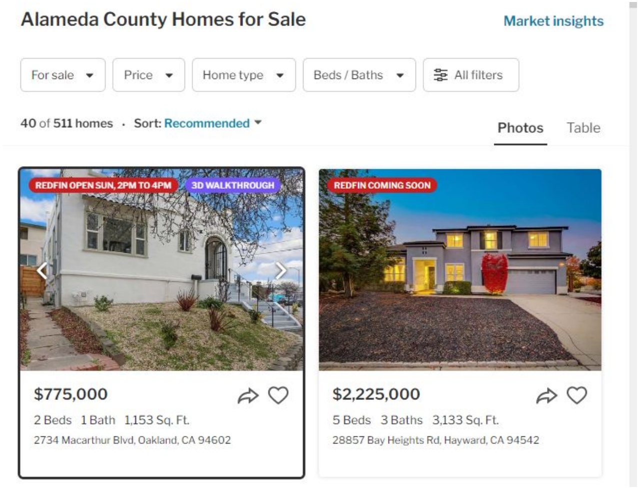
BAN 612
Housing Project
Group #1

Siraj Abbasi
Anton Zyarko
Vedantini Bogawat
Akshat Verma
Mihir Jain
Jasmeen Kaur

Housing Data Source - Redfin

- Redfin is a real-estate brokerage based out of Seattle
- House Listings on Redfin are not limited to Redfin Agents - collection of a wide range of listings
- **Goal:** Get overview of housing market in bay area and use different ML models to predict a listing price



Data Scraping

Urllib & Beautiful Soup Libraries

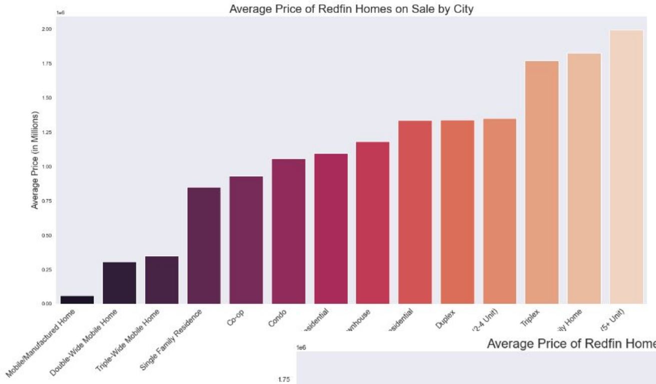
1. Obtained a series of page links for a few Bay Area Counties
2. Used BeautifulSoup to extract all the house links from each Redfin page
3. Used BeautifulSoup to locate the right tags & add specific key values to a Python Dictionary
4. Created a DataFrame using that Dictionary

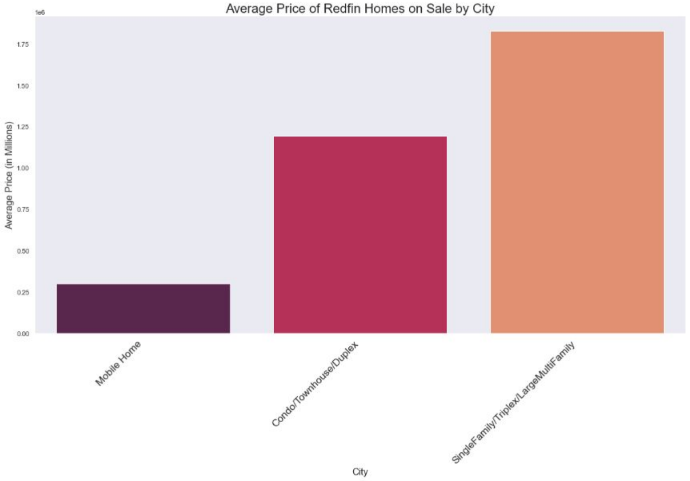
```python
i=1
housing_dictionary1={'street':[],'county':[],'state':[],'zipcode':[],'price':[],'bed':[],'bath':[],'sqft':[],'walkscore':[],
                     'transitscore':[],'bikescore':[],'competitivescore':[],'1st School Rating':[],'status':[],'house_type':
                     'year_built':[],'Lotsize':[],'perSq_Ft':[],'url':[] }

for x in url:
    housing = Request(x, headers={'User-Agent': 'Mozilla/5.0'})
    website_addr = urlopen(housing).read()
    attri_soup = BS( website_addr,'html.parser')
    parent   = attri_soup.findChildren('span',{'class':'header font-color-gray-light inline-block'})
    children = attri_soup.findChildren('span',{'class':'content text-right'})

    street = attri_soup.find("div",{"class":"street-address"}).text.replace(',','')

    city_state=attri_soup.find('div',{'data-rf-test-id':'abp-cityStateZip'}).text
    city_state = city_state.split(",")
    city_state[1] = city_state[1].split()


    price=attri_soup.find('div',{'class':'statsValue'}).text.replace('$','').replace(',','').replace('+','')

    try:
        bed=attri_soup.find_all('div',{'class':'stat-block beds-section'})[1].text
        bed  = bed.replace("Beds", " ").replace("Bed", " ")
    except:
        bed = 0
    try:
        bath=attri_soup.find('div',{'class':'stat-block baths-section'}).text
        bath = bath.replace("Baths", " ").replace("Bath", " ")
    except:
        bath = 0
    try:
        sqft=attri_soup.find('div',{'class':'stat-block sqft-section'}).text
        sqft  = sqft.replace("Sq Ft", " ")
    except:
        sqft = 0
```

Data Cleaning

Data Cleaning Highlights

- Dropped duplicates
- Removed Outliers (price, sqft..)
- Ensured all variables are of the right data type
- Filled missing numerical data w/ city median
- Reclassified certain Categorical variables to reduce number of dummies





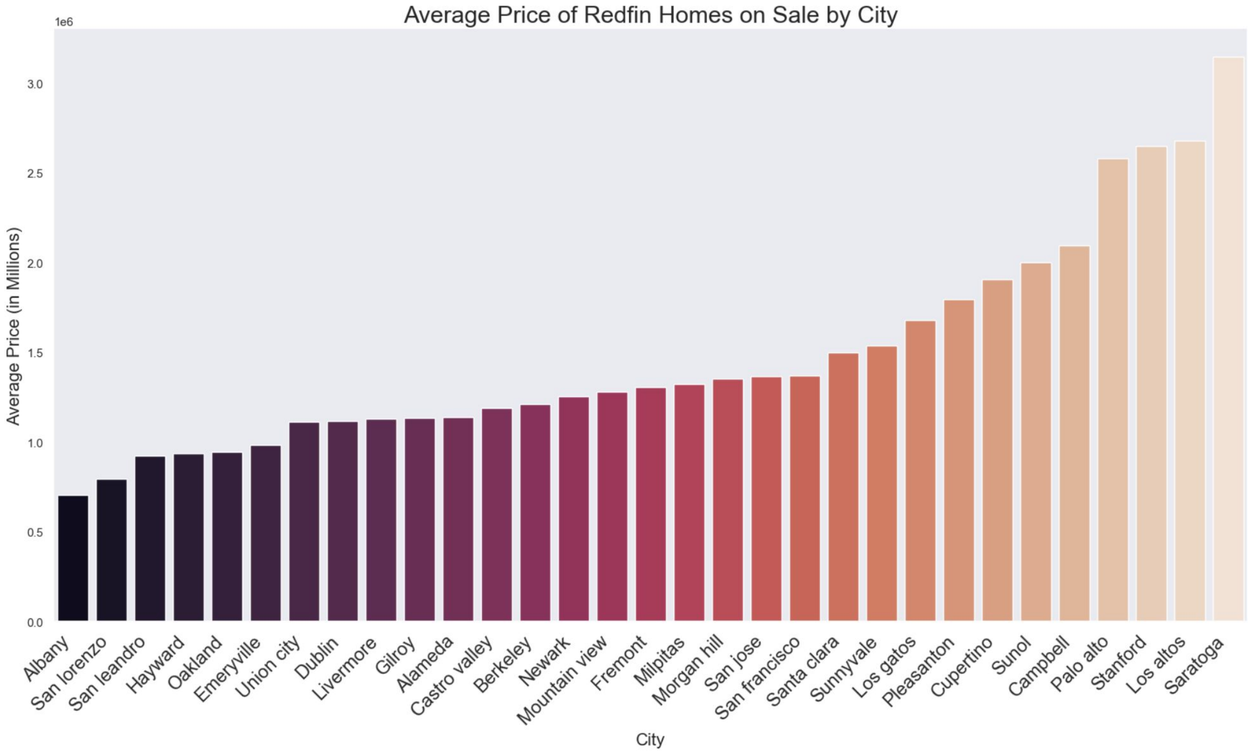
Tail Snippet of Clean Data

	street	city	state	county	zipcode	price	bed	bath	sqft	House_Type	walkscore	transitscore	bikescore
1040	382 Fontanelle Dr	San jose	Ca	Santa Clara County	95111	1049000	4	2	1542	SingleFamily/Triplex/LargeMultiFamily	9	27	43
1041	1882 Johnston Ave	San jose	Ca	Santa Clara County	95125	3395000	4	4	2931	SingleFamily/Triplex/LargeMultiFamily	53	38	58
1042	809 Auzerais Ave #401	San jose	Ca	Santa Clara County	95126	900000	2	2	1274	Condo/Townhouse/Duplex	83	60	88
1043	47 N Claremont Ave	San jose	Ca	Santa Clara County	95127	948000	3	2	1320	SingleFamily/Triplex/LargeMultiFamily	42	0	44
1044	3300 NARVAEZ Ave #45	San jose	Ca	Santa Clara County	95136	349000	3	2	1584	Mobile Home	50	46	56

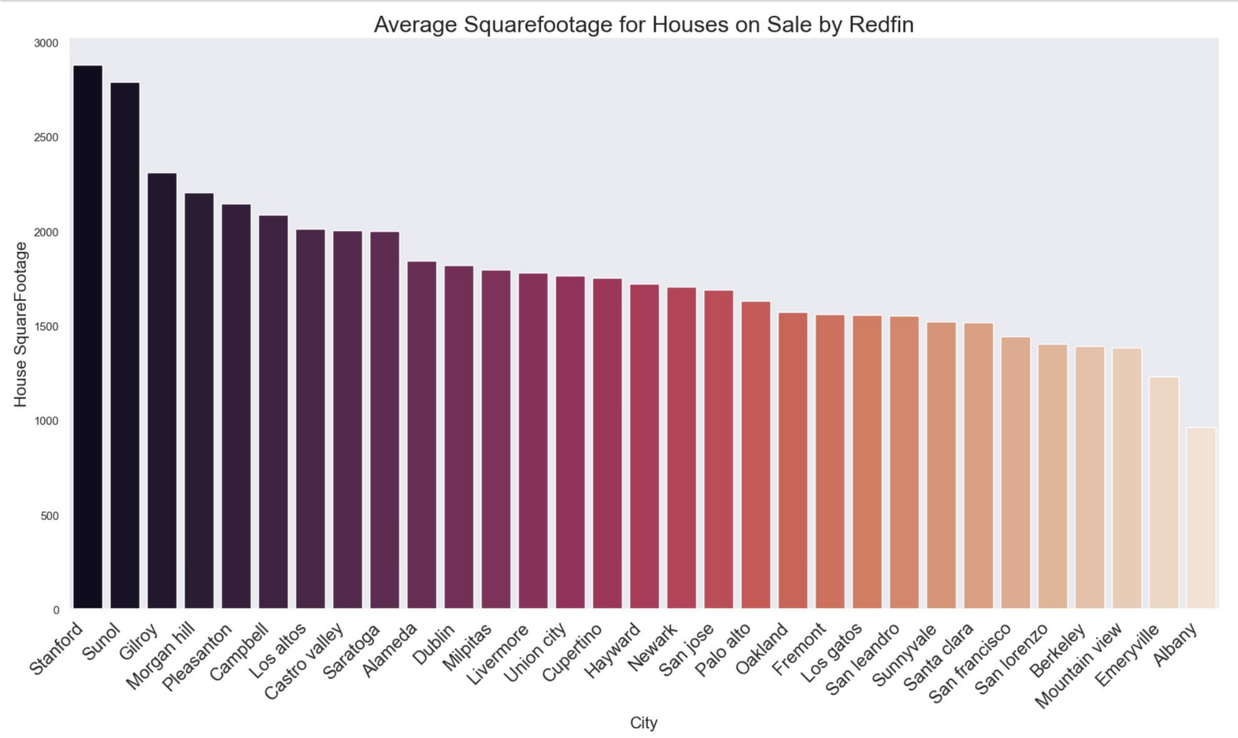
Data Analysis - Logic Check

(Analysis by City Groupings)

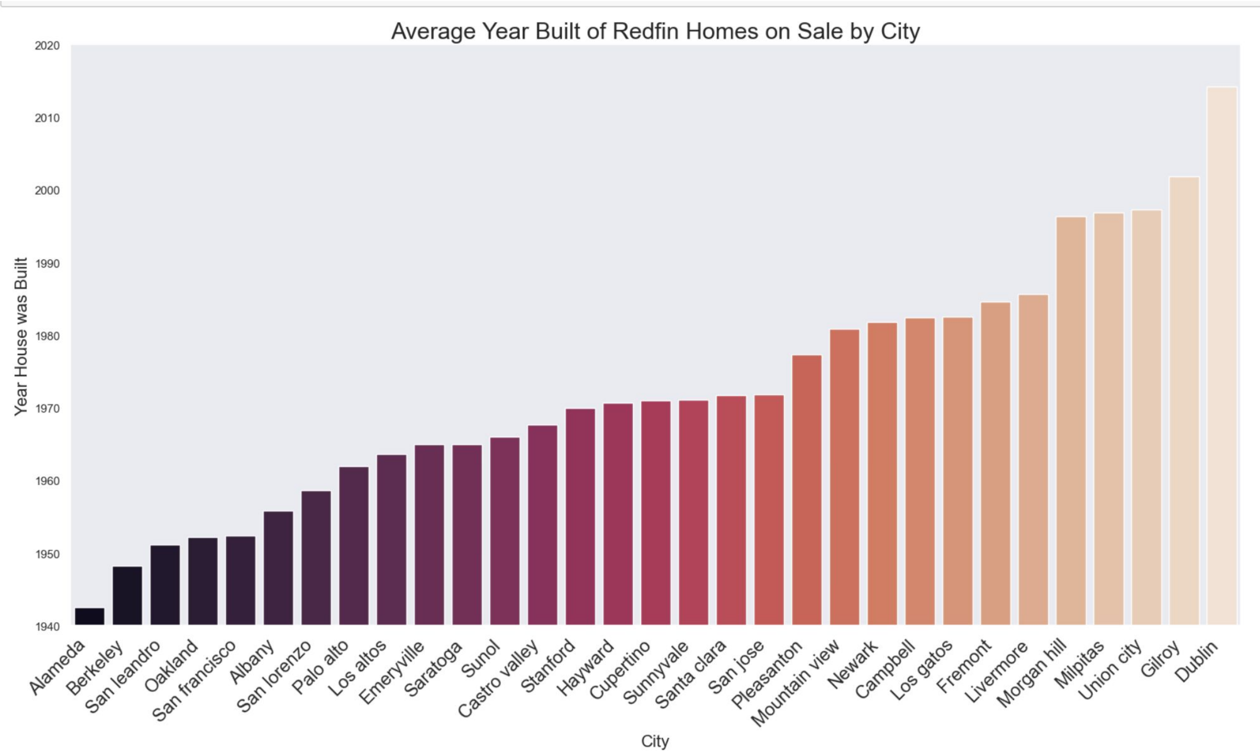
Avg Listing Prices in Bay by City



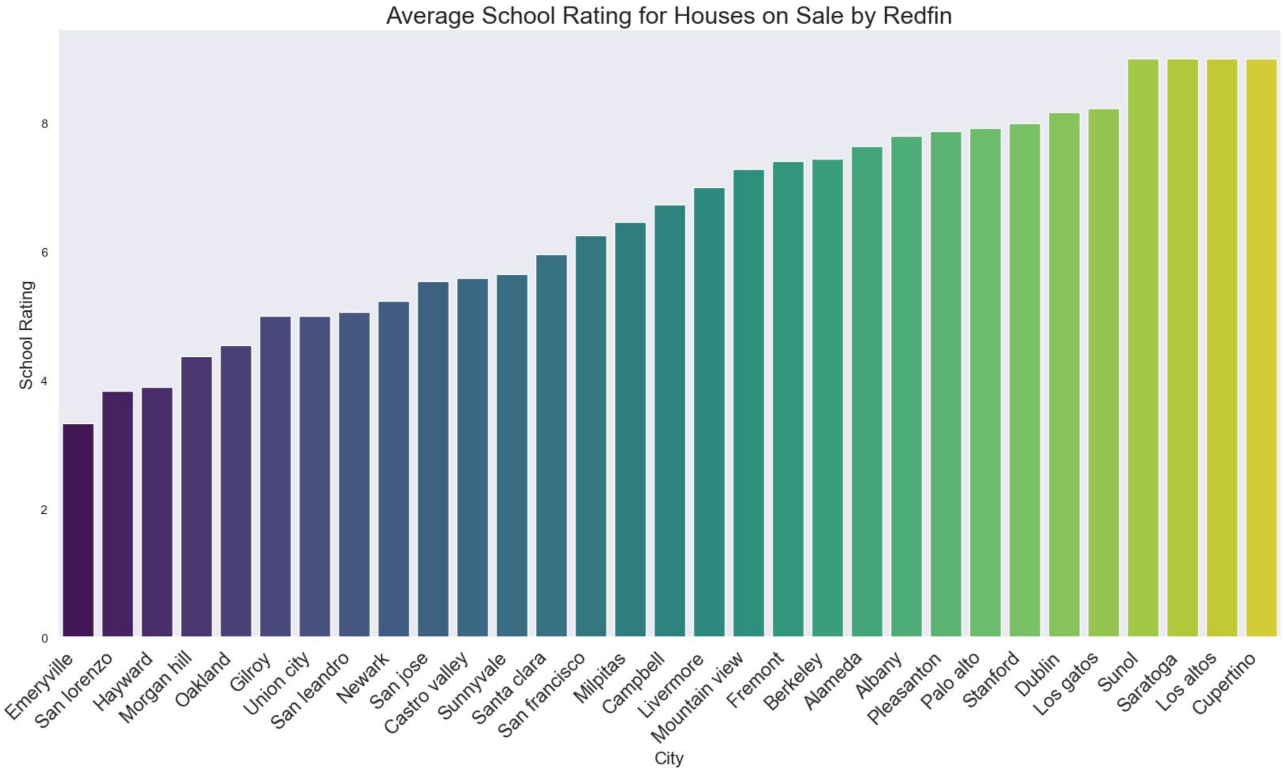
Avg Listing Sqft in Bay by City



Average Listing Year Built in Bay by City

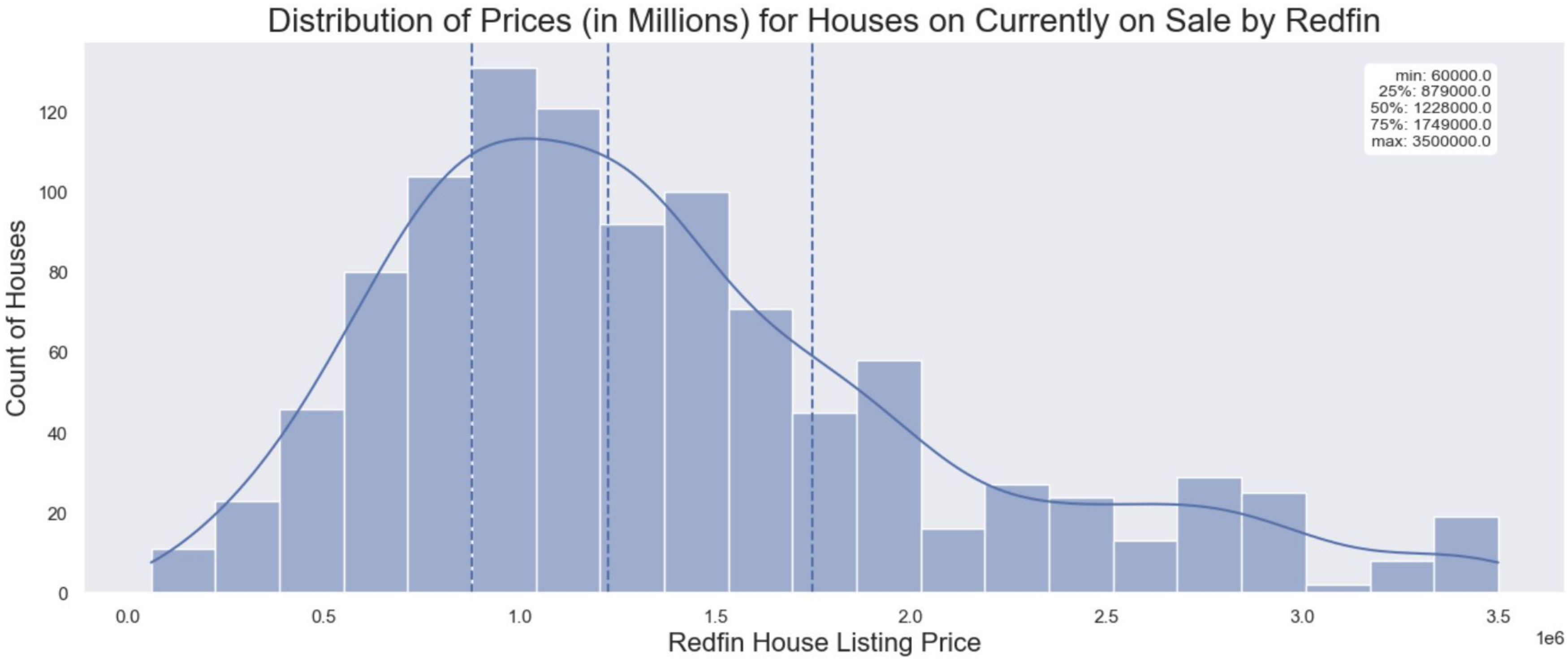


Avg Rating of Closest School by City

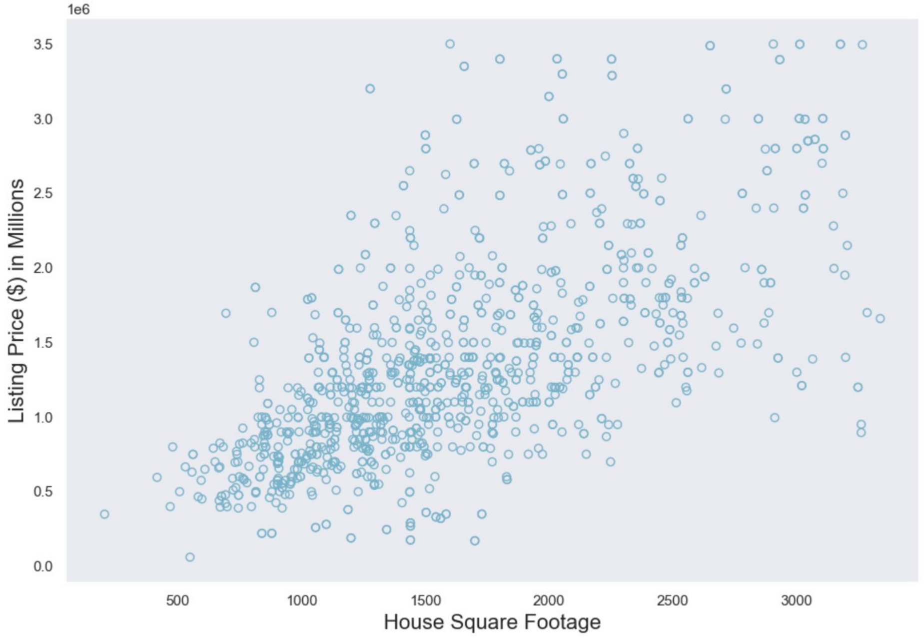


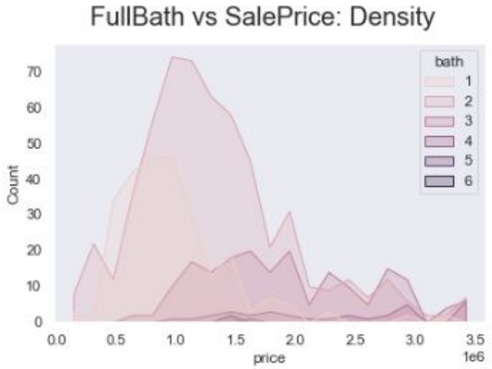
Data Analysis - Price Prediction

Distribution of Price (outliers removed)

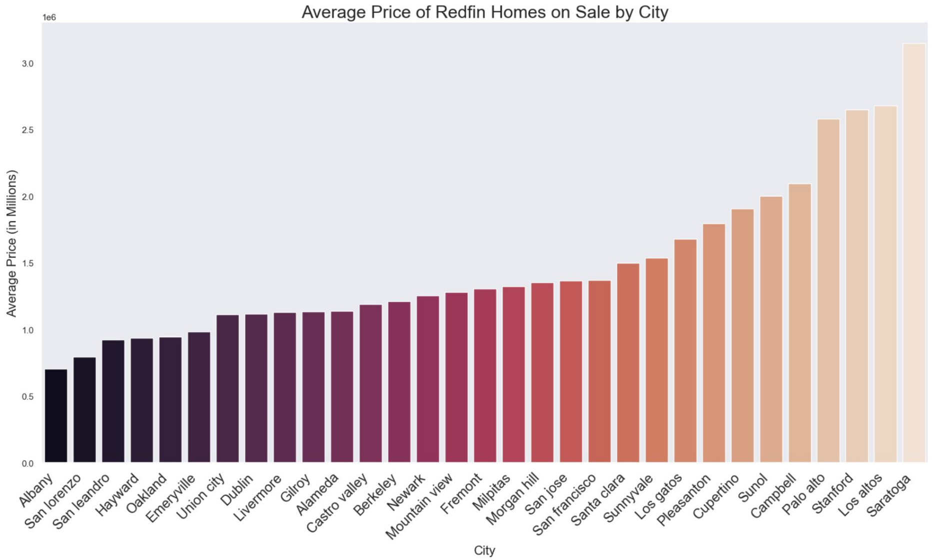


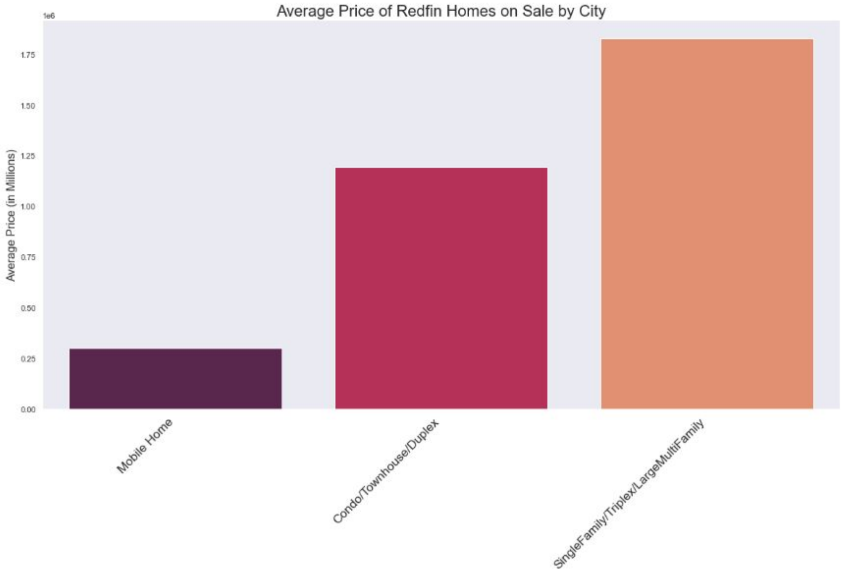
Numerical Variables - Correlated with Price



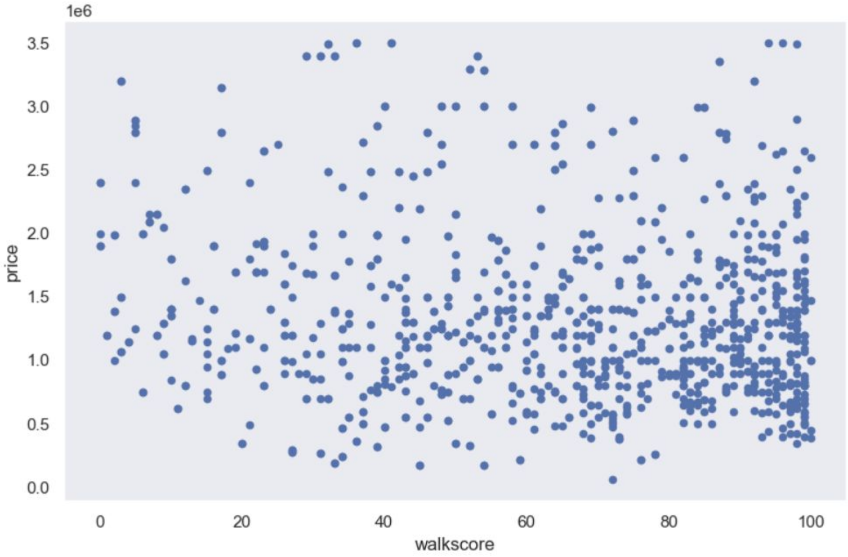


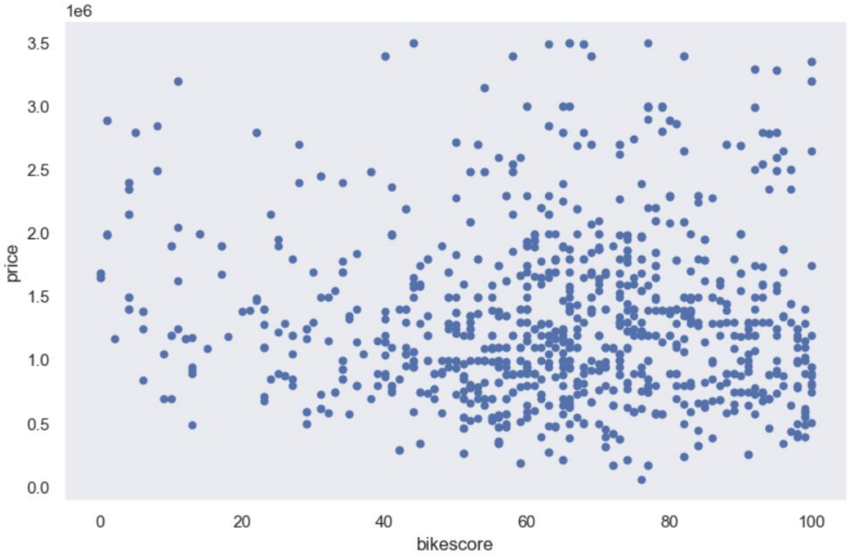
Categorical Variables - Correlated with Price



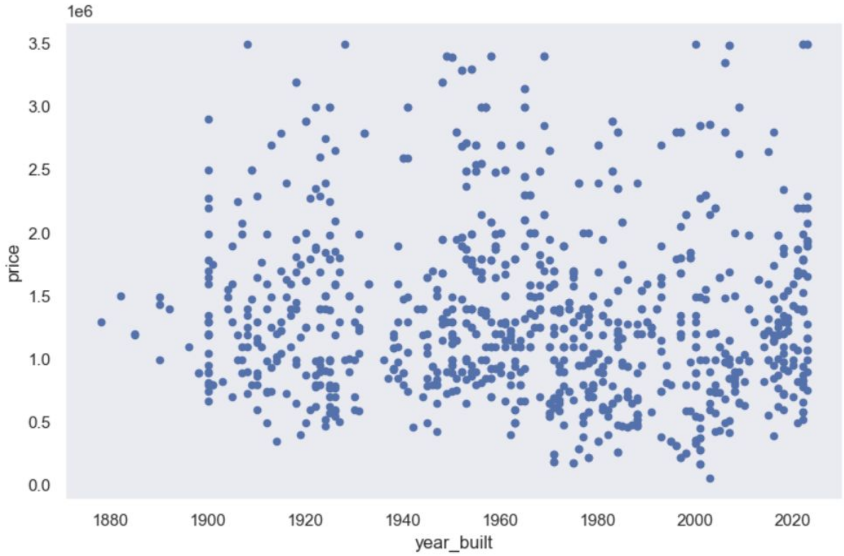


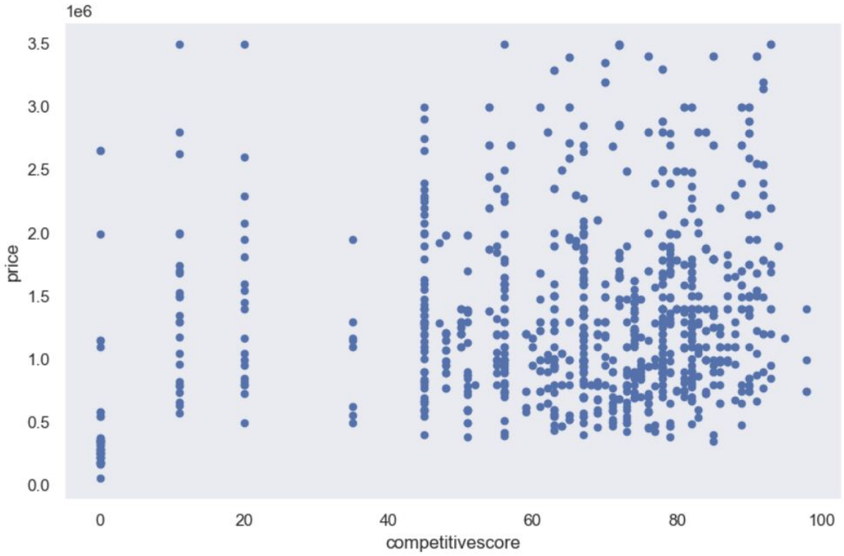
Numerical Variables - Not Correlated with Price



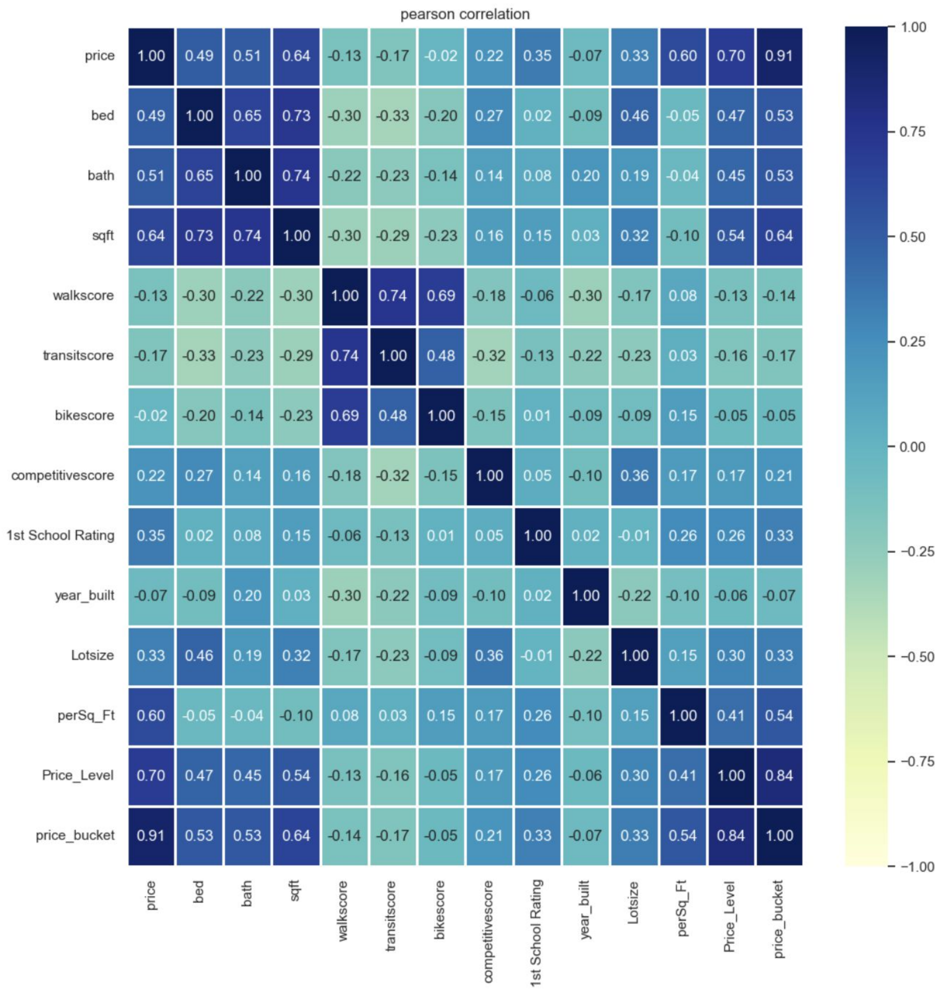


Numerical Variables - Not Correlated with Price





Pearson Correlation



Data Processing

Adding Dummies to Training & Validation Data

```
                                 House_Type  House_Type_Mobile Home  \
146                  Condo/Townhouse/Duplex                       0
118                  Condo/Townhouse/Duplex                       0
1025  SingleFamily/Triplex/LargeMultiFamily                       0
982   SingleFamily/Triplex/LargeMultiFamily                       0
836   SingleFamily/Triplex/LargeMultiFamily                       0

      House_Type_SingleFamily/Triplex/LargeMultiFamily
146                                                  0
118                                                  0
1025
982
836
```

```
            city  city_Albany  city_Berkeley  city_Campbell  \
146      Oakland            0              0              0
118    Livermore            0              0              0
1025    Campbell            0              0              1
982   Morgan hill           0              0              0
836     San jose            0              0              0

      city_Castro valley  city_Cupertino  city_Dublin  city_Emeryville  \
146                    0               0            0                0
118                    0               0            0                0
1025                   0               0            0                0
982                    0               0            0                0
836                    0               0            0                0
```

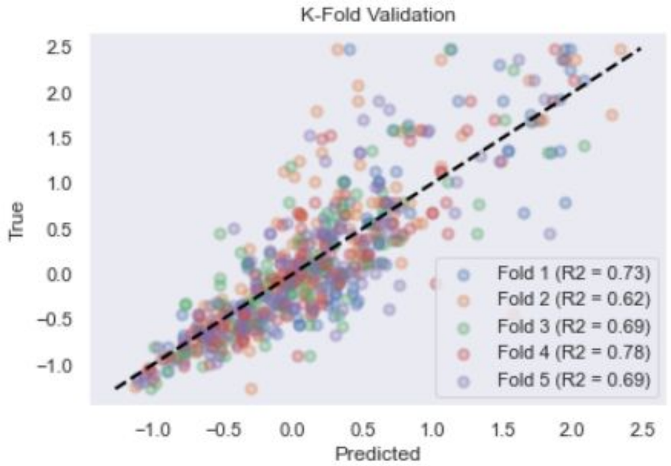
Normalizing Training & Validation Data

	bed	bath	sqft	year_built	House_Type_Mobile Home	House_Type_SingleFamily/Triplex/LargeMultiFamily	city_Albany	city_Berkeley	city_Campbell
146	1.0	0.0	0.892365	-1.085714	0.0	0.0	0.0	0.0	0.0
118	0.5	0.0	0.180225	0.419048	0.0	0.0	0.0	0.0	0.0
1025	1.0	2.0	1.912390	0.647619	0.0	1.0	0.0	0.0	1.0
982	0.5	0.0	0.538173	0.190476	0.0	1.0	0.0	0.0	0.0
836	0.5	0.0	0.700876	0.152381	0.0	1.0	0.0	0.0	0.0

Machine Learning Methods

#1. Multiple Regression

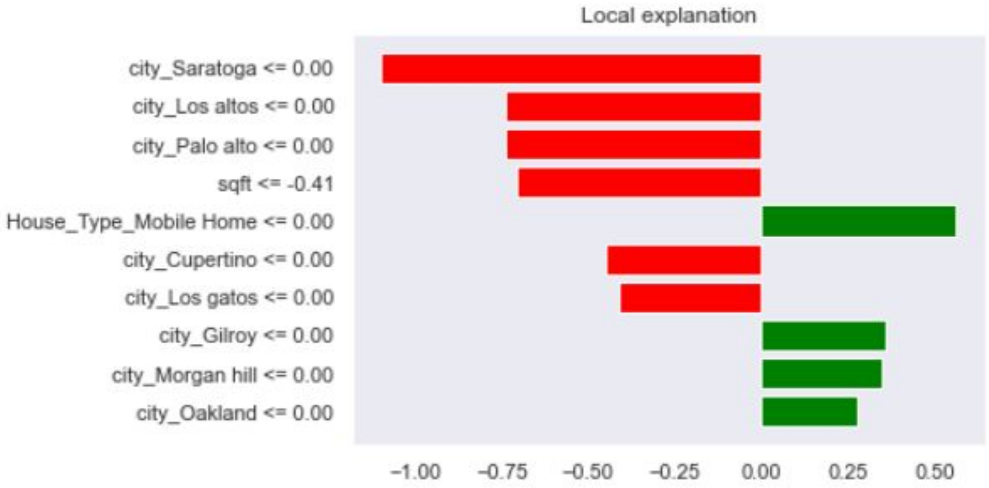
- Variables Selected (post trial & error)
    - Bath
    - Sqft
    - House Type
    - City
- On the right are the k-fold regression results post gradient boosting




```
Regression statistics

                      Mean Error (ME) : -49293.8686
       Root Mean Squared Error (RMSE) : 373385.3633
            Mean Absolute Error (MAE) : 285197.1125
          Mean Percentage Error (MPE) : -7.7365
Mean Absolute Percentage Error (MAPE) : 30.6946
```

#1. Multiple Regression Coefficient Analysis

- Variables with a positive effect on price:
  - Bath
  - Sqft
  - Single Family House Type
  - Cities such as Cupertino, Los Gatos, etc
- Variables with a negative effect on price:
  - Mobile Home House Type
  - Cities such as Oakland, Gilroy, Newark, etc

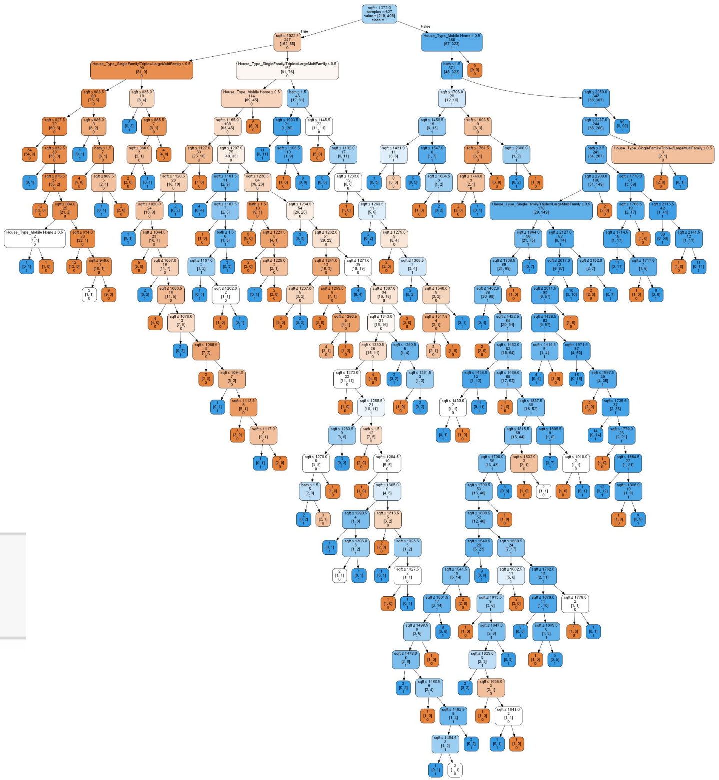


#2. CART Analysis (Binary Classification)

- First tried to do a CART to predict whether a Redfin listing would be less than or greater than $1M
- Predictors:
  - Bath, Sqft, House Type
- Cannot read tree on right but showed similar trends as Regression

```
#classificationSummary(valid_y, boost.predict(valid_X))
accuracy_score(valid_y, boost.predict(valid_X))
```

```
0.8229665071770335
```



#2. CART Analysis (Non-Binary Classification)

- Grouped house prices into 5 categories:
  - 1 if price <= $250k
  - 2 if price <= $500k
  - 3 if price <= $1M
  - 4 if price <= $1.5M
  - 5 if price > $1.5M
- For same predictors (bath, sqft, house type) – get a lower predictive score on validation data

Confusion Matrix (Accuracy 0.8459)

	Prediction				
Actual	0	1	2	3	4
0	12	0	1	0	0
1	6	36	8	0	2
2	0	7	285	23	13
3	0	0	32	240	42
4	0	1	9	17	311

```
#Calculate accuracy for validation data
accuracy_score(valid_y, NewClassTree.predict(valid_X))
```

```
0.6578947368421053
```

#2. CART Analysis (Non-Binary Classification)

- Added in city as a predictor to try and increase the classification score
- New predictors:
  - Bath
  - Sqft
  - City
  - House Type
- Algorithm is able to correctly classify housing level for roughly 70% of the houses in the validation data set

Confusion Matrix (Accuracy 0.8670)

	Prediction				
Actual	0	1	2	3	4
0	12	1	0	0	0
1	0	39	12	0	1
2	0	7	288	23	10
3	0	0	34	250	30
4	0	1	3	17	317

```
#Calculate accuracy for validation data
accuracy_score(valid_y, NewClassTree.predict(valid_X))
```

```
0.6961722488038278
```

#3. kNN

First model used a continuous price variable - the goal was to build a broad kNN Model, which would be refined later

Categorical variables were converted into binary dummy variables

```
#Start with continuous price model:
X_names = ['bath', 'sqft', 'House_Type_Mobile Home',
       'House_Type_SingleFamily/Triplex/LargeMultiFamily', 'city_Albany',
       'city_Berkeley', 'city_Campbell',
       'city_Cupertino', 'city_Dublin', 'city_Emeryville', 'city_Fremont',
       'city_Gilroy', 'city_Hayward', 'city_Livermore', 'city_Los altos',
       'city_Los gatos', 'city_Milpitas', 'city_Morgan hill',
       'city_Mountain view', 'city_Newark', 'city_Oakland', 'city_Palo alto',
       'city_Pleasanton', 'city_San francisco', 'city_San jose',
       'city_San leandro', 'city_San lorenzo', 'city_Santa clara',
       'city_Saratoga', 'city_Stanford', 'city_Sunnyvale',
       'city_Union city']
```

#3. kNN cont...

Root Mean Square Error (RMSE) where Number of Neighbors is 3

Models suffers from overfitting since RMSE on training data is less than RMSE on validation data

```
from sklearn.metrics import mean_squared_error
from math import sqrt
train_preds = knn_model.predict(X_train)
mse = mean_squared_error(y_train, train_preds)
rmse = sqrt(mse)
rmse
```

```
0.32768971834177363
```

```
test_preds = knn_model.predict(X_test)
mse = mean_squared_error(y_test, test_preds)
rmse = sqrt(mse)
rmse
```

```
0.4899092890860622
```

#3. kNN cont...

Improved RMSE with gridsearch - 8 nearest neighbors

```
#Compare the fit of the model with 8 nearest nei
knn_model8 = KNeighborsRegressor(n_neighbors=8)
knn_model8.fit(X_train, y_train)
train_preds8 = knn_model8.predict(X_train)
mse = mean_squared_error(y_train, train_preds8)
rmse = sqrt(mse)
rmse
```

```
0.4178482948516704
```

```
#Confirm above result for the validation data
test_preds8 = knn_model8.predict(X_test)
mse = mean_squared_error(y_test, test_preds8)
rmse = sqrt(mse)
rmse
#Conclusion: 8 nearest neighbors produces a bett
#but performance is worse on the validation data
```

```
0.516886003517522 2
```

#3. kNN cont...

Repeat process for categorical price variable

```
Index(['zSQFT', 'zYEAR_BUILT', 'zBED', 'zBATH', 'price_bucket',
       'House_Type_Mobile Home',
       'House_Type_SingleFamily/Triplex/LargeMultiFamily', 'city_Albany',
       'city_Berkeley', 'city_Campbell', 'city_Castro valley',
       'city_Cupertino', 'city_Dublin', 'city_Emeryville', 'city_Fremont',
       'city_Gilroy', 'city_Hayward', 'city_Livermore', 'city_Los altos',
       'city_Los gatos', 'city_Milpitas', 'city_Morgan hill',
       'city_Mountain view', 'city_Newark', 'city_Oakland', 'city_Palo alto',
       'city_Pleasanton', 'city_San francisco', 'city_San jose',
       'city_San leandro', 'city_San lorenzo', 'city_Santa clara',
       'city_Saratoga', 'city_Stanford', 'city_Sunnyvale', 'city_Sunol',
       'city_Union city'],
      dtype='object')
```

#3. kNN cont...

Better fit on the Validation data

Confusion Matrix (Accuracy 0.5789)

		Prediction				
	Actual	0	1	2	3	4
<=$60000	0	69	18	3	2	1
<=$798978	1	16	36	30	9	1
<=$1096800	2	5	10	40	23	11
<=$1395000	3	2	0	15	32	14
<=$3500000	4	0	0	6	10	65

#4. Logistic Regression

```
intercept -6.431316309152559
          zSQFT  zYEAR_BUILT     zBED     zBATH  House_Type_Mobile Home  \
coeff -3.226426     0.071906 -0.48088  0.344737               18.217774

       House_Type_SingleFamily/Triplex/LargeMultiFamily  city_Albany  \
coeff                                         -3.042326     3.619037

       city_Berkeley  city_Campbell  city_Castro valley  city_Cupertino  \
coeff       0.935837       4.226227           -5.864802       -6.811834

       city_Dublin  city_Emeryville  city_Fremont  city_Gilroy  city_Hayward  \
coeff    -4.97703         2.700597      3.242575     7.696592     -5.113248

       city_Livermore  city_Los altos  city_Los gatos  city_Milpitas  \
coeff        3.503967       -0.001908         2.96611       2.630427

       city_Morgan hill  city_Mountain view  city_Newark  city_Oakland  \
coeff         -3.475936            4.176835     4.850274      5.351323

       city_Palo alto  city_Pleasanton  city_San francisco  city_San jose  \
coeff       -7.673467        -1.967396            1.597245       3.645924

       city_San leandro  city_San lorenzo  city_Santa clara  city_Saratoga  \
coeff          5.863706          4.755488          2.553526      -0.070444

       city_Stanford  city_Sunnyvale  city_Sunol  city_Union city
coeff       0.006468       -3.178064   -0.015847          5.36696
AIC 1227.857088696052
```

#4. Logistic Regression cont...

Result is comparable to the grid search kNN with 8 nearest neighbors

Confusion Matrix (Accuracy 0.5167)

Actual \ Prediction	<=$60000	<=$798978	<=$1096800	<=$1395000	<=$3500000
<=$60000	72	17	3	0	1
<=$798978	25	33	18	15	1
<=$1096800	4	22	21	32	10
<=$1395000	1	4	11	35	12
<=$3500000	0	4	4	18	55

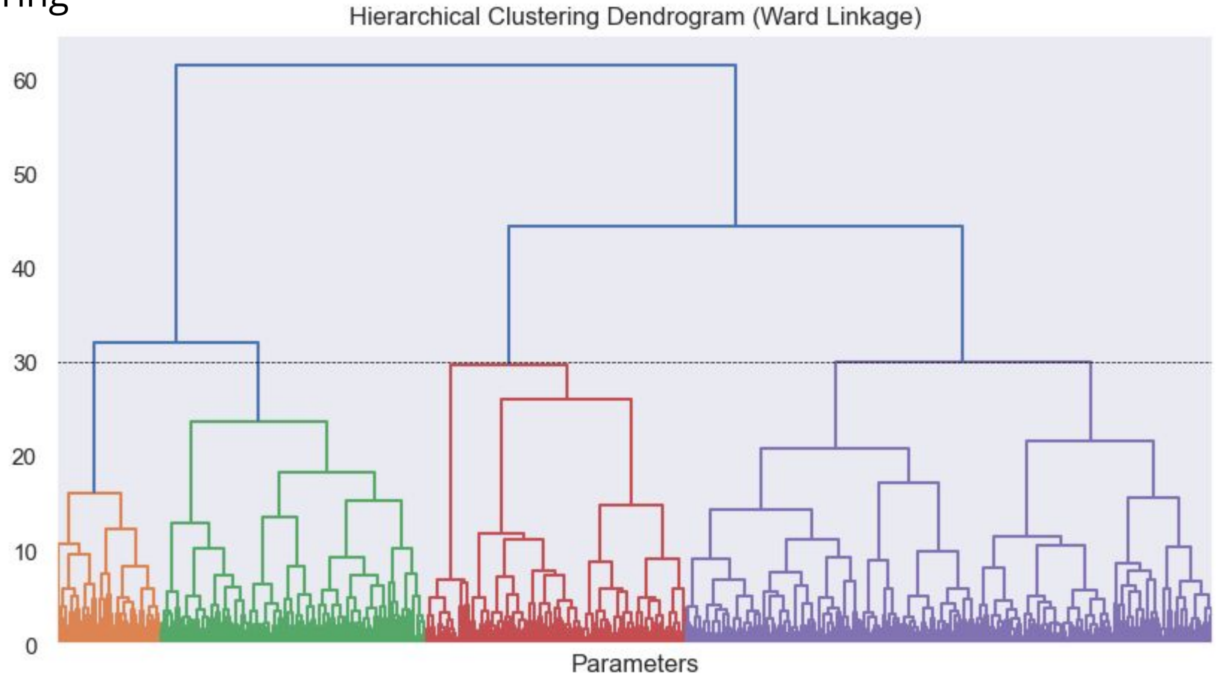
#5. Clustering

Variables used for this analysis (normalized prior to analysis):

	price	bed	bath	sqft	walkscore	transitscore	bikescore	competitivescore	1st School Rating	year_built
0	649000	2	2.0	1037.0	96	86	91	76	2.0	2003
1	699000	2	2.0	990.0	82	51	47	71	3.0	1930
2	685000	1	1.5	1010.0	94	59	86	69	4.0	2006
3	1000000	2	2.5	1492.0	76	41	56	77	7.0	2007
4	749000	4	2.0	1836.0	91	57	68	73	2.0	1941

#5. Clustering (Dendrogram)

Dendrogram with a color threshold of 30 - preliminary visualization of subsequent clustering



#5. Clustering

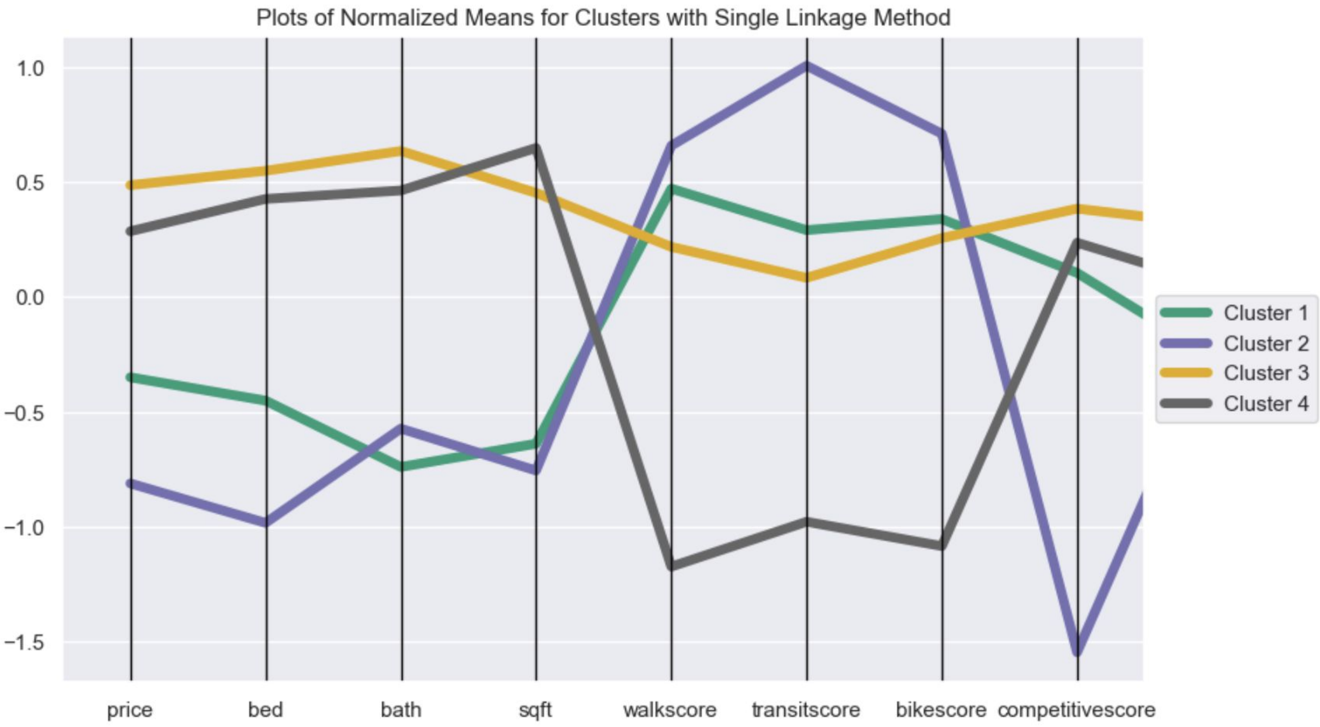
De-normalized representation of 4 clusters

Means of Input Variables for Clusters with Ward Linkage Method

	price	bed	bath	sqft	walkscore	transitscore	bikescore
1	9.055e+05	2.285	1.552	1182.659	77.470	60.339	71.610
2	1.160e+06	3.253	2.000	1680.094	29.953	21.076	37.641
3	1.883e+06	3.907	2.577	2182.814	88.515	67.515	76.330
4	1.979e+06	3.705	2.462	2039.884	52.976	30.283	62.954

	competitivescore	1st School Rating	year_built	Price_Level_New	Cluster
1	57.742	5.510	1966.076	3.185	Cluster 1
2	69.682	5.429	1974.253	3.729	Cluster 2
3	60.289	6.577	1918.062	4.711	Cluster 3
4	76.456	6.836	1978.900	4.638	Cluster 4

#5. Clustering - Visual Representation




THANK
YOU!