Analyzing Real Estate Trends by Metropolitan Statistical Areas (MSA)

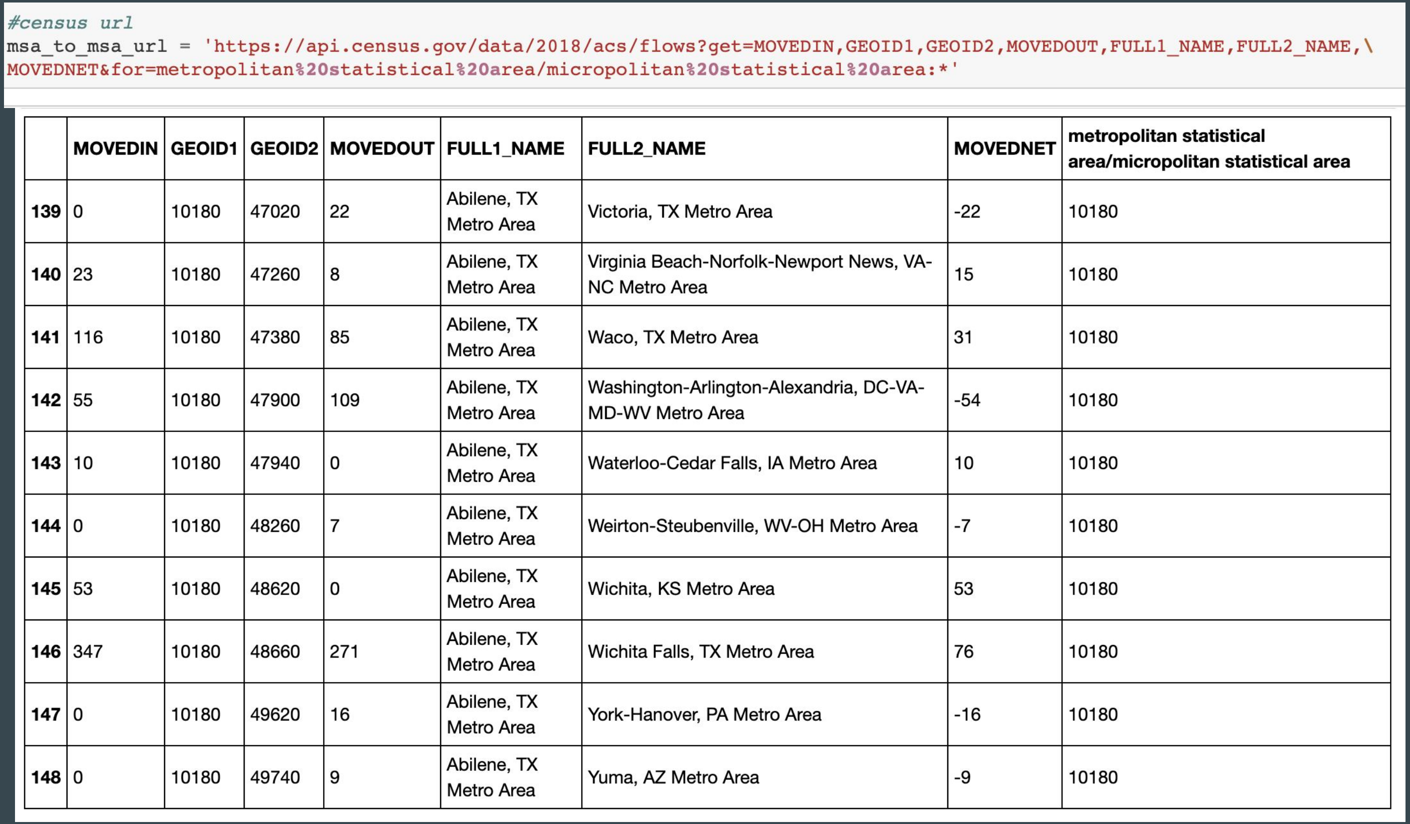
•••

Target: Project Future Home Value with Multiple Linear Regression

Can we project which city to invest in based on certain criteria?

1. Migration
2. Population
3. Crime
4. Education
5. Environmental Factors

Census Data to Evaluate Migration Trends



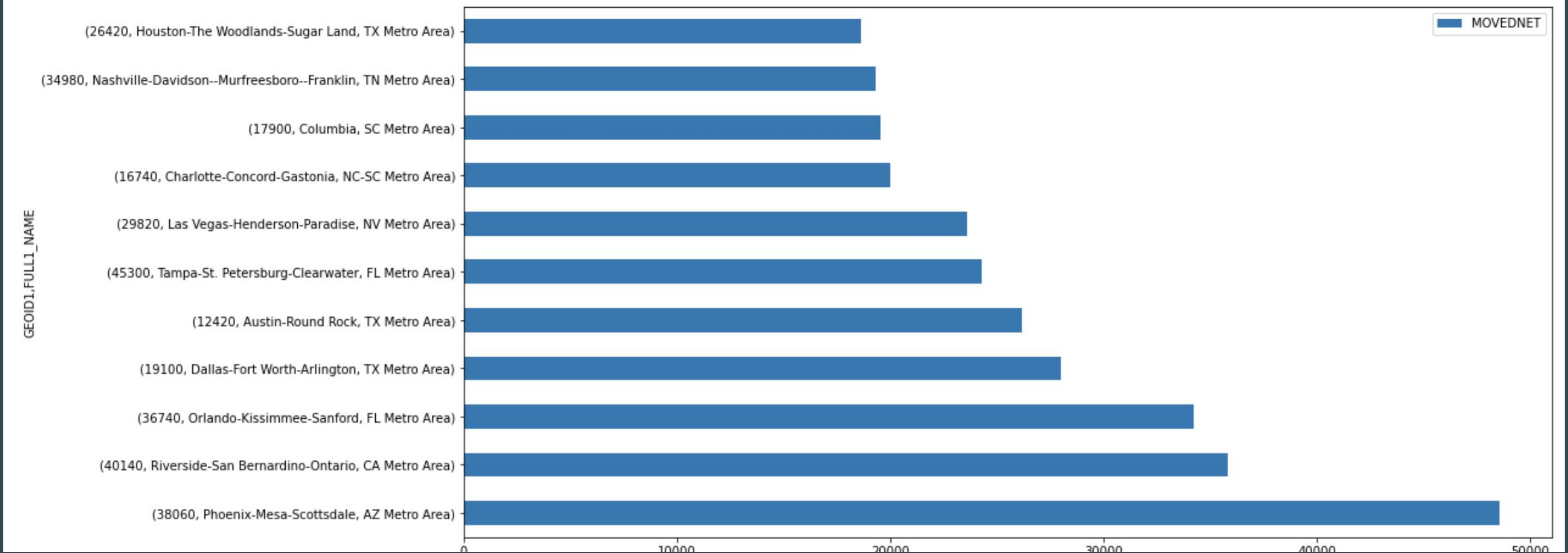
```
#census url
msa_to_msa_url = 'https://api.census.gov/data/2018/acs/flows?get=MOVEDIN,GEOID1,GEOID2,MOVEDOUT,FULL1_NAME,FULL2_NAME,\
MOVEDNET&for=metropolitan%20statistical%20area/micropolitan%20statistical%20area:*'
```

	MOVEDIN	GEOID1	GEOID2	MOVEDOUT	FULL1_NAME	FULL2_NAME	MOVEDNET	metropolitan statistical area/micropolitan statistical area
139	0	10180	47020	22	Abilene, TX Metro Area	Victoria, TX Metro Area	-22	10180
140	23	10180	47260	8	Abilene, TX Metro Area	Virginia Beach-Norfolk-Newport News, VA-NC Metro Area	15	10180
141	116	10180	47380	85	Abilene, TX Metro Area	Waco, TX Metro Area	31	10180
142	55	10180	47900	109	Abilene, TX Metro Area	Washington-Arlington-Alexandria, DC-VA-MD-WV Metro Area	-54	10180
143	10	10180	47940	0	Abilene, TX Metro Area	Waterloo-Cedar Falls, IA Metro Area	10	10180
144	0	10180	48260	7	Abilene, TX Metro Area	Weirton-Steubenville, WV-OH Metro Area	-7	10180
145	53	10180	48620	0	Abilene, TX Metro Area	Wichita, KS Metro Area	53	10180
146	347	10180	48660	271	Abilene, TX Metro Area	Wichita Falls, TX Metro Area	76	10180
147	0	10180	49620	16	Abilene, TX Metro Area	York-Hanover, PA Metro Area	-16	10180
148	0	10180	49740	9	Abilene, TX Metro Area	Yuma, AZ Metro Area	-9	10180

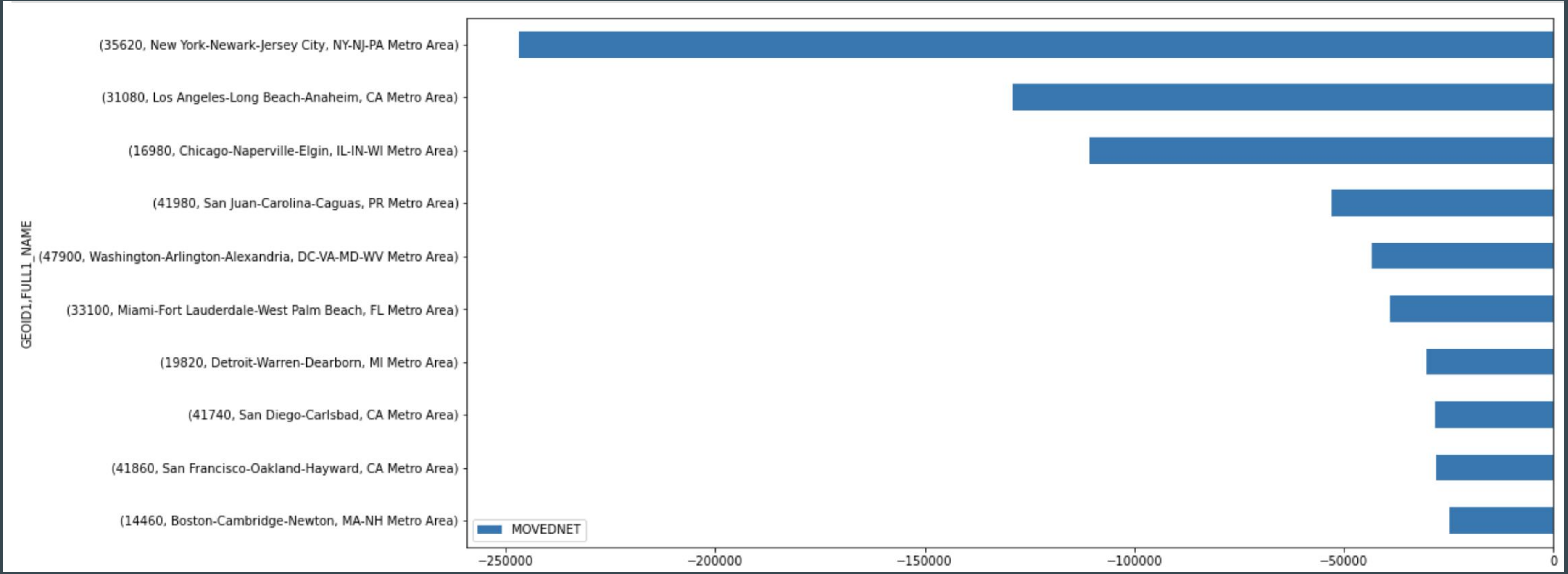
1. Pull Down US Census Data showing Migration from GEO1D1 to GEO1D2

2. Group by GEO1D1 to get Agg of MovedNet (Migration Sum)

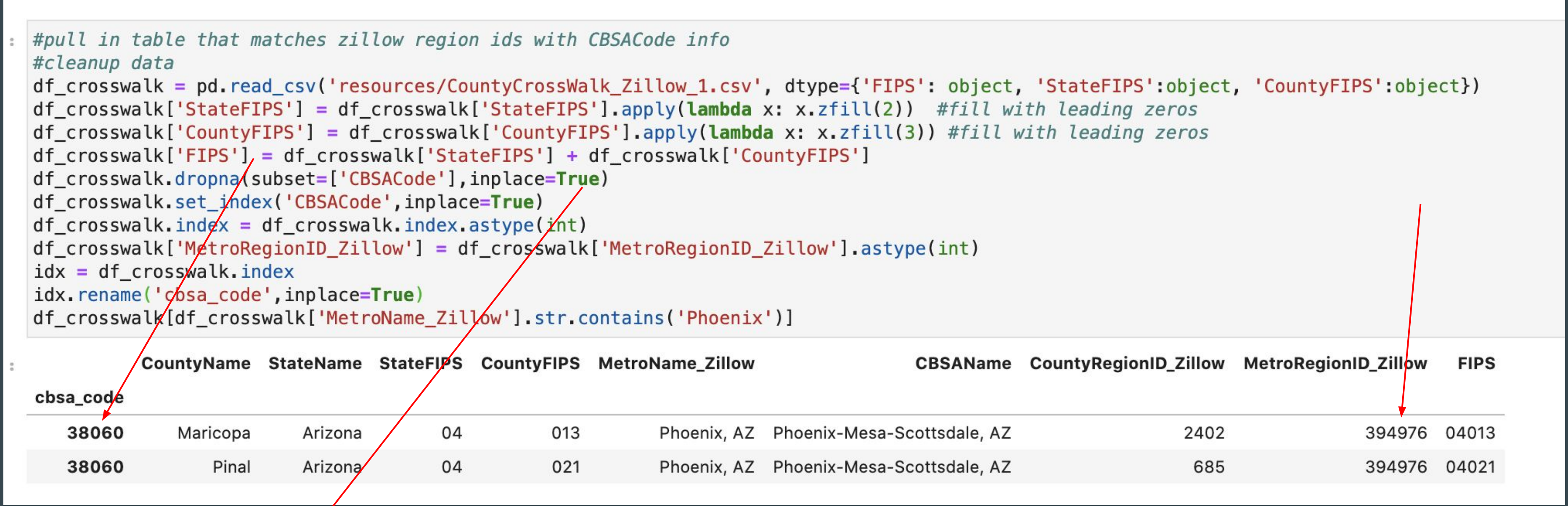
Top 10 Migrated Cities from 2014-2018



Census Data to Evaluate Migration Trends - Top 10 Cities losing Population



Matching Zillow Data to Census Data via Crosswalk

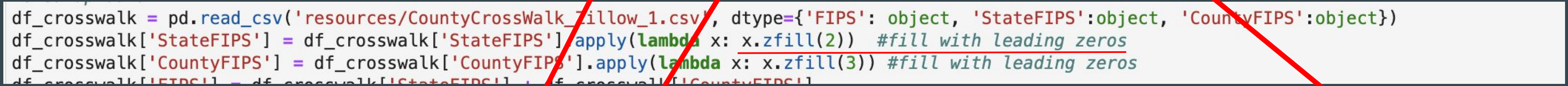


```
#pull in table that matches zillow region ids with CBSACode info
#cleanup data
df_crosswalk = pd.read_csv('resources/CountyCrossWalk_Zillow_1.csv', dtype={'FIPS': object, 'StateFIPS':object, 'CountyFIPS':object})
df_crosswalk['StateFIPS'] = df_crosswalk['StateFIPS'].apply(lambda x: x.zfill(2))  #fill with leading zeros
df_crosswalk['CountyFIPS'] = df_crosswalk['CountyFIPS'].apply(lambda x: x.zfill(3)) #fill with leading zeros
df_crosswalk['FIPS'] = df_crosswalk['StateFIPS'] + df_crosswalk['CountyFIPS']
df_crosswalk.dropna(subset=['CBSACode'],inplace=True)
df_crosswalk.set_index('CBSACode',inplace=True)
df_crosswalk.index = df_crosswalk.index.astype(int)
df_crosswalk['MetroRegionID_Zillow'] = df_crosswalk['MetroRegionID_Zillow'].astype(int)
idx = df_crosswalk.index
idx.rename('cbsa_code',inplace=True)
df_crosswalk[df_crosswalk['MetroName_Zillow'].str.contains('Phoenix')]
```

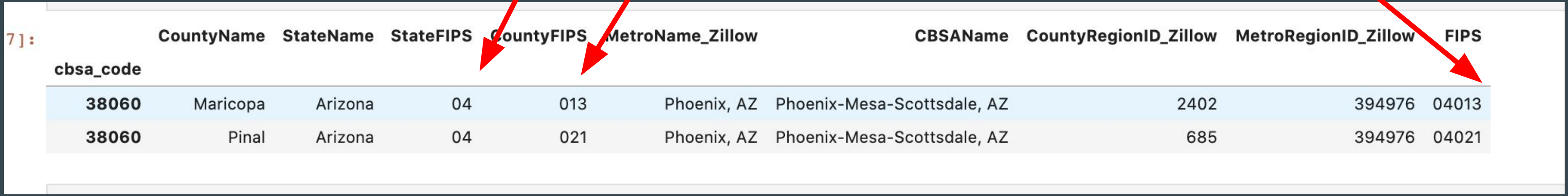
	CountyName	StateName	StateFIPS	CountyFIPS	MetroName_Zillow	CBSAName	CountyRegionID_Zillow	MetroRegionID_Zillow	FIPS
cbsa_code									
38060	Maricopa	Arizona	04	013	Phoenix, AZ	Phoenix-Mesa-Scottsdale, AZ	2402	394976	04013
38060	Pinal	Arizona	04	021	Phoenix, AZ	Phoenix-Mesa-Scottsdale, AZ	685	394976	04021

(38060, Phoenix-Mesa-Scottsdale, AZ Metro Area)
0
10000
20000
30000
40000
50000

Matching Zillow Data to Census Data via Crosswalk: Data Cleanup



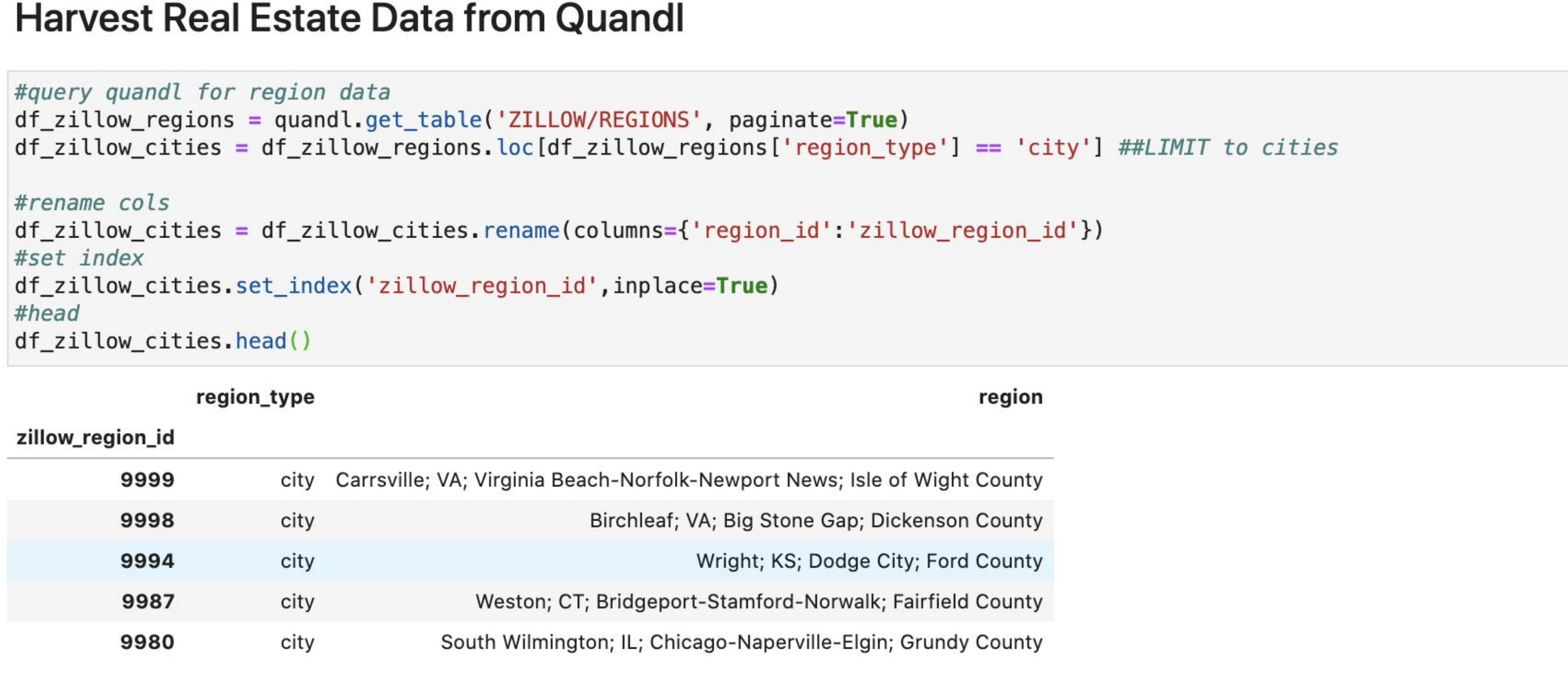
```
df_crosswalk = pd.read_csv('resources/CountyCrossWalk_Zillow_1.csv', dtype={'FIPS': object, 'StateFIPS':object, 'CountyFIPS':object})
df_crosswalk['StateFIPS'] = df_crosswalk['StateFIPS'].apply(lambda x: x.zfill(2))  #fill with leading zeros
df_crosswalk['CountyFIPS'] = df_crosswalk['CountyFIPS'].apply(lambda x: x.zfill(3)) #fill with leading zeros
df_crosswalk['FIPS'] = df_crosswalk['StateFIPS'] + df_crosswalk['CountyFIPS']
```



7]:

	CountyName	StateName	StateFIPS	CountyFIPS	MetroName_Zillow	CBSAName	CountyRegionID_Zillow	MetroRegionID_Zillow	FIPS
cbsa_code									
38060	Maricopa	Arizona	04	013	Phoenix, AZ	Phoenix-Mesa-Scottsdale, AZ	2402	394976	04013
38060	Pinal	Arizona	04	021	Phoenix, AZ	Phoenix-Mesa-Scottsdale, AZ	685	394976	04021

Grab All of Zillows Cities via Quandl



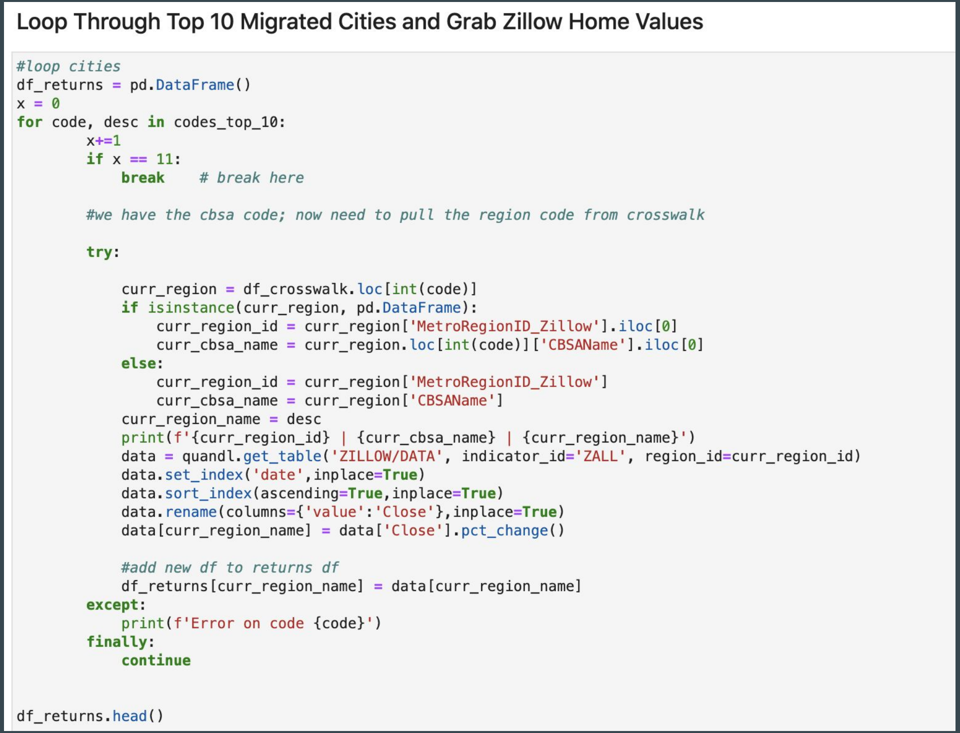
Harvest Real Estate Data from Quandl

```python
#query quandl for region data
df_zillow_regions = quandl.get_table('ZILLOW/REGIONS', paginate=True)
df_zillow_cities = df_zillow_regions.loc[df_zillow_regions['region_type'] == 'city'] ##LIMIT to cities

#rename cols
df_zillow_cities = df_zillow_cities.rename(columns={'region_id':'zillow_region_id'})
#set index
df_zillow_cities.set_index('zillow_region_id',inplace=True)
#head
df_zillow_cities.head()
```

	region_type	region
zillow_region_id		
9999	city	Carrsville; VA; Virginia Beach-Norfolk-Newport News; Isle of Wight County
9998	city	Birchleaf; VA; Big Stone Gap; Dickenson County
9994	city	Wright; KS; Dodge City; Ford County
9987	city	Weston; CT; Bridgeport-Stamford-Norwalk; Fairfield County
9980	city	South Wilmington; IL; Chicago-Naperville-Elgin; Grundy County

Grab Zillow Home Values by MSA (Top Migrated Cities) from Quandl

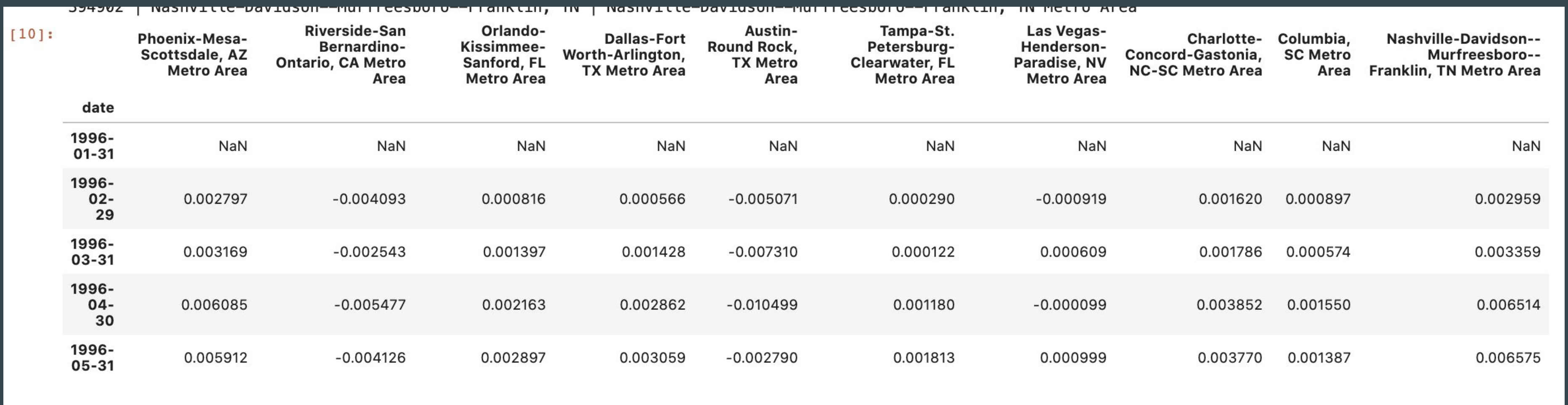
1. Loop through the Top 10 Migrated Cities
2. Match MSA Code to Zillow Region Id via Crosswalk
3. Call Quandl API for Zillow Values and convert to pct_change to mimic returns
4. Add the MSA to Returns Dataframe



Loop Through Top 10 Migrated Cities and Grab Zillow Home Values

```
#loop cities
df_returns = pd.DataFrame()
x = 0
for code, desc in codes_top_10:
        x+=1
        if x == 11:
            break    # break here

        #we have the cbsa code; now need to pull the region code from crosswalk

        try:

            curr_region = df_crosswalk.loc[int(code)]
            if isinstance(curr_region, pd.DataFrame):
                curr_region_id = curr_region['MetroRegionID_Zillow'].iloc[0]
                curr_cbsa_name = curr_region.loc[int(code)]['CBSAName'].iloc[0]
            else:
                curr_region_id = curr_region['MetroRegionID_Zillow']
                curr_cbsa_name = curr_region['CBSAName']
            curr_region_name = desc
            print(f'{curr_region_id} | {curr_cbsa_name} | {curr_region_name}')
            data = quandl.get_table('ZILLOW/DATA', indicator_id='ZALL', region_id=curr_region_id)
            data.set_index('date',inplace=True)
            data.sort_index(ascending=True,inplace=True)
            data.rename(columns={'value':'Close'},inplace=True)
            data[curr_region_name] = data['Close'].pct_change()

            #add new df to returns df
            df_returns[curr_region_name] = data[curr_region_name]
        except:
            print(f'Error on code {code}')
        finally:
            continue


df_returns.head()
```

Create Dataframe showing Home Value Returns by Month/MSA (Top Migrated Cities)

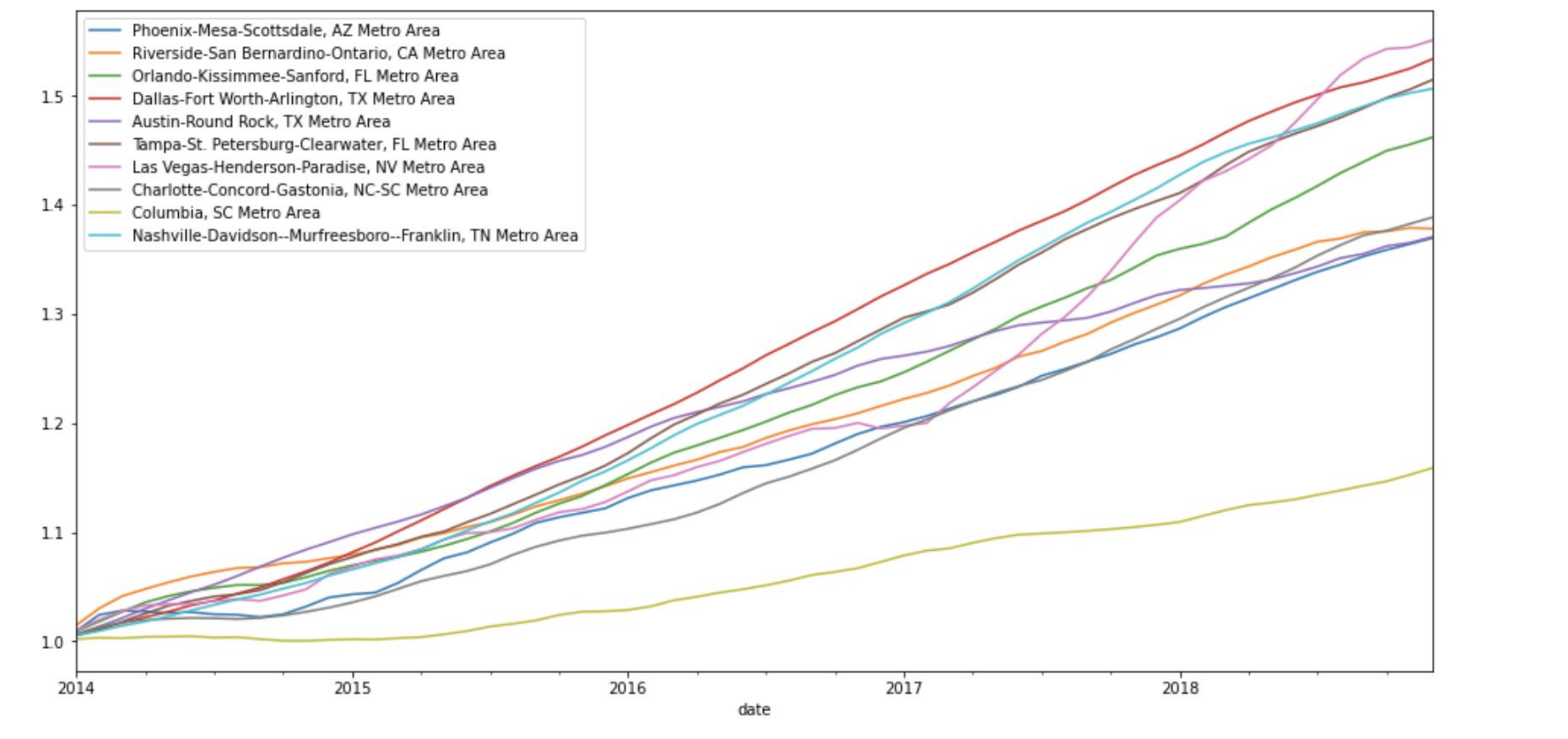


```
394902 | Nashville-Davidson--Murfreesboro--Franklin, TN | Nashville-Davidson--Murfreesboro--Franklin, TN Metro Area
```

[10]:

date	Phoenix-Mesa-Scottsdale, AZ Metro Area	Riverside-San Bernardino-Ontario, CA Metro Area	Orlando-Kissimmee-Sanford, FL Metro Area	Dallas-Fort Worth-Arlington, TX Metro Area	Austin-Round Rock, TX Metro Area	Tampa-St. Petersburg-Clearwater, FL Metro Area	Las Vegas-Henderson-Paradise, NV Metro Area	Charlotte-Concord-Gastonia, NC-SC Metro Area	Columbia, SC Metro Area	Nashville-Davidson--Murfreesboro--Franklin, TN Metro Area
1996-01-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1996-02-29	0.002797	-0.004093	0.000816	0.000566	-0.005071	0.000290	-0.000919	0.001620	0.000897	0.002959
1996-03-31	0.003169	-0.002543	0.001397	0.001428	-0.007310	0.000122	0.000609	0.001786	0.000574	0.003359
1996-04-30	0.006085	-0.005477	0.002163	0.002862	-0.010499	0.001180	-0.000099	0.003852	0.001550	0.006514
1996-05-31	0.005912	-0.004126	0.002897	0.003059	-0.002790	0.001813	0.000999	0.003770	0.001387	0.006575

Show Returns by Top Migrated MSA

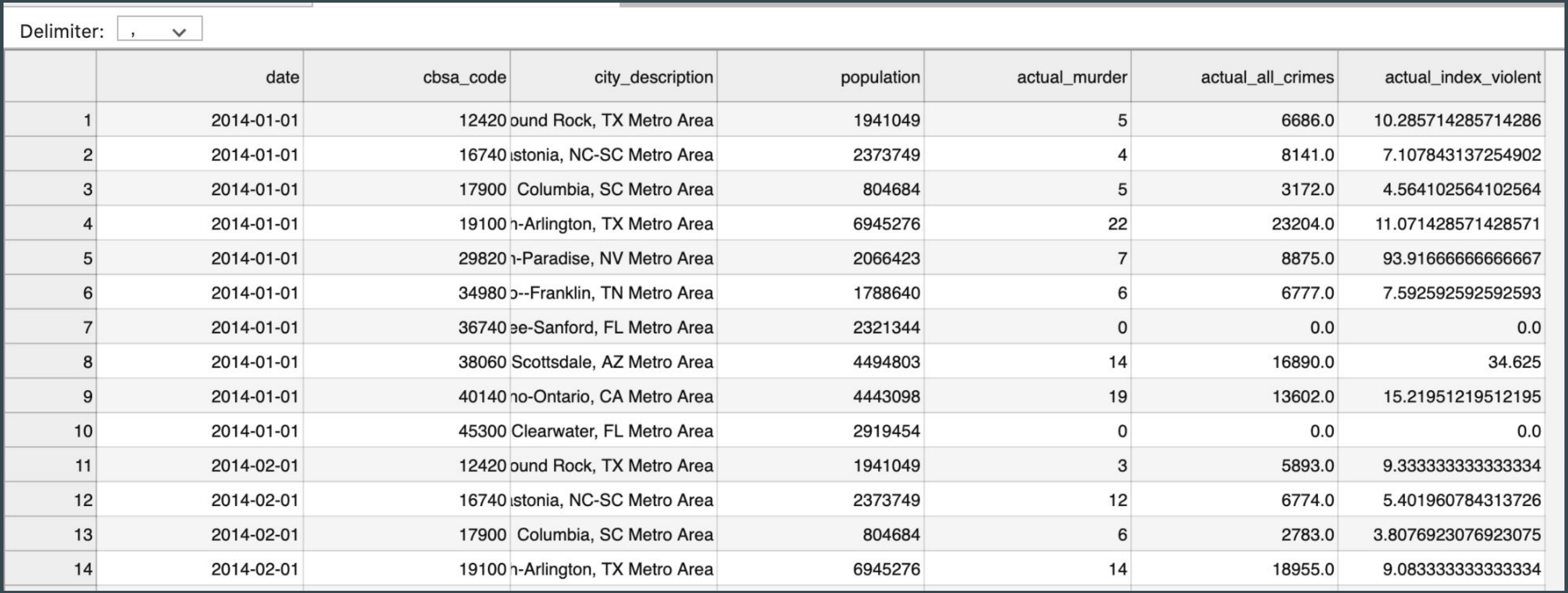


Findings

- Was challenging to locate additional Time Series Data by Year/Month and MSA to properly run Multiple Linear Regressions against Home Values

    - Education
    - Income
    - Crime
    - Environmental Factors
- Learning Curve and Lucky Find to Match MSA Codes with Zillow Codes via Crosswalk
- We limited our Returns to Top 10 Cities to filter results.
- We wish all Data was Standardized and accessible like Stock Tickers

Crime Data

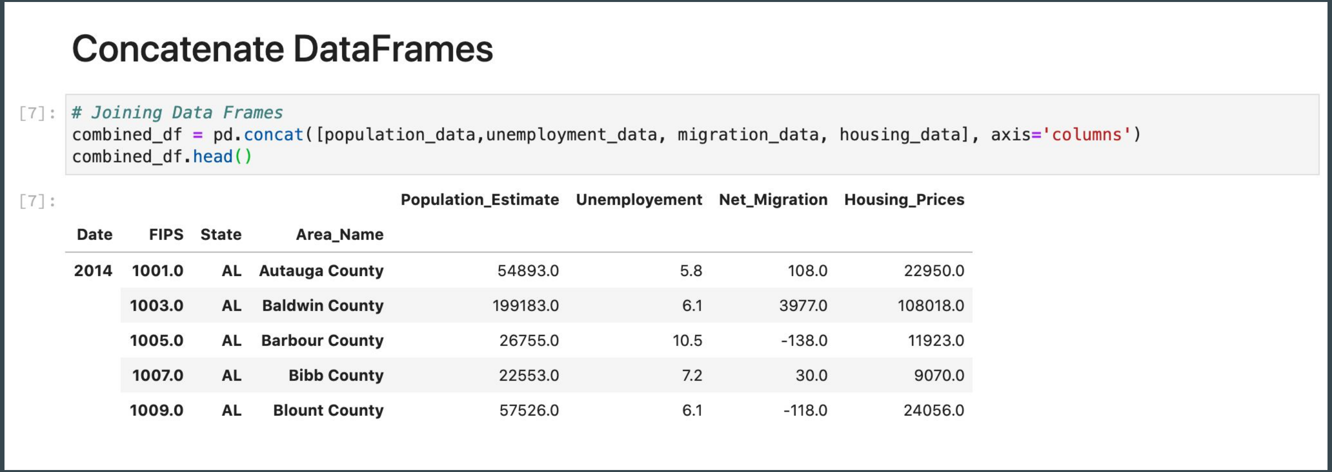


Delimiter: ,

	date	cbsa_code	city_description	population	actual_murder	actual_all_crimes	actual_index_violent
1	2014-01-01	12420	ound Rock, TX Metro Area	1941049	5	6686.0	10.285714285714286
2	2014-01-01	16740	ıstonia, NC-SC Metro Area	2373749	4	8141.0	7.107843137254902
3	2014-01-01	17900	Columbia, SC Metro Area	804684	5	3172.0	4.564102564102564
4	2014-01-01	19100	n-Arlington, TX Metro Area	6945276	22	23204.0	11.071428571428571
5	2014-01-01	29820	n-Paradise, NV Metro Area	2066423	7	8875.0	93.91666666666667
6	2014-01-01	34980	o--Franklin, TN Metro Area	1788640	6	6777.0	7.592592592592593
7	2014-01-01	36740	ee-Sanford, FL Metro Area	2321344	0	0.0	0.0
8	2014-01-01	38060	Scottsdale, AZ Metro Area	4494803	14	16890.0	34.625
9	2014-01-01	40140	no-Ontario, CA Metro Area	4443098	19	13602.0	15.21951219512195
10	2014-01-01	45300	Clearwater, FL Metro Area	2919454	0	0.0	0.0
11	2014-02-01	12420	ound Rock, TX Metro Area	1941049	3	5893.0	9.333333333333334
12	2014-02-01	16740	ıstonia, NC-SC Metro Area	2373749	12	6774.0	5.401960784313726
13	2014-02-01	17900	Columbia, SC Metro Area	804684	6	2783.0	3.8076923076923075
14	2014-02-01	19100	n-Arlington, TX Metro Area	6945276	14	18955.0	9.083333333333334

GRAPHICAL ANALYSIS

Joined DataFrame



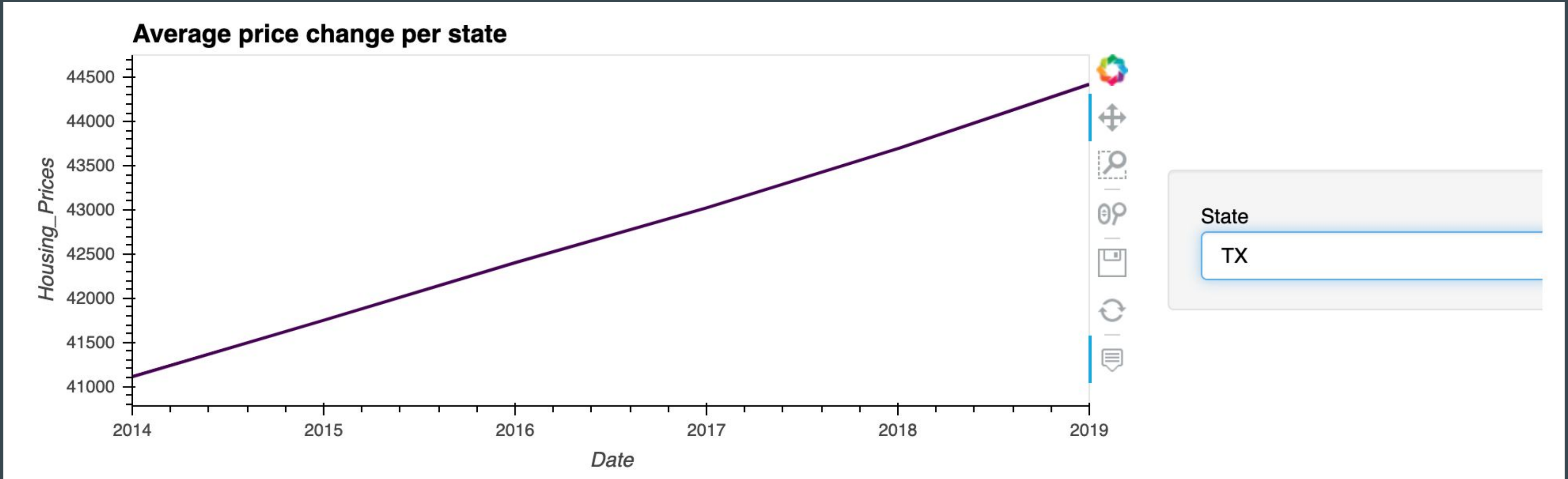
Concatenate DataFrames

```
[7]: # Joining Data Frames
     combined_df = pd.concat([population_data,unemployment_data, migration_data, housing_data], axis='columns')
     combined_df.head()
```

```
[7]:                                    Population_Estimate  Unemployement  Net_Migration  Housing_Prices
     Date  FIPS    State  Area_Name
     2014  1001.0  AL     Autauga County           54893.0            5.8          108.0         22950.0
           1003.0  AL     Baldwin County          199183.0            6.1         3977.0        108018.0
           1005.0  AL     Barbour County           26755.0           10.5         -138.0         11923.0
           1007.0  AL        Bibb County           22553.0            7.2           30.0          9070.0
           1009.0  AL      Blount County           57526.0            6.1         -118.0         24056.0
```

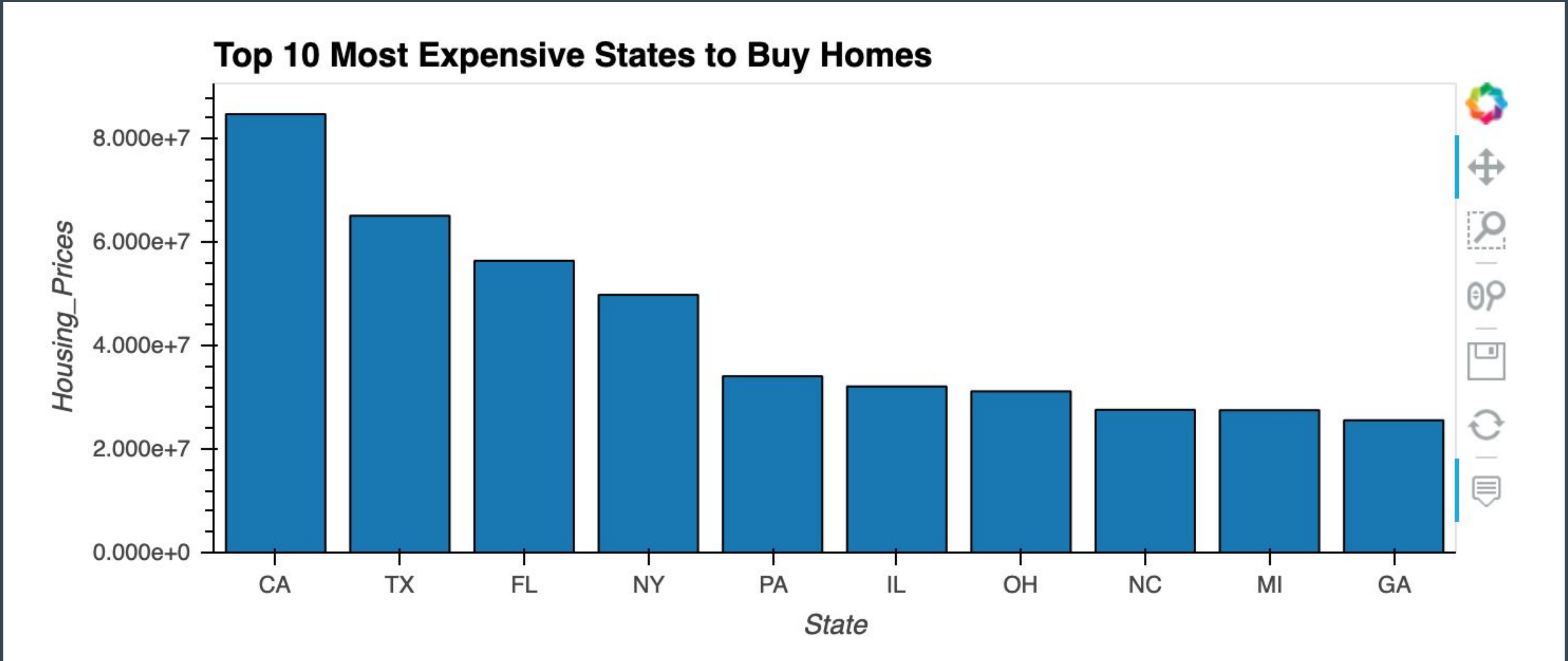
DataFrames were joined based on the FIPS and Date to get yearly data on all the counties in the USA.

Economic Research Service: United States Department of Agriculture

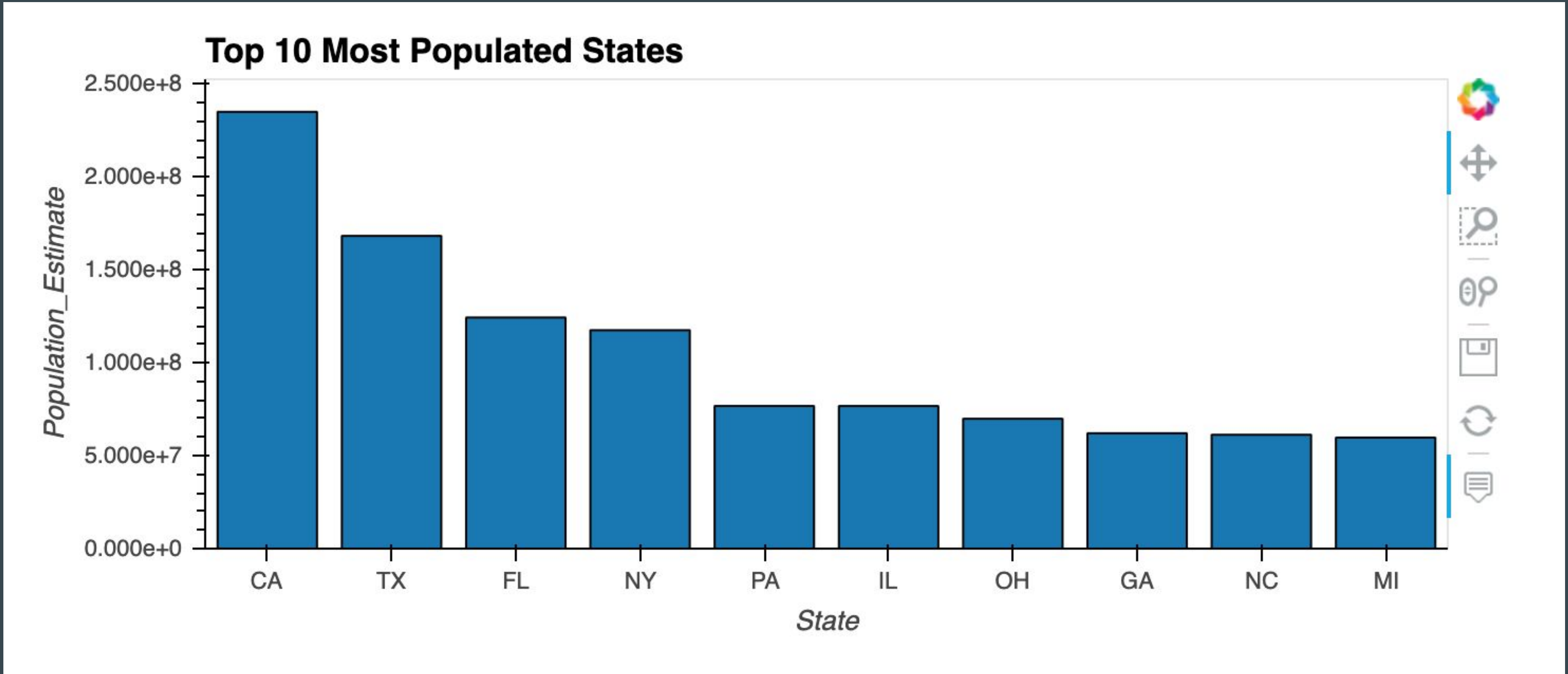

Average price change per state
44500
44000
43500
43000
42500
42000
41500
41000
Housing_Prices
2014
2015
2016
2017
2018
2019
Date

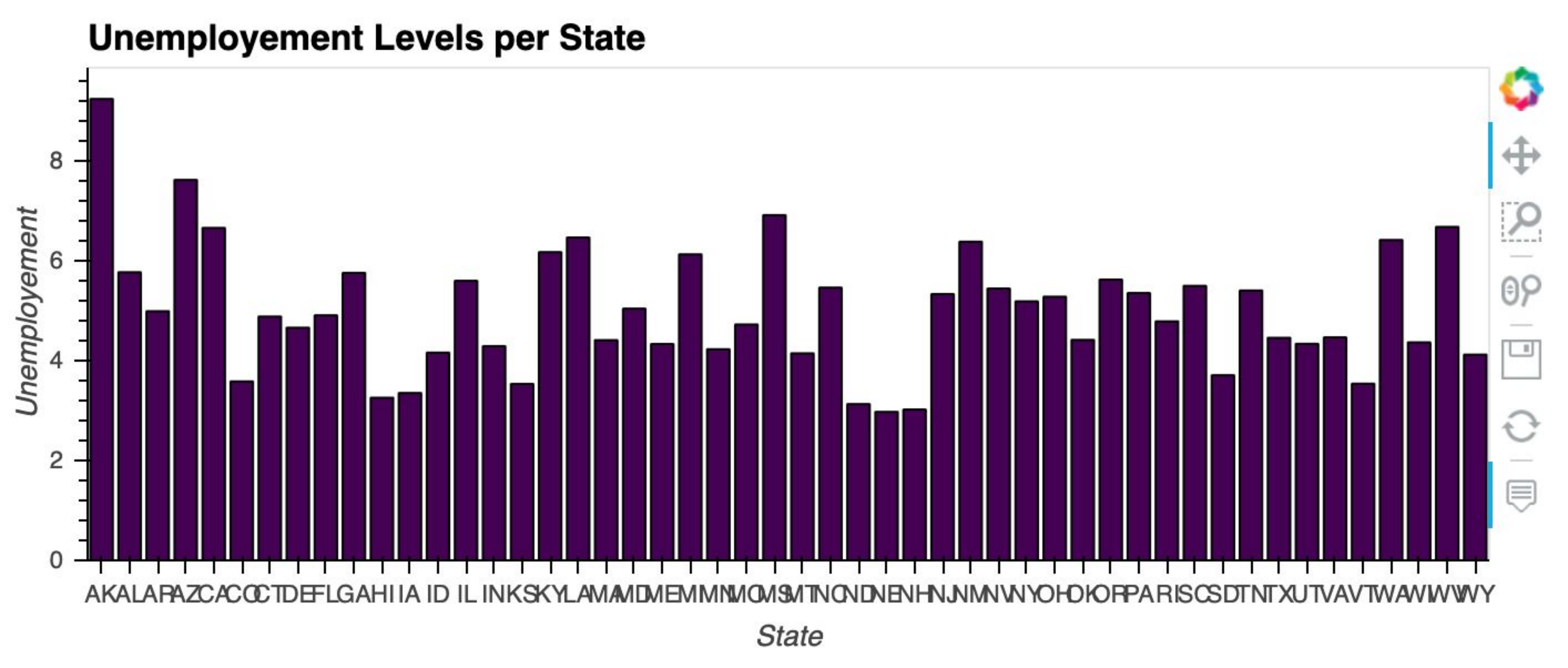
State

TX



Top 10 Most Expensive States to Buy Homes


Top 10 Most Populated States
2.500e+8
2.000e+8
1.500e+8
1.000e+8
5.000e+7
0.000e+0
Population_Estimate
CA
TX
FL
NY
PA
IL
OH
GA
NC
MI
State

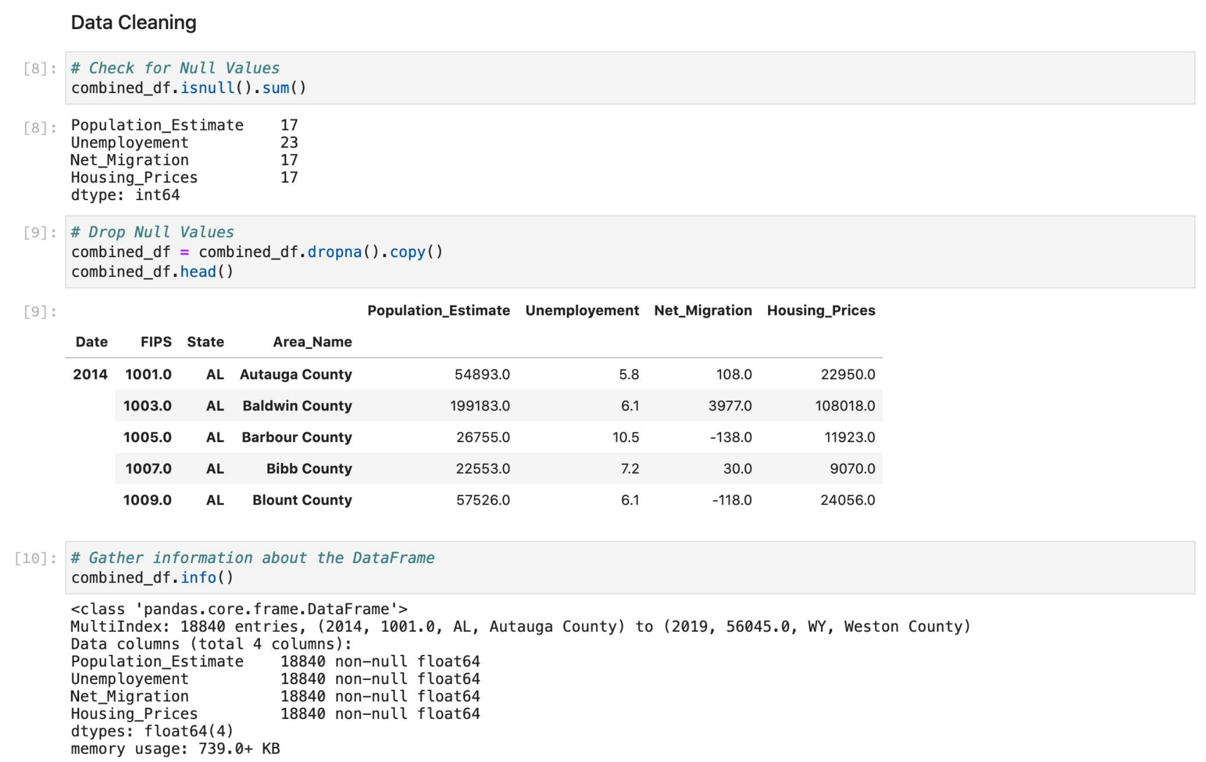

Unemployement Levels per State
Unemployement
0
2
4
6
8
State
AK AL AR AZ CA CO CT DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC ND NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY

MULTIPLE LINEAR REGRESSION

The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

Step by Step Process

Step 1: Data Cleaning



Data Cleaning

[8]:
```
# Check for Null Values
combined_df.isnull().sum()
```

[8]:
```
Population_Estimate    17
Unemployement          23
Net_Migration          17
Housing_Prices         17
dtype: int64
```

[9]:
```
# Drop Null Values
combined_df = combined_df.dropna().copy()
combined_df.head()
```

[9]:

				Population_Estimate	Unemployement	Net_Migration	Housing_Prices
Date	FIPS	State	Area_Name				
2014	1001.0	AL	Autauga County	54893.0	5.8	108.0	22950.0
	1003.0	AL	Baldwin County	199183.0	6.1	3977.0	108018.0
	1005.0	AL	Barbour County	26755.0	10.5	-138.0	11923.0
	1007.0	AL	Bibb County	22553.0	7.2	30.0	9070.0
	1009.0	AL	Blount County	57526.0	6.1	-118.0	24056.0

[10]:
```
# Gather information about the DataFrame
combined_df.info()
```

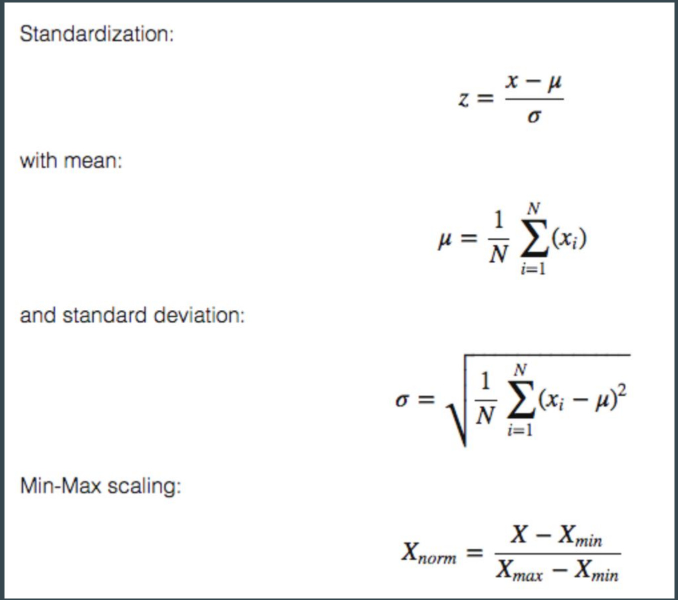
```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 18840 entries, (2014, 1001.0, AL, Autauga County) to (2019, 56045.0, WY, Weston County)
Data columns (total 4 columns):
Population_Estimate    18840 non-null float64
Unemployement          18840 non-null float64
Net_Migration          18840 non-null float64
Housing_Prices         18840 non-null float64
dtypes: float64(4)
memory usage: 739.0+ KB
```

Usual data cleaning process:

1. Check for null
2. Drop nulls
3. Final information shows we have 18840 non-null data points

Step 2: Data Scaling

To make the data comparable, MinMax Scalar was used.



Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

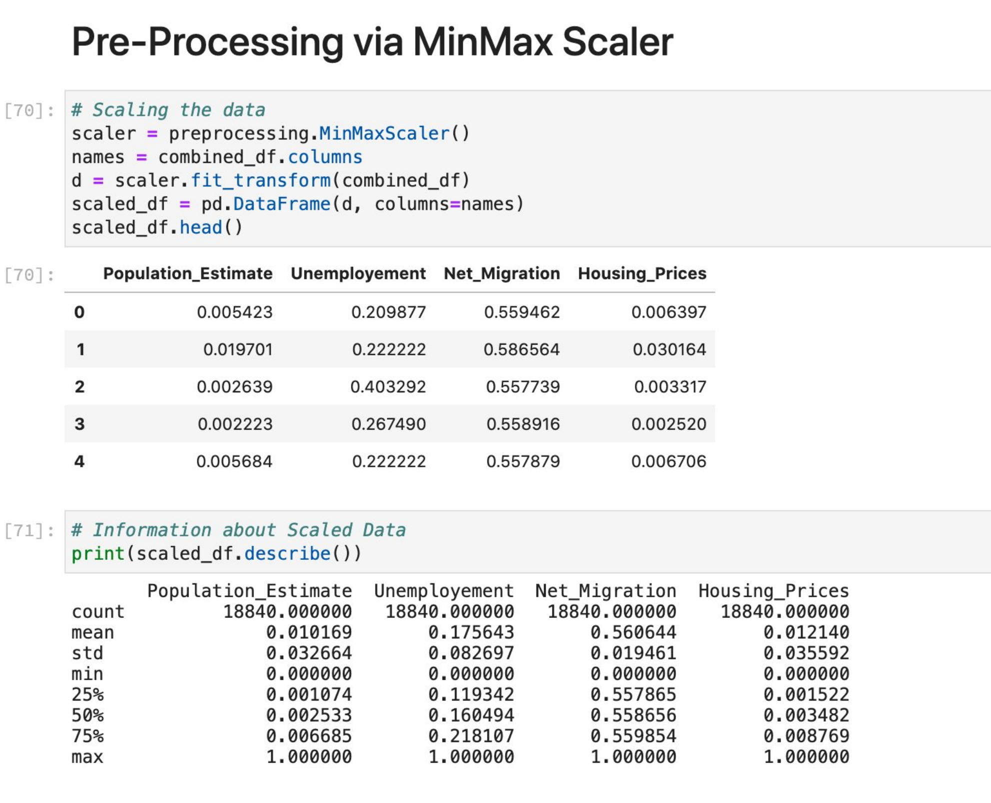
$$\mu = \frac{1}{N} \sum_{i=1}^{N} (x_i)$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}$$

Min-Max scaling:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



Pre-Processing via MinMax Scaler

[70]:

```
# Scaling the data
scaler = preprocessing.MinMaxScaler()
names = combined_df.columns
d = scaler.fit_transform(combined_df)
scaled_df = pd.DataFrame(d, columns=names)
scaled_df.head()
```

[70]:

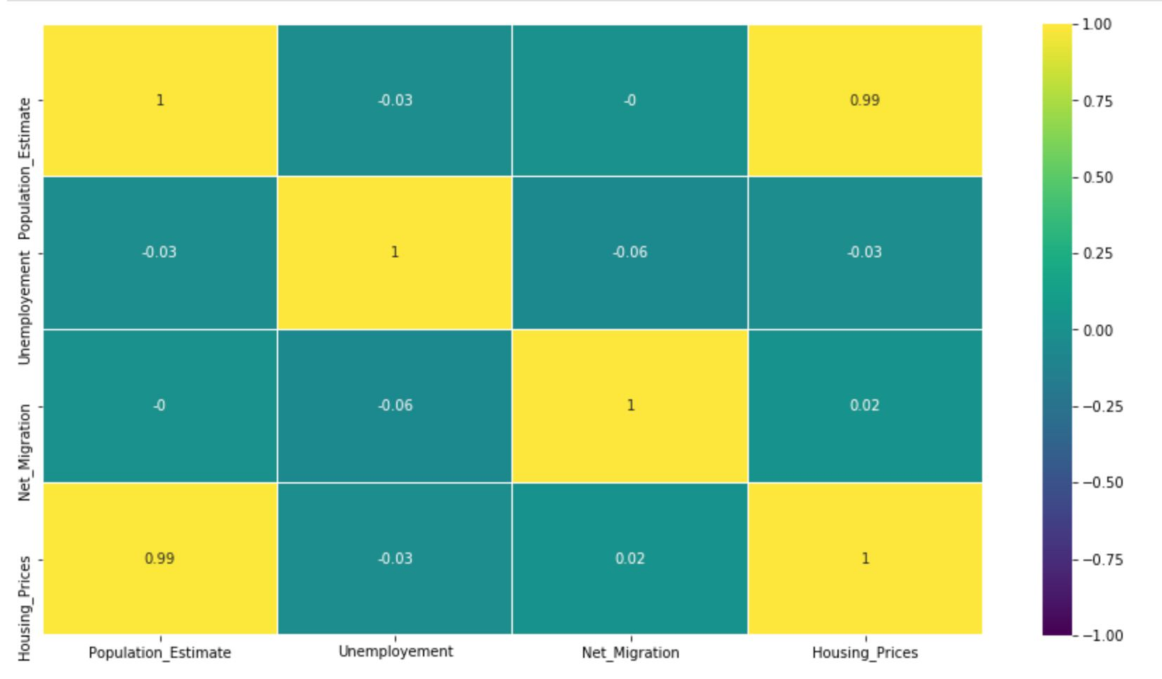
	Population_Estimate	Unemployement	Net_Migration	Housing_Prices
0	0.005423	0.209877	0.559462	0.006397
1	0.019701	0.222222	0.586564	0.030164
2	0.002639	0.403292	0.557739	0.003317
3	0.002223	0.267490	0.558916	0.002520
4	0.005684	0.222222	0.557879	0.006706

[71]:

```
# Information about Scaled Data
print(scaled_df.describe())
```

```
       Population_Estimate  Unemployement  Net_Migration  Housing_Prices
count         18840.000000   18840.000000   18840.000000    18840.000000
mean              0.010169       0.175643       0.560644        0.012140
std               0.032664       0.082697       0.019461        0.035592
min               0.000000       0.000000       0.000000        0.000000
25%               0.001074       0.119342       0.557865        0.001522
50%               0.002533       0.160494       0.558656        0.003482
75%               0.006685       0.218107       0.559854        0.008769
max               1.000000       1.000000       1.000000        1.000000
```

Step 3: Correlation



	Population_Estimate	Unemployement	Net_Migration	Housing_Prices
Population_Estimate	1	-0.03	-0	0.99
Unemployement	-0.03	1	-0.06	-0.03
Net_Migration	-0	-0.06	1	0.02
Housing_Prices	0.99	-0.03	0.02	1

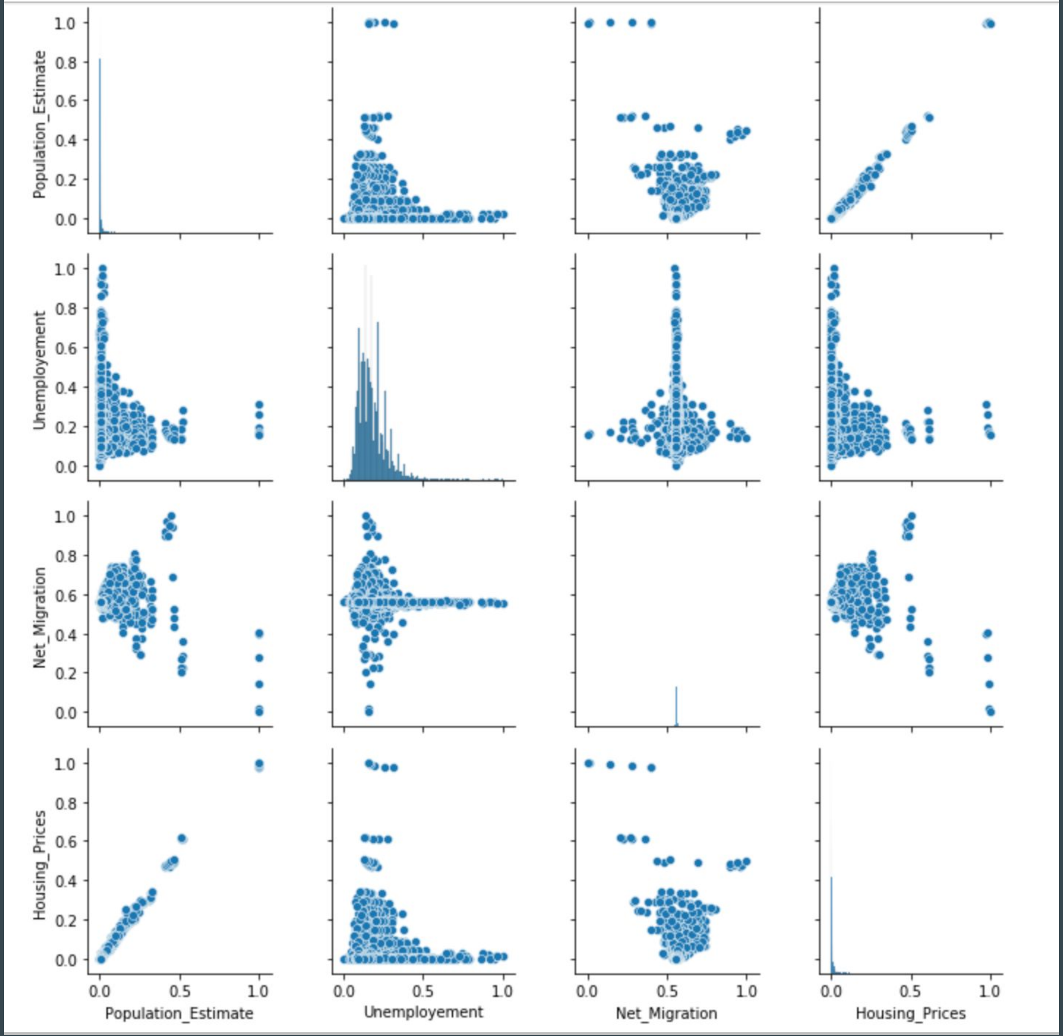
Observations:

1. Great correlation between Housing Prices and Population
2. Not that great correlation between variables: No heteroskedasticity
3. But also not that great correlation between housing prices and other variables. So is this even valid?

Step 3: Correlation

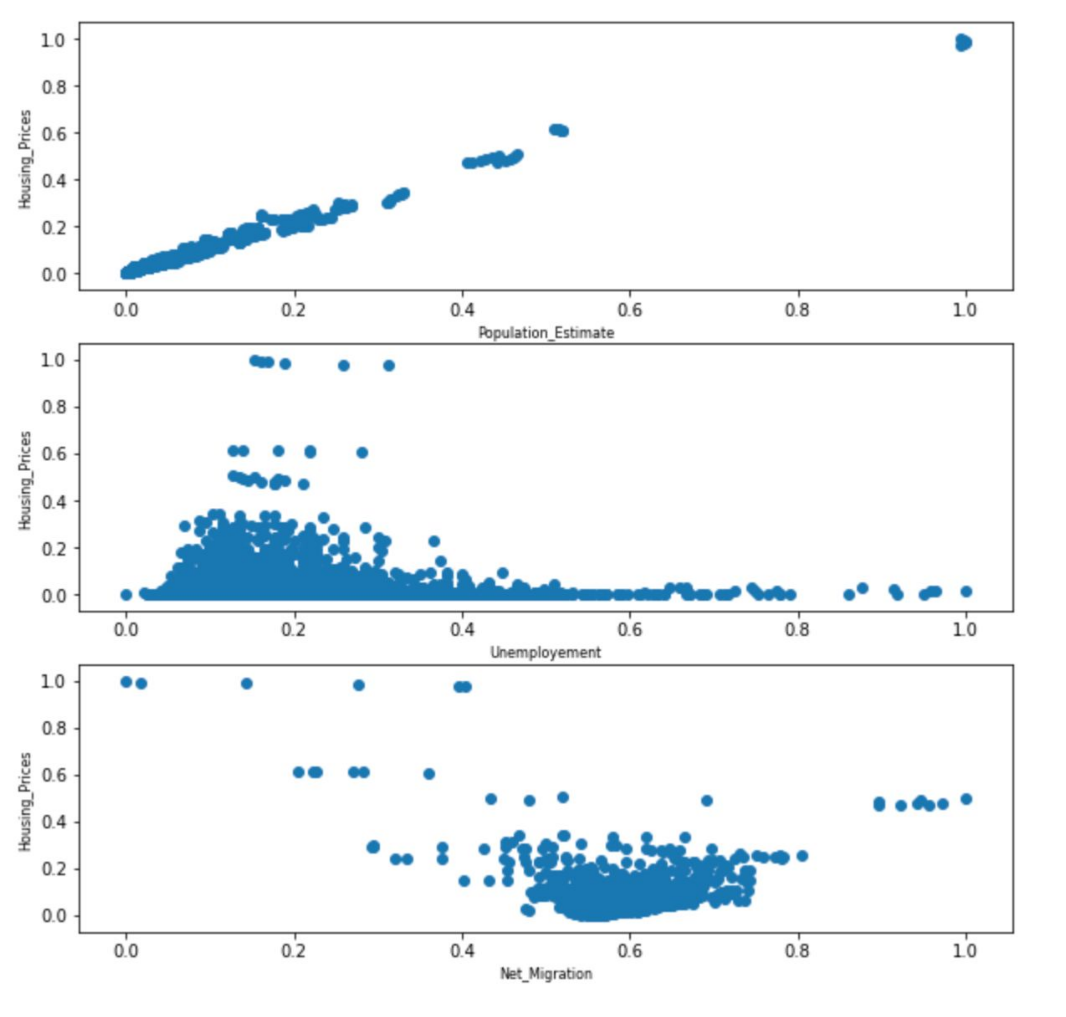
A pairplot just to show the relationships and distributions of each variable.

Kind of like a spoiler for what we will see in our regression results



Step 3: Correlation

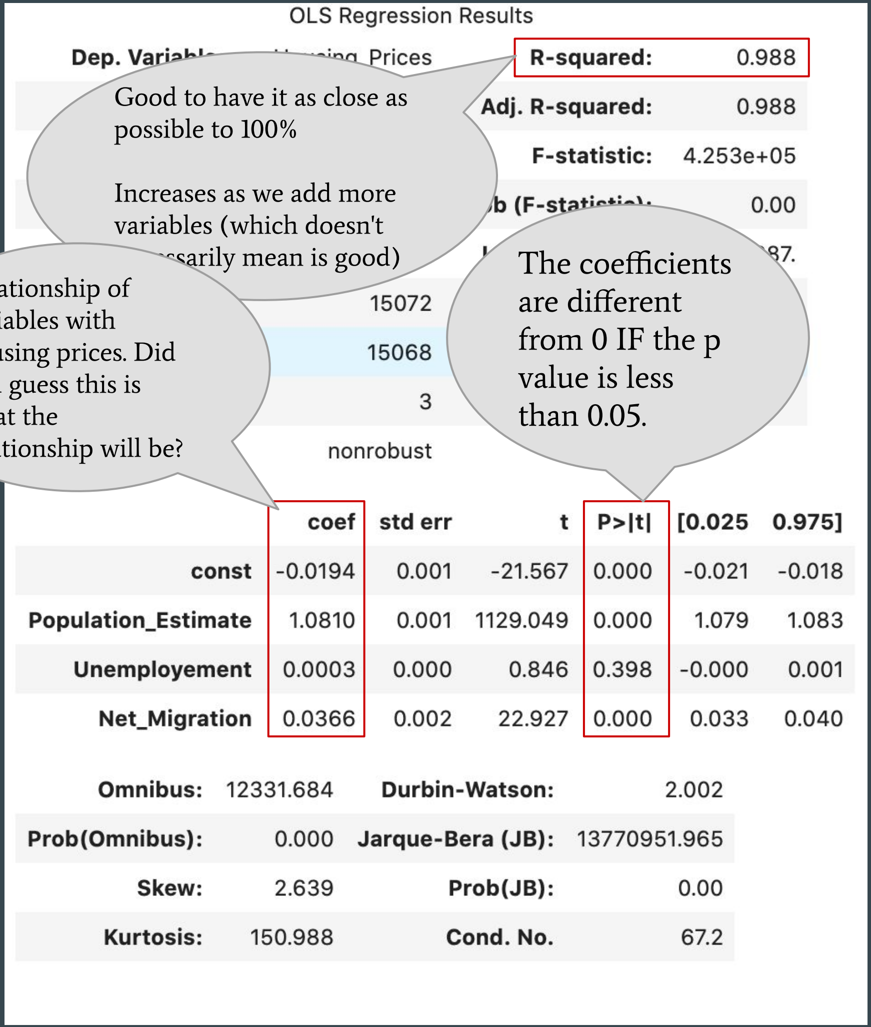
To better understand the relationship between target variable (Housing Prices) and predictor variables (Population, Unemployment and Migration).



Step 4: Regression

1. Data was partitioned into training and test data (20% of the data) sets.
2. Training data was regressed on the y-variable (Housing Prices).
3. Relationships + Significance
4. The R^2 value shows that 98% of the error is accounted for in the model.

Does this mean it's a great model?



OLS Regression Results

Dep. Variabl	Prices	R-squared:	0.988
		Adj. R-squared:	0.988
		F-statistic:	4.253e+05
		b (F-statistic):	0.00
			87.
	15072		
	15068		
	3		
	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0194	0.001	-21.567	0.000	-0.021	-0.018
Population_Estimate	1.0810	0.001	1129.049	0.000	1.079	1.083
Unemployement	0.0003	0.000	0.846	0.398	-0.000	0.001
Net_Migration	0.0366	0.002	22.927	0.000	0.033	0.040

Omnibus:	12331.684	Durbin-Watson:	2.002
Prob(Omnibus):	0.000	Jarque-Bera (JB):	13770951.965
Skew:	2.639	Prob(JB):	0.00
Kurtosis:	150.988	Cond. No.	67.2

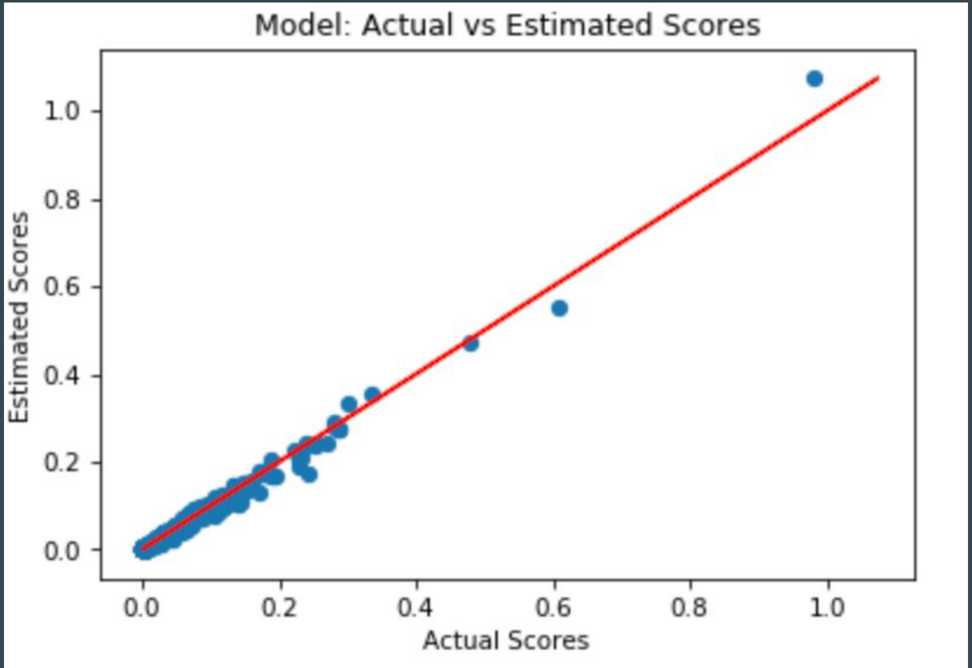
Good to have it as close as possible to 100%

Increases as we add more variables (which doesn't ssarily mean is good)

Relationship of variables with housing prices. Did you guess this is what the relationship will be?

The coefficients are different from 0 IF the p value is less than 0.05.

Step 5: Fitting the model

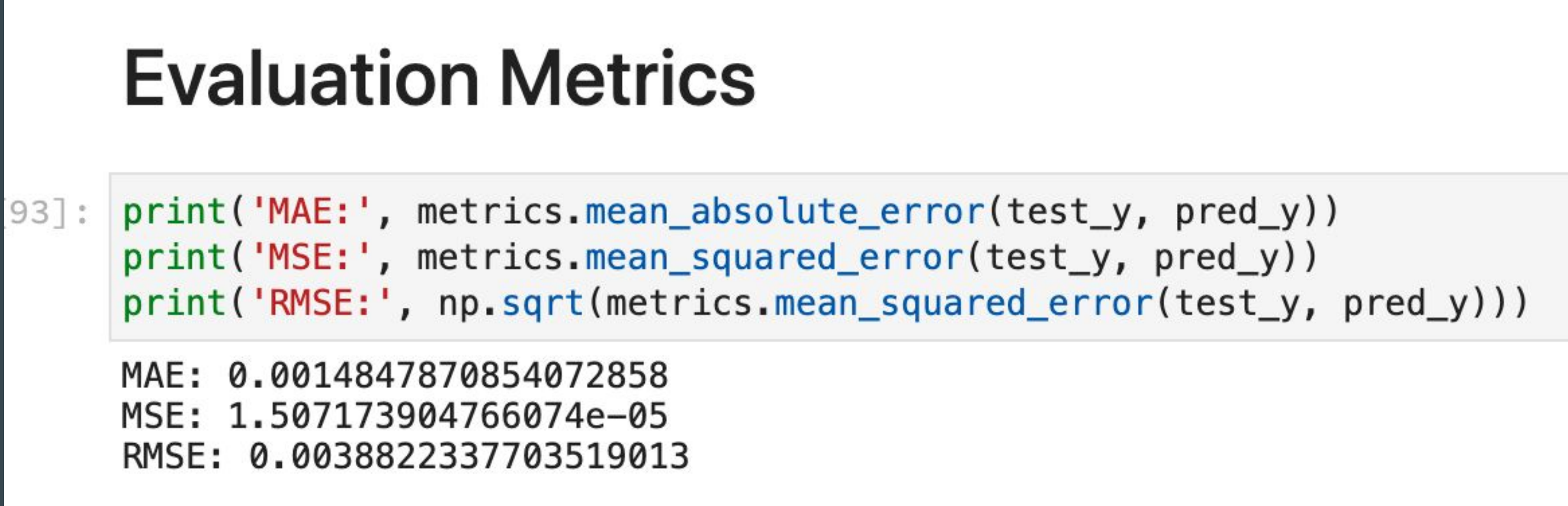


1. Red line shows the predicted value of y based on our regression model

AND

2. The scatter points show test_y along with predicted_y based on the test_x.
3. Fits very closely as indicated by our R^2 value (the fit of the model is great)

Step 6: Evaluation



Evaluation Metrics

```
[93]: print('MAE:', metrics.mean_absolute_error(test_y, pred_y))
      print('MSE:', metrics.mean_squared_error(test_y, pred_y))
      print('RMSE:', np.sqrt(metrics.mean_squared_error(test_y, pred_y)))
```

```
MAE: 0.0014847870854072858
MSE: 1.507173904766074e-05
RMSE: 0.0038822337703519013
```

MSE shows that the model is 98.5% accurate. But because of the low statistical significance of our coefficients, this model, in isolation cannot be used for much prediction.

Implications and Evaluation

1. DATA, DATA, DATA
2. Challenges: find appropriate data for each variable and be able to join them to make one useful data set that can be used for regression and prediction
3. Further work: More variables which are better correlated with the target variable could yield a model that can be used for prediction
4. Other models such as Lasso/ Logistic model that allow for time series analysis and autoregressive models could maybe fit the data better.