BAN612 Project

House Price Prediction

Arushi Aesha, Sukirti Bhat, Vishwajit S Bhosale, Kumar Abhinav, Apoorva Tyagi

Agenda

01
Data Scraping

02
Data Wrangling

03
Univariate Analysis

04
Bivariate Analysis

05
Price Prediction

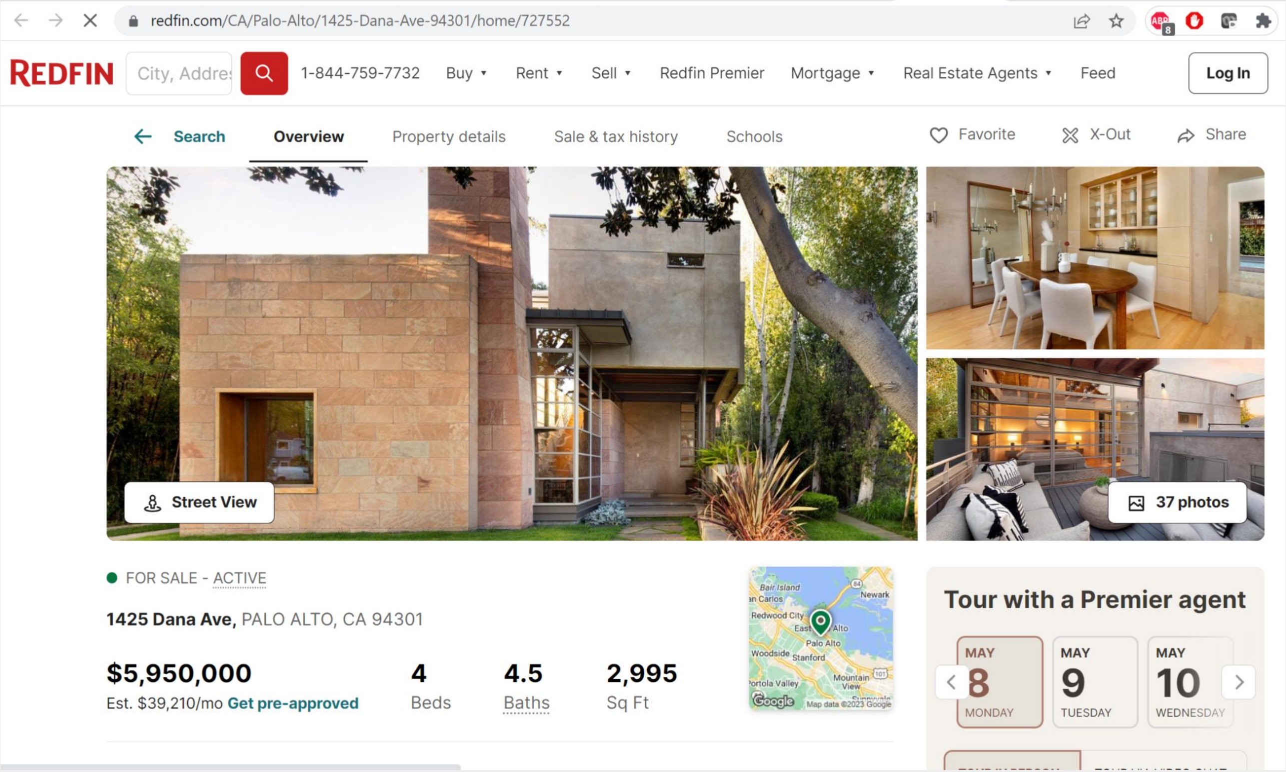
06
Conclusion

Data scraping was done on redfin.com

Redfin is very popular real estate service platform which offers multiple services to the users and has variety of data.

- Redfin is an extensive listing database which provides users real estate listing in their desired neighbourhood. It has all the details on the listing such as previous price, photos, taxes, location etc.
- Estimate Price- Redfin also provides current price estimate of the property. Which we will be predicting in our project as well.
- In our project we have used selenium to scrape data for Bay Area and nearby areas.
- All the houses are listed on an individual web page and we have used that path to as input to get the detail information of the house listed.
- After pulling the data we have stored the data as pandas dataframe and then dropped the irrelevant columns before starting the data wrangling step.

Redfin has page for individual listing; we are scraping data from these pages



Using selenium to scrape the data

```
import os
import pandas as pd

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

driver = webdriver.Chrome()
driver.get('https://www.redfin.com/')

driver.maximize_window()
elem = driver.find_element("xpath","//button[@class='onetrust-close-btn-handler onetrust-close-btn-ui banner-close-button ot-
elem.click()
elem = driver.find_element("xpath","//input[@title='City, Address, School, Agent, ZIP']")
elem.click()
elem.clear()
elem.send_keys("fremont")
elem.send_keys(Keys.RETURN)
time.sleep(5)
elem = driver.find_element("xpath","//a[@id='download-and-save']")
driver.execute_script("arguments[0].scrollIntoView({behavior: 'auto', block: 'center', inline: 'center'});", elem)
elem.click()
time.sleep(5)

Homepagelink= driver.find_element("xpath","//a[@data-rf-test-id='breadcrumbTitle'][1]")
Homepagelink.click()
time.sleep(5)
elem = driver.find_element("xpath","//input[@title='City, Address, School, Agent, ZIP']")
elem.click()
elem.clear()
elem.send_keys("san francisco")
elem.send_keys(Keys.RETURN)
time.sleep(10)
```

Final Dataset after dropping irrelevant columns and ready for the data cleaning

Out[189]:

	SALE TYPE	PROPERTY TYPE	CITY	STATE OR PROVINCE	PRICE	BEDS	BATHS	LOCATION	SQUARE FEET	YEAR BUILT	DAYS ON MARKET	$/SQUARE FEET	STATUS	SOURCE	MLS#	L
0	Redfin Data	Single Family Residential	Fremont	CA	1300000	3.0	2.0	Fremont	1067.0	1961.0	7.0	1218.0	Pre On-Market	Coming Soon	92185	[illegible]
1	Redfin Data	Single Family Residential	Fremont	CA	1380000	4.0	2.5	Fremont	1866.0	1998.0	21.0	740.0	Pre On-Market	Coming Soon	91889	[illegible]
2	MLS Listing	Single Family Residential	Fremont	CA	900000	3.0	2.0	Niles Area	1120.0	1955.0	1.0	804.0	Active	bridgeMLS, Bay East AOR, or Contra Costa AOR	41025752	[illegible]
3	MLS Listing	Townhouse	Fremont	CA	990000	3.0	1.5	Valle De La Paz	1242.0	1971.0	1.0	797.0	Active	bridgeMLS, Bay East AOR, or Contra Costa AOR	41025224	[illegible]
4	MLS Listing	Townhouse	Fremont	CA	748888	3.0	1.5	Cherry Lane	1180.0	1971.0	2.0	635.0	Active	bridgeMLS, Bay East AOR, or Contra Costa AOR	41025667	[illegible]

[190]:

```
union_df.columns
```

Out[190]:

```
Index(['SALE TYPE', 'PROPERTY TYPE', 'CITY', 'STATE OR PROVINCE', 'PRICE',
       'BEDS', 'BATHS', 'LOCATION', 'SQUARE FEET', 'YEAR BUILT',
       'DAYS ON MARKET', '$/SQUARE FEET', 'STATUS', 'SOURCE', 'MLS#',
       'LATITUDE', 'LONGITUDE'],
      dtype='object')
```

Dropping the rows with null values

Doing the sum of Null values in the data set ¶

In [192]:
```
#checking if any null values
num_missing = union_df.isna().sum()
num_missing
```

Out[192]:
```
SALE TYPE                0
PROPERTY TYPE            0
CITY                     0
STATE OR PROVINCE        0
PRICE                    0
BEDS                    41
BATHS                   54
LOCATION                 0
SQUARE FEET             78
YEAR BUILT             126
DAYS ON MARKET          62
$/SQUARE FEET           78
STATUS                   0
SOURCE                   0
MLS#                     0
LATITUDE                 0
LONGITUDE                0
dtype: int64
```

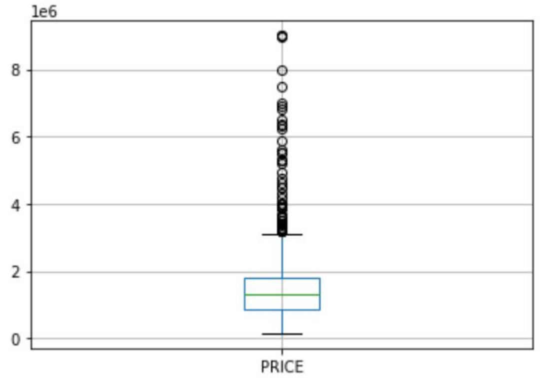
In [193]:
```
#droping the null values
union_df=union_df.dropna()
```

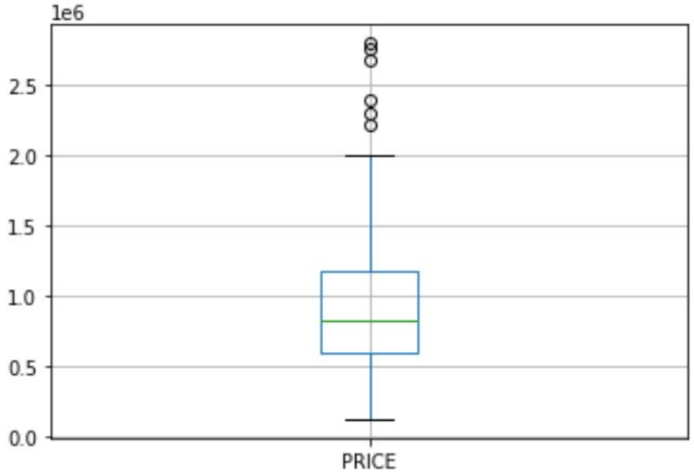
Finding the outliers and dropping the rows with outside IQR and exporting final data

Drawing a Box Plot to identify the outliers.

```python
union_df.boxplot(column=['PRICE'])
```

```
95]: <AxesSubplot:>
```






```python
# write the resulting DataFrame to a CSV file
output_path = "output.csv"
union_df.to_csv(output_path, index=False)

print(f"Successfully wrote DataFrame to {output_path}")
```

```
Successfully wrote DataFrame to output.csv
```

Statistical Summary of all numerical features

```
#summay of dataset
df.describe()
```

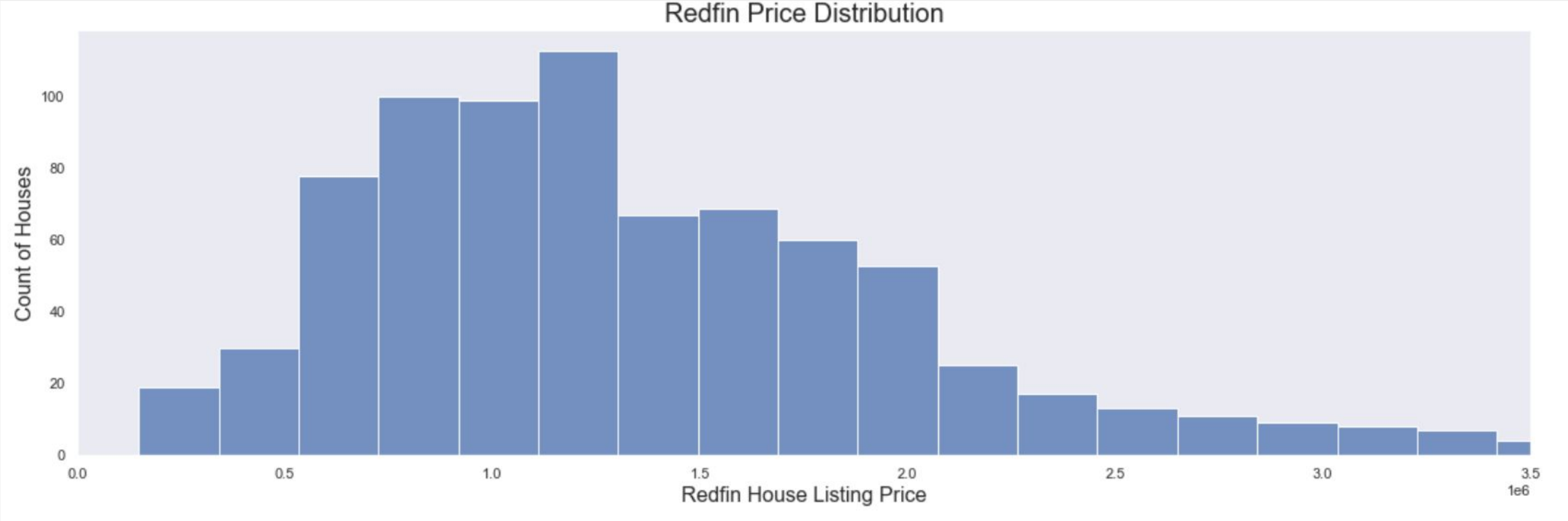
	PRICE	BEDS	BATHS	SQUARE FEET	YEAR BUILT	DAYS ON MARKET	$/SQUARE FEET	LATITUDE	LONGITUDE
count	8.190000e+02	819.000000	819.000000	819.000000	819.000000	819.000000	819.000000	819.000000	819.000000
mean	1.528086e+06	3.195360	2.407814	1844.884005	1968.473748	15.957265	845.849817	37.535712	-122.096611
std	1.117984e+06	1.707435	1.333594	1164.475359	36.464844	28.174567	355.438337	0.209899	0.252694
min	1.480000e+05	0.000000	1.000000	480.000000	1885.000000	1.000000	126.000000	37.175114	-122.508079
25%	8.770000e+05	2.000000	2.000000	1180.000000	1947.000000	4.000000	633.500000	37.313972	-122.412104
50%	1.295000e+06	3.000000	2.000000	1542.000000	1971.000000	10.000000	811.000000	37.560114	-121.970475
75%	1.795000e+06	4.000000	3.000000	2180.000000	2000.000000	16.000000	999.500000	37.745982	-121.886056
max	8.999000e+06	24.000000	19.000000	14390.000000	2023.000000	342.000000	6153.000000	37.805312	-121.738352

```
#summary of column price only
pd.set_option('display.float_format', lambda x: '%.2f' % x)
df["PRICE"].describe()
```

```
count       819.00
mean    1528085.96
std     1117983.59
min      148000.00
25%      877000.00
50%     1295000.00
75%     1795000.00
max     8999000.00
Name: PRICE, dtype: float64
```

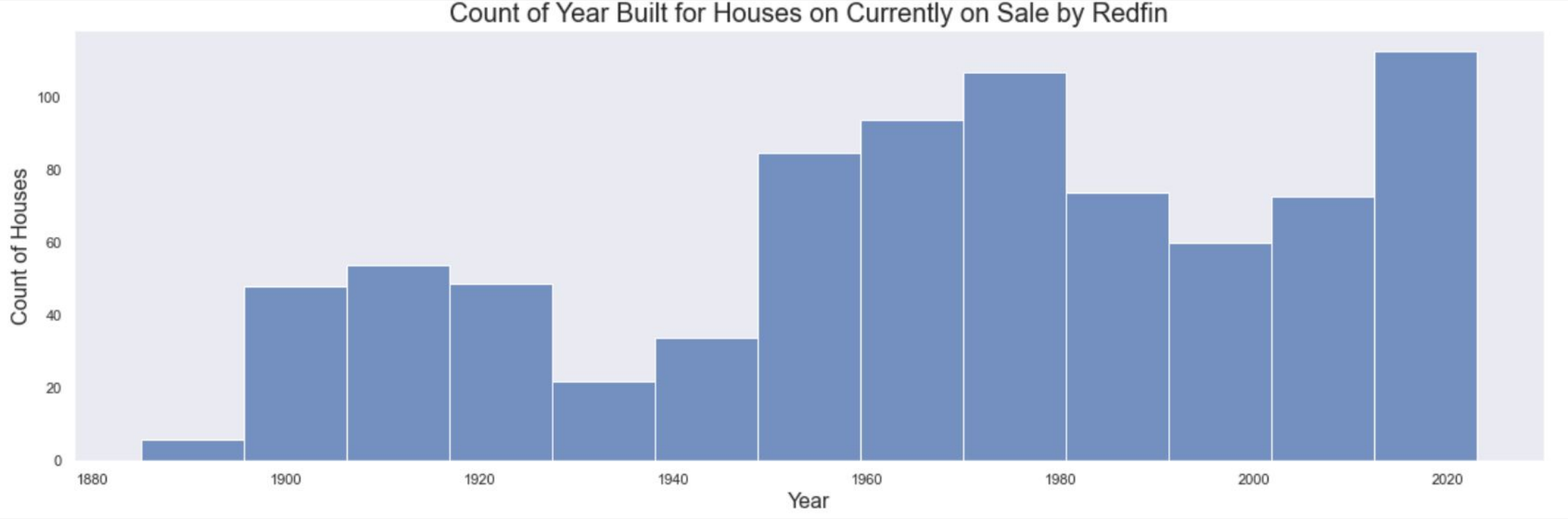
House Price follows the normal distribution

Price for houses listed in Redfin is right skewed. This means that the distribution of house prices is not symmetrical, and the tail of the distribution is stretched out towards the right-hand side. In other words, there are a few houses that are priced much higher than the majority of houses, causing the distribution to be skewed to the right. For example, it might indicate that there is a high demand for luxury homes in the area or that certain neighborhoods are becoming more exclusive and expensive.

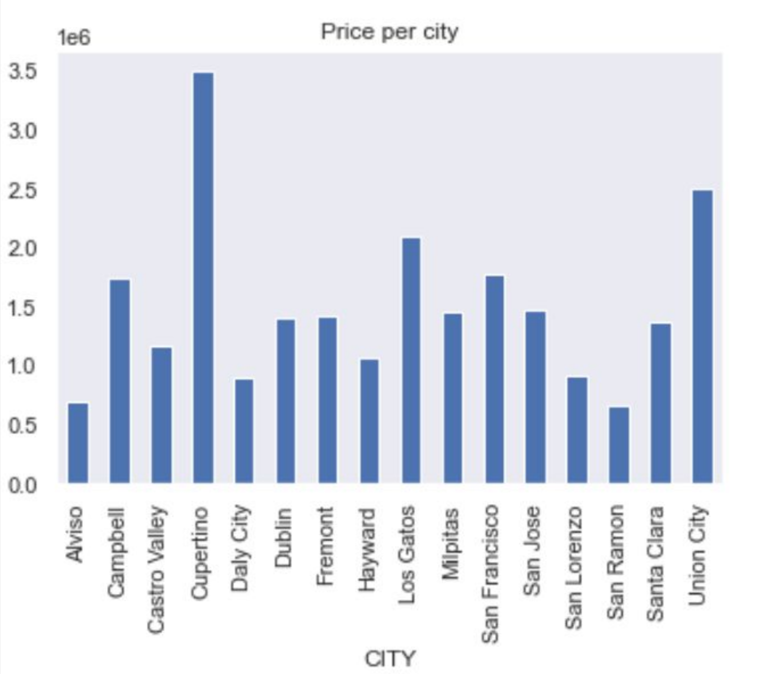


Finding the distribution of Houses built year

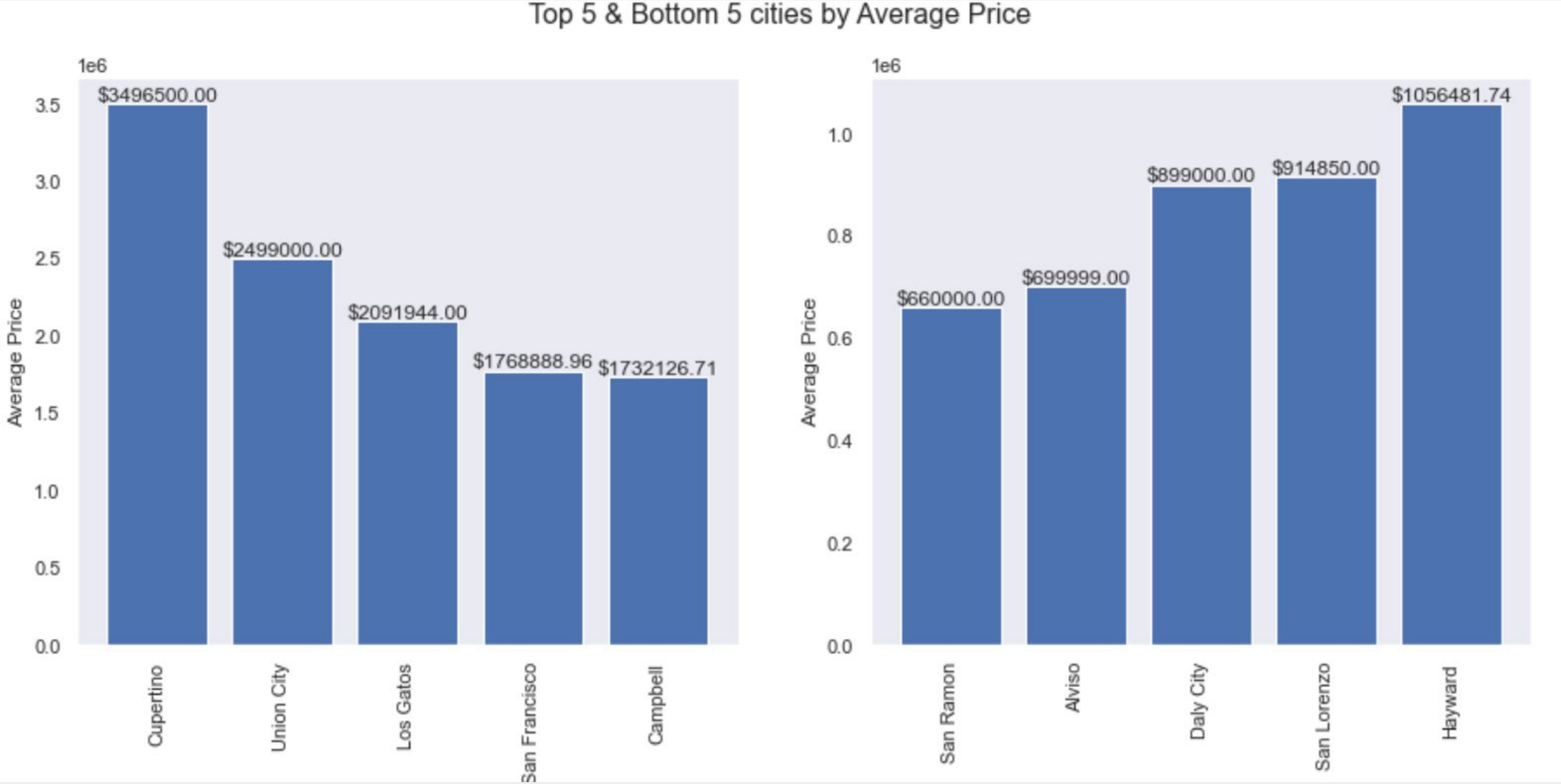
After the year 1940 the distribution is uniform and but it looks like houses built in the year 1980-2010 are relatively less available in the market for sale.



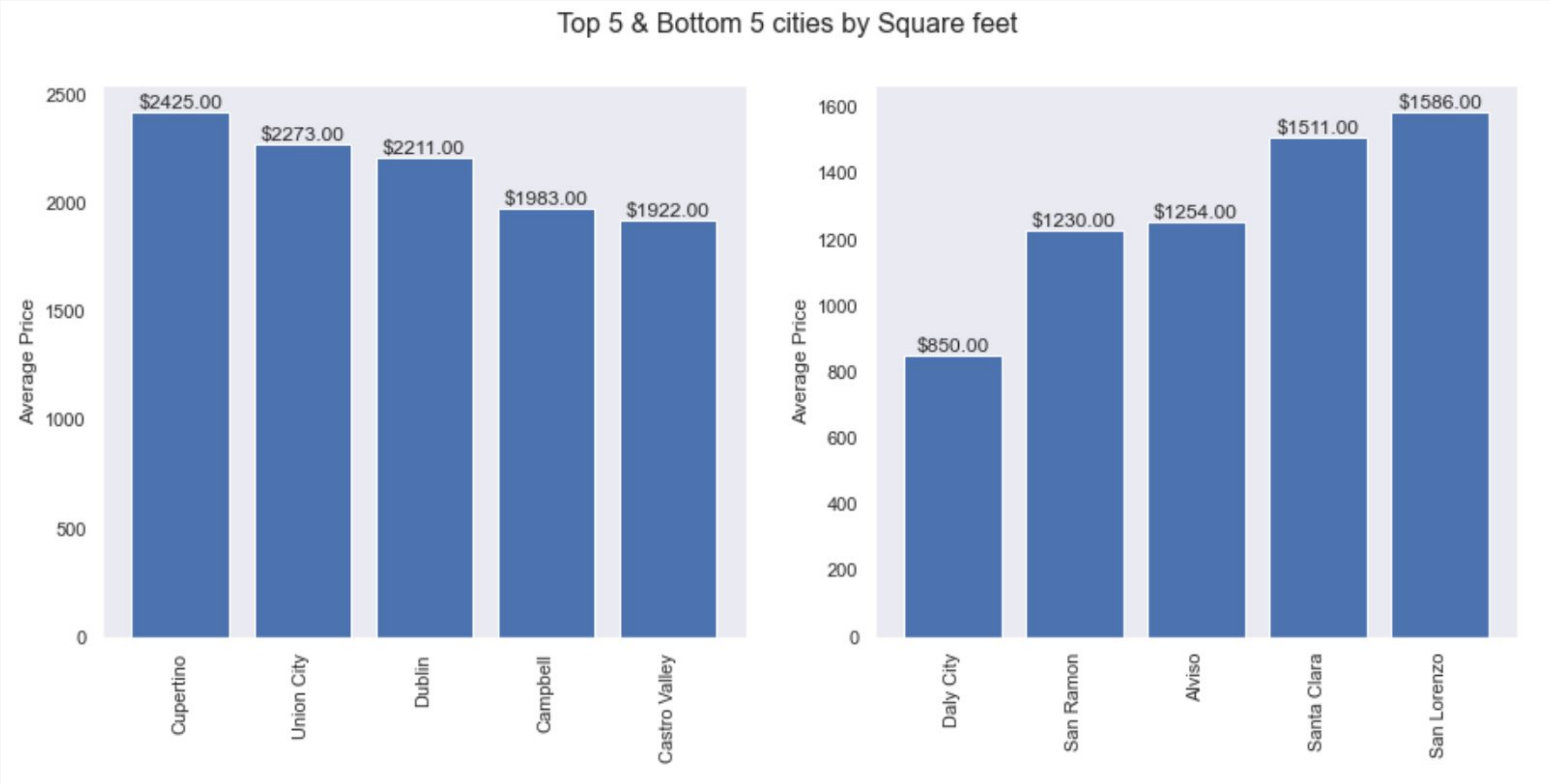
San Jose and SF have the highest houses; Cupertino houses are the costliest in the area



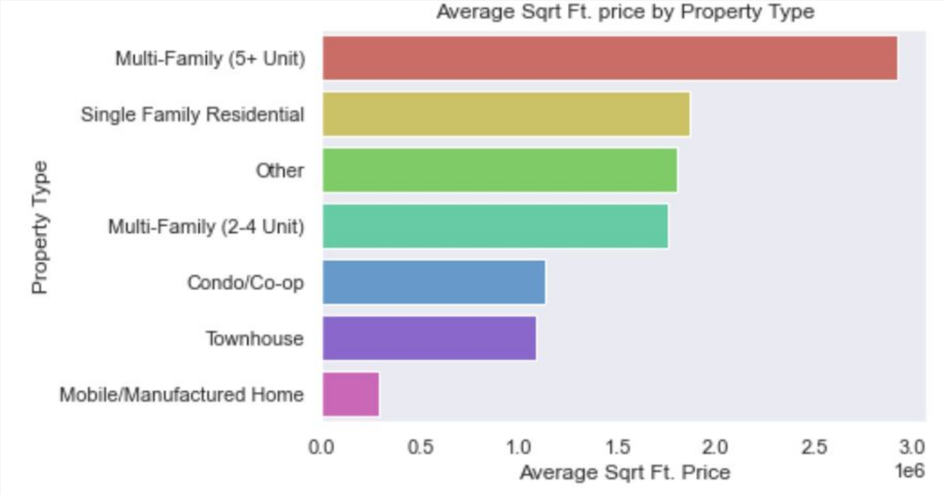
Average price distribution for top 5 and bottom 5 areas



Average $ square feet distribution of top 5 and bottom 5 areas



Multi-Family houses have higher average price



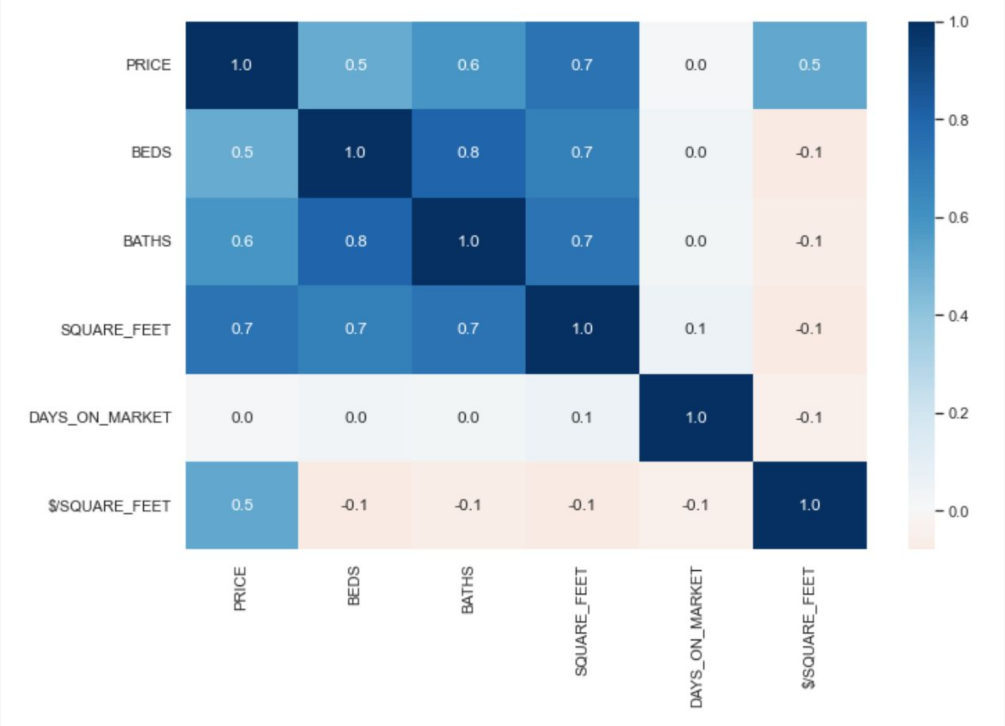
2,3,4 Beds houses are most popular; SF and San Jose have the highest active houses

BEDS CITY	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	21.0	24.0	Total
Alviso	0	0	0	1	0	0	0	0	0	0	0	0	0	1
Campbell	0	0	0	3	3	0	1	0	0	0	0	0	0	7
Castro Valley	0	0	0	1	1	0	0	0	0	0	0	0	0	2
Cupertino	0	0	0	1	0	1	0	0	0	0	0	0	0	2
Daly City	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Dublin	0	1	3	5	15	2	0	0	0	0	0	0	0	26
Fremont	0	8	26	26	18	4	1	2	1	0	0	0	0	86
Hayward	0	0	24	24	21	7	1	0	1	1	1	0	0	80
Los Gatos	0	0	1	1	2	0	0	0	0	0	0	0	0	4
Milpitas	0	0	0	1	2	0	0	0	0	0	0	0	0	3
San Francisco	8	44	84	62	50	15	6	8	1	0	1	1	0	280
San Jose	0	12	63	109	91	31	9	1	2	2	0	0	1	321
San Lorenzo	0	0	0	1	1	0	0	0	0	0	0	0	0	2
San Ramon	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Santa Clara	0	0	2	0	0	0	0	0	0	0	0	0	0	2
Union City	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Total	8	65	205	235	205	60	18	11	5	3	2	1	1	819

STATUS CITY	Active	Pre On-Market	Total
Alviso	1	0	1
Campbell	7	0	7
Castro Valley	2	0	2
Cupertino	2	0	2
Daly City	1	0	1
Dublin	26	0	26
Fremont	84	2	86
Hayward	80	0	80
Los Gatos	4	0	4
Milpitas	3	0	3
San Francisco	280	0	280
San Jose	319	2	321
San Lorenzo	2	0	2
San Ramon	1	0	1
Santa Clara	2	0	2
Union City	1	0	1
Total	815	4	819

Price seems to correlated with all the features

$/square_feet seems to have no correlation or small negative correlation which indicates that $/square_feet is independent of other features. Price is independent of days on market.



Predicting the price of House- with all variables

This is the first model which is used to predict the price. We are using all the variables to predict the price of the house.

```
predictors = ['BEDS', 'BATHS', 'SQUARE_FEET','DAYS_ON_MARKET', '$/SQUARE_FEET','CITY','STATUS']
outcome = 'PRICE'
```

Training Results

```
# print performance measures
regressionSummary(train_y, redfin_lm.predict(train_X))
```

```
intercept  -1204516.8189803772
                Predictor  coefficient
0                    BEDS   -126159.66
1                   BATHS    66845.38
2             SQUARE_FEET      942.92
3          DAYS_ON_MARKET     -777.71
4           $/SQUARE_FEET     1753.66
5           CITY_Campbell  -175269.09
6      CITY_Castro Valley  -235388.15
7          CITY_Cupertino   155609.54
8          CITY_Daly City        0.00
9             CITY_Dublin  -331739.23
10           CITY_Fremont  -151373.85
11           CITY_Hayward  -235557.58
12         CITY_Los Gatos  -198813.52
13          CITY_Milpitas  -111825.96
14     CITY_San Francisco  -202646.21
15          CITY_San Jose  -205473.58
16       CITY_San Lorenzo  -171713.82
17         CITY_San Ramon        0.00
18       CITY_Santa Clara        0.00
19        CITY_Union City        0.00
20  STATUS_Pre On-Market     90121.01

Regression statistics

                      Mean Error (ME) : -0.0000
       Root Mean Squared Error (RMSE) : 391231.4748
            Mean Absolute Error (MAE) : 202768.8649
          Mean Percentage Error (MPE) : 1.3799
Mean Absolute Percentage Error (MAPE) : 16.2125
```

In our first predictive model the results of training and validation sets are comparable so this is not an overfitting model but we will still want to try other optimizations to improve the performance of the model. The R^2 for the training set on this model is ~89%.

Validation Results

```
# Compute common accuracy measures
regressionSummary(valid_y, redfin_lm_pred)
```

```
     Predicted   Actual    Residual
81   809473.14  1000000   190526.86
684 2878737.12  2985000   106262.88
8    592735.67   669000    76264.33
241 1576446.75   995000  -581446.75
265 1660298.99   890000  -770298.99
154 2634215.60  2695000    60784.40
101 1242502.49   698000  -544502.49
17  1878733.41  1799000   -79733.41
233 1301743.40  1200000  -101743.40
797  616566.75   699000    82433.25
488  -13562.47   289000   302562.47
540 1387873.76  1398000    10126.24
774  466956.03   750000   283043.97
806 1118165.64  1175000    56834.36
518 1154920.23  1188000    33079.77
60   204933.53   480000   275066.47
577 1425360.24  1299999  -125361.24
634 1958149.65  1950000    -8149.65
107 4947938.89   825000 -4122938.89
744 2468971.52  1800000  -668971.52

Regression statistics

                      Mean Error (ME) : -83471.6672
       Root Mean Squared Error (RMSE) : 501863.2572
            Mean Absolute Error (MAE) : 227137.0902
          Mean Percentage Error (MPE) : -0.6973
Mean Absolute Percentage Error (MAPE) : 21.8421
```

Using stepwise model to predict the price

We have used stepwise model to predict the price of the house.

```python
#Using stepwise
bestSW_model, best_variables = stepwise_selection(train_X.columns, train_model, score_model, verbose=True)

# Fit the linear regression model using the selected variables
bestSW_model.fit(train_X[best_variables], train_y)

# Calculate adjusted R-squared for the validation set
n = len(valid_y)
p = len(best_variables)
adj_r2 = 1 - (1 - bestSW_model.score(valid_X[best_variables], valid_y)) * (n - 1) / (n - p - 1)

# Print adjusted R-squared and other performance measures
print('Adjusted R-squared:', adj_r2)
regressionSummary(valid_y, bestSW_model.predict(valid_X[best_variables]))
```

```
Variables: BEDS, BATHS, SQUARE_FEET, DAYS_ON_MARKET, $/SQUARE_FEET
Start: score=15160.99, constant
Step: score=14772.01, add SQUARE_FEET
Step: score=14099.60, add $/SQUARE_FEET
Step: score=14062.53, add BEDS
Step: score=14057.47, add BATHS
Step: score=14057.47, unchanged None
Adjusted R-squared: 0.7139033818009509

Regression statistics

                      Mean Error (ME) : -79561.9211
       Root Mean Squared Error (RMSE) : 494451.5331
            Mean Absolute Error (MAE) : 221907.7031
          Mean Percentage Error (MPE) : -0.3213
Mean Absolute Percentage Error (MAPE) : 21.3958
```

This model also has error metrics values within the range of our initial model in the last step so we will still look for better models in the next steps. The R^2 in this model however is lesser than the initially expected model

Using only numeric variables to predict

We have used only numeric variable to predict in this iteration. The left side is the result of the training set and right side below is validation set. The R^2 in this is higher as compared to the previous iterations. Also, predicting with less number of variables is better as it is less complex and more interpretable model in terms on understanding what the features means. The AIC and BIC scores are also less as compared to the initial model.

```
intercept  -1419296.0012642643
        Predictor  coefficient
0            BEDS   -124356.36
1           BATHS     64214.57
2     SQUARE_FEET       942.68
3  DAYS_ON_MARKET      -853.33
4   $/SQUARE_FEET      1768.56

Regression statistics

                      Mean Error (ME) : 0.0000
       Root Mean Squared Error (RMSE) : 393327.7967
            Mean Absolute Error (MAE) : 203979.0923
          Mean Percentage Error (MPE) : 1.4373
Mean Absolute Percentage Error (MAPE) : 16.3226
```

```python
pred_y = redfin_lm.predict(train_X)

print('adjusted r2 : ', adjusted_r2_score(train_y, pred_y, redfin_lm))
print('AIC : ', AIC_score(train_y, pred_y, redfin_lm))
print('BIC : ', BIC_score(train_y, pred_y, redfin_lm))
```

```
adjusted r2 :  0.8953052974332667
AIC :  14057.913095840664
BIC :  14087.288204735225
```

```python
# Compute common accuracy measures
regressionSummary(valid_y, redfin_lm_pred)
```

```
     Predicted   Actual    Residual
81   734061.06  1000000   265938.94
684 2878057.81  2985000   106942.19
8    537386.39   669000   131613.61
241 1565054.22   995000  -570054.22
265 1668241.26   890000  -778241.26
154 2636660.33  2695000    58339.67
101 1247783.73   698000  -549783.73
17  1827875.42  1799000   -28875.42
233 1300328.48  1200000  -100328.48
797  647697.75   699000    51302.25
488  -20142.91   289000   309142.91
540 1396546.93  1398000     1453.07
774  498902.42   750000   251097.58
806 1140322.68  1175000    34677.32
518 1158029.60  1188000    29970.40
60   146456.55   480000   333543.45
577 1421561.51  1299999  -121562.51
634 1960831.11  1950000   -10831.11
107 4935125.99   825000 -4110125.99
744 2494522.98  1800000  -694522.98

Regression statistics

                      Mean Error (ME) : -78226.6440
       Root Mean Squared Error (RMSE) : 498358.1600
            Mean Absolute Error (MAE) : 224667.1025
          Mean Percentage Error (MPE) : -0.0530
Mean Absolute Percentage Error (MAPE) : 21.7798
```

Finding the best fit model

Using the exhaustive search function and considering model evaluation metrics R^2 adj, AIC and MAE we will finalize the model in which we are only using the numerical variables to predict the price.

```python
# Display the Exhaustive Search results.
print(pd.DataFrame(data, columns=('n', 'r2adj', 'AIC','MAE' ) + tuple(sorted(allVariables))))
# Define the width of output presentation to be wider to display results in two rows (instead of more rows otherwise).
pd.set_option('display.width', 100)


# Reset the output width to the default.
pd.reset_option('display.width')
```

```
   n  r2adj       AIC        MAE  $/SQUARE_FEET  BATHS   BEDS  DAYS_ON_MARKET  \
0  1   0.55  14772.01  431004.24          False  False  False           False
1  2   0.89  14099.60  188240.54           True  False  False           False
2  3   0.89  14062.53  221835.04           True  False   True           False
3  4   0.90  14057.47  221907.70           True   True   True           False
4  5   0.90  14057.91  224667.10           True   True   True            True

   SQUARE_FEET
0         True
1         True
2         True
3         True
4         True
```

The best fit model

Below is our best fit model which can be used to predict the price of the houses listed on the redfin website. We will be using only 3 numerical features to predict the independent variable. In our initial analysis also we saw that Bath is correlated with Beds and day in market had little to no correlation to the price.

```
#Final Model based on Exhaustive search ( We looked into training adjusted R square and Validation MAE)
# final model we choose
model_idx = 2
final_predictors = results[model_idx]['variables']
final_model = results[model_idx]['model']
val_mae = results[model_idx]['val_MAE']
val_r2adj = results[model_idx]['val_MAE']
```

```
print("Final model information:\n")
print("Predictors: \n{} ".format(final_predictors))
print("\nValidation MAE: \n{} ".format(val_mae))
```

```
Final model information:

Predictors:
['BEDS', 'SQUARE_FEET', '$/SQUARE_FEET']

Validation MAE:
221835.0371141757
```

```
# print coefficients
print('intercept ', final_model.intercept_)
print(pd.DataFrame({'Predictor': final_predictors, 'coefficient': final_model.coef_}))
```

```
intercept  -1424041.1362984339
       Predictor  coefficient
0           BEDS    -99474.45
1    SQUARE_FEET       976.62
2  $/SQUARE_FEET      1772.64
```

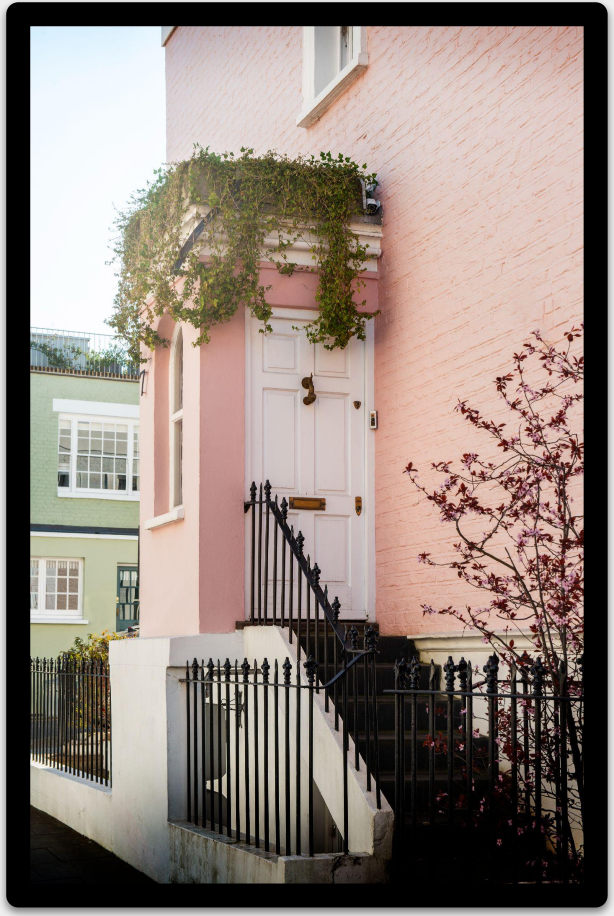
Conclusion

- To predict the price which is a continuous numerical variable we needed a regression model and we used various forms of multiple linear regression model to predict it.

- In the initial steps we did the univariate and bivariate analysis to lay the land for us to understand how each variable is correlated and what is the distribution of those variables.

- Using our understanding of the housing market the results of the model and the analysis are in-line and yields that no. of beds, area are probably the best predictors of a house for sale.

- If there are any follow up analysis happens then it would be very interesting to see how the latitude and longitude(basically location) plays a role in the price of house. However, these variables are complicated to interpret and would need some third party data to estimate the location closest to the city center or any major business or technology hub.



Thanks!