Data Science Zillow Case Study

Drew Fingerhut

4/1/2022

Background

As a Data Analyst I am always seeking oppurtunites to increase my skill set and better myself. I have completed a variety of courses surrounding the Data Science field including machine learning, neural networks, NLP, and webscraping. As a way to show case this I took it upon myself to create a WebScraper using the BeautifulSoup library in Python.

Overview

In this case study I set out to search Indeed for Zillow house postings and gather a few key features from the listings. I thought it most important to gather the price, address, number of beds, baths, and square feet if listed to compare listings in the current market. I did so by generating a WebScraper in Jupyter Notebook using the BeautifulSoup Library in Python. A spreadsheet was then generated and imported into Excel, where using the get data function, it was cleaned and organized for better interpretation. The cleaned data was then loaded into Tableau for visualization and analysis. I did this to quickly gain insight into markets for buying a future home or to potentially invest in.

Data Extracting

In Jupyter notebook I created a webscraper that will iterate through a selected number of pages on Zillow and pull all the data from the listings and format the price, address, beds, baths, and square feet seperated into indivivual columns. I did so by using the BeautifulSoup library for html parsing as well as the Pandas library for creating and structing data frames. After populating my data frame with all the metrics I was looking for I exported this data frame into an excel file so I could do additional cleaning and visualization. On the next page the code and the associated commenting is shown.

Python Script

```python
#Zillow Webscraper by Drew Fingerhut
#4/1/2022

#Importing the necessary Libraries
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import requests

#Establishing the lists to be populated outside of iteration
price = []
bed = []
address = []

#Creating the loop for a specified range representing the number of pages to scrape
for i in range(30):

#Passing in a request header as to get the dynamic page rather than the static page that can not be scraped
    req_headers = {
        'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
        'accept-encoding': 'gzip, deflate, br',
        'accept-language': 'en-US,en;q=0.8',
        'upgrade-insecure-requests': '1',
        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.316
                    }

#Generating a new url for each page that is to be requested
    with requests.Session() as s:
       page_number = i+1
       page = str(page_number)+'_p/'
       city = 'salt-lake-city/' #*****change this city to what you want*****
       url = 'https://www.zillow.com/homes/for_sale/'+city+page
       r = s.get(url, headers=req_headers)
```

```python

#Parse the requested html and look for each title card that has the listing info enclosed in it
    soup = BeautifulSoup(r.content, 'html.parser')
    house_elements = soup.find_all("div", class_="list-card-info")

#Search through each title card for price, beds, and address and populating to the preestablished lists only keeping the
    for house_char in house_elements:
        price_element = house_char.find("div", class_="list-card-price")
        bed_element = house_char.find("ul", class_ ="list-card-details" )
        address_element = house_char.find("address", class_="list-card-addr")
        if price_element != None:
            price.append(price_element.text)
        elif price_element == None:
            price.append('Blank')
        if bed_element != None:
            bed.append(bed_element.text)
        elif bed_element == None:
            bed.append('Blank')
        if address_element != None:
            address.append(address_element.text)
        elif address_element == None:
            address.append('Blank')

#Populate the data frame with the listes that were just appended to
    df = pd.DataFrame ({
                    "Price": price,
                    "Bed": bed,
                    "Address": address
                        })
#Create a save point of the data frame before altering the data to ensure there is not loss
    df_new = df.copy()
```

Python Script (continued)

```
65
66 #Replacing unnecessary text in Bed column with spaces so that it can be seperated into individual columns
67     df_new['Beds'] = df_new['Bed'].str.replace(' bds',' ')
68     df_new['Beds'] = df_new['Beds'].str.replace(' ba',' ')
69     df_new['Beds'] = df_new['Beds'].str.replace(' sqft',' ')
70     df_new['Beds'] = df_new['Beds'].str.replace(',','')
71     df_new['Beds'] = df_new['Beds'].str.replace('--','')
72     df_new['Beds'] = df_new['Beds'].str.replace('/',' ')
73     df_new['Beds'] = df_new['Beds'].str.split('-')
74     df_new['Beds'] = df_new['Beds'].fillna(0)
75     df_new.drop(df_new.tail(1).index,inplace=True)
76
77  #CLeaning and dropping the wording such as Multi-family for sale or Townhouse for sale from data frame
78     split_df = pd.DataFrame(df_new['Beds'].tolist(),columns=['Info','Lose','Ex'])
79     split_df = split_df.drop(['Lose','Ex'],axis=1)
80
81  #Splititng Bed coloumn post cleaning into 3 seperate columns for bed, bath, and square feet
82     split_df[['Bedrooms','Bathrooms','Sq_Feet']] = split_df['Info'].str.split(' ', n=2,expand=True)
83     split_df = split_df.drop(['Info'],axis=1)
84
85 #Concatting our split data frame to the original and dropping now unnecessary columns
86     full_df = pd.concat([df_new, split_df],axis=1)
87     full_df = full_df.drop(['Bed','Beds'],axis=1)
88
89 #Displaying our full data frame after iterating through all the pages we defined it to
90 full_df
```

Out[1]:

	Price	Address	Bedrooms	Bathrooms	Sq_Feet
0	$799,900	1768 S 1400 E, Salt Lake City, UT 84105	4	2	2222
1	$569,000	32 W 1700 S #A13, Salt Lake City, UT 84115	3	3	1725
2	$485,000	886 N Catherine St, Salt Lake City, UT 84116	3	2	1670
3	$355,000	1162 S Emery St, Salt Lake City, UT 84104	2	1	772
4	$525,000	1627 S Denver St, Salt Lake City, UT 84115	2	1	1683
...	...	...	...	...	...
244	$815,000	166 S 1200 E, Salt Lake City, UT 84102	4	2	2096
245	$679,900	1710 S West Temple St W UNIT 11, Salt Lake Cit...	3	4	2096
246	$695,000	1202 E Princeton Ave, Salt Lake City, UT 84105	3	1	1730
247	$525,000	1627 S Denver St, Salt Lake City, UT 84115	2	1	1683
248	$589,900	753 E Loveland Ave, Salt Lake City, UT 84106	3	2	1702

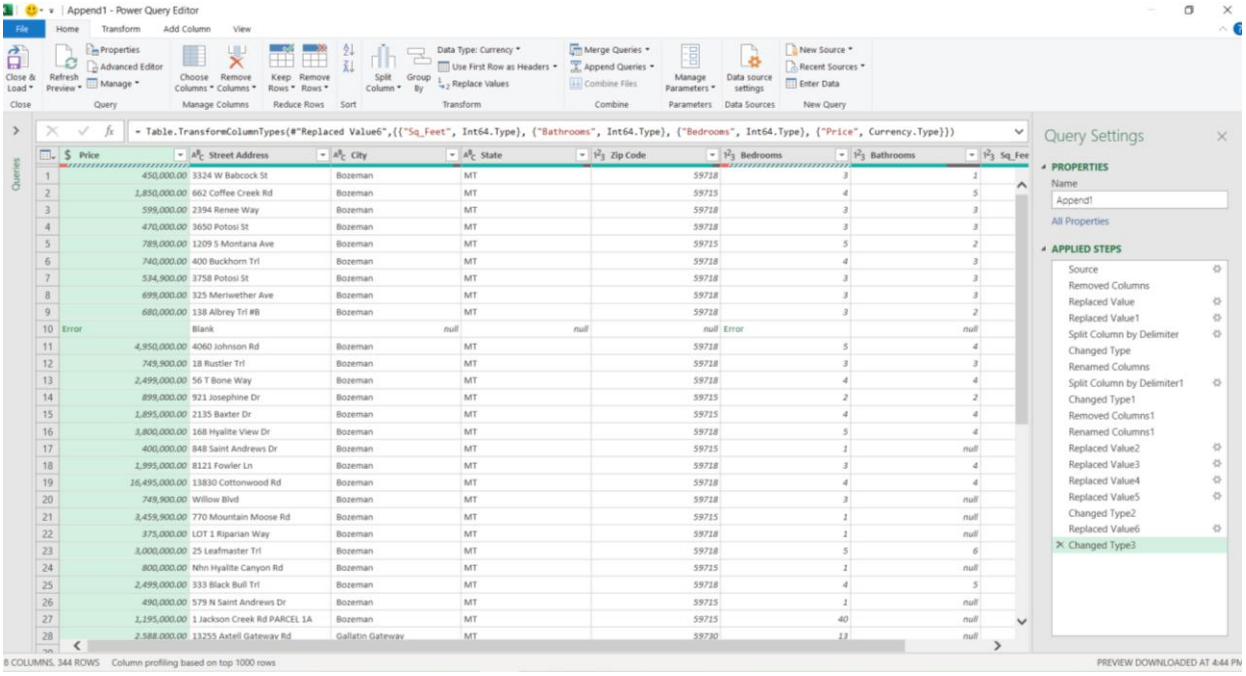
249 rows × 5 columns

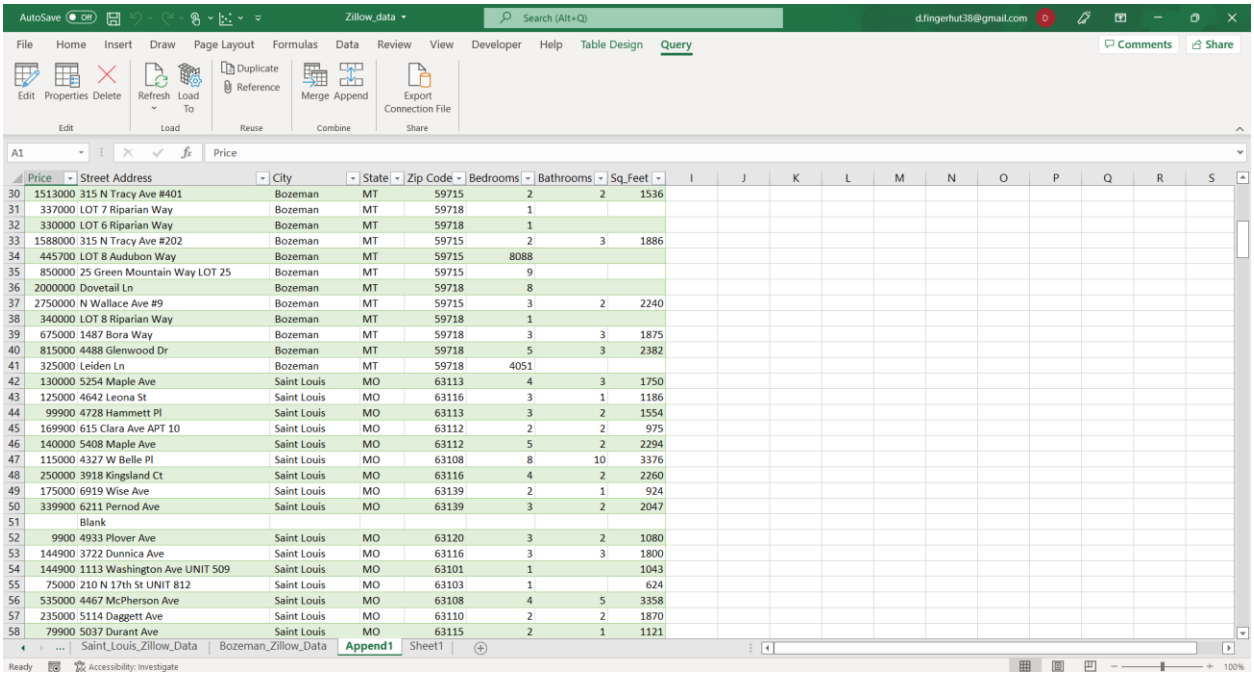
In [2]:
```
1 final_df = full_df.drop_duplicates()
```

In [4]:
```
1 final_df.to_csv('Salt_Lake_City_Zillow_Data.csv')
```

Transforming the Data

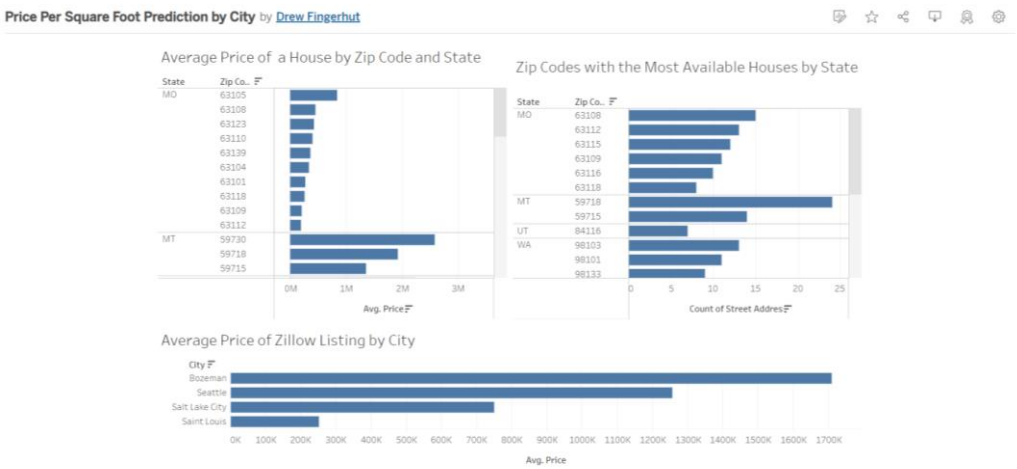
After running the script above for Seatlle, Salt Lake City, Bozeman, and Saint Louis and saving each of them as separate csv files I used the get data function in Excel to gather them all into one sheet and do some transformation to make visualization easier. I went through and removed commas and dollar signs. I seperated the address into individual address, city, and state columns. I also replaced certain verbage that was not initially caught in the Python script from the bed column. After this I then loaded the data back into Excel.



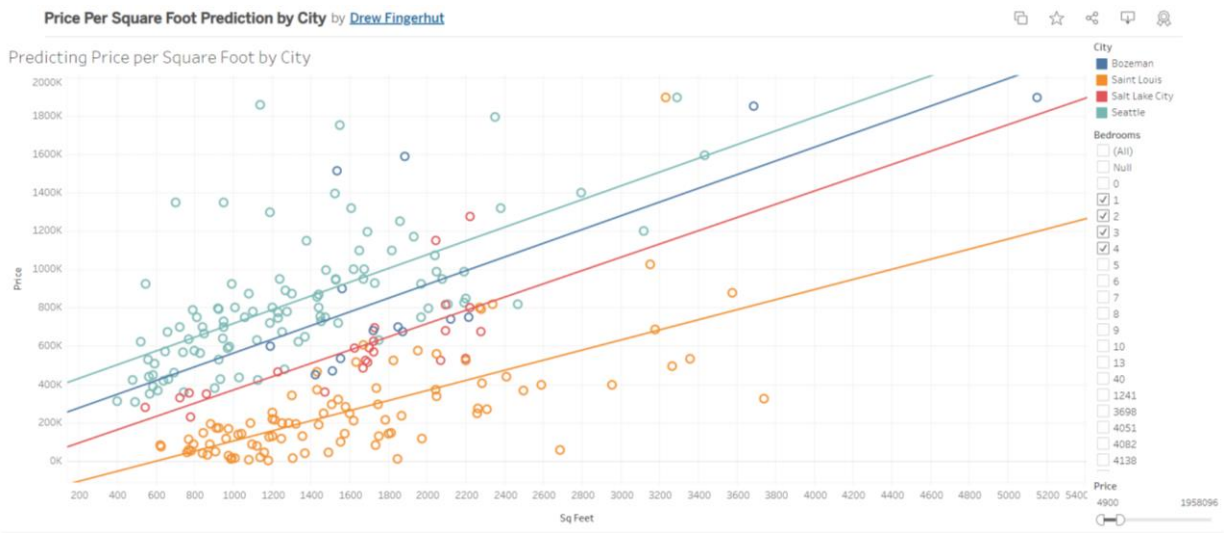


Data Visualization

With this new compounded Excel sheet I uploaded it into Tableau for visualization. I thought it best to break it down into a few sepereate visuals. I first broke it into Average Price by City, Average Price by Zip Code, and Number of house available by Zip Code. I then combined these into a dashboard that can be filtered out by price. I did this for a large overarching visualization to show you a few key metrics about the different areas for further investigation depending on investing constraint or market availability.



I then created a visualization that can be filtered out by price and number of bedrooms. I then added coloring to the cities for easier interpretation and a trendline based on the price compared to the square footage. This is a key metric that when comparing the markets will show you how much house you are getting for the price.



Reviewing Success

At the end of this exercise, I view it as a massive success. I was able to write a universal script that can pull data for any city and number of pages from Zillow and save it to a csv file. I was then able to transform this data for further analysis and visualize it for key insights. I was able to identify that Seattle was the most expensive price by square foot and Saint Louis was the cheapest. I was able to create a visualization that based on the square feet and city could infer a price it would be listed for. I was also able to see how many houses were available per zip code and what the average price in that zip code was. This will give me the necessary key metrics I would need when starting to house hunt and give me the flexibility I need to sort by number of beds and price.

Future Use

Moving forward I will be adding more cities to this and narrow in on a particular region to invest in. With the prebuilt scripting and excel queries it will be a quick and painless process to analyze any area for future investing. Thanks for reading through this and feel free to reach out if there are any questions!