# Clustering

## 1. DBSCAN

Using DBSCAN iterate (for-loop) through different values of `min_samples` (1 to 10) and `epsilon` (.05 to .5, in steps of .01) to find clusters in the road-data used in the Lesson and calculate the Silohouette Coeff for `min_samples` and `epsilon`. Plot ***one*** line plot with the multiple lines generated from the min_samples and epsilon values. Use a 2D array to store the SilCoeff values, one dimension represents `min_samples`, the other represents epsilon.

Expecting a plot of `epsilon vs sil_score`.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import DBSCAN
```

In [3]:
```
X = pd.read_csv('data/3D_spatial_network.txt.gz', header=None, names=['osm', 'lat
X = X.drop(['osm'], axis=1).sample(10000)
X.head()
```

Out[3]:

| | lat | lon | alt |
|---|---|---|---|
| **381299** | 8.697499 | 56.960387 | 28.030476 |
| **26533** | 10.295159 | 57.448328 | 80.049196 |
| **338384** | 9.863616 | 56.856573 | 36.967641 |
| **18231** | 8.404013 | 56.710705 | 38.030832 |
| **73387** | 9.939837 | 57.043799 | 2.781029 |

In [4]:
```
from sklearn.metrics import silhouette_score
all_scores = []
for min_sample in range(1, 11):
    scores = []
    for epsilon in np.arange(0.05, 0.51, 0.01):

        dbscan = DBSCAN(eps=epsilon, min_samples=min_sample)
        cluster = dbscan.fit_predict(X[['lat','lon', 'alt']])

        # calculate silouette score here
        score = silhouette_score(X[['lat','lon', 'alt']], cluster)

        scores.append(score)

    all_scores.append(scores)
```

In [5]:
```
all_scores
```

Out[5]:
```
[[0.14167556171737952,
  0.15443633536017393,
  -0.08357335422511171,
  -0.5267896323589104,
  -0.7670910550898974,
  -0.818814892195846,
  -0.7271052596828594,
  -0.6577408960680218,
  -0.6339131158133907,
  -0.5795370729286866],
 [-0.5431270304730578,
  -0.1470111133011138,
  -0.1380767322390031,
  -0.5137128516938345,
  -0.755206267878820З,
  -0.7261236530625375,
  -0.5958381958586478,
  -0.6378910200493381,
  -0.6282012487493992,
  -0.5794750866981738],
 [-0.7519154272442823,
  -0.38537600561470015,
  -0.2207678741783689,
  -0.5195176833062682,
  -0.7210155148792909,
  -0.7119244182131933,
  -0.5973724253641395,
  -0.6113901252497976,
  -0.5907237377325854,
  -0.5406488277282301],
 [-0.7729054714764483,
  -0.5614706907270581,
  -0.3076101591565368,
  -0.47763010656910027,
  -0.7185213850852973,
  -0.6275993231272161,
  -0.60208985518152,
  -0.6153897106186198,
  -0.5923887792108483,
  -0.5410585064913552],
 [-0.7004050783414434,
  -0.6564659755837862,
  -0.4101199780046713,
  -0.41831176848285667,
  -0.6838671779346042,
  -0.6018247156087672,
  -0.607795824284823,
  -0.6092480725315045,
  -0.5863909021687825,
  -0.5420780476473915],
 [-0.6426490546522841,
  -0.723800932445474,
  -0.5097664030223825,
  -0.4361462271372893,
```

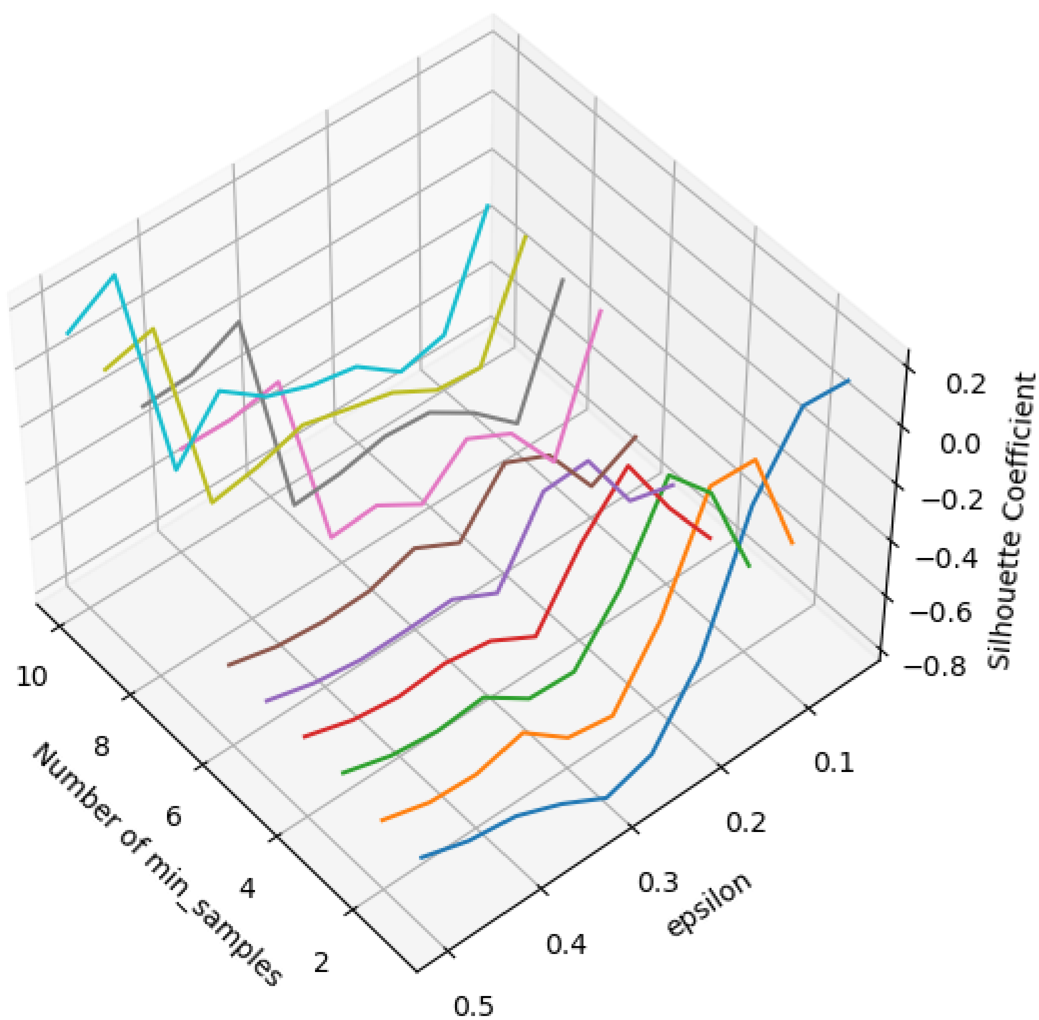
```
  -0.6231197014826763,
  -0.5396103962797842,
  -0.5917376167213056,
  -0.5970990446296525,
  -0.5813672489178768,
  -0.5405705957286739],
 [-0.30573151474430854,
  -0.7536954198590619,
  -0.5497580952359485,
  -0.4696093573390804,
  -0.6061351258747132,
  -0.5083302355372277,
  -0.5195973780890814,
  0.12272227035228407,
  0.09295714217107712,
  0.094688344027736],
 [-0.3094662043024607,
  -0.730050095940898,
  -0.5924082274477486,
  -0.49254835039002925,
  -0.4805034589558532,
  -0.5140476778338426,
  -0.5262506524252248,
  0.2093342852939561,
  0.12923212866448353,
  0.119960372239723],
 [-0.26867478667558053,
  -0.6407560773454459,
  -0.6262281352070399,
  -0.5374833755545523,
  -0.49896214869197586,
  -0.458001809079244,
  -0.5085945511741227,
  -0.5372234072888684,
  0.16777300586229524,
  0.12712710027372862],
 [-0.26892511136115677,
  -0.6370924132333707,
  -0.6752361502605713,
  -0.5595943889254104,
  -0.5302457842467561,
  -0.4729058248246201,
  -0.3547655671385737,
  -0.5392452424047037,
  0.23448495764050337,
  0.13463303425996043]]
```

In [6]:
```
fig = plt.figure(1)
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)

plt.cla()
for min_sample in range(1, 11):
    ax.plot([min_sample] * len(all_scores[0]), np.arange(0.05, 0.51, 0.01), all_s

ax.set_xlabel('Number of min_samples')
ax.set_ylabel('epsilon')
ax.set_zlabel('Silhouette Coefficient')
plt.show()
```

```
C:\Users\Lite\AppData\Local\Temp\ipykernel_52048\3981562889.py:3: MatplotlibDep
recationWarning: Axes3D(fig) adding itself to the figure is deprecated since 3.
4. Pass the keyword argument auto_add_to_figure=False and use fig.add_axes(ax)
to suppress this warning. The default value of auto_add_to_figure will change t
o False in mpl3.5 and True values will no longer work in 3.6.  This is consiste
nt with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```



## 2. Clustering your own data

Using your own data, find relevant clusters/groups within your data (repeat the above). If your data is labeled with a class that you are attempting to predict, be sure to not use it in training and clustering.

You may use the labels to compare with predictions to show how well the clustering performed using one of the clustering metrics (http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation (http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation)).

If you don't have labels, use the silhouette coefficient to show performance. Find the optimal fit for your data but you don't need to be as exhaustive as above.

Additionally, show the clusters in 2D or 3D plots.

As a bonus, try using PCA first to condense your data from N columns to less than N.

Two items are expected:

- Metric Evaluation Plot (like in 1.)
- Plots of the clustered data

In [34]:
```
from sklearn.preprocessing import OrdinalEncoder

enc = OrdinalEncoder()

data = pd.read_csv("data\\iris.data", names=["SepalLength", "Sepal width","Petal

new_data = data.copy()

new_data['Class'] = enc.fit_transform(data[['Class']])
labels = new_data.iloc[:, -1].to_numpy()
X = new_data.iloc[:, :-1].to_numpy()
print(X[:5])
print(labels[:5])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
[0. 0. 0. 0. 0.]
```

In [37]:
```
# PCA
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
print(X.shape)
X
```

```
(150, 3)
```

In [57]:
```
from sklearn.metrics import v_measure_score

all_scores = []
all_clusters = []
for min_sample in range(1, 11):
    scores = []
    clusters = []
    for epsilon in np.arange(0.05, 0.51, 0.05):

        dbscan = DBSCAN(eps=epsilon, min_samples=min_sample)
        cluster = dbscan.fit_predict(X)
        clusters.append(cluster)

        # calculate silouette score here
        score = v_measure_score(labels, cluster)

        scores.append(score)

    all_scores.append(scores)
    all_clusters.append(clusters)

all_scores
```

Out[57]:
```
[[0.3615025522111045,
  0.36925353066227723,
  0.39493263106083876,
  0.42770845923582096,
  0.48777765586775623,
  0.5564179633793501,
  0.5659825855518522,
  0.5993539628084387,
  0.6427550164649745,
  0.6599508222832073],
 [0.05830415764818812,
  0.20495319967612752,
  0.3221856002433511,
  0.39131479397508223,
  0.46360374298122314,
  0.5386456621648364,
  0.5515573926263604,
  0.5871757823399452,
  0.6257793784771467,
  0.6571788607036102],
 [0.03740984619206994,
  0.09188335656811052,
  0.26706857973771697,
  0.38345611614709163,
  0.4620476637872152,
  0.5442953169081833,
  0.5524056388279177,
  0.5802059398068086,
  0.627585329642541,
  0.6625376669466911],
 [0.0,
  0.060708642450685156,
  0.19304007797420558,
```

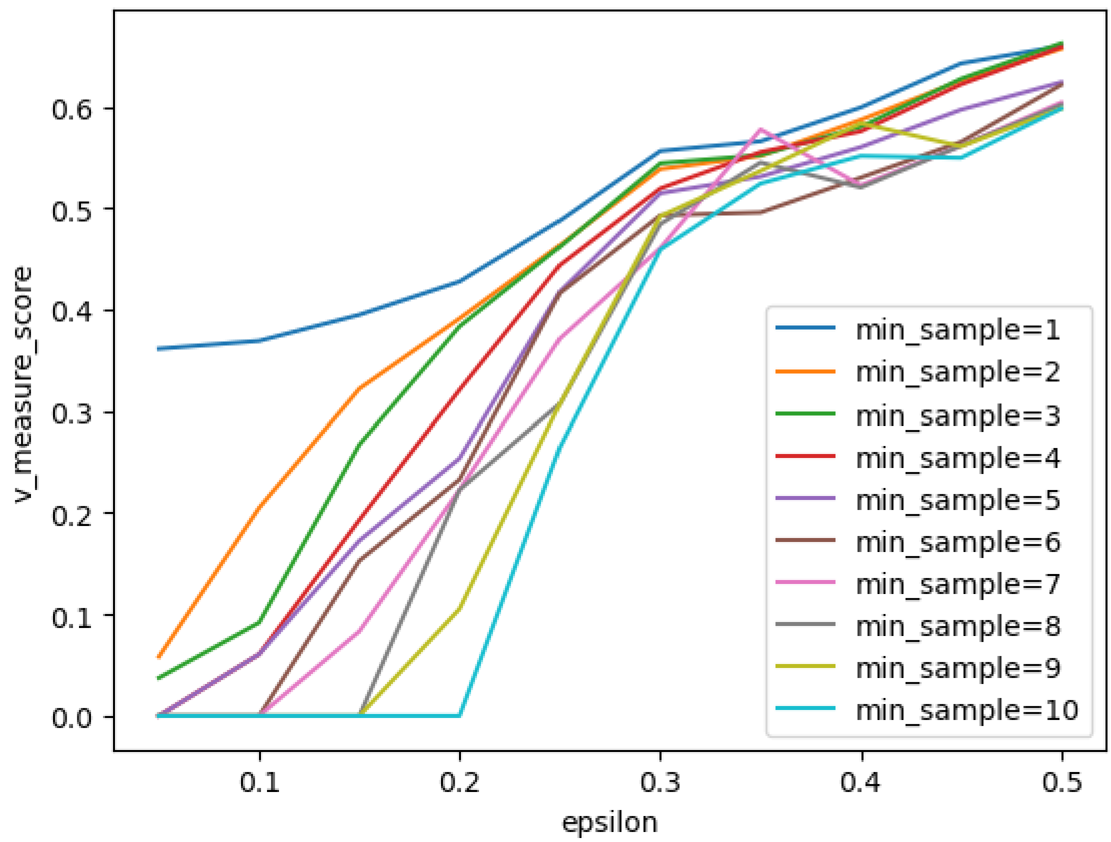
```
  0.3210854864936306,
  0.44410060067451906,
  0.5194934837499603,
  0.5553331480460819,
  0.575816911358352,
  0.6220887542320666,
  0.6587633493101253],
 [0.0,
  0.060708642450685156,
  0.17265776246056863,
  0.25366546045140775,
  0.41803347743449043,
  0.5147455530776229,
  0.5311886542157567,
  0.5602657153982252,
  0.5971044371912548,
  0.624197660512502],
 [0.0,
  0.0,
  0.152841021413629,
  0.23289097163319733,
  0.4165819264164717,
  0.4930311451058567,
  0.4958262120167615,
  0.5301824890414011,
  0.5653431509037828,
  0.621591697560461],
 [0.0,
  0.0,
  0.08327127597195305,
  0.22268566970593898,
  0.37172861260535695,
  0.4608484252203195,
  0.5776127387142849,
  0.5226556649246591,
  0.5614281211107915,
  0.6041447533328671],
 [0.0,
  0.0,
  0.0,
  0.22268566970593898,
  0.3075303769721428,
  0.48437191241305044,
  0.5450436872466629,
  0.5204567150384597,
  0.5614281211107915,
  0.6019449419123793],
 [0.0,
  0.0,
  0.0,
  0.1053672659196585,
  0.3075303769721428,
  0.4922342629423773,
  0.536815949795598,
  0.5837652778470633,
  0.5609862211242184,
  0.5981978860354958],
```

```
[0.0,
 0.0,
 0.0,
 0.0,
 0.26477927249793715,
 0.45930426820037745,
 0.524342624353444,
 0.5515187536478221,
 0.5496743707015539,
 0.5981978860354958]]
```

In [53]:
```
fig = plt.figure(3)
plt.clf()

plt.cla()
for min_sample in range(len(all_scores)):
    # ax.plot([min_sample] * len(all_scores[0]), np.arange(0.05, 0.51, 0.01), all
    plt.plot(np.arange(0.05, 0.51, 0.05), all_scores[min_sample], label=f"min_sam

plt.legend()
plt.xlabel("epsilon")
plt.ylabel("v_measure_score")
plt.show()
```




In [64]:
```
np.array(all_clusters).shape
```

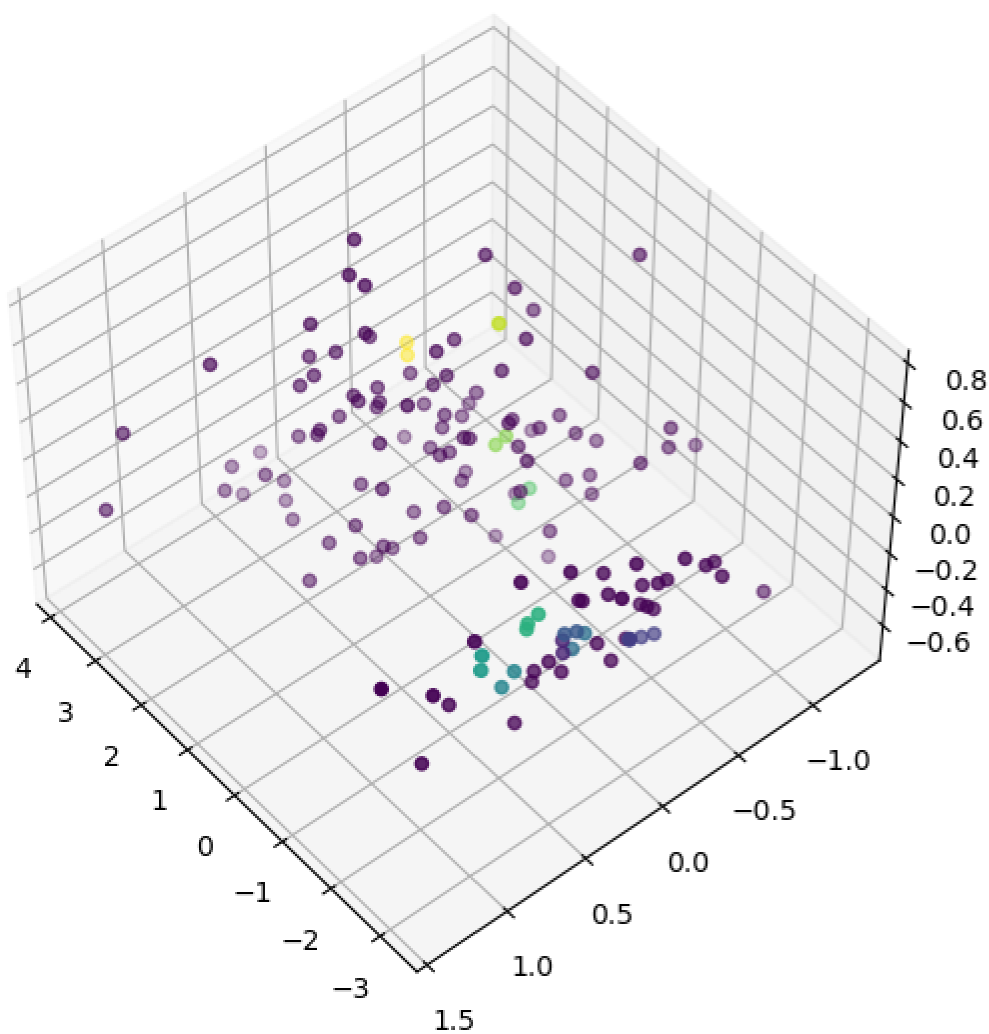
Out[64]: (10, 10, 150)

In [76]:
```python
def plot_clustered(cluster, X):
    fig = plt.figure(3)
    plt.clf()
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
    plt.cla()

    ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=cluster)
    plt.show()
```

In [77]:
```python
min_sample_idx = 1  # idx of min_sample in all_clusters[0,9]
epsilon_idx = 1  # idx of epsilon in all_clusters[0,9]
cluster = all_clusters[min_sample_idx][epsilon_idx]
plot_clustered(cluster, X)
```

```
C:\Users\Lite\AppData\Local\Temp\ipykernel_36076\3122014412.py:4: MatplotlibDep
recationWarning: Axes3D(fig) adding itself to the figure is deprecated since 3.
4. Pass the keyword argument auto_add_to_figure=False and use fig.add_axes(ax)
to suppress this warning. The default value of auto_add_to_figure will change t
o False in mpl3.5 and True values will no longer work in 3.6.  This is consiste
nt with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```



In [78]:
```
min_sample_idx = 9  # idx of min_sample in all_clusters[0,9]
epsilon_idx = 9  # idx of epsilon in all_clusters[0,9]
cluster = all_clusters[min_sample_idx][epsilon_idx]
plot_clustered(cluster, X)
```

```
C:\Users\Lite\AppData\Local\Temp\ipykernel_36076\3122014412.py:4: MatplotlibDep
recationWarning: Axes3D(fig) adding itself to the figure is deprecated since 3.
4. Pass the keyword argument auto_add_to_figure=False and use fig.add_axes(ax)
to suppress this warning. The default value of auto_add_to_figure will change t
o False in mpl3.5 and True values will no longer work in 3.6.  This is consiste
nt with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```