**An EDA Dashboard: Real Estate in Vancouver, BC, Canada**



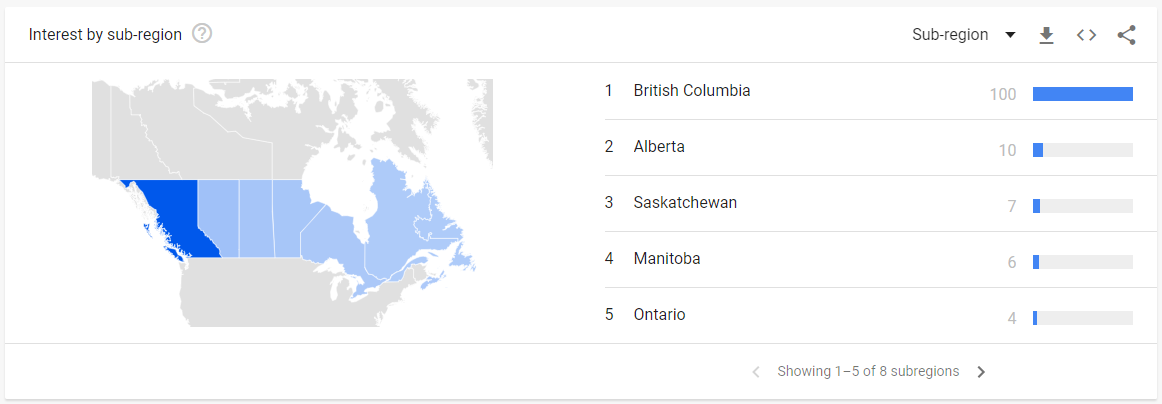
VANCOUVER LOOKOUT AT HARBOUR CENTRE TOWER, Source: [https://www.aaa.com]

**Author**: Kasra Heidarinezhad

**Last Update**: 10/11/2022

**Last Data Captured**: Jan-16-2023

Nowadays, finding an affordable house in Canada is very difficult. Study of Google Trends data [https://trends.google.com] shows that British Columbia, especially Vancouver got higher ranking in search keywords between searchers among all provinces in Canada.



Google Trends output for “Vancouver real estate” comparison in Canada. Image by the author.

According to Statista (https://www.statista.com/statistics/1040698/most-expensive-property-markets-worldwide/), Vancouver ranks 7th most expensive residential property markets worldwide in 2020. To finding a reasonable price property, there are couple of different resource in the market. For example websites such as Realtor.com and Redfin.com. In this project, we decided to analysis real estate market in Vancouver, BC, Canada with retrieving listing detail from one of existing resource.

**Goal 1:** To predict the sales price in house listing. For each item in the test set, we must predict the value of the sale-price variable.

I describe the progress of this project step by step here:

**The Project Steps:**

Step 1: Creating Data Source, Data Cleaning, Missing values

Step 2: Exploratory Data Analysis

Step 3: Query/Manipulate Data

Step 4: Push CSV to Google Drive and Connect to Tableau

Step 5: Analysis

Step 6: Data Exploration/Visualization

**The Tools:**

Python

SQL

Tableau (Since currently I don’t have a Tableau Desktop account, I used Tableau Public)

GCP (Google Cloud Platform)

Google Trends

**Step 1: Creating Data Source**

First step is to get data to analysis. You have two choice here: First is to link a government agency or real estate license to use as a part of research (Dose not work for me). Second choice is scraping data. In this article, I kept going with the second option. Finding resource is next challenge of this project. To choice a good option among the existing resource, I considered two following options:

1- Comprehensively of data that providing by a website

2- Difficulty of accessing to data within website

After a quite study and considering above mentioned factors, I decided to going to Zillow (<https://Zillow.com>).

Google Trends data

**Web Scraping with Python**

**Required Libraries**

Here we list a major required libraries. For easiness, we could throw all of them in a RequiredLibraries.txt file and run: pip install RequiredLibraries.txt.

import os

from bs4 import BeautifulSoup

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

import warnings

import numpy as np

import pandas as pd

import lxml

from lxml.html.soupparser import fromstring

import prettify

import htmltext

import requests

import re

import json

Web Headers: Zillow throw Captchas so when you try and run a request.get(url) type of function. So way to get around this is by adding headers to the request function as you can see below:

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/61.0.3163.100 Safari/537.36'

}

Following step is done to complete job:

Parse data from urls in looping through pages

Create and append Data Frames

Export Data frame to csv file

Complete source

Plot to rule them all

Here is complete code and result

warnings.filterwarnings('ignore')

#add headers in case you use chromedriver (captchas are no fun); namely used for chromedriver

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/61.0.3163.100 Safari/537.36'

}

with requests.Session() as s:

    data\_list = []

    resp = s.get('https://www.zillow.com/homes/for\_sale/Vancouver,-BC\_rb/', headers=req\_headers)

    data = json.loads(re.search(r'!--(\{"queryState".\*?)-->', resp.text).group(1))

    data\_list.append(data)

    for pages in range(1,20):   #just grabbing the first 20 pages

       resp = s.get('https://www.zillow.com/homes/for\_sale/Vancouver,-BC\_rb/'+str(pages+1)+'\_p/', headers=req\_headers)

       data = json.loads(re.search(r'!--(\{"queryState".\*?)-->', resp.text).group(1))

       data\_list.append(data)

df = pd.DataFrame()

def make\_frame(frame):

    for i in data\_list:

        for item in i['cat1']['searchResults']['listResults']:

            frame = frame.append(item, ignore\_index=True)

    return frame

df = make\_frame(df)

df = df.drop('hdpData', 1) #drop cols

df = df.drop\_duplicates(subset='zpid', keep="last") #drop dupes

#filters

df['zestimate'] = df['zestimate'].fillna(0)

df['best\_deal'] = df['unformattedPrice'] - df['zestimate']

df = df.sort\_values(by='best\_deal',ascending=True)

df.to\_csv('zillow\_original.csv', encoding='utf-8')

df

Unnamed: 0 int64

zpid int64

id int64

providerListingId float64

imgSrc object

hasImage object

carouselPhotos object

detailUrl object

statusType object

statusText object

countryCurrency object

price object

unformattedPrice int64

address object

addressStreet object

addressCity object

addressState object

addressZipcode object

isUndisclosedAddress bool

beds int64

baths int64

area float64

latLong object

isZillowOwned bool

variableData object

badgeInfo float64

isSaved bool

isUserClaimingOwner bool

isUserConfirmedClaim bool

pgapt object

sgapt object

zestimate int64

shouldShowZestimateAsPrice bool

has3DModel bool

hasVideo bool

isHomeRec bool

info1String object

brokerName object

hasAdditionalAttributions bool

isFeaturedListing bool

availabilityDate float64

list bool

relaxed bool

hasOpenHouse object

openHouseStartDate object

openHouseEndDate object

openHouseDescription object

streetViewMetadataURL object

streetViewURL object

best\_deal int64

dtype: object

**Note**: keep in mind that it uses anti-scraping techniques like captchas, IP blocking, and honeypot traps to prevent its data from scraping. I do think my IP/device is probably blacklisted by now

Congratulation: First step done!

**Data Cleaning**

Data cleaning involves identifying and correcting any errors or inconsistencies in the data, such as missing values, duplicate records or incorrect data types. This is an important step because it helps to ensure that the data is complete and accurate, which is necessary for building reliable models. For this purpose, some columns that are not likely to help in data analysis and predicting the target variable are dropped from the original data frame.

**Missing Values**

It is important to fill in missing data with NaN (Not a Number) because it allows you to identify missing values in your data clearly. Filling missing values with NaN allows us to identify which values are missing and take appropriate action easily. Here, the missing values in the columns ‘Volume’, ‘Interior’, ‘Availability’, ‘Garage’, ‘Upkeep Status,’ ‘Specification’, ‘Location Type’, ‘Number of floors’, ‘Details of Garden’, ‘Details of Storage’, ‘Number of Bedrooms’, ‘Details of Balcony’, ‘Number of Bathrooms’ and ‘Description of Storage’ are addressed by filling it with NaN values.

**Step 2: Exploratory Data Analysis**

After data cleaning, shape of data is like result:

df = pd.read\_csv("zillow.csv")

print(f"Number of samples: {df.shape[0]}")

print(f"Number of features in set: {df.shape[1]}")

print("Features:")

print(df.dtypes)

Result:

**Number of samples: 800**

**Number of features in set: 18**

**Features:**

**index int64**

**zpid int64**

**id int64**

**imgSrc object**

**detailUrl object**

**TypeofProperty object**

**formattedprice object**

**Price int64**

**address object**

**addressStreet object**

**addressZipcode object**

**beds float64**

**baths float64**

**area int64**

**latLong object**

**has3DModel bool**

**brokerName object**

**best\_deal int64**

**dtype: object**

Here we will some codding to create a shape of data and histogram.

For plot histogram:

# histogram of target=price variable

sns.set\_theme()

graph = sns.displot(data=propertyListFrame, x="Price", kde=True, log\_scale=True, bins=50)

graph.set(title="Histogram of Sale Price")

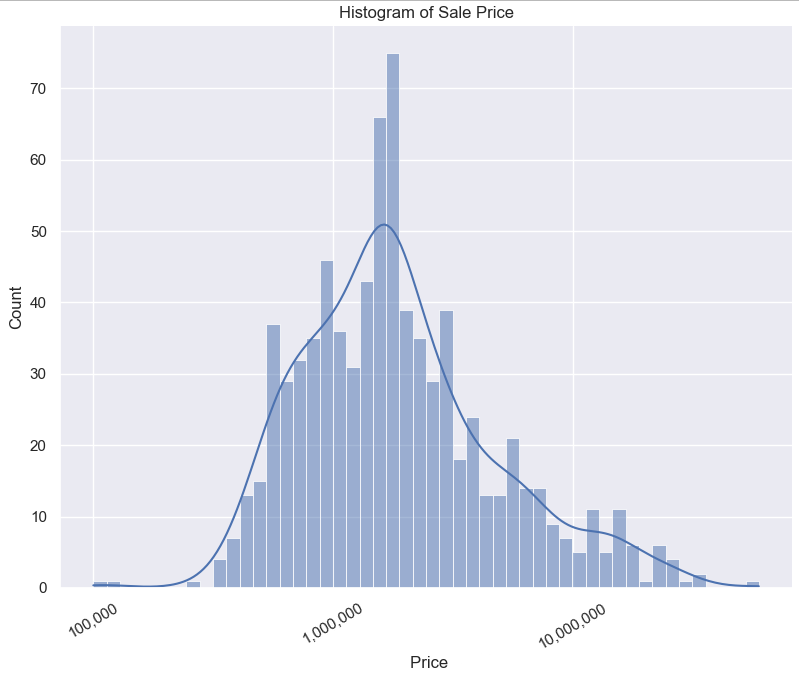
for ax in graph.axes.flat:

   ax.xaxis.set\_major\_formatter(tkr.FuncFormatter(lambda x, p: format(int(x), ',')))

plt.xticks(rotation=30)

plt.show()

Result:



**(Image by the author)**

Let’s have a look in to the other four major statistic parameter:

**Mean: 3,164,698.75 CA$**

**Median: 1,675,000.00 CA$**

**Skewness: 4.6598**

**Kurtosis: 33.9900**

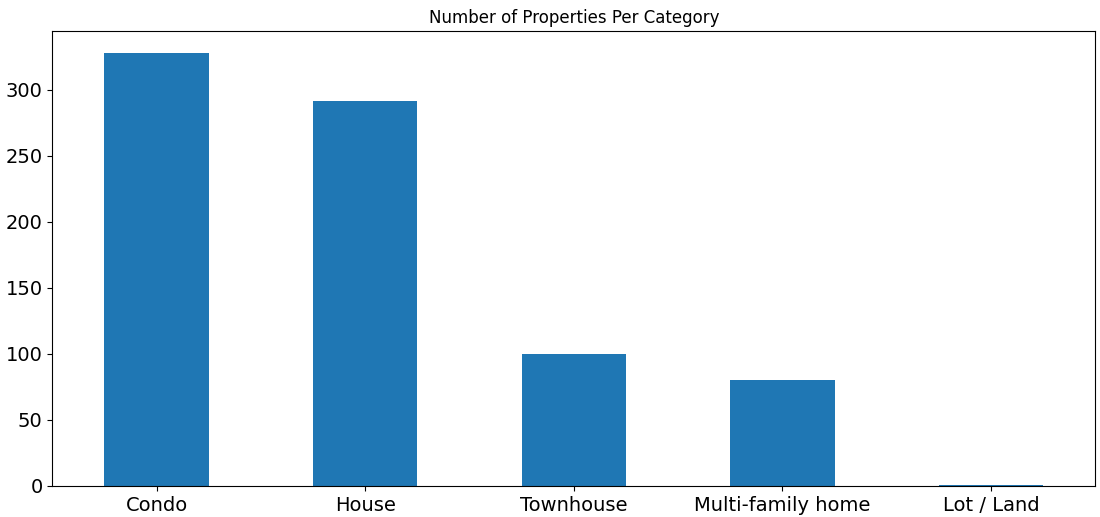
Analysis: It shows the distribution of Price house in Vancouver, BC, is relatively high positive skewness (the mean is greater than the median). In the prediction model, this will cause, the less accurate in the price prediction. In the other hand, high kurtosis represents heavy tails meaning more outliers.

df['TypeofProperty'] = df['TypeofProperty'].str.strip()

graph = df['TypeofProperty'].value\_counts()

graph.plot(kind='bar', fontsize=14, title="Number of Properties Per Category", rot=0)

plt.show()



**(Image by the author)**

**Activity of Brokers**

Most of listing is created by 119 broker/agent how have most activity in the market

propertyListFrame['brokerName'] = propertyListFrame['brokerName'].str.strip()

type(propertyListFrame['brokerName'][0])

print(propertyListFrame['brokerName'].nunique())

119

**Histogram - Group by: Type of Property/ Price range science**

df['TypeofProperty'].value\_counts()

**Condo 328**

**House 291**

**Townhouse 100**

**Multi-family home 80**

df1 = pd.read\_csv('zillow.cs

df = df1[['Price', 'TypeofProperty' ]]

df['binned\_price'] = pd.cut(df.Price, [1, 1000000, 2000000, 3000000, 4000000, 5000000, 6000000, 7000000, 8000000, 9000000, 10000000])

df.groupby('binned\_price')['TypeofProperty'].value\_counts()

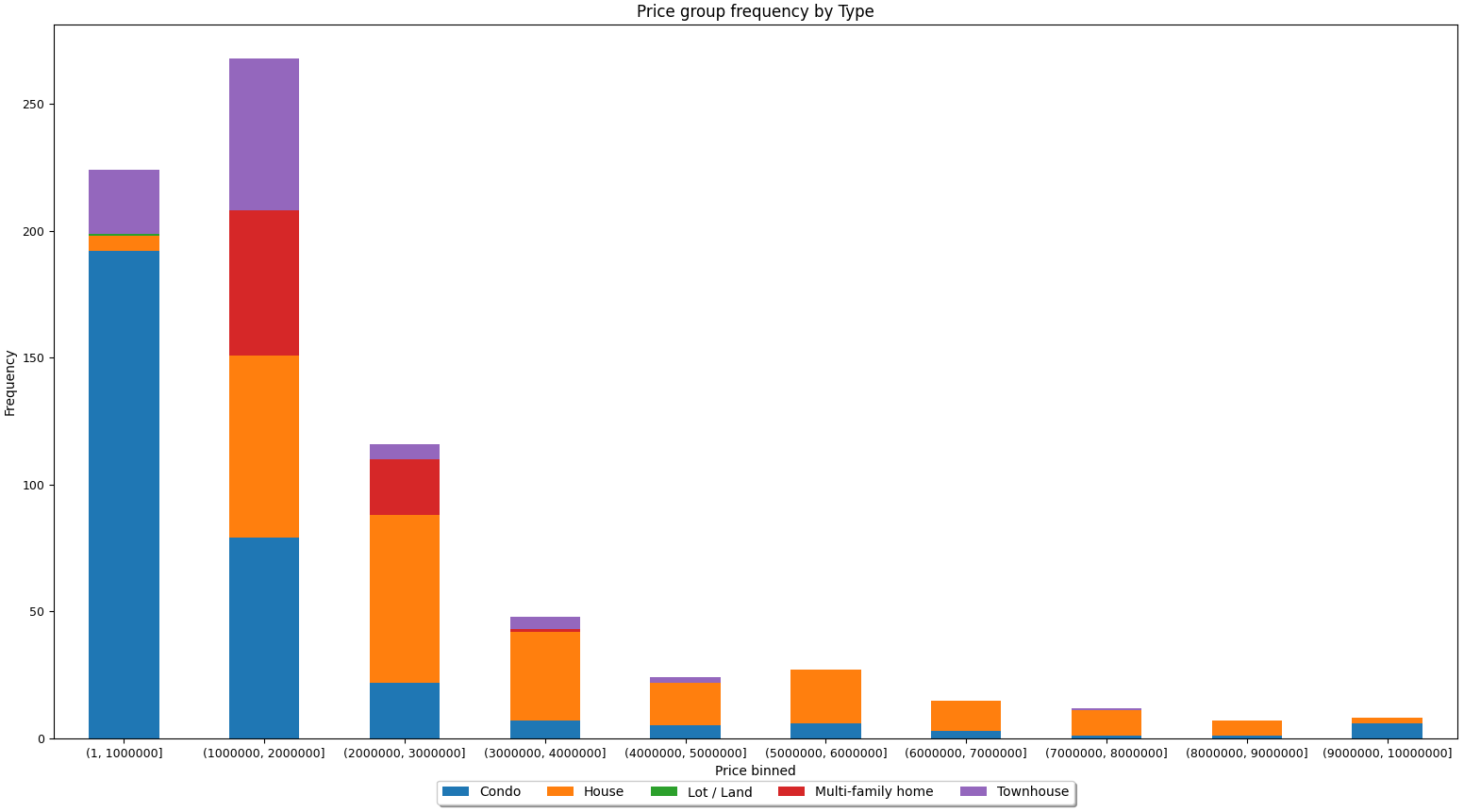
df.drop('Price',axis=1,inplace=True)

df\_reshaped = df.pivot\_table(index=['binned\_price'], columns=['TypeofProperty'], aggfunc=len)

df\_reshaped.plot(kind='bar', stacked=True, ylabel='Frequency', xlabel='Price binned',title='Price group frequency by Type', rot=0, fontsize=9)

plt.legend(loc='upper center', bbox\_to\_anchor=(0.5, -0.05),fancybox=True, shadow=True, ncol=8)

plt.show()



**(Image by the author)**

This is the top 20 realtor histogram

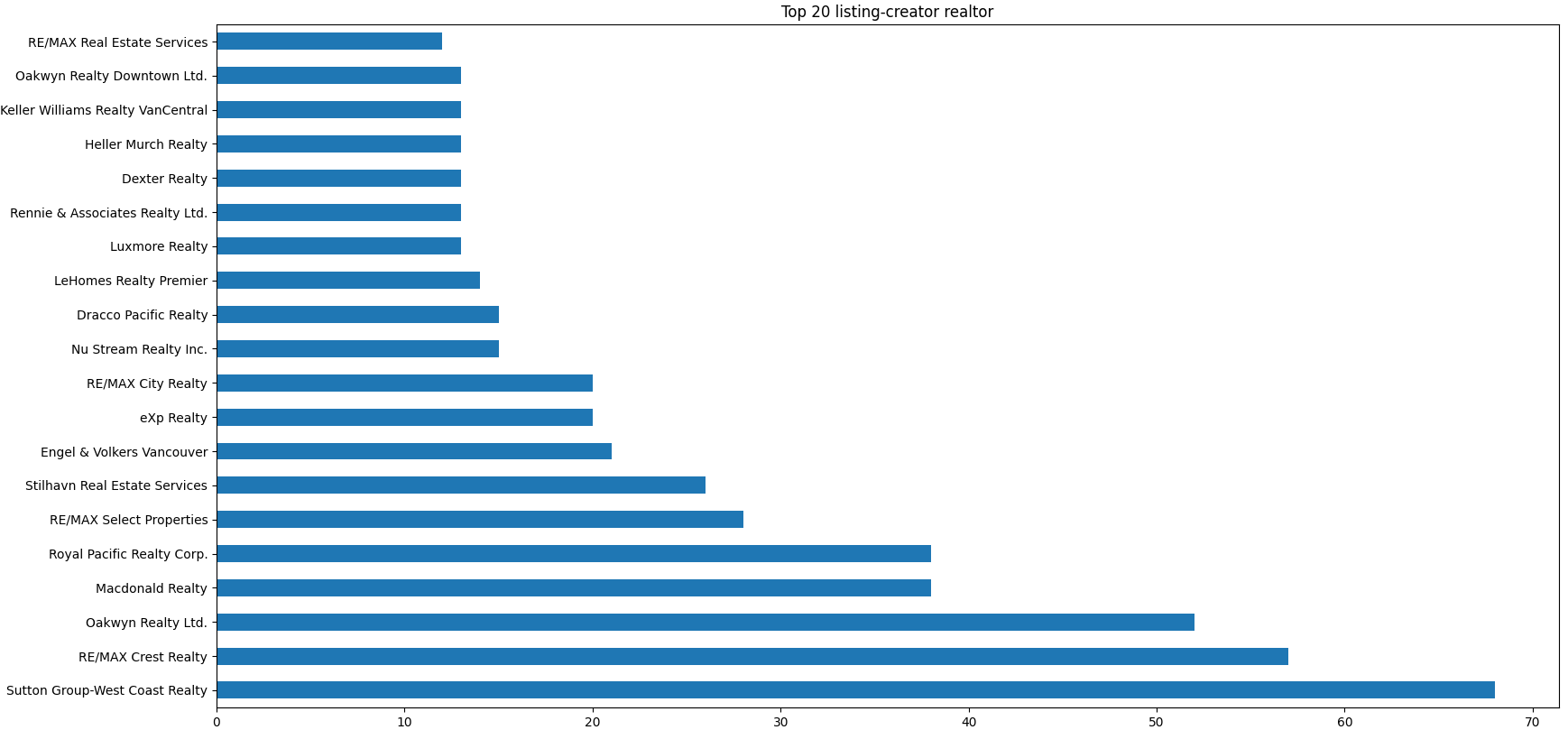
df['brokerName'] = df['brokerName'].str.strip()

#print(df['brokerName'].nunique())   #print number of brokers

graph = df['brokerName'].value\_counts()[:20]

graph.plot(kind='barh', fontsize=10, title="Top 20 listing-creator realtor")

plt.show()



**(Image by the author)**

properties = pd.read\_csv("zillow.csv")

plt.rcParams.update( {'figure.figsize':(10,8), 'figure.dpi':100})

plt.scatter(x=properties.Price, y=properties.area, c=properties.beds, cmap='Spectral')

plt.colorbar(label="Number of Beds")

plt.title('Relationships between Price and Area')

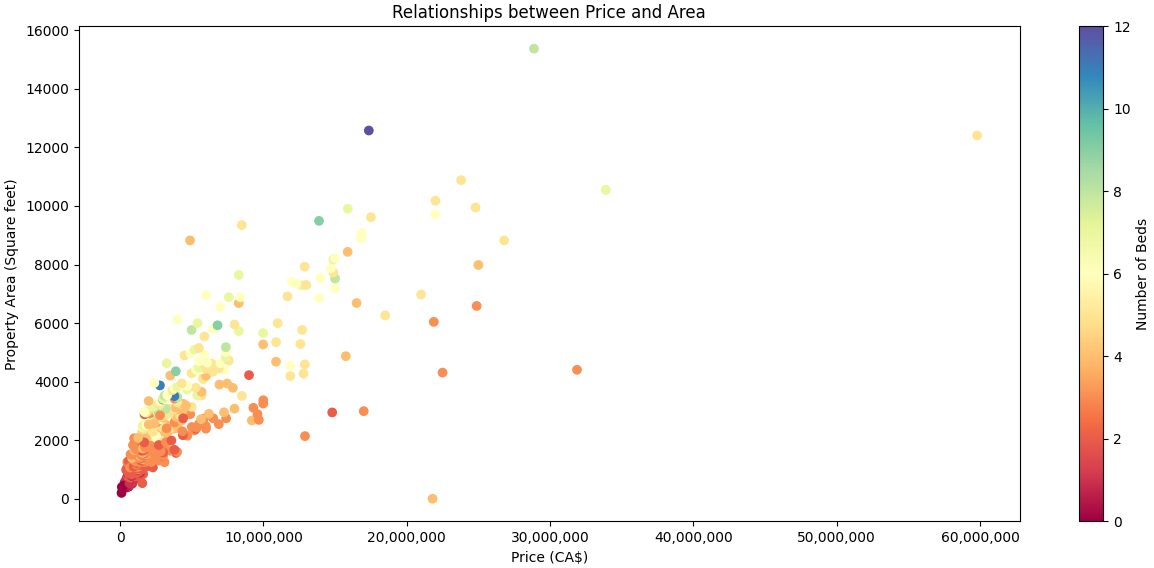
plt.xlabel('Price')

plt.ylabel('Property Area')

current\_values = plt.gca().get\_xticks()

plt.gca().set\_xticklabels(['{:,.0f}'.format(x) for x in current\_values])

plt.show()



**Scatter plot of distribution of price based of area and number of beds (Image by the author)**

Finally, let’s take a look to geographical property distribution in the Vancouver area. For this purpose I used folium library. I used different color for each category of property. As you can see, it increase performance of visualization considerably.

import pandas as pd

import folium

from folium.plugins import MarkerCluster    # Import folium MousePosition plugin

from folium.plugins import MousePosition    # Import folium DivIcon plugin

from folium.features import DivIcon

df = pd.read\_csv('zillow.csv')

drawable\_df = df[df.Lat > 0.0]

mapit = folium.Map( location=[49.261505,  -123.05453], zoom\_start=12 )

for index, drawable\_df in drawable\_df.iterrows():

    if drawable\_df["TypeofProperty"]=='Condo': c = 'blue'

    elif drawable\_df["TypeofProperty"]=='Townhouse': c = 'green'

    elif   drawable\_df["TypeofProperty"]=='Multi-family home': c = 'gray'

    else: c = 'red'

    folium.Marker(

                location = [drawable\_df['Lat'], drawable\_df['Long']],

                radius=15 ,

                icon=folium.Icon(color= c ) ,

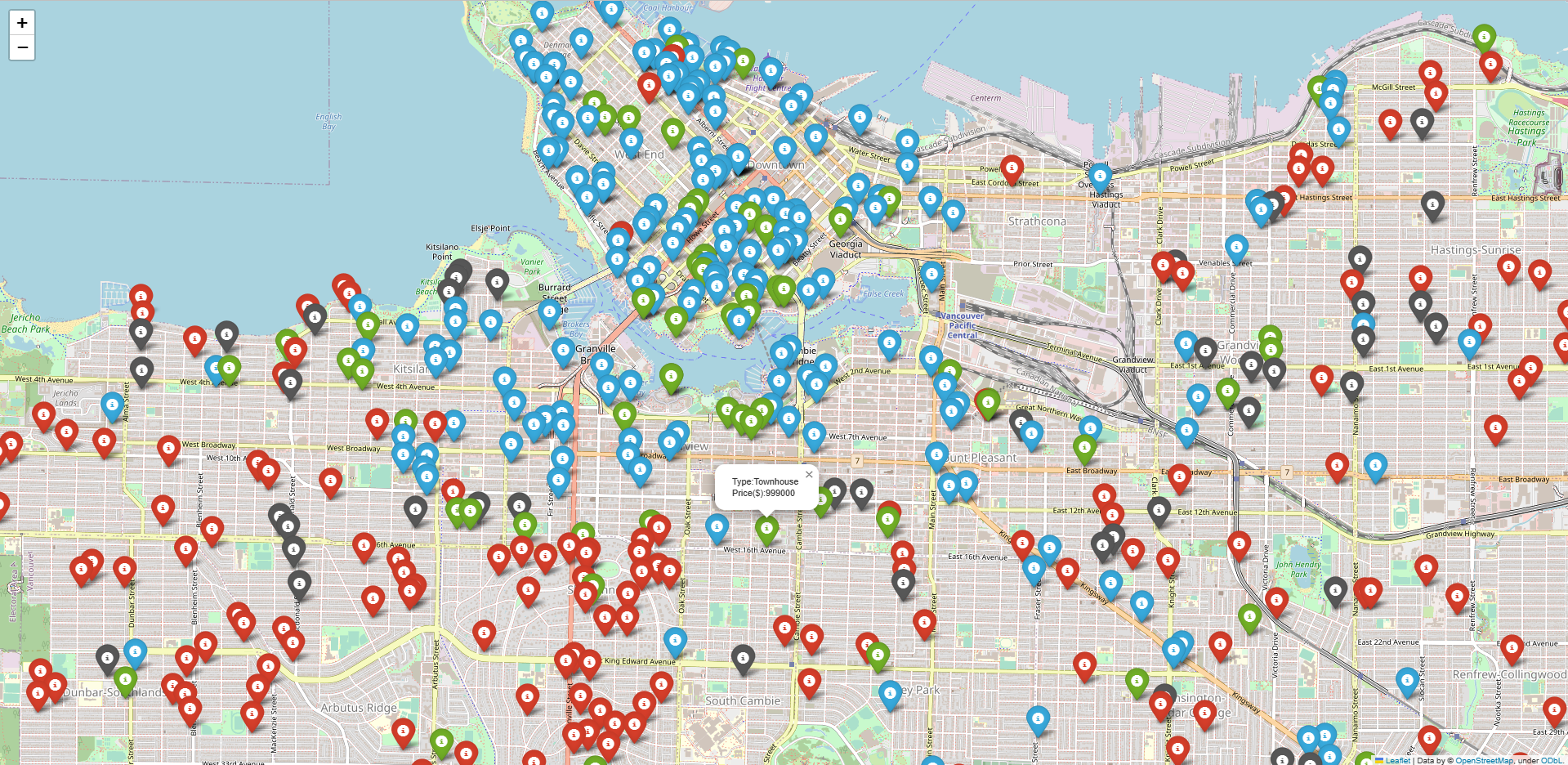
                popup = f'Type:{drawable\_df["TypeofProperty"]}\n Price($):{drawable\_df["Price"]}'

                ).add\_to(mapit)

mapit.save('map.html')

mapit.show\_in\_browser()

Result:



**Geographical property distribution in the Vancouver map (Image by the author)**

**Working with DataBase and SQL**

Lets create a sqlite database and table from CSVs

from pathlib import Path

properties = pd.read\_csv("zillow.csv")

Path('VancouverProperties.db').touch()

db\_conn = sqlite3.connect('VancouverProperties.db')

db\_cursor = db\_conn.cursor()

properties.to\_sql('VancouverPropertiesTable', db\_conn, if\_exists='append', index=False)

db\_vancouverproperties\_init\_query = pd.read\_sql(''' SELECT \* FROM VancouverPropertiesTable ''', db\_conn)

db\_vancouverproperties\_init\_query

index zpid id imgSrc ... latLong has3DModel brokerName best\_deal

0 614 2070285804 2070285804 https://photos.zillowstatic.com/fp/f67e9aa377a... ... {'latitude': 49.286182, 'longitude': -123.11589} 0 None 1058000

1 49 2066436476 2066436476 https://photos.zillowstatic.com/fp/b0831a45b64... ... {'latitude': 49.261505, 'longitude': -123.05453} 0 eXp Realty 419900

2 136 2067490635 2067490635 https://photos.zillowstatic.com/fp/76674f8faeb... ... {} 0 Park Georgia Realty Ltd. 438800

3 411 2060499969 2060499969 https://photos.zillowstatic.com/fp/85669420760... ... {'latitude': 49.243015, 'longitude': -123.05983} 0 Nu Stream Realty Inc. 449000

4 636 2060854306 2060854306 https://photos.zillowstatic.com/fp/91c1798b32d... ... {'latitude': 49.286224, 'longitude': -123.14085} 0 Team 3000 Realty Ltd. 599900

.. ... ... ... ... ... ... ... ... ...

795 323 314427630 314427630 https://photos.zillowstatic.com/fp/02c6a12b34b... ... {'latitude': 49.289257, 'longitude': -123.12097} 0 Macdonald Realty 4350000

796 201 2061702036 2061702036 https://photos.zillowstatic.com/fp/cc0fd71a6dd... ... {'latitude': 49.27208, 'longitude': -123.12038} 0 RE/MAX Crest Realty 3250000

797 750 314427938 314427938 https://photos.zillowstatic.com/fp/3dd1f4e2611... ... {'latitude': 49.274944, 'longitude': -123.12518} 0 Macdonald Realty 4398000

798 632 2071582184 2071582184 https://photos.zillowstatic.com/fp/47de178e771... ... {} 0 eXp Realty 7870000

799 549 314346775 314346775 https://photos.zillowstatic.com/fp/0d5302ad0c1... ... {'latitude': 49.256756, 'longitude': -123.13778} 0 Royal Pacific Realty Corp. 13888000

Next: create a sql query to select all properties in Kingsway Street

data\_stat = pd.read\_sql(''' SELECT zpid, TypeofProperty, Price, beds, addressStreet, area

                                     FROM VancouverPropertiesTable

                                     WHERE addressStreet LIKE '%Kingsway%'

                                     ORDER BY Price DESC ''', db\_conn)

data\_stat

Result:

0 2061711245 Condo 990000 3.0 2220 Kingsway #NE802 1140

1 2063168941 Condo 830000 2.0 2220 Kingsway #1612 812

2 2065888823 Condo 748888 2.0 760 Kingsway #310 880

3 2063497103 Condo 699000 2.0 2689 Kingsway #910 746

4 2060658439 Condo 659900 2.0 488 Kingsway #W407 780

5 2060941923 Townhouse 649800 1.0 2435 Kingsway #204 528

6 2060248755 Condo 618000 2.0 2973 Kingsway #102 861

7 2060723406 Condo 575000 1.0 1239 Kingsway #208 569

8 2060561370 Condo 459000 0.0 1432 Kingsway St #351 459

9 2060499969 Condo 449000 0.0 2239 Kingsway #109 415

Average condo price (CA$) and area(Square feet)

data\_stat\_Avg = pd.read\_sql(''' SELECT AVG(Price)AS Avg\_price\_condo , TypeofProperty, AVG(area) AS Avg\_area

                                     FROM VancouverPropertiesTable

                                     WHERE TypeofProperty == 'Condo'  ''', db\_conn)

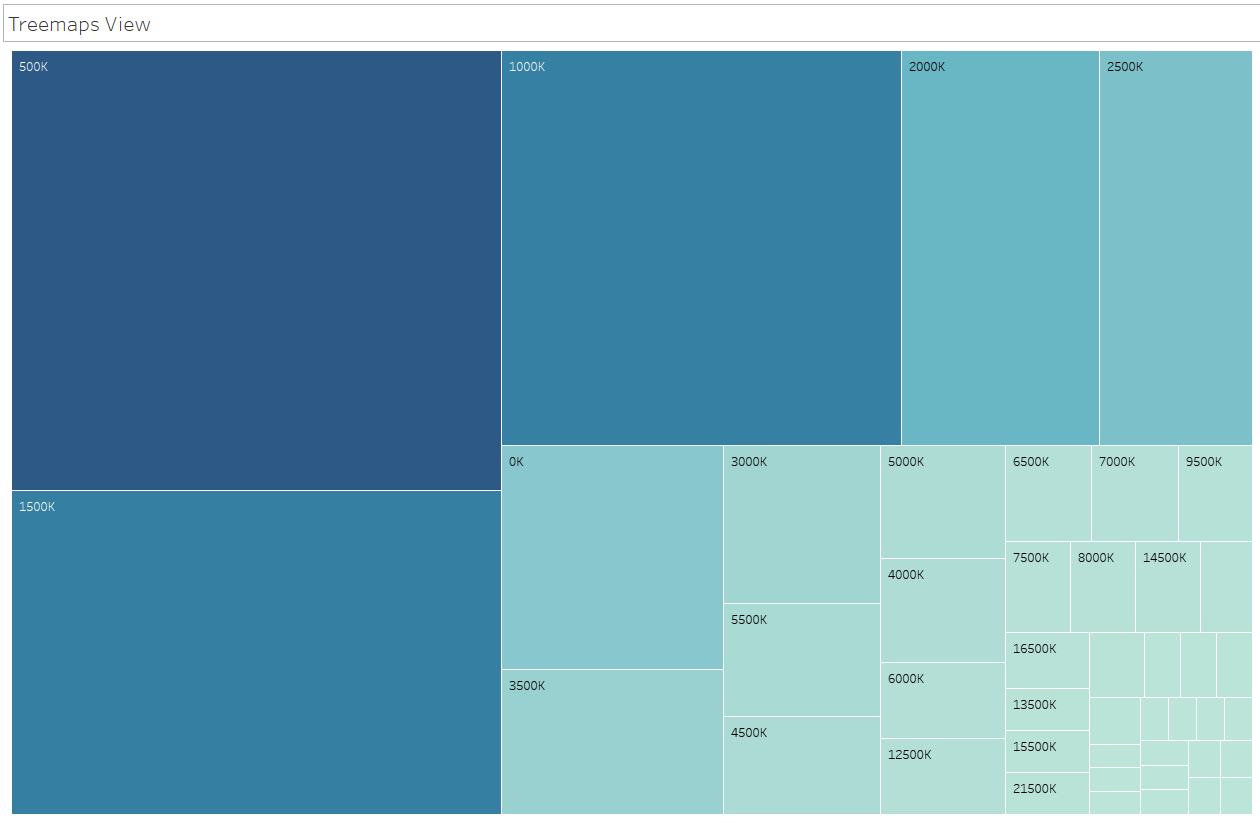
Result:

Avg\_price\_condo TypeofProperty Avg\_area

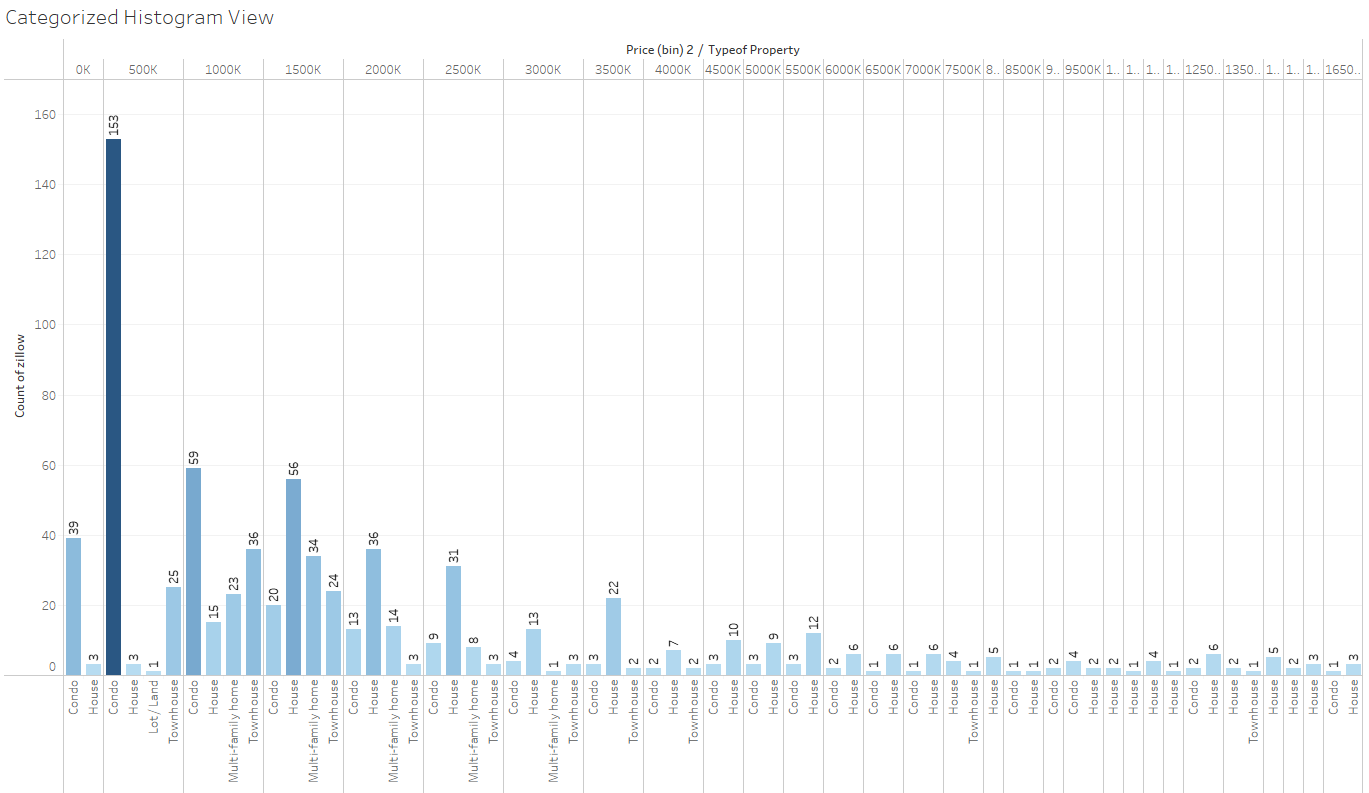
1.797490e+06 Condo 1129.859756

Property location Analysis with Folium

**Data visualization with Tableau**



**Treemap view shows the distribution based on price-range category (Image by the author)**



**Categorized histogram view shows the distribution based on price-type category (Image by the author)**

Link: [BCRealStateAnalysis | Tableau Public](https://public.tableau.com/app/profile/kasra.heidarinezhad/viz/BCRealStateAnalysis/Sheet1?publish=yes)

There are various tools and technologies available for creating a data pipeline, such as Apache NiFi, Apache Kafka, Apache Airflow and AWS Glue, etc. The choice of the tool depends on the specific needs of the organization and the type of data being processed.

⭐ HashTags ⭐

#Data\_pipline

#Data\_science

#Data\_mining

#web\_scraping

#beautifulsoup

#scraping

#web\_scraping\_python

#python\_beautifulsoup

#Python\_Tutorials

#Python\_web\_scraper\_tutorial

#Web\_Scraper\_Python

#octoparse

#python\_scrapy

#web\_crawler\_python

#Data\_Visualization

#Tableau