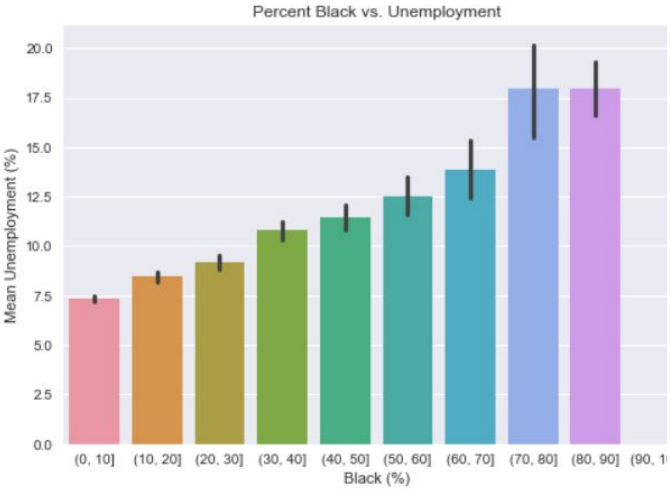
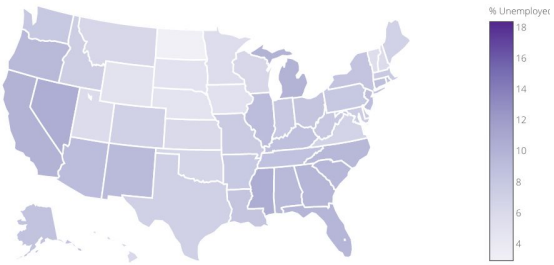
Project 1: Data Cleaning, Visualization, & Mining

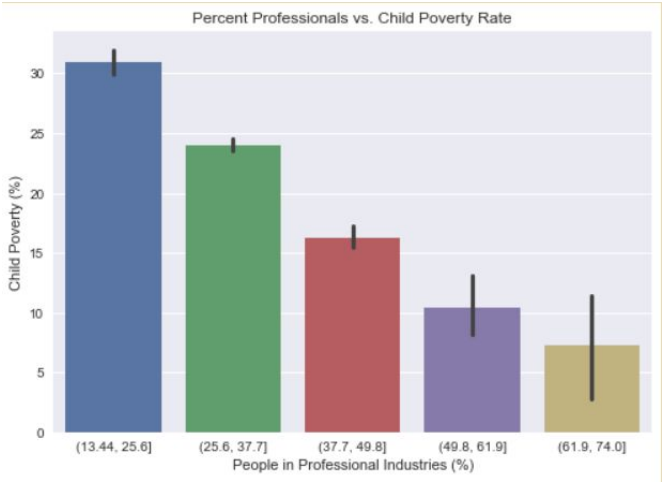
Avneesh Mehta, Newman Hu, Zachary Golan-Strieb

Data Visualization

The first correlation we found was between racial demographics and unemployment. A higher percentage of Whites was negatively correlated with unemployment , while a higher percentage of Blacks was positively correlated with unemployment (Graph 1 and 2). The higher the percentage of Blacks in a county, the higher the unemployment rate. We also found correlations between the percent of professionals and the child poverty rate and the percent of professionals and the income per capita. There was a negative correlation between the percentage of professionals in a county and the child poverty rate (Graph 3). The more professionals in a county, the lower child poverty was. There was a positive correlation between the percentage of professionals in a county and the income per capita (Graph 4). The more professionals there were in a county, the higher the income per capita was. We decided to regress on % Professionals vs. Child Poverty, % Professionals vs. Income per Capita, and % Black vs. Unemployment.

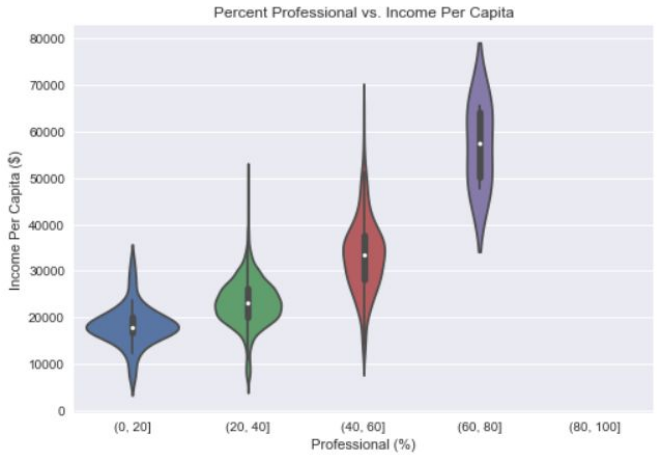


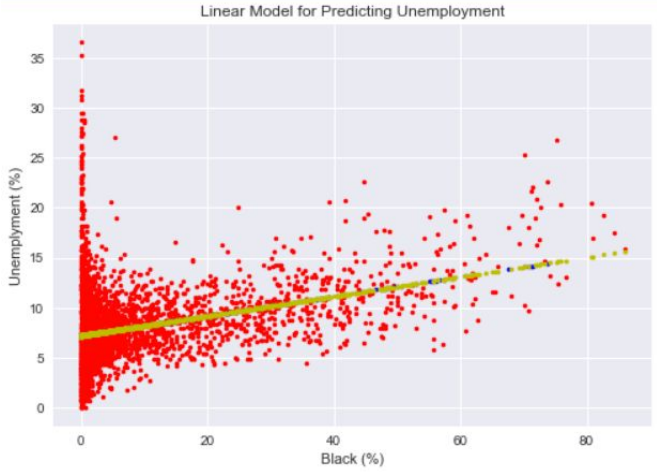


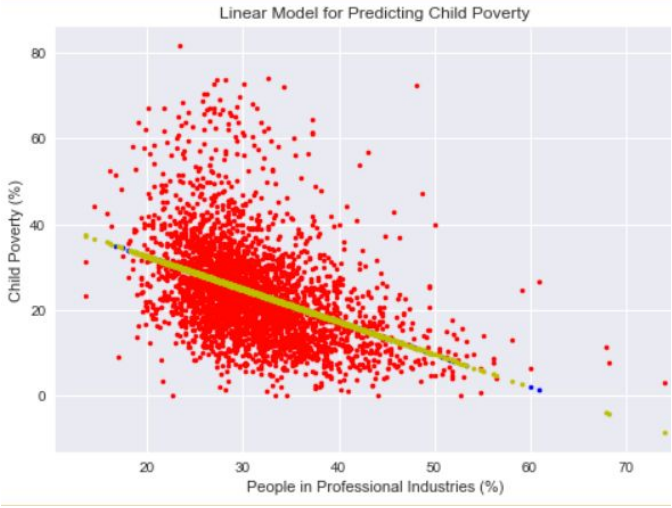


Regression

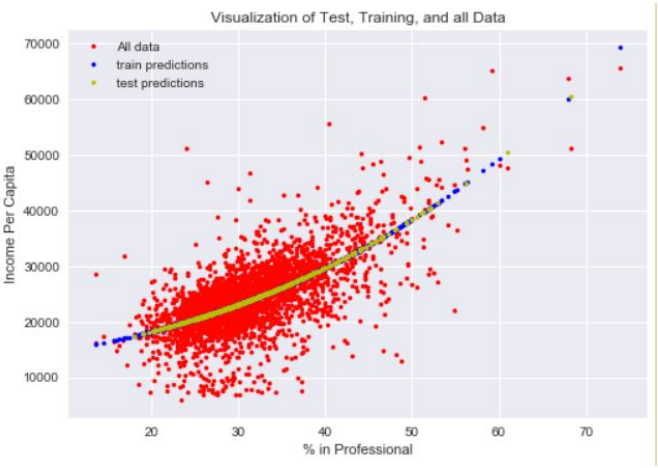
We investigated three financial indicators with our linear regressions: (1) percent unemployment (2) percent of children in poverty, and (3) income per capita. We used a simple linear regression for the black percentage vs unemployment percentage model and found a slight positive correlation (Graph 5). The accuracy of a linear model is very limited due to the distribution of the data and our best fit had an R squared score of 0.16. Similarly, we used a simple linear regression for the model predicting child poverty % vs % of people in professional industry (Graph 6). There was a negative correlation because the higher percentage of people in professional industry meant lower percent of children in poverty. The accuracy of this model is also quite limited at and R squared around 0.17. Finally, for modeling % of people in professional industry vs income per capita we used a ridge regression to regularize using the l2 norm (Graph 7). We explored the fitting of higher order polynomials and found that the accuracy improved marginally but







did not show any major improvements past a 2nd degree. The model we used to describe the data consisted of a 3rd degree polynomial. This gave us the most accurate model with an R squared score of about 0.4.



Conclusion

The accuracy of linear regression modeling for this data is severely hampered by the nonlinear relationships between variables in the census. Although linear relationships do exist, most data is distributed over a wide range that makes linear models inefficient.

Census Data Analysis

March 1, 2018

In [1]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import missingno as msno
import seaborn as sns

import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff
py.init_notebook_mode(connected=True)

sns.set()
```

In [2]:
```
census = pd.read_csv('acs2015_county_data.csv')
```

In [3]:
```
census
```

Out[3]:

	CensusId	State	County	TotalPop	Men	Women	\
0	1001	Alabama	Autauga	55221	26745	28476	
1	1003	Alabama	Baldwin	195121	95314	99807	
2	1005	Alabama	Barbour	26932	14497	12435	
3	1007	Alabama	Bibb	22604	12073	10531	
4	1009	Alabama	Blount	57710	28512	29198	
5	1011	Alabama	Bullock	10678	5660	5018	
6	1013	Alabama	Butler	20354	9502	10852	
7	1015	Alabama	Calhoun	116648	56274	60374	
8	1017	Alabama	Chambers	34079	16258	17821	
9	1019	Alabama	Cherokee	26008	12975	13033	
10	1021	Alabama	Chilton	43819	21619	22200	
11	1023	Alabama	Choctaw	13395	6382	7013	
12	1025	Alabama	Clarke	25070	11834	13236	
13	1027	Alabama	Clay	13537	6671	6866	
14	1029	Alabama	Cleburne	15002	7334	7668	
15	1031	Alabama	Coffee	50884	25174	25710	

16	1033	Alabama	Colbert	54444	26303	28141	
17	1035	Alabama	Conecuh	12865	6176	6689	
18	1037	Alabama	Coosa	11027	5579	5448	
19	1039	Alabama	Covington	37886	18339	19547	
20	1041	Alabama	Crenshaw	13938	6863	7075	
21	1043	Alabama	Cullman	80965	40081	40884	
22	1045	Alabama	Dale	49866	24708	25158	
23	1047	Alabama	Dallas	42154	19450	22704	
24	1049	Alabama	DeKalb	71068	35474	35594	
25	1051	Alabama	Elmore	80763	39362	41401	
26	1053	Alabama	Escambia	37935	19524	18411	
27	1055	Alabama	Etowah	103766	50207	53559	
28	1057	Alabama	Fayette	16896	8477	8419	
29	1059	Alabama	Franklin	31634	15311	16323	
...	...	...	...	...	...	...	
3190	72095	Puerto Rico	Maunabo	11701	5779	5922	
3191	72097	Puerto Rico	Mayagüez	83418	39978	43440	
3192	72099	Puerto Rico	Moca	38815	19039	19776	
3193	72101	Puerto Rico	Morovis	32294	16067	16227	
3194	72103	Puerto Rico	Naguabo	26804	12733	14071	
3195	72105	Puerto Rico	Naranjito	29751	14700	15051	
3196	72107	Puerto Rico	Orocovis	22595	11376	11219	
3197	72109	Puerto Rico	Patillas	18441	9075	9366	
3198	72111	Puerto Rico	Peñuelas	22770	11107	11663	
3199	72113	Puerto Rico	Ponce	156054	75153	80901	
3200	72115	Puerto Rico	Quebradillas	25205	12258	12947	
3201	72117	Puerto Rico	Rincón	14841	7249	7592	
3202	72119	Puerto Rico	Río Grande	53029	25845	27184	
3203	72121	Puerto Rico	Sabana Grande	24307	11536	12771	
3204	72123	Puerto Rico	Salinas	30114	14631	15483	
3205	72125	Puerto Rico	San Germán	34125	16590	17535	
3206	72127	Puerto Rico	San Juan	371400	170416	200984	
3207	72129	Puerto Rico	San Lorenzo	39778	19485	20293	
3208	72131	Puerto Rico	San Sebastián	40471	19723	20748	
3209	72133	Puerto Rico	Santa Isabel	22913	11085	11828	
3210	72135	Puerto Rico	Toa Alta	74603	36028	38575	
3211	72137	Puerto Rico	Toa Baja	85242	40215	45027	
3212	72139	Puerto Rico	Trujillo Alto	71886	34036	37850	
3213	72141	Puerto Rico	Utuado	31474	15368	16106	
3214	72143	Puerto Rico	Vega Alta	39319	18762	20557	
3215	72145	Puerto Rico	Vega Baja	56858	27379	29479	
3216	72147	Puerto Rico	Vieques	9130	4585	4545	
3217	72149	Puerto Rico	Villalba	24685	12086	12599	
3218	72151	Puerto Rico	Yabucoa	36279	17648	18631	
3219	72153	Puerto Rico	Yauco	39474	19047	20427	

	Hispanic	White	Black	Native	...	Walk	OtherTransp	\
0	2.6	75.8	18.5	0.4	...	0.5	1.3	

1	4.5	83.1	9.5	0.6	...	1.0	1.4
2	4.6	46.2	46.7	0.2	...	1.8	1.5
3	2.2	74.5	21.4	0.4	...	0.6	1.5
4	8.6	87.9	1.5	0.3	...	0.9	0.4
5	4.4	22.2	70.7	1.2	...	5.0	1.7
6	1.2	53.3	43.8	0.1	...	0.8	0.6
7	3.5	73.0	20.3	0.2	...	1.2	1.2
8	0.4	57.3	40.3	0.2	...	0.3	0.4
9	1.5	91.7	4.8	0.6	...	0.6	0.7
10	7.6	80.5	10.2	0.4	...	1.1	1.4
11	0.4	55.9	42.9	0.0	...	1.9	1.4
12	0.3	53.4	45.3	0.0	...	1.3	2.6
13	3.2	79.9	14.4	0.7	...	1.8	0.9
14	2.3	92.5	2.9	0.2	...	0.2	1.0
15	6.4	71.5	17.2	0.8	...	1.0	1.3
16	2.4	78.9	15.6	0.6	...	0.9	0.7
17	1.6	51.0	44.7	0.3	...	0.6	0.3
18	2.1	65.2	30.7	0.1	...	0.9	1.0
19	1.5	83.3	13.0	0.5	...	1.5	1.9
20	1.7	70.5	23.5	0.8	...	1.3	0.8
21	4.3	92.2	1.1	0.4	...	1.5	1.0
22	6.0	69.8	19.4	0.5	...	1.6	1.7
23	0.3	28.6	69.2	0.2	...	2.2	1.6
24	14.0	80.9	1.8	1.1	...	0.6	0.9
25	2.8	73.6	21.0	0.2	...	0.4	1.2
26	1.2	60.7	33.4	3.2	...	1.2	0.5
27	3.6	78.5	15.4	0.3	...	0.4	0.5
28	0.7	85.3	12.0	0.0	...	0.3	0.2
29	15.7	78.5	4.1	0.7	...	0.2	4.7
...	...	...	...	...	...	...	...
3190	99.9	0.1	0.0	0.0	...	4.7	2.5
3191	99.0	0.7	0.1	0.0	...	4.1	1.2
3192	99.2	0.4	0.0	0.0	...	4.3	0.8
3193	99.8	0.1	0.0	0.0	...	4.7	1.2
3194	99.7	0.1	0.0	0.0	...	4.7	0.2
3195	99.6	0.1	0.0	0.0	...	1.7	0.1
3196	99.9	0.1	0.0	0.0	...	5.6	1.1
3197	99.8	0.1	0.0	0.0	...	3.2	2.3
3198	99.3	0.7	0.0	0.0	...	3.8	1.9
3199	99.3	0.5	0.1	0.0	...	2.8	1.8
3200	99.2	0.5	0.2	0.0	...	6.5	0.3
3201	91.0	5.0	0.0	0.0	...	4.8	0.0
3202	99.3	0.6	0.1	0.0	...	2.1	4.1
3203	99.0	1.0	0.0	0.0	...	3.1	0.9
3204	99.3	0.3	0.1	0.0	...	6.0	2.3
3205	99.6	0.3	0.1	0.0	...	3.0	0.9
3206	98.2	1.2	0.2	0.0	...	6.1	3.0
3207	99.8	0.2	0.0	0.0	...	3.6	0.4

3208	99.3	0.6	0.0	0.0	...	3.4	0.9
3209	99.8	0.2	0.0	0.0	...	3.4	0.8
3210	99.4	0.3	0.1	0.0	...	1.4	2.0
3211	99.5	0.3	0.1	0.0	...	2.5	4.8
3212	99.6	0.4	0.0	0.0	...	1.4	2.9
3213	99.6	0.3	0.1	0.0	...	7.8	0.7
3214	98.6	1.1	0.0	0.0	...	3.1	2.3
3215	96.4	3.4	0.1	0.0	...	1.2	1.3
3216	96.7	2.9	0.0	0.0	...	10.8	0.0
3217	99.7	0.0	0.0	0.0	...	3.2	0.0
3218	99.8	0.2	0.0	0.0	...	2.3	2.3
3219	99.5	0.5	0.0	0.0	...	1.6	0.7

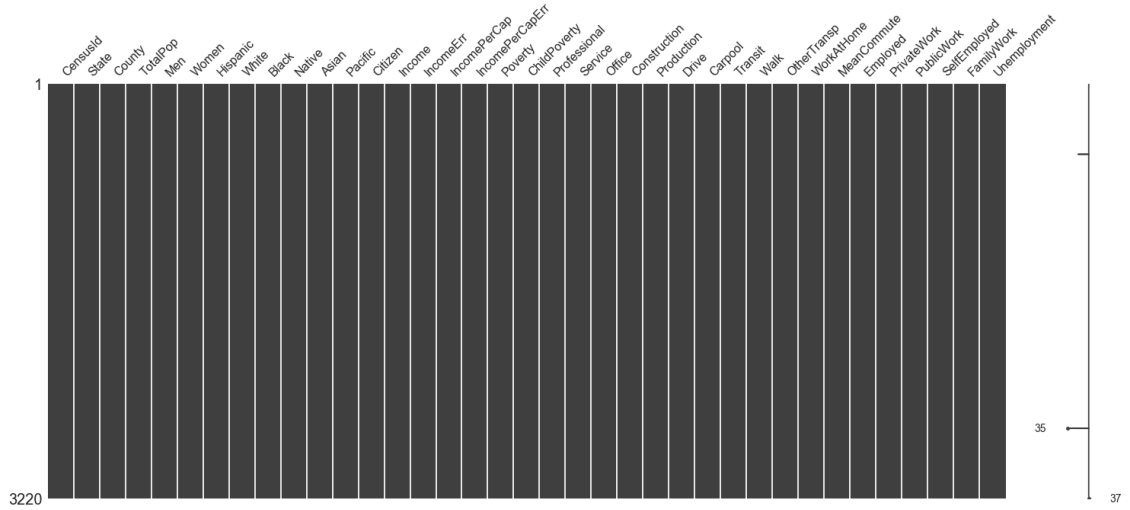
	WorkAtHome	MeanCommute	Employed	PrivateWork	PublicWork	\
0	1.8	26.5	23986	73.6	20.9	
1	3.9	26.4	85953	81.5	12.3	
2	1.6	24.1	8597	71.8	20.8	
3	0.7	28.8	8294	76.8	16.1	
4	2.3	34.9	22189	82.0	13.5	
5	2.8	27.5	3865	79.5	15.1	
6	1.7	24.6	7813	77.4	16.2	
7	2.7	24.1	47401	74.1	20.8	
8	2.1	25.1	13689	85.1	12.1	
9	2.5	27.4	10155	73.1	18.5	
10	1.9	32.0	17895	80.3	15.2	
11	0.8	30.7	4405	77.5	16.8	
12	1.1	23.0	8161	83.1	13.6	
13	2.1	30.3	5180	77.5	14.7	
14	3.4	33.3	6065	76.3	15.3	
15	2.8	20.9	20912	70.3	23.5	
16	1.5	22.8	21290	78.6	16.1	
17	1.3	29.7	3718	77.8	13.0	
18	2.1	28.6	3852	76.6	17.5	
19	2.2	25.5	14202	77.1	15.8	
20	4.6	27.9	5620	74.1	17.8	
21	2.6	26.4	32282	81.6	12.2	
22	2.6	20.4	18735	73.1	20.9	
23	2.2	23.9	14094	76.6	16.7	
24	2.1	22.8	27778	78.8	12.6	
25	2.2	28.5	33636	70.4	24.1	
26	1.1	23.1	12294	74.5	19.0	
27	3.8	23.4	41236	79.8	14.4	
28	1.9	28.4	6122	81.9	14.2	
29	1.4	29.0	12249	78.5	14.6	
...	...	...	...	...	...	
3190	0.0	28.8	2459	46.7	48.5	
3191	2.2	21.8	21048	65.9	25.3	
3192	0.6	26.0	9245	59.1	29.1	

3193	1.0	37.9	7740	72.5	22.9
3194	2.7	33.2	7342	63.1	23.0
3195	1.6	35.4	7315	70.5	22.9
3196	1.9	34.8	4552	58.3	32.6
3197	1.3	27.3	3884	55.1	39.3
3198	0.9	26.4	5412	60.8	31.0
3199	2.5	23.0	41711	69.8	20.4
3200	1.7	27.7	6284	66.7	24.1
3201	5.1	25.0	4406	71.3	19.5
3202	2.6	38.6	17033	69.8	23.4
3203	1.4	24.7	5339	67.2	29.8
3204	1.3	24.1	7804	61.6	32.4
3205	3.7	26.5	8803	67.1	22.7
3206	2.8	26.0	133186	67.6	18.9
3207	0.9	34.2	11327	71.5	22.2
3208	2.8	27.9	8785	64.6	23.7
3209	1.7	25.6	6895	68.7	25.3
3210	0.6	41.2	26760	70.3	21.6
3211	2.8	35.6	30020	68.9	23.3
3212	3.2	34.5	28059	68.9	21.0
3213	1.0	32.4	6819	58.0	34.0
3214	0.9	35.0	9804	75.7	20.3
3215	0.3	32.0	13660	78.3	17.6
3216	1.4	14.0	2860	44.5	41.6
3217	3.3	26.9	6795	59.2	27.5
3218	1.5	29.5	8083	65.1	27.6
3219	3.1	24.6	8923	68.0	27.6

	SelfEmployed	FamilyWork	Unemployment
0	5.5	0.0	7.6
1	5.8	0.4	7.5
2	7.3	0.1	17.6
3	6.7	0.4	8.3
4	4.2	0.4	7.7
5	5.4	0.0	18.0
6	6.2	0.2	10.9
7	5.0	0.1	12.3
8	2.8	0.0	8.9
9	7.9	0.5	7.9
10	4.1	0.5	9.1
11	5.4	0.3	13.6
12	3.3	0.0	19.4
13	7.8	0.0	9.4
14	8.4	0.0	8.3
15	5.7	0.5	7.1
16	5.2	0.0	9.0
17	8.3	0.8	22.6
18	5.9	0.0	17.0

```
19            6.7         0.4          11.2
20            7.9         0.3           9.7
21            5.8         0.4           7.3
22            5.9         0.1          10.9
23            6.2         0.4          16.4
24            8.5         0.1           7.7
25            5.5         0.0           8.3
26            6.4         0.1          15.6
27            5.6         0.2           9.0
28            3.8         0.1          10.0
29            6.9         0.0           9.8
...           ...         ...           ...
3190          4.8         0.0          27.9
3191          8.5         0.2          22.9
3192         11.7         0.0          23.0
3193          4.4         0.2          24.7
3194         14.0         0.0          10.9
3195          6.5         0.0          21.9
3196          8.9         0.2          31.2
3197          4.2         1.4          27.5
3198          8.2         0.0          26.1
3199          9.5         0.3          20.0
3200          9.3         0.0          16.1
3201          9.1         0.0          17.6
3202          6.4         0.4          23.9
3203          3.0         0.0          24.3
3204          6.0         0.0          12.1
3205          9.8         0.4           6.8
3206         13.4         0.2          15.8
3207          6.2         0.1          15.8
3208         10.9         0.9          29.4
3209          6.0         0.0          12.2
3210          7.8         0.3          15.7
3211          7.7         0.2          19.0
3212         10.1         0.0           7.9
3213          7.3         0.7          28.8
3214          4.1         0.0          21.7
3215          4.1         0.0          15.2
3216         13.6         0.3          12.2
3217         13.1         0.2          25.9
3218          7.3         0.0          24.3
3219          4.4         0.0          27.1

[3220 rows x 37 columns]
```

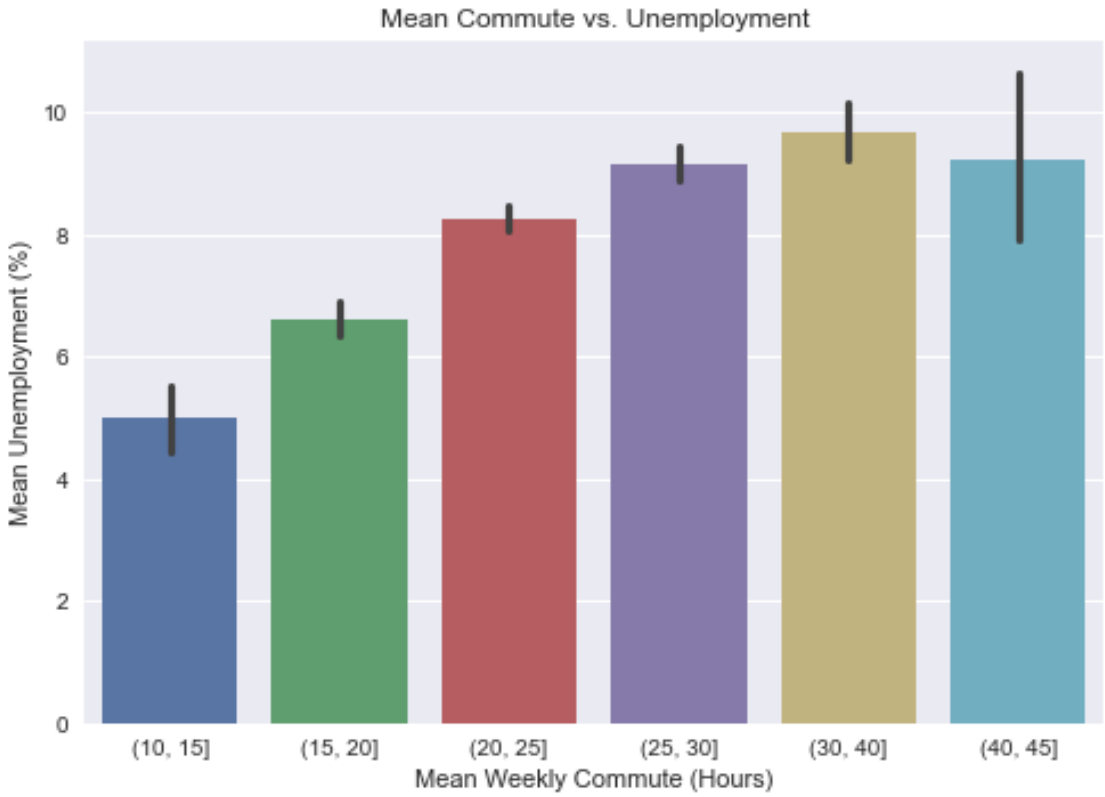
In [4]:
```
msno.matrix(census)
```



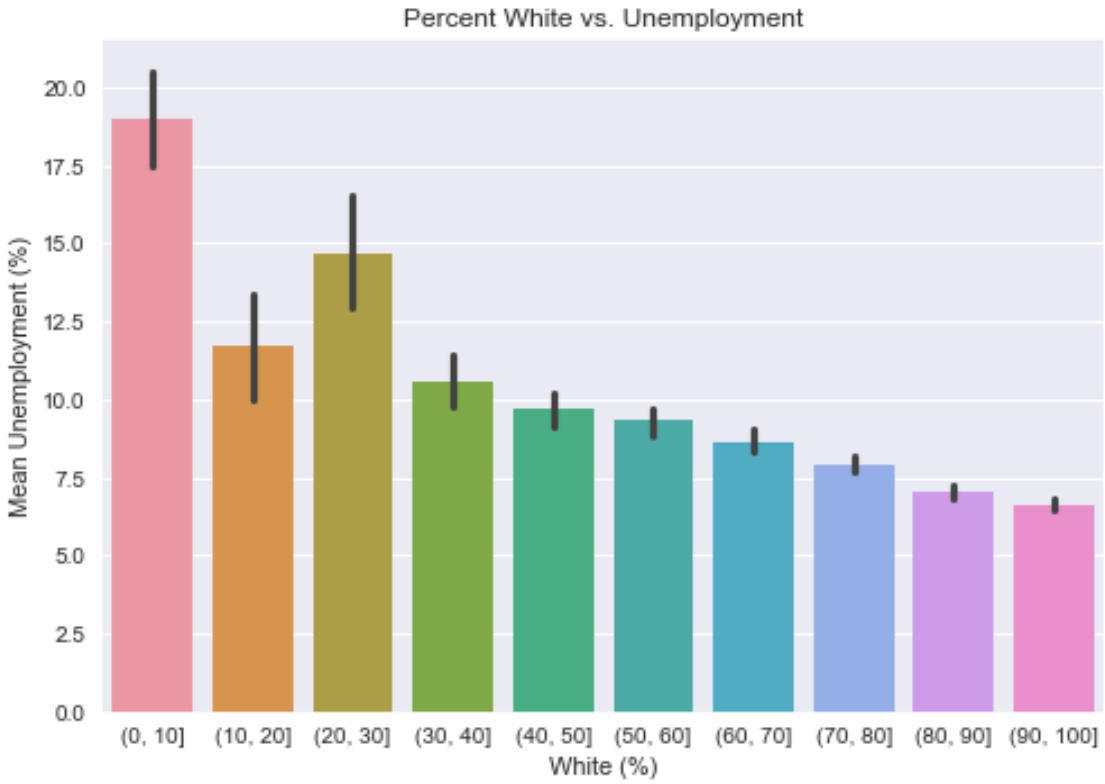
In [5]:
```
census['PercentMen'] = 100 * census['Men'] / census['TotalPop']
census['PercentWomen'] = 100 * census['Women'] / census['TotalPop']
census['PercentCitizen'] = 100 * census['Citizen'] / census['TotalPop']
census['PercentNonCitizen'] = 100 - census['PercentCitizen']
```

In [6]:
```
mean_commute6 = pd.cut(census['MeanCommute'], [10, 15, 20, 25, 30, 40, 45])
unemployment = census['Unemployment']
fig = sns.barplot(x=mean_commute6, y=unemployment)
fig.set(xlabel='Mean Weekly Commute (Hours)', ylabel='Mean Unemployment (%)')
fig.set_title('Mean Commute vs. Unemployment')
fig
```

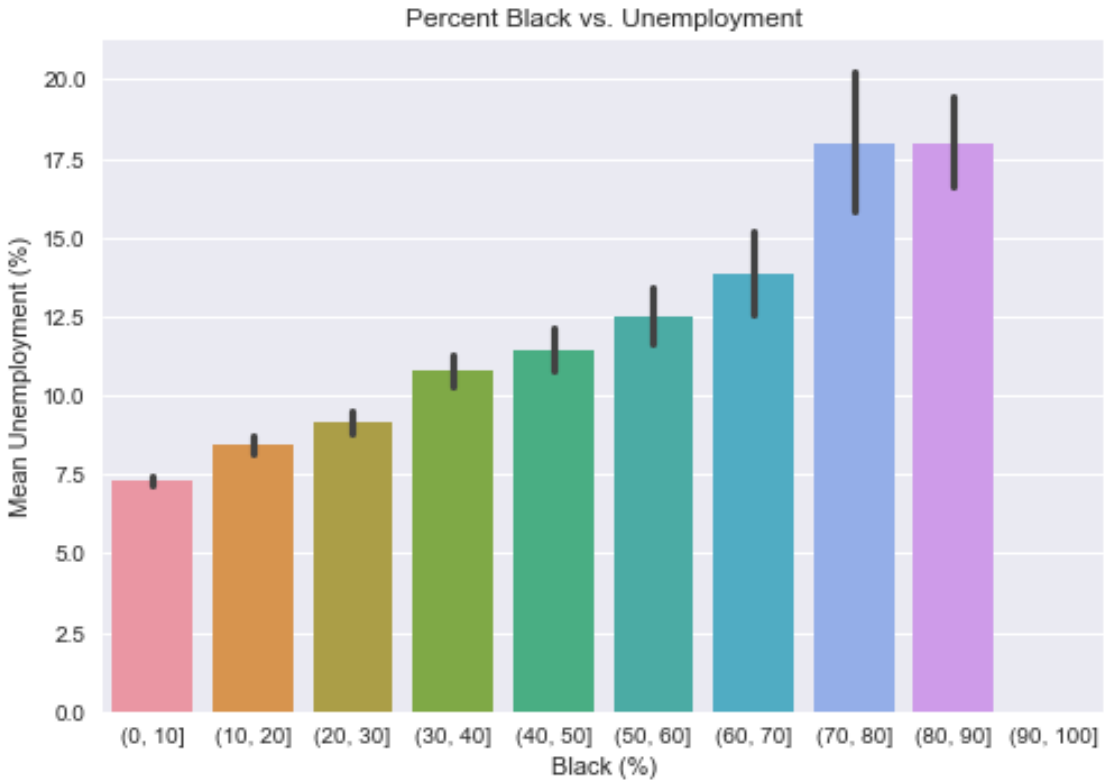
Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11a892b70>
```



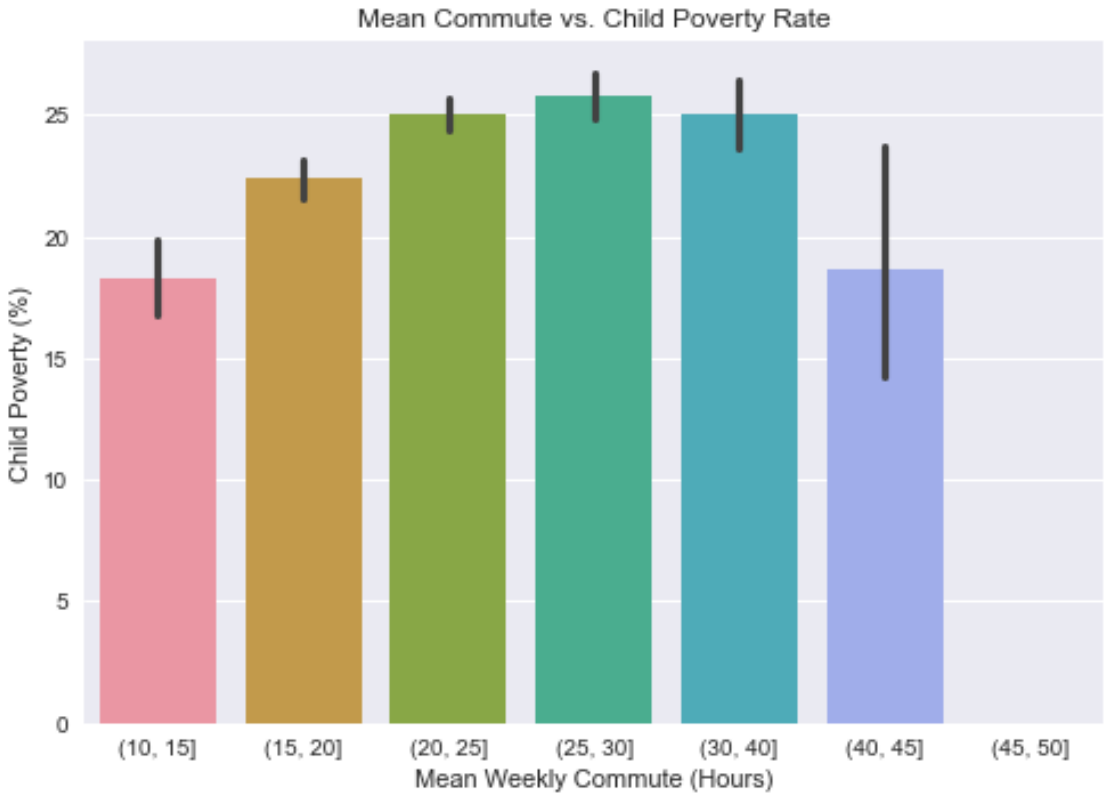

```
In [7]: white10 = pd.cut(census['White'], [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
        unemployment = census['Unemployment']
        fig = sns.barplot(x=white10, y=unemployment)
        fig.set(xlabel='White (%)', ylabel='Mean Unemployment (%)')
        fig.set_title('Percent White vs. Unemployment')
        fig

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21aa6630>
```



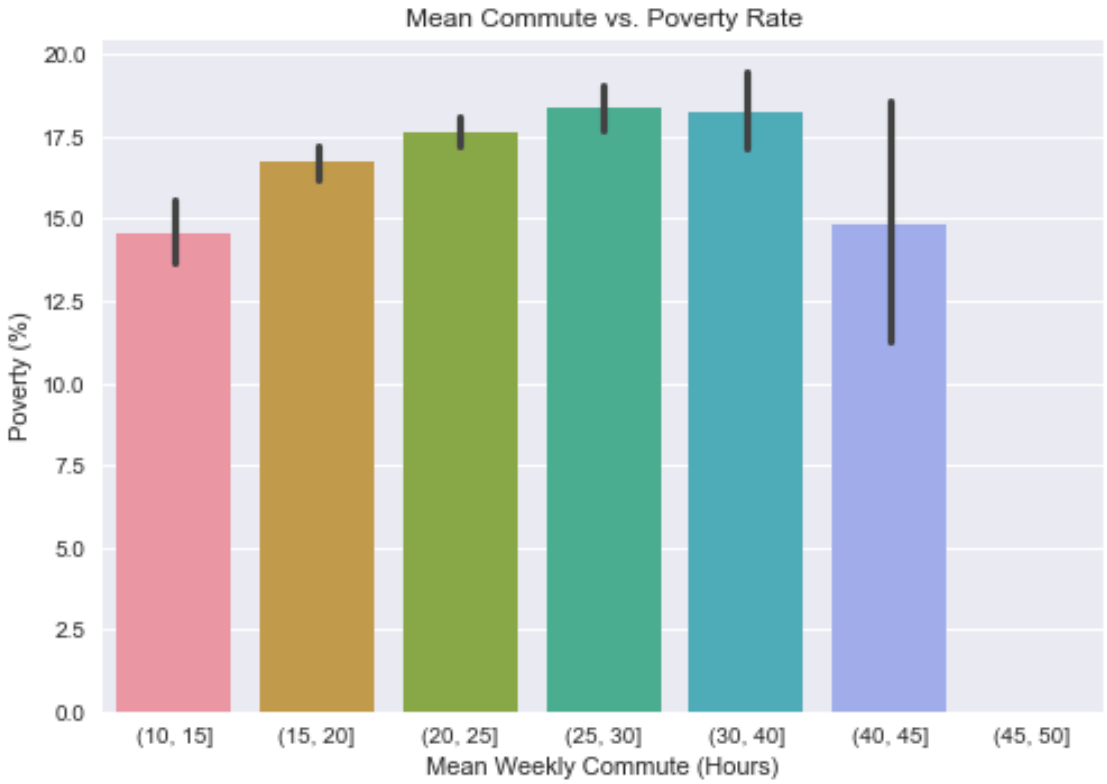

```
In [8]: black10 = pd.cut(census['Black'], [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
        unemployment = census['Unemployment']
        fig = sns.barplot(x=black10, y=unemployment)
        fig.set(xlabel='Black (%)', ylabel='Mean Unemployment (%)')
        fig.set_title('Percent Black vs. Unemployment')
        fig

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21b7ffd0>
```



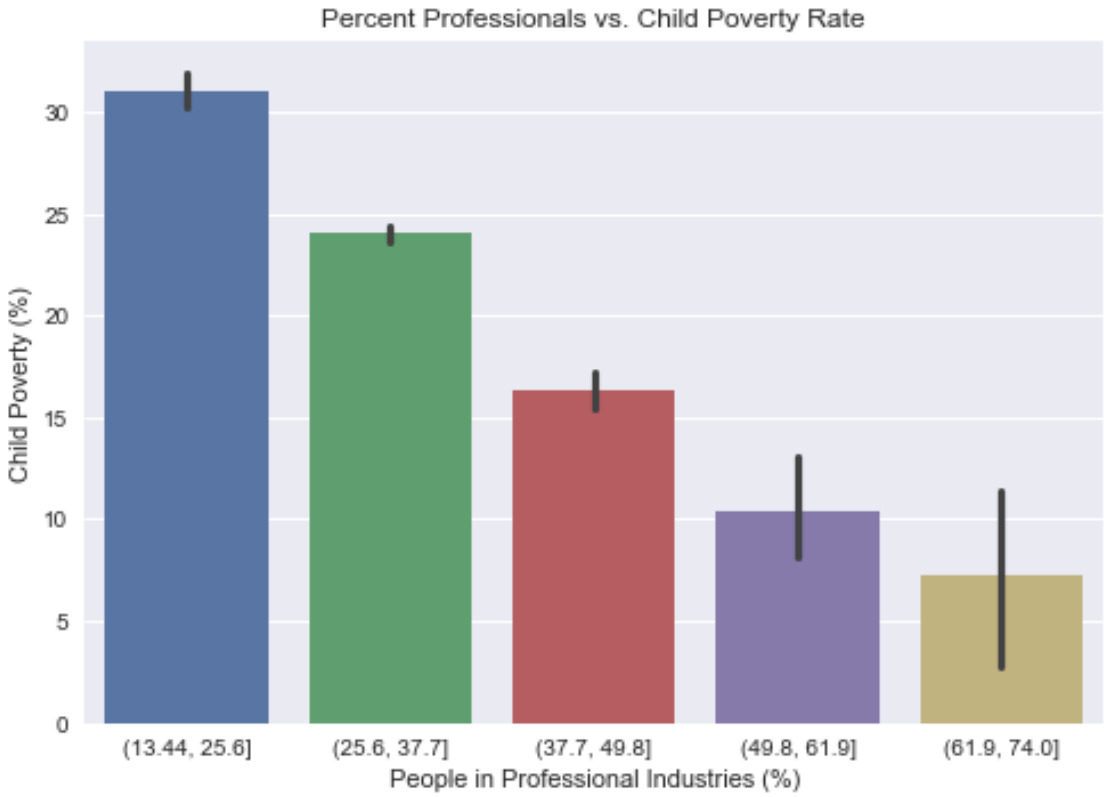

```
In [9]: mean_commute6 = pd.cut(census['MeanCommute'], [10, 15, 20, 25, 30, 40, 45, 50])
        child_poverty = census['ChildPoverty']
        fig = sns.barplot(x=mean_commute6, y=child_poverty)
        fig.set(xlabel='Mean Weekly Commute (Hours)', ylabel='Child Poverty (%)')
        fig.set_title('Mean Commute vs. Child Poverty Rate')
        fig

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21d9e630>
```




```
In [10]: mean_commute6 = pd.cut(census['MeanCommute'], [10, 15, 20, 25, 30, 40, 45, 50])
         poverty = census['Poverty']
         fig = sns.barplot(x=mean_commute6, y=poverty)
         fig.set(xlabel='Mean Weekly Commute (Hours)', ylabel='Poverty (%)')
         fig.set_title('Mean Commute vs. Poverty Rate')
         fig

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21f29cf8>
```

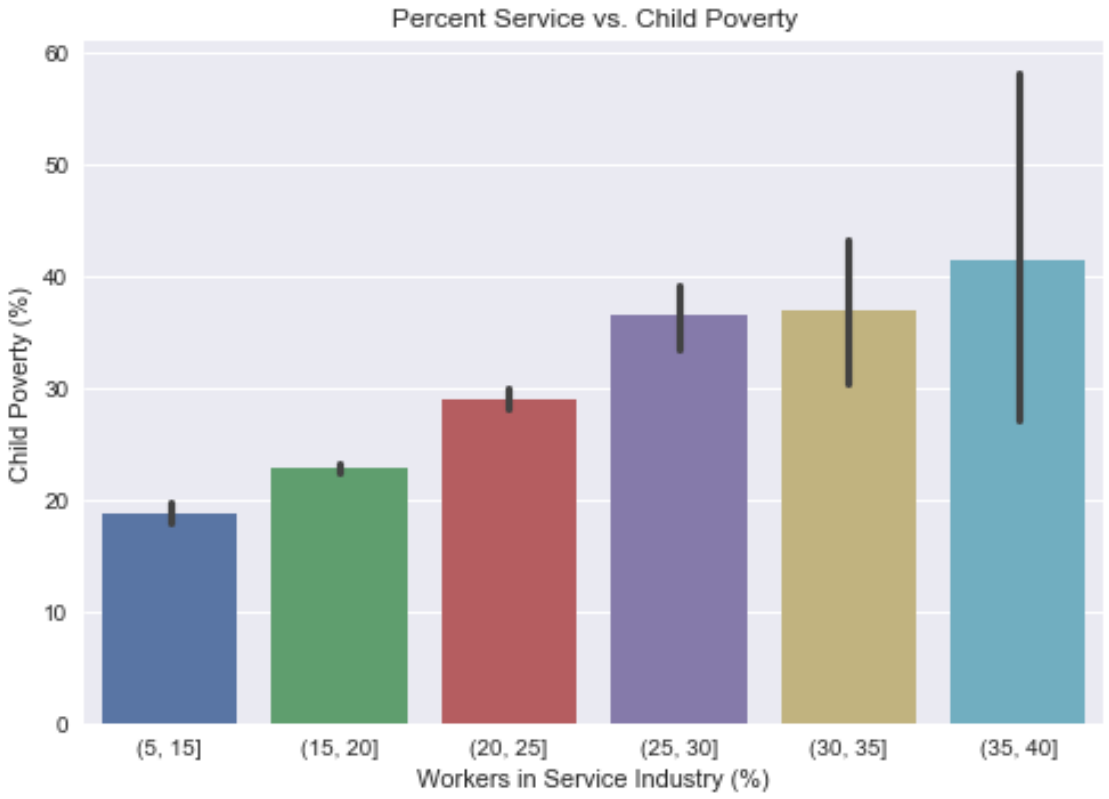



```
In [11]: professional5 = pd.cut(census['Professional'], 5)
         child_poverty = census['ChildPoverty']
         fig = sns.barplot(x=professional5, y=child_poverty)
         fig.set(xlabel='People in Professional Industries (%)', ylabel='Child Poverty (%)')
         fig.set_title('Percent Professionals vs. Child Poverty Rate')
         fig

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a22071e48>
```

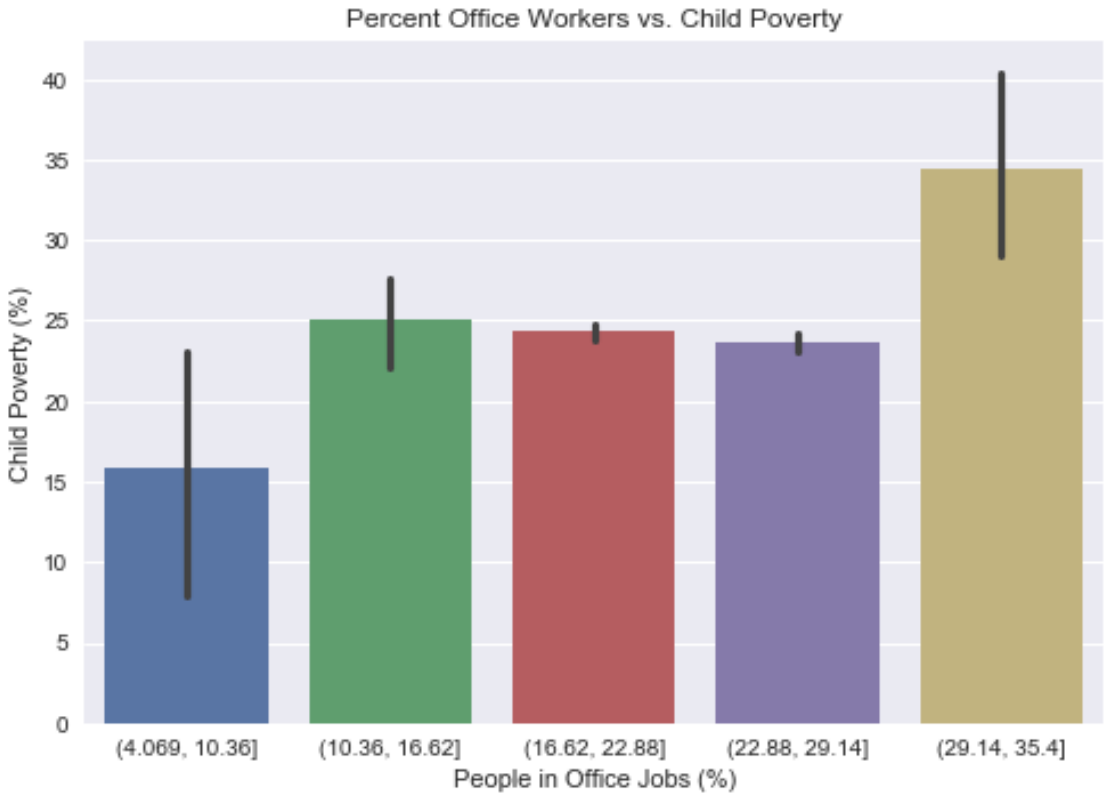



```
In [12]: professional6 = pd.cut(census['Service'], [5, 15, 20, 25, 30, 35, 40])
         child_poverty = census['ChildPoverty']
         fig = sns.barplot(x=professional6, y=child_poverty)
         fig.set(xlabel='Workers in Service Industry (%)', ylabel='Child Poverty (%)')
         fig.set_title('Percent Service vs. Child Poverty')
         fig
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x11a400080>
```



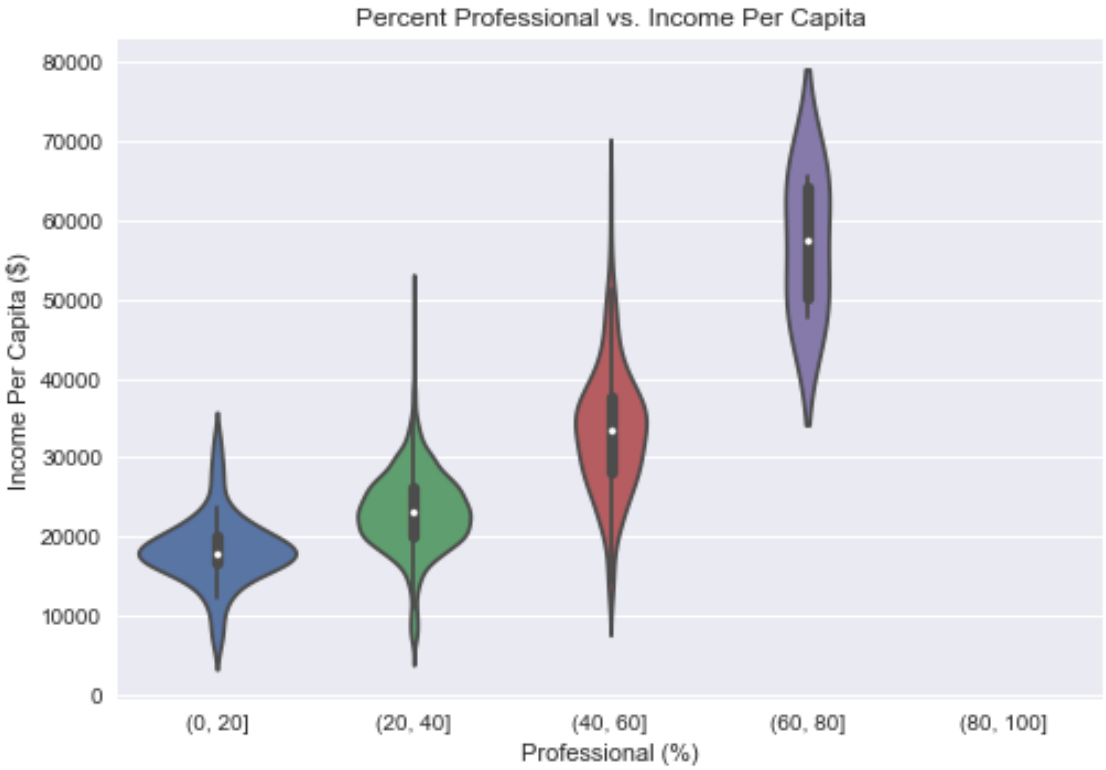

```
In [13]: office5 = pd.cut(census['Office'], 5)
         child_poverty = census['ChildPoverty']
         fig = sns.barplot(x=office5, y=child_poverty)
         fig.set(xlabel='People in Office Jobs (%)', ylabel='Child Poverty (%)')
         fig.set_title('Percent Office Workers vs. Child Poverty')
         fig

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a220c1eb8>
```




```
In [14]: census['ProfessionalRange'] = pd.cut(census['Professional'], [0, 20, 40, 60, 80, 100])
```

```
In [37]: fig = sns.violinplot(x='ProfessionalRange', y='IncomePerCap', data=census)
         fig.set(xlabel='Professional (%)', ylabel='Income Per Capita ($)')
         fig.set_title('Percent Professional vs. Income Per Capita')
         fig
```

```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1a231e6ef0>
```



In [18]:
```
#Mean Commute vs. Unemployment - Positive Correlation
#Mean Commute vs. Poverty - Positive Correlation
#Mean Commute vs. Income Per Cap - Negative Correlation
#Black vs. MeanCommute - Positive Correlation
#White vs. MeanCommute - Negative Correlation

state_abbrev_dict = {
        'AK': 'Alaska',
        'AL': 'Alabama',
        'AR': 'Arkansas',
        'AS': 'American Samoa',
        'AZ': 'Arizona',
        'CA': 'California',
        'CO': 'Colorado',
        'CT': 'Connecticut',
        'DC': 'District of Columbia',
        'DE': 'Delaware',
        'FL': 'Florida',
        'GA': 'Georgia',
        'GU': 'Guam',
        'HI': 'Hawaii',
        'IA': 'Iowa',
```

```
        'ID': 'Idaho',
        'IL': 'Illinois',
        'IN': 'Indiana',
        'KS': 'Kansas',
        'KY': 'Kentucky',
        'LA': 'Louisiana',
        'MA': 'Massachusetts',
        'MD': 'Maryland',
        'ME': 'Maine',
        'MI': 'Michigan',
        'MN': 'Minnesota',
        'MO': 'Missouri',
        'MP': 'Northern Mariana Islands',
        'MS': 'Mississippi',
        'MT': 'Montana',
        'NA': 'National',
        'NC': 'North Carolina',
        'ND': 'North Dakota',
        'NE': 'Nebraska',
        'NH': 'New Hampshire',
        'NJ': 'New Jersey',
        'NM': 'New Mexico',
        'NV': 'Nevada',
        'NY': 'New York',
        'OH': 'Ohio',
        'OK': 'Oklahoma',
        'OR': 'Oregon',
        'PA': 'Pennsylvania',
        'PR': 'Puerto Rico',
        'RI': 'Rhode Island',
        'SC': 'South Carolina',
        'SD': 'South Dakota',
        'TN': 'Tennessee',
        'TX': 'Texas',
        'UT': 'Utah',
        'VA': 'Virginia',
        'VI': 'Virgin Islands',
        'VT': 'Vermont',
        'WA': 'Washington',
        'WI': 'Wisconsin',
        'WV': 'West Virginia',
        'WY': 'Wyoming'
}

state_abbrev_dict = {state: abbrev for abbrev, state in state_abbrev_dict.items()}

census['StateAbbrev'] = census['State'].map(state_abbrev_dict.get)
census['Unemployed'] = census['TotalPop'] * census['Unemployment']
```

```
census_grouped = census.groupby('StateAbbrev', as_index=False)
census_by_state = census_grouped.sum()
census_by_state['StateUnemployment'] = census_by_state['Unemployed'] / census_by_state
census_by_state['StateAbbrev']
```

Out[18]:
```
0     AK
1     AL
2     AR
3     AZ
4     CA
5     CO
6     CT
7     DC
8     DE
9     FL
10    GA
11    HI
12    IA
13    ID
14    IL
15    IN
16    KS
17    KY
18    LA
19    MA
20    MD
21    ME
22    MI
23    MN
24    MO
25    MS
26    MT
27    NC
28    ND
29    NE
30    NH
31    NJ
32    NM
33    NV
34    NY
35    OH
36    OK
37    OR
38    PA
39    PR
40    RI
41    SC
42    SD
```

```
43    TN
44    TX
45    UT
46    VA
47    VT
48    WA
49    WI
50    WV
51    WY
Name: StateAbbrev, dtype: object
```

In [19]:
```
states = census_by_state['StateAbbrev']
unemployment = census_by_state['StateUnemployment'].tolist()

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)

data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = states,
        z = unemployment,
        locationmode = 'USA-states',
        marker = dict(
            line = dict (
                color = 'rgb(255,255,255)',
                width = 2
            ) ),
        colorbar = dict(
            title = "% Unemployed")
        ) ]

layout = dict(
        title = 'Unemployment Rate by State',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)'),
             )

fig = dict( data=data, layout=layout )
py.iplot( fig, filename='us-map' )
```

In [20]:
```
from sklearn.linear_model import LinearRegression, Ridge, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
```

```
from sklearn.preprocessing import normalize, PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

```
In [21]: census
```

Out[21]:

	CensusId	State	County	TotalPop	Men	Women
0	1001	Alabama	Autauga	55221	26745	28476
1	1003	Alabama	Baldwin	195121	95314	99807
2	1005	Alabama	Barbour	26932	14497	12435
3	1007	Alabama	Bibb	22604	12073	10531
4	1009	Alabama	Blount	57710	28512	29198
5	1011	Alabama	Bullock	10678	5660	5018
6	1013	Alabama	Butler	20354	9502	10852
7	1015	Alabama	Calhoun	116648	56274	60374
8	1017	Alabama	Chambers	34079	16258	17821
9	1019	Alabama	Cherokee	26008	12975	13033
10	1021	Alabama	Chilton	43819	21619	22200
11	1023	Alabama	Choctaw	13395	6382	7013
12	1025	Alabama	Clarke	25070	11834	13236
13	1027	Alabama	Clay	13537	6671	6866
14	1029	Alabama	Cleburne	15002	7334	7668
15	1031	Alabama	Coffee	50884	25174	25710
16	1033	Alabama	Colbert	54444	26303	28141
17	1035	Alabama	Conecuh	12865	6176	6689
18	1037	Alabama	Coosa	11027	5579	5448
19	1039	Alabama	Covington	37886	18339	19547
20	1041	Alabama	Crenshaw	13938	6863	7075
21	1043	Alabama	Cullman	80965	40081	40884
22	1045	Alabama	Dale	49866	24708	25158
23	1047	Alabama	Dallas	42154	19450	22704
24	1049	Alabama	DeKalb	71068	35474	35594
25	1051	Alabama	Elmore	80763	39362	41401
26	1053	Alabama	Escambia	37935	19524	18411
27	1055	Alabama	Etowah	103766	50207	53559
28	1057	Alabama	Fayette	16896	8477	8419
29	1059	Alabama	Franklin	31634	15311	16323
...	...	...	...	...	...	...
3190	72095	Puerto Rico	Maunabo	11701	5779	5922
3191	72097	Puerto Rico	Mayagüez	83418	39978	43440
3192	72099	Puerto Rico	Moca	38815	19039	19776
3193	72101	Puerto Rico	Morovis	32294	16067	16227
3194	72103	Puerto Rico	Naguabo	26804	12733	14071
3195	72105	Puerto Rico	Naranjito	29751	14700	15051
3196	72107	Puerto Rico	Orocovis	22595	11376	11219
3197	72109	Puerto Rico	Patillas	18441	9075	9366
3198	72111	Puerto Rico	Peñuelas	22770	11107	11663
3199	72113	Puerto Rico	Ponce	156054	75153	80901
3200	72115	Puerto Rico	Quebradillas	25205	12258	12947

3201	72117	Puerto Rico	Rincón	14841	7249	7592
3202	72119	Puerto Rico	Río Grande	53029	25845	27184
3203	72121	Puerto Rico	Sabana Grande	24307	11536	12771
3204	72123	Puerto Rico	Salinas	30114	14631	15483
3205	72125	Puerto Rico	San Germán	34125	16590	17535
3206	72127	Puerto Rico	San Juan	371400	170416	200984
3207	72129	Puerto Rico	San Lorenzo	39778	19485	20293
3208	72131	Puerto Rico	San Sebastián	40471	19723	20748
3209	72133	Puerto Rico	Santa Isabel	22913	11085	11828
3210	72135	Puerto Rico	Toa Alta	74603	36028	38575
3211	72137	Puerto Rico	Toa Baja	85242	40215	45027
3212	72139	Puerto Rico	Trujillo Alto	71886	34036	37850
3213	72141	Puerto Rico	Utuado	31474	15368	16106
3214	72143	Puerto Rico	Vega Alta	39319	18762	20557
3215	72145	Puerto Rico	Vega Baja	56858	27379	29479
3216	72147	Puerto Rico	Vieques	9130	4585	4545
3217	72149	Puerto Rico	Villalba	24685	12086	12599
3218	72151	Puerto Rico	Yabucoa	36279	17648	18631
3219	72153	Puerto Rico	Yauco	39474	19047	20427

	Hispanic	White	Black	Native	...	FamilyWork	Unemployment	\
0	2.6	75.8	18.5	0.4	...	0.0	7.6	
1	4.5	83.1	9.5	0.6	...	0.4	7.5	
2	4.6	46.2	46.7	0.2	...	0.1	17.6	
3	2.2	74.5	21.4	0.4	...	0.4	8.3	
4	8.6	87.9	1.5	0.3	...	0.4	7.7	
5	4.4	22.2	70.7	1.2	...	0.0	18.0	
6	1.2	53.3	43.8	0.1	...	0.2	10.9	
7	3.5	73.0	20.3	0.2	...	0.1	12.3	
8	0.4	57.3	40.3	0.2	...	0.0	8.9	
9	1.5	91.7	4.8	0.6	...	0.5	7.9	
10	7.6	80.5	10.2	0.4	...	0.5	9.1	
11	0.4	55.9	42.9	0.0	...	0.3	13.6	
12	0.3	53.4	45.3	0.0	...	0.0	19.4	
13	3.2	79.9	14.4	0.7	...	0.0	9.4	
14	2.3	92.5	2.9	0.2	...	0.0	8.3	
15	6.4	71.5	17.2	0.8	...	0.5	7.1	
16	2.4	78.9	15.6	0.6	...	0.0	9.0	
17	1.6	51.0	44.7	0.3	...	0.8	22.6	
18	2.1	65.2	30.7	0.1	...	0.0	17.0	
19	1.5	83.3	13.0	0.5	...	0.4	11.2	
20	1.7	70.5	23.5	0.8	...	0.3	9.7	
21	4.3	92.2	1.1	0.4	...	0.4	7.3	
22	6.0	69.8	19.4	0.5	...	0.1	10.9	
23	0.3	28.6	69.2	0.2	...	0.4	16.4	
24	14.0	80.9	1.8	1.1	...	0.1	7.7	
25	2.8	73.6	21.0	0.2	...	0.0	8.3	
26	1.2	60.7	33.4	3.2	...	0.1	15.6	

27	3.6	78.5	15.4	0.3	...	0.2	9.0
28	0.7	85.3	12.0	0.0	...	0.1	10.0
29	15.7	78.5	4.1	0.7	...	0.0	9.8
...	...	...	...	...	...	...	...
3190	99.9	0.1	0.0	0.0	...	0.0	27.9
3191	99.0	0.7	0.1	0.0	...	0.2	22.9
3192	99.2	0.4	0.0	0.0	...	0.0	23.0
3193	99.8	0.1	0.0	0.0	...	0.2	24.7
3194	99.7	0.1	0.0	0.0	...	0.0	10.9
3195	99.6	0.1	0.0	0.0	...	0.0	21.9
3196	99.9	0.1	0.0	0.0	...	0.2	31.2
3197	99.8	0.1	0.0	0.0	...	1.4	27.5
3198	99.3	0.7	0.0	0.0	...	0.0	26.1
3199	99.3	0.5	0.1	0.0	...	0.3	20.0
3200	99.2	0.5	0.2	0.0	...	0.0	16.1
3201	91.0	5.0	0.0	0.0	...	0.0	17.6
3202	99.3	0.6	0.1	0.0	...	0.4	23.9
3203	99.0	1.0	0.0	0.0	...	0.0	24.3
3204	99.3	0.3	0.1	0.0	...	0.0	12.1
3205	99.6	0.3	0.1	0.0	...	0.4	6.8
3206	98.2	1.2	0.2	0.0	...	0.2	15.8
3207	99.8	0.2	0.0	0.0	...	0.1	15.8
3208	99.3	0.6	0.0	0.0	...	0.9	29.4
3209	99.8	0.2	0.0	0.0	...	0.0	12.2
3210	99.4	0.3	0.1	0.0	...	0.3	15.7
3211	99.5	0.3	0.1	0.0	...	0.2	19.0
3212	99.6	0.4	0.0	0.0	...	0.0	7.9
3213	99.6	0.3	0.1	0.0	...	0.7	28.8
3214	98.6	1.1	0.0	0.0	...	0.0	21.7
3215	96.4	3.4	0.1	0.0	...	0.0	15.2
3216	96.7	2.9	0.0	0.0	...	0.3	12.2
3217	99.7	0.0	0.0	0.0	...	0.2	25.9
3218	99.8	0.2	0.0	0.0	...	0.0	24.3
3219	99.5	0.5	0.0	0.0	...	0.0	27.1

	PercentMen	PercentWomen	PercentCitizen	PercentNonCitizen	\
0	48.432661	51.567339	73.749117	26.250883	
1	48.848663	51.151337	75.694057	24.305943	
2	53.828160	46.171840	76.912223	23.087777	
3	53.410901	46.589099	77.397806	22.602194	
4	49.405649	50.594351	73.375498	26.624502	
5	53.006181	46.993819	75.454205	24.545795	
6	46.683699	53.316301	76.550064	23.449936	
7	48.242576	51.757424	75.965297	24.034703	
8	47.706799	52.293201	77.648992	22.351008	
9	49.888496	50.111504	79.206398	20.793602	
10	49.337046	50.662954	72.406947	27.593053	
11	47.644644	52.355356	78.895110	21.104890	

12	47.203829	52.796171	76.816913	23.183087
13	49.279752	50.720248	76.176405	23.823595
14	48.886815	51.113185	75.769897	24.230103
15	49.473312	50.526688	73.844430	26.155570
16	48.312027	51.687973	77.281243	22.718757
17	48.006218	51.993782	77.186164	22.813836
18	50.593997	49.406003	79.831323	20.168677
19	48.405744	51.594256	77.490366	22.509634
20	49.239489	50.760511	75.986512	24.013488
21	49.504107	50.495893	75.615389	24.384611
22	49.548791	50.451209	74.862632	25.137368
23	46.140343	53.859657	74.087868	25.912132
24	49.915574	50.084426	69.427872	30.572128
25	48.737665	51.262335	75.742605	24.257395
26	51.466983	48.533017	77.498352	22.501648
27	48.384827	51.615173	76.018156	23.981844
28	50.171638	49.828362	78.053977	21.946023
29	48.400455	51.599545	69.937409	30.062591
...	...	...	...	...
3190	49.388941	50.611059	77.779677	22.220323
3191	47.924908	52.075092	80.131387	19.868613
3192	49.050625	50.949375	75.669200	24.330800
3193	49.752276	50.247724	74.753824	25.246176
3194	47.504104	52.495896	74.201612	25.798388
3195	49.410104	50.589896	76.562132	23.437868
3196	50.347422	49.652578	75.184775	24.815225
3197	49.210997	50.789003	77.821159	22.178841
3198	48.779095	51.220905	73.987703	26.012297
3199	48.158330	51.841670	76.556833	23.443167
3200	48.633208	51.366792	76.695100	23.304900
3201	48.844417	51.155583	79.132134	20.867866
3202	48.737483	51.262517	76.431764	23.568236
3203	47.459580	52.540420	77.010738	22.989262
3204	48.585376	51.414624	74.809059	25.190941
3205	48.615385	51.384615	78.347253	21.652747
3206	45.884760	54.115240	72.682014	27.317986
3207	48.984363	51.015637	77.354316	22.645684
3208	48.733661	51.266339	77.107064	22.892936
3209	48.378650	51.621350	73.333915	26.666085
3210	48.292964	51.707036	74.570728	25.429272
3211	47.177448	52.822552	75.726754	24.273246
3212	47.347189	52.652811	75.551568	24.448432
3213	48.827604	51.172396	77.152570	22.847430
3214	47.717389	52.282611	75.205371	24.794629
3215	48.153294	51.846706	76.780752	23.219248
3216	50.219058	49.780942	77.601314	22.398686
3217	48.960907	51.039093	74.774154	25.225846
3218	48.645222	51.354778	76.970148	23.029852

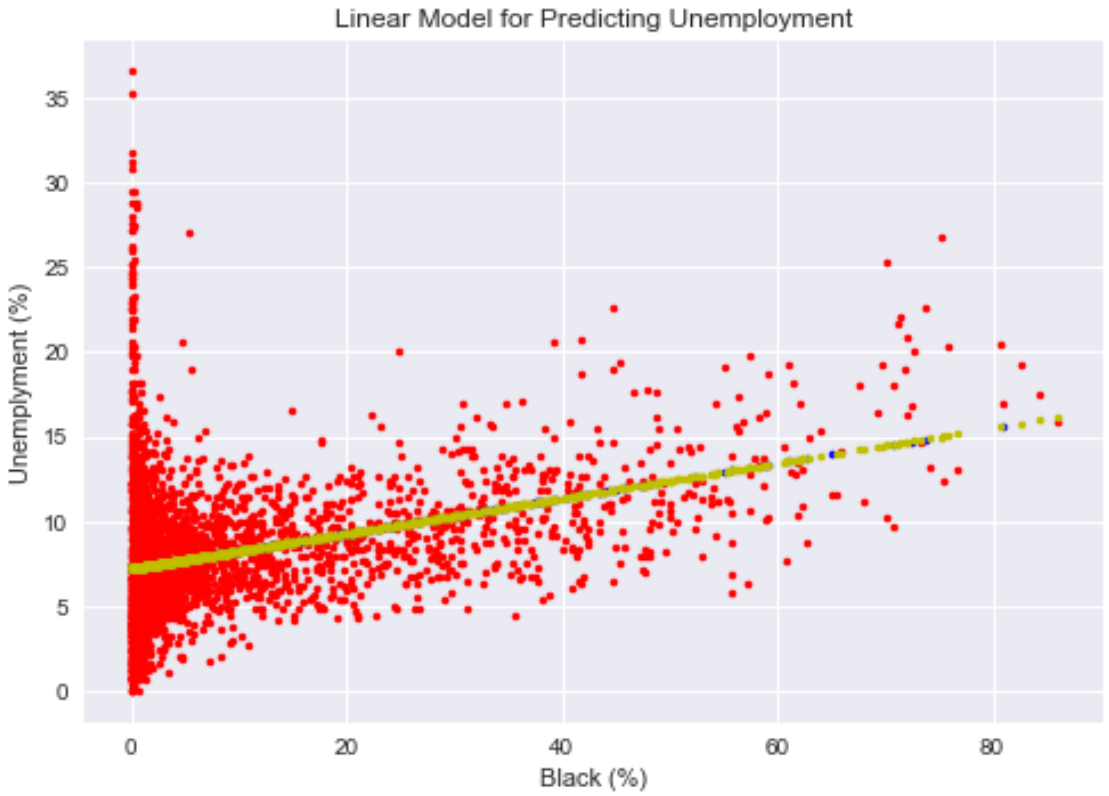
```
3219   48.252014     51.747986       77.673912          22.326088

      ProfessionalRange  PercentBlack  StateAbbrev  Unemployed
0              (20, 40]      (10, 20]           AL    419679.6
1              (20, 40]       (0, 10]           AL   1463407.5
2              (20, 40]      (40, 50]           AL    474003.2
3              (20, 40]      (20, 30]           AL    187613.2
4              (20, 40]       (0, 10]           AL    444367.0
5               (0, 20]      (70, 80]           AL    192204.0
6              (20, 40]      (40, 50]           AL    221858.6
7              (20, 40]      (20, 30]           AL   1434770.4
8              (20, 40]      (40, 50]           AL    303303.1
9              (20, 40]       (0, 10]           AL    205463.2
10             (20, 40]      (10, 20]           AL    398752.9
11             (20, 40]      (40, 50]           AL    182172.0
12             (20, 40]      (40, 50]           AL    486358.0
13             (20, 40]      (10, 20]           AL    127247.8
14             (20, 40]       (0, 10]           AL    124516.6
15             (20, 40]      (10, 20]           AL    361276.4
16             (20, 40]      (10, 20]           AL    489996.0
17              (0, 20]      (40, 50]           AL    290749.0
18              (0, 20]      (30, 40]           AL    187459.0
19             (20, 40]      (10, 20]           AL    424323.2
20             (20, 40]      (20, 30]           AL    135198.6
21             (20, 40]       (0, 10]           AL    591044.5
22             (20, 40]      (10, 20]           AL    543539.4
23             (20, 40]      (60, 70]           AL    691325.6
24             (20, 40]       (0, 10]           AL    547223.6
25             (20, 40]      (20, 30]           AL    670332.9
26             (20, 40]      (30, 40]           AL    591786.0
27             (20, 40]      (10, 20]           AL    933894.0
28             (20, 40]      (10, 20]           AL    168960.0
29             (20, 40]       (0, 10]           AL    310013.2
...                 ...           ...          ...         ...
3190           (20, 40]           NaN           PR    326457.9
3191           (20, 40]       (0, 10]           PR   1910272.2
3192           (20, 40]           NaN           PR    892745.0
3193           (20, 40]           NaN           PR    797661.8
3194           (20, 40]           NaN           PR    292163.6
3195           (20, 40]           NaN           PR    651546.9
3196           (20, 40]           NaN           PR    704964.0
3197           (20, 40]           NaN           PR    507127.5
3198           (20, 40]           NaN           PR    594297.0
3199           (20, 40]       (0, 10]           PR   3121080.0
3200           (20, 40]       (0, 10]           PR    405800.5
3201           (20, 40]           NaN           PR    261201.6
3202           (20, 40]       (0, 10]           PR   1267393.1
3203           (20, 40]           NaN           PR    590660.1
```

```
3204          (20, 40]       (0, 10]          PR    364379.4
3205          (20, 40]       (0, 10]          PR    232050.0
3206          (20, 40]       (0, 10]          PR   5868120.0
3207          (20, 40]           NaN          PR    628492.4
3208          (20, 40]           NaN          PR   1189847.4
3209          (20, 40]           NaN          PR    279538.6
3210          (20, 40]       (0, 10]          PR   1171267.1
3211          (20, 40]       (0, 10]          PR   1619598.0
3212          (20, 40]           NaN          PR    567899.4
3213          (20, 40]       (0, 10]          PR    906451.2
3214          (20, 40]           NaN          PR    853222.3
3215          (20, 40]       (0, 10]          PR    864241.6
3216           (0, 20]           NaN          PR    111386.0
3217          (20, 40]           NaN          PR    639341.5
3218          (20, 40]           NaN          PR    881579.7
3219          (20, 40]           NaN          PR   1069745.4

[3220 rows x 45 columns]
```

Linear Model for Percent Black to Unemployment

```
In [22]: train, test = train_test_split(census[['Black', 'Unemployment']], test_size=0.2)
         train_x1, train_y1 = train[[ 'Black']], train[['Unemployment']]
         test_x1, test_y1 = test[['Black']], test[['Unemployment']]
         # train_x1, train_y1 = normalize(train_x1), normalize(train_y1)
         # test_x1, test_y1 = normalize(test_x1), normalize(test_y1)

In [23]: l1 = LinearRegression()
         l1.fit(train_x1, train_y1)

Out[23]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [24]: y_train_pred = l1.predict(train_x1)
         y_test_pred = l1.predict(test_x1)
         print('train R^2: ',
               r2_score(train_y1, y_train_pred, multioutput = 'uniform_average'))
         print('test R^2: ',
               r2_score(test_y1, y_test_pred, multioutput = 'uniform_average'))
         # train_y1['IncomePerCap'].values.astype(float)
         # y_train_pred

train R^2:  0.128538703052
test R^2:  0.104783937126


In [25]: plt.plot(census['Black'], census['Unemployment'], 'r.', test_x1, y_test_pred, 'b.', t
         plt.xlabel('Black (%)')
         plt.ylabel('Unemplyment (%)')
         plt.title("Linear Model for Predicting Unemployment")
```

```
Out[25]: <matplotlib.text.Text at 0x1a22e0ca58>
```



Linear Model for Percent in Professional Industry to Child Poverty Rate

```
In [26]: l2_train, l2_test = train_test_split(census[['Professional', 'ChildPoverty']].dropna()
         l2_train_x, l2_train_y = l2_train[['Professional']], l2_train[['ChildPoverty']]
         l2_test_x, l2_test_y = l2_test[['Professional']], l2_test[['ChildPoverty']]

In [27]: # pcp = census[['Professional', 'ChildPoverty']]
         # pcp = pcp.dropna()
         # pcp_nan = pcp[pcp.isnull().any(axis=1)]
         # pcp_nan.head()

In [28]: l2 = LinearRegression()
         l2.fit(l2_train_x, l2_train_y)

Out[28]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [29]: l2_train_y_pred = l2.predict(l2_train_x)
         l2_test_y_pred = l2.predict(l2_test_x)
         print('train R^2: ',
               r2_score(l2_train_y, l2_train_y_pred, multioutput = 'uniform_average'))
         print('test R^2: ',
               r2_score(l2_test_y, l2_test_y_pred, multioutput = 'uniform_average'))
```

```
train R^2:  0.182940149457
test R^2:  0.151914701093
```

```
In [30]: plt.plot(census['Professional'], census['ChildPoverty'], 'r.', l2_test_x, l2_test_y_pr
         plt.xlabel('People in Professional Industries (%)')
         plt.ylabel('Child Poverty (%)')
         plt.title("Linear Model for Predicting Child Poverty")
```

```
Out[30]: <matplotlib.text.Text at 0x1a2303e668>
```

Model for % in Professional Work to Income Per Capita
(Professional work includes management, business, science, and the arts)

```
In [31]: l3_train, l3_test = train_test_split(census[['Professional', 'IncomePerCap']], test_si
         l3_train_x, l3_train_y = l3_train[['Professional']], l3_train[['IncomePerCap']]
         l3_test_x, l3_test_y = l3_test[['Professional']], l3_test[['IncomePerCap']]
```

```
In [32]: # l3_train_x.fillna(l3_train_x.mean())
         # l3_train_y.fillna(l3_train_y.mean())
         print(np.any(np.isnan(l3_train_x))) #if any value is nan
         print(np.all(np.isfinite(l3_train_x))) #if all values are finite
         np.where(np.isnan(l3_train_x))
```

```
False
True
```

Out[32]: (array([], dtype=int64), array([], dtype=int64))

In [33]:
```
# l3 = LinearRegression()
l3 = make_pipeline(PolynomialFeatures(2), Ridge())
l3.fit(l3_train_x, l3_train_y)
```

Out[33]:
```
Pipeline(memory=None,
     steps=[('polynomialfeatures', PolynomialFeatures(degree=2, include_bias=True, int
   normalize=False, random_state=None, solver='auto', tol=0.001))])
```

In [34]:
```
print(np.any(np.isnan(l3_test_x))) #if any value is nan
print(np.all(np.isfinite(l3_test_x))) #if all values are finite
np.where(np.isnan(l3_test_x))
```

```
False
True
```

Out[34]: (array([], dtype=int64), array([], dtype=int64))

In [35]:
```
l3_train_y_pred = l3.predict(l3_train_x)
l3_test_y_pred = l3.predict(l3_test_x)
print('train R^2: ',
      r2_score(l3_train_y, l3_train_y_pred))
print('test R^2: ',
      r2_score(l3_test_y, l3_test_y_pred))
```

```
train R^2:  0.448574220854
test R^2:  0.405784260144
```

In [36]:
```
# a = plt.plot(l3_train_x, l3_train_y, 'r.', l3_train_x, l3_train_y_pred, 'b.',
#              l3_test_x, l3_test_y_pred, 'y.')
data = plt.plot(census['Professional'], census['IncomePerCap'], 'r.', label='All data
train_3 = plt.plot(l3_train_x, l3_train_y_pred, 'b.', label='train predictions')
test_3 = plt.plot(l3_test_x, l3_test_y_pred, 'y.', label='test predictions')
plt.xlabel("% in Professional")
plt.ylabel("Income Per Capita")
plt.title("Visualization of Test, Training, and all Data")
plt.legend()
```

Out[36]: <matplotlib.legend.Legend at 0x1a231de438>


Visualization of Test, Training, and all Data
All data
train predictions
test predictions
Income Per Capita
% in Professional
60000
50000
40000
30000
20000
10000
20
30
40
50
60
70

In [ ]: