**Modeling single-family home prices in the Bay Area: (subtitle)**

****

*Photo credit: Unsplash (https://unsplash.com/photos/6b9rqGI\_w1s)*

**Abstract**

This article describes the collection, visualization, and modeling of single-family home prices currently listed across the San Francisco Bay Area. Complementing listing information (number of bedrooms and bathrooms, home size, and lot size) with location data (school quality and commute times) was found to significantly improve explanatory power of the model, and enabled the identification of undervalued listings and neighborhoods.

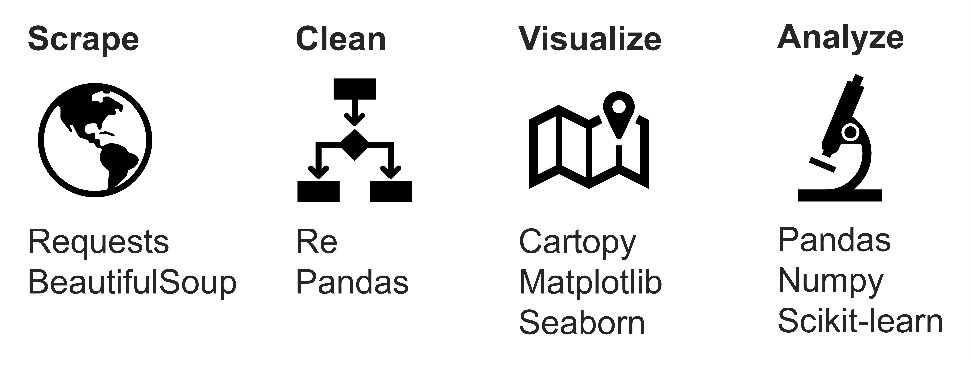
**Introduction**

While the performance of the S&P 500 and the overall US housing market has been nearly identical since 2000 (both up ~100%), home price indices in the San Francisco Bay Area have risen by approximately 167% (St. Louis Fed). As such, Bay Area homeowners have enjoyed an opportunity to build wealth through real estate in a way that is not accessible to most of the rest of the country.

For those already bought into the market, this near-tripling of real estate values since 2000 has undoubtedly been a good thing. However, for those newly relocated to the region, saving towards a down payment and choosing where to buy can be a daunting task. Inspired by discussions I’ve had with friends and family, basic concepts in investing (i.e., buy undervalued assets), and a desire to hone my data science skillset, I set out to gather as much information about current prices of single-family homes in the Bay Area and apply machine learning techniques to tease out the most important factors driving home values.

**Methods**

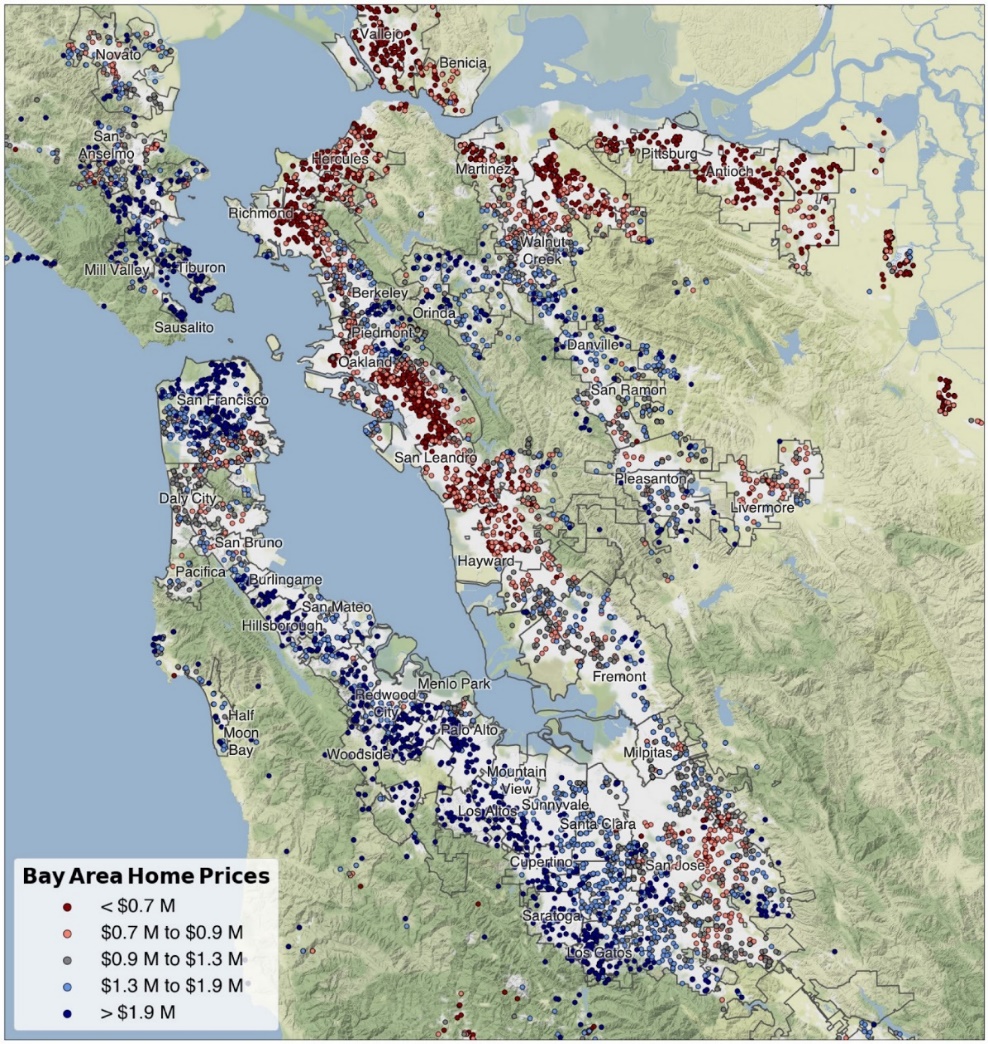
Single-family home listings (address, beds, baths, home size, lot size, latitude/longitude, and price) across the Bay Area were scraped in June 2019 from a real estate webpage (www.mlslistings.com) using the Requests and BeautifulSoup Python libraries, and cleaned and tabulated using Regex and Pandas. Complementing this listing data, commute times were obtained from Google Maps and school quality data pulled from the 2018 California Assessment of Student Performance and Progress (CAASPP). The data was plotted on top of maps using Cartopy, Matplotlib, shapefiles (town, zip code, and neighborhood borders) from Stanford Earthworks. Box/strip plots and pairwise relationships between variables were visualized using Seaborn. Ordinary least squares regression analysis was applied to the data using Statsmodels and scikit-learn packages. The full source code for this project is available on my personal GitHub page (github.com/mboles01).



**Results**

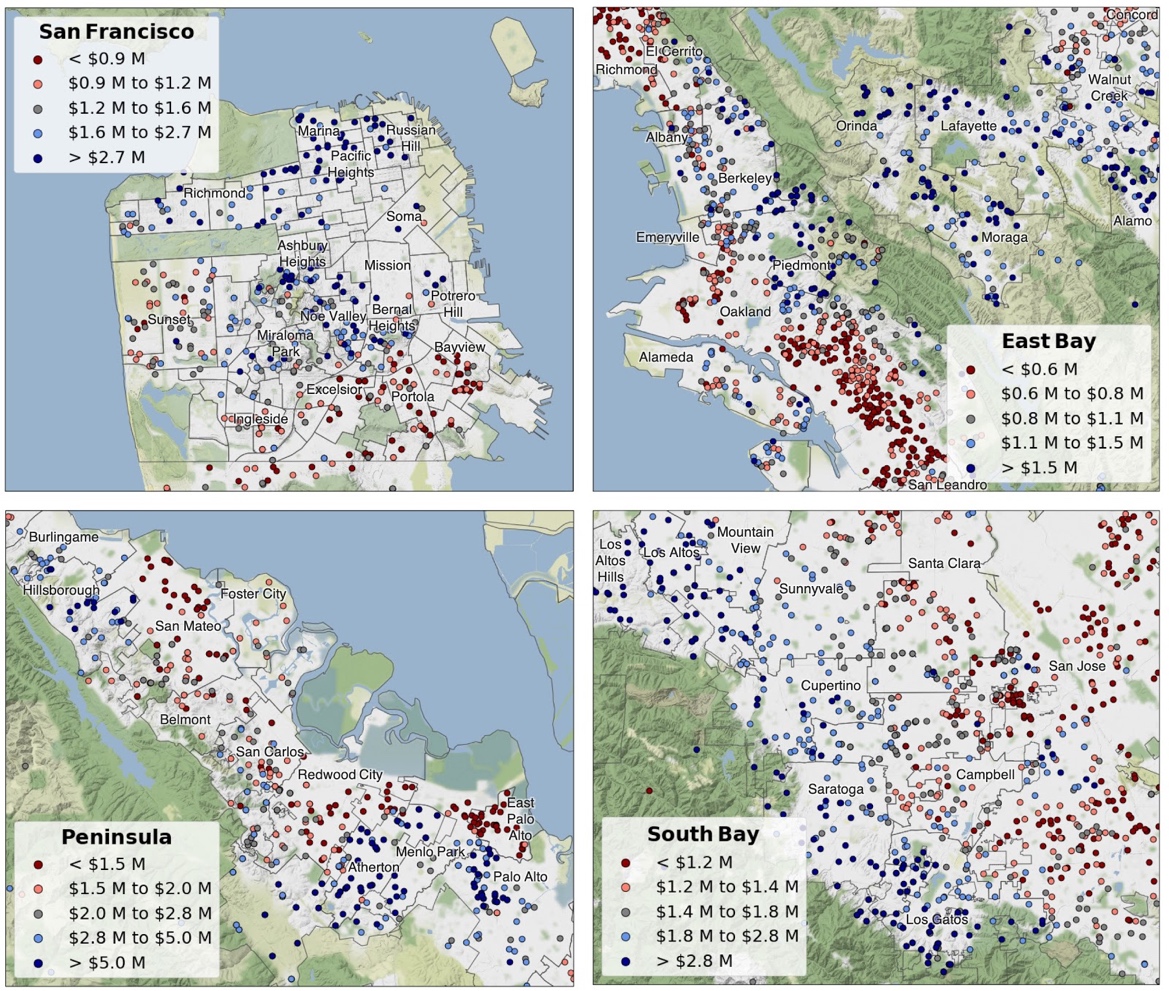
This study began with the collection of listing data for all single family homes on a popular real estate webpage (www.mlslistings.com). Looping over a list of Bay Area zip codes, location details (address, latitude and longitude coordinates), property characteristics (number of bedrooms, number of bathrooms, home size, and lot size), and list price were scraped for each listing using the Python Requests and BeautifulSoup libraries (see source code at end of article). Across 214 zip codes, location, property, and price information was collected for 7151 listings in June 2019. The data was cleaned using regular expressions (re.findall(), re.sub() functions) to remove extra whitespace, unwanted characters, and entries with missing values.

With information for several thousand properties currently on the market across the Bay Area, the location of all listings was plotted on a terrain map, color-coded by price, using the Python Cartopy package together with city border information obtained from Stanford Earthworks (Figure 1). This map makes it easy to compare the relative cost of homes across the region. Listings in San Francisco, Marin County, and the Peninsula are typically in the most expensive quintile (dark blue data points), while those in Oakland, San Leandro, and Richmond are typically in the least expensive quintile (dark red data points). Similarly, in the South Bay, homes closer to the Santa Cruz Mountains are typically more expensive than those tucked into the Diablo Range.



**Figure 1.** Overview of single-family homes listed for sale in the Bay Area in June 2019. The 7153 entries are split into quintiles by price, with list prices falling within the bottom and top 20% colored red and blue, respectively.

From this



**Figure 2.** Zoom showing detail of single-family home list prices in the San Francisco, East Bay, Peninsula, and South Bay regions. In each case, price quintiles have been recalculated to reflect the distribution of prices within the highlighted region.

**Results**

a. Maps describe home price trends across the region (Figures 1,2)

b. Box/strip plots enable ranking cities by cost of house, cost of land (Figure 3)

c. Plots of list price vs. listing data (beds, baths, home size, lot size) show weak positive correlations that narrow upon zooming into one zip code (Figure 4)

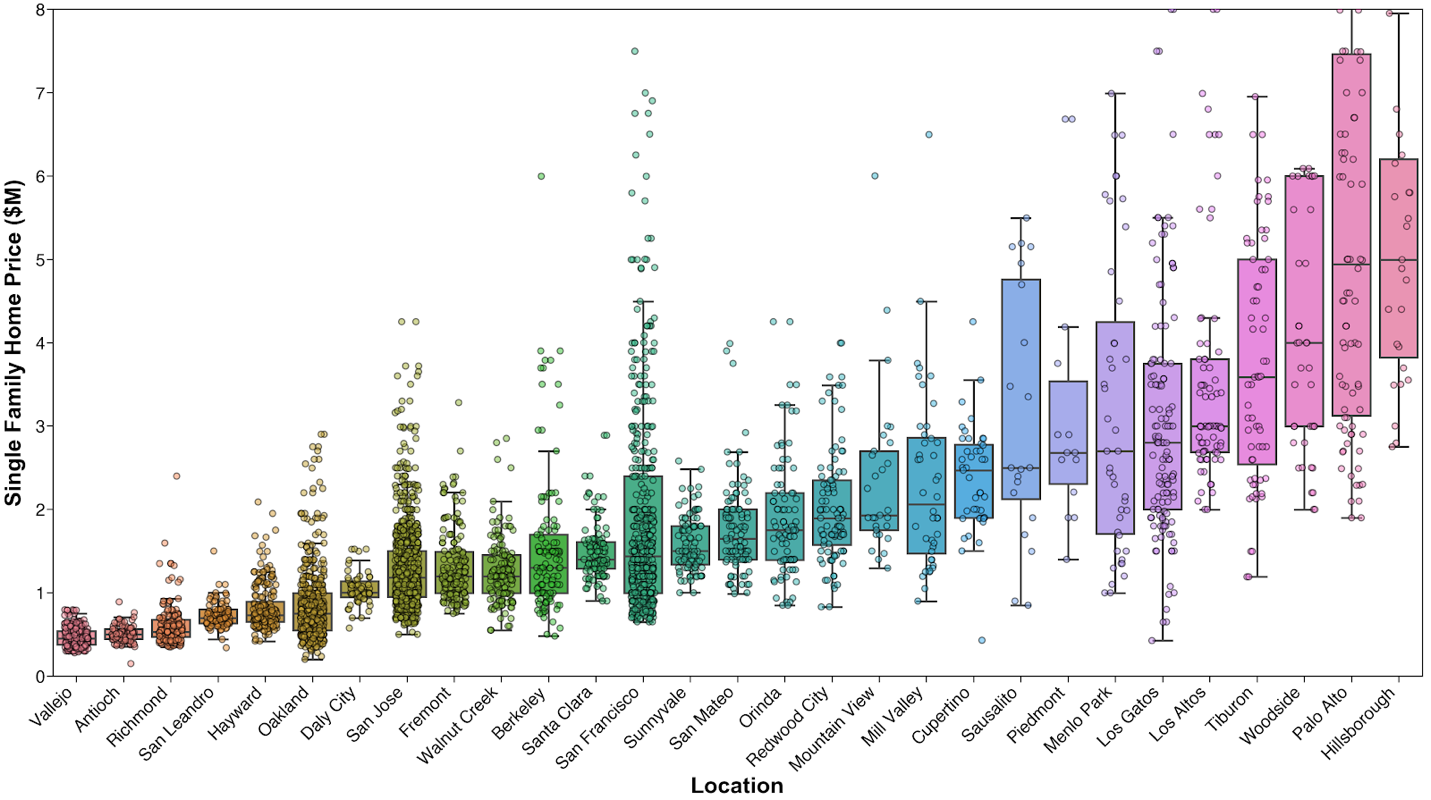
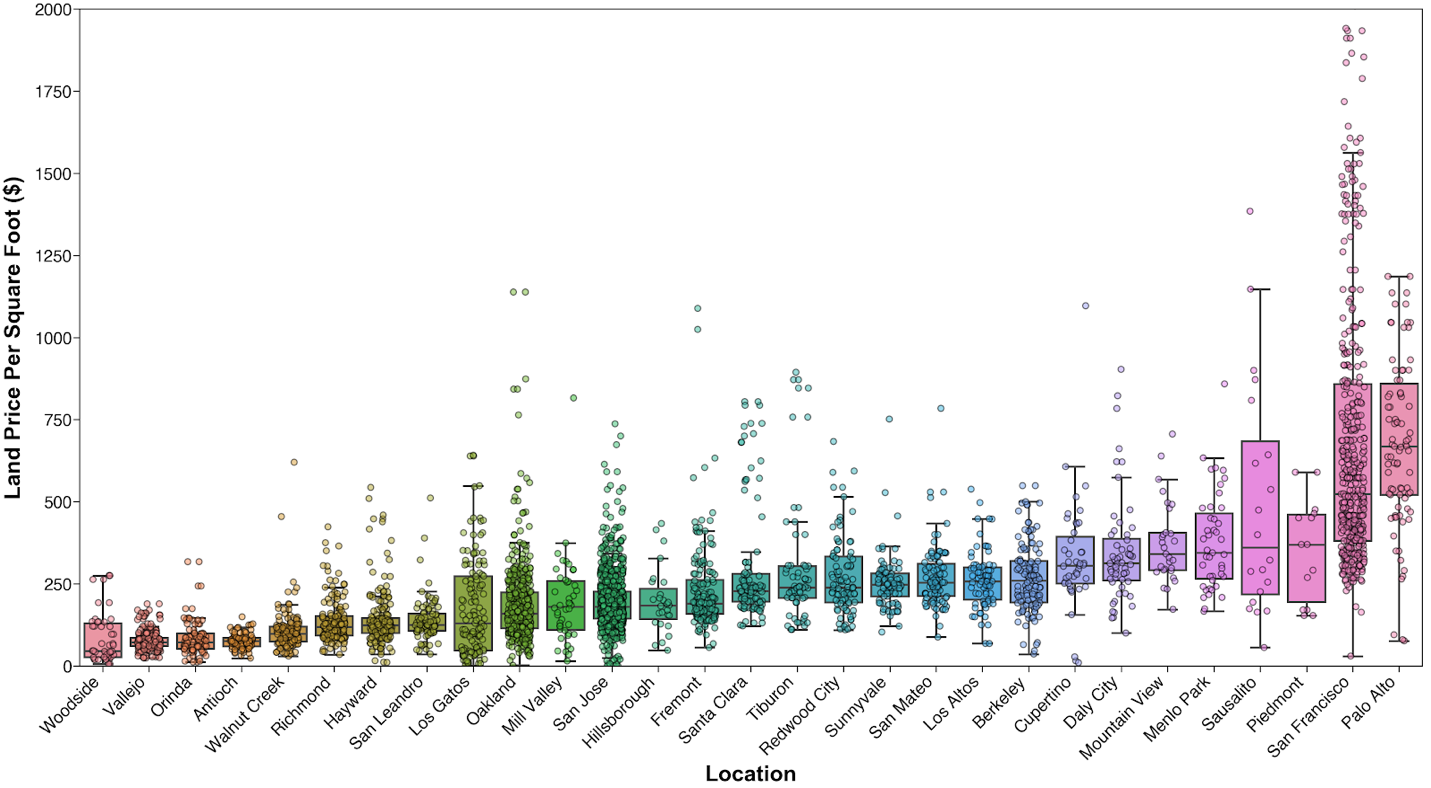
d. Aiming to capture other aspects of “place”, collected data on commute times, school quality, and crime rate (Figure 5?) – rationale for incorporating such metrics

e. Ran linear regression fit on the full data set, evaluating the effect (coefficients) and statistical significance (P-values) of the inputs to devise a model for home prices (Figure 6? Equation 1?)

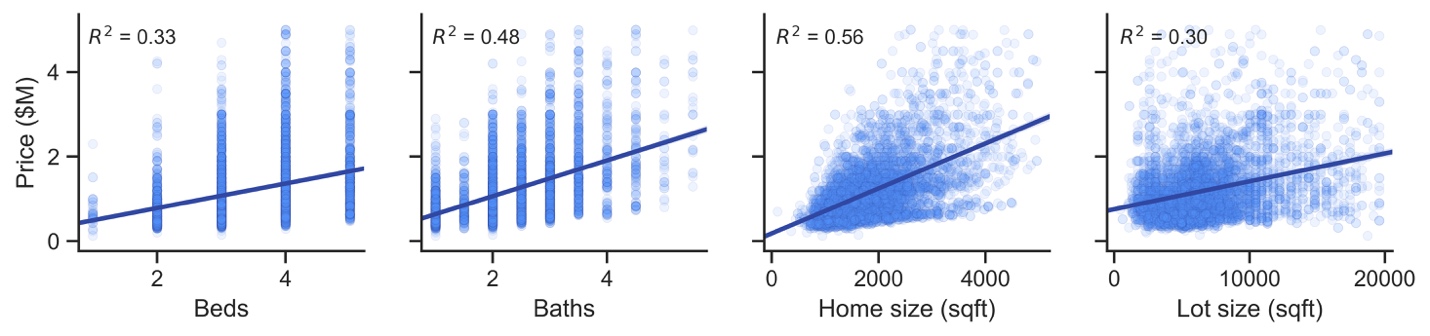
f. Calculated the difference between predicted and actual list prices, used this to identify potentially undervalued homes (Figure 7? – histogram of *P*pred – *P*list, Table 1 – 10 undervalued, 10 overvalued houses?)

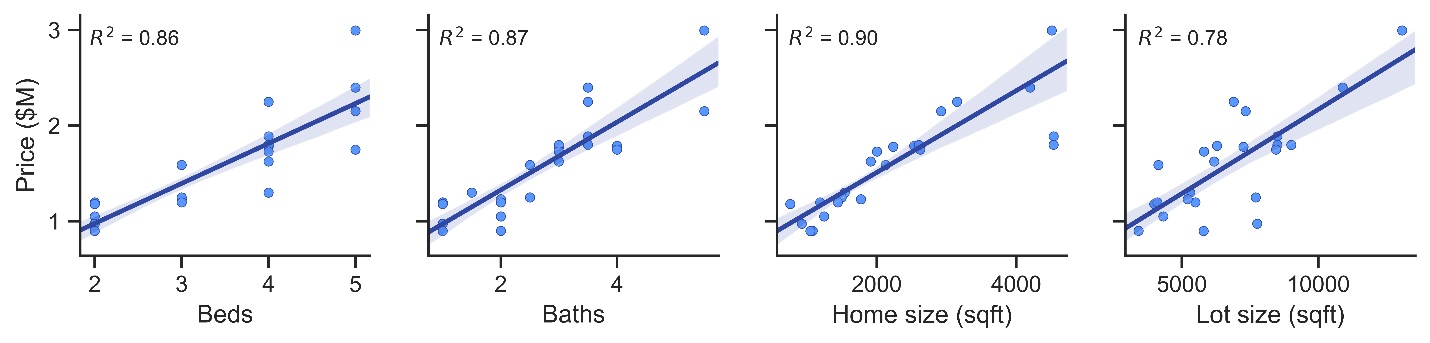
**Conclusions**

The application of data science techniques to inform real estate investment decisions need not be only pursued by corporate investors – free, open-source packages for use with Python empower the individual to scrape webpages, visualize data, apply machine learning to identify deals that may be overlooked by other market participants.

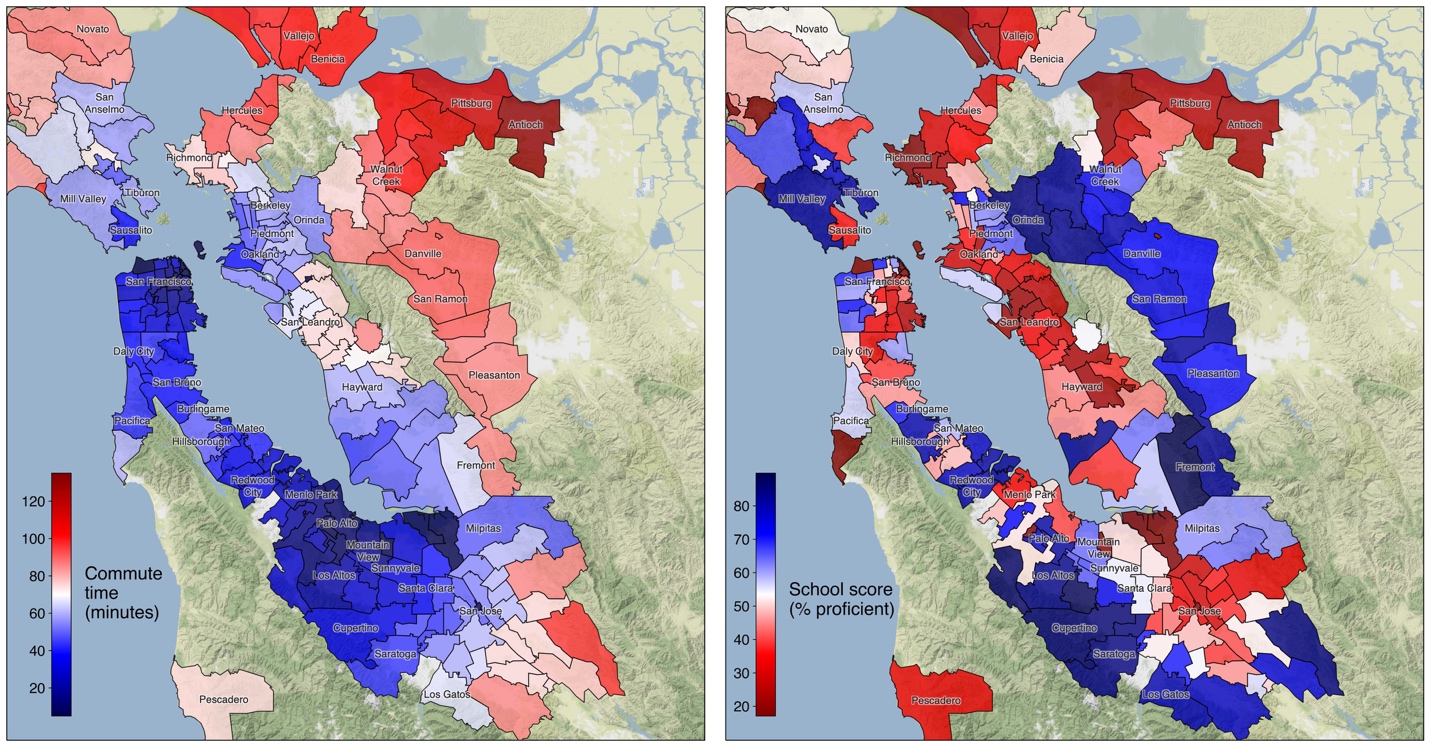
  
  


**Figure 3.** Box plots displaying home price (top) and land price (bottom) for selected Bay Area cities, with individual observations superimposed to reveal sample size and distribution.



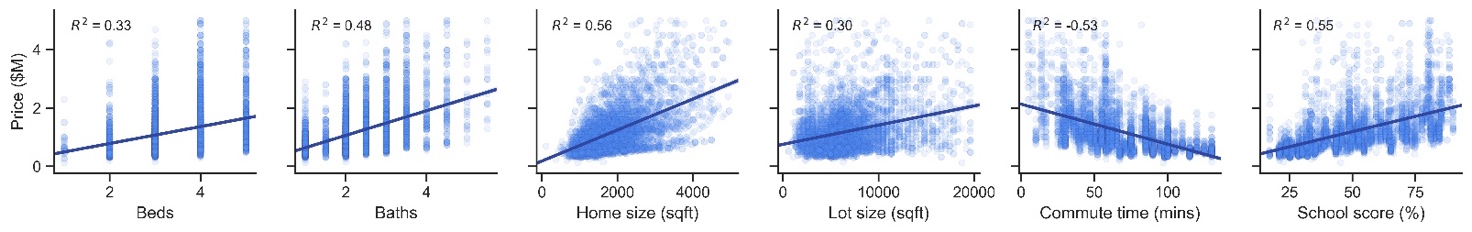


**Figure 4**. Correlation between price and listing factors is improved when full set of listings (top) is narrowed to a single zip code (bottom).

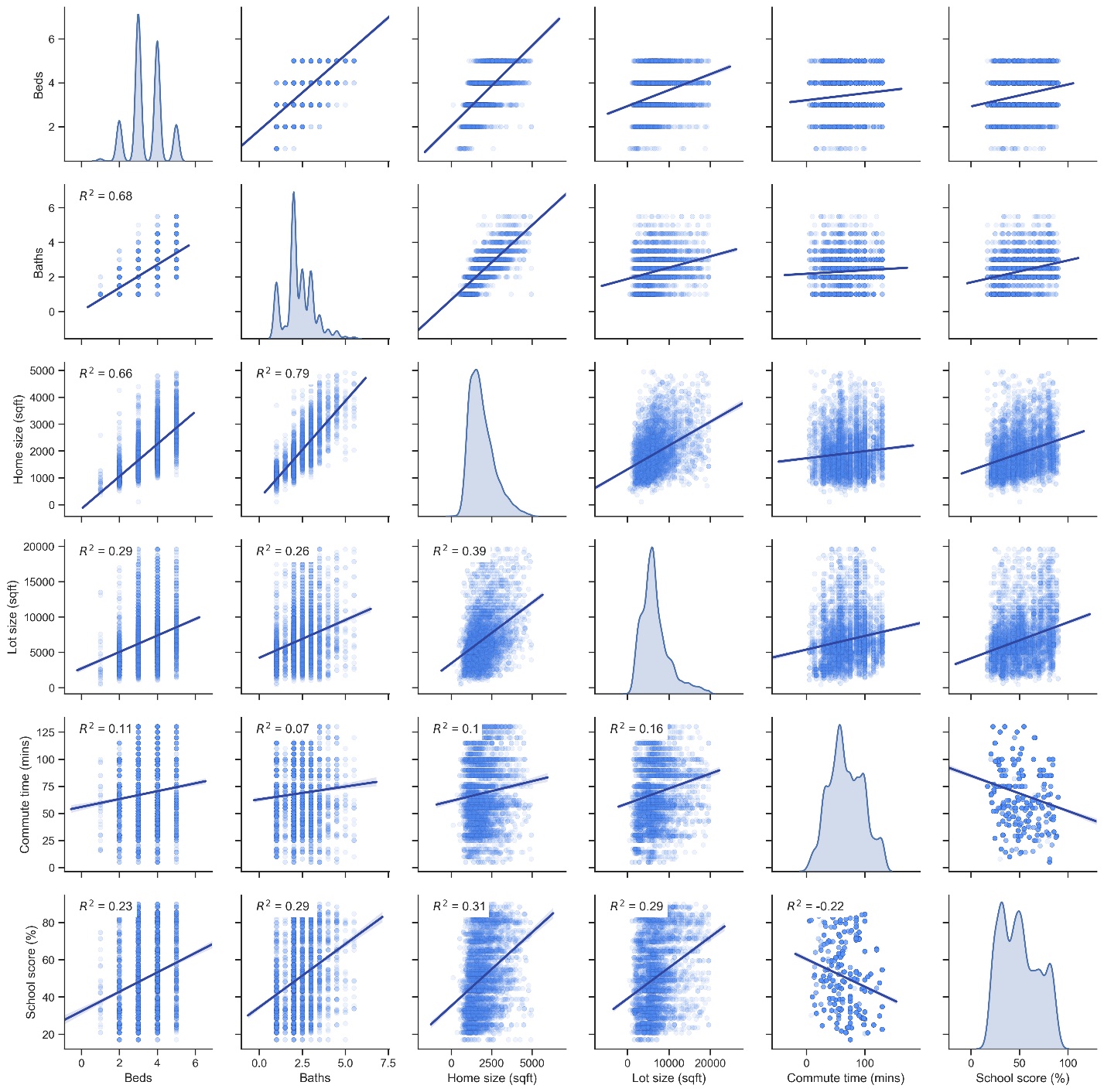


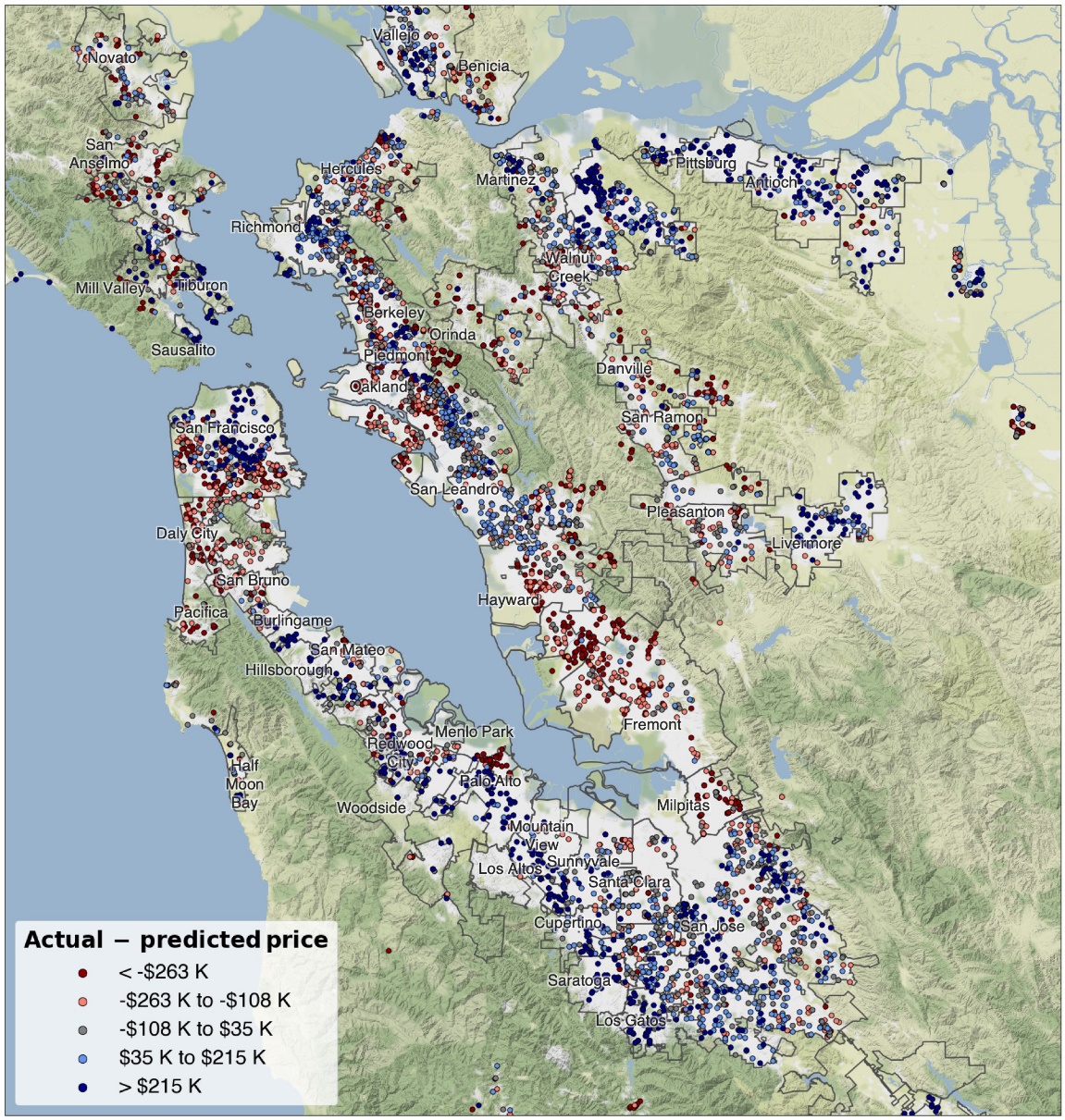
**Figure 5**. Commute times (left) and school quality (right) for zip codes across the Bay Area.

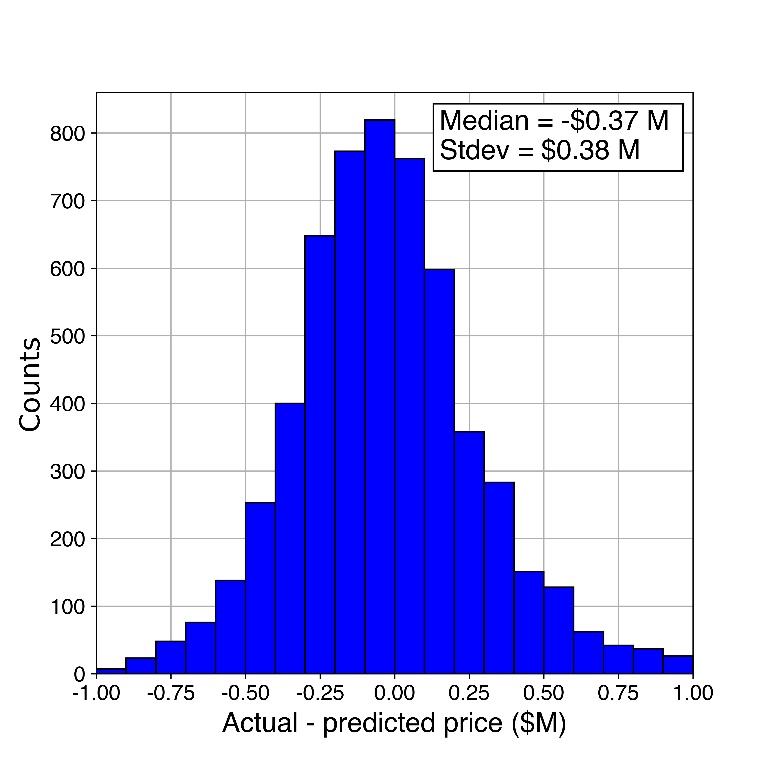
New parameters commute time, school score are significantly related to home price

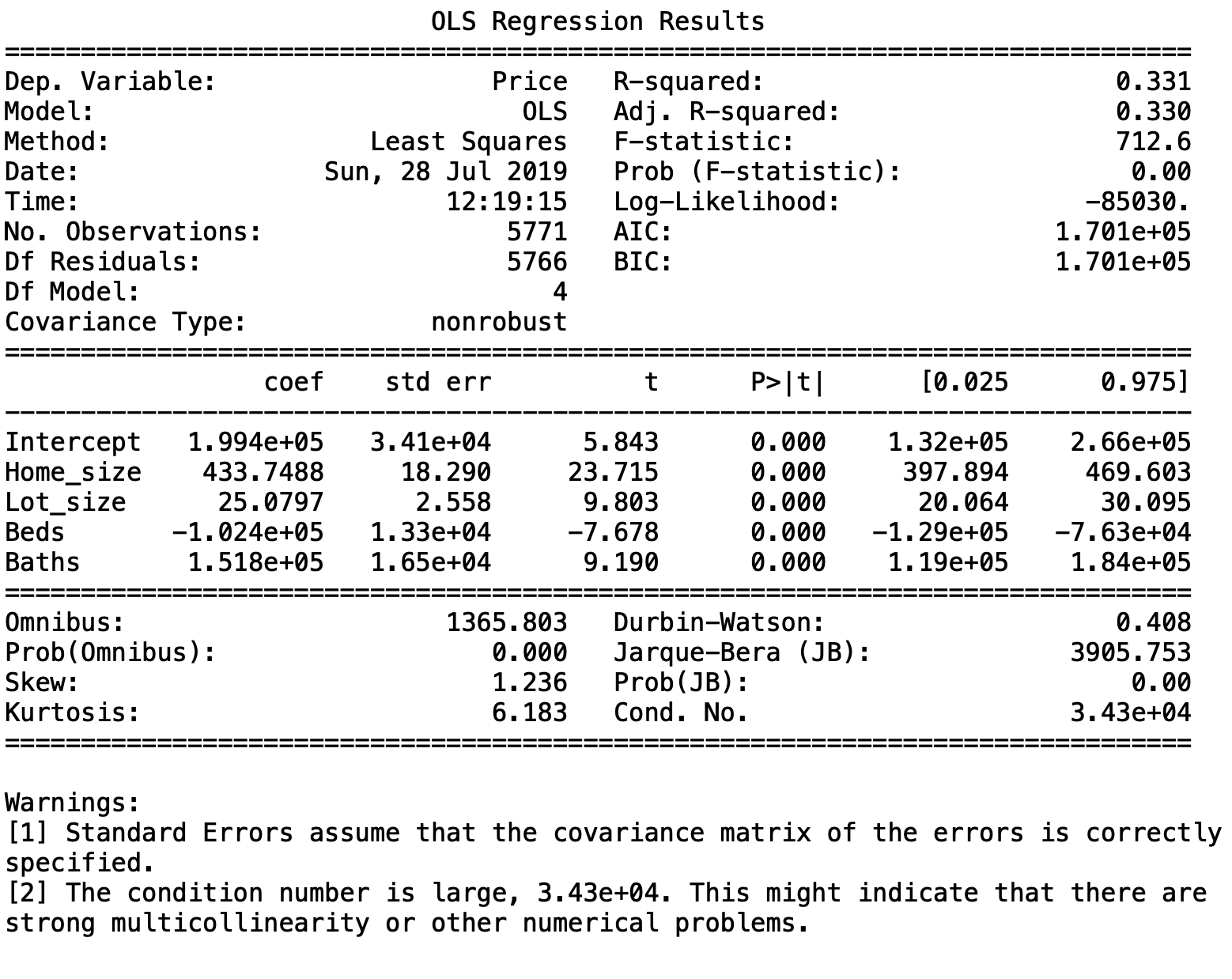


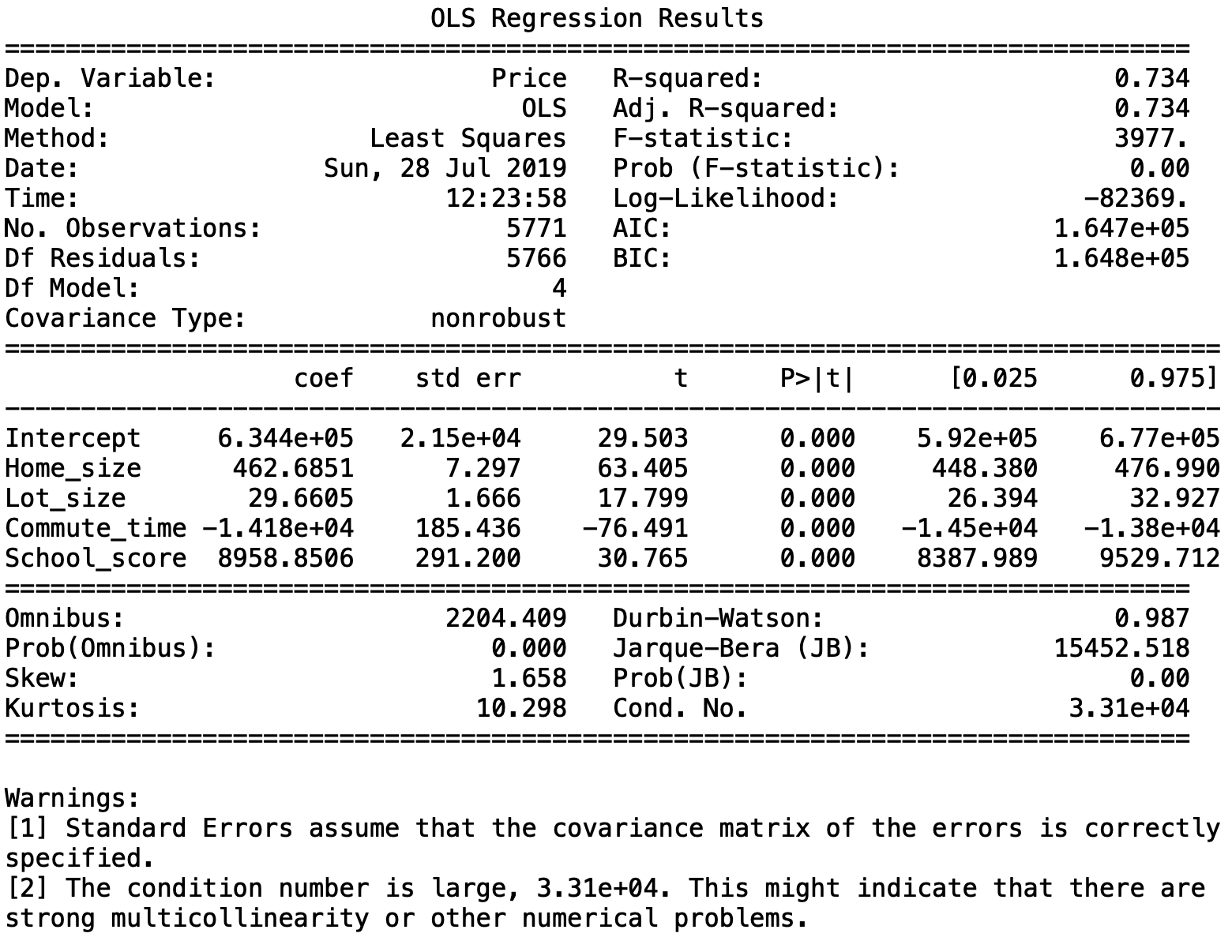
New parameters are not related to previous set – home, lot size. Three variables are however related – beds, baths, and home size (sqft). Because home size is most strongly correlated (R2 = 0.56) with price, beds and baths were discarded from the OLS model.











**Sources**

Shapefiles: Stanford Earthworks (https://earthworks.stanford.edu/catalog/stanford-vj593xs7263)

**Source code**

*Web scraping*

Mozilla Firefox developer tools was used to identify xpaths for items to be scraped and key-value pairs for data dictionary in the session.post() request

# import modules

from bs4 import BeautifulSoup

from lxml import html

import requests

import pandas as pd

import time

def webscrape(zipcodes):

# create empty data frame

data\_all = pd.DataFrame()

for counter, zipcode in enumerate(zipcodes,1):

# get homepage session

session = requests.Session()

homepage = session.get('https://www.mlslistings.com/')

soup = BeautifulSoup(homepage.content, "html.parser")

# get security token, post search data

token = soup.find("input", attrs={"name" : "\_\_RequestVerificationToken"})['value']

data = {'transactionType': 'buy', 'listing\_status': 'Active', 'searchTextType': '', 'searchText': zipcode,'\_\_RequestVerificationToken': token, 'property\_type': 'SingleFamilyResidence'}

search\_results = session.post("https://www.mlslistings.com/Search/ResultPost", data=data)

tree = html.fromstring(search\_results.content)

# update status

print('Scraping data for zipcode (%s/%s): ' % (counter,len(zipcodes)) + str(zipcode))

# scrape desired information

address = list(map(str, tree.xpath('//a[@class="search-nav-link"]//text()')))

price = list(map(str, tree.xpath('//span[@class="font-weight-bold listing-price d-block pull-left pr-25"]//text()')))

beds = list(map(str, tree.xpath('//span[@class="listing-info-item font-size-sm line-height-base d-block pull-left pr-50 listing-beds"]//text()')))

baths = list(map(str, tree.xpath('//span[@class="listing-info-item font-size-sm line-height-base d-block pull-left pr-50 listing-baths"]//text()')))

homesize = list(map(str, tree.xpath('//span[@class="font-weight-bold info-item-value d-block pull-left pr-25"]//text()')))

lot = list(map(str, tree.xpath('//span[@class="listing-info-item font-size-sm line-height-base d-block pull-left pr-50 listing-lot-size"]//text()')))

yearbuilt = list(map(str, tree.xpath('//span[@class="listing-info-item font-size-sm line-height-base d-block pull-left pr-50 listing-sqft last"]//text()')))

# create data frame from scraped, cleaned data

data\_temp = {'Address': address, 'City': city, 'Zip': zip\_code,

'Beds': beds, 'Baths': baths, 'Home size': homesize,

'Lot size': lot, 'Year built': yearbuilt, 'Garage': garage,

'Home type': hometype, 'Price': price}

dataframe\_temp = pd.DataFrame(data\_temp)

data\_all = data\_all.append(dataframe\_temp)

print('Zipcode %s was skipped' % zipcode)

# wait, then scrape next zipcode

time.sleep(1)

return data\_all