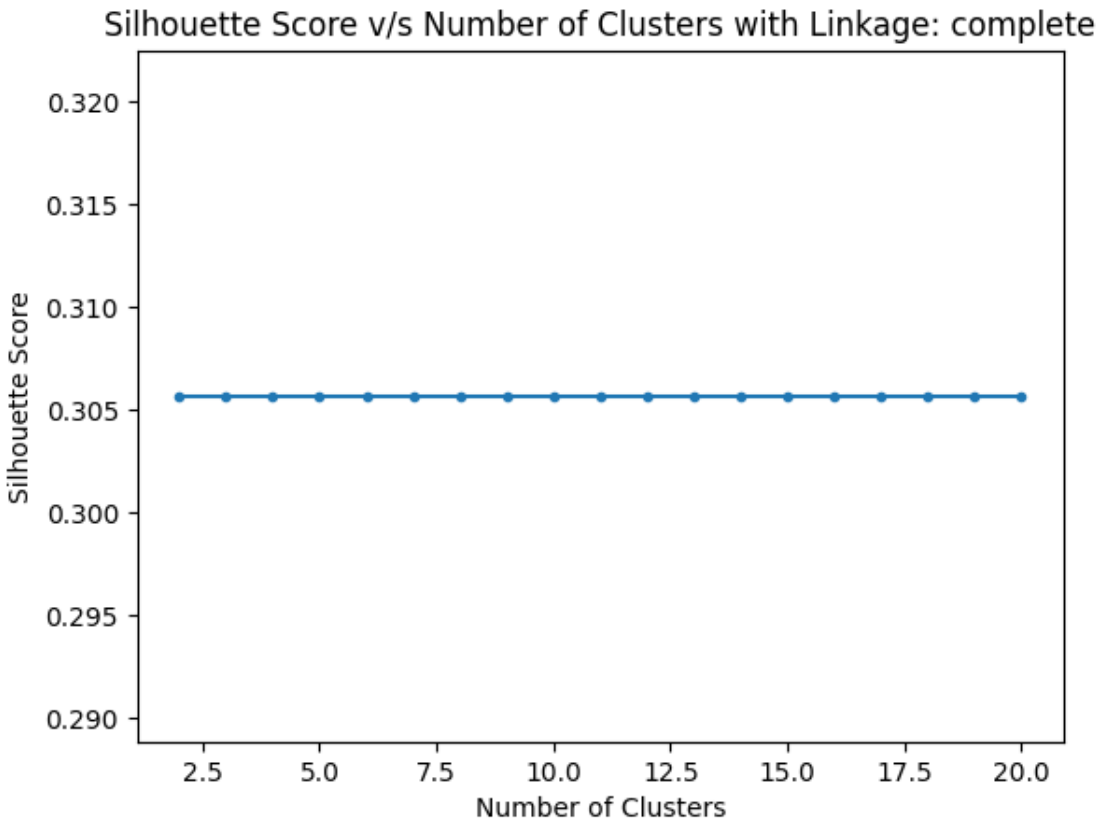
[ ]:
```
for linkage in ['ward', 'complete', 'average', 'single']:
    dist_arr = [.1, .4, 2, 5, 10, 20, 50, 60, 100]
    scores = []
    for dist in dist_arr:
        agg = AgglomerativeClustering(n_clusters=None, distance_threshold=dist,␣
 ↪linkage=linkage)
        clusters = agg.fit_predict(X)
        scores.append(silhouette_score(X, kmeans_clusters))

    plt.plot(dist_arr, scores, marker='.')
    plt.xlabel("Distance Threshold")
    plt.ylabel("Silhouette Score")
    plt.title(f"Silhouette Score v/s Distance Threshold with Linkage:␣
 ↪{linkage}")
    plt.show()

    scores = []
```

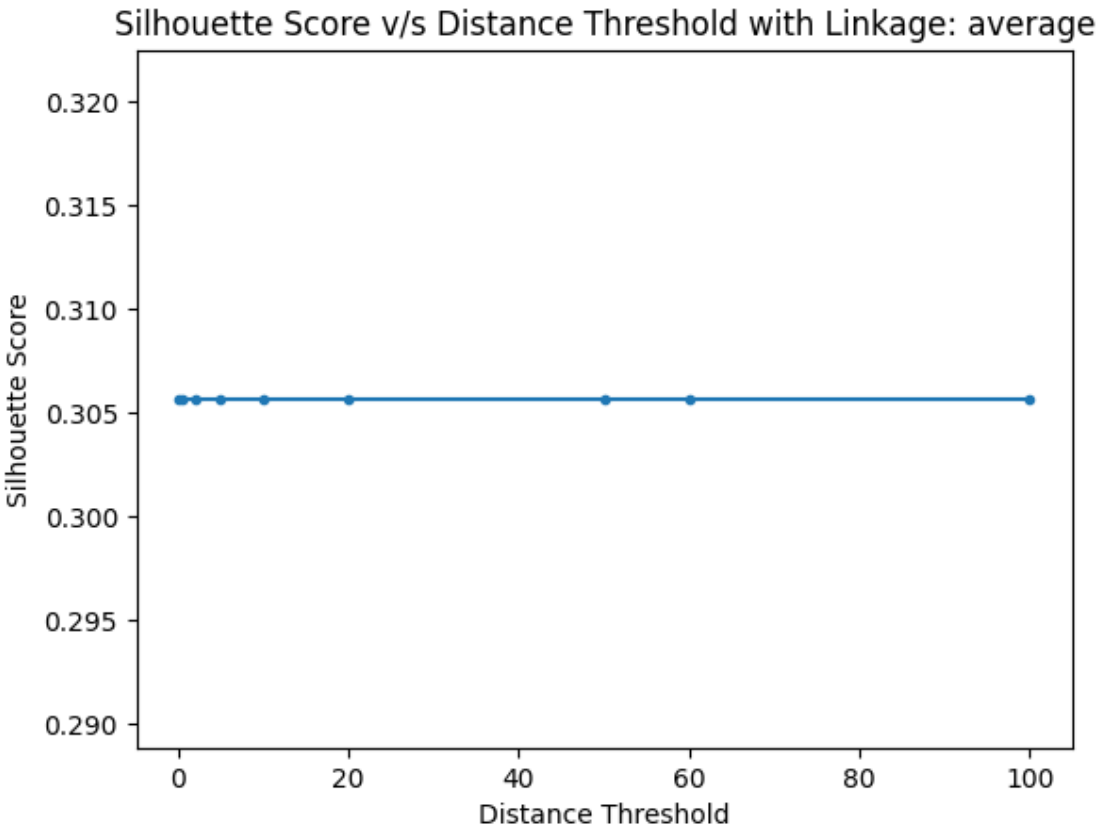
```
    for i in range(2, max_num_clusters + 1):
        agg = AgglomerativeClustering(n_clusters=i, linkage=linkage)
        clusters = agg.fit_predict(X)
        scores.append(silhouette_score(X, kmeans_clusters))

    plt.plot(range(2, max_num_clusters + 1), scores, marker='.')
    plt.xlabel("Number of Clusters")
    plt.ylabel("Silhouette Score")
    plt.title(f"Silhouette Score v/s Number of Clusters with Linkage:␣
↪{linkage}")
    plt.show()
```

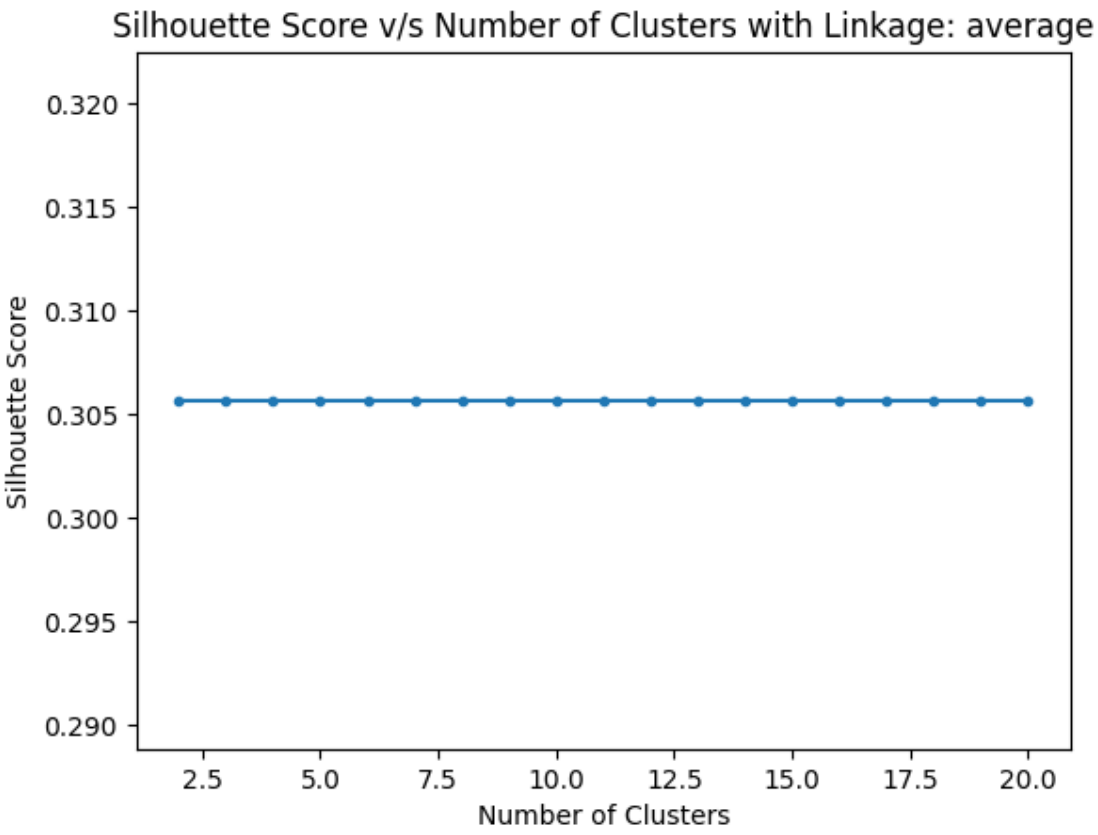


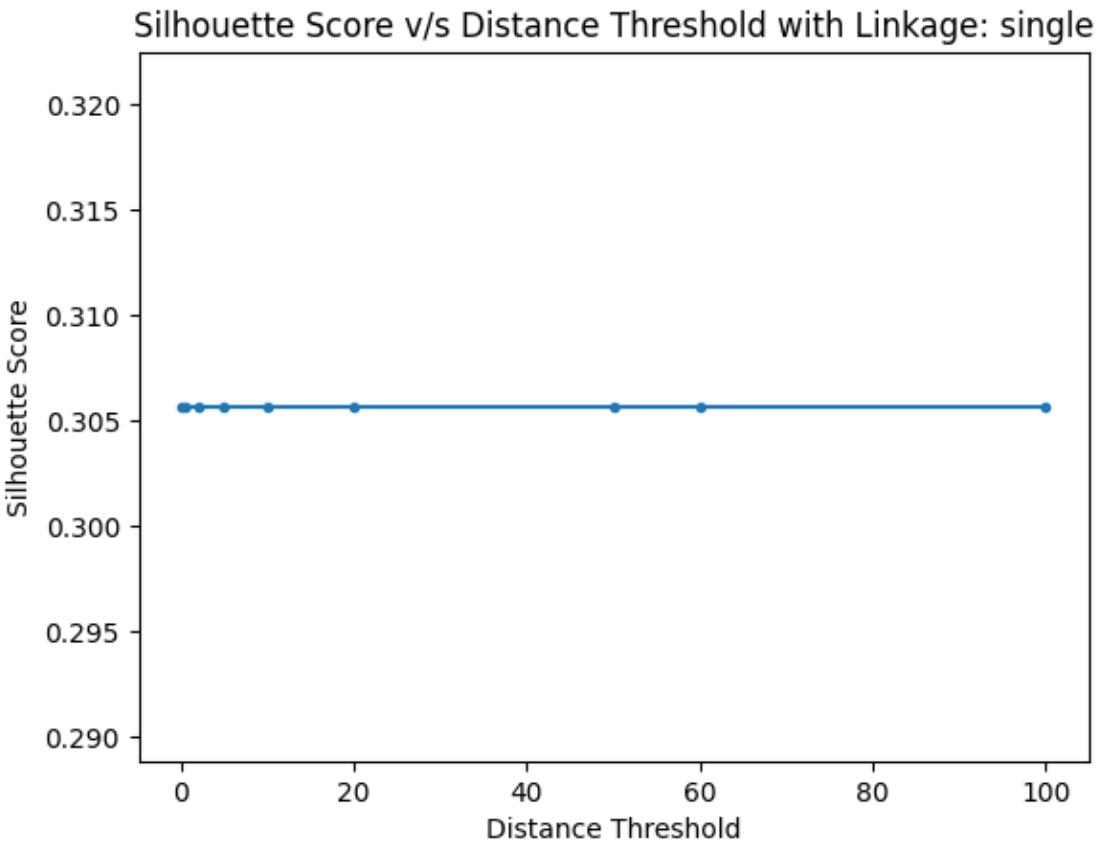

Silhouette Score v/s Number of Clusters with Linkage: ward
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Clusters

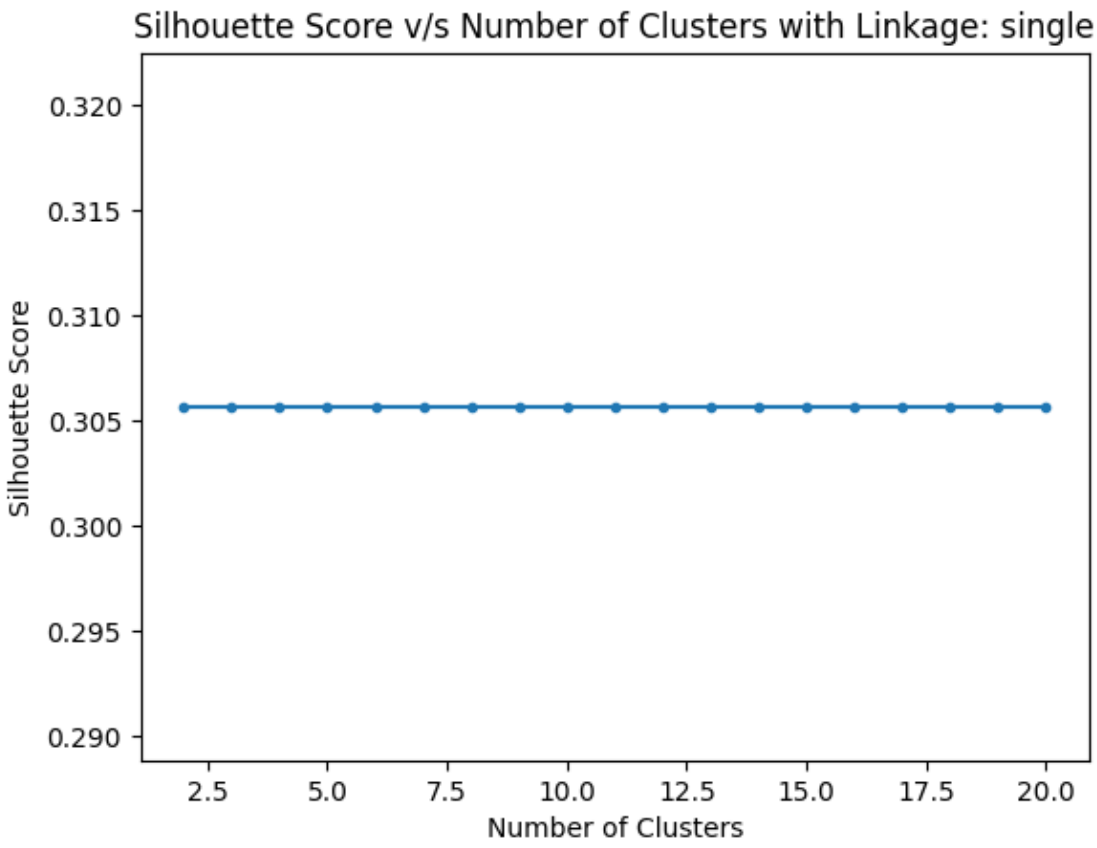

Silhouette Score v/s Distance Threshold with Linkage: complete
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
0
20
40
60
80
100
Distance Threshold


Silhouette Score v/s Number of Clusters with Linkage: complete
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Clusters


Silhouette Score v/s Distance Threshold with Linkage: average
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
0
20
40
60
80
100
Distance Threshold


Silhouette Score v/s Number of Clusters with Linkage: average
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Clusters


Silhouette Score v/s Distance Threshold with Linkage: single
Silhouette Score
0.320
0.315
0.310
0.305
0.300
0.295
0.290
0
20
40
60
80
100
Distance Threshold

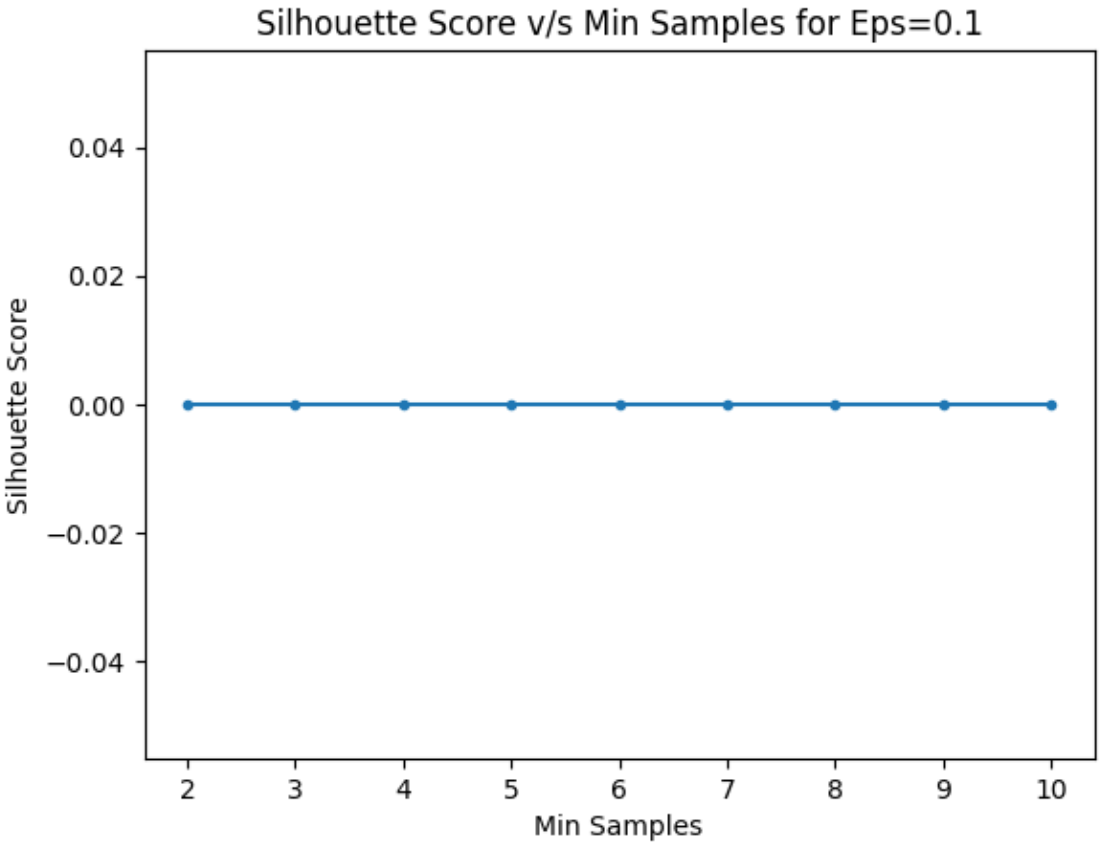


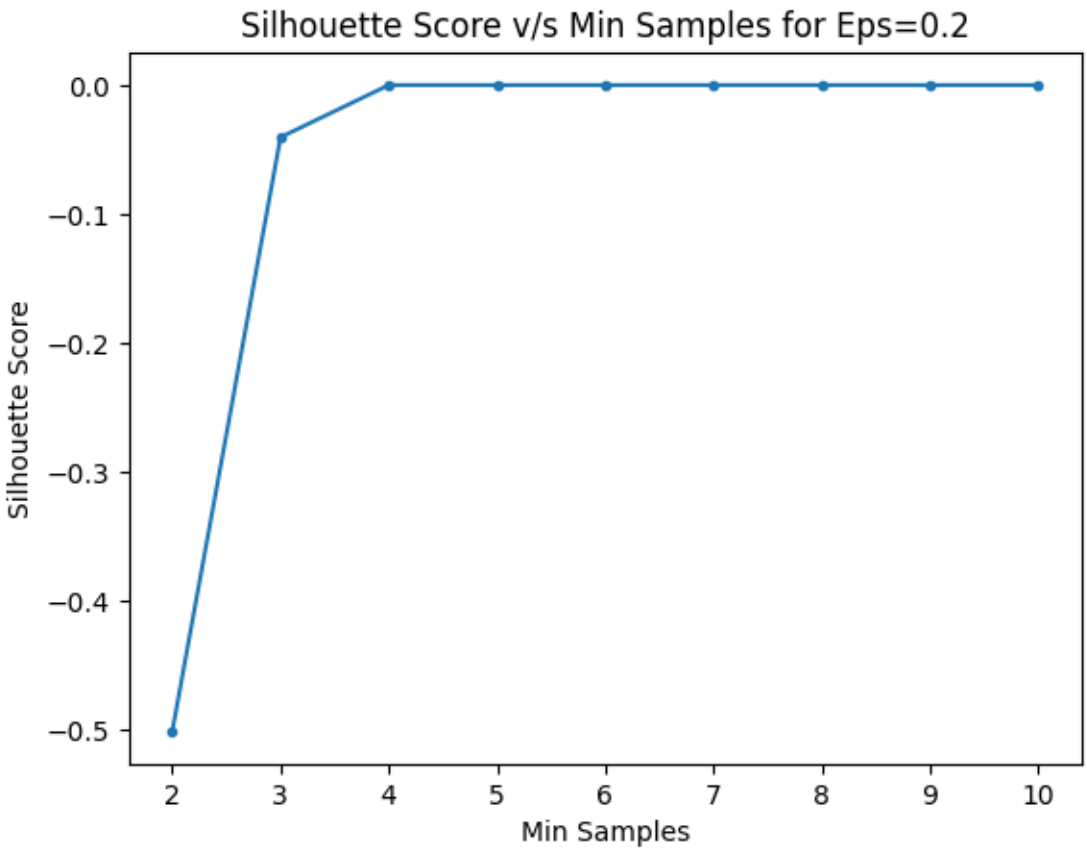
DBSCAN Clustering

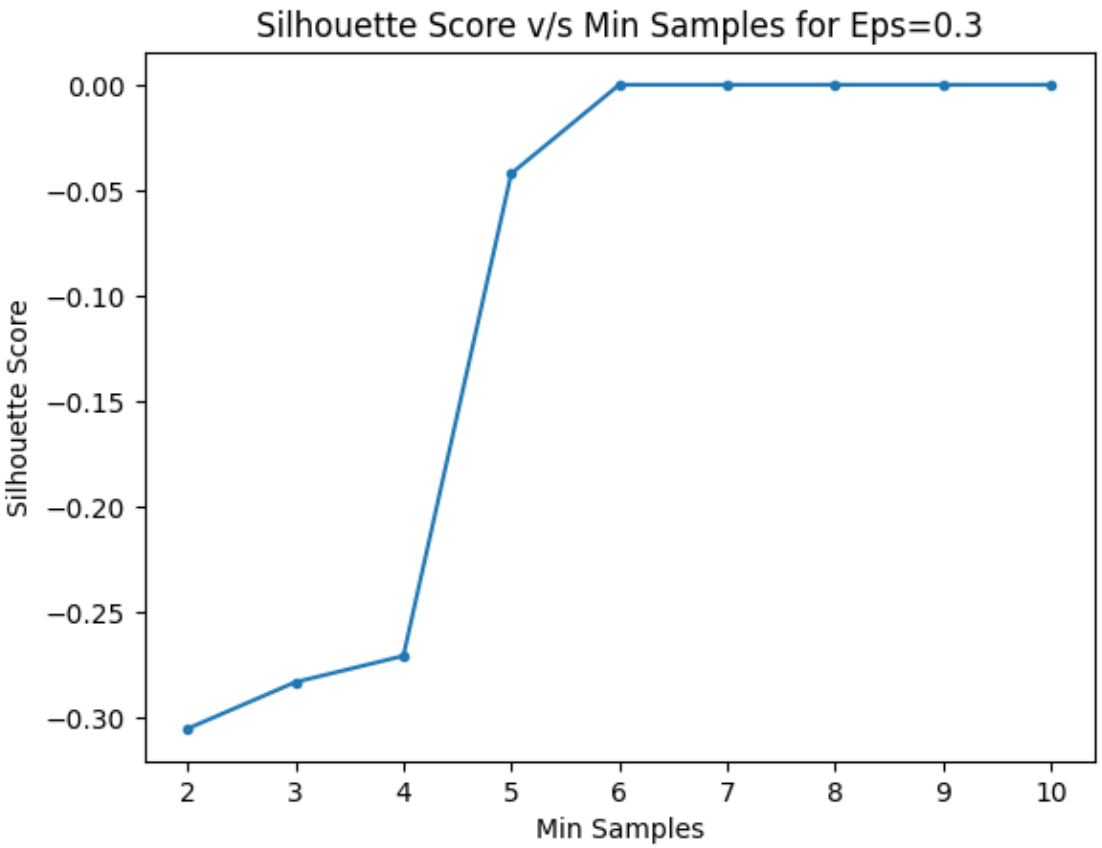
[ ]:
```
from sklearn.cluster import DBSCAN
```

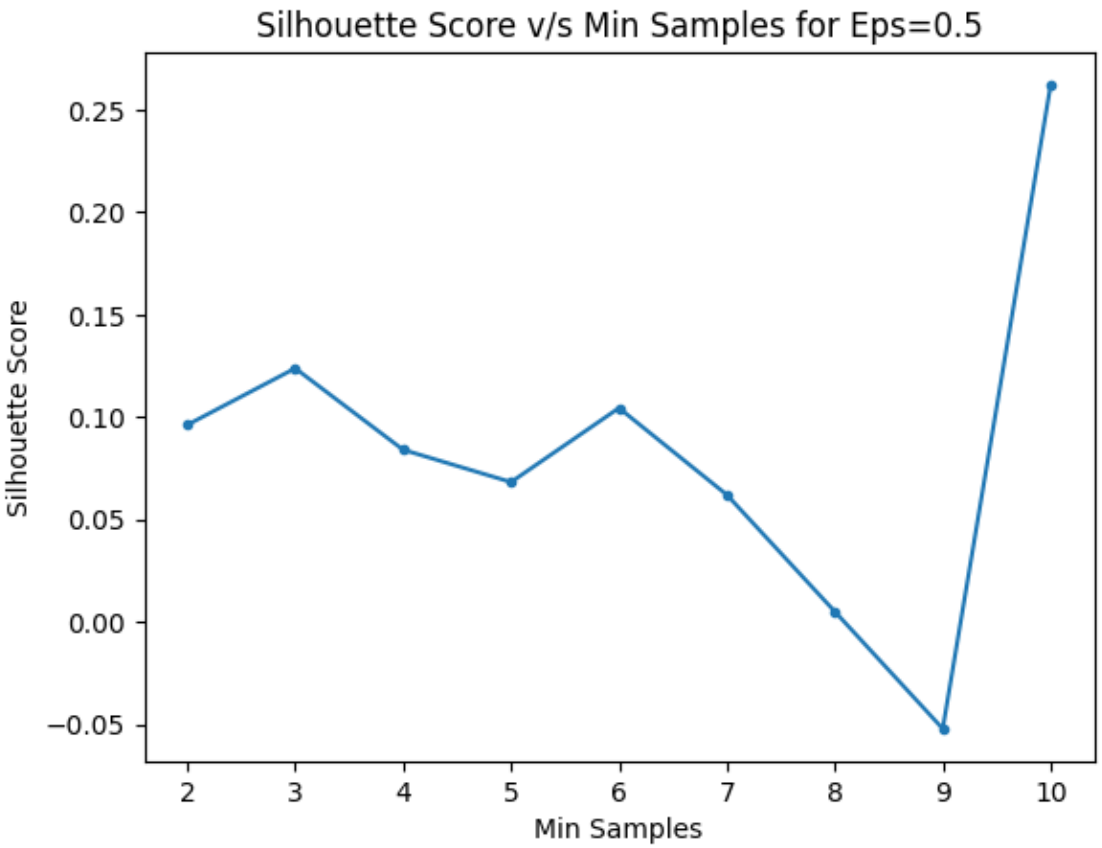
[ ]:
```
eps_arr = [0.1, .2, .3, .5, .8, 1, 1.2, 1.5, 1.8, 2, 2.5]
for eps in eps_arr:
    scores = []
    max_min_samples = 10
    for min_samples in range(2, max_min_samples + 1):
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        clusters = dbscan.fit_predict(X)
        try:
            scores.append(silhouette_score(X, clusters))
        except ValueError:
            scores.append(0)

    plt.plot(range(2, max_min_samples + 1), scores, marker='.')
    plt.xlabel("Min Samples")
    plt.ylabel("Silhouette Score")
    plt.title(f"Silhouette Score v/s Min Samples for Eps={eps}")
```

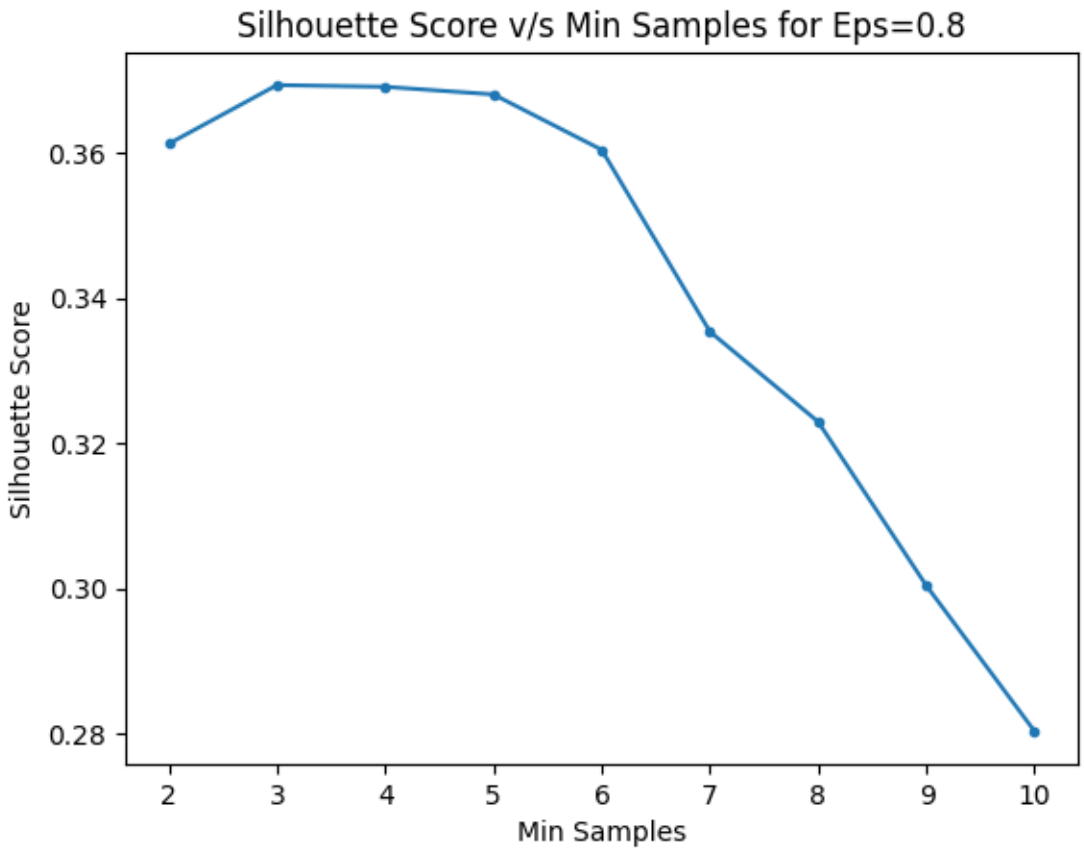
```
plt.show()
```

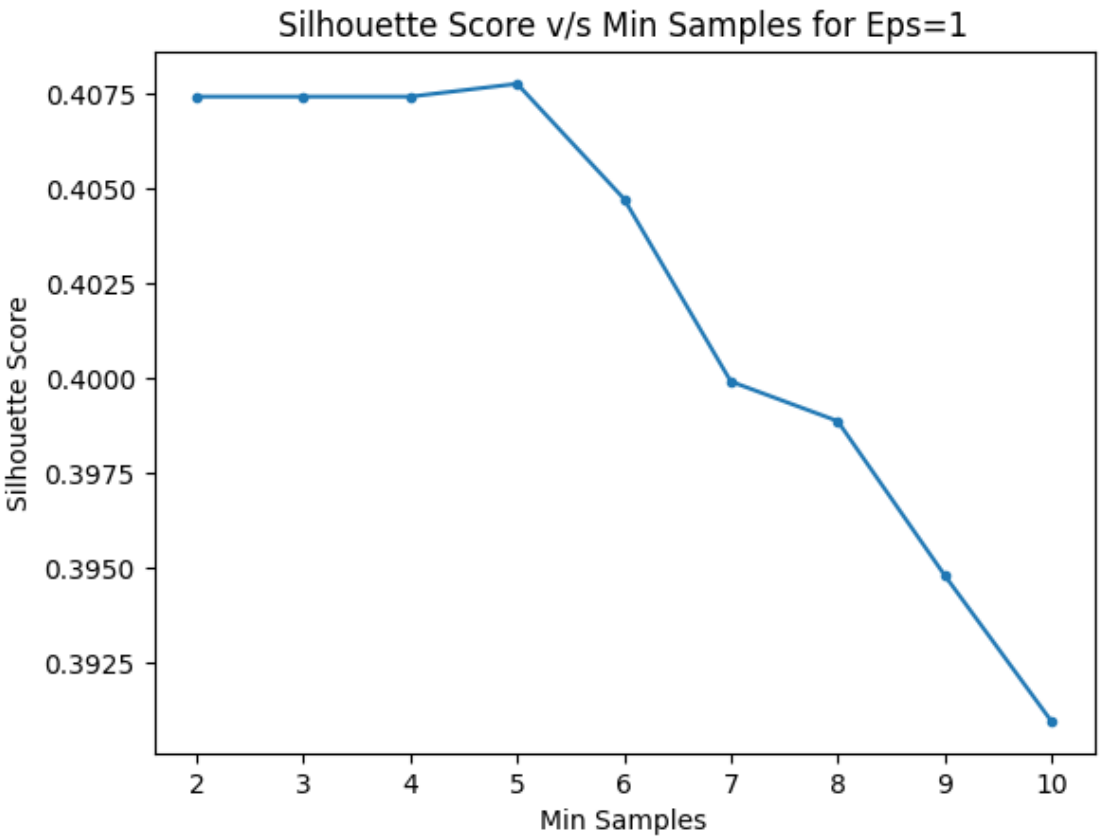

Silhouette Score v/s Min Samples for Eps=0.1
Silhouette Score
0.04
0.02
0.00
−0.02
−0.04
2
3
4
5
6
7
8
9
10
Min Samples

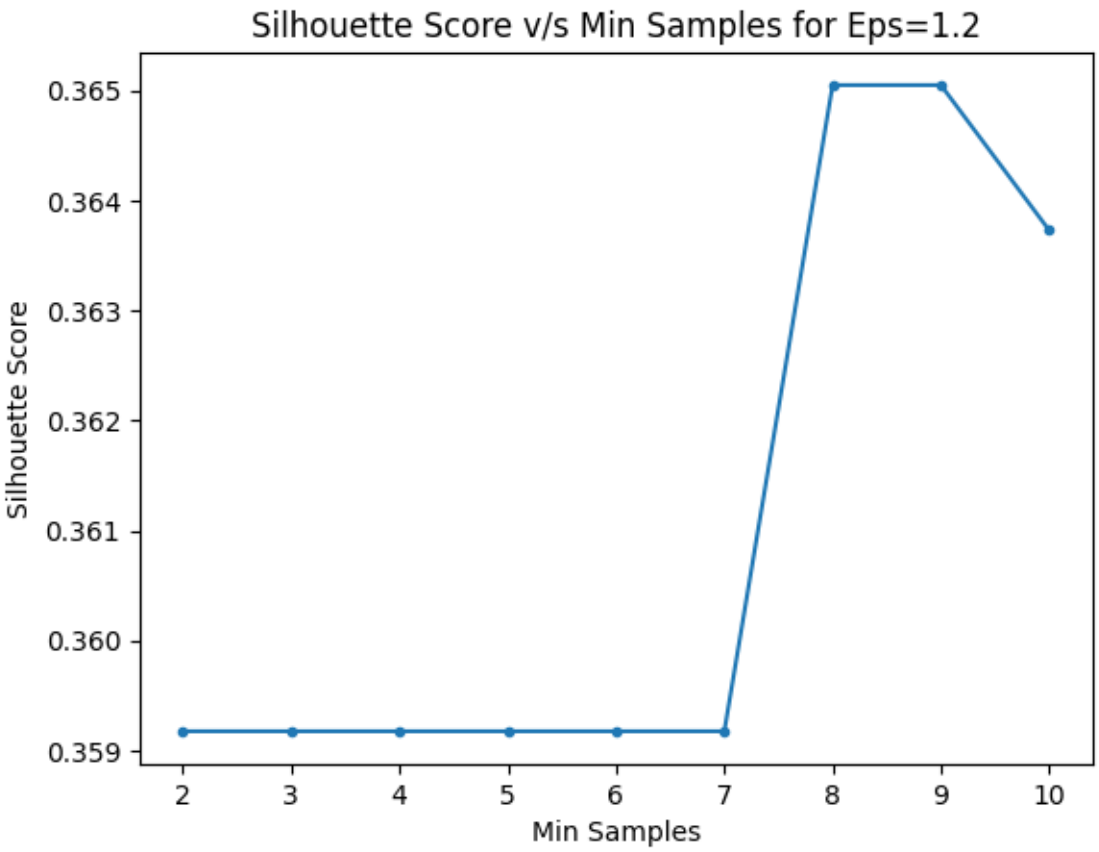

Silhouette Score v/s Min Samples for Eps=0.2
Silhouette Score
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
2
3
4
5
6
7
8
9
10
Min Samples

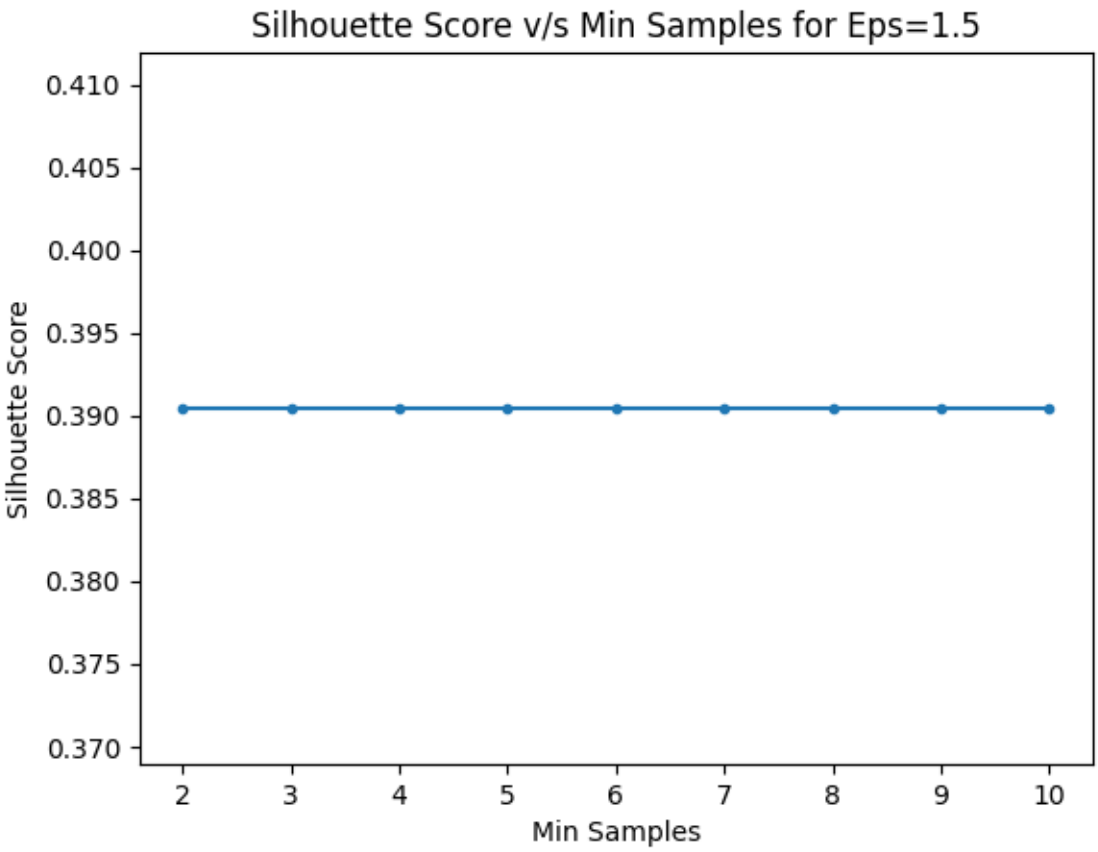

Silhouette Score v/s Min Samples for Eps=0.3
Silhouette Score
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
2
3
4
5
6
7
8
9
10
Min Samples

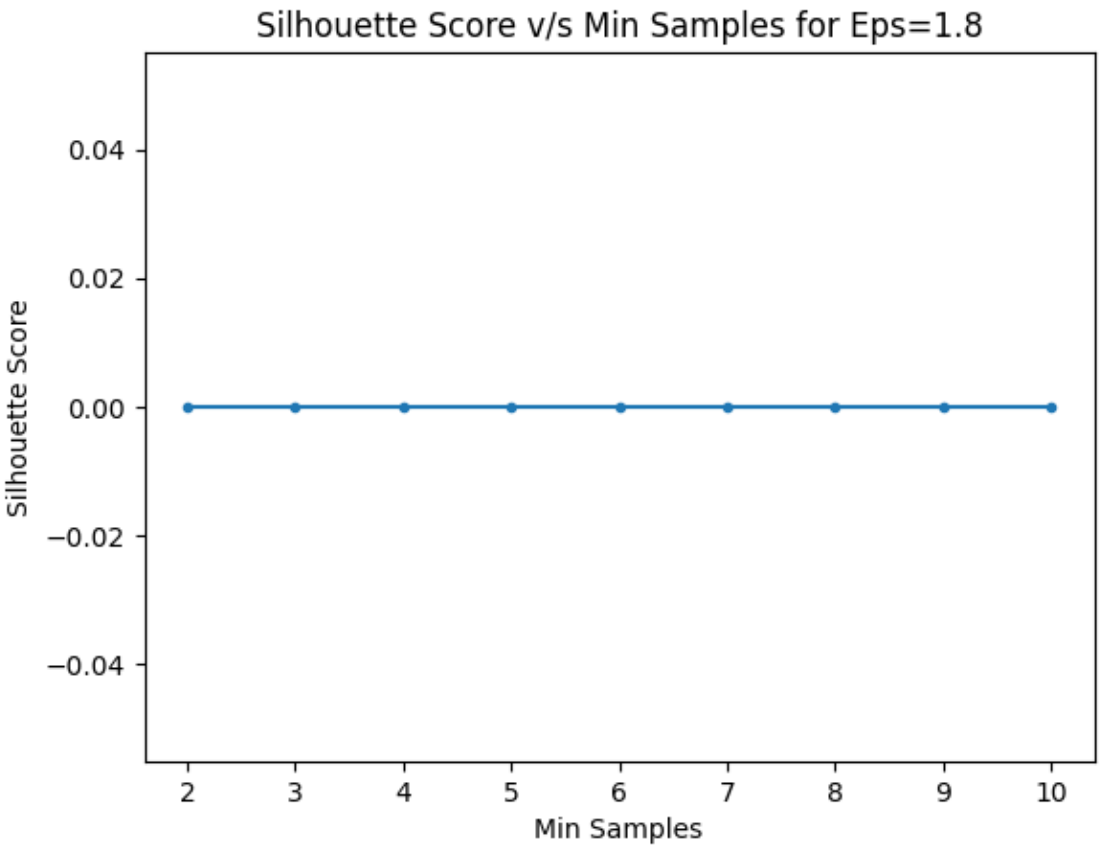

Silhouette Score v/s Min Samples for Eps=0.5
Silhouette Score
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
2
3
4
5
6
7
8
9
10
Min Samples

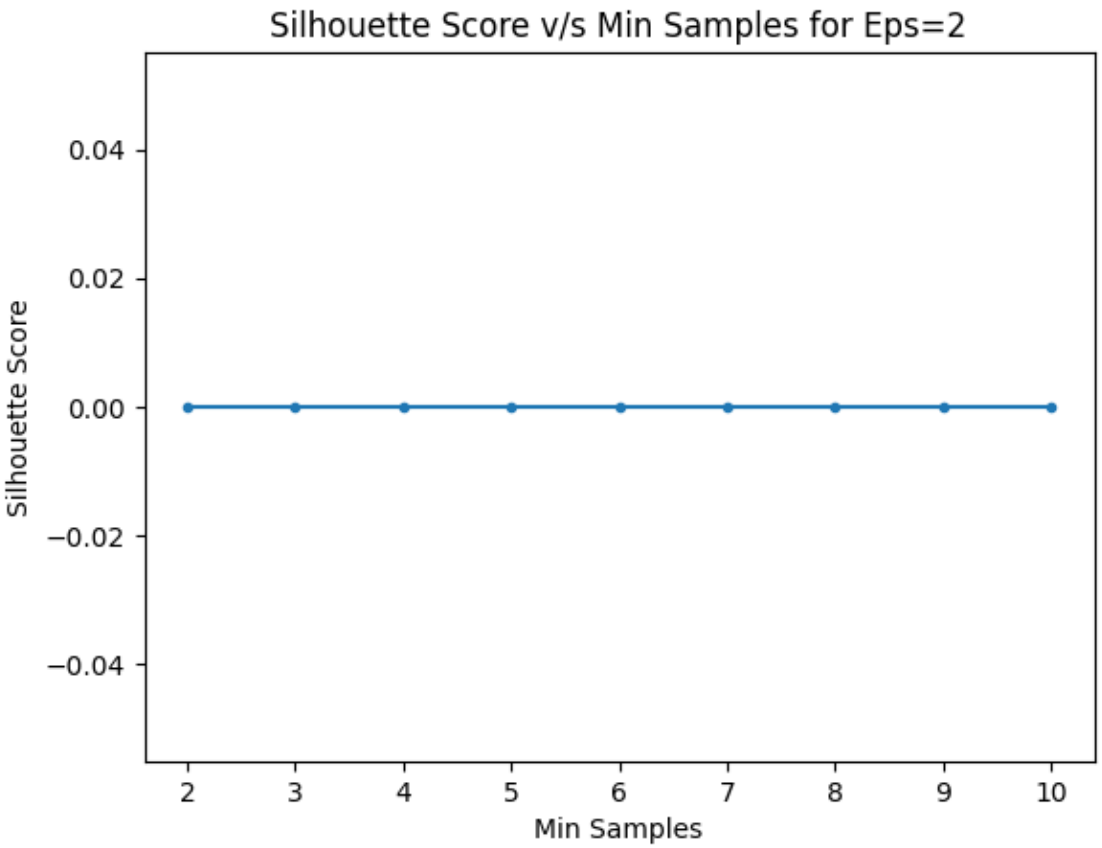

Silhouette Score v/s Min Samples for Eps=0.8
Silhouette Score
0.36
0.34
0.32
0.30
0.28
2
3
4
5
6
7
8
9
10
Min Samples

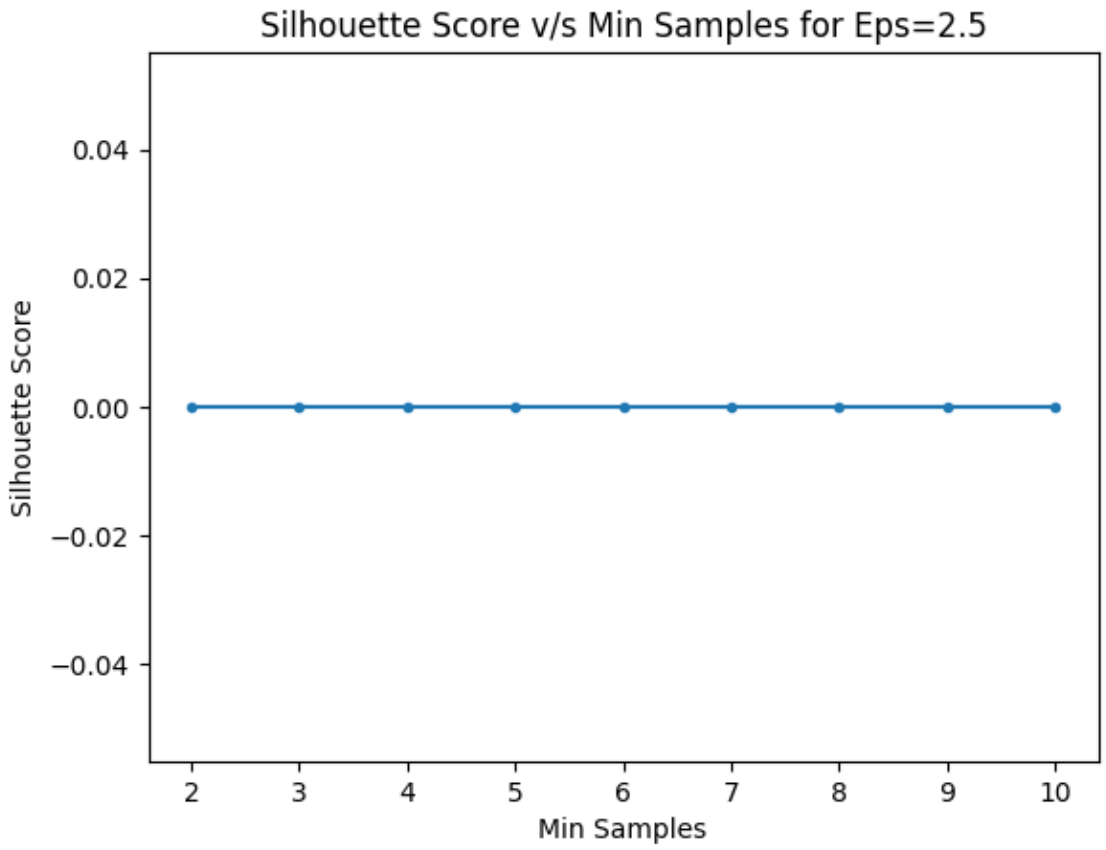

Silhouette Score v/s Min Samples for Eps=1
Silhouette Score
0.4075
0.4050
0.4025
0.4000
0.3975
0.3950
0.3925
2
3
4
5
6
7
8
9
10
Min Samples


Silhouette Score v/s Min Samples for Eps=1.2
Silhouette Score
0.365
0.364
0.363
0.362
0.361
0.360
0.359
2
3
4
5
6
7
8
9
10
Min Samples


Silhouette Score v/s Min Samples for Eps=1.5
Silhouette Score
0.410
0.405
0.400
0.395
0.390
0.385
0.380
0.375
0.370
2
3
4
5
6
7
8
9
10
Min Samples


Silhouette Score v/s Min Samples for Eps=1.8
Silhouette Score
0.04
0.02
0.00
−0.02
−0.04
2
3
4
5
6
7
8
9
10
Min Samples


Silhouette Score v/s Min Samples for Eps=2
Silhouette Score
0.04
0.02
0.00
−0.02
−0.04
2
3
4
5
6
7
8
9
10
Min Samples



Jaccard Score

[ ]:
```
from sklearn.metrics import jaccard_score
from sklearn.preprocessing import LabelEncoder
```

[ ]:
```
y_enc = LabelEncoder().fit_transform(y)
```

[ ]:
```
kmeans_clusters = KMeans(n_clusters=3).fit_predict(X)
```

```
/home/oneautumleaf/.local/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[ ]:
```
import copy
yy = copy.deepcopy(y_enc)
```

[ ]:
```
yy[yy == 1] = 5
yy[yy == 2] = 6
```

```
yy[yy == 5] = 2
yy[yy == 6] = 1
```

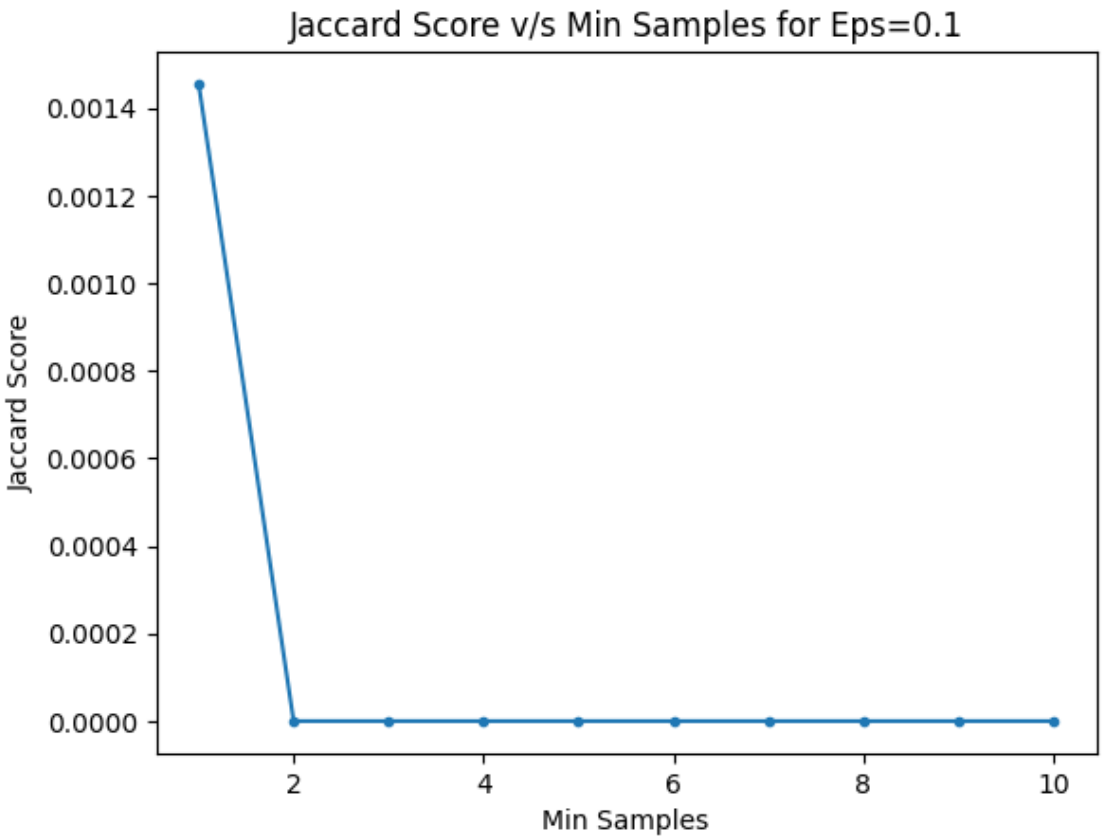
[ ]:
```
jaccard_score(y_enc, kmeans_clusters, average='micro')
```

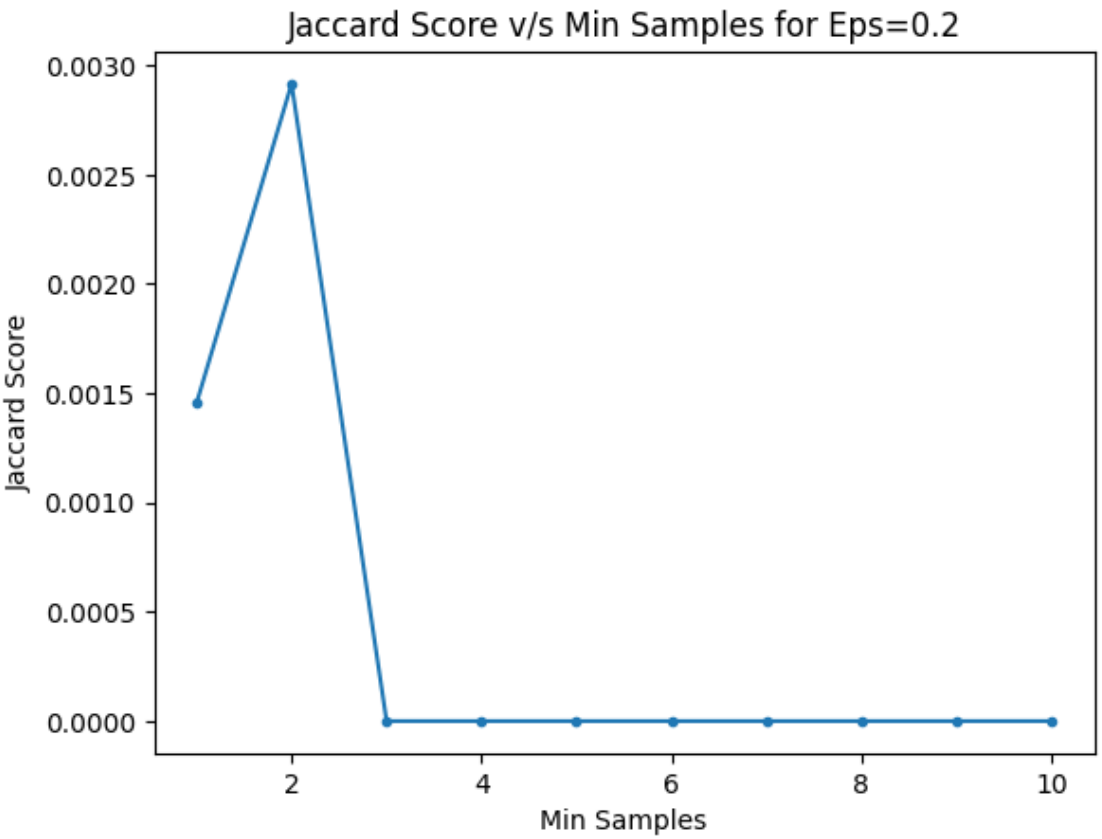
[ ]:
```
0.09554140127388536
```

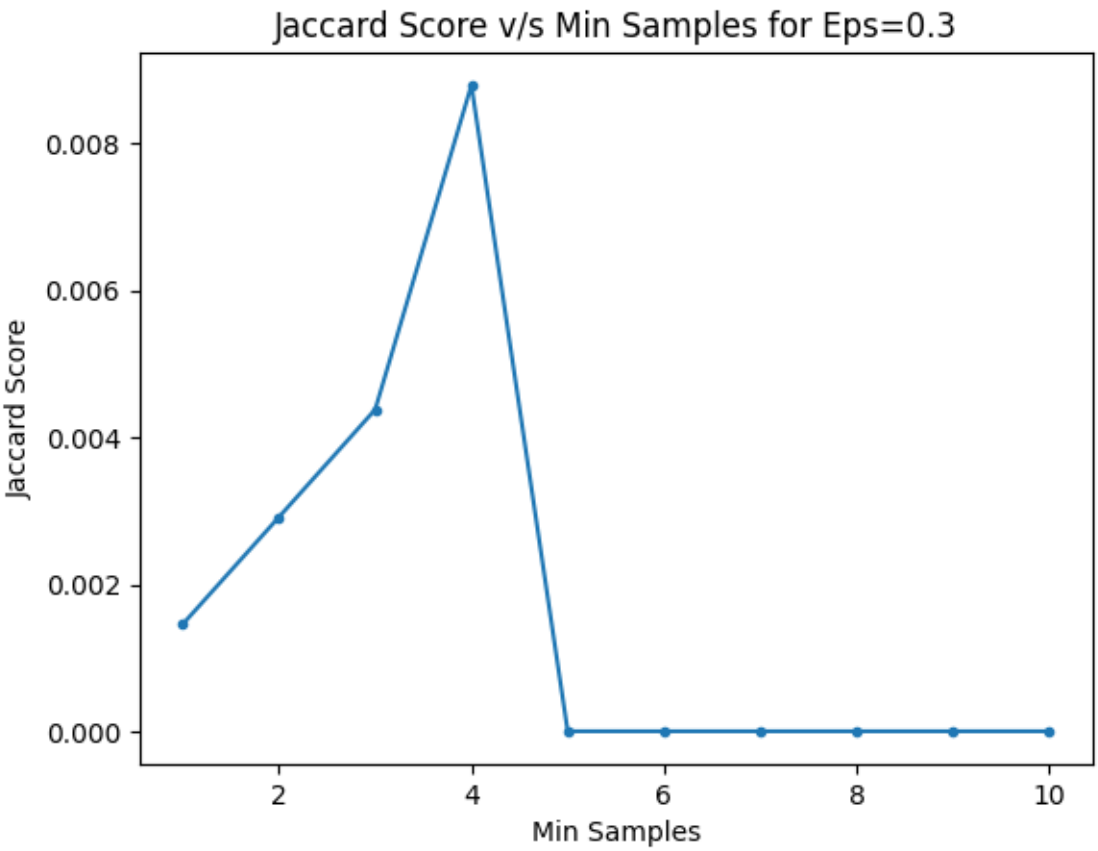
[ ]:
```
agg = AgglomerativeClustering(n_clusters=3)
agg_clusters = agg.fit_predict(X)
jaccard_score(y_enc, agg_clusters, average='micro')
```

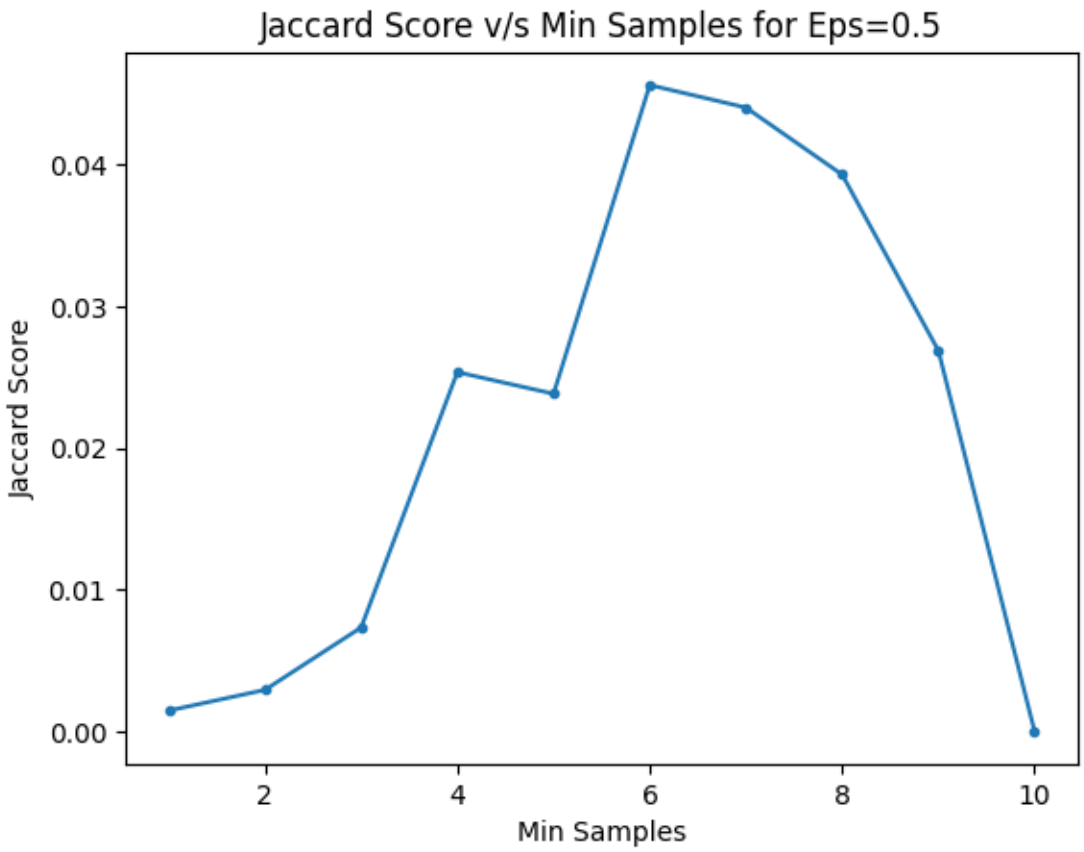
[ ]:
```
0.2835820895522388
```

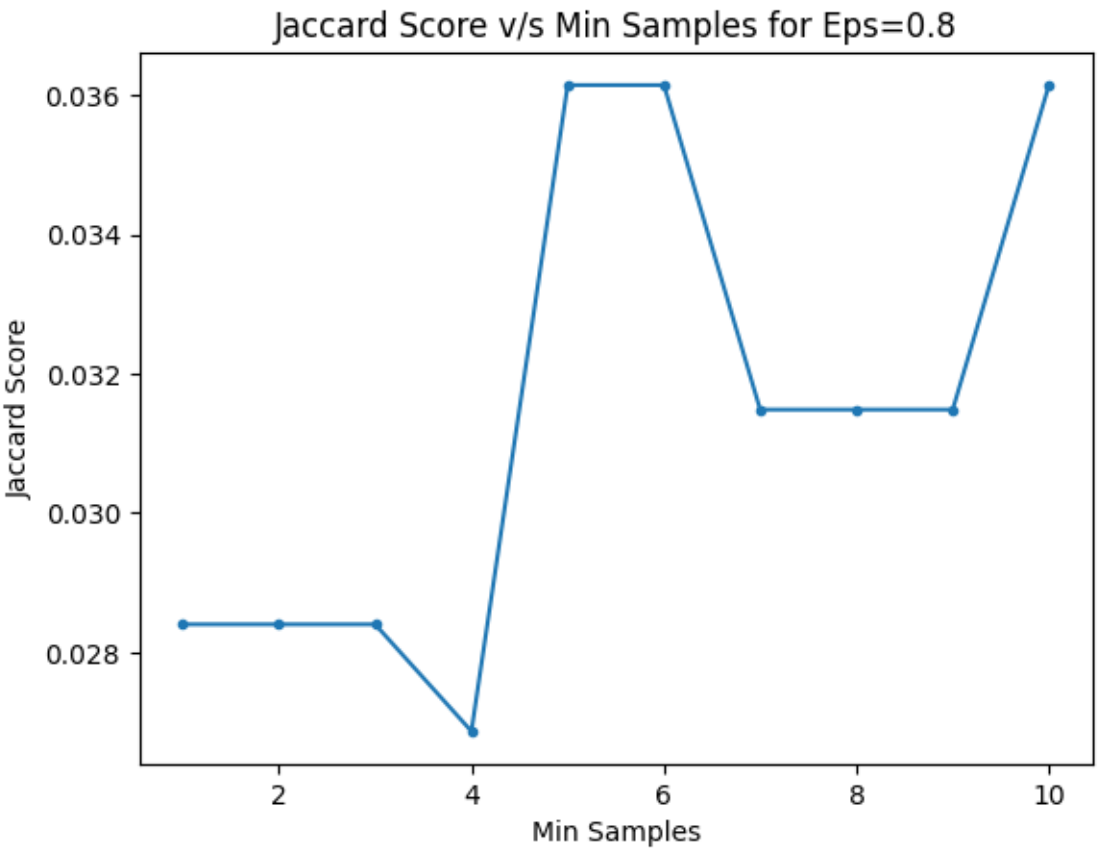
[ ]:
```
eps_arr = [0.1, .2, .3, .5, .8, 1, 1.2, 1.5, 1.8, 2, 2.5]
for eps in eps_arr:
    scores = []
    max_min_samples = 10
    for min_samples in range(1, max_min_samples + 1):
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        clusters = dbscan.fit_predict(X)
        scores.append(jaccard_score(y_enc, clusters, average='micro'))

    plt.plot(range(1, max_min_samples + 1), scores, marker='.')
    plt.xlabel("Min Samples")
    plt.ylabel("Jaccard Score")
    plt.title(f"Jaccard Score v/s Min Samples for Eps={eps}")
    plt.show()
```

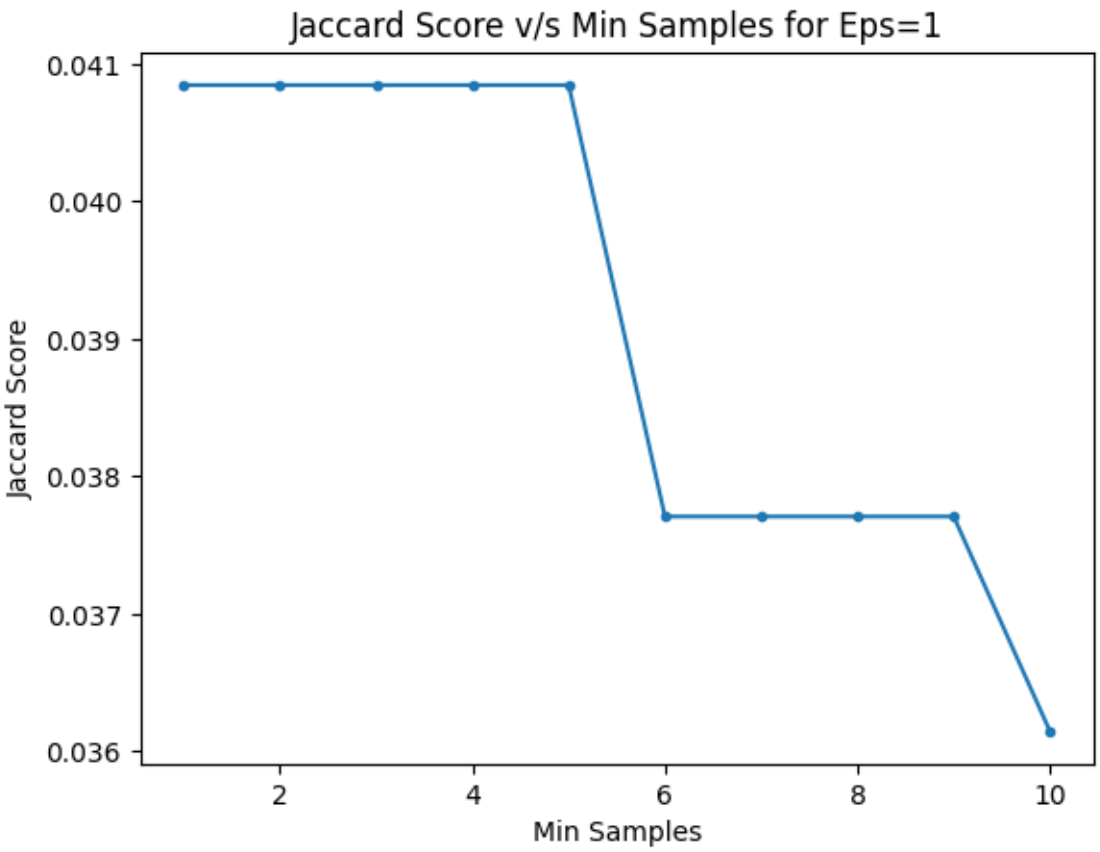

Jaccard Score v/s Min Samples for Eps=0.1
Jaccard Score
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
2
4
6
8
10
Min Samples

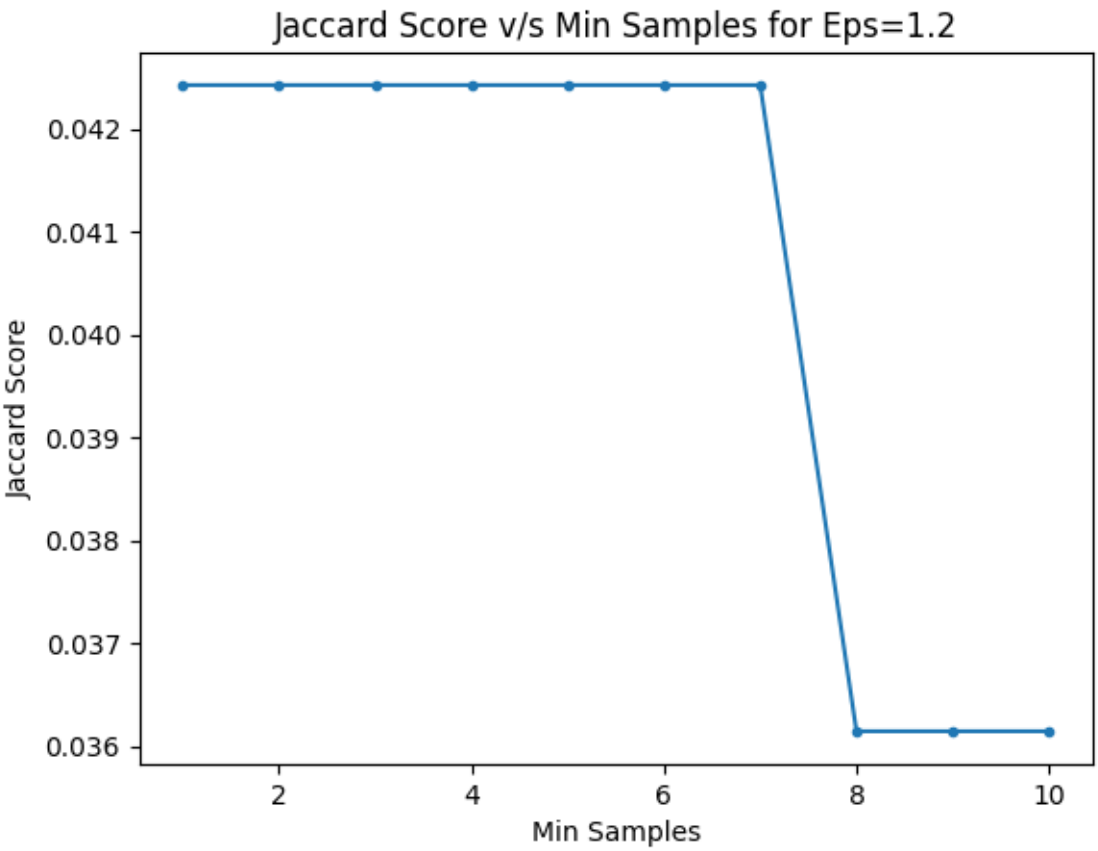

Jaccard Score v/s Min Samples for Eps=0.2
Jaccard Score
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
2
4
6
8
10
Min Samples

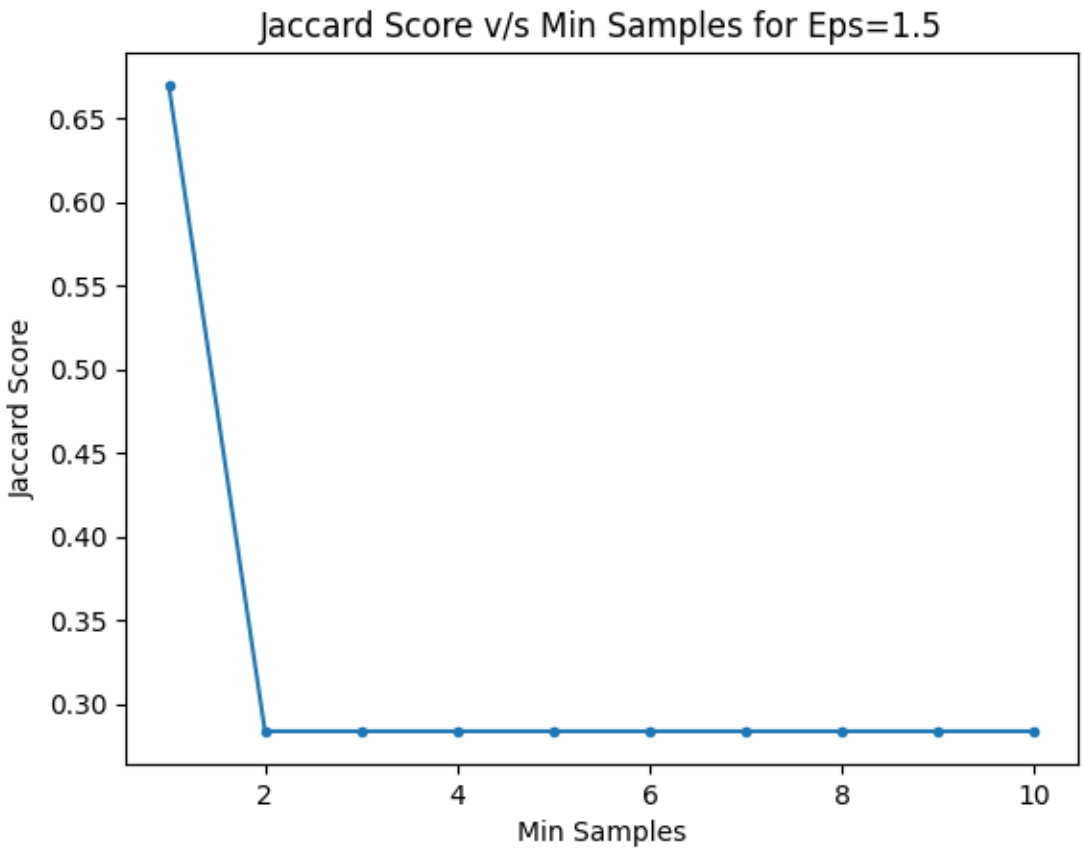

Jaccard Score v/s Min Samples for Eps=0.3
Jaccard Score
0.008
0.006
0.004
0.002
0.000
2
4
6
8
10
Min Samples

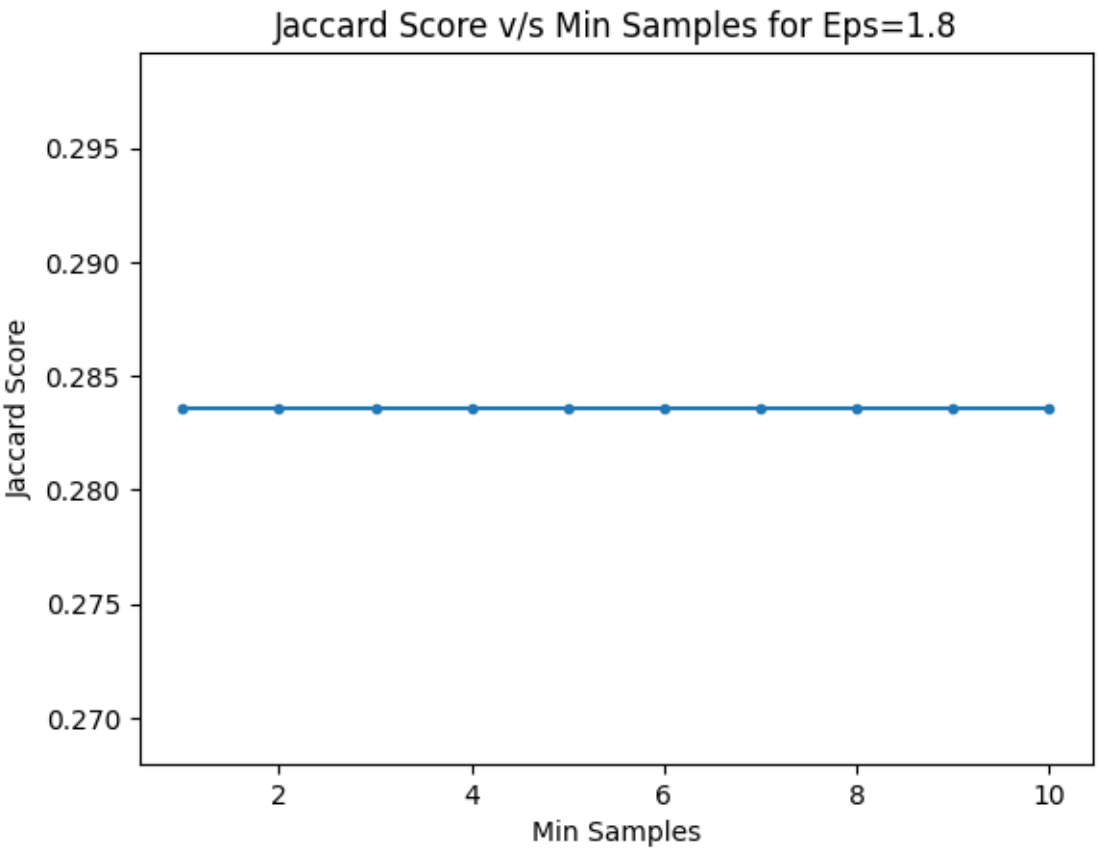

Jaccard Score v/s Min Samples for Eps=0.5
Jaccard Score
0.00
0.01
0.02
0.03
0.04
2
4
6
8
10
Min Samples

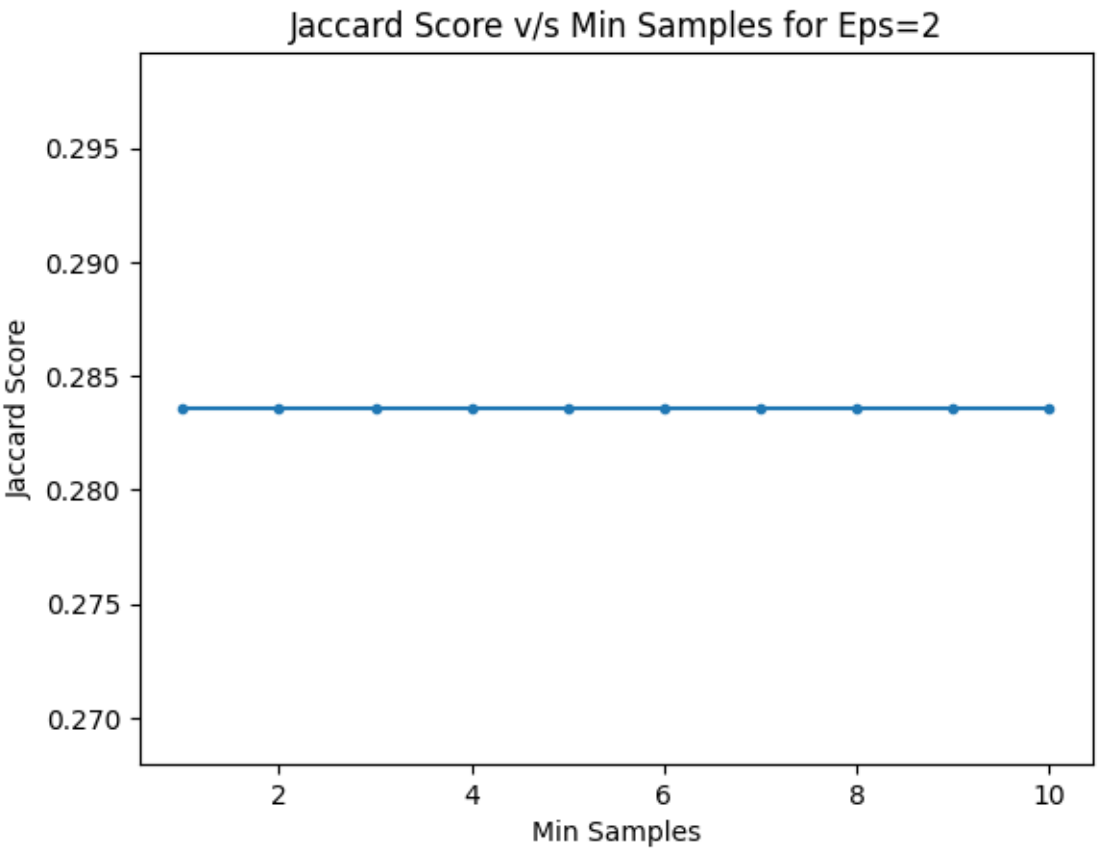

Jaccard Score v/s Min Samples for Eps=0.8
Jaccard Score
0.036
0.034
0.032
0.030
0.028
2
4
6
8
10
Min Samples

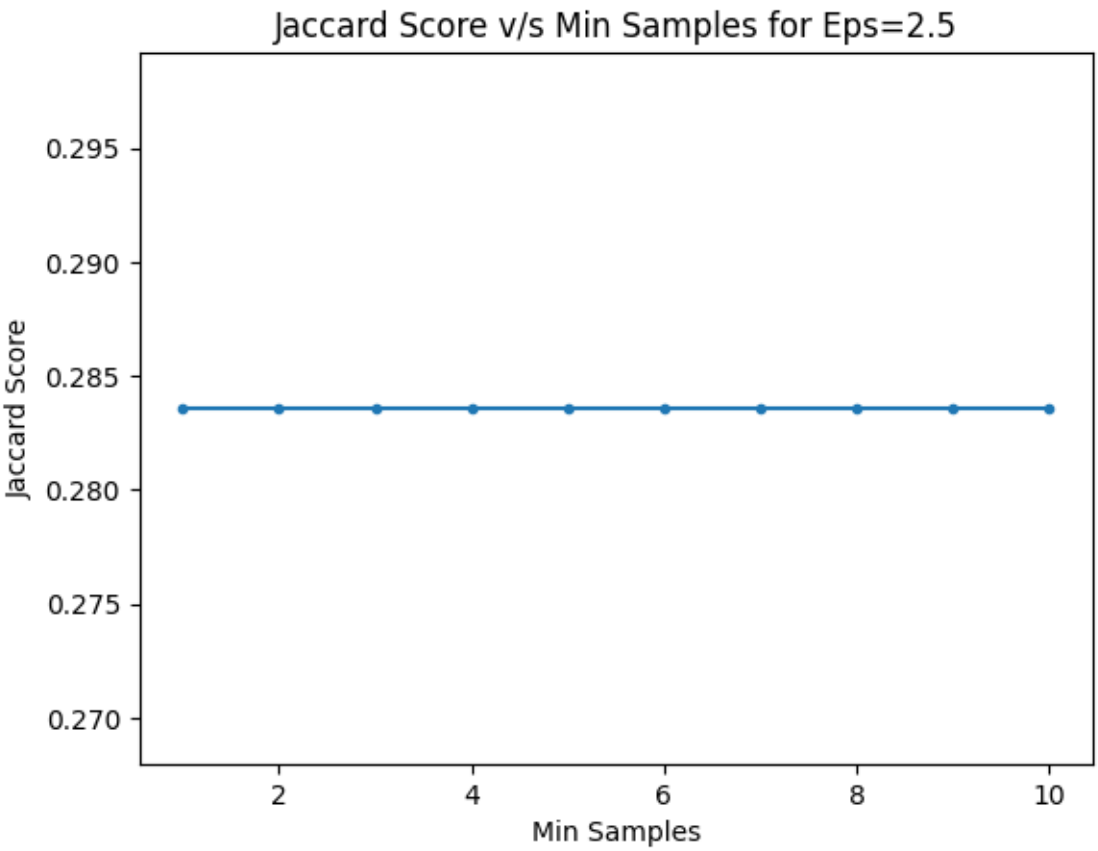

Jaccard Score v/s Min Samples for Eps=1
Jaccard Score
0.041
0.040
0.039
0.038
0.037
0.036
2
4
6
8
10
Min Samples


Jaccard Score v/s Min Samples for Eps=1.2
Jaccard Score
0.042
0.041
0.040
0.039
0.038
0.037
0.036
2
4
6
8
10
Min Samples


Jaccard Score v/s Min Samples for Eps=1.5
Jaccard Score
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
2
4
6
8
10
Min Samples


Jaccard Score v/s Min Samples for Eps=1.8
Jaccard Score
0.295
0.290
0.285
0.280
0.275
0.270
2
4
6
8
10
Min Samples


Jaccard Score v/s Min Samples for Eps=2
Jaccard Score
0.295
0.290
0.285
0.280
0.275
0.270
2
4
6
8
10
Min Samples


Jaccard Score v/s Min Samples for Eps=2.5
Jaccard Score
0.295
0.290
0.285
0.280
0.275
0.270
2
4
6
8
10
Min Samples