Clustering

In [57]:
```
import pandas as pd
import numpy as np
# allow plots to appear in the notebook
%matplotlib notebook
import matplotlib.pyplot as plt
import seaborn
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['font.size'] = 14
plt.rcParams['figure.figsize'] = (8.0, 5.0)
```

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 [85]:
```
X = pd.read_csv('../data/3D_spatial_network.txt.gz', header=None, names=['osm', 'lat'
X = X.drop(['osm'], axis=1).sample(1000)
X.head()
```

Out[85]:

	lat	lon	alt
393597	8.291569	56.878087	12.596548
342317	9.796822	56.814117	22.395665
157827	8.781154	56.738636	32.078124
175178	10.137708	56.809133	31.997393
222729	9.718259	57.379248	9.278766

In [14]:
```
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[14]:

	lat	lon	alt
2118	-0.440213	-1.572362	0.895279
153569	-1.377963	-1.484812	-0.287326
291441	-1.278020	-1.631540	1.537482
197666	0.124493	1.146735	0.008990
434214	0.259403	-0.250058	-0.605039

In [15]:

```
from sklearn.cluster import DBSCAN
from sklearn import metrics
```

In [59]:

```
min_samples = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
epsilons = np.arange(0.05, 0.51, 0.01)
all_scores = []
for ms in min_samples:
    scores = []
    for e in epsilons:
        dbscan = DBSCAN(eps = e, min_samples = ms)
        labels = dbscan.fit_predict(XX[['lon', 'lat', 'alt']])

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

        except ValueError:
            print("NULL")

        scores.append(score)

    all_scores.append(scores)
```

```
NULL
NULL
NULL
NULL
NULL
```

In [60]:

```
all_scores
```

Out[60]:
```
[[0.1429720480594681,
  0.16493398993073244,
  0.18167158925613786,
  0.21946115674989006,
  0.24125681795192971,
  0.2560401432362994,
  0.2718365966283104,
  0.27736756410885904,
  0.26663790753468647,
  0.2667010625915522,
  0.26214836426437715,
  0.2632335909395823,
  0.23459406301582109,
  0.1759506977603093,
  0.15169241414735007,
  0.15343934814415836,
  0.07124840442716374,
  0.039476478074380636,
  0.037329477557041316,
  -0.0019096687056215705,
  -0.11684568709343082,
  -0.1804782769848567,
  -0.19025316630128303,
  -0.19928344843170775,
  -0.2965500194707005,
  -0.4138574328848015,
  -0.41740241738588807,
  -0.46757139418864224,
  -0.49454121981402993,
  -0.5070445970393045,
  -0.5124416284678023,
  -0.35938581738852826,
  -0.27643054929061645,
  -0.2766729619400737,
  -0.26709237507525496,
  -0.2544243326163745,
  -0.25274810881003573,
  -0.236966706674665,
  -0.18748196339398301,
  -0.18698023641887535,
  -0.1905148354397641,
  -0.1905148354397641,
  -0.1905148354397641,
  -0.1905148354397641,
  -0.1905148354397641,
  -0.1808746219797845],
 [-0.44585713404639715,
  -0.377692820717411,
  -0.3235998731921976,
  -0.23572703548598098,
  -0.14653327271516622,
  -0.07511576236360597,
  -0.03171131449508599,
  0.007831435002657653,
  0.03818362806584487,
  0.06984019296069453,
  0.0988357102207119,
  0.13823361360900926,
  0.14927330769040603,
  0.13128345343424444,
```

```
 0.11900395452545344,
 0.14246538707409676,
 0.07063375066415437,
 0.06229309461323855,
 0.06124748234974148,
 0.023559952594499278,
 -0.1016933247874809,
 -0.1581388669749202,
 -0.13481270198629092,
 -0.12362097595509036,
 -0.2106467182295879,
 -0.3487246958625422,
 -0.34196312380510324,
 -0.3852790143370273,
 -0.4038232213408012,
 -0.407372510125861,
 -0.33497723693796183,
 -0.12766900226493813,
 -0.12803318649128,
 -0.1285980981643968,
 -0.1288249047054631,
 -0.12675767546911856,
 -0.12171713888299927,
 -0.12184678193706391,
 -0.12810274138721764,
 -0.12695567414159836,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.1257275241275713],
[-0.4636398080178269,
 -0.4566156302435946,
 -0.44718628791674625,
 -0.40937282363588023,
 -0.35951411229785823,
 -0.30040626117162916,
 -0.26261440641814116,
 -0.2130594388205667,
 -0.16999579001397905,
 -0.11183527182306065,
 -0.07688263216884902,
 -0.04077558766489591,
 0.003999340170026871,
 -0.0029367745797238484,
 0.00038258962345594493,
 0.03375612181606558,
 -0.008098696865707208,
 0.004067303819267396,
 0.012838115038890386,
 -0.03358044209222803,
 -0.13920810222799962,
 -0.1855673114640855,
 -0.1476917458411725,
 -0.11874128779573073,
 -0.18217733608044062,
 -0.3028084879397919,
 -0.2765569522763602,
 -0.30821910829947263,
```

```
 -0.31680702373977865,
 -0.31459497322138863,
 -0.171805572898192,
 -0.13069742079832838,
 -0.13085804665920198,
 -0.13121129378105706,
 -0.13115705582581139,
 -0.12851719708002216,
 -0.12320056458858104,
 -0.12327609444517719,
 -0.12909300720987216,
 -0.1282248720722851,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.12996049189439052,
 -0.1257275241275713],
[-0.3429829080980562,
 -0.38836336213151035,
 -0.42258340155405044,
 -0.4455374973931988,
 -0.4105790024705078,
 -0.38181693260192723,
 -0.35839313636898634,
 -0.3225349893029172,
 -0.28481974351038253,
 -0.2509277251323321,
 -0.20332563215152868,
 -0.15833890399779124,
 -0.10842152373685678,
 -0.11942895177278062,
 -0.1085528149613384,
 -0.0829938403767965,
 -0.08829863699058295,
 -0.03946499024961483,
 -0.02128782169338406,
 -0.06853355073598177,
 -0.16701797181396905,
 -0.22432860569909036,
 -0.17244542856381997,
 -0.13821146520284702,
 -0.2037263342184978,
 -0.323407665845563,
 -0.2777310923053042,
 -0.261454257396005,
 -0.29926923927045856,
 -0.29950483555419455,
 -0.13483206631513606,
 -0.08463905867794225,
 -0.08318265326346738,
 -0.07016264343808157,
 -0.07007087428898003,
 -0.11229481543157573,
 -0.10690516597358127,
 -0.10904356802329888,
 -0.10904356802329888,
 -0.10907742284205373,
 -0.1104777239283307,
 -0.1104777239283307,
```

```
 -0.12436112599422791,
 -0.12438034766145381,
 -0.12438034766145381,
 -0.11409640731467675],
[-0.346121037489257,
 -0.3148746948378042,
 -0.31384408219770904,
 -0.3017992824627466,
 -0.28115606583317865,
 -0.3141066402164979,
 -0.2848780359364851,
 -0.2570006112085126,
 -0.2922966498573396,
 -0.3011132543960506,
 -0.2731400139587545,
 -0.2198359541115795,
 -0.20484198595623304,
 -0.1865334089666986,
 -0.15288025296494223,
 -0.12811402557049686,
 -0.11183700097405727,
 -0.1157689445347473,
 -0.0770907374850573,
 -0.07043475614343687,
 -0.12399960925894629,
 -0.1334417549609433,
 -0.18144774219099902,
 -0.16813820863485762,
 -0.15274903660709027,
 -0.18816112507584962,
 -0.16067049095509417,
 -0.13763162083123606,
 -0.2731156776924738,
 -0.2712927512091043,
 -0.12688814734335402,
 -0.12490833170871636,
 0.037649640369429,
 0.04069072361484621,
 0.043409748791549914,
 0.04636574937522271,
 0.049161072352750544,
 -0.0761251528622603,
 -0.07665618427513374,
 -0.10904356802329888,
 -0.1104629605154662,
 -0.1104629605154662,
 -0.1104777239283307,
 -0.11055523914473786,
 -0.11055523914473786,
 -0.1112896809946638],
[0.03494165406147326,
 -0.24142937905465334,
 -0.2643847045385302,
 -0.2933873024971091,
 -0.2734042970342209,
 -0.2885928647886776,
 -0.27550548298493954,
 -0.26870503227106185,
 -0.25247996728232813,
 -0.2288743608115638,
```

```
 -0.28617061340067057,
 -0.2335655759102645,
 -0.23768937374606974,
 -0.24848777985737144,
 -0.21065198494572143,
 -0.19351277675025083,
 -0.1800691945156537,
 -0.13228008997887183,
 -0.12424355912622635,
 -0.07104693397540401,
 -0.13381763210360623,
 -0.1083669310766594,
 -0.05402884721812087,
 -0.033332034997206085,
 0.0031551119159365947,
 -0.14098129845826807,
 -0.13097879797513798,
 -0.12667048401368258,
 -0.15540703187857452,
 -0.254708728341456,
 -0.24925948662260133,
 -0.24775394824618877,
 -0.23939159017734848,
 -0.2211006595265945,
 -0.22196141501038716,
 -0.22200202556812118,
 0.04913387532233257,
 0.050392148080013015,
 0.05026505392565656,
 0.05026505392565656,
 0.045849056906699416,
 0.045849056906699416,
 0.04784996013785795,
 0.04784996013785795,
 0.05019390110469622,
 0.04730601177323088],
[0.01880993583879694,
 0.027616324390340993,
 -0.2750325830356104,
 -0.2506355746004277,
 -0.21611403570836724,
 -0.27190324868182053,
 -0.26245221807147473,
 -0.24454960705801915,
 -0.23526173027733677,
 -0.22814753123736198,
 -0.24518093595990154,
 -0.2290361302693935,
 -0.18070744113093168,
 -0.18156188454215436,
 -0.2511975271669084,
 -0.23033725471244051,
 -0.20788457780879355,
 -0.17746210085272912,
 -0.16335658172636466,
 -0.12449704902373031,
 -0.126092020654594,
 -0.1891800826813939,
 -0.16662151087722105,
 -0.10101761604334483,
```

```
 -0.04344910570538677,
 -0.15309596988944582,
 -0.14122242411273508,
 -0.1382545549481973,
 -0.12602703061882908,
 -0.11204180679201806,
 -0.15964226479676644,
 -0.15996708983777141,
 -0.11771894693785213,
 0.01194346503340349,
 0.012531063769395206,
 0.01820673103135468 7,
 0.03092875314021777,
 0.039891288265107926,
 0.042986175313764745,
 0.042986175313764745,
 0.04174334314148857,
 0.043353917443931245,
 0.046231213301419334,
 0.04784996013785795,
 0.04784996013785795,
 0.04730601177323088],
[0.01880993583879694,
 0.01880993583879694,
 0.03107316244436622,
 0.03913693253592997,
 0.004555337103453274,
 -0.24108121451300968,
 -0.25467158891868485,
 -0.24267179001539285,
 -0.22711818102050985,
 -0.21635874195421156,
 -0.2087307053849957,
 -0.29360163577322235,
 -0.22602677150042927,
 -0.23314478517219378,
 -0.19618968993963187,
 -0.21907674137495892,
 -0.19695193918163595,
 -0.19041561571469356,
 -0.1689806060669074,
 -0.15961807909467382,
 -0.14888003387925347,
 -0.11757263534908428,
 -0.09226091527458676,
 -0.0493297434919437,
 -0.03265386254331351,
 -0.037055098390725814,
 -0.09128027315117561,
 -0.10748826532634034,
 -0.1148369125383745,
 -0.13607783169866655,
 -0.07411501030700414,
 -0.21624519761152874,
 -0.20649762486862164,
 -0.1559100001747144,
 -0.15408622643149494,
 0.013454259989108944,
 0.012465862660999407,
 0.0329854192093857,
```

```
 0.03540282688373866,
 0.03761310739180322,
 0.04343742110630606,
 0.04318154432880078,
 0.0444484070682789,
 0.0444484070682789,
 0.045849056906699416,
 0.045134892876576245],
[0.045134892876576245,
 0.045134892876576245,
 0.017084037683563008,
 0.03913693253592997,
 0.04884998947926799,
 0.003991491854499655,
 -0.22982649819550605,
 -0.28430794126877945,
 -0.24360505511895653,
 -0.21976304217353448,
 -0.21631462018681338,
 -0.27356276618924874,
 -0.21954732297174073,
 -0.24589449838828256,
 -0.2228667943105415,
 -0.19629042301445895,
 -0.15341176965585418,
 -0.20373197635882948,
 -0.1834550073912677,
 -0.11825088765510637,
 -0.14629561555141532,
 -0.1192012525794607,
 -0.12590702467103346,
 -0.11333755178598157,
 -0.060207092552693825,
 -0.017679994979973048,
 -0.08369203419126402,
 -0.1732882590241293,
 -0.13162111283546318,
 -0.08198719745127574,
 0.00726779271145872,
 0.020193726040761858,
 0.030717046513128173,
 -0.02486690636905354,
 -0.14867403870000817,
 -0.07381450453468952,
 0.02971144808694271,
 0.019148422603736094,
 0.026647598283072782,
 0.03204998209017488,
 0.043550230825837824,
 0.04430527857027793,
 0.04430527857027793,
 0.04430527857027793,
 0.04592112265721499,
 0.04295461763112808],
[0.04295461763112808,
 0.04295461763112808,
 0.04295461763112808,
 0.020172982078140868,
 0.045391093436643604,
 0.001924665762453822,
```

```
-0.23580724989579263,
-0.23128441885458909,
-0.2123940958609421З,
-0.24487236506716328,
-0.21053392589652103,
-0.21233630244128643,
-0.26349774364854023,
-0.22781225819831818,
-0.25710833318847665,
-0.215984855117687,
-0.191427861939241,
-0.13556217045248622,
-0.17748049869385635,
-0.14046649100229477,
-0.13830121447268,
-0.09592912843484529,
-0.08863672441228826,
-0.1161864031209964,
-0.10031421536759015,
-0.07669204557211574,
-0.04944283753748759,
0.010528996914505144,
-0.04695404687808589,
-0.03632720048455556,
-0.07927231924863967,
-0.05325001277137058,
-0.016659460486898914,
0.0291405813632043,
-0.132039737094006,
-0.12476895844254701,
-0.12009666696176252,
0.025360273727905256,
0.010168655323542523,
0.017246264001192887,
0.025630828798847545,
0.025474395540269618,
0.03675722527057428,
0.03675722527057428,
0.04175573636606895,
0.041289376470531346]]
```

In [61]:

```
sc_df = pd.DataFrame(all_scores, columns = epsilons).T
sc_df.columns = ["one", "two", "three", "four", "five", "six", "seven", "eight", "nin
```

In [62]:

```
sc_df
```

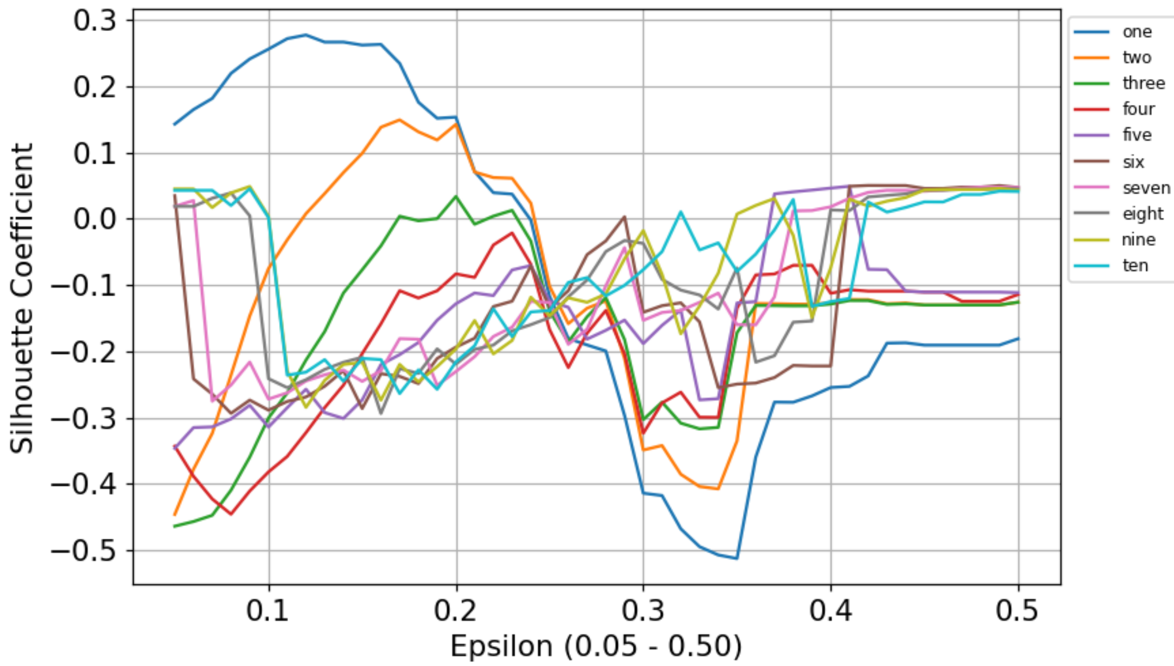
Out[62]:

	one	two	three	four	five	six	seven	eight	nine
0.05	0.142972	-0.445857	-0.463640	-0.342983	-0.346121	0.034942	0.018810	0.018810	0.045135
0.06	0.164934	-0.377693	-0.456616	-0.388363	-0.314875	-0.241429	0.027616	0.018810	0.045135
0.07	0.181672	-0.323600	-0.447186	-0.422583	-0.313844	-0.264385	-0.275033	0.031073	0.017084
0.08	0.219461	-0.235727	-0.409373	-0.445537	-0.301799	-0.293387	-0.250636	0.039137	0.039137
0.09	0.241257	-0.146533	-0.359514	-0.410579	-0.281156	-0.273404	-0.216114	0.004555	0.048850
0.10	0.256040	-0.075116	-0.300406	-0.381817	-0.314107	-0.288593	-0.271903	-0.241081	0.003991
0.11	0.271837	-0.031711	-0.262614	-0.358393	-0.284878	-0.275505	-0.262452	-0.254672	-0.229826
0.12	0.277368	0.007831	-0.213059	-0.322535	-0.257001	-0.268705	-0.244550	-0.242672	-0.284308
0.13	0.266638	0.038184	-0.169996	-0.284820	-0.292297	-0.252480	-0.235262	-0.227118	-0.243605
0.14	0.266701	0.069840	-0.111835	-0.250928	-0.301113	-0.228874	-0.228148	-0.216359	-0.219763
0.15	0.262148	0.098836	-0.076883	-0.203326	-0.273140	-0.286171	-0.245181	-0.208731	-0.216315
0.16	0.263234	0.138234	-0.040776	-0.158339	-0.219836	-0.233566	-0.229036	-0.293602	-0.273563
0.17	0.234594	0.149273	0.003999	-0.108422	-0.204842	-0.237689	-0.180707	-0.226027	-0.219547
0.18	0.175951	0.131283	-0.002937	-0.119429	-0.186533	-0.248488	-0.181562	-0.233145	-0.245894
0.19	0.151692	0.119004	0.000383	-0.108553	-0.152880	-0.210652	-0.251198	-0.196190	-0.222867
0.20	0.153439	0.142465	0.033756	-0.082994	-0.128114	-0.193513	-0.230337	-0.219077	-0.196290
0.21	0.071248	0.070634	-0.008099	-0.088299	-0.111837	-0.180069	-0.207885	-0.196952	-0.153412
0.22	0.039476	0.062293	0.004067	-0.039465	-0.115769	-0.132280	-0.177462	-0.190416	-0.203732
0.23	0.037329	0.061247	0.012838	-0.021288	-0.077091	-0.124244	-0.163357	-0.168981	-0.183455
0.24	-0.001910	0.023560	-0.033580	-0.068534	-0.070435	-0.071047	-0.124497	-0.159618	-0.118251
0.25	-0.116846	-0.101693	-0.139208	-0.167018	-0.124000	-0.133818	-0.126092	-0.148880	-0.146296
0.26	-0.180478	-0.158139	-0.185567	-0.224329	-0.133442	-0.108367	-0.189180	-0.117573	-0.119201
0.27	-0.190253	-0.134813	-0.147692	-0.172445	-0.181448	-0.054029	-0.166622	-0.092261	-0.125907
0.28	-0.199283	-0.123621	-0.118741	-0.138211	-0.168138	-0.033332	-0.101018	-0.049330	-0.113338
0.29	-0.296550	-0.210647	-0.182177	-0.203726	-0.152749	0.003155	-0.043449	-0.032654	-0.060207
0.30	-0.413857	-0.348725	-0.302808	-0.323408	-0.188161	-0.140981	-0.153096	-0.037055	-0.017680
0.31	-0.417402	-0.341963	-0.276557	-0.277731	-0.160670	-0.130979	-0.141222	-0.091280	-0.083692
0.32	-0.467571	-0.385279	-0.308219	-0.261454	-0.137632	-0.126670	-0.138255	-0.107488	-0.173288
0.33	-0.494541	-0.403823	-0.316807	-0.299269	-0.273116	-0.155407	-0.126027	-0.114837	-0.131621
0.34	-0.507045	-0.407373	-0.314595	-0.299505	-0.271293	-0.254709	-0.112042	-0.136078	-0.081987
0.35	-0.512442	-0.334977	-0.171806	-0.134832	-0.126888	-0.249259	-0.159642	-0.074115	0.007268
0.36	-0.359386	-0.127669	-0.130697	-0.084639	-0.124908	-0.247754	-0.159967	-0.216245	0.020194
0.37	-0.276431	-0.128033	-0.130858	-0.083183	0.037650	-0.239392	-0.117719	-0.206498	0.030717

	one	two	three	four	five	six	seven	eight	nine
0.38	-0.276673	-0.128598	-0.131211	-0.070163	0.040691	-0.221101	0.011943	-0.155910	-0.024867
0.39	-0.267092	-0.128825	-0.131157	-0.070071	0.043410	-0.221961	0.012531	-0.154086	-0.148674
0.40	-0.254424	-0.126758	-0.128517	-0.112295	0.046366	-0.222002	0.018207	0.013454	-0.073815
0.41	-0.252748	-0.121717	-0.123201	-0.106905	0.049161	0.049134	0.030929	0.012466	0.029711
0.42	-0.236967	-0.121847	-0.123276	-0.109044	-0.076125	0.050392	0.039891	0.032985	0.019148
0.43	-0.187482	-0.128103	-0.129093	-0.109044	-0.076656	0.050265	0.042986	0.035403	0.026648
0.44	-0.186980	-0.126956	-0.128225	-0.109077	-0.109044	0.050265	0.042986	0.037613	0.032050
0.45	-0.190515	-0.129960	-0.129960	-0.110478	-0.110463	0.045849	0.041743	0.043437	0.043550
0.46	-0.190515	-0.129960	-0.129960	-0.110478	-0.110463	0.045849	0.043354	0.043182	0.044305
0.47	-0.190515	-0.129960	-0.129960	-0.124361	-0.110478	0.047850	0.046231	0.044448	0.044305
0.48	-0.190515	-0.129960	-0.129960	-0.124380	-0.110555	0.047850	0.047850	0.044448	0.044305
0.49	-0.190515	-0.129960	-0.129960	-0.124380	-0.110555	0.050194	0.047850	0.045849	0.045921
0.50	-0.180875	-0.125728	-0.125728	-0.114096	-0.111290	0.047306	0.047306	0.045135	0.042955

In [63]:

```
plt.figure()
plt.plot(sc_df["one"])
plt.plot(sc_df["two"])
plt.plot(sc_df["three"])
plt.plot(sc_df["four"])
plt.plot(sc_df["five"])
plt.plot(sc_df["six"])
plt.plot(sc_df["seven"])
plt.plot(sc_df["eight"])
plt.plot(sc_df["nine"])
plt.plot(sc_df["ten"])
plt.legend(sc_df, bbox_to_anchor=(1, 1), loc='upper left', fontsize = "8")
plt.xlabel('Epsilon (0.05 - 0.50)')
plt.ylabel('Silhouette Coefficient')
plt.grid(True)
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 [75]:
```
wine = pd.read_csv('../data/WineQT.csv', names=['fixedacid', 'volatileacid','citricac
wine = wine.iloc[1: , :]
wine
```

Out[75]:

	fixedacid	volatileacid	citricacid	residualsugar	chlorides	freesulfurdio	totalsulfurdio	density
1	7.4	0.7	0	1.9	0.076	11	34	0.9978
2	7.8	0.88	0	2.6	0.098	25	67	0.9968
3	7.8	0.76	0.04	2.3	0.092	15	54	0.997
4	11.2	0.28	0.56	1.9	0.075	17	60	0.998
5	7.4	0.7	0	1.9	0.076	11	34	0.9978
...	...	...	...	...	...	...	...	..
1138	5.4	0.74	0.09	1.7	0.089	16	26	0.99402
1139	6.3	0.51	0.13	2.3	0.076	29	40	0.99574
1140	6.8	0.62	0.08	1.9	0.068	28	38	0.99651
1141	6.2	0.6	0.08	2	0.09	32	44	0.9949
1142	5.9	0.55	0.1	2.2	0.062	39	51	0.99512

1142 rows × 12 columns

In [76]:

```
wine.dtypes
```

Out[76]:

```
fixedacid         object
volatileacid      object
citricacid        object
residualsugar     object
chlorides         object
freesulfurdio     object
totalsulfurdio    object
density           object
pH                object
sulphates         object
alcohol           object
quality           object
dtype: object
```

In [77]:

```
wine["fixedacid"] = pd.to_numeric(wine.fixedacid, errors='coerce')
wine["volatileacid"] = pd.to_numeric(wine.volatileacid, errors='coerce')
wine["citricacid"] = pd.to_numeric(wine.citricacid, errors='coerce')
wine["residualsugar"] = pd.to_numeric(wine.residualsugar, errors='coerce')
wine["chlorides"] = pd.to_numeric(wine.chlorides, errors='coerce')
wine["freesulfurdio"] = pd.to_numeric(wine.freesulfurdio, errors='coerce')
wine["totalsulfurdio"] = pd.to_numeric(wine.totalsulfurdio, errors='coerce')
wine["density"] = pd.to_numeric(wine.density, errors='coerce')
wine["pH"] = pd.to_numeric(wine.pH, errors='coerce')
wine["sulphates"] = pd.to_numeric(wine.sulphates, errors='coerce')
wine["alcohol"] = pd.to_numeric(wine.alcohol, errors='coerce')
wine["quality"] = pd.to_numeric(wine.quality, errors='coerce')
```

In [78]:

```
wine.dtypes
```

Out[78]:
```
fixedacid         float64
volatileacid      float64
citricacid        float64
residualsugar     float64
chlorides         float64
freesulfurdio     float64
totalsulfurdio    float64
density           float64
pH                float64
sulphates         float64
alcohol           float64
quality             int64
dtype: object
```

In [79]:
```
wine_x = wine.copy()
wine_x['fixedacid'] = (wine.fixedacid - wine.fixedacid.mean())/wine.fixedacid.std()
wine_x['volatileacid'] = (wine.volatileacid - wine.volatileacid.mean())/wine.volatile
wine_x['citricacid'] = (wine.citricacid - wine.citricacid.mean())/wine.citricacid.std
wine_x['residualsugar'] = (wine.residualsugar - wine.residualsugar.mean())/wine.resid
wine_x['chlorides'] = (wine.chlorides - wine.chlorides.mean())/wine.chlorides.std()
wine_x['freesulfurdio'] = (wine.freesulfurdio - wine.freesulfurdio.mean())/wine.frees
wine_x['totalsulfurdio'] = (wine.totalsulfurdio - wine.totalsulfurdio.mean())/wine.to
wine_x['density'] = (wine.density - wine.density.mean())/wine.density.std()
wine_x['pH'] = (wine.pH - wine.pH.mean())/wine.pH.std()
wine_x['sulphates'] = (wine.sulphates - wine.sulphates.mean())/wine.sulphates.std()
wine_x['alcohol'] = (wine.alcohol - wine.alcohol.mean())/wine.alcohol.std()
wine_x['quality'] = (wine.quality - wine.quality.mean())/wine.quality.std()
wine_x.head()
```

Out[79]:

	fixedacid	volatileacid	citricacid	residualsugar	chlorides	freesulfurdio	totalsulfurdio	density
1	-0.522767	0.939229	-1.364832	-0.466388	-0.231420	-0.449177	-0.363344	0.554899
2	-0.293790	1.941008	-1.364832	0.049676	0.233827	0.917545	0.642863	0.035567
3	-0.293790	1.273155	-1.161501	-0.171494	0.106942	-0.058685	0.246479	0.139433
4	1.652513	-1.398256	1.481810	-0.466388	-0.252567	0.136561	0.429425	0.658766
5	-0.522767	0.939229	-1.364832	-0.466388	-0.231420	-0.449177	-0.363344	0.554899

In [82]:
```
min_samples = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
epsilons = np.arange(0.05, 0.51, 0.01)
all_scores = []
for ms in min_samples:
    scores = []
    for e in epsilons:
        dbscan = DBSCAN(eps = e, min_samples = ms)
        labels = dbscan.fit_predict(wine_x[['fixedacid', 'volatileacid','citricacid',

        # calculate silouette score here
        try:
            score = metrics.silhouette_score(wine_x[['fixedacid', 'volatileacid','cit

        except ValueError:
            pass

        scores.append(score)
```

```
    all_scores.append(scores)
```

In [83]:

```
all_scores
```

```
Out[83]: [[0.21272127210230227,
  0.2144390455007918,
  0.2144390455007918,
  0.2144390455007918,
  0.2144390455007918,
  0.2144390455007918,
  0.2144390455007918,
  0.21507165852514076,
  0.21672910173248144,
  0.21672910173248144,
  0.21794795189864422,
  0.2195543546701948,
  0.2200695182149805,
  0.22207203606217346,
  0.22207203606217346,
  0.22561999660070883,
  0.22561999660070883,
  0.22561999660070883,
  0.22714046003858346,
  0.22714046003858346,
  0.22913451854823905,
  0.23029653139098655,
  0.23029653139098655,
  0.23029653139098655,
  0.23029653139098655,
  0.23057587092650975,
  0.23057587092650975,
  0.23057587092650975,
  0.23057587092650975,
  0.23287132019419382,
  0.2339771563263904,
  0.23568617916048804,
  0.23568617916048804,
  0.23680220526740262,
  0.23680220526740262,
  0.2360164156184699,
  0.2360164156184699,
  0.2360164156184699,
  0.23627875838148096,
  0.23735040780830594,
  0.23735040780830594,
  0.23670474178742704,
  0.23670474178742704,
  0.23670474178742704,
  0.23696162457060416,
  0.23943760512627024],
 [-0.2615586699278065,
  -0.25850218095017874,
  -0.25850218095017874,
  -0.25850218095017874,
  -0.25850218095017874,
  -0.25850218095017874,
  -0.25850218095017874,
  -0.2567797624938396,
  -0.25442443068735837,
  -0.25442443068735837,
  -0.2526202664460744,
  -0.250254623459601,
  -0.24887638660526823,
  -0.24427566060469633,
```

```
 -0.24427566060469633,
 -0.2386974198421971,
 -0.2386974198421971,
 -0.2386974198421971,
 -0.23626897075478226,
 -0.23626897075478226,
 -0.2333338265613444,
 -0.2310029426359906,
 -0.2310029426359906,
 -0.2310029426359906,
 -0.2310029426359906,
 -0.22988929826208165,
 -0.22988929826208165,
 -0.22988929826208165,
 -0.22988929826208165,
 -0.22581520034821426,
 -0.22400845548691295,
 -0.21885324389313962,
 -0.21885324389313962,
 -0.2164116699226785,
 -0.2164116699226785,
 -0.21582798608060363,
 -0.21582798608060363,
 -0.21582798608060363,
 -0.21467925834843346,
 -0.21101849937897624,
 -0.21101849937897624,
 -0.2104932548789519,
 -0.2104932548789519,
 -0.2104932548789519,
 -0.20803308995542233,
 -0.20373917375515516],
[-0.37609789719591663,
 -0.37609789719591663,
 -0.37609789719591663,
 -0.37609789719591663,
 -0.37609789719591663,
 -0.37609789719591663,
 -0.37609789719591663,
 -0.37695871660169544,
 -0.37695871660169544,
 -0.37695871660169544,
 -0.37695871660169544,
 -0.37695871660169544,
 -0.3830043403858295,
 -0.3830043403858295,
 -0.3830043403858295,
 -0.3817324253828289,
 -0.3817324253828289,
 -0.3817324253828289,
 -0.3817324253828289,
 -0.3817324253828289,
 -0.38411916661139034,
 -0.38411916661139034,
 -0.38411916661139034,
 -0.38411916661139034,
 -0.38411916661139034,
 -0.3876665581011336,
 -0.3876665581011336,
 -0.3876665581011336,
```

```
 -0.3876665581011336,
 -0.3876665581011336,
 -0.3876665581011336,
 -0.38700482620993554,
 -0.38700482620993554,
 -0.38700482620993554,
 -0.38700482620993554,
 -0.38412243865865275,
 -0.38412243865865275,
 -0.38412243865865275,
 -0.39925476516457814,
 -0.39925476516457814,
 -0.39925476516457814,
 -0.39968884804077376,
 -0.39968884804077376,
 -0.39968884804077376,
 -0.40004803977699394,
 -0.40074149155016425],
[-0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.25737106653123987,
 -0.28886599390756734,
 -0.28886599390756734,
 -0.28886599390756734,
 -0.28886599390756734,
 -0.28886599390756734,
 -0.28886599390756734,
 -0.30532997477543233,
```

```
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233],
        [-0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.30532997477543233,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364],
        [-0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
         -0.12266326478526364,
```

```
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364],
[-0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
```

```
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364],
[-0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
```

```
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364],
[-0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364],
[-0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
```

```
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364,
 -0.12266326478526364]]
```

In [88]:

```
wine_sc_df = pd.DataFrame(all_scores, columns = epsilons).T
wine_sc_df.columns = ["one", "two", "three", "four", "five", "six", "seven", "eight",
```

In [89]:

```
wine_sc_df
```

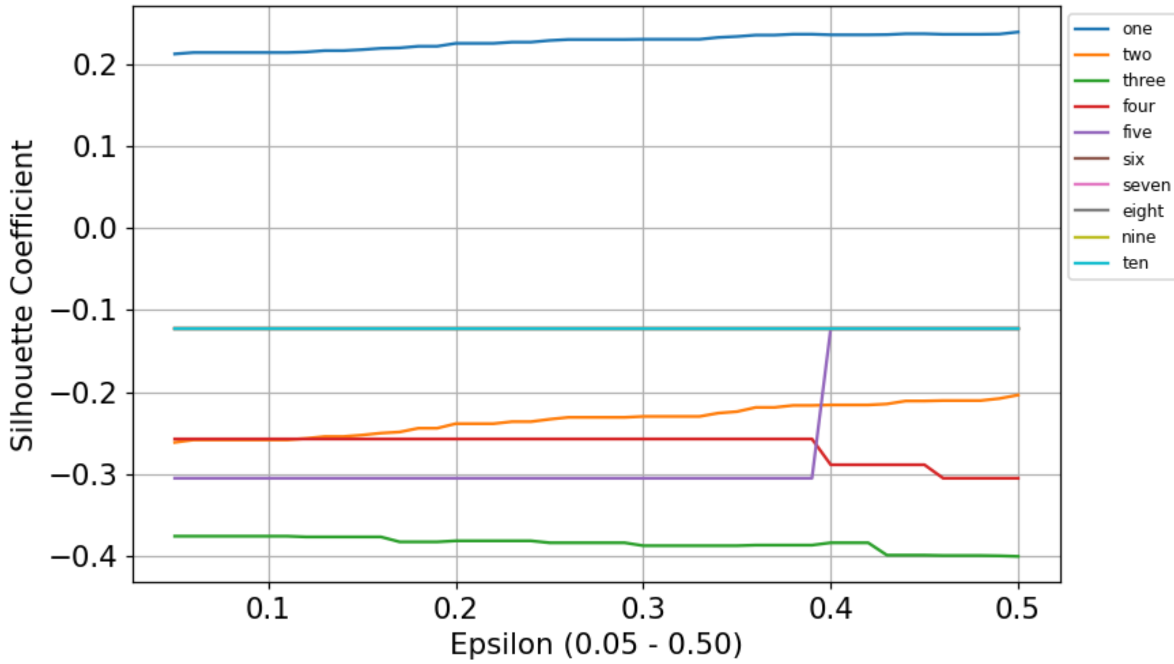
Out[89]:

	one	two	three	four	five	six	seven	eight	nine
0.05	0.212721	-0.261559	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.06	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.07	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.08	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.09	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.10	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.11	0.214439	-0.258502	-0.376098	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.12	0.215072	-0.256780	-0.376959	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.13	0.216729	-0.254424	-0.376959	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.14	0.216729	-0.254424	-0.376959	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.15	0.217948	-0.252620	-0.376959	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.16	0.219554	-0.250255	-0.376959	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.17	0.220070	-0.248876	-0.383004	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.18	0.222072	-0.244276	-0.383004	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.19	0.222072	-0.244276	-0.383004	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.20	0.225620	-0.238697	-0.381732	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.21	0.225620	-0.238697	-0.381732	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.22	0.225620	-0.238697	-0.381732	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.23	0.227140	-0.236269	-0.381732	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.24	0.227140	-0.236269	-0.381732	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.25	0.229135	-0.233334	-0.384119	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.26	0.230297	-0.231003	-0.384119	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.27	0.230297	-0.231003	-0.384119	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.28	0.230297	-0.231003	-0.384119	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.29	0.230297	-0.231003	-0.384119	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.30	0.230576	-0.229889	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.31	0.230576	-0.229889	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.32	0.230576	-0.229889	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.33	0.230576	-0.229889	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.34	0.232871	-0.225815	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.35	0.233977	-0.224008	-0.387667	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.36	0.235686	-0.218853	-0.387005	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.37	0.235686	-0.218853	-0.387005	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663

	one	two	three	four	five	six	seven	eight	nine
0.38	0.236802	-0.216412	-0.387005	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.39	0.236802	-0.216412	-0.387005	-0.257371	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663
0.40	0.236016	-0.215828	-0.384122	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.41	0.236016	-0.215828	-0.384122	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.42	0.236016	-0.215828	-0.384122	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.43	0.236279	-0.214679	-0.399255	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.44	0.237350	-0.211018	-0.399255	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.45	0.237350	-0.211018	-0.399255	-0.288866	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.46	0.236705	-0.210493	-0.399689	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.47	0.236705	-0.210493	-0.399689	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.48	0.236705	-0.210493	-0.399689	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.49	0.236962	-0.208033	-0.400048	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663
0.50	0.239438	-0.203739	-0.400741	-0.305330	-0.122663	-0.122663	-0.122663	-0.122663	-0.122663

In [90]:

```
plt.figure()
plt.plot(wine_sc_df["one"])
plt.plot(wine_sc_df["two"])
plt.plot(wine_sc_df["three"])
plt.plot(wine_sc_df["four"])
plt.plot(wine_sc_df["five"])
plt.plot(wine_sc_df["six"])
plt.plot(wine_sc_df["seven"])
plt.plot(wine_sc_df["eight"])
plt.plot(wine_sc_df["nine"])
plt.plot(wine_sc_df["ten"])
plt.legend(wine_sc_df, bbox_to_anchor=(1, 1), loc='upper left', fontsize = "8")
plt.xlabel('Epsilon (0.05 - 0.50)')
plt.ylabel('Silhouette Coefficient')
plt.grid(True)
plt.show()
```



My dataset contains the make up of just over 1,100 different wines. In the code above, I did a dbscan to cluster the wines into 10 different clusters. However, the last 5 clusters are all identical. This makes sense as wines can be very similar. For that reason, this dataset would probably best fit into 4 or 5 clusters. Regardless, I have graphed each cluster's epsilon and silhouette coeffiient above. Because the final 5 clusters are identical, they are all represented by the same line.

In [ ]: