

```
In [1]: import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import numpy.random as nr
import math
from sklearn import preprocessing
import sklearn.model_selection as ms
from sklearn import linear_model
import sklearn.metrics as sklm

%matplotlib inline
```

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

Out[2]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	NaN	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0
1561	1561	38ccdfd	1dcfd4e	650813.0	92680.0	37.789	NaN	30.687	0.704111	0

1562 rows × 45 columns



```
In [3]: test_values = pd.read_csv('test_values.csv')
test_values
```

0	0	8e7613e	9d1e27d	52842.0	5403.0	26.840	NaN	27.960	0.924234
1	1	694a5e8	a952566	212287.0	53502.0	57.534	6032.0	33.072	0.398318
2	2	b3f0726	20d32fc	81263.0	13368.0	39.994	1012.0	32.044	0.483789
3	3	fd922f0	1b0d913	122870.0	19359.0	41.865	NaN	30.724	0.468043
4	4	3bd551e	698ab34	146153.0	15766.0	30.681	644.0	30.860	0.651511
...
1571	1571	dae1584	52acab4	44414.0	6275.0	35.026	254.0	30.569	0.726121
1572	1572	b95e0ba	485e9af	16160.0	1871.0	31.922	NaN	33.899	0.663196
1573	1573	f44348c	1dcfd4e	28688.0	4478.0	33.897	NaN	28.644	0.809853
1574	1574	c9450a6	52acab4	29234.0	4528.0	40.479	57.0	24.803	0.707838
1575	1575	c229ad2	78e8330	4036.0	327.0	18.052	NaN	22.122	0.967377

In [4]:

```
for col in train_values:  
    train_values[col].describe()  
    print(col,train_values[col].describe(),'\n')
```

```
row_id count 1562.000000
```

```
mean 780.500000
```

```
std 451.054875
```

```
min 0.000000
```

```
25% 390.250000
```

```
50% 780.500000
```

```
75% 1170.750000
```

```
max 1561.000000
```

```
Name: row_id, dtype: float64
```

```
county_code count 1562
```

```
unique 1562
```

```
top 2d693e1
```

```
freq 1
```

```
Name: county_code, dtype: object
```

```
state count 1562
```

```
unique 50
```

```
top 1b0d913
```

```
freq 121
```

```
In [5]: def count_unique(train_values, cols):
    for col in cols:
        print('\n' + 'For column ' + col)
        print(train_values[col].value_counts())
    cat_cols = ['rucc', 'urban_influence', 'economic_typerology']
    count_unique(train_values, cat_cols)
```

For column rucc

Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area	301
Metro - Counties in metro areas of 1 million population or more	219
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area	215
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area	206
Metro - Counties in metro areas of 250,000 to 1 million population	193
Metro - Counties in metro areas of fewer than 250,000 population	165
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area	120
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area	101
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area	42
Name: rucc, dtype: int64	

For column urban_influence

Small-in a metro area with fewer than 1 million residents	358
Large-in a metro area with at least 1 million residents or more	219
Noncore adjacent to a small metro with town of at least 2,500 residents	178
Micropolitan not adjacent to a metro area	131
Micropolitan adjacent to a small metro area	113
Noncore adjacent to micro area and does not contain a town of at least 2,500 residents	97
Noncore adjacent to a small metro and does not contain a town of at least 2,500 residents	88
Noncore not adjacent to a metro/micro area and does not contain a town of at least 2,500 residents	86
Noncore adjacent to micro area and contains a town of 2,500-19,999 residents	86
Noncore adjacent to a large metro area	79
Micropolitan adjacent to a large metro area	64
Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents	63
Name: urban_influence, dtype: int64	

For column economic_typerology

Nonspecialized	631
Manufacturing-dependent	244
Farm-dependent	217
Federal/State government-dependent	191
Recreation	166

Mining-dependent
Name: economic_tpyology, dtype: int64

In [6]: `train_labels = pd.read_csv('train_labels.csv')`
`train_labels`

Out[6]:

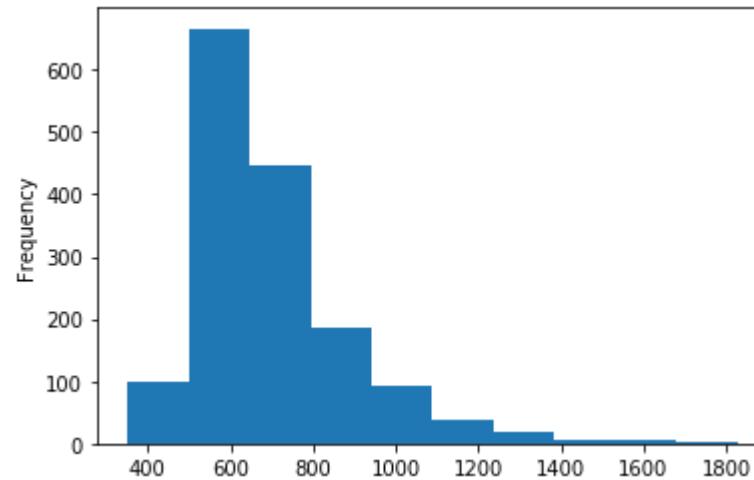
	row_id	gross_rent
0	0	577
1	1	844
2	2	700
3	3	592
4	4	444
...
1557	1557	928
1558	1558	640
1559	1559	754
1560	1560	640
1561	1561	976

1562 rows × 2 columns

```
In [7]: import matplotlib.pyplot as plt  
%matplotlib inline
```

```
train_labels['gross_rent'].plot.hist()
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x243f5126048>
```



```
In [8]: train_labels['gross_rent'].describe()
```

```
Out[8]: count    1562.000000  
mean     701.142125  
std      192.883110  
min      351.000000  
25%     578.000000  
50%     650.000000  
75%     773.750000  
max     1827.000000  
Name: gross_rent, dtype: float64
```

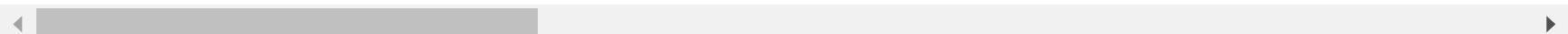
In [9]:

```
train_values_labels = pd.merge(train_values, train_labels, on='row_id')
train_values_labels
```

Out[9]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	NaN	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0
1561	1561	38ccdfd	1dcfd4e	650813.0	92680.0	37.789	NaN	30.687	0.704111	0

1562 rows × 46 columns



In [10]:

```
train_values_labels['gross_rent'].describe()
```

Out[10]:

count	1562.000000
mean	701.142125
std	192.883110
min	351.000000
25%	578.000000
50%	650.000000
75%	773.750000
max	1827.000000
Name:	gross_rent, dtype: float64

```
In [11]: train_values_labels['gross_rent'].mean()
```

```
Out[11]: 701.1421254801536
```

```
In [12]: train_values_labels['gross_rent'].median()
```

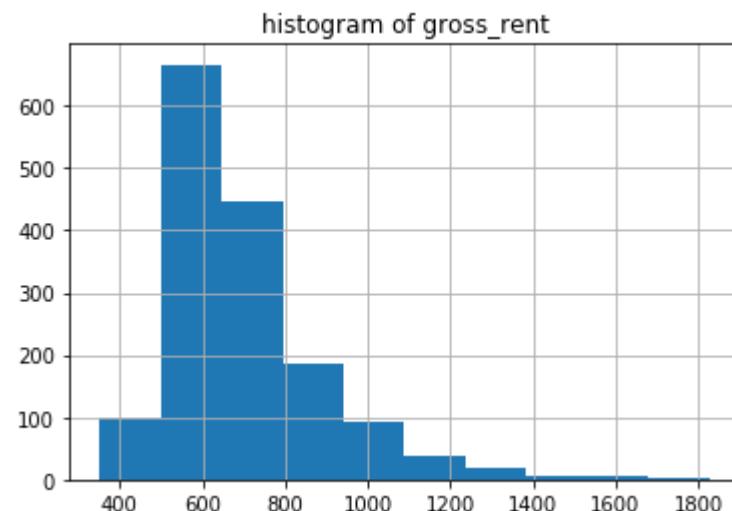
```
Out[12]: 650.0
```

```
In [13]: train_values_labels['gross_rent'].std()
```

```
Out[13]: 192.88310953787266
```

```
In [14]: train_values_labels['gross_rent'].hist()  
plt.title('histogram of gross_rent')
```

```
Out[14]: Text(0.5, 1.0, 'histogram of gross_rent')
```



```
In [15]: num_cols = ['gross_rent','population',
    'renter_occupied_households', 'pct_renter_occupied', 'evictions',
    'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
    'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
    'pct_civilian_labor',
    'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
    'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
    'pct_low_birthweight', 'pct_excessive_drinking',
    'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
    'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
    'heart_disease_mortality_per_100k', 'pop_per_dentist',
    'pop_per_primary_care_physician', 'pct_female',
    'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
    'pct_adults_less_than_a_high_school_diploma',
    'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
    'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
    'death_rate_per_1k']
```

```
In [16]: Correlations = train_values_labels[num_cols]

corr = Correlations.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[16]:

	gross_rent	population	renter_occupied_households	pct_renter_occupied	evictions	re
gross_rent	1	0.398343	0.345154	0.278613	0.325653	
population	0.398343	1	0.979015	0.285874	0.800173	
renter_occupied_households	0.345154	0.979015	1	0.320175	0.790723	
pct_renter_occupied	0.278613	0.285874	0.320175	1	0.357836	
evictions	0.325653	0.800173	0.790723	0.357836	1	
rent_burden	0.237213	0.162747	0.146213	0.229203	0.147673	
pct_white	-0.226985	-0.233149	-0.227568	-0.472168	-0.286329	
pct_af_am	0.0161472	0.065498	0.0760577	0.283012	0.187811	
pct_hispanic	0.170457	0.194042	0.17686	0.211543	0.146569	
pct_am_ind	-0.0179417	-0.0476048	-0.0403485	0.140909	-0.0365775	
count	131.000000	0.45000000	0.105010	0.107700	0.000115	

```
In [17]: state_one = train_values_labels[train_values_labels['state']=='1b0d913']
```

```
In [18]: state_two = train_values_labels[train_values_labels['state']=='a952566']
```

```
In [19]: state_one['gross_rent'].describe()
```

```
Out[19]: count    131.000000
mean     691.229008
std      123.938419
min      433.000000
25%      617.500000
50%      675.000000
75%      759.500000
max     1002.000000
Name: gross_rent, dtype: float64
```

```
In [20]: state_one['gross_rent'].mean()
```

```
Out[20]: 691.2290076335878
```

```
In [21]: state_two['gross_rent'].describe()
```

```
Out[21]: count      69.000000
mean      894.710145
std       315.577440
min       473.000000
25%       680.000000
50%       861.000000
75%       1031.000000
max       1827.000000
Name: gross_rent, dtype: float64
```

```
In [22]: state_two['gross_rent'].mean()
```

```
Out[22]: 894.7101449275362
```

```
In [23]: train_values_labels['state'].unique()
```

```
Out[23]: array(['fb8cab1', '842bd12', '2b7da97', '5029ed4', 'b795815', '4522abc',
   '4c72956', '78e8330', '9e0007d', 'e74aca3', '1b0d913', '485e9af',
   '4cd9667', '1646cf6', '20d32fc', 'a952566', '8036085', '9dda412',
   '5086a32', 'e899d7f', '158df01', '6d287d7', '08f8fb4', '1dcfd4e',
   '9d0874a', '528ea9f', '7dd3518', 'e2f94fa', 'c479f0c', 'dc9ae72',
   '698ab34', '176f5f0', '0f8930b', '52acab4', '9d1e27d', 'c3dbf0a',
   'b44cfe6', '7572db1', 'bc77872', 'dfc21f3', '3745933', '64ffe5d',
   '09d8cd0', '105e445', 'fa605d5', '9e065a4', 'd233cec', '3337bbb',
   '375d4d3', '914c15f'], dtype=object)
```

```
In [24]: states=['fb8cab1', '842bd12', '2b7da97', '5029ed4', 'b795815', '4522abc',  
    '4c72956', '78e8330', '9e0007d', 'e74aca3', '1b0d913', '485e9af',  
    '4cd9667', '1646cf6', '20d32fc', 'a952566', '8036085', '9dda412',  
    '5086a32', 'e899d7f', '158df01', '6d287d7', '08f8fb4', '1dcfd4e',  
    '9d0874a', '528ea9f', '7dd3518', 'e2f94fa', 'c479f0c', 'dc9ae72',  
    '698ab34', '176f5f0', '0f8930b', '52acab4', '9d1e27d', 'c3dbf0a',  
    'b44cfe6', '7572db1', 'bc77872', 'dfc21f3', '3745933', '64ffe5d',  
    '09d8cd0', '105e445', 'fa605d5', '9e065a4', 'd233cec', '3337bbb',  
    '375d4d3', '914c15f']
```

```
In [25]: nums = list(range(0, 50))  
  
for i in nums:  
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]  
    county_codes_state = specific_state[['county_code', 'gross_rent']]  
    print(county_codes_state.describe(), '\n' )
```

```
gross_rent  
count    30.000000  
mean    577.900000  
std     125.367969  
min     351.000000  
25%    488.750000  
50%    562.500000  
75%    663.500000  
max    898.000000
```

```
gross_rent  
count    26.000000  
mean    663.653846  
std     91.172778  
min     528.000000  
25%    592.250000  
50%    652.000000  
75%    717.750000  
max    844.000000
```

```
In [26]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','gross_rent']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	gross_rent
0	fb8cab1	8e686a7	577
11	fb8cab1	170b0b3	406
65	fb8cab1	43b66dd	582
94	fb8cab1	555fe97	467
98	fb8cab1	118ca99	558
153	fb8cab1	3b9505f	481
277	fb8cab1	a6f2901	548
291	fb8cab1	e46e2df	650
364	fb8cab1	b301c5b	898
399	fb8cab1	b66aeb7	512
456	fb8cab1	62c7e21	515
458	fb8cab1	d27aeaf	443
531	fb8cab1	5a9c263	351
604	fb8cab1	6074b91	673
650	fb8cab1	978d233	514
787	fb8cab1	f62fc23	442
792	fb8cab1	26886b3	565
923	fb8cab1	62cb873	560
977	fb8cab1	5504177	510

```
In [27]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state.describe(), '\n' )
```

```
          homicides_per_100k
count              0.0
mean                NaN
std                 NaN
min                 NaN
25%                NaN
50%                NaN
75%                NaN
max                 NaN
```

```
          homicides_per_100k
count      3.000000
mean      1.593333
std       0.483356
min       1.280000
25%      1.315000
50%      1.350000
75%      1.750000
max      2.150000
```

```
In [28]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	homicides_per_100k
0	fb8cab1	8e686a7	NaN
11	fb8cab1	170b0b3	NaN
65	fb8cab1	43b66dd	NaN
94	fb8cab1	555fe97	NaN
98	fb8cab1	118ca99	NaN
153	fb8cab1	3b9505f	NaN
277	fb8cab1	a6f2901	NaN
291	fb8cab1	e46e2df	NaN
364	fb8cab1	b301c5b	NaN
399	fb8cab1	b66aeb7	NaN
456	fb8cab1	62c7e21	NaN
458	fb8cab1	d27aeaf	NaN
531	fb8cab1	5a9c263	NaN
604	fb8cab1	6074b91	NaN
650	fb8cab1	978d233	NaN
787	fb8cab1	f62fc23	NaN
792	fb8cab1	26886b3	NaN
923	fb8cab1	62cb873	NaN
~	fb8cab1	5ec0172	~

```
In [29]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_excessive_drinking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_excessive_drinking
0	fb8cab1	8e686a7	0.262
11	fb8cab1	170b0b3	0.179
65	fb8cab1	43b66dd	0.205
94	fb8cab1	555fe97	0.233
98	fb8cab1	118ca99	0.202
153	fb8cab1	3b9505f	0.167
277	fb8cab1	a6f2901	0.265
291	fb8cab1	e46e2df	0.238
364	fb8cab1	b301c5b	0.272
399	fb8cab1	b66aeb7	0.264
456	fb8cab1	62c7e21	0.235
458	fb8cab1	d27aeaf	0.237
531	fb8cab1	5a9c263	0.282
604	fb8cab1	6074b91	0.191
650	fb8cab1	978d233	0.232
787	fb8cab1	f62fc23	0.361
792	fb8cab1	26886b3	0.152
923	fb8cab1	62cb873	0.160
~	fb8cab1	5ec0172	~ ~12

```
In [30]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_adult_smoking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_adult_smoking
0	fb8cab1	8e686a7	0.166
11	fb8cab1	170b0b3	0.157
65	fb8cab1	43b66dd	0.110
94	fb8cab1	555fe97	0.177
98	fb8cab1	118ca99	0.185
153	fb8cab1	3b9505f	0.112
277	fb8cab1	a6f2901	0.166
291	fb8cab1	e46e2df	0.246
364	fb8cab1	b301c5b	0.300
399	fb8cab1	b66aeb7	0.189
456	fb8cab1	62c7e21	0.175
458	fb8cab1	d27aeaf	0.153
531	fb8cab1	5a9c263	0.361
604	fb8cab1	6074b91	0.238
650	fb8cab1	978d233	0.155
787	fb8cab1	f62fc23	NaN
792	fb8cab1	26886b3	0.083
923	fb8cab1	62cb873	0.133
~--	fb8cab1	fb8cab1	~ 161

```
In [31]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_adult_obesity']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_adult_obesity
0	fb8cab1	8e686a7	0.310
11	fb8cab1	170b0b3	0.303
65	fb8cab1	43b66dd	0.317
94	fb8cab1	555fe97	0.306
98	fb8cab1	118ca99	0.303
153	fb8cab1	3b9505f	0.329
277	fb8cab1	a6f2901	0.298
291	fb8cab1	e46e2df	0.304
364	fb8cab1	b301c5b	0.306
399	fb8cab1	b66aeb7	0.304
456	fb8cab1	62c7e21	0.329
458	fb8cab1	d27aeaf	0.288
531	fb8cab1	5a9c263	0.403
604	fb8cab1	6074b91	0.354
650	fb8cab1	978d233	0.331
787	fb8cab1	f62fc23	0.359
792	fb8cab1	26886b3	0.264
923	fb8cab1	62cb873	0.355
~	fb8cab1	fb8cab1	~ ~

```
In [32]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','evictions']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	evictions
0	fb8cab1	8e686a7	NaN
11	fb8cab1	170b0b3	NaN
65	fb8cab1	43b66dd	NaN
94	fb8cab1	555fe97	NaN
98	fb8cab1	118ca99	NaN
153	fb8cab1	3b9505f	NaN
277	fb8cab1	a6f2901	NaN
291	fb8cab1	e46e2df	NaN
364	fb8cab1	b301c5b	NaN
399	fb8cab1	b66aeb7	NaN
456	fb8cab1	62c7e21	NaN
458	fb8cab1	d27aeaf	NaN
531	fb8cab1	5a9c263	NaN
604	fb8cab1	6074b91	NaN
650	fb8cab1	978d233	NaN
787	fb8cab1	f62fc23	NaN
792	fb8cab1	26886b3	NaN
923	fb8cab1	62cb873	NaN
~7~	fb8cab1	5ec0177	~ ~

```
In [33]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state', 'county_code', 'pct_low_birthweight']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_low_birthweight
0	fb8cab1	8e686a7	NaN
11	fb8cab1	170b0b3	NaN
65	fb8cab1	43b66dd	NaN
94	fb8cab1	555fe97	NaN
98	fb8cab1	118ca99	NaN
153	fb8cab1	3b9505f	NaN
277	fb8cab1	a6f2901	0.050
291	fb8cab1	e46e2df	NaN
364	fb8cab1	b301c5b	0.060
399	fb8cab1	b66aeb7	NaN
456	fb8cab1	62c7e21	0.079
458	fb8cab1	d27aeaf	NaN
531	fb8cab1	5a9c263	0.091
604	fb8cab1	6074b91	0.081
650	fb8cab1	978d233	0.060
787	fb8cab1	f62fc23	NaN
792	fb8cab1	26886b3	NaN
923	fb8cab1	62cb873	0.050
~	fb8cab1	fb8cab1	~ 0.071

```
In [34]: null_value_one = train_values_labels[train_values_labels['row_id']==231]
null_value_one
```

Out[34]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_	
231	231	34d700b	7dd3518	6360.0		693.0	28.388	NaN	23.09	0.460225	0.0

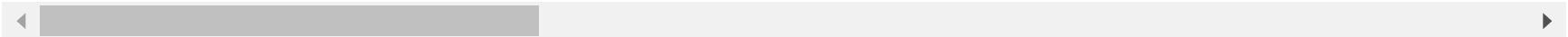
1 rows × 46 columns

```
In [35]: null_value_two = train_values_labels[train_values_labels['row_id']==1478]  
null_value_two
```

Out[35]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
	1478	1478	ece4b49	7dd3518	2118.0	299.0	33.943	NaN	22.104	0.495419 0

1 rows × 46 columns



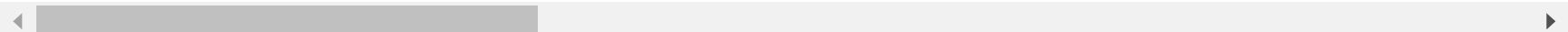
In [36]:

```
state_median = train_values_labels[train_values_labels['state']=='7dd3518']
state_median
```

Out[36]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
35	35	bc192e7	7dd3518	3302.0	276.0	46.884	NaN	22.064	0.149208	0
182	182	e0370f9	7dd3518	14005.0	2104.0	40.190	NaN	29.683	0.516275	0
190	190	072036b	7dd3518	5694.0	840.0	68.336	NaN	19.237	0.278452	0
231	231	34d700b	7dd3518	6360.0	693.0	28.388	NaN	23.090	0.460225	0
243	243	e840e47	7dd3518	961.0	207.0	45.837	NaN	21.259	0.532234	0
331	331	8282823	7dd3518	100623.0	16090.0	42.227	NaN	30.838	0.719742	0
372	372	a48f4a1	7dd3518	13709.0	2327.0	41.394	NaN	28.484	0.649735	0
481	481	f2607e5	7dd3518	2566.0	336.0	30.712	NaN	23.033	0.793519	0
825	825	2d693e1	7dd3518	9689.0	1372.0	29.342	NaN	19.720	0.701697	0
849	849	6afba47	7dd3518	9811.0	1338.0	43.685	NaN	25.904	0.164267	0
891	891	78296f7	7dd3518	6981.0	857.0	29.689	NaN	24.663	0.766605	0
897	897	d465f03	7dd3518	8901.0	1604.0	41.129	NaN	27.644	0.627646	0
1009	1009	3900385	7dd3518	5684.0	653.0	29.133	NaN	22.985	0.217068	0
1196	1196	3fa9943	7dd3518	9575.0	1117.0	46.110	NaN	17.865	0.321428	0
1266	1266	76ff4a9	7dd3518	7761.0	980.0	44.728	NaN	18.372	0.118455	0
1389	1389	9b53095	7dd3518	17676.0	2048.0	35.446	NaN	24.938	0.113347	0
1470	1470	86fe9c7	7dd3518	643.0	120.0	45.318	NaN	25.048	0.396316	0
1478	1478	ece4b49	7dd3518	2118.0	299.0	33.943	NaN	22.104	0.495419	0

18 rows × 46 columns



In [37]:

```
#train_values_labels.drop(null_value_one.index, inplace = True)
```

```
In [38]: #train_values_labels.drop(null_value_two.index, inplace = True)
```

```
In [39]: train_values_labels.columns
```

```
Out[39]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent'],
      dtype='object')
```

In [40]: `test_values.columns`

Out[40]: `Index(['row_id', 'county_code', 'state', 'population',
 'renter_occupied_households', 'pct_renter_occupied', 'evictions',
 'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
 'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
 'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
 'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
 'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
 'pct_low_birthweight', 'pct_excessive_drinking',
 'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
 'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k', 'pop_per_dentist',
 'pop_per_primary_care_physician', 'pct_female',
 'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
 'death_rate_per_1k'],
 dtype='object')`

```
In [41]: train_values_labels.isnull().any()
```

```
Out[41]: row_id                         False  
county_code                      False  
state                            False  
population                        False  
renter_occupied_households      False  
pct_renter_occupied                False  
evictions                          True  
rent_burden                        False  
pct_white                           False  
pct_af_am                            False  
pct_hispanic                         False  
pct_am_ind                            False  
pct_asian                            False  
pct_nh_pi                            False  
pct_multiple                         False  
pct_other                            False  
poverty_rate                         False  
rucc                                False  
urban_influence                      False  
economic_typerology                  False  
pct_civilian_labor                  False  
pct_unemployment                     False  
pct_uninsured_adults                 True  
pct_uninsured_children                True  
pct_adult_obesity                    True  
pct_adult_smoking                     True  
pct_diabetes                          True  
pct_low_birthweight                   True  
pct_excessive_drinking                True  
pct_physical_inactivity                True  
air_pollution_particulate_matter_value True  
homicides_per_100k                     True  
motor_vehicle_crash_deaths_per_100k    True  
heart_disease_mortality_per_100k       False  
pop_per_dentist                       True  
pop_per_primary_care_physician        True  
pct_female                            True  
pct_below_18_years_of_age              True  
pct_aged_65_years_and_older            True  
pct_adults_less_than_a_high_school_diploma False
```

```
pct_adults_with_high_school_diploma      False
pct_adults_with_some_college              False
pct_adults_bachelors_or_higher           False
birth_rate_per_1k                        False
death_rate_per_1k                        False
gross_rent                               False
dtype: bool
```

In [42]: `test_values.isnull().any()`

Out[42]:

row_id	False
county_code	False
state	False
population	False
renter_occupied_households	False
pct_renter_occupied	False
evictions	True
rent_burden	False
pct_white	False
pct_af_am	False
pct_hispanic	False
pct_am_ind	False
pct_asian	False
pct_nh_pi	False
pct_multiple	False
pct_other	False
poverty_rate	False
rucc	False
urban_influence	False
economic_typerology	False
pct_civilian_labor	False
pct_unemployment	False
pct_uninsured_adults	True
pct_uninsured_children	True
pct_adult_obesity	True
pct_adult_smoking	True
pct_diabetes	True
pct_low_birthweight	True
pct_excessive_drinking	True
pct_physical_inactivity	True
air_pollution_particulate_matter_value	True
homicides_per_100k	True
motor_vehicle_crash_deaths_per_100k	True
heart_disease_mortality_per_100k	False
pop_per_dentist	True
pop_per_primary_care_physician	True
pct_female	True
pct_below_18_years_of_age	True
pct_aged_65_years_and_older	True
pct_adults_less_than_a_high_school_diploma	False

```
pct_adults_with_high_school_diploma      False
pct_adults_with_some_college            False
pct_adults_bachelors_or_higher          False
birth_rate_per_1k                      False
death_rate_per_1k                      False
dtype: bool
```

```
In [43]: train_values_labels.homicides_per_100k.fillna(0, inplace=True)
train_values_labels.motor_vehicle_crash_deaths_per_100k.fillna(0, inplace=True)
train_values_labels[['homicides_per_100k','motor_vehicle_crash_deaths_per_100k']]
```

Out[43]:

	homicides_per_100k	motor_vehicle_crash_deaths_per_100k
0	0.0	0.00
1	0.0	18.26
2	0.0	19.94
3	0.0	30.31
4	0.0	0.00
...
1557	0.0	28.94
1558	0.0	18.02
1559	0.0	13.78
1560	0.0	0.00
1561	4.2	9.88

1562 rows × 2 columns

```
In [44]: test_values.homicides_per_100k.fillna(0, inplace=True)
test_values.motor_vehicle_crash_deaths_per_100k.fillna(0, inplace=True)
test_values[['homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k']]
```

Out[44]:

	homicides_per_100k	motor_vehicle_crash_deaths_per_100k
0	3.31	19.83
1	26.93	10.83
2	10.26	23.25
3	8.25	21.27
4	5.30	16.52
...
1571	5.67	25.89
1572	0.00	30.34
1573	0.00	10.74
1574	0.00	18.86
1575	0.00	0.00

1576 rows × 2 columns

```
In [45]: test_values.homicides_per_100k.fillna(0, inplace=True)
test_values.motor_vehicle_crash_deaths_per_100k.fillna(0, inplace=True)
test_values[['homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k']]
```

Out[45]:

	homicides_per_100k	motor_vehicle_crash_deaths_per_100k
0	3.31	19.83
1	26.93	10.83
2	10.26	23.25
3	8.25	21.27
4	5.30	16.52
...
1571	5.67	25.89
1572	0.00	30.34
1573	0.00	10.74
1574	0.00	18.86
1575	0.00	0.00

1576 rows × 2 columns

```
In [46]: test_values.homicides_per_100k.fillna(0, inplace=True)
test_values.motor_vehicle_crash_deaths_per_100k.fillna(0, inplace=True)
test_values[['homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k']]
```

Out[46]:

	homicides_per_100k	motor_vehicle_crash_deaths_per_100k
0	3.31	19.83
1	26.93	10.83
2	10.26	23.25
3	8.25	21.27
4	5.30	16.52
...
1571	5.67	25.89
1572	0.00	30.34
1573	0.00	10.74
1574	0.00	18.86
1575	0.00	0.00

1576 rows × 2 columns

```
In [47]: train_values_labels.columns
```

```
Out[47]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent'],
      dtype='object')
```

```
In [48]: train_values_labels.pct_adult_smoking.fillna(0, inplace=True)
train_values_labels.pct_adult_smoking.fillna(0, inplace=True)
train_values_labels['pct_adult_smoking']
```

```
Out[48]: 0      0.166
1      0.102
2      0.215
3      0.182
4      0.118
...
1557    0.000
1558    0.169
1559    0.193
1560    0.171
1561    0.178
Name: pct_adult_smoking, Length: 1562, dtype: float64
```

```
In [49]: test_values.pct_adult_smoking.fillna(0, inplace=True)
test_values.pct_adult_smoking.fillna(0, inplace=True)
test_values['pct_adult_smoking']
```

```
Out[49]: 0      0.257
1      0.187
2      0.121
3      0.226
4      0.191
...
1571    0.259
1572    0.195
1573    0.096
1574    0.161
1575    0.130
Name: pct_adult_smoking, Length: 1576, dtype: float64
```

```
In [50]: train_values_labels.pct_excessive_drinking.fillna(0, inplace=True)
train_values_labels.pct_excessive_drinking.fillna(0, inplace=True)
train_values_labels['pct_excessive_drinking']
```

```
Out[50]: 0      0.262
1      0.200
2      0.187
3      0.169
4      0.178
...
1557    0.000
1558    0.166
1559    0.168
1560    0.131
1561    0.149
Name: pct_excessive_drinking, Length: 1562, dtype: float64
```

```
In [51]: test_values.pct_excessive_drinking.fillna(0, inplace=True)
test_values.pct_excessive_drinking.fillna(0, inplace=True)
test_values['pct_excessive_drinking']
```

```
Out[51]: 0      0.048
1      0.164
2      0.000
3      0.150
4      0.153
...
1571    0.091
1572    0.000
1573    0.331
1574    0.107
1575    0.202
Name: pct_excessive_drinking, Length: 1576, dtype: float64
```

```
In [52]: train_values_labels.air_pollution_particulate_matter_value.fillna(0, inplace=True)
train_values_labels.air_pollution_particulate_matter_value.fillna(0, inplace=True)
train_values_labels['air_pollution_particulate_matter_value']
```

```
Out[52]: 0      11.022908
1      9.904099
2      11.011502
3      12.845770
4      11.565750
...
1557    12.942281
1558    10.669456
1559    12.962100
1560    10.244994
1561    13.099012
Name: air_pollution_particulate_matter_value, Length: 1562, dtype: float64
```

```
In [53]: test_values.air_pollution_particulate_matter_value.fillna(0, inplace=True)
test_values.air_pollution_particulate_matter_value.fillna(0, inplace=True)
test_values['air_pollution_particulate_matter_value']
```

```
Out[53]: 0      13.194028
1      12.419433
2      11.877446
3      9.826327
4      12.530045
...
1571   10.200021
1572   10.897362
1573   12.726635
1574   9.967003
1575   9.667330
Name: air_pollution_particulate_matter_value, Length: 1576, dtype: float64
```

```
In [54]: #train_values_labels.evictions.fillna(-1, inplace=True)
#train_values_labels.evictions.fillna(-1, inplace=True)
#train_values_labels['evictions']
```

```
In [55]: #test_values.evictions.fillna(-1, inplace=True)
#test_values.evictions.fillna(-1, inplace=True)
#test_values['evictions']
```

```
In [56]: #train_values_labels.pct_adult_obesity.fillna(0, inplace=True)
#train_values_labels.pct_adult_obesity.fillna(0, inplace=True)
#train_values_labels['pct_adult_obesity']
```

```
In [57]: #test_values.pct_adult_obesity.fillna(0, inplace=True)
#test_values.pct_adult_obesity.fillna(0, inplace=True)
#test_values['pct_adult_obesity']
```

```
In [58]: #train_values_labels.pct_diabetes.fillna(0, inplace=True)
#train_values_labels.pct_diabetes.fillna(0, inplace=True)
#train_values_labels['pct_diabetes']
```

```
In [59]: #test_values.pct_diabetes.fillna(0, inplace=True)
#test_values.pct_diabetes.fillna(0, inplace=True)
#test_values['pct_diabetes']
```

```
In [60]: #train_values_labels.pct_physical_inactivity.fillna(0, inplace=True)
#train_values_labels.pct_physical_inactivity.fillna(0, inplace=True)
#train_values_labels['pct_physical_inactivity']
```

```
In [61]: #test_values.pct_physical_inactivity.fillna(0, inplace=True)
#test_values.pct_physical_inactivity.fillna(0, inplace=True)
#test_values['pct_physical_inactivity']
```

```
In [62]: #train_values_labels.pct_low_birthweight.fillna(0, inplace=True)
#train_values_labels.pct_low_birthweight.fillna(0, inplace=True)
#train_values_labels['pct_low_birthweight']
```

```
In [63]: #test_values.pct_low_birthweight.fillna(0, inplace=True)
#test_values.pct_low_birthweight.fillna(0, inplace=True)
#test_values['pct_low_birthweight']
```

```
In [64]: #train_values_labels.pop_per_dentist.fillna(0, inplace=True)
#train_values_labels.pop_per_dentist.fillna(0, inplace=True)
#train_values_labels['pop_per_dentist']
```

```
In [65]: #test_values.pop_per_dentist.fillna(0, inplace=True)
#test_values.pop_per_dentist.fillna(0, inplace=True)
#test_values['pop_per_dentist']
```

```
In [66]: #train_values_labels.pop_per_primary_care_physician.fillna(0, inplace=True)
#train_values_labels.pop_per_primary_care_physician.fillna(0, inplace=True)
#train_values_labels['pop_per_primary_care_physician']
```

```
In [67]: #test_values.pop_per_primary_care_physician.fillna(0, inplace=True)
#test_values.pop_per_primary_care_physician.fillna(0, inplace=True)
#test_values['pop_per_primary_care_physician']
```

```
In [68]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state.describe(), '\n' )
```

```
          homicides_per_100k
count              30.0
mean               0.0
std                0.0
min                0.0
25%               0.0
50%               0.0
75%               0.0
max               0.0
```

```
          homicides_per_100k
count      26.000000
mean       0.183846
std        0.536828
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       2.150000
```

```
In [69]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	homicides_per_100k
0	fb8cab1	8e686a7	0.0
11	fb8cab1	170b0b3	0.0
65	fb8cab1	43b66dd	0.0
94	fb8cab1	555fe97	0.0
98	fb8cab1	118ca99	0.0
153	fb8cab1	3b9505f	0.0
277	fb8cab1	a6f2901	0.0
291	fb8cab1	e46e2df	0.0
364	fb8cab1	b301c5b	0.0
399	fb8cab1	b66aeb7	0.0
456	fb8cab1	62c7e21	0.0
458	fb8cab1	d27aeaf	0.0
531	fb8cab1	5a9c263	0.0
604	fb8cab1	6074b91	0.0
650	fb8cab1	978d233	0.0
787	fb8cab1	f62fc23	0.0
792	fb8cab1	26886b3	0.0
923	fb8cab1	62cb873	0.0
~	fb8cab1	eccb172	~ ~

```
In [70]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_excessive_drinking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_excessive_drinking
0	fb8cab1	8e686a7	0.262
11	fb8cab1	170b0b3	0.179
65	fb8cab1	43b66dd	0.205
94	fb8cab1	555fe97	0.233
98	fb8cab1	118ca99	0.202
153	fb8cab1	3b9505f	0.167
277	fb8cab1	a6f2901	0.265
291	fb8cab1	e46e2df	0.238
364	fb8cab1	b301c5b	0.272
399	fb8cab1	b66aeb7	0.264
456	fb8cab1	62c7e21	0.235
458	fb8cab1	d27aeaf	0.237
531	fb8cab1	5a9c263	0.282
604	fb8cab1	6074b91	0.191
650	fb8cab1	978d233	0.232
787	fb8cab1	f62fc23	0.361
792	fb8cab1	26886b3	0.152
923	fb8cab1	62cb873	0.160
~	fb8cab1	5ec0177	~ ~12

```
In [71]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_adult_smoking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

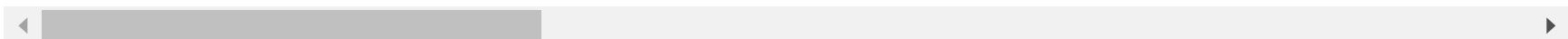
	state	county_code	pct_adult_smoking
0	fb8cab1	8e686a7	0.166
11	fb8cab1	170b0b3	0.157
65	fb8cab1	43b66dd	0.110
94	fb8cab1	555fe97	0.177
98	fb8cab1	118ca99	0.185
153	fb8cab1	3b9505f	0.112
277	fb8cab1	a6f2901	0.166
291	fb8cab1	e46e2df	0.246
364	fb8cab1	b301c5b	0.300
399	fb8cab1	b66aeb7	0.189
456	fb8cab1	62c7e21	0.175
458	fb8cab1	d27aeaf	0.153
531	fb8cab1	5a9c263	0.361
604	fb8cab1	6074b91	0.238
650	fb8cab1	978d233	0.155
787	fb8cab1	f62fc23	0.000
792	fb8cab1	26886b3	0.083
923	fb8cab1	62cb873	0.133
~	fb8cab1	fc0b172	0.161

```
In [72]: train_values_labels.fillna(train_values_labels.median(), inplace=True)  
train_values_labels
```

Out[72]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0
1561	1561	38ccdfd	1dcfd4e	650813.0	92680.0	37.789	27.0	30.687	0.704111	0

1562 rows × 46 columns



```
In [73]: test_values.fillna(train_values_labels.median(), inplace=True)  
test_values
```

Out[73]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white
0	0	8e7613e	9d1e27d	52842.0	5403.0	26.840	27.0	27.960	0.924234
1	1	694a5e8	a952566	212287.0	53502.0	57.534	6032.0	33.072	0.398318
2	2	b3f0726	20d32fc	81263.0	13368.0	39.994	1012.0	32.044	0.483789
3	3	fd922f0	1b0d913	122870.0	19359.0	41.865	27.0	30.724	0.468043
4	4	3bd551e	698ab34	146153.0	15766.0	30.681	644.0	30.860	0.651511
...
1571	1571	dae1584	52acab4	44414.0	6275.0	35.026	254.0	30.569	0.726121
1572	1572	b95e0ba	485e9af	16160.0	1871.0	31.922	27.0	33.899	0.663196
1573	1573	f44348c	1dcfd4e	28688.0	4478.0	33.897	27.0	28.644	0.809853
1574	1574	c9450a6	52acab4	29234.0	4528.0	40.479	57.0	24.803	0.707838
1575	1575

```
In [74]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state.describe(), '\n' )
```

```
          homicides_per_100k
count              30.0
mean               0.0
std                0.0
min                0.0
25%               0.0
50%               0.0
75%               0.0
max               0.0
```

```
          homicides_per_100k
count      26.000000
mean       0.183846
std        0.536828
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       2.150000
```

```
In [75]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','homicides_per_100k']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	homicides_per_100k
0	fb8cab1	8e686a7	0.0
11	fb8cab1	170b0b3	0.0
65	fb8cab1	43b66dd	0.0
94	fb8cab1	555fe97	0.0
98	fb8cab1	118ca99	0.0
153	fb8cab1	3b9505f	0.0
277	fb8cab1	a6f2901	0.0
291	fb8cab1	e46e2df	0.0
364	fb8cab1	b301c5b	0.0
399	fb8cab1	b66aeb7	0.0
456	fb8cab1	62c7e21	0.0
458	fb8cab1	d27aeaf	0.0
531	fb8cab1	5a9c263	0.0
604	fb8cab1	6074b91	0.0
650	fb8cab1	978d233	0.0
787	fb8cab1	f62fc23	0.0
792	fb8cab1	26886b3	0.0
923	fb8cab1	62cb873	0.0
~	fb8cab1	eccb172	~ ~

```
In [76]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_excessive_drinking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_excessive_drinking
0	fb8cab1	8e686a7	0.262
11	fb8cab1	170b0b3	0.179
65	fb8cab1	43b66dd	0.205
94	fb8cab1	555fe97	0.233
98	fb8cab1	118ca99	0.202
153	fb8cab1	3b9505f	0.167
277	fb8cab1	a6f2901	0.265
291	fb8cab1	e46e2df	0.238
364	fb8cab1	b301c5b	0.272
399	fb8cab1	b66aeb7	0.264
456	fb8cab1	62c7e21	0.235
458	fb8cab1	d27aeaf	0.237
531	fb8cab1	5a9c263	0.282
604	fb8cab1	6074b91	0.191
650	fb8cab1	978d233	0.232
787	fb8cab1	f62fc23	0.361
792	fb8cab1	26886b3	0.152
923	fb8cab1	62cb873	0.160
~	fb8cab1	5ec0177	~ ~12

```
In [77]: nums = list(range(0, 50))

for i in nums:
    specific_state = train_values_labels[train_values_labels['state'] == states[i]]
    county_codes_state = specific_state[['state','county_code','pct_adult_smoking']]
    print(county_codes_state, '\n', county_codes_state.describe(), '\n' )
```

	state	county_code	pct_adult_smoking
0	fb8cab1	8e686a7	0.166
11	fb8cab1	170b0b3	0.157
65	fb8cab1	43b66dd	0.110
94	fb8cab1	555fe97	0.177
98	fb8cab1	118ca99	0.185
153	fb8cab1	3b9505f	0.112
277	fb8cab1	a6f2901	0.166
291	fb8cab1	e46e2df	0.246
364	fb8cab1	b301c5b	0.300
399	fb8cab1	b66aeb7	0.189
456	fb8cab1	62c7e21	0.175
458	fb8cab1	d27aeaf	0.153
531	fb8cab1	5a9c263	0.361
604	fb8cab1	6074b91	0.238
650	fb8cab1	978d233	0.155
787	fb8cab1	f62fc23	0.000
792	fb8cab1	26886b3	0.083
923	fb8cab1	62cb873	0.133
~	fb8cab1	fc0b172	0.161

```
In [78]: train_values_labels.columns
```

```
Out[78]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent'],
      dtype='object')
```

```
In [79]: train_values_labels['rucc'].value_counts()
```

```
Out[79]: Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area    301
Metro - Counties in metro areas of 1 million population or more                    219
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area 215
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area           206
Metro - Counties in metro areas of 250,000 to 1 million population                  193
Metro - Counties in metro areas of fewer than 250,000 population                   165
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area 120
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area                101
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area             42
Name: rucc, dtype: int64
```

```
In [80]: low_evictions = train_values_labels[train_values_labels['evictions'] == -1]
low_evictions
```

Out[80]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
32	32	49e4d3b	8036085	1043.0		125.0	25.425	-1.0	17.492	0.971436
39	39	f59fa79	c479f0c	21758.0		2017.0	23.901	-1.0	29.654	0.863019
61	61	dafe063	1b0d913	18240.0		1793.0	29.867	-1.0	23.228	0.169795
100	100	c5c7967	1b0d913	1426.0		149.0	19.095	-1.0	31.547	0.755442
101	101	e24aadb	485e9af	10332.0		923.0	25.519	-1.0	24.556	0.660098
...
1480	1480	d968a45	1b0d913	3460.0		370.0	28.363	-1.0	16.899	0.666335
1513	1513	8919d15	1b0d913	7233.0		625.0	29.491	-1.0	28.766	0.155347
1546	1546	151b7a5	2b7da97	7001.0		568.0	20.640	-1.0	33.245	0.936286
1551	1551	f592e2d	842bd12	4239.0		306.0	20.508	-1.0	27.847	0.944484
1555	1555	6e49271	9d0874a	14055.0		729.0	18.999	-1.0	33.118	0.713404

82 rows × 46 columns

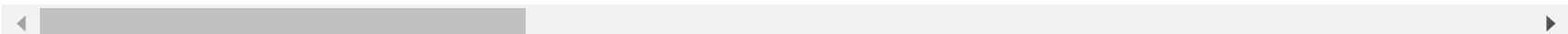


```
In [81]: low_evictions.describe()
```

Out[81]:

	row_id	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am	pct_hi
count	82.000000	82.000000	82.000000	82.000000	82.0	82.000000	82.000000	82.000000	82.0
mean	723.170732	7457.439024	780.085366	26.568659	-1.0	25.478256	0.729019	0.081178	0.1
std	443.764029	6561.909862	745.466201	8.292358	0.0	6.355004	0.257032	0.161151	0.2
min	32.000000	269.000000	64.000000	12.153000	-1.0	9.909000	0.050944	0.000000	0.0
25%	379.000000	2605.500000	294.250000	21.582250	-1.0	20.804250	0.560238	0.000954	0.0
50%	667.000000	6218.500000	575.000000	25.657500	-1.0	26.003500	0.832817	0.008335	0.0
75%	1042.250000	9360.500000	976.250000	30.020250	-1.0	29.774750	0.939651	0.042972	0.2
max	1555.000000	33342.000000	4384.000000	69.813000	-1.0	39.122000	0.995141	0.732029	0.9

8 rows × 41 columns



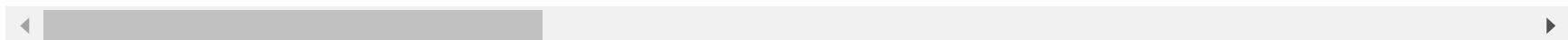
```
In [82]: zeroevictions = train_values_labels[train_values_labels['evictions']==0]
zeroevictions
```

Out[82]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
	3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835
	5	5	abf992b	4522abc	8540.0	751.0	22.543	0.0	34.937	0.560637
	27	27	9e8fcfd3	8036085	4203.0	504.0	27.531	0.0	24.138	0.855246
	46	46	dc0571b	4522abc	7306.0	1135.0	39.341	0.0	29.593	0.334105
	55	55	2fa2d23	4522abc	5114.0	576.0	35.694	0.0	25.538	0.710549

	1523	1523	d069cab	b795815	1954.0	202.0	24.789	0.0	18.753	0.888687
	1532	1532	798c659	9d0874a	22965.0	1561.0	14.438	0.0	32.590	0.878718
	1535	1535	c604e53	1b0d913	18504.0	2104.0	22.532	0.0	26.377	0.791739
	1541	1541	a1bf7f3	9d1e27d	13179.0	1331.0	20.516	0.0	27.429	0.919745
	1552	1552	103631f	0f8930b	2205.0	111.0	11.277	0.0	36.995	0.977926

110 rows × 46 columns



In [83]: `zeroevictions.describe()`

Out[83]:

	row_id	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am	pct_l
count	110.000000	110.000000	110.000000	110.000000	110.0	110.000000	110.000000	110.000000	110.000000
mean	798.090909	8004.572727	834.809091	25.228418	0.0	26.246627	0.766992	0.075711	0.000000
std	472.952595	7466.430563	785.523100	6.865803	0.0	5.445414	0.217541	0.150867	0.000000
min	3.000000	710.000000	92.000000	7.279000	0.0	12.925000	0.099335	0.000000	0.000000
25%	410.250000	3071.000000	314.500000	20.965000	0.0	23.164250	0.596758	0.001494	0.000000
50%	769.500000	5811.000000	576.500000	25.100000	0.0	25.546500	0.875195	0.006740	0.000000
75%	1230.250000	10380.250000	1090.250000	29.970500	0.0	29.518250	0.950102	0.053475	0.000000
max	1552.000000	38911.000000	4565.000000	42.419000	0.0	44.096000	0.991058	0.621763	0.000000

8 rows × 41 columns



In [84]: `#train_values_labels['evictions'].replace(to_replace =[0],
value =-1, inplace=True)`

In [85]: `#test_values['evictions'].replace(to_replace =[0],
value =-1, inplace=True)`

In [86]: `low_murder = train_values_labels[train_values_labels['homicides_per_100k'] < 0]
low_murder`

Out[86]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_a
	713	713	da2e0ef	64ffe5d	547371.0	49957.0	32.959	246.0	29.96	0.833345

1 rows × 46 columns



```
In [87]: low_evictions['rucc']
```

```
Out[87]: 32    Nonmetro - Completely rural or less than 2,500...
39    Nonmetro - Urban population of 2,500 to 19,999...
61    Nonmetro - Urban population of 2,500 to 19,999...
100   Nonmetro - Completely rural or less than 2,500...
101   Nonmetro - Completely rural or less than 2,500...
...
1480   Nonmetro - Completely rural or less than 2,500...
1513   Nonmetro - Urban population of 2,500 to 19,999...
1546   Metro - Counties in metro areas of fewer than ...
1551   Nonmetro - Completely rural or less than 2,500...
1555   Nonmetro - Completely rural or less than 2,500...
Name: rucc, Length: 82, dtype: object
```

```
In [88]: low_murder['rucc']
```

```
Out[88]: 713    Metro - Counties in metro areas of 250,000 to ...
Name: rucc, dtype: object
```

```
In [89]: train_values_labels['urban_influence'].value_counts()
```

```
Out[89]: Small-in a metro area with fewer than 1 million residents          358
Large-in a metro area with at least 1 million residents or more           219
Noncore adjacent to a small metro with town of at least 2,500 residents  178
Micropolitan not adjacent to a metro area                            131
Micropolitan adjacent to a small metro area                         113
Noncore adjacent to micro area and does not contain a town of at least 2,500 residents 97
Noncore adjacent to a small metro and does not contain a town of at least 2,500 residents 88
Noncore not adjacent to a metro/micro area and does not contain a town of at least 2,500 residents 86
Noncore adjacent to micro area and contains a town of 2,500-19,999 residents 86
Noncore adjacent to a large metro area                           79
Micropolitan adjacent to a large metro area                         64
Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents 63
Name: urban_influence, dtype: int64
```

```
In [90]: low_evictions['urban_influence']
```

```
Out[90]: 32    Noncore not adjacent to a metro/micro area and...
39    Noncore adjacent to a small metro with town of...
61          Noncore adjacent to a large metro area
100   Noncore not adjacent to a metro/micro area and...
101   Noncore adjacent to micro area and does not co...
...
1480   Noncore not adjacent to a metro/micro area and...
1513   Noncore adjacent to a small metro with town of...
1546   Small-in a metro area with fewer than 1 millio...
1551   Noncore adjacent to a small metro and does not...
1555   Noncore adjacent to a small metro and does not...
Name: urban_influence, Length: 82, dtype: object
```

```
In [91]: low_murder['urban_influence']
```

```
Out[91]: 713    Small-in a metro area with fewer than 1 millio...
Name: urban_influence, dtype: object
```

```
In [92]: train_values_labels['economic_typerology'].value_counts()
```

```
Out[92]: Nonspecialized           631
Manufacturing-dependent        244
Farm-dependent                 217
Federal/State government-dependent 191
Recreation                      166
Mining-dependent                  113
Name: economic_typerology, dtype: int64
```

```
In [93]: low_evictions['economic_typology']
```

```
Out[93]: 32          Farm-dependent
39      Federal/State government-dependent
61          Mining-dependent
100         Mining-dependent
101         Nonspecialized
...
1480         Mining-dependent
1513         Mining-dependent
1546         Recreation
1551         Farm-dependent
1555         Nonspecialized
Name: economic_typology, Length: 82, dtype: object
```

```
In [94]: zeroevictions['economic_typology']
```

```
Out[94]: 3          Nonspecialized
5          Mining-dependent
27         Farm-dependent
46         Nonspecialized
55         Manufacturing-dependent
...
1523        Farm-dependent
1532        Nonspecialized
1535        Mining-dependent
1541    Federal/State government-dependent
1552        Recreation
Name: economic_typology, Length: 110, dtype: object
```

```
In [95]: low_murder['economic_typology']
```

```
Out[95]: 713    Nonspecialized
Name: economic_typology, dtype: object
```

```
In [96]: train_values_labels.isnull().any()
```

```
Out[96]: row_id                         False  
county_code                      False  
state                            False  
population                        False  
renter_occupied_households      False  
pct_renter_occupied                False  
evictions                          False  
rent_burden                        False  
pct_white                           False  
pct_af_am                            False  
pct_hispanic                        False  
pct_am_ind                           False  
pct_asian                            False  
pct_nh_pi                            False  
pct_multiple                        False  
pct_other                            False  
poverty_rate                        False  
rucc                                False  
urban_influence                      False  
economic_typerology                  False  
pct_civilian_labor                  False  
pct_unemployment                     False  
pct_uninsured_adults                 False  
pct_uninsured_children                False  
pct_adult_obesity                    False  
pct_adult_smoking                     False  
pct_diabetes                          False  
pct_low_birthweight                   False  
pct_excessive_drinking                False  
pct_physical_inactivity                False  
air_pollution_particulate_matter_value False  
homicides_per_100k                     False  
motor_vehicle_crash_deaths_per_100k    False  
heart_disease_mortality_per_100k       False  
pop_per_dentist                       False  
pop_per_primary_care_physician        False  
pct_female                            False  
pct_below_18_years_of_age              False  
pct_aged_65_years_and_older            False  
pct_adults_less_than_a_high_school_diploma False
```

```
pct_adults_with_high_school_diploma      False
pct_adults_with_some_college            False
pct_adults_bachelors_or_higher          False
birth_rate_per_1k                      False
death_rate_per_1k                      False
gross_rent                            False
dtype: bool
```

In [97]: train_values_labels

Out[97]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0
1561	1561	38ccdfd	1dcfd4e	650813.0	92680.0	37.789	27.0	30.687	0.704111	0

1562 rows × 46 columns

In [98]: train_values_labels.to_csv(r'C:\Users\Jorge\OneDrive\DAT102x.2\train_values_labels.csv', index = None, header=Tr

```
In [99]: train_values_labels.columns
```

```
Out[99]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent'],
      dtype='object')
```

```
In [100]: train_values_labels['state'].value_counts()
```

```
Out[100]: 1b0d913    131
4522abc     93
a952566     69
528ea9f      57
20d32fc      56
c479f0c      56
08f8fb4      54
dc9ae72      50
9d1e27d      47
09d8cd0      47
b795815      45
0f8930b      44
e2f94fa      44
7572db1      40
9d0874a      39
9dda412      39
158df01      38
1dcfd4e      37
e74aca3      36
8036085      35
c3dbf0a      34
78e8330      33
9e0007d      33
e899d7f      31
4c72956      31
52acab4      31
fb8cab1      30
5086a32      28
1646cf6      27
842bd12      26
485e9af      25
698ab34      21
4cd9667      20
3745933      19
7dd3518      18
176f5f0      16
64ffe5d      15
d233cec      10
5029ed4       8
bc77872       7
```

```
2b7da97      6  
fa605d5      6  
dfc21f3      6  
3337bbb      6  
b44cf6e      6  
105e445      4  
6d287d7      4  
375d4d3      2  
914c15f      1  
9e065a4      1  
Name: state, dtype: int64
```

```
In [101]: train_values_labels['state'].unique()
```

```
Out[101]: array(['fb8cab1', '842bd12', '2b7da97', '5029ed4', 'b795815', '4522abc',  
        '4c72956', '78e8330', '9e0007d', 'e74aca3', '1b0d913', '485e9af',  
        '4cd9667', '1646cf6', '20d32fc', 'a952566', '8036085', '9dda412',  
        '5086a32', 'e899d7f', '158df01', '6d287d7', '08f8fb4', '1dcfd4e',  
        '9d0874a', '528ea9f', '7dd3518', 'e2f94fa', 'c479f0c', 'dc9ae72',  
        '698ab34', '176f5f0', '0f8930b', '52acab4', '9d1e27d', 'c3dbf0a',  
        'b44cf6e', '7572db1', 'bc77872', 'dfc21f3', '3745933', '64ffe5d',  
        '09d8cd0', '105e445', 'fa605d5', '9e065a4', 'd233cec', '3337bbb',  
        '375d4d3', '914c15f'], dtype=object)
```

```
In [102]: train_values_labels['county_code'].value_counts()
```

```
Out[102]: 2d693e1    1  
7fa2453    1  
91c3edf    1  
6190dfd    1  
f3bc345    1  
..  
afba46f    1  
98b941b    1  
5fcbae1    1  
1eb4681    1  
03227cb    1  
Name: county_code, Length: 1562, dtype: int64
```

```
In [103]: outlier = train_values_labels[train_values_labels['gross_rent'] > 1800]
```

```
In [104]: outlier['gross_rent']
```

```
Out[104]: 564    1827  
Name: gross_rent, dtype: int64
```

```
In [105]: #train_values_labels.drop(outlier.index, inplace = True)
```

```
In [106]: train_values_labels['gross_rent'].describe()
```

```
Out[106]: count    1562.000000  
mean     701.142125  
std      192.883110  
min      351.000000  
25%      578.000000  
50%      650.000000  
75%      773.750000  
max     1827.000000  
Name: gross_rent, dtype: float64
```

```
In [107]: #train_values_labels.drop(low_murder.index, inplace = True)
```

```
In [108]: num_cols = ['population',  
                 'renter_occupied_households', 'pct_renter_occupied', 'evictions',  
                 'rent_burden', 'poverty_rate',  
                 'pct_civilian_labor',  
                 'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',  
                 'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',  
                 'pct_low_birthweight', 'pct_excessive_drinking',  
                 'pct_physical_inactivity', 'air_pollution_particulate_matter_value',  
                 'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',  
                 'heart_disease_mortality_per_100k', 'pop_per_dentist',  
                 'pop_per_primary_care_physician',  
                 'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',  
                 'pct_adults_less_than_a_high_school_diploma',  
                 'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',  
                 'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',  
                 'death_rate_per_1k', 'gross_rent', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',  
                 'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other']
```

```
In [109]: cat_cols = ['rucc', 'urban_influence', 'economic_typology']
```

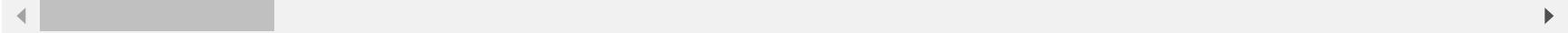
```
In [110]: Correlations = train_values_labels[num_cols]
```

```
corr = Correlations.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[110]:

	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pov
population	1	0.979015	0.285874	0.763193	0.162747	-0.
renter_occupied_households	0.979015	1	0.320175	0.76267	0.146213	-0.0
pct_renter_occupied	0.285874	0.320175	1	0.288118	0.229203	(
evictions	0.763193	0.76267	0.288118	1	0.128273	0.
rent_burden	0.162747	0.146213	0.229203	0.128273	1	(
poverty_rate	-0.0296009	-0.00251764	0.311045	0.0207987	0.387541	(
pct_civilian_labor	0.0745728	0.0658938	-0.00993314	0.0719095	-0.36754	-0.
pct_unemployment	0.00402801	0.00999491	0.123263	0.00654586	0.429351	(
pct_uninsured_adults	7.83003e-05	0.0148575	0.182201	0.0359564	0.0593151	(
pct_uninsured_children	-0.0591799	-0.0474549	0.0113586	-0.0328941	-0.210452	(
pct_adult_obesity	-0.217104	-0.197823	-0.0529688	-0.10484	-0.00809	(
pct_adult_smoking	-0.0406649	-0.0390331	-0.0114915	-0.00756537	0.17016	(
pct_diabetes	-0.172004	-0.150242	-0.134406	-0.0933151	0.144596	(
pct_low_birthweight	-0.0256529	-0.0106273	0.154435	0.0527257	0.275532	(
pct_excessive_drinking	0.138904	0.116266	0.0814671	0.111202	-0.0360056	-0.
pct_physical_inactivity	-0.240359	-0.206249	-0.154269	-0.153729	-0.129752	(
air_pollution_particulate_matter_value	-0.0688334	-0.0699489	-0.143764	-0.0135237	0.157737	0.
homicides_per_100k	0.194127	0.194908	0.32057	0.277229	0.313355	(
motor_vehicle_crash_deaths_per_100k	-0.15935	-0.143949	-0.148669	-0.133605	0.0793123	(
heart_disease_mortality_per_100k	-0.133897	-0.112827	0.0309446	-0.074842	0.124221	(
pop_per_dentist	-0.160052	-0.144359	-0.228345	-0.140227	-0.00568969	(
pop_per_primary_care_physician	-0.128322	-0.116253	-0.227424	-0.11299	0.00536428	(

	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pov
pct_below_18_years_of_age	0.0676451	0.0408083	0.110914	0.0744428	-0.180496	(0.00)
pct_aged_65_years_and_older	-0.202788	-0.182354	-0.457253	-0.206524	-0.115864	(0.00)
pct_adults_less_than_a_high_school_diploma	-0.029638	-0.0113852	0.1038	-0.0303286	0.146306	(0.00)
pct_adults_with_high_school_diploma	-0.303886	-0.271734	-0.372389	-0.246811	-0.166862	(0.00)
pct_adults_with_some_college	-0.0689123	-0.0759798	-0.0763535	-0.0459329	-0.170873	(0.00)
pct_adults_bachelors_or_higher	0.295888	0.261647	0.255215	0.239207	0.118751	(0.00)
birth_rate_per_1k	0.0999571	0.0992008	0.31906	0.125148	-0.116683	(0.00)
death_rate_per_1k	-0.252709	-0.221893	-0.320552	-0.205856	-0.0569726	(0.00)
gross_rent	0.398343	0.345154	0.278613	0.251339	0.237213	(0.00)
pct_white	-0.233149	-0.227568	-0.472168	-0.22895	-0.192858	(0.00)
pct_af_am	0.065498	0.0760577	0.283012	0.164462	0.33612	(0.00)
pct_hispanic	0.194042	0.17686	0.211543	0.119254	-0.0153675	(0.00)
pct_am_ind	-0.0476048	-0.0403485	0.140909	-0.0394346	-0.144384	(0.00)
pct_asian	0.459903	0.435849	0.407788	0.289347	0.127979	(0.00)
pct_nh_pi	0.0861408	0.0768735	0.172797	0.0154869	-0.00685285	(0.00)
pct_multiple	0.0745288	0.0690093	0.195395	0.0601595	0.0185502	(0.00)
pct_other	0.219683	0.218633	0.231417	0.171253	0.154167	(0.00)



```
In [111]: for col in train_values_labels:  
    train_values_labels[col].describe()  
    print(col, train_values_labels[col].describe(), '\n')
```

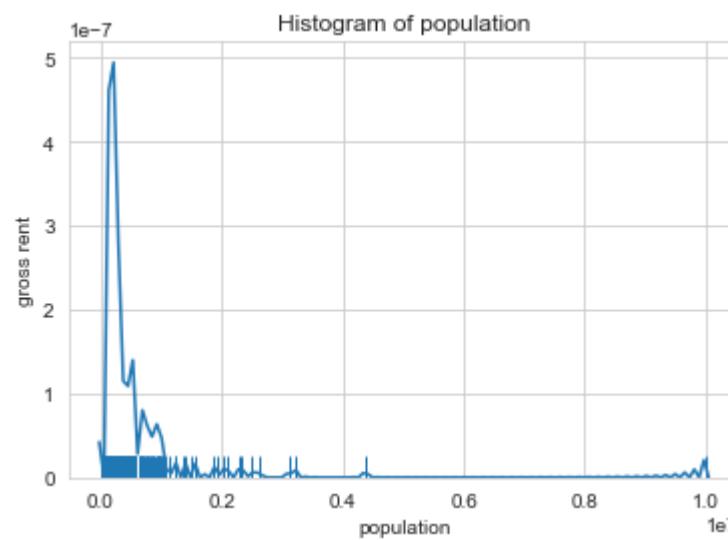
```
row_id count      1562.000000  
mean      780.500000  
std       451.054875  
min       0.000000  
25%     390.250000  
50%     780.500000  
75%   1170.750000  
max   1561.000000  
Name: row_id, dtype: float64
```

```
county_code count      1562  
unique      1562  
top        2d693e1  
freq        1  
Name: county_code, dtype: object
```

```
state count      1562  
unique      50  
top        1b0d913  
...
```

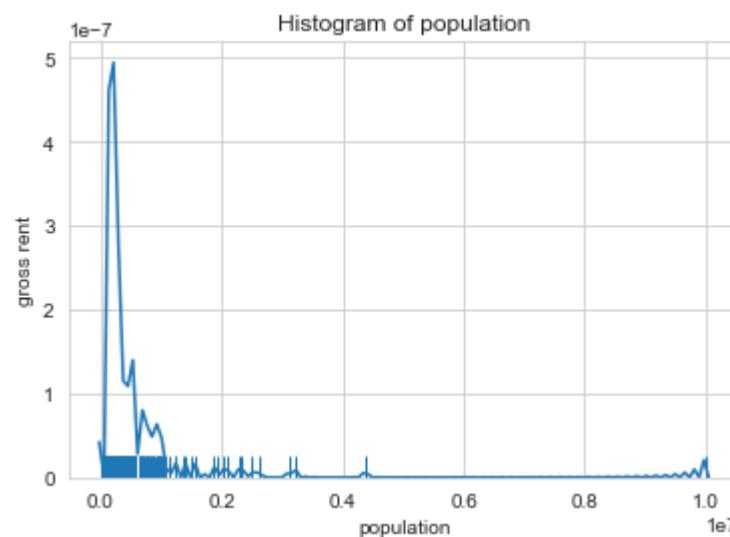
```
In [112]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_density_hist(gross_rent, cols, bins = 10, hist = False):
    for col in cols:
        sns.set_style("whitegrid")
        sns.distplot(train_values_labels[col], bins = bins, rug=True, hist = hist)
        plt.title('Histogram of ' + col) # Give the plot a main title
        plt.xlabel(col) # Set text for the x axis
        plt.ylabel('gross rent')# Set text for y axis
        plt.show()

plot_density_hist(train_values_labels, num_cols)
```



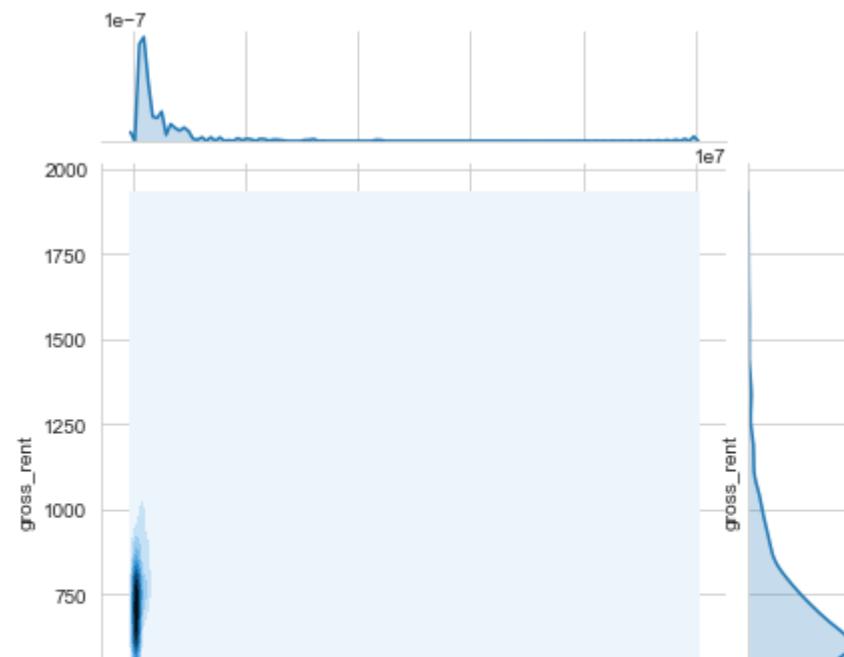
```
In [113]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_density_hist(gross_rent, cols, bins = 10, hist = False):
    for col in cols:
        sns.set_style("whitegrid")
        sns.distplot(train_values_labels[col], bins = bins, rug=True, hist = hist)
        plt.title('Histogram of ' + col) # Give the plot a main title
        plt.xlabel(col) # Set text for the x axis
        plt.ylabel('gross rent')# Set text for y axis
        plt.show()

plot_density_hist(test_values, num_cols)
```



```
In [114]: def plot_density_2d(auto_prices, cols, col_y = 'gross_rent', kind ='kde'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.jointplot(col, col_y, data=train_values_labels, kind=kind)
        #plt.xlabel(col) # Set text for the x axis
        plt.ylabel(col_y)# Set text for y axis
        plt.show()

plot_density_2d(train_values_labels, num_cols)
```



```
In [115]: train_values_labels['evictionslog'] = np.log(train_values_labels['evictions']+25)
```

```
In [116]: test_values['evictionslog'] = np.log(test_values['evictions']+25)
```

```
In [117]:
```

```
train_values_labels['homicides_per_100klog'] = np.log(train_values_labels['homicides_per_100k']+30)
```

In [118]:

```
test_values['homicides_per_100klog'] = np.log(test_values['homicides_per_100k']+30)
```

In [119]:

```
train_values_labels['populationlog'] = np.log(train_values_labels['population'])
```

In [120]:

```
test_values['populationlog'] = np.log(test_values['population'])
```

In [121]:

```
train_values_labels['renter_occupied_householdslog'] = np.log(train_values_labels['renter_occupied_households'])
```

In [122]:

```
test_values['renter_occupied_householdslog'] = np.log(test_values['renter_occupied_households'])
```

In [123]:

```
train_values_labels['pct_renter_occupiedlog'] = np.log(train_values_labels['pct_renter_occupied'])
```

In [124]:

```
test_values['pct_renter_occupiedlog'] = np.log(test_values['pct_renter_occupied'])
```

In [125]:

```
train_values_labels['rent_burdenlog'] = np.log(train_values_labels['rent_burden'])
```

In [126]:

```
test_values['rent_burdenlog'] = np.log(test_values['rent_burden'])
```

In [127]:

```
train_values_labels['poverty_ratelog'] = np.sqrt(train_values_labels['poverty_rate'])
```

In [128]:

```
test_values['poverty_ratelog'] = np.sqrt(test_values['poverty_rate'])
```

In [129]:

```
train_values_labels['motor_vehicle_crash_deaths_per_100klog'] = np.log(train_values_labels['motor_vehicle_crash_
```

In [130]:

```
test_values['motor_vehicle_crash_deaths_per_100klog'] = np.log(test_values['motor_vehicle_crash_deaths_per_100k']
```

```
In [131]: train_values_labels['pop_per_dentistlog'] = np.log(train_values_labels['pop_per_dentist'])
```

```
In [132]: test_values['pop_per_dentistlog'] = np.log(test_values['pop_per_dentist'])
```

```
In [133]: train_values_labels['pop_per_primary_care_physicianlog'] = np.log(train_values_labels['pop_per_primary_care_physi...']
```

```
In [134]: test_values['pop_per_primary_care_physicianlog'] = np.log(test_values['pop_per_primary_care_physician'])
```

```
In [135]: train_values_labels['pct_adults_less_than_a_high_school_diplomalog'] = np.log(train_values_labels['pct_adults_le...']
```

```
In [136]: test_values['pct_adults_less_than_a_high_school_diplomalog'] = np.log(test_values['pct_adults_less_than_a_high_s...']
```

```
In [137]: train_values_labels['pct_adults_bachelors_or_higherlog'] = np.log(train_values_labels['pct_adults_bachelors_or_hi...']
```

```
In [138]: test_values['pct_adults_bachelors_or_higherlog'] = (test_values['pct_adults_bachelors_or_higher'])**2
```

```
In [139]: train_values_labels['gross_rentlog'] = np.log(train_values_labels['gross_rent'])
```

```
In [140]: train_values_labels['air_pollution_particulate_matter_valueolog'] = np.log(train_values_labels['air_pollution_pa...']
```

```
In [141]: test_values['air_pollution_particulate_matter_valueolog'] = np.log(test_values['air_pollution_particulate_matter_...']
```

```
In [142]: train_values_labels['pct_uninsured_childrenlog'] = np.log(train_values_labels['pct_uninsured_children'])
```

```
In [143]: test_values['pct_uninsured_childrenlog'] = np.log(test_values['pct_uninsured_children'])
```

```
In [144]: train_values_labels['pct_whitelog'] = np.log(train_values_labels['pct_white'])
```

```
In [145]: test_values['pct_whitelog'] = np.log(test_values['pct_white'])
```

```
In [146]: train_values_labels['pct_af_amlog'] = np.sqrt(train_values_labels['pct_af_am'])
```

```
In [147]: test_values['pct_af_amlog'] = np.sqrt(test_values['pct_af_am'])
```

```
In [148]: train_values_labels['pct_hispaniclog'] = np.sqrt(train_values_labels['pct_hispanic'])
```

```
In [149]: test_values['pct_hispaniclog'] = np.sqrt(test_values['pct_hispanic'])
```

```
In [150]: train_values_labels['pct_am_indlog'] = np.sqrt(train_values_labels['pct_am_ind'])
```

```
In [151]: test_values['pct_am_indlog'] = np.sqrt(test_values['pct_am_ind'])
```

```
In [152]: train_values_labels['pct_asianlog'] = np.sqrt(train_values_labels['pct_asian'])
```

```
In [153]: test_values['pct_asianlog'] = np.sqrt(test_values['pct_asian'])
```

```
In [154]: train_values_labels['pct_nh_pilog'] = np.sqrt(train_values_labels['pct_nh_pi'])
```

```
In [155]: test_values['pct_nh_pilog'] = np.sqrt(test_values['pct_nh_pi'])
```

```
In [156]: train_values_labels['pct_multiplelog'] = np.sqrt(train_values_labels['pct_multiple'])
```

```
In [157]: test_values['pct_multiplelog'] = np.sqrt(test_values['pct_multiple'])
```

```
In [158]: train_values_labels['pct_otherlog'] = np.sqrt(train_values_labels['pct_other'])
```

```
In [159]: test_values['pct_otherlog'] = np.sqrt(test_values['pct_other'])

In [160]: train_values_labels['pct_aged_65_years_and_olderlog'] = np.log(train_values_labels['pct_aged_65_years_and_older'])

In [161]: test_values['pct_aged_65_years_and_olderlog'] = np.log(test_values['pct_aged_65_years_and_older'])

In [162]: train_values_labels['heart_disease_mortality_per_100klog'] = np.log(train_values_labels['heart_disease_mortality_per_100k'])

In [163]: test_values['heart_disease_mortality_per_100klog'] = np.log(test_values['heart_disease_mortality_per_100k'])

In [164]: train_values_labels['death_rate_per_1klog'] = np.log(train_values_labels['death_rate_per_1k'])

In [165]: test_values['death_rate_per_1klog'] = np.log(test_values['death_rate_per_1k'])

In [166]: train_values_labels['pct_adult_obesitylog'] = np.log(train_values_labels['pct_adult_obesity'])

In [167]: test_values['pct_adult_obesitylog'] = np.log(test_values['pct_adult_obesity'])

In [168]: train_values_labels['pct_adult_smokinglog'] = np.log(train_values_labels['pct_adult_smoking']+25)

In [169]: test_values['pct_adult_smokinglog'] = np.log(test_values['pct_adult_smoking']+25)

In [170]: train_values_labels['pct_diabeteslog'] = np.log(train_values_labels['pct_diabetes'])

In [171]: test_values['pct_diabeteslog'] = np.log(test_values['pct_diabetes'])

In [172]: train_values_labels['pct_low_birthweightlog'] = np.log(train_values_labels['pct_low_birthweight'])
```

```
In [173]: test_values['pct_low_birthweightlog'] = np.log(test_values['pct_low_birthweight'])
```

```
In [174]: train_values_labels['pct_physical_inactivitylog'] = np.log(train_values_labels['pct_physical_inactivity'])
```

```
In [175]: test_values['pct_physical_inactivitylog'] = np.log(test_values['pct_physical_inactivity'])
```

```
In [176]: train_values_labels['pct_excessive_drinkinglog'] = np.log(train_values_labels['pct_excessive_drinking']+25)
```

```
In [177]: test_values['pct_excessive_drinkinglog'] = np.log(test_values['pct_excessive_drinking']+25)
```

```
In [178]: train_values_labels['pct_below_18_years_of_agelog'] = np.log(train_values_labels['pct_below_18_years_of_age']+25)
```

```
In [179]: test_values['pct_below_18_years_of_agelog'] = np.log(test_values['pct_below_18_years_of_age'])
```

```
In [180]: train_values_labels['pct_adults_with_high_school_diplomalog'] = np.log(train_values_labels['pct_adults_with_high_school_diploma'])
```

```
In [181]: test_values['pct_adults_with_high_school_diplomalog'] = np.log(test_values['pct_adults_with_high_school_diploma'])
```

```
In [182]: train_values_labels['pct_adults_with_some_collegelog'] = np.log(train_values_labels['pct_adults_with_some_college'])
```

```
In [183]: test_values['pct_adults_with_some_collegelog'] = np.log(test_values['pct_adults_with_some_college'])
```

```
In [184]: train_values_labels['birth_rate_per_1klog'] = np.log(train_values_labels['birth_rate_per_1k'])
```

```
In [185]: test_values['birth_rate_per_1klog'] = np.log(test_values['birth_rate_per_1k'])
```

```
In [186]: train_values_labels['pct_civilian_laborlog'] = np.log(train_values_labels['pct_civilian_labor'])
```

```
In [187]: test_values['pct_civilian_laborlog'] = np.log(test_values['pct_civilian_labor'])
```

```
In [188]: train_values_labels['pct_unemploymentlog'] = np.log(train_values_labels['pct_unemployment'])
```

```
In [189]: test_values['pct_unemploymentlog'] = np.log(test_values['pct_unemployment'])
```

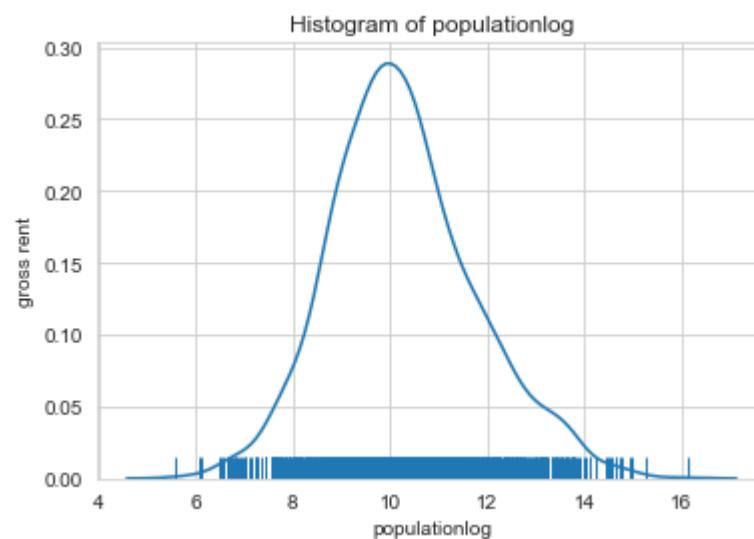
```
In [190]: train_values_labels['pct_uninsured_adultslog'] = np.log(train_values_labels['pct_uninsured_adults']+25)
```

```
In [191]: test_values['pct_uninsured_adultslog'] = np.log(test_values['pct_uninsured_adults']+25)
```

```
In [192]: updated_num_cols = ['populationlog','homicides_per_100klog',
'motor_vehicle_crash_deaths_per_100klog', 'evictionslog',
'renter_occupied_householdslog',
'pct_renter_occupied',
'rent_burden',
'poverty_ratelog',
'pct_civilian_laborlog',
'pct_unemploymentlog',
'pct_uninsured_adults',
'pct_uninsured_children',
'pct_adult_obesity',
'pct_adult_smokinglog',
'pct_diabeteslog',
'pct_low_birthweightlog',
'pct_excessive_drinkinglog',
'pct_physical_inactivitylog',
'air_pollution_particulate_matter_value',
'heart_disease_mortality_per_100klog',
'pop_per_dentistlog',
'pop_per_primary_care_physicianlog',
'pct_below_18_years_of_agelog',
'pct_aged_65_years_and_olderlog',
'pct_adults_less_than_a_high_school_diplomalog',
'pct_adults_with_high_school_diploma',
'pct_adults_with_some_college',
'pct_adults_bachelors_or_higher',
'birth_rate_per_1k',
'death_rate_per_1k']
```

```
In [193]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_density_hist(gross_rent, cols, bins = 10, hist = False):
    for col in cols:
        sns.set_style("whitegrid")
        sns.distplot(train_values_labels[col], bins = bins, rug=True, hist = hist)
        plt.title('Histogram of ' + col) # Give the plot a main title
        plt.xlabel(col) # Set text for the x axis
        plt.ylabel('gross rent')# Set text for y axis
        plt.show()

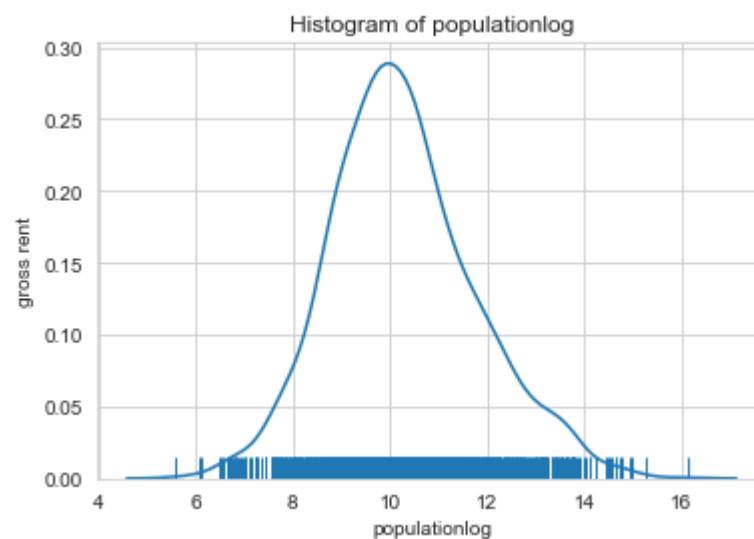
plot_density_hist(train_values_labels, updated_num_cols)
```



In [194]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_density_hist(gross_rent, cols, bins = 10, hist = False):
    for col in cols:
        sns.set_style("whitegrid")
        sns.distplot(train_values_labels[col], bins = bins, rug=True, hist = hist)
        plt.title('Histogram of ' + col) # Give the plot a main title
        plt.xlabel(col) # Set text for the x axis
        plt.ylabel('gross rent')# Set text for y axis
        plt.show()

plot_density_hist(test_values, updated_num_cols)
```

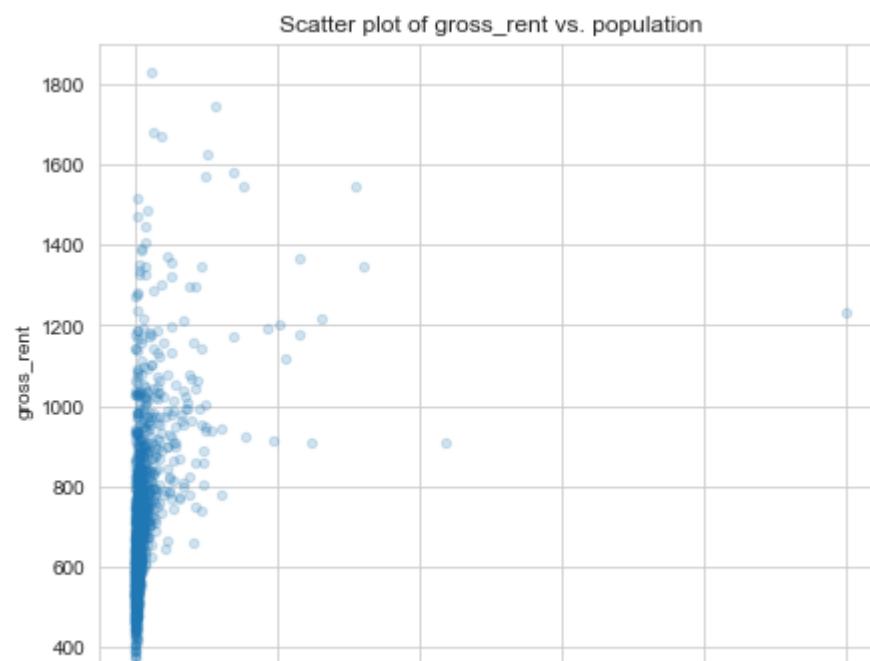


```
In [195]: train_values_labels.columns
```

```
Out[195]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [196]: def plot_scatter(train_values_labels, cols, col_y = 'gross_rent', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
        plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```

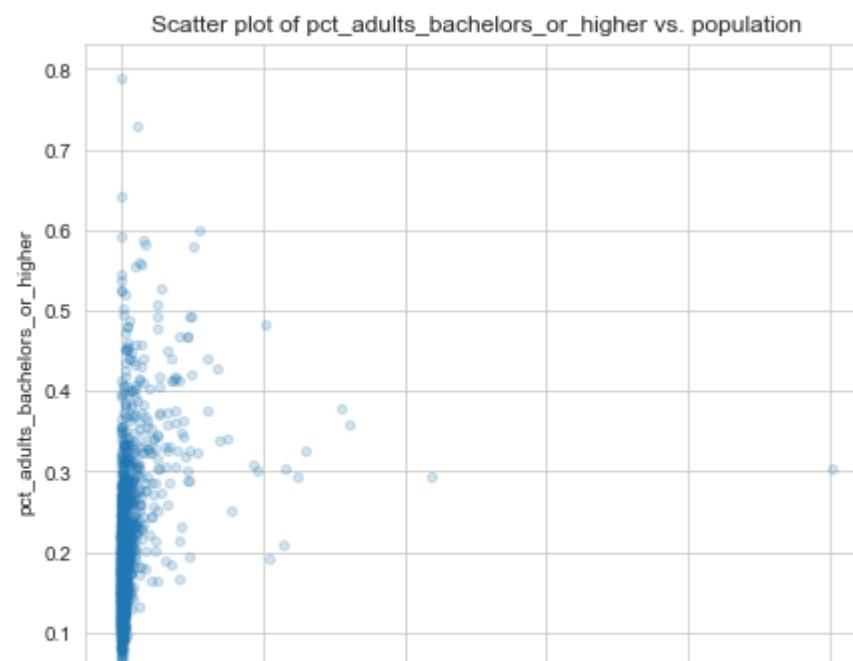


```
In [197]: train_values_labels.columns
```

```
Out[197]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [198]: def plot_scatter(train_values_labels, cols, col_y = 'pct_adults_bachelors_or_higher', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```

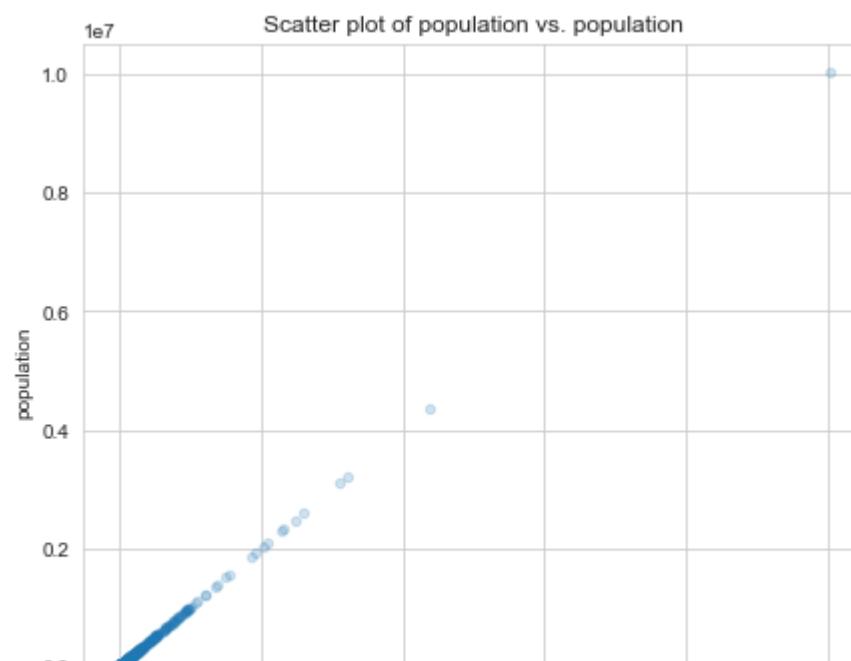


```
In [199]: train_values_labels.columns
```

```
Out[199]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

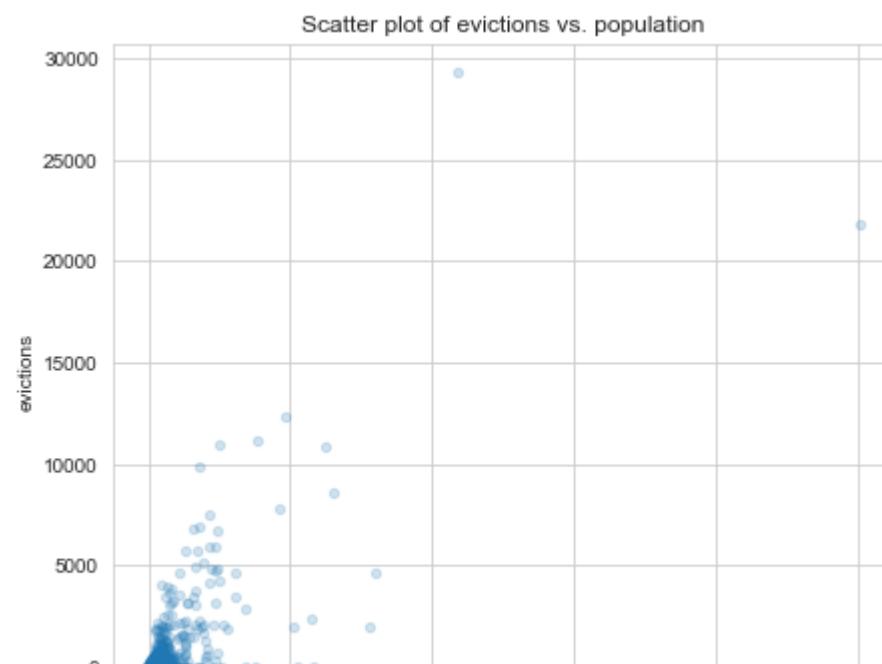
```
In [200]: def plot_scatter(train_values_labels, cols, col_y = 'population', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
        plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



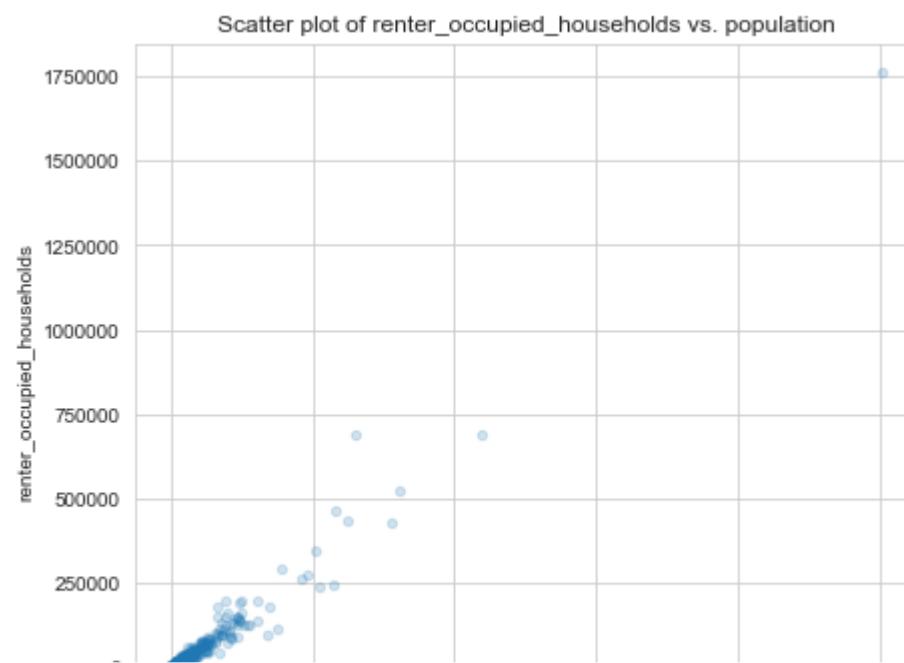
```
In [201]: def plot_scatter(train_values_labels, cols, col_y = 'evictions', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



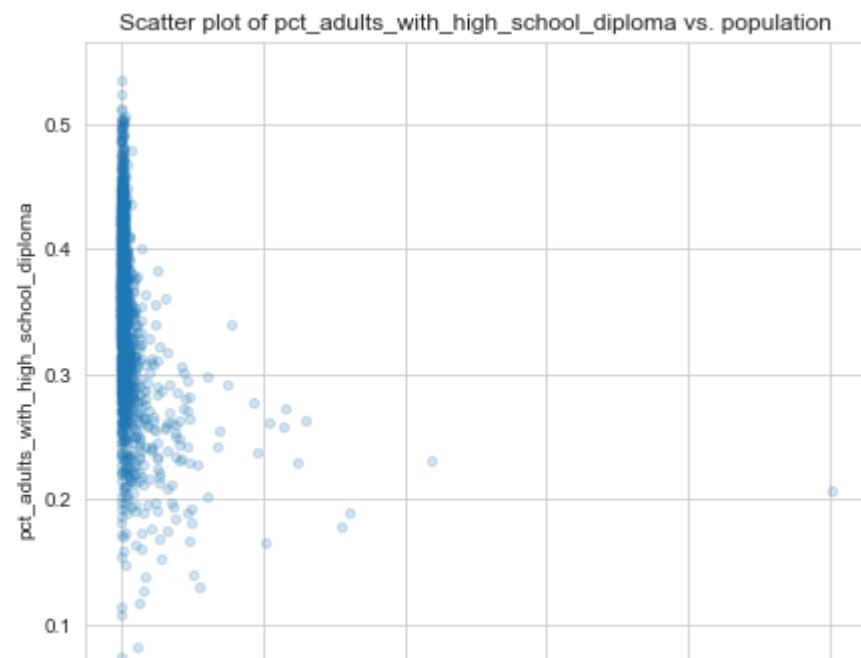
```
In [202]: def plot_scatter(train_values_labels, cols, col_y = 'renter_occupied_households', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



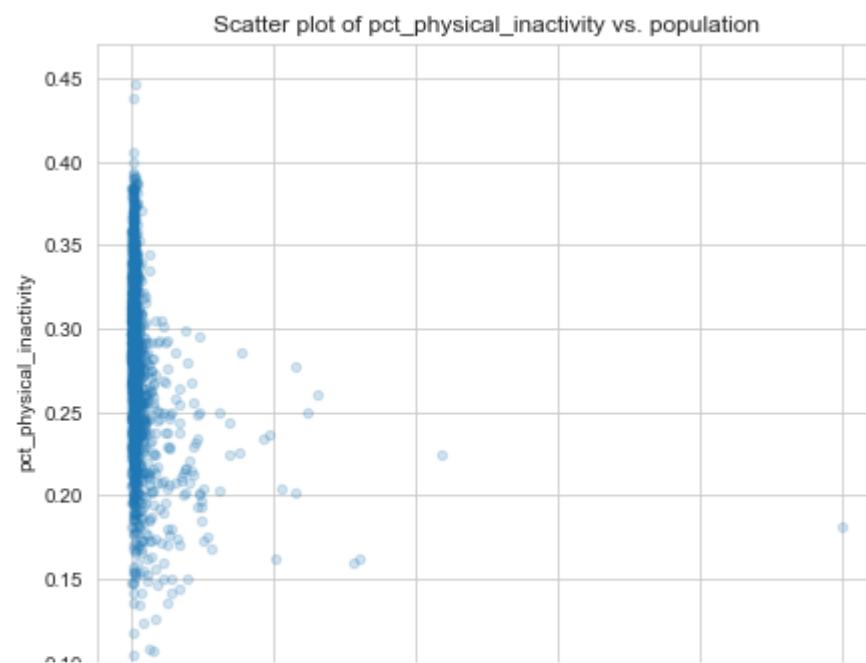
```
In [203]: def plot_scatter(train_values_labels, cols, col_y = 'pct_adults_with_high_school_diploma', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
        plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



```
In [204]: def plot_scatter(train_values_labels, cols, col_y = 'pct_physical_inactivity', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



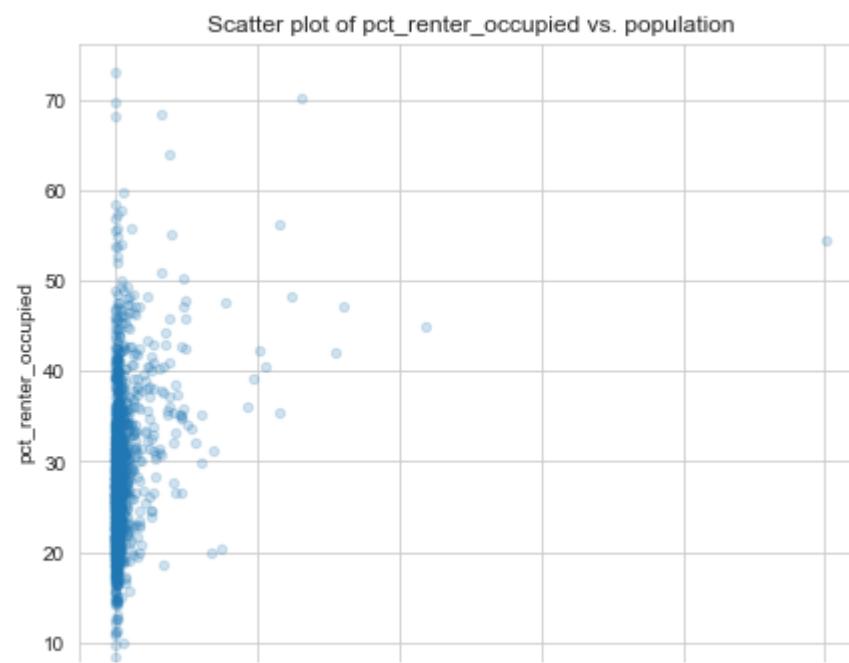
```
In [205]: def plot_scatter(train_values_labels, cols, col_y = 'pct_uninsured_adults', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



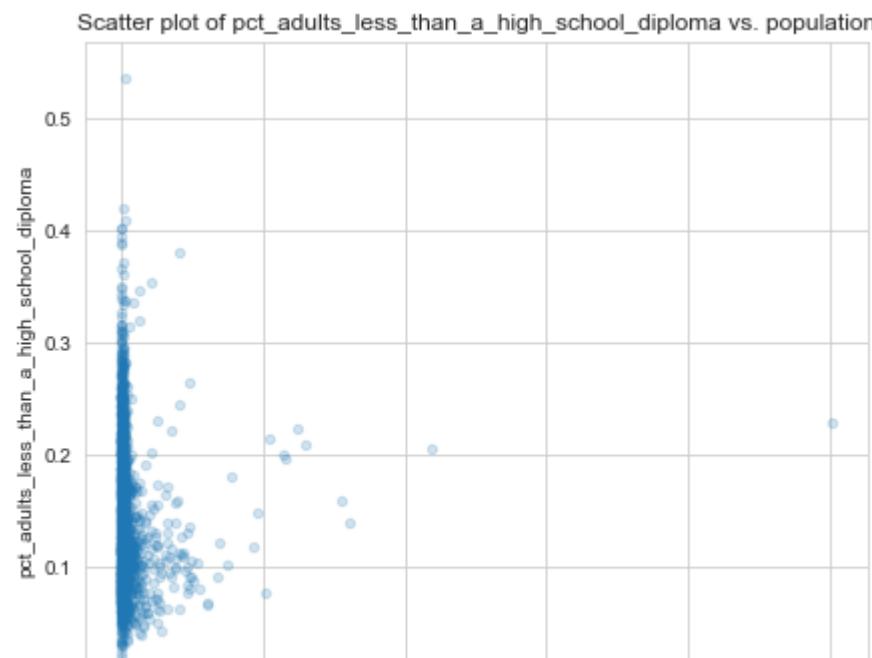
```
In [206]: def plot_scatter(train_values_labels, cols, col_y = 'pct_renter_occupied', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
            # Set text for y axis
    plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



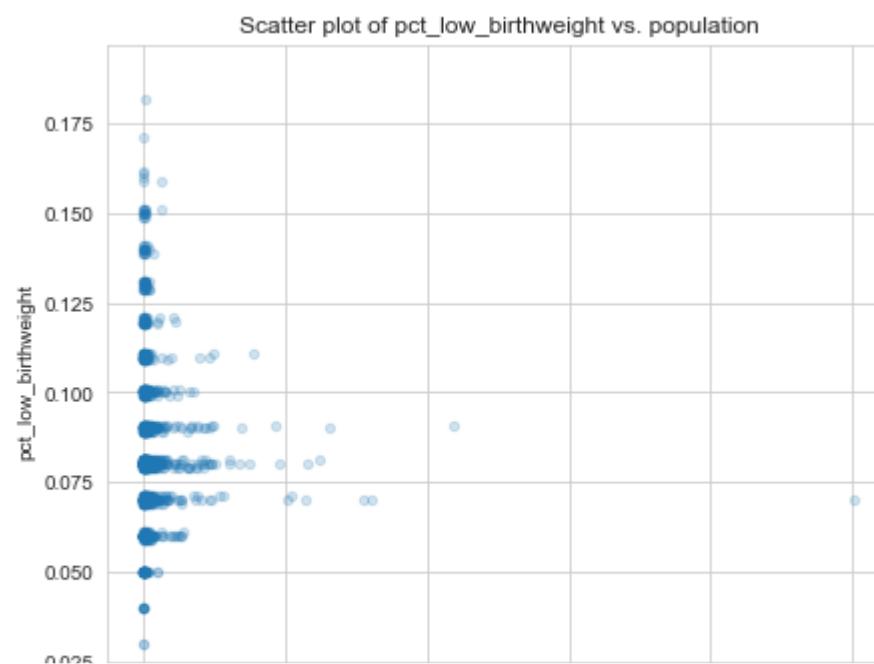
```
In [207]: def plot_scatter(train_values_labels, cols, col_y = 'pct_adults_less_than_a_high_school_diploma', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
        plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



```
In [208]: def plot_scatter(train_values_labels, cols, col_y = 'pct_low_birthweight', alpha = 1.0):
    for col in cols:
        fig = plt.figure(figsize=(7,6)) # define plot area
        ax = fig.gca() # define axis
        train_values_labels.plot.scatter(x = col, y = col_y, ax = ax, alpha = alpha)
        ax.set_title('Scatter plot of ' + col_y + ' vs. ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel(col_y)
        plt.show()

plot_scatter(train_values_labels, num_cols, alpha = 0.2)
```



```
In [209]: train_values_labels[['pct_adults_bachelors_or_higher','motor_vehicle_crash_deaths_per_100k']].corr()
```

Out[209]:

	pct_adults_bachelors_or_higher	motor_vehicle_crash_deaths_per_100k
pct_adults_bachelors_or_higher	1.000000	-0.407155
motor_vehicle_crash_deaths_per_100k	-0.407155	1.000000

```
In [210]: train_values_labels[['pct_adults_bachelors_or_higher','pct_adult_smoking']].corr()
```

Out[210]:

	pct_adults_bachelors_or_higher	pct_adult_smoking
pct_adults_bachelors_or_higher	1.000000	-0.226848
pct_adult_smoking	-0.226848	1.000000

```
In [211]: train_values_labels[['pct_adults_bachelors_or_higher','pct_excessive_drinking']].corr()
```

Out[211]:

	pct_adults_bachelors_or_higher	pct_excessive_drinking
pct_adults_bachelors_or_higher	1.000000	0.367517
pct_excessive_drinking	0.367517	1.000000

```
In [212]: train_values_labels[['pct_adults_bachelors_or_higher','evictions']].corr()
```

Out[212]:

	pct_adults_bachelors_or_higher	evictions
pct_adults_bachelors_or_higher	1.000000	0.239207
evictions	0.239207	1.000000

```
In [213]: train_values_labels[['renter_occupied_households','evictions']].corr()
```

Out[213]:

	renter_occupied_households	evictions
renter_occupied_households	1.000000	0.76267
evictions	0.76267	1.000000

```
In [214]: train_values_labels[['renter_occupied_households','pct_adults_bachelors_or_higher']].corr()
```

Out[214]:

	renter_occupied_households	pct_adults_bachelors_or_higher
renter_occupied_households	1.000000	0.261647
pct_adults_bachelors_or_higher	0.261647	1.000000

```
In [215]: train_values_labels[['renter_occupied_households','population']].corr()
```

Out[215]:

	renter_occupied_households	population
renter_occupied_households	1.000000	0.979015
population	0.979015	1.000000

```
In [216]: train_values_labels[['pct_adults_bachelors_or_higher','population']].corr()
```

Out[216]:

	pct_adults_bachelors_or_higher	population
pct_adults_bachelors_or_higher	1.000000	0.295888
population	0.295888	1.000000

```
In [217]: train_values_labels[['pct_adults_bachelors_or_higher','pct_adults_with_high_school_diploma']].corr()
```

Out[217]:

	pct_adults_bachelors_or_higher	pct_adults_with_high_school_diploma
pct_adults_bachelors_or_higher	1.000000	-0.763686
pct_adults_with_high_school_diploma	-0.763686	1.000000

```
In [218]: train_values_labels[['pct_physical_inactivity','pct_adults_with_high_school_diploma']].corr()
```

Out[218]:

	pct_physical_inactivity	pct_adults_with_high_school_diploma
pct_physical_inactivity	1.000000	0.599897
pct_adults_with_high_school_diploma	0.599897	1.000000

```
In [219]: train_values_labels[['pop_per_dentist','pop_per_primary_care_physician']].corr()
```

Out[219]:

	pop_per_dentist	pop_per_primary_care_physician
pop_per_dentist	1.000000	0.408681
pop_per_primary_care_physician	0.408681	1.000000

```
In [220]: train_values_labels.columns
```

```
Out[220]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [221]: train_values_labels[['pct_adults_less_than_a_high_school_diploma','pct_uninsured_adults']].corr()
```

Out[221]:

	pct_adults_less_than_a_high_school_diploma	pct_uninsured_adults
pct_adults_less_than_a_high_school_diploma	1.000000	0.633243
pct_uninsured_adults	0.633243	1.000000

```
In [222]: cat_cols
```

Out[222]: ['rucc', 'urban_influence', 'economic_typerology']

```
In [223]: #train_values_labels.drop(pollution.index, inplace = True)
```

```
In [224]: smoke = train_values_labels[train_values_labels['pct_adult_smoking']>0.2]
smoke
```

Out[224]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
	2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621
	6	6	279f041	4522abc	22712.0	1901.0	24.602	19.0	29.242	0.929799
	8	8	f11285d	78e8330	2027.0	295.0	55.572	27.0	21.808	0.184899
	10	10	3cc65a1	e74aca3	963541.0	191320.0	50.311	6749.0	32.229	0.528537
	13	13	7516a19	485e9af	244240.0	21112.0	22.737	512.0	31.793	0.795076

	1547	1547	e1f74dd	52acab4	5854.0	628.0	27.376	9.0	27.979	0.710822
	1548	1548	31d2cdb	1646cf6	146816.0	15574.0	25.178	472.0	26.750	0.923420
	1550	1550	6938a5d	09d8cd0	16946.0	1405.0	20.001	5.0	27.990	0.951139
	1554	1554	47d466b	528ea9f	31697.0	4603.0	37.487	105.0	31.634	0.962354
	1555	1555	6e49271	9d0874a	14055.0	729.0	18.999	-1.0	33.118	0.713404

724 rows × 85 columns

```
In [225]: smokecollege = smoke[smoke['pct_adults_bachelors_or_higher']>0.5]
smokecollege
```

Out[225]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am

0 rows × 85 columns

```
In [226]: smokecollege['pct_adult_smoking']
```

Out[226]: Series([], Name: pct_adult_smoking, dtype: float64)

```
In [227]: #train_values_labels.drop(smokecollege.index, inplace = True)
```

```
In [228]: crash = train_values_labels[train_values_labels['motor_vehicle_crash_deaths_per_100k'] == 0]
crash
```

Out[228]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
8	8	f11285d	78e8330	2027.0	295.0	55.572	27.0	21.808	0.184899	0
11	11	170b0b3	fb8cab1	2352.0	277.0	25.940	27.0	21.172	0.965858	0
14	14	5c62775	4cd9667	17882.0	1898.0	25.902	27.0	23.007	0.740954	0
...
1546	1546	151b7a5	2b7da97	7001.0	568.0	20.640	-1.0	33.245	0.936286	0
1547	1547	e1f74dd	52acab4	5854.0	628.0	27.376	9.0	27.979	0.710822	0
1551	1551	f592e2d	842bd12	4239.0	306.0	20.508	-1.0	27.847	0.944484	0
1552	1552	103631f	0f8930b	2205.0	111.0	11.277	0.0	36.995	0.977926	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0

190 rows × 85 columns

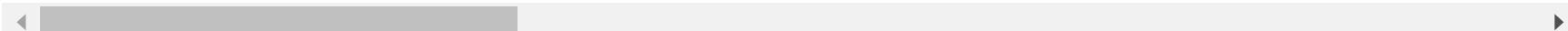


In [229]: `crash.describe()`

Out[229]:

	row_id	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am	pc
count	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	1
mean	795.584211	4478.963158	513.373684	27.497789	10.026316	24.407779	0.796990	0.032960	
std	472.721739	3522.988787	511.260371	10.137707	14.747050	6.052635	0.227426	0.107226	
min	0.000000	269.000000	64.000000	7.279000	-1.000000	12.925000	0.113347	0.000000	
25%	400.500000	2192.000000	222.500000	21.130000	0.000000	19.745500	0.712271	0.000528	
50%	789.500000	3484.500000	369.000000	25.856000	1.000000	23.519000	0.915834	0.003363	
75%	1249.750000	5868.250000	591.500000	30.848500	27.000000	27.775250	0.953105	0.010742	
max	1560.000000	18433.000000	3755.000000	73.008000	92.000000	48.980000	0.995141	0.648357	

8 rows × 80 columns



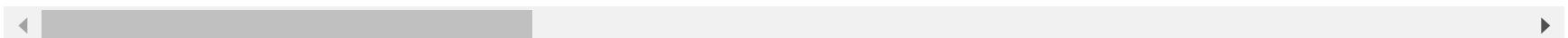
In [230]: `#train_values_labels.drop(crash.index, inplace = True)`

```
In [231]: nomurder = train_values_labels[train_values_labels['homicides_per_100k']==0]
nomurder
```

Out[231]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1555	1555	6e49271	9d0874a	14055.0	729.0	18.999	-1.0	33.118	0.713404	0
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0

949 rows × 85 columns



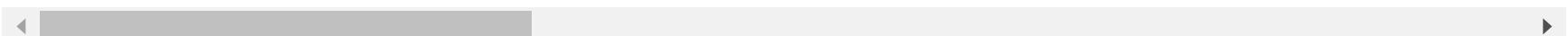
```
In [232]: nomurderzero = nomurder[(nomurder['rent_burden']>27.5) & (nomurder['rent_burden']<31)]  
nomurderzero
```

Out[232]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
	3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835
	6	6	279f041	4522abc	22712.0	1901.0	24.602	19.0	29.242	0.929799
	7	7	83adc69	4c72956	27974.0	3873.0	38.012	49.0	29.966	0.537084
	9	9	e34c58f	9e0007d	17165.0	2223.0	29.769	27.0	28.664	0.961332
	12	12	0921795	1b0d913	7218.0	823.0	26.178	3.0	30.468	0.428118

	1545	1545	e6e9351	3745933	11213.0	1219.0	29.393	7.0	29.246	0.883128
	1547	1547	e1f74dd	52acab4	5854.0	628.0	27.376	9.0	27.979	0.710822
	1550	1550	6938a5d	09d8cd0	16946.0	1405.0	20.001	5.0	27.990	0.951139
	1551	1551	f592e2d	842bd12	4239.0	306.0	20.508	-1.0	27.847	0.944484
	1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936

267 rows × 85 columns



```
In [233]: train_values_labels.columns
```

```
Out[233]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [234]: nosmoking = train_values_labels[train_values_labels['pct_adult_smoking']==0]
nosmoking
```

Out[234]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
5	5	abf992b	4522abc	8540.0	751.0	22.543	0.0	34.937	0.560637	0
12	12	0921795	1b0d913	7218.0	823.0	26.178	3.0	30.468	0.428118	0
29	29	a0b1eb5	08f8fb4	9472.0	869.0	22.716	3.0	33.620	0.959306	0
37	37	cef221b	e2f94fa	6179.0	593.0	21.550	1.0	21.684	0.917266	0
38	38	9d68ed9	9e0007d	7940.0	1028.0	34.115	27.0	26.498	0.537051	0
...
1536	1536	718b409	1b0d913	3613.0	330.0	31.340	2.0	15.847	0.307728	0
1541	1541	a1bf7f3	9d1e27d	13179.0	1331.0	20.516	0.0	27.429	0.919745	0
1543	1543	558cad5	08f8fb4	12117.0	1403.0	26.491	9.0	28.708	0.932306	0
1552	1552	103631f	0f8930b	2205.0	111.0	11.277	0.0	36.995	0.977926	0
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0

218 rows × 85 columns

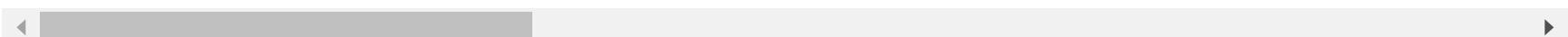


In [235]: `nodrinking = train_values_labels[train_values_labels['pct_excessive_drinking'] == 0]`
`nodrinking`

Out[235]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
5	5	abf992b	4522abc	8540.0	751.0	22.543	0.0	34.937	0.560637	0
9	9	e34c58f	9e0007d	17165.0	2223.0	29.769	27.0	28.664	0.961332	0
12	12	0921795	1b0d913	7218.0	823.0	26.178	3.0	30.468	0.428118	0
24	24	0a4702d	e899d7f	13429.0	1034.0	18.362	10.0	32.843	0.557809	0
29	29	a0b1eb5	08f8fb4	9472.0	869.0	22.716	3.0	33.620	0.959306	0
...
1551	1551	f592e2d	842bd12	4239.0	306.0	20.508	-1.0	27.847	0.944484	0
1552	1552	103631f	0f8930b	2205.0	111.0	11.277	0.0	36.995	0.977926	0
1554	1554	47d466b	528ea9f	31697.0	4603.0	37.487	105.0	31.634	0.962354	0
1555	1555	6e49271	9d0874a	14055.0	729.0	18.999	-1.0	33.118	0.713404	0
1557	1557	8a15c79	a952566	18983.0	1480.0	22.621	41.0	26.905	0.847652	0

462 rows × 85 columns



In [236]: `#train_values_labels.drop(nomurder.index, inplace = True)`

In [237]: `#train_values_labels.drop(nosmoking.index, inplace = True)`

In [238]: `#train_values_labels.drop(nodrinking.index, inplace = True)`

In [239]:

```
murdercrashzero = train_values_labels[(train_values_labels['motor_vehicle_crash_deaths_per_100k'] == 0)&(train_values_labels['motor_vehicle_crash_deaths_per_100k'].isna())]
murdercrashzero
```

Out[239]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
8	8	f11285d	78e8330	2027.0	295.0	55.572	27.0	21.808	0.184899	0
11	11	170b0b3	fb8cab1	2352.0	277.0	25.940	27.0	21.172	0.965858	0
14	14	5c62775	4cd9667	17882.0	1898.0	25.902	27.0	23.007	0.740954	0
...
1546	1546	151b7a5	2b7da97	7001.0	568.0	20.640	-1.0	33.245	0.936286	0
1547	1547	e1f74dd	52acab4	5854.0	628.0	27.376	9.0	27.979	0.710822	0
1551	1551	f592e2d	842bd12	4239.0	306.0	20.508	-1.0	27.847	0.944484	0
1552	1552	103631f	0f8930b	2205.0	111.0	11.277	0.0	36.995	0.977926	0
1560	1560	ff00193	4cd9667	4698.0	402.0	20.107	1.0	48.980	0.173909	0

189 rows × 85 columns

In [240]:

```
annoyingrow = train_values_labels[train_values_labels['row_id']==368]
annoyingrow
```

Out[240]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_
368	368	827ea49	1b0d913	16604.0	1957.0	29.049	9.0	33.622	0.58366	0

1 rows × 85 columns

In [241]:

```
#train_values_labels.drop(annoyingrow.index, inplace = True)
```

```
In [242]: #train_values_labels.drop(murdercrashzero.index, inplace = True)
```

```
In [243]: drinkingmedian = train_values_labels[(train_values_labels['pct_excessive_drinking'] > .15) & (train_values_labels['pct_excessive_drinking'] < .25)]
drinkingmedian
```

Out[243]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
7	7	83adc69	4c72956	27974.0	3873.0	38.012	49.0	29.966	0.537084	0
17	17	d5b0571	1646cf6	87762.0	11465.0	29.449	112.0	32.547	0.939076	0
20	20	58d1c1f	a952566	33020.0	4079.0	29.231	114.0	27.552	0.609131	0
28	28	f34b543	20d32fc	39788.0	5457.0	37.846	27.0	35.006	0.546908	0
...
1542	1542	24045b9	8036085	154197.0	21373.0	31.468	306.0	28.734	0.873540	0
1544	1544	0081051	78e8330	3218.0	351.0	38.213	27.0	17.891	0.451239	0
1553	1553	88faf13	09d8cd0	79274.0	9225.0	29.021	227.0	24.510	0.849798	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0

192 rows × 85 columns

In [244]:

```
drinkingmediantwo = train_values_labels[(train_values_labels['pct_excessive_drinking'] > .15)]
drinkingmediantwo
```

Out[244]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct
0	0	8e686a7	fb8cab1	3876.0	408.0	24.583	27.0	18.380	0.945945	0
1	1	d1b5fc5	842bd12	10224.0	1166.0	28.346	3.0	26.694	0.808959	0
2	2	19a463b	2b7da97	27023.0	2927.0	21.641	9.0	31.028	0.956621	0
3	3	1711ab7	5029ed4	8735.0	1039.0	23.110	0.0	27.734	0.894835	0
4	4	1eb4681	b795815	3681.0	365.0	21.985	2.0	19.673	0.923886	0
...
1548	1548	31d2cdb	1646cf6	146816.0	15574.0	25.178	472.0	26.750	0.923420	0
1549	1549	3a3a857	e74aca3	86559.0	8481.0	23.459	46.0	25.631	0.925444	0
1553	1553	88faf13	09d8cd0	79274.0	9225.0	29.021	227.0	24.510	0.849798	0
1558	1558	ffaffbd	e2f94fa	18837.0	2144.0	28.649	7.0	26.515	0.925500	0
1559	1559	4268c79	a952566	77224.0	4677.0	18.928	137.0	28.827	0.883936	0

642 rows × 85 columns



In [245]:

```
eldronkodos = drinkingmedian[drinkingmedian['pop_per_dentist'] > 8000]
eldronkodos
```

Out[245]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am
0									

0 rows × 85 columns

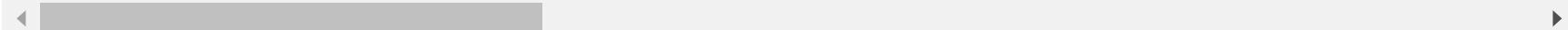


```
In [246]: eldronkotres = drinkingmedian[two[drinkingmedian['pop_per_primary_care_physician'] > 10000]]  
eldronkotres
```

Out[246]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am
--------	-------------	-------	------------	----------------------------	---------------------	-----------	-------------	-----------	-----------

0 rows × 85 columns



```
In [247]: #train_values_labels.drop(eldronkodos.index, inplace = True)
```

```
In [248]: #train_values_labels.drop(eldronkotres.index, inplace = True)
```

```
In [249]: train_values_labels['population'].describe()
```

```
Out[249]: count    1.562000e+03  
mean     1.083407e+05  
std      3.745229e+05  
min      2.690000e+02  
25%     1.045275e+04  
50%     2.528200e+04  
75%     6.836150e+04  
max      1.002029e+07  
Name: population, dtype: float64
```

```
In [250]: train_values_labels['evictions'].describe()
```

```
Out[250]: count    1562.000000  
mean     319.866197  
std      1362.305841  
min     -1.000000  
25%      6.000000  
50%     27.000000  
75%     99.000000  
max     29251.000000  
Name: evictions, dtype: float64
```

```
In [251]: train_values_labels[train_values_labels['renter_occupied_households'] > 1750000]
```

Out[251]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white
953	953	454aabf	4c72956	10020287.0	1760277.0	54.552	21783.0	34.916

1 rows × 85 columns

```
In [252]: outliers = train_values_labels[train_values_labels['evictions'] > 20000]
outliers
```

Out[252]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_
70	70	6c5ae9d	1b0d913	4371311.0	688268.0	44.981	29251.0	29.758	0.317401
953	953	454aabf	4c72956	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376

2 rows × 85 columns

```
In [253]: outliers.describe()
```

Out[253]:

	row_id	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_am	pct
count	2.000000	2.000000e+00		2.000000e+00	2.000000	2.000000	2.000000	2.000000	2.000000
mean	511.500000	7.195799e+06		1.224272e+06	49.766500	25517.000000	32.337000	0.293388	0.131504
std	624.375288	3.994429e+06		7.580248e+05	6.767719	5280.673442	3.647257	0.033959	0.073351
min	70.000000	4.371311e+06		6.882680e+05	44.981000	21783.000000	29.758000	0.269376	0.079637
25%	290.750000	5.783555e+06		9.562702e+05	47.373750	23650.000000	31.047500	0.281382	0.105571
50%	511.500000	7.195799e+06		1.224272e+06	49.766500	25517.000000	32.337000	0.293388	0.131504
75%	732.250000	8.608043e+06		1.492275e+06	52.159250	27384.000000	33.626500	0.305395	0.157437
max	953.000000	1.002029e+07		1.760277e+06	54.552000	29251.000000	34.916000	0.317401	0.183371

8 rows × 80 columns

```
In [254]: #train_values_labels.drop(outliers.index, inplace = True)
```

```
In [255]: #train_values_labels.replace(outliers,(train_values_labels.mean()+2*np.std(train_values_labels)),inplace=True)
```

```
In [256]: def count_unique(train_values_labels, cols):
    for col in cols:
        print('\n' + 'For column ' + col)
        print(train_values_labels[col].value_counts())
    count_unique(train_values_labels, cat_cols)
```

For column rucc

Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area	301
Metro - Counties in metro areas of 1 million population or more	219
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area	215
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area	206
Metro - Counties in metro areas of 250,000 to 1 million population	193
Metro - Counties in metro areas of fewer than 250,000 population	165
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area	120
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area	101
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area	42

Name: rucc, dtype: int64

For column urban_influence

Small-in a metro area with fewer than 1 million residents	358
Large-in a metro area with at least 1 million residents or more	219
Noncore adjacent to a small metro with town of at least 2,500 residents	178
Micropolitan not adjacent to a metro area	131
Micropolitan adjacent to a small metro area	113
Noncore adjacent to micro area and does not contain a town of at least 2,500 residents	97
Noncore adjacent to a small metro and does not contain a town of at least 2,500 residents	88
Noncore not adjacent to a metro/micro area and does not contain a town of at least 2,500 residents	86
Noncore adjacent to micro area and contains a town of 2,500-19,999 residents	86
Noncore adjacent to a large metro area	79
Micropolitan adjacent to a large metro area	64
Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents	63

Name: urban_influence, dtype: int64

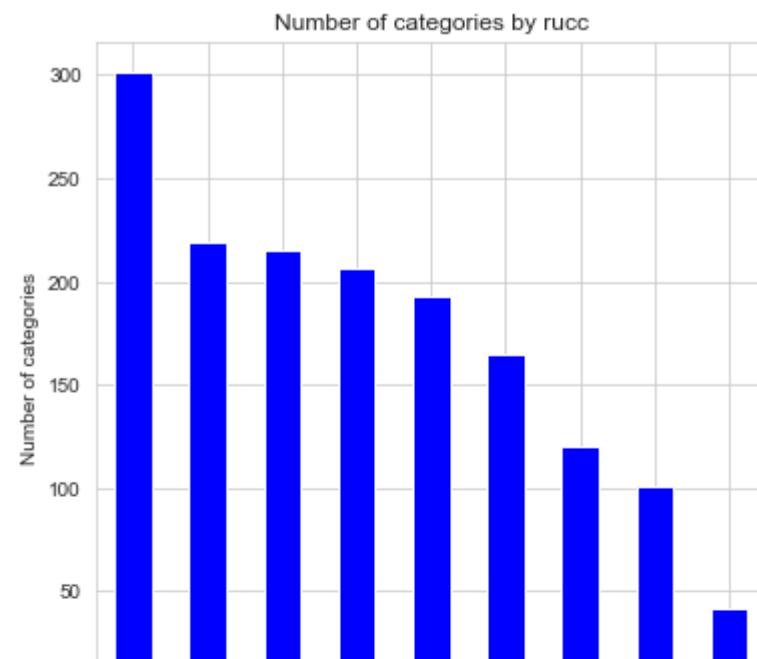
For column economic_t typology

Nonspecialized	631
Manufacturing-dependent	244
Farm-dependent	217
Federal/State government-dependent	191
Recreation	166

Mining-dependent 113
Name: economic_typerology, dtype: int64

```
In [257]: def plot_bars(dataset, cols):
    for col in cols:
        fig = plt.figure(figsize=(6,6)) # define plot area
        ax = fig.gca() # define axis
        counts = dataset[col].value_counts() # find the counts for each unique category
        counts.plot.bar(ax = ax, color = 'blue') # Use the plot.bar method on the counts data frame
        ax.set_title('Number of categories by ' + col) # Give the plot a main title
        ax.set_xlabel(col) # Set text for the x axis
        ax.set_ylabel('Number of categories')# Set text for y axis
        plt.show()

plot_bars(train_values_labels, cat_cols)
```



```
In [258]: def factorize_col(cols, df):
    for col in df[cols]:
        if(df[col].dtype == 'object'):
            df[col]= df[col].astype('category')
            df[col] = df[col].cat.codes
    cols = ['state','county_code']
    factorize_col(cols,train_values_labels)
train_values_labels
```

Out[258]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
0	0	869	49	3876.0	408.0	24.583	27.0	18.380	0.945945	0.010
1	1	1271	29	10224.0	1166.0	28.346	3.0	26.694	0.808959	0.000
2	2	158	10	27023.0	2927.0	21.641	9.0	31.028	0.956621	0.000
3	3	144	18	8735.0	1039.0	23.110	0.0	27.734	0.894835	0.000
4	4	183	38	3681.0	365.0	21.985	2.0	19.673	0.923886	0.000
...
1557	1557	840	36	18983.0	1480.0	22.621	41.0	26.905	0.847652	0.060
1558	1558	1560	45	18837.0	2144.0	28.649	7.0	26.515	0.925500	0.010
1559	1559	401	36	77224.0	4677.0	18.928	137.0	28.827	0.883936	0.070
1560	1560	1557	17	4698.0	402.0	20.107	1.0	48.980	0.173909	0.000
1561	1561	344	8	650813.0	92680.0	37.789	27.0	30.687	0.704111	0.050

1562 rows × 85 columns



```
In [259]: def factorize_col(cols, df):
    for col in df[cols]:
        if(df[col].dtype == 'object'):
            df[col]= df[col].astype('category')
            df[col] = df[col].cat.codes
    cols = ['state','county_code']
    factorize_col(cols,test_values)
test_values
```

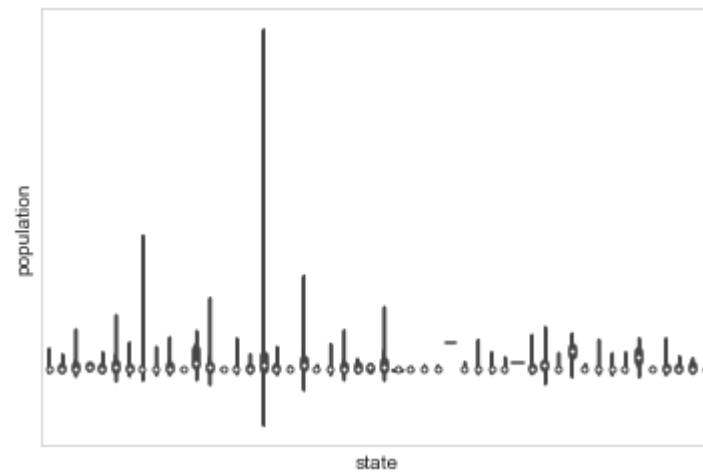
Out[259]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
0	0	873	32	52842.0	5403.0	26.840	27.0	27.960	0.924234	0
1	1	675	37	212287.0	53502.0	57.534	6032.0	33.072	0.398318	0
2	2	1100	9	81263.0	13368.0	39.994	1012.0	32.044	0.483789	0
3	3	1562	7	122870.0	19359.0	41.865	27.0	30.724	0.468043	0
4	4	379	23	146153.0	15766.0	30.681	644.0	30.860	0.651511	0
...
1571	1571	1347	21	44414.0	6275.0	35.026	254.0	30.569	0.726121	0
1572	1572	1130	15	16160.0	1871.0	31.922	27.0	33.899	0.663196	0
1573	1573	1496	8	28688.0	4478.0	33.897	27.0	28.644	0.809853	0
1574	1574	1217	21	29234.0	4528.0	40.479	57.0	24.803	0.707838	0
1575	1575	1175	26	4036.0	327.0	18.052	27.0	22.122	0.967377	0

1576 rows × 83 columns

```
In [260]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'state'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.xticks([])
        plt.yticks([])
        plt.show()

plot_violin(train_values_labels, num_cols)
```



```
In [261]: #import seaborn as sns
#import matplotlib.pyplot as plt
#%matplotlib inline
#def plot_violin(train_values_labels, cols, col_x = 'county_code'):
#    for col in cols:
#        sns.set_style("whitegrid")
#        sns.violinplot(col_x, col, data=train_values_labels)
#        plt.xlabel(col_x) # Set text for the x axis
#        plt.ylabel(col)# Set text for y axis
#        plt.show()
#
#plot_violin(train_values_labels, num_cols)
```

```
In [262]: #import seaborn as sns
#import matplotlib.pyplot as plt
#%matplotlib inline
#def plot_violin(train_values_labels, cols, col_x = 'gross_rent'):
#    for col in cols:
#        sns.set_style("whitegrid")
#        sns.violinplot(col_x, col, data=train_values_labels)
#        plt.xlabel(col_x) # Set text for the x axis
#        plt.ylabel(col)# Set text for y axis
#        plt.show()
#
#plot_violin(train_values_labels, num_cols)
```

```
In [263]: statethirty=train_values_labels[train_values_labels['state']==30]
statethirty
```

Out[263]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	1443	1443	909	30	974468.0	140785.0	45.773	735.0	33.905	0.193638

1 rows × 85 columns



```
In [264]: statethirty['gross_rent']
```

```
Out[264]: 1443    1569  
Name: gross_rent, dtype: int64
```

```
In [265]: statethirty['homicides_per_100k'].describe()
```

```
Out[265]: count      1.0  
mean       1.0  
std        NaN  
min        1.0  
25%        1.0  
50%        1.0  
75%        1.0  
max        1.0  
Name: homicides_per_100k, dtype: float64
```

```
In [266]: train_values_labels.drop(statethirty.index, inplace = True)
```

```
In [267]: statethirtyfive=train_values_labels[train_values_labels['state']==35]  
statethirtyfive
```

```
Out[267]:
```

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_
483	483	27	35	207627.0	18978.0	22.449	953.0	29.387	0.748492	0.123

1 rows × 85 columns



```
In [268]: statethirtyfive['homicides_per_100k'].describe()
```

```
Out[268]: count    1.00
mean     2.99
std      NaN
min     2.99
25%     2.99
50%     2.99
75%     2.99
max     2.99
Name: homicides_per_100k, dtype: float64
```

```
In [269]: train_values_labels.drop(statethirtyfive.index, inplace = True)
```

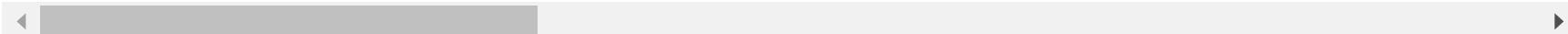
```
In [270]: statetwentysix=train_values_labels[train_values_labels['state']==26]
statetwentysix
```

Out[270]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	8	8	1466	26	2027.0	295.0	55.572	27.0	21.808	0.184899
	108	108	768	26	4262.0	510.0	31.050	27.0	28.175	0.884318
	221	221	567	26	10246.0	1203.0	25.236	27.0	29.159	0.917340
	304	304	1061	26	5585.0	607.0	27.046	27.0	29.263	0.869725
	305	305	1375	26	6841.0	1020.0	28.936	27.0	27.266	0.856343
	330	330	1024	26	1403.0	152.0	18.238	27.0	16.153	0.876144
	375	375	483	26	24467.0	4017.0	32.892	27.0	26.385	0.913017
	424	424	1472	26	2725.0	269.0	21.617	27.0	20.837	0.900433
	561	561	735	26	22460.0	2839.0	32.791	27.0	25.646	0.902089
	562	562	604	26	7010.0	567.0	21.648	27.0	24.865	0.880201
	563	563	797	26	7173.0	636.0	24.538	27.0	29.060	0.954706
	641	641	1348	26	1559.0	115.0	19.647	27.0	22.846	0.948433
	656	656	611	26	5154.0	528.0	23.268	27.0	19.873	0.953383
	663	663	240	26	2833.0	425.0	46.789	27.0	29.458	0.235613
	681	681	1514	26	2086.0	231.0	35.248	27.0	30.406	0.396084
	697	697	1491	26	3396.0	399.0	29.851	27.0	21.881	0.974314
	820	820	165	26	1464.0	137.0	31.008	27.0	13.128	0.976257
	844	844	906	26	2248.0	195.0	25.472	27.0	16.761	0.914089
	858	858	968	26	788.0	129.0	31.468	27.0	22.095	0.942399
	961	961	1292	26	6455.0	669.0	28.251	27.0	19.964	0.793214
	1032	1032	1238	26	6606.0	675.0	26.359	27.0	21.504	0.950673
	1116	1116	1306	26	2305.0	195.0	19.050	27.0	19.227	0.926745
	1147	1147	422	26	2965.0	335.0	28.114	27.0	25.237	0.956002
	1260	1260	110	26	4180.0	583.0	48.848	27.0	21.958	0.308533

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
1315	1315	1380	26	5967.0	388.0	19.926	27.0	23.479	0.953199	0.00
1339	1339	221	26	3364.0	145.0	16.545	27.0	25.168	0.926980	0.03
1353	1353	1325	26	8355.0	863.0	18.550	27.0	24.438	0.911314	0.00
1422	1422	1208	26	2948.0	285.0	19.222	27.0	25.676	0.891379	0.00
1439	1439	1414	26	2327.0	232.0	23.227	27.0	19.075	0.947020	0.00
1462	1462	119	26	4666.0	490.0	27.747	27.0	18.485	0.833189	0.00
1505	1505	879	26	19941.0	3320.0	37.320	27.0	25.176	0.924782	0.00
1510	1510	506	26	33217.0	5571.0	40.072	27.0	30.070	0.912488	0.01
1544	1544	3	26	3218.0	351.0	38.213	27.0	17.891	0.451239	0.00

33 rows × 85 columns



In [271]: `statetwentysix['gross_rent'].describe()`

Out[271]:

count	33.000000
mean	536.333333
std	87.719962
min	391.000000
25%	476.000000
50%	536.000000
75%	589.000000
max	796.000000

Name: gross_rent, dtype: float64

```
In [272]: statetwentysix['homicides_per_100k'].describe()
```

```
Out[272]: count    33.0
mean     0.0
std      0.0
min     0.0
25%     0.0
50%     0.0
75%     0.0
max     0.0
Name: homicides_per_100k, dtype: float64
```

```
In [273]: statetwentysix.describe()
```

```
Out[273]:
```

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_whit
count	33.000000	33.000000	33.0	33.000000	33.000000	33.000000	33.0	33.000000	33.000000
mean	849.272727	879.333333	26.0	6673.969697	859.878788	28.901788	27.0	23.406455	0.82322
std	461.710764	481.070010	0.0	7479.165296	1243.055805	9.280357	0.0	4.457049	0.22489
min	8.000000	3.000000	26.0	788.000000	115.000000	16.545000	27.0	13.128000	0.18489
25%	561.000000	506.000000	26.0	2327.000000	232.000000	21.648000	27.0	19.964000	0.86972
50%	820.000000	906.000000	26.0	4180.000000	425.000000	27.747000	27.0	23.479000	0.91248
75%	1315.000000	1325.000000	26.0	6841.000000	669.000000	32.791000	27.0	26.385000	0.94702
max	1544.000000	1514.000000	26.0	33217.000000	5571.000000	55.572000	27.0	30.406000	0.97625

8 rows × 82 columns



In [274]:

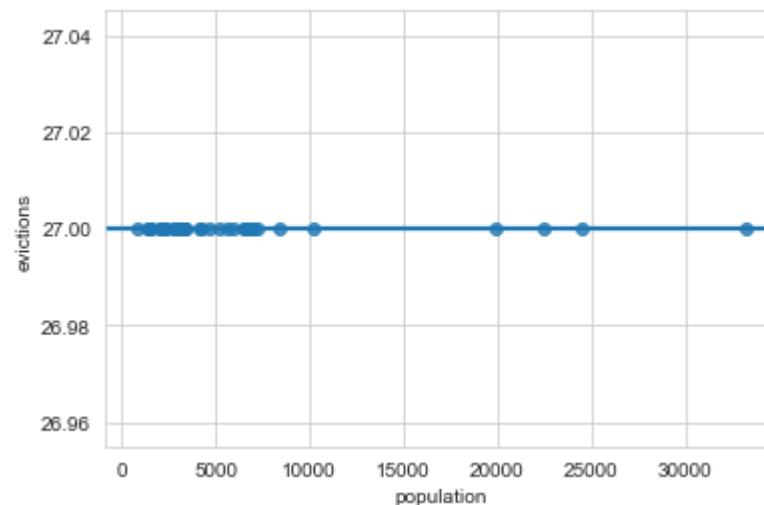
```
for col in num_cols:  
    print(col, '\n', statetwentysix[col].describe(),'\n' )
```

population
count 33.000000
mean 6673.969697
std 7479.165296
min 788.000000
25% 2327.000000
50% 4180.000000
75% 6841.000000
max 33217.000000
Name: population, dtype: float64

renter_occupied_households
count 33.000000
mean 859.878788
std 1243.055805
min 115.000000
25% 232.000000
50% 425.000000
75% 669.000000

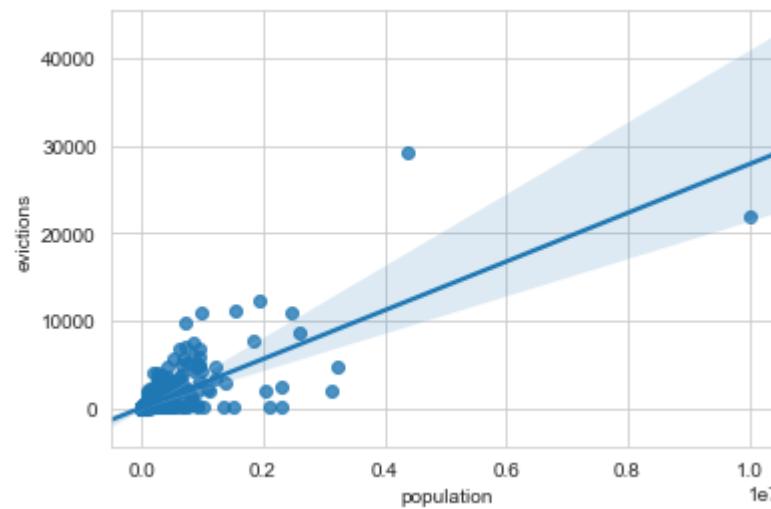
```
In [275]: import seaborn as sns
```

```
ax = sns.regplot(x='population', y='evictions', data=statetwentysix)
```



```
In [276]: import seaborn as sns
```

```
ax = sns.regplot(x='population', y='evictions', data=train_values_labels)
```



```
In [277]: train_values_labels.drop(statetwentysix.index, inplace = True)
```

```
In [278]: train_values_labels[['evictions', 'population']].corr()
```

Out[278]:

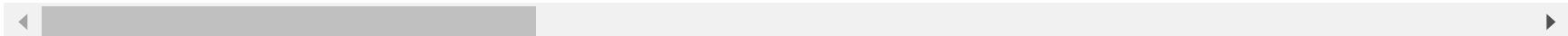
	evictions	population
evictions	1.000000	0.763799
population	0.763799	1.000000

In [279]: `statetwentyseven=train_values_labels[train_values_labels['state']==27]`
`statetwentyseven`

Out[279]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
35	35	1143	27	3302.0	276.0	46.884	27.0	22.064	0.149208	0.09
182	182	1354	27	14005.0	2104.0	40.190	27.0	29.683	0.516275	0.00
190	190	47	27	5694.0	840.0	68.336	27.0	19.237	0.278452	0.04
231	231	315	27	6360.0	693.0	28.388	27.0	23.090	0.460225	0.00
243	243	1410	27	961.0	207.0	45.837	27.0	21.259	0.532234	0.00
331	331	788	27	100623.0	16090.0	42.227	27.0	30.838	0.719742	0.04
372	372	1003	27	13709.0	2327.0	41.394	27.0	28.484	0.649735	0.00
481	481	1478	27	2566.0	336.0	30.712	27.0	23.033	0.793519	0.00
825	825	272	27	9689.0	1372.0	29.342	27.0	19.720	0.701697	0.00
849	849	636	27	9811.0	1338.0	43.685	27.0	25.904	0.164267	0.00
891	891	727	27	6981.0	857.0	29.689	27.0	24.663	0.766605	0.01
897	897	1286	27	8901.0	1604.0	41.129	27.0	27.644	0.627646	0.01
1009	1009	347	27	5684.0	653.0	29.133	27.0	22.985	0.217068	0.00
1196	1196	390	27	9575.0	1117.0	46.110	27.0	17.865	0.321428	0.00
1266	1266	722	27	7761.0	980.0	44.728	27.0	18.372	0.118455	0.00
1389	1389	951	27	17676.0	2048.0	35.446	27.0	24.938	0.113347	0.00
1470	1470	818	27	643.0	120.0	45.318	27.0	25.048	0.396316	0.01
1478	1478	1443	27	2118.0	299.0	33.943	27.0	22.104	0.495419	0.02

18 rows × 85 columns



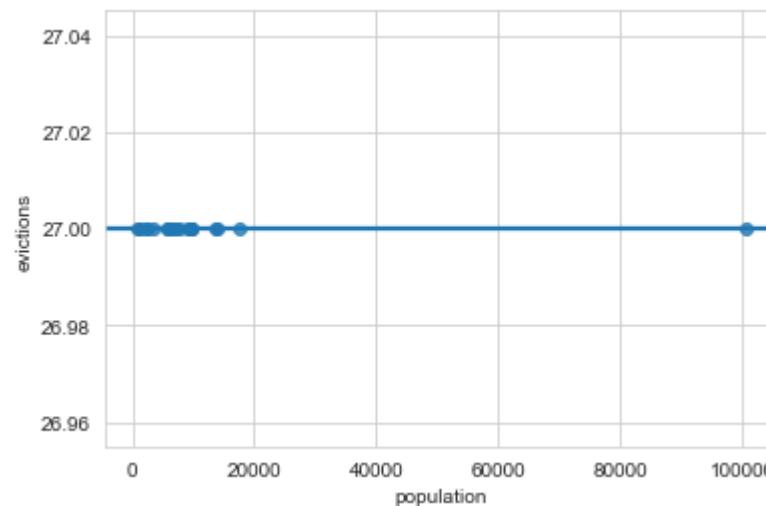
```
In [280]: statetwentyseven['gross_rent'].describe()
```

```
Out[280]: count      18.000000
mean      1015.111111
std       174.028358
min       638.000000
25%      933.750000
50%      1045.000000
75%      1162.000000
max      1270.000000
Name: gross_rent, dtype: float64
```

```
In [281]: statetwentyseven['homicides_per_100k'].describe()
```

```
Out[281]: count      18.000000
mean      0.927222
std       3.096145
min       0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      12.770000
Name: homicides_per_100k, dtype: float64
```

```
In [282]: import seaborn as sns  
  
ax = sns.regplot(x='population', y='evictions', data=statetwentyseven)
```



```
In [283]: train_values_labels.drop(statetwentyseven.index, inplace = True)
```

In [284]:

```
statethirtyfour=train_values_labels[train_values_labels['state']==34]
statethirtyfour
```

Out[284]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
	9	9	1383	34	17165.0	2223.0	29.769	27.0	28.664	0.961332
	38	38	966	34	7940.0	1028.0	34.115	27.0	26.498	0.537051
	50	50	942	34	11143.0	1188.0	24.243	27.0	26.851	0.877104
	78	78	1338	34	16927.0	1699.0	22.231	27.0	33.343	0.940959
	155	155	1517	34	27345.0	4149.0	44.735	27.0	29.671	0.410982
	174	174	1170	34	22015.0	2305.0	26.207	27.0	27.697	0.915777
	199	199	1036	34	70385.0	7290.0	29.398	27.0	26.372	0.871526
	236	236	250	34	18173.0	1670.0	22.874	27.0	23.359	0.930232
	249	249	892	34	9126.0	824.0	20.305	27.0	28.605	0.919109
	351	351	1552	34	24328.0	3372.0	31.905	27.0	31.827	0.592844
	353	353	1399	34	78396.0	10009.0	31.504	27.0	29.667	0.885746
	409	409	1346	34	9900.0	1449.0	42.766	27.0	32.812	0.408769
	412	412	1461	34	40955.0	4705.0	24.472	27.0	29.017	0.953221
	446	446	1551	34	5211.0	530.0	19.816	27.0	30.930	0.730539
	493	493	196	34	12262.0	1124.0	24.117	27.0	29.260	0.958999
	714	714	464	34	21237.0	2517.0	24.426	27.0	29.789	0.681511
	759	759	90	34	10338.0	911.0	20.040	27.0	24.423	0.926133
	822	822	936	34	15638.0	1454.0	23.885	27.0	29.613	0.909374
	829	829	237	34	391432.0	68529.0	39.724	27.0	30.296	0.543865
	850	850	1509	34	22574.0	3295.0	36.543	27.0	30.711	0.697391
	884	884	969	34	33414.0	3240.0	28.167	27.0	32.812	0.828885
	937	937	454	34	21724.0	2598.0	32.368	27.0	25.587	0.762080
	967	967	1177	34	240270.0	29497.0	33.077	27.0	24.074	0.749303
	970	970	1322	34	18699.0	2844.0	33.518	27.0	27.786	0.700164

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
991	991	1304	34	97454.0	13505.0	32.512	27.0	32.835	0.829915	0
1008	1008	563	34	13569.0	1692.0	34.411	27.0	26.059	0.659227	0
1012	1012	389	34	11344.0	1678.0	32.157	27.0	31.595	0.396128	0
1207	1207	195	34	43429.0	5679.0	35.242	27.0	28.568	0.945741	0
1345	1345	1520	34	11296.0	1445.0	32.769	27.0	34.686	0.572662	0
1409	1409	95	34	6957.0	1154.0	38.696	27.0	32.139	0.694039	0
1437	1437	176	34	17065.0	1750.0	23.224	27.0	31.786	0.936805	0
1441	1441	1373	34	27780.0	3500.0	24.571	27.0	26.147	0.816603	0
1539	1539	48	34	73565.0	10669.0	36.105	27.0	31.934	0.405282	0

33 rows × 85 columns

In [285]: `statethirtyfour['gross_rent'].describe()`

Out[285]:

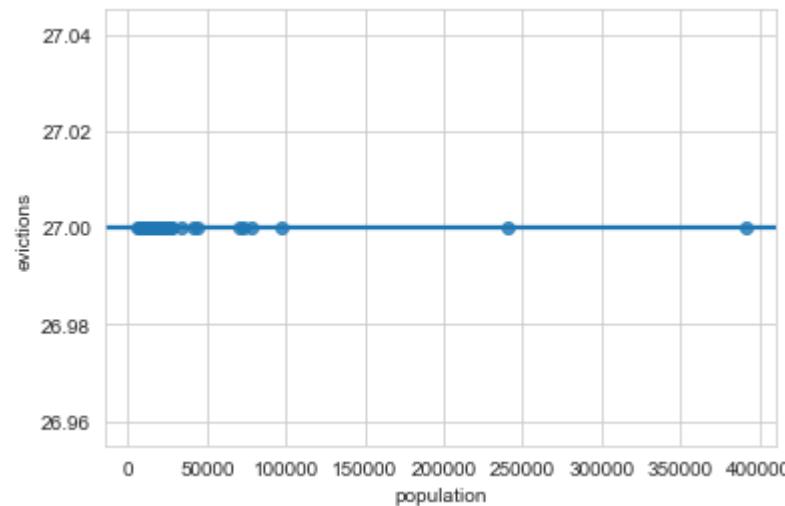
count	33.000000
mean	605.151515
std	77.309492
min	468.000000
25%	549.000000
50%	596.000000
75%	641.000000
max	796.000000
Name:	gross_rent, dtype: float64

```
In [286]: statethirtyfour['homicides_per_100k'].describe()
```

```
Out[286]: count    33.000000
mean      5.585758
std       7.162072
min      0.000000
25%     0.000000
50%     2.680000
75%     9.010000
max     26.160000
Name: homicides_per_100k, dtype: float64
```

```
In [287]: import seaborn as sns

ax = sns.regplot(x='population', y='evictions', data=statethirtyfour)
```



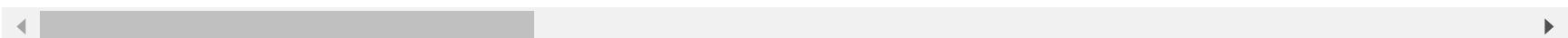
```
In [288]: train_values_labels.drop(statethirtyfour.index, inplace = True)
```

```
In [289]: statethirtyseven=train_values_labels[train_values_labels['state']==37]
statethirtyseven
```

Out[289]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	82	82	1390	37	1014148.0	125464.0	34.065	27.0	30.436	0.465959
	163	163	534	37	153229.0	12298.0	22.138	27.0	34.077	0.452282
	357	357	102	37	101870.0	14661.0	37.538	27.0	32.796	0.650666
	728	728	685	37	243215.0	22615.0	25.935	27.0	30.400	0.759955
	1171	1171	677	37	26158.0	3317.0	34.911	27.0	40.195	0.517783
	1201	1201	346	37	149184.0	20376.0	35.483	27.0	27.799	0.814674

6 rows × 85 columns



```
In [290]: statethirtyseven['gross_rent'].describe()
```

```
Out[290]: count      6.000000
mean     1159.000000
std      365.580634
min      703.000000
25%     892.000000
50%    1139.500000
75%    1436.500000
max    1627.000000
Name: gross_rent, dtype: float64
```

```
In [291]: statethirtyseven['homicides_per_100k'].describe()
```

```
Out[291]: count    6.000000
mean     2.756667
std      2.288333
min     0.000000
25%    1.517500
50%    2.080000
75%    4.165000
max    6.190000
Name: homicides_per_100k, dtype: float64
```

```
In [292]: train_values_labels.drop(statethirtyseven.index, inplace = True)
```

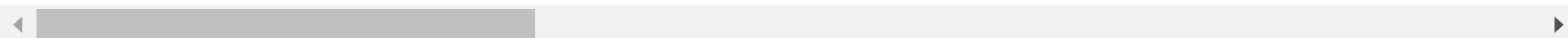
In [293]: statefortynine=train_values_labels[train_values_labels['state']==49]
statefortynine

Out[293]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
0	0	869	49	3876.0	408.0	24.583	27.0	18.380	0.945945	0.011
11	11	143	49	2352.0	277.0	25.940	27.0	21.172	0.965858	0.011
65	65	411	49	3213.0	403.0	26.003	27.0	16.022	0.935264	0.001
94	94	504	49	3050.0	321.0	22.585	27.0	16.355	0.945344	0.011
98	98	104	49	2744.0	286.0	17.787	27.0	22.095	0.958078	0.001
153	153	365	49	2337.0	322.0	26.382	27.0	18.675	0.923389	0.011
277	277	1016	49	5218.0	652.0	28.030	27.0	22.014	0.940592	0.011
291	291	1389	49	2191.0	217.0	32.524	27.0	22.289	0.930284	0.001
364	364	1084	49	29713.0	4426.0	36.334	27.0	20.142	0.868544	0.011
399	399	1104	49	3359.0	410.0	26.090	27.0	21.787	0.965013	0.001
456	456	584	49	7202.0	769.0	21.833	27.0	22.058	0.927284	0.001
458	458	1275	49	4189.0	448.0	21.698	27.0	19.888	0.977662	0.001
531	531	539	49	14474.0	1668.0	32.048	27.0	27.307	0.199088	0.001
604	604	571	49	9257.0	1097.0	30.294	27.0	17.740	0.643903	0.001
650	650	926	49	11034.0	1245.0	25.605	27.0	25.300	0.857445	0.001
787	787	1505	49	2369.0	268.0	18.594	27.0	25.707	0.970262	0.001
792	792	229	49	2610.0	182.0	17.038	27.0	19.081	0.941162	0.001
923	923	585	49	8085.0	1030.0	26.973	27.0	24.984	0.936535	0.001
977	977	515	49	163071.0	34420.0	48.073	27.0	26.063	0.887579	0.031
980	980	952	49	2441.0	237.0	18.913	27.0	21.112	0.947278	0.001
1070	1070	130	49	16280.0	2001.0	28.664	27.0	22.380	0.926021	0.001
1132	1132	681	49	6577.0	623.0	25.708	27.0	22.276	0.928181	0.001
1135	1135	1068	49	87435.0	12507.0	30.000	27.0	24.161	0.910490	0.001
1192	1192	1419	49	6837.0	768.0	35.552	27.0	19.996	0.421421	0.001

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
1203	1203	1504	49	2546.0	262.0	21.969	27.0	19.680	0.958350	0.00
1292	1292	145	49	8693.0	790.0	17.141	27.0	18.590	0.938897	0.00
1324	1324	756	49	1931.0	149.0	16.495	27.0	26.194	0.965652	0.00
1377	1377	795	49	28942.0	3395.0	24.610	27.0	24.472	0.913270	0.00
1482	1482	1555	49	4401.0	503.0	26.560	27.0	29.157	0.957195	0.00
1517	1517	523	49	68874.0	14573.0	50.003	27.0	30.314	0.863820	0.02

30 rows × 85 columns



In [294]: `statefortynine['gross_rent'].describe()`

Out[294]:

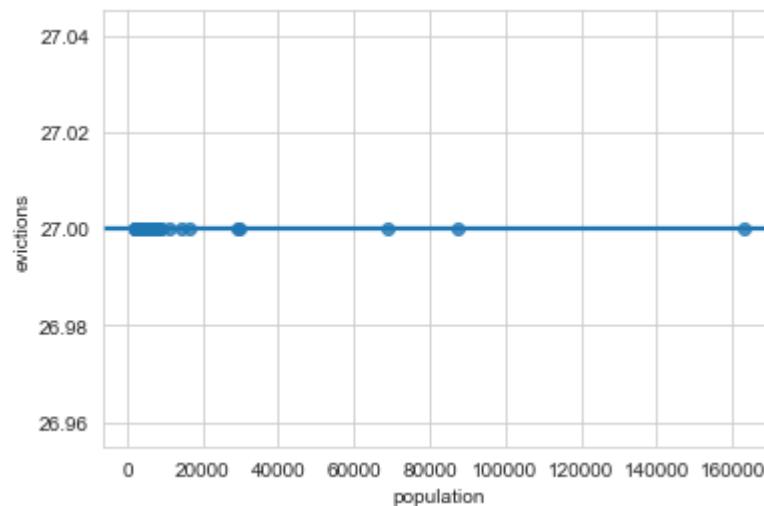
count	30.000000
mean	577.900000
std	125.367969
min	351.000000
25%	488.750000
50%	562.500000
75%	663.500000
max	898.000000
Name:	gross_rent, dtype: float64

In [295]: `statefortynine['homicides_per_100k'].describe()`

Out[295]:

count	30.0
mean	0.0
std	0.0
min	0.0
25%	0.0
50%	0.0
75%	0.0
max	0.0
Name:	homicides_per_100k, dtype: float64

```
In [296]: import seaborn as sns  
  
ax = sns.regplot(x='population', y='evictions', data=statefortynine)
```



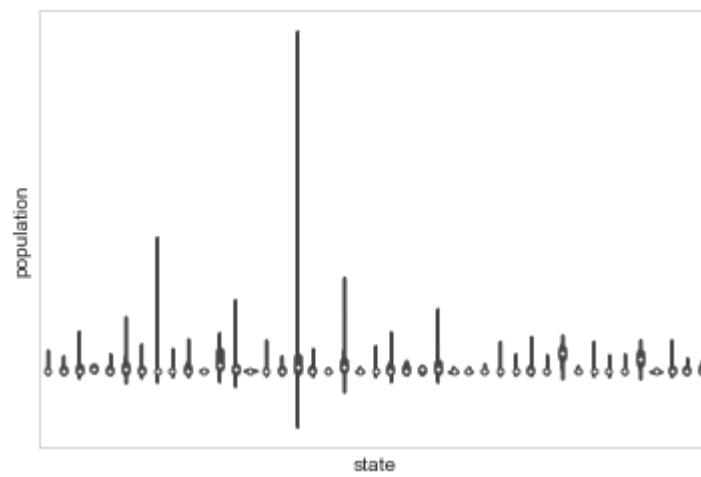
```
In [297]: train_values_labels.drop(statefortynine.index, inplace = True)
```

```
In [298]: train_values_labels['evictions'].median()
```

```
Out[298]: 27.0
```

```
In [299]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'state'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.xticks([])
        plt.yticks([])
        plt.show()

plot_violin(train_values_labels, num_cols)
```

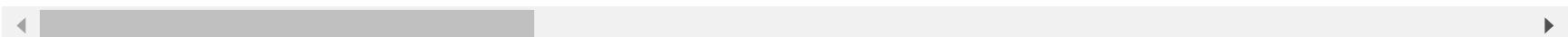


```
In [300]: statethirteen = train_values_labels[train_values_labels['state']==13]
statethirteen
```

Out[300]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
1256	1256	170	13	32129.0		4345.0	30.152	99.0	28.712	0.944723
1322	1322	265	13	47370.0		4783.0	20.300	83.0	30.916	0.961649

2 rows × 85 columns

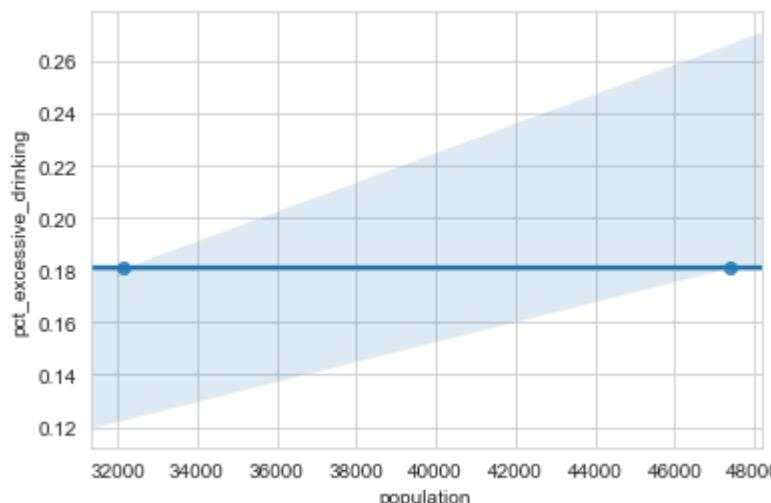


```
In [301]: statethirteen['pct_excessive_drinking']
```

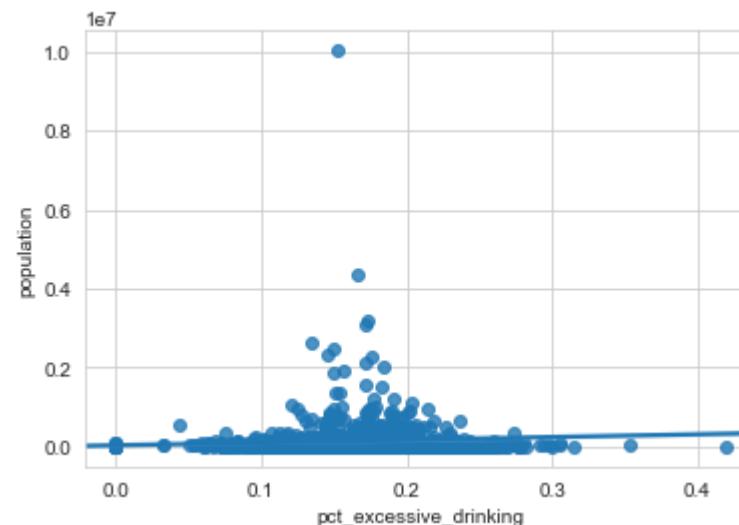
Out[301]: 1256 0.181
1322 0.181
Name: pct_excessive_drinking, dtype: float64

```
In [302]: import seaborn as sns
```

```
ax = sns.regplot(x='population', y='pct_excessive_drinking', data=statethirteen)
```



```
In [303]: import seaborn as sns  
  
ax = sns.regplot(x="pct_excessive_drinking", y="population", data=train_values_labels)
```



```
In [304]: eldronko = train_values_labels[train_values_labels['population']>1*10**7]  
eldronko
```

Out[304]:

row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_	
953	953	419	16	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.079

1 rows × 85 columns

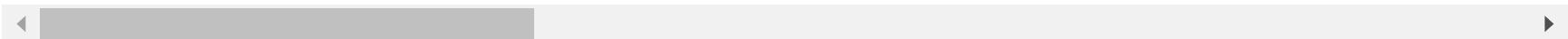
◀ ▶

In [305]: `eldronko.describe()`

Out[305]:

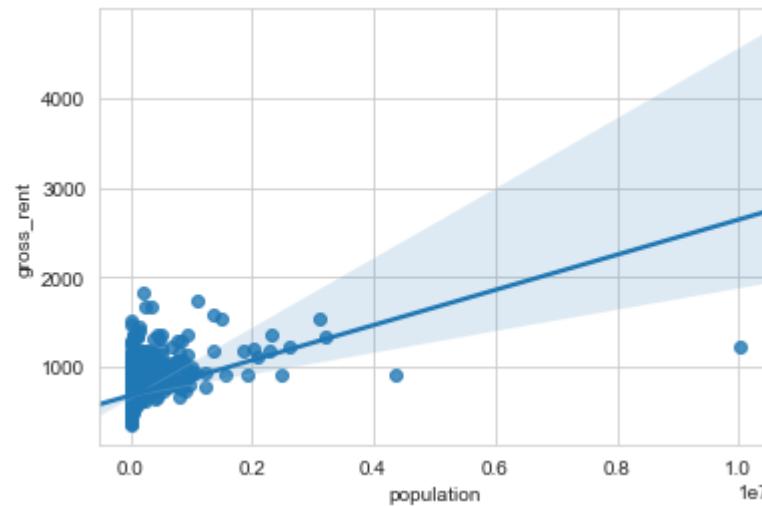
	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_i
count	1.0	1.0	1.0	1.0	1.0	1.000	1.0	1.000	1.000000	1.0
mean	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0
25%	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0
50%	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0
75%	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0
max	953.0	419.0	16.0	10020287.0	1760277.0	54.552	21783.0	34.916	0.269376	0.0

8 rows × 82 columns



```
In [306]: import seaborn as sns
```

```
ax = sns.regplot(x='population', y='gross_rent', data=train_values_labels)
```



```
In [307]: highpop = train_values_labels[train_values_labels['population'] > 400000]
highpop
```

Out[307]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
10	10	372	46	963541.0	191320.0	50.311	6749.0	32.229	0.528537	0
15	15	1049	14	991145.0	195220.0	47.798	4190.0	30.317	0.403998	0
16	16	777	5	1218942.0	198288.0	35.131	3434.0	28.559	0.795641	0
23	23	1451	19	2318272.0	465328.0	56.318	27.0	33.998	0.263296	0
25	25	317	4	537958.0	76931.0	33.949	3192.0	29.999	0.787563	0
...
1463	1463	67	36	1117611.0	127238.0	31.998	1917.0	27.856	0.523858	0
1500	1500	335	19	463808.0	69065.0	34.859	215.0	30.391	0.779398	0
1527	1527	710	7	2481822.0	432841.0	48.287	10865.0	29.232	0.313163	0
1556	1556	1048	17	667796.0	103780.0	37.620	4928.0	31.909	0.407850	0
1561	1561	344	8	650813.0	92680.0	37.789	27.0	30.687	0.704111	0

90 rows × 85 columns

```
In [308]: lowrent = highpop[highpop['gross_rent']<1000]
lowrent
```

Out[308]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
10	10	372	46	963541.0	191320.0	50.311	6749.0	32.229	0.528537	0
16	16	777	5	1218942.0	198288.0	35.131	3434.0	28.559	0.795641	0
25	25	317	4	537958.0	76931.0	33.949	3192.0	29.999	0.787563	0
41	41	1355	7	725489.0	97063.0	35.276	2268.0	28.412	0.619152	0
70	70	647	7	4371311.0	688268.0	44.981	29251.0	29.758	0.317401	0
120	120	669	44	466320.0	71053.0	38.401	1621.0	32.890	0.653465	0
137	137	836	12	448216.0	57613.0	35.707	656.0	29.894	0.803549	0
161	161	260	19	915659.0	139688.0	34.828	27.0	29.740	0.763446	0
189	189	1242	25	480738.0	50142.0	26.215	1496.0	32.647	0.778386	0
203	203	291	2	1221929.0	138466.0	29.774	4667.0	28.196	0.736175	0
232	232	1426	33	532599.0	82622.0	41.036	952.0	30.354	0.643960	0
296	296	1082	5	1547703.0	288255.0	47.576	11181.0	34.277	0.357644	0
306	306	49	25	621120.0	76655.0	30.969	1472.0	31.983	0.625155	0
369	369	1294	9	992748.0	162065.0	42.571	10913.0	29.201	0.491648	0
433	433	960	39	510930.0	64730.0	32.613	2244.0	33.225	0.587128	0
444	444	234	6	773504.0	151084.0	45.763	1695.0	32.538	0.711724	0
478	478	1044	25	937809.0	151015.0	35.047	4701.0	32.424	0.755926	0
503	503	736	33	408195.0	37740.0	25.450	495.0	28.470	0.808673	0
517	517	746	38	540161.0	83364.0	38.055	3116.0	29.292	0.708709	0
572	572	34	15	442595.0	77014.0	40.510	4676.0	33.150	0.460094	0
617	617	662	22	1069646.0	123043.0	33.638	2104.0	29.273	0.727127	0
629	629	92	0	681375.0	110332.0	40.431	5690.0	30.639	0.629170	0
631	631	1080	7	1931166.0	272734.0	39.212	12310.0	29.403	0.498917	0
633	633	820	25	658233.0	88363.0	30.782	2038.0	32.032	0.696760	0

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
643	643	589	8	654494.0	147601.0	50.872	3729.0	29.701	0.533719	C
693	693	4	15	438520.0	67433.0	38.084	2137.0	32.485	0.542860	C
695	695	121	7	811865.0	72937.0	32.192	533.0	32.576	0.072819	C
703	703	915	9	969850.0	137959.0	35.875	4854.0	28.584	0.613351	C
713	713	1312	22	547371.0	49957.0	32.959	246.0	29.960	0.833345	C
716	716	1098	12	478945.0	70297.0	37.193	203.0	31.358	0.856704	C
772	772	676	46	509680.0	89031.0	41.590	623.0	29.739	0.807504	C
838	838	870	21	621133.0	99411.0	40.344	6762.0	28.528	0.641392	C
869	869	718	42	438845.0	74565.0	43.425	27.0	30.458	0.647465	C
921	921	913	7	836421.0	106031.0	38.591	552.0	30.029	0.131791	C
992	992	1494	7	418947.0	43592.0	33.295	27.0	31.983	0.101058	C
1020	1020	242	2	622847.0	73427.0	31.513	3469.0	29.853	0.749488	C
1038	1038	601	2	854736.0	82957.0	26.624	5925.0	31.220	0.814691	C
1047	1047	1038	21	761102.0	123821.0	41.011	5149.0	30.040	0.581041	C
1108	1108	420	32	942616.0	150697.0	42.746	5872.0	33.522	0.373562	C
1112	1112	828	25	558978.0	69345.0	28.381	1873.0	32.384	0.761906	C
1218	1218	917	11	888825.0	128169.0	35.148	4822.0	30.374	0.636748	C
1221	1221	678	16	955734.0	137127.0	47.140	3186.0	34.393	0.310215	C
1258	1258	700	14	709491.0	129167.0	44.323	9846.0	32.177	0.296293	C
1265	1265	1268	4	433457.0	70647.0	38.964	3577.0	30.910	0.699696	C
1336	1336	628	47	413398.0	55001.0	33.420	1329.0	33.181	0.582567	C
1350	1350	627	43	566835.0	67662.0	30.782	1468.0	26.297	0.809906	C
1368	1368	157	19	753311.0	111989.0	36.152	27.0	33.135	0.716121	C
1390	1390	349	9	510815.0	83695.0	40.120	5722.0	30.728	0.523185	C
1392	1392	701	16	526369.0	68713.0	43.009	27.0	34.577	0.447179	C
1413	1413	1200	1	486938.0	60774.0	31.356	2193.0	31.375	0.548256	C
1459	1459	1528	5	525419.0	64812.0	31.311	1081.0	30.816	0.833251	C

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pc
1500	1500	335	19	463808.0	69065.0	34.859	215.0	30.391	0.779398	0
1527	1527	710	7	2481822.0	432841.0	48.287	10865.0	29.232	0.313163	0
1556	1556	1048	17	667796.0	103780.0	37.620	4928.0	31.909	0.407850	0
1561	1561	344	8	650813.0	92680.0	37.789	27.0	30.687	0.704111	0

55 rows × 85 columns

In [309]: `lowrent.describe()`

Out[309]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden
count	55.000000	55.000000	55.000000	5.500000e+01	55.000000	55.000000	55.000000	55.000000
mean	769.563636	728.254545	17.581818	8.090740e+05	120236.927273	37.330927	3786.218182	30.930673
std	481.212310	403.977778	13.566968	6.182705e+05	103011.303374	6.056323	4708.278736	1.799487
min	10.000000	4.000000	0.000000	4.081950e+05	37740.000000	25.450000	27.000000	26.297000
25%	401.000000	360.500000	7.000000	5.102475e+05	69821.000000	33.127000	804.000000	29.739500
50%	703.000000	710.000000	15.000000	6.508130e+05	88363.000000	37.193000	2244.000000	30.687000
75%	1219.500000	1041.000000	25.000000	9.022420e+05	137543.000000	41.023500	4891.000000	32.404000
max	1561.000000	1528.000000	47.000000	4.371311e+06	688268.000000	50.872000	29251.000000	34.577000

8 rows × 82 columns

In [310]: `#train_values_labels.drop(lowrent.index, inplace = True)`In [311]: `#train_values_labels.drop(eldronko.index, inplace = True)`In [312]: `#train_values_labels.drop(nodrinking.index, inplace = True)`

```
In [313]: statethirteen['gross_rent']
```

```
Out[313]: 1256    691  
1322    899  
Name: gross_rent, dtype: int64
```

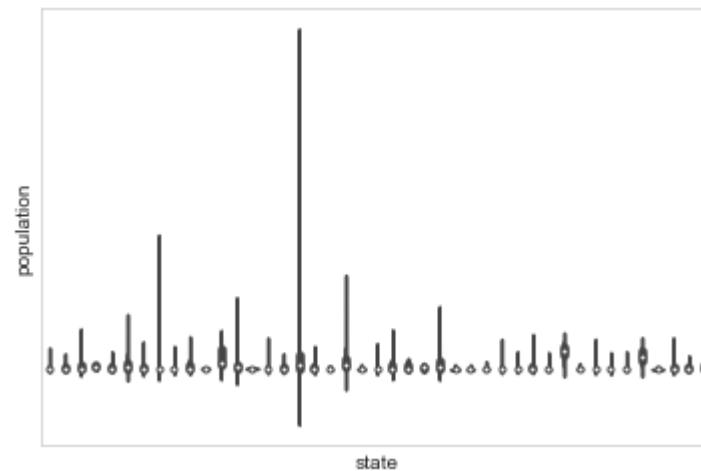
```
In [314]: statethirteen['homicides_per_100k'].describe()
```

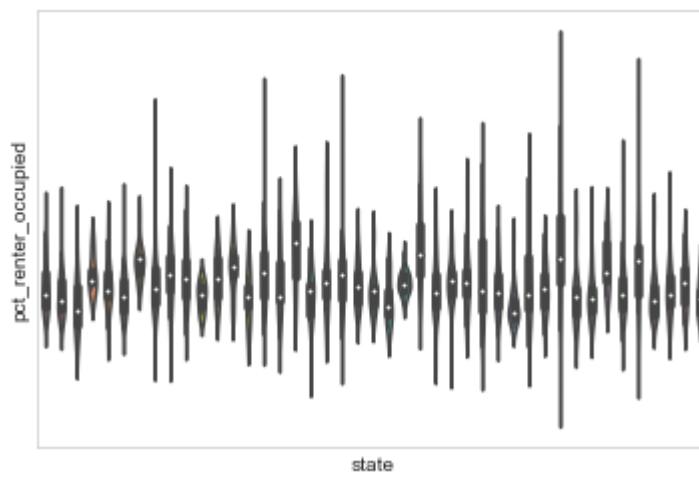
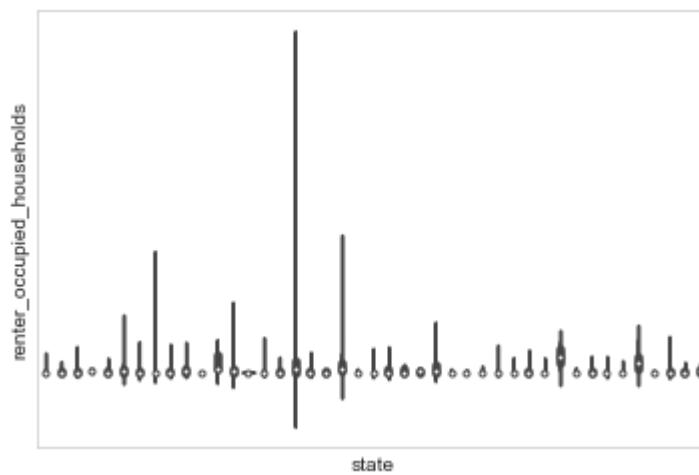
```
Out[314]: count    2.0  
mean     0.0  
std      0.0  
min     0.0  
25%     0.0  
50%     0.0  
75%     0.0  
max     0.0  
Name: homicides_per_100k, dtype: float64
```

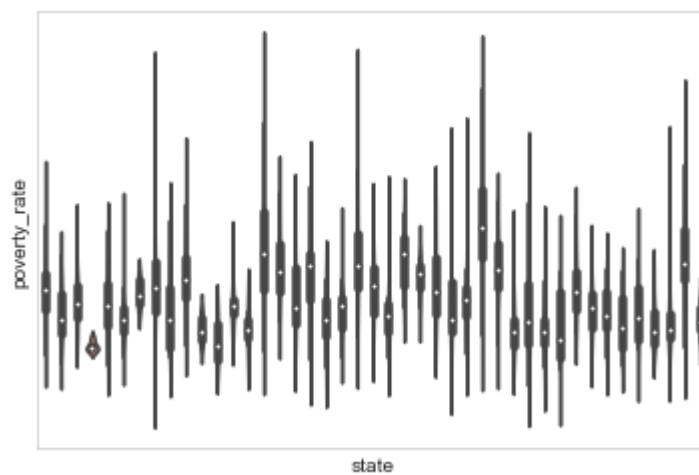
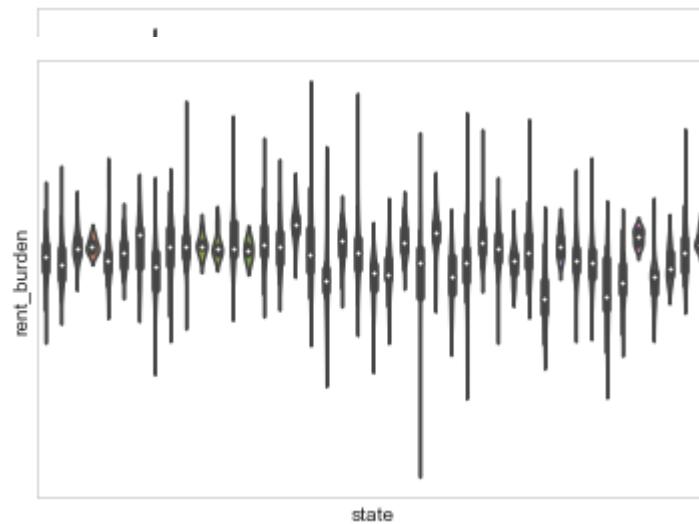
```
In [315]: #train_values_labels.drop(statethirteen.index, inplace = True)
```

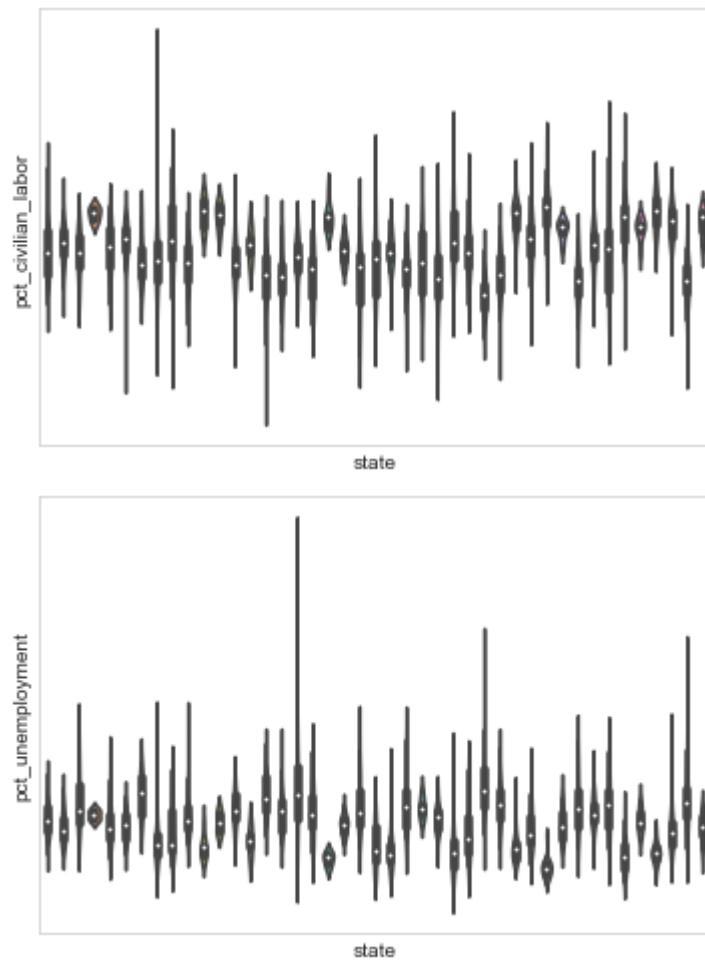
```
In [316]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'state'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.xticks([])
        plt.yticks([])
        plt.show()

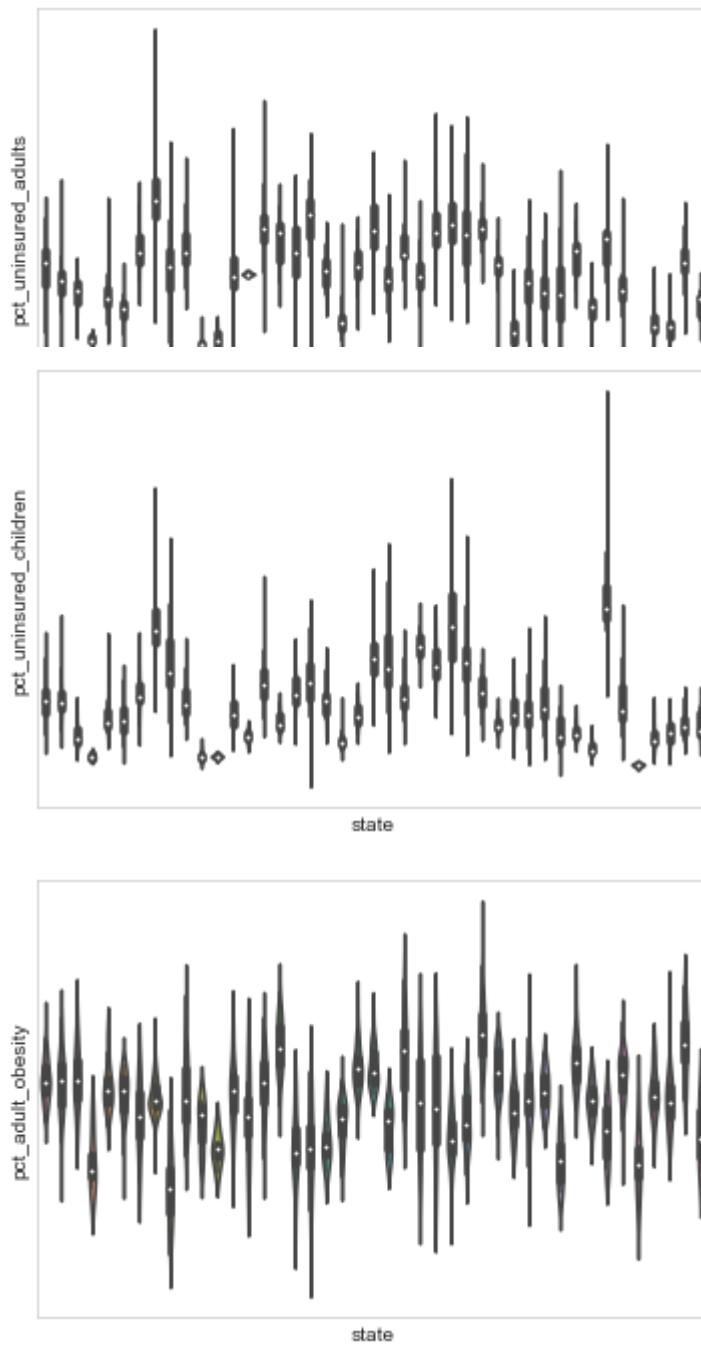
plot_violin(train_values_labels, num_cols)
```

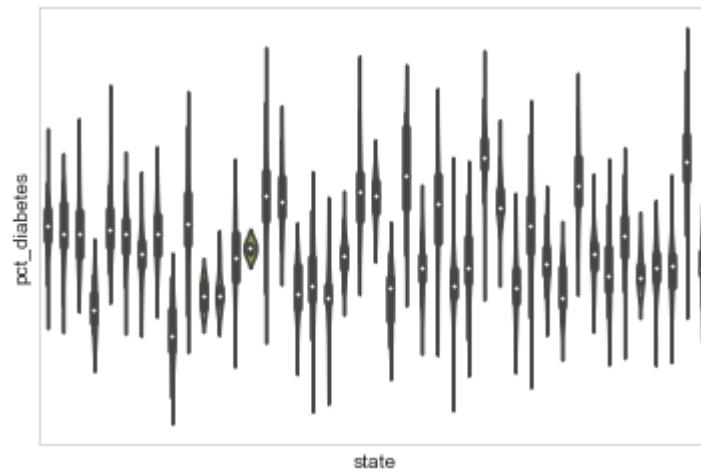
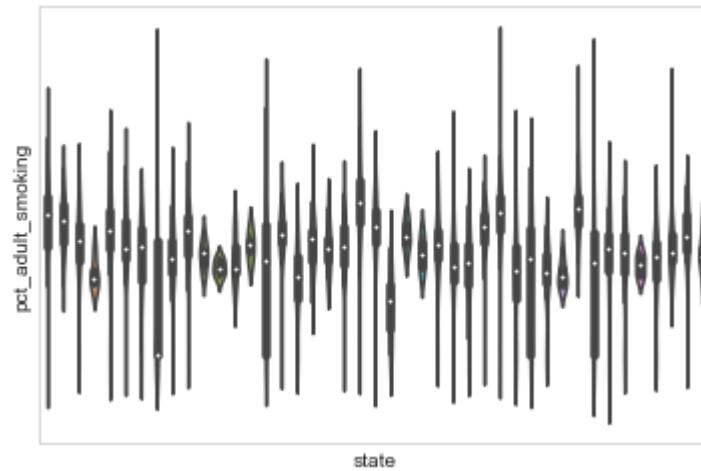


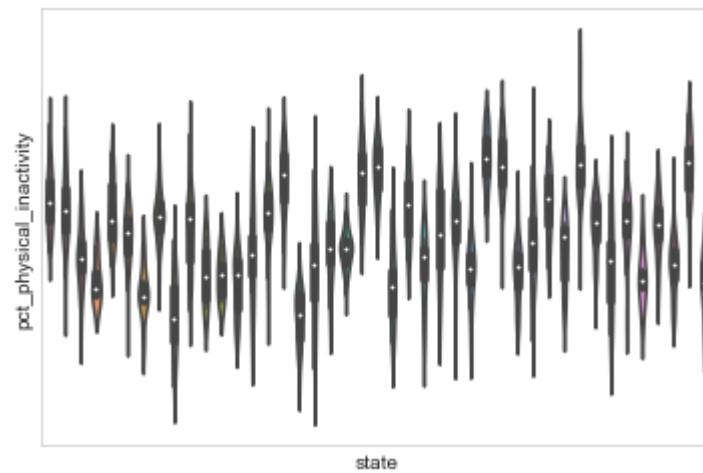
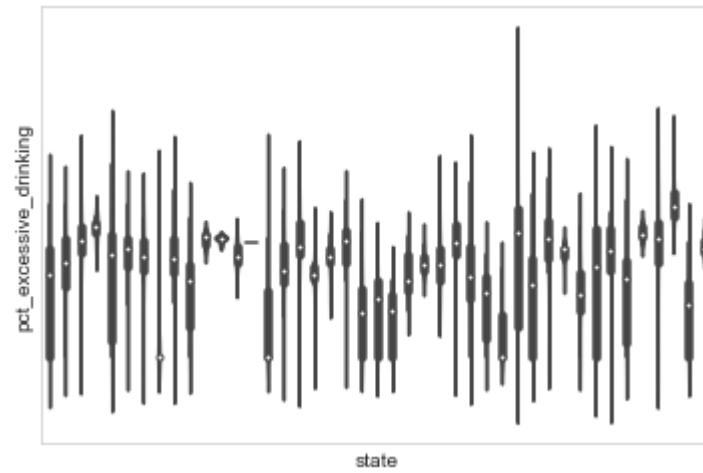
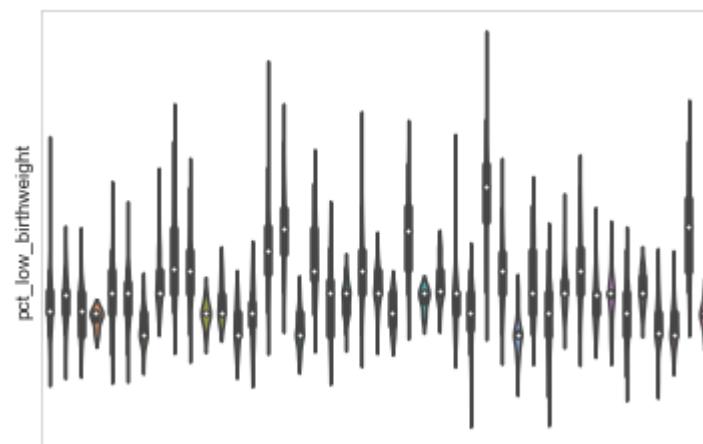


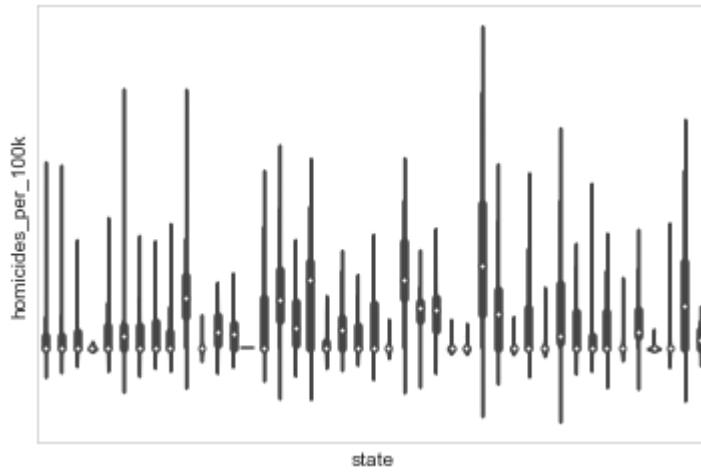
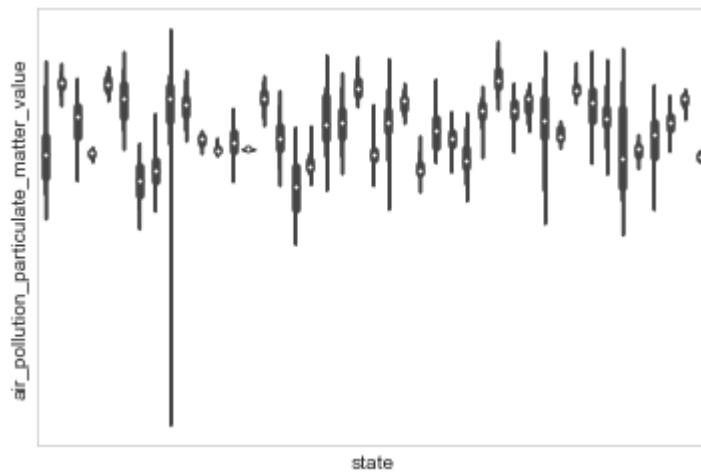


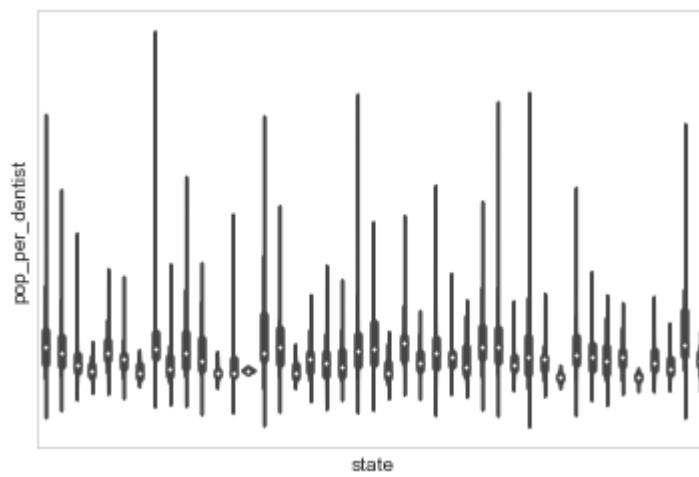
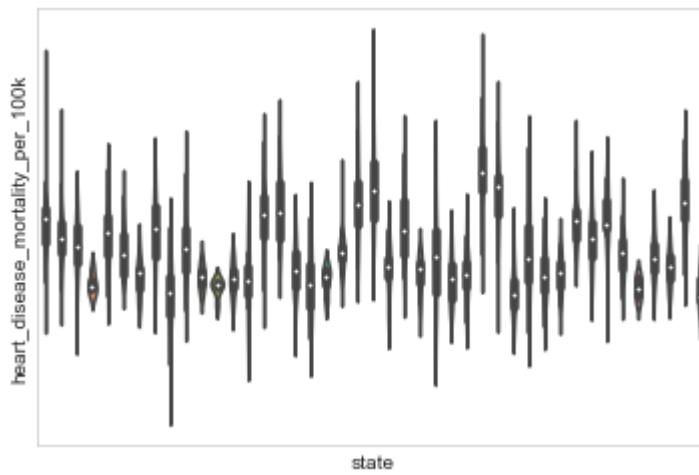
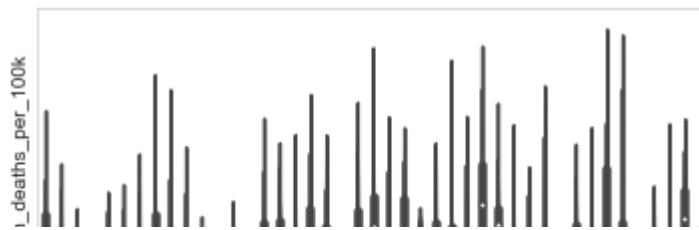


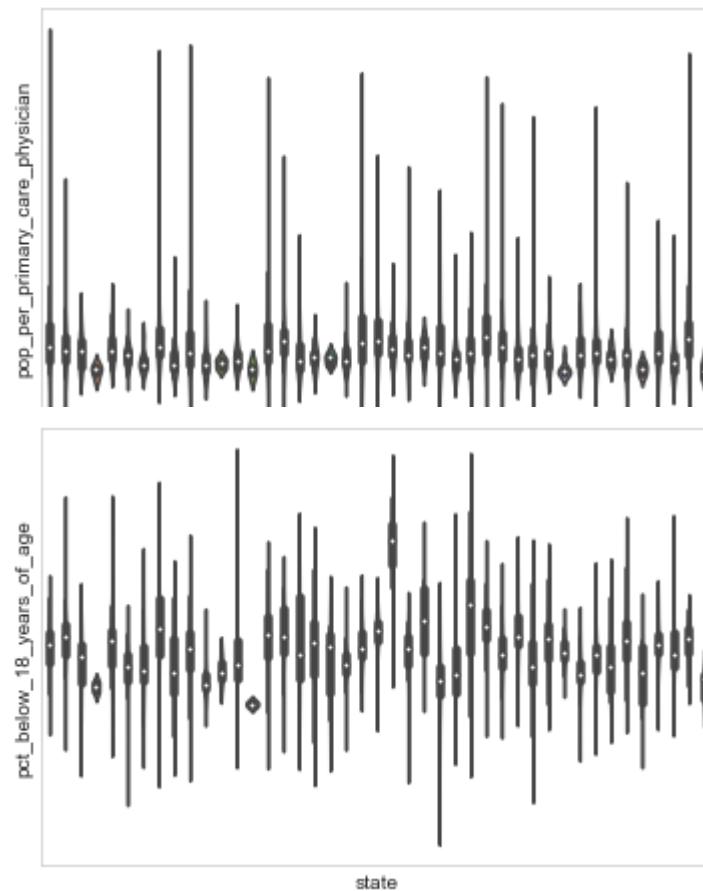


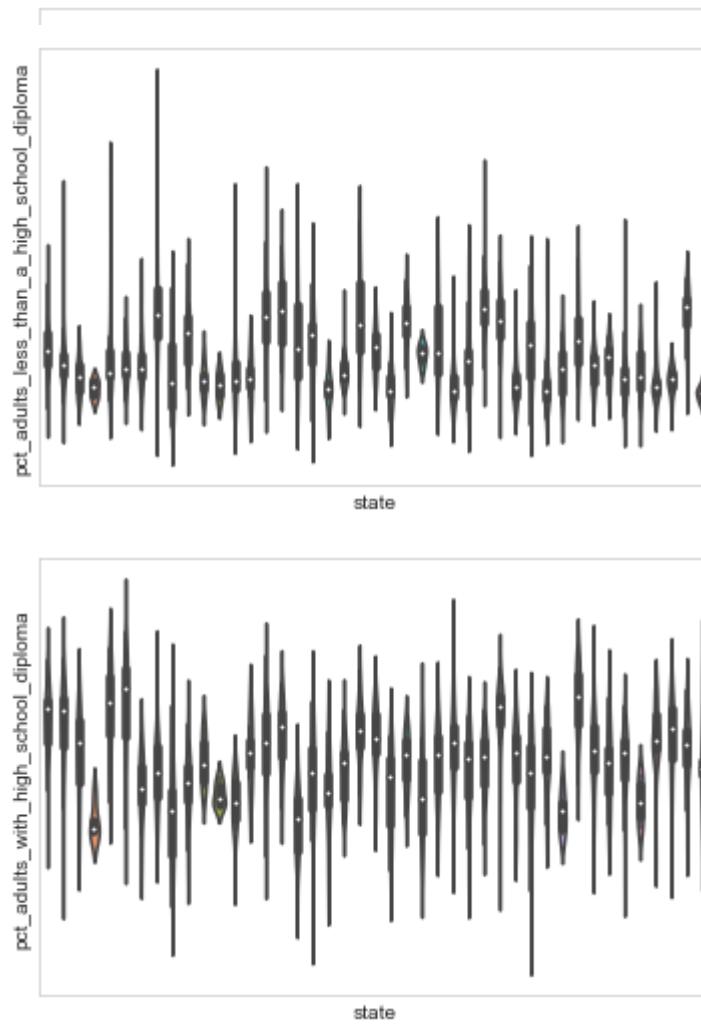


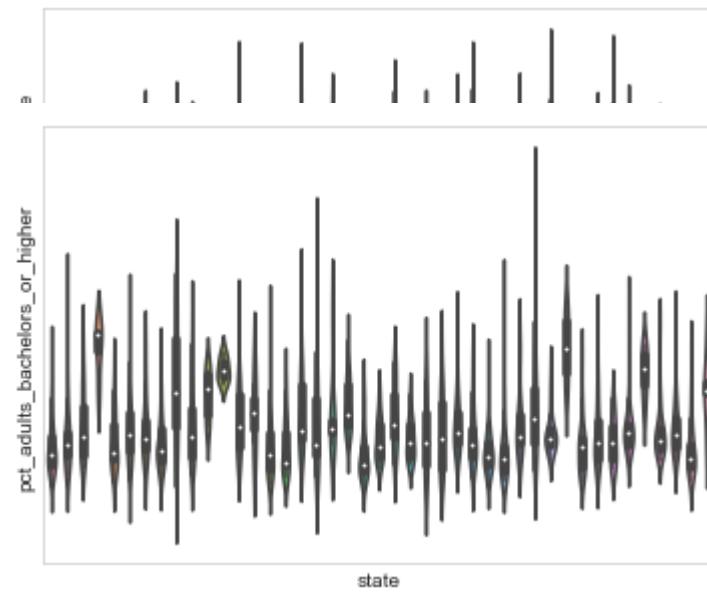


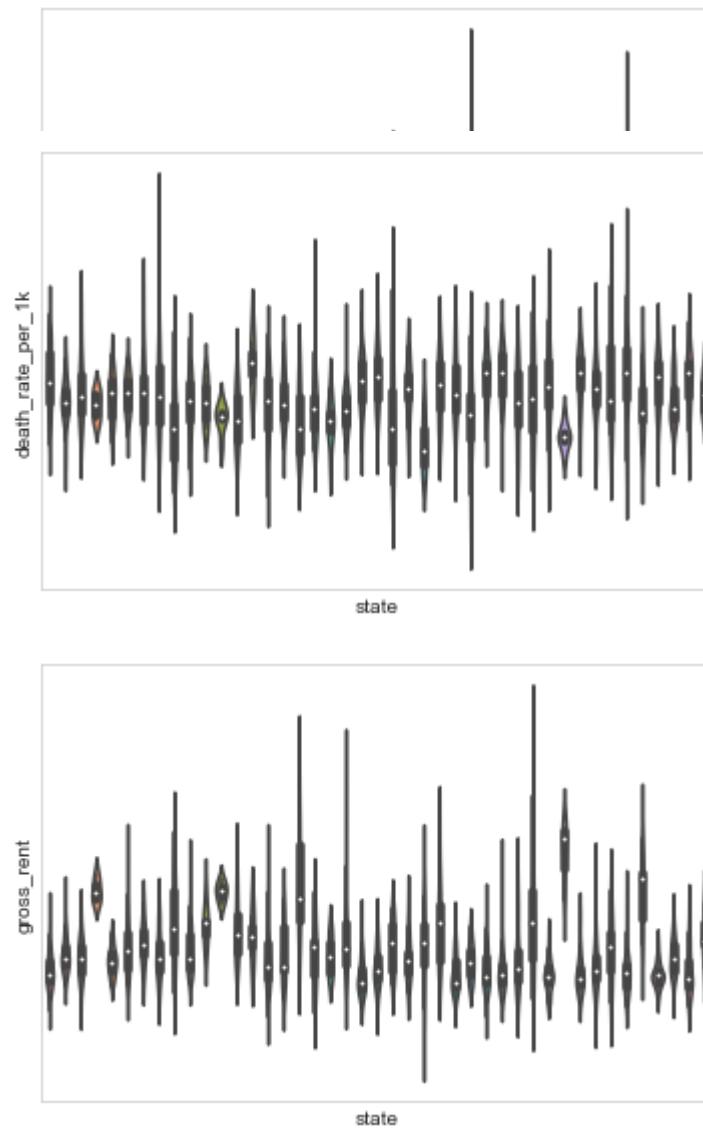


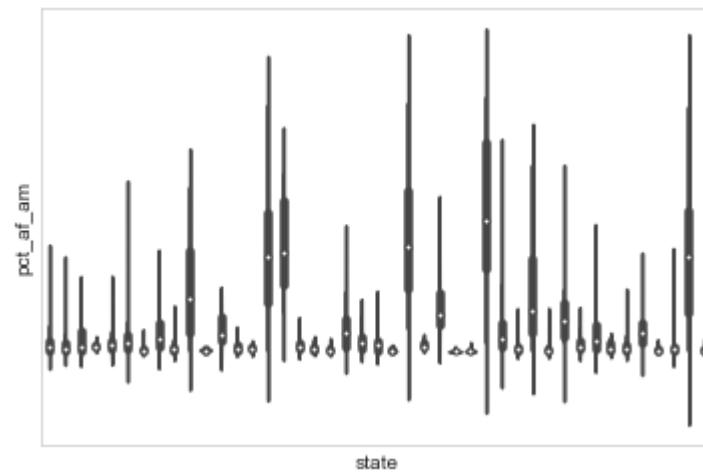
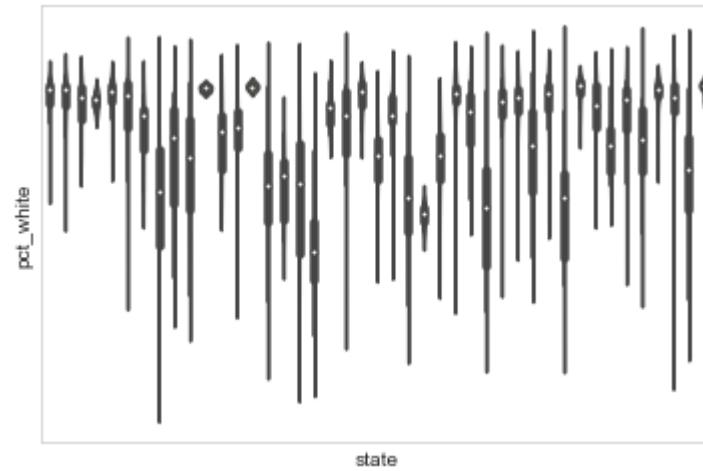


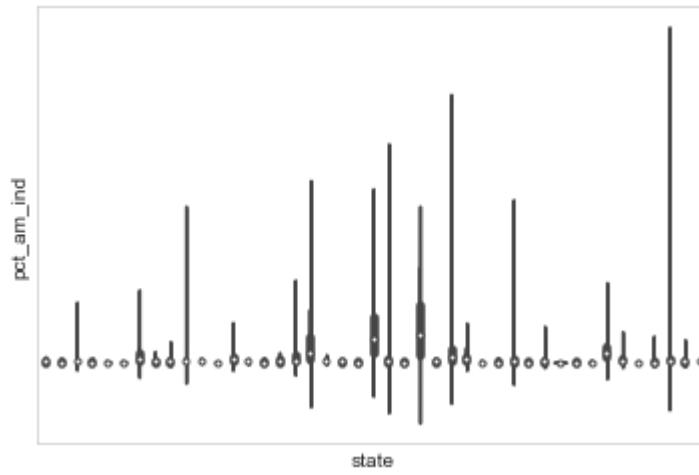
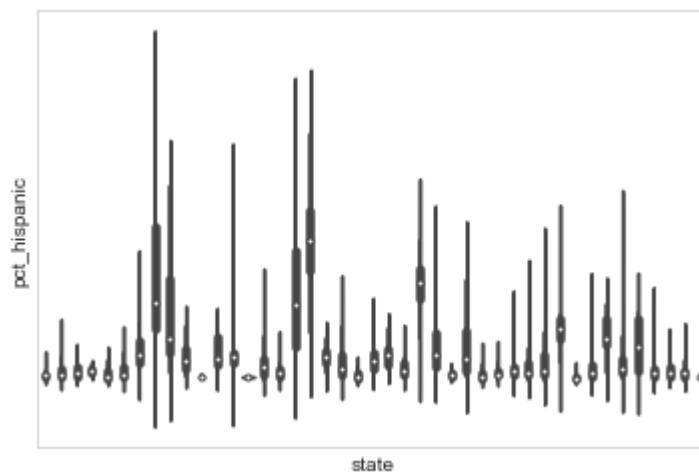


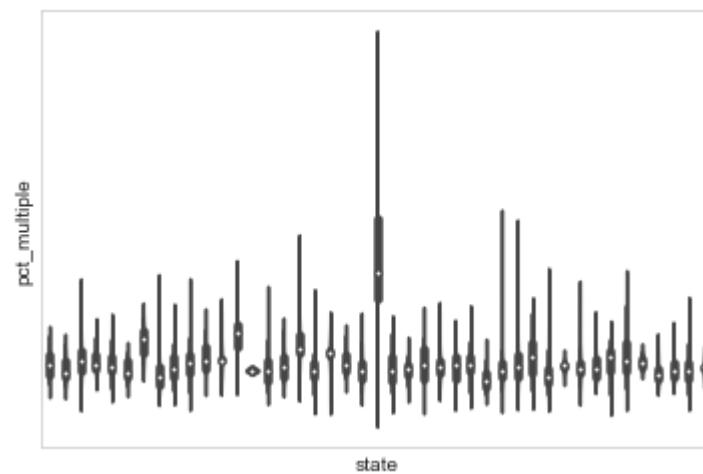
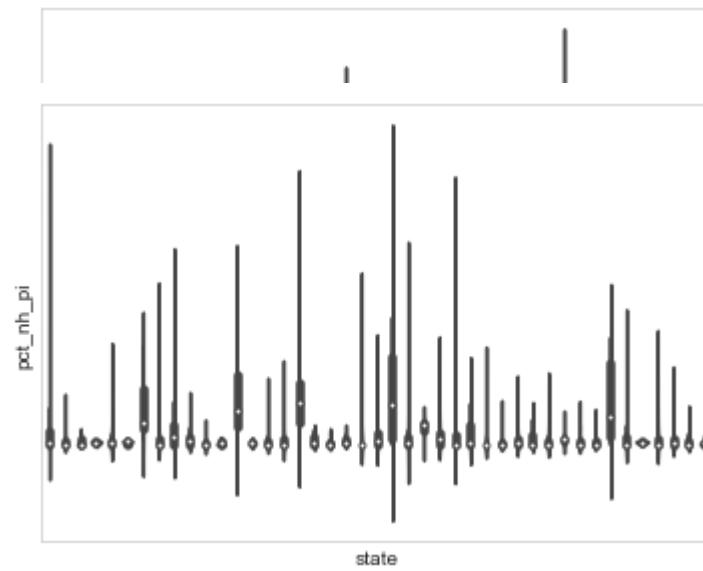


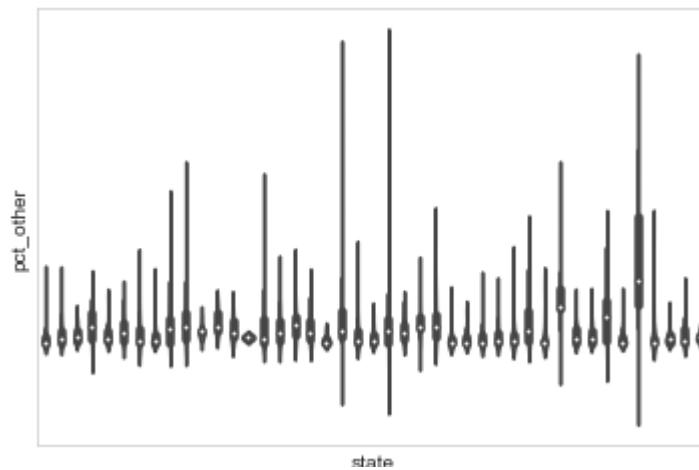












```
In [317]: #pollution = train_values_labels[train_values_labels['air_pollution_particulate_matter_value']==0]
#pollution
```

```
In [318]: #train_values_labels.drop(pollution.index, inplace = True)
```

```
In [319]: train_values_labels.columns
```

```
Out[319]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [320]: Correlations = train_values_labels[['gross_rent', 'pct_adults_bachelors_or_higher', 'pct_adults_with_high_school_diploma', 'pct_adult_obesity']]
corr = Correlations.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[320]:

	gross_rent	pct_adults_bachelors_or_higher	pct_adults_with_high_school_diploma	pct_adult_obesity
gross_rent	1	0.710152	-0.634984	-0.482307
pct_adults_bachelors_or_higher	0.710152	1	-0.761688	-0.594428
pct_adults_with_high_school_diploma	-0.634984	-0.761688	1	0.49217
pct_adult_obesity	-0.482307	-0.594428	0.49217	1
pct_diabetes	-0.438652	-0.559656	0.480347	0.731772
pct_physical_inactivity	-0.577041	-0.63927	0.602733	0.705268

```
In [321]: Correlations=train_values_labels[['pct_excessive_drinking', 'pct_adult_smoking','motor_vehicle_crash_deaths_per_100k', 'pct_adult_obesity']]
corr = Correlations.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[321]:

	pct_excessive_drinking	pct_adult_smoking	motor_vehicle_crash_deaths_per_100k	pct_adult_obesity
pct_excessive_drinking	1	0.191254	-0.246779	-0.250937
pct_adult_smoking	0.191254	1	0.258843	0.256894
motor_vehicle_crash_deaths_per_100k	-0.246779	0.258843	1	0.343313
pct_adult_obesity	-0.250937	0.256894	0.343313	1
pct_diabetes	-0.390954	0.251431	0.393515	0.731772
pct_physical_inactivity	-0.385806	0.289787	0.405683	0.705268

```
In [322]: Correlations= ['pct_adults_bachelors_or_higher', 'pct_adults_with_high_school_diploma', 'pct_adult_obesity', 'pct_diabetes', 'pct_physical_inactivity']
```

```
In [323]: Correlations2 = ['pct_adult_obesity','pct_diabetes','pct_physical_inactivity']
```

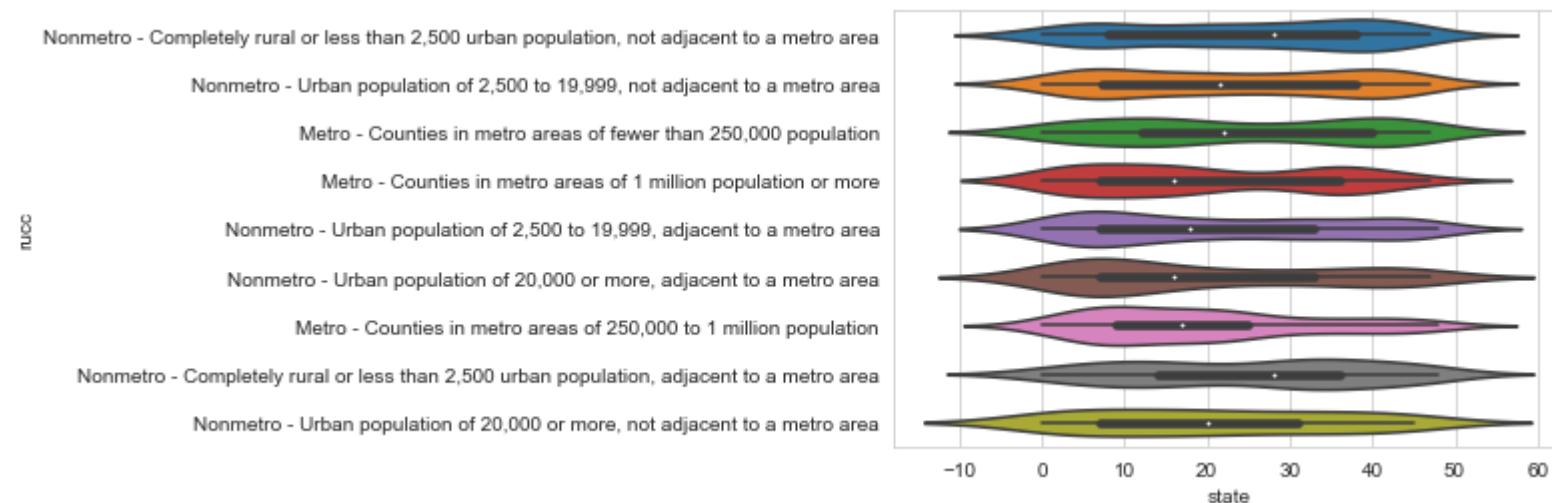
```
In [324]: def cond_plot(cols):
    import ipywidgets
    import seaborn as sns
    for col in cols:
        g = sns.FacetGrid(train_values_labels, col="urban_influence",
                           hue="economic_typology", palette="Set2", margin_titles=True, col_wrap=1)
        g.map(sns.regplot, col, "gross_rent", fit_reg = False)

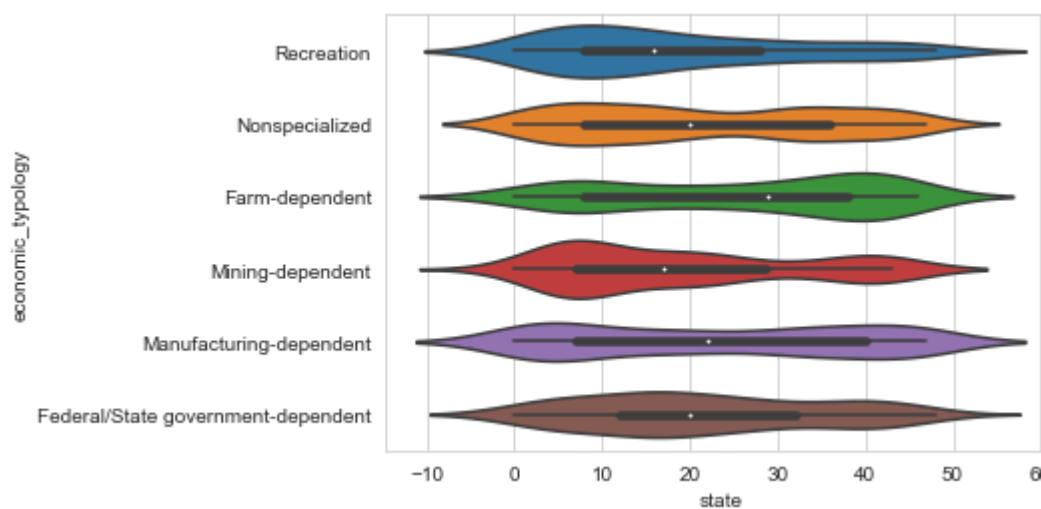
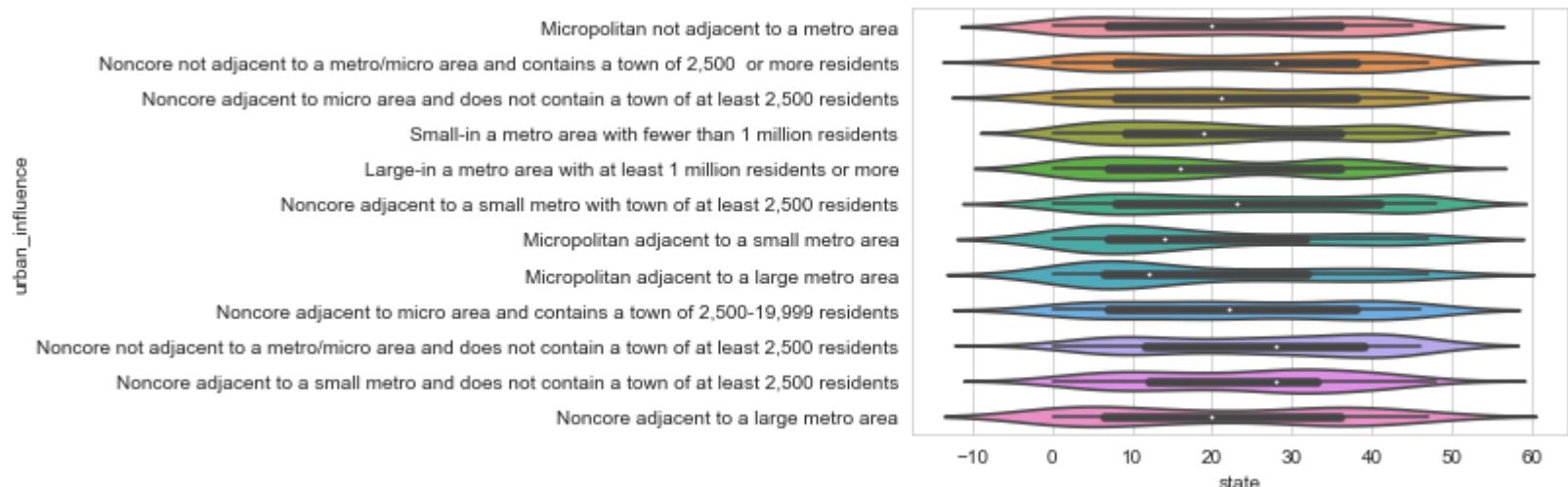
cond_plot(Correlations)
```

C:\Users\Jorge\Anaconda3\lib\site-packages\seaborn\axisgrid.py:848: UserWarning: Tight layout not applied. The left and right margins cannot be made large enough to accommodate all axes decorations.
self.fig.tight_layout()

```
In [325]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'state'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

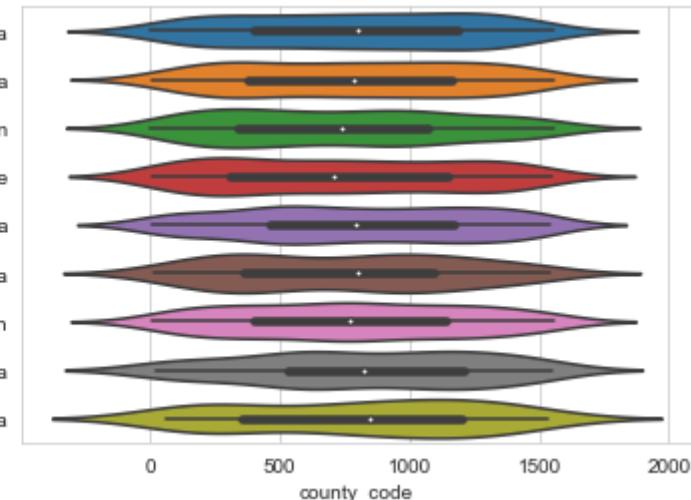


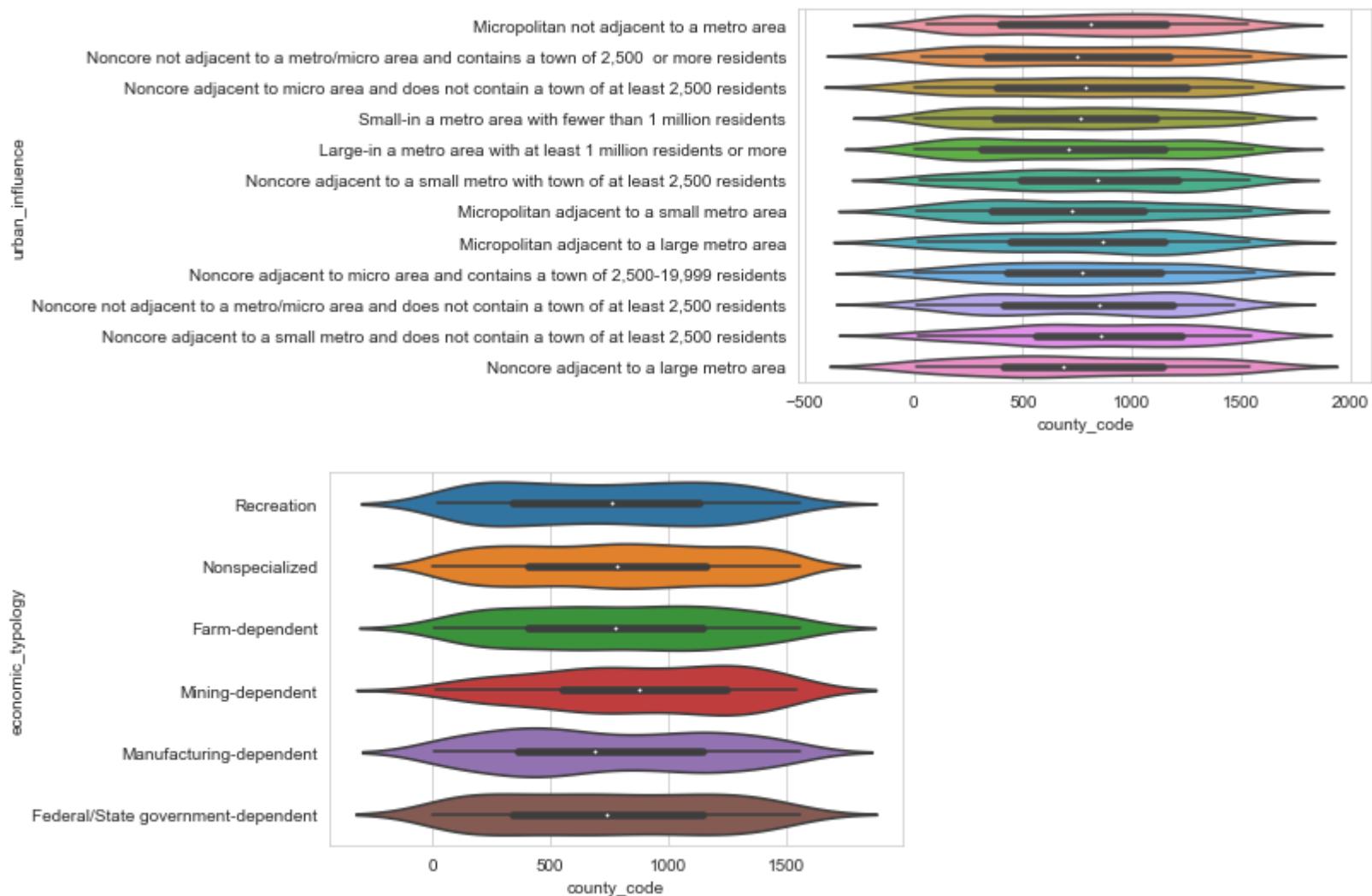



```
In [326]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'county_code'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





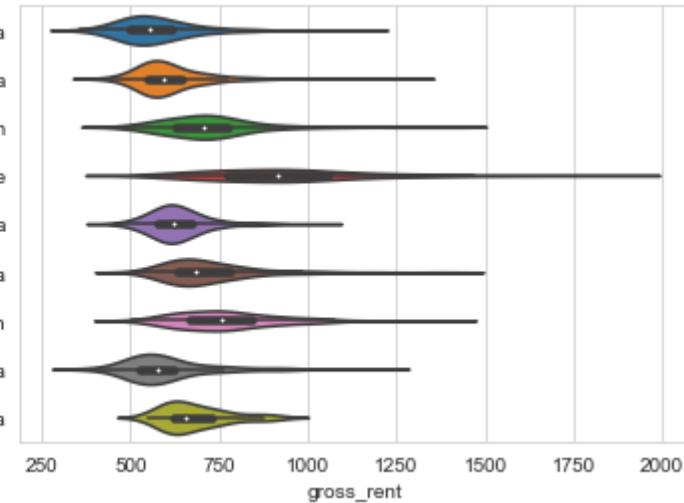
In [327]: `cat_cols`

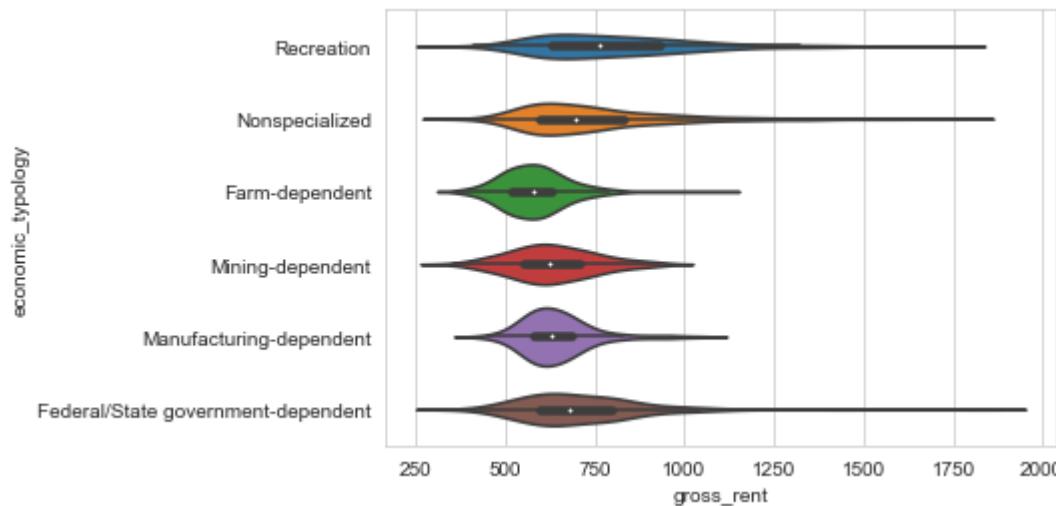
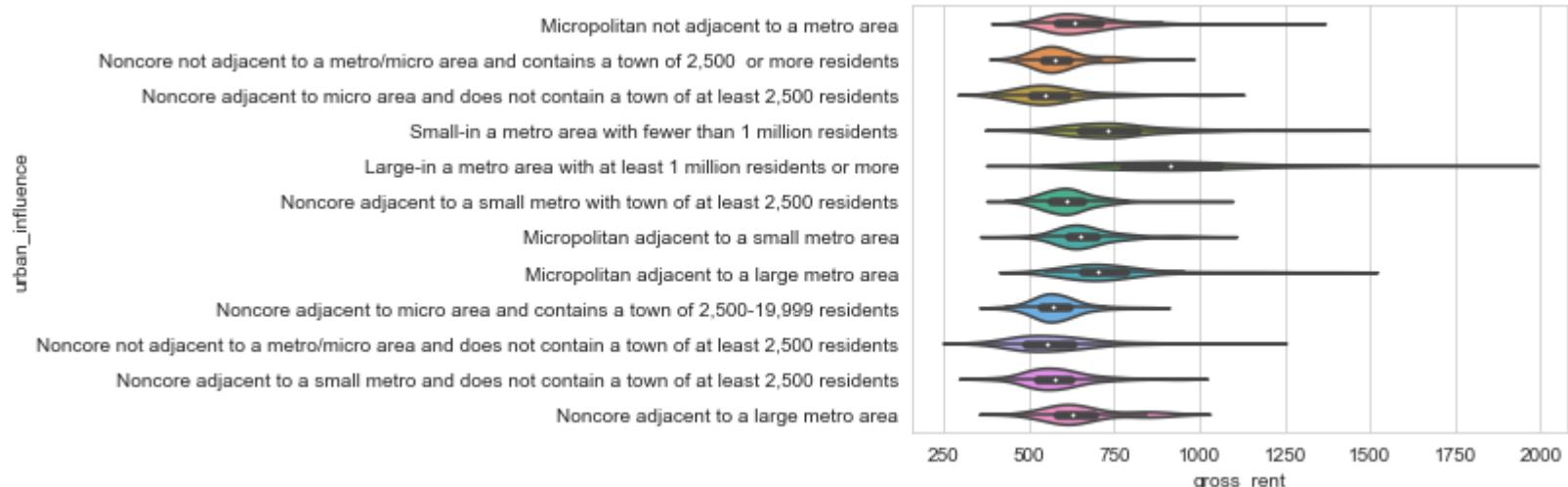
Out[327]: `['rucc', 'urban_influence', 'economic_typology']`

```
In [328]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'gross_rent'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





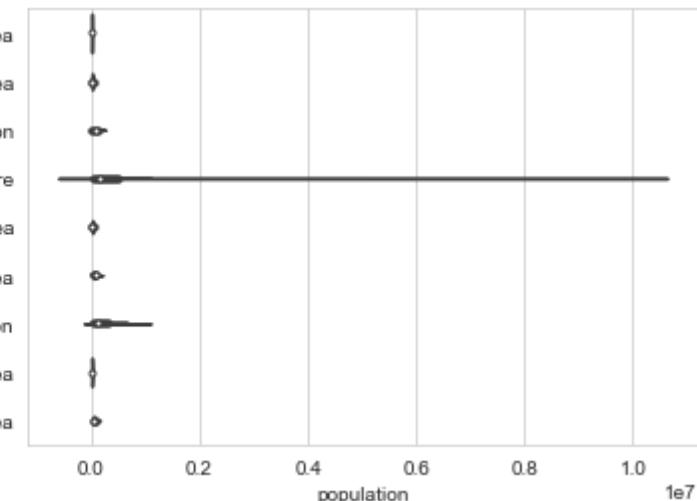
In [329]: num_cols

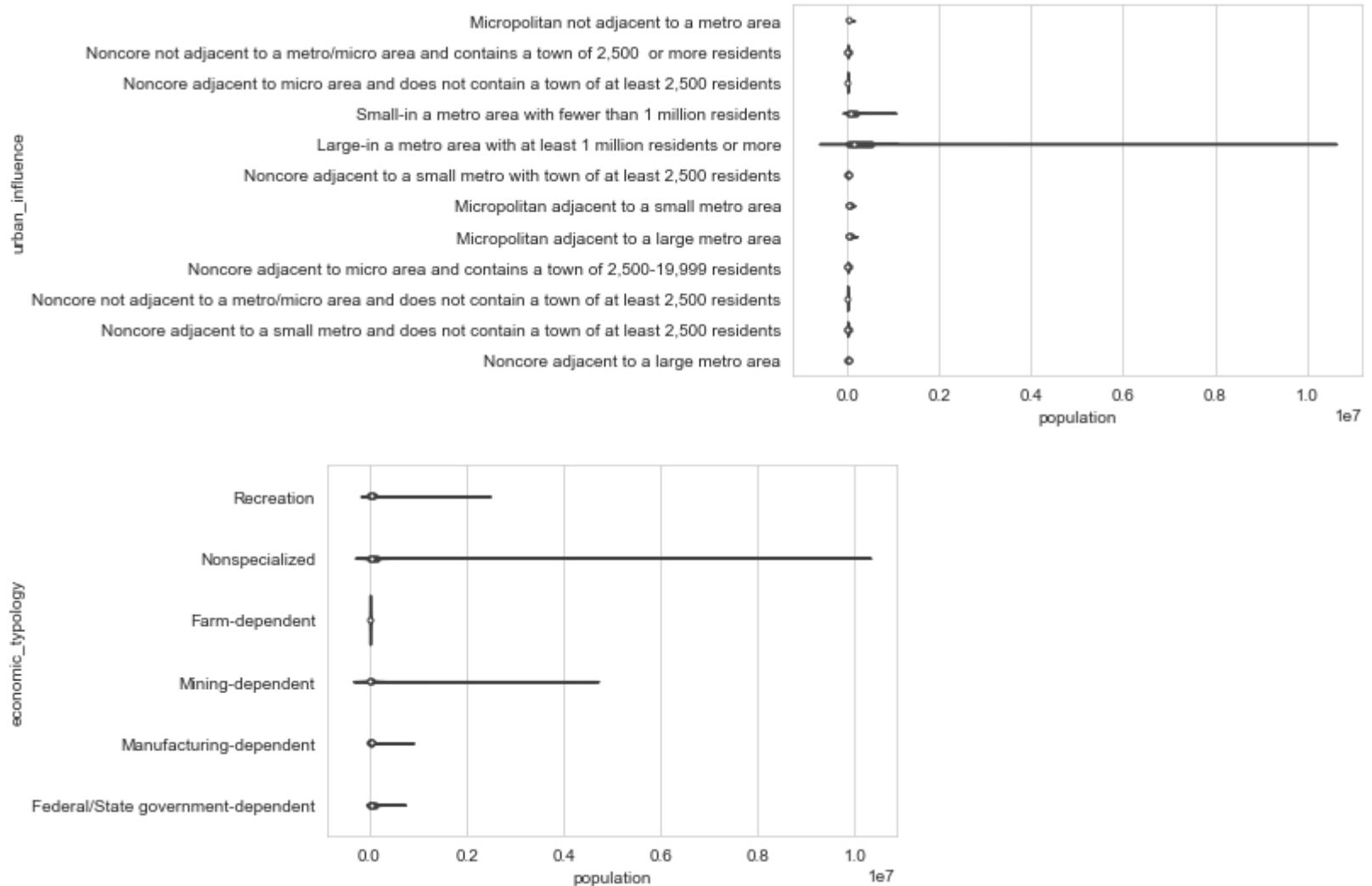
Out[329]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [330]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'population'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

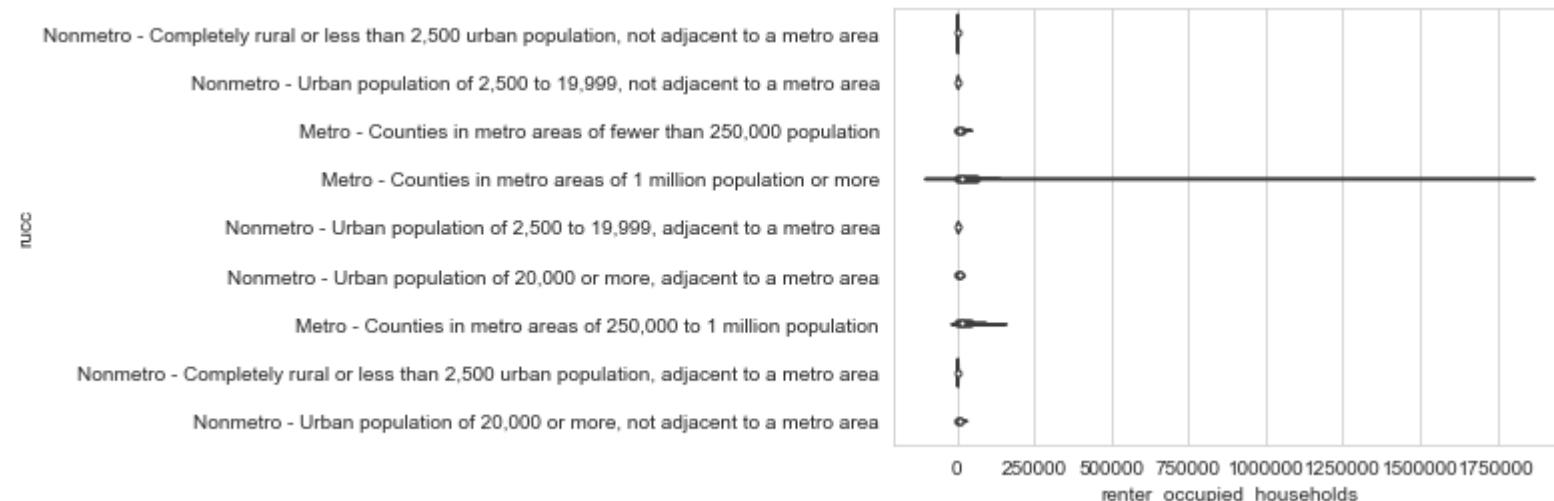
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

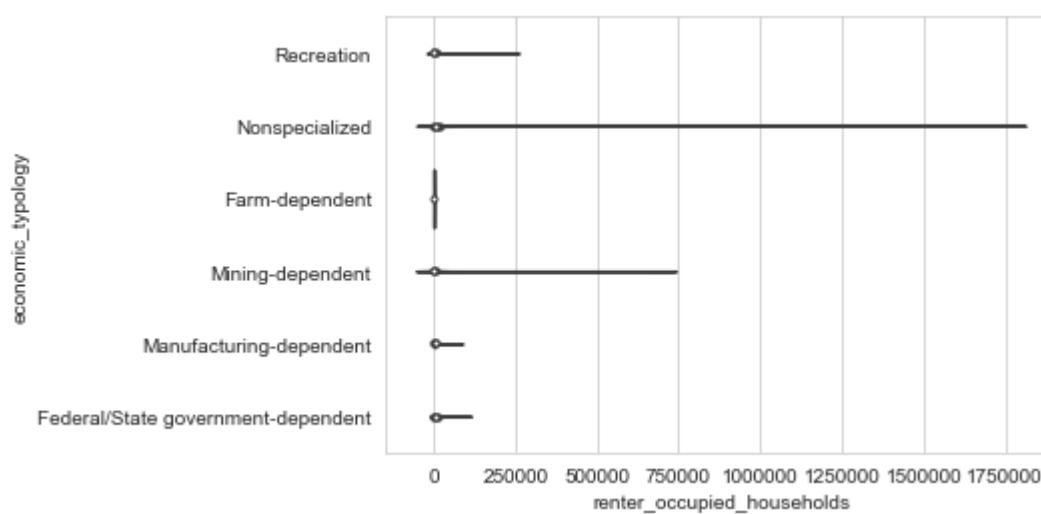
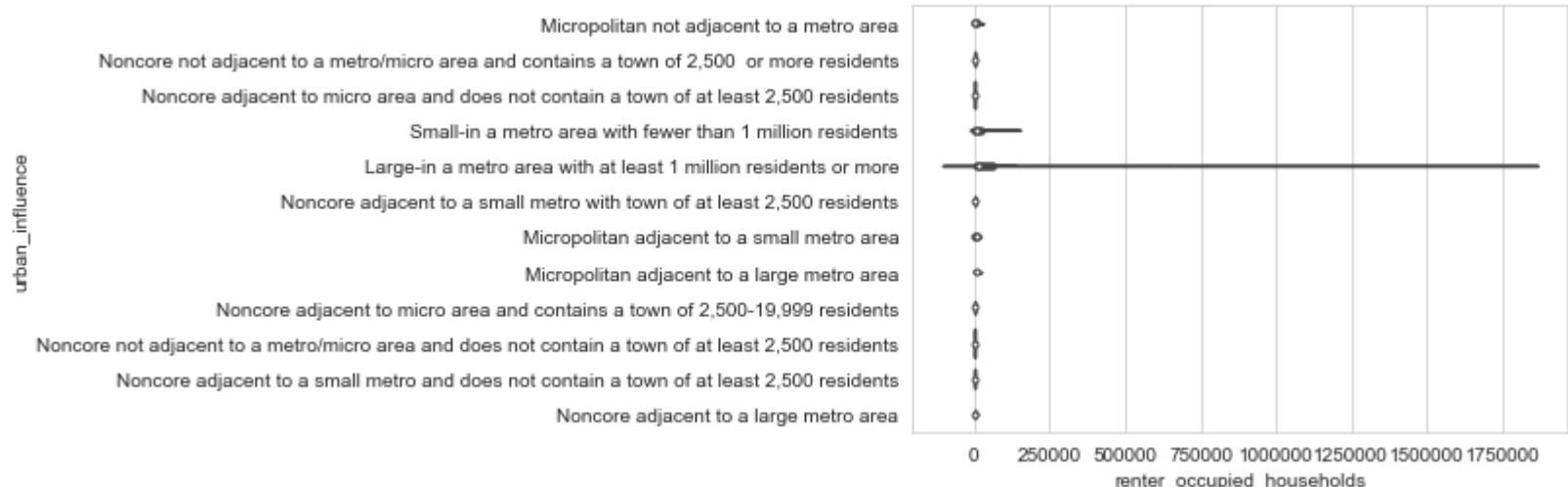




```
In [331]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'renter_occupied_households'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```





In [332]: num_cols

Out[332]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [333]: train_values_labels['rucc'].value_counts()
```

```
Out[333]: Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area    287  
Metro - Counties in metro areas of 1 million population or more                      216  
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area          186  
Metro - Counties in metro areas of 250,000 to 1 million population                  182  
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area 165  
Metro - Counties in metro areas of fewer than 250,000 population                   157  
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area 107  
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area                 99  
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area            41  
Name: rucc, dtype: int64
```

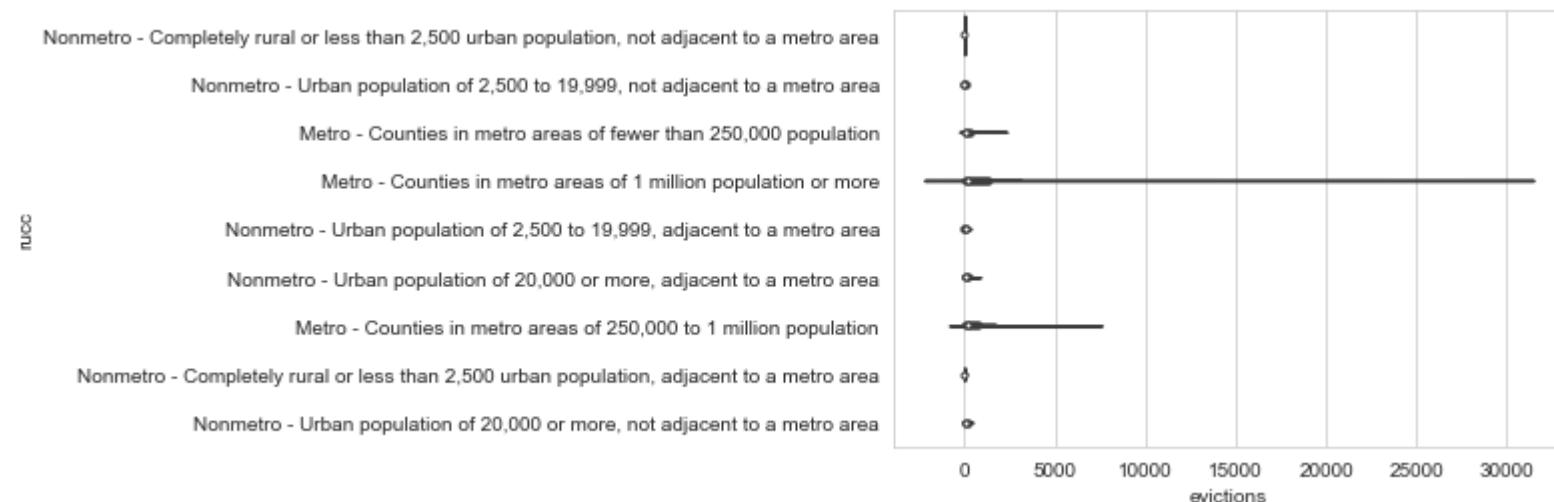
```
In [334]: train_values_labels.columns
```

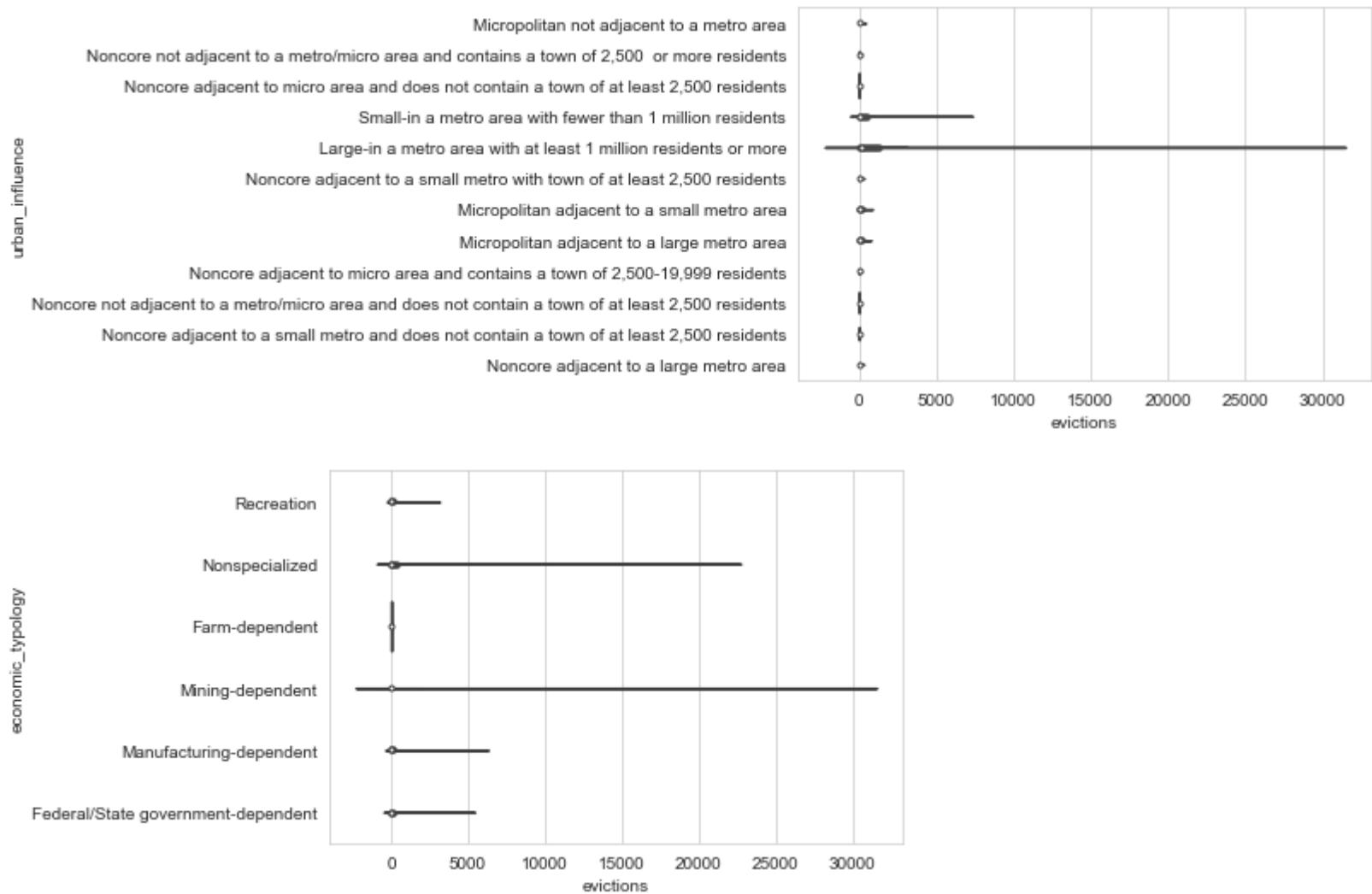
```
Out[334]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

In [335]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'evictions'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```



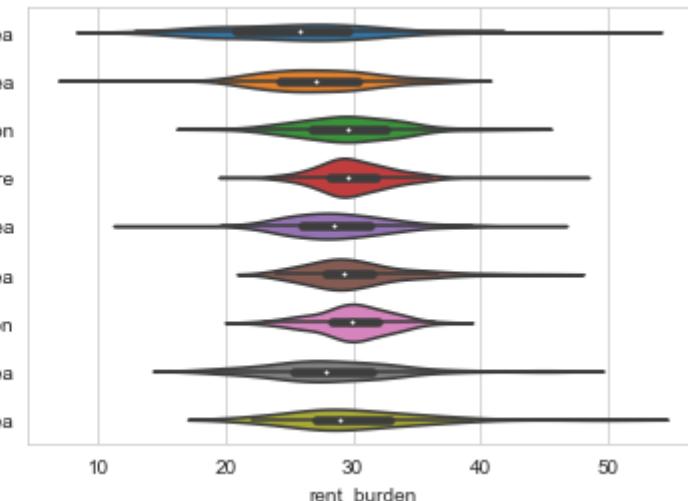


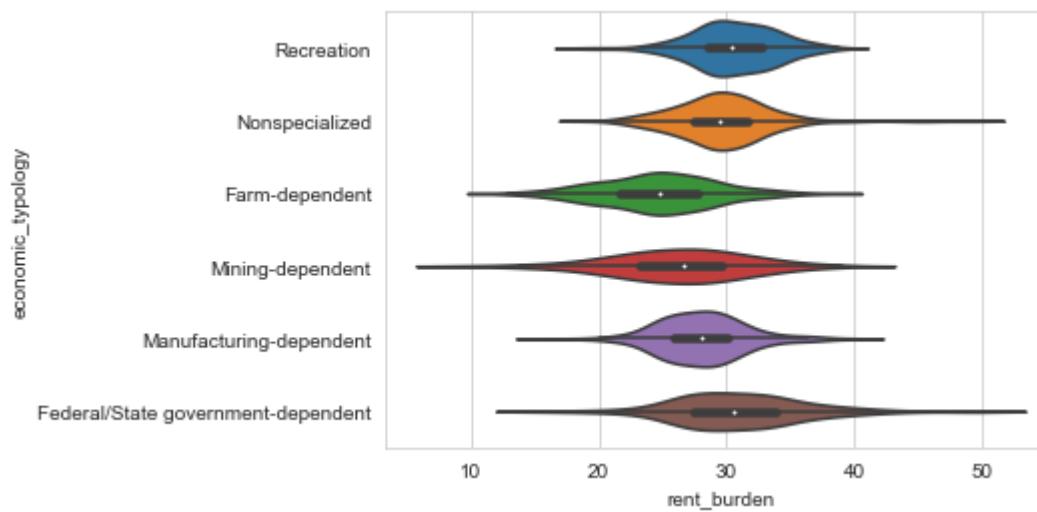
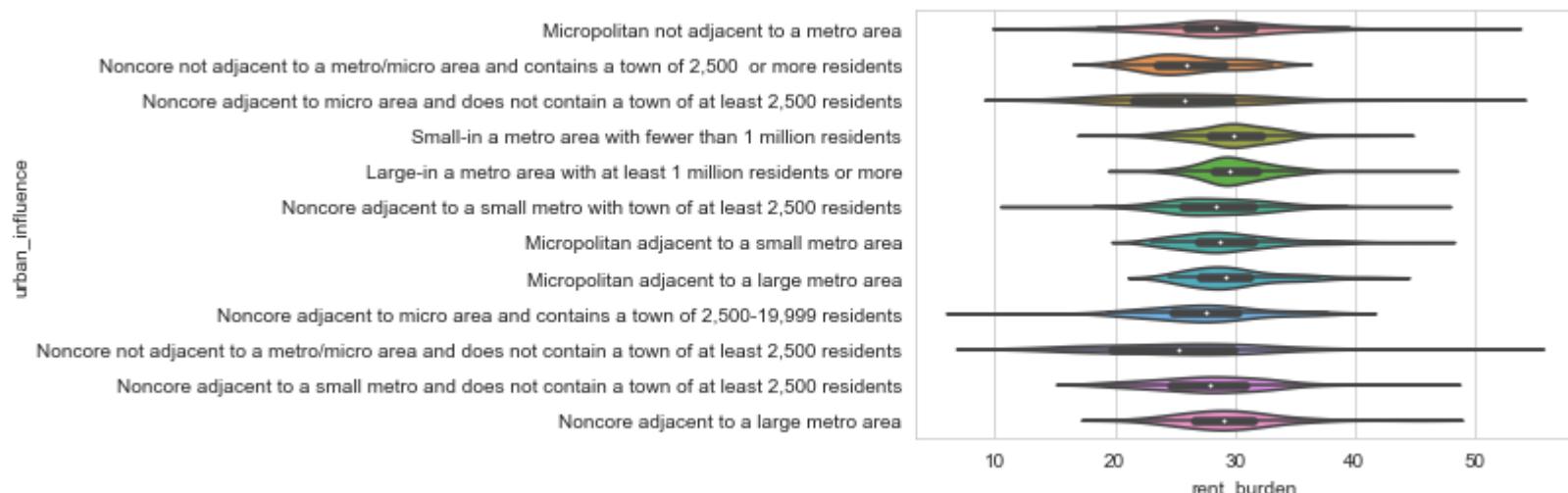
```
In [336]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'rent_burden'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more

Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

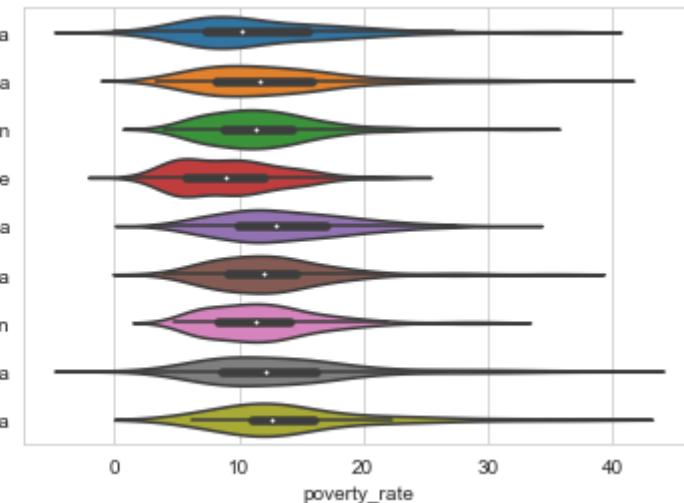


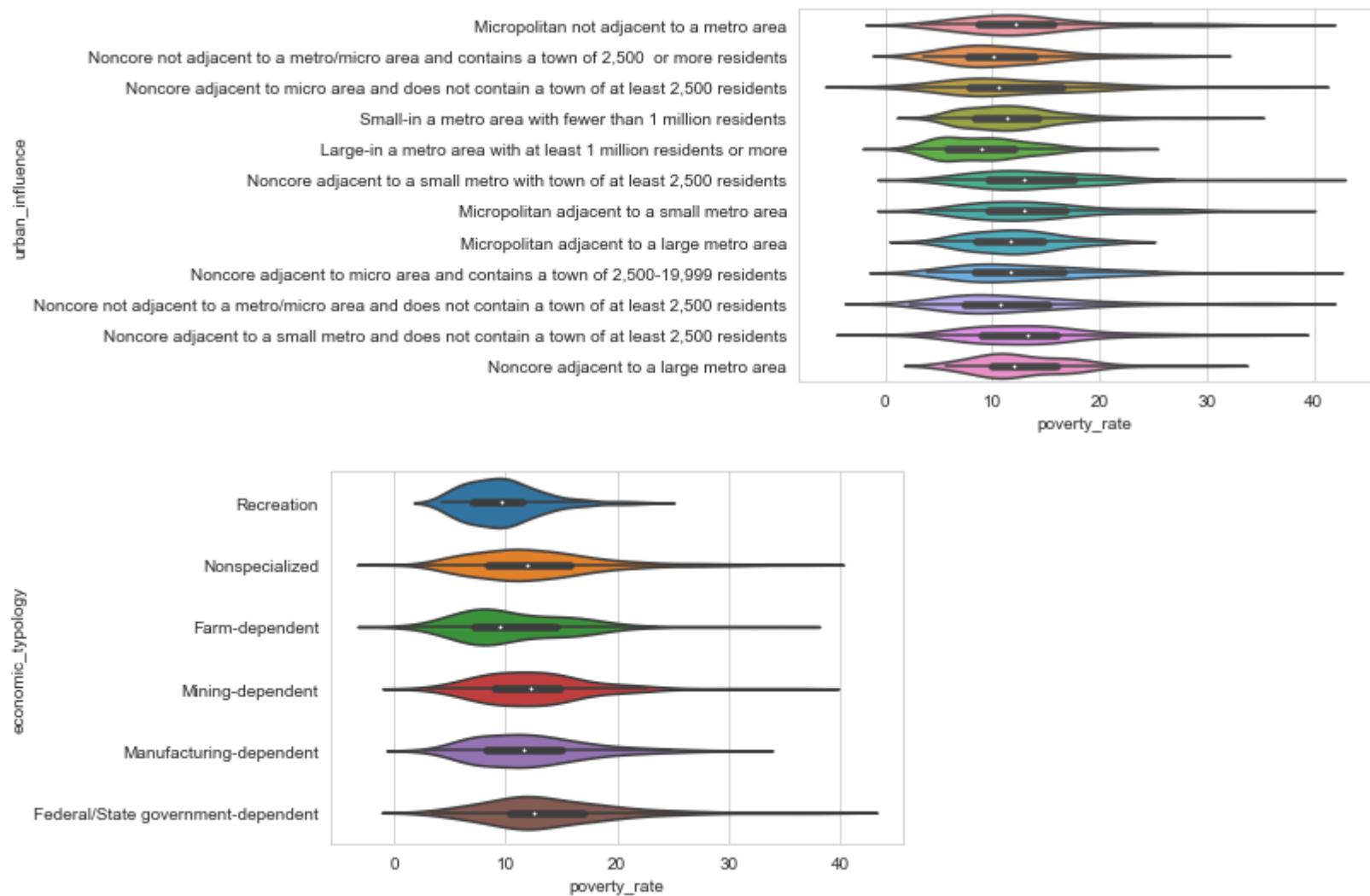



```
In [337]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'poverty_rate'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





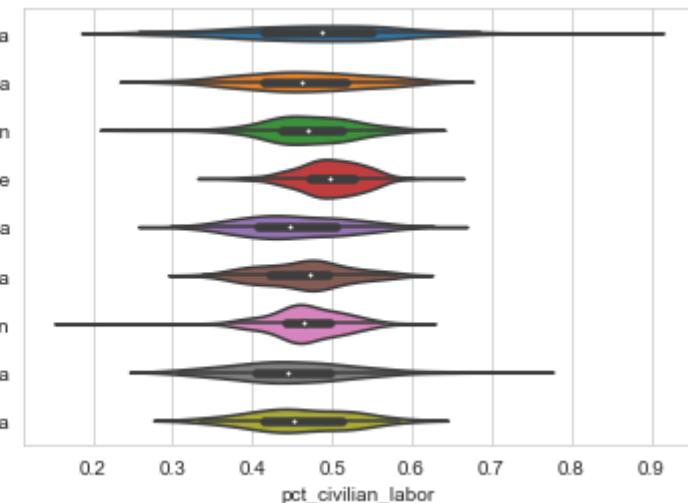
In [338]: num_cols

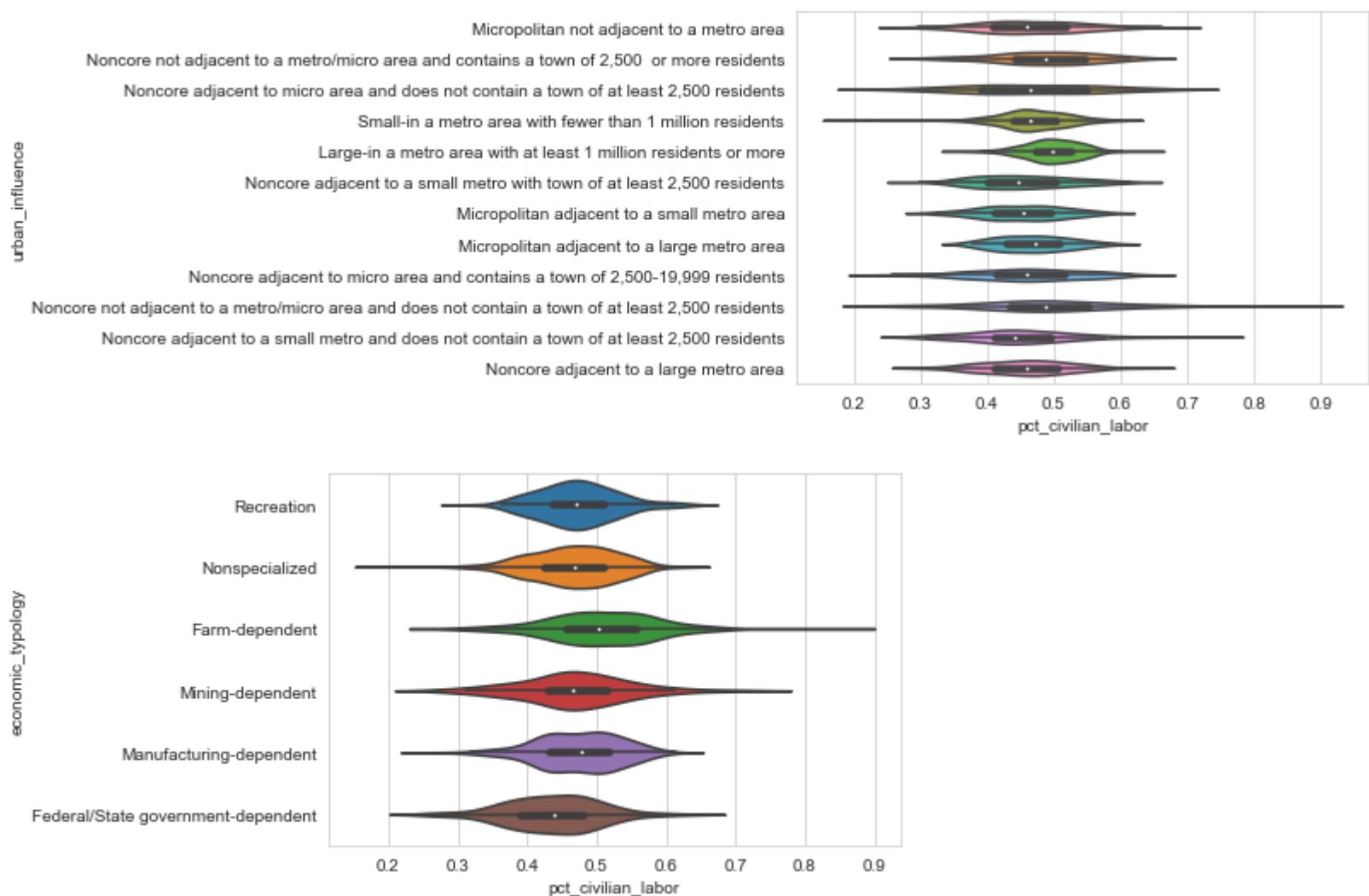
Out[338]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [339]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_civilian_labor'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area



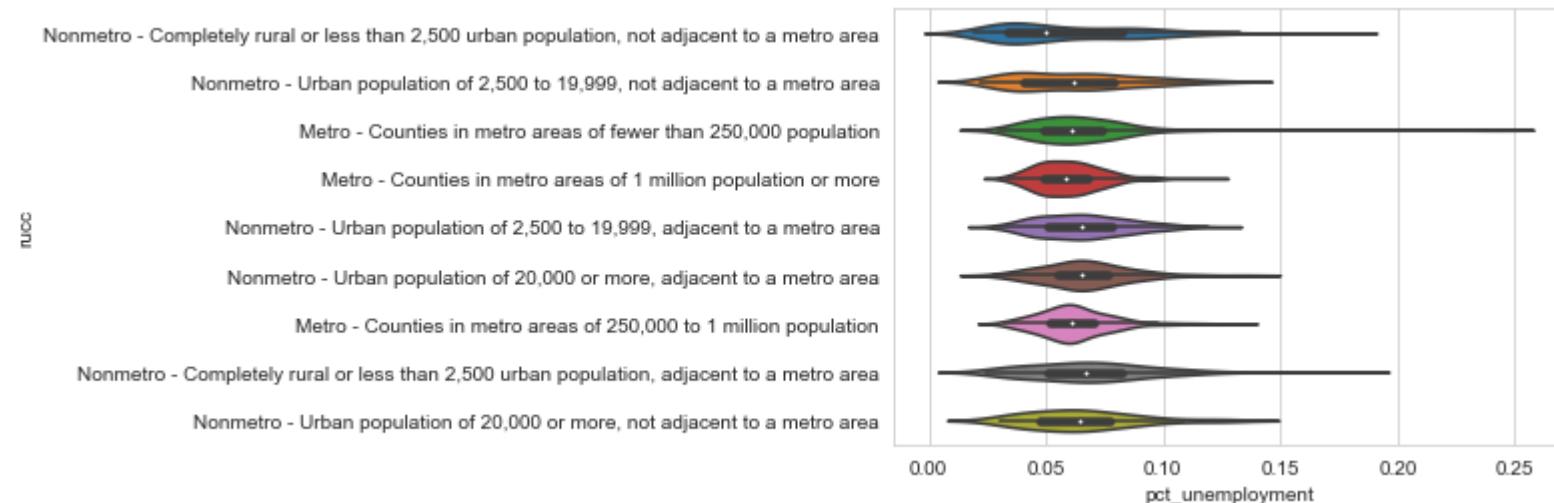


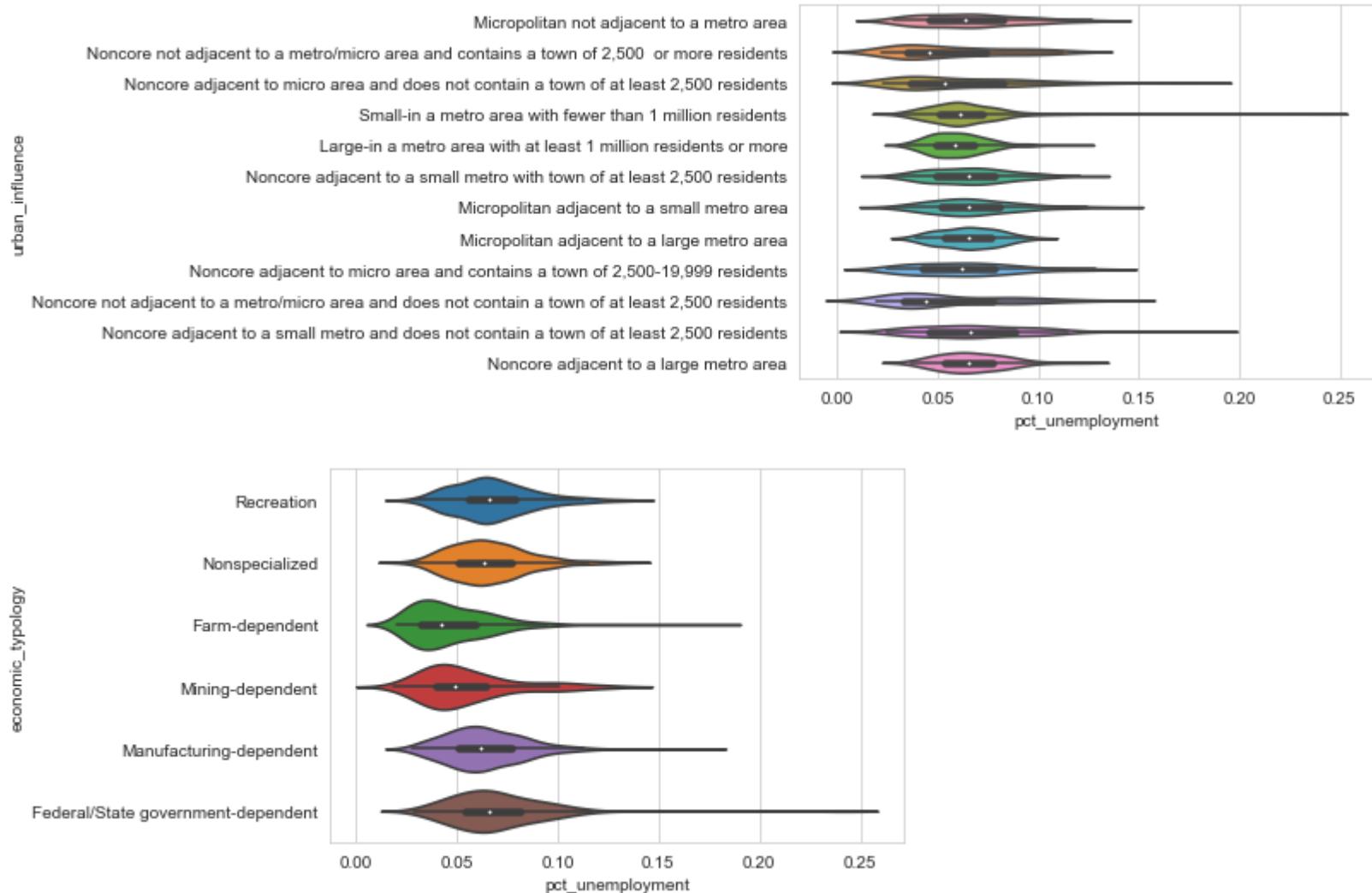
In [340]: num_cols

Out[340]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [341]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_unemployment'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

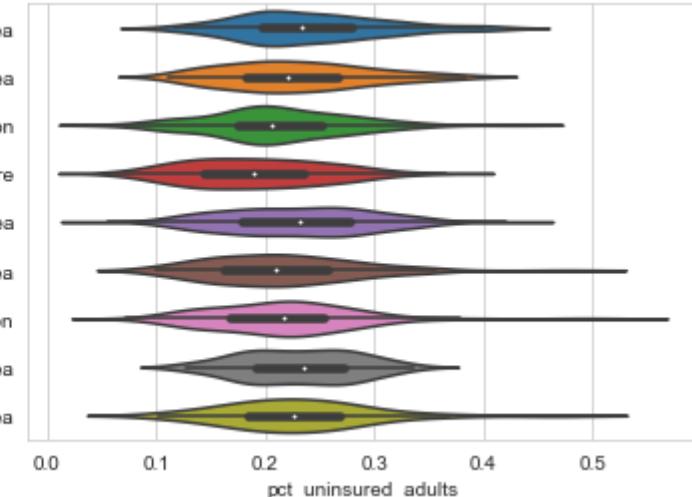


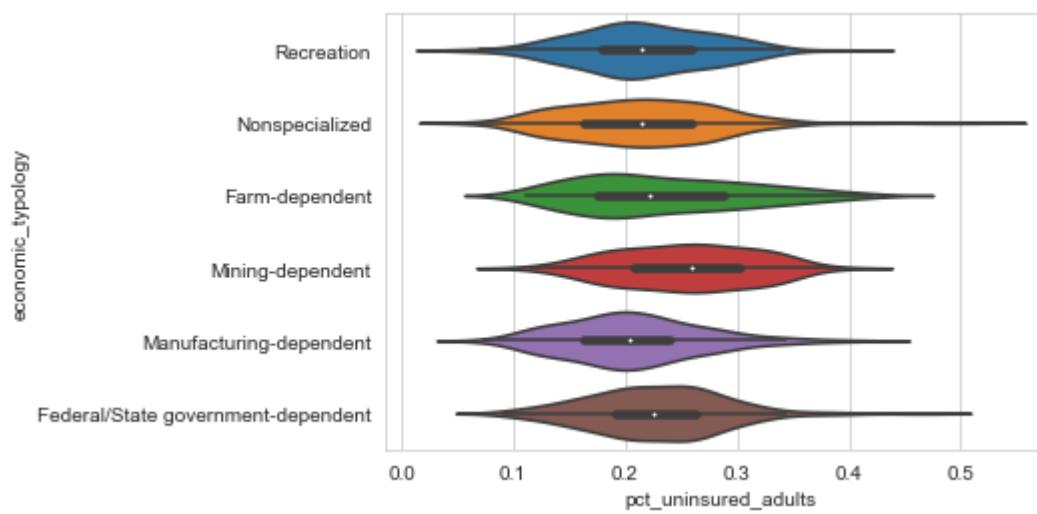
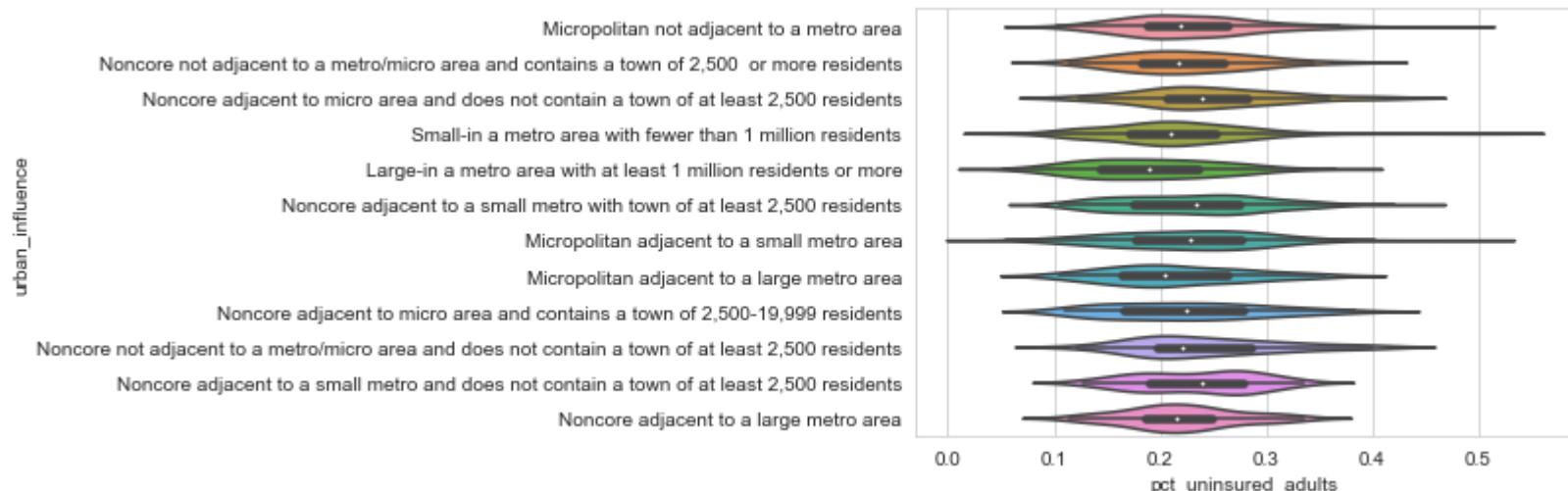


```
In [342]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_uninsured_adults'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area



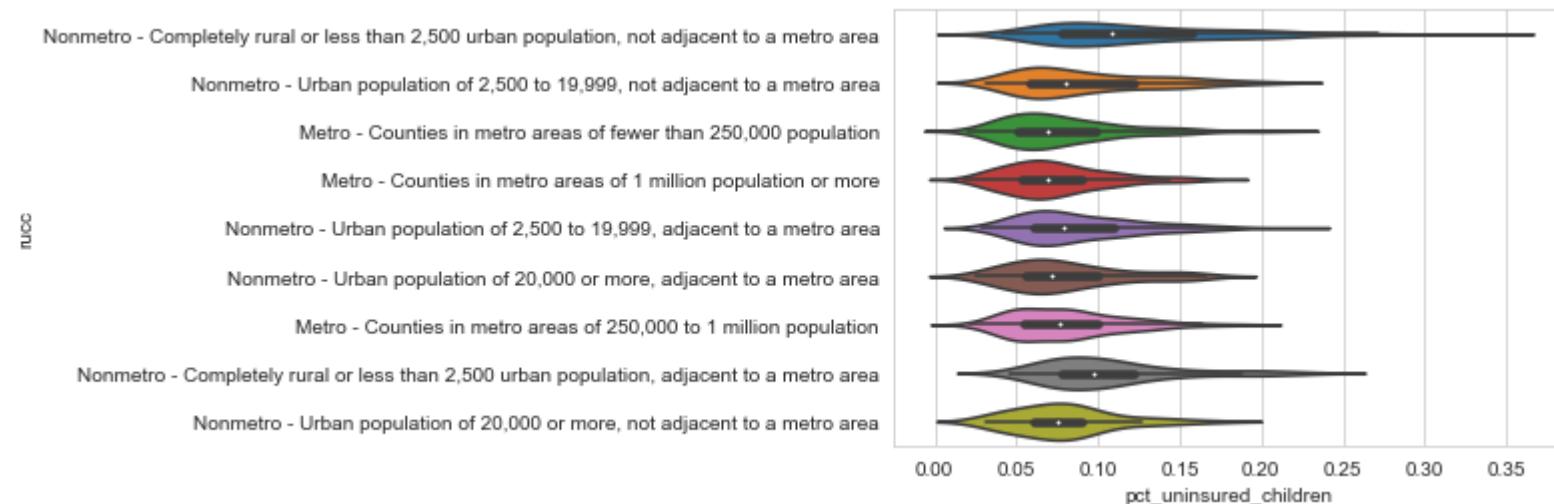


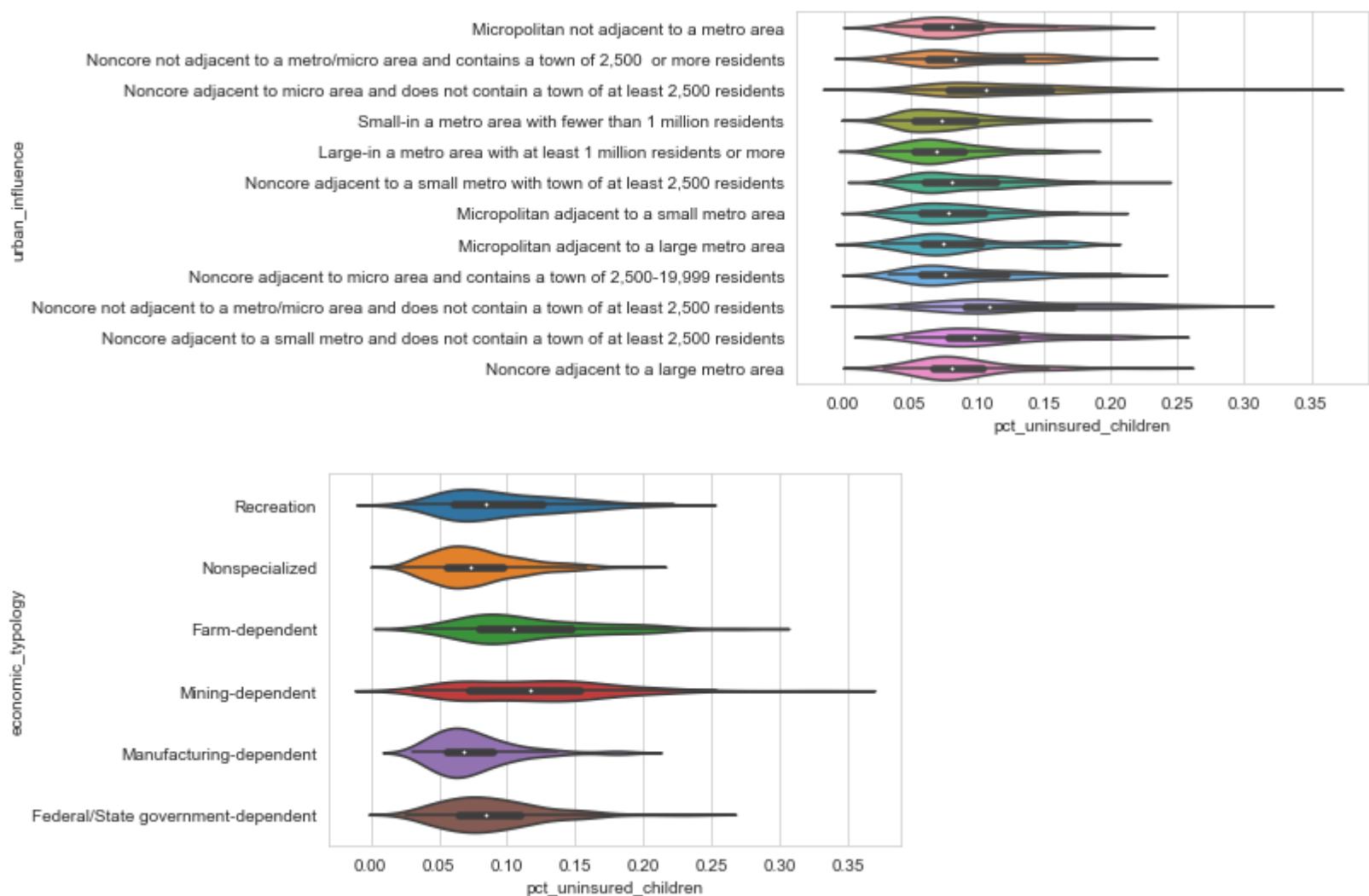
In [343]: num_cols

Out[343]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [344]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_uninsured_children'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

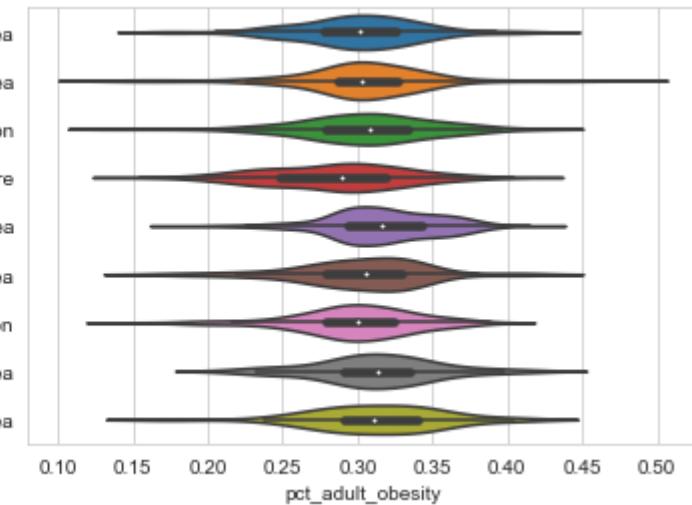


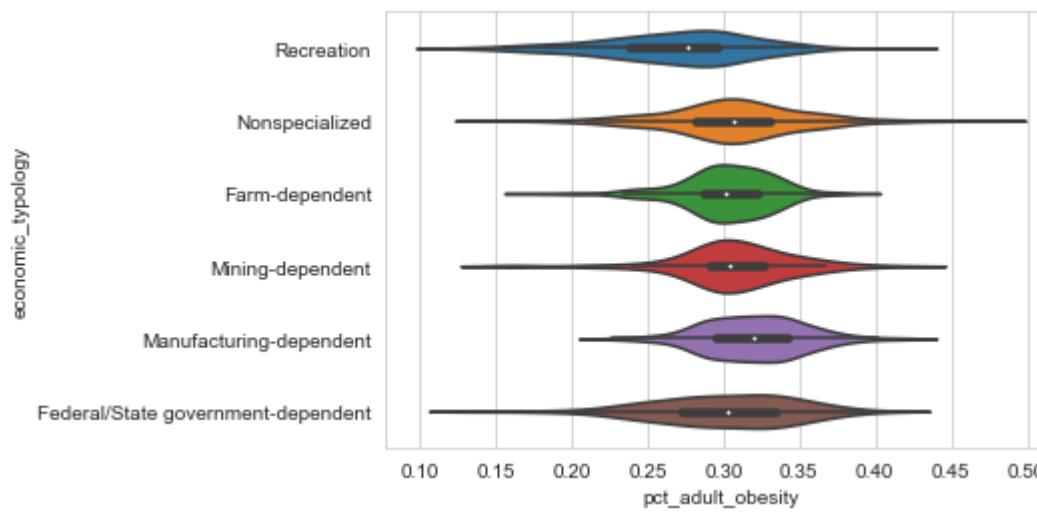
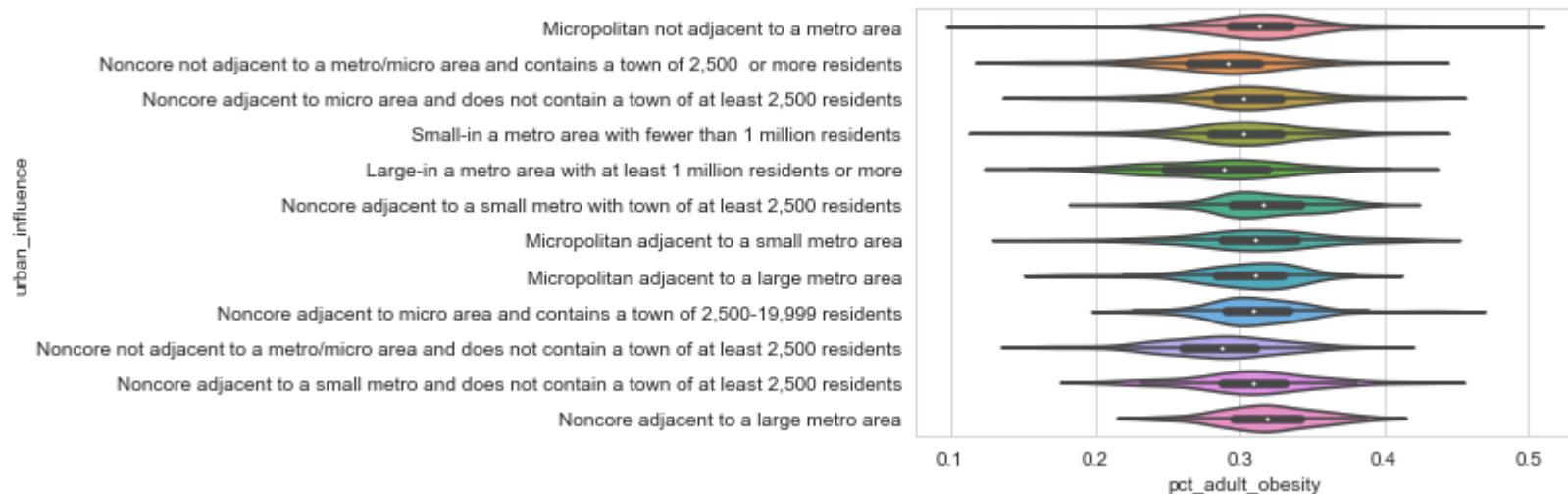


```
In [345]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adult_obesity'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

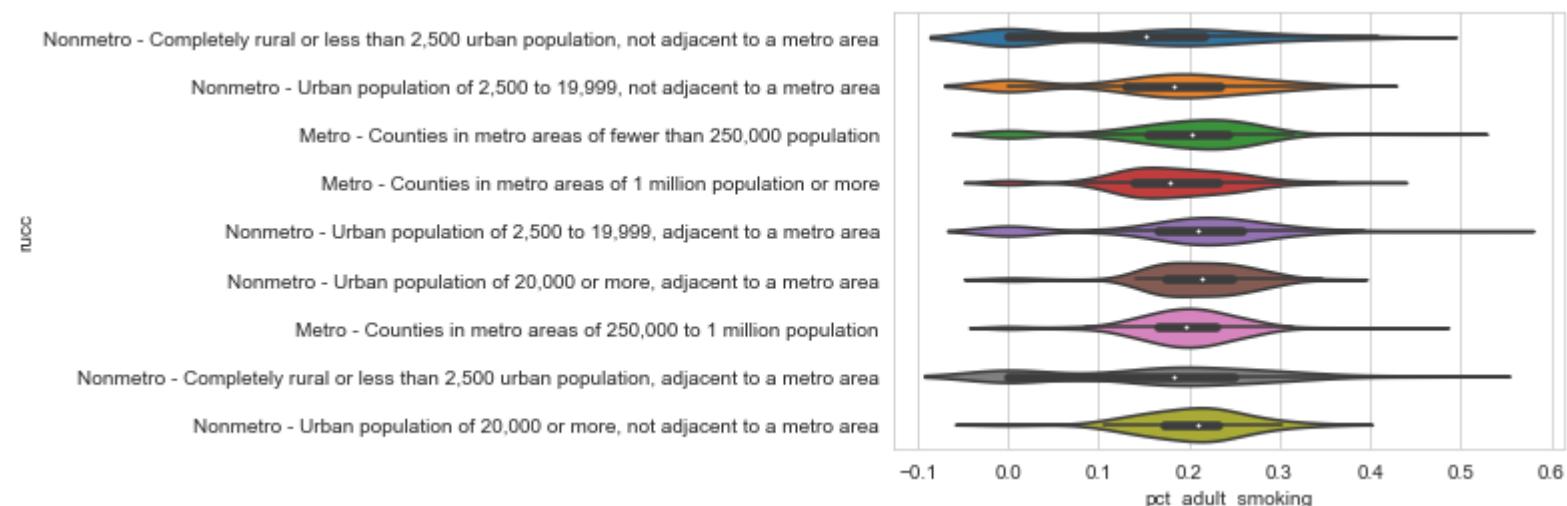
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

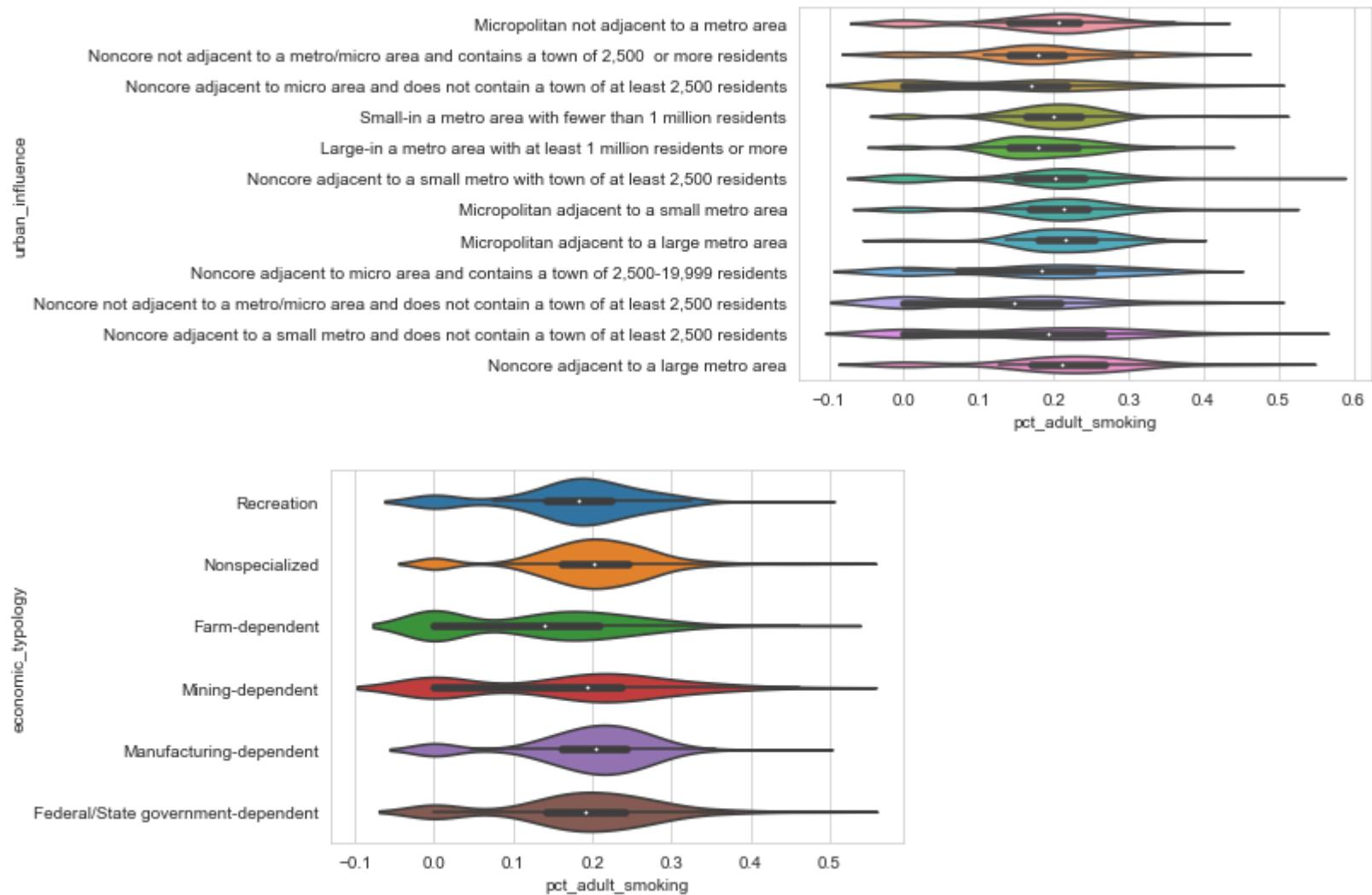





```
In [346]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adult_smoking'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

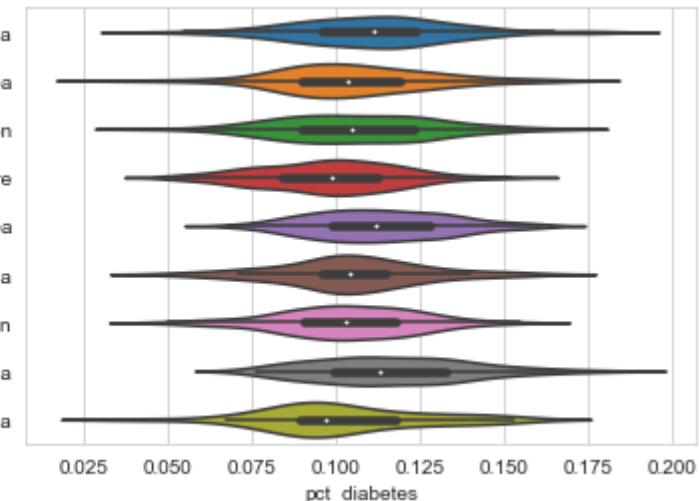


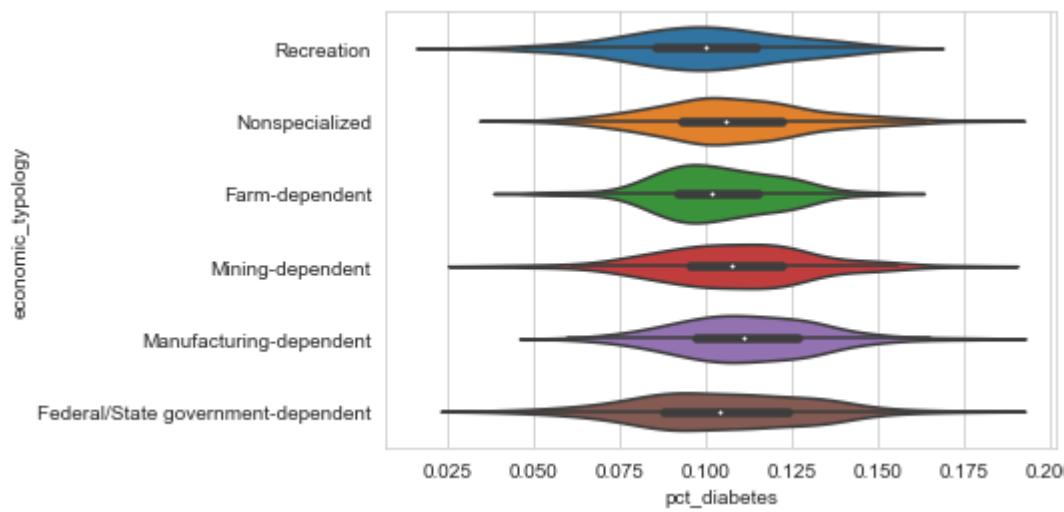
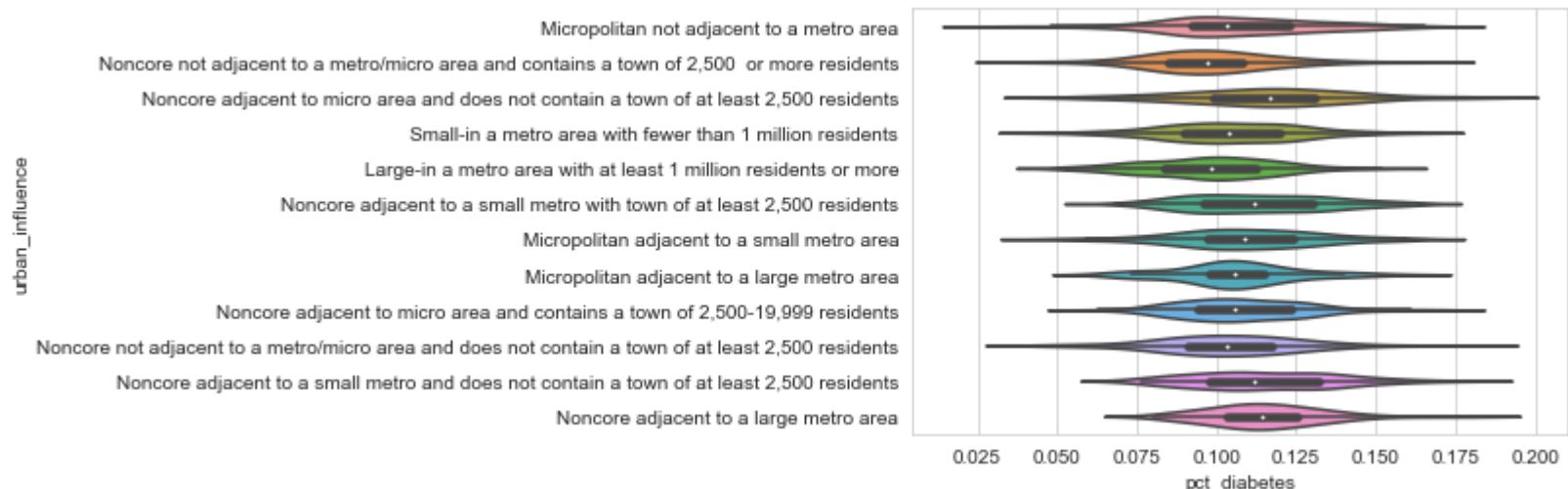


```
In [347]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_diabetes'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

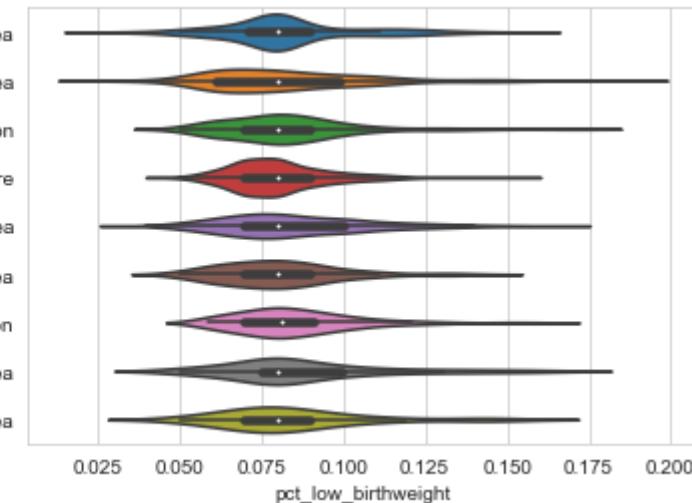


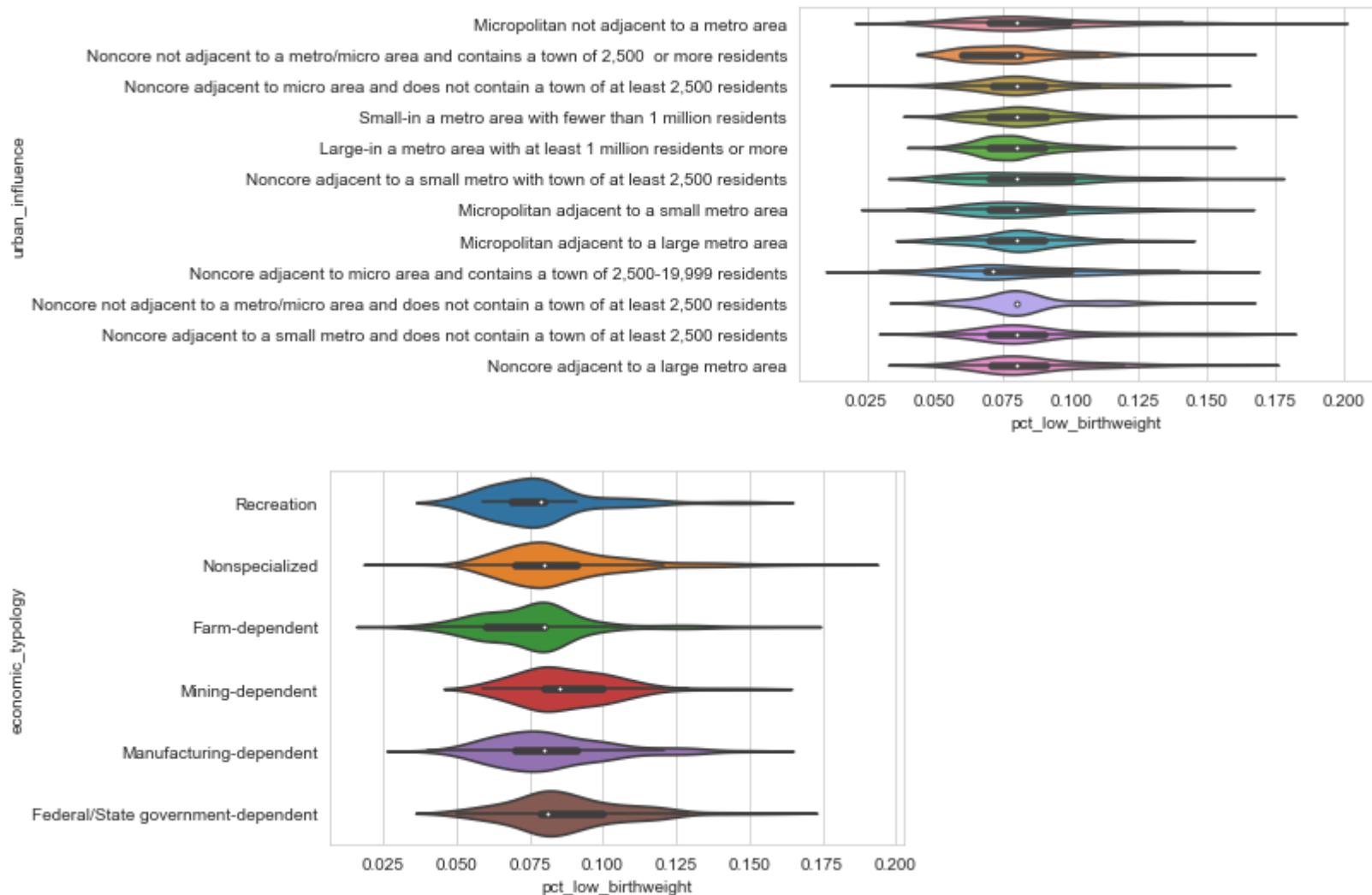



```
In [348]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_low_birthweight'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





In [349]: num_cols

Out[349]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

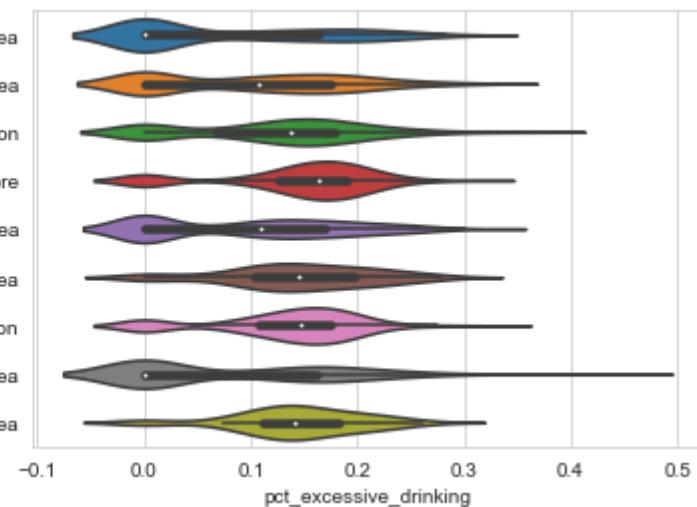
In [350]: num_cols

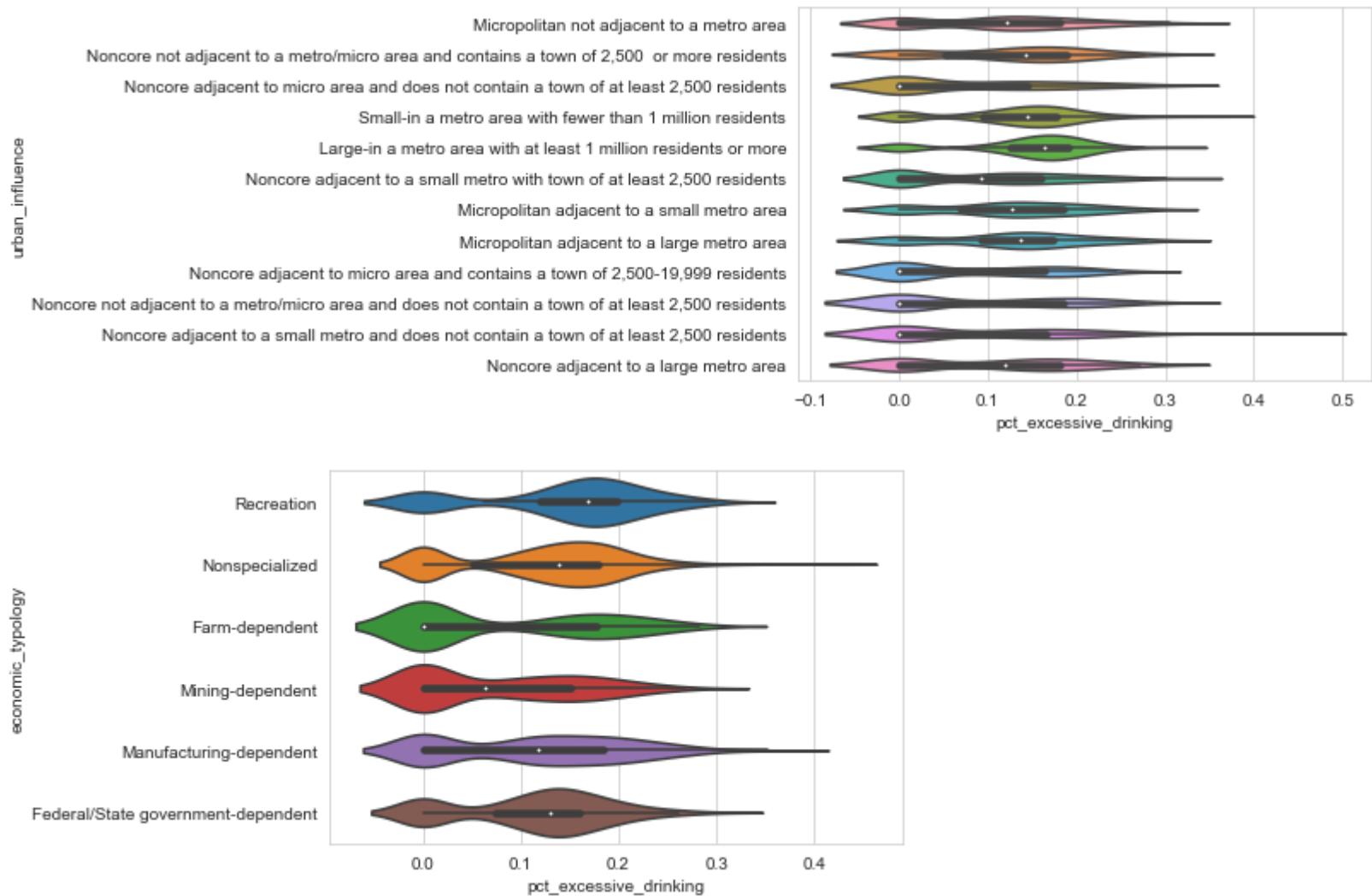
Out[350]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [351]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_excessive_drinking'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





```
In [352]: train_values_labels['urban_influence'].value_counts()
```

```
Out[352]: Small-in a metro area with fewer than 1 million residents           339
Large-in a metro area with at least 1 million residents or more             216
Noncore adjacent to a small metro with town of at least 2,500 residents     168
Micropolitan not adjacent to a metro area                                119
Micropolitan adjacent to a small metro area                            106
Noncore adjacent to micro area and contains a town of 2,500-19,999 residents  80
Noncore adjacent to a large metro area                               79
Noncore adjacent to a small metro and does not contain a town of at least 2,500 residents 77
Noncore adjacent to micro area and does not contain a town of at least 2,500 residents 73
Noncore not adjacent to a metro/micro area and does not contain a town of at least 2,500 residents 67
Micropolitan adjacent to a large metro area                           63
Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents 53
Name: urban_influence, dtype: int64
```

```
In [353]: small = train_values_labels[train_values_labels['urban_influence'] == 'Noncore adjacent to micro area and contains small'
```

```
Out[353]:
```

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
19	19	125	5	17479.0	1801.0	22.804	12.0	30.329	0.968958	0.001
22	22	899	33	10111.0	920.0	21.095	3.0	28.268	0.908087	0.001
31	31	1057	38	10591.0	999.0	26.133	1.0	21.008	0.539935	0.011
37	37	1254	45	6179.0	593.0	21.550	1.0	21.684	0.917266	0.001
40	40	751	14	20981.0	2437.0	29.936	63.0	32.960	0.436677	0.521
...
1433	1433	1326	45	8003.0	768.0	25.455	5.0	22.697	0.946675	0.001
1477	1477	1199	14	18394.0	2023.0	30.032	1.0	23.085	0.696391	0.181
1487	1487	373	45	9637.0	1055.0	27.954	7.0	25.519	0.945073	0.001
1521	1521	467	4	22505.0	2593.0	28.002	43.0	25.894	0.968066	0.001
1558	1558	1560	45	18837.0	2144.0	28.649	7.0	26.515	0.925500	0.011

80 rows × 85 columns

In [354]:

```
for col in small:  
    small[col].describe()  
    print(col,small[col].describe(),'\n')
```

```
row_id count      80.000000
```

```
mean     737.737500
```

```
std      426.222612
```

```
min      19.000000
```

```
25%     405.750000
```

```
50%     738.500000
```

```
75%    1080.500000
```

```
max     1558.000000
```

```
Name: row_id, dtype: float64
```

```
county_code count      80.000000
```

```
mean     792.550000
```

```
std      434.690837
```

```
min      5.000000
```

```
25%     423.250000
```

```
50%     768.500000
```

```
75%    1128.750000
```

```
max     1560.000000
```

```
Name: county_code, dtype: float64
```

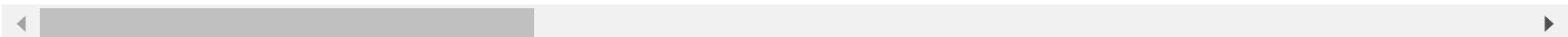
```
In [355]: train_values_labels[train_values_labels['evictions'] == -1]
```

Out[355]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	32	32	435	28	1043.0	125.0	25.425	-1.0	17.492	0.971436
	39	39	1502	41	21758.0	2017.0	23.901	-1.0	29.654	0.863019
	61	61	1319	7	18240.0	1793.0	29.867	-1.0	23.228	0.169795
	100	100	1193	7	1426.0	149.0	19.095	-1.0	31.547	0.755442
	101	101	1374	15	10332.0	923.0	25.519	-1.0	24.556	0.660098

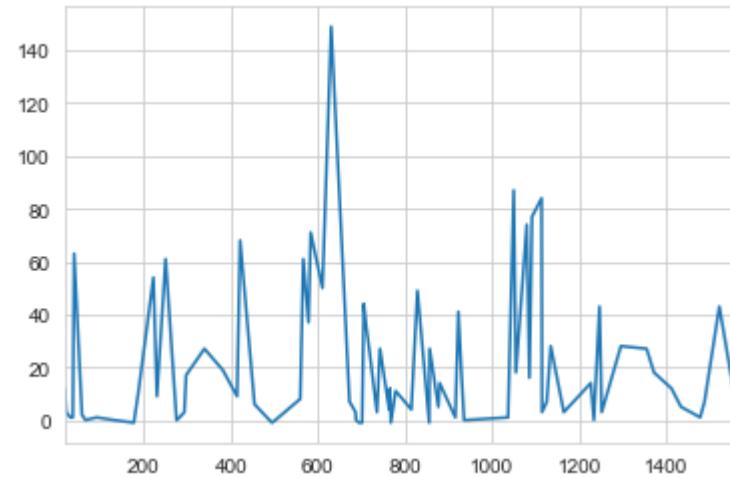
	1480	1480	1305	7	3460.0	370.0	28.363	-1.0	16.899	0.666335
	1513	1513	834	7	7233.0	625.0	29.491	-1.0	28.766	0.155347
	1546	1546	128	10	7001.0	568.0	20.640	-1.0	33.245	0.936286
	1551	1551	1501	29	4239.0	306.0	20.508	-1.0	27.847	0.944484
	1555	1555	661	31	14055.0	729.0	18.999	-1.0	33.118	0.713404

82 rows × 85 columns



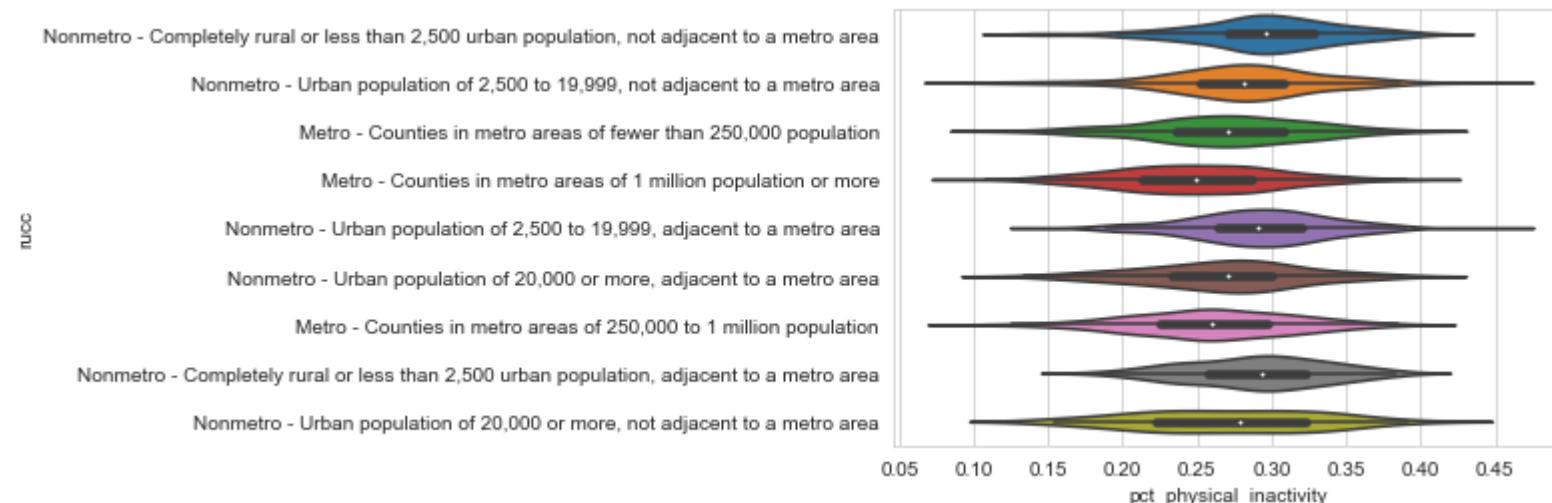
```
In [356]: small['evictions'].plot()
```

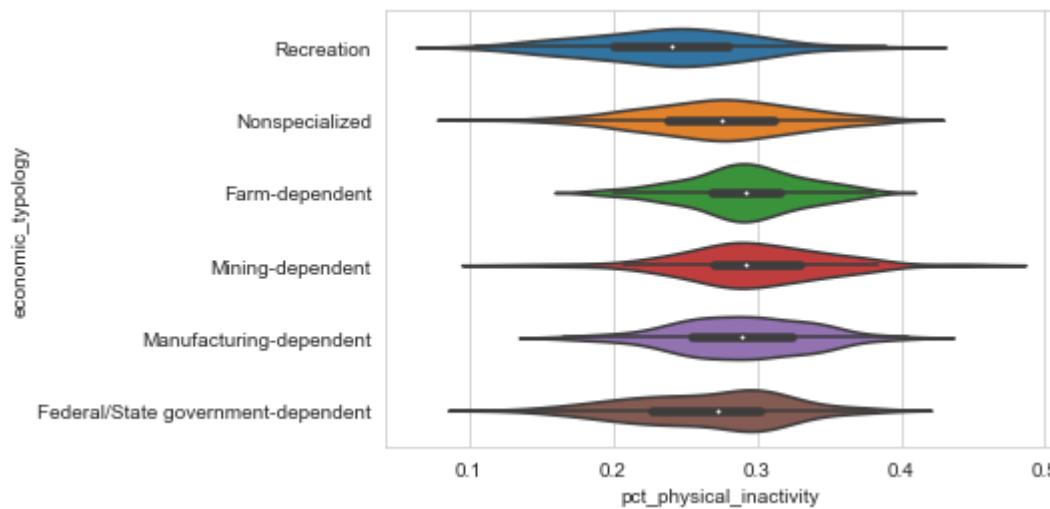
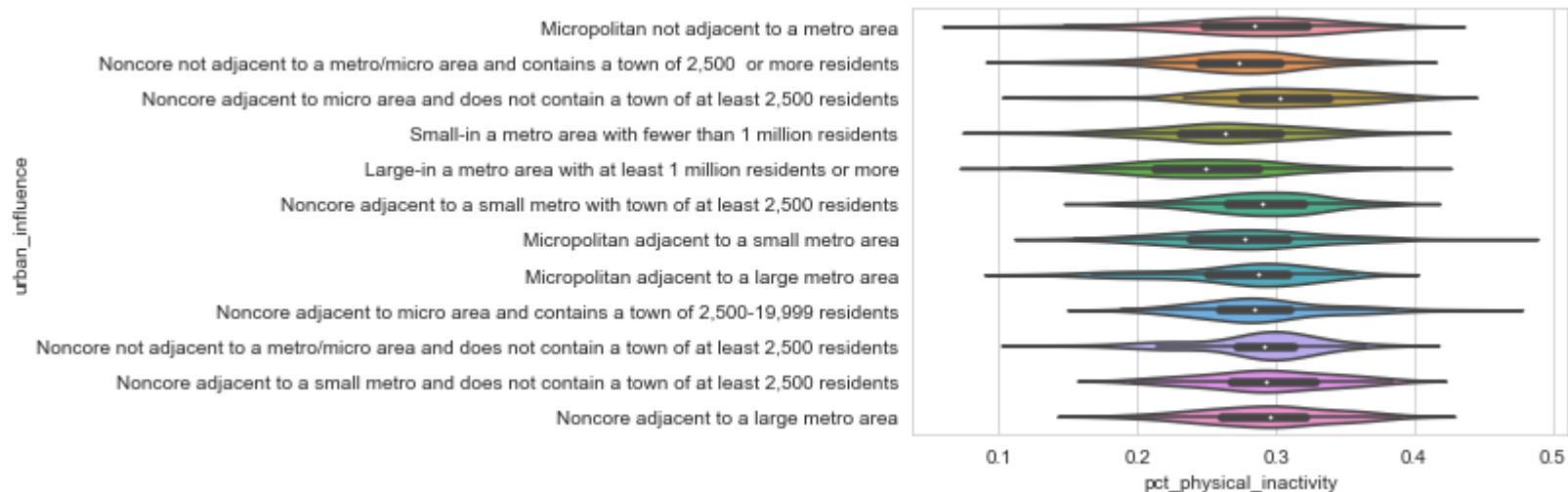
```
Out[356]: <matplotlib.axes._subplots.AxesSubplot at 0x243fb39ee80>
```



```
In [357]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_physical_inactivity'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```



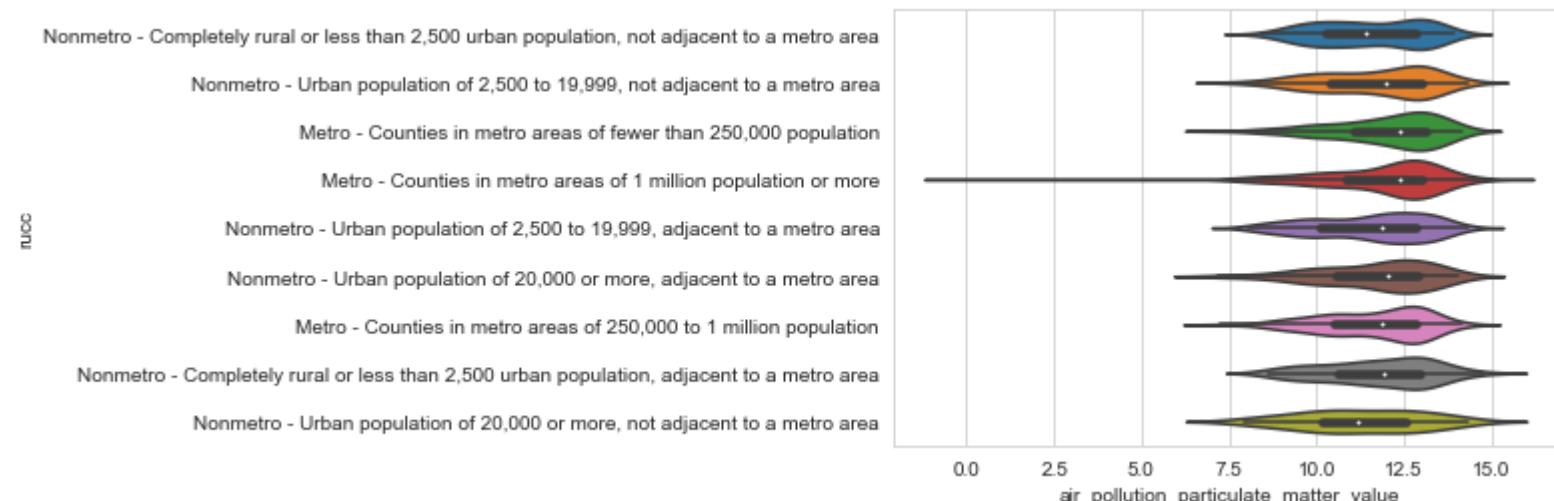


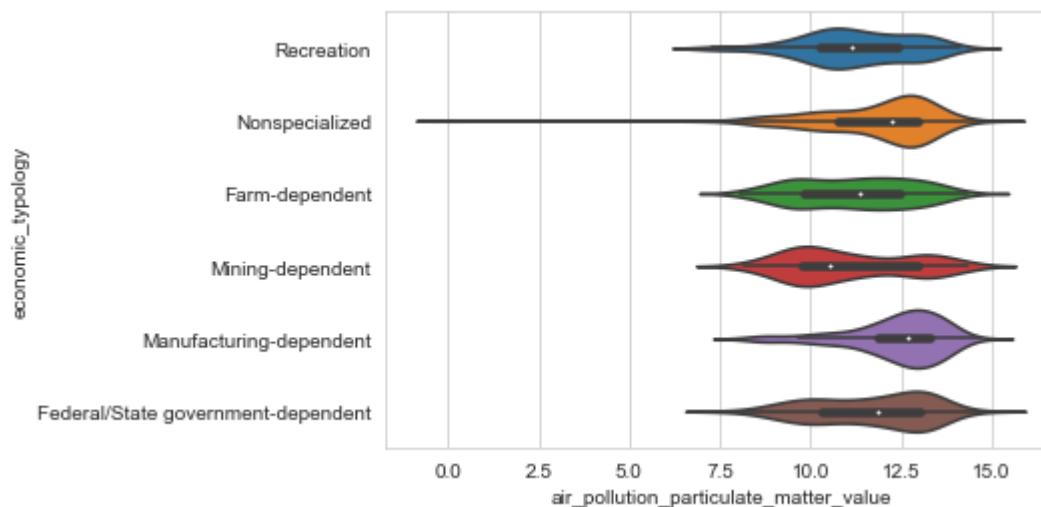
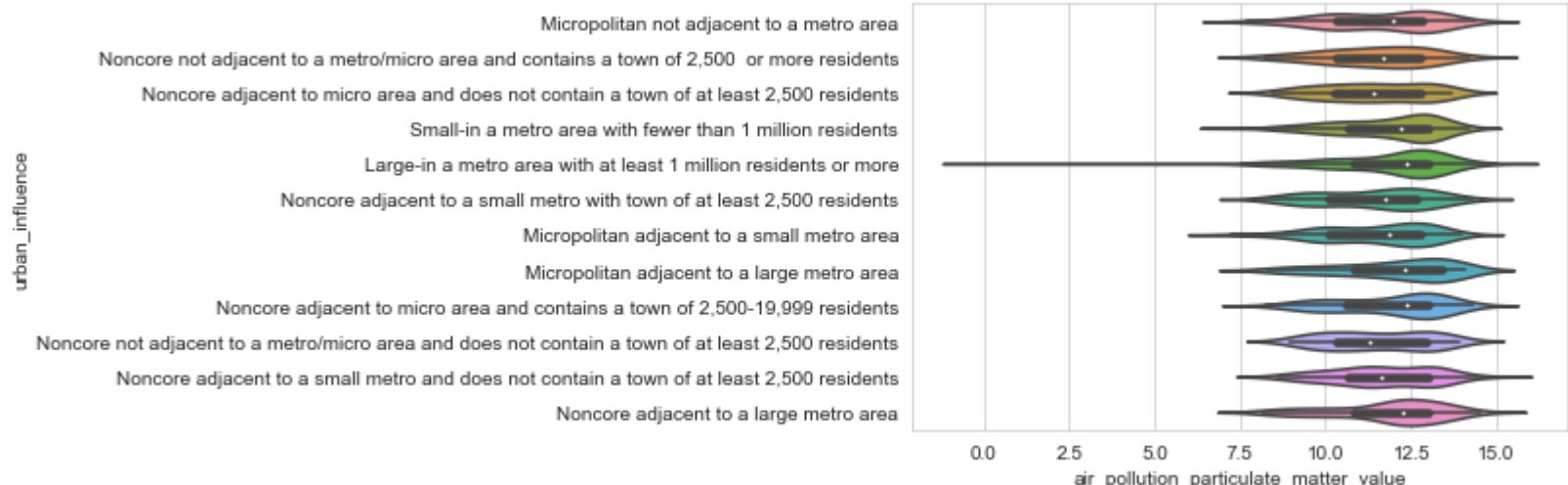
In [358]: num_cols

Out[358]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [359]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'air_pollution_particulate_matter_value'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```





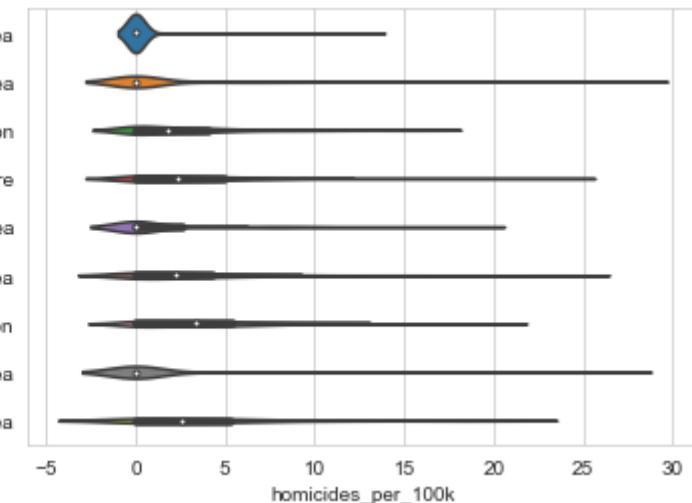
In [360]: num_cols

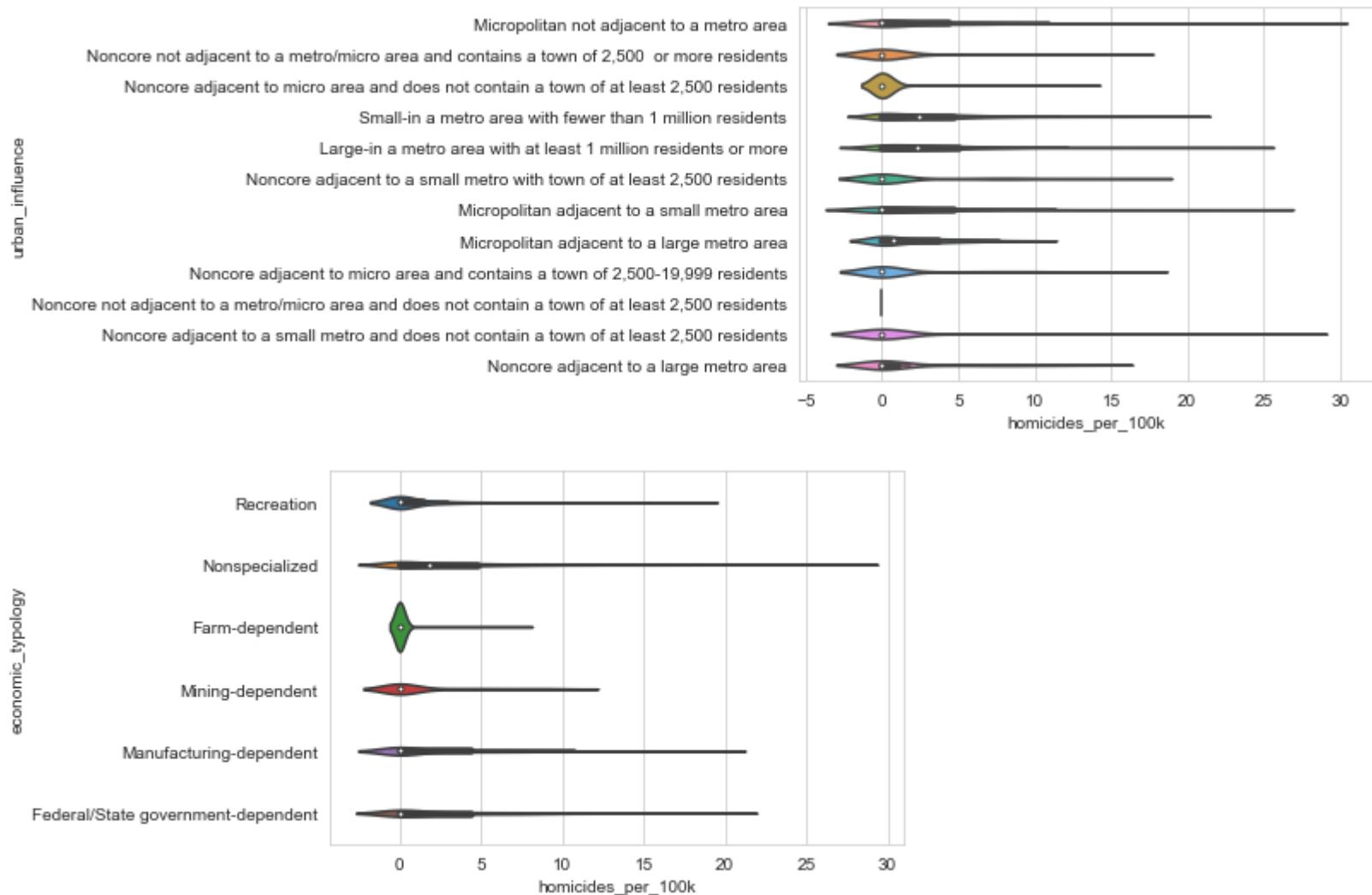
Out[360]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [361]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'homicides_per_100k'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





```
In [362]: train_values_labels['urban_influence'].value_counts()
```

```
Out[362]: Small-in a metro area with fewer than 1 million residents           339
          Large-in a metro area with at least 1 million residents or more       216
          Noncore adjacent to a small metro with town of at least 2,500 residents 168
          Micropolitan not adjacent to a metro area                           119
          Micropolitan adjacent to a small metro area                         106
          Noncore adjacent to micro area and contains a town of 2,500-19,999 residents 80
          Noncore adjacent to a large metro area                            79
          Noncore adjacent to a small metro and does not contain a town of at least 2,500 residents 77
          Noncore adjacent to micro area and does not contain a town of at least 2,500 residents 73
          Noncore not adjacent to a metro/micro area and does not contain a town of at least 2,500 residents 67
          Micropolitan adjacent to a large metro area                          63
          Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents 53
          Name: urban_influence, dtype: int64
```

```
In [363]: flatzone = train_values_labels[train_values_labels['urban_influence']=='Noncore not adjacent to a metro/micro area']
```

```
Out[363]:
```

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
27	27	974	28	4203.0	504.0	27.531	0.0	24.138	0.855246	0.001
32	32	435	28	1043.0	125.0	25.425	-1.0	17.492	0.971436	0.001
75	75	1349	16	18895.0	2824.0	27.891	35.0	30.305	0.843737	0.001
84	84	244	43	4763.0	518.0	21.716	0.0	22.497	0.967279	0.001
100	100	1193	7	1426.0	149.0	19.095	-1.0	31.547	0.755442	0.031
...
1412	1412	198	14	21531.0	2137.0	21.248	18.0	29.606	0.943763	0.001
1468	1468	1095	20	12676.0	1133.0	28.765	1.0	34.340	0.931585	0.041
1480	1480	1305	7	3460.0	370.0	28.363	-1.0	16.899	0.666335	0.001
1515	1515	1181	22	2742.0	270.0	17.127	0.0	19.399	0.925314	0.001
1523	1523	1262	38	1954.0	202.0	24.789	0.0	18.753	0.888687	0.001

67 rows × 85 columns

```
In [364]: train_values_labels.columns
```

```
Out[364]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [365]: flatzone['homicides_per_100k'].describe()
```

```
Out[365]: count    67.0
mean      0.0
std       0.0
min      0.0
25%      0.0
50%      0.0
75%      0.0
max      0.0
Name: homicides_per_100k, dtype: float64
```

```
In [366]: flatzone['motor_vehicle_crash_deaths_per_100k'].describe()
```

```
Out[366]: count    67.000000
mean     12.793582
std      17.338330
min     0.000000
25%     0.000000
50%     0.000000
75%     24.485000
max     52.170000
Name: motor_vehicle_crash_deaths_per_100k, dtype: float64
```

```
In [367]: flatzone['gross_rent']
```

```
Out[367]: 27      490
32      497
75      836
84      474
100     439
...
1412     625
1468     420
1480     662
1515     544
1523     607
Name: gross_rent, Length: 67, dtype: int64
```

```
In [368]: flatzone.isnull().any()
```

```
Out[368]: row_id           False
county_code        False
state             False
population        False
renter_occupied_households  False
...
pct_adults_with_some_collegelog  False
birth_rate_per_1klog      False
pct_civilian_laborlog    False
pct_unemploymentlog     False
pct_uninsured_adultslog  False
Length: 85, dtype: bool
```

```
In [369]: #train_values_labels.drop(flatzone.index, inplace = True)
```

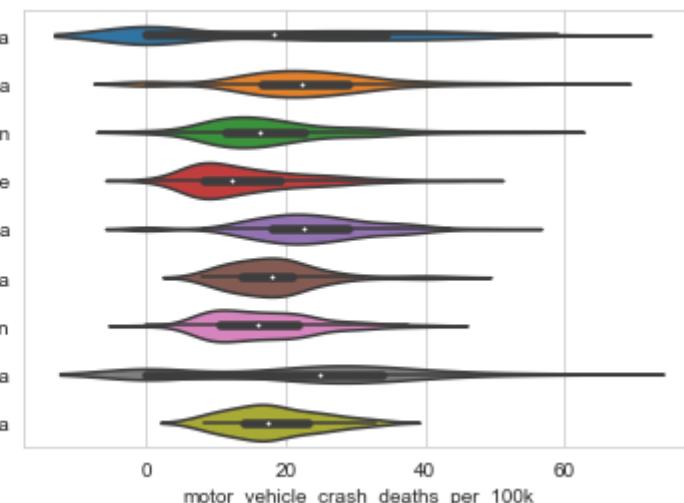
In [370]: num_cols

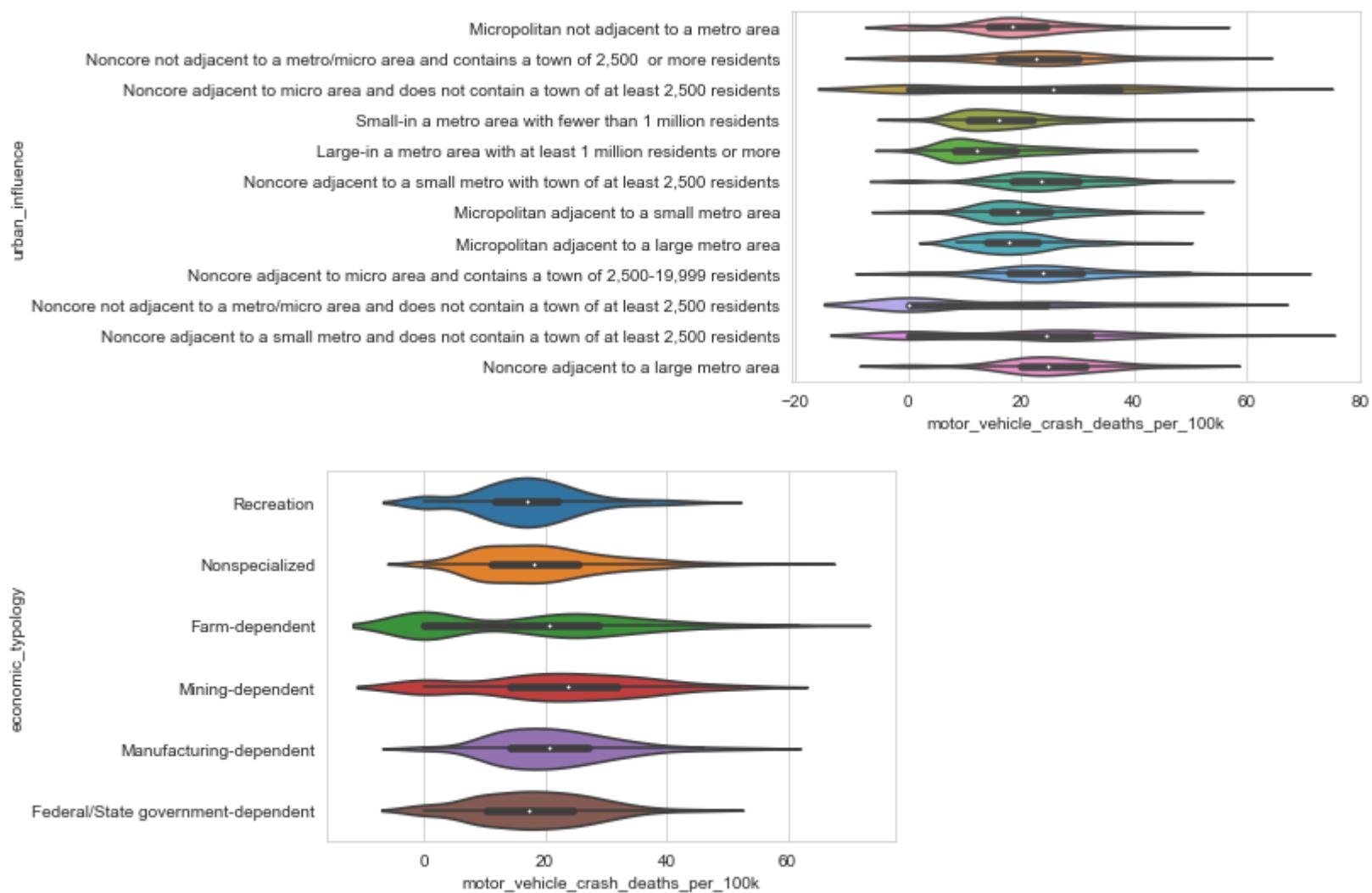
Out[370]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [371]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'motor_vehicle_crash_deaths_per_100k'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





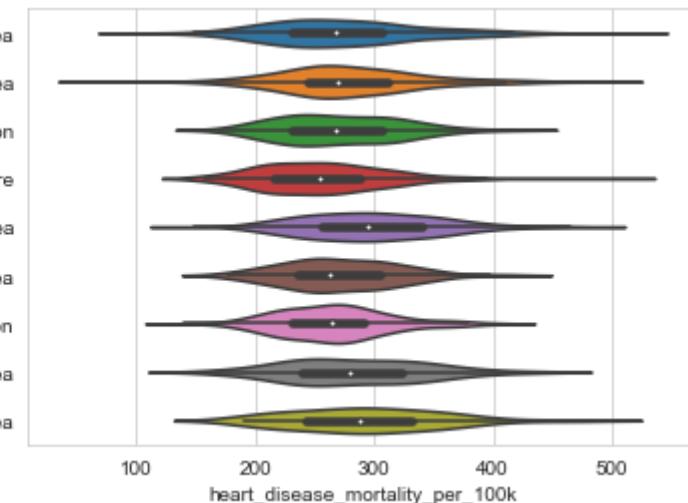
In [372]: num_cols

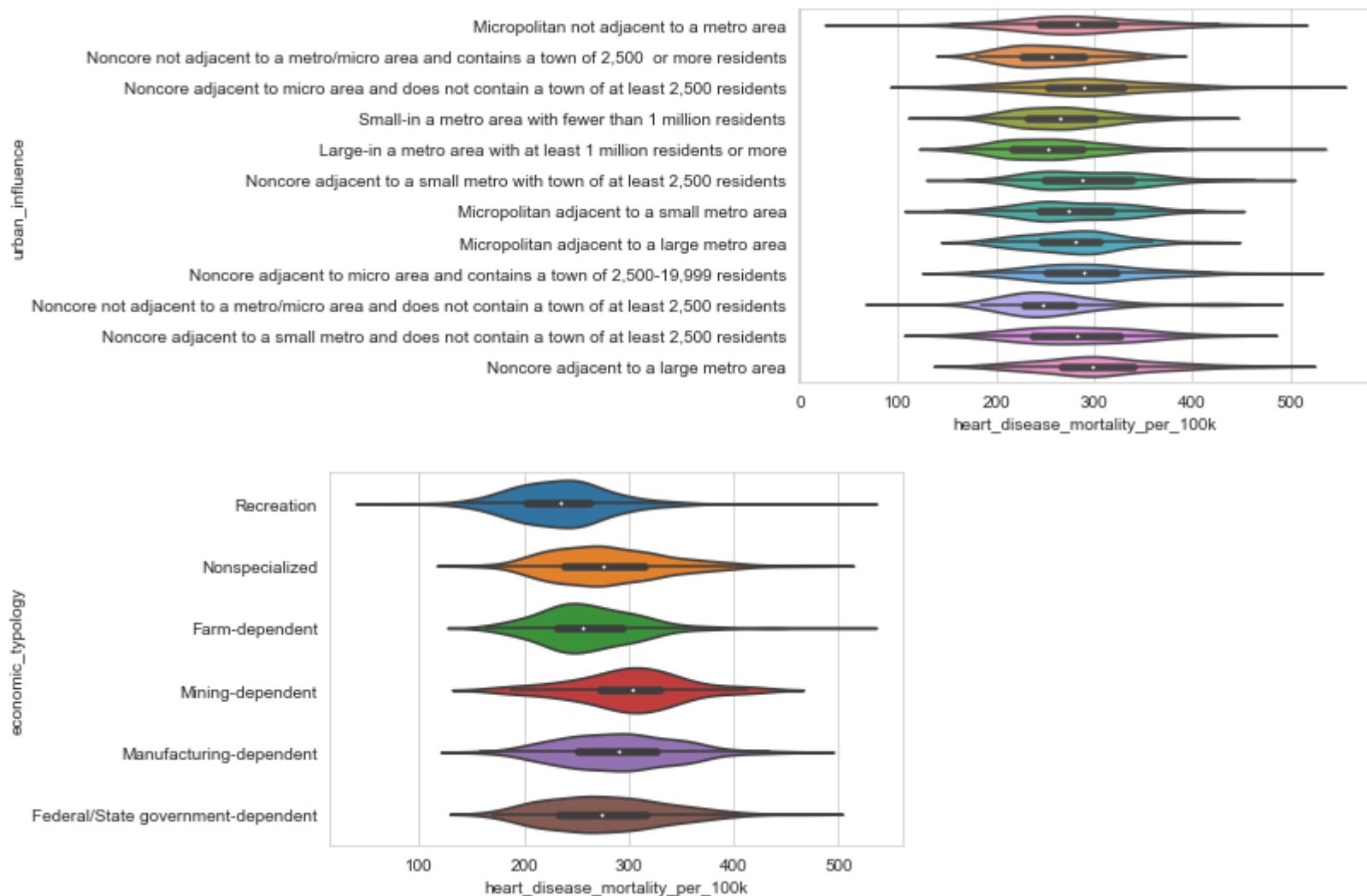
Out[372]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [373]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'heart_disease_mortality_per_100k'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area



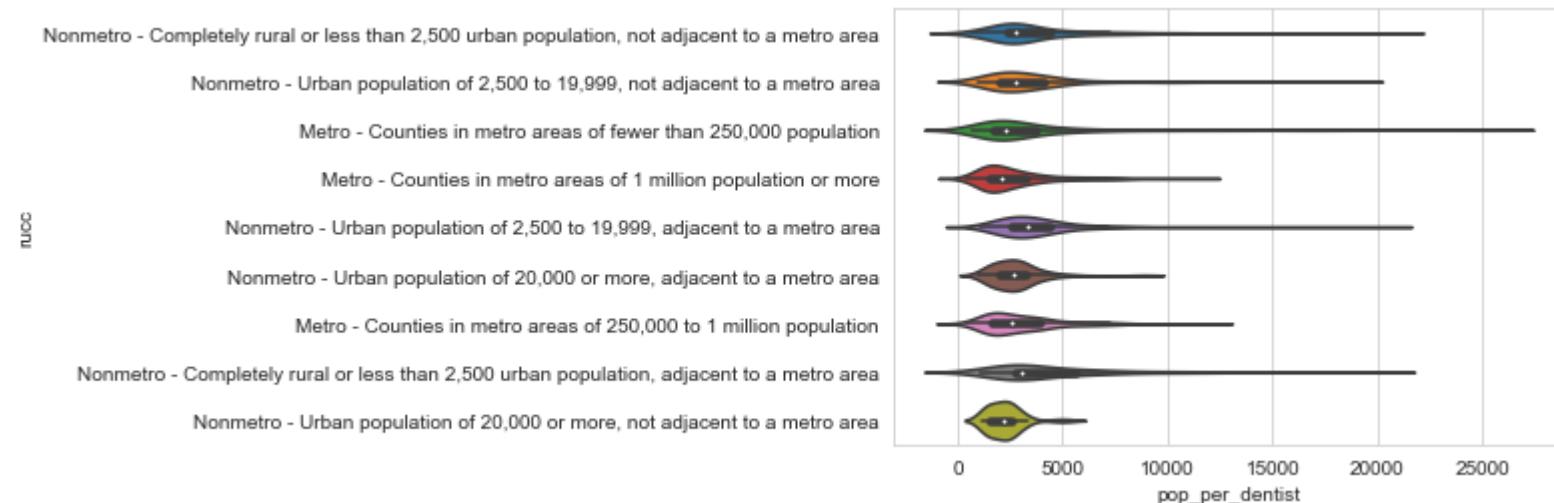


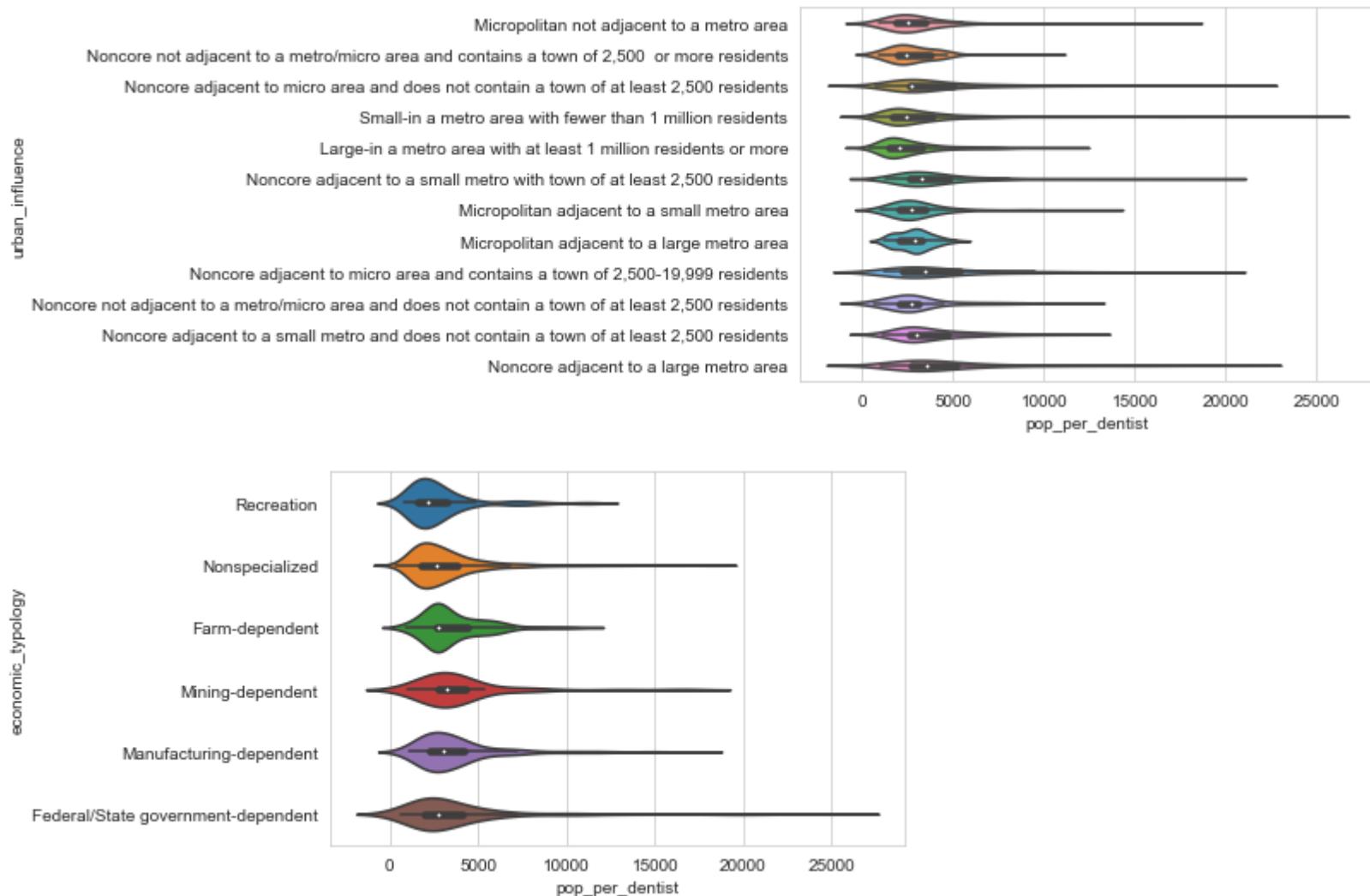
In [374]: num_cols

Out[374]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [375]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pop_per_dentist'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```





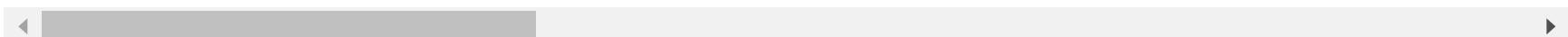
In [376]: `Farmers = train_values_labels[train_values_labels['economic_typology']=='Farm-dependent']
Farmers`

Out[376]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	4	4	183	38	3681.0	365.0	21.985	2.0	19.673	0.923886
	7	7	794	16	27974.0	3873.0	38.012	49.0	29.966	0.537084
	22	22	899	33	10111.0	920.0	21.095	3.0	28.268	0.908087
	27	27	974	28	4203.0	504.0	27.531	0.0	24.138	0.855246
	32	32	435	28	1043.0	125.0	25.425	-1.0	17.492	0.971436

	1457	1457	1487	38	6352.0	665.0	21.695	3.0	25.864	0.960195
	1495	1495	1398	7	2989.0	274.0	21.083	0.0	19.503	0.383372
	1523	1523	1262	38	1954.0	202.0	24.789	0.0	18.753	0.888687
	1547	1547	1368	21	5854.0	628.0	27.376	9.0	27.979	0.710822
	1551	1551	1501	29	4239.0	306.0	20.508	-1.0	27.847	0.944484

172 rows × 85 columns



In [377]: `Farmers['evictions'].describe()`

Out[377]:

count	172.000000
mean	4.052326
std	8.215849
min	-1.000000
25%	0.000000
50%	1.000000
75%	4.250000
max	55.000000
Name:	evictions, dtype: float64

In []:

```
In [378]: sixteen = Farmers[Farmers['evictions']>30]
sixteen
```

Out[378]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af
	7	7	794	16	27974.0	3873.0	38.012	49.0	29.966	0.537084
	1018	1018	614	7	16739.0	1495.0	25.238	34.0	27.527	0.642629
	1408	1408	524	20	20192.0	2123.0	27.348	55.0	25.674	0.955261

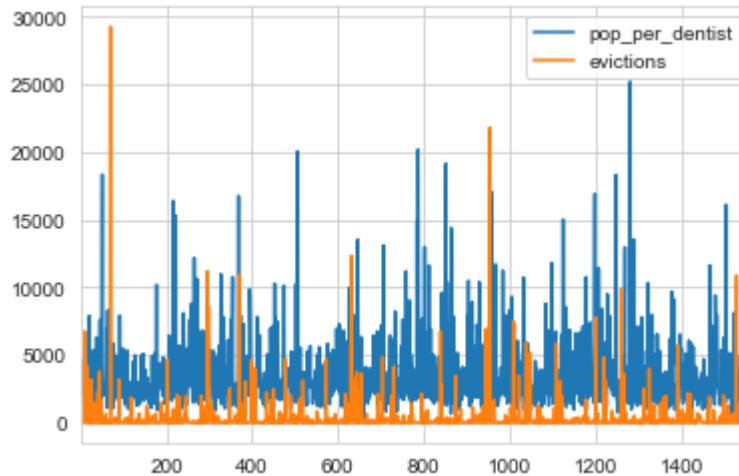
3 rows × 85 columns

```
In [379]: sixteen['pop_per_dentist']
```

```
Out[379]: 7      1410.0
1018    3419.0
1408    5099.0
Name: pop_per_dentist, dtype: float64
```

```
In [380]: train_values_labels[['pop_per_dentist','evictions']].plot()
```

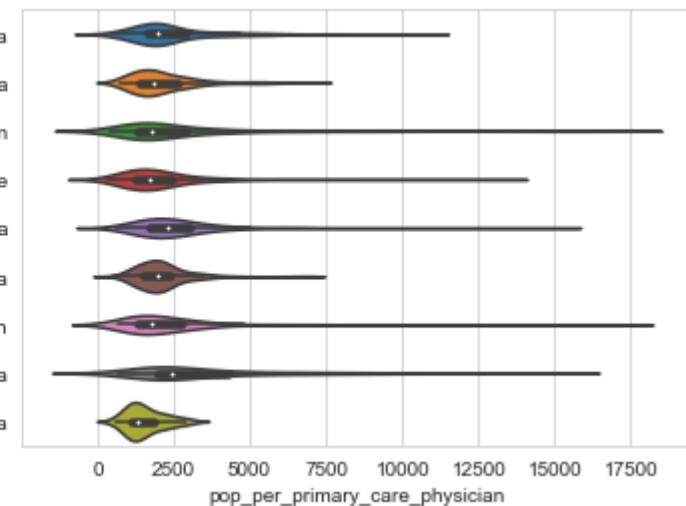
```
Out[380]: <matplotlib.axes._subplots.AxesSubplot at 0x243f75207b8>
```

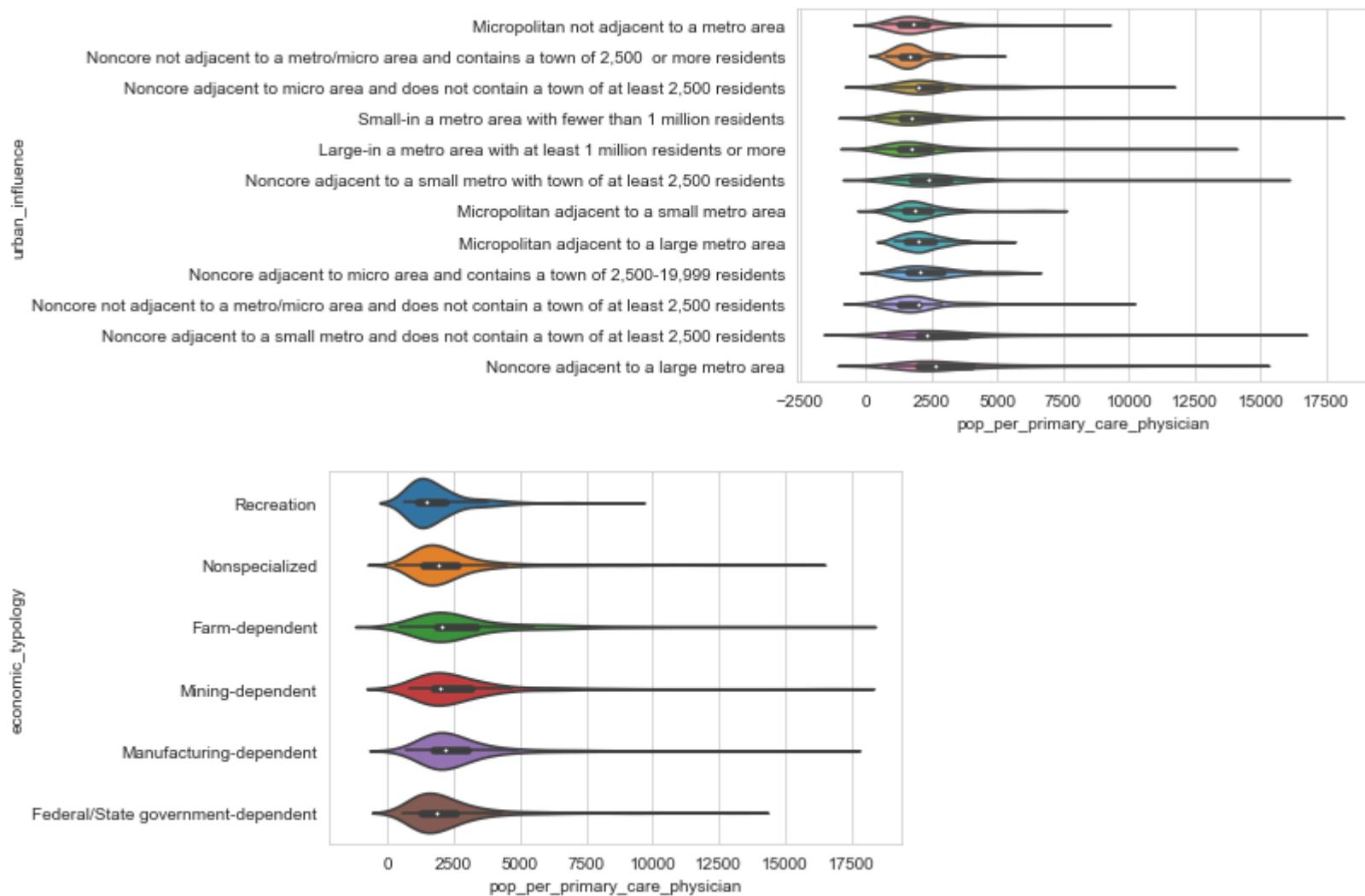


```
In [381]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pop_per_primary_care_physician'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





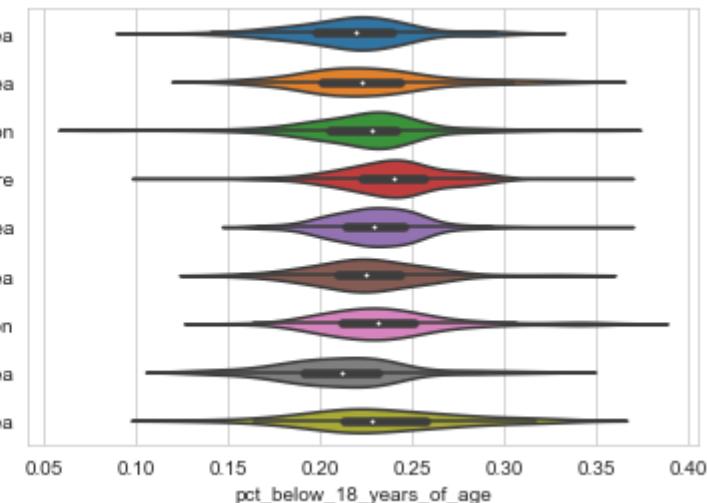
In [382]: num_cols

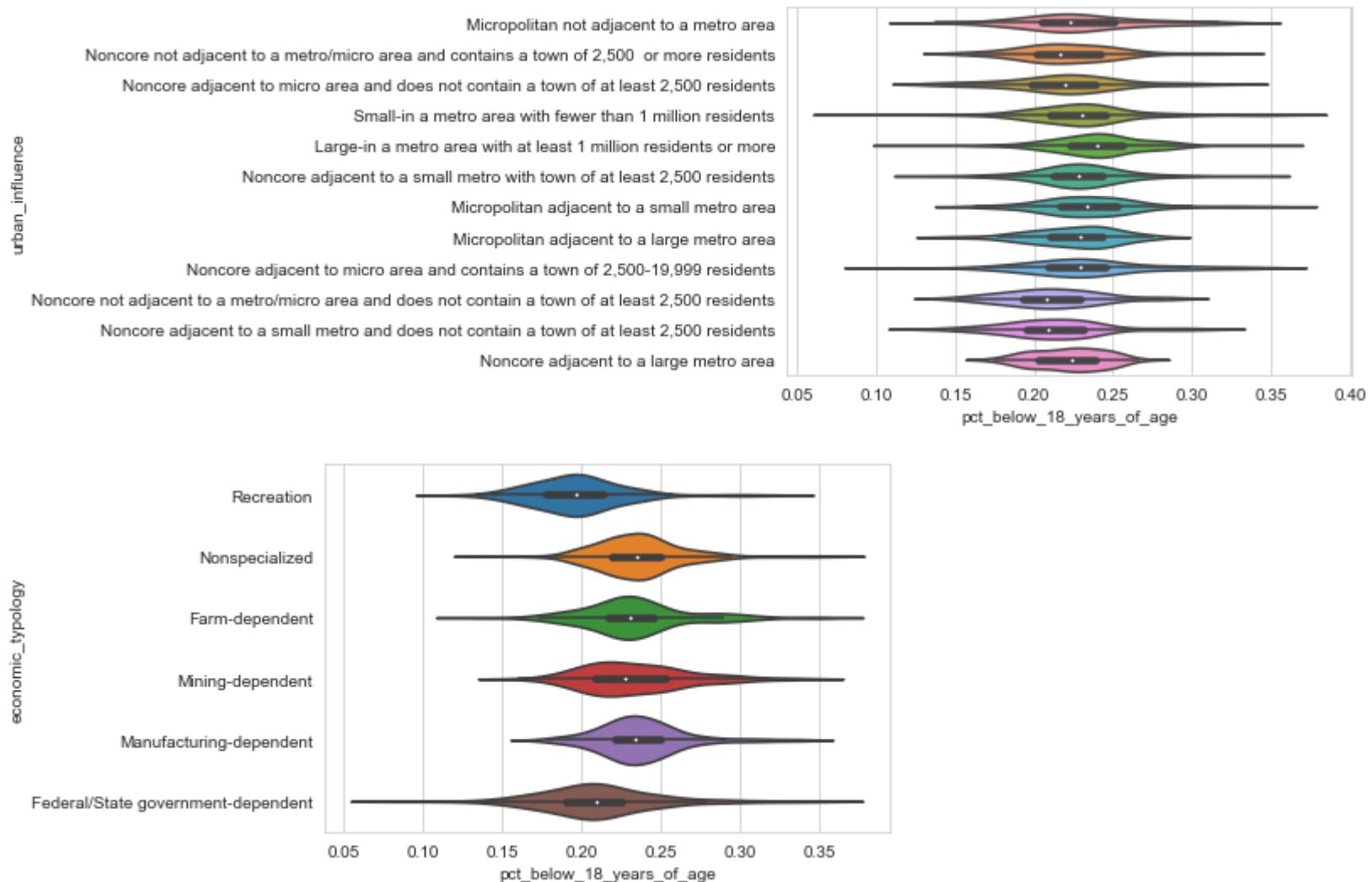
Out[382]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [383]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_below_18_years_of_age'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area



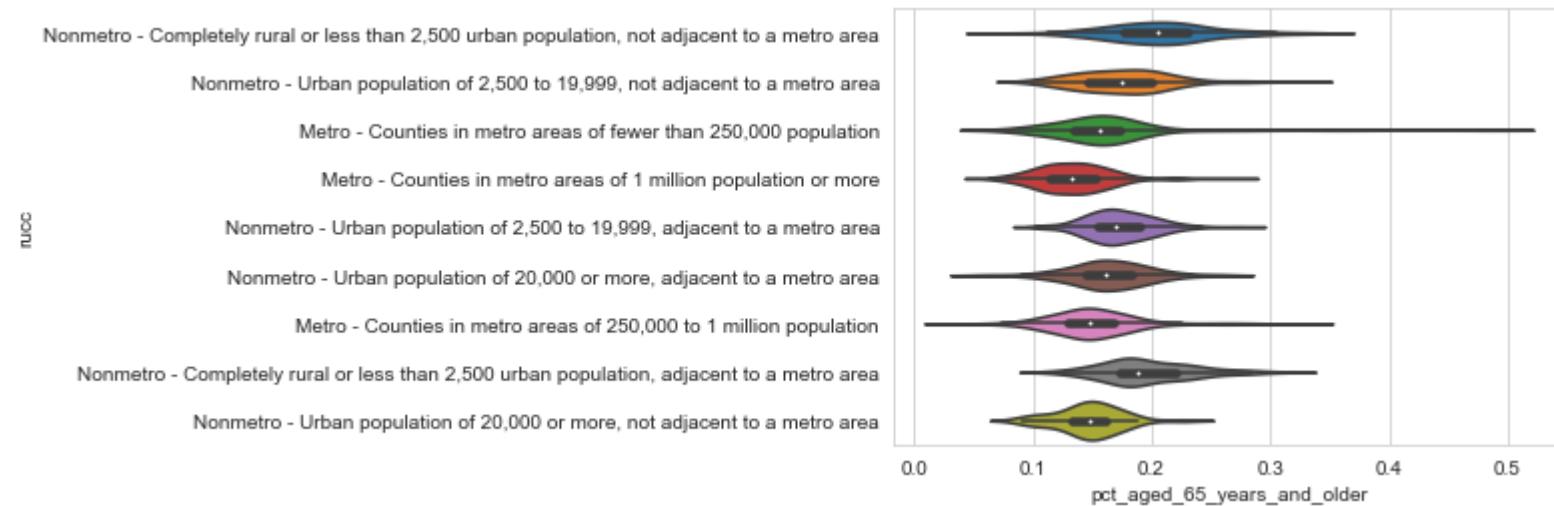


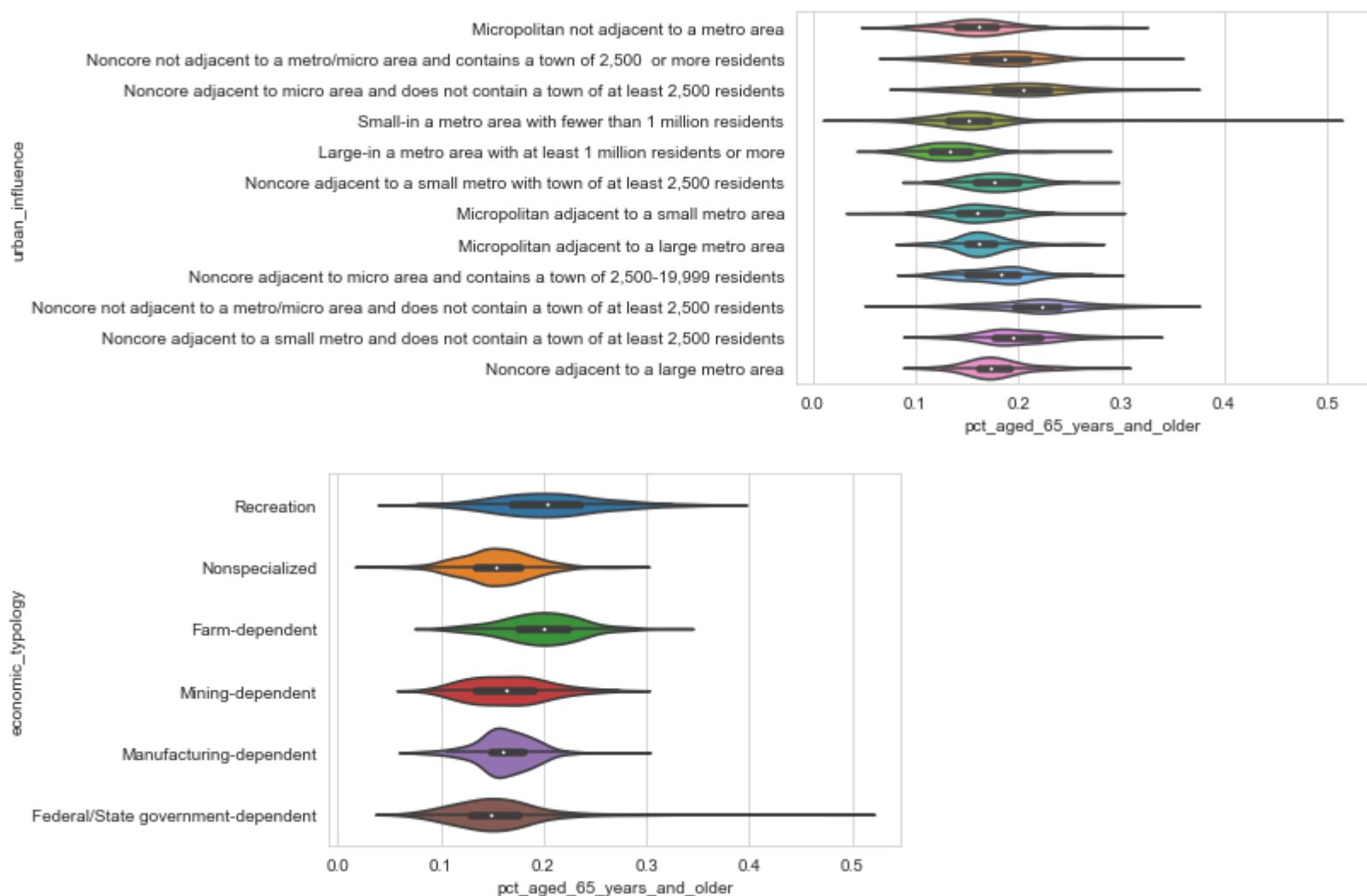
In [384]: num_cols

Out[384]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [385]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_aged_65_years_and_older'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```



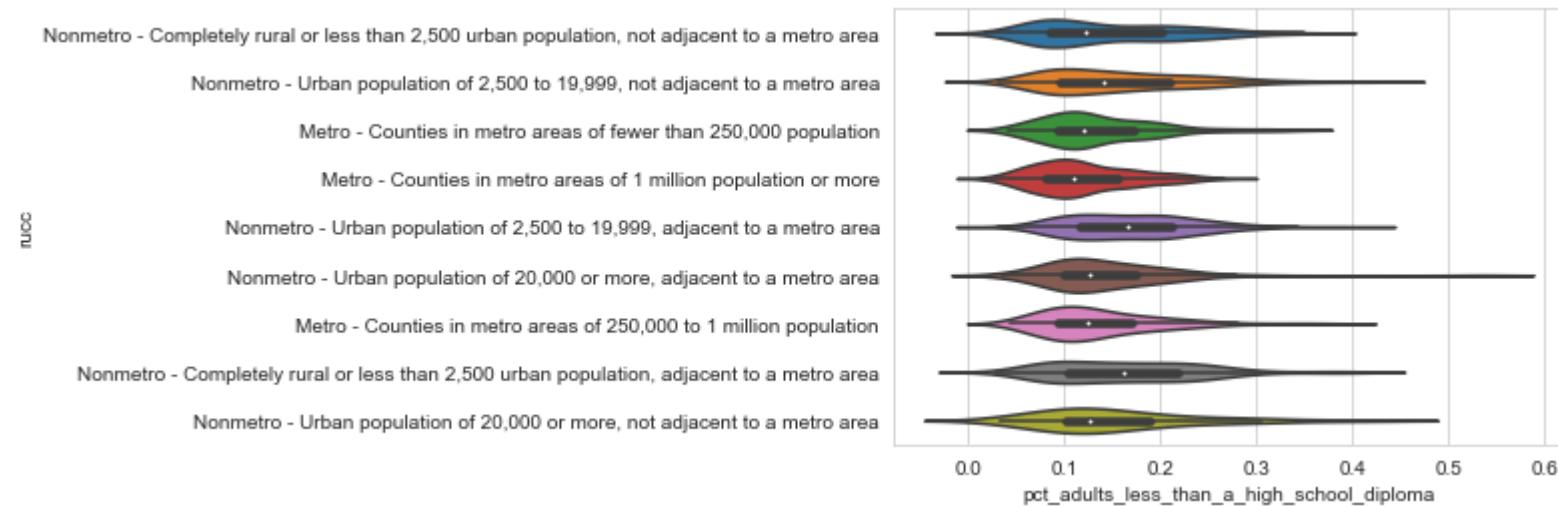


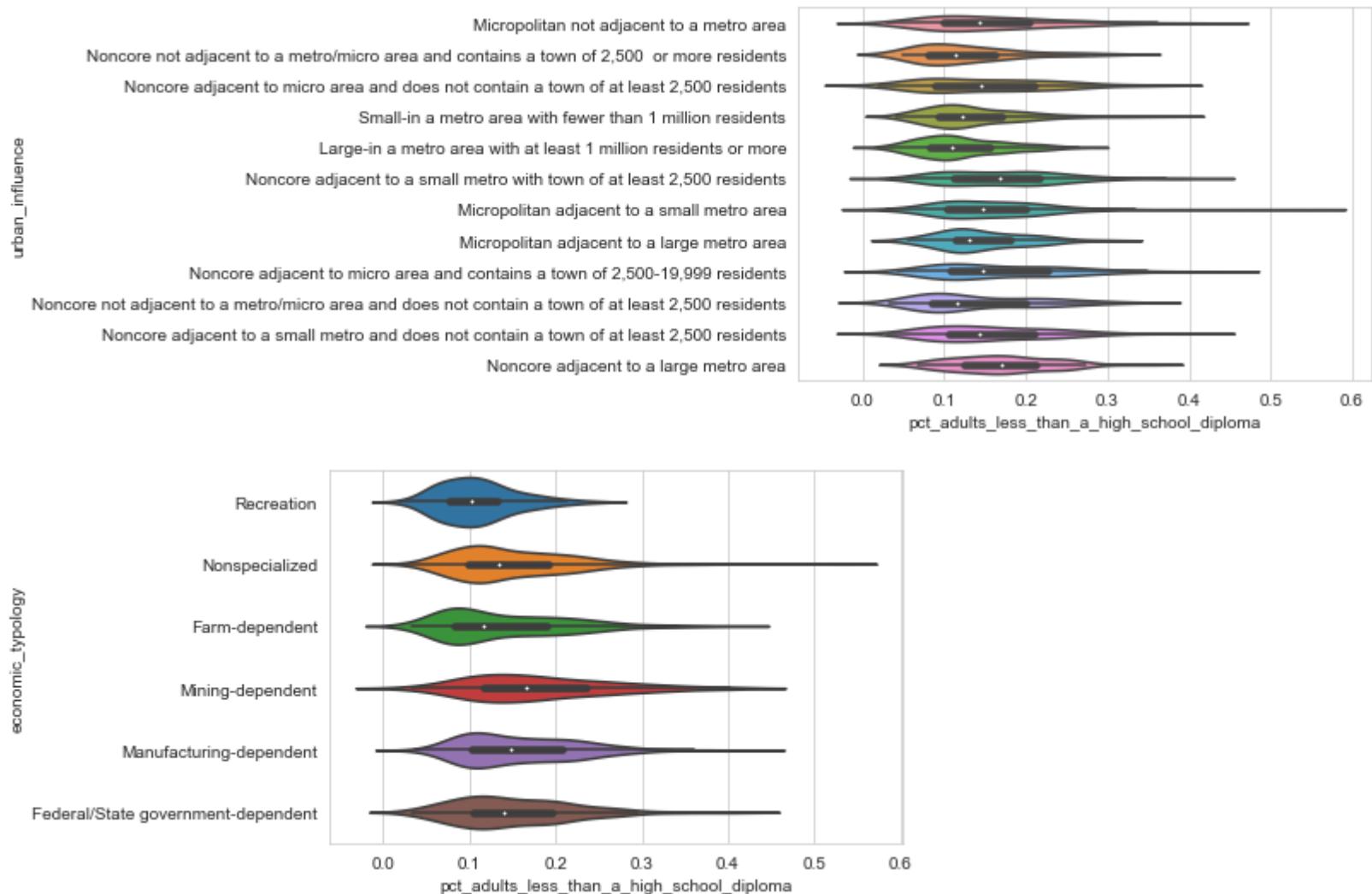
In [386]: num_cols

Out[386]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [387]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adults_less_than_a_high_school_diploma'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

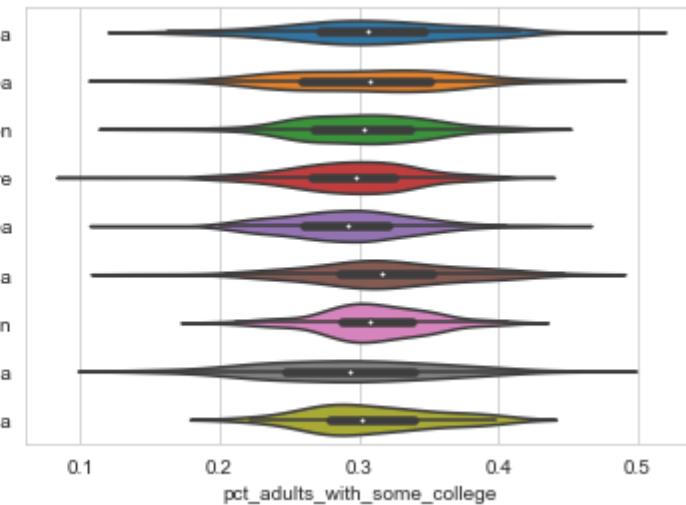


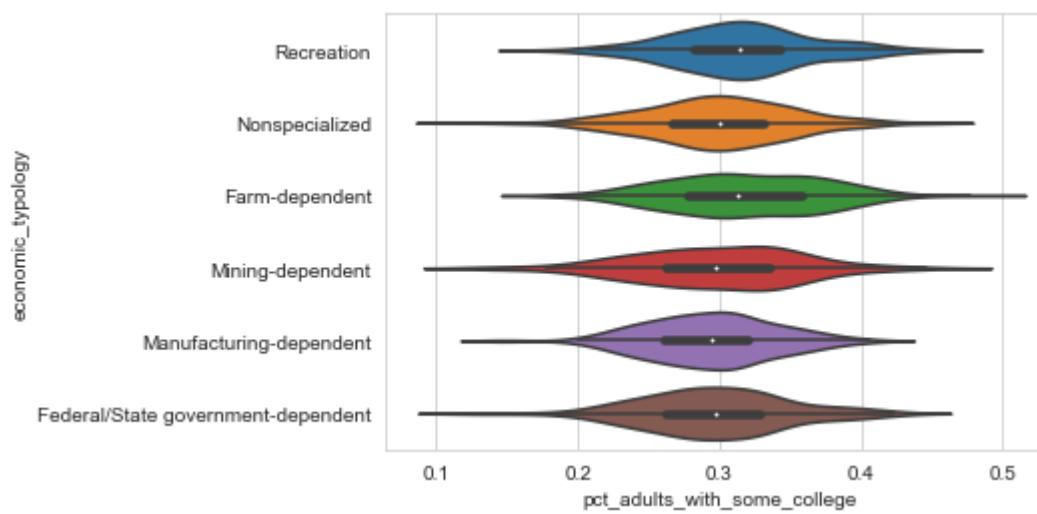
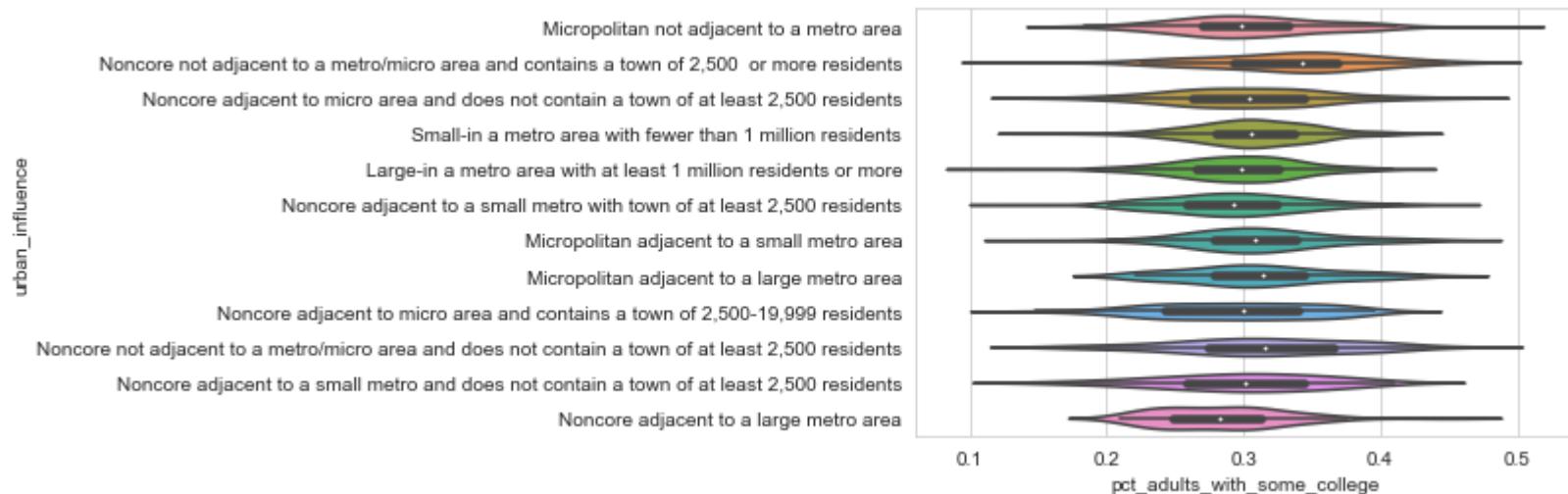


```
In [388]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adults_with_some_college'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

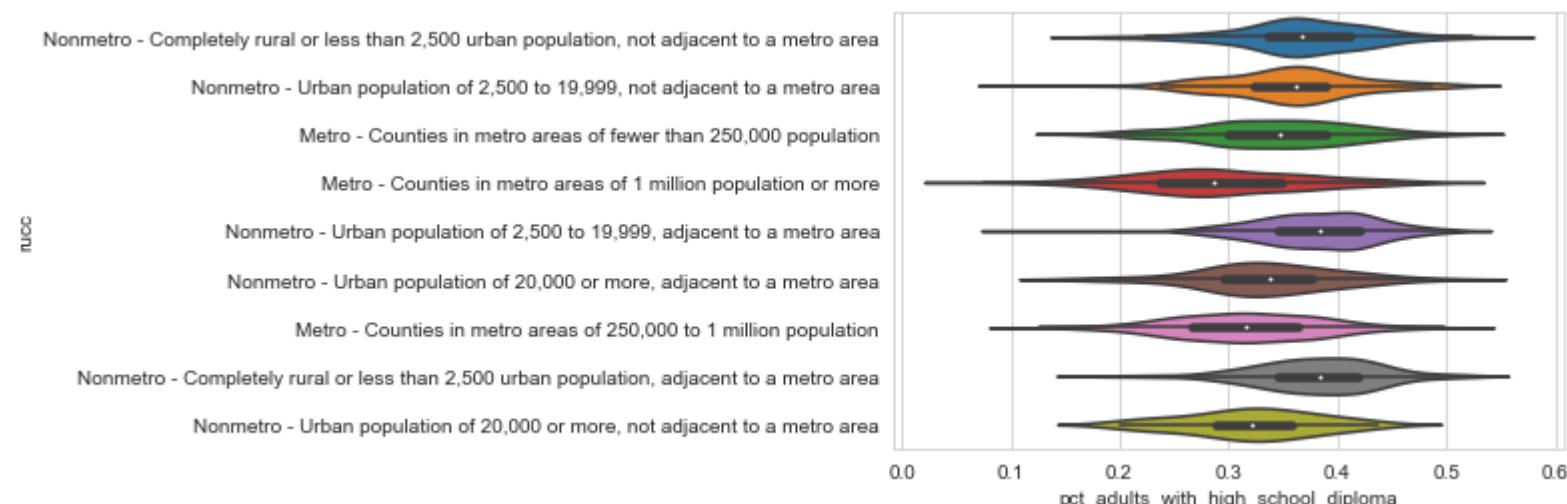
Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

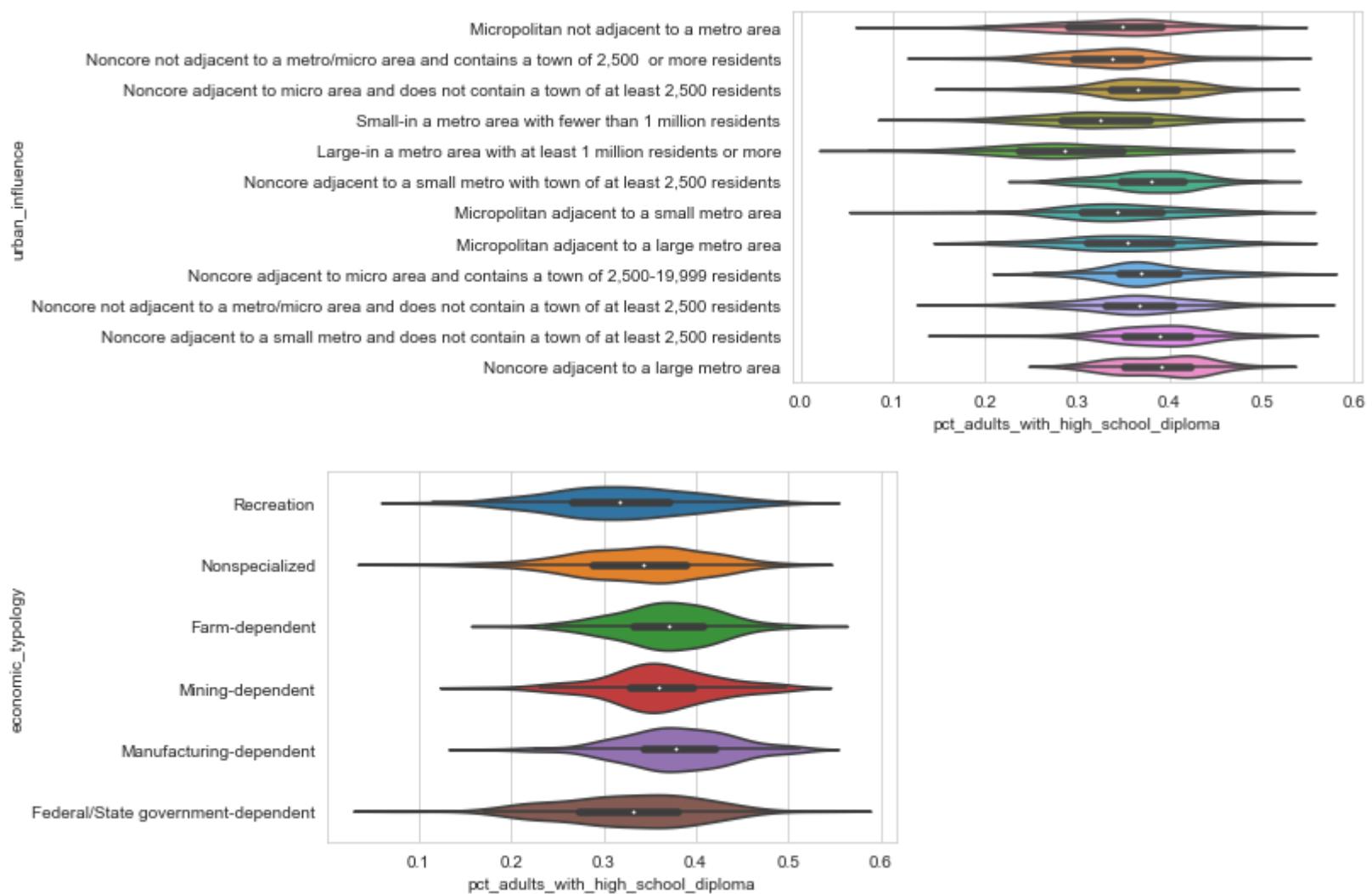





```
In [389]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adults_with_high_school_diploma'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

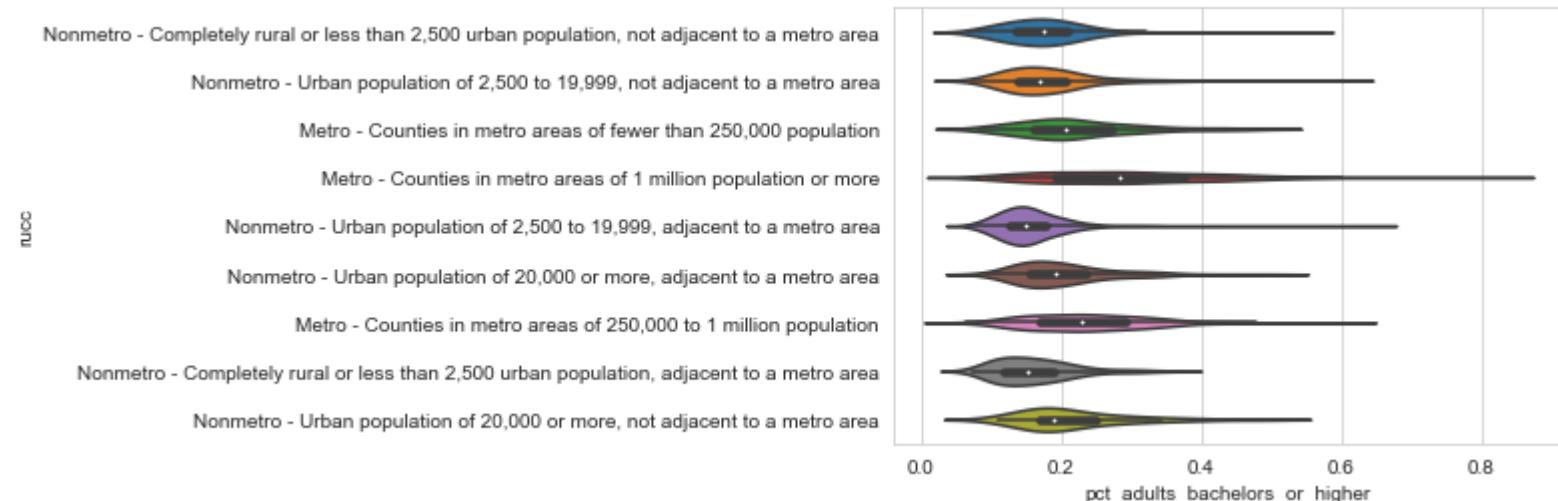
plot_violin(train_values_labels, cat_cols)
```

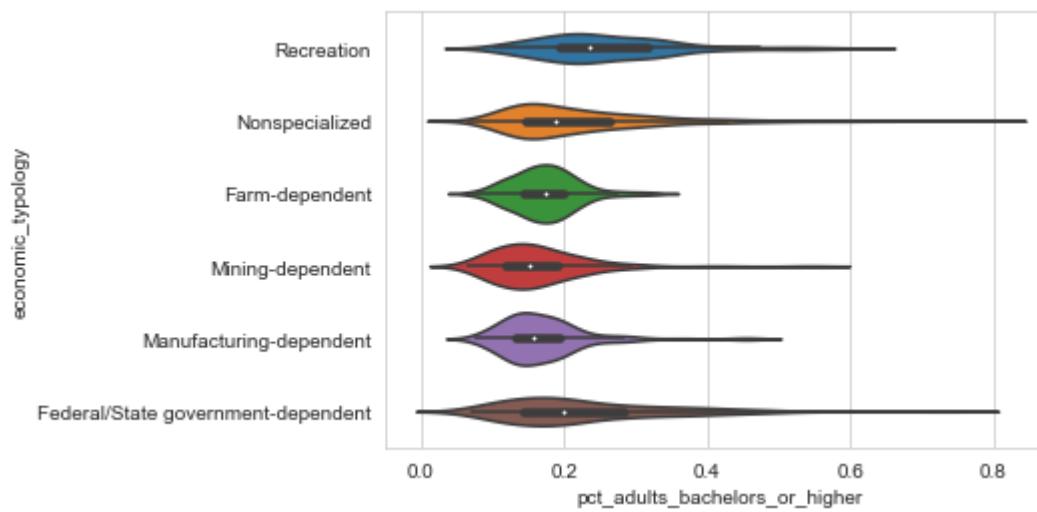
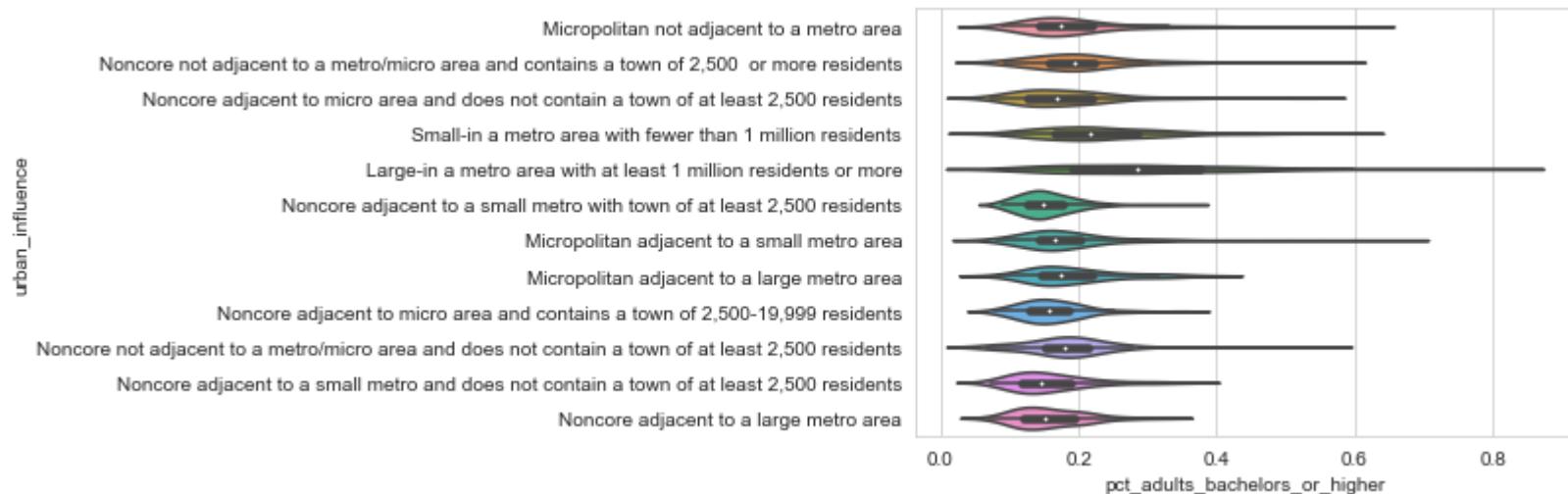




```
In [390]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'pct_adults_bachelors_or_higher'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```





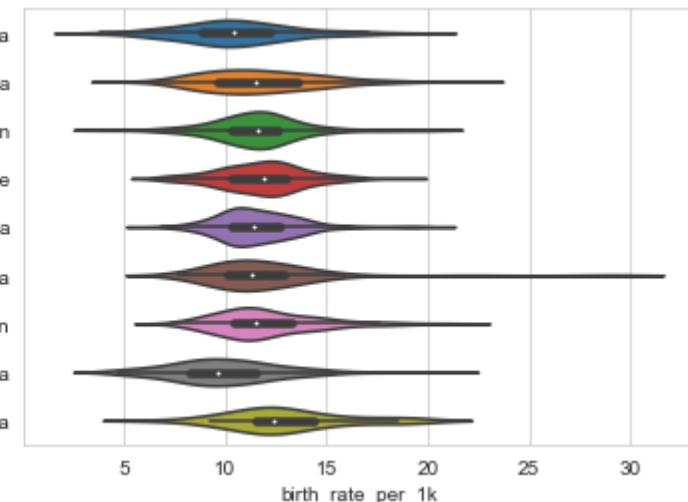
In [391]: num_cols

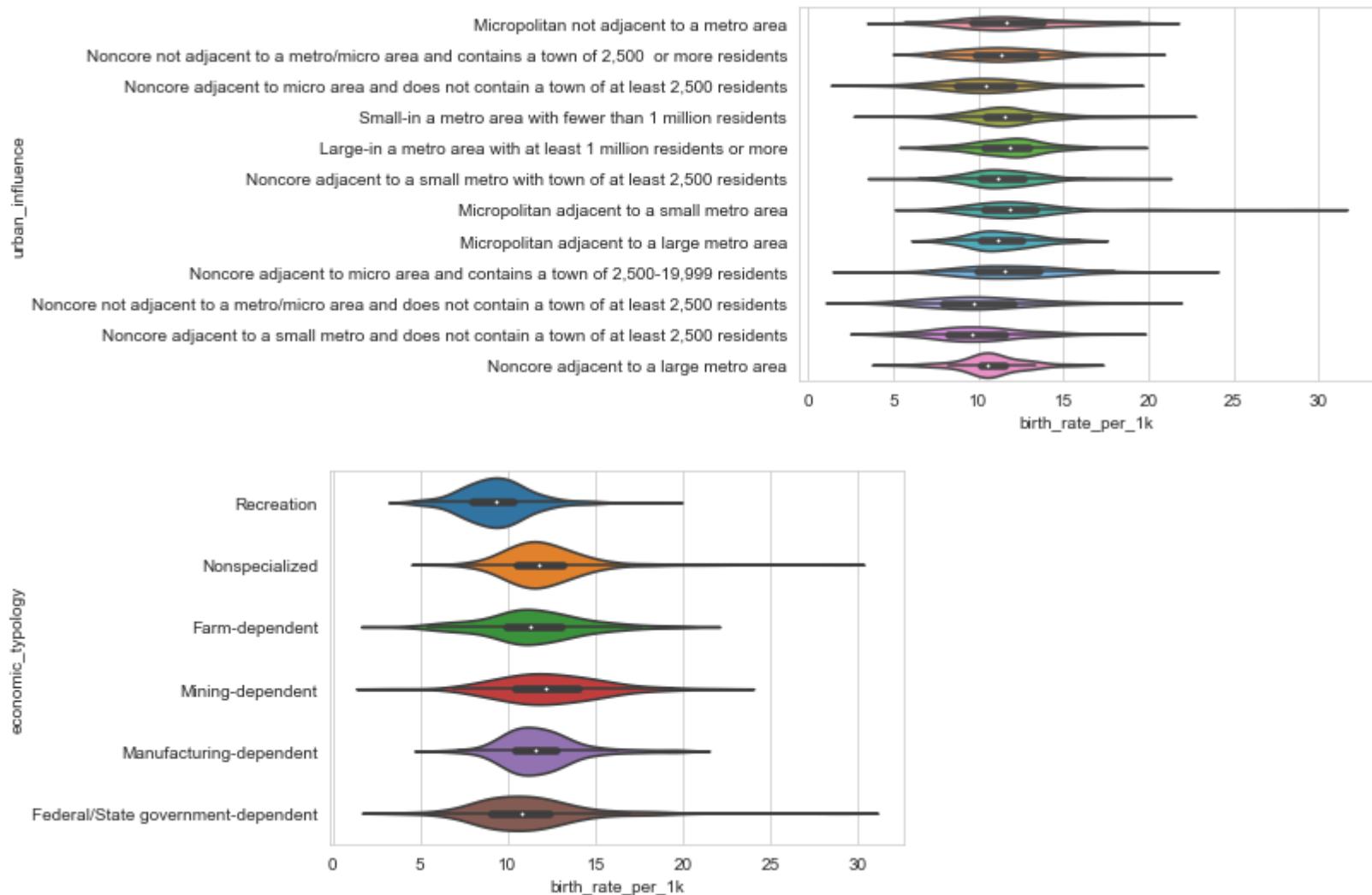
Out[391]: ['population',
 'renter_occupied_households',
 'pct_renter_occupied',
 'evictions',
 'rent_burden',
 'poverty_rate',
 'pct_civilian_labor',
 'pct_unemployment',
 'pct_uninsured_adults',
 'pct_uninsured_children',
 'pct_adult_obesity',
 'pct_adult_smoking',
 'pct_diabetes',
 'pct_low_birthweight',
 'pct_excessive_drinking',
 'pct_physical_inactivity',
 'air_pollution_particulate_matter_value',
 'homicides_per_100k',
 'motor_vehicle_crash_deaths_per_100k',
 'heart_disease_mortality_per_100k',
 'pop_per_dentist',
 'pop_per_primary_care_physician',
 'pct_below_18_years_of_age',
 'pct_aged_65_years_and_older',
 'pct_adults_less_than_a_high_school_diploma',
 'pct_adults_with_high_school_diploma',
 'pct_adults_with_some_college',
 'pct_adults_bachelors_or_higher',
 'birth_rate_per_1k',
 'death_rate_per_1k',
 'gross_rent',
 'pct_white',
 'pct_af_am',
 'pct_hispanic',
 'pct_am_ind',
 'pct_asian',
 'pct_nh_pi',
 'pct_multiple',
 'pct_other']

```
In [392]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'birth_rate_per_1k'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area

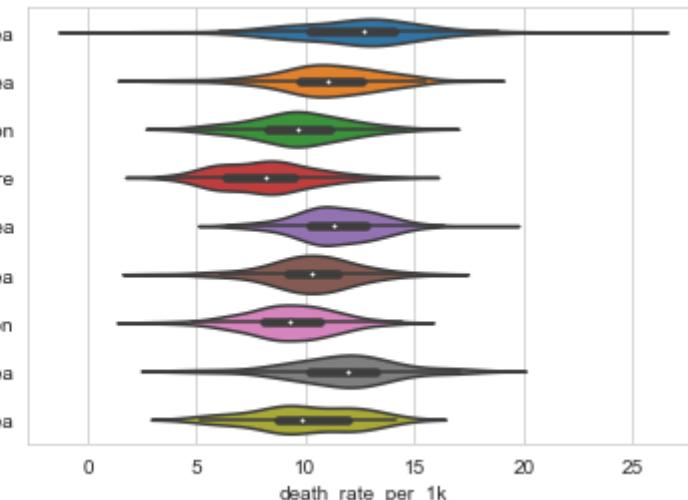


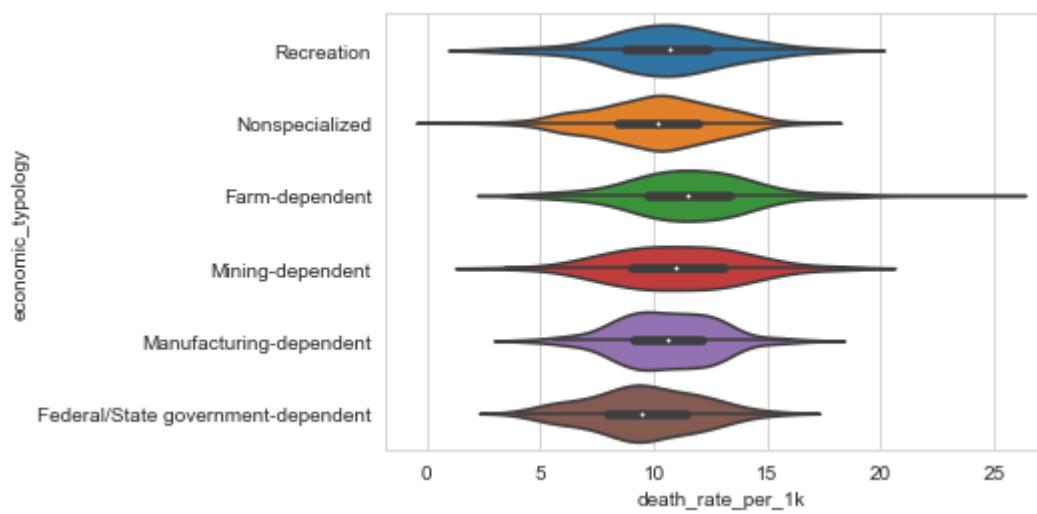
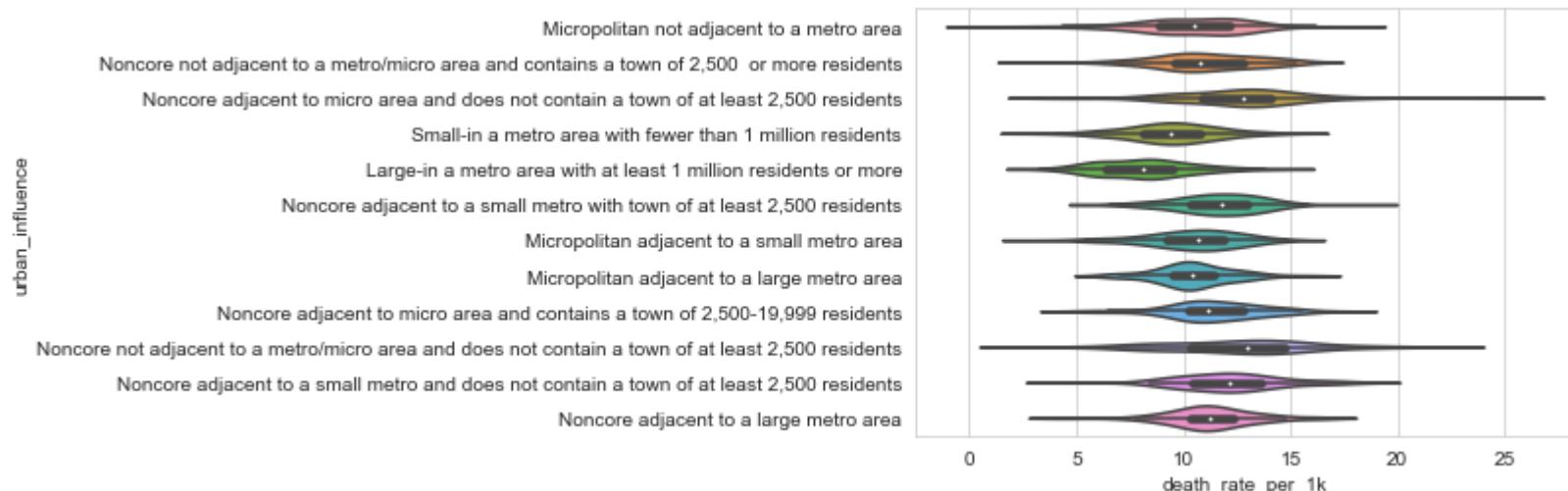


```
In [393]: import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_violin(train_values_labels, cols, col_x = 'death_rate_per_1k'):
    for col in cols:
        sns.set_style("whitegrid")
        sns.violinplot(col_x, col, data=train_values_labels)
        plt.xlabel(col_x) # Set text for the x axis
        plt.ylabel(col)# Set text for y axis
        plt.show()

plot_violin(train_values_labels, cat_cols)
```

Nonmetro - Completely rural or less than 2,500 urban population, not adjacent to a metro area
Nonmetro - Urban population of 2,500 to 19,999, not adjacent to a metro area
Metro - Counties in metro areas of fewer than 250,000 population
Metro - Counties in metro areas of 1 million population or more
Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, adjacent to a metro area
Metro - Counties in metro areas of 250,000 to 1 million population
Nonmetro - Completely rural or less than 2,500 urban population, adjacent to a metro area
Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area





```
In [394]: labels = np.array(np.log(train_values_labels['gross_rent']))
```

```
In [395]: cat_cols
```

```
Out[395]: ['rucc', 'urban_influence', 'economic_t typology']
```

```
In [396]: def encode_string(cat_features):
    ## First encode the strings to numeric categories
    enc = preprocessing.LabelEncoder()
    enc.fit(cat_features)
    enc_cat_features = enc.transform(cat_features)
    ## Now, apply one hot encoding
    ohe = preprocessing.OneHotEncoder()
    encoded = ohe.fit(enc_cat_features.reshape(-1,1))
    return encoded.transform(enc_cat_features.reshape(-1,1)).toarray()

categorical_columns = ['urban_influence', 'economic_typology']

TestFeatures = encode_string(test_values['rucc'])
for col in categorical_columns:
    temp = encode_string(test_values[col])
    TestFeatures = np.concatenate([TestFeatures, temp], axis = 1)

print(TestFeatures.shape)
print(TestFeatures[:2, :])
```

```
(1576, 27)
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0.]]
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.

If you want the future behaviour and silence this warning, you can specify "categories='auto'".

In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

```
warnings.warn(msg, FutureWarning)
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.

If you want the future behaviour and silence this warning, you can specify "categories='auto'".

In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

```
warnings.warn(msg, FutureWarning)
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.

If you want the future behaviour and silence this warning, you can specify "categories='auto'". In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

```
warnings.warn(msg, FutureWarning)
```

In [397]: cat_cols

Out[397]: ['rucc', 'urban_influence', 'economic_t typology']

```
In [398]: def encode_string(cat_features):
    ## First encode the strings to numeric categories
    enc = preprocessing.LabelEncoder()
    enc.fit(cat_features)
    enc_cat_features = enc.transform(cat_features)
    ## Now, apply one hot encoding
    ohe = preprocessing.OneHotEncoder()
    encoded = ohe.fit(enc_cat_features.reshape(-1,1))
    return encoded.transform(enc_cat_features.reshape(-1,1)).toarray()

categorical_columns = ['urban_influence','economic_typology']

Features = encode_string(train_values_labels['rucc'])
for col in categorical_columns:
    temp = encode_string(train_values_labels[col])
    Features = np.concatenate([Features, temp], axis = 1)

print(Features.shape)
print(Features[:2, :])
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.

If you want the future behaviour and silence this warning, you can specify "categories='auto'".

In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

```
warnings.warn(msg, FutureWarning)
```

```
(1440, 27)
[[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1.]
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1.]]
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.

If you want the future behaviour and silence this warning, you can specify "categories='auto'".

In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

```
warnings.warn(msg, FutureWarning)
```

C:\Users\Jorge\Anaconda3\lib\site-packages\sklearn\preprocessing_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range

[0, max(values)], while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.
warnings.warn(msg, FutureWarning)

In [399]: updated_num_cols

Out[399]: ['populationlog',
'homicides_per_100klog',
'motor_vehicle_crash_deaths_per_100klog',
'evictionslog',
'renter_occupied_householdslog',
'pct_renter_occupied',
'rent_burden',
'poverty_ratelog',
'pct_civilian_laborlog',
'pct_unemploymentlog',
'pct_uninsured_adults',
'pct_uninsured_children',
'pct_adult_obesity',
'pct_adult_smokinglog',
'pct_diabeteslog',
'pct_low_birthweightlog',
'pct_excessive_drinkinglog',
'pct_physical_inactivitylog',
'air_pollution_particulate_matter_value',
'heart_disease_mortality_per_100klog',
'pop_per_dentistlog',
'pop_per_primary_care_physicianlog',
'pct_below_18_years_of_agelog',
'pct_aged_65_years_and_olderlog',
'pct_adults_less_than_a_high_school_diplomalog',
'pct_adults_with_high_school_diploma',
'pct_adults_with_some_college',
'pct_adults_bachelors_or_higher',
'birth_rate_per_1k',
'death_rate_per_1k']

```
In [400]: TestFeatures = np.concatenate([TestFeatures, np.array(test_values[['populationlog','homicides_per_100klog',
'motor_vehicle_crash_deaths_per_100klog', 'evictionslog',
'renter_occupied_householdslog',
'pct_renter_occupied',
'rent_burden',
'poverty_ratelog',
'pct_civilian_laborlog',
'pct_unemploymentlog',
'pct_uninsured_adults',
'pct_uninsured_children',
'pct_adult_obesity',
'pct_adult_smokinglog',
'pct_diabeteslog',
'pct_low_birthweightlog',
'pct_physical_inactivitylog',
'pct_excessive_drinkinglog',
'air_pollution_particulate_matter_value',
'heart_disease_mortality_per_100klog',
'pop_per_dentistlog',
'pop_per_primary_care_physicianlog',
'pct_below_18_years_of_agelog',
'pct_aged_65_years_and_olderlog',
'pct_adults_less_than_a_high_school_diplomalog',
'pct_adults_with_high_school_diploma',
'pct_adults_with_some_college',
'pct_adults_bachelors_or_higher',
'birth_rate_per_1k',
'death_rate_per_1k',
]]]), axis = 1)
print(TestFeatures.shape)
print(TestFeatures[:2, :])
```

```
(1576, 57)
[[ 0.0000000e+00  0.0000000e+00  1.0000000e+00  0.0000000e+00
  0.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
  0.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
  0.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
  0.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
  1.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
  0.0000000e+00  1.0000000e+00  0.0000000e+00  1.08750616e+01
```

```
3.50585765e+00 7.61076863e+00 3.95124372e+00 8.59470963e+00
2.68400000e+01 2.79600000e+01 3.38659121e+00 -8.30113036e-01
-2.59026717e+00 2.18000000e-01 5.70000000e-02 2.84000000e-01
3.22910335e+00 -2.17155683e+00 -2.52572864e+00 -1.10262031e+00
3.22079398e+00 1.31940276e+01 5.68017261e+00 8.49290050e+00
7.90470391e+00 -1.53711725e+00 -1.73160555e+00 -1.74496731e+00
3.95209581e-01 2.74451098e-01 1.55688623e-01 9.82829062e+00
1.06282678e+01]
[ 1.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.22656944e+01
 4.04182244e+00 7.60630285e+00 8.70896991e+00 1.08874743e+01
 5.75340000e+01 3.30720000e+01 4.37835586e+00 -6.46263595e-01
-2.78062089e+00 2.25000000e-01 5.80000000e-02 2.82000000e-01
3.22632799e+00 -2.33304430e+00 -2.12026354e+00 -1.37832619e+00
3.22541440e+00 1.24194326e+01 5.73009978e+00 6.74523635e+00
6.91671502e+00 -1.65025991e+00 -2.18036746e+00 -1.78184859e+00
2.30079681e-01 2.41035857e-01 3.60557769e-01 1.32855463e+01
8.81829296e+00]]
```

In [401]:

```
scaler = preprocessing.StandardScaler().fit(TestFeatures[:,27:])
TestFeatures[:,27:] = scaler.transform(TestFeatures[:,27:])
print(TestFeatures.shape)
TestFeatures[:5,:]
```

(1576, 57)

```
Out[401]: array([[ 0.          ,  0.          ,  0.          ,  1.          ,  0.          ,  0.          ,  0.          ],
   0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
   0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
   0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
   1.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
   1.          ,  0.          ,  0.41335036,  0.3222024 ,  0.07587219 ,
 -0.36352645,  0.32900981, -0.22028465, -0.11902964, -0.02118923 ,
 -0.44063107,  0.68204434,  0.04605988, -0.73581445, -0.55004182 ,
  0.77815552,  0.3699882 , -0.06677921,  0.93857876, -0.79800864 ,
  0.90528602,  0.34477884,  0.91951599,  0.43568858, -0.37411997 ,
  0.34726014,  0.61392532,  0.66171513, -0.5434046 , -0.53981712 ,
 -0.75070766,  0.08507029],  
[ 1.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  1.          ,  0.          ],
  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  1.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  1.37684123,  5.04412896, -0.69827687 ,
  3.5239058 ,  1.83211807,  3.61197381,  1.02625059,  1.21883068 ,
  0.73803587,  0.16290426,  0.1543114 , -0.70994802, -0.59862887 ,
  0.0292093 , -0.37458398,  1.69561517, -0.42332556,  0.54016557 ,
  0.4861897 ,  0.57413142, -2.11042758, -1.21125026, -1.1854456 ,
 -1.42160777,  0.53366453, -1.68857503, -1.19694032,  1.78103137 ,
  0.59916495, -0.56332416],  
[ 0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  1.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  1.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  0.          ,  0.          ,  1.          ,  0.          ,  0.          ],
  0.          ,  0.          ,  0.71153944,  1.9917313 ,  0.36914473 ,
  2.0818594 ,  0.92291287,  1.42204049,  0.79593994,  0.96118253 ,
 -0.04159286,  1.29787305,  0.54092396, -0.11502025,  1.0290374 ,
 -0.6788489 ,  1.21735622,  0.85945162,  0.56792746, -1.35354957 ,
  0.19294689, -0.04780804,  0.064412 ,  0.2736702 ,  0.41467666 ,
 -0.22708661,  0.92853788, -0.26684224, -0.28646713, -0.25731965 ]]
```

```
-0.06994919, -0.09124498],  
[ 0. , 1. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 1. , 0. , 0. , 0. , 0. , 0. ,  
 1. , 0. , 0.99798593, 1.54052105, 0.19941584,  
-0.36352645, 1.16567281, 1.6556417 , 0.5002103 , 1.13102843,  
0.0134379 , -1.03232161, 2.36573524, 1.61803024, 0.78610214,  
0.44673574, -0.23414535, 0.85945162, 0.75672878, 0.37898948,  
-0.91681594, 1.02820372, -0.41582763, -0.86760355, 1.57110153,  
-1.49199647, 1.30166091, -1.01597713, 0.62117584, -0.61439515,  
2.12437056, -0.10801583],  
[ 0. , 1. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 0. , 0. , 0. , 0. , 0. , 0. ,  
 1. , 0. , 0. , 0. , 0. , 0. ,  
 1. , 0. , 1.11821307, 0.83341371, -0.20844086,  
1.72373165, 1.03107955, 0.25927829, 0.53067941, -0.21173609,  
0.04077715, -0.01897905, -0.06219163, -0.03742097, -0.40428066,  
0.07206227, -0.05330995, 0.44517645, -0.54236023, 0.41353477,  
0.54603669, -0.15143069, -0.69862705, 0.3072253 , 0.96101526,  
-1.59999397, -0.59738147, -0.845846 , 1.13680987, 0.53141891,  
0.27834825, -1.41790648]])
```

```
In [402]: #from sklearn.preprocessing import QuantileTransformer  
#transformer = QuantileTransformer().fit(X_train)  
#transformer.transform(X_train)
```

```
In [403]: train_values_labels.columns
```

```
Out[403]: Index(['row_id', 'county_code', 'state', 'population',
       'renter_occupied_households', 'pct_renter_occupied', 'evictions',
       'rent_burden', 'pct_white', 'pct_af_am', 'pct_hispanic', 'pct_am_ind',
       'pct_asian', 'pct_nh_pi', 'pct_multiple', 'pct_other', 'poverty_rate',
       'rucc', 'urban_influence', 'economic_typerology', 'pct_civilian_labor',
       'pct_unemployment', 'pct_uninsured_adults', 'pct_uninsured_children',
       'pct_adult_obesity', 'pct_adult_smoking', 'pct_diabetes',
       'pct_low_birthweight', 'pct_excessive_drinking',
       'pct_physical_inactivity', 'air_pollution_particulate_matter_value',
       'homicides_per_100k', 'motor_vehicle_crash_deaths_per_100k',
       'heart_disease_mortality_per_100k', 'pop_per_dentist',
       'pop_per_primary_care_physician', 'pct_female',
       'pct_below_18_years_of_age', 'pct_aged_65_years_and_older',
       'pct_adults_less_than_a_high_school_diploma',
       'pct_adults_with_high_school_diploma', 'pct_adults_with_some_college',
       'pct_adults_bachelors_or_higher', 'birth_rate_per_1k',
       'death_rate_per_1k', 'gross_rent', 'evictionslog',
       'homicides_per_100klog', 'populationlog',
       'renter_occupied_householdslog', 'pct_renter_occupiedlog',
       'rent_burdenlog', 'poverty_ratelog',
       'motor_vehicle_crash_deaths_per_100klog', 'pop_per_dentistlog',
       'pop_per_primary_care_physicianlog',
       'pct_adults_less_than_a_high_school_diplomalog',
       'pct_adults_bachelors_or_higherlog', 'gross_rentlog',
       'air_pollution_particulate_matter_valuelog',
       'pct_uninsured_childrenlog', 'pct_whitelog', 'pct_af_amlog',
       'pct_hispaniclog', 'pct_am_indlog', 'pct_asianlog', 'pct_nh_pilog',
       'pct_multiplelog', 'pct_otherlog', 'pct_aged_65_years_and_olderlog',
       'heart_disease_mortality_per_100klog', 'death_rate_per_1klog',
       'pct_adult_obesitylog', 'pct_adult_smokinglog', 'pct_diabeteslog',
       'pct_low_birthweightlog', 'pct_physical_inactivitylog',
       'pct_excessive_drinkinglog', 'pct_below_18_years_of_agelog',
       'pct_adults_with_high_school_diplomalog',
       'pct_adults_with_some_collegelog', 'birth_rate_per_1klog',
       'pct_civilian_laborlog', 'pct_unemploymentlog',
       'pct_uninsured_adultslog'],
      dtype='object')
```

```
In [404]: Features = np.concatenate([Features, np.array(train_values_labels[['populationlog','homicides_per_100klog',
'motor_vehicle_crash_deaths_per_100klog', 'evictionslog',
'renter_occupied_householdslog',
'pct_renter_occupied',
'rent_burden',
'poverty_ratelog',
'pct_civilian_laborlog',
'pct_unemploymentlog',
'pct_uninsured_adults',
'pct_uninsured_children',
'pct_adult_obesity',
'pct_adult_smokinglog',
'pct_diabeteslog',
'pct_low_birthweightlog',
'pct_physical_inactivitylog',
'pct_excessive_drinkinglog',
'air_pollution_particulate_matter_value',
'heart_disease_mortality_per_100klog',
'pop_per_dentistlog',
'pop_per_primary_care_physicianlog',
'pct_below_18_years_of_agelog',
'pct_aged_65_years_and_olderlog',
'pct_adults_less_than_a_high_school_diplomalog',
'pct_adults_with_high_school_diploma',
'pct_adults_with_some_college',
'pct_adults_bachelors_or_higher',
'birth_rate_per_1k',
'death_rate_per_1k',
]]]), axis = 1)
print(Features.shape)
print(Features[:2, :])
```

```
(1440, 57)
[[ 0.        0.        0.        0.        1.        0.
   0.        0.        0.        0.        0.        0.
   1.        0.        0.        0.        0.        0.
   0.        0.        0.        0.        0.        0.
   0.        0.        1.        9.23249318  3.40119738  7.60999103
   3.33220451  7.06133437  28.346     26.694     2.77740886 -0.60696948
  -3.21887582  0.324      0.201      0.253      3.22294752 -2.71810054
  -2.65926004 -1.91054301  3.22684399  9.90409859  5.19849703  7.60539236
```

```
7.43248381 3.23040906 -2.56394986 -2.31163493 0.2022022 0.3023023
0.3963964 14.48289269 3.01321928]
[ 0.          0.          0.          0.          0.          0.
  1.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  1.          0.          0.          0.          0.          0.
  0.          0.          1.          10.20444363 3.40119738 7.61082309
  3.52636052 7.98173329 21.641      31.028      3.12873776 -0.67727383
-2.76462055 0.129       0.032       0.296       3.22743906 -2.44184716
-2.65926004 -1.35479569 3.22632799 11.01150213 5.45532112 8.4446225
  7.20785987 3.2271614   -1.65025991 -1.9824996  0.38722555 0.25548902
  0.21956088 9.62034648 10.87357169]]
```

In [405]:

```
#scaler = preprocessing.StandardScaler().fit(Features[:,27:])
#Features[:,27:] = scaler.transform(Features[:,27:])
#print(Features.shape)
#Features[:5,:]
```

```
In [406]: scaler = preprocessing.StandardScaler().fit(Features[:,27:])
Features[:,27:] = scaler.transform(Features[:,27:])
print(Features.shape)
Features[:5,:]
```

(1440, 57)

```
Out[406]: array([[ 0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  1.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   1.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   -7.58497601e-01, -6.32501427e-01, -9.08478744e-02,
  -8.08246103e-01, -7.00967707e-01, -8.88885489e-04,
  -4.66108561e-01, -8.12827288e-01,  1.07326146e+00,
  -1.15483813e+00,  1.56499057e+00,  2.69375944e+00,
  -1.15012527e+00, -8.37388331e-01, -2.08300243e+00,
  -6.22781543e-01, -2.87038803e+00,  1.01595762e+00,
  -1.11210848e+00, -1.93339279e+00, -6.13685354e-01,
  -3.57977556e-01,  1.83041928e+00, -2.86612222e+00,
  -6.01000938e-01, -2.00275874e+00,  1.03873134e-02,
   2.04967729e+00,  1.12514123e+00, -2.76230729e+00],
 [ 0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   1.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   1.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   0.0000000e+00,  0.0000000e+00,  0.0000000e+00,
   -1.00000158e-01, -6.32501427e-01,  5.52002314e-02,
  -6.63096440e-01, -1.10190690e-01, -8.39256295e-01,
  4.88554374e-01, -3.62937966e-01,  6.03341375e-01,
  1.64469320e-01, -1.36870071e+00, -1.36836209e+00,
  -1.65775230e-01,  3.64197078e-01, -8.29994274e-01,
  -6.22781543e-01, -2.12776834e-01,  8.64785135e-01,
  -4.14716199e-01, -6.94773604e-01,  8.46262932e-01,
  -7.53101559e-01, -6.10078262e-01,  6.56912058e-01,
```

```
8.84606558e-02, 5.80404372e-01, -8.92085292e-01,  
1.52135901e-01, -7.28660144e-01, 1.73150142e-01],  
[ 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
1.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
1.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 1.00000000e+00, 0.00000000e+00,  
-8.65136233e-01, -6.32501427e-01, 9.54019390e-01,  
-8.92969826e-01, -7.74988467e-01, -6.55578186e-01,  
-2.37024700e-01, 3.76364575e-01, 1.07326146e+00,  
-1.30381081e+00, -1.95224200e-01, -1.18478545e-01,  
-1.10434155e+00, 1.38491692e-02, -2.04462338e-02,  
-4.35202572e-03, 7.49520180e-02, 6.55340526e-01,  
7.40422025e-01, -7.57004546e-01, 7.58015909e-01,  
-2.97614694e-01, -1.05736955e+00, 1.09317105e+00,  
-1.12534801e+00, -5.17068469e-01, 1.87063632e+00,  
1.07818576e-01, -8.19060688e-01, 1.72118418e-01],  
[ 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 1.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 1.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
1.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
-1.45060112e+00, -6.32501427e-01, -1.68613983e+00,  
-8.35434296e-01, -1.44645999e+00, -7.96243859e-01,  
-2.012644489e+00, -6.98703174e-01, 1.40307375e-01,  
-1.62684681e+00, -6.46561320e-01, -3.34804544e-01,  
4.75196890e-01, -6.66924007e-01, 3.56004611e-01,  
-5.69881826e-02, 1.10700206e+00, 7.60081547e-01,  
-6.56760995e-02, -4.62240066e-03, -8.02031455e-02,  
-8.89887428e-01, -1.92846466e-01, 1.35321325e+00,  
-7.44510021e-01, 2.30862818e-01, 8.57976504e-01,  
-6.84401311e-02, -7.58038895e-01, 1.61106942e+00],  
[ 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,  
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
```

```
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
-8.80432130e-01, -6.32501427e-01, 1.71746930e+00,
-8.92969826e-01, -9.83345020e-01, -7.26473685e-01,
1.34960131e+00, 1.50587521e+00, -1.67996467e+00,
1.78318527e+00, 3.91514057e-01, 2.66101009e-01,
4.29413167e-01, -1.92665928e+00, 1.57985297e+00,
1.96746473e+00, 7.72048922e-01, -1.31844878e+00,
7.25993723e-01, 1.74405601e+00, -8.02031455e-02,
2.53848336e+00, -8.78421083e-01, 5.13708232e-01,
1.83475437e+00, 5.91216702e-01, -2.00036625e+00,
-1.16300905e+00, -4.09970494e-01, 1.39290206e+00]])
```

```
In [407]: import pandas as pd
from sklearn import preprocessing
import sklearn.model_selection as ms
from sklearn import linear_model
import sklearn.metrics as sklm
from sklearn import feature_selection as fs
from sklearn.model_selection import cross_validate
import numpy as np
import numpy.random as nr
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as ss
import math

%matplotlib inline
```

```
In [408]: ## Define the variance threshold and fit the threshold to the feature array.
sel = fs.VarianceThreshold(threshold=.8 * (1 - .8))
Features = sel.fit_transform(Features)
```

```
In [409]: TestFeatures = sel.fit_transform(TestFeatures)
```

```
In [410]: ## Print the support and shape for the transformed features
```

```
print(sel.get_support())
print(Features.shape)
```

```
[False False  
 False False False False False False False False True False False False  
 False True False True  
 True True True True True True True True True True True True True  
 True True True True True True True True True True True True]  
(1440, 32)
```

```
In [411]: ## Print the support and shape for the transformed features
```

```
print(sel.get_support())
print(TestFeatures.shape)
```

```
[False False  
 False False False False False False False False True False False False  
 False True False True  
 True True True True True True True True True True True True True  
 True True True True True True True True True True True True]  
(1576, 32)
```

```
In [412]: ## Randomly sample cases to create independent training and test data
```

```
nr.seed(9988)
indx = range(Features.shape[0])
indx = ms.train_test_split(indx, test_size = 469)
X_train = Features[indx[0],:]
y_train = np.ravel(labels[indx[0]])
X_test = Features[indx[1],:]
y_test = np.ravel(labels[indx[1]])
```

```
In [413]: import pandas as pd
import sklearn.model_selection as ms
from sklearn import linear_model
import sklearn.metrics as sklm
import sklearn.decomposition as skde
import numpy as np
import numpy.random as nr
import matplotlib.pyplot as plt
import math

%matplotlib inline
```

```
In [414]: ## define and fit the linear regression model
lin_mod = linear_model.LinearRegression()
lin_mod.fit(X_train, y_train)

def print_metrics(y_true, y_predicted):
    ## First compute R^2 and the adjusted R^2
    r2 = sklm.r2_score(y_true, y_predicted)

    ## Print the usual metrics and the R^2 values
    print('Mean Square Error      = ' + str(sklm.mean_squared_error(y_true, y_predicted)))
    print('Root Mean Square Error = ' + str(math.sqrt(sklm.mean_squared_error(y_true, y_predicted))))
    print('Mean Absolute Error     = ' + str(sklm.mean_absolute_error(y_true, y_predicted)))
    print('Median Absolute Error   = ' + str(sklm.median_absolute_error(y_true, y_predicted)))
    print('R^2                     = ' + str(r2))

def resid_plot(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test.reshape(-1,1), y_score.reshape(-1,1))
    ## now make the residual plots
    sns.regplot(y_score, resids, fit_reg=False)
    plt.title('Residuals vs. predicted values')
    plt.xlabel('Predicted values')
    plt.ylabel('Residual')
    plt.show()

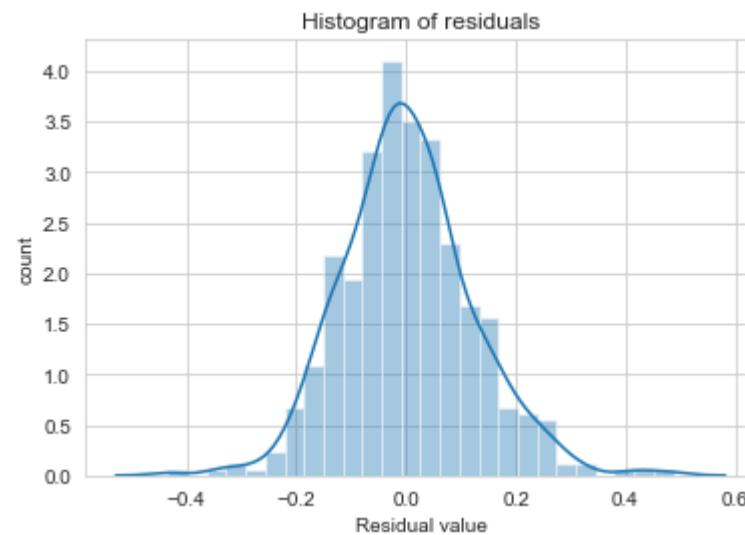
def hist_resids(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test.reshape(-1,1), y_score.reshape(-1,1))
    ## now make the residual plots
    sns.distplot(resids)
    plt.title('Histogram of residuals')
    plt.xlabel('Residual value')
    plt.ylabel('count')
    plt.show()

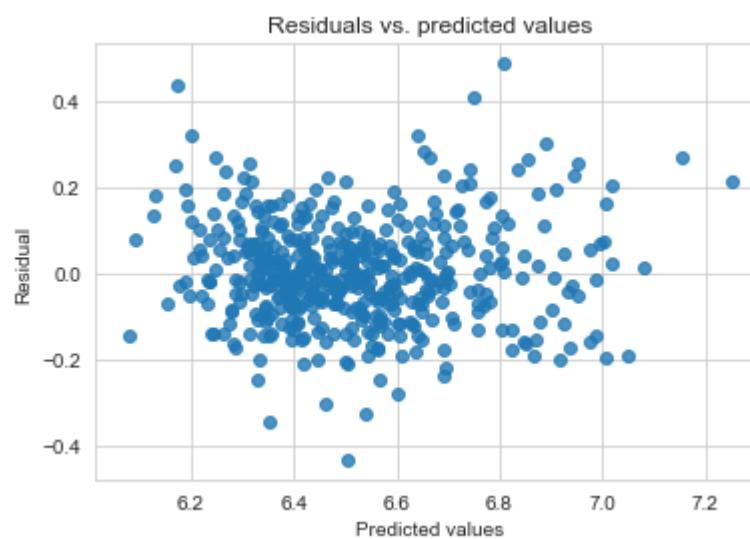
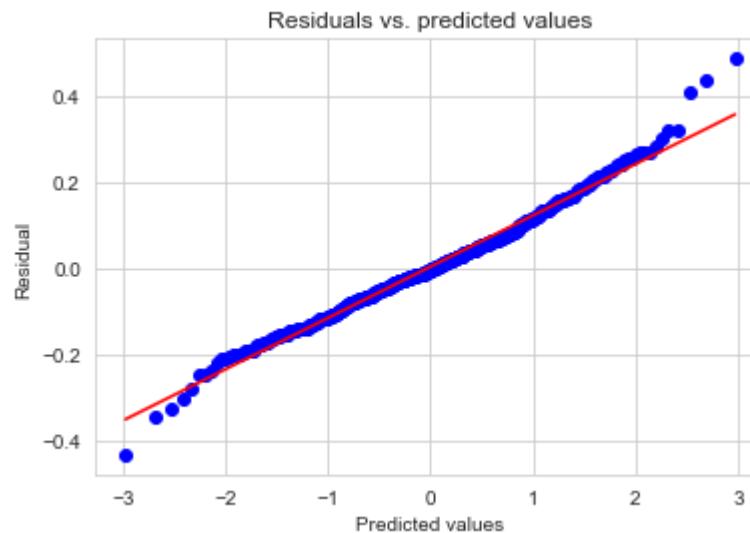
def resid_qq(y_test, y_score):
    ## first compute vector of residuals.
    resids = np.subtract(y_test, y_score)
    ## now make the residual plots
    ss.probplot(resids.flatten(), plot = plt)
    plt.title('Residuals vs. predicted values')
    plt.xlabel('Predicted values')
```

```
plt.ylabel('Residual')
plt.show()

y_score = lin_mod.predict(X_test)
print_metrics(y_test, y_score)
hist_resids(y_test, y_score)
resid_qq(y_test, y_score)
resid_plot(y_test, y_score)
```

Mean Square Error = 0.014236399579523135
Root Mean Square Error = 0.11931638437164921
Mean Absolute Error = 0.09130573283864644
Median Absolute Error = 0.06955492997663004
 R^2 = 0.7376117678489192





```
In [415]: yhat = lin_mod.predict(TestFeatures)  
yhat
```

```
Out[415]: array([6.52047821, 6.81112063, 6.59988789, ..., 6.98892003, 6.53822911,  
6.38788306])
```

```
In [416]: #test_values['gross_rent'] = np.exp(yhat).astype(int)
```

```
In [417]: #test_values.head(5)
```

```
In [418]: #Prediction = test_values[['row_id', 'gross_rent']]
```

```
In [419]: #Prediction.to_csv(r'C:\Users\Jorge\OneDrive\DAT102x.2\Prediction.csv', index = None, header=True)
```

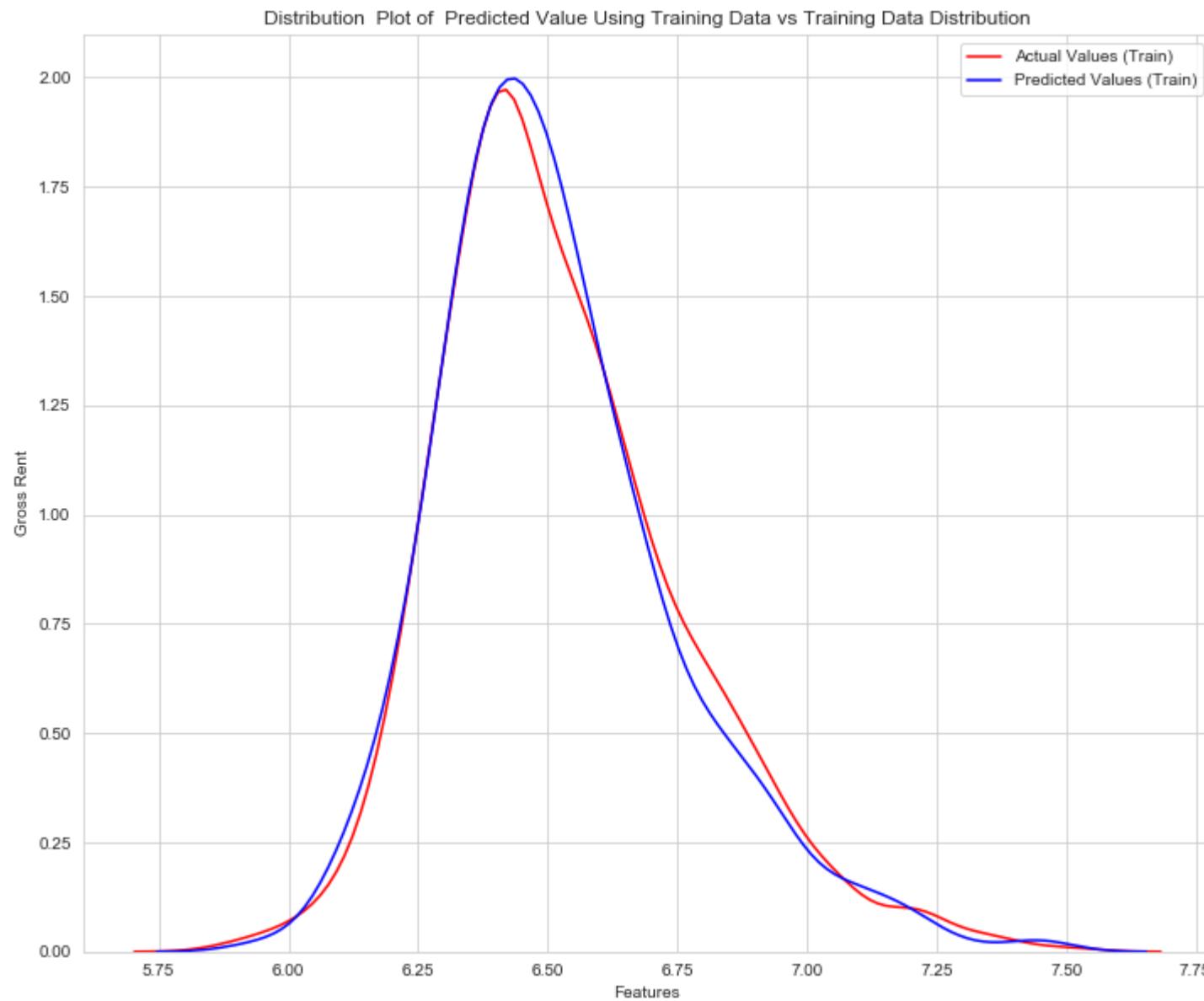
```
In [420]: def DistributionPlot(RedFunction, BlueFunction, RedName, BlueName, Title):
    width = 12
    height = 10
    plt.figure(figsize=(width, height))

    ax1 = sns.distplot(RedFunction, hist=False, color="r", label=RedName)
    ax2 = sns.distplot(BlueFunction, hist=False, color="b", label=BlueName, ax=ax1)

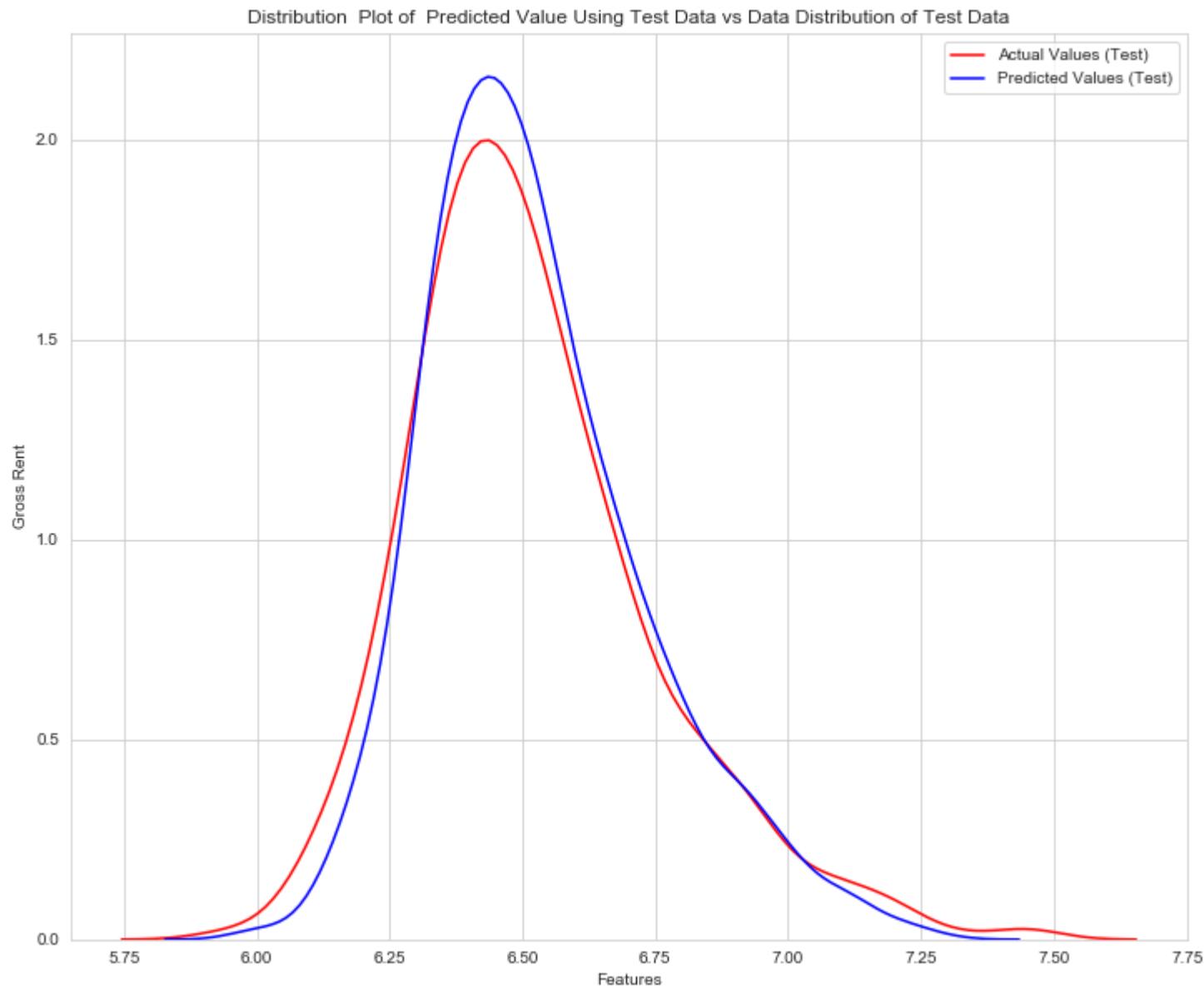
    plt.title>Title)
    plt.xlabel('Features')
    plt.ylabel('Gross Rent')

    plt.show()
    plt.close()
```

```
In [421]: Title = 'Distribution Plot of Predicted Value Using Training Data vs Training Data Distribution'  
DistributionPlot(y_train, y_test, "Actual Values (Train)", "Predicted Values (Train)", Title)
```




```
In [422]: Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'  
DistributionPlot(y_test,yhat,"Actual Values (Test)","Predicted Values (Test)",Title)
```



```
In [423]: from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear_model import LinearRegression
```

```
In [424]: pr = PolynomialFeatures(degree=1)  
X_train_pr = pr.fit_transform(X_train)  
X_test_pr = pr.fit_transform(X_test)  
pr
```

```
Out[424]: PolynomialFeatures(degree=1, include_bias=True, interaction_only=False,  
                                order='C')
```

```
In [425]: poly = LinearRegression()  
poly.fit(X_train_pr, y_train)
```

```
Out[425]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [426]: yhat5 = poly.predict(X_test_pr)
yhat5[0:5]
```

```
Out[426]: array([6.56376644, 6.55658476, 6.26272404, 6.95347763, 6.84551315])
```

```
In [427]: TrueTestpr = pr.fit_transform(TestFeatures)
```

```
In [428]: yhat6 = poly.predict(TrueTestpr)
yhat6
```

```
Out[428]: array([6.52047821, 6.81112063, 6.59988789, ..., 6.98892003, 6.53822911,
6.38788306])
```

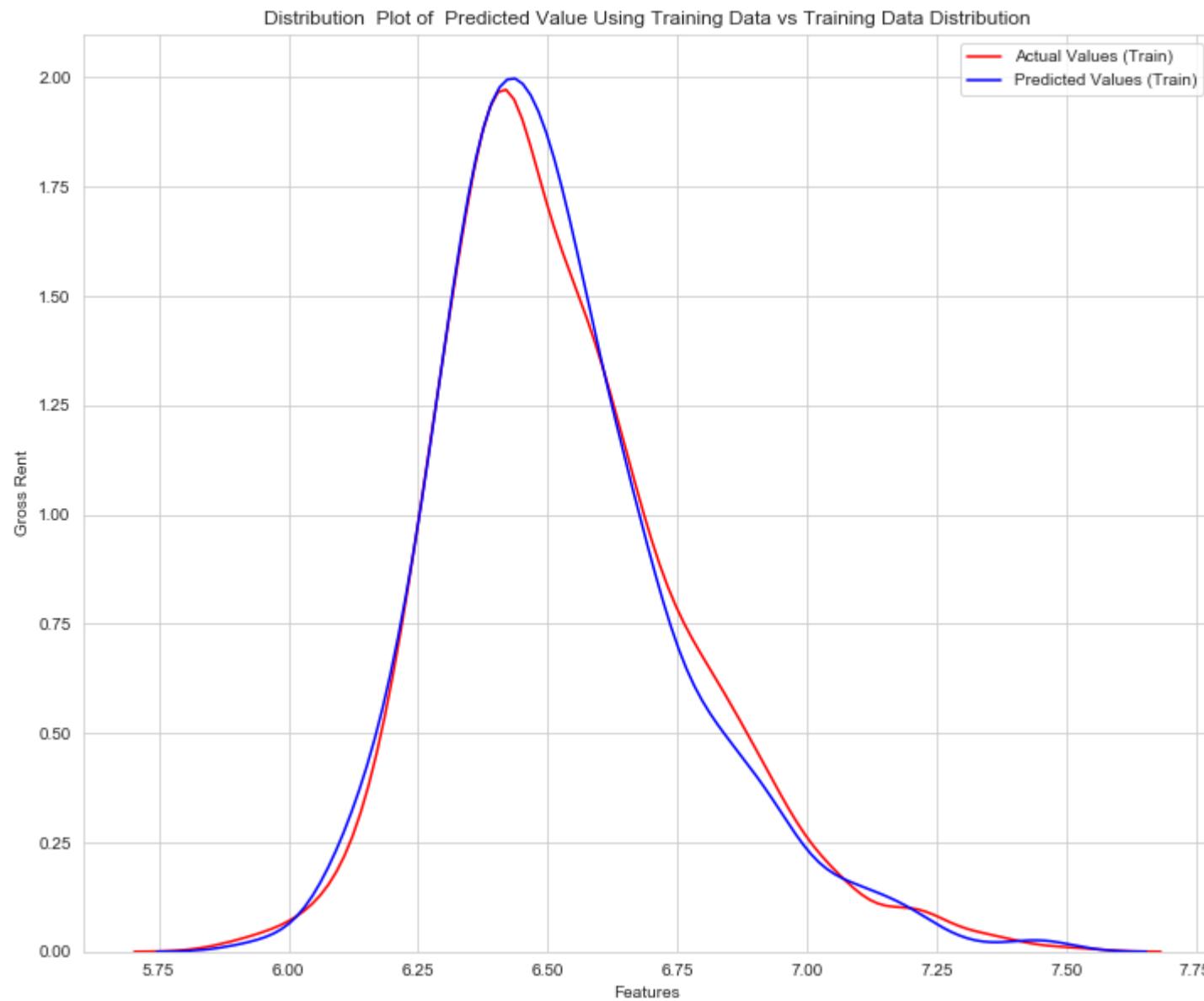
```
In [429]: poly.score(X_train_pr, y_train)
```

```
Out[429]: 0.7751981001715167
```

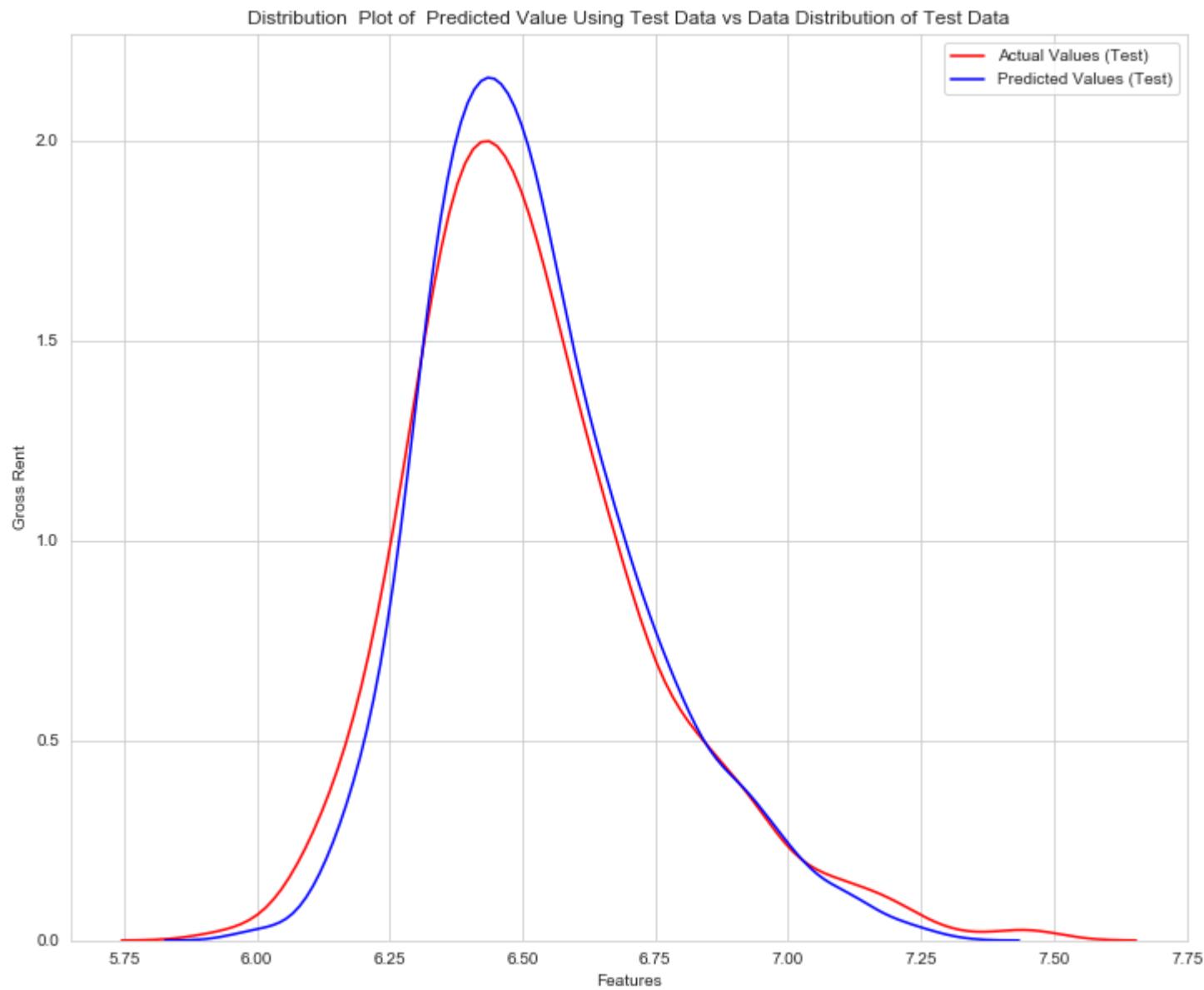
```
In [430]: poly.score(X_test_pr, y_test)
```

```
Out[430]: 0.7376117678489187
```

```
In [431]: Title = 'Distribution Plot of Predicted Value Using Training Data vs Training Data Distribution'  
DistributionPlot(y_train, y_test, "Actual Values (Train)", "Predicted Values (Train)", Title)
```




```
In [432]: Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'  
DistributionPlot(y_test,yhat6,"Actual Values (Test)","Predicted Values (Test)",Title)
```



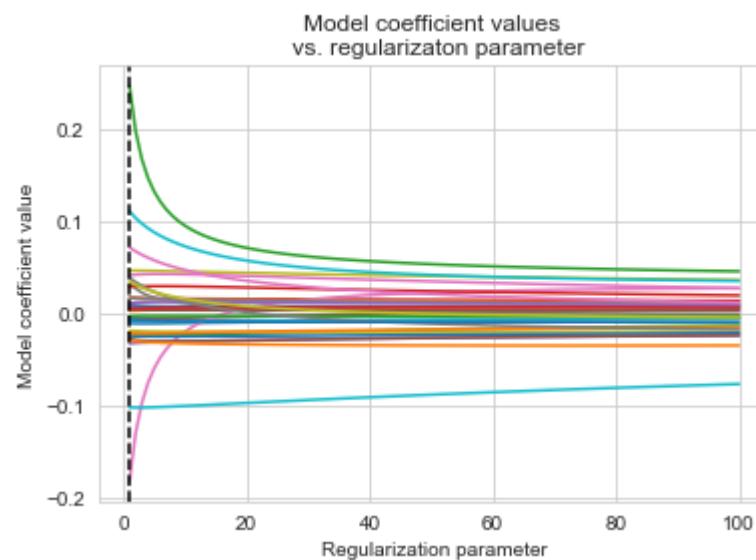
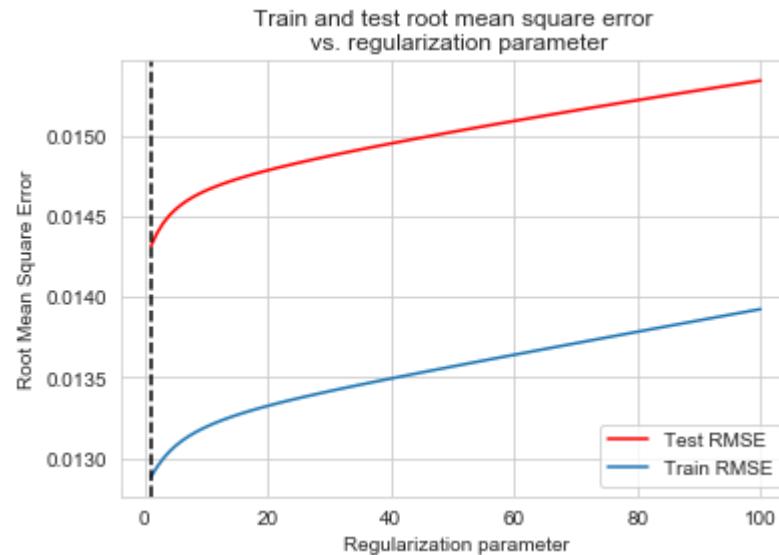

```
In [433]: def plot_regularization(l, train_RMSE, test_RMSE, coefs, min_idx, title):
    plt.plot(l, test_RMSE, color = 'red', label = 'Test RMSE')
    plt.plot(l, train_RMSE, label = 'Train RMSE')
    plt.axvline(min_idx, color = 'black', linestyle = '--')
    plt.legend()
    plt.xlabel('Regularization parameter')
    plt.ylabel('Root Mean Square Error')
    plt.title(title)
    plt.show()

    plt.plot(l, coefs)
    plt.axvline(min_idx, color = 'black', linestyle = '--')
    plt.title('Model coefficient values \n vs. regularizaton parameter')
    plt.xlabel('Regularization parameter')
    plt.ylabel('Model coefficient value')
    plt.show()

def test_regularization_l2(x_train, y_train, x_test, y_test, l2):
    train_RMSE = []
    test_RMSE = []
    coefs = []
    for reg in l2:
        lin_mod = linear_model.Ridge(alpha = reg)
        lin_mod.fit(x_train, y_train)
        coefs.append(lin_mod.coef_)
        y_score_train = lin_mod.predict(x_train)
        train_RMSE.append(sklearn.mean_squared_error(y_train, y_score_train))
        y_score = lin_mod.predict(x_test)
        test_RMSE.append(sklearn.mean_squared_error(y_test, y_score))
    min_idx = np.argmin(test_RMSE)
    min_l2 = l2[min_idx]
    min_RMSE = test_RMSE[min_idx]

    title = 'Train and test root mean square error \n vs. regularization parameter'
    plot_regularization(l2, train_RMSE, test_RMSE, coefs, min_l2, title)
    return min_l2, min_RMSE

l2 = [x for x in range(1,101)]
out_l2 = test_regularization_l2(X_train, y_train, X_test, y_test, l2)
print(out_l2)
```

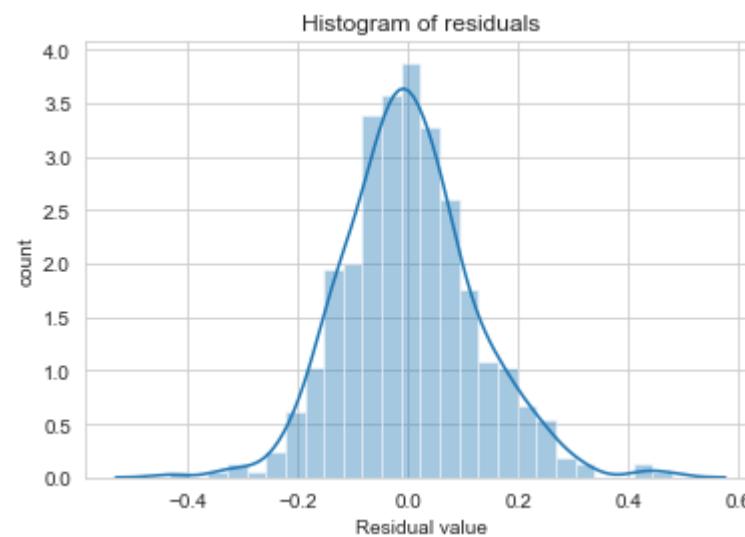


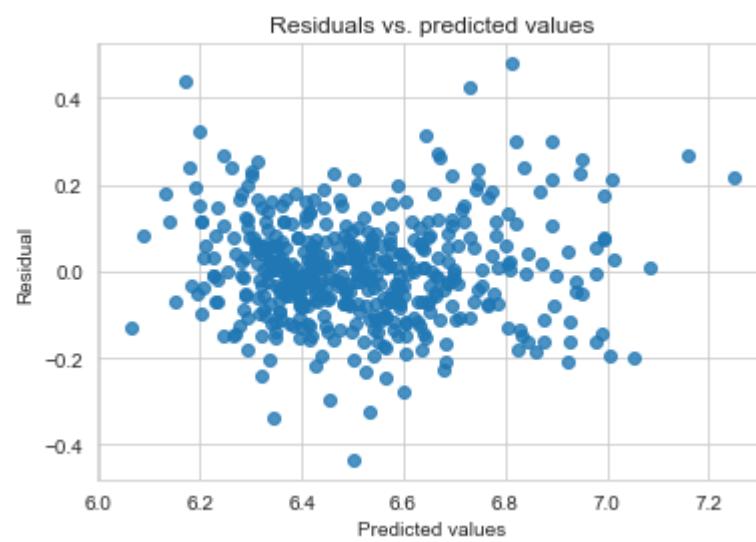
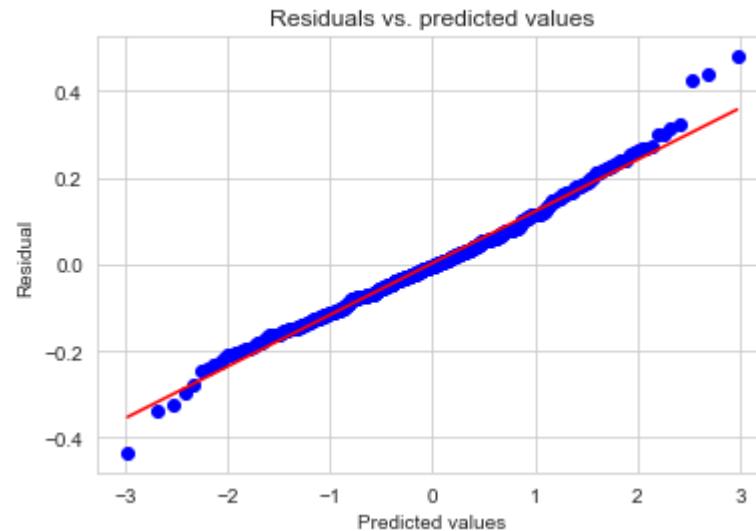
(1, 0.014319236227959923)

```
In [434]: lin_mod_12 = linear_model.Ridge(alpha = out_12[0])
lin_mod_12.fit(X_train, y_train)
y_score_12 = lin_mod_12.predict(X_test)

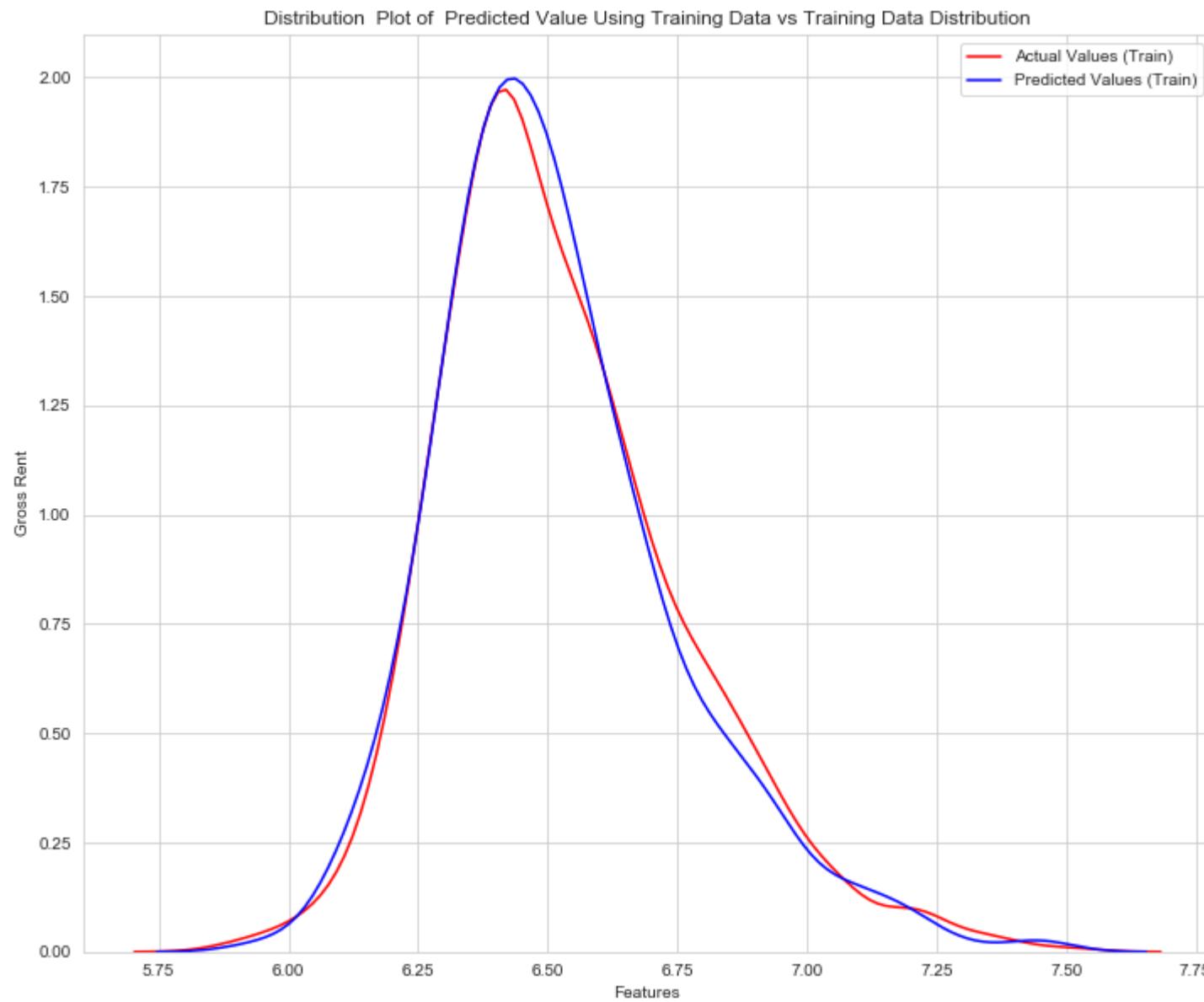
print_metrics(y_test, y_score_12)
hist_resids(y_test, y_score_12)
resid_qq(y_test, y_score_12)
resid_plot(y_test, y_score_12)
```

Mean Square Error = 0.014319236227959923
Root Mean Square Error = 0.1196630111102003
Mean Absolute Error = 0.09155710001076296
Median Absolute Error = 0.07266996323015373
 R^2 = 0.7360850221560046

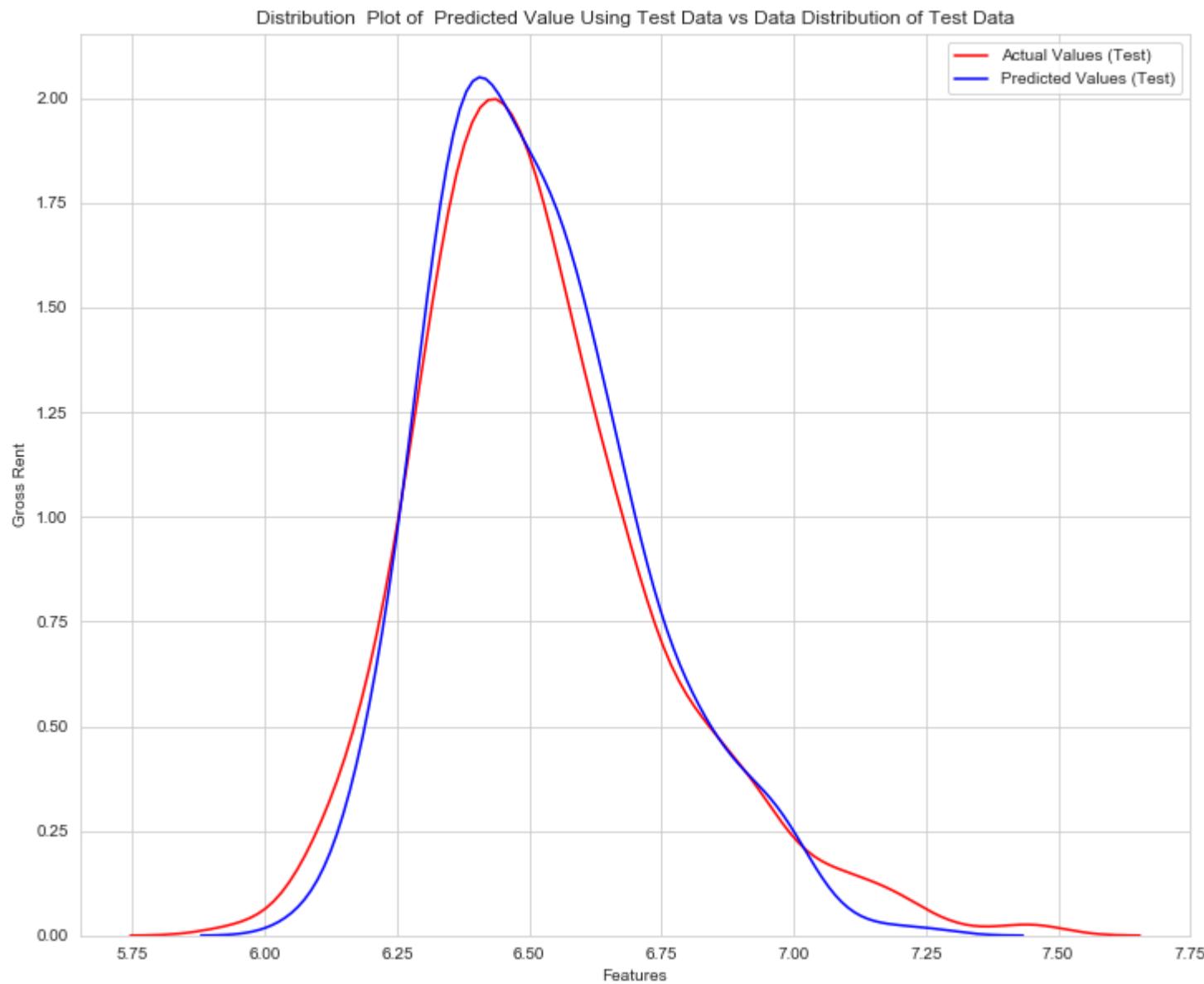




```
In [435]: Title = 'Distribution Plot of Predicted Value Using Training Data vs Training Data Distribution'  
DistributionPlot(y_train, y_test, "Actual Values (Train)", "Predicted Values (Train)", Title)
```



```
In [436]: Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'  
DistributionPlot(y_test,y_score_12,"Actual Values (Test)","Predicted Values (Test)",Title)
```



```
In [437]: #yhat9 = lin_mod_l2.predict(TestFeatures)
```

```
In [438]: #test_values['gross_rent'] = np.exp(yhat9).astype(int)
```

```
In [439]: #test_values.head(5)
```

```
In [440]: #Prediction = test_values[['row_id','gross_rent']]
```

```
In [441]: #Prediction.to_csv(r'C:\Users\Jorge\OneDrive\DAT102x.2\Prediction.csv', index = None, header=True)
```

```
In [442]: import math
import pandas
#from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV, cross_validate
from sklearn.utils import shuffle
```

```
In [443]: #def svr_model(X, y):
#    gsc = GridSearchCV(
#        estimator=SVR(kernel='rbf'),
#        param_grid={
#            'C': [0.1, 1, 100, 1000],
#            'epsilon': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10],
#            'gamma': [0.0001, 0.001, 0.005, 0.1, 1, 3, 5]
#        },
#        cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=-1)
#
#    grid_result = gsc.fit(X, y)
#    best_params = grid_result.best_params_
#    best_svr = SVR(kernel='rbf', C=best_params["C"], epsilon=best_params["epsilon"], gamma=best_params["gamma"],
#                    coef0=0.1, shrinking=True,
#                    tol=0.001, cache_size=200, verbose=False, max_iter=-1)
#
#    scoring = {
#        'abs_error': 'neg_mean_absolute_error',
#        'squared_error': 'neg_mean_squared_error'}
#
#    scores = cross_validate(best_svr, X, y, cv=10, scoring=scoring, return_train_score=True)
#    return "MAE :", abs(scores['test_abs_error'].mean()), "| RMSE :", math.sqrt(abs(scores['test_squared_error']))
## Run
#print(svr_model(X_train,y_train))
```

```
In [444]: best_svr = SVR(C=3301, cache_size=200, coef0=0.1, degree=3, epsilon=0.0005, gamma=0.0001,
kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=False)
```

```
In [445]: #best_svr = SVR(C=1000, cache_size=200, coef0=0.1, degree=3, epsilon=0.0005, gamma=0.0001,
#    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
In [446]: #def svr_model(X, y):
#    gsc = GridSearchCV(
#        estimator=SVR(kernel='linear'),
#        param_grid={
#            'C': [0.1, 1, 100, 1000],
#            'epsilon': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10],
#        },
#        cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=-1)
#
#    grid_result = gsc.fit(X, y)
#    best_params = grid_result.best_params_
#    best_svr = SVR(kernel='linear', C=best_params["C"], epsilon=best_params["epsilon"], coef0=0.1, shrinking=True,
#                    tol=0.001, cache_size=200, verbose=False, max_iter=-1)
#
#    scoring = {
#        'abs_error': 'neg_mean_absolute_error',
#        'squared_error': 'neg_mean_squared_error'}
#
#    scores = cross_validate(best_svr, X, y, cv=10, scoring=scoring, return_train_score=True)
#    return "MAE :", abs(scores['test_abs_error'].mean()), "| RMSE :", math.sqrt(abs(scores['test_squared_error']))
# Run
#print(svr_model(X_train,y_train))
```

```
In [447]: #best_svr = SVR(C=1000, cache_size=200, coef0=0.1, degree=3, epsilon=0.0005, gamma=0.0001,
#    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
In [448]: #from sklearn.preprocessing import QuantileTransformer
#transformer = QuantileTransformer().fit(X_train)
#transformer.transform(X_train)
```

```
In [449]: best_svr.fit(X_train, y_train)
```

```
Out[449]: SVR(C=3301, cache_size=200, coef0=0.1, degree=3, epsilon=0.0005, gamma=0.0001,
    kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=False)
```

```
In [450]: best_svr.score(X_train, y_train)
```

```
Out[450]: 0.845568218919428
```

```
In [451]: best_svr.score(X_test, y_test)
```

```
Out[451]: 0.7844550069048879
```

```
In [452]: yhat10 = best_svr.predict(TestFeatures)
```

```
In [453]: #test_values['gross_rent'] = np.exp(yhat10).astype(int)
```

```
In [454]: #test_values.head(5)
```

```
In [455]: #Prediction = test_values[['row_id', 'gross_rent']]
```

```
In [456]: #Prediction.to_csv(r'C:\Users\Jorge\OneDrive\DAT102x.2\Prediction.csv', index = None, header=True)
```

```
In [457]:  
#from sklearn.ensemble import GradientBoostingRegressor
```

```
In [458]:  
#model_GradR2_GS= GradientBoostingRegressor(learning_rate=.1, n_estimators=230, max_depth=3, min_samples_Leaf=3,  
#model_GradR2_GS.fit(X_train,y_train)
```

```
In [459]: #from sklearn.preprocessing import QuantileTransformer  
#transformer = QuantileTransformer().fit(X_train)  
#transformer.transform(X_train)
```

```
In [460]: #model_GradR2_GS.score(X_train, y_train)
```

```
In [461]: #model_GradR2_GS.score(X_test, y_test)
```

```
In [462]: #yhat11 = model_GradR2_GS.predict(TestFeatures)
```

```
In [463]: test_values['gross_rent'] = np.exp(yhat10).astype(int)
```

```
In [464]: test_values['gross_rent'] = np.exp(yhat10).astype(int)
```

```
In [465]: Prediction = test_values[['row_id','gross_rent']]
```

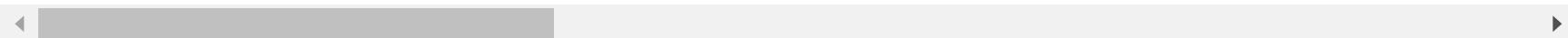
```
In [466]: Prediction.to_csv(r'C:\Users\Jorge\OneDrive\DAT102x.2\Prediction.csv', index = None, header=True)
```

```
In [467]: test_values.head(5)
```

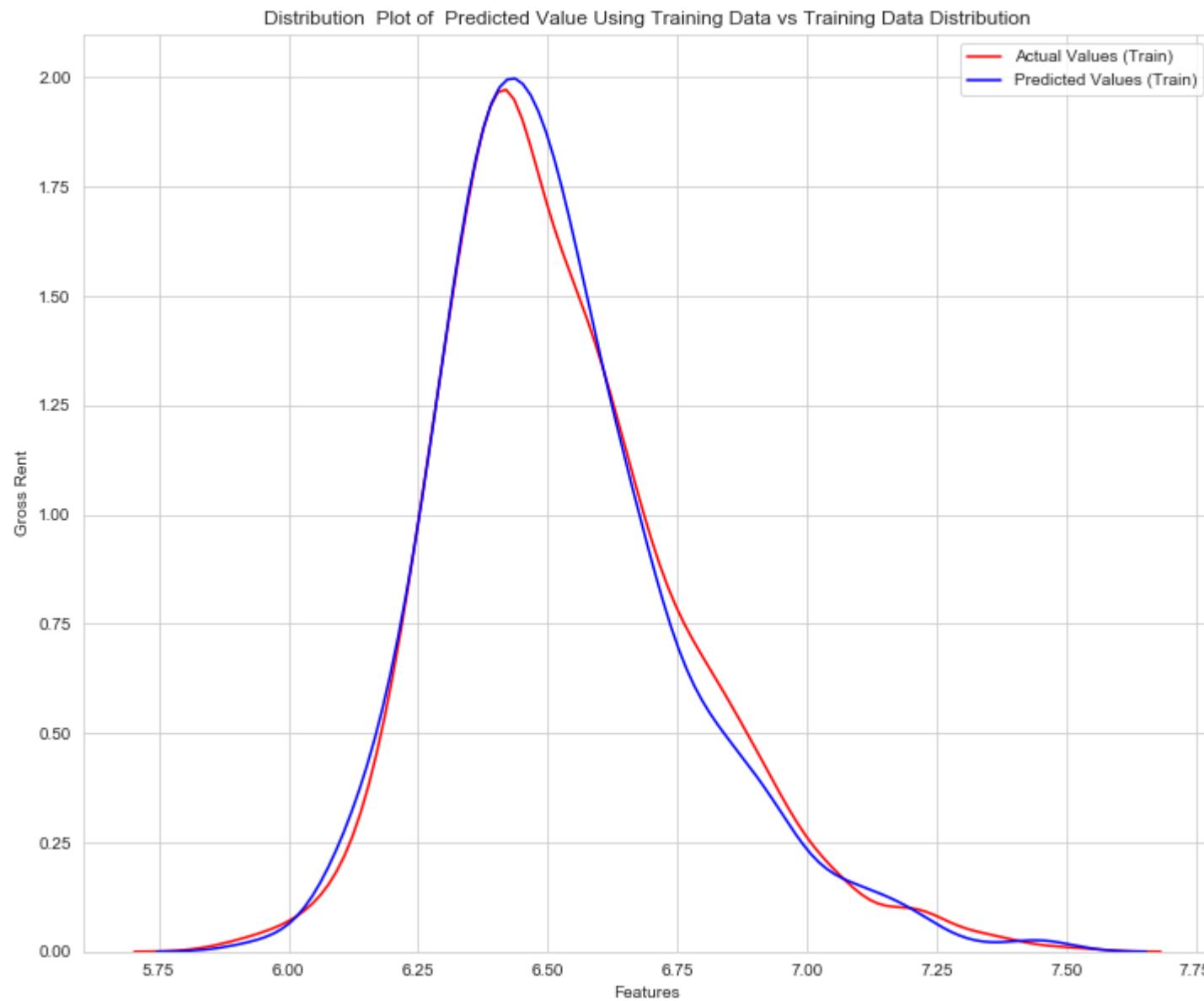
Out[467]:

	row_id	county_code	state	population	renter_occupied_households	pct_renter_occupied	evictions	rent_burden	pct_white	pct_af_an
0	0	873	32	52842.0	5403.0	26.840	27.0	27.960	0.924234	0.025041
1	1	675	37	212287.0	53502.0	57.534	6032.0	33.072	0.398318	0.484631
2	2	1100	9	81263.0	13368.0	39.994	1012.0	32.044	0.483789	0.381910
3	3	1562	7	122870.0	19359.0	41.865	27.0	30.724	0.468043	0.088391
4	4	379	23	146153.0	15766.0	30.681	644.0	30.860	0.651511	0.251311

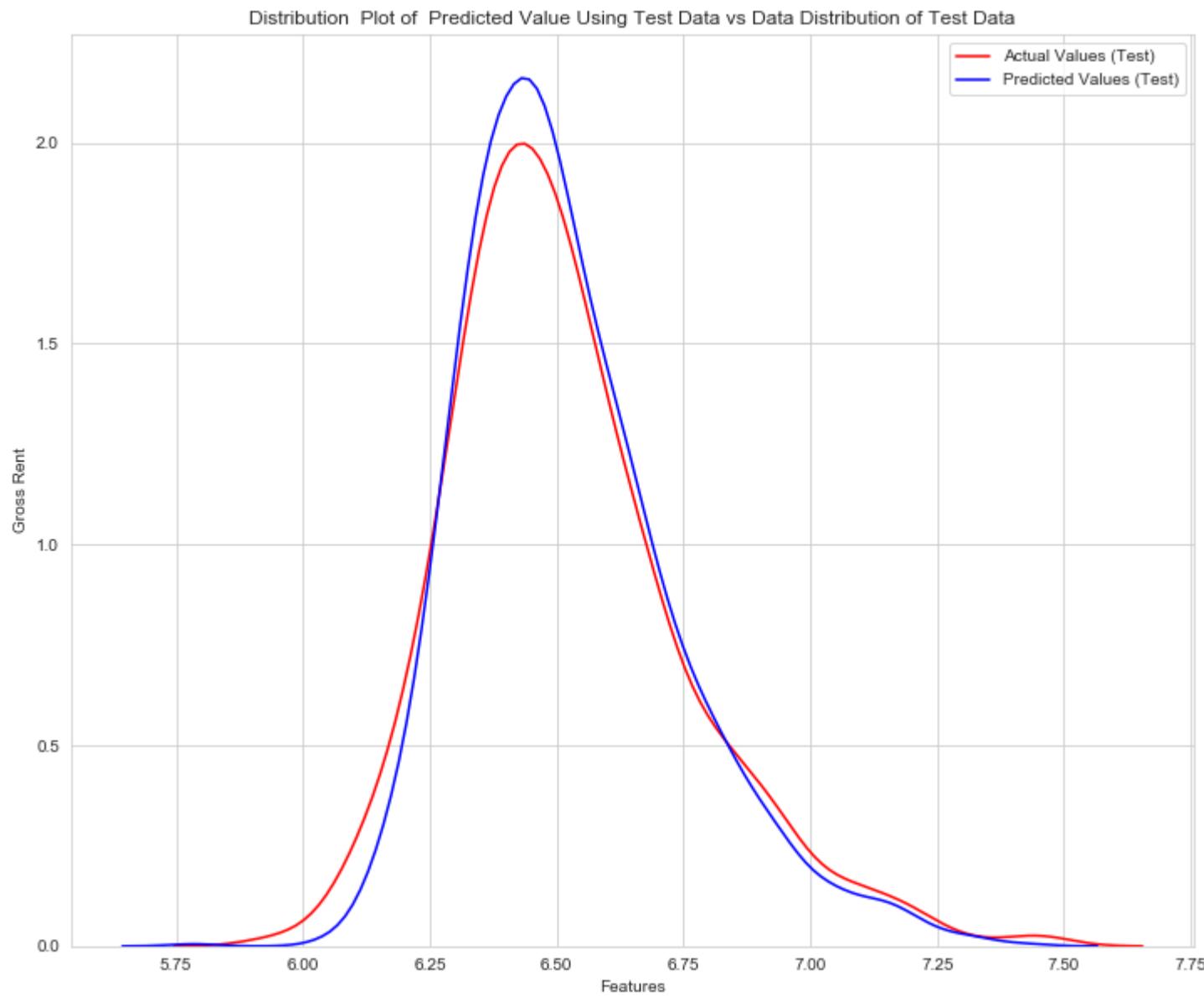
5 rows × 84 columns



```
In [468]: Title = 'Distribution Plot of Predicted Value Using Training Data vs Training Data Distribution'  
DistributionPlot(y_train, y_test, "Actual Values (Train)", "Predicted Values (Train)", Title)
```




```
In [469]: Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'  
DistributionPlot(y_test, yhat10,"Actual Values (Test)","Predicted Values (Test)",Title)
```




```
In [470]: #def test_regularization_l1(x_train, y_train, x_test, y_test, l1):
#    train_RMSE = []
#    test_RMSE = []
#    coefs = []
#    for reg in l1:
#        lin_mod = Linear_model.Lasso(alpha = reg, max_iter=1e7) #tol=0.01
#        lin_mod.fit(x_train, y_train)
#        coefs.append(lin_mod.coef_)
#        y_score_train = lin_mod.predict(x_train)
#        train_RMSE.append(sklm.mean_squared_error(y_train, y_score_train))
#        y_score = lin_mod.predict(x_test)
#        test_RMSE.append(sklm.mean_squared_error(y_test, y_score))
#    min_idx = np.argmin(test_RMSE)
#    min_l1 = l1[min_idx]
#    min_RMSE = test_RMSE[min_idx]
#
#    title = 'Train and test root mean square error \n vs. regularization parameter'
#    plot_regularization(l1, train_RMSE, test_RMSE, coefs, min_l1, title)
#    return min_l1, min_RMSE
#
#l1 = [x/5000 for x in range(1,101)]
#out_l1 = test_regularization_l1(X_train, y_train, X_test, y_test, l1)
#print(out_l1)
```

```
In [471]: #lin_mod_l1 = Linear_model.Lasso(alpha = out_l1[0], max_iter=1e7)
#lin_mod_l1.fit(X_train, y_train)
#y_score_l1 = lin_mod_l1.predict(X_test)
#
#print_metrics(y_test, y_score_l1)
#hist_resids(y_test, y_score_l1)
#resid_qq(y_test, y_score_l1)
#resid_plot(y_test, y_score_l1)
```

```
In [472]: #yhat12 = lin_mod_l1.predict(TestFeatures)
```

```
In [473]: #test_values['gross_rent'] = np.exp(yhat11).astype(int)
```

```
In [474]: #test_values.head(5)
```

```
In [475]: #Title = 'Distribution Plot of Predicted Value Using Training Data vs Training Data Distribution'
#DistributionPlot(y_train, y_test, "Actual Values (Train)", "Predicted Values (Train)", Title)
```

```
In [476]: #Title='Distribution Plot of Predicted Value Using Test Data vs Data Distribution of Test Data'
#DistributionPlot(y_test, yhat12,"Actual Values (Test)","Predicted Values (Test)",Title)
```

```
In [477]: #MLPRegressor(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9,
#                  beta_2=0.999, early_stopping=False, epsilon=1e-08,
#                  hidden_layer_sizes=3, learning_rate='constant',
#                  learning_rate_init=0.001, max_iter=200, momentum=0.9,
#                  n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
#                  random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
#                  validation_fraction=0.1, verbose=False, warm_start=False)
```

```
In [478]: def print_metrics(y_true, y_predicted, n_parameters):
    ## First compute R^2 and the adjusted R^2
    r2 = sklm.r2_score(y_true, y_predicted)
    r2_adj = r2 - (n_parameters - 1)/(y_true.shape[0] - n_parameters) * (1 - r2)

    ## Print the usual metrics and the R^2 values
    print('Mean Square Error      = ' + str(sklm.mean_squared_error(y_true, y_predicted)))
    print('Root Mean Square Error = ' + str(math.sqrt(sklm.mean_squared_error(y_true, y_predicted))))
    print('Mean Absolute Error     = ' + str(sklm.mean_absolute_error(y_true, y_predicted)))
    print('Median Absolute Error   = ' + str(sklm.median_absolute_error(y_true, y_predicted)))
    print('R^2                     = ' + str(r2))
    print('Adjusted R^2            = ' + str(r2_adj))

    y_score = best_svr.predict(X_test)
    print_metrics(y_test, y_score, 28)
```

```
Mean Square Error      = 0.011694825731745122
Root Mean Square Error = 0.1081426175554537
Mean Absolute Error     = 0.08335513894390041
Median Absolute Error   = 0.06924876656249168
R^2                     = 0.7844550069048879
Adjusted R^2            = 0.7712583746745749
```

```
In [ ]:
```

