DALHOUSIE UNIVERSITY

Home Price Prediction

Arman Andisheh, Jafar Pashami

April 2022

Contents

1 Abstract

*Nowadays one a most crucial problems in Canada, especially in Halifax is housing. Lack of availability, exponential increasing prices, and dealing with a reasonable price are a couple of such matters. Considering this issue, We have come up with an idea which is combination of data mining and machine learning and housing. The main goal of our idea is firstly estimation of fair value, then predicting the price of properties. Title of the project is **"Home Price Prediction"**. It is a study for finding the attributes affecting on price of selling/renting a home and creating a machine learning based model for estimation and prediction of the price based on given input parameters.The geographical scope of the project will be Halifax, Nova Scotia, Canada.*

1.1 Project Goals

- Scraping Data from related advertising platform websites
- Data Prepossessing: Scraped data need to be cleans and transformed to become available for analytical works.
- Attribute Selection: Finding the most relevant attributes that affect the price
- Building a Regression Model for Price Prediction
- Applying the model on other real data to find the accuracy of the model

1.2 Data set Description

We will scrape data from selling/rental advertisements from the web. some websites contains these kind of advertisements we focused on Realtor.ca website.

2 Introduction

2.1 Problem Description

As most of us have touched, one of the most important issues these days, especially in Nova Scotia and Halifax, is housing.Lack of availability and exponential rising in prices are two major issues that cause people to have many problems in planning and budgeting to own a home. Also, Dealing with a fair price is another matter. The idea that came to our mind is to use data mining technique in this context and analyze and manipulate the data to achieve a useful and practical model, also providing comprehensives directory of hoses with reliable details.

2.2 Key Questions

After reviewing and digging into literature and based on our own experience we produced two main questions.

1. Is that possible to predict a reliable price through data mining techniques?
2. Which attributes do most affect the home price?

3 Related Works

We browsed various sources and deeply studied literature, and ultimately found several relatively similar examples. We also came across articles on the use of data mining and machine learning techniques in estimating home prices. For instance there are a couple of websites that provide home value estimation services, such as *RedFin* and *Zillow*.

Zillow is a useful starting point to help you determine an independent and unbiased assessment of what your home might be worth in today's market.They claim their error estimation is below 5%, also they have over 7.5 million properties advertisements. They analyze 58 attributes for a accurate price prediction. Zillow consider a $ 1 million award for minimizing logerror index.

RedFin has a complete and direct access to multiple listing services (MLSs), the databases that real estate agents use to list properties. They use MLS data on recently sold homes in your area to calculate your property's current market value. The team uses a combination of many different techniques to keep track of this complexity, including random forest and gradient boosting. Ensemble and hierarchical models are also present, like calculating a walk score and feeding that into the overall estimate. Since it's a fairly complex problem, the team has built a fairly complex model with a multitude of techniques to make the estimate as accurate as possible. The generate an updated Automating the Comparative Market Analysis (CMA) document, which is a process that real estate professionals use to determine the market value of a property by comparing it to similar properties that have recently sold, as well as to those currently listed for sale.

4 Methodology

To begin, we started to extract data through crawling and scraping techniques on Python and Selenium library.Then we did preprocessing step including cleaning the data on Python and Excel. We also carried out data mining and models evaluation on WEKA and final phase is Knowledge discovery and machine learning stage (figure 1).

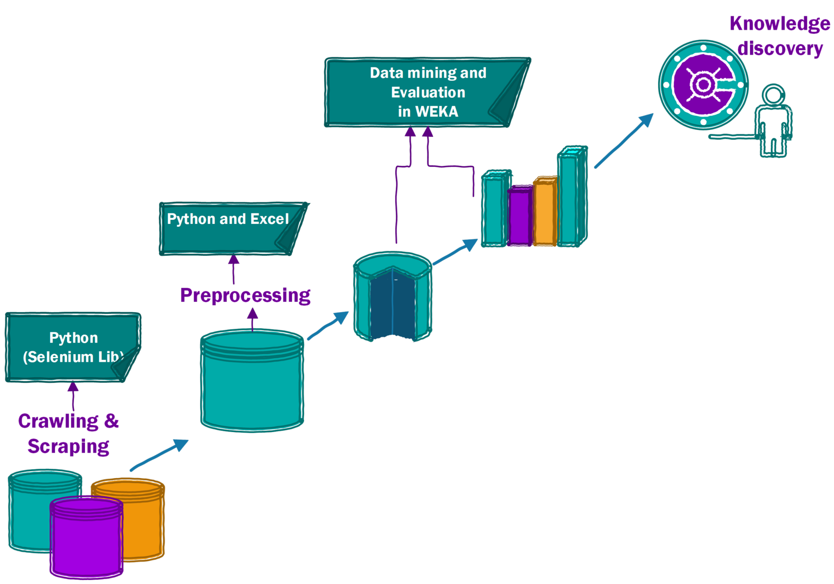


Figure 1: Methodology

5 Data set and Experimental setup

In preparing the data set, we investigated the websites of housing advertisements in Nova Scotia and Canada for scrapping data.Because some of these platforms, such as Kijiji, contain fewer variables and, importantly, did not have a coherent structure in data presentation, we decided to find a venue with a more cohesive design and multiple attributes.

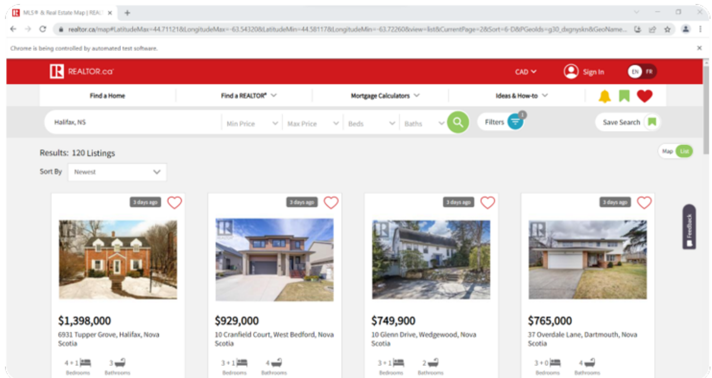


Figure 2: Realtor.ca search results

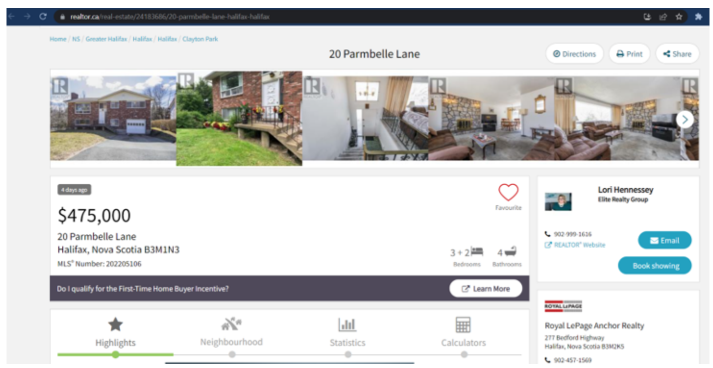


Figure 3: Realtor.ca post page

The websites chosen for the data collection was the website of Realtor.ca, which is owned by the Canadian Real Estate Association and receives more than 240 million views annually. This platform has a precisely defined structure for inserting ads and provides more variables from each ad. We collected about 20 variables and 510 selling posts in and around Halifax.

To scrape the data, we defined a two-step process. We collected all the links to posts about Halifax properties from realtor.ca website as the first step. Then, using the Selenium and Pandas libraries in Python, we collected each post's content and saved them in a CSV file. After that, it was needed to accomplish a data preprocessing stage to prepare data for analytical purposes. the tools we applied for data preprocessing were Python (Jupyter Notebook and Microsoft Excel). Data set contains 510 instances ith 20 attributes (7 numeric attributes and 13 nominal attributes). Name of the attributes are in the following:

index	Column	Non-Null Count	Dtype
1	Price	510 non-null	int64
2	Number of Bedrooms	510 non-null	int64
3	Number of Bedrooms including other rooms	510 non-null	int64
4	Number of Bathrooms	510 non-null	int64
5	Property Type	510 non-null	object
6	Building Type	509 non-null	object
7	Storeys	510 non-null	int64
8	Community Name	510 non-null	object
9	Title	510 non-null	object
10	Land Size	489 non-null	object
11	Built-in Date	404 non-null	float64
12	Parking Type	388 non-null	object
13	Total Finished Area	509 non-null	float64
14	Appliances Included	422 non-null	object
15	Foundation Type	479 non-null	object
16	style	448 non-null	object
17	Architecture Style	199 non-null	object
18	Basement Type	350 non-null	object
19	Postal Code (4 first digit)	510 non-null	object
20	Postal Code (3 first digit)	510 non-null	object

Selected attributes for modeling

The following picture shows the distribution of variables:

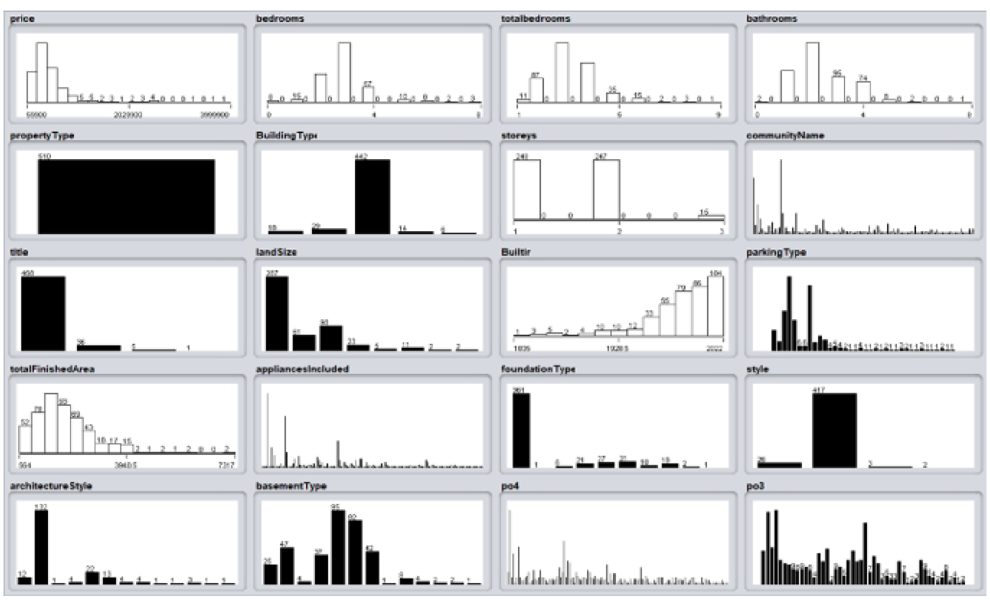


Figure 4: attributes' distributions

For understanding beter about data, using Seaborn library in Phyton we drawed PairPlot for numeric data. Here is the results:

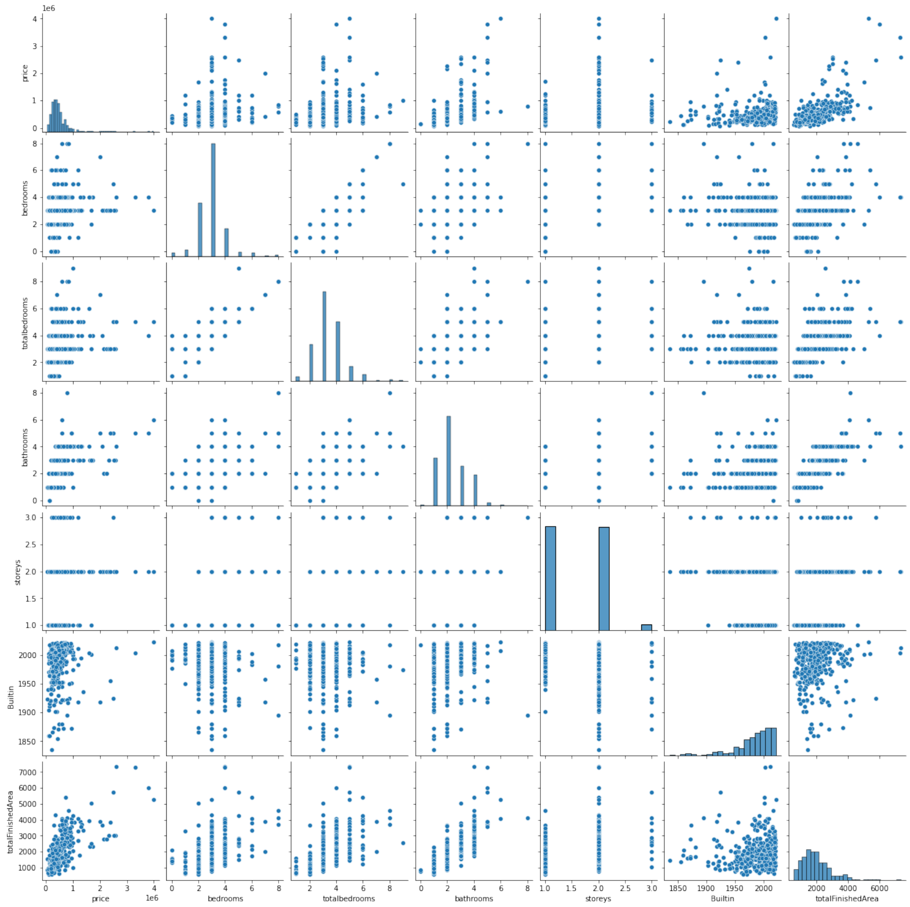


Figure 5: Pair Plot to show relation between numeric attributes

As you see in this figure (Pair Plot Chart), we can see the relationship between some attributes shown here. for example there is a strong relationship between Price and Total Finished Area or between price and number of bedrooms and bathrooms. It chart leads us to find some influential attributes on price attribute.

The following figure is a heat-map chart that demonstrates the correlation between numeric attributes by colors and numbers:

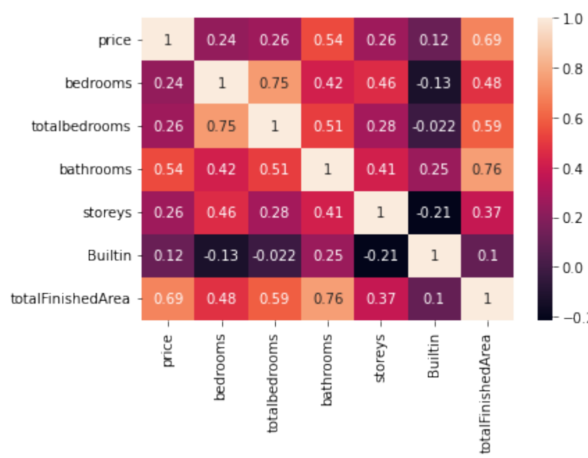


Figure 6: Heat map chart for illustrating correlation by color

As you can see here the is strong relationship between price and total finished area.
In Appendix 2 what we did for data cleaning and its steps is precisely explained in jupyter notebook.

6 Results

After preparing the data set we uploaded data to Weka as our machine learning tools. Because of the nature of our data set and our question we applied regression techniques on data.
Based on our literature review we selected four regression method to apply on data:

6.1 Linear Regression

After applying Linear regression on 20 attribute data set the results are below:

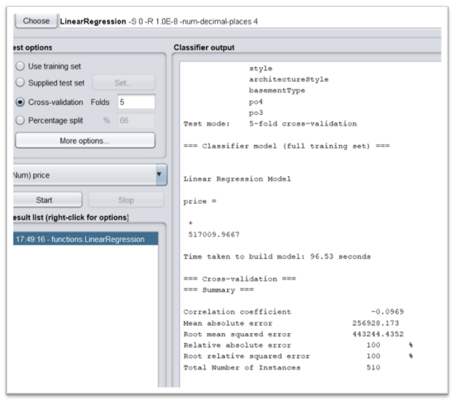


Figure 7: Linear regression results in Weka

In this picture as you see, Correlation coefficient is -0.0969 and Root relative squared error is 100% both of these indicator means that we couldn't find a meaningful model using regression.

6.2 SMO Regression

We applied SMO regression on our data set in Weka. the results are as below:

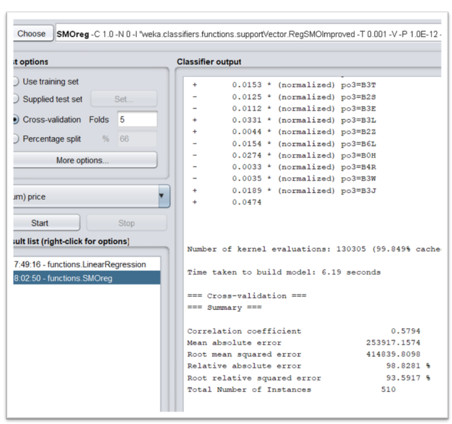


Figure 8: SMO regression results in Weka

In this figure, it is shown that Correlation coefficient is 0.58 and Root relative squared error is 93.59%. it is better in correlation coefficient however the error is still very high.

6.3 K-nearest neighbor's classifier

thee third algorithm that we applied on data was K-nearest neighbor. the results are as below:

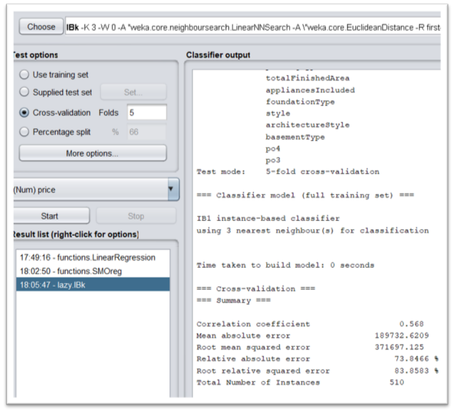


Figure 9: K-nearest neighbor's results in Weka

As the result, Correlation coefficient is 0.57 and Root relative squared error is about 83.86%. The method is also near to SMOreg with lower RRSe (Root Relative Square error).

6.4 Random Forrest classifier

thee forth algorithm was Random Forrest. the results are as below:

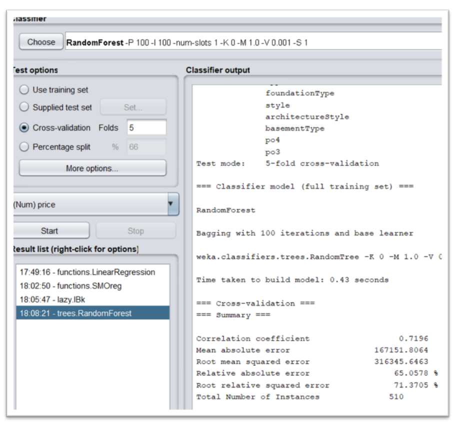


Figure 10: Random Forest results in Weka

As the result, Correlation coefficient is 0.72 and Root relative squared error is 71.37%. Random Forrest demonstrate much better performance between other three classification/Regression algorithms.

6.5 Models evaluation

After applying these four algorithm on our data set, now we tried to enhance our results by performing some changes in our data and attributes.

At first, we normalized data using normalizer filter in Weka. then we applied all algorithms again to see the results. as you can see normalization didn't make any significant changes in our results.

MODEL EVALUATION

Algorithm	Index	Base	Normalization	Removing imbalanced attributes (14 Attr.)	Literature Review (7 Attr.)
Random Forrest	Correlation coefficient	*0.72*	*0.72*	*0.72*	*0.72*
	Root mean squared error	*316345*	*0.08*	*0.08*	*0.08*
	Relative squared error	*71%*	*71%*	*72%*	*70%*
Lazy.IBK	Correlation coefficient	*0.57*	*0.57*	*0.6*	*0.45*
	Root mean squared error	*371697*	*0.09*	*0.09*	*0.1*
	Relative squared error	*83%*	*82.40%*	*81%*	*90%*
Linear Reg	Correlation coefficient	*-0.1*	*-0.1*	*0.72*	*0.74*
	Root mean squared error	*443244*	*0.11*	*0.08*	*0.08*
	Relative squared error	*100%*	*100%*	*73%*	*68%*
SMO Reg	Correlation coefficient	*0.58*	*0.58*	*0.72*	*0.73*
	Root mean squared error	*414839*	*0.11*	*0.08*	*0.08*
	Relative squared error	*94%*	*94%*	*70%*	*71%*

Figure 11: Model evaluation based on Correlation coef. and rrse in four examination using each algorithm

In the next step, we reduce the number of attributes from 20 attributes to 14 attributes by removeing imbalanced distribution of the attributes.for finding imbalanced attributes we looked at the distribution of each attribute then we decided to remove the attributes that were imballanced and also don't use in other studies we found in our litreature review.For learning more how to deal with imbalanced data please see the below links:
https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data
Following table indicates the remaining variables:

index	Column	Non-Null Count	Dtype
1	Price	510 non-null	int64
2	Number of Bedrooms	510 non-null	int64
3	Number of Bedrooms including other rooms	510 non-null	int64
4	Number of Bathrooms	510 non-null	int64
5	Building Type	509 non-null	object
6	Storeys	510 non-null	int64
7	Title	510 non-null	object
8	Land Size	489 non-null	object
9	Built-in Date	404 non-null	float64
10	Total Finished Area	509 non-null	float64
11	style	448 non-null	object
12	Basement Type	350 non-null	object
13	Postal Code (4 first digit)	510 non-null	object
14	Postal Code (3 first digit)	510 non-null	object

Selected attributes for modeling after attribute reduction

After reducing the attributes to 14 attributes we can see a significant improvement in Linear Regression results and also in time of executing and creating model. In addition, we saw enhancement in other algorithms results except random forest.

In the last step, again we reduced the number of attributes to 7 attributes and examine all four algorithms based on these 7 attributes. As you see, by decreasing more attributes, in some algorithms, there are slightly enhancement and on the other hand for k-nearest neighbor algorithm we got less accuracy and higher error. The list of seven remaining attributes are in the following table:

index	Column	Non-Null Count	Dtype
1	Price	510 non-null	int64
2	Number of Bedrooms	510 non-null	int64
3	Number of Bedrooms including other rooms	510 non-null	int64
4	Number of Bathrooms	510 non-null	int64
5	Built-in Date	404 non-null	float64
6	Total Finished Area	509 non-null	float64
7	Postal Code (3 first digit)	510 non-null	object

Selected attributes for modeling in last step

7 Conclusion and future works

In this study, which aimed to create a comprehensive platform of Halifax homes with the ability to estimate and predict prices, due to some time constraints in data collection, did not lead to the expected result or machine learning process.

Doing this project gave us interesting information that can be divided into two main parts: technical part and theoretical part. First, the issues of data mining and data collection are far more complex than they seem, and not merely a science dependent on tools and algorithms.

According to what we did in this study and limitations in terms of data and time, we have several suggestions to researchers in the future:

- By increasing the data as much as possible, the reliability of the models can be better, and would decrease the errors.
- Another factor that we think can improve outputs is the increase in the geographic area in which data is collected, would give us a more comprehensive insight.
- This study is based on sell and buy price, however digging into rental market could be another work in the future.
- A couple of macro variables such as fluctuation can be considered.
- One of the variables that can affect our independent variable is the time series attribute. That is, considering price changes over different periods.
- Social media impact: REVIEWS AND FEED BACKS and bring it into analyzing and price prediction

8 References

9 Appendix 1: Scarping codes

10 Appendix 2: Data Cleaning Instructions

11 Appendix 3: Data analysing and Data Visualization using python

Bibliography

Ade-Ojo, J. (2021, 01 16). *Predicting House Prices with Machine Learning*. (towardsdatascience.com) Retrieved from https://towardsdatascience.com/predicting-house-prices-with-machine-learning-62d5bcd0d68f

Hu, L., He, S., Han, Z., Xiao, H., Su, S., Weng, M., & Gai, Z. (2019). Monitoring housing rental prices based on social media- An integrated approach of machine-learning algorithms and hedonic modeling to informe quitable housing policies. *LandUsePolicy, 82*, 657-673.

Imbalanced Data. (n.d.). Retrieved from Developers.google: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Kang, Y., Fan, Z., Wenzhe, P., Gao, S., Rao, J., Duarte, F., & Ratti, C. (2021). Understanding house price appreciation using multi-source big geo-data and machine learning. *Land Use Policy, 111*(104919).

Kaul, A. (2021, 10 26). *House Hunters: Machine Learning at Redfin with Akshat Kaul*. (https://twimlai.com/) Retrieved from https://twimlai.com/house-hunters-machine-learning-at-redfin-with-akshat-kaul/

Shalloway, B. (2020, 08 17). *Linear Regression in Pricing Analysis, Essential Things to Know*. (https://www.bryanshalloway.com/) Retrieved from https://www.bryanshalloway.com/2020/08/17/pricing-insights-from-historical-data-part-1/

Tientcheu, D. (2021, 12 23). *Predicting House Prices on Zillow Using Machine Learning*. (https://blog.jovian.ai/) Retrieved from https://blog.jovian.ai/predicting-house-prices-on-zillow-using-machine-learning-bfa5b35f547b

Wang, C., & Wu, H. (2018). A new machine learning approach to house price estimation. *New Trends in Mathematical Sciences, 4*(6), 165-171.

Zillow Prize: Zillow's Home Value Prediction (Zestimate). (2018). (https://www.kaggle.com/) Retrieved from https://www.kaggle.com/competitions/zillow-prize-1/rules

Realtor_linkScrap3

April 6, 2022

1 Scraping link from Realtor.ca

[1]:
```
from json.tool import main
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import csv
from datetime import date
# Import writer class from csv module
from csv import writer
import pandas as pd
```

Opening Chrome web driver

[4]:
```
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)
#url1 = "https://www.realtor.ca/map#ZoomLevel=8&Center=44.595693%2C-61.
 ↪953830&LatitudeMax=45.64606&LongitudeMax=-58.65793&LatitudeMin=43.
 ↪52599&LongitudeMin=-65.
 ↪24973&view=list&Sort=6-D&PGeoIds=g30_dxgnyskn&GeoName=Halifax%20Regional%20Municipality%2C%20
url2 = 'https://www.realtor.ca/map#LatitudeMax=44.71121&LongitudeMax=-63.
 ↪54320&LatitudeMin=44.58117&LongitudeMin=-63.
 ↪72260&view=list&CurrentPage=2&Sort=6-D&PGeoIds=g30_dxgnyskn&GeoName=Halifax%2C%20NS&PropertyT
driver.get(url2)
```

```
C:\Users\JPASHAMI\AppData\Local\Temp/ipykernel_18140/1165801800.py:2:
DeprecationWarning: executable_path has been deprecated, please pass in a
Service object
  driver = webdriver.Chrome(PATH)
```

Initializing variables

[6]:
```
resultNum = int(driver.find_element(By.ID, 'listViewResultsNumVal').text)
pageNum = resultNum//12 + 1
print(resultNum, pageNum)
```

121 11

[7]:
```
list = []
page = 1
```

Main Scripts to collect link from search result of realtor.ca

[8]:
```
if page == 1:
    url = 'https://www.realtor.ca/map#LatitudeMax=44.71121&LongitudeMax=-63.
↪54320&LatitudeMin=44.58117&LongitudeMin=-63.
↪72260&view=list&Sort=6-D&PGeoIds=g30_dxgnyskn&GeoName=Halifax%2C%20NS&PropertyTypeGroupID=1&P
else:
    url_p1 = 'https://www.realtor.ca/map#LatitudeMax=44.71121&LongitudeMax=-63.
↪54320&LatitudeMin=44.58117&LongitudeMin=-63.72260&view=list&CurrentPage='
    url_p2 =␣
↪'&Sort=6-D&PGeoIds=g30_dxgnyskn&GeoName=Halifax%2C%20NS&PropertyTypeGroupID=1&PropertySearchT
    url = url_p1 + str(page) + url_p2
    #while page <= pageNum:
    #try:
driver.get(url)
WebDriverWait(driver, 10 )
element_present = EC.presence_of_element_located((By.ID, 'listViewFooter'))
WebDriverWait(driver, 30).until(element_present)
links_f = driver.find_elements(By.CLASS_NAME, 'blockLink.listingDetailsLink')
for link in links_f:
    link_url = link.get_attribute('href')
    #print(link_url)

    list.extend([link_url])
#print(page, url)
#print(len(list))
page += 1
driver.refresh()
#except:
    #    print('Can not load page number:', page)
    #    page += 1
```

Printing the result here:

[9]:
```
list
```

[9]:
```
['https://www.realtor.ca/real-estate/24226590/770-young-avenue-halifax-halifax',
 'https://www.realtor.ca/real-estate/24225856/3628-barrington-street-halifax-
halifax',
 'https://www.realtor.ca/real-estate/24225175/5052-shore-road-dartmouth-
dartmouth',
 'https://www.realtor.ca/real-estate/24224874/206-30-brookdale-crescent-
dartmouth-dartmouth',
```

```
 'https://www.realtor.ca/real-estate/24224870/103-wentworth-drive-halifax-
halifax',
 'https://www.realtor.ca/real-estate/24224360/lot-6-78-307-marketway-lane-
brunello-estates-timberlea-timberlea',
 'https://www.realtor.ca/real-estate/24223133/5870-merkel-street-halifax-
peninsula-halifax-peninsula',
 'https://www.realtor.ca/real-estate/24218110/55-hilden-drive-spryfield-
spryfield',
 'https://www.realtor.ca/real-estate/24217850/lot-1015-higgins-avenue-
beechville-beechville',
 'https://www.realtor.ca/real-estate/24216544/1710-oxford-street-halifax-
halifax',
 'https://www.realtor.ca/real-estate/24216276/203-94-bedros-lane-halifax-
halifax',
 'https://www.realtor.ca/real-estate/24216058/22-maple-grove-avenue-timberlea-
timberlea']
```

Checking the list to remove duplicate links:

[10]:
```
New_df = pd.DataFrame(list , columns=['url']).drop_duplicates()
New_df
```

[10]:

	url
0	https://www.realtor.ca/real-estate/24226590/77...
1	https://www.realtor.ca/real-estate/24225856/36...
2	https://www.realtor.ca/real-estate/24225175/50...
3	https://www.realtor.ca/real-estate/24224874/20...
4	https://www.realtor.ca/real-estate/24224870/10...
5	https://www.realtor.ca/real-estate/24224360/lo...
6	https://www.realtor.ca/real-estate/24223133/58...
7	https://www.realtor.ca/real-estate/24218110/55...
8	https://www.realtor.ca/real-estate/24217850/lo...
9	https://www.realtor.ca/real-estate/24216544/17...
10	https://www.realtor.ca/real-estate/24216276/20...
11	https://www.realtor.ca/real-estate/24216058/22...

[11]:
```
New_df.shape
```

[11]:
```
(12, 1)
```

After preparing the list of links we will write it to a CSV file:

[12]:
```
today = date.today()
csv_name = "Realtorlinks_test" + str(today) +".csv"
# find web links
New_df.to_csv(csv_name, index=None)
```

Finally, we close the chrome driver:

[13]:
```
driver.quit()
```

RealtorCa_SeleniumScaring_RealEstatePage_v3.6

April 6, 2022

1 Collecting the post's content data

Using this code we can open a Realtor post and scrape data from that web page. After that we converted the collected data to dataframe and finally append it as a row to a CSV file.

[1]:
```
from json.tool import main
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
from datetime import datetime
import csv
from csv import writer
import pandas as pd
import os
import wget
import sys
```

Opening the file that contains posts' link:

[2]:
```
# Open links file
df = pd. read_csv('Realtorlinks_2022-03-28.csv', header =None, usecols=[0])
df.head()
```

[2]:
```
                                                 0
0                                              url
1  https://www.realtor.ca/real-estate/24181975/10...
2  https://www.realtor.ca/real-estate/24181978/69...
3  https://www.realtor.ca/real-estate/24181886/10...
4  https://www.realtor.ca/real-estate/24181361/32...
```

Making a directory for downloading and saving images

[ ]:
```
# Make directory for downloading and saving images
path2 = os.getcwd()
path2 = os.path.join(path2 , 'homeimages')
```

```
os.mkdir(path2)
path2
```

[ ]:
```
# Just for manually setting start point of getting links
counter = 425
except_count = 0
```

Opening Chrome Driver

[ ]:
```
# Opening Chrome Driver
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)
```

The following part is the main body of the code. As you see we implemented a while loop that crwal of links and then open their URL. After opening each post, selected attributes been scraped and saved in a list. Then they save as a row in output file.

[ ]:
```
# Main Body
while counter < len(df):
    url = df.iat[counter,0]
    print(counter)
    print(url)
    driver.get(url)
    try:
        element_present = EC.presence_of_element_located((By.ID,␣
↪'listingDetailsTopCon'))
        WebDriverWait(driver, 30).until(element_present)
        now = datetime.now() # current date and time
        scrapeDatetime = now.strftime("%m/%d/%Y, %H:%M:%S")
        #print (scrapeDatetime)
        listingDetailsTopCon = driver.find_element(by=By.ID,␣
↪value="listingDetailsTopCon")
        ls = listingDetailsTopCon.text.split('\n')
        bedrooms = ''
        totalbedrooms = 0
        bathrooms = ''
        mls = ''
        price = ''
        addressline1 = ''
        addressline2 = ''
        po = ''
        for l in ls:
            if (l.find('$') != -1):
                i= ls.index(l)
                price = ls[i]
                addressline1 = ls[i+2]
            if (l.find('MLS') != -1):
                k = ls.index(l)
```

```
                mls = ls[k]
                addressline2 = ls[k-1]
                po = addressline2[-6:]
            if (l.find('Bedrooms') != -1):
                i = ls.index(l)
                bedrooms = ls[i-1]
                totalbedrooms = int(ls[i-1][0])+ int(ls[i-1][-1:])
            if (l.find('Bathrooms') != -1):
                i = ls.index(l)
                bathrooms = ls[i-1]
        #print(ls)
        obj = [scrapeDatetime, url, price, addressline1, addressline2, po, mls, bedrooms, totalbedrooms, bathrooms]
        #print(obj)
        PropertySummary = driver.find_element(by=By.ID, value="PropertySummary")
        ps = PropertySummary.text.split('\n')
        #print(ps)
        propertyType = ''
        BuildingType = ''
        storeys = ''
        communityName = ''
        title = ''
        landSize = ''
        Builtin = ''
        parkingType = ''
        publishTime = ''
        for item in ps:
            if item == 'Property Type':
                k = ps.index(item)
                propertyType = ps[k+1]
            if item == 'Building Type':
                k = ps.index(item)
                BuildingType = ps[k+1]
            if item == 'Storeys':
                k = ps.index(item)
                storeys = ps[k+1]
            if item == 'Community Name':
                k = ps.index(item)
                communityName  = ps[k+1]
            if item == 'Title':
                k = ps.index(item)
                title = ps[k+1]
            if item == 'Land Size':
                k = ps.index(item)
                landSize = ps[k+1]
            if item == 'Built in':
                k = ps.index(item)
```

```
                Builtin = ps[k+1]
            if item == 'Parking Type':
                k = ps.index(item)
                parkingType = ps[k+1]
            if item == 'Time on REALTOR.ca':
                k = ps.index(item)
                publishTime = ps[k+1]
        obj.extend([propertyType, BuildingType, storeys, communityName, title,␣
↪landSize, Builtin, parkingType, publishTime])
        #print(obj)
        try:
            PriceHistory = driver.find_element(by=By.ID,␣
↪value="historyDetailSection")
            ph = PriceHistory.text.split('\n')
            priceHistory = ph[2]
        except:
            ph[2] = ''
        #print(ph[2])
        obj.append(ph[2])
        listingDetailsBuildingCon = driver.find_element(by=By.ID,␣
↪value="listingDetailsBuildingCon")
        lsd = listingDetailsBuildingCon.text.split('\n')
        totalFinishedArea = ''
        appliancesIncluded = ''
        foundationType = ''
        style = ''
        architectureStyle = ''
        basementType = ''

        for item in lsd:
            if item == 'Total Finished Area':
                k = lsd.index(item)
                totalFinishedArea = lsd[k+1]
            if item == 'Appliances Included':
                k = lsd.index(item)
                appliancesIncluded = lsd[k+1]
            if item == 'Foundation Type':
                k = lsd.index(item)
                foundationType = lsd[k+1]
            if item == 'Style':
                k = lsd.index(item)
                style = lsd[k+1]
            if item == 'Architecture Style':
                k = lsd.index(item)
                architectureStyle = lsd[k+1]
            if item == 'Basement Type':
                k = lsd.index(item)
```

```
                basementType = lsd[k+1]
        obj.extend([totalFinishedArea, appliancesIncluded, foundationType,␣
↪style, architectureStyle, basementType])
        obj.extend(lsd)
        images = driver.find_elements(by=By.ID, value='propimg_1')
        images = [image.get_attribute('src') for image in images]
        #print(images)
        for image in images:
            imglnk = image.split('/')
            file_name = imglnk[len(imglnk)-1]
            #print(file_name)
            save_as = os.path.join(path2,  file_name)
            wget.download(image, save_as)
            obj.extend([file_name])
        counter = counter + 1
        print(obj)
        # writing in the CSV file
        csv_name = "Realtor_RealEstatePage_JP_S2.csv"
        with open(csv_name, 'a', newline='') as f_object:
            writer_object = writer(f_object)
            writer_object.writerow(obj)
            f_object.close()
        except_count = 0
    except Exception as e:
        print('Error:', e.__class__)
        except_count = except_count + 1
        print("Try number:",except_count)
        if except_count >= 3:
            counter = counter + 1
            print("Trys exceeded! Go to next link...")
```

To close Chrome driver:

[ ]:
```
# To close Chrome driver
driver.quit()
```

DataCleaning

April 6, 2022

1 The process of Data Cleaning and Data Transformation

First remove duplicates:

```
[ ]: #import pandas
     import pandas as pd
     import numpy as np
     import csv
```

```
[ ]: home_df = pd.read_csv('Home0_563.csv', encoding='utf-8').drop_duplicates()
```

```
[ ]: hdf = home_df.iloc[:,:26]
     len(hdf)
```

```
[ ]: hdf.url.duplicated().sum()
```

```
[ ]: hdf = hdf.drop_duplicates(subset='url', keep='last', ignore_index=True)
```

```
[ ]: display(hdf)
```

Extracting Price from Price field

```
[ ]: p = hdf['price'].str
     hdf['price'] = np.where(p.startswith('$'),p.replace(',','',regex=True), p)
     hdf['price'] = np.where(p.startswith('$'),p.replace(' ','',regex=True), p)
     hdf['price'] = np.where(p.startswith('$'),p.replace('$','',regex=True), p)
     print(hdf['price'])
```

```
[ ]: display(hdf.iloc[0,26])
```

```
[ ]: hdf.shape
```

Spliting the 3 frist digit from postal code and creating a new column

```
[ ]: count = 0
     hdf['po3'] = hdf['po']
     for x in range(510):
         s = hdf.iloc[count][5]
         hdf['po3'][count] = s[:3]
```

```
    print(count, s[:3])
    count = count +1
print(hdf['po3'])
```

[ ]:
```
display(hdf)
```

Extracting Total finished Area from related field

[ ]:
```
p = hdf['totalFinishedArea'].str
hdf['totalFinishedArea'] = np.where(p.endswith('sqft'),p.replace('␣
↪sqft','',regex=True), p)
```

Extracting Price History for calculating annual growth rate (CAGR)

[ ]:
```
py = hdf['PriceHistory'][0]
py = py[py.find('$')+1:]
py = py.replace(',','')
py_int = int(py)
print(py_int)
```

[ ]:
```
print(hdf.iloc[0][19])
```

[ ]:
```
y = hdf['PriceHistory'][0]
loc = y.find('Sold')
y = y[loc-5:loc-1]
y_int = int(y)
print(loc)
print(y_int)
```

[ ]:
```
pn_int = int(hdf['price'][0])
print(pn_int)
```

[ ]:
```
CAGR = (((pn_int/py_int)**(1/(2022-y_int)))-1)
print(CAGR)
```

[ ]:
```
count = 0
py = ''
y = ''
pn = ''
CAGR = np.zeros(510)
while count<510:
    s = str(hdf['PriceHistory'][count])
    if (s != 'nan'):
        if s.find('$') != -1:
            py = s[s.find('$')+1:]
            py = py.replace(' (CAD)','')
            py = py.replace(',','')
            py_int = int(py)
```

```
            loc = s.find('Sold')
            y = s[loc-5:loc-1]
            y_int = int(y)
            pn_int = int(hdf['price'][count])
            CAGR[count] = (((pn_int/py_int)**(1/(2022-y_int)))-1)
    else:
        py = ''
        py_int = 0
        pn = ''
        pn_int = 0
        y = '0'
        y_int = 0
    print([count, s, py_int , pn_int, y_int, CAGR[count]])
    count = count +1
```

[ ]:
```
hdf['CAGR'] = CAGR
print(hdf['CAGR'])
```

[ ]:
```
hdf.head()
```

Write the result to a new file

[ ]:
```
hdf.to_csv('home0_510.csv', index=False)
```

Home Price Estimator_v1

April 6, 2022

1 Data Analysis and Data Visualization of the Data set using Python

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
# importing OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
```

Loading Data from CSV file as output of scraping process

[2]:
```
home_df = pd.read_csv('home0_510.csv',␣
 ↪usecols=['price','bedrooms','totalbedrooms','bathrooms','propertyType','BuildingType','storey
 ↪'landSize', 'Builtin' , 'parkingType', 'totalFinishedArea' ,␣
 ↪'appliancesIncluded', 'foundationType','style',␣
 ↪'architectureStyle','basementType','po4','po3']).drop_duplicates()
home_df.head()
```

[2]:
```
    price  bedrooms  totalbedrooms  bathrooms   propertyType     BuildingType  \
0  399900         3              3          3  Single Family  Row / Townhouse
1  329900         2              2          1  Single Family        Apartment
2  165000         3              4          2  Single Family            House
3  339000         3              3          2  Single Family            House
4  750000         3              4          3  Single Family            House

   storeys     communityName               title        landSize  Builtin  \
0        2         Dartmouth            Freehold  under 1/2 acre   2010.0
1        1         Dartmouth  Condominium/Strata  under 1/2 acre   2012.0
2        2          Kingston            Freehold  under 1/2 acre   1995.0
3        2           Bedford            Freehold  under 1/2 acre   1980.0
4        2  Robinsons Corner            Freehold  under 1/2 acre   2020.0

              parkingType  totalFinishedArea  \
0                  Garage             1815.0
1      Garage, Underground              986.0
2                     NaN             1400.0
```

```
3                     NaN             1470.0
4  Garage, Detached Garage             3120.0

                               appliancesIncluded   foundationType  \
0  Stove, Dishwasher, Washer, Microwave Range Hoo...  Poured Concrete
1                                          Intercom  Poured Concrete
2  Stove, Dryer, Washer, Microwave, Microwave Ran...  Poured Concrete
3    Stove, Dishwasher, Dryer, Washer, Refrigerator  Poured Concrete
4  Cooktop - Electric, Oven, Dishwasher, Dryer, W...  Poured Concrete

          style architectureStyle              basementType   po4  po3
0           NaN               NaN                       NaN  B2W0  B2W
1           NaN               NaN                      Full  B2W0  B2W
2  Semi-detached           3 Level  Full (Partially finished)  B0P1  B0P
3  Semi-detached               NaN                       NaN  B4A1  B4A
4       Detached               NaN                       NaN  B0J1  B0J
```

Extracting object features of Data set

[3]:
```python
# checking features
cat = home_df.select_dtypes(include='O').keys()
# display variabels
#cat.shape
home_df_obj =  home_df.select_dtypes(include='O')
display(home_df_obj.head())
```

```
    propertyType     BuildingType     communityName               title  \
0  Single Family  Row / Townhouse         Dartmouth            Freehold
1  Single Family        Apartment         Dartmouth  Condominium/Strata
2  Single Family            House          Kingston            Freehold
3  Single Family            House           Bedford            Freehold
4  Single Family            House  Robinsons Corner            Freehold

         landSize              parkingType  \
0  under 1/2 acre                   Garage
1  under 1/2 acre      Garage, Underground
2  under 1/2 acre                      NaN
3  under 1/2 acre                      NaN
4  under 1/2 acre  Garage, Detached Garage

                               appliancesIncluded   foundationType  \
0  Stove, Dishwasher, Washer, Microwave Range Hoo...  Poured Concrete
1                                          Intercom  Poured Concrete
2  Stove, Dryer, Washer, Microwave, Microwave Ran...  Poured Concrete
3    Stove, Dishwasher, Dryer, Washer, Refrigerator  Poured Concrete
4  Cooktop - Electric, Oven, Dishwasher, Dryer, W...  Poured Concrete

          style architectureStyle              basementType   po4  po3
```

```
0           NaN              NaN                          NaN  B2W0  B2W
1           NaN              NaN                         Full  B2W0  B2W
2  Semi-detached         3 Level  Full (Partially finished)  B0P1  B0P
3  Semi-detached             NaN                          NaN  B4A1  B4A
4       Detached             NaN                          NaN  B0J1  B0J
```

Extracting numeric features of Data set

[4]:
```
home_df_numerics =  home_df.select_dtypes(include=['int', 'float'])
display(home_df_numerics.head())
```

```
    price  bedrooms  totalbedrooms  bathrooms  storeys  Builtin  \
0  399900         3              3          3        2   2010.0
1  329900         2              2          1        1   2012.0
2  165000         3              4          2        2   1995.0
3  339000         3              3          2        2   1980.0
4  750000         3              4          3        2   2020.0

   totalFinishedArea
0             1815.0
1              986.0
2             1400.0
3             1470.0
4             3120.0
```

Information about Data set attributes

[5]:
```
home_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 0 to 509
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   price               510 non-null    int64
 1   bedrooms            510 non-null    int64
 2   totalbedrooms       510 non-null    int64
 3   bathrooms           510 non-null    int64
 4   propertyType        510 non-null    object
 5   BuildingType        509 non-null    object
 6   storeys             510 non-null    int64
 7   communityName       510 non-null    object
 8   title               510 non-null    object
 9   landSize            489 non-null    object
 10  Builtin             404 non-null    float64
 11  parkingType         388 non-null    object
 12  totalFinishedArea   509 non-null    float64
 13  appliancesIncluded  422 non-null    object
 14  foundationType      479 non-null    object
 15  style               448 non-null    object
```

```
 16  architectureStyle   199 non-null    object
 17  basementType        350 non-null    object
 18  po4                 510 non-null    object
 19  po3                 510 non-null    object
dtypes: float64(2), int64(5), object(13)
memory usage: 83.7+ KB
```

Data set description (Numeric attributes)

[6]:
```
home_df.describe()
```

[6]:
```
              price    bedrooms  totalbedrooms   bathrooms     storeys  \
count  5.100000e+02  510.000000     510.000000  510.000000  510.000000
mean   5.170100e+05    2.905882       3.347059    2.319608    1.543137
std    4.425132e+05    1.031415       1.112150    1.086535    0.554586
min    5.990000e+04    0.000000       1.000000    0.000000    1.000000
25%    2.899250e+05    2.000000       3.000000    2.000000    1.000000
50%    4.000000e+05    3.000000       3.000000    2.000000    2.000000
75%    5.996750e+05    3.000000       4.000000    3.000000    2.000000
max    3.999900e+06    8.000000       9.000000    8.000000    3.000000

           Builtin  totalFinishedArea
count   404.000000         509.000000
mean   1983.967822        1970.322200
std      33.173451         933.624424
min    1835.000000         564.000000
25%    1972.000000        1344.000000
50%    1991.500000        1800.000000
75%    2008.000000        2400.000000
max    2022.000000        7317.000000
```

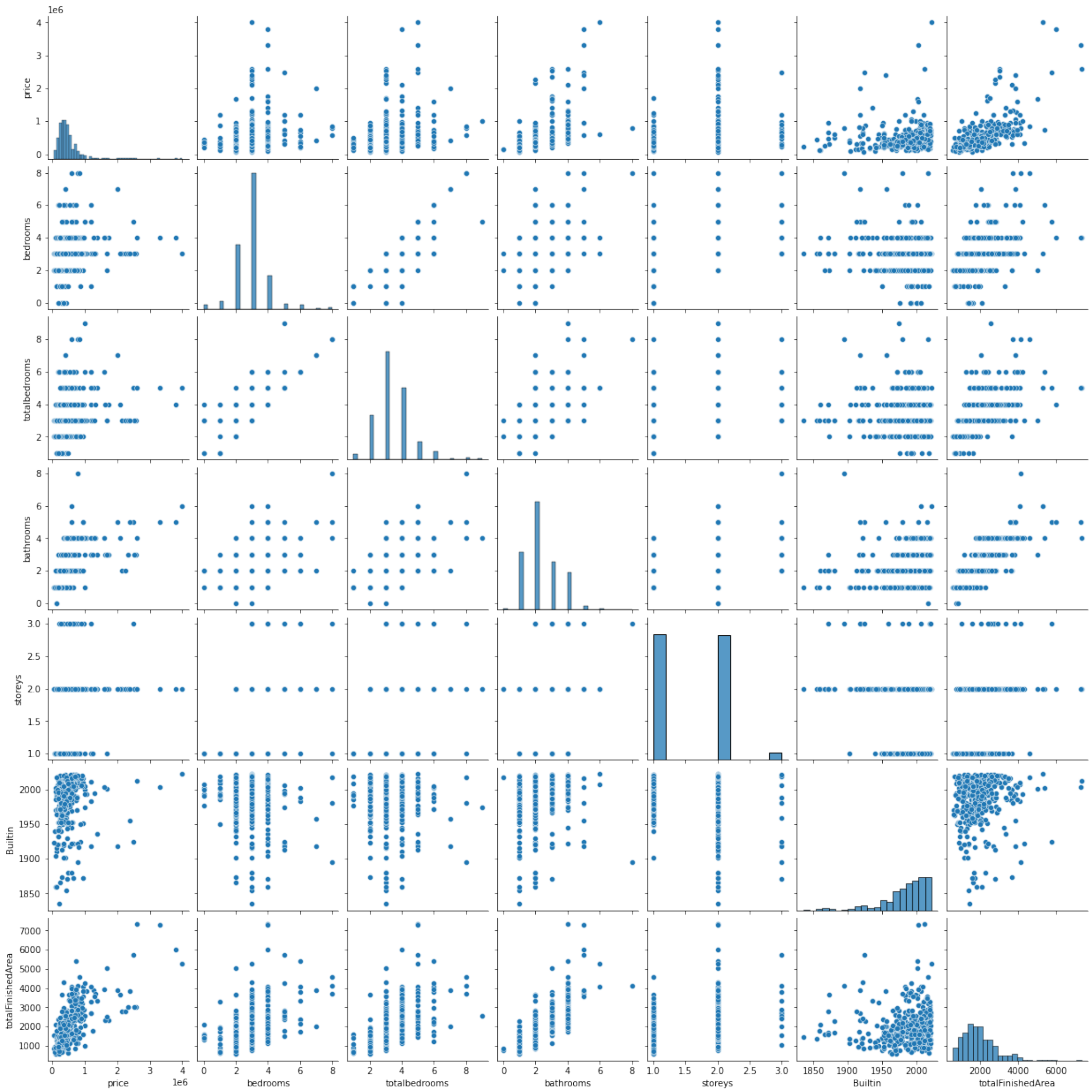
[7]:
```
home_df.shape
```

[7]:
```
(510, 20)
```

Using Pair Plot to compare attributes

[8]:
```
sns.pairplot(home_df)
```

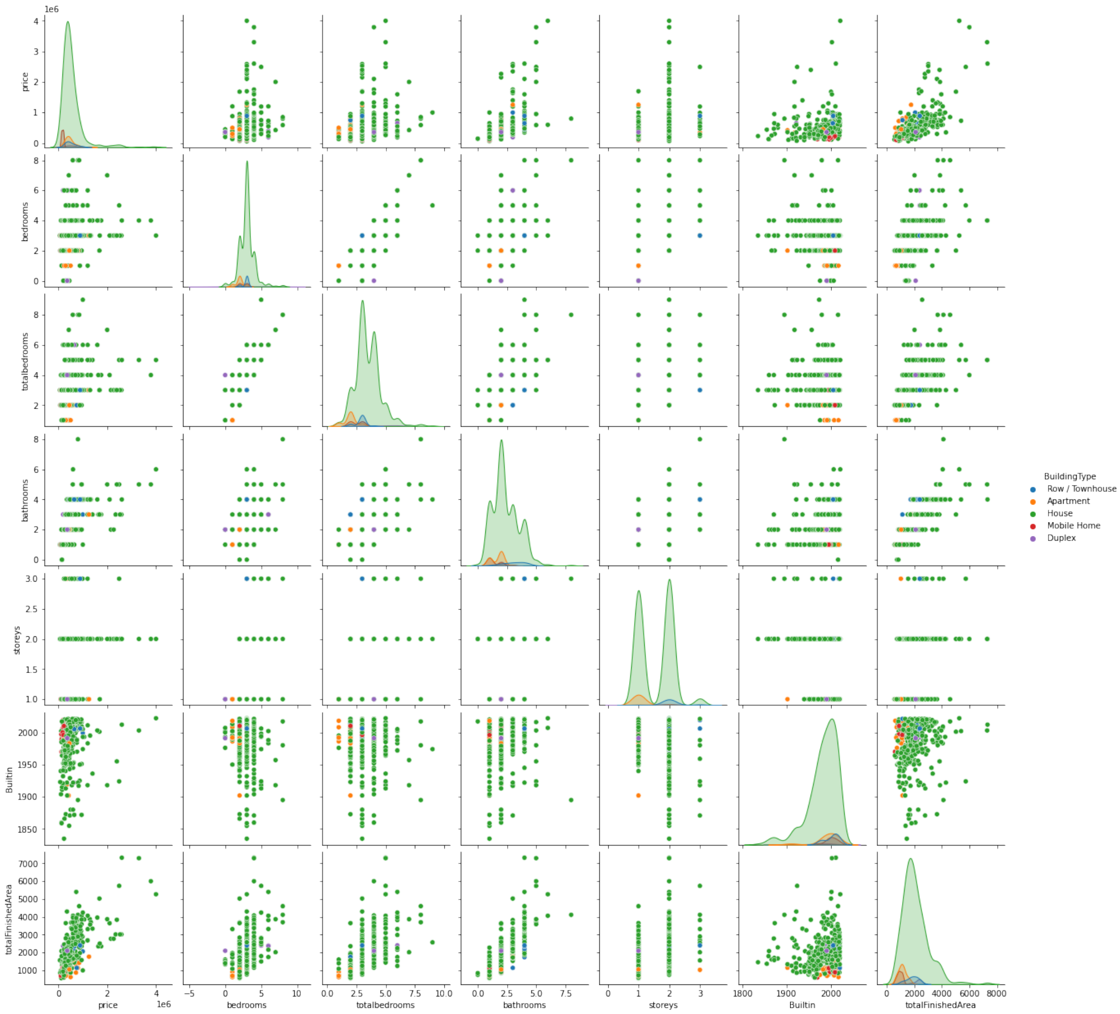
[8]:
```
<seaborn.axisgrid.PairGrid at 0x1ee11639be0>
```



Using Pair Plot to compare attributes with a hue attribute

[9]:
```
sns.pairplot(home_df , hue="BuildingType")
```

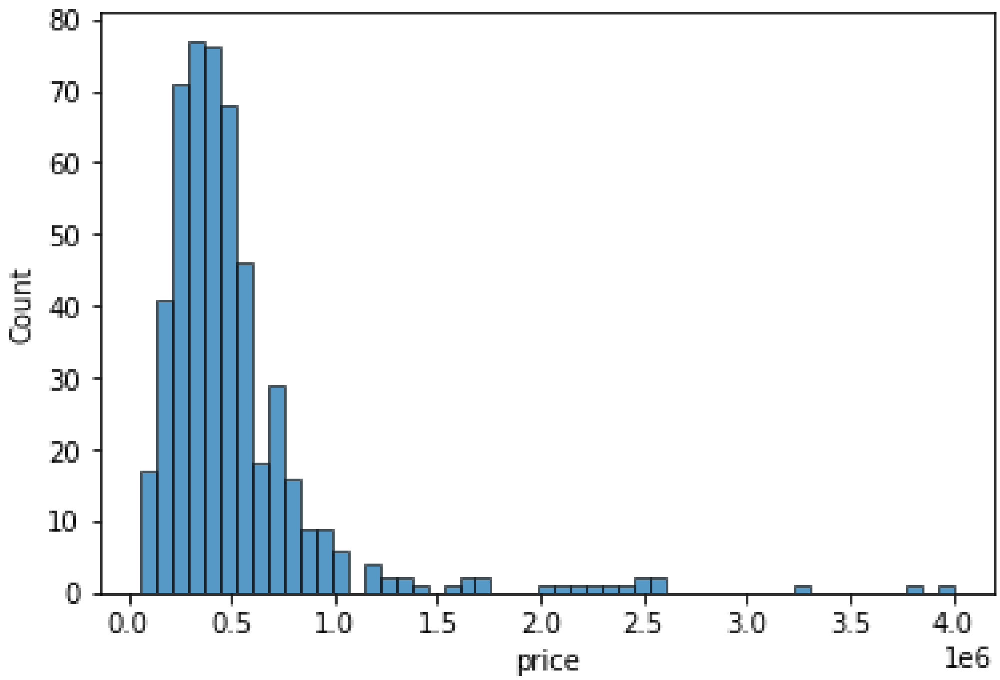
[9]: <seaborn.axisgrid.PairGrid at 0x1ee147ac130>



Distribution plot for variables

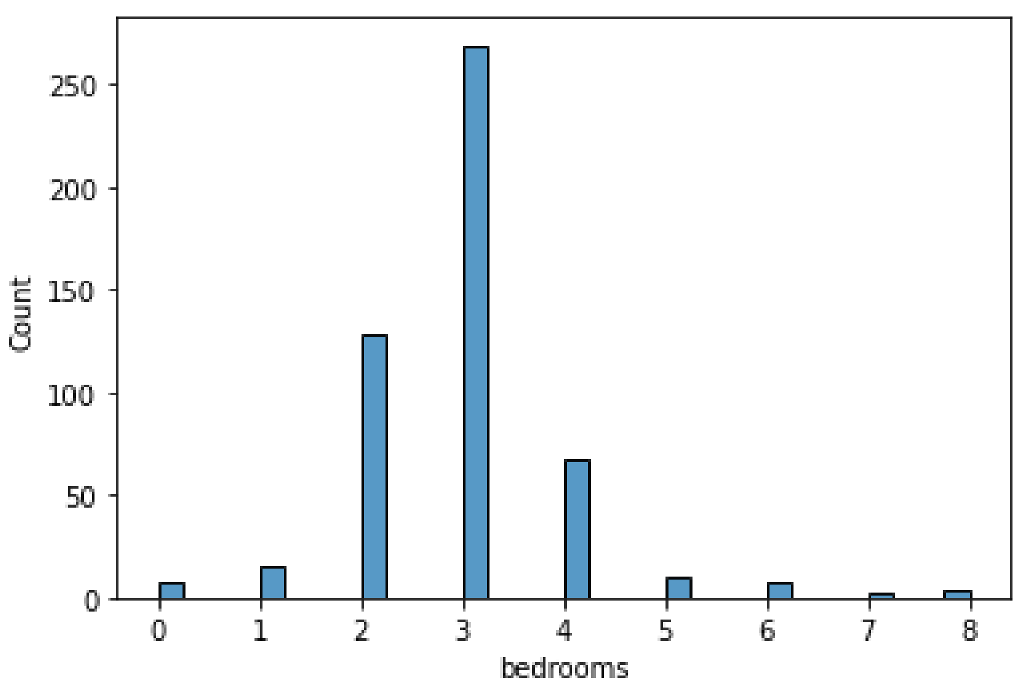
[10]:
```
sns.histplot(home_df['price'])
```

[10]: `<AxesSubplot:xlabel='price', ylabel='Count'>`



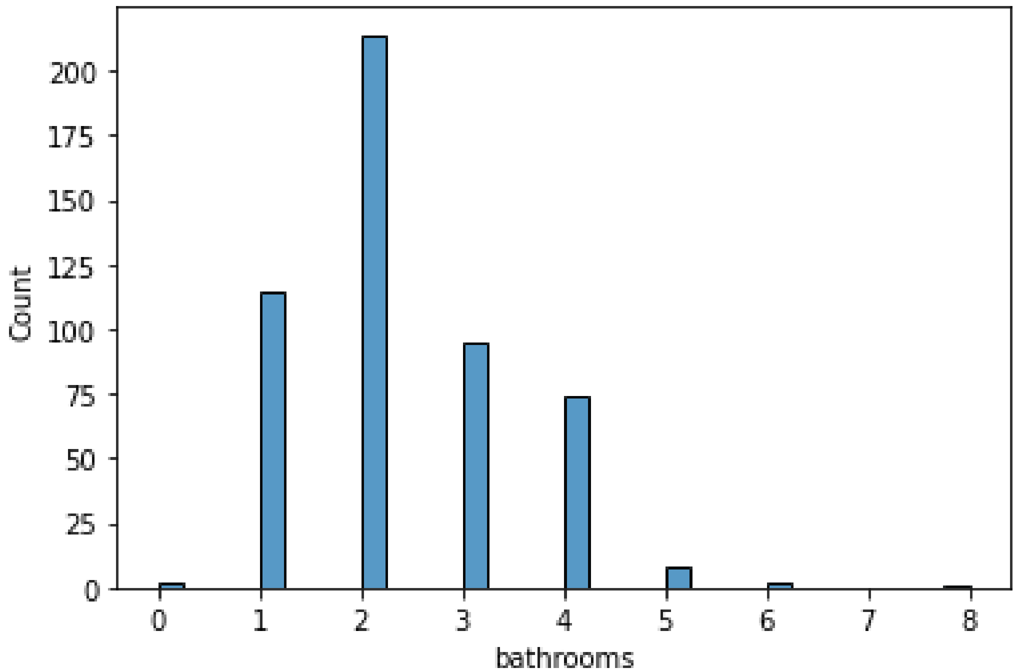
[11]:
```
sns.histplot(home_df['bedrooms'])
```

[11]: <AxesSubplot:xlabel='bedrooms', ylabel='Count'>



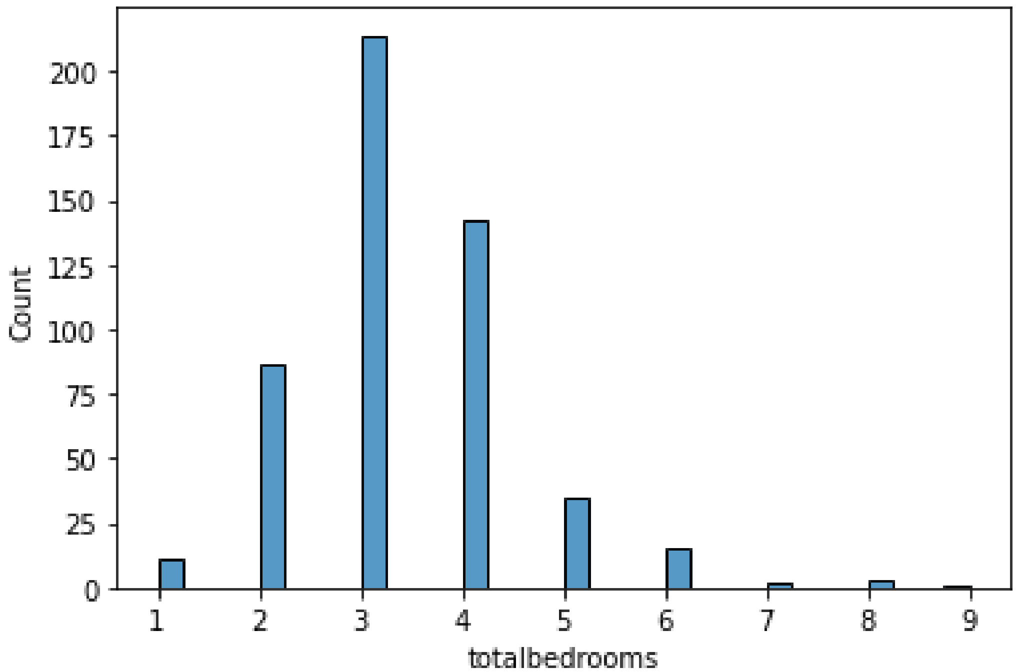
[12]:
```
sns.histplot(home_df['bathrooms'])
```

[12]: <AxesSubplot:xlabel='bathrooms', ylabel='Count'>



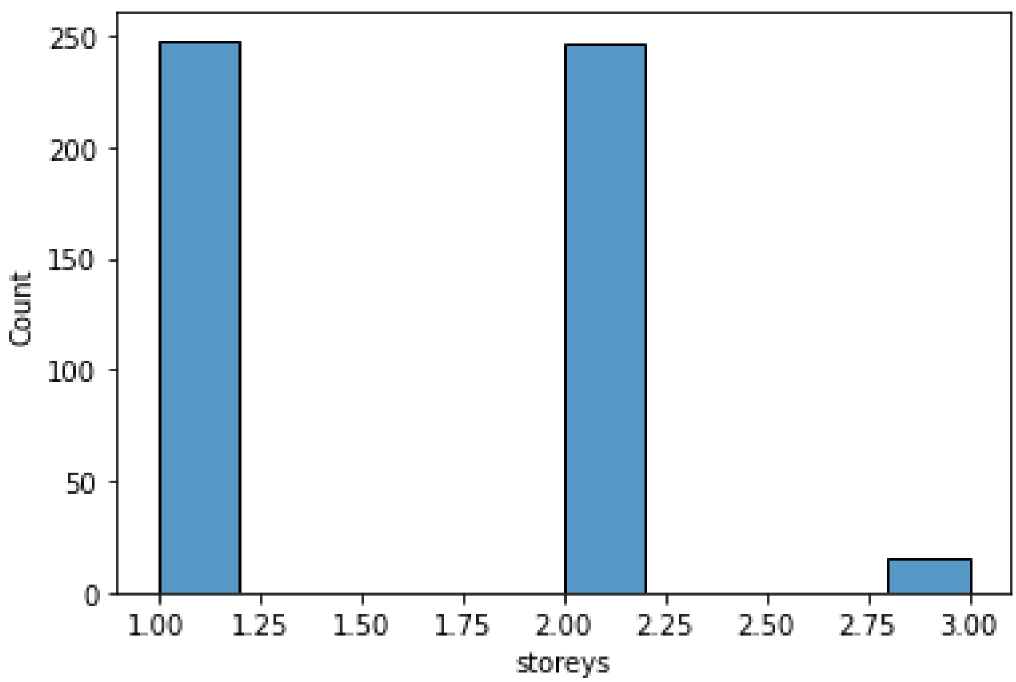
[13]:
```
sns.histplot(home_df['totalbedrooms'])
```

[13]: <AxesSubplot:xlabel='totalbedrooms', ylabel='Count'>



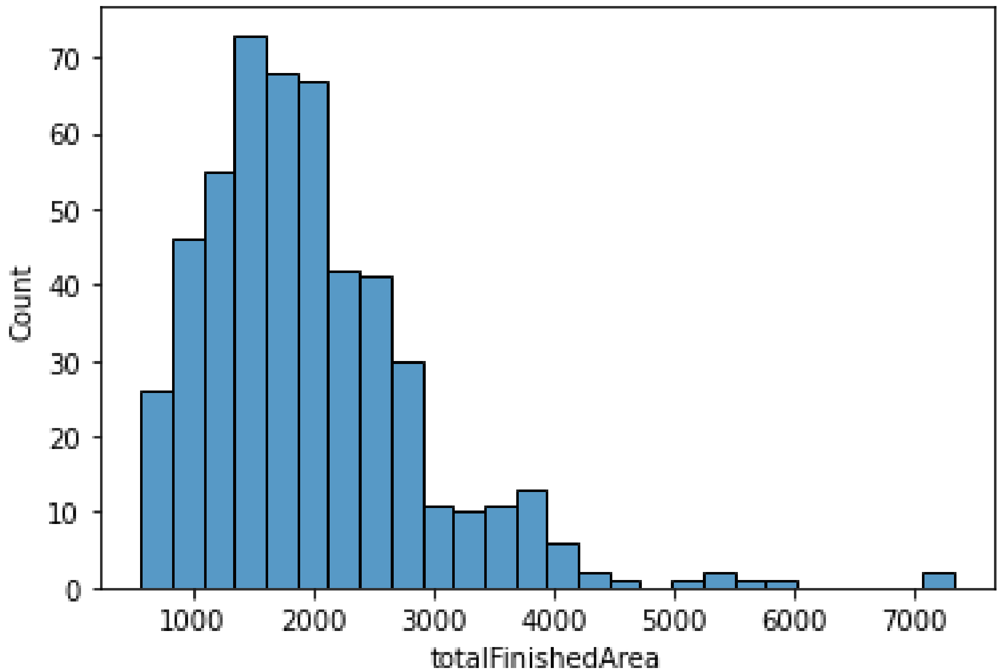
[14]:
```
sns.histplot(home_df['storeys'])
```

[14]: <AxesSubplot:xlabel='storeys', ylabel='Count'>



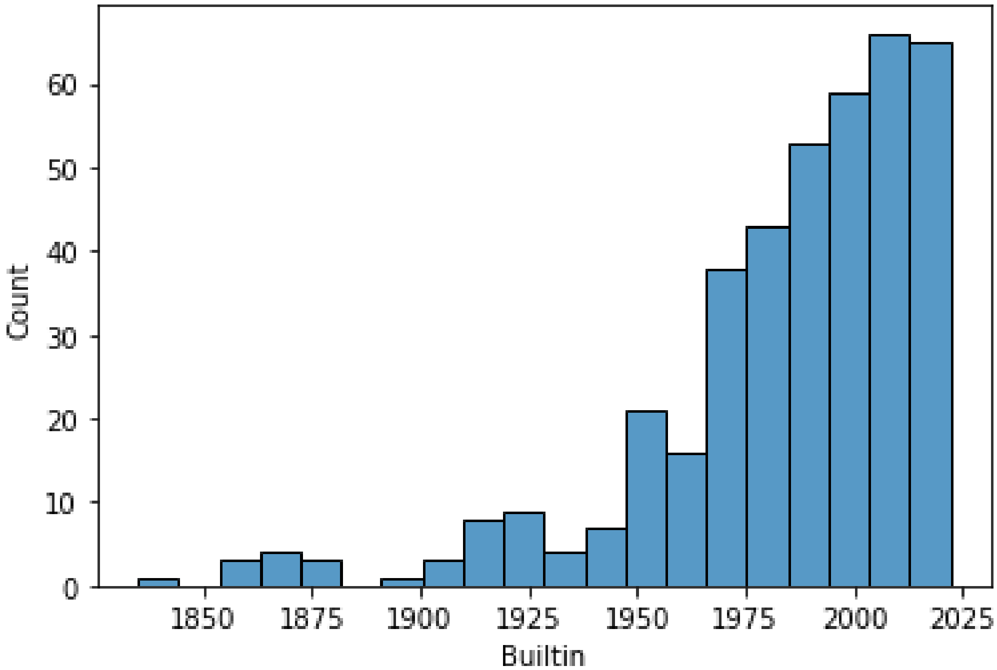
[15]:
```
sns.histplot(home_df['totalFinishedArea'])
```

[15]: <AxesSubplot:xlabel='totalFinishedArea', ylabel='Count'>



[16]:
```
sns.histplot(home_df['Builtin'])
```

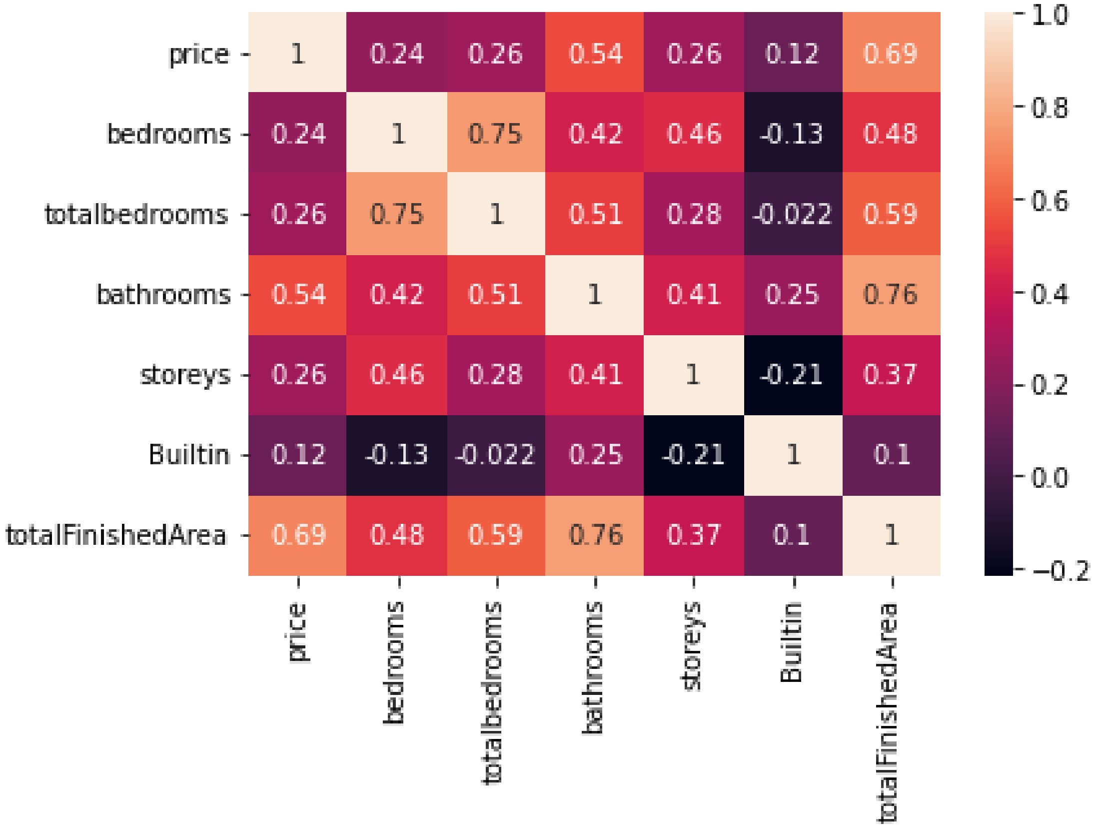
[16]: <AxesSubplot:xlabel='Builtin', ylabel='Count'>



Heat Map chart to show the relation between attributes

[17]:
```
sns.heatmap(home_df.corr(), annot = True)
```

[17]: <AxesSubplot:>


price
bedrooms
totalbedrooms
bathrooms
storeys
Builtin
totalFinishedArea
1 0.24 0.26 0.54 0.26 0.12 0.69
0.24 1 0.75 0.42 0.46 -0.13 0.48
0.26 0.75 1 0.51 0.28 -0.022 0.59
0.54 0.42 0.51 1 0.41 0.25 0.76
0.26 0.46 0.28 0.41 1 -0.21 0.37
0.12 -0.13 -0.022 0.25 -0.21 1 0.1
0.69 0.48 0.59 0.76 0.37 0.1 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2