Real Estate Model Building

The Real Estate Market is complicated and for many requires a professional Real Estate Agent to navigate the market and buy the property they are looking for at a reasonable price. However, it is hard to objectively determine whether the price of the property sold/bought for is actually a "fair price" and it requires a lot of faith and trust from buyer/seller that the agent is acting in good faith. Therefore the goal of this project is to see if a model could be built that could accurately determine a "fair price" for a property given information about the property which would allow even laymens to determine if the property is fair.

Audience

Any person who is looking into real estate could benefit from being able to quickly determine a fair price for the property.

Data Source

The Data Source was obtained by utilizing Rapid Api to extract real estate data. Due to the nature of the API it has a limit of 200 real estate properties per call. To circumvent this, I included the 10 highest populated cities in CA allowing me to extract 2000 real estate properties in a given call. Given my inexperience with API and learning that after a certain amount of calls I would get charged, I decided to only call the API once when all the bugs were worked out (this will be further elaborated in "Potential Improvements" section).

About the Data

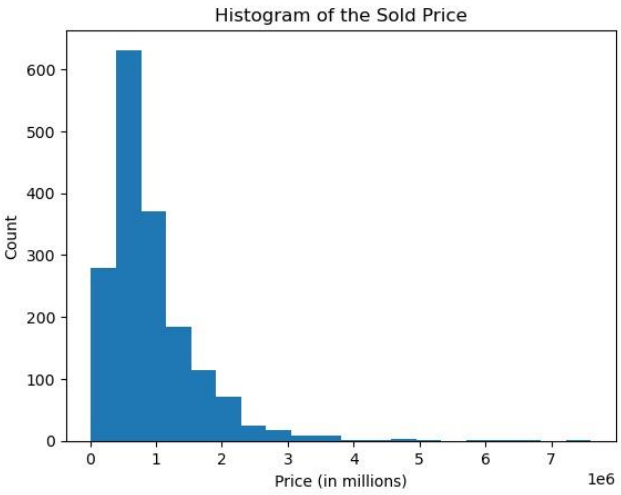
The data contained information about the most currently sold real estate property in the given city at the time of extraction. The properties were sold between August 9th 2021 and September 1st 2021. The Data was turned into a table with the following features:

- Number of Baths including number of (total, full, 3qtr, half, 1qtr) baths
- Number of beds
- The City it property was sold in
- Number of Garages
- The List Price
- Sold Price
- Date Sold
- Sqft
- Lot Sqft
- Stories
- Year built
- Type
- Sub Type

Data Wrangling

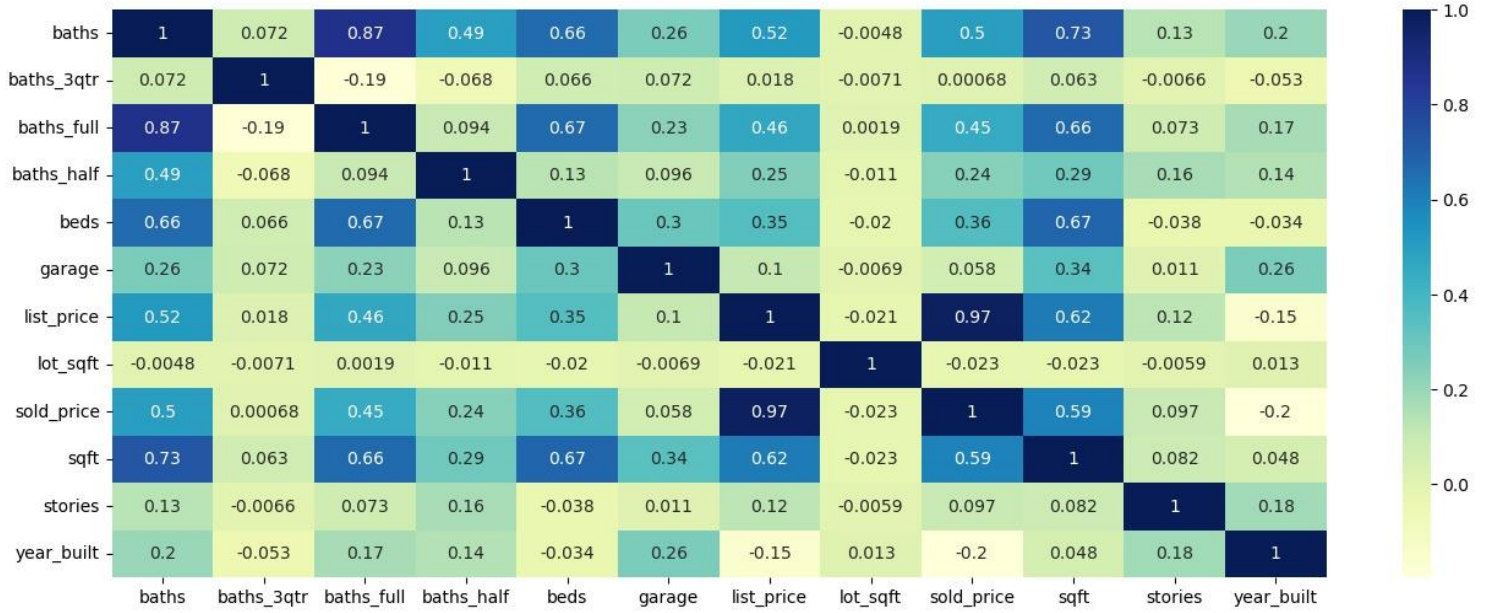
- Feature Analysis
    - Find which features do not provide much information to the overall data
        - Learned that Sub-Type could easily be replaced with Type, removed Sub-Type
        - Utilized only total number of Baths rather than all the extra information of whether the baths were full, 3qtr, half etc
- Problem 2: Row Analysis
    - Look for interesting or weird observations
        - Due to lack of domain knowledge I removed the observations that had “land” as their type. Since the real estate was just land the only value it had was its square footage, since I was looking towards modeling home prices I removed these observations.
        - I was unsure what real estate could have 0 baths so I removed all these observations as well.

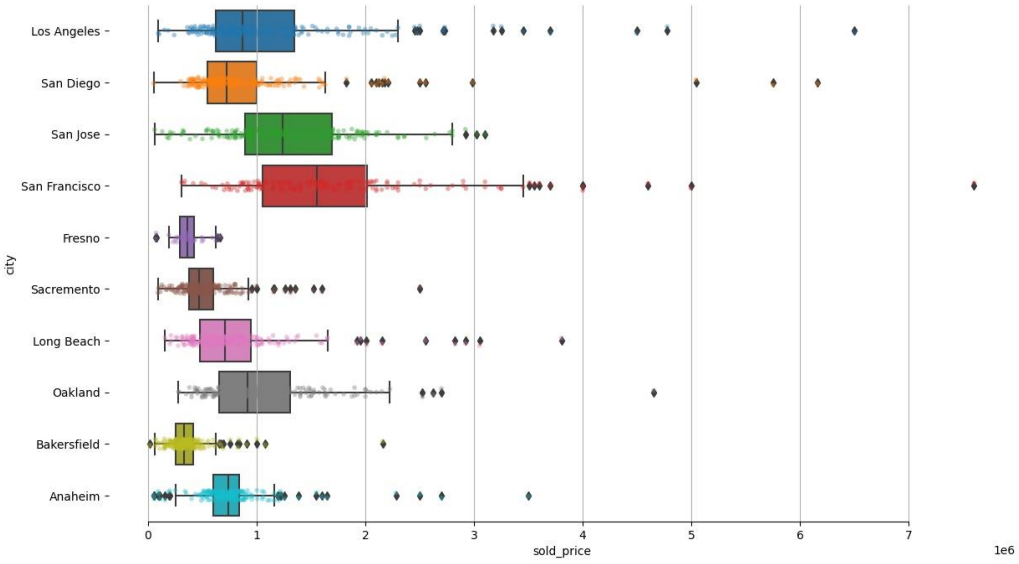
EDA: Full Observation can be found here



Most of the real estate sold was between 0-1.5 millions dollars.

Highest correlated features with sold price (not including listing price) are number of beds, baths, garages, and square footage.





The sold price distribution between cities differ by around 1 million with some cities (fresno and bakersfield) having similar distributions.

Modeling

I utilized a package known as pycaret to determine the best modeling algorithm to use for the data set. I analyzed models based on two target variables, the first was the sold price and the second was the log(sold price). The results were as followed:

Sold Price

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	199452.1343	126807453611.7003	350583.1489	0.7657	0.3149	0.2590	0.0200
lightgbm	Light Gradient Boosting Machine	212754.0115	140476135581.3947	369102.1896	0.7392	0.3428	0.2646	0.0140
et	Extra Trees Regressor	207905.0983	144481421966.1154	376155.7301	0.7299	0.3253	0.2533	0.0630
rf	Random Forest Regressor	206299.9497	145827374083.7755	375144.7572	0.7297	0.3310	0.2723	0.0720
ridge	Ridge Regression	249142.8156	168812668518.4000	401267.2000	0.6927	0.5267	0.4100	0.0070
lasso	Lasso Regression	251951.3828	169169197465.6000	401998.8156	0.6901	0.5697	0.4278	0.2590
llar	Lasso Least Angle Regression	251945.6944	169188297995.0217	402025.4545	0.6901	0.5699	0.4277	0.0040
lar	Least Angle Regression	251981.0144	169208813220.3622	402052.7337	0.6900	0.5696	0.4279	0.0040
lr	Linear Regression	252527.6922	169211602534.4000	402163.5625	0.6898	0.6341	0.4298	0.4430
knn	K Neighbors Regressor	233496.9281	186775002316.8000	422312.2812	0.6629	0.3667	0.2911	0.0060
dt	Decision Tree Regressor	267540.9220	234022815112.0876	478300.6810	0.5320	0.4269	0.3371	0.0060
omp	Orthogonal Matching Pursuit	360398.0634	268536290760.1895	512671.6652	0.4943	0.6361	0.6209	0.0040
huber	Huber Regressor	276654.4569	301751898049.8934	540454.6355	0.4498	0.4178	0.3237	0.0060
ada	AdaBoost Regressor	458923.1988	321042559071.3930	562889.0060	0.3835	0.7467	1.0623	0.0220
en	Elastic Net	447974.7719	458750699110.4000	671757.5062	0.1438	0.7038	0.8569	0.0040
par	Passive Aggressive Regressor	411800.7948	498787347261.8597	701132.6026	0.0656	0.6248	0.5619	0.0190
br	Bayesian Ridge	496629.5856	533822100357.9896	725943.5915	-0.0032	0.7671	0.9726	0.0040
dummy	Dummy Regressor	496629.5844	533822101913.6000	725943.5938	-0.0032	0.7671	0.9726	0.0040

Log Sold Price

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	0.2104	0.0865	0.2922	0.8328	0.0205	0.0157	0.0240
lightgbm	Light Gradient Boosting Machine	0.2176	0.0956	0.3065	0.8167	0.0215	0.0162	0.0160
rf	Random Forest Regressor	0.2206	0.0993	0.3122	0.8096	0.0220	0.0165	0.0840
et	Extra Trees Regressor	0.2239	0.1026	0.3164	0.8036	0.0222	0.0167	0.0620
ridge	Ridge Regression	0.2405	0.1192	0.3405	0.7702	0.0235	0.0179	0.0080
huber	Huber Regressor	0.2243	0.1284	0.3446	0.7528	0.0233	0.0166	0.0100
br	Bayesian Ridge	0.2350	0.1302	0.3488	0.7499	0.0237	0.0175	0.0070
knn	K Neighbors Regressor	0.2560	0.1318	0.3595	0.7470	0.0252	0.0191	0.0070
dt	Decision Tree Regressor	0.2946	0.1762	0.4173	0.6621	0.0292	0.0220	0.0060
ada	AdaBoost Regressor	0.3815	0.2278	0.4768	0.5549	0.0333	0.0286	0.0260
par	Passive Aggressive Regressor	0.4422	0.3284	0.5593	0.3371	0.0386	0.0329	0.0050
omp	Orthogonal Matching Pursuit	0.4681	0.3856	0.6172	0.2559	0.0430	0.0353	0.0040
lasso	Lasso Regression	0.5640	0.5259	0.7235	-0.0157	0.0506	0.0423	0.4560
en	Elastic Net	0.5640	0.5259	0.7235	-0.0157	0.0506	0.0423	0.0050
llar	Lasso Least Angle Regression	0.5640	0.5259	0.7235	-0.0157	0.0506	0.0423	0.0050
dummy	Dummy Regressor	0.5640	0.5259	0.7235	-0.0157	0.0506	0.0423	0.0090
lar	Least Angle Regression	0.6294	133.1679	3.9647	-292.8231	0.0572	0.0479	0.0050
lr	Linear Regression	0.6294	133.1705	3.9647	-292.8287	0.0572	0.0479	0.5150

Pycaret shows that overall the Gradient Boosting Regressor is the best model for both predicting the sold price as well as the log of the sold price. Interestingly it may be better to preprocess the data using the log of the sold price as it seems that the R2 is much higher indicating that the model works better when the sold price is logged.

Potential Improvements

- Get more domain knowledge of real estate to fully understand other real estate properties such as “land” and properties with 18 garages but nothing else.
- Utilize the API better and call it every few weeks to import more data over time. Due to fear of getting charged I only used the API call once which reduced my data to only 2000 observations.
- Use PyCaret better and understand the whole functionality.
- Improve readability of EDA and cleaning.