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 [1]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
# 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
```

In [2]:
```
network = pd.read_csv('3D_spatial_network.txt.gz', header=None, names=
network = network.drop(['osm'], axis=1).sample(10000)
```

In [3]:
```
network.head()
```

Out[3]:

	lat	lon	alt
262171	9.286047	57.002584	1.387053
328563	10.277626	57.618400	4.226585
344611	10.005608	56.984130	25.729497
364779	9.680198	56.704976	51.819028
408068	10.424713	57.522217	4.785734

In [4]:
```
from sklearn import metrics
```

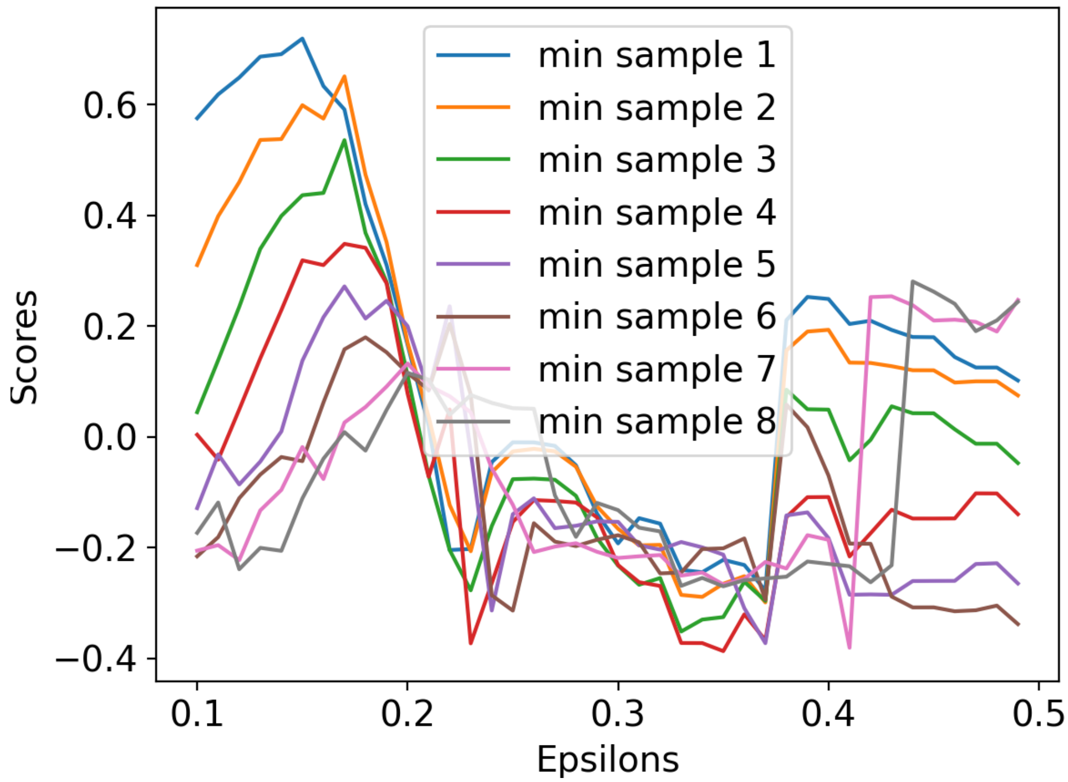
In [5]:
```python
all_scores = []
min_samples = range(1,9) # arguments: start, stop, step
epsilons = np.arange(0.1, 0.5, 0.01) # arguments: start, stop, step

for min_sample in min_samples:
    scores = []
    for epsilon in epsilons:

        dbscan_sample = DBSCAN(eps=epsilon, min_samples=min_sample)
       # print(dbscan_sample)
        network['cluster'] = dbscan_sample.fit_predict(network[['lat',
       # print(network.cluster)
        print(network.head())
        labels = dbscan_sample.labels_
       # print(labels)

        # calculate silouette score here
        score = metrics.silhouette_score(network, labels)

        scores.append(score)

    all_scores.append(scores)
```

```
              lat        lon        alt  cluster
262171   9.286047  57.002584   1.387053        0
328563  10.277626  57.618400   4.226585        1
344611  10.005608  56.984130  25.729497        2
364779   9.680198  56.704976  51.819028        3
408068  10.424713  57.522217   4.785734        4
              lat        lon        alt  cluster
262171   9.286047  57.002584   1.387053        0
328563  10.277626  57.618400   4.226585        1
344611  10.005608  56.984130  25.729497        2
364779   9.680198  56.704976  51.819028        3
408068  10.424713  57.522217   4.785734        4
              lat        lon        alt  cluster
262171   9.286047  57.002584   1.387053        0
328563  10.277626  57.618400   4.226585        1
344611  10.005608  56.984130  25.729497        2
364779   9.680198  56.704976  51.819028        3
408068  10.424713  57.522217   4.785734        4
              lat        lon        alt  cluster
```

In [6]:
```
#scores, all_scores
all_scores[0]
```

Out[6]:
```
[0.5751652322642922,
 0.6182275809549107,
 0.6485869582829002,
 0.6863994564619772,
 0.6908629399990878,
 0.7189757837785473,
 0.6332166141610083,
 0.5909412045980004,
 0.42027215506865795,
 0.3102217056677673,
 0.1660827186283245,
 0.020113260451353816,
 -0.20422003713790673,
 -0.20224898074564676,
 -0.04551609442636838,
 -0.009837190102803994,
 -0.00995035240978921,
 -0.01626718695364404,
 -0.05132628788327207,
 -0.13853911493037827,
 -0.19230404125317901,
 -0.14675049363692932,
 -0.15660682849482246,
 -0.24008508264343145,
 -0.24433887799137147,
 -0.221947690858483,
 -0.23127105646221685,
 -0.27734323742936845,
 0.2105741064197895,
 0.25260754939402064,
 0.248904590342101,
 0.2038099602421482,
 0.2096109867222518,
 0.19305643160956182,
 0.18027788290174424,
 0.17986870054706666,
 0.14411800439722933,
 0.12522780462721378,
 0.12522780462721378,
 0.1017386373502212]
```

In [7]:
```
plt.figure()
plt.plot(epsilons, all_scores[0], label='min sample 1')
plt.plot(epsilons, all_scores[1], label='min sample 2')
plt.plot(epsilons, all_scores[2], label='min sample 3')
plt.plot(epsilons, all_scores[3], label='min sample 4')
plt.plot(epsilons, all_scores[4], label='min sample 5')
plt.plot(epsilons, all_scores[5], label='min sample 6')
plt.plot(epsilons, all_scores[6], label='min sample 7')
plt.plot(epsilons, all_scores[7], label='min sample 8')
plt.xlabel("Epsilons")
plt.ylabel("Scores")
plt.legend()
```



Out[7]: <matplotlib.legend.Legend at 0x10698fdf0>

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

Dataset: https://www.kaggle.com/datasets/muratkokludataset/dry-bean-dataset (https://www.kaggle.com/datasets/muratkokludataset/dry-bean-dataset)

In [8]:
```
import pandas as pd

beans = pd.read_excel('Dry_Bean_Dataset/Dry_Bean_Dataset.xlsx')

beans.head()
```

Out[8]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexAre
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	2871
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	2917
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	2969
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	3072
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	3041

In [9]:
```
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [10]:
```
beans_copy = beans.copy()

enc.fit(beans_copy[["Class"]])
enc.categories_
```

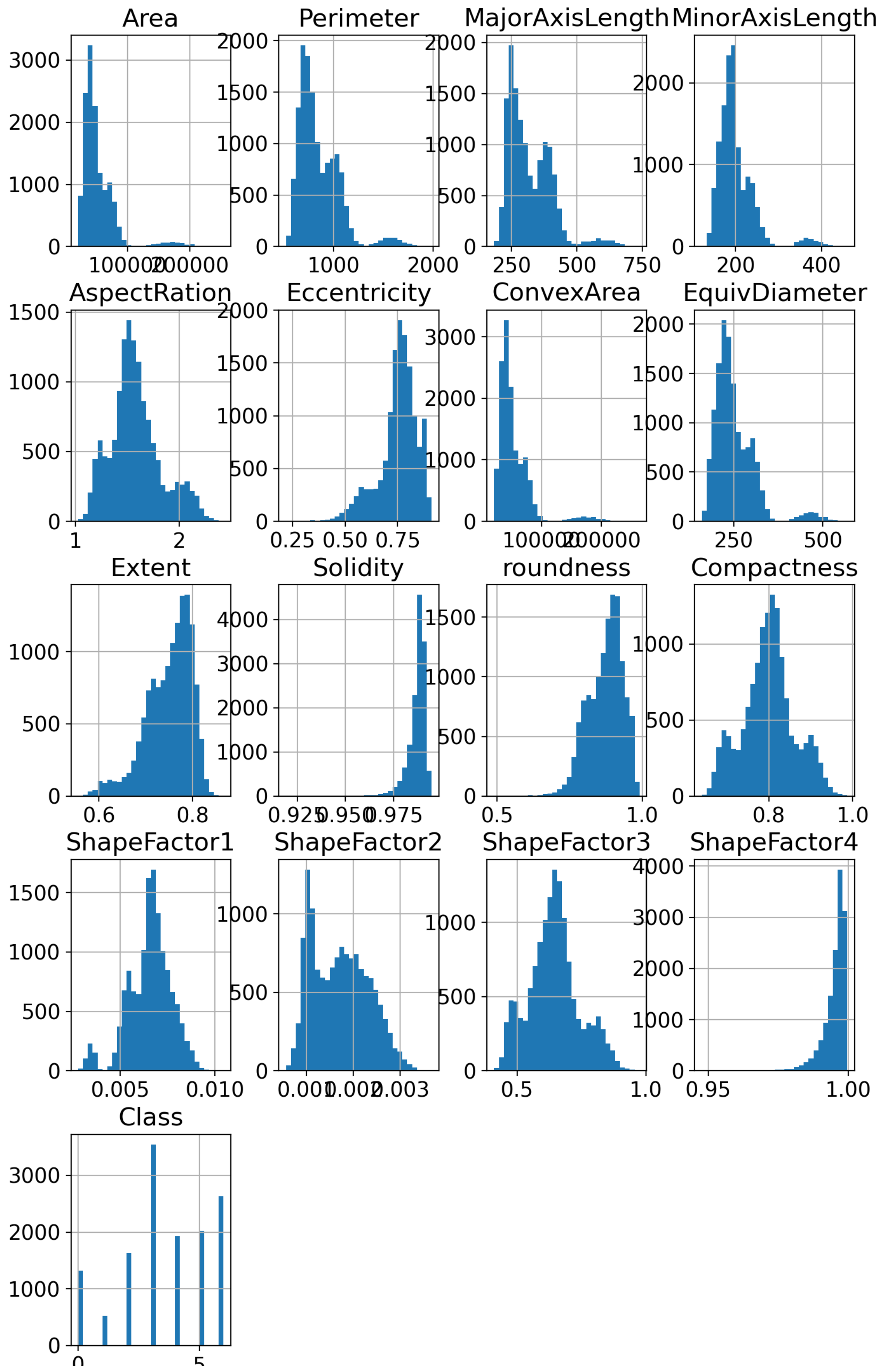
Out[10]:
```
[array(['BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', '
SIRA'],
       dtype=object)]
```

In [11]:
```
beans_copy["Class"] = enc.fit_transform(beans_copy[["Class"]])
beans_copy.head()
```

Out[11]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexAre
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	2871
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	2917
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	2969
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	3072
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	3041

In [12]:
```
#look at the data spread
beans_copy.hist(bins=30, figsize=(9.5,16))
plt.show()
```



In [13]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(beans_copy.drop(['
#beans.shape, train.shape, test.shape
x_train.head()
```

Out[13]:

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	Conve
171	34097	680.595	227.819142	190.810171	1.193957	0.546359	
8753	44642	796.470	305.442642	186.755899	1.635518	0.791301	
11270	29842	637.688	230.432165	165.446548	1.392789	0.696060	
11226	29687	655.459	236.983437	160.933892	1.472551	0.734052	
9247	46908	807.242	302.797882	197.798943	1.530837	0.757153	

In [14]:
```
y_train.head()
```

Out[14]:
```
171      5.0
8753     6.0
11270    3.0
11226    3.0
9247     6.0
Name: Class, dtype: float64
```

In [15]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[15]:
```
((10208, 16), (3403, 16), (10208,), (3403,))
```

In [16]:
```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=5, random_state=1, n_init='auto')
km.fit(x_train)
```

Out[16]:
```
▾ KMeans
KMeans(n_clusters=5, n_init='auto', random_state=1)
```

In [17]:
```
km.labels_
```

Out[17]:
```
array([0, 4, 0, ..., 0, 4, 4], dtype=int32)
```

In [18]:
```
set(km.labels_)
```

Out[18]:
```
{0, 1, 2, 3, 4}
```

In [19]:
```
x_train['cluster'] = km.predict(x_train)
```

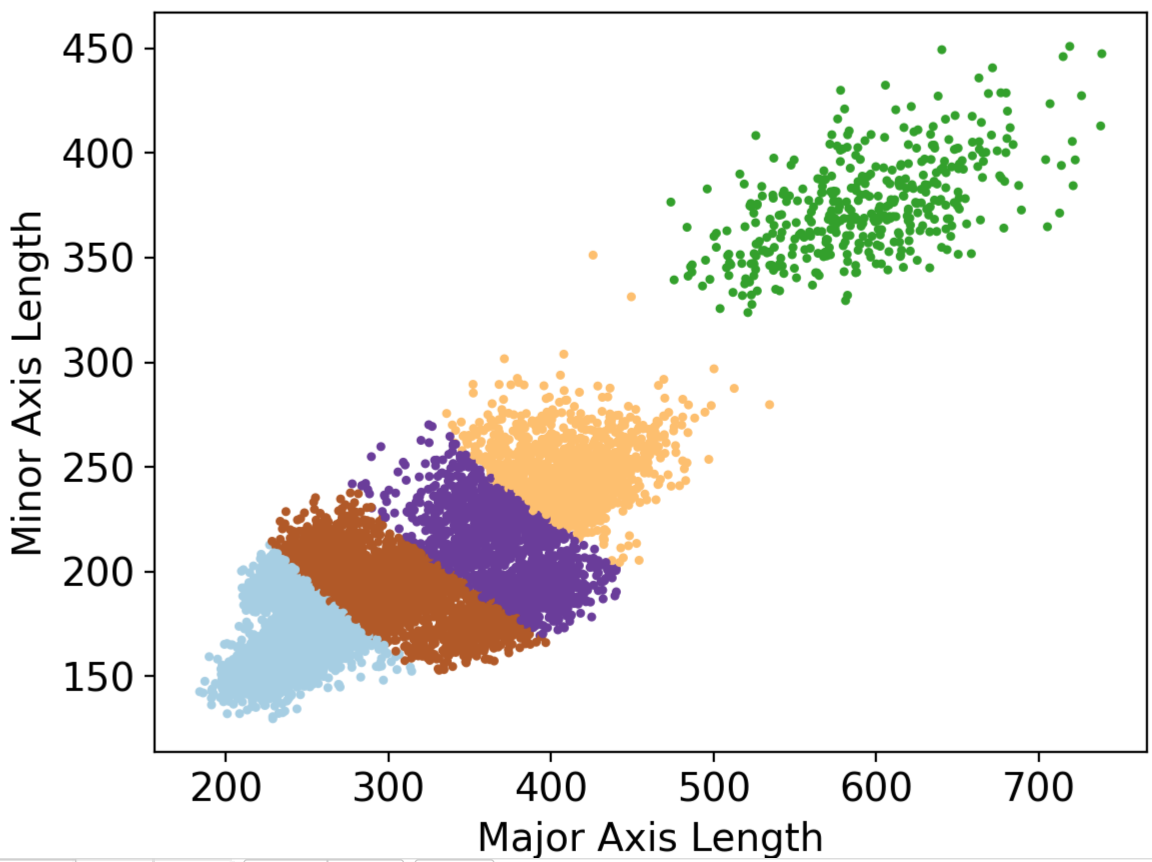
In [20]:
```
x_train.cluster
```

Out[20]:
```
171      0
8753     4
11270    0
11226    0
9247     4
        ..
7197     3
9842     4
759      0
1334     4
6100     4
Name: cluster, Length: 10208, dtype: int32
```

In [21]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
import seaborn
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['font.size'] = 14
```

In [22]:
```
fig = plt.figure()
plt.scatter(x_train.MajorAxisLength, x_train.MinorAxisLength, c=x_trai

plt.xlabel('Major Axis Length')
plt.ylabel('Minor Axis Length')
plt.show()
```

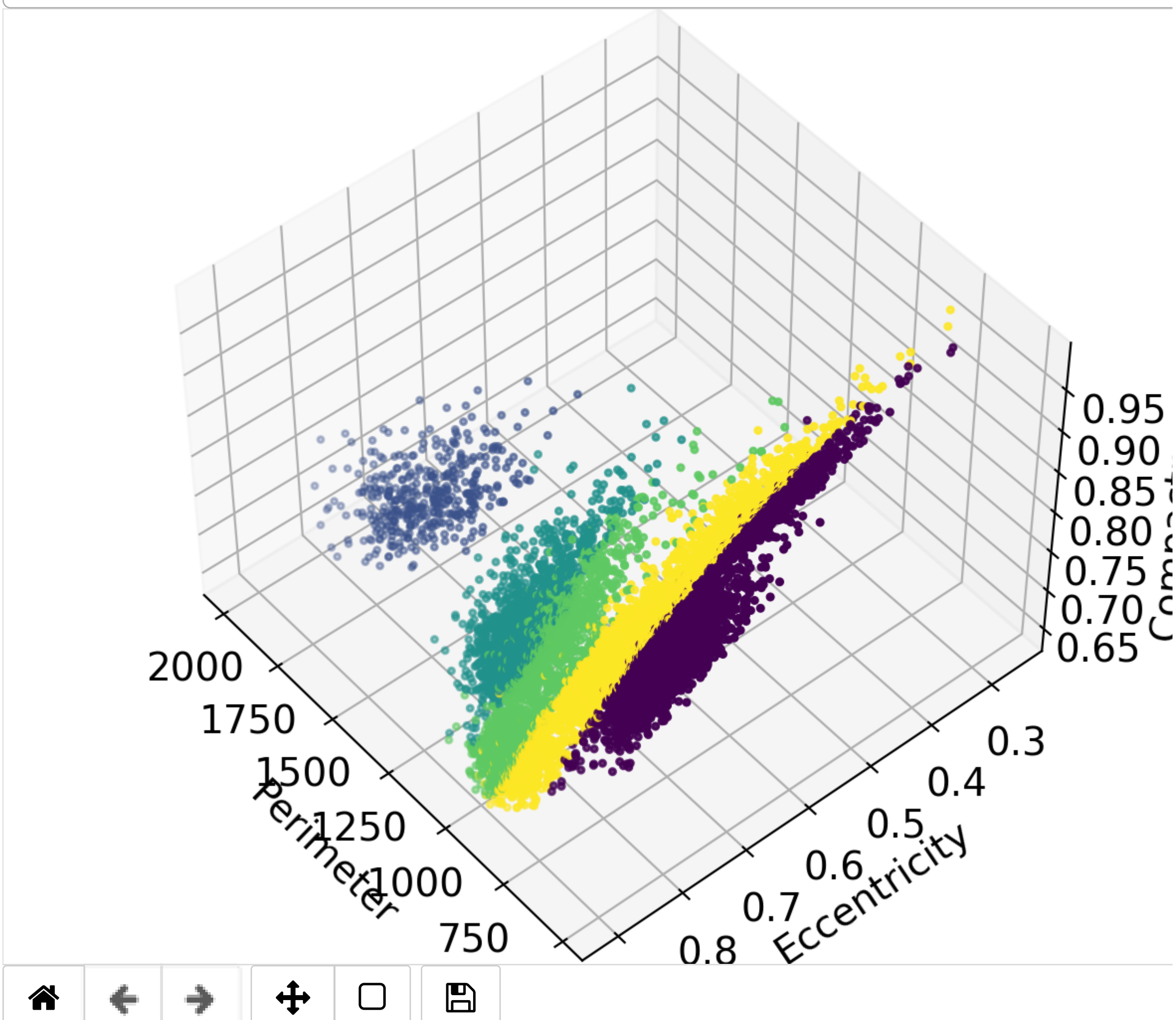
Figure 1



Reset orig

In [23]:
```
fig = plt.figure()
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)

plt.cla()

ax.scatter(x_train['Perimeter'], x_train['Eccentricity'], x_train['Com

ax.set_xlabel('Perimeter')
ax.set_ylabel('Eccentricity')
ax.set_zlabel('Compactness')
plt.show()
```

Figure 2




```
/var/folders/9c/3ylcvqn54zs1r5tlfz4_znf00000gn/T/ipykernel_30730/2949
999719.py:3: MatplotlibDeprecationWarning: 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 warnin
g. The default value of auto_add_to_figure will change to False in mp
l3.5 and True values will no longer work in 3.6.  This is consistent
with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```

In [24]:
```
from sklearn.metrics import silhouette_score

silhouette_score(x_train, km.labels_, metric='euclidean')
```

Out[24]: 0.542391975258459

In [25]:
```
K = range(2, 8)
fits = []
score = []

for k in K:
    print(k)
    model = KMeans(n_clusters = k, random_state = 1, n_init='auto')
    print(model)
    x_train['clusterk'] = model.fit_predict(x_train)
    print(x_train.clusterk)

    fits.append(x_train.clusterk)

    score.append(silhouette_score(x_train, model.labels_, metric='eucl
```

```
2
KMeans(n_clusters=2, n_init='auto', random_state=1)
171      0
8753     0
11270    0
11226    0
9247     0
        ..
7197     0
9842     0
759      0
1334     0
6100     0
Name: clusterk, Length: 10208, dtype: int32
3
KMeans(n_clusters=3, n_init='auto', random_state=1)
171      0
8753     0
11270    0
11226    0
9247     0
        ..
7197     2
9842     0
759      0
1334     0
6100     0
Name: clusterk, Length: 10208, dtype: int32
4
KMeans(n_clusters=4, n_init='auto', random_state=1)
171      0
8753     3
11270    0
11226    0
9247     3
        ..
7197     3
9842     3
759      0
1334     0
6100     3
```

```
Name: clusterk, Length: 10208, dtype: int32
5
KMeans(n_clusters=5, n_init='auto', random_state=1)
171      0
8753     4
11270    0
11226    0
9247     4
        ..
7197     3
9842     4
759      0
1334     4
6100     4
Name: clusterk, Length: 10208, dtype: int32
6
KMeans(n_clusters=6, n_init='auto', random_state=1)
171      0
8753     4
11270    0
11226    0
9247     4
        ..
7197     3
9842     4
759      0
1334     4
6100     4
Name: clusterk, Length: 10208, dtype: int32
7
KMeans(n_clusters=7, n_init='auto', random_state=1)
171      0
8753     4
11270    0
11226    0
9247     4
        ..
7197     2
9842     3
759      4
1334     4
6100     3
Name: clusterk, Length: 10208, dtype: int32
```

In [26]:
```
fits, score
```

Out[26]:
```
([171      0
  8753     0
  11270    0
  11226    0
  9247     0
          ..
  7197     0
  9842     0
  759      0
  1334     0
  6100     0
  Name: clusterk, Length: 10208, dtype: int32,
  171      0
  8753     0
  11270    0
  11226    0
  9247     0
          ..
  7197     2
  9842     0
  759      0
  1334     0
  6100     0
  Name: clusterk, Length: 10208, dtype: int32,
  171      0
  8753     3
  11270    0
  11226    0
  9247     3
          ..
  7197     3
  9842     3
  759      0
  1334     0
  6100     3
  Name: clusterk, Length: 10208, dtype: int32,
  171      0
  8753     4
  11270    0
  11226    0
  9247     4
          ..
  7197     3
  9842     4
  759      0
  1334     4
  6100     4
  Name: clusterk, Length: 10208, dtype: int32,
  171      0
  8753     4
  11270    0
  11226    0
  9247     4
          ..
```

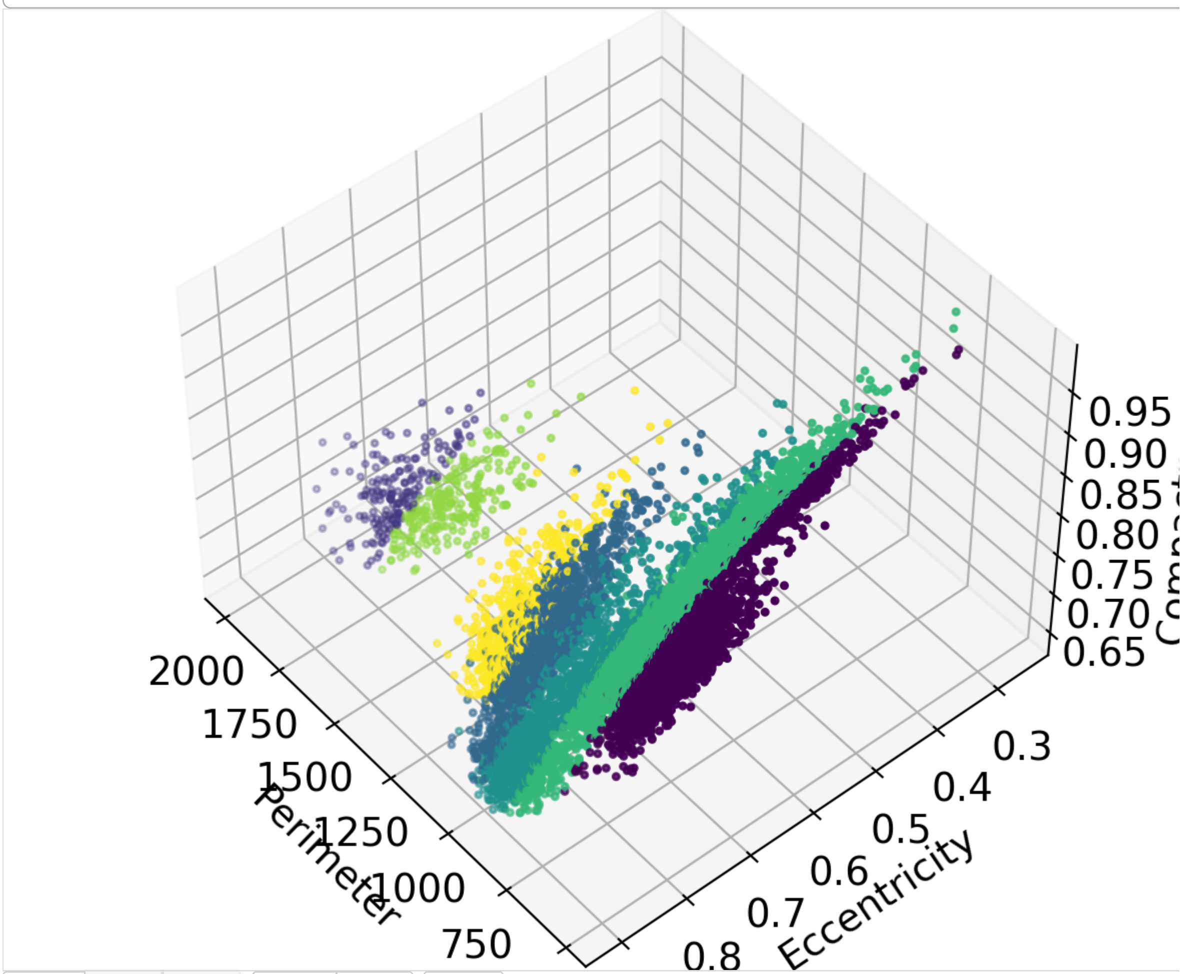
```
 7197     3
 9842     4
 759      0
 1334     4
 6100     4
 Name: clusterk, Length: 10208, dtype: int32,
 171      0
 8753     4
 11270    0
 11226    0
 9247     4
         ..
 7197     2
 9842     3
 759      4
 1334     4
 6100     3
 Name: clusterk, Length: 10208, dtype: int32],
[0.8410175157304038,
 0.6666806386552977,
 0.5703964851265705,
 0.5423919969091973,
 0.5362377726318746,
 0.5345520789470923])
```

In [27]:
```
model.labels_
```

Out[27]:
```
array([0, 4, 0, ..., 4, 4, 3], dtype=int32)
```

In [28]:
```
fig = plt.figure()
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)

plt.cla()

ax.scatter(x_train['Perimeter'], x_train['Eccentricity'], x_train['Com

ax.set_xlabel('Perimeter')
ax.set_ylabel('Eccentricity')
ax.set_zlabel('Compactness')
plt.show()
```

Figure 3

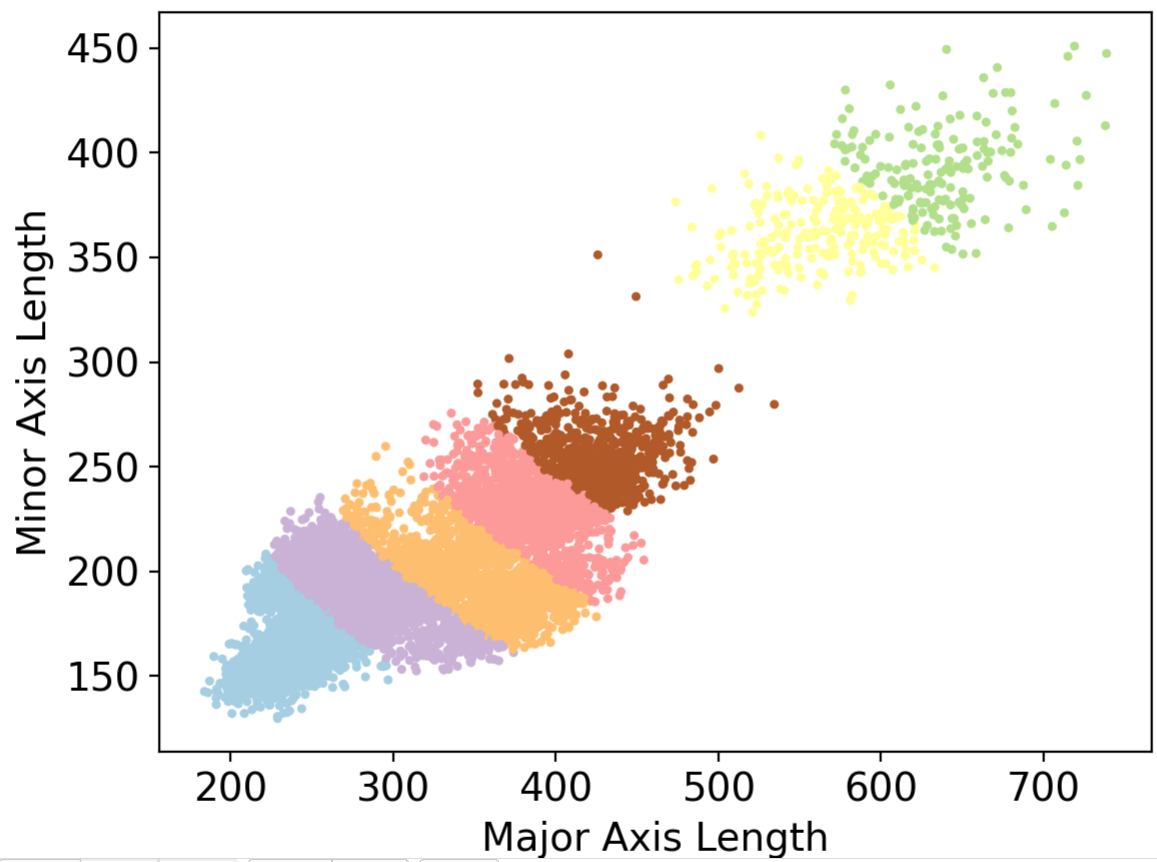


x=1378.1081, y=1.1860, z

```
/var/folders/9c/3ylcvqn54zs1r5tlfz4_znf00000gn/T/ipykernel_30730/8413
01993.py:3: MatplotlibDeprecationWarning: Axes3D(fig) adding itself t
o 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 warnin
g. The default value of auto_add_to_figure will change to False in mp
l3.5 and True values will no longer work in 3.6.  This is consistent
with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```

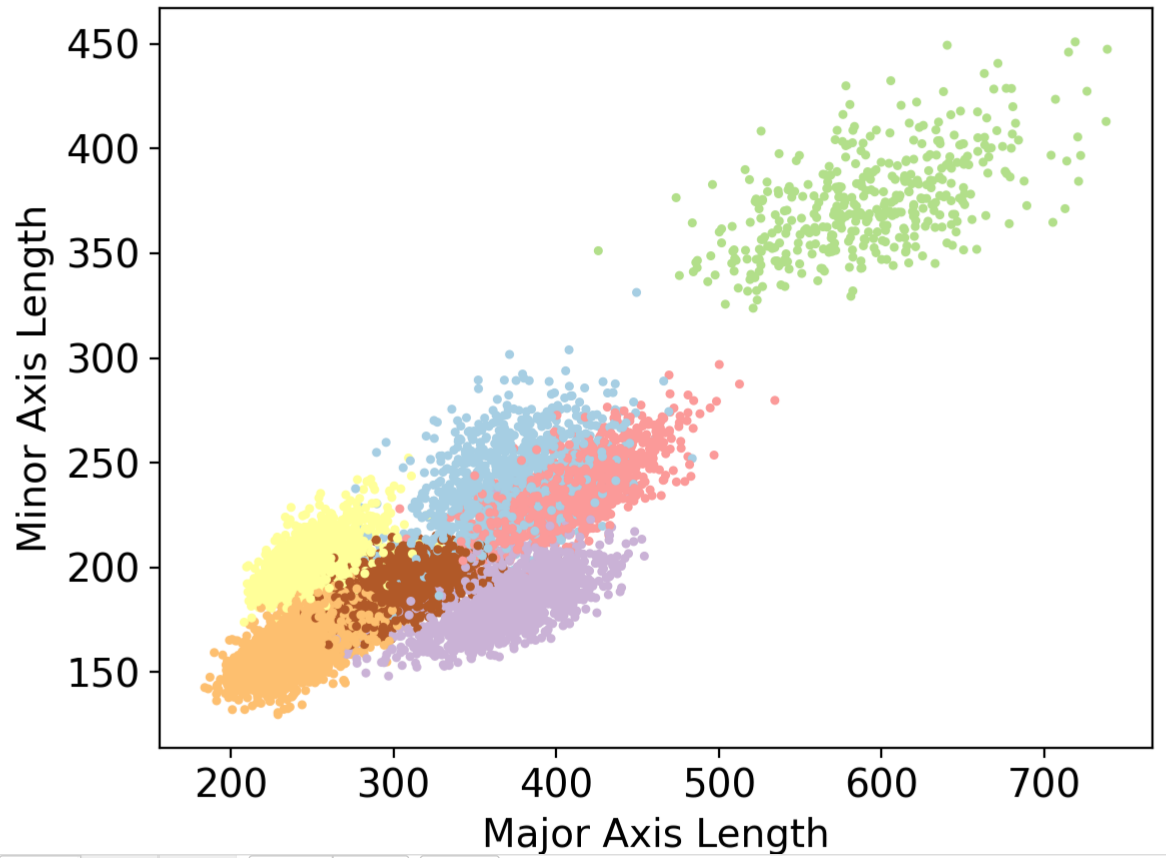
In [29]:
```
fig = plt.figure()
plt.scatter(x_train.MajorAxisLength, x_train.MinorAxisLength, c=model.

plt.xlabel('Major Axis Length')
plt.ylabel('Minor Axis Length')
plt.show()
```

Figure 4



In [30]:
```
#comparison with the actual classification in the data
fig = plt.figure()
plt.scatter(x_train.MajorAxisLength, x_train.MinorAxisLength, c=y_trai

plt.xlabel('Major Axis Length')
plt.ylabel('Minor Axis Length')
plt.show()
```

Figure 5



In [31]:
```
all_scores = []
min_samples = range(2,8) # arguments: start, stop, step
epsilons = np.arange(0.1, 0.5, 0.01) # arguments: start, stop, step

for min_sample in min_samples:
    scores = []
    for epsilon in epsilons:

        dbscan_sample = DBSCAN(eps=epsilon, min_samples=min_sample)
       # print(dbscan_sample)
        x_train['clusterdb'] = dbscan_sample.fit_predict(x_train[['Per
       # print(network.cluster)
        print(x_train.head())
        labels = dbscan_sample.labels_
       # print(labels)

        # calculate silouette score here
        score = metrics.silhouette_score(x_train, labels)

        scores.append(score)

    all_scores.append(scores)
```

```
        Area  Perimeter  MajorAxisLength  MinorAxisLength  AspectRati
on  \
171    34097    680.595       227.819142       190.810171      1.1939
57
8753   44642    796.470       305.442642       186.755899      1.6355
18
11270  29842    637.688       230.432165       165.446548      1.3927
89
11226  29687    655.459       236.983437       160.933892      1.4725
51
9247   46908    807.242       302.797882       197.798943      1.5308
37

       Eccentricity  ConvexArea  EquivDiameter    Extent  Solidity  r
oundness  \
171        0.546359       34502     208.359422  0.762251  0.988262
0.925014
8753       0.791301       45159     238.411325  0.717763  0.988552
0.884331
```

In [32]:
```
x_train.clusterdb
```

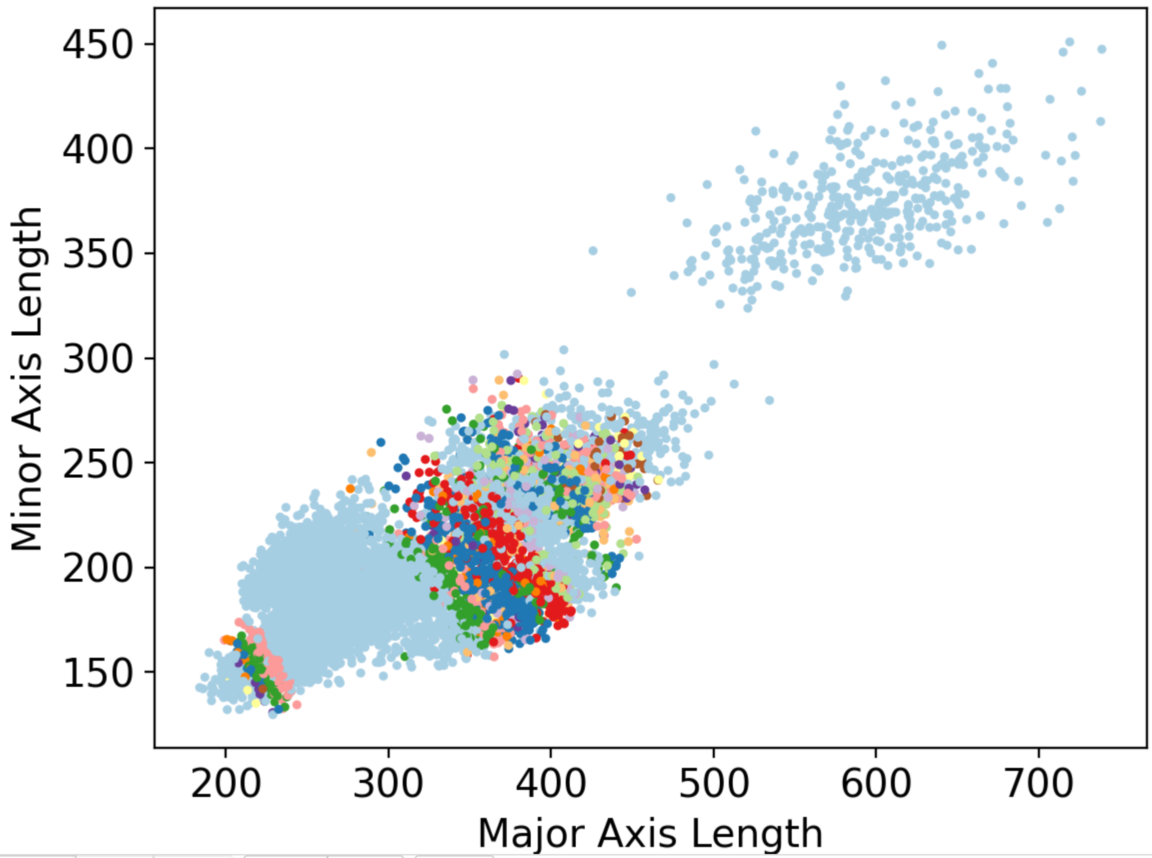
Out[32]:
```
171       0
8753      0
11270     0
11226     0
9247      0
         ..
7197     31
9842      1
759       0
1334      0
6100      2
Name: clusterdb, Length: 10208, dtype: int64
```

In [33]:
```
scores
```

Out[33]:
```
[-0.8393545503519146,
 -0.827660437717611З,
 -0.8136238937630195,
 -0.7829410446219166,
 -0.7590155414441326,
 -0.735661816250З671,
 -0.7063114084460643,
 -0.670584228702312,
 -0.634404895425919,
 -0.6103118919104871,
 -0.5827096544360227,
 -0.5575909328263029,
 -0.5378707173180183,
 -0.5131225497068075,
 -0.47848035895795465,
 -0.4415145465644813,
 -0.39543791239719445,
 -0.3960459763120294,
 -0.34791161802283394,
 -0.33834338951189474,
 -0.28945514996062166,
 -0.22939885312600317,
 -0.20391885455224534,
 -0.17447259650406186,
 -0.1783812109711314,
 -0.23039324105115638,
 -0.22736891172290224,
 -0.22022691789463072,
 -0.22580434089341267,
 -0.20606383976585677,
 -0.20344150966757796,
 -0.19079174786319142,
 -0.1829432998367496,
 -0.17333649284709712,
 -0.16631590313593514,
 -0.16052902873764496,
 -0.15876268706937258,
 -0.12228551729882176,
 -0.10596644617315923,
 -0.10280755718069963]
```

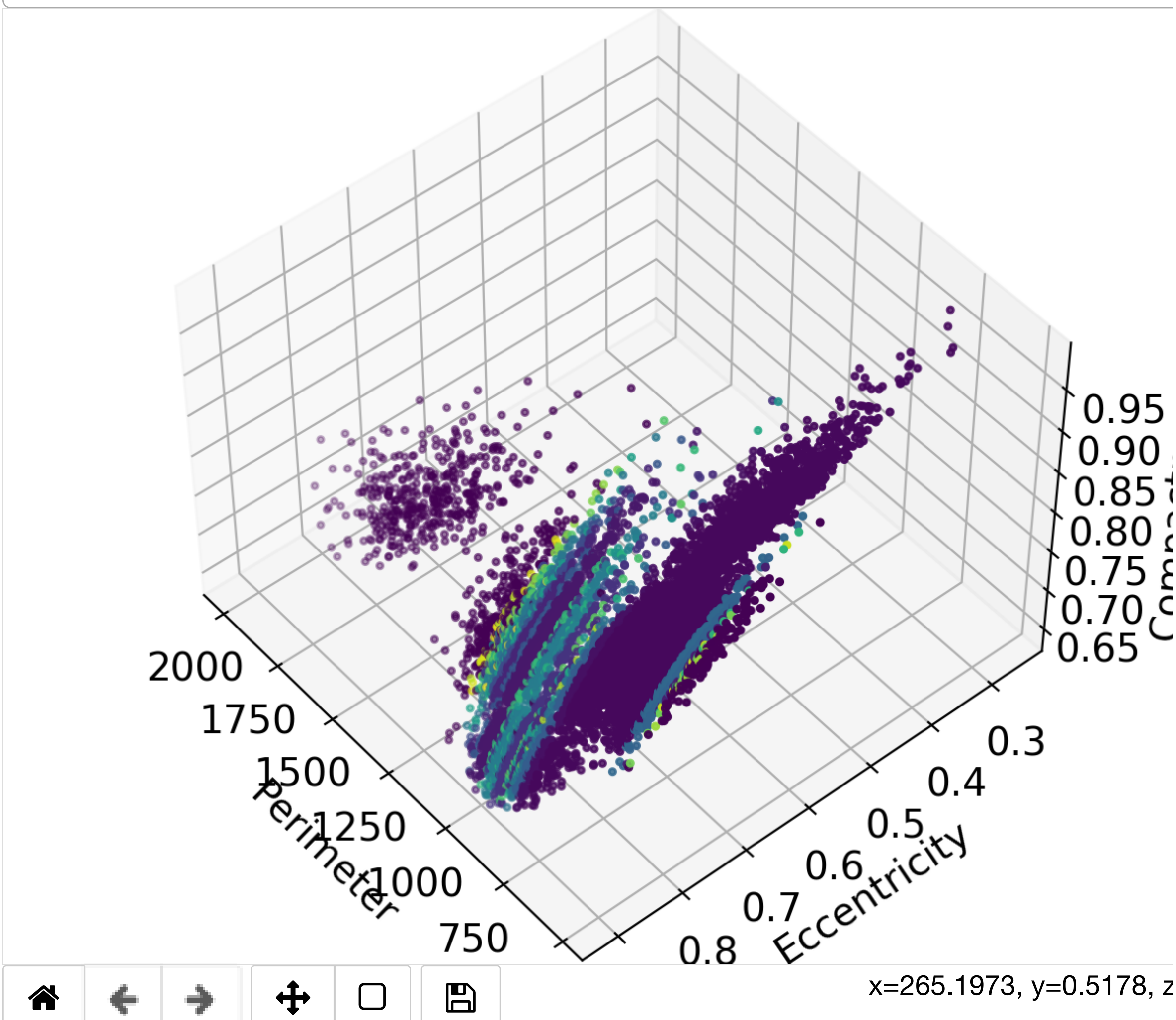
In [34]:
```
fig = plt.figure()
plt.scatter(x_train.MajorAxisLength, x_train.MinorAxisLength, c=x_trai

plt.xlabel('Major Axis Length')
plt.ylabel('Minor Axis Length')
plt.show()
```

Figure 6



In [35]:
```
fig = plt.figure()
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)

plt.cla()

ax.scatter(x_train['Perimeter'], x_train['Eccentricity'], x_train['Com

ax.set_xlabel('Perimeter')
ax.set_ylabel('Eccentricity')
ax.set_zlabel('Compactness')
plt.show()
```

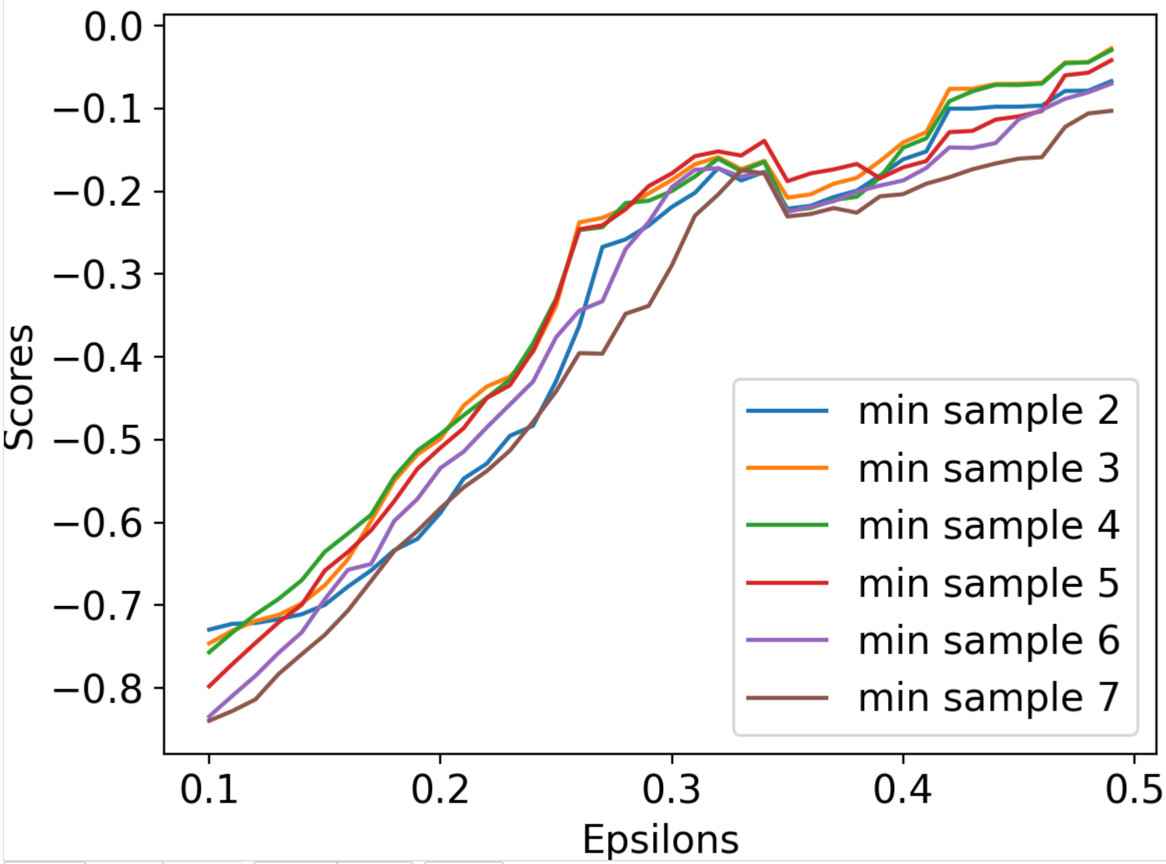
Figure 7




```
/var/folders/9c/3ylcvqn54zs1r5tlfz4_znf00000gn/T/ipykernel_30730/4155
452839.py:3: MatplotlibDeprecationWarning: 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 warnin
g. The default value of auto_add_to_figure will change to False in mp
l3.5 and True values will no longer work in 3.6.  This is consistent
with other Axes classes.
  ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=140)
```

In [37]:
```
plt.figure()
plt.plot(epsilons, all_scores[0], label='min sample 2')
plt.plot(epsilons, all_scores[1], label='min sample 3')
plt.plot(epsilons, all_scores[2], label='min sample 4')
plt.plot(epsilons, all_scores[3], label='min sample 5')
plt.plot(epsilons, all_scores[4], label='min sample 6')
plt.plot(epsilons, all_scores[5], label='min sample 7')
plt.xlabel("Epsilons")
plt.ylabel("Scores")
plt.legend()
```

Figure 8



Out[37]: <matplotlib.legend.Legend at 0x1210b7c70>

In [ ]: