Model Comparison of the Linear Model, the Regression Tree
&
The Random Forest on Queens Apartment Price Prediction

Final Project for Risk Management 742: Data Science via Machine Learning

May 25th, 2022

By: Lamae Maharaj

In Collaboration with:
Peter Antonaros
Javendeen Naipaul

Abstract

This paper discusses the comparison between three types of machine learning models used in data science and statistics. Many factors impact the final price of apartment sales. Data can only be collected based on what is known and given. Together with many missing pieces to predict apartment prices, the Regression, the Linear, and the Random Forest Algorithms proceeds to complete the task. They are set side by side as to which model gives the most accurate price. The dataset features and the algorithms construct statistical models to predict apartment prices. The dataset comes from Amazon MTruk and contains data from February 2016 until February 2017.

1. Introduction

New York City consists of one of the most diverse boroughs globally. Not only are the people diverse, but so are apartment prices. They range from one hundred thousand in some areas and even millions of dollars in the inner city. Queen, however, is the most diverse borough in terms of culture according to popular thought. What else is diverse in this borough? Correct guess! The apartment prices! There are many types of properties in the Queens, New York area. This paper examines how diverse these prices are regarding the features that play a role in their value. Many features play a role in this phenomenon. Only nature can honestly know the main causal drivers.

Like the phenomenon above, many questions lurk in the minds of inquisitive thinkers. These questions lead to an experiment. The Experiments are then to collect data. Data is said to hold observations and those observations are from what we learn. Those observations, in many ways, engineer a model. Some may model it as a globe from observations taken from out of space, a car as a mechanical prototype design, and even statistical models built on variables. The statistical model becomes a predictive model when used to make predictions about unknown events that will happen in the future. After being trained with data, some models can interpolate based on what they know already. Alternatively, sometimes extrapolate to guess an event based on data it never learned.

Predictive models are not perfect and carry a series of issues. Three types of errors play a significant role in prediction. The first is a model misspecification error. Model misspecification error occurs when a model's function does not correctly meet the full standard to map across data points. In other words, the model's function does not account for all data points. Therefore, it will not give a near-perfect output as expected. The second type of error is estimation error, where the error is extracted from the model the function that is created and subtracted from the best function within the candidate set of choices. The third type of error is ignorance. This error plays a significant role in this paper. Ignorance holds the place for all features not within the given data set.

Some of these models show high-performance metrics, such as the Random Forest model. Some variables that play a prominent role in making sure these models can predict an appropriate level are the apartment's square footage, the average price of the apartments in the area, and many more. Like all significant excavations, it starts with finding accurate clues about what we are trying to discover. Nevertheless, the excavation starts with cleaning the artifacts at the surface.

2. Data

There are many ways information is collected. Information is collected in a survey or sifting through old records, and sometimes they are slipped away from right under our noses. This data, however, is retrievable from the MLSI using Amazon's MTurk. This data ranges from a collection of observations from February 2016 up to February 2017. The data consists of 55 columns, also known as features, and 2230 rows, also known as observations. Like all excavations, dirt, sand, and many useless materials appear when looking for true gems.

Within the raw data frame, large sums of disreputable junk exist, such as the URL from where the data is from, the creation time of the post, and many more. Others, such as the square footage, sale price, and the number of total rooms, seemed respectable for future analysis. There were numerical, characterized, and logical entries within this large data frame. The data set from the MLSI is the housing data set that contains information for two categories of property. The first being apartments and the second being co-ops.

After looking at all the features within the dataset, some features are not valuable. They are like dirt and sand while looking for an old necklace. This is where the necessary tools come into use to pull and recreate a new data frame of valuable features. While looking at similar features to the *url, HITId, Title Keywords,* and many more, the procedure was to extract the numbers and categories that may play a role. Most of the useless information that is in the original data set contains web information of where the data came from.

2.2. Featurization

Some features within the data set that seem to be useful based from what I think an apartment price can be used for prediction stayed. Some new features are included from Zillow housing market retrieval tool and a similar tool produced by Redfin. The features that are kept consist of 6 categorical features, and 14 numerical features. The 6 categorical features are dining_room_type, fuel_type, kitchen_type, cats_allowed, dogs_allowed, and coop_condo. The 14 numerical features that are kept are approx_year_built, maintenance_cost, num_bedrooms, num_floors_in_building, num_full_bathrooms, num_total_rooms, parking_charges, sale_price, sq_footage, walk_score, listing_price_to_nearest_1000, and three newly made features that is in the next section. They are zipcode, avg_prices, and price_per_sqft. Underneath there is a summary statistic that describes the kept features and new features after proper cleaning.

The zipcode feature is to only make sure proper categorizing of the avg_prices which is the average price of condos within that area in 2022. I thought it would have been a great variable to include since it may or may not be close to the original prices. The approx_year_built shows the average age of these properties data back to around 1961. The standard deviation between these years are around 78 years. The youngest properties were built in 2016. The oldest properties date back to 1915. The dining_room_type feature consists of different types of dining rooms. There are a total of five categories. They are none, other, combo, dining area, and formal. These features are dummified as none = 0, combo = 1, dining area = 2, formal = 3, and other = 4. The fuel_type feature is like the categorical variable above. There are 5 categories. They were dummified in the following order. The categories are none = 0, electric = 1, gas = 2, oil = 3, and other = 4. The kitchen_type was also dummified in the following order. The categories are none=0, Eat In = 1, efficiency = 2, 1995 = 3, and Combo = 4. Maintenance costs are shown by the feature maintenance_cost which is a numerical value. The mean maintenance cost is $799.62 while the minimum is $155, and the maximum is $4659.0. The standard deviation for this variable is $373.67. The num_bedroom variable shows the amount bedrooms within the apartment. The maximum number of bedrooms are 3, the minimum is around 1 and the mean is around 2 bedrooms. The num_floors_in_building shows the number of floors in a building that the apartment is located in. The minimum number of floors are 1 while the maximum is 33 floors. The average amount of floors within this data set is around 7 floors. The num_full_bathrooms represent the number of full bathrooms. This means it has a toilet, sink and a shower. The average amount of full bathrooms is 1. The maximum is 3, while the min is also 1. The num_of total_rooms are a numerical value just as the previous features. The minimum number of rooms is around 1 room, the mean however is 4 rooms. The maximum is 8. The next variable is the amount of money that users pay for parking when they have these condos. They parking_charges variable is numerical as the mean amount is $107.31, the minimum is $9.00 and the maximum charge is $500.00. The standard deviation for this feature is $64.79. The sale_price is the price that the apartment is sold for. The lowest sale price was $66,000.00, the maximum was $999,999.00 and the mean of the price is $324,369.23. The standard deviation between prices is $177,243.83. The sq_footage is a feature that measures the square footing of the apartments. The mean square footage is 909 square feet, the minimum size was 450 square feet and the maximum is 6215 square feet. The walk_score feature measures something called walkability. This is the metric that allows people to know if they can do their regular errands without the use of a car. The average walk score is 83.71 which is very walkable. The maximum walk score is 99 which is considered a walker's paradise (Walk Score Methodology). The minimum is 15 which is very car dependent (Walk Score Methodology). The next feature is a numerical feature identified as listing_price_to_nearest_1000. This feature is the listing price of the house to the nearest $1000. The minimum is $179,000.00, the maximum is $759,000.00 and the mean is $459,259.00. This feature was later multiplied b 1000 in attempt to make a better prediction. The feature was later renamed to price_listings. The avg_prices represent the average prices of condos within the same zip code in the year of 2022. This was an idea of mine. I included it to have a better range of numbers closer to the original price. The minimum is $190,000.00, the maximum is $804,446.00, the mean is $416,206.97. The standard deviation amongst these prices is $118,140.71. The next variable is a categorical variable is cats_allowed. The cats_allowed variable is dummified to say whether cats are allowed in the apartments or not. The same information is given for the next categorical feature dogs_allowed but instead in relations to dog. The coop_condo tells if the apartment is a coop or a condo. The price_per_sqft feature is a new creation made by dividing the values in the sale_prices by sq_footage. The table underneath gives more information on all statistics related to the data prepared before the analysis begins. The mean price per square foot is $357.12, the maximum is $1169.51, and the minimum is $78.68.

	count	mean	std	min	25%	50%	75%	max
zipcode	524.0	11359.368321	79.136979	11004.000000	11360.000000	11372.000000	11375.000000	11435.00000
approx_year_built	524.0	1961.320611	21.090905	1915.000000	1950.000000	1955.000000	1965.000000	2016.00000
dining_room_type	524.0	2.002443	1.090339	1.000000	1.000000	1.490000	3.000000	4.00000
fuel_type	524.0	2.368607	0.586004	0.000000	2.000000	2.000000	3.000000	4.00000
kitchen_type	524.0	1.914294	1.011482	1.000000	1.000000	2.000000	2.000000	4.00000
maintenance_cost	524.0	813.793282	379.252478	155.000000	620.150000	720.000000	860.500000	4659.00000
num_bedrooms	524.0	1.629771	0.669145	1.000000	1.000000	2.000000	2.000000	3.00000
num_floors_in_building	524.0	6.561069	5.734910	1.000000	3.000000	6.000000	6.000000	33.00000
num_full_bathrooms	524.0	1.208015	0.424684	1.000000	1.000000	1.000000	1.000000	3.00000
num_total_rooms	524.0	4.124046	1.134322	1.000000	3.000000	4.000000	5.000000	8.00000
parking_charges	524.0	107.191240	64.251036	9.000000	72.982500	99.090000	124.027500	500.00000
sale_price	524.0	324369.230916	177243.836326	66000.000000	179750.000000	275000.000000	435000.000000	999999.00000
sq_footage	524.0	908.496298	359.431178	450.000000	730.845000	850.000000	987.950000	6215.00000
walk_score	524.0	83.723282	13.052100	15.000000	76.000000	86.000000	94.000000	99.00000
price_listings	524.0	459259.541985	61542.666297	179000.000000	416800.000000	451600.000000	492200.000000	759000.00000
avg_prices	524.0	416206.967111	118140.724637	190000.000000	316500.000000	412000.000000	490000.000000	804446.00000
cats_allowed	524.0	0.471374	0.499657	0.000000	0.000000	0.000000	1.000000	1.00000
dogs_allowed	524.0	0.286260	0.452444	0.000000	0.000000	0.000000	1.000000	1.00000
coop_condo	524.0	0.246183	0.431198	0.000000	0.000000	0.000000	0.000000	1.00000
price_per_sqft	524.0	357.122938	171.450097	78.680611	234.984919	301.593881	458.676798	1169.51122

2.3. Errors & Missingness

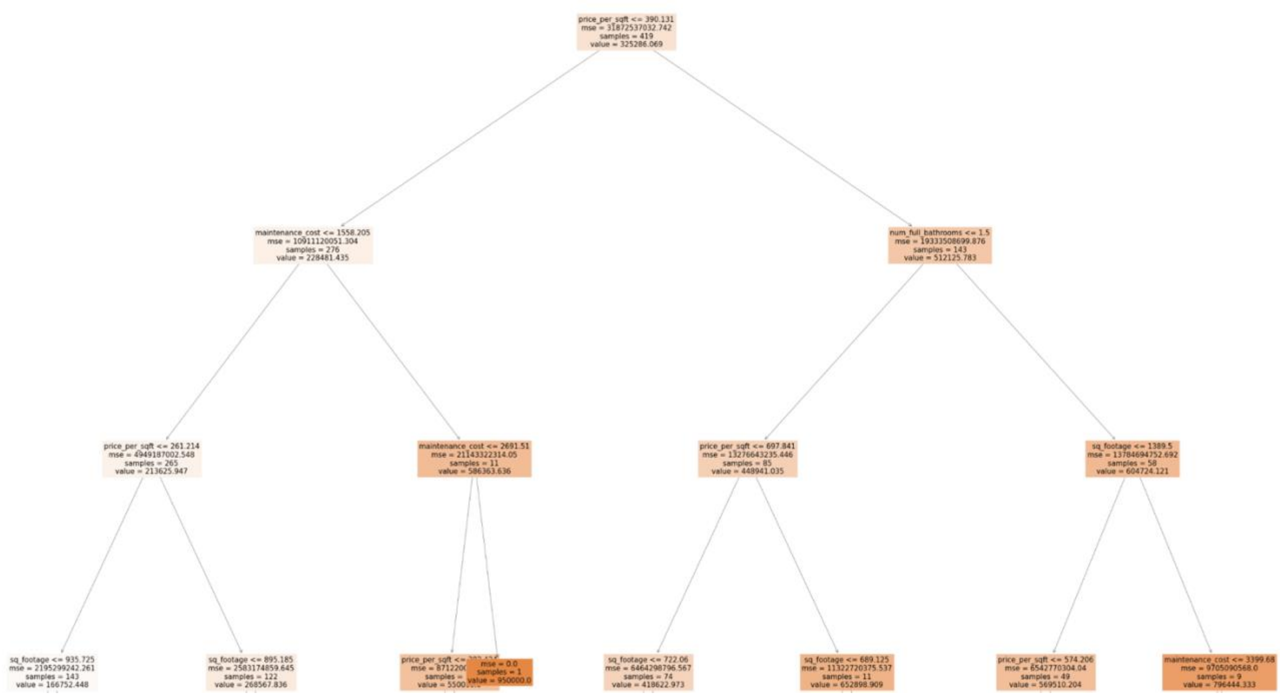
This data set consisted of many missing values. The main challenge was to create a high-powered predictive model with the limited amount of data that is useful. This task became harder after finding that the y variable which is needed to train the model had a total of 1700 missing values. Many things could be done here but this was just one of the issues. The other issues included the string type feature full_address_or_zipcode. Being a string certain parsing procedures were done to clean the entire feature. This was done using the str.extract() method within Python. Whilst cleaning the problem occurred on sight that some observations were missing zip codes. I proceeded to use the Zillow Home Value Index and the Redfin Housing Market Trends to manually find the missing addresses with the zip codes. Another issue occurred with the dogs_allowed and cats_allowed feature. The dogs_allowed contained the following responses no, yes ang yes89. All responses for yes89 were replaced with yes via the replace() function. The cat s_allowed column contained no, yes and y. The response y was replaced with yes with the same replace() function.The kitchen_type feature contained eat in, efficiency, Combo, combo, Eat In, Eat in, 1995, eatin, efficiency kitchene, efficiency kitchen, efficiemcy, efficiency ktchem. The replacements for all observations that contained Combo for all that contained combo, other for all containing Other,and efficiency for all that contained misspellings and meant efficiency. Some of the features that contained money values had dollar signs and commas. With the use of regex these were cleansed allowing strings to become free of the unnecessary symbols and later are turned into floats. These features were sale_price, listing_price_to_nearest_1000, parking_charges, maintenance_cost,and common_charges. Realizing there were 1700 missing sales_price observations those rows were then dropped. The model was now thought to be built with 555 rows of data. Imputation is now key within this dataset. The 555 rows are now imputed using the MissForest algorithm from the package missingpy. However, when looking at the numbers on the statistics table prior to the one shown above, odd values appeared where some apartments didn't have rooms, or there were negative prices. The missing values from the rows that are remaining is now further cleansed from negative price values, and apartments with no rooms. This brought the data set to 524 rows.

3. Modeling

To begin modeling, the data remaining data set is divided into 2 parts. The X_data which consist of all features and observations that will predict on the y_data which will be all the values that will be training the data as to what should be predicted. The X_data contained approx_year_built, dining_room_type, num_total_rooms, fuel_type, maintenance_cost, num_bedrooms, num_full_bathrooms, sq_footage, walk_score, avg_prices, coop_condo, and price_per_sqft. The reason for choosing num_bedrooms, num_full_bathrooms, and num_total_rooms is due to having no variable for living rooms and other rooms. Therefore, the remaining number of rooms can be predicted for if they may be living rooms or other rooms. To give an honest comparison of the model's predicting power, all variables mentioned above were kept when regressing with the linear model, the regression tree, and the random forest. The X_train together with the y_train data contains of 419 rows, and the X_test data together with the y_test data contains 105 rows. Both the Regression tree and the Random Forest were set to a max_depth of 5.

3.1. Regression Tree Modeling

The Regression Tree Algorithm creates features that are split based in nature. This model is built with the use of the sklearn.tree API and the use of the imported DecicionTreeRegressor(). The decision tree could make use of the observed features in order to train a model in order to predict the future. This model is meant to produce useful outputs. The Regression Tree model has a total of 5 layers depicted underneath. There are 27 nodes, and 28 leaves according to the visualization below. The root node starts with splitting price_per_sqft less than or equal to $390.13. The second layer is shown where if the root node split is less than $390.13 the decision tree decides to split on if the maintenance_cost is less than or equal to $1558.20. If the price_per_sqft is more than $390.13, then the decision tree splits num_full_bathrooms being less than or equal to 1.5 full bathrooms. If the num_full_bathrooms is less than 1.5 full bathrooms than the third layer node will be making a decision on sq_footage being less than or equal to 697.8 square feet. If the num_full_bathrooms are more 1.5 then the third layer node will be deciding if the sq_footage is 1389.5 square feet. For the third layer, if the maintenance_cost is less than $1558.20 then the decision tree decides to split again on if price_per_sqft is less than or equal to $261.21. If maintenance_cost is greater than $1558.20 then the decision goes to the next node as maintenance_cost less than or equal to $2991.51. After looking at the first three layers the most valuable features in deciding are between price_per_sqft, maintenance_cost, num_full_bathrooms, and sq_footage. If it's shown that the root node makes its decision with a sample of 419. This means that the first split was made whilst the algorithm looks for the most samples before the node is set. The in-sample performance metrics gave an R-Squared of 96.40% while the RMSE is $33,781.80. The out of sample metrics produced an R-Squared of 91.96% and an RMSE of $48,504.96. This model in my opinion has a high R-Squared due to the number of features placed within the model and I believe that this model could produce more honest metrics if trained with more data. Underneath shows a depiction of the first three layers of the decision tree.



3.2. The Linear Model

This model is built from the sklearn.linear_model API together with the LinearRegression() function. This linear model is created using the same features as all other models discussed in this paper. The feature's coefficient that has the most impact is the num_full_bathrooms with a total of 79011.3312. The second highest or most impactful is the num_bedrooms at 31910.49. The variables that is shown above in the Decision Tree map is also impactful within this model. The price_per_sqft coefficient is 805.701116. The second feature that made an impact in the Decision Tree model above is the maintenance_cost and its coefficient is 96.7462217.

Some coefficients were not as impactful in this model except for the num_full_bathrooms. The weakest coefficient in this model was the num_total_rooms which was -1002.28454. These features according to the linear model seem to be the most impactful. The in-sample metrics returns an R-Sqaured of 94.98% and an RMSE of $39,383.81. The out of sample metrics produced an R-Sqaured of 94.55% and an RMSE of $39,938.25. The Linear model thus far based on performances, defeats the Regression Tree's performance metrics. My opinion of this model is that these numbers may decrease with later predictions due to have a small training and testing data set. According to the out of sample metrics, this model may be good enough for prediction. However, there is room for error within this model.

```
                        Coefficients:
Intercept:               990089.82416215
approx_year_built       -6.46172707e+02
dining_room_type         4.18813169e+03
fuel_type                2.95274608e+03
maintenance_cost         9.67462217e+01
num_bedrooms             3. .19104981e+04
num_full_bathrooms       7.90113312e+04
num_total_rooms          1.00228454e+03
sq_footage               8.45837749e+01
walk_score               9.21564278e+01
avg_prices              -2.08020323e-02
coop_condo               7.16620379e+03
price_per_sqft           8.05701116e+02
```

3.3. The Random Forest Model

This is the third model is in use for this model comparison experiment on housing prices. The Random Forest model is an algorithm that use the same ideology of the decision tree algorithm we fist spoke about in the beginning. The Random Forest algorithm is a nonparametric algorithm. This means that this algorithm does not make assumptions about the type o mapping functions when mapping input to output data. This also means that this algorithm has the ability to choose any form of data training functions it chooses to produce quality predictions. This model is a supervised learning algorithm that uses “bagging” to solve regression problems. The “bagging” technique is used to predict out of sample data. This algorithm can also be used to solve classification problems as well. However, for this experiment the Random Forest Regression Algorithm will be used. This model built using the sklearn.ensemble API as the RandomForestRegressor() was imported. This algorithm constructs a significant amount of decision trees at the point of where the model is trained. Then it outputs the mean of the prediction of the individual decision trees. If it were the classification version of the algorithm, then the procedure would be the same, but the mode of the decision tree would have been produced instead. What was gained by choosing this model is having a better R-Squared and a lower RMSE which is in the next section. I believe that this model is overfit to some degree because of the high R-Squared shown below.

4. The Random Forest Model Results

The Random Forest model on this data set produced higher performance metrics than both the Linear model and the Decision Tree Regression model. The in-sample metrics produced an R-Squared of 97.92% and an RMSE of $25,735.93. The RMSE in this model like all others show

how much in the predictive value the model's output is off by. This means model is known to have a higher performance than both Linear and Decision Trees. The out of sample metrics produced an R-Sqaured of 96.16% and an RMSE of $33,528.20. This is in fact the lowest out of sample RMSE between all three models. Although it is the case that the Random Forest model can produce a higher out of sample metrics than the in-sample metrics, it is not what happened in my experiment. I do believe that this model is not ready for deployment as there is work that needs to be done. This model can make predictions, but I do fear extrapolation demolishing these high-performance metrics. However, if this model can be trained with more data, then it will be possible to continue to flourish pass the other two compared models.

5. Discussion

The goal of this experiment is to build three models and compare their performance metrics based on their prediction on the sale_price feature within the Queens area. However, a major issue within this experiment was due to a lack of data for model training purposes. Because of 1700 missing sales_price observations, these models may have learned too well and thus may have been overfit. The Random Forest model, as expected, did produce the highest performance metrics. Secondly, the Linear model performed great. Out of the three models the Decision Tree was outperformed by the above-mentioned models. 77.21% of the data was not used due to the missing sales_price. So, within only 22.78% of the data being used, these models may have been overfit. This is due to learning from the same pattern of some original and some imputed data. I do not think that this model can beat Zillow. In the future, I believe these models can be successful. Therefore, I am taking the initiative to get data from the Zillow API to train all three models with a larger sum of clean, ordered, and reputable data.

Acknowledgements

I would like to thank my fellow classmates and collaborators on this project. I would like to extend a special thanks to Peter Antonaros and Javendeen Naipaul for discussing this project to lead me through some creative ideas for my features. It was a pleasure learning and planning this project. Special thanks to Zillow and Redfin for their tools. I do hope this project is continued later with the same collaborators to make better machine learning models.

References:

Walk Score. (2022). *Walk Score Methodology*. Walk Score Methodology.
Retrieved May 23, 2022, from https://www.walkscore.com/methodology.shtml

Redfin. (2022). *Redfin United States Housing Market*. Redfin United States Housing Market.
Retrieved May 23, 2022, from https://www.redfin.com/us-housing-market

Zillow. (2022). *United States Home Values*. United States Home Values.
Retrieved May 23, 2022, from https://www.zillow.com/home-values/

```
cd ~/desktop
```

```
/Users/lamaemaharaj/Desktop
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import sklearn.neighbors._base
import sys
sys.modules['sklearn.neighbors.base'] = sklearn.neighbors._base
from missingpy import MissForest
```

```
data = pd.read_csv('housing_data_2016_2017.csv')
data
```

```
                                HITId  
HITTypeId  \
0     3OID399FXG7F26JWONXF0Y86J90FD4  36BILMLQB75QQNBTYKGYCZWDN8TVAU

1     3MQY1YVHS3K2MF90MWR2LPQH7KJ2B0  36BILMLQB75QQNBTYKGYCZWDN8TVAU

2     3DGDV62G7O94Q9AA5193G9V6OOY2PL  36BILMLQB75QQNBTYKGYCZWDN8TVAU

3     3087LXLJ6MGL3MI2CB9KLRONPKRF0B  36BILMLQB75QQNBTYKGYCZWDN8TVAU

4     3FULMHZ7OUX88KSKHZ0ZSKY93XJ4MN  36BILMLQB75QQNBTYKGYCZWDN8TVAU

...                              ...                             ...

2225                             NaN                             NaN

2226                             NaN                             NaN

2227                             NaN                             NaN

2228                             NaN                             NaN

2229                             NaN                             NaN


                                                  Title  \
0     Find Information about Housing To Help a Stude...   
1     Find Information about Housing To Help a Stude...   
2     Find Information about Housing To Help a Stude...   
3     Find Information about Housing To Help a Stude...   
4     Find Information about Housing To Help a Stude...   
...                                                 ...   
2225                                                NaN   
2226                                                NaN   
```

```
2227                                                  NaN
2228                                                  NaN
2229                                                  NaN

                                           Description  Keywords  Reward
\
0     Go to a link and copy information into the HIT       NaN  $0.05

1     Go to a link and copy information into the HIT       NaN  $0.05

2     Go to a link and copy information into the HIT       NaN  $0.05

3     Go to a link and copy information into the HIT       NaN  $0.05

4     Go to a link and copy information into the HIT       NaN  $0.05

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

2225                                               NaN       NaN     NaN

2226                                               NaN       NaN     NaN

2227                                               NaN       NaN     NaN

2228                                               NaN       NaN     NaN

2229                                               NaN       NaN     NaN


                     CreationTime  MaxAssignments  \
0     Wed Feb 15 22:13:37 PST 2017             1.0
1     Wed Feb 15 22:13:37 PST 2017             1.0
2     Wed Feb 15 22:13:41 PST 2017             1.0
3     Wed Feb 15 22:13:33 PST 2017             1.0
4     Wed Feb 15 22:13:38 PST 2017             1.0
...                           ...             ...
2225                          NaN             NaN
2226                          NaN             NaN
2227                          NaN             NaN
2228                          NaN             NaN
2229                          NaN             NaN

                                 RequesterAnnotation  \
0     BatchId:2689947;OriginalHitTemplateId:920937336;
1     BatchId:2689947;OriginalHitTemplateId:920937336;
2     BatchId:2689947;OriginalHitTemplateId:920937336;
3     BatchId:2689947;OriginalHitTemplateId:920937336;
4     BatchId:2689947;OriginalHitTemplateId:920937336;
...                                                ...
```

```
2225                                                NaN
2226                                                NaN
2227                                                NaN
2228                                                NaN
2229                                                NaN

      AssignmentDurationInSeconds  ...  num_half_bathrooms
num_total_rooms  \
0                           900.0  ...                 NaN
5.0
1                           900.0  ...                 NaN
4.0
2                           900.0  ...                 NaN
3.0
3                           900.0  ...                 NaN
5.0
4                           900.0  ...                 NaN
4.0
...                           ...  ...                 ...
...
2225                          NaN  ...                 NaN
4.0
2226                          NaN  ...                 NaN
5.0
2227                          NaN  ...                 NaN
6.0
2228                          NaN  ...                 NaN
6.0
2229                          NaN  ...                 NaN
5.0

      parking_charges  pct_tax_deductibl sale_price sq_footage
total_taxes  \
0                 NaN                NaN  $228,000        NaN
NaN
1                 NaN                NaN  $235,500      890.0
NaN
2                 NaN                NaN  $137,550      550.0
$5,500
3                 NaN                NaN  $545,000        NaN
$2,260
4                 NaN               39.0  $241,700      675.0
NaN
...               ...                ...        ...        ...
...
2225              NaN                NaN        NaN        NaN
$3,588
2226             $99                 NaN        NaN        NaN
$5,100
2227              NaN                NaN        NaN     1500.0
```

```
$250
2228              NaN                  NaN        NaN     1600.0
$250
2229              NaN                  NaN        NaN     1134.0
$3,785

     walk_score listing_price_to_nearest_1000  \
0            82                           NaN
1            89                           NaN
2            90                           NaN
3            94                           NaN
4            71                           NaN
...         ...                           ...
2225         97                          $628
2226         82                          $988
2227         96                          $850
2228         96                          $850
2229         82                          $899

                                                     url
0                                                    NaN
1                                                    NaN
2                                                    NaN
3                                                    NaN
4                                                    NaN
...                                                  ...
2225  http://www.mlsli.com/homes-for-sale/address-no...
2226  http://www.mlsli.com/homes-for-sale/One-Bay-Cl...
2227  http://www.mlsli.com/homes-for-sale/address-no...
2228  http://www.mlsli.com/homes-for-sale/address-no...
2229  http://www.mlsli.com/homes-for-sale/Two-Bay-Cl...

[2230 rows x 55 columns]
```

```
data['common_charges']
```

```
0       $767
1         NaN
2       $167
3       $275
4         NaN
        ...
2225    $480
2226    $956
2227    $250
2228    $250
2229    $792
Name: common_charges, Length: 2230, dtype: object
```

```python
prices_zips = ['11361: $265,000',"11362: $490,000","11363:
$250,000",'11364: $315,000', '11354: $490,000','11355: $553,000 ',
              '11356: $612,000','11357: $316,000','11358:
$476,000','11359: $329,000','11360: $412,000','11365: $483,000',
             ' 11366: $550,000','11367: $525,000','11412:
$600,000','11423: $195,000','11432: $190,000','11433: $250,000',
              '11434: NaN','11435: $274,000','11436: NaN','11101:
$804,446', '11102: $667,000', "11103: $608,000", '11104: $477,000',
               '11105: $607,000','11106: $600,000','11374:
$669,000','11375: $480,000','11379: $486,000','11385: $554,000',
               '11004: $316,500','11005: NaN', '11411: NaN','11413:
NaN', '11422: $400,000', '11426: $281,000', '11427: $291,000',
               '11428: NaN', '11429: NaN','11414: $371,000','11415:
$369,000','11416: NaN','11417: $441,000', '11418: $369,000',
               '11419: NaN', '11420: $470,000', '11421: $235,000',
'11368: $220,000','11369:$313,000', '11370: $476,000', '11372:
$410,000',
               '11372: $320,000','11373: $420,000','11377:
$450,000','11378: $423,000']

zips  = [i.split(':', 1)[0] for i in prices_zips]
prices =[i.split(':', 1)[1] for i in prices_zips]
zips = pd.DataFrame(zips,columns=['Zipcodes'])
prices = pd.DataFrame(prices,columns=['avg_prices'])
prices['avg_prices'] = prices['avg_prices'].replace({'\$': '', ',':
''}, regex=True)
prices['avg_prices']  = prices['avg_prices'].astype(float)
price_means = prices['avg_prices'].mean()
prices = prices.fillna(price_means)
prices['zips'] = zips['Zipcodes']
a_z = prices
a_z['zips'] = a_z['zips'].astype(str)
a_z.info()
a_z
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   avg_prices  56 non-null     float64
 1   zips        56 non-null     object
dtypes: float64(1), object(1)
memory usage: 1.0+ KB
```

	avg_prices	zips
0	265000.000000	11361
1	490000.000000	11362
2	250000.000000	11363
3	315000.000000	11364

4	490000.000000	11354
5	553000.000000	11355
6	612000.000000	11356
7	316000.000000	11357
8	476000.000000	11358
9	329000.000000	11359
10	412000.000000	11360
11	483000.000000	11365
12	550000.000000	11366
13	525000.000000	11367
14	600000.000000	11412
15	195000.000000	11423
16	190000.000000	11432
17	250000.000000	11433
18	427722.255319	11434
19	274000.000000	11435
20	427722.255319	11436
21	804446.000000	11101
22	667000.000000	11102
23	608000.000000	11103
24	477000.000000	11104
25	607000.000000	11105
26	600000.000000	11106
27	669000.000000	11374
28	480000.000000	11375
29	486000.000000	11379
30	554000.000000	11385
31	316500.000000	11004
32	427722.255319	11005
33	427722.255319	11411
34	427722.255319	11413
35	400000.000000	11422
36	281000.000000	11426
37	291000.000000	11427
38	427722.255319	11428
39	427722.255319	11429
40	371000.000000	11414
41	369000.000000	11415
42	427722.255319	11416
43	441000.000000	11417
44	369000.000000	11418
45	427722.255319	11419
46	470000.000000	11420
47	235000.000000	11421
48	220000.000000	11368
49	313000.000000	11369
50	476000.000000	11370
51	410000.000000	11372
52	320000.000000	11372
53	420000.000000	11373

```
54  450000.000000   11377
55  423000.000000   11378
```

```
datan =
data[['approx_year_built','cats_allowed','dining_room_type','dogs_allo
wed','fuel_type','full_address_or_zip_code',

'garage_exists','kitchen_type','maintenance_cost','num_bedrooms','num_
floors_in_building','num_full_bathrooms',

'num_total_rooms','parking_charges','sale_price','sq_footage','walk_sc
ore','listing_price_to_nearest_1000','coop_condo','common_charges',]]
datan['full_address_or_zip_code'] =
datan['full_address_or_zip_code'].astype(str)
datan['zipcode'] = datan['full_address_or_zip_code'].str.extract(r'(\
d{5}\-?\d{0,4})')
#datan = pd.get_dummies(datan, columns=['cats_allowed'])
datan.isnull().sum()
```

```
/var/folders/y1/cj1dr0kd7lng8yfkrtd5_r_00000gn/T/
ipykernel_24584/2627950147.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  datan['full_address_or_zip_code'] =
datan['full_address_or_zip_code'].astype(str)
/var/folders/y1/cj1dr0kd7lng8yfkrtd5_r_00000gn/T/ipykernel_24584/26279
50147.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  datan['zipcode'] =
datan['full_address_or_zip_code'].str.extract(r'(\d{5}\-?\d{0,4})')

approx_year_built                 40
cats_allowed                       0
dining_room_type                 448
dogs_allowed                       0
fuel_type                        112
full_address_or_zip_code           0
garage_exists                   1826
kitchen_type                      16
maintenance_cost                 623
num_bedrooms                     115
```

```
num_floors_in_building           650
num_full_bathrooms                 0
num_total_rooms                    2
parking_charges                 1671
sale_price                      1700
sq_footage                      1210
walk_score                         0
listing_price_to_nearest_1000    534
coop_condo                         0
common_charges                  1684
zipcode                           15
dtype: int64
```

```
datan = datan.set_index('zipcode')
a_z = a_z.set_index('zips')
datan = pd.merge(datan, a_z, left_index=True, right_index=True)
datan
```

```
       approx_year_built cats_allowed dining_room_type dogs_allowed
fuel_type  \
11004             1950.0          yes              NaN          yes
oil
11004             1950.0          yes            combo          yes
oil
11004             1950.0          yes           formal          yes
oil
11004             1950.0          yes            other          yes
gas
11004             1950.0          yes            combo          yes
gas
...                  ...          ...              ...          ...
...
11435             1950.0          yes           formal          yes
NaN
11435             1956.0          yes            combo          yes
NaN
11435             1956.0          yes           formal          yes
gas
11435             1959.0           no           formal           no
gas
11435             1952.0           no              NaN           no
oil

                          full_address_or_zip_code garage_exists  \
11004  71-12 Little Neck Pky,  Glen Oaks NY, 11004           NaN
11004   264-03B Langston Ave,  Glen Oaks NY, 11004           NaN
11004        255-17 74th Ave,  Glen Oaks NY, 11004           NaN
11004         73-43 255th St,  Glen Oaks NY, 11004           NaN
11004         70-52 260th St,  Glen Oaks NY, 11004           NaN
...                                            ...           ...
```

```
11435      150-77 Village Rd,  Briarwood NY, 11435           NaN
11435          84-01 Main St,  Briarwood NY, 11435           NaN
11435      84-01 Main Street,  Briarwood NY, 11435           NaN
11435    84-31 Van Wyck Expy,  Briarwood NY, 11435           yes
11435       84-55 Daniels St,  Briarwood NY, 11435           NaN

             kitchen_type maintenance_cost  num_bedrooms  ...  \
11004               Combo             $684           1.0  ...
11004              eat in             $698           2.0  ...
11004              eat in             $698           2.0  ...
11004              eat in             $765           3.0  ...
11004              eat in             $789           3.0  ...
...                   ...              ...           ...  ...
11435   efficiency kitchen          $1,310           2.0  ...
11435               eatin             $933           2.0  ...
11435               eatin             $983           2.0  ...
11435               eatin             $805           2.0  ...
11435               eatin             $393           1.0  ...

       num_full_bathrooms  num_total_rooms  parking_charges sale_price
\
11004                   1              4.0              NaN  $135,000

11004                   1              4.0              NaN  $267,000

11004                   1              3.0              NaN  $275,000

11004                   1              5.0              NaN  $298,000

11004                   1              5.0              NaN  $301,000

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

11435                   1              5.0             $75         NaN

11435                   1              5.0             $75         NaN

11435                   1              5.0             $90         NaN

11435                   1              5.0              NaN        NaN

11435                   1              2.0              NaN        NaN


      sq_footage  walk_score  listing_price_to_nearest_1000 coop_condo
\
11004      590.0          58                            NaN      co-op

11004        NaN          55                            NaN      co-op
```

```
11004         NaN          77                              NaN      co-op

11004       812.0          77                              NaN      co-op

11004       856.0          58                              NaN      co-op

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

11435      1000.0          87                            $210       co-op

11435         NaN          82                            $229       co-op

11435      1200.0          82                            $283       co-op

11435         NaN          93                            $239       co-op

11435         NaN          75                            $145       co-op


      common_charges avg_prices
11004            NaN   316500.0
11004            NaN   316500.0
11004            NaN   316500.0
11004            NaN   316500.0
11004            NaN   316500.0
...              ...        ...
11435            NaN   274000.0
11435            NaN   274000.0
11435            NaN   274000.0
11435            NaN   274000.0
11435            NaN   274000.0

[2300 rows x 21 columns]
```

```
data['kitchen_type'].unique()
```

```
array(['eat in', 'efficiency', 'Combo', 'combo', 'Eat In', nan, 'Eat
in',
       '1955', 'eatin', 'efficiency kitchene', 'efficiency kitchen',
       'efficiemcy', 'none', 'efficiency ktchen'], dtype=object)
```

```
datan.cats_allowed=datan.cats_allowed.replace('y',"yes")
datan.dogs_allowed=datan.dogs_allowed.replace('yes89',"yes")
datan.kitchen_type = datan.kitchen_type.replace(['eat in','eatin','Eat
in'],'Eat In')
datan.kitchen_type = datan.kitchen_type.replace(['efficiency
kitchene','efficiency kitchen','efficiemcy',
                                                'efficiency
ktchen'],'efficiency')
```

```
datan.kitchen_type = datan.kitchen_type.replace('combo','Combo')
datan.fuel_type = datan.fuel_type.replace('Other','other')
datan =  pd.get_dummies(datan,columns=['cats_allowed',
'dogs_allowed','coop_condo'],drop_first=True)
```

```
datan.kitchen_type=datan.kitchen_type.replace('none',0)
datan.kitchen_type=datan.kitchen_type.replace('Eat In',1)
datan.kitchen_type=datan.kitchen_type.replace('efficiency',2)
datan.kitchen_type=datan.kitchen_type.replace('1955',3)
datan.kitchen_type=datan.kitchen_type.replace('Combo',4)
datan['kitchen_type'].unique()
```

```
array([ 4.,  1.,  2.,  0., nan,  3.])
```

```
datan
```

```
       approx_year_built dining_room_type fuel_type  \
11004             1950.0              NaN       oil
11004             1950.0            combo       oil
11004             1950.0           formal       oil
11004             1950.0            other       gas
11004             1950.0            combo       gas
...                  ...              ...       ...
11435             1950.0           formal       NaN
11435             1956.0            combo       NaN
11435             1956.0           formal       gas
11435             1959.0           formal       gas
11435             1952.0              NaN       oil

                          full_address_or_zip_code garage_exists  \
11004  71-12 Little Neck Pky,  Glen Oaks NY, 11004           NaN
11004   264-03B Langston Ave,  Glen Oaks NY, 11004           NaN
11004        255-17 74th Ave,  Glen Oaks NY, 11004           NaN
11004         73-43 255th St,  Glen Oaks NY, 11004           NaN
11004         70-52 260th St,  Glen Oaks NY, 11004           NaN
...                                            ...           ...
11435       150-77 Village Rd,  Briarwood NY, 11435           NaN
11435           84-01 Main St,  Briarwood NY, 11435           NaN
11435       84-01 Main Street,  Briarwood NY, 11435           NaN
11435     84-31 Van Wyck Expy,  Briarwood NY, 11435           yes
11435        84-55 Daniels St,  Briarwood NY, 11435           NaN

       kitchen_type maintenance_cost  num_bedrooms
num_floors_in_building  \
11004           4.0             $684           1.0
2.0
11004           1.0             $698           2.0
2.0
11004           1.0             $698           2.0
2.0
11004           1.0             $765           3.0
```

```
2.0
11004           1.0           $789           3.0
2.0
...             ...             ...          ...
...
11435           2.0         $1,310           2.0
2.0
11435           1.0           $933           2.0
NaN
11435           1.0           $983           2.0
7.0
11435           1.0           $805           2.0
NaN
11435           1.0           $393           1.0
1.0

       num_full_bathrooms  ...  parking_charges sale_price sq_footage
\
11004                   1  ...              NaN  $135,000       590.0

11004                   1  ...              NaN  $267,000         NaN

11004                   1  ...              NaN  $275,000         NaN

11004                   1  ...              NaN  $298,000       812.0

11004                   1  ...              NaN  $301,000       856.0

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

11435                   1  ...             $75         NaN      1000.0

11435                   1  ...             $75         NaN         NaN

11435                   1  ...             $90         NaN      1200.0

11435                   1  ...              NaN        NaN         NaN

11435                   1  ...              NaN        NaN         NaN


       walk_score  listing_price_to_nearest_1000 common_charges
avg_prices  \
11004          58                            NaN            NaN
316500.0
11004          55                            NaN            NaN
316500.0
11004          77                            NaN            NaN
316500.0
```

```
11004          77                          NaN           NaN
316500.0
11004          58                          NaN           NaN
316500.0
...           ...                          ...           ...
...
11435          87                        $210            NaN
274000.0
11435          82                        $229            NaN
274000.0
11435          82                        $283            NaN
274000.0
11435          93                        $239            NaN
274000.0
11435          75                        $145            NaN
274000.0

       cats_allowed_yes  dogs_allowed_yes  coop_condo_condo
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
...                 ...               ...               ...
11435                 1                 1                 0
11435                 1                 1                 0
11435                 1                 1                 0
11435                 0                 0                 0
11435                 0                 0                 0

[2300 rows x 21 columns]
```

```
datan.fuel_type=datan.fuel_type.replace('none',0)
datan.fuel_type=datan.fuel_type.replace('electric',1)
datan.fuel_type=datan.fuel_type.replace('gas',2)
datan.fuel_type=datan.fuel_type.replace('oil',3)
datan.fuel_type=datan.fuel_type.replace('other',4)
datan["fuel_type"].unique()
```

```
array([ 3.,  2., nan,  4.,  1.,  0.])
```

```
datan
```

```
       approx_year_built dining_room_type  fuel_type  \
11004             1950.0              NaN        3.0
11004             1950.0            combo        3.0
11004             1950.0           formal        3.0
11004             1950.0            other        2.0
11004             1950.0            combo        2.0
...                  ...              ...        ...
11435             1950.0           formal        NaN
```

```
11435             1956.0           combo        NaN
11435             1956.0          formal        2.0
11435             1959.0          formal        2.0
11435             1952.0             NaN        3.0

                          full_address_or_zip_code garage_exists  \
11004  71-12 Little Neck Pky,  Glen Oaks NY, 11004           NaN
11004   264-03B Langston Ave,  Glen Oaks NY, 11004           NaN
11004        255-17 74th Ave,  Glen Oaks NY, 11004           NaN
11004         73-43 255th St,  Glen Oaks NY, 11004           NaN
11004         70-52 260th St,  Glen Oaks NY, 11004           NaN
...                                            ...           ...
11435      150-77 Village Rd,  Briarwood NY, 11435           NaN
11435          84-01 Main St,  Briarwood NY, 11435           NaN
11435      84-01 Main Street,  Briarwood NY, 11435           NaN
11435    84-31 Van Wyck Expy,  Briarwood NY, 11435           yes
11435       84-55 Daniels St,  Briarwood NY, 11435           NaN

       kitchen_type maintenance_cost  num_bedrooms
num_floors_in_building  \
11004           4.0            $684           1.0
2.0
11004           1.0            $698           2.0
2.0
11004           1.0            $698           2.0
2.0
11004           1.0            $765           3.0
2.0
11004           1.0            $789           3.0
2.0
...             ...              ...          ...
...
11435           2.0          $1,310           2.0
2.0
11435           1.0            $933           2.0
NaN
11435           1.0            $983           2.0
7.0
11435           1.0            $805           2.0
NaN
11435           1.0            $393           1.0
1.0

       num_full_bathrooms  ...  parking_charges sale_price sq_footage
\
11004                   1  ...              NaN  $135,000      590.0

11004                   1  ...              NaN  $267,000        NaN

11004                   1  ...              NaN  $275,000        NaN
```

```
11004                    1  ...              NaN  $298,000       812.0

11004                    1  ...              NaN  $301,000       856.0

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

11435                    1  ...             $75        NaN      1000.0

11435                    1  ...             $75        NaN         NaN

11435                    1  ...             $90        NaN      1200.0

11435                    1  ...              NaN       NaN         NaN

11435                    1  ...              NaN       NaN         NaN


       walk_score  listing_price_to_nearest_1000 common_charges
avg_prices  \
11004         58                             NaN            NaN
316500.0
11004         55                             NaN            NaN
316500.0
11004         77                             NaN            NaN
316500.0
11004         77                             NaN            NaN
316500.0
11004         58                             NaN            NaN
316500.0
...          ...                             ...            ...
...
11435         87                           $210             NaN
274000.0
11435         82                           $229             NaN
274000.0
11435         82                           $283             NaN
274000.0
11435         93                           $239             NaN
274000.0
11435         75                           $145             NaN
274000.0

       cats_allowed_yes  dogs_allowed_yes  coop_condo_condo
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
```

```
...                    ...               ...               ...
11435                    1                 1                 0
11435                    1                 1                 0
11435                    1                 1                 0
11435                    0                 0                 0
11435                    0                 0                 0

[2300 rows x 21 columns]
```

```
datan.dining_room_type=datan.dining_room_type.replace('none',0)
datan.dining_room_type=datan.dining_room_type.replace('combo',1)
datan.dining_room_type=datan.dining_room_type.replace('dining area',2)
datan.dining_room_type=datan.dining_room_type.replace('formal',3)
datan.dining_room_type=datan.dining_room_type.replace('other',4)
datan["dining_room_type"].unique()
```

```
array([nan,  1.,  3.,  4.,  2.,  0.])
```

```
datan['sale_price'] = datan['sale_price'].replace({'\$': '', ',': ''},
regex=True)
datan['listing_price_to_nearest_1000'] =
datan['listing_price_to_nearest_1000'].replace({'\$': '', ',': ''},
regex=True)
datan['parking_charges'] = datan['parking_charges'].replace({'\$': '',
',': ''}, regex=True)
datan['maintenance_cost'] = datan['maintenance_cost'].replace({'\$':
'', ',': ''}, regex=True)
datan["common_charges"] = datan["common_charges"].replace({'\$': '',
',': ''}, regex=True)
datan['sale_price'] = datan['sale_price'].astype(float)
del datan['full_address_or_zip_code']
del datan['garage_exists']
```

```
del datan['common_charges']
```

```
datan
```

```
       approx_year_built  dining_room_type  fuel_type  kitchen_type  \
11004             1950.0               NaN        3.0           4.0
11004             1950.0               1.0        3.0           1.0
11004             1950.0               3.0        3.0           1.0
11004             1950.0               4.0        2.0           1.0
11004             1950.0               1.0        2.0           1.0
...                  ...               ...        ...           ...
11435             1950.0               3.0        NaN           2.0
11435             1956.0               1.0        NaN           1.0
11435             1956.0               3.0        2.0           1.0
11435             1959.0               3.0        2.0           1.0
11435             1952.0               NaN        3.0           1.0

      maintenance_cost  num_bedrooms  num_floors_in_building  \
```

```
11004             684          1.0                        2.0
11004             698          2.0                        2.0
11004             698          2.0                        2.0
11004             765          3.0                        2.0
11004             789          3.0                        2.0
...               ...          ...                        ...
11435            1310          2.0                        2.0
11435             933          2.0                        NaN
11435             983          2.0                        7.0
11435             805          2.0                        NaN
11435             393          1.0                        1.0

       num_full_bathrooms  num_total_rooms parking_charges  sale_price
\
11004                   1              4.0             NaN    135000.0

11004                   1              4.0             NaN    267000.0

11004                   1              3.0             NaN    275000.0

11004                   1              5.0             NaN    298000.0

11004                   1              5.0             NaN    301000.0

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

11435                   1              5.0             75          NaN

11435                   1              5.0             75          NaN

11435                   1              5.0             90          NaN

11435                   1              5.0             NaN         NaN

11435                   1              2.0             NaN         NaN


       sq_footage  walk_score listing_price_to_nearest_1000
avg_prices  \
11004       590.0          58                            NaN
316500.0
11004         NaN          55                            NaN
316500.0
11004         NaN          77                            NaN
316500.0
11004       812.0          77                            NaN
316500.0
11004       856.0          58                            NaN
316500.0
```

```
...           ...         ...                              ...         ..
.
11435      1000.0          87                              210
274000.0
11435         NaN          82                              229
274000.0
11435      1200.0          82                              283
274000.0
11435         NaN          93                              239
274000.0
11435         NaN          75                              145
274000.0

       cats_allowed_yes  dogs_allowed_yes  coop_condo_condo
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
11004                 1                 1                 0
...                 ...               ...               ...
11435                 1                 1                 0
11435                 1                 1                 0
11435                 1                 1                 0
11435                 0                 0                 0
11435                 0                 0                 0

[2300 rows x 18 columns]
```

```
datan = datan.reset_index()
datan = datan.rename(columns={"index":'zipcode'})
datan['zipcode'] = datan['zipcode'].astype(float)
datan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2300 entries, 0 to 2299
Data columns (total 19 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   zipcode                       2300 non-null   float64
 1   approx_year_built             2257 non-null   float64
 2   dining_room_type              1841 non-null   float64
 3   fuel_type                     2183 non-null   float64
 4   kitchen_type                  2286 non-null   float64
 5   maintenance_cost              1662 non-null   object 
 6   num_bedrooms                  2182 non-null   float64
 7   num_floors_in_building        1622 non-null   float64
 8   num_full_bathrooms            2300 non-null   int64  
 9   num_total_rooms               2298 non-null   float64
 10  parking_charges               527 non-null    object 
 11  sale_price                    555 non-null    float64
```

```
 12  sq_footage                     1048 non-null   float64
 13  walk_score                     2300 non-null   int64
 14  listing_price_to_nearest_1000  1743 non-null   object
 15  avg_prices                     2300 non-null   float64
 16  cats_allowed_yes               2300 non-null   uint8
 17  dogs_allowed_yes               2300 non-null   uint8
 18  coop_condo_condo               2300 non-null   uint8
dtypes: float64(11), int64(2), object(3), uint8(3)
memory usage: 294.4+ KB
```

datan

```
      zipcode  approx_year_built  dining_room_type  fuel_type
kitchen_type  \
0     11004.0             1950.0               NaN        3.0
4.0
1     11004.0             1950.0               1.0        3.0
1.0
2     11004.0             1950.0               3.0        3.0
1.0
3     11004.0             1950.0               4.0        2.0
1.0
4     11004.0             1950.0               1.0        2.0
1.0
...       ...                ...               ...        ...
...
2295  11435.0             1950.0               3.0        NaN
2.0
2296  11435.0             1956.0               1.0        NaN
1.0
2297  11435.0             1956.0               3.0        2.0
1.0
2298  11435.0             1959.0               3.0        2.0
1.0
2299  11435.0             1952.0               NaN        3.0
1.0

     maintenance_cost  num_bedrooms  num_floors_in_building  \
0                 684           1.0                     2.0
1                 698           2.0                     2.0
2                 698           2.0                     2.0
3                 765           3.0                     2.0
4                 789           3.0                     2.0
...               ...           ...                     ...
2295             1310           2.0                     2.0
2296              933           2.0                     NaN
2297              983           2.0                     7.0
2298              805           2.0                     NaN
2299              393           1.0                     1.0
```

```
      num_full_bathrooms  num_total_rooms parking_charges  sale_price  \
0                      1              4.0             NaN    135000.0
1                      1              4.0             NaN    267000.0
2                      1              3.0             NaN    275000.0
3                      1              5.0             NaN    298000.0
4                      1              5.0             NaN    301000.0
...                  ...              ...             ...         ...
2295                   1              5.0             75          NaN
2296                   1              5.0             75          NaN
2297                   1              5.0             90          NaN
2298                   1              5.0             NaN         NaN
2299                   1              2.0             NaN         NaN

      sq_footage  walk_score listing_price_to_nearest_1000  avg_prices  \
0          590.0          58                           NaN    316500.0
1            NaN          55                           NaN    316500.0
2            NaN          77                           NaN    316500.0
3          812.0          77                           NaN    316500.0
4          856.0          58                           NaN    316500.0
...          ...         ...                           ...         ...
2295      1000.0          87                          210     274000.0
2296         NaN          82                          229     274000.0
2297      1200.0          82                          283     274000.0
2298         NaN          93                          239     274000.0
2299         NaN          75                          145     274000.0
```

```
      cats_allowed_yes  dogs_allowed_yes  coop_condo_condo
0                    1                 1                 0
1                    1                 1                 0
2                    1                 1                 0
3                    1                 1                 0
4                    1                 1                 0
...                ...               ...               ...
2295                 1                 1                 0
2296                 1                 1                 0
2297                 1                 1                 0
2298                 0                 0                 0
2299                 0                 0                 0

[2300 rows x 19 columns]
```

```
data_sp = datan[~datan['sale_price'].isnull()]
data_sp_vals = data_sp.values.astype(float)
data_sp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 555 entries, 0 to 2262
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   zipcode                         555 non-null    float64
 1   approx_year_built               548 non-null    float64
 2   dining_room_type                432 non-null    float64
 3   fuel_type                       529 non-null    float64
 4   kitchen_type                    549 non-null    float64
 5   maintenance_cost                405 non-null    object
 6   num_bedrooms                    555 non-null    float64
 7   num_floors_in_building          437 non-null    float64
 8   num_full_bathrooms              555 non-null    int64
 9   num_total_rooms                 555 non-null    float64
 10  parking_charges                 135 non-null    object
 11  sale_price                      555 non-null    float64
 12  sq_footage                      230 non-null    float64
 13  walk_score                      555 non-null    int64
 14  listing_price_to_nearest_1000  2 non-null      object
 15  avg_prices                      555 non-null    float64
 16  cats_allowed_yes                555 non-null    uint8
 17  dogs_allowed_yes                555 non-null    uint8
 18  coop_condo_condo                555 non-null    uint8
dtypes: float64(11), int64(2), object(3), uint8(3)
memory usage: 75.3+ KB
```

```
imputer = MissForest()
X_imputed = imputer.fit_transform(data_sp_vals)
```

```python
t_data = pd.DataFrame()
t_data['zipcode'] = X_imputed[:, 0]
t_data['approx_year_built'] = X_imputed[:, 1]
t_data['dining_room_type'] = X_imputed[:, 2]
t_data['fuel_type'] = X_imputed[:, 3]
t_data['kitchen_type'] = X_imputed[:, 4]
t_data['maintenance_cost'] = X_imputed[:, 5]
t_data['num_bedrooms'] = X_imputed[:, 6]
t_data['num_floors_in_building'] = X_imputed[:, 7]
t_data['num_full_bathrooms'] = X_imputed[:, 8]
t_data['num_total_rooms'] = X_imputed[:, 9]
t_data['parking_charges'] = X_imputed[:, 10]
t_data['sale_price'] = X_imputed[:,11]
t_data['sq_footage'] = X_imputed[:,12]
t_data['walk_score'] = X_imputed[:,13]
t_data['price_listings'] = X_imputed[:,14]*1000
t_data['avg_prices'] = X_imputed[:,15]
t_data['cats_allowed'] = X_imputed[:,16]
t_data['dogs_allowed'] = X_imputed[:,17]
t_data['coop_condo'] = X_imputed[:,18]
t_data['price_per_sqft'] = X_imputed[:,11]/X_imputed[:,12]
t_data
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4

     zipcode  approx_year_built  dining_room_type  fuel_type
kitchen_type  \
0    11004.0             1950.0              2.01        3.0
4.0
1    11004.0             1950.0              1.00        3.0
1.0
2    11004.0             1950.0              3.00        3.0
1.0
3    11004.0             1950.0              4.00        2.0
1.0
4    11004.0             1950.0              1.00        2.0
1.0
..       ...                ...               ...        ...
...
550  11435.0             1952.0              1.00        3.0
2.0
551  11435.0             1958.0              1.00        3.0
2.0
552  11435.0             1950.0              1.00        2.0
4.0
553  11435.0             1950.0              1.94        3.0
2.0
```

```
554  11435.0             1950.0              1.34        2.0
1.0

     maintenance_cost  num_bedrooms  num_floors_in_building  \
0               684.0           1.0                     2.0
1               698.0           2.0                     2.0
2               698.0           2.0                     2.0
3               765.0           3.0                     2.0
4               789.0           3.0                     2.0
..                ...           ...                     ...
550             723.0           1.0                     6.0
551             740.0           1.0                     7.0
552             503.0           1.0                     6.0
553             852.0           1.0                     2.0
554             725.0           2.0                     6.0

     num_full_bathrooms  num_total_rooms  parking_charges  sale_price
\
0                   1.0              4.0           233.49    135000.0

1                   1.0              4.0           230.40    267000.0

2                   1.0              3.0           231.54    275000.0

3                   1.0              5.0           243.66    298000.0

4                   1.0              5.0           233.22    301000.0

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

550                 1.0              4.0           100.00    145000.0

551                 1.0              3.0            70.00    158000.0

552                 1.0              3.0           125.00    142000.0

553                 1.0              3.0            74.71    113000.0

554                 1.0              4.0           125.00    216000.0


     sq_footage  walk_score  price_listings  avg_prices  cats_allowed
\
0        590.00        58.0        358800.0    316500.0           1.0

1        804.28        55.0        405200.0    316500.0           1.0

2        807.10        77.0        387800.0    316500.0           1.0
```

```
3        812.00        77.0         416800.0    316500.0            1.0

4        856.00        58.0         469000.0    316500.0            1.0

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

550      710.13        83.0         445800.0    274000.0            0.0

551      750.00        85.0         422600.0    274000.0            1.0

552      750.00        83.0         440000.0    274000.0            0.0

553      756.84        78.0         399400.0    274000.0            0.0

554      907.98        83.0         503800.0    274000.0            0.0


     dogs_allowed  coop_condo  price_per_sqft
0             1.0         0.0      228.813559
1             1.0         0.0      331.973939
2             1.0         0.0      340.726056
3             1.0         0.0      366.995074
4             1.0         0.0      351.635514
..            ...         ...             ...
550           0.0         0.0      204.187966
551           1.0         0.0      210.666667
552           0.0         0.0      189.333333
553           0.0         0.0      149.305005
554           0.0         0.0      237.890702

[555 rows x 20 columns]
```

```
t_data.describe().T
```

```
                        count          mean          std
min  \
zipcode                 555.0  11359.482883    78.685461
11004.000000
approx_year_built       555.0   1961.214360    20.808743
1915.000000
dining_room_type        555.0      2.005099     1.084671
1.000000
fuel_type               555.0      2.378505     0.587780
0.000000
kitchen_type            555.0      1.919586     1.006281
1.000000
maintenance_cost        555.0    803.237622   368.784713
155.000000
```

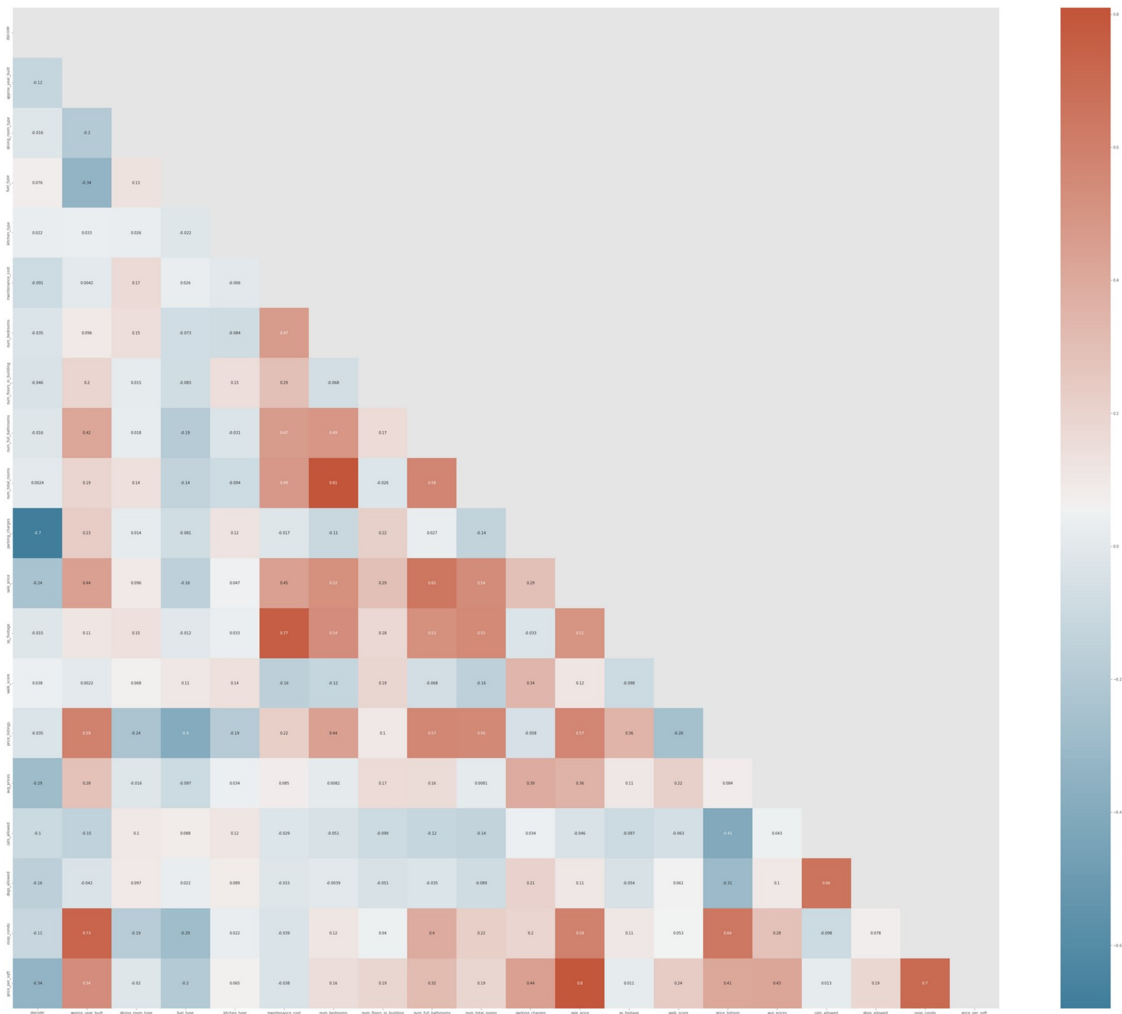
```
num_bedrooms             555.0       1.538739       0.750351
0.000000
num_floors_in_building  555.0       6.826432       5.798692
1.000000
num_full_bathrooms      555.0       1.196396       0.415392
1.000000
num_total_rooms         555.0       4.012613       1.200866
1.000000
parking_charges         555.0     109.433946      65.204117
9.000000
sale_price              555.0  316303.720721  176853.795570
55000.000000
sq_footage              555.0     891.482198     361.562259
375.000000
walk_score              555.0      84.142342      12.932046
15.000000
price_listings          555.0  455383.063063   64487.218681
179000.000000
avg_prices              555.0  418205.219398  118709.292833
190000.000000
cats_allowed            555.0       0.470270       0.499566
0.000000
dogs_allowed            555.0       0.290090       0.454213
0.000000
coop_condo              555.0       0.239640       0.427249
0.000000
price_per_sqft          555.0     354.069446     169.241807
78.680611

                                  25%            50%            75%  \
zipcode                  11360.000000   11372.000000   11375.000000
approx_year_built         1950.000000    1955.000000    1965.000000
dining_room_type             1.000000       1.520000       3.000000
fuel_type                    2.000000       2.000000       3.000000
kitchen_type                 1.000000       2.000000       2.000000
maintenance_cost           605.000000     711.000000     870.915000
num_bedrooms                 1.000000       1.000000       2.000000
num_floors_in_building       3.000000       6.000000       7.000000
num_full_bathrooms           1.000000       1.000000       1.000000
num_total_rooms              3.000000       4.000000       5.000000
parking_charges             73.350000     100.000000     128.265000
sale_price              175000.000000  265000.000000  429250.000000
sq_footage                 709.585000     817.790000     976.800000
walk_score                  76.500000      87.000000      95.000000
price_listings          416800.000000  445800.000000  492200.000000
avg_prices              318250.000000  412000.000000  490000.000000
cats_allowed                 0.000000       0.000000       1.000000
dogs_allowed                 0.000000       0.000000       1.000000
coop_condo                   0.000000       0.000000       0.000000
price_per_sqft             236.973892     301.896557     444.039068
```

```
                                    max
zipcode                     11435.00000
approx_year_built            2016.00000
dining_room_type                4.00000
fuel_type                       4.00000
kitchen_type                    4.00000
maintenance_cost             4659.00000
num_bedrooms                    3.00000
num_floors_in_building         33.00000
num_full_bathrooms              3.00000
num_total_rooms                 8.00000
parking_charges               500.00000
sale_price                 999999.00000
sq_footage                   6215.00000
walk_score                     99.00000
price_listings             759000.00000
avg_prices                 804446.00000
cats_allowed                    1.00000
dogs_allowed                    1.00000
coop_condo                      1.00000
price_per_sqft               1163.56383
```

```
import seaborn as sns
corr = t_data.corr()
f, ax = plt.subplots(figsize=(60, 50))
mask = np.triu(np.ones_like(corr, dtype=bool))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(corr, annot=True, mask = mask, cmap=cmap)
```

```
<AxesSubplot:>
```



```
from pandas.plotting import scatter_matrix
scatter_matrix(t_data,figsize=(50,50),alpha=0.8)
```

```
array([[<AxesSubplot:xlabel='zipcode', ylabel='zipcode'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='zipcode'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='zipcode'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='zipcode'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='zipcode'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='zipcode'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='zipcode'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='zipcode'>,
        <AxesSubplot:xlabel='num_full_bathrooms', ylabel='zipcode'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='zipcode'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='zipcode'>,
        <AxesSubplot:xlabel='sale_price', ylabel='zipcode'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='zipcode'>,
        <AxesSubplot:xlabel='walk_score', ylabel='zipcode'>,
        <AxesSubplot:xlabel='price_listings', ylabel='zipcode'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='zipcode'>,
```

```
        <AxesSubplot:xlabel='cats_allowed', ylabel='zipcode'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='zipcode'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='zipcode'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='zipcode'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='kitchen_type',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='num_bedrooms',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='sale_price', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='walk_score', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='cats_allowed',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='dogs_allowed',
ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='approx_year_built'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='approx_year_built'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='kitchen_type',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='num_bedrooms',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='num_floors_in_building',
```

```
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='sale_price', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='walk_score', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='cats_allowed',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='dogs_allowed',
ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='dining_room_type'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='dining_room_type'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='fuel_type'>,
        <AxesSubplot:xlabel='num_full_bathrooms', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='sale_price', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='walk_score', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='price_listings', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='fuel_type'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='fuel_type'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='kitchen_type'>,
```

```
        <AxesSubplot:xlabel='num_bedrooms', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='sale_price', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='walk_score', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='price_listings', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='kitchen_type'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='kitchen_type'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='kitchen_type',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='num_bedrooms',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='sale_price', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='walk_score', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='cats_allowed',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='dogs_allowed',
ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='maintenance_cost'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='maintenance_cost'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='num_bedrooms'>,
```

```
        <AxesSubplot:xlabel='approx_year_built',
ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='sale_price', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='walk_score', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='price_listings', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='num_bedrooms'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='num_bedrooms'>],
       [<AxesSubplot:xlabel='zipcode',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='fuel_type',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='kitchen_type',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='num_bedrooms',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='sale_price',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='sq_footage',
ylabel='num_floors_in_building'>,
```

```
        <AxesSubplot:xlabel='walk_score',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='avg_prices',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='cats_allowed',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='dogs_allowed',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='coop_condo',
ylabel='num_floors_in_building'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='num_floors_in_building'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='kitchen_type',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='num_bedrooms',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='sale_price',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='sq_footage',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='walk_score',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='avg_prices',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='cats_allowed',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='dogs_allowed',
ylabel='num_full_bathrooms'>,
        <AxesSubplot:xlabel='coop_condo',
ylabel='num_full_bathrooms'>,
```

```
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='num_full_bathrooms'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='sale_price', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='walk_score', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='num_total_rooms'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='num_total_rooms'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='sale_price', ylabel='parking_charges'>,
```

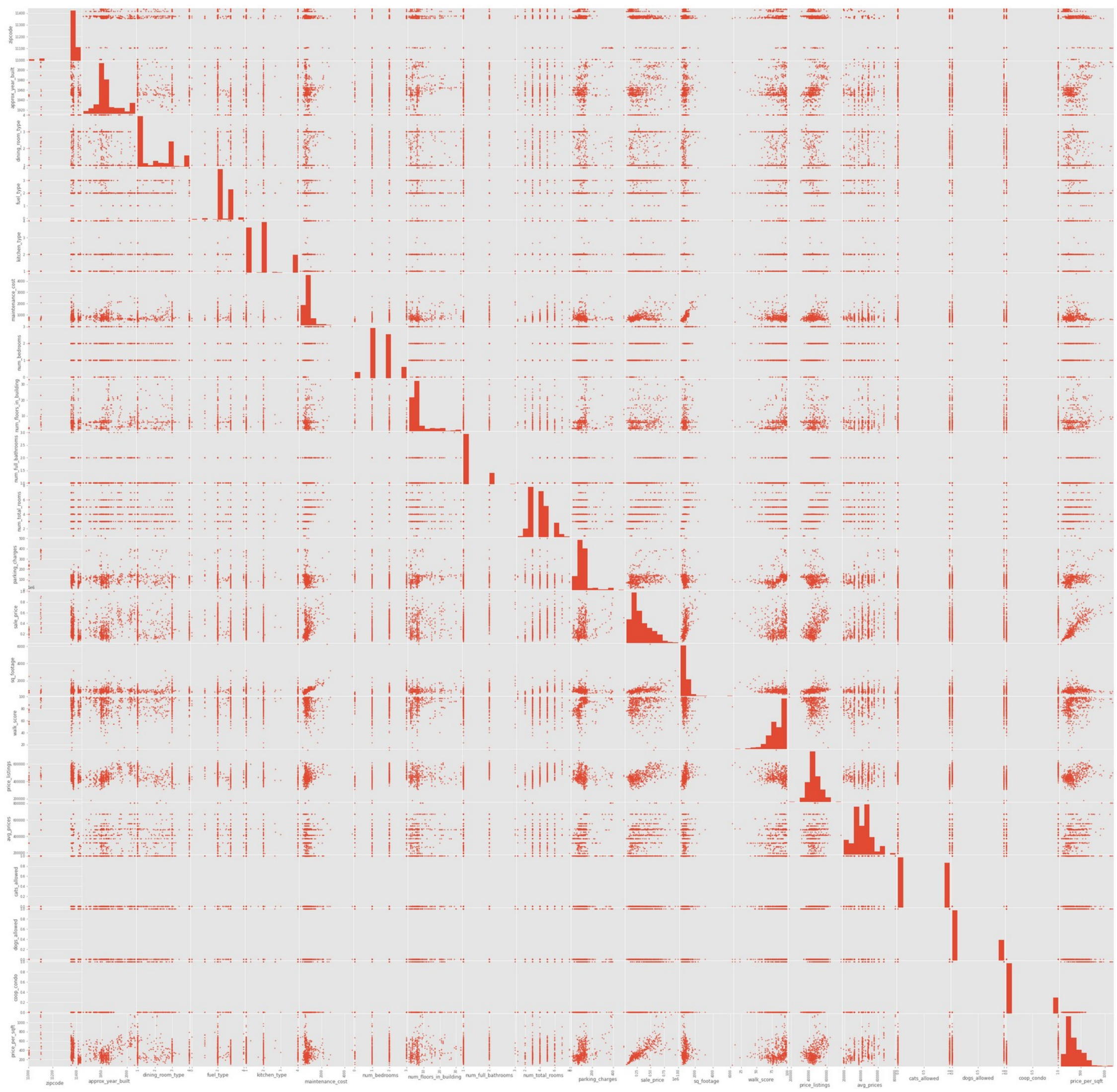
```
        <AxesSubplot:xlabel='sq_footage', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='walk_score', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='parking_charges'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='parking_charges'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='parking_charges'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='sale_price'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='sale_price'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='sale_price'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='sale_price'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='sale_price'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='sale_price'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='sale_price'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='sale_price'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='sale_price'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='sale_price'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='sale_price'>,
        <AxesSubplot:xlabel='sale_price', ylabel='sale_price'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='sale_price'>,
        <AxesSubplot:xlabel='walk_score', ylabel='sale_price'>,
        <AxesSubplot:xlabel='price_listings', ylabel='sale_price'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='sale_price'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='sale_price'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='sale_price'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='sale_price'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='sale_price'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='sq_footage'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='sq_footage'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='sale_price', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='walk_score', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='price_listings', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='sq_footage'>,
```

```
        <AxesSubplot:xlabel='cats_allowed', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='sq_footage'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='sq_footage'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='walk_score'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='walk_score'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='walk_score'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='walk_score'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='walk_score'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='walk_score'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='walk_score'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='walk_score'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='walk_score'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='walk_score'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='walk_score'>,
        <AxesSubplot:xlabel='sale_price', ylabel='walk_score'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='walk_score'>,
        <AxesSubplot:xlabel='walk_score', ylabel='walk_score'>,
        <AxesSubplot:xlabel='price_listings', ylabel='walk_score'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='walk_score'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='walk_score'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='walk_score'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='walk_score'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='walk_score'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='price_listings'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='price_listings'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='price_listings'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='price_listings'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='sale_price', ylabel='price_listings'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='price_listings'>,
        <AxesSubplot:xlabel='walk_score', ylabel='price_listings'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='price_listings'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='price_listings'>,
```

```
        <AxesSubplot:xlabel='cats_allowed', ylabel='price_listings'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='price_listings'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='price_listings'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='price_listings'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='avg_prices'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='avg_prices'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='sale_price', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='walk_score', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='price_listings', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='avg_prices'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='avg_prices'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='sale_price', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='walk_score', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='price_listings', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='cats_allowed'>,
```

```
        <AxesSubplot:xlabel='coop_condo', ylabel='cats_allowed'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='cats_allowed'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='approx_year_built',
ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='sale_price', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='walk_score', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='price_listings', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='dogs_allowed'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='dogs_allowed'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='approx_year_built', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='dining_room_type', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='maintenance_cost', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='coop_condo'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='coop_condo'>,
        <AxesSubplot:xlabel='num_total_rooms', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='parking_charges', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='sale_price', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='walk_score', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='price_listings', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='coop_condo'>,
        <AxesSubplot:xlabel='price_per_sqft', ylabel='coop_condo'>],
       [<AxesSubplot:xlabel='zipcode', ylabel='price_per_sqft'>,
```

```
        <AxesSubplot:xlabel='approx_year_built',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='dining_room_type',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='fuel_type', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='kitchen_type', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='maintenance_cost',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='num_bedrooms', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='num_floors_in_building',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='num_full_bathrooms',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='num_total_rooms',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='parking_charges',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='sale_price', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='sq_footage', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='walk_score', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='price_listings',
ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='avg_prices', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='cats_allowed', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='dogs_allowed', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='coop_condo', ylabel='price_per_sqft'>,
        <AxesSubplot:xlabel='price_per_sqft',
ylabel='price_per_sqft'>]],
      dtype=object)
```



```
t_data['zipcode'] = t_data['zipcode'].astype(int)
t_data['approx_year_built'] = t_data['approx_year_built'].astype(int)
t_data['dining_room_type'] = t_data['dining_room_type'].astype(int)
t_data['num_bedrooms'] = t_data['num_bedrooms'].astype(int)
t_data['num_floors_in_building'] =
t_data['num_floors_in_building'].astype(int)
t_data['num_full_bathrooms'] =
t_data['num_full_bathrooms'].astype(int)
t_data['num_total_rooms'] = t_data['num_total_rooms'].astype(int)
t_data['dogs_allowed'] = t_data['dogs_allowed'].astype(int)
t_data['cats_allowed'] = t_data['cats_allowed'].astype(int)
t_data['fuel_type'] = t_data['fuel_type'].astype(int)
t_data = t_data[~(t_data['num_bedrooms'] <= 0)]
t_data.describe().T
```

```
                        count          mean          std
min  \
zipcode                 524.0  11359.368321    79.136979
```

```
11004.000000
approx_year_built       524.0    1961.374046      21.048459
1915.000000
dining_room_type        524.0       1.938931       1.102791
1.000000
fuel_type               524.0       2.349237       0.591672
0.000000
kitchen_type            524.0       1.916737       1.012293
1.000000
maintenance_cost        524.0     820.488664     371.769747
155.000000
num_bedrooms            524.0       1.629771       0.669145
1.000000
num_floors_in_building  524.0       6.580153       5.737674
1.000000
num_full_bathrooms      524.0       1.208015       0.424684
1.000000
num_total_rooms         524.0       4.124046       1.134322
1.000000
parking_charges         524.0     107.518034      65.023422
9.000000
sale_price              524.0  324369.230916  177243.836326
66000.000000
sq_footage              524.0     912.322748     361.176158
450.000000
walk_score              524.0      83.723282      13.052100
15.000000
price_listings          524.0  458440.458015   64163.988374
179000.000000
avg_prices              524.0  416206.967111  118140.724637
190000.000000
cats_allowed            524.0       0.471374       0.499657
0.000000
dogs_allowed            524.0       0.286260       0.452444
0.000000
coop_condo              524.0       0.246183       0.431198
0.000000
price_per_sqft          524.0     355.209905     168.857888
78.680611

                                  25%           50%          75%  \
zipcode                 11360.000000  11372.000000  11375.00000
approx_year_built        1950.000000   1955.000000   1965.00000
dining_room_type            1.000000      1.000000      3.00000
fuel_type                   2.000000      2.000000      3.00000
kitchen_type                1.000000      2.000000      2.00000
maintenance_cost          628.117500    723.310000    880.18500
num_bedrooms                1.000000      2.000000      2.00000
num_floors_in_building      3.000000      6.000000      7.00000
num_full_bathrooms          1.000000      1.000000      1.00000
```

```
num_total_rooms                3.000000      4.000000      5.00000
parking_charges               72.342500     98.425000    125.90750
sale_price               179750.000000 275000.000000 435000.00000
sq_footage                   732.077500    850.000000    982.61000
walk_score                    76.000000     86.000000     94.00000
price_listings           416800.000000 445800.000000 493650.00000
avg_prices               316500.000000 412000.000000 490000.00000
cats_allowed                   0.000000      0.000000      1.00000
dogs_allowed                   0.000000      0.000000      1.00000
coop_condo                     0.000000      0.000000      0.00000
price_per_sqft               236.516313    302.037393    454.10766

                                  max
zipcode                   11435.00000
approx_year_built          2016.00000
dining_room_type              4.00000
fuel_type                     4.00000
kitchen_type                  4.00000
maintenance_cost           4659.00000
num_bedrooms                  3.00000
num_floors_in_building       33.00000
num_full_bathrooms            3.00000
num_total_rooms               8.00000
parking_charges             500.00000
sale_price               999999.00000
sq_footage                 6215.00000
walk_score                   99.00000
price_listings           759000.00000
avg_prices               804446.00000
cats_allowed                  1.00000
dogs_allowed                  1.00000
coop_condo                    1.00000
price_per_sqft             1163.56383
```

```
t_data
```

```
from sklearn.model_selection import train_test_split
X_data = 
t_data[['approx_year_built','dining_room_type','fuel_type','maintenanc
e_cost','num_bedrooms',

'num_full_bathrooms','num_total_rooms','sq_footage','walk_score','avg_
prices','coop_condo',
                 'price_per_sqft']]
y_data = t_data[['sale_price']]
X_train,X_test,y_train,y_test = 
train_test_split(X_data,y_data,test_size=0.2,random_state=10)
```

```
X_train.max()
```

```
approx_year_built       2016.00000
dining_room_type           4.00000
fuel_type                  4.00000
maintenance_cost        4659.00000
num_bedrooms               3.00000
num_full_bathrooms         3.00000
num_total_rooms            8.00000
sq_footage              6215.00000
walk_score                99.00000
avg_prices            804446.00000
coop_condo                 1.00000
price_per_sqft          1163.56383
dtype: float64
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg_model  = reg.fit(X_train,y_train)
```

```
from sklearn.metrics import mean_squared_error
y_pred_ISE = reg_model.predict(X_train)
MSE = mean_squared_error(y_train,y_pred_ISE)
RMSE = np.sqrt(MSE)
print("In Sample Errors for Linear Regression Model")
print("RMSE:",RMSE)
print("R-Squared:",reg_model.score(X_train,y_train)*100,"%")
print("Coefficients:",reg_model.coef_)
print("Intercept:", reg_model.intercept_)
```

```
In Sample Errors for Linear Regression Model
RMSE: 39989.84079389918
R-Squared: 94.9825538987418 %
Coefficients: [[-6.46172707e+02  4.18813169e+03  2.95274608e+03
9.67462217e+01
   3.19104981e+04  7.90113312e+04 -1.00228454e+03  8.45837749e+01
   9.21564278e+01 -2.08020323e-02  7.16620379e+03  8.05701116e+02]]
Intercept: [990089.82416215]
```

```
reg_y_pred_OOS = reg_model.predict(X_test)
MSE = mean_squared_error(y_test,reg_y_pred_OOS)
RMSE = np.sqrt(MSE)
print("Out of Sample Errors for Linear Regression Model")
print("RMSE:",RMSE)
print("R-Squared:",reg_model.score(X_test,y_test)*100,"%")
```

```
Out of Sample Errors for Linear Regression Model
RMSE: 39938.256687374706
R-Squared: 94.55147510771577 %
```

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor(random_state=100,max_depth=5)
tree_model = tree_reg.fit(X_train,y_train)
```

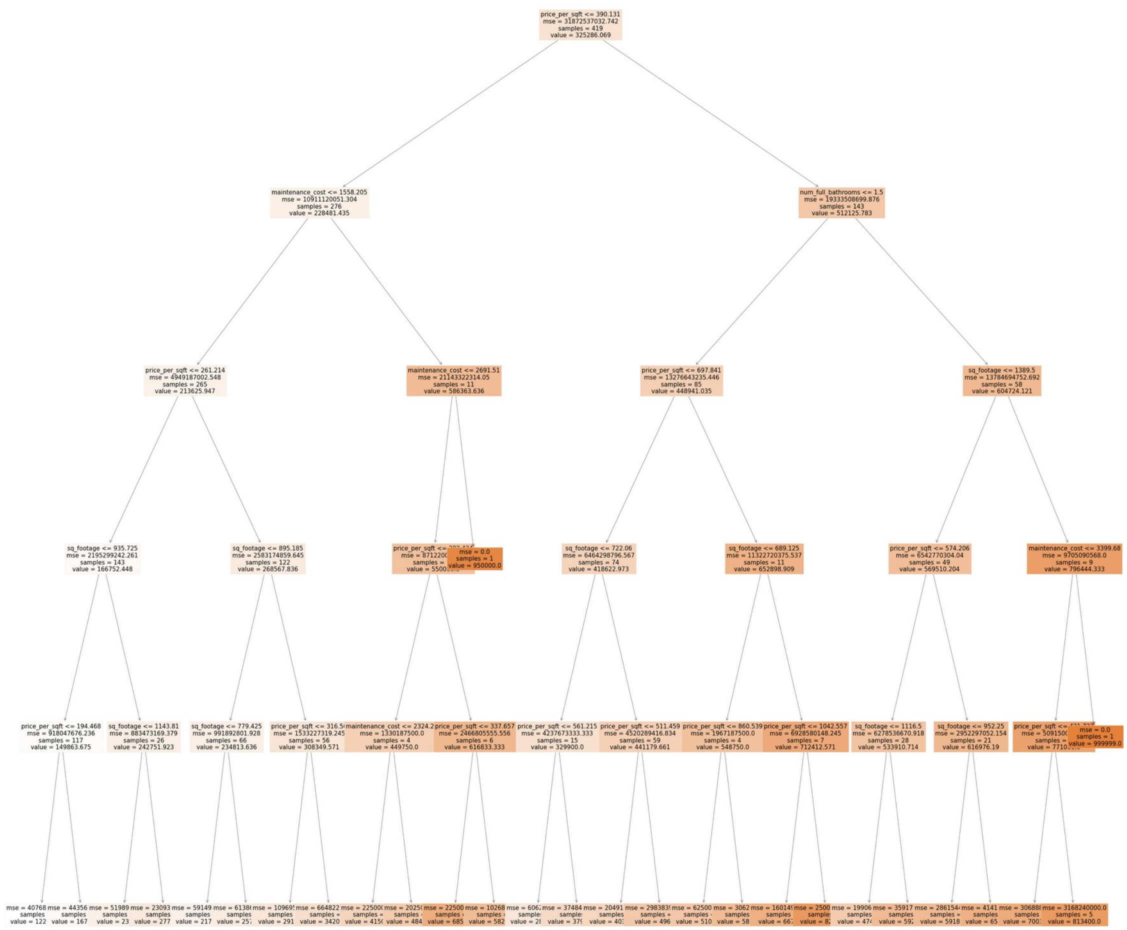
```
tree_y_pred_ISE = tree_reg.predict(X_train)
MSE = mean_squared_error(y_train,tree_y_pred_ISE)
RMSE = np.sqrt(MSE)
print("In Sample Errors for Tree Regression Model")
print("RMSE:",RMSE)
print("R-Squared:",tree_reg.score(X_train,y_train)*100,"%")
```

```
In Sample Errors for Tree Regression Model
RMSE: 33871.807257475506
R-Squared: 96.40035141943999 %
```

```
tree_y_pred_OOS = tree_reg.predict(X_test)
MSE = mean_squared_error(y_test,tree_y_pred_OOS)
RMSE = np.sqrt(MSE)
print("Out of Sample Errors for Tree Regression Model")
print("OOS RMSE:",RMSE)
print("OOS R-Squared:",tree_model.score(X_test,y_test)*100,"%")
```

```
Out of Sample Errors for Tree Regression Model
OOS RMSE: 48504.96391683467
OOS R-Squared: 91.96338611743867 %
```

```
from sklearn import tree
plt.figure(figsize=(50,50))
features = X_data.columns
tree.plot_tree(tree_reg,feature_names=features,filled=True,fontsize=15
)
plt.show()
```



```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(max_depth=5)
rf_model = rf.fit(X_train,y_train)
```

```
/var/folders/y1/cj1dr0kd7lng8yfkrtd5_r_00000gn/T/
ipykernel_24584/1441387015.py:3: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  rf_model = rf.fit(X_train,y_train)
```

```
rf_y_pred_ISE = rf_model.predict(X_train)
MSE = mean_squared_error(y_train,rf_y_pred_ISE)
RMSE = np.sqrt(MSE)
print("In Sample Errors for Random Forest Regression Model")
print("RMSE:",RMSE)
print("R-Squared:",rf_model.score(X_train,y_train)*100,"%")
```

```
In Sample Errors for Random Forest Regression Model
RMSE: 25735.93058832218
R-Squared: 97.92191590344213 %
```

```
rf_y_pred_OOS = rf_model.predict(X_test)
MSE = mean_squared_error(y_test,rf_y_pred_OOS)
RMSE = np.sqrt(MSE)
print("Out of Sample Errors for Random Forest Regression Model")
print("RMSE:",RMSE)
print("R-Squared:",rf_model.score(X_test,y_test)*100,"%")
```

```
Out of Sample Errors for Random Forest Regression Model
RMSE: 33528.202805024004
R-Squared: 96.1600879164084 %
```