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 [ ]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt

from mpl_toolkits import mplot3d

from sklearn.cluster import DBSCAN
from sklearn import metrics

plt.rcParams['font.size'] = 14
```

In [ ]:
```
# Load the data
file = 'C:/Users/jomors/OneDrive/_JHU/AS.470.667 Machine Learning/mlnn-main/data/3D_sp
# Rename column headings as they load
X = pd.read_csv(file, header=None, names=['osm', 'lat','lon','alt'])
X = X.drop(['osm'], axis=1).sample(10000)   # sample should take a random sample of th
X.head()
```

Out[ ]:

	lat	lon	alt
311358	11.179962	57.322866	2.019803
83702	9.503374	56.922242	24.698545
251765	9.740749	57.296361	8.437094
209399	10.538270	57.463233	1.411779
193568	9.489505	56.775696	36.337757

In [ ]:
```
# Make a copy of the data and normalize the columns
XX = X.copy()
XX['alt'] = (X.alt - X.alt.mean())/X.alt.std()
XX['lat'] = (X.lat - X.lat.mean())/X.lat.std()
XX['lon'] = (X.lon - X.lon.mean())/X.lon.std()
XX.head()
```

Out[ ]:

	lat	lon	alt
311358	2.317207	0.827437	-1.080755
83702	-0.358606	-0.550832	0.126130
251765	0.020242	0.736252	-0.739248
209399	1.293075	1.310342	-1.113112
193568	-0.380740	-1.054995	0.745530

In [ ]:
```
# Create our ranges of values to cycle through.
min_samples = np.arange(1, 11)
epsilons = np.arange(.05, .55, .05)
```

In [ ]:
```
# Initialize an empty array
all_scores = []

# Loop through the min_samples values
for min_sample in min_samples:
    #scores = []

    # Loop through the epsilon values
    for epsilon in epsilons:
        # Create the DBSCAN model with the current min_sample and epsilon values
        dbscan = DBSCAN(eps=epsilon, min_samples=min_sample)

        # Fit the model
        cluster = dbscan.fit_predict(XX[['lat','lon', 'alt']])

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

        # Append the values into a 3D array
        #scores.append(score)
        all_scores.append([min_sample, epsilon, score])

    #all_scores.append(scores)
```

In [ ]:
```
# Convert the 3D array to a data frame
results = pd.DataFrame(all_scores)
# Assign meaningful column names
results.columns = ['min_samples', 'epsilon', 'sil_score']
results

#results = np.array(min_samples, epsilons, all_scores)
```

Out[ ]:

	min_samples	epsilon	sil_score
0	1	0.05	0.173288
1	1	0.10	-0.208377
2	1	0.15	-0.677241
3	1	0.20	-0.545692
4	1	0.25	-0.464297
...	...	...	...
95	10	0.30	0.100371
96	10	0.35	0.102618
97	10	0.40	0.120641
98	10	0.45	0.141536
99	10	0.50	0.138930

100 rows × 3 columns

In [ ]:

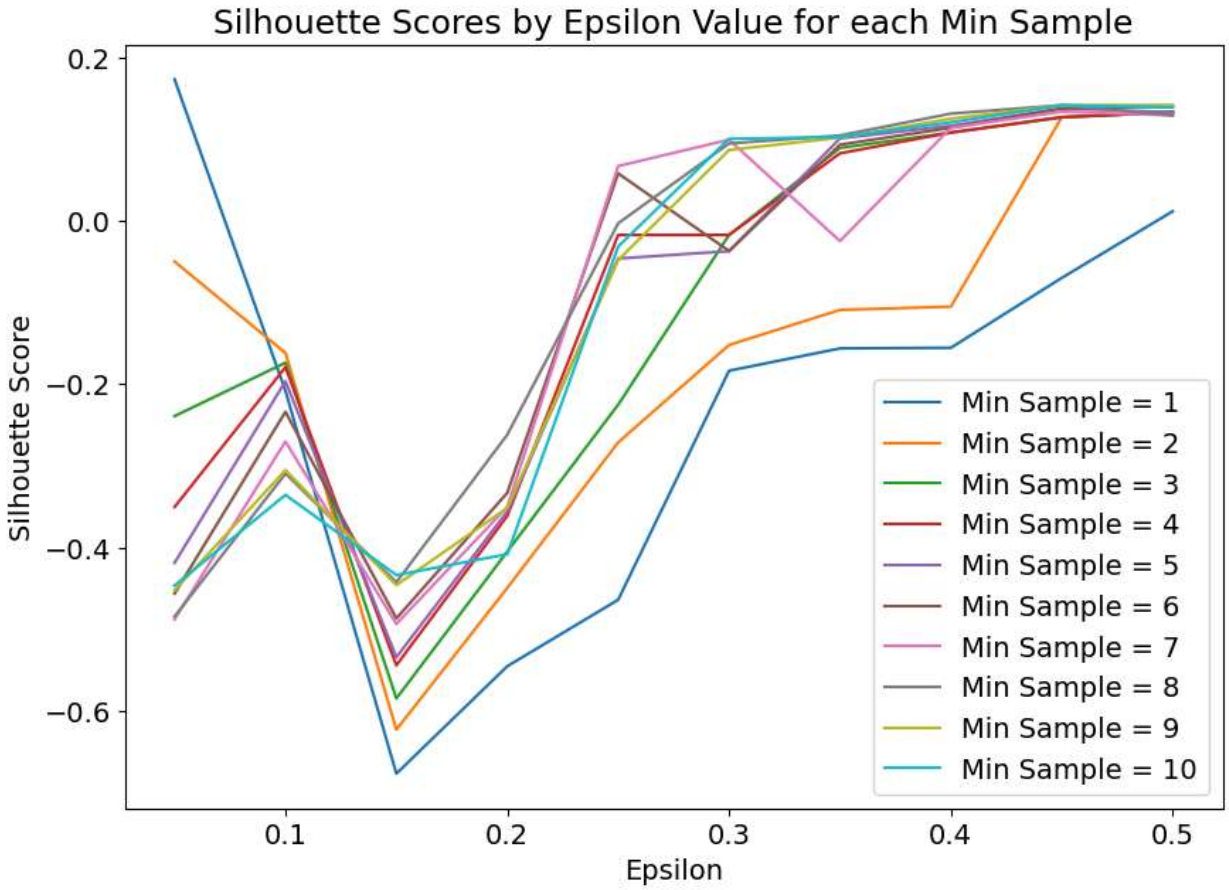
```
plt.rcParams['figure.figsize'] = (10.0, 7.0)

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()

# Loop through the min_samples values to create
# a separate line plot for each min_sample value.
for min_sample in min_samples:
    ax.plot(results[results.min_samples==min_sample].epsilon,
            results[results.min_samples==min_sample].sil_score,
            label='Min Sample = ' + str(min_sample))

# Set labels and titles
ax.set_xlabel('Epsilon')
ax.set_ylabel('Silhouette Score')
ax.legend(loc='best')
ax.set_title("Silhouette Scores by Epsilon Value for each Min Sample")

# Display the line chart
plt.show()
```



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).

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 [ ]:
```
from sklearn.datasets import load_wine
wine = load_wine()
```

```
winedf = pd.DataFrame(wine.data, columns=wine.feature_names)
winedf.head()
```

Out[ ]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phe
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

The **wine** dataset is a sample dataset included with scikit-learn. This includes a set of target data that classifies the observations into 3 types. Thus, we shoudl expect 3 classifications of the dataset.

Here, I start with a PCA decomposition of the data into 3 variables. This facilitates 3D plotting.

In [ ]:

```
# Apply PCA to reduce to 3 variables for easy plotting
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
wine_pca = pca.fit(wine.data).transform(wine.data)
wine_pca
```

Out[ ]:
```
array([[ 3.18562979e+02,  2.14921307e+01,  3.13073470e+00],
       [ 3.03097420e+02, -5.36471768e+00,  6.82283550e+00],
       [ 4.38061133e+02, -6.53730945e+00, -1.11322298e+00],
       [ 7.33240139e+02,  1.92729032e-01, -9.17257016e-01],
       [-1.15714285e+01,  1.84899946e+01, -5.54422076e-01],
       [ 7.03231192e+02, -3.32158674e-01,  9.49375334e-01],
       [ 5.42971581e+02, -1.35189666e+01,  2.12694283e+00],
       [ 5.48401860e+02,  1.14494324e+01,  4.04924202e-02],
       [ 2.98036863e+02, -8.18015784e+00,  3.88097517e+00],
       [ 2.98049553e+02, -7.10154294e+00,  1.55845533e+00],
       [ 7.63079712e+02, -8.33431723e+00, -1.88629037e+00],
       [ 5.32943228e+02, -1.42876338e+01,  1.30335240e-01],
       [ 5.72834410e+02, -2.10050143e+01,  3.72614859e-01],
       [ 4.02925358e+02, -1.61026352e+01,  5.67513986e+00],
       [ 8.00053394e+02, -1.21184472e+01,  3.04652991e+00],
       [ 5.63245578e+02,  2.21482866e+00, -5.25510985e-01],
       [ 5.33379651e+02,  1.08048022e+01, -2.47652734e+00],
       [ 3.83317591e+02,  8.47741982e+00, -1.98974501e+00],
       [ 9.33118387e+02, -8.35447844e+00, -1.93291276e+00],
       [ 9.84031775e+01,  1.43926594e+01,  4.10374616e+00],
       [ 3.35935940e+01,  2.55679565e+01,  4.03358615e+00],
       [ 2.31464375e+01,  1.81747309e+00,  8.87160841e-01],
       [ 2.88093030e+02, -3.96304175e+00,  2.05371381e+00],
       [ 2.67981513e+02, -9.57070401e+00,  7.58411832e-01],
       [ 9.80198858e+01, -5.49584606e+00, -3.87297661e-01],
       [ 8.34987440e+01,  2.28916215e+01, -4.27883653e+00],
       [ 4.47925596e+02, -1.47973313e+01,  1.21983445e+00],
       [ 5.37919165e+02, -1.53883461e+01,  1.64692329e-01],
       [ 1.68210468e+02,  4.25531096e+00, -1.03384408e-01],
       [ 2.88008247e+02, -8.95973155e+00,  2.16149821e+00],
       [ 5.38026452e+02, -8.21273882e+00, -5.16839628e+00],
       [ 7.68092939e+02, -7.37989737e+00, -3.22996241e+00],
       [ 2.43150751e+02, -1.43914928e-01,  1.68080273e+00],
       [ 4.88601280e+02,  2.35653250e+01, -1.17162642e+00],
       [ 3.48231007e+02,  4.03808015e+00, -4.47674995e-01],
       [ 1.73079957e+02, -2.79292165e+00, -1.50979198e+00],
       [ 1.33286424e+02,  7.77272958e+00,  3.71926608e+00],
       [ 3.58018559e+02, -8.15798412e+00,  7.32006055e-02],
       [ 2.73044432e+02, -6.72507431e+00,  3.03479237e+00],
       [ 1.36430021e+01,  2.78468321e+01,  6.64055073e+00],
       [ 4.84296422e+01,  1.63219498e+01,  3.36121013e+00],
       [ 2.87884092e+02, -1.48851185e+01, -8.06936528e-01],
       [ 3.48095348e+02, -5.04342609e+00,  2.83896837e+00],
       [-6.68146554e+01,  4.38482992e+00,  2.36654298e+00],
       [ 1.38227010e+02,  4.73068836e+00,  2.19316584e+00],
       [ 3.33254806e+02,  5.31933116e+00, -7.83758129e-01],
       [ 3.18111848e+02, -3.48694494e+00,  1.99385738e+00],
       [ 2.38111485e+02, -3.05152100e+00,  2.22732714e+00],
       [ 3.13119915e+02, -2.30536316e+00, -8.55903527e-01],
       [ 5.13187748e+02, -8.63759235e-01, -9.63892767e-01],
       [ 4.02939017e+02, -1.50533028e+01,  4.25268054e+00],
       [ 5.17928963e+02, -1.49975549e+01, -2.95433477e-01],
       [ 4.43266411e+02,  3.26050204e+00,  3.22533487e+00],
       [ 6.28286071e+02,  4.04114538e+00, -3.57931528e-01],
       [ 3.13397613e+02,  1.26089135e+01,  2.00469936e+00],
       [ 3.73333291e+02,  9.66419863e+00, -2.29204575e+00],
       [ 2.23417106e+02,  1.42168924e+01,  2.38786504e+00],
       [ 5.23072851e+02, -7.09662526e+00,  1.29288369e-01],
       [ 5.38182097e+02, -1.35678013e+00,  2.30978511e-01],
       [-2.27032690e+02, -7.99496797e+00,  9.94713134e+00],
```

```
[-6.68501220e+01,  2.31986654e+00,  4.12689933e+00],
[-2.96824437e+02,  5.46847570e+00,  3.66175522e+00],
[-1.16970473e+02, -3.72638342e+00,  2.21999884e+00],
[-3.27059126e+02, -6.94813081e+00,  2.08570803e+00],
[-3.91752486e+02,  1.11716783e+01,  3.02474464e+00],
[-6.89057604e+01, -5.58620537e-01,  1.89506922e+00],
[-2.45212524e+02, -1.74936393e+01,  4.99323593e+00],
[-2.37241410e+02, -1.75310256e+01,  6.56014227e-01],
[ 3.29615599e+00,  1.00958116e+01,  3.21779225e+00],
[-2.79661207e+01,  5.16301252e+01,  4.99946128e+00],
[ 1.23130138e+02,  1.05983451e+00, -8.11022883e-01],
[-3.37104122e+02, -7.62385512e+00, -3.33557626e+00],
[-2.75096026e+02, -7.75315213e+00, -2.93117724e+00],
[ 2.38716757e+02,  3.52518274e+01, -9.00507574e+00],
[ 1.39094057e+02, -1.20747620e+00, -1.19825468e+00],
[-3.18876329e+02,  2.81077113e+00,  5.00732155e+00],
[-3.55060253e+02, -7.53070183e+00,  4.80195138e+00],
[-2.46633597e+02,  1.65584367e+01,  3.44015940e+00],
[ 3.77168966e+00,  3.60310924e+01,  6.13187473e+00],
[-2.83842248e+02,  6.36186080e+00, -1.30886277e+00],
[-4.69059350e+02, -5.45892398e+00,  3.20131486e+00],
[-3.31313392e+01, -1.31871050e+01,  8.94219904e-01],
[-1.17290392e+02, -1.95786125e+01, -3.39719663e+00],
[-2.32132988e+02, -1.05301033e+01, -2.43010911e+00],
[-2.26953299e+02, -1.77605879e+00,  3.09059588e+00],
[-2.96852152e+02,  4.45280309e+00,  3.64545501e+00],
[-2.52046392e+02, -5.21261759e+00, -1.45231083e+00],
[-1.85108857e+02, -8.31858202e+00, -4.81593129e+00],
[-6.71807518e+01, -1.45246902e+01, -1.47148703e+00],
[-1.22431326e+02, -2.75040387e+01, -3.08038596e+00],
[-2.67185333e+02, -1.40538901e+01,  2.34355042e+00],
[-2.97104415e+02, -8.41503161e+00, -9.72715452e-01],
[-2.52213885e+02, -1.52515702e+01, -1.05599736e-02],
[-4.57022215e+02, -3.69733793e+00,  4.01928491e+00],
[-4.01851274e+02,  5.33140022e+00,  3.84371981e+00],
[ 1.91183397e+02,  5.87937624e+01,  2.36881154e+00],
[-1.21279178e+02,  3.64175149e+01,  3.23171033e-01],
[-3.19088273e+02, -9.19307820e+00,  5.13148563e+00],
[-8.70802926e+01, -1.02265727e+01,  1.53970742e+00],
[-3.41040900e+02, -5.75056559e+00,  3.48018898e+00],
[-3.69316146e+01, -2.17041593e+00,  2.59030958e+00],
[-1.85073713e+02, -8.52051902e+00,  2.25097671e+00],
[-3.08882387e+02,  3.75165655e+00,  6.45735750e-01],
[-3.32089296e+02, -7.88372875e+00,  2.04129847e+00],
[-7.51531378e+01, -1.34138279e+01, -2.69440610e-01],
[-4.32009742e+02, -2.03294673e+00, -1.29785520e-02],
[-2.37206697e+02, -1.55619679e+01,  1.54724661e+00],
[-2.59148368e+02, -1.10788163e+01, -1.59997285e+00],
[-4.34957780e+02, -5.95260295e-02,  3.09505248e+00],
[-6.69906958e+01, -4.57145764e+00,  6.49008692e-01],
[-1.84736439e+02,  1.05164632e+01,  1.70289086e+00],
[-4.22042874e+02, -4.24492194e+00,  1.07980318e+00],
[-1.39818663e+02,  5.73555207e+00,  5.34029189e-01],
[-3.13060948e+02, -6.16709401e+00,  4.29583739e-01],
[-3.62129633e+02, -9.25929540e+00, -8.39044347e-01],
[-3.40115006e+02, -8.68973443e+00,  3.45930570e-01],
[-2.52109472e+02, -9.27587015e+00,  5.57705742e-01],
[-4.01695973e+02,  1.54258903e+01,  2.47856188e-01],
[-3.75171092e+02, -1.31850503e+01,  4.60945469e+00],
[-1.83097929e+02, -9.55354432e+00,  1.90848316e+00],
```

```
[-1.21945241e+02, -1.58889428e+00,  6.55635430e-01],
[-2.81534569e+02,  2.45082612e+01, -6.48504587e+00],
[-3.81826778e+02,  9.18910976e+00, -4.13334205e+00],
[-3.67089380e+02, -7.18821589e+00, -2.18655027e-01],
[-3.67157175e+02, -1.11963202e+01,  3.65508871e-01],
[-3.69103601e+02, -8.17580491e+00,  6.39848719e-01],
[-3.95078836e+02, -6.67983187e+00,  6.18945571e-02],
[-2.81030369e+02, -2.53586007e+00, -6.78592530e+00],
[-4.05061288e+02, -4.45107367e+00, -2.20721536e+00],
[-1.67237642e+02, -1.67311857e+01, -1.73687914e+00],
[-1.16473000e+02,  2.42660201e+01,  2.78924950e+00],
[-2.16784540e+02,  8.13044956e+00,  2.39926810e-01],
[-1.86915112e+02,  1.71005736e+00, -3.85614652e+00],
[-1.46769530e+02,  8.92033411e+00, -1.36448130e+00],
[-9.71329736e+01, -1.30582838e+01,  1.67585772e+00],
[-5.19820469e+01, -4.81229310e+00,  1.60730122e-01],
[-2.70979307e+01, -1.02299962e+01, -1.83765584e+00],
[-2.31951534e+02,  5.34907410e-01, -4.83149311e+00],
[-1.67076246e+02, -8.75672311e+00, -1.77377625e-01],
[-1.56868045e+02,  4.16316249e+00, -3.70307659e+00],
[-1.46946490e+02, -1.09672896e+00, -1.00960254e+00],
[ 3.29042927e+01, -1.13053656e+01, -1.34137698e+00],
[-2.26926210e+02,  1.38827837e+00, -2.96295367e+00],
[-1.97004347e+02, -4.22842870e+00, -1.22874042e-01],
[ 1.08312677e+02,  1.03513304e+01, -4.81085430e-01],
[ 8.31182626e+01,  8.06869695e-01, -1.79684297e+00],
[-3.32195840e+02, -1.38132989e+01, -1.70935837e-01],
[-1.22126092e+02, -1.14699259e+01, -3.00064371e+00],
[-9.70193893e+01, -5.90644258e+00, -2.98282629e+00],
[-1.96624807e+02,  1.68731139e+01, -2.25389963e+00],
[-2.46449199e+02,  2.78100954e+01, -3.75074929e+00],
[-2.66628788e+02,  1.71502320e+01, -2.68311926e+00],
[-3.21570741e+02,  2.21643401e+01, -4.42434279e+00],
[-7.18976587e+01, -3.99656400e-01, -7.07753163e-01],
[-1.06816618e+02,  5.20612307e+00, -7.33294017e-01],
[-2.20202568e+01, -6.22943925e+00, -3.86483282e+00],
[-2.67031732e+02, -5.92308914e+00, -7.03284458e-01],
[ 1.32999841e+02, -4.86770544e+00, -8.75379841e+00],
[-8.69139986e+01,  7.96357772e-02, -7.17131368e+00],
[-1.27061290e+02, -8.29141684e+00, -4.57728089e+00],
[-2.27068836e+02, -7.61863030e+00, -1.85508543e+00],
[-6.67554312e+01,  8.47028990e+00, -5.49136535e-01],
[-1.76765098e+02,  9.47319000e+00, -1.84287292e+00],
[-7.17683979e+01,  7.51063784e+00,  1.01222585e+00],
[-1.32048248e+02, -7.25503031e+00, -3.66332810e+00],
[-2.27077845e+02, -7.59337930e+00, -3.10400630e+00],
[-5.16895065e+01,  1.23580850e+01, -4.74361265e+00],
[-6.20850111e+01, -1.05540153e+01, -1.94213604e+00],
[ 3.18276465e+00,  5.39136150e+00, -5.85723969e+00],
[-1.16674818e+02,  1.45333703e+01, -5.75495650e+00],
[-2.36921208e+02,  4.63036998e-01,  9.70193743e-01],
[-2.77083578e+02, -8.74033191e+00, -8.39693618e-01],
[-8.70274026e+01, -7.10459575e+00, -1.96051582e+00],
[-6.98021096e+00, -4.54113657e+00, -2.47470686e+00],
[ 3.13160468e+00,  2.33519051e+00, -4.30993061e+00],
[ 8.84580737e+01,  1.87762846e+01, -2.23757651e+00],
[ 9.34562419e+01,  1.86708191e+01, -1.78839152e+00],
[-1.86943190e+02, -2.13330803e-01, -5.63050984e+00]])
```

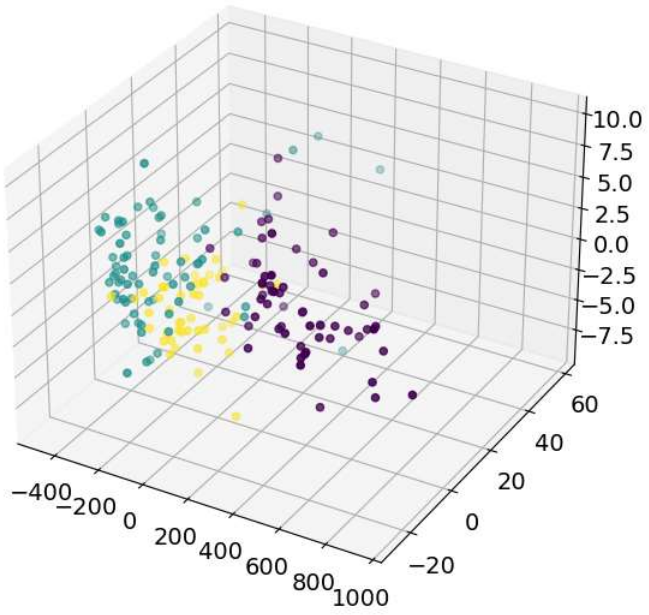
Next, I plot the 3 new variables from the PCA model and color the results based on the original classification provided in the **wine** dataset.

In [ ]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

ax.scatter3D(wine_pca[:,0], wine_pca[:,1], wine_pca[:,2], c=wine.target, cmap='viridis

plt.show()
```

Figure



Next, I cycle through the **epsilon** and **min_samples** values as in part 1, applying a DBSCAN classification on the variables that came from the PCA decomposition.

In [ ]:
```
results = pd.DataFrame(wine_pca, columns=['one', 'two', 'three'])

# Initialize an empty array
wine_scores = []

# Loop through the min_samples values
for min_sample in min_samples:

    # Loop through the epsilon values
    for epsilon in epsilons:
        # Create the DBSCAN model with the current min_sample and epsilon values
        dbscan = DBSCAN(eps=epsilon, min_samples=min_sample)

        # Fit the model
        #cluster = dbscan.fit_predict(wine_pca)
        cluster = dbscan.fit_predict(results[['one', 'two', 'three']])
```

```
        # calculate silouette score here
        score = metrics.silhouette_score(results[['one', 'two', 'three']], cluster)

        # Append the values into a 3D array
        #scores.append(score)
        wine_scores.append([min_sample, epsilon, score])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[85], line 19
     16 cluster = dbscan.fit_predict(results[['one', 'two', 'three']])
     18 # calculate silouette score here
---> 19 score = metrics.silhouette_score(results[['one', 'two', 'three']], cluster)
     21 # Append the values into a 3D array
     22 #scores.append(score)
     23 wine_scores.append([min_sample, epsilon, score])

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\_param_v
alidation.py:211, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwa
rgs)
    205 try:
    206     with config_context(
    207         skip_parameter_validation=(
    208             prefer_skip_nested_validation or global_skip_validation
    209         )
    210     ):
--> 211         return func(*args, **kwargs)
    212 except InvalidParameterError as e:
    213     # When the function is just a wrapper around an estimator, we allow
    214     # the function to delegate validation to the estimator, but we replace
    215     # the name of the estimator by the name of the function in the error
    216     # message to avoid confusion.
    217     msg = re.sub(
    218         r"parameter of \w+ must be",
    219         f"parameter of {func.__qualname__} must be",
    220         str(e),
    221     )

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:131, in silhouette_score(X, labels, metric, sample_size, random_st
ate, **kwds)
    129     else:
    130         X, labels = X[indices], labels[indices]
--> 131 return np.mean(silhouette_samples(X, labels, metric=metric, **kwds))

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\_param_v
alidation.py:184, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwa
rgs)
    182 global_skip_validation = get_config()["skip_parameter_validation"]
    183 if global_skip_validation:
--> 184     return func(*args, **kwargs)
    186 func_sig = signature(func)
    188 # Map *args/**kwargs to the function signature

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:277, in silhouette_samples(X, labels, metric, **kwds)
    275 n_samples = len(labels)
    276 label_freqs = np.bincount(labels)
--> 277 check_number_of_labels(len(le.classes_), n_samples)
    279 kwds["metric"] = metric
    280 reduce_func = functools.partial(
    281     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    282 )

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:37, in check_number_of_labels(n_labels, n_samples)
     26 """Check that number of labels are valid.
```

```
     27
     28 Parameters
   (...)
     34     Number of samples.
     35 """
     36 if not 1 < n_labels < n_samples:
---> 37     raise ValueError(
     38         "Number of labels is %d. Valid values are 2 to n_samples - 1 (inclusi
ve)"
     39         % n_labels
     40     )

ValueError: Number of labels is 178. Valid values are 2 to n_samples - 1 (inclusive)
```

In [ ]:
```
dbscan = DBSCAN(eps=0.3, min_samples=5)
#winesmall = wine.data[:,10:13]
cluster = dbscan.fit_predict(wine_pca.data)

#cluster = dbscan.fit_predict(winesmall)
#cluster = dbscan.fit_predict(wine.data)
#winedf.cluster.value_counts()
```

In [ ]:
```
results = pd.DataFrame(wine_pca, columns=['one', 'two', 'three'])
results['cluster'] = cluster
#silhouette_score(wine_pca, cluster)
results
```

Out[ ]:

	one	two	three	cluster
0	318.562979	21.492131	3.130735	0
1	303.097420	-5.364718	6.822835	1
2	438.061133	-6.537309	-1.113223	2
3	733.240139	0.192729	-0.917257	3
4	-11.571428	18.489995	-0.554422	4
...	...	...	...	...
173	-6.980211	-4.541137	-2.474707	173
174	3.131605	2.335191	-4.309931	174
175	88.458074	18.776285	-2.237577	175
176	93.456242	18.670819	-1.788392	176
177	-186.943190	-0.213331	-5.630510	177

178 rows × 4 columns

In [ ]:
```
#metrics.silhouette_score(XX[['lon', 'lat', 'alt']], XX.cluster)
metrics.silhouette_score(results[['one', 'two', 'three']], cluster)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[87], line 2
      1 #metrics.silhouette_score(XX[['lon', 'lat', 'alt']], XX.cluster)
----> 2 metrics.silhouette_score(results[['one', 'two', 'three']], cluster)

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\_param_v
alidation.py:211, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwa
rgs)
    205 try:
    206     with config_context(
    207         skip_parameter_validation=(
    208             prefer_skip_nested_validation or global_skip_validation
    209         )
    210     ):
--> 211         return func(*args, **kwargs)
    212 except InvalidParameterError as e:
    213     # When the function is just a wrapper around an estimator, we allow
    214     # the function to delegate validation to the estimator, but we replace
    215     # the name of the estimator by the name of the function in the error
    216     # message to avoid confusion.
    217     msg = re.sub(
    218         r"parameter of \w+ must be",
    219         f"parameter of {func.__qualname__} must be",
    220         str(e),
    221     )

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:131, in silhouette_score(X, labels, metric, sample_size, random_st
ate, **kwds)
    129     else:
    130         X, labels = X[indices], labels[indices]
--> 131 return np.mean(silhouette_samples(X, labels, metric=metric, **kwds))

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\_param_v
alidation.py:184, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwa
rgs)
    182 global_skip_validation = get_config()["skip_parameter_validation"]
    183 if global_skip_validation:
--> 184     return func(*args, **kwargs)
    186 func_sig = signature(func)
    188 # Map *args/**kwargs to the function signature

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:277, in silhouette_samples(X, labels, metric, **kwds)
    275 n_samples = len(labels)
    276 label_freqs = np.bincount(labels)
--> 277 check_number_of_labels(len(le.classes_), n_samples)
    279 kwds["metric"] = metric
    280 reduce_func = functools.partial(
    281     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    282 )

File c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\metrics\cluste
r\_unsupervised.py:37, in check_number_of_labels(n_labels, n_samples)
     26 """Check that number of labels are valid.
     27
     28 Parameters
   (...)
     34     Number of samples.
```

```
     35 """
     36 if not 1 < n_labels < n_samples:
---> 37     raise ValueError(
     38         "Number of labels is %d. Valid values are 2 to n_samples - 1 (inclusi
ve)"
     39         % n_labels
     40     )

ValueError: Number of labels is 178. Valid values are 2 to n_samples - 1 (inclusive)
```

It seems I am unable to get the DBSCAN and silhouette score to work with this data. I tried several different data sets and could not get valid results. Always I had a single classifier, and could not get scores.

Just for the effort to do something, I will do a KMeans Cluster here instead, as that seems to work much better with the data.

In [ ]:
```
pca_data = pd.DataFrame(wine_pca, columns=['one', 'two', 'three'])

N = 3
from sklearn.cluster import KMeans
km = KMeans(n_clusters=N, random_state=1)
km.fit(pca_data)
pca_data['cluster'] = km.predict(pca_data)
pca_data.cluster.value_counts()
```

```
c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:
1412: 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
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\jomors\AppData\Local\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:
1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when th
ere are less chunks than available threads. You can avoid it by setting the environme
nt variable OMP_NUM_THREADS=1.
  warnings.warn(
```

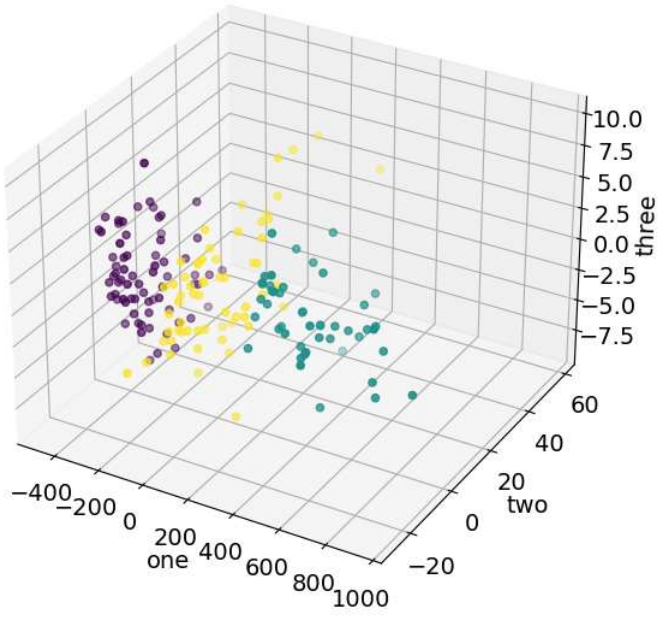
Out[ ]:
```
cluster
0    69
2    62
1    47
Name: count, dtype: int64
```

In [ ]:
```
# 3D Plot
fig = plt.figure()
ax = plt.axes(projection='3d')

ax.scatter3D(pca_data.one, pca_data.two, pca_data.three, c=pca_data.cluster, cmap='vir
ax.set_xlabel('one')
ax.set_ylabel('two')
ax.set_zlabel('three')

plt.show()
```

Figure


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
three
60
40
20
0
−20
two
−400
−200
0
200
400
600
800
1000
one