Lab2

April 18, 2020

[16]:

```
# Import packages.
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from IPython.display import display
from sklearn.cluster import AffinityPropagation
import matplotlib.pyplot as plt
from itertools import cycle


# Import and preview data with complete views
filename = 'Lab2data.csv'
data = pd.read_csv(filename)
pd.set_option("max_rows", None)
pd.set_option("max_columns", None)
display(data)
```

```
    Community Area     Community Area Name  Birth Rate  \
0                1            Rogers Park        16.4
1                2            West Ridge         17.3
2                3                Uptown         13.1
3                4         Lincoln Square        17.1
4                5           North Center        22.4
5                6              Lake View        13.5
6                7           Lincoln Park        13.2
7                8        Near North Side        10.7
8                9            Edison Park        11.3
9               10           Norwood Park        10.4
10              11         Jefferson Park        13.8
11              12            Forest Glen        10.0
12              13             North Park        10.9
13              14            Albany Park        18.3
14              15           Portage Park        14.2
15              16            Irving Park        15.8
16              17                Dunning        12.5
17              18             Montclaire        17.1
18              19         Belmont Cragin        20.0
19              20                Hermosa        20.3
```

20	21	Avondale	18.5
21	22	Logan Square	18.2
22	23	Humboldt Park	19.2
23	24	West Town	18.8
24	25	Austin	18.0
25	26	West Garfield Park	20.1
26	27	East Garfield Park	19.4
27	28	Near West Side	18.2
28	29	North Lawndale	20.6
29	30	South Lawndale	19.5
30	31	Lower West Side	16.5
31	32	Loop	9.4
32	33	Near South Side	21.4
33	34	Armour Square	11.5
34	35	Douglas	10.3
35	36	Oakland	17.5
36	37	Fuller Park	11.9
37	38	Grand Boulevard	14.3
38	39	Kenwood	12.2
39	40	Washington Park	19.3
40	41	Hyde Park	9.7
41	42	Woodlawn	15.1
42	43	South Shore	15.7
43	44	Chatham	14.4
44	45	Avalon Park	13.3
45	46	South Chicago	18.1
46	47	Burnside	12.9
47	48	Calumet Heights	10.0
48	49	Roseland	14.3
49	50	Pullman	14.3
50	51	South Deering	15.8
51	52	East Side	17.7
52	53	West Pullman	15.3
53	54	Riverdale	12.5
54	55	Hegewisch	13.7
55	56	Garfield Ridge	13.4
56	57	Archer Heights	18.1
57	58	Brighton Park	20.6
58	59	McKinley Park	16.7
59	60	Bridgeport	11.7
60	61	New City	21.4
61	62	West Elsdon	20.8
62	63	Gage Park	21.8
63	64	Clearing	14.6
64	65	West Lawn	18.8
65	66	Chicago Lawn	20.1
66	67	West Englewood	20.3
67	68	Englewood	20.0

68	69	Greater Grand Crossing	18.2
69	70	Ashburn	14.7
70	71	Auburn Gresham	15.1
71	72	Beverly	11.0
72	73	Washington Heights	12.0
73	74	Mount Greenwood	12.5
74	75	Morgan Park	13.2
75	76	O'Hare	15.8
76	77	Edgewater	12.1

	General Fertility Rate	Low Birth Weight	\
0	62.0	11.0	
1	83.3	8.1	
2	50.5	8.3	
3	61.0	8.1	
4	76.2	9.1	
5	38.7	6.3	
6	38.7	6.6	
7	35.9	8.6	
8	59.5	7.9	
9	59.6	4.9	
10	67.8	6.6	
11	60.6	7.6	
12	54.2	9.7	
13	76.5	8.5	
14	66.1	6.9	
15	67.1	7.7	
16	64.7	6.8	
17	83.5	8.3	
18	88.6	6.9	
19	86.7	6.7	
20	77.7	7.3	
21	63.5	7.2	
22	80.7	12.3	
23	60.4	9.1	
24	80.1	15.4	
25	88.4	17.0	
26	80.8	17.5	
27	55.6	9.0	
28	86.3	15.3	
29	94.9	7.6	
30	68.2	4.5	
31	27.7	5.3	
32	72.9	8.8	
33	57.1	12.4	
34	42.2	11.7	
35	63.9	13.5	
36	60.4	17.1	

37	58.2	12.7
38	51.1	11.4
39	72.1	17.7
40	33.9	5.9
41	60.1	17.4
42	67.9	13.8
43	71.3	15.4
44	69.6	19.7
45	82.2	13.0
46	64.0	7.9
47	57.4	9.3
48	68.8	12.2
49	66.5	11.2
50	76.7	14.9
51	85.8	6.4
52	71.2	14.9
53	46.1	15.3
54	73.2	7.7
55	69.0	8.0
56	80.0	8.7
57	90.6	7.2
58	73.9	7.3
59	51.7	8.0
60	93.4	11.8
61	92.3	4.6
62	93.2	6.8
63	68.3	7.4
64	83.3	7.6
65	85.3	9.4
66	93.3	16.1
67	85.8	14.5
68	82.4	12.9
69	69.0	9.0
70	70.5	11.6
71	60.7	4.9
72	61.0	19.6
73	59.0	8.4
74	67.5	10.6
75	70.0	3.5
76	48.1	7.5

	Prenatal Care Beginning in First Trimester	Preterm Births \
0	73.0	11.2
1	71.1	8.3
2	77.7	10.3
3	80.5	9.7
4	80.4	9.8
5	79.1	8.1

6	75.7	7.8
7	69.7	9.6
8	86.6	12.6
9	89.4	8.3
10	82.9	7.7
11	79.3	10.3
12	79.1	10.2
13	73.3	8.3
14	79.8	8.7
15	79.9	10.2
16	82.7	9.9
17	77.6	8.8
18	74.1	7.6
19	77.5	8.8
20	74.4	7.5
21	78.2	9.4
22	70.9	11.7
23	75.5	10.8
24	72.9	14.3
25	71.4	17.5
26	73.2	16.3
27	77.6	10.5
28	75.8	15.2
29	85.1	9.6
30	80.8	8.8
31	78.2	6.9
32	78.1	10.9
33	79.1	11.8
34	76.0	10.2
35	75.0	11.5
36	71.4	14.3
37	74.2	13.7
38	77.6	11.9
39	74.9	16.9
40	80.3	5.5
41	73.3	16.3
42	71.9	13.4
43	72.5	15.6
44	74.5	14.6
45	75.3	15.1
46	68.4	13.2
47	75.0	16.4
48	69.0	12.8
49	81.3	12.1
50	76.9	15.7
51	73.3	11.0
52	71.3	14.4
53	74.1	16.5

54	66.9	13.1
55	81.7	9.5
56	74.3	10.0
57	82.7	8.3
58	80.5	9.9
59	80.3	11.4
60	75.9	12.2
61	81.2	7.5
62	80.4	6.8
63	85.8	8.0
64	83.5	8.3
65	78.4	10.8
66	63.6	13.8
67	69.7	13.1
68	72.3	15.1
69	82.4	11.3
70	71.8	13.9
71	84.8	9.9
72	75.4	16.2
73	94.5	15.1
74	74.5	12.3
75	82.0	5.0
76	76.1	7.4

	Teen Birth Rate	Assault (Homicide)	Breast cancer in females \
0	40.8	7.7	23.3
1	29.9	5.8	20.2
2	35.1	5.4	21.3
3	38.4	5.0	21.7
4	8.4	1.0	16.6
5	15.8	1.4	20.1
6	2.1	0.7	23.7
7	34.0	3.7	24.0
8	3.9	0.0	13.8
9	3.4	4.7	20.7
10	28.6	4.8	18.4
11	6.3	3.3	25.0
12	10.5	3.0	20.4
13	44.5	4.7	22.9
14	41.7	3.3	23.3
15	37.0	4.1	29.9
16	19.9	3.7	23.7
17	61.5	8.6	29.9
18	68.2	7.0	14.4
19	69.7	12.7	18.4
20	63.4	4.7	16.6
21	66.1	8.6	9.2
22	77.9	29.0	26.3

23	49.2	8.5	14.5
24	81.8	34.4	33.7
25	114.9	40.0	54.7
26	93.2	38.4	21.7
27	36.7	12.7	33.4
28	108.9	46.7	45.8
29	77.5	11.1	13.2
30	49.0	11.7	27.2
31	1.3	0.7	20.2
32	50.9	4.8	31.9
33	16.2	1.8	10.7
34	34.2	13.6	34.3
35	54.5	18.9	20.6
36	69.2	49.6	8.5
37	54.8	32.1	22.6
38	25.7	15.2	30.9
39	82.6	44.6	26.0
40	7.6	5.8	33.6
41	51.1	31.1	32.9
42	65.9	33.4	32.4
43	68.2	45.2	41.9
44	63.9	22.1	33.8
45	76.8	36.9	31.9
46	68.7	70.3	7.6
47	39.3	19.5	33.5
48	79.7	40.9	43.5
49	67.9	32.2	34.6
50	65.3	41.3	25.3
51	53.0	10.7	15.3
52	67.8	43.9	20.3
53	64.5	33.0	25.0
54	47.1	6.9	20.0
55	33.7	9.9	31.0
56	50.3	16.6	25.2
57	58.1	11.1	26.8
58	47.9	5.1	32.7
59	28.4	4.9	16.5
60	94.3	26.6	23.6
61	45.5	5.9	12.1
62	61.4	10.8	23.4
63	38.7	9.4	23.6
64	44.6	10.7	16.9
65	67.4	22.4	25.0
66	116.9	47.2	39.2
67	105.3	45.1	32.9
68	84.3	49.7	29.1
69	38.3	12.4	37.2
70	83.1	37.6	41.9

71	11.9	3.5	42.0
72	65.0	38.0	47.9
73	7.7	2.2	34.6
74	46.7	19.9	32.4
75	15.9	5.6	20.5
76	15.1	5.8	18.5

	Cancer (All Sites)	Colorectal Cancer	Diabetes-related	Firearm-related \
0	176.9	25.3	77.1	5.2
1	155.9	17.3	60.5	3.7
2	183.3	20.5	80.0	4.6
3	153.2	8.6	55.4	6.1
4	152.1	26.1	49.8	1.0
5	126.9	13.0	38.5	1.8
6	152.9	16.7	50.1	2.3
7	142.7	15.1	27.0	3.2
8	189.7	15.1	53.0	7.1
9	180.8	18.9	47.3	8.7
10	208.2	23.2	49.2	4.2
11	138.7	14.3	37.2	6.2
12	143.7	21.9	58.9	3.1
13	158.1	16.8	72.1	5.3
14	168.7	15.9	48.2	4.7
15	169.4	19.2	60.2	5.7
16	191.5	25.9	42.5	5.2
17	151.0	15.4	89.6	6.5
18	152.6	17.7	58.6	5.5
19	135.2	15.6	63.6	11.8
20	133.9	13.4	52.7	4.6
21	148.7	13.7	75.7	10.2
22	211.1	26.6	94.1	22.7
23	139.6	12.4	107.0	6.6
24	261.9	29.8	113.9	28.5
25	291.5	31.4	118.2	36.0
26	236.8	24.8	97.3	37.1
27	202.0	19.2	62.3	9.3
28	261.5	34.8	99.2	37.6
29	127.4	9.2	65.0	8.6
30	141.3	11.9	61.9	11.7
31	120.1	10.8	26.8	4.0
32	169.0	19.2	61.5	6.6
33	162.9	23.1	42.5	1.8
34	269.9	33.2	119.1	9.1
35	159.7	14.5	88.7	12.6
36	258.9	21.1	111.7	22.6
37	218.3	27.7	82.6	25.8
38	196.4	34.2	45.5	17.4
39	258.0	31.9	88.2	39.5

40	144.0	11.7	34.0	5.0
41	241.3	23.9	82.1	25.0
42	246.0	25.1	95.4	30.0
43	213.5	24.1	73.2	37.9
44	239.6	27.7	83.9	18.5
45	227.3	21.1	86.9	33.7
46	191.2	32.8	86.1	70.3
47	216.6	36.0	81.4	24.5
48	258.5	32.0	95.5	37.7
49	262.5	28.1	78.5	32.1
50	224.0	32.5	80.6	35.1
51	182.9	12.9	73.9	10.1
52	263.6	32.6	83.4	46.5
53	258.3	39.4	115.9	32.8
54	210.0	16.7	80.0	13.4
55	231.9	24.6	80.3	9.2
56	166.3	9.0	67.7	15.4
57	139.1	12.8	69.7	10.6
58	148.4	14.2	61.4	6.5
59	168.9	16.1	49.8	4.5
60	235.2	29.2	83.7	24.6
61	180.6	11.5	68.5	7.7
62	171.0	13.6	65.0	11.1
63	189.4	22.5	72.0	12.7
64	145.1	15.7	61.5	8.8
65	179.3	26.1	73.0	19.4
66	247.6	24.4	88.2	39.3
67	252.2	30.2	101.8	44.9
68	274.4	31.5	92.3	44.6
69	229.3	22.8	80.1	11.6
70	243.0	24.5	83.6	32.6
71	197.6	24.8	59.6	3.5
72	260.6	29.7	79.5	35.6
73	201.1	24.8	66.5	7.4
74	218.2	27.1	75.4	15.8
75	138.5	8.7	47.3	11.8
76	162.0	16.2	48.8	3.9

	Infant Mortality Rate	Lung Cancer	Prostate Cancer in Males	\
0	6.4	36.7	21.7	
1	5.1	36.0	14.2	
2	6.5	50.5	25.2	
3	3.8	43.1	27.6	
4	2.7	42.4	15.1	
5	2.2	32.5	17.0	
6	2.4	40.0	27.3	
7	6.5	33.6	15.1	
8	4.6	45.2	28.0	

9	4.4	44.5	26.4
10	8.3	55.7	32.1
11	3.8	27.0	20.3
12	5.4	34.7	14.6
13	4.9	36.9	13.1
14	4.7	44.9	14.8
15	5.3	41.3	17.9
16	4.9	53.9	24.4
17	4.6	33.4	21.9
18	5.6	37.8	27.3
19	9.3	27.7	25.6
20	5.7	32.5	37.7
21	4.3	37.7	17.5
22	9.8	48.0	52.5
23	5.1	27.4	16.6
24	13.3	74.6	69.8
25	19.0	65.3	75.0
26	11.0	56.3	78.1
27	9.1	66.2	33.6
28	14.1	61.7	54.0
29	5.9	15.9	32.7
30	5.4	27.8	14.3
31	5.7	29.2	17.2
32	4.8	46.2	51.4
33	1.5	54.3	17.2
34	13.4	74.5	85.5
35	8.2	54.5	54.2
36	22.6	89.6	70.5
37	12.1	63.8	39.0
38	8.9	49.1	46.2
39	19.3	79.8	67.0
40	10.4	34.9	24.1
41	11.5	69.8	58.0
42	11.4	70.0	54.8
43	10.9	51.5	47.9
44	11.4	57.9	45.1
45	17.7	50.9	56.3
46	13.0	69.5	44.0
47	13.9	48.0	40.4
48	9.6	70.3	57.6
49	13.6	83.8	92.9
50	11.8	59.2	48.7
51	3.7	45.9	26.2
52	11.9	78.6	62.9
53	8.7	86.1	42.5
54	8.4	57.9	28.6
55	4.5	59.2	30.9
56	5.2	49.6	20.5

57	5.9	27.7	15.1
58	7.3	38.3	0.0
59	8.0	54.4	15.8
60	7.9	69.4	40.7
61	8.1	57.8	44.7
62	5.4	34.1	41.1
63	6.7	58.6	18.7
64	8.4	36.7	19.7
65	11.1	42.9	22.1
66	13.3	66.3	75.0
67	13.4	76.8	57.7
68	14.2	77.5	70.3
69	10.2	62.8	44.5
70	15.6	65.1	43.5
71	10.0	47.9	44.7
72	11.2	70.0	56.2
73	3.3	55.0	16.9
74	13.1	50.0	39.8
75	2.0	37.4	2.8
76	6.9	40.1	23.7

	Stroke (Cerebrovascular Disease)	Childhood Blood Lead Level Screening \
0	33.7	364.7
1	34.7	331.4
2	41.7	353.7
3	36.9	273.3
4	41.6	178.1
5	24.4	179.2
6	35.3	173.3
7	22.0	311.2
8	38.9	134.7
9	45.2	163.1
10	41.9	236.8
11	31.0	160.8
12	26.7	317.3
13	39.1	374.1
14	37.9	287.3
15	42.0	333.5
16	32.0	233.9
17	40.3	315.5
18	38.2	421.5
19	27.6	394.7
20	36.0	373.3
21	31.9	364.4
22	53.5	447.7
23	33.3	316.6
24	56.8	492.3
25	77.0	516.1

26	47.5	503.7
27	43.5	396.7
28	70.2	565.7
29	37.3	605.9
30	39.2	589.0
31	39.0	297.3
32	52.4	301.3
33	38.7	490.3
34	62.1	482.2
35	43.7	435.4
36	82.4	489.9
37	46.7	590.4
38	31.5	397.9
39	43.3	502.4
40	26.0	369.0
41	60.0	444.3
42	53.2	470.3
43	50.3	418.3
44	44.5	434.8
45	50.6	493.4
46	99.1	375.0
47	39.2	383.2
48	55.6	369.5
49	65.3	398.5
50	54.9	427.1
51	33.3	398.5
52	63.9	397.2
53	80.6	NaN
54	47.1	298.8
55	42.2	354.9
56	41.8	543.5
57	38.3	580.4
58	51.7	529.6
59	40.1	384.9
60	50.2	555.0
61	42.6	492.6
62	51.2	590.3
63	53.5	302.0
64	43.1	508.3
65	61.7	524.6
66	64.6	450.6
67	71.6	401.0
68	58.2	453.7
69	47.4	368.3
70	63.7	393.8
71	57.2	179.1
72	57.6	371.4
73	26.7	133.6

74	47.9	298.8
75	40.4	182.9
76	31.5	308.6

	Childhood Lead Poisoning	Gonorrhea in Females	Gonorrhea in Males \
0	0.5	322.5	423.3
1	1.0	141.0	205.7
2	0.5	170.8	468.7
3	0.4	98.8	195.5
4	0.9	85.4	188.6
5	0.4	81.8	357.6
6	0.6	50.3	93.1
7	0.1	244.4	235.8
8	0.0	NaN	.
9	0.0	NaN	.
10	0.2	NaN	.
11	0.0	NaN	.
12	0.4	NaN	.
13	1.2	72.9	101.8
14	0.5	87.7	84.9
15	0.6	159.6	74.5
16	0.1	NaN	70.1
17	0.9	NaN	.
18	0.8	95.9	140.9
19	0.7	154.5	114.3
20	0.7	85.3	92.2
21	0.9	209.0	159.3
22	1.3	1234.7	937.5
23	0.5	177.5	182.8
24	2.0	1741.1	1678.9
25	1.4	3193.3	2336.7
26	0.9	2325.1	1730.2
27	0.5	629.2	466.5
28	1.4	2529.9	2236.3
29	0.8	289.5	106.8
30	0.8	92.3	78.7
31	0.3	129.9	200.5
32	0.2	300.6	318.7
33	0.2	222.6	218
34	0.0	1063.3	727.4
35	0.3	1655.4	1629.3
36	2.5	1061.9	1556.4
37	1.0	1454.6	1680
38	0.4	610.2	549.1
39	0.4	2145.8	2058
40	0.0	216.6	168.4
41	1.6	1382.0	1818.6
42	1.1	1718.6	1357.1

43	0.5	1896.3	1855.8
44	0.6	1139.9	2059.9
45	1.6	1774.7	893.5
46	3.6	3108.8	1574.8
47	0.3	1188.0	1106.5
48	1.9	1512.0	1725.3
49	3.7	829.1	1480.4
50	0.3	907.0	1018.2
51	0.9	191.2	98.6
52	1.1	1656.9	1673.4
53	NaN	1699.7	1397.9
54	1.1	NaN	.
55	0.1	180.0	101.8
56	0.5	NaN	.
57	0.9	97.2	52.7
58	0.8	141.4	.
59	0.7	110.8	65
60	0.8	1052.6	579.7
61	0.2	122.0	115.9
62	0.8	171.6	149.2
63	0.3	NaN	.
64	0.2	93.1	87.2
65	1.2	1189.5	1159.9
66	2.6	2762.2	2545.7
67	2.8	2594.9	2323.5
68	1.9	2500.3	2034.2
69	0.2	529.0	602.9
70	1.5	2032.2	1986.7
71	1.3	195.5	469.5
72	1.5	1298.2	1274.2
73	0.0	NaN	.
74	1.3	800.5	741.1
75	0.5	NaN	.
76	0.9	120.1	427.5

	Tuberculosis	Below Poverty Level	Crowded Housing	Dependency	\
0	11.4	22.7	7.9	28.8	
1	8.9	15.1	7.0	38.3	
2	13.6	22.7	4.6	22.2	
3	8.5	9.5	3.1	25.6	
4	1.9	7.1	0.2	25.5	
5	3.2	10.5	1.2	16.5	
6	1.2	11.8	0.6	20.4	
7	5.5	13.4	2.0	23.3	
8	1.8	5.1	0.6	36.6	
9	1.6	5.9	2.3	40.6	
10	7.1	6.4	1.9	34.4	
11	2.2	6.1	1.3	40.6	

12	8.9	12.4	3.8	39.7
13	16.8	17.1	11.2	32.1
14	4.0	12.3	4.4	34.6
15	9.3	10.8	5.6	31.6
16	2.4	8.3	4.8	34.9
17	3.0	12.8	5.8	35.0
18	9.4	18.6	10.0	36.9
19	15.8	19.1	8.4	36.3
20	10.1	14.6	5.8	30.4
21	8.4	17.2	3.2	26.7
22	9.7	32.6	11.2	38.3
23	3.6	15.7	2.0	22.9
24	5.8	27.0	5.7	39.0
25	6.5	40.3	8.9	42.5
26	13.6	39.7	7.5	43.2
27	14.1	21.6	3.8	22.9
28	9.3	38.6	7.2	40.9
29	7.9	28.1	17.6	33.1
30	11.4	27.2	10.4	35.2
31	6.5	11.1	2.0	15.5
32	5.0	11.1	1.4	21.0
33	22.7	35.8	5.9	37.9
34	4.2	26.1	1.6	31.0
35	6.7	38.1	3.5	40.5
36	0.0	55.5	4.5	38.2
37	13.2	28.3	2.7	41.7
38	0.0	23.1	2.3	34.2
39	5.0	39.1	4.9	40.9
40	5.3	18.2	2.5	26.7
41	17.4	28.3	1.8	37.6
42	11.7	31.5	2.9	37.6
43	8.2	25.3	2.2	40.0
44	1.9	16.7	0.6	41.9
45	6.9	28.0	5.9	43.1
46	6.8	22.5	5.5	40.4
47	2.8	12.0	1.8	42.3
48	7.9	19.5	3.1	40.9
49	2.7	20.1	1.4	42.0
50	2.6	24.5	6.0	41.4
51	5.2	18.7	8.3	42.5
52	9.2	24.3	3.3	42.2
53	5.8	61.4	5.1	50.2
54	2.1	12.1	4.4	41.6
55	3.5	9.0	2.6	39.5
56	1.5	13.0	8.5	40.5
57	11.5	23.0	13.2	39.8
58	11.5	16.1	6.9	33.7
59	9.9	17.3	4.8	32.3

60	9.3	30.6	12.2	42.0
61	4.5	9.8	8.7	38.7
62	7.0	20.8	17.4	40.4
63	0.9	5.9	3.4	36.4
64	4.3	15.3	6.8	41.9
65	5.3	22.2	6.5	40.0
66	10.9	32.3	6.9	40.9
67	11.3	42.2	4.8	43.4
68	4.8	25.6	4.2	42.9
69	4.4	9.5	4.2	36.7
70	7.3	24.5	4.1	42.1
71	0.0	5.2	0.7	38.7
72	3.0	15.7	1.1	42.4
73	0.0	3.1	1.1	37.0
74	2.6	13.7	0.8	39.4
75	6.3	9.5	1.9	26.5
76	10.5	16.6	3.9	23.4

	No High School Diploma	Per Capita Income	Unemployment
0	18.1	23714	7.5
1	19.6	21375	7.9
2	13.6	32355	7.7
3	12.5	35503	6.8
4	5.4	51615	4.5
5	2.9	58227	4.7
6	4.3	71403	4.5
7	3.4	87163	5.2
8	8.5	38337	7.4
9	13.5	31659	7.3
10	13.5	27280	9.0
11	6.3	41509	5.5
12	18.2	24941	7.5
13	34.9	20355	9.0
14	18.7	23617	10.6
15	22.0	26713	10.3
16	18.0	26347	8.6
17	28.4	21257	10.8
18	37.0	15246	11.5
19	41.9	15411	12.9
20	25.7	20489	9.3
21	18.5	29026	7.5
22	36.8	13391	12.3
23	13.4	39596	6.0
24	25.0	15920	21.0
25	26.2	10951	25.2
26	26.2	13596	16.4
27	11.2	41488	10.7
28	30.4	12548	18.5

29	58.7	10697	11.5
30	44.3	15467	13.0
31	3.4	67699	4.2
32	7.1	60593	5.7
33	37.5	16942	11.6
34	16.9	23098	16.7
35	17.6	19312	26.6
36	33.7	9016	40.0
37	19.4	22056	20.6
38	10.8	37519	11.0
39	28.3	13087	23.2
40	5.3	39243	6.9
41	17.9	18928	17.3
42	14.9	18366	17.7
43	13.7	20320	19.0
44	13.3	23495	16.6
45	28.2	15393	17.7
46	18.6	13756	23.4
47	11.2	28977	17.2
48	17.4	17974	17.8
49	15.6	19007	21.0
50	21.9	15506	11.8
51	35.5	15347	14.5
52	22.6	16228	17.0
53	24.6	8535	26.4
54	17.9	22561	9.6
55	19.4	24684	8.1
56	36.4	16145	14.2
57	48.2	13138	11.2
58	31.8	17577	11.9
59	25.6	24969	11.2
60	42.4	12524	17.4
61	39.6	16938	13.5
62	54.1	12014	14.0
63	18.5	23920	9.6
64	33.4	15898	7.8
65	31.6	14405	11.9
66	30.3	10559	34.7
67	29.4	11993	21.3
68	17.9	17213	18.9
69	18.3	22078	8.8
70	19.5	16022	24.2
71	5.1	40107	7.8
72	15.6	19709	18.3
73	4.5	34221	6.9
74	10.9	26185	14.9
75	11.0	29402	4.7
76	9.0	33364	9.0

[95]:
```python
#Filter out and compare teen birth rate and per capita income
babyMoney = data[['Per Capita Income', 'Teen Birth Rate']]
#Display data for two columns
display(babyMoney)
#Calculate descriptive statistics and display
babyMoney.describe()
```

	Per Capita Income	Teen Birth Rate
0	23714	40.8
1	21375	29.9
2	32355	35.1
3	35503	38.4
4	51615	8.4
5	58227	15.8
6	71403	2.1
7	87163	34.0
8	38337	3.9
9	31659	3.4
10	27280	28.6
11	41509	6.3
12	24941	10.5
13	20355	44.5
14	23617	41.7
15	26713	37.0
16	26347	19.9
17	21257	61.5
18	15246	68.2
19	15411	69.7
20	20489	63.4
21	29026	66.1
22	13391	77.9
23	39596	49.2
24	15920	81.8
25	10951	114.9
26	13596	93.2
27	41488	36.7
28	12548	108.9
29	10697	77.5
30	15467	49.0
31	67699	1.3
32	60593	50.9
33	16942	16.2
34	23098	34.2
35	19312	54.5
36	9016	69.2
37	22056	54.8

38	37519	25.7
39	13087	82.6
40	39243	7.6
41	18928	51.1
42	18366	65.9
43	20320	68.2
44	23495	63.9
45	15393	76.8
46	13756	68.7
47	28977	39.3
48	17974	79.7
49	19007	67.9
50	15506	65.3
51	15347	53.0
52	16228	67.8
53	8535	64.5
54	22561	47.1
55	24684	33.7
56	16145	50.3
57	13138	58.1
58	17577	47.9
59	24969	28.4
60	12524	94.3
61	16938	45.5
62	12014	61.4
63	23920	38.7
64	15898	44.6
65	14405	67.4
66	10559	116.9
67	11993	105.3
68	17213	84.3
69	22078	38.3
70	16022	83.1
71	40107	11.9
72	19709	65.0
73	34221	7.7
74	26185	46.7
75	29402	15.9
76	33364	15.1

[95]:

	Per Capita Income	Teen Birth Rate
count	77.000000	77.000000
mean	25106.740260	50.064935
std	14952.672297	28.097817
min	8535.000000	1.300000
25%	15467.000000	33.700000
50%	20489.000000	49.200000

```
75%          29026.000000        67.900000
max          87163.000000       116.900000
```

[96]:

```
#Normalize teen birth rate and per capita income
tbr = np.asarray(data['Teen Birth Rate'])
pci = np.asarray(data['Per Capita Income'])
tbrNorm = (tbr - tbr.min())/(np.ptp(tbr))
pciNorm = (pci - pci.min())/(np.ptp(pci))


#Create numpy array with normalized data
bmData = np.asarray(babyMoney)
for i in range(77):
    array = np.array([pciNorm[i], tbrNorm[i]])
    bmData[i] = array


print(tbrNorm)
print(pciNorm)
print("New Data Vector:")
print(bmData)
```

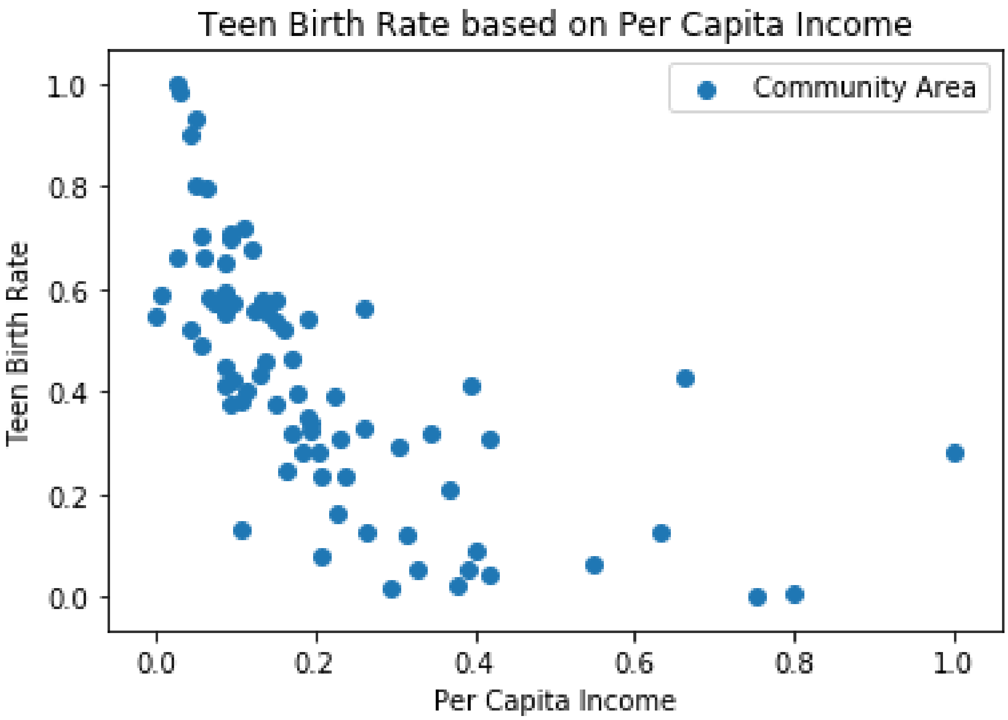
```
[0.3416955  0.24740484 0.29238754 0.32093426 0.06141869 0.12543253
 0.00692042 0.28287197 0.02249135 0.01816609 0.23615917 0.0432526
 0.07958478 0.37370242 0.34948097 0.30882353 0.16089965 0.52076125
 0.57871972 0.5916955  0.53719723 0.56055363 0.66262976 0.41435986
 0.69636678 0.98269896 0.7949827  0.30622837 0.93079585 0.65916955
 0.41262976 0.         0.42906574 0.12889273 0.28460208 0.46020761
 0.58737024 0.46280277 0.21107266 0.7032872  0.05449827 0.43079585
 0.55882353 0.57871972 0.54152249 0.65311419 0.58304498 0.32871972
 0.67820069 0.57612457 0.55363322 0.44723183 0.57525952 0.5467128
 0.39619377 0.28027682 0.42387543 0.49134948 0.40311419 0.23442907
 0.80449827 0.38235294 0.51989619 0.32352941 0.37456747 0.57179931
 1.         0.89965398 0.71799308 0.3200692  0.70761246 0.0916955
 0.55103806 0.05536332 0.39273356 0.12629758 0.11937716]
[0.19304828 0.16330061 0.30294552 0.34298214 0.54789642 0.6319886
 0.7995625  1.         0.37902528 0.29409371 0.23840108 0.41936715
 0.20865341 0.15032813 0.19181462 0.23118991 0.22653508 0.16179987
 0.08535127 0.08744976 0.15203235 0.26060691 0.06175917 0.39503739
 0.09392328 0.03072697 0.06436638 0.41910007 0.0510378  0.02749657
 0.08816198 0.7524546  0.66207967 0.1069212  0.18521392 0.13706313
 0.00611741 0.17196164 0.36862186 0.05789286 0.3905479  0.13217938
 0.1250318  0.14988299 0.19026301 0.08722084 0.06640128 0.25998372
 0.12004629 0.13318411 0.08865798 0.0866358  0.09784046 0.
 0.17838429 0.20538485 0.09678486 0.05854149 0.1149972  0.20900951
 0.05073256 0.10687033 0.04424632 0.19566821 0.09364349 0.07465534
 0.02574147 0.04397924 0.11036781 0.17224144 0.09522053 0.40153635
```

```
 0.14211222 0.32667752 0.22447474 0.26538892 0.31577809]
New Data Vector:
[[0.19304828 0.3416955 ]
 [0.16330061 0.24740484]
 [0.30294552 0.29238754]
 [0.34298214 0.32093426]
 [0.54789642 0.06141869]
 [0.6319886  0.12543253]
 [0.7995625  0.00692042]
 [1.         0.28287197]
 [0.37902528 0.02249135]
 [0.29409371 0.01816609]
 [0.23840108 0.23615917]
 [0.41936715 0.0432526 ]
 [0.20865341 0.07958478]
 [0.15032813 0.37370242]
 [0.19181462 0.34948097]
 [0.23118991 0.30882353]
 [0.22653508 0.16089965]
 [0.16179987 0.52076125]
 [0.08535127 0.57871972]
 [0.08744976 0.5916955 ]
 [0.15203235 0.53719723]
 [0.26060691 0.56055363]
 [0.06175917 0.66262976]
 [0.39503739 0.41435986]
 [0.09392328 0.69636678]
 [0.03072697 0.98269896]
 [0.06436638 0.7949827 ]
 [0.41910007 0.30622837]
 [0.0510378  0.93079585]
 [0.02749657 0.65916955]
 [0.08816198 0.41262976]
 [0.7524546  0.        ]
 [0.66207967 0.42906574]
 [0.1069212  0.12889273]
 [0.18521392 0.28460208]
 [0.13706313 0.46020761]
 [0.00611741 0.58737024]
 [0.17196164 0.46280277]
 [0.36862186 0.21107266]
 [0.05789286 0.7032872 ]
 [0.3905479  0.05449827]
 [0.13217938 0.43079585]
 [0.1250318  0.55882353]
 [0.14988299 0.57871972]
 [0.19026301 0.54152249]
 [0.08722084 0.65311419]
```

```
 [0.06640128 0.58304498]
 [0.25998372 0.32871972]
 [0.12004629 0.67820069]
 [0.13318411 0.57612457]
 [0.08865798 0.55363322]
 [0.0866358  0.44723183]
 [0.09784046 0.57525952]
 [0.         0.5467128 ]
 [0.17838429 0.39619377]
 [0.20538485 0.28027682]
 [0.09678486 0.42387543]
 [0.05854149 0.49134948]
 [0.1149972  0.40311419]
 [0.20900951 0.23442907]
 [0.05073256 0.80449827]
 [0.10687033 0.38235294]
 [0.04424632 0.51989619]
 [0.19566821 0.32352941]
 [0.09364349 0.37456747]
 [0.07465534 0.57179931]
 [0.02574147 1.        ]
 [0.04397924 0.89965398]
 [0.11036781 0.71799308]
 [0.17224144 0.3200692 ]
 [0.09522053 0.70761246]
 [0.40153635 0.0916955 ]
 [0.14211222 0.55103806]
 [0.32667752 0.05536332]
 [0.22447474 0.39273356]
 [0.26538892 0.12629758]
 [0.31577809 0.11937716]]
```

[97]:

```
#Plot the data
plt.ion()
plt.scatter(bmData[:, 0], bmData[:, 1], label='Community Area')
plt.xlabel('Per Capita Income')
plt.ylabel('Teen Birth Rate')
plt.title('Teen Birth Rate based on Per Capita Income')
plt.legend()
plt.show()
```

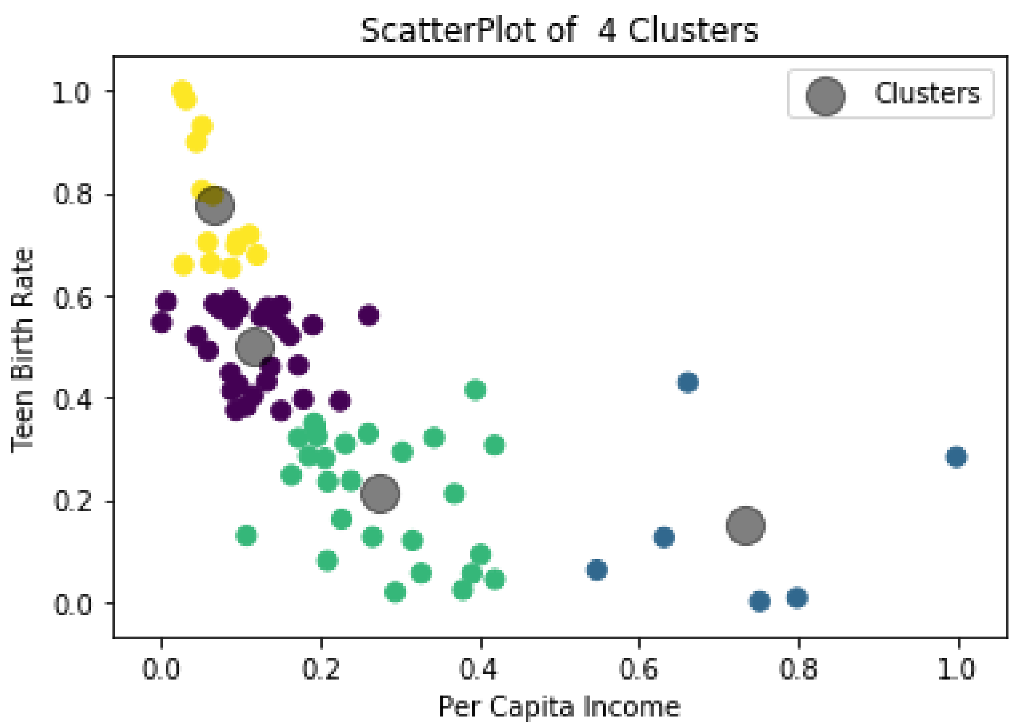


[137]:
```
# Use kmeans to cluster vector into 4 groups

kmeans = KMeans(n_clusters = 4, random_state = 0).fit(bmData)
labels = kmeans.labels_
groupReps = kmeans.cluster_centers_
jClust = kmeans.inertia_

#Print number of iterations
print("Number of iterations: ")
print(kmeans.n_iter_)
print()
print("Labels: ")
print(labels)
print()
print("Cluster Means: ")
print(groupReps)
print()
print("Jclust: ")
print(jClust)
print()

#Plot clusters
```

```
plt.scatter(bmData[:, 0], bmData[:, 1], c = kmeans.predict(bmData), s = 50,␣
 ↪cmap = 'viridis')
#plt.scatter()
plt.title("ScatterPlot of  4 Clusters")
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5,␣
 ↪label='Clusters');
plt.xlabel('Per Capita Income')
plt.ylabel('Teen Birth Rate')
plt.legend()
plt.show()

# Find neighborhoods in each cluster
nH = np.asarray(data['Community Area Name'])
print("Group 1:")
for i in range(len(labels)):
    if (labels[i] == 3):
        print(nH[i])
print()
print("Group 2:")
for i in range(len(labels)):
    if (labels[i] == 2):
        print(nH[i])
print()
print("Group 3:")
for i in range(len(labels)):
    if (labels[i] == 0):
        print(nH[i])
print()
print("Group 4:")
for i in range(len(labels)):
    if (labels[i] == 1):
        print(nH[i])
```

```
Number of iterations:
4

Labels:
[2 2 2 2 1 1 1 1 2 2 2 2 2 0 2 2 2 0 0 0 0 0 3 2 3 3 3 2 3 3 0 1 1 2 2 0 0
 0 2 3 2 0 0 0 0 3 0 2 3 0 0 0 0 0 0 2 0 0 0 2 3 0 0 2 0 0 3 3 3 2 3 2 0 2
 0 2 2]

Cluster Means:
[[0.11685532 0.49948097]
 [0.7323303  0.15095156]
 [0.27438769 0.21113674]
 [0.06575084 0.77792882]]
```

```
Jclust:
1.3430157168960097
```




```
Group 1:
Humboldt Park
Austin
West Garfield Park
East Garfield Park
North Lawndale
South Lawndale
Washington Park
South Chicago
Roseland
New City
West Englewood
Englewood
Greater Grand Crossing
Auburn Gresham

Group 2:
Rogers Park
West Ridge
```

Uptown
Lincoln Square
Edison Park
Norwood Park
Jefferson Park
Forest Glen
North Park
Portage Park
Irving Park
Dunning
West Town
Near West Side
Armour Square
Douglas
Kenwood
Hyde Park
Calumet Heights
Garfield Ridge
Bridgeport
Clearing
Ashburn
Beverly
Mount Greenwood
O'Hare
Edgewater

Group 3:
Albany Park
Montclaire
Belmont Cragin
Hermosa
Avondale
Logan Square
Lower West Side
Oakland
Fuller Park
Grand Boulevard
Woodlawn
South Shore
Chatham
Avalon Park
Burnside
Pullman
South Deering
East Side
West Pullman
Riverdale
Hegewisch

```
Archer Heights
Brighton Park
McKinley Park
West Elsdon
Gage Park
West Lawn
Chicago Lawn
Washington Heights
Morgan Park

Group 4:
North Center
Lake View
Lincoln Park
Near North Side
Loop
Near South Side
```

The dataset was cleaned to include 'Teen birth rates' and 'per capita income' levels. They were normalized and convereted to numpy arrays. Trying a few different cluster quantities, I settled on 4 because the other options did not show much better results, and the Jclust number wasn't getting any better. The kmeans algorithm ran 4 times to achieve the clusters.

The normalized data shows that communities with higher per capita income have a lower teen birth rate and vice versa, the communities with lowest per capita income show the highest teen birth rates. Each community is grouped into one of four clusters, listed beneath the graph. Then comparing the cluster means to the graph, we can list the specific groups from left to right, which are listed underneath the graph. From this data, we can see directly which communities not only have the highest per capita income, but also the lowest teen birth rates. These areas, such as Lakeview, Lincoln Park, and the Loop are probably considered the nicer areas in the city. Whereas the poorest communities, such as Englewood, South Chicago, and Garfield Park have very low income levels and high teen birth rates.

Perhaps one conclusion from this dataset would be that areas with high per capita income have less teen births.

[128]:
```
#Filter out and compare percentage of people without high school diplomas
#and number of assaults(homicides)
dumbKill = data[['No High School Diploma', 'Assault (Homicide)']]
#Display data for two columns
display(dumbKill)
#Calculate descriptive statistics and display
dumbKill.describe()
```

```
    No High School Diploma  Assault (Homicide)
0                     18.1                 7.7
1                     19.6                 5.8
2                     13.6                 5.4
3                     12.5                 5.0
```

4	5.4	1.0
5	2.9	1.4
6	4.3	0.7
7	3.4	3.7
8	8.5	0.0
9	13.5	4.7
10	13.5	4.8
11	6.3	3.3
12	18.2	3.0
13	34.9	4.7
14	18.7	3.3
15	22.0	4.1
16	18.0	3.7
17	28.4	8.6
18	37.0	7.0
19	41.9	12.7
20	25.7	4.7
21	18.5	8.6
22	36.8	29.0
23	13.4	8.5
24	25.0	34.4
25	26.2	40.0
26	26.2	38.4
27	11.2	12.7
28	30.4	46.7
29	58.7	11.1
30	44.3	11.7
31	3.4	0.7
32	7.1	4.8
33	37.5	1.8
34	16.9	13.6
35	17.6	18.9
36	33.7	49.6
37	19.4	32.1
38	10.8	15.2
39	28.3	44.6
40	5.3	5.8
41	17.9	31.1
42	14.9	33.4
43	13.7	45.2
44	13.3	22.1
45	28.2	36.9
46	18.6	70.3
47	11.2	19.5
48	17.4	40.9
49	15.6	32.2
50	21.9	41.3
51	35.5	10.7

52	22.6	43.9
53	24.6	33.0
54	17.9	6.9
55	19.4	9.9
56	36.4	16.6
57	48.2	11.1
58	31.8	5.1
59	25.6	4.9
60	42.4	26.6
61	39.6	5.9
62	54.1	10.8
63	18.5	9.4
64	33.4	10.7
65	31.6	22.4
66	30.3	47.2
67	29.4	45.1
68	17.9	49.7
69	18.3	12.4
70	19.5	37.6
71	5.1	3.5
72	15.6	38.0
73	4.5	2.2
74	10.9	19.9
75	11.0	5.6
76	9.0	5.8

[128]:

	No High School Diploma	Assault (Homicide)
count	77.000000	77.000000
mean	21.596104	18.068831
std	12.354995	16.561077
min	2.900000	0.000000
25%	13.400000	4.900000
50%	18.500000	10.800000
75%	29.400000	32.200000
max	58.700000	70.300000

[129]:

```
#Normalize teen birth rate and per capita income
hsd = np.asarray(data['No High School Diploma'])
ah = np.asarray(data['Assault (Homicide)'])
hsdNorm = (hsd - hsd.min())/(np.ptp(hsd))
ahNorm = (ah - ah.min())/(np.ptp(ah))


#Create numpy array with normalized data
dkData = np.asarray(dumbKill)
for i in range(77):
    array = np.array([hsdNorm[i], ahNorm[i]])
```

```python
    dkData[i] = array


print(hsdNorm)
print(ahNorm)
print("New Data Vector:")
print(dkData)
```

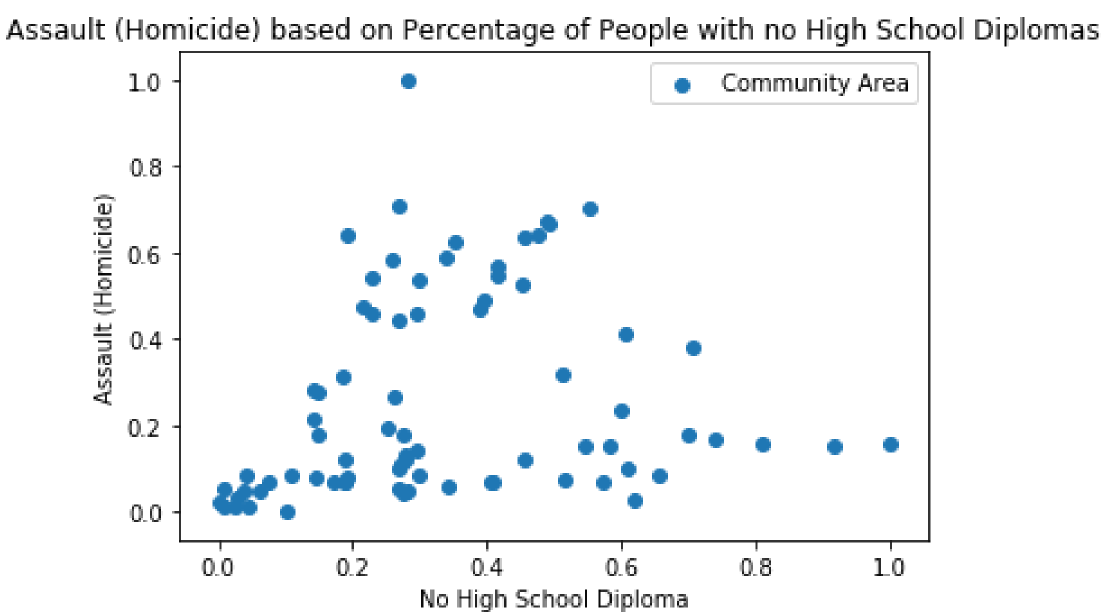
```
[0.27240143 0.29928315 0.19175627 0.17204301 0.04480287 0.
 0.02508961 0.00896057 0.10035842 0.18996416 0.18996416 0.0609319
 0.27419355 0.5734767  0.28315412 0.34229391 0.27060932 0.45698925
 0.61111111 0.69892473 0.40860215 0.27956989 0.60752688 0.18817204
 0.39605735 0.41756272 0.41756272 0.14874552 0.49283154 1.
 0.74193548 0.00896057 0.07526882 0.62007168 0.25089606 0.26344086
 0.55197133 0.29569892 0.14157706 0.45519713 0.04301075 0.2688172
 0.21505376 0.19354839 0.18637993 0.45340502 0.28136201 0.14874552
 0.25985663 0.22759857 0.34050179 0.58422939 0.35304659 0.38888889
 0.2688172  0.29569892 0.60035842 0.81182796 0.51792115 0.40681004
 0.7078853  0.65770609 0.91756272 0.27956989 0.54659498 0.51433692
 0.49103943 0.47491039 0.2688172  0.27598566 0.29749104 0.03942652
 0.22759857 0.02867384 0.14336918 0.14516129 0.109319  ]
[0.10953058 0.08250356 0.07681366 0.07112376 0.01422475 0.01991465
 0.00995733 0.05263158 0.         0.06685633 0.06827881 0.04694168
 0.04267425 0.06685633 0.04694168 0.05832148 0.05263158 0.12233286
 0.09957326 0.18065434 0.06685633 0.12233286 0.41251778 0.12091038
 0.48933144 0.56899004 0.54623044 0.18065434 0.66429587 0.15789474
 0.16642959 0.00995733 0.06827881 0.02560455 0.19345661 0.2688478
 0.70554765 0.45661451 0.21621622 0.6344239  0.08250356 0.44238976
 0.47510669 0.64295875 0.314367   0.52489331 1.         0.27738265
 0.58179232 0.45803698 0.58748222 0.15220484 0.62446657 0.46941679
 0.09815078 0.14082504 0.23613087 0.15789474 0.07254623 0.06970128
 0.37837838 0.08392603 0.15362731 0.13371266 0.15220484 0.31863442
 0.67140825 0.64153627 0.70697013 0.17638691 0.53485064 0.04978663
 0.54054054 0.03129445 0.28307255 0.07965861 0.08250356]
New Data Vector:
[[0.27240143 0.10953058]
 [0.29928315 0.08250356]
 [0.19175627 0.07681366]
 [0.17204301 0.07112376]
 [0.04480287 0.01422475]
 [0.         0.01991465]
 [0.02508961 0.00995733]
 [0.00896057 0.05263158]
 [0.10035842 0.        ]
 [0.18996416 0.06685633]
 [0.18996416 0.06827881]
 [0.0609319  0.04694168]
```

```
[0.27419355 0.04267425]
[0.5734767  0.06685633]
[0.28315412 0.04694168]
[0.34229391 0.05832148]
[0.27060932 0.05263158]
[0.45698925 0.12233286]
[0.61111111 0.09957326]
[0.69892473 0.18065434]
[0.40860215 0.06685633]
[0.27956989 0.12233286]
[0.60752688 0.41251778]
[0.18817204 0.12091038]
[0.39605735 0.48933144]
[0.41756272 0.56899004]
[0.41756272 0.54623044]
[0.14874552 0.18065434]
[0.49283154 0.66429587]
[1.         0.15789474]
[0.74193548 0.16642959]
[0.00896057 0.00995733]
[0.07526882 0.06827881]
[0.62007168 0.02560455]
[0.25089606 0.19345661]
[0.26344086 0.2688478 ]
[0.55197133 0.70554765]
[0.29569892 0.45661451]
[0.14157706 0.21621622]
[0.45519713 0.6344239 ]
[0.04301075 0.08250356]
[0.2688172  0.44238976]
[0.21505376 0.47510669]
[0.19354839 0.64295875]
[0.18637993 0.314367  ]
[0.45340502 0.52489331]
[0.28136201 1.        ]
[0.14874552 0.27738265]
[0.25985663 0.58179232]
[0.22759857 0.45803698]
[0.34050179 0.58748222]
[0.58422939 0.15220484]
[0.35304659 0.62446657]
[0.38888889 0.46941679]
[0.2688172  0.09815078]
[0.29569892 0.14082504]
[0.60035842 0.23613087]
[0.81182796 0.15789474]
[0.51792115 0.07254623]
[0.40681004 0.06970128]
```

```
[0.7078853  0.37837838]
[0.65770609 0.08392603]
[0.91756272 0.15362731]
[0.27956989 0.13371266]
[0.54659498 0.15220484]
[0.51433692 0.31863442]
[0.49103943 0.67140825]
[0.47491039 0.64153627]
[0.2688172  0.70697013]
[0.27598566 0.17638691]
[0.29749104 0.53485064]
[0.03942652 0.04978663]
[0.22759857 0.54054054]
[0.02867384 0.03129445]
[0.14336918 0.28307255]
[0.14516129 0.07965861]
[0.109319   0.08250356]]
```

[130]:

```
#Plot the data
plt.ion()
plt.scatter(dkData[:, 0], dkData[:, 1], label='Community Area')
plt.xlabel('No High School Diploma')
plt.ylabel('Assault (Homicide)')
plt.title('Assault (Homicide) based on Percentage of People with no High School␣
 ↪Diplomas')
plt.legend()
plt.show()
```

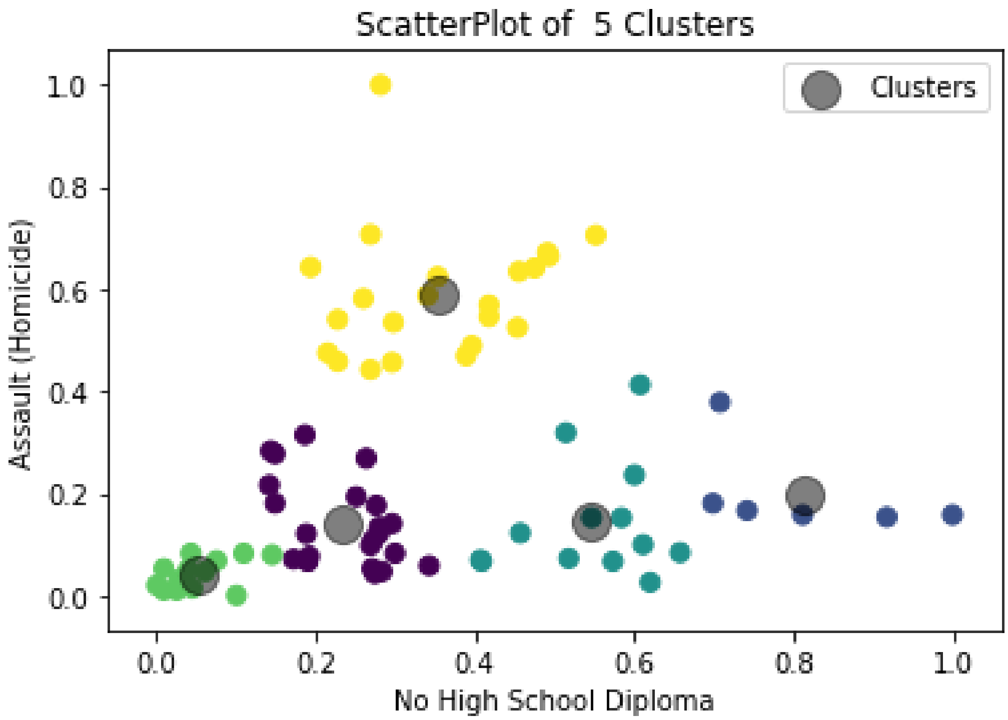


This graphs shows the amount of homicides per community based on the percentage of people without high school diplomas. Looking at the data, it doesn't appear that much pattern can be discerned. One thing that looks evident is that areas where almost everyone has a high school diploma has very little homicide. While the rest of chicagoland seems to have quite a bit of homicide. More accurately, areas above the 0.15 threshhold all seem to have more cases of assault. However, it is a bit odd that the communities with the least high school diplomas do not have as much homicide. Mostly the middle percentage areas.

[170]:
```
# Use kmeans to cluster vector into 5 groups

kmeans2 = KMeans(n_clusters = 5, random_state = 0).fit(dkData)
labels = kmeans2.labels_
groupReps = kmeans2.cluster_centers_
jClust = kmeans2.inertia_

#Print number of iterations
print("Number of iterations: ")
print(kmeans.n_iter_)
print()
print("Labels: ")
print(labels)
print()
print("Cluster Means: ")
print(groupReps)
print()
print("Jclust: ")
print(jClust)
print()

#Plot clusters

plt.scatter(dkData[:, 0], dkData[:, 1], c = kmeans2.predict(dkData), s = 50,␣
 ↪cmap = 'viridis')
#plt.scatter()
plt.title("ScatterPlot of  5 Clusters")
centers = kmeans2.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5,␣
 ↪label='Clusters');
plt.xlabel('No High School Diploma')
plt.ylabel('Assault (Homicide)')
plt.legend()
plt.show()

# Find neighborhoods in each cluster
nH = np.asarray(data['Community Area Name'])
print("Group 1:")
for i in range(len(labels)):
```

```
    if (labels[i] == 3):
        print(nH[i])
print()
print("Group 2:")
for i in range(len(labels)):
    if (labels[i] == 0):
        print(nH[i])
print()
print("Group 3:")
for i in range(len(labels)):
    if (labels[i] == 4):
        print(nH[i])
print()
print("Group 4:")
for i in range(len(labels)):
    if (labels[i] == 2):
        print(nH[i])
print()
print("Group 5:")
for i in range(len(labels)):
    if (labels[i] == 1):
        print(nH[i])
```

```
Number of iterations:
4

Labels:
[0 0 0 0 3 3 3 3 3 0 0 3 0 2 0 0 0 2 2 1 2 0 2 0 4 4 4 0 4 1 1 3 3 2 0 0 4
 4 0 4 3 4 4 4 0 4 4 0 4 4 4 2 4 4 0 0 2 1 2 2 1 2 1 0 2 2 4 4 4 0 4 3 4 3
 0 3 3]

Cluster Means:
[[0.23289699 0.13921702]
 [0.8130227  0.19914651]
 [0.54659498 0.14454536]
 [0.05307417 0.04212715]
 [0.35312805 0.58942196]]

Jclust:
1.1754203082836758
```




```
Group 1:
North Center
Lake View
Lincoln Park
Near North Side
Edison Park
Forest Glen
Loop
Near South Side
Hyde Park
Beverly
Mount Greenwood
O'Hare
Edgewater

Group 2:
Rogers Park
West Ridge
Uptown
Lincoln Square
Norwood Park
Jefferson Park
North Park
Portage Park
```

Irving Park
Dunning
Logan Square
West Town
Near West Side
Douglas
Oakland
Kenwood
Avalon Park
Calumet Heights
Hegewisch
Garfield Ridge
Clearing
Ashburn
Morgan Park

Group 3:
Austin
West Garfield Park
East Garfield Park
North Lawndale
Fuller Park
Grand Boulevard
Washington Park
Woodlawn
South Shore
Chatham
South Chicago
Burnside
Roseland
Pullman
South Deering
West Pullman
Riverdale
West Englewood
Englewood
Greater Grand Crossing
Auburn Gresham
Washington Heights

Group 4:
Albany Park
Montclaire
Belmont Cragin
Avondale
Humboldt Park
Armour Square
East Side

```
Archer Heights
McKinley Park
Bridgeport
West Elsdon
West Lawn
Chicago Lawn

Group 5:
Hermosa
South Lawndale
Lower West Side
Brighton Park
New City
Gage Park
```

In the analysis of the amount of assault (homicides) compared to the percentage of people without high school diplomas, 5 groupings were made using the kmeans algorithm, using 4 iterations to achieve the best clusters. Once again, using the cluster means array, we can see which groups are which in the graph.

The green dots represent the area with the most high school graduates, which also has the lowest amount of homicides. This area includes nicer areas such as Lincoln Park and Lakeview. The purple and light blue dots represent areas with less high school graduates but slightly higher homicide rates. The darker blue dots represent the areas with the least amount of high shool graduates, which also have a similar homicide count as the purple and light blue areas. The outlying group is the yellow group, which has the highest number of assaults, but an average amount of high school diplomas compared to the other areas.

Looking at the communities in group 3, which has the highest number of assaults, it can be seen that these are all actually areas that are closer to each other and on the South Side of Chicago where crime is bad. These areas include Englewood, Garfield Park, and South Chicago.

Based on these findings we can see that group 1 are intellectual areas with low homicide rates. Groups 2, 4, and 5 have lower high school diploma rates, and higher assault rates. However the highest percentage of assaults happen in the areas of Group 3. While this group does not have the highest or lowest amount of high school graduates, it is clear that the area itself is very dangerous.

[143]:
```
#Filter out and compare the amount of people below poverty level in an area and␣
↪cases of cancer
poorCancer = data[['Below Poverty Level', 'Cancer (All Sites)']]
#Display data for two columns
display(poorCancer)
#Calculate descriptive statistics and display
poorCancer.describe()

#Normalize teen birth rate and per capita income
bpl = np.asarray(data['Below Poverty Level'])
cas = np.asarray(data['Cancer (All Sites)'])
bplNorm = (bpl - bpl.min())/(np.ptp(bpl))
```

```python
casNorm = (cas - cas.min())/(np.ptp(cas))


#Create numpy array with normalized data
pcData = np.asarray(poorCancer)
for i in range(77):
    array = np.array([bplNorm[i], casNorm[i]])
    pcData[i] = array

print("New Data Vector:")
print(pcData)
```

```
    Below Poverty Level  Cancer (All Sites)
0                  22.7               176.9
1                  15.1               155.9
2                  22.7               183.3
3                   9.5               153.2
4                   7.1               152.1
5                  10.5               126.9
6                  11.8               152.9
7                  13.4               142.7
8                   5.1               189.7
9                   5.9               180.8
10                  6.4               208.2
11                  6.1               138.7
12                 12.4               143.7
13                 17.1               158.1
14                 12.3               168.7
15                 10.8               169.4
16                  8.3               191.5
17                 12.8               151.0
18                 18.6               152.6
19                 19.1               135.2
20                 14.6               133.9
21                 17.2               148.7
22                 32.6               211.1
23                 15.7               139.6
24                 27.0               261.9
25                 40.3               291.5
26                 39.7               236.8
27                 21.6               202.0
28                 38.6               261.5
29                 28.1               127.4
30                 27.2               141.3
31                 11.1               120.1
32                 11.1               169.0
33                 35.8               162.9
```

34	26.1	269.9
35	38.1	159.7
36	55.5	258.9
37	28.3	218.3
38	23.1	196.4
39	39.1	258.0
40	18.2	144.0
41	28.3	241.3
42	31.5	246.0
43	25.3	213.5
44	16.7	239.6
45	28.0	227.3
46	22.5	191.2
47	12.0	216.6
48	19.5	258.5
49	20.1	262.5
50	24.5	224.0
51	18.7	182.9
52	24.3	263.6
53	61.4	258.3
54	12.1	210.0
55	9.0	231.9
56	13.0	166.3
57	23.0	139.1
58	16.1	148.4
59	17.3	168.9
60	30.6	235.2
61	9.8	180.6
62	20.8	171.0
63	5.9	189.4
64	15.3	145.1
65	22.2	179.3
66	32.3	247.6
67	42.2	252.2
68	25.6	274.4
69	9.5	229.3
70	24.5	243.0
71	5.2	197.6
72	15.7	260.6
73	3.1	201.1
74	13.7	218.2
75	9.5	138.5
76	16.6	162.0

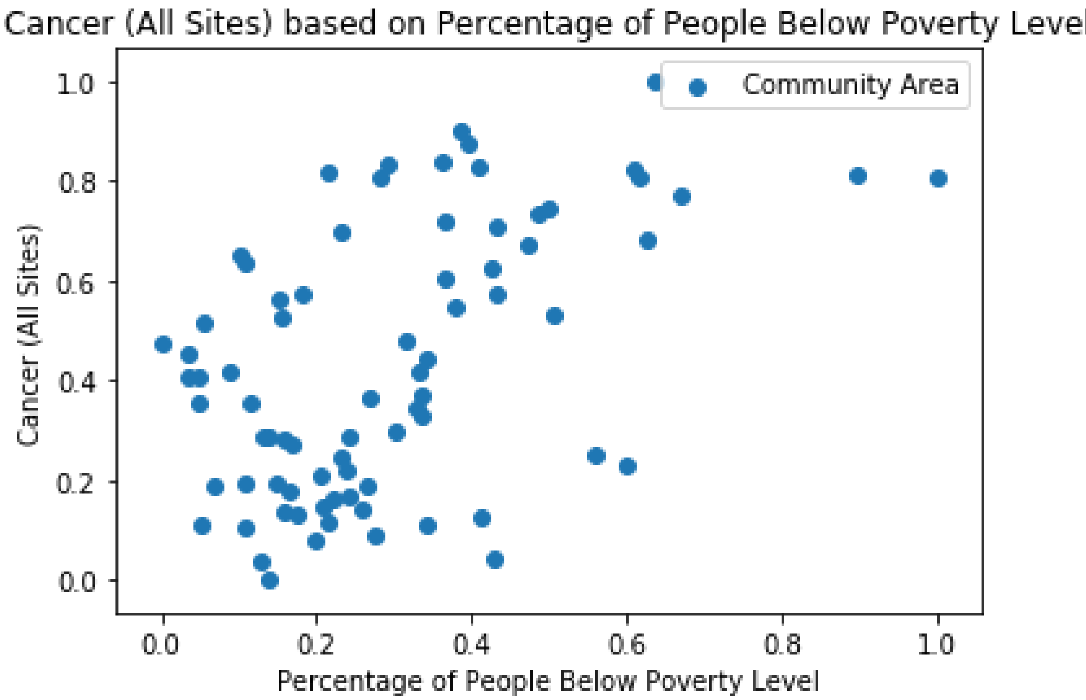
```
New Data Vector:
[[0.33619211 0.33138856]
 [0.2058319  0.20886814]]
```

```
[0.33619211 0.36872812]
[0.10977702 0.19311552]
[0.06861063 0.18669778]
[0.12692967 0.03967328]
[0.14922813 0.19136523]
[0.17667238 0.13185531]
[0.03430532 0.40606768]
[0.04802744 0.35414236]
[0.05660377 0.51400233]
[0.05145798 0.10851809]
[0.15951973 0.13768961]
[0.24013722 0.22170362]
[0.15780446 0.28354726]
[0.13207547 0.28763127]
[0.08919383 0.41656943]
[0.16638079 0.18028005]
[0.26586621 0.18961494]
[0.27444254 0.08809802]
[0.19725557 0.08051342]
[0.24185249 0.16686114]
[0.50600343 0.53092182]
[0.2161235  0.11376896]
[0.40994854 0.82730455]
[0.6380789  1.        ]
[0.62778731 0.68086348]
[0.31732419 0.47782964]
[0.60891938 0.82497083]
[0.42881647 0.04259043]
[0.41337907 0.12368728]
[0.13722127 0.        ]
[0.13722127 0.28529755]
[0.56089194 0.24970828]
[0.39451115 0.873979  ]
[0.60034305 0.23103851]
[0.89879931 0.80980163]
[0.432247   0.57292882]
[0.34305317 0.44515753]
[0.61749571 0.80455076]
[0.25900515 0.13943991]
[0.432247   0.70711785]
[0.48713551 0.73453909]
[0.38078902 0.54492415]
[0.23327616 0.69719953]
[0.4271012  0.62543757]
[0.33276158 0.41481914]
[0.15265866 0.5630105 ]
[0.2813036  0.80746791]
[0.2915952  0.83080513]
```

```
 [0.3670669  0.60618436]
 [0.26758148 0.3663944 ]
 [0.36363636 0.83722287]
 [1.         0.80630105]
 [0.15437393 0.52450408]
 [0.10120069 0.65227538]
 [0.16981132 0.26954492]
 [0.34133791 0.11085181]
 [0.22298456 0.16511085]
 [0.24356775 0.28471412]
 [0.47169811 0.67152859]
 [0.11492281 0.3529755 ]
 [0.30360206 0.29696616]
 [0.04802744 0.40431739]
 [0.20926244 0.14585764]
 [0.32761578 0.3453909 ]
 [0.50085763 0.74387398]
 [0.67066895 0.77071179]
 [0.38593482 0.90023337]
 [0.10977702 0.63710618]
 [0.3670669  0.71703617]
 [0.03602058 0.45215869]
 [0.2161235  0.81971995]
 [0.         0.47257876]
 [0.18181818 0.57234539]
 [0.10977702 0.10735123]
 [0.23156089 0.24445741]]
```

[162]:

```
#Plot the data
plt.ion()
plt.scatter(pcData[:, 0], pcData[:, 1], label='Community Area')
plt.xlabel('Percentage of People Below Poverty Level')
plt.ylabel('Cancer (All Sites)')
plt.title('Cancer (All Sites) based on Percentage of People Below Poverty␣
 ↪Level')
plt.legend()
plt.show()
```

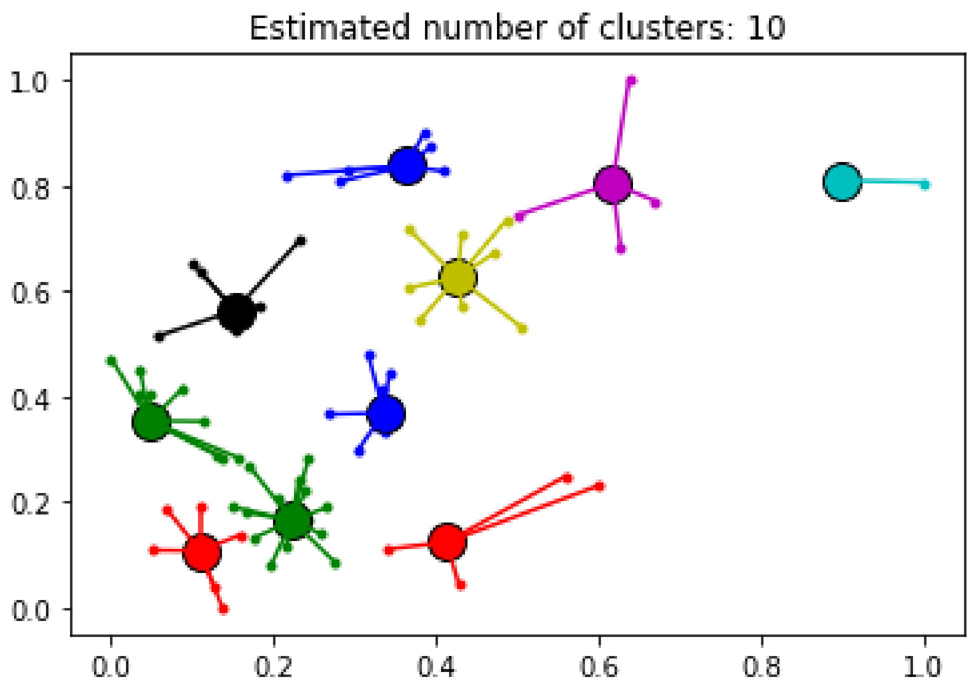


This is a scatterplot of community areas and the amount of cancer cases based on the the percentage of people below the poverty level. There is a general upward movement of the dots, which shows that there is a slight pattern of places with less people belew poverty level having less cases of cancer than the areas with the most people below poverty levels. A linear trendline would have a positive slope in this case.

[167]:
```
#Use affinity propagation to get number of clusters
af = AffinityPropagation().fit(pcData)

clusterCenterIndices = af.cluster_centers_indices_
numClusters = len(clusterCenterIndices)
labels = af.labels_

print(clusterCenterIndices)
print(numClusters)
print(labels)


plt.close('all')
plt.figure(1)
plt.clf()

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(numClusters), colors):
```

```
    class_members = labels == k
    cluster_center = pcData[clusterCenterIndices[k]]
    plt.plot(pcData[class_members, 0], pcData[class_members, 1], col + '.')
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,␣
↪markeredgecolor='k', markersize=14)
    for x in pcData[class_members]:
        plt.plot([cluster_center[0], x[0]], [cluster_center[1], x[1]], col)

plt.title('Estimated number of clusters: %d' % numClusters)
plt.show()
```

```
[ 2  9 30 36 39 45 47 52 58 75]
10
[0 8 0 9 9 9 8 8 1 1 6 9 9 8 1 1 1 8 8 8 8 8 5 8 7 4 4 0 4 2 2 9 1 2 7 2 3
 5 0 4 8 5 5 5 6 5 0 6 7 7 5 0 7 3 6 6 8 2 8 8 5 1 0 1 8 0 4 4 7 6 5 1 7 1
 6 9 8]
```

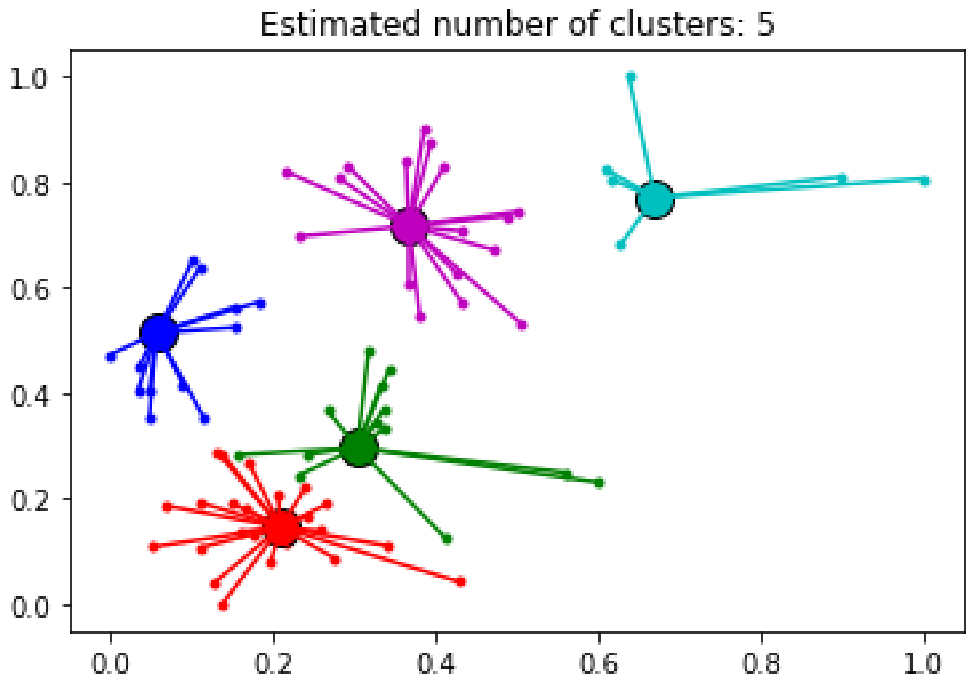


Affinity propagation results in 10 clusters. We can try to see if we can lower the number of clusters. Perhaps normalizing the data skews the algorithm a bit.

[171]:
```
#Use affinity propagation to get number of clusters with set preference
#Set preference near minimal negative squared distance in dataset
af = AffinityPropagation(preference= -.5).fit(pcData)

clusterCenterIndices = af.cluster_centers_indices_
```

```
numClusters = len(clusterCenterIndices)
labels = af.labels_

print(clusterCenterIndices)
print(numClusters)
print(labels)


plt.close('all')
plt.figure(1)
plt.clf()

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(numClusters), colors):
    class_members = labels == k
    cluster_center = pcData[clusterCenterIndices[k]]
    plt.plot(pcData[class_members, 0], pcData[class_members, 1], col + '.')
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col, markeredgecolor='k', markersize=14)
    for x in pcData[class_members]:
        plt.plot([cluster_center[0], x[0]], [cluster_center[1], x[1]], col)

plt.title('Estimated number of clusters: %d' % numClusters)
plt.show()
```

```
[10 62 64 67 70]
5
[1 2 1 2 2 2 2 2 0 0 0 2 2 2 1 2 0 2 2 2 2 2 4 2 4 3 3 1 3 2 1 2 2 1 4 1 3
 4 1 3 2 4 4 4 4 4 1 0 4 4 4 1 4 3 0 0 2 2 2 1 4 0 1 0 2 1 4 3 4 0 4 0 4 0
 0 2 1]
```



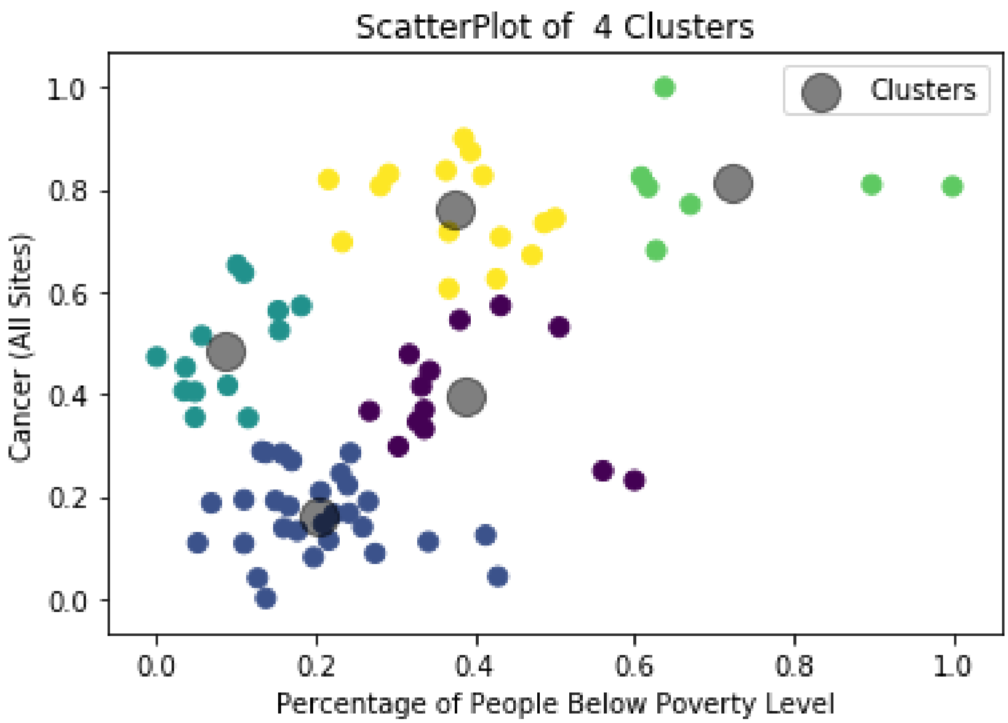
Setting the preference to -.5 leads to 5 clusters which looks much better.

[173]:
```
# Use kmeans to cluster vector into 5 groups after affinity propagation

kmeans2 = KMeans(n_clusters = 5, random_state = 0).fit(pcData)
labels = kmeans2.labels_
groupReps = kmeans2.cluster_centers_
jClust = kmeans2.inertia_

#Print number of iterations
print("Number of iterations: ")
print(kmeans.n_iter_)
print()
print("Labels: ")
print(labels)
print()
print("Cluster Means: ")
print(groupReps)
print()
print("Jclust: ")
print(jClust)
print()

#Plot clusters
```

```
plt.scatter(pcData[:, 0], pcData[:, 1], c = kmeans2.predict(pcData), s = 50,␣
 ↪cmap = 'viridis')
#plt.scatter()
plt.title("ScatterPlot of  4 Clusters")
centers = kmeans2.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5,␣
 ↪label='Clusters');
plt.xlabel('Percentage of People Below Poverty Level')
plt.ylabel('Cancer (All Sites)')
plt.legend()
plt.show()

# Find neighborhoods in each cluster
nH = np.asarray(data['Community Area Name'])
print("Group 1:")
for i in range(len(labels)):
    if (labels[i] == 2):
        print(nH[i])
print()
print("Group 2:")
for i in range(len(labels)):
    if (labels[i] == 1):
        print(nH[i])
print()
print("Group 3:")
for i in range(len(labels)):
    if (labels[i] == 4):
        print(nH[i])
print()
print("Group 4:")
for i in range(len(labels)):
    if (labels[i] == 0):
        print(nH[i])
print()
print("Group 5:")
for i in range(len(labels)):
    if (labels[i] == 3):
        print(nH[i])
```

```
Number of iterations:
4

Labels:
[0 1 0 1 1 1 1 1 2 2 2 1 1 1 1 1 2 1 1 1 1 1 0 1 4 3 3 0 3 1 1 1 1 0 4 0 3
 0 0 3 1 4 4 0 4 4 0 2 4 4 4 0 4 3 2 2 1 1 1 1 4 2 0 2 1 0 4 3 4 2 4 2 4 2
 2 1 1]
```

```
Cluster Means:
[[0.38804592 0.39816893]
 [0.20151417 0.16305879]
 [0.0866869  0.48631182]
 [0.72310708 0.81388565]
 [0.37530017 0.75997666]]

Jclust:
1.2522390306736588
```




```
Group 1:
Edison Park
Norwood Park
Jefferson Park
Dunning
Calumet Heights
Hegewisch
Garfield Ridge
West Elsdon
Clearing
Ashburn
Beverly
Mount Greenwood
```

Morgan Park

Group 2:
West Ridge
Lincoln Square
North Center
Lake View
Lincoln Park
Near North Side
Forest Glen
North Park
Albany Park
Portage Park
Irving Park
Montclaire
Belmont Cragin
Hermosa
Avondale
Logan Square
West Town
South Lawndale
Lower West Side
Loop
Near South Side
Hyde Park
Archer Heights
Brighton Park
McKinley Park
Bridgeport
West Lawn
O'Hare
Edgewater

Group 3:
Austin
Douglas
Woodlawn
South Shore
Avalon Park
South Chicago
Roseland
Pullman
South Deering
West Pullman
New City
West Englewood
Greater Grand Crossing
Auburn Gresham

```
Washington Heights

Group 4:
Rogers Park
Uptown
Humboldt Park
Near West Side
Armour Square
Oakland
Grand Boulevard
Kenwood
Chatham
Burnside
East Side
Gage Park
Chicago Lawn

Group 5:
West Garfield Park
East Garfield Park
North Lawndale
Fuller Park
Washington Park
Riverdale
Englewood
```

Once again, comparing the cluster means to the graph, we can see which dots represent which group and the communities within those groups. It seems apparent that the green dots, represent the worst area, with not only the highest percentage of people belwo the poverty level, but the highest number of cancer cases as well. These communites in group five are definitely some of the lowest income areas in Chicagoland, like Englewood, Fuller Park, and Garfield Park. It can also be seen that groups 1 and 3 also have higher rates of cancer, despite having less people below poverty rates. Looking at the communities in these groups, it looks like many of these communities are on the South Side, or on the edge of chicago's city limits. The South Side seems to have a lot of cancer cases, and there might be a lot of people without insurance as well. Groups 2 and 4 represent the communities with the lowest amount of cancer cases. These areas are closer to the center of chicago and are generally regarded as nicer areas.

Overall, with these three charts of Chicagoland data, I think it can be seen that there are definitely nicer areas and lower class areas in chicago. The disparage can be noticed by geographical area if all the communities are mapped out. Richer areas near the loop, north side and west side seem to be wealthier, smarter, safer, and healthier; whereas areas like the south side, south west, and far west sides are the opposite.