



Explanatory And Predictive Analysis of Real Estate Listings on Redfin

Web Analytics (BYGB-7978-003) | Fall 2020 | Instructor: Professor Apostolos Filippas

Team 1 :

Yuge Ma

Prannoiy Chandran

Shu Guan

Muhetear Mayila

1. Executive Summary

Redfin is a real estate brokerage that offers property search capabilities online. It is widely used by landlords, brokers, buyers and renters to carry out research and make decisions pertaining to real estate. The wealth of data available on the website offers many opportunities for deriving data-backed insights that can aid in maximizing returns on investment and optimizing a listing to sell at a desired price range.

The purpose of this project was to help prospective landlords and investors (who intend to buy or invest in a property on Redfin) understand what drives profitability in the New York City market. The approach was to uncover trends from historical data and apply these insights to active listings. The goal was to maximize returns, identify features with a significant impact on selling price and uncover opportunities in both fast-growing neighborhoods, where landlords can benefit from being among the first to cater to high demand.

Web analytics techniques were used to automate the collection of accurate and current data from Redfin's website. Redfin's anti-scraping measures (which are triggered upon detecting high traffic) limited the number of listings that could be scraped in a single attempt. Nevertheless, comprehensive datasets were scraped, containing sold and active listings in New York City. A variety of analytical and statistical techniques were then used to identify trends and craft specific business recommendations. Recent developments and external circumstances were also considered in the evaluation.

Categorical analysis was used to estimate what price range a listing could expect to sell for based on the number of bedrooms and bathrooms, and the neighborhood's walkability and transit options. The evaluation also emphasized the range of opportunities available for landlords in growing neighborhoods (due to ever-changing trends) as well as in vacant lots.

2. Business Goal Analysis

2.1. Introduction to the Redfin platform

Redfin is a real estate brokerage that also offers online, map-based property search capabilities. Founded in 2004, the company has sought to carve a niche for itself in the real estate web portal market by earning revenue from users buying or selling with its agents, as opposed to its competitors' reliance on advertising and lead generation. Today, it is active in more than 80 markets across the United States and has been involved in transactions for over 75,000 homes.

Redfin has been active in the New York market for several years but only began listing properties in Manhattan in 2017. This is partly due to the peculiarities of the Manhattan real

estate market. Most real estate markets have a Multiple Listing Service that is traditionally operated by a local realtor association. Agents pay dues to access this shared database of listings compiled by brokers. However, Manhattan lacks a traditional MLS, compounding the complexity involved in buying and selling homes in the borough. The rest of the city is also known for being challenging to navigate for both buyers and sellers.

2.2. Project goals


The New York City market is known for numerous quirks, including trends like housing co-ops and land leases, that distinguish it from other markets. Credible and accurate information is often hard to come by for prospective buyers and even brokers. Brokers and investors who are just entering the market find themselves at a disadvantage, as they are not yet accustomed to the latest trends and policies. Redfin, meanwhile, has positioned itself as a portal that makes listing and buying homes more accessible and transparent for seller and buyer/renter alike.

The purpose of this project was to help prospective landlords and investors (who intend to buy or invest in a property on Redfin) understand what drives profitability in the New York City market. Landlords have many decisions to make, including the property type, location, size, and the number of bedrooms and bathrooms. In addition, they have to consider factors like the convenience of public transit, the neighborhood's walkability, and the ease of biking. With data-driven estimates on which features translate to high returns and how much a particular property will realistically sell for, landlords will be well-equipped to plan effectively and maximize returns.

Web analytics techniques for scraping websites allowed for the automation of the data collection process, while gathering up-to-date and accurate data. Data points collected for each listing included the price (listed/selling), neighborhood, and home type. After performing data cleaning to handle outliers and missing values, regression analysis and visualization were used to uncover price trends (both expected and unexpected) and to identify the most impactful features. Recent developments in the housing market were taken into account to add context to the analysis. The analysis also yielded insights regarding future opportunities for brokers.

3. Dataset Description

The dataset was acquired by using web scraping techniques on the Redfin website, aided by Selenium and its associated Webdriver. Selenium was used to circumvent anti-scraping measures and interact with elements on the dynamic page (mainly, the "Next" button to view more listings). The first dataset consisted of averages from Redfin comparing cities in New York state. The other datasets consisted of active listings in New York City and homes sold on Redfin in New York City.

Building Name		Price	Beds	Baths
	94-11 46 Ave #3	\$680,000	3	2
Elmhurst, NY 11373		Price	Beds	Baths
Status: Active		Redfin Estimate: \$666,548 On Redfin: 12 days		

Building Location

Estimate Price

Duplex 3 bedroom 2 Full Bath Approx 1,100 Sqft with Balcony, Open Kitchen & Laundry

Listing by Huayan Shi • Winzone Realty Inc
Redfin last checked: 20 minutes ago | Last updated Nov 5, 2020 • Source: OneKey

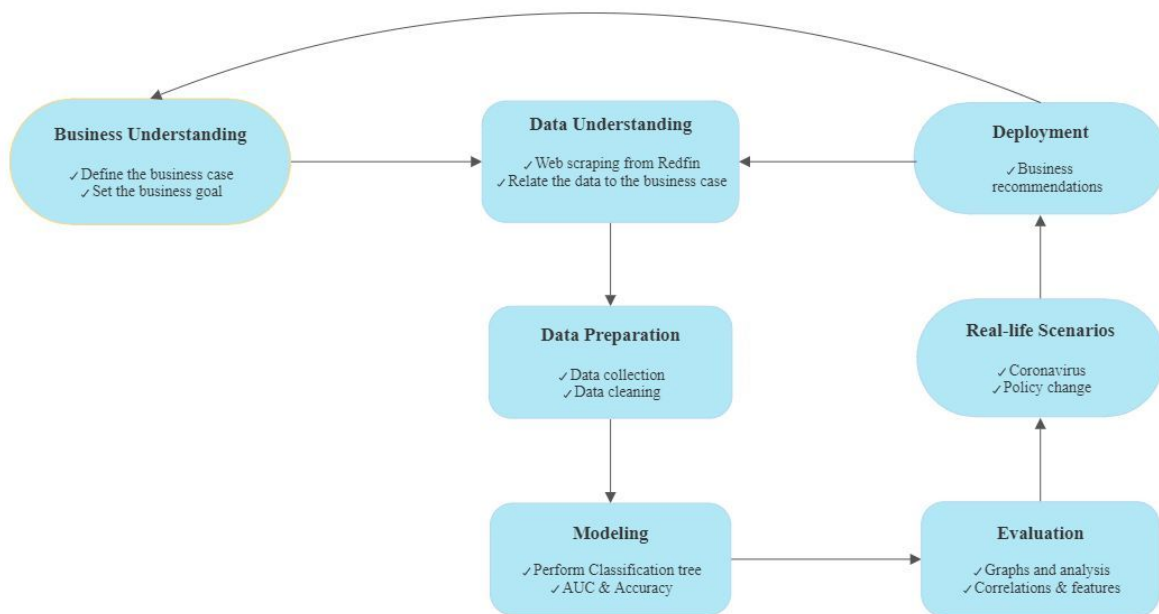
Walker's Paradise Walk Score®	Rider's Paradise Transit Score®	Very Bikeable Bike Score®
96 / 100	100 / 100	82 / 100

Baths	—	Style	Condo/Co-op
Finished Sq. Ft.	—	Year Built	—
Unfinished Sq. Ft.	—	Year Renovated	—
Total Sq. Ft.	1,034	County	Queens County
Stories	1	APN	16021070

Redfin Scraping example page in New York state

From the Redfin website, we scraped all the elements highlighted in the above figure, in a total of 34 pages in both buildings on the active listing and sold buildings in the past three months. Due to regulations and privacy issues of the website, it only allowed us to scrape around 330 house listing information in each category. In total, we obtained 660 records with 13 attributes.

4. System Design



Project workflow

5. System Implementation

5.1. Model purpose

A range of factors affects whether a property is valuable in a particular housing market, in other ways, whether a property will be sold out easily or not. Using predictive analytics, landlords can easily find profitable investment properties. To choose a classification technique that performs well across a wide range of situations without requiring much effort from the agents while being readily understandable by the landlords of the analysis and enhancing their house hunting experience, a strong contender would be the tree methodology.

5.2. Data collection and cleaning

Redfin's web scraping data from <https://www.redfin.com/state/New-York> was combined, and created into two datasets, one containing properties that are still listed, one containing properties that are already being sold. For the predictive model that we built, we used the BuildingOnListing file to predict the valuable properties in the particular New York housing market. However, the data set we choose to train models is not the latest updated data collection, since real estate websites like Redfin are updating every minute and hour.

The clean process was completed on Jupyter Notebook and Google Colab. We removed the

double quotations and dollar signs in the columns. We dropped the text variables like ‘building_website’, ‘building_name’. Then we converted the string variables like ‘property_type’ which represents single-family residential, condo, multi-family residential, other types and ‘building_location’ which stands for some specific locations ‘Flushing, NY’, ‘Brooklyn, NY’, ‘Queens, NY’ and others to categorical variables before putting the model (fit(X,y)). For NA values in the dataset, we could use interpretation methods to fill out the missing data, but since our NA values are only about 10% of total values, and to increase the accuracy of real estate data, we just eliminated all the records with 0 directly. After going through all the processes, the data was clean, organized, and ready to be trained.

building_name	building_price	building_beds	building_baths	redfin_estimates	building_location	property_type	property_size	property_story	property_story	walking_score	transit_score	bike_score	building_description	building_website
33-51 157th St	1588000	4.0	3.0	NaN	Flushing,NY	Single Family Residential	2,750	2.5	2.5	92.0	61.0	56.0	"Completely clean, Tudor Colonial on a quiet st...	https://www.redfin.com/NY/Flushing/33-51-157th...
67-11 223rd Pl Unit B	329000	2.0	1.0	NaN	Bayside,NY	Condo/Co-op	NaN	NaN	NaN	81.0	82.0	63.0	"Second floor 2 bedroom, 1 bath nestled on la...	https://www.redfin.com/NY/Brooklyn/935-Pacific...
935 Pacific St #101	880000	1.0	1.5	NaN	Brooklyn,NY	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
245-49 78th Ave Unit B	299999	1.0	1.0	NaN	Bellerose,NY	Condo/Co-op	NaN	NaN	NaN	69.0	45.0	55.0	"Glen Oaks Village South facing upper corner ...	https://www.redfin.com/NY/Jamaica/245-49-78th...
Unit at 220 E 59th St	899000	1.0	1.0	NaN	New York,NY	Condo/Co-op	605	26.0	26.0	100.0	100.0	83.0	"The unit will be delivered several OPTION TO P...	https://www.redfin.com/NY/New-York/220-E-69th...
...
74-49 Calanus Cr #1	699000	3.0	1.5	68046.0	Elmhurst,NY	Condo/Co-op	NaN	NaN	NaN	86.0	95.0	56.0	"Convenient, comfortable and easy living! Thi...	https://www.redfin.com/NY/Queens/74-49-Calanus...
705 Henderson Ave	799999	4.0	3.0	780606.0	Staten Island,NY	Multi-Family (2-4 Unit)	NaN	NaN	NaN	72.0	63.0	60.0	"Photos coming soon"	https://www.redfin.com/NY/Staten-Island/705-Hen...
8794 17 Ave #304	768000	2.0	2.0	751316.0	Brooklyn,NY	Condo/Co-op	990	3.5	3.5	93.0	75.0	77.0	"Prepare to be impressed when you enter this s...	https://www.redfin.com/NY/Brooklyn/8794-17th-A...
14803 34th Ave	1136000	2.0	1.5	1091043.0	Flushing,NY	Single Family Residential	NaN	NaN	NaN	94.0	81.0	59.0	"Heart Of North Flushing Beautiful Mixed Use I...	https://www.redfin.com/NY/Flushing/148-03-34th...
4613 Avenue J	579000	3.0	2.5	571648.0	Brooklyn,NY	Multi-Family (2-4 Unit)	1,806	2.0	2.0	83.0	86.0	60.0	"Beautiful Semi-detached 2 Family located in E...	https://www.redfin.com/NY/Brooklyn/4613-Avenue...

Active Listings in NYC

building_name	sold_price	building_beds	building_baths	redfin_estimates	building_location	property_type	property_size	walking_score	transit_score	transit_score	bike_score	building_description	building_website
9420 238th St	570,000	4.0	2.0	598,867	Floral Park,NY	Single Family Residential	4,000	72.0	NaN	NaN	59.0	"Won't Last! This Spacious 4-Bedroom, 2-Bath B...	https://www.redfin.com/NY/Floral-Park/9420-238...
95 Locustwood Blvd	525,000	3.0	2.0	559,541	Elmont,NY	Single Family Residential	4,000	39.0	55.0	55.0	42.0	"Magnificent Expanded Cape That's Much Larger ...	https://www.redfin.com/NY/Elmont/95-Locustwood...
852 Cypress Ave Unit 2F	700,000	NaN	NaN	NaN	Queens,NY	Condo/Co-op	NaN	99.0	98.0	98.0	81.0	NaN	https://www.redfin.com/NY/Queens/852-Cypress-A...
75-05 Utopia Pkwy	775,000	3.0	3.0	803,383	Flushing,NY	Single Family Residential	1,440	90.0	63.0	63.0	64.0	"Large home conveniently located in the heart ...	https://www.redfin.com/NY/Flushing/75-05-Utopia...
146-30 177th St	900,000	6.0	4.0	922,498	Jamaica,NY	Multi-Family (2-4 Unit)	4,000	64.0	62.0	62.0	50.0	"Don't miss this meticulously maintained 3 ove...	https://www.redfin.com/NY/Queens/146-30-177th...
209-39 23rd Ave Ave Unit 1K	445,000	3.0	2.0	431,237	Bayside,NY	Condo/Co-op	NaN	72.0	50.0	50.0	48.0	"Don't miss this totally renovated 3 Bedroom I...	https://www.redfin.com/NY/Queens/209-39-23rd-A...
153-40 58th Ave	800,000	NaN	NaN	858,014	Flushing,NY	Single Family Residential	1,800	75.0	68.0	68.0	76.0	NaN	https://www.redfin.com/NY/Flushing/153-40-58th...
91 N Lotus Oval	629,000	4.0	3.0	634,592	Valley Stream,NY	Single Family Residential	6,000	60.0	59.0	59.0	45.0	"Great opportunity in Mill Brook section of Va...	https://www.redfin.com/NY/Valley-Stream/91-Lot...
107-74 111th St	770,000	4.0	2.0	777,808	Richmond Hill 5,NY	Multi-Family (2-4 Unit)	3,503	89.0	81.0	81.0	52.0	"Opportunity awaits in this large 2-Family Hom...	https://www.redfin.com/NY/Queens/107-74-111th...
194-43 114th Rd	615,000	NaN	NaN	662,398	Queens,NY	Multi-Family (2-4 Unit)	3,000	72.0	58.0	58.0	51.0	NaN	https://www.redfin.com/NY/Queens/194-43-114th...

Sold Listings in NYC

5.3. Variables

The dataset contains 349 observations of a variety of properties on listing selections from the New York region. It is divided among 14 other variables that can be used as measurements to predict the profitability of a property. Building location (categorical), which specifies the borough or neighborhood a listing is located in, is one of the most crucial attributes to examine and group the listings. The transit score (numeric) is a measure of how well a location is served by public transit on a scale from 0 to 100. The higher the transit score, the more convenient the location is in terms of proximity to public transit. The property type (categorical) is a field to describe the kind of property for sale. Examples are: Single Family Residential, Condo,

Multifamily, and Other. Each type has its own benefits and drawbacks. Condos are popular options, but some buyers may be concerned about having enough space as families grow larger, and would consider single family homes. The final attributes are the number of bedrooms and baths (both numeric), another key consideration for many buyers.

5.4. Model building

We defined our predictive model to answer the specific question “Is this listing going to be profitable?”, in which we have a target variable ‘good_value’ that is labeled and binarized in order to better utilize investor’s business decisions. Therefore, we recognized our predictive model as supervised. The candidate algorithm is decision trees, after distinguishing between categorical variables and numerical variables, we drop the transit score column, which is the benchmark for judging whether it is a ‘good_value’ property. Then based on the AUC and accuracy, we import GridSearchCV to get different trees. The final step is to split the cleaned data into 20% test dataset and 80% into a training dataset.

Based on optimal AUC	Based on optimal accuracy
The level of depth of the best pruned tree is 3 and the AUC is 0.93303.	The level of depth of the best pruned tree is 3 and accuracy is 0.87142.

Results from model building

5.5. Model evaluation

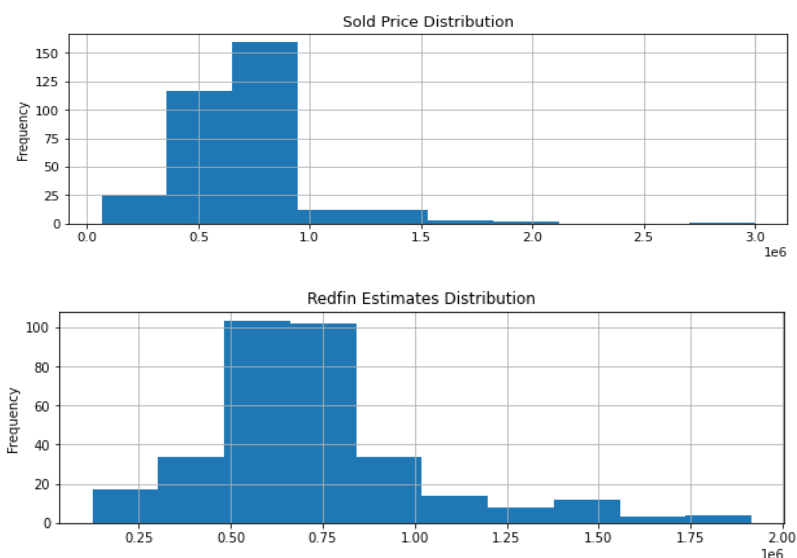
In the above context, Accuracy and AUC values are defined and compared. The results show that the model reaches a reliable performance (around 90% each) with insignificant differences. Therefore, we believe the model offers reliable results and essential information to property landlords, which can enhance the investor’s property hunting experience by lessening their time and increasing the opportunity of finding a profitable investment property.

The model may not be as effective if it is applied to a dataset that is significantly larger than the dataset used for training. It was trained on roughly 350 records, so it would be limited if deployed on a dataset with thousands of rows, for example. Collecting more data would help to train future models to be performant on larger datasets. Gathering data on more features may also produce models with better generalization performance. The current model has already been trained on a comprehensive variety of features, however, the web scraping process gathered data on the majority of the features listed on Redfin. Improving on this may entail scraping from other websites, or even combining data from multiple sources.

Nevertheless, the results from the current model would be a good starting point for a prospective investor looking to create preliminary estimates and make decisions (including which property type to develop and which features to prioritize).

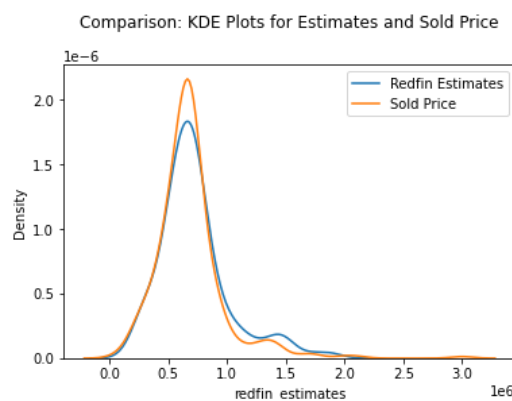
6. Evaluation

6.1. Selling price trends across property types



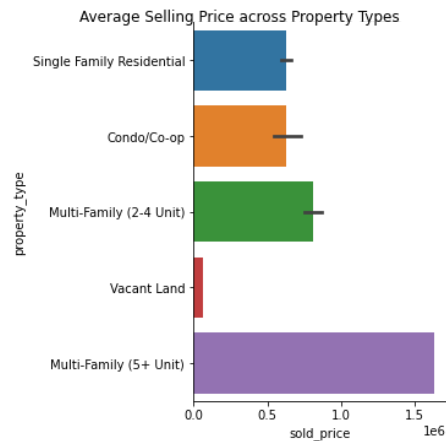
Histograms of price distributions from Sold Listings

Based on the histograms, most listings in New York City were estimated by Redfin to sell in the \$0.5M-1M range but ended up selling at a slightly lower range (\$0.4M-0.9M). The difference is more pronounced at prices greater than \$1M. This is the first indication that Redfin's estimates, while fairly close to the final selling prices, are not always reliable for planning purposes. The method used to arrive at the estimates is unclear, and many of them may be arbitrary figures or based on an overly simple calculation.



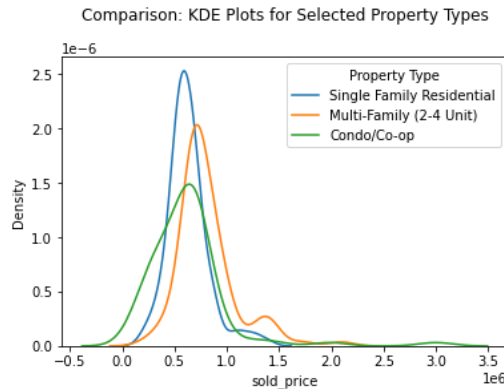
Smoothed price distributions comparison from Sold Listings

The kernel density estimate (KDE) plot, which reduces noise, largely confirms the observation that final selling prices are slightly lower than the Redfin estimates, especially beyond the \$1M mark. It would help to examine the average selling price of each property type and consider the unique properties of each type.



Categorical plot of average price from Sold Listings

Condos and single family homes appear to have similar average prices, at around \$0.7M (matching the median selling price recorded in the city in 2019, which was \$670,000). Meanwhile, multi-family homes (with more than 5 units) sell for more than \$1.5M on average. With condos and cooperatives (co-ops), there is often the need to provide additional luxuries or amenities for residents, which is not always the case with the average single family home. Landlords will have to weigh the costs and benefits of collecting rent from multiple residents in a condo (along with providing a variety of services) against the revenue they would get from selling a single family home. For instance, full-service buildings are becoming increasingly popular in the city and market themselves as providing a luxurious and convenient lifestyle. In many cases, they are expected to provide a range of amenities, including a comprehensive staff on-site. Co-ops often come with their own expectations, and sellers have to consider if their expenses can be recouped by the co-op fees they charge.



Smoothed price distributions from Sold Listings

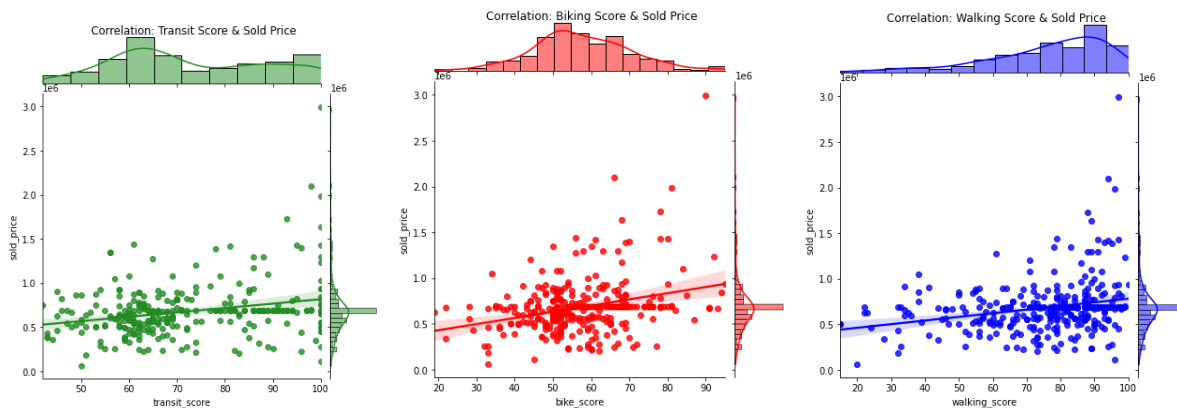
Examining the smoothed distribution functions of the selling price can also offer additional insights into trends within property types. All 3 distributions are skewed to the left, indicating that most listings sold were within the \$0.5M-1M range. The KDE plot confirms that single-family homes and condos have roughly the same average price, while multi-family units (with 2-4 units) have a slightly larger mean. Overall, these 3 property types sell for similar prices, allowing landlords to leverage differences among the properties themselves to maximize revenue. For instance, a single-family home could possibly be sold for a similar price to a co-op, without the additional amenities or features that a condo may be expected to provide, or the expenses of running a full-service condo.

Vacant land is also a crucial prospect for landlords to consider. Land sells for an average of less than \$0.1M based on the Sold Buildings dataset, but this is a limited data point as the dataset only contains a single vacant land listing (for a 1,600 square-foot lot). New York City has long retained its status as the world's most expensive city to build in, costing roughly \$360 per square foot¹, but selling prices are also very high in many parts of the city (and steadily increasing). Once a unit is on the market, the median selling price is \$1,319 per square foot in Manhattan and \$718 in Brooklyn². Taking a vacant land listing of 5,310 square feet from the Active Listings dataset as an example, the lot may cost \$1.9M to develop and sell for \$3.8M. The Brooklyn-based lot is listed for \$1.35M, so a developer could expect \$552,980, or 17%, in total returns. In similar cases where developers buy vacant lots to build on, the objective would be to optimize the features of the building to increase the returns on investment.

¹ <https://therealdeal.com/2018/05/15/surprise-nyc-remains-most-expensive-place-to-build-in-the-world-report/>

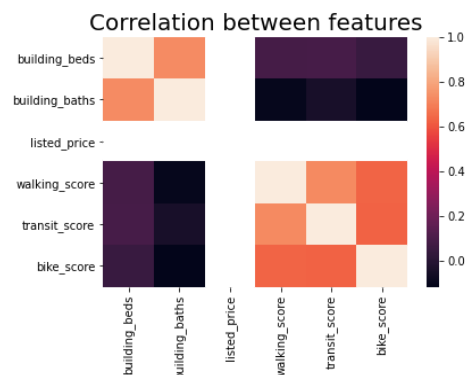
² <https://streeteasy.com/blog/cost-per-square-foot-nyc-manhattan-brooklyn-queens-price/>

6.2. Significant features of sold buildings



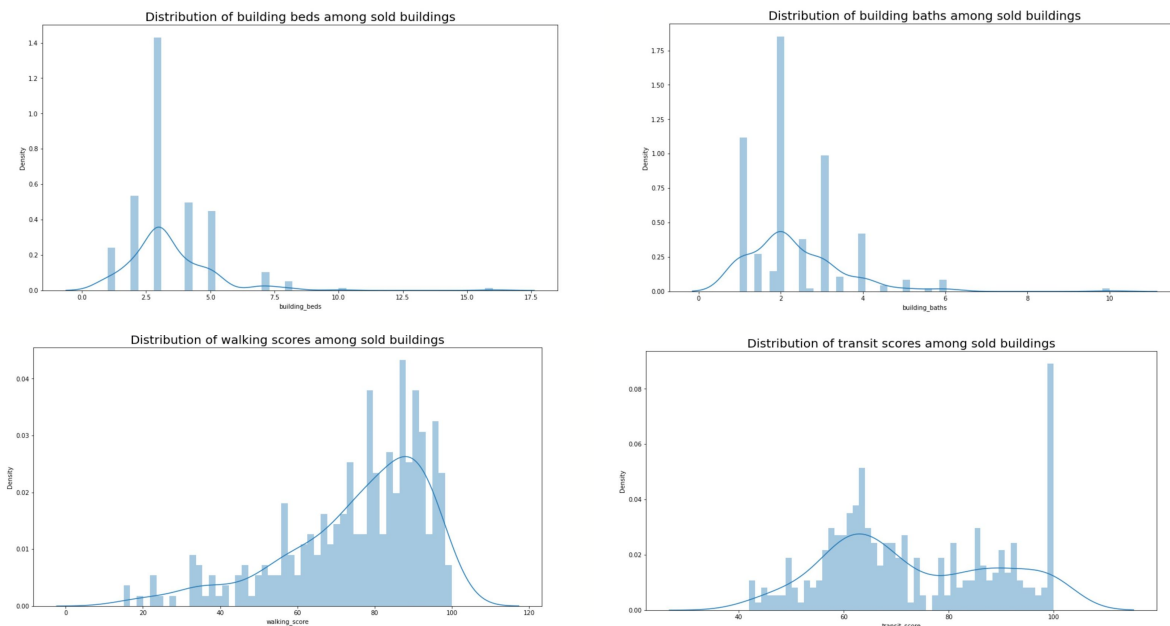
Correlations between features and sold price

The Sold Buildings dataset includes transit, biking, and walking scores, all of which are crucial factors in an urban environment. The relative strengths of the relationships between these scores and the selling prices can indicate which features are strongly tied to a listing selling at a high price. From the correlation plots, the walking score appears to have the highest correlation coefficient, followed by the transit score. This suggests that the walking score for a listing is partially reliable for predicting the price the listing was sold for. Nevertheless, the correlation factors do not appear large enough to suggest a highly predictive relationship. Another form of visualization may be needed to examine if these relationships are significant.



Heatmap comparing building features from Sold Buildings

According to the heatmap, the relationship between building beds and building baths has a very strong positive correlation since the value is close to 1, the transit scores and walking scores show a fairly strong positive relationship too, with its correlation coefficient equals to 0.8. Both of them tell that when customers choose to buy the building, building beds and building baths would be the top 2 factors they look into, and the traffic factors - transit scores and walking scores would also be the second most conservative factors for customers.



Histograms comparing features among Sold Buildings

The four histograms examine the distribution of building beds vs building baths and walking score vs transit score among sold buildings. From the histograms, building beds equals 3 and building baths equals 2 appears to be the most popular investor's purchase choice. Walking scores that fall into 80 between 90 tends to be the favored groups for investor's purchases, with specifically walking score equals 85 appears to be the most popular choice. The transit score equals 100 appears to be the most popular property purchase choice among other sold buildings, it makes sense in life since the higher the transit score, the more convenient the property is in terms of proximity to public transit. Except for landlords who were lucky enough to purchase the 100 point transit score property, it appears the second most popular purchase choices are properties with transit score between 60 to 65.

6.3. Text Analysis

A good, concise description will catch the eye of the viewer more easily. We expected to see some keywords related to the covid-19 in the description part but as the word cloud, the words such as quarantine or safety didn't show in the high-frequency list. In the 2-gram high-frequency list, some words always appear together: 'hardwood' with 'floor', 'private' with 'driveway', 'finish' with 'basement' and 'car' with 'garage'. These findings indicate that many sold buildings were equipped with garages, hardwood floors, private driveways, or finished basements. These are features that landlords may have to consider adding if they want to sell above the market rate, or in the luxury building range. However, some of these features are costly to develop, which will have to be taken into consideration. Hardwood floors and finished basements appear to be

relatively cost-effective and easy for landlords to develop and may be well worth the higher returns they help to deliver.



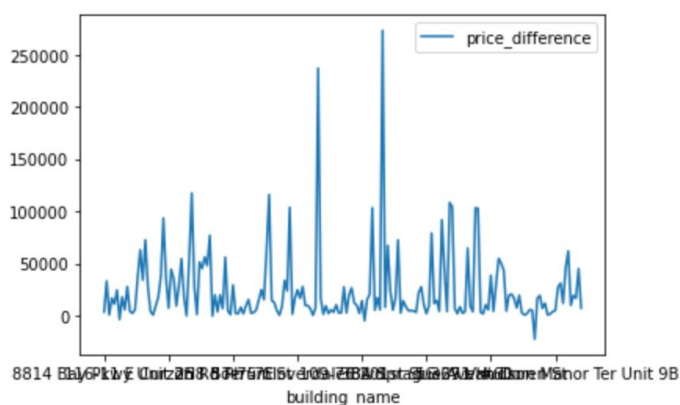
Word clouds from Sold Listings



Word clouds from Active Listings

6.4. Applying insights to Active Listings in NYC

6.4.1 Listing price

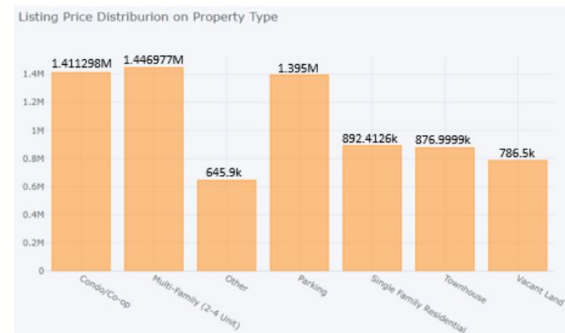


Listing price comparison

As the evaluation part states that the estimated sold price is at a slightly higher range than the actual sold price in about \$0.1M. However, the graph above shows that landlords tend to list the price higher than the estimated price. The mean price difference of listing price to estimated price is approximately \$24627, from the lowest \$331 (greater than 0) to the highest \$117360. There is a trend that landlords tend to set their prices higher than the estimated price, but the buyers tended to offer a much lower price than the estimated price to buy their property. Many landlords may therefore need to lower their listing price, or plan for selling below the original price. These price trends should also be taken into account for their future listings.

	property_type	building_price
0	Condo/Co-op	1.411298e+06
1	Multi-Family (2-4 Unit)	1.446977e+06
2	Other	6.459000e+05
3	Parking	1.395000e+06
4	Single Family Residential	8.924126e+05
5	Townhouse	8.769999e+05
6	Vacant Land	7.865000e+05

Building type with sold price



Listing price distribution on property type

In the sold building list, it is shown that the type of Condo and Single Family has a similar sold price. However, in the active listing building data, the average condo listing price is listed at about 1.41M, which is 0.7M higher than the sold condos. That might be due to the reason that there is an additional property type in the listing building ‘Townhouse’. According to the definition of these two building types, a condo is a single-residential unit that’s housed within a larger building, which may have units above, below, or next to; a townhouse is typically two or three stories tall and shares walls with the next-door properties, but it doesn't have any units above or below. So the reason that condos in the listing have a much higher price than that in the sold building might be because that the condos are housed in the luxury building or due to other amenity reasons, which could be further evaluated in the location.

6.4.2 Amenities

	walking_score	transit_score	bike_score
building_price			
lower than 500k	78.784615	73.229508	61.092308
500k-700k	81.986667	80.926471	61.853333
700k-900k	83.935484	80.964912	66.096774
900k-1.5M	86.012048	84.584416	70.445783
greater than 1.5M	94.075472	95.346154	81.056604

Average categorical building price comparison

We created 5 bins of all the building prices listed above and calculated the mean walking score, transit score, and bike score in different bins. The more expensive the properties are, the higher scores they obtain. The categorical comparison confirms the observations from the Sold Buildings dataset that more convenient neighborhoods (in terms of transit and walkability) attract higher prices listed on the website.

	building_beds	building_baths		building_beds	building_baths
property_type			building_price		
Condo/Co-op	1.970414	1.611111	lower than 500k	1.847458	1.303030
Multi-Family (2-4 Unit)	5.539683	3.250000	500k-700k	2.432432	1.925325
Other	3.000000	4.000000	700k-900k	3.111111	2.066406
Parking	2.000000	2.000000	900k-1.5M	3.962500	2.628049
Single Family Residential	3.338235	2.566176	greater than 1.5M	3.687500	3.208333
Townhouse	3.800000	3.050000			
Vacant Land	NaN	NaN			

Categorical analysis of average building beds and baths in different property types

The figure above shows the average building beds and baths in different property types. From the evaluation for the old buildings, when the building is listed lower than \$900k, prospective landlords would tend to choose the property with an average of 3 beds and 2 baths. However, for the properties where the building is listed higher than \$900k, even though the building beds in the price range of \$900k to \$1.5M are more than the building beds in the price range of greater than 1.5M, the number of building baths in the former price range is less than the number of that in the latter price range. Therefore, there is a trend that when the number of building baths is close to the number of building beds, a higher price would be offered. In this case, according to the figures, these properties are all fit into the profile.

Additionally, the above graphs also represent the basic features in different price ranges which could also conclude recommendations for the current landlords, if they have more building beds and building baths, or walking, transit and biking score in this price range, they could slightly increase their listed price or this stands for those current landlords could sell their properties faster than the rest of the property owners. In other words, if properties in a certain price range, and fail to meet the standard basic features in either scores or number of beds and baths, we would recommend the current landlords to lower their price to sell their properties. On the other hand, as for the investors, we would recommend them to invest in properties that at least meet the basic criteria for this price range.

6.4.3 Listing description text

We also compared the description of buildings sold and the description of buildings in the Active Listings dataset. The description of the unsold buildings has some high-frequency keywords that do not show often in the description of sold buildings. The words include: 'washer' with 'dryer'. These listings may go on to sell for high prices if in-unit washing machines prove to be an attractive feature (which they very likely are).

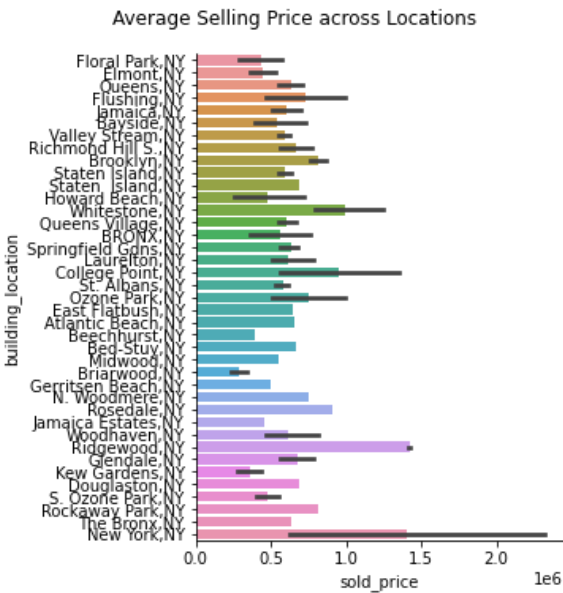
Another set of keywords are: ('lincoln', 'center'), ('park', 'west') and ('central', 'park'). These word sets appeared totally 51 times. These words have much meaning in the location element. As the

location-based trend shows, Central Park and other parks are concentrated in the Manhattan area where buildings' prices are relatively high. Proximity to any park, in general, seems to be correlated with higher prices and greater demand. 'private', 'driveway', referring to private driveways, also appears frequently in the Sold Buildings dataset. While this may be a rarer feature, it appears to be closely linked with high selling prices and may mark a listing as a luxury offering. So for sellers, if their house has this property, remember to include it in the description, which is more or less helpful.

The descriptions of unsold and sold buildings feature some common high-frequency words. Such as 'finish', 'basement' appeared in unsold buildings 49 times and appeared in sold buildings 88 times. Based on the selling situation, buildings with a finished basement would be relatively attractive to investors. Also, 'hardwood floor' shows over 80 times in both two lists. Both hardwood floors and finished basements should be relatively cost-effective for landlords to implement, as mentioned earlier, and may help deliver high returns on the overall listing.

'Public', 'remark' is also a common phrase in both sold and unsold buildings' description, 98 times for sold buildings and 46 times for active listings. In terms of the outcome, “public remark” has effects on the attention of the viewer.

6.5. Location-based trends



Categorical plot of mean price by location from Sold Listings

Manhattan (usually indicated by the “New York, NY” location tag) may be the crown jewel of the city’s housing market, but the other boroughs (especially Brooklyn and Queens) have also drawn attention in recent years due to new developments. In addition, many buyers and renters

have been looking outside of Manhattan to maximize what they spend on housing or to find a location that caters to a specific lifestyle. The ‘building_location’ attribute from the Sold Listings dataset can help uncover neighborhoods where properties can sell for remarkably high prices.

From the categorical plot, Ridgewood (a neighborhood in Queens) has significantly high selling prices, almost on par with Manhattan at roughly \$1.4M. Upon further research, this neighborhood is considered high-income relative to the rest of the city and is known for its well-preserved single-family homes, so properties receive considerable attention upon entering the market. This can justify a high price, along with the neighborhood’s relative suburban feel, its convenient location, and access to various amenities. It is a worthwhile proposition for landlords to closely watch this neighborhood for any openings which they can acquire and re-sell.

Two other neighborhoods that bear mention are Whitestone and College Point, both in Queens and with relatively high average prices. Further research reveals these are both classified as upper-to-middle class neighborhoods and have seen a great deal of new development recently. The increasing levels of demand in these locations bode well for charging at or above the market rate, while leveraging potentially cheaper development costs.

Examining the average selling prices in each neighborhood can help uncover neighborhoods where the market rate has undergone a significant increase in recent years, suggesting that these are the locations where the latest opportunities exist for landlords. These areas may be relatively cheaper to develop in while commanding high selling prices that would yield high returns.

6.6. External circumstances

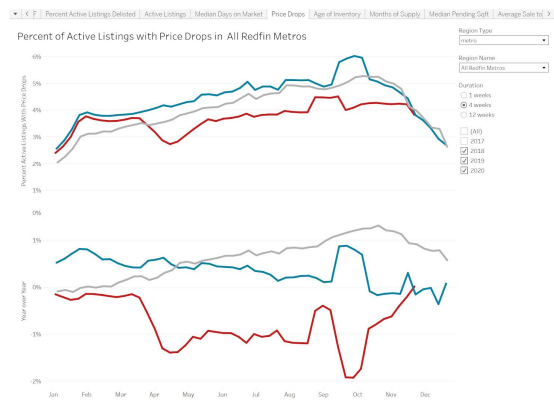
6.6.1 COVID-19

When the coronavirus pandemic hit the United States, it had an immediate impact on the country’s economy³. The real estate market, in particular, has felt the widespread effects of the pandemic⁴ and has been used as an indicator of how the population and the economy are responding. Redfin’s publicly-available dashboards track some of the trends and the changes that have occurred as a result.

³ speedcres.com/blog/how-will-coronavirus-affect-real-estate-market/

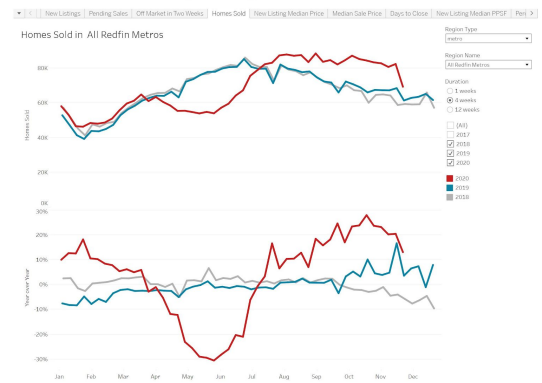
⁴ speedcres.com/blog/how-will-coronavirus-affect-real-estate-market/

Redfin COVID-19 Weekly Housing Market Data



% of active listing with price drops

Redfin COVID-19 Weekly Housing Market Data



No. of homes sold in Redfin metros

From studying trends from Redfin’s dashboards, it appears the percent of active listings with price drops in all Redfin metros including New York reached the first falling peaks in mid-April, the year-over-year growth is -1.4%. It then reached the most falling peaks in October, in which the year-over-year growth is -1.9%. It makes sense because the time frame of these effects matches with the real pandemic time. The initial outbreak of the coronavirus pandemic was in late March, early April. And the second outbreak of pandemic was before the Presidential election in October when the deaths from pandemic were over 200k. The above peaking price drops explains the demand for property decreases. Because financially in uncertain economic times, people tend to put off making bigger purchases⁵, and the high unemployment rate keeps away those who want to have their own house. And we also found out that sellers in Redfin tend to put a higher listing price for the active listings at first during the pandemic period, but the actual sold price is a lot cheaper than the listing price during that period, which also shows the effects of a pandemic caused many sellers to have to drop the price for the properties in order for them to sell it. There’s also another reason that buying a house involves close contact with real estate professionals, which makes social distancing difficult⁵.

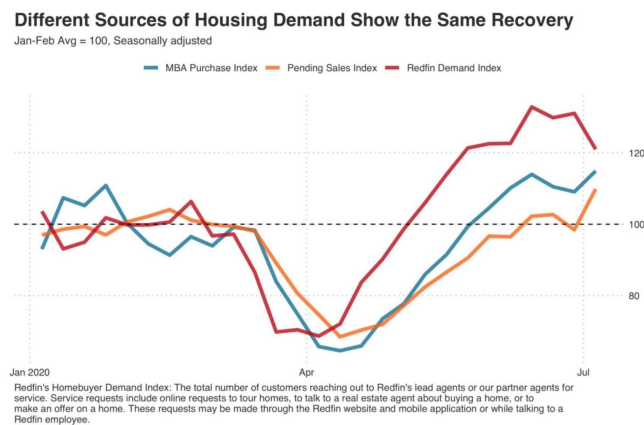
According to the data provided by Redfin, the number of homes sold in all Redfin metros including New York declined 30.7% year-over-year in late May. It makes sense that during that time of the pandemic, with the travel restrictions and other lockdown restrictions, people had to wait to move, and with the possible financial reasons mentioned in the above text would both corporate to the factors that may lead to the drops of home sales. But then the homes sold trend increased from June after, the homes sold quantities and year-over-year growth in the year 2020 even exceed the year 2018 and year 2019’s data. It’s interesting how fast the trend was picking up itself, in particular, several factors contribute to it: First, the summer season tends to have better sales than others for the real estate market. Second, people are buying down because the price of a property drops so that it is a good investment opportunity to “buy low”. Third, there

⁵ speedcres.com/blog/how-will-coronavirus-affect-real-estate-market/

has been a spike in the market due to people wanting to get into a better housing situation after the stay-at-home order⁶, which proves how tightly the real estate market is tied to the economic changes caused by the coronavirus pandemic.

6.6.2. Policy changes

Another factor that affects the up trend towards home sold is the policy change. To prevent a collapse in the housing market, the federal government issued a moratorium on foreclosures⁷. In which is the action that FHFA Extends Foreclosure and Eviction Moratorium to help borrowers and renters who are at risk of losing their home due to the coronavirus national emergency⁸. Low rates and low inventory pushed home sale prices up 7% from a year earlier. This pandemic policy helps to drive the mortgage rate to a record low of 3.03%, and this low rates and low inventory pushed home sale prices up 7% from a year earlier⁹, which explains why the housing is still doing well during the pandemic period.



Housing demand in different indexes¹⁰

According to the data given by Redfin, the housing demand on different sources in which include Redfin Demand Index itself, it shows started from early May, the housing demand is constantly increasing which shows some certain extent of recovery for not only the real estate market but also the economy in the United States.

⁶communityimpact.com/houston/katy/housing-real-estate/2020/07/23/4-katy-area-covid-19-residential-real-estate-trends-to-keep-up-with-in-2020/

⁷speedcres.com/blog/how-will-coronavirus-affect-real-estate-market/

⁸www.fhfa.gov/Media/PublicAffairs/Pages/FHFA-Extends-Foreclosure-and-Eviction-Moratorium.aspx

⁹www.redfin.com/news/home-sales-above-pre-coronavirus-levels/

¹⁰www.redfin.com/news/home-sales-above-pre-coronavirus-levels/

7. Conclusions and Future Direction

7.1. Evaluation results

The evaluation of both the Sold and Active Listings datasets makes it apparent that many listings are listed at a higher price than they would be expected to sell for. The significant appeal of purchasing a home in New York City notwithstanding, the market is highly competitive that buyers take many factors into account. Buyers wish to maximize what they get out of their purchase, and buyers who are looking to purchase one of the more expensive properties are particularly concerned with features and amenities as they have a particular lifestyle or experience in mind. While the housing market is unlikely to crash, the economic effects of the pandemic are expected to remain for several months, if not a few years, in the form of lower demand and prices. Therefore, landlords need to take an approach that optimizes the features and amenities available if they wish to price their listings above the market rate. Otherwise, they will have to plan to sell below (roughly \$0.1M) their initial estimates.

7.2. Business recommendations

Landlords and investors are advised to strongly consider single-family homes, which appear to be uniquely profitable in terms of selling for a higher price relative to the cost of developing them as well as lower expenses compared to a condo or co-op. The categorical analysis that was performed to identify the archetypal properties within certain price ranges yielded crucial insights for calibrating listing prices. The typical property sold for \$1M or less has 3 bedrooms and 2 bathrooms, while properties selling for \$1.5M and above tend to have 3 or more bedrooms and bathrooms. Choosing the property with walking scores that are between the range of 80 to 90 is a safe choice for profitability. And if possible, try to pick the property with a high transit score, 100 would be the best option if it is within the budget. Otherwise, the properties with transit scores between 60 to 65 would be ideal. These estimates, based on historical data, are largely reliable for predicting or adjusting a listing's selling price, or planning a new development.

Regarding the location of a listing, landlords would also benefit from periodically examining the average selling prices in different neighborhoods (especially in Brooklyn and Queens) to identify the latest trends. With new development increasingly spreading outside of Manhattan and even to previously-underserved neighborhoods, opportunities are arising frequently. Crucially, the market may be less saturated here, providing many listings to choose from to invest in. Landlords stand to benefit from meeting high demand in areas that may be relatively inexpensive to develop in. Vacant lots are also worth exploring for investors who have the capital to develop a building, due to the high demand for newly-constructed homes.

7.3. Future direction

To improve the accuracy of our predictive model, collecting more data would help to train future models to be performant on larger datasets. But the biggest difficulty is the collecting process

due to the unique features of Redfin. Redfin, like many other websites nowadays, has anti-scraping measures that are triggered upon detecting high traffic from an IP address. This also limits the number of listings that can be scraped in a single attempt to around 300 to 350. Scraping several times was not necessarily the best solution, as listings are updated often and information for previously-scraped listings may change, resulting in an inconsistent dataset. The evaluation was therefore done using listings that could be scraped in a single attempt.

Another limitation that was identified during evaluation was the lack of listing prices in the Sold Listings dataset (the dataset only had Redfin estimates and final selling prices). There are also other data points that are only available to registered agents. Combining proprietary and publicly-available data would improve the effectiveness of any future analysis.

8. References

Bokhari, Sheharyar. “Home Sales Are Above Pre-Coronavirus Levels as Mortgage Rates Hit New Low.” *Redfin Real Estate News*, 7 Oct. 2020, www.redfin.com/news/home-sales-above-pre-coronavirus-levels/.

Brenzel, Kathryn. “Surprise! NYC remains most expensive place to build in the world: report”. *The Real Deal*, <https://therealdeal.com/2018/05/15/surprise-nyc-remains-most-expensive-place-to-build-in-the-world-report/>

“Downloadable Housing Market Data - Redfin.” *Redfin Real Estate News*, 16 Oct. 2020, www.redfin.com/news/data-center/.

“Get Started Here:” *FHFA Extends Foreclosure and Eviction Moratorium | Federal Housing Finance Agency*, www.fhfa.gov/Media/PublicAffairs/Pages/FHFA-Extends-Foreclosure-and-Eviction-Moratorium.aspx.

“How Will Coronavirus Affect the Real Estate Market?” *SPEED Commercial Real Estate*, 13 July 2020, speedcres.com/blog/how-will-coronavirus-affect-real-estate-market/.

Para, Jen. “4 Katy-Area COVID-19 Residential Real Estate Trends to Keep up with in 2020.” *Impact*, Impact, 23 July 2020, communityimpact.com/houston/katy/housing-real-estate/2020/07/23/4-katy-area-covid-19-residential-real-estate-trends-to-keep-up-with-in-2020/.

Quintana, Mariela. “The Wildly Differing Cost of 900 Square Feet Across NYC”. *StreetEasy Reads*, <https://streeteasy.com/blog/cost-per-square-foot-nyc-manhattan-brooklyn-queens-price/>.

9. Code

Web Scraping

```
import requests # to get the website
import time    # to force our code to wait a little before re-trying to grab a webpage
import re      # to grab the exact element we need
from bs4 import BeautifulSoup # to grab the html elements we need
import pandas as pd # to create dataframe
from selenium import webdriver #use selenium to avoid website's anti-scraping
from selenium.common.exceptions import NoSuchElementException
path = 'C:\\Users\\yugem\\webdrivers\\chromedriver.exe'
browser = webdriver.Chrome(path)
#In this data, we mainly focus on data in New York State
browser.get('https://www.redfin.com/state/New-York')
def check_exists_by_xpath(xpath):
    try:
        browser.find_element_by_xpath(xpath)
    except NoSuchElementException:
        return False
    return True
data = []
page_source = browser.page_source
soup = BeautifulSoup(page_source, 'xml')
cities = 'NA'
avg_price = 'NA'
avg_ppersqr = 'NA'
avg_day = 'NA'
cities_link = 'NA'
if (check_exists_by_xpath('//a[@class="ui_button nav next primary "]')):
    browser.find_element_by_xpath('//a[@class="ui_button nav next primary "]').click()
    time.sleep(4)
table = soup.find('table')
tbody = table.find('tbody')
trs = tbody.find_all('tr')
for tr in trs:
    lnk = tr.find_all('a')
    lnk = lnk[0]
```

```

link = lnk.get('href')
cities_link = "https://www.redfin.com" + link
cities = lnk.text.strip()
td1 = tr.find_all('td',{'class':'c1'})
td1 = td1[0]
avg_price = td1.text.strip()
td2 = tr.find_all('td',{'class':'c2'})
td2 = td2[0]
avg_ppersqr = td2.text.strip()
td3 = tr.find_all('td',{'class':'c3'})
td3 = td3[0]
avg_day = td3.text.strip()

data.append([cities, avg_price, avg_ppersqr, avg_day, cities_link])
#get each cities' average price, average price/ sqr ft., average days on redfin, and link
df = pd.DataFrame(data, columns = ['Cities', 'Avg.Price', 'Avg.price/sqr', 'Avg.days', 'link'])

#In this dataset, we only focus on NYC's data
building_list = []
nyc = data[0][4]
webpages = 17
start = 1
for webpage in range(webpages):
    if (check_exists_by_xpath('//a[@class="ui_button nav next primary "]')):
        browser.find_element_by_xpath('//a[@class="ui_button nav next primary "]').click()
        time.sleep(4)
    pattern = '/page-'
    webpage = nyc + pattern + str(start)
    browser.get(webpage)
    page_source1 = browser.page_source
    soup1 = BeautifulSoup(page_source1, 'lxml')
    start += 1
    div = soup1.findAll('div', {'class':re.compile('HomeCardContainer')})
    for item in div:
        if (check_exists_by_xpath('//a[@class="ui_button nav next primary "]')):
            browser.find_element_by_xpath('//a[@class="ui_button nav next primary "]').click()
            time.sleep(4)
        a = item.find('a')
        if a == None:
            continue

```

```

else:
    building_link = a.get('href')
    website_pattern = 'https://www.redfin.com'
    building_link = website_pattern + building_link
    browser.get(building_link)
    page_source2 = browser.page_source
    soup2 = BeautifulSoup(page_source2, 'lxml')
    span_address = soup2.find('span', {'class':re.compile('street-address')})
    building_name = span_address.text.strip()
    div_price = soup2.find('div', {'class':re.compile('statsValue')})
    building_price = div_price.text.strip()
    div_beds = soup2.find('div', {'data-rf-test-id':'abp-beds'})
    building_beds = div_beds.text.strip()
    if building_beds == '—Beds':
        building_beds = 'NA'
    div_baths = soup2.find('div', {'data-rf-test-id':'abp-baths'})
    building_baths = div_baths.text.strip()
    if building_baths == '—Baths':
        building_baths = 'NA'
    description_p = soup2.find('p', {'class':re.compile('text-base')})
    if description_p == None:
        building_description = 'NA'
    else:
        building_description = description_p.text.strip()
    span_estimate = soup2.find('span', {'data-rf-test-id': 'avmLdpPrice'})
    if span_estimate == None:
        redfin_estimates = 'NA'
    else:
        subspan_estimate = span_estimate.find('span', {'class':re.compile('value')})
        redfin_estimates = subspan_estimate.text.strip()
    span_location = soup2.find('span', {'class':re.compile('locality')})
    span_region = soup2.find('span', {'class':re.compile('region')})
    building_location = span_location.text.strip()
    building_region = span_region.text.strip()
    div_type = soup2.findAll('div', {'class':re.compile('table-value')})
    if len(div_type) == 0:
        property_type = 'NA'
        property_size = 'NA'
        property_story = 'NA'
    else:

```



```

property_type = div_type[7].text.strip()
if property_type == '—':
    property_type = 'NA'
property_size = div_type[4].text.strip()
if property_size == '—':
    property_size = 'NA'
property_story = div_type[5].text.strip()
if property_story == '—':
    property_size = 'NA'
walking_div = soup2.find('div', {'class':re.compile('transport-icon-and-percentage
walkscore'}}))
if walking_div == None:
    walking_score = 'NA'
else:
    walking_span = walking_div.find('span', {'class':re.compile('value'}}))
    walking_score = walking_span.text.strip()
transit_div = soup2.find('div', {'class':re.compile('transport-icon-and-percentage
transitscore'}}))
if transit_div == None:
    transit_score = 'NA'
else:
    transit_span = transit_div.find('span', {'class':re.compile('value'}}))
    transit_score = transit_span.text.strip()
bike_div = soup2.find('div', {'class':re.compile('transport-icon-and-percentage
bikescore'}}))
if bike_div == None:
    bike_score = 'NA'
else:
    bike_span = bike_div.find('span', {'class':re.compile('value'}}))
    bike_score = bike_span.text.strip()

building_list.append([
    building_name, building_price, building_beds,
    building_baths, redfin_estimates, building_location,
    property_type, property_size, property_story,
    walking_score, transit_score, bike_score, building_description, building_link])
#create dataframe for building_list of active listing building's information
df1 = pd.DataFrame(building_list, columns = ['building_name', 'building_price', 'building_beds',
    'building_baths', 'redfin_estimates', 'building_location',
    'property_type', 'property_size', 'property_story', 'walking_score', 'transit_score',

```

```
'bike_score', 'building_description', 'building_website']])
```

```
#Now extract sold building information of nyc
sold_list = []
webpages = 17
filter_pattern = '/filter/include=sold-3mo'
start = 1
for webpage in range(webpages):
    if (check_exists_by_xpath('//a[@class="ui_button nav next primary "]')):
        browser.find_element_by_xpath('//a[@class="ui_button nav next primary "]').click()
        time.sleep(4)
    pattern = '/page-'
    webpage1 = nyc + filter_pattern + pattern + str(start)
    browser.get(webpage1)
    page_source3 = browser.page_source
    soup3 = BeautifulSoup(page_source3, 'xml')
    start += 1
    divs = soup3.findAll('div', {'class':re.compile('HomeCardContainer')})
    for item in divs:
        if (check_exists_by_xpath('//a[@class="ui_button nav next primary "]')):
            browser.find_element_by_xpath('//a[@class="ui_button nav next primary "]').click()
            time.sleep(4)
        a = item.find('a')
        if a == None:
            continue
        else:
            sold_building_link = a.get('href')
            website_pattern = 'https://www.redfin.com'
            sold_building_link = website_pattern + sold_building_link
            browser.get(sold_building_link)
            page_source4 = browser.page_source
            soup4 = BeautifulSoup(page_source4, 'xml')
            span_address = soup4.find('span', {'class':re.compile('street-address')})
            if span_address == None:
                building_name = 'NA'
            else:
                building_name = span_address.text.strip()
            div_price = soup4.find('div', {'data-rf-test-id':'abp-price'})
            if div_price == None:
```

```

    sold_price = 'NA'
else:
    subdiv_price = div_price.find('div')
    sold_price = subdiv_price.text.strip()
redfin_div = soup4.find('div', {'data-rf-test-id': 'avm-price'})
if redfin_div == None:
    redfin_estimates = 'NA'
else:
    subredfin_div = redfin_div.find('div')
    redfin_estimates = subredfin_div.text.strip()
listed_div = soup4.find('div', {'class': re.compile('price-col number')})
if listed_div == None:
    listed_price = 'NA'
else:
    listed_price = listed_div.text.strip()
div_beds = soup4.find('div', {'data-rf-test-id': 'abp-beds'})
if div_beds == None:
    building_beds = 'NA'
else:
    building_beds = div_beds.text.strip()
if building_beds == '—Beds':
    building_beds = 'NA'
div_baths = soup4.find('div', {'data-rf-test-id': 'abp-baths'})
if div_baths == None:
    building_baths = 'NA'
else:
    building_baths = div_baths.text.strip()
if building_baths == '—Baths':
    building_baths = 'NA'
description_p = soup4.find('p', {'class': re.compile('text-base')})
if description_p == None:
    building_description = 'NA'
else:
    building_description = description_p.text.strip()
    building_description = "" + building_description + ""
span_location = soup4.find('span', {'class': re.compile('locality')})
if span_location == None:
    building_location = 'NA'
else:
    span_region = soup4.find('span', {'class': re.compile('region')})

```

```

        building_location = span_location.text.strip()
        building_region = span_region.text.strip()
        building_location = "" + building_location + building_region + ""
    div_type = soup4.findAll('div', {'class':re.compile('table-value')})
    if len(div_type) == 0:
        property_type = 'NA'
        property_size = 'NA'
        property_story = 'NA'
    else:
        property_type = div_type[5].text.strip()
        if property_size == '—':
            property_size = 'NA'
        property_size = div_type[4].text.strip()
        if property_size == '—':
            property_size = 'NA'
        property_story = div_type[5].text.strip()
        if property_story == '—':
            property_story = 'NA'
    walking_div = soup4.find('div', {'class':re.compile('transport-icon-and-percentage
walkscore')})
    if walking_div == None:
        walking_score = 'NA'
    else:
        walking_span = walking_div.find('span', {'class':re.compile('value')})
        walking_score = walking_span.text.strip()
    transit_div = soup4.find('div', {'class':re.compile('transport-icon-and-percentage
transitscore')})
    if transit_div == None:
        transit_score = 'NA'
    else:
        transit_span = transit_div.find('span', {'class':re.compile('value')})
        transit_score = transit_span.text.strip()
    bike_div = soup4.find('div', {'class':re.compile('transport-icon-and-percentage
bikescore')})
    if bike_div == None:
        bike_score = 'NA'
    else:
        bike_span = bike_div.find('span', {'class':re.compile('value')})
        bike_score = bike_span.text.strip()

```

```

sold_list.append([
    building_name, sold_price, building_beds,
    building_baths, redfin_estimates, listed_price, building_location,
    property_type, property_size, property_story,
    walking_score, transit_score, bike_score, building_description, sold_building_link])
#create dataframe for building_list of sold building's information
df2 = pd.DataFrame(sold_list, columns = ['building_name', 'sold_price', 'building_beds',
'building_baths',
                                'redfin_estimates', 'listed_price', 'building_location',
                                'property_type', 'property_size', 'property_story',
                                'walking_score', 'transit_score', 'bike_score',
                                'building_description', 'building_website'])
df2.head(10)

```

Classification tree - AUC

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns

from google.colab import files
uploaded = files.upload()

# read the original dataset, and add a new column to it
filename ="new BuildingOnListing - BuildingOnListing (version 1).csv"
df = pd.read_csv(filename)
df

rvar_list =['transit_score','building_website','building_name','building_description']
df_sample1 = df.drop(columns=rvar_list)

cvar_list = ['building_location', 'property_type','good_value']
nvar_list = ['building_price', 'building_beds', 'building_baths', 'redfin_estimates',
'property_size','property_story', 'walking_score','bike_score']

```

```

df_sample3 = df_sample1.copy()
df_sample3[cvar_list] = df_sample1[cvar_list].astype('category')
df_sample3[nvar_list] = df_sample1[nvar_list].astype('float64')

# Required package: scikit-learn. Package name in Python: sklearn
# Required subpackage: model_selection. Required function name: train_test_split
from sklearn.model_selection import train_test_split

# Placeholder variables: df4partition, testpart_size
# test_size specifies the percentage for the test partition
df4partition = df_sample3
testpart_size = 0.2

# random_state specifies the seed for the random number generator.
# random_state = 1 unless otherwise noted
df_nontestData, df_testData = train_test_split(df4partition, test_size=testpart_size,
random_state=1)

# Required package: scikit-learn. Package name in Python: sklearn
# Required subpackage: tree
# Required function name: DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

# Separate the predictor values and the DV values into X and y respectively
# Placeholder variable: DV
DV = 'good_value'
y = df_nontestData[DV]
X = df_nontestData.drop(columns=[DV])

from sklearn.tree import export_graphviz
from io import StringIO
import pydotplus
from IPython.display import Image

```

```

# A user-defined function summary_tree to display a classification tree
def summary_tree(model_object):
    dot_data = StringIO()
    export_graphviz(model_object, out_file=dot_data, filled=True,
                    rounded=True, special_characters=True, feature_names=X.columns.values,
                    class_names=['0', '1'])
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    output_imagefile = 'tree.png'
    graph.write_png(output_imagefile)
    return output_imagefile
kfolds = 5

# Here we specify within which range of depths we will search for the best pruned tree
kfolds = 5

# Here we specify within which range of depths we will search for the best pruned tree
maximum_depth = 100
minimum_depth = 1

param_grid = {'max_depth': list(range(minimum_depth, maximum_depth+1))}

from sklearn.model_selection import GridSearchCV

gridsearch = GridSearchCV(DecisionTreeClassifier(criterion='entropy', random_state=1),
param_grid, scoring='roc_auc', cv=kfolds, n_jobs=-1)
gridsearch.fit(X,y)
clf_BPT = gridsearch.best_estimator_

# Display the resulting best pruned tree
Image(summary_tree(clf_BPT))

# Display the level of depth of the best pruned tree

```

```

print(clf_BPT.get_depth())

# y_test_actual is the actual values of the DV in the test partition
y_test_actual = df_testData[DV]

# X_test is the predictor values in the test partition
X_test = df_testData.drop(columns=[DV])

# Get the AUC of the best pruned tree model
from sklearn.metrics import roc_auc_score
print(roc_auc_score(y_test_actual, clf_BPT.predict_proba(X_test)[:,-1]))

import numpy as np

def get_treepaths(dtc, df):
    rules_list = []
    values_path = []
    values = dtc.tree_.value

    def RevTraverseTree(tree, node, rules, pathValues):
        try:
            prevnode = tree[2].index(node)
            leftright = '<='
            pathValues.append(values[prevnode])
        except ValueError:
            # failed, so find it as a right node
            prevnode = tree[3].index(node)
            leftright = '>'
            pathValues.append(values[prevnode])

        # now let's get the rule that caused prevnode to -> node
        p1 = df.columns[tree[0][prevnode]]
        p2 = tree[1][prevnode]
        rules.append(str(p1) + ' ' + leftright + ' ' + str(p2))

```



```

# if we've not yet reached the top, go up the tree one more step
if prevnode != 0:
    RevTraverseTree(tree, prevnode, rules, pathValues)

# get the nodes which are leaves
leaves = dtc.tree_.children_left == -1
leaves = np.arange(0,dtc.tree_.node_count)[leaves]

# build a simpler tree as a nested list: [split feature, split threshold, left node, right node]
thistree = [dtc.tree_.feature.tolist()]
thistree.append(dtc.tree_.threshold.tolist())
thistree.append(dtc.tree_.children_left.tolist())
thistree.append(dtc.tree_.children_right.tolist())

# get the decision rules for each leaf node & apply them
for (ind,nod) in enumerate(leaves):

    # get the decision rules
    rules = []
    pathValues = []
    RevTraverseTree(thistree, nod, rules, pathValues)

    pathValues.insert(0, values[nod])
    pathValues = list(reversed(pathValues))

    rules = list(reversed(rules))

    rules_list.append(rules)
    values_path.append(pathValues)

for i in range(len(rules_list)):

    print('\nLeaf node ID =', i+1)

```

```

    print('Path =', rules_list[i])
    distro = sum(values_path[i][-1])
    print('sample =', int(sum(distro)))
    print('value =', list([int(distro[0]), int(distro[1])]))
    predicted_class = 1 if distro[1] > distro[0] else 0
    print('class = ', predicted_class)

return None

get_treepaths(dtc=clf_BPT, df=df_nontestData)

Accuracy

gridsearch = GridSearchCV(DecisionTreeClassifier(criterion='entropy', random_state=1),
param_grid, scoring='accuracy', cv=kfolds, n_jobs=-1)
gridsearch.fit(X,y)
clf_BPT = gridsearch.best_estimator_

# Display the resulting best pruned tree
Image(summary_tree(clf_BPT))

# Display the level of depth of the best pruned tree
print(clf_BPT.get_depth())

# y_test_actual is the actual values of the DV in the test partition
y_test_actual = df_testData[DV]

# X_test is the predictor values in the test partition
X_test = df_testData.drop(columns=[DV])

from sklearn import metrics
metrics.confusion_matrix(y_test_actual, clf_BPT.predict(X_test))

clf_BPT.score(X_test, y_test_actual )

```

```
get_treepaths(dtc=clf_BPT, df=df_nontestData)
```

Plots

```
import pandas as pd
import numpy as np
import matplotlib as plt
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# new dataframe to store data on listings marked as sold
sold_df = pd.read_csv('SoldBuilding.csv', index_col=0, thousands=',')
sold_df
# no.of missing values for each variable
sold_df.isnull().sum()
# replacing missing values inplace with the column's mean
sold_df['sold_price'].fillna((sold_df['sold_price'].mean()), inplace=True)
sold_df['redfin_estimates'].fillna((sold_df['redfin_estimates'].mean()), inplace=True)
# drop rows with irrelevant property_type values
sold_df.drop(sold_df.loc[sold_df['property_type']=='Yes'].index, inplace=True)
sold_df.drop(sold_df.loc[sold_df['property_type']=='No'].index, inplace=True)

# average sold price
sold_df.sold_price.mean()
# average price per property type
sold_df.groupby("property_type").sold_price.mean()
# visualize avg sold price across property types
sns.catplot(x="sold_price", y="property_type",
            kind='bar', data=sold_df).fig.suptitle("Average Selling Price across Property Types",
y=1);
# examining relationship between transit score & sold price, with bivariate & univariate graphs
sns.jointplot(x="transit_score", y="sold_price", data=sold_df, kind="reg",
            color='forestgreen').fig.suptitle("Correlation: Transit Score & Sold Price", y=1);
```

```

# examining relationship between bike score & sold price
sns.jointplot(x="bike_score", y="sold_price", data=sold_df, kind="reg",
              color='red').fig.suptitle("Correlation: Biking Score & Sold Price", y=1.0);
# examining relationship between walking score & sold price
sns.jointplot(x="walking_score", y="sold_price", data=sold_df, kind="reg",
              color='blue').fig.suptitle("Correlation: Walking Score & Sold Price", y=1);

# visualize avg sold price by location
sns.catplot(x="sold_price", y="building_location",
            kind='bar', data=sold_df).fig.suptitle("Average Selling Price across Locations", y=1.05);

# histogram to look at distribution of sold price
sold_df['sold_price'].plot(figsize=(10,3), kind="hist", title="Sold Price Distribution", grid=True);
# histogram to look at distribution of redfin estimates
sold_df['redfin_estimates'].plot(figsize=(10,3), kind="hist", title="Redfin Estimates
Distribution", grid=True);
# kde plots for estimates & sold price
sns.kdeplot(sold_df.redfin_estimates,
            label='Redfin Estimates');
sns.kdeplot(sold_df.sold_price,
            label="Sold Price").set_title("Comparison: KDE Plots for Estimates and Sold Price",
y=1.1);
plt.legend(loc='upper right',
          labels=['Redfin Estimates', 'Sold Price']);

# kde plots comparing distributions of 3 diff. property types
sns.kdeplot(sold_df.sold_price[sold_df.property_type=="Single Family Residential"],
            label='Single Family Residential');
sns.kdeplot(sold_df.sold_price[sold_df.property_type=="Multi-Family (2-4 Unit)"],
            label="Multi-Family (2-4 Unit)");
sns.kdeplot(sold_df.sold_price[sold_df.property_type=="Condo/Co-op"],
            label="Condo/Co-op").set_title("Comparison: KDE Plots for Selected Property Types",
y=1.1);
plt.legend(title='Property Type', loc='upper right',

```

```
labels=['Single Family Residential', 'Multi-Family (2-4 Unit)', 'Condo/Co-op']);
```

Text Analysis

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

df = pd.read_csv(r'D:\2020 fall\BYGB-7978 Web Analytics(R)\project\SoldBuilding.csv')
df['sold_price'] = df['sold_price'].str.replace(',', '')

import warnings
warnings.filterwarnings('ignore')
import re
from nltk.stem import WordNetLemmatizer, PorterStemmer, SnowballStemmer
stop_words_file = 'SmartStoplist.txt'

stop_words = []

with open(stop_words_file, "r") as f:
    for line in f:
        stop_words.extend(line.split())

stop_words = stop_words

def preprocess(raw_text):

    #regular expression keeping only letters
    letters_only_text = re.sub("[^a-zA-Z]", " ", str(raw_text))

    # convert to lower case and split into words -> convert string into list ( 'hello world' -> ['hello',
    'world'])
    words = letters_only_text.lower().split()

    cleaned_words = []
```

```

lemmatizer = PorterStemmer()

# remove stopwords
for word in words:
    if word not in stop_words:
        cleaned_words.append(word)

# stemm or lemmatise words
stemmed_words = []
for word in cleaned_words:
    word = lemmatizer.stem(word)
    stemmed_words.append(word)

# converting list back to string
return " ".join(stemmed_words)

import nltk
from collections import Counter
frequency=df.building_description.str.split(expand=True).stack().value_counts()
frequency[:40]

from collections import Counter
Counter(" ".join(df["prep"]).split()).most_common(30)

#nice library to produce wordclouds
from wordcloud import WordCloud

import matplotlib.pyplot as plt
# if uising a Jupyter notebook, include:
%matplotlib inline

all_words = "

#looping through all incidents and joining them to one text, to extract most common words

```

```

for arg in df["prep"]:

    tokens = arg.split()

    all_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 700, height = 700,
                        background_color ='white',
                        min_font_size = 10).generate(all_words)

# plot the WordCloud image
plt.figure(figsize = (5, 5), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()

from nltk.util import ngrams
n_gram = 2
n_gram_dic = dict(Counter(ngrams(all_words.split(), n_gram)))

for i in n_gram_dic:
    if n_gram_dic[i] >= 2:
        print(i, n_gram_dic[i])
#Above part is for the description of sold building

from nltk.util import ngrams
n_gram = 2
n_gram_dic = dict(Counter(ngrams(all_words.split(), n_gram)))
onlist={}
freq=[]
for i in n_gram_dic:
    if n_gram_dic[i] >= 2:

```

```
onlist[i] = n_gram_dic[i]  
freq.qppend()  
print(i, n_gram_dic[i])
```