Predicting House Prices in Los Angeles Using Machine Learning Methods

Ali Tahsili – Student No.: 0704606

A project submitted to the School of Graduate Studies as a partial fulfillment of the Master of Science degree in Financial Analytic

Prof. David Riegert

April 29, 2022

INTRODUCTION

Prediction is an exciting and fundamental aspect of statistics and inference; it is used to estimate model parameters, validate those models, and importantly: make decisions in the present about the unknown future. It anticipates the sense of what the outcome would look like from its features. There are traditional ways to predict a variable from its characteristics: Ordinary Least Square (OLS) is one of them. However, there are some limitations in traditional prediction methods like linear regression. For instance, most of the time, the relationship between the target and observed variables is not linear. Also, the error's heteroskedasticity and non-normality might cause critical problems. We could not validate the result of linear regression if the assumptions were violated. Therefore, new methods have been developed to overcome these issues. Nowadays, many studies use nonstructural data like text, images, or videos, which cannot be processed and modelled by traditional OLS methods. Supervised Machine Learning (ML) methods come in handy to predict phenomena that are not the typical data structure. In addition, advanced ML techniques can solve data issues with nonlinearity and heteroskedasticity.

In this study, my attempt was to find the best models to predict housing prices in Los Angeles, California. After adjusting for inflation, We applied linear regression with OLS (traditional hedonic regression) and various supervised ML methods, Regularized Linear Regression (LASSO, Ridge, Elastic Net) and Tree-Based Methods (Decision tree, Random Forest, Boosted Regression Trees like XGBoost), to predict real estate prices. We created the models with data from June 1964 to August 2021. To assess the models, we compared the result of each model with the test data from August 2021 to January 2022. There are various methods to assess the prediction performance: R^2, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We compared all methods to find which model has the highest R^2, lowest MAE and lowest RMSE.

Advance ML methods like random forest or boosted regression trees had a superior performance metrics compared to a classical linear regression model. In this project, the boosted regression trees model had an R^2 value of 86% which was more than twice of the OLS baseline method. MAE could not define what the relative size of the error is. Mean Absolute Percentage Error (MAPE) which is the division of MAE by the mean of data could tell us how accurate the model is compared to other models.

Based on the folds from cross-validation, on average, we achieve an estimated MAPE of 14.4% with boosted regression trees compared to 28.3% with OLS. In monetary units, the enhanced pricing accuracy relates to a reduction in pricing error by approximately 143,000 USD for an average property price of 1,000,000 USD. We infer that nonlinearities and interaction effects captured by complex ML methods are relevant for real estate pricing.

DATA PREPARATION

– *Scraping and Cleaning Data*

We used the https://www.realtor.com website to scrape the data. There were 206 pages containing 42 properties on each page resulting from a search for "Los Angeles, CA". This data was from the properties which have been sold between October 26, 2021 to January 26, 2022 since only 3 month worth of prices were available on that website. Each sold house had a specific link for further information related to its characteristics.[1]

After scraping the website, Realtor.com, there were 8652 saved HTML-type variables in a list. We extracted information from all these pages with the help of the "rvest" package (Wickham, 2021). We cleaned the data using regular expression (regex) and determined each feature's type. These features were including number of bedrooms, bathrooms, parking, lot size, building size, price/sqft, build year, renovated year, having a garage, number of garages, type of heating, cooling, water source and sewer system. In addition, there is information related to the historical transaction prices of each property. We combined historical and current prices into one dataset and created about 55,000 observations of 23 variables.

– *Preprocessing data*

There were two types of data in the dataset: nominal and numeric. Some of the nominal variables such as type of the water source and sewer system did not add valuable information. We factorized the nominal columns and made their values more descriptive.

To consider the effect of inflation in the time series dataset, we collected the Consumer Price Index (CPI) for each month. We assumed January 2020 as a base point for inflation and changed the adjusted factor by that date. We implemented this factor into the "Price"" and "Price/SQFT" columns to change them as adjusted prices.

The address column by itself does not have any predictive power, however geocodes can be extracted from addresses to add some valuable information for analysis. We extracted Longitude and Latitude using a Google Sheets Add-on to use address information. There were some limitations to extract geocode from the US addresses that added time to the pre-processing stage.[2]

[1] During the scraping process, an error came up every now and then. To circumvent this issue, we used a VPN to access the website and changed the country IP for every 600 links. To have access and retrieve the data, we stored all the 42*206 (8652) pages in HTML format into a list. Allocating several HTML-type variables into one list was not straightforward. When an HTML variable is stored into a list, there is an external pointer defined during the process, and at the time of retrieving that list, external pointer will lose. So, we needed to change each HTML variable into characters and save them. Then turn them into HTML type and keep them as RDS.

[2] Every account has a 1000 address limitation for retrieving geocode information from addresses. We created nine google accounts to access 8500 address geocodes.

– *Dealing with missing data*

Some features such as Heating, Cooling, Water Source, Sewer systems have more than 50% missing values. We did not consider these features for the data analysis. The rest of the features have less than 10% missing values. Thus, we removed all the rows with missing values. At last, about 48,500 observations and 15 features remained for analysis.

– *Outliers*

In this study, we employed different methods from classical linear regression to boosted regression trees. Linear and regularized regression models were not robust to outliers. To get a reasonable result for all models, we decided to remove extreme outliers.

Also, some of the variables in the dataset were extremely right skewed. To filter those observations, we needed to trim the data. IQR is the difference between the 75th and the 25th percentile values of the data. In general, data values outside of the following are considered outliers: +1.5 x IQR + 3rd Quartile Upper Bound; and -1.5 x IQR + 2nd Quartile Lower Bound. Using a multiple of 3.0 (instead of 1.5) times, IQR would indicate extreme values.

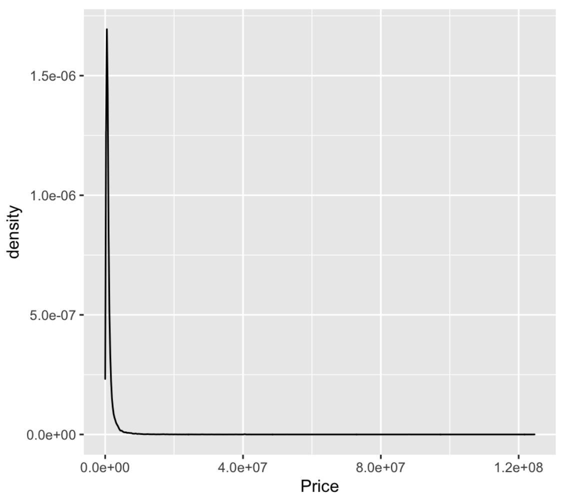
There are several practical methods for handling outliers such as winsorization and trimming. When extreme values and outliers are replaced with the maximum (for large value outliers) and minimum (for small value outliers) values of data points that are not outliers, the process is known as winsorization. When extreme values and outliers are simply removed from the dataset, it is known as trimming (also called truncation). We used the trimming method and multiple of 3 to handle data outliers. I prepared several datasets for analysis. Long_data is the dataset with outliers left in and neat_data uses the trimming method to remove outliers. (Appendix 3 & 4)

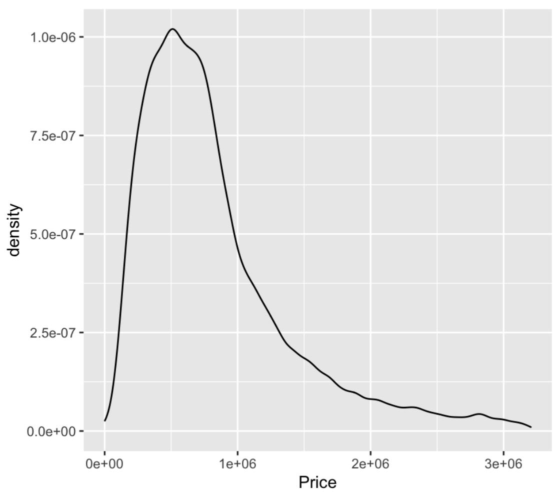
	Before	**After**	**Before**	**After**	**Before**	**After**	**Before**	**After**	**Before**	**After**
	Price (Target Variable)	Price (Target Variable)	Bath	Bath	Building size	Building size	lot size	Lot size	Bed	Bed
Min	1570	1570	0.5	1	202	202	100	100	0	0
1st Qu.	422414	429773	2	2	1184	1196	5976	5600	2	2
Median	684769	681346	2	2	1583	1567	7797	7153	3	3
Mean	1089864	823083	2.616	2.396	1970	1756	39396	12655	3.192	3.098
3rd Qu.	1118284	1033849	3	3	2232	2101	22216	13068	4	4
Max	124612663	3204326	54	6	32664	4748	41991840	69260	64	10

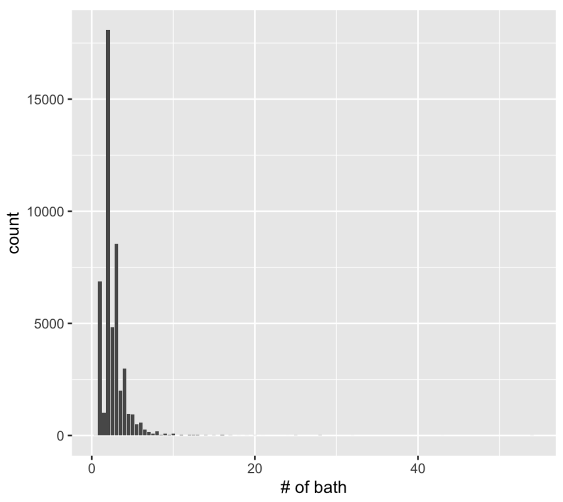
Table1. long_data and neat_data numerical variable demonstration, with and without outliers respectively.

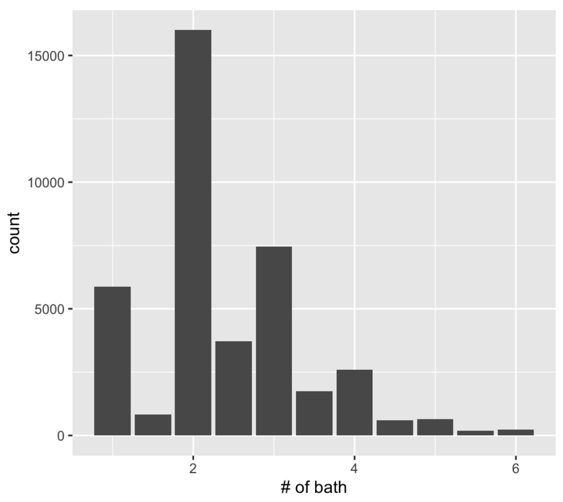
Before

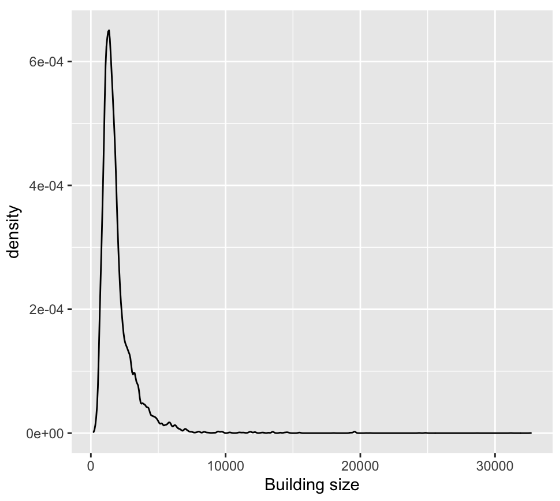
After

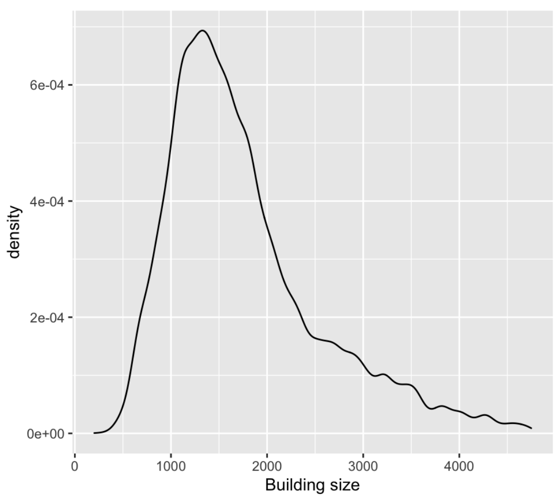












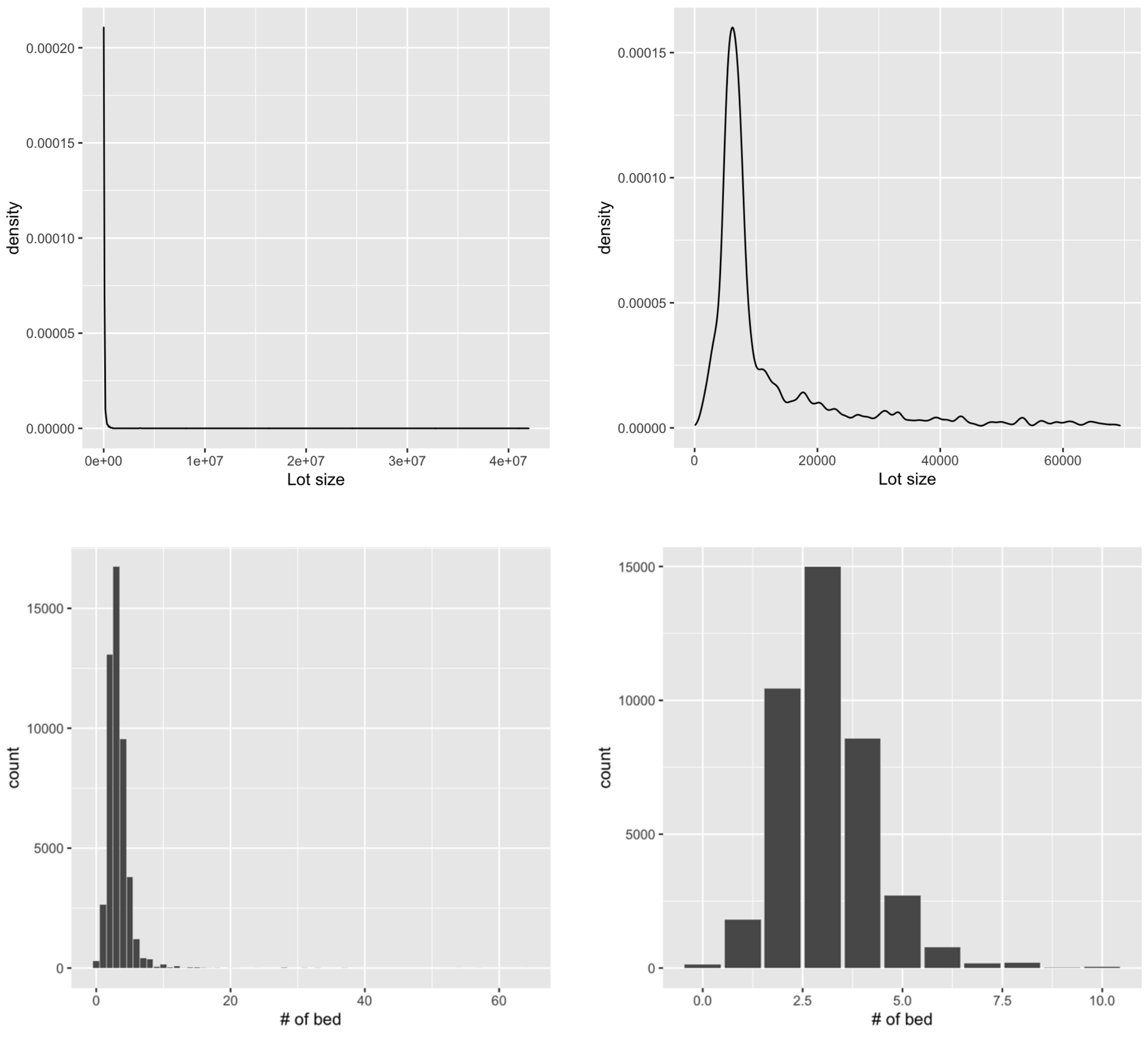


Figure1. Density plots of the numerical variables before and after trimming

DATA DESCRIPTION

In this project, we predicted the housing price in Los Angeles city from August 2021 to January 2022. To do that, We scraped the data from https://www.realtor.com to find information on sold properties. We extracted all data related to these properties, such as number of bedrooms, bathrooms, parking, lot size, building size etc. We obtained more than 8,500 observations from these three months. In addition, there is information related to the history transaction prices of property since it has been built. We combined these two datasets, and after cleaning and preprocessing data, about 40,000 observations and 14 variables were available for analysis. (Appendix 4)

As shown in the below figure, the left map indicates the distribution of properties in the Los Angeles area. Houses close to the beach and in the middle part of the city close to downtown have a relatively higher price than those properties farther away. It is worth mentioning again that all property prices are adjusted for inflation which is crucial to get relevant results for prediction.

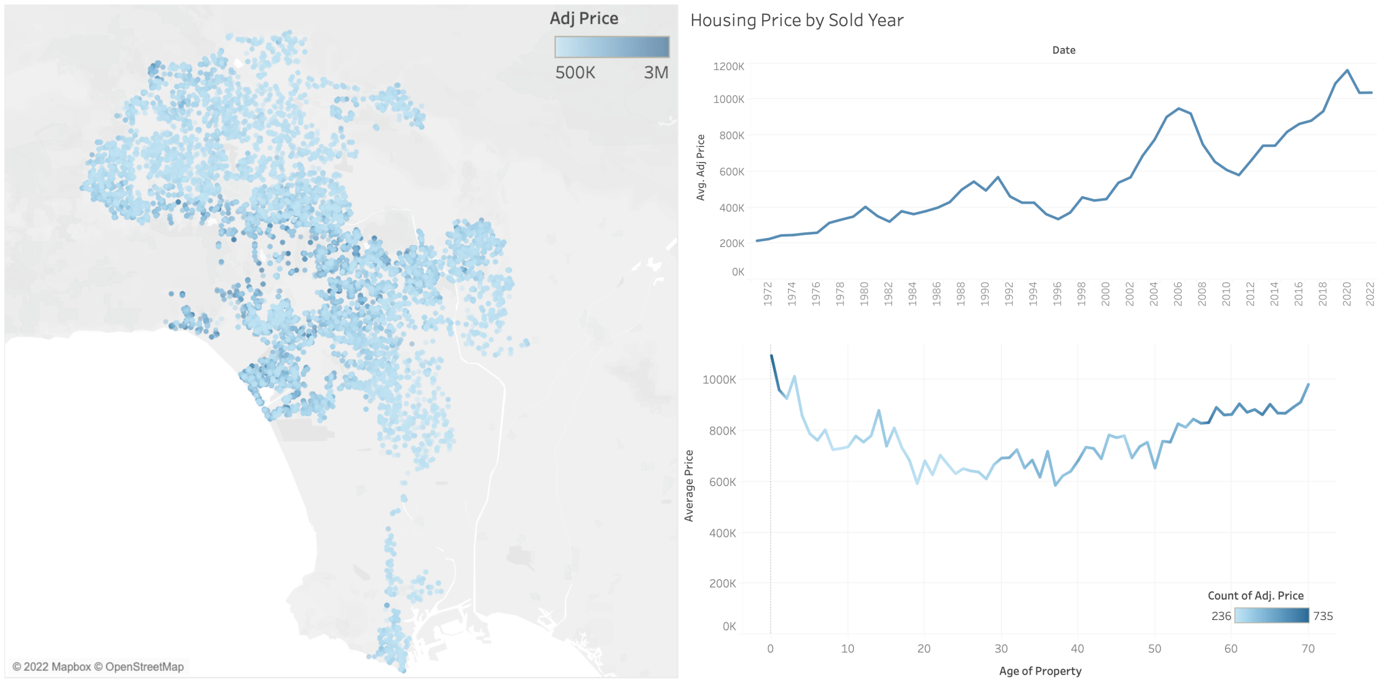


Figure 2. Left map shows the house price distribution in Los Angeles City. Upper right plot demonstrates the adjusted price by year. Bottom plot shows the relationship between average adjusted price by property age

In the upper right plot, the prices increase as time passes even by considering the effect of inflation. There are other factors than inflation that can raise the price of houses. Technology can play a critical part in this cause. (Kattan, n.d.). Facilities and amenities are employed in different parts of the house, which raise the price (Anas, et al., n.d.). Other causes can be named, such as demographics, interest rates, economy and government policies/subsidies which might affect the rising housing price during these years (NGUYEN, 2021). The last plot, which is on the lower right, shows the average prices are not dependent on the build year. However, a fascinating finding can be realized from this plot. On average, houses built in the 1970s had a relatively lower price than houses built in the 1920-1950s. A plausible cause is that the older properties are renovated which would affect the price.

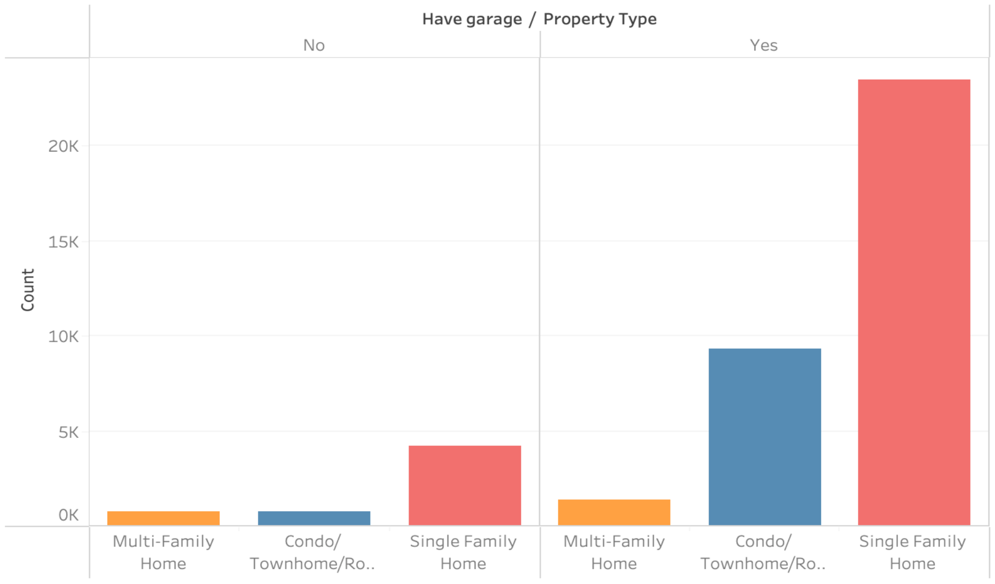
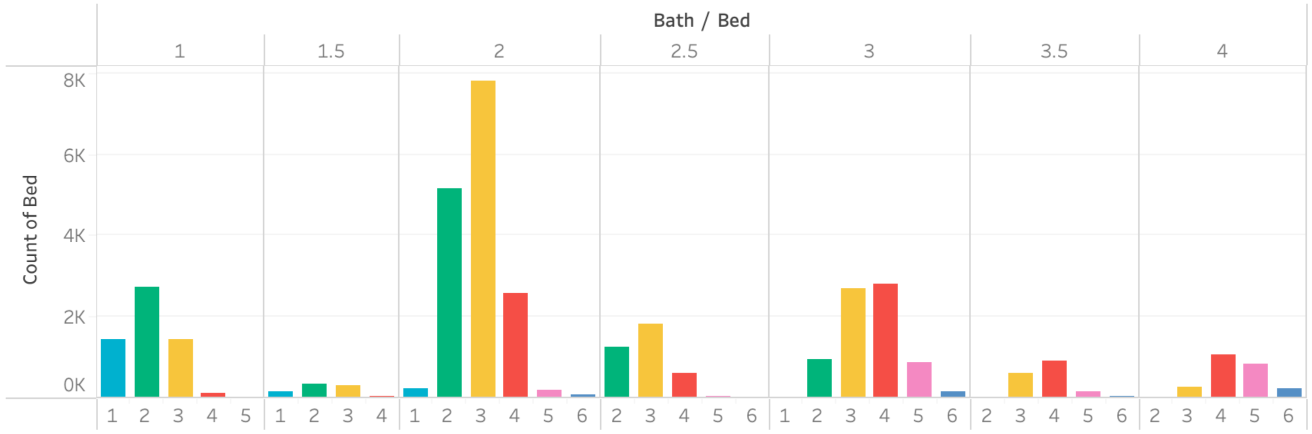


Figure 3. Type of the property and either having garage or not

The above figure compares the number of properties with, or without, a garage and what type those properties are. Most of the observations are Single Family Homes with garages, followed by Condo/Townhome/Co-Op with garages.



Count of Bed for each Bed broken down by Bath. Color shows details about Bed. The data is filtered on Bath and Bed. The Bath filter ranges from 1 to 4. The Bed filter ranges from 1 to 6.

Figure 4. The combination of the number of bathrooms and bedrooms

The above two figures show how the number of bathrooms and bedrooms spread in the dataset. Most of the houses are 3-bedroom and 2-bathroom.

METHODOLOGY

This study aims to predict not only the housing price but also the features which are useful for distinguishing the accurate prediction. An important caveat is that we are exploring the relationships between price and these other variables, and no claim is made about causality.

Our preprocessing procedure left us with about 55,000 observations and 23 variables. We called this dataset the main_data (see Table 2 and Appendix 1). Some variables in the main_data had missing values and some of them were not as useful to predict the model. To fully employ the whole dataset, variables with more than 7% missing values are removed from dataset which we named that long_data. Long_data had the greatest number of observations and fewer features (48,440 observations and 14 variables). Another dataset was designated to utilize the variables with more than 7% missing values which was called wide_data (Appendix 2). Wide_data had the most number of features and the lowest amount of observation (20,270 observations and 17 variables). Neat_data was extracted from long_data with extreme outliers removed. (39,852 observations and 14 variables).

55,625 observations	Variable Type	NAs	Percentages of NAs
lot_size	Numerical	836	1.50%
building_size	Numerical	1252	2.25%
age	Numerical	2325	4.18%
Renovated_age	Numerical	2325	4.18%
Bed	Numerical	3507	6.30%
No_Garage	Numerical	13805	24.81%
Have_garage	Nominal	363	0.65%
PropertyType	Nominal	792	1.42%
Heating	Nominal	20871	37.52%
Cooling	Nominal	26016	46.77%

Table 2. Counts and percentages of NAs in the main_data

To find which dataset would be the most informative one to pass through the model for comparison, we investigated the pros and cons for all of them. Most of the nominal variables in wide_data are imbalanced; for instance, only 0.2% of properties have solar panel systems (Appendix 2). Therefore, these variables were not as useful in predicting the target variable since they were not enough sample to estimate the contrast.

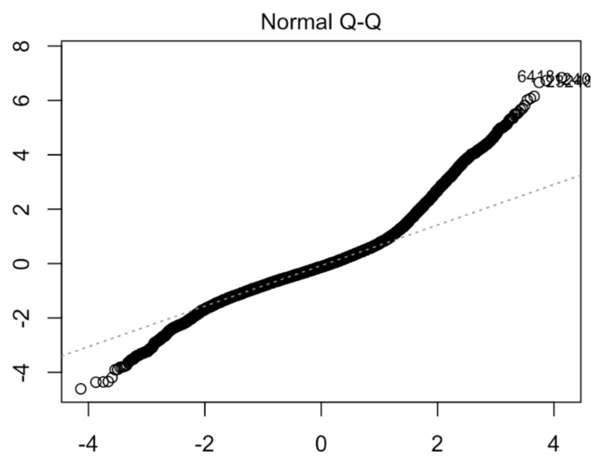
In this study, we employed different methods from classical linear regression to boosted regression trees. Regression models were not robust to outliers. To compare the model with one dataset, we decided to work with neat_data and applied it to the different models.

We compared the models' performances and errors to select the best ML method among them. Finally, we passed all three datasets to the best model and compared whether having more features or outliers contribute to a different prediction or not. To start applying the dataset in the models, we divided the data into two training and test datasets. Training data consists of observation from 1964/06/03 to 2021/08/31. Test data are from 2021/09/01 to 2022/01/26. For the models with hyperparameters, cross-validation (k = 6) of the training dataset has been used to tune the hyperparameter with the lowest MAE.

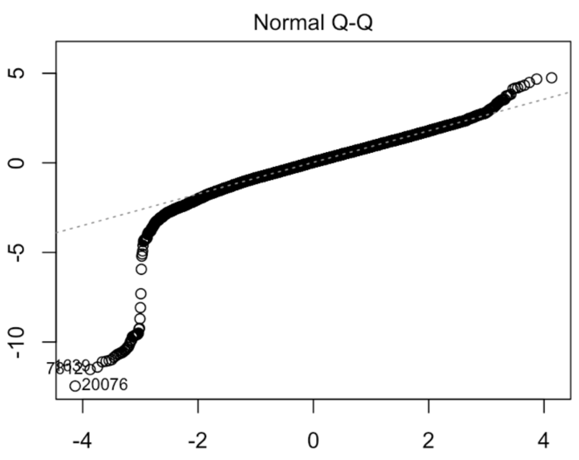
– ***Multiple Linear Regression***

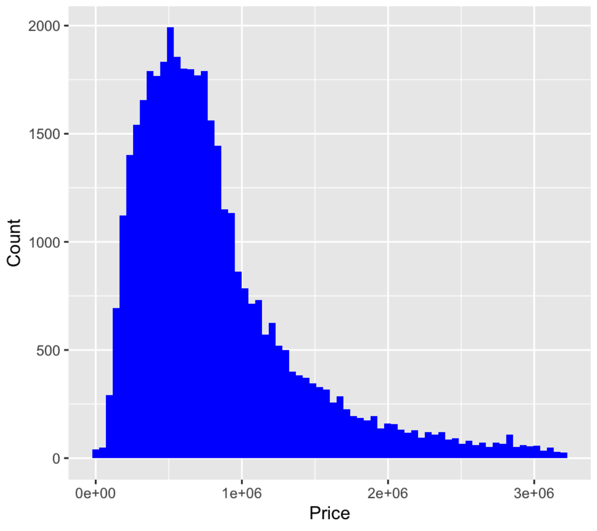
Multiple linear regression with the Ordinary Least Square (OLS) estimator is the simplest model implemented in this study. Real estate asset pricing is a complex high-dimensional problem due to a large number of property characteristics, nonlinearities, and interaction effects. (For instance, any change in construction material can alter the housing price drastically). Due to the nonlinear nature of housing pricing with its characteristics, some assumptions of multiple linear regression have been violated; in particular the assumption of homoscedasticity and normality of the residuals. In addition, due to historical prices on the properties, the house prices are also going to be correlated, i.e., the independence assumption is also violated. Moreover, our data were extracted from 1964 to 2022. So, we have the challenge of working with the time series variable. To have a valid result for multiple linear regression, the residual errors should be uncorrelated. In addition, price as the target variable has a right-skewed distribution. For most cases, utilizing log transformation can make the distribution more normal. Figure 5 shows the Normal Q-Q plot and histogram of the price before and after log-transformation.

Before Log-Transformation



After Log Transformation





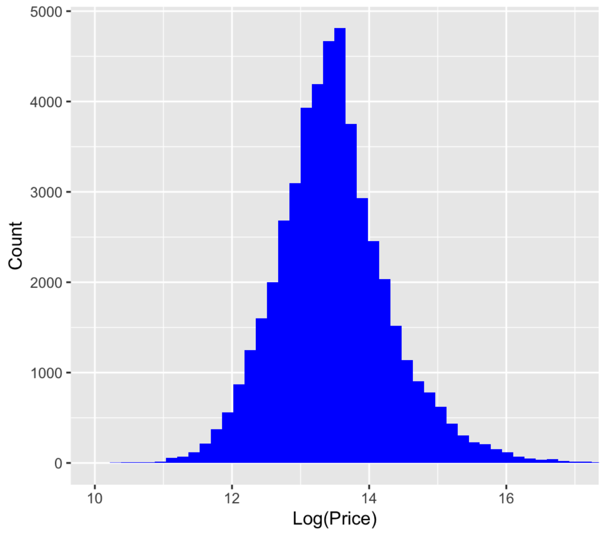


Figure 5. Normal Q-Q and histogram plots before and after the Log-transformation

– ***Regularized Linear Regression***

The only goal of the prediction problem is to maximize prediction performance, so they do not require the unbiasedness of variable coefficients. Since OLS is the best linear unbiased estimator (BLUE), a way to improve prediction performance is to introduce bias. Regularized/Penalized linear methods can introduce bias to improve OLS prediction performance (Hastie, et al., 2009). Penalized regression is a computationally efficient technique used in prediction problems.

Practically, penalized regression's duty is to reduce several features to a controllable set and make reasonable predictions in various large datasets, particularly where features are correlated (i.e., when classical linear regression works poorly

A model that fits the training data well is not necessarily a good thing. It shows that the model may closely reflect the characteristics of the specific training data, which will not perform well on new data. This problem occurs when the data becomes overfitted, and regularized linear regression can ultimately remove irrelevant variables.

As expressed by the quote below by (Anon., 2019), "In prediction, out-of-sample performance is critical, so relatively parsimonious models (that is, models in which each variable plays an essential role) tend to work well because they are less subject to overfitting"

There are three Regularized linear methods used in this study, Ridge, LASSO and Elastic Net Regression. Ridge regression uses a loss function to minimize the sum of squared residuals and penalize the sum of squared coefficients for shrinking them towards zero. LASSO regression works the same, but instead of penalizing the sum of squared coefficients, it regularizes the sum of absolute values of the coefficients. Elastic Net regression is a convex combination of Ridge and Lasso (Zou & Hastie, 2005). Here are the differences in penalty terms between these methods.

Multiple Linear Regression:

$$min\ \sum_{i=1}^{n}(\mathrm{Y}_i - \hat{Y}_i)^2$$

Ridge Regression:

$$min\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda\sum_{k=i}^{K}\hat{b}_k^2$$

LASSO Regression:

$$min\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda\sum_{k=i}^{K}|\hat{b}_k|$$

Elastic Net Regression:

$$min\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda(\alpha\sum_{k=i}^{K}|\hat{b}_k| + (1-\alpha)\sum_{k=i}^{K}\hat{b}_k^2)$$

Lambda (λ) and alpha (α) are hyperparameters—parameters whose value must be set by the researcher before learning begins—of the regression model and will determine the balance between fitting the model versus keeping the model parsimonious. Tuning is a method to repeatedly set the hyperparameters by reviewing model performance at different validation sets. Hence, the test set is also essential to avoid overfitting hyperparameters to the validation data.

There is a slight difference between LASSO and ridge regression models. The penalty term in LASSO makes non-relevant coefficients exactly zero. Thus, LASSO is often used for variable selection and pure prediction problems. Therefore, it is more powerful when interpretation is necessary. In contrast, ridge regression uses a penalty term that does not drive coefficients to precisely zero and is, therefore, less interpretable. However, ridge regression often provides superior prediction performance compared to LASSO. Elastic net regression incorporates LASSO and ridge strengths by selecting penalty term as a linear combination of both methods. λ and α are the hyperparameters for elastic net regression, which the researcher can tune to get the best prediction performance out of penalized regression models.

– ***Decision Tree***

Because of the interaction between variables and nonlinearity, tree-based models are better options for predicting real estate asset pricing. A regression tree model maps each characteristic to an expected value like a linear function. The prediction function has a decision-making role and splits the data into two nodes. At each node, the value of a single variable (say, the number value of building size) determines whether the left (less than 2000) or the right (more than 2000) child node is considered the next node (Mullainathan & Spiess, 2017). Various variables with different thresholds can be set in the first node. Subsequent nodes can be built by picking the suitable variable and threshold; this process continues for the lower level of trees. Among all combinations of a tree, the best decision tree will be selected based on the minimum sum of the squared residual at the terminal node (see Figure 6). A prediction is returned when a terminal node—a leaf—is reached.

Several hyperparameters are defined to provide the model's best performance, such as tree depth, the minimum number of observations in each node, and the complexity parameter. Tuning these parameters can prevent the model from overfitting.

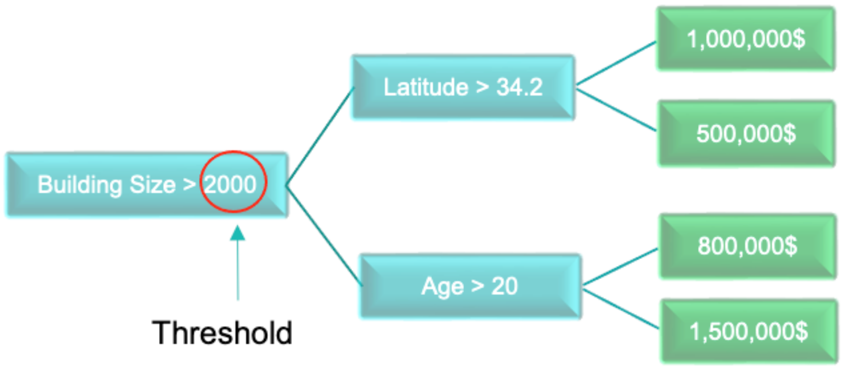


Figure 6. Demonstration of decision tree (regression tree)

Decision trees are intuitively straightforward to explain. They follow the human decision-making process compared to other regression and classification methods. You can present them graphically, and they can easily handle qualitative predictors without the need to create dummy variables. However, decision trees are generally not accurate in prediction since they aren't quite robust. A minor change in the data can cause a substantial difference in the final decision tree, which will not get the best result for a new dataset (Kotsiantis, 2011).

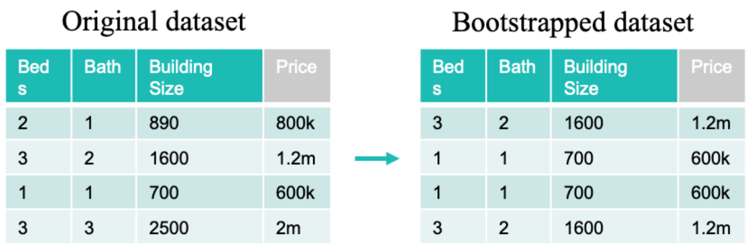
– ***Bagging and Random Forest***

Decision trees tend to suffer from high variance when they are applied to new data. This means that if we cross validate[3] the training data into k-folds (k could be between 5 to 10 for optimal result) at random and fit a decision tree to each fold, the performance result of each set could be entirely different. On the contrary, a procedure with low variance will produce similar performance results if applied repeatedly to distinct datasets. We can

[3] "Cross-validation is a resampling technique that uses evenly percentages of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice." (Allen, 2012)

aggregate numerous decision trees using methods like bagging, random forests, and boosting to improve decision trees' predictive performance substantially.

Bagging, or bootstrap aggregation, uses the combination of different sub-samples from the same dataset to reduce the variance of the predictions. Bagging has been determined to improve accuracy and mitigate overfitting problems by combining hundreds or even thousands of trees into a single procedure (Breiman, 2001).



Original dataset

Beds	Bath	Building Size	Price
2	1	890	800k
3	2	1600	1.2m
1	1	700	600k
3	3	2500	2m

Bootstrapped dataset

Beds	Bath	Building Size	Price
3	2	1600	1.2m
1	1	700	600k
1	1	700	600k
3	2	1600	1.2m

Figure 7. Creating bootstrapped dataset from original dataset

Figure 8. Bunch of decision trees to create Random Forest

Bagging creates several copies of the original training dataset using the bootstrap. Then it fits a particular decision tree to each replica and combines all the trees to create a distinct predictive model. Each tree is established from a bootstrapped dataset, not dependent on the other trees. In bagging, we pick randomly from all predictor variables when splitting a new node happens. Whereas, in Random Forest, there is a minor tweak in the number of predictor variables we can choose. In Random Forest, we choose a number between two and the total number of predictor variables randomly. For instance, we want to create a node, and the dataset has ten features to pick for the node to split the data. In bagging, we randomly select a variable from 10 features; however, we choose randomly from the k features ($2 < k < 10$) in a random forest. This change would make the procedure less dependent on the characteristics of training data. In addition, data with a large proportion of missing values can be estimated. Random Forest also can adequately balance the errors in case of imbalanced target variables. The most prominent benefit of using Random Forest is that it can work reasonably well with large datasets with many columns. However, using Random Forest has some disadvantages, such as less interpretability than the decision tree, and nonstructural data cannot be used with this model.

– ***Gradient Boosting and XGBoost***

Like bagging and random forests, gradient boosting is a common approach to solving regression and classification problems. Gradient boosting works the same as bagging, except each tree grows based on the error made by the previous tree. Each tree is called a weak learner and is produced using information from previously grown trees. Boosting does not use a bootstrapped dataset before creating a weak learner. Each tree is built from an improved version of the original dataset. Boosting works well on a large dataset data and can be huge and intricate. This method has been used to solve many challenging regression and classification problems, including risk analysis (Tian, et al., 2020), sentiment analysis (Hew, et al., 2020), predictive advertising, price modelling, and sales estimation (Wisesa, et al., 2020). For example, Credit Risk Analysis (CRA) is a vital task in banking. Usually, obtaining a loan is necessary for a corporate or individual to maintain the business. So, banks and financial organizations perform credit analyses for their customers. With the help of XGBoost risk analysis this can be accomplished promptly with high accuracy.

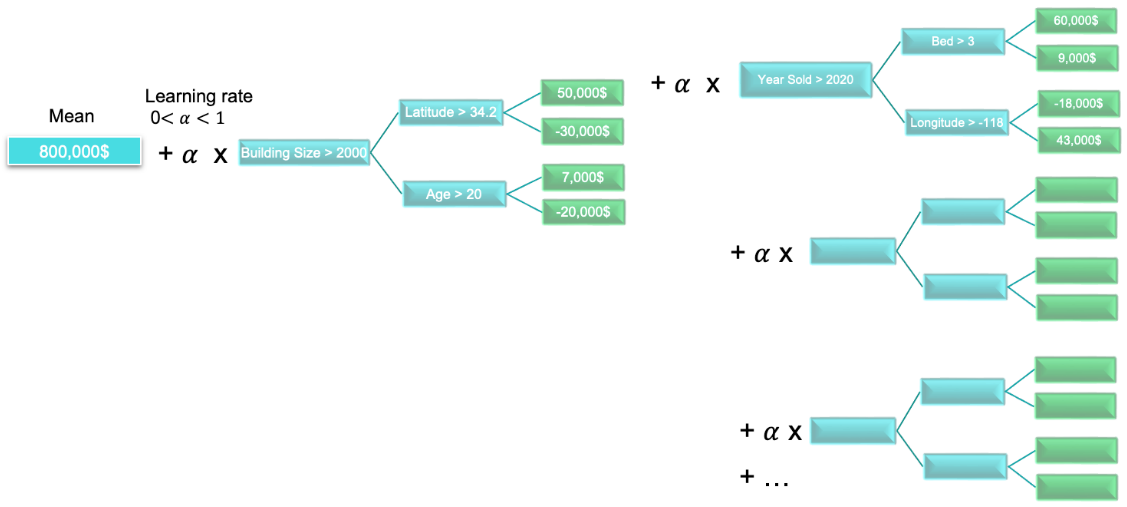


Figure 9. XGBoost Basic Structure

Figure 9 shows the structure of Gradient Boosting with a bunch of weak learners. Weak learners will be built by putting more weight on trees with extreme errors and inaccurate predictions. The idea behind this technique is that the model learns from past mistakes. For instance, weak learners cannot accurately predict the observations that are difficult for the model, so those will be weighted more during learning. Therefore, gradient boosting does not create the next model from scratch, and this approach makes this technique superior in working with the new data.

Gradient boosting usually outperforms Random Forest, but Random Forest is easier to implement. In Random Forest, the only tuning parameters are the number of trees and the number of predictors for a node. In contrast, in boosting, more tuning parameters are required besides the number of trees, including the shrinkage and the interaction depth.

XGBoost stands for Extreme Gradient Boosting, which uses more accurate approximations to find the best tree-based model. In gradient boosting, when we want to find a threshold for each variable assigned to a node, we employ all the observations to decide on an acceptable threshold. Conversely, in XGBoost, we use an Approximate Greedy Algorithm. Instead of testing every single observation, this algorithm divides the data into quantiles. It only uses the quantiles as nominee thresholds to split the nodes, which would make the best threshold very fast. To find the quantile in a sizable dataset, XGBoost uses the Weighted Quantile Sketch Algorithm, which proportionately splits the data into several datasets based on the weight. The weight for each observation only matters in the classification problem, which is the second-order gradient of the loss function called the Hessian. Each core in the computer can handle the task for each dataset. Then all cores work together in parallel and ultimately combine the values to make an approximate result. XGBoost also is an excellent method when there is missing data. Some other optimization techniques in XGBoost take the computer hardware into account. Therefore, this method is not just an applied statistical technique (Chen & Guestrin, 2016).

- ***Tuning***

As discussed in the models, some models need to be tuned to get the best performance result. Hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. We must cross-validate the training data beforehand to find the best hyperparameter for applying new data to the model. Using cross-validation gives us a set of MAE to configure which hyperparameter has the lowest MAE value. There are different methods to grid the hyperparameter depending on the number to be estimated. In this project, I used two approaches to find suitable hyperparameters. First, randomly grid the hyperparameter to discover the best performing hyperparameters. Then, we used the combination of chosen hyperparameters to find which combination achieved the lowest MAE. The figure below shows the optimization of the XGBoost hyperparameter. Learning rate = 0.06, Proportion Observation Sampled = 0.9, Tree Depth = 10 had the lowest MAE which was less than 0.15.

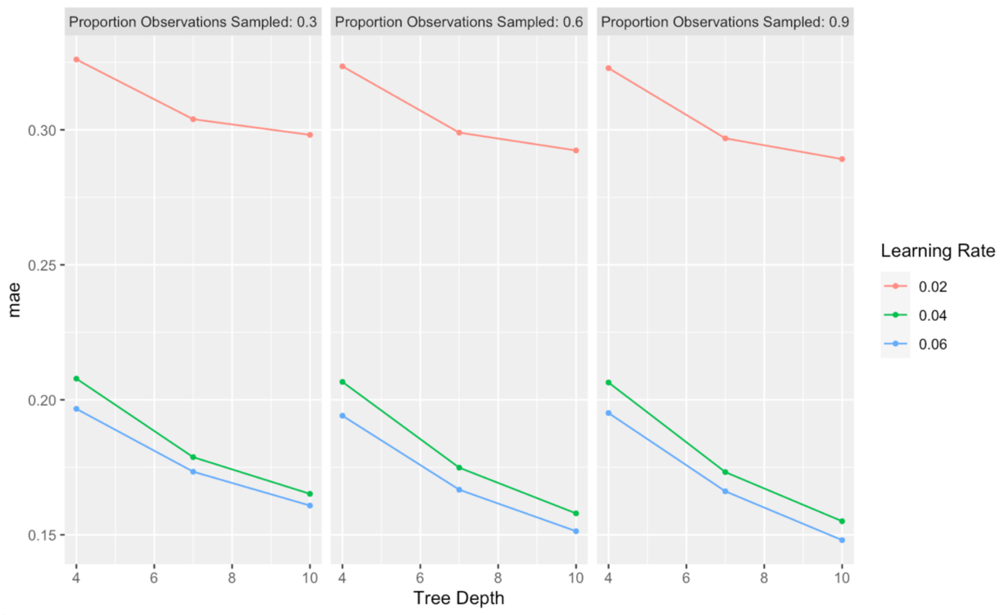


Figure 10. The result of MAE from the tuning of three hyperparameters in the XGBoost model

DISCUSSION OF FINDINGS

Figure 11 depicts the two models' pricing accuracy. On average, the price predictions from the ML approach like XGBoost are much closer to the actual prices than the OLS estimates like linear regression. At higher prices, the difference between these two methods is more distinguishable; however, some observations in the higher price do not suit both models well.

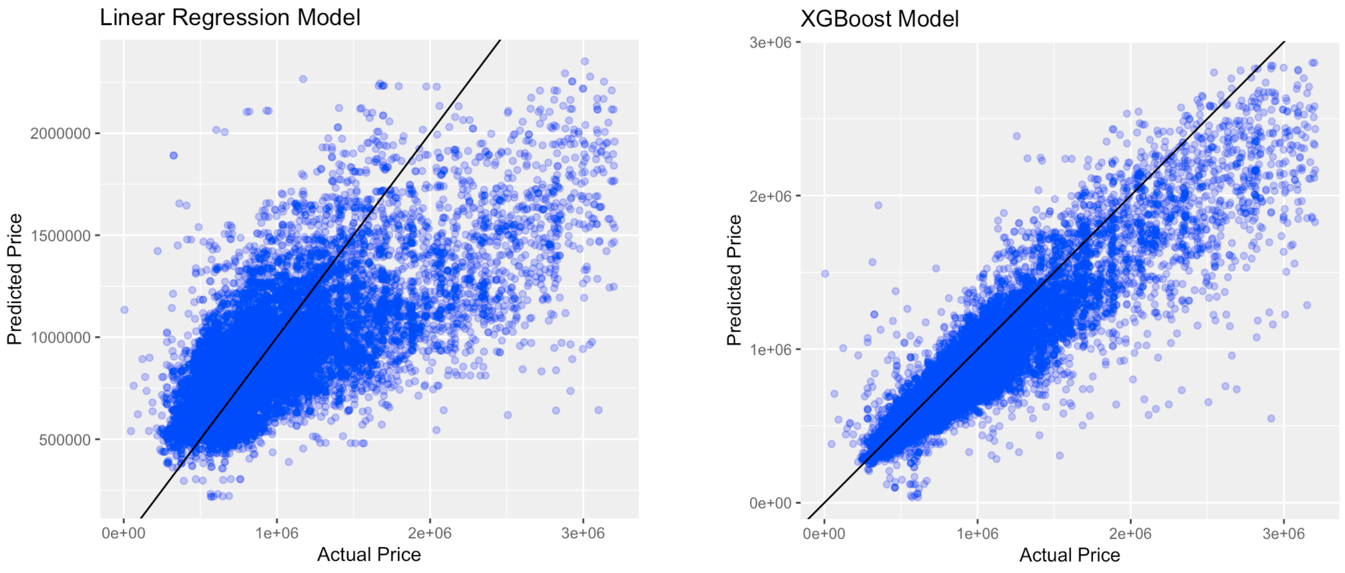


Figure 11. Comparison of pricing accuracy between Linear Regression and XGBoost

All errors are wrong, but not all are equally bad. Sometimes significant prediction errors are disproportionately more harmful than minor errors. RMSE and MAE are both metrics to measure the error in the models. RMSE squares large errors, punishing extreme outliers more harshly than more minor ones. However, MAE treats every error the same as the formulas below.

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} i\text{th squared\_diff}}$$

$$MAE = \frac{1}{n} . \sum_{i=1}^{n} i\text{th absolute\_diff}$$

Table 3 indicates that every ML method outperformed the basic OLS method in all performance metrics. Regularized linear regressions and classic regression have the same result since the best performing hyperparameters are zeros in the penalized regression. XGBoost got a slightly better result than Random Forest with an R^2 value of 81.3% and 0.145 for the estimated MAE. Our results strongly indicate that the nonlinearities and interaction effects captured by more complex ML methods contain relevant information for real estate pricing. MAPE is a meaningful measure to interpret the economic magnitude of our findings. MAPE in XGBoost is almost half of the OLS estimate's MAPE with a value of 14.4%. This result can be concluded from MAE as well.

The improvement in pricing error in Random Forest and XGBoost is not only statistically significant but also economically large. Table 4 shows the performance result in a case when actual price is a target variable. To quantify the result in USD, mean absolute error is a good metric. MAE in XGBoost is 149,000 USD which is the half the amount of MAE in linear regression estimate.

Model	R^2	MAE	RMSE	MAPE (mean = 13.7)
Linear Regression	0.495	0.281	0.370	2.05%
Lass Regression	0.495	0.281	0.370	2.05%
Ridge Regression	0.495	0.281	0.370	2.05%
Elastic Net Regression	0.495	0.281	0.370	2.05%
Decision Tree	0.702	0.191	0.284	1.39%
Random Forest	0.803	0.159	0.235	1.16%
XGBoost Tree	0.813	0.145	0.221	1.06%

Table 3: Results of R^2, MAE, RMSE and MAPE for five different models, ln(price) as the target variable[4]

Result of accuracy and performance of the neat data on different models with the actual price.

Model	R^2	MAE	RMSE	MAPE (mean = 1,034,263)
Linear Regression	0.460	292,694	442,868	28.30%
Lass Regression	0.460	292,694	442,868	28.30%
Ridge Regression	0.460	292,694	442,868	28.30%
Elastic Net Regression	0.460	292,694	442,868	28.30%
Decision Tree	0.743	190,685	293,852	18.44%
Random Forest	0.826	164,067	260,980	15.87%
XGBoost Tree	0.840	148,992	235,281	14.41%

Table 4: Results of R^2, MAE, RMSE and MAPE for five different models, price as target variable

Table 5 shows the performance result of three different datasets on the XGBoost model. RMSE and MAE become larger as the outliers exist in the dataset. Moreover, R Squared becomes bigger with the existence of outliers and more features.

Model	R^2	MAE	RMSE
Neat_data	0.813	0.145	0.221
Long_data	0.847	0.167	0.270
Wide_data	0.853	0.174	0.289

Table 5: Results of R^2, MAE and RMSE modelled by XGBoost with three different datasets

Figure 11 demonstrates the variable importance in the XGBoost model. The size of the building is a clear feature in predicting house prices. It is interesting to see that the second important variable is the year of the sold property, which plays a vital role in price fluctuation over time. Moreover, as it is observable from the

[4] The best tuning parameter for Lasso, Ridge and Elastic Net Regression is the penalty = 0, making all the regression performance results the same.

importance of latitude and longitude variables, knowing the property location can be a piece of critical information in price prediction.

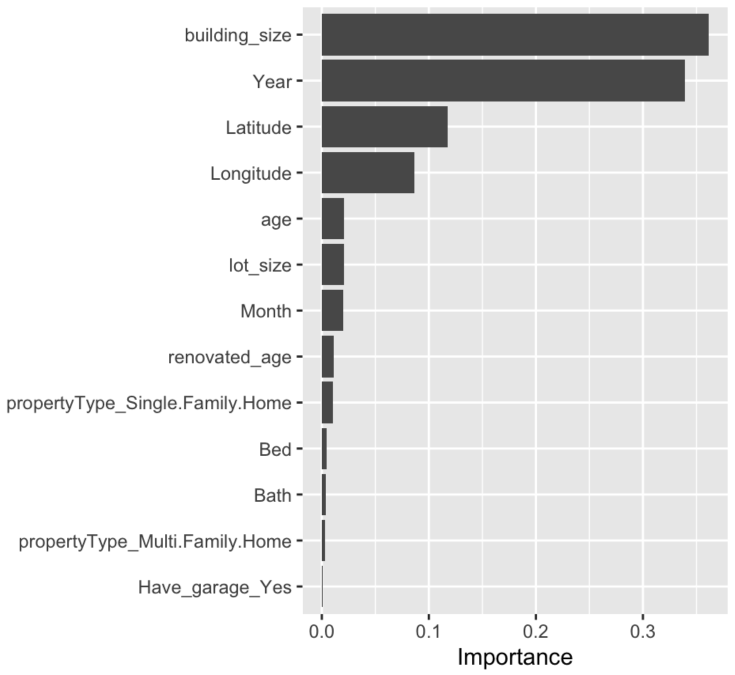


Figure 12. The variable importance in XGBoost Regression Tree Model

CONCLUSION

We started scraping data from an online source to extract reliable information related to the properties in Los Angeles city.

After cleaning and preprocessing the data, we applied five models to this data and tested the models to compare them in terms of performance accuracy and interpretability. We needed to train the models with historical prices, meaning real estate prices before September 2021. Therefore, our models were not acquainted with the data after September 2021. While simpler models like penalized regressions and decision trees already performed superior to the classic regression method, more complex models like Random Forest and XGBoost yielded much lower pricing errors. The mean absolute error percentage was 28% for linear regression and about a half that for XGBoost. In monetary units, the enhanced pricing accuracy corresponds to a reduction in pricing error by approximately 208,000 USD for an average property price of 1,034,000 EUR.

We conclude that interaction effects and nonlinearities captured by complex ML methods influence real estate pricing. Most of the data were extracted from October 2021 to February 2022, which can be a bias factor for considering month as a predictor variable in our models.

Future research might also consider GDP as a predictor variable, which could provide additional information relevant to real estate prices but is not yet included in our dataset. However, all prices were adjusted by inflation based on CPI in the United States.

According to the regression tree model, the essential feature was building size, which indicates a reality in the housing market.

BIBLIOGRAPHY

Allen, D. M., 2012. The Relationship Between Variable Selection and Data Agumentation and a Method for Prediction, Technometrics. 9 April, Volume 16:1, pp. 125-127.

Anas, Alex, Arnott & Richard J., n.d. Technological Progress in a Model of the Housing – Land Cycle. Elsevier.

Anon., 2019. CFA Program Curriculum Level I. s.l.:CFA Institute.

Breiman, L., 2001. Random forests. Machine learning 45, Volume 1, pp. 5-32.

Chen, T. & Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining , pp. 785-794.

Hastie, T., Friedman, J. & Tibshirani, R., 2009. The Elements of Statistical Learning. s.l.:s.n.

Hew, K. F., Hu, X., Qiao, C. & Tang, Y., 2020. What predicts student satisfaction with MOOCs: A gradient boosting trees supervised machine learning and sentiment analysis approach. Computers & Education, Volume 145.

Kattan, D., n.d. Forbes. [Online]
Available at: https://www.forbes.com/sites/forbesrealestatecouncil/2020/12/09/the-impact-of-technology-on-real-estate/?sh=8deaa4724341

Kotsiantis, S., 2011. Decision trees: a recent overview. Artificial Intelligence Review, 29 June.pp. 261-283.

Mullainathan, S. & Spiess, J., 2017. Machine Learning: An Applied Econometric Approach.. Journal of Economic Perspectives.

NGUYEN, J., 2021. Investopedia. [Online]
Available at: https://www.investopedia.com/articles/mortages-real-estate/11/factors-affecting-real-estate-market.asp

Tian, Z., Xiao, J., Feng, H. & Wei, Y., 2020. Credit Risk Assessment based on Gradient Boosting Decision Tree. Procedia Computer Science, 174(1877-0509), pp. 150-160.

Wickham, H., 2021. rvest package. s.l.:s.n.

Wisesa, O., Adriansyah, A. & Khalaf, O. I., 2020. Prediction Analysis Sales for Corporate Services Telecommunications Company using Gradient Boost Algorithm. Wireless Sensors and Powering (BCWSP), pp. 101-106.

Zou, H. & Hastie, T., 2005. Regularization and variable selection via the elastic net.

APPENDIX 1

	Observations	Variables	Target Variables	ID Variables	Predictor Variables	Nominal Predictor Variables	Numerical Predictor Variables
Main_data	55,625	23	1 (Price)	2 (Date & Address)	20	5	15
Wide_data	20,270	17	1 (Price)	1 (Date)	15	4	11
Long_data	48,480	14	1 (Price)	1 (Date)	12	2	10
Neat_data	39,852	14	1 (Price)	1 (Date)	12	2	10

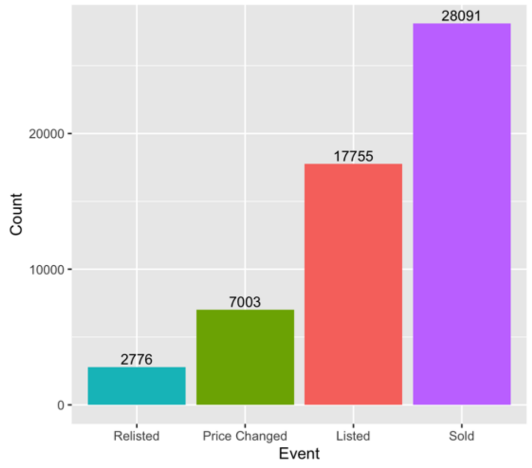
ID variables are just descriptive variables and do not have a prediction role for the models.

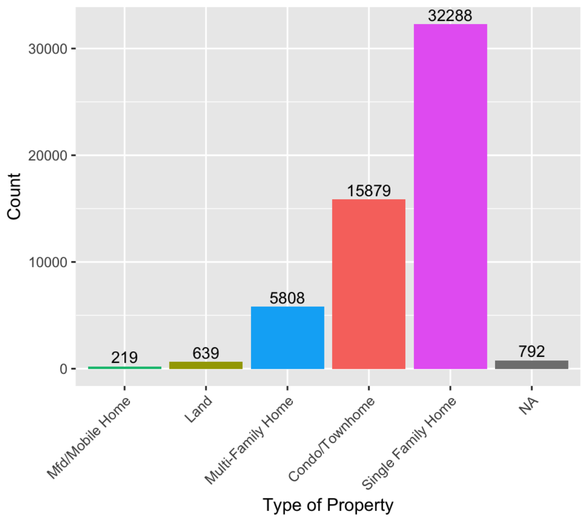
Main_data:

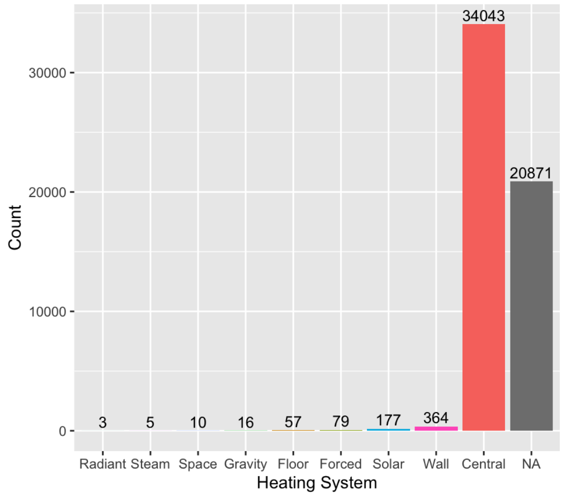
Numerical Variables:

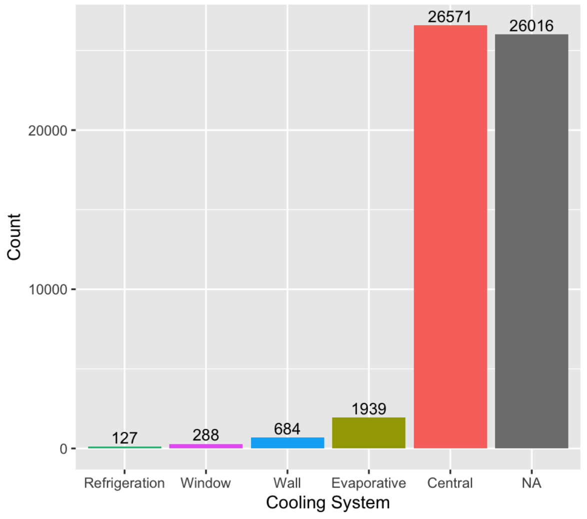
	Min	1st Qu.	Median	Mean	3rd Qu.	Max	NA's
Price (Target Variable)	1500	335000	638000	1076229	1095000	133000000	-
Latitude	33.71	34.04	34.10	34.10	34.18	34.64	-
Longitude	-118.7	-118.5	-118.4	-118.4	-118.3	-118.0	-
Year	1959	2007	2016	2012	2021	2022	-
Month	1	5	8	7.449	11	12	-
adj_factor	0.1135	0.7851	0.9315	0.8775	1.0587	1.0752	-
adj_price	1,570	416,009	684,459	1,143,177	1,128,831	124,612,663	-
Price_per_Sqft	0.1	206.6	391.6	475.2	597.4	7,379.5	-
adj_price_sqft	0.1	264.3	421.5	507.5	605.5	6914.1	-
Bath	0.5	2	2	2.86	3	114	3849
building_size	60	1214	1636	2233	2420	298782	1252
lot_size	100	5841	7535	42187	19602	41991840	836
Bed	0	2	3	3.376	4	99	3507
No_Garage	0	1	2	1.593	2	26	13805
age	0	29	52	51.55	73	198	2325
Renovated_age	0	25	44	46.2	65	198	2325

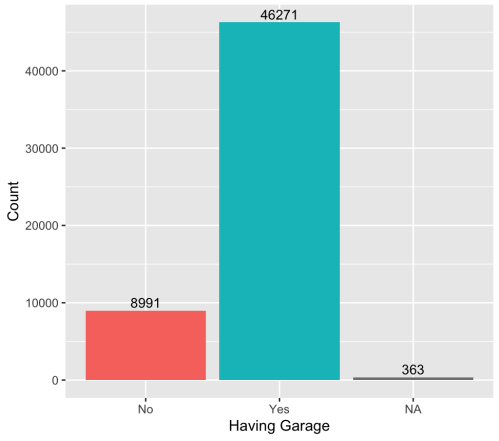
Nominal Variables:











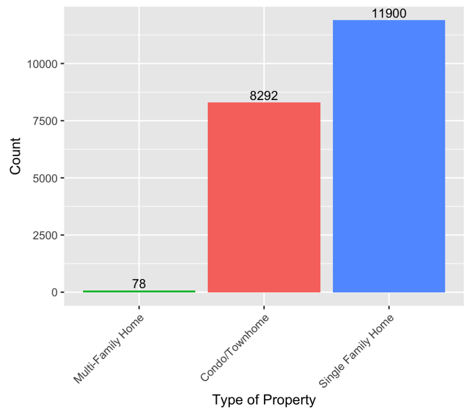
APPENDIX 2:

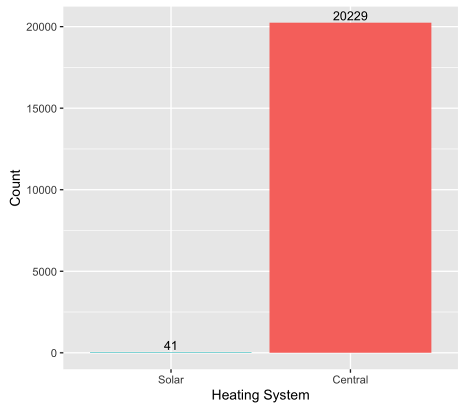
Wide Data

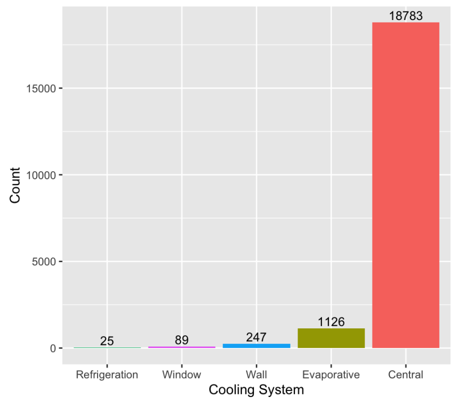
Numerical Variables:

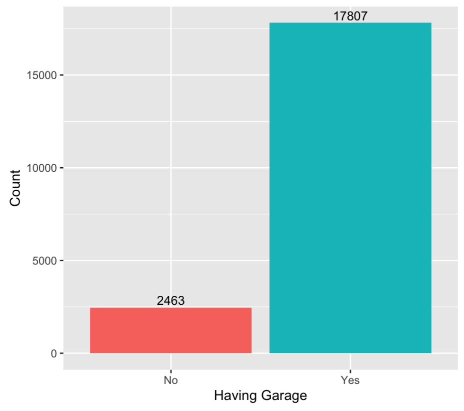
	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Adj_price (Target Variable)	1809	436402	706600	1242403	1144183	124612663
Latitude	33.72	34.09	34.17	34.15	34.22	34.64
Longitude	-118.7	-118.5	-118.5	-118.5	-118.3	-118.2
Year	1959	2007	2016	2012	2021	2022
Month	1	5	8	7.432	11	12
Bath	0.5	2	2.5	2.846	3	16
building_size(Sqft)	202	1300	1746	2187	2562	24837
lot_size(Sqft)	123	7325	12632	54782	38333	3588908
Bed	0	2	3	3.228	4	12
No_Garage	0	2	2	1.8	2	26
age	0	17	35	36.89	55	198
Renovated_age	0	15	32	31.89	46	198

Nominal Variables:









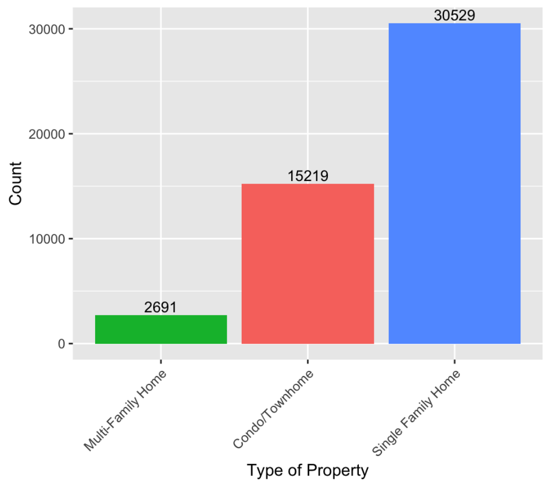
APPENDIX 3:

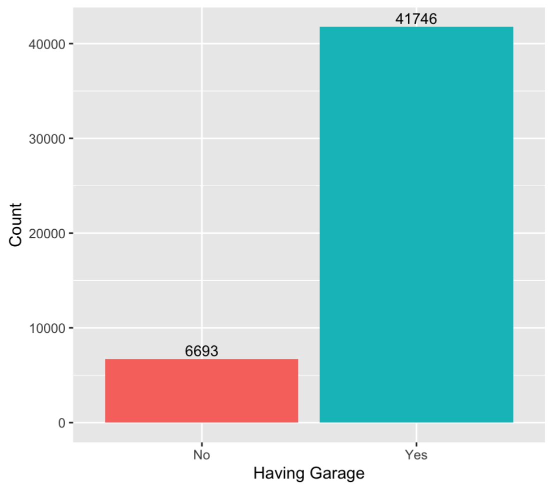
Long Data

Numerical Variables:

	Min	1st Qu.	Median	Mean	3^{rd} Qu.	Max
Adj_price (Target Variable)	1570	422410	684758	1089864	1118257	124612663
Latitude	33.71	34.04	34.10	34.10	34.19	34.64
Longitude	-118.7	-118.5	-118.4	-118.4	-118.3	-118.1
Year	1959	2007	2016	2012	2021	2022
Month	1	5	8	7.479	11	12
Bath	0.5	2	2	2.616	3	54
building_size(Sqft)	202	1184	1583	1970	2232	32664
lot_size(Sqft)	100	5976	7797	39397	22216	41991840
Bed	0	2	3	3.192	4	64
age	0	28	51	50.47	71	198
Renovated_age	0	24	43	44.86	64	198

Nominal Variables:





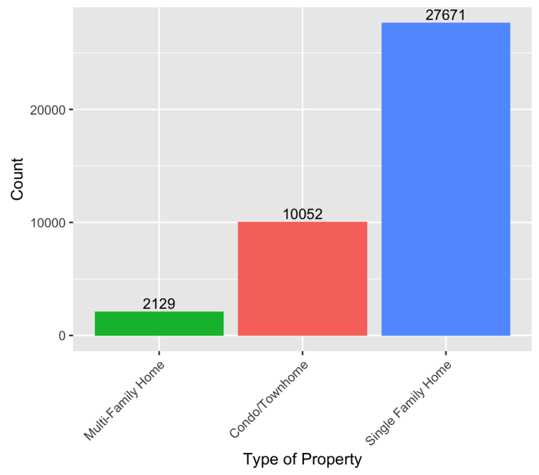
APPENDIX 4

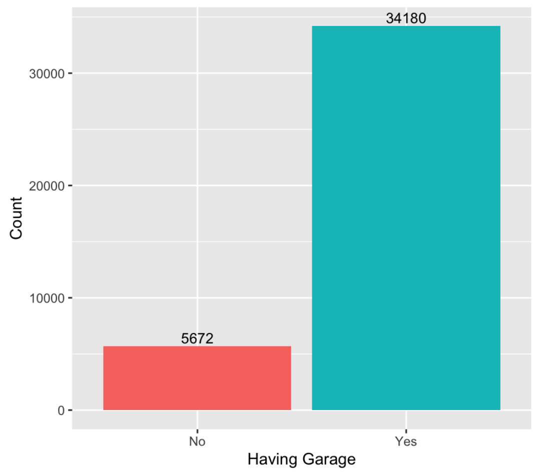
Neat Data

Numerical Variables:

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Adj_price (Target Variable)	1570	429773	681346	823083	1033849	3204326
Latitude	33.71	34.04	34.11	34.10	34.19	34.33
Longitude	-118.7	-118.5	-118.4	-118.4	-118.3	-118.1
Year	1962	2007	2016	2012	2021	2022
Month	1	5	8	7.509	11	12
Bath	1	2	2	2.396	3	6
building_size(Sqft)	202	1196	1567	1756	2101	4748
lot_size(Sqft)	100	5600	7153	12655	13068	69260
Bed	0	2	3	3.098	4	10
age	0	31	56	53.7	74	172
Renovated_age	0	27	47	47.68	66	172

Nominal Variables:





APPENDIX 5

Reflection

During this project, I have learnt how machine learning techniques such as decision tree, random forest and XGBoost work with the continuous target variable. In detail, I came up with how to restore HTML variables in R. How to scrape information from webpages without API and use VPN to change the IP addresses to bypass IP restrictions on the websites. Extracting geocodes from US addresses via Google Sheets was another exciting assignment. Doing these tasks also came along with different challenges that took several days to fix. There were other challenges in learning new ML techniques; however, I put enough time and effort into resolving those issues.

YouTube was an excellent resource for teaching me all the steps to accomplish this project. As an example, I learnt decision tree, Random Forest and XGboost through a YouTube channel called "StatQuest with Josh Starmer."

My instructor, Prof. Riegert, was supportive in all aspects of the project, from providing valuable resources and ideas to commenting and reviewing the presentation and report. His method was not to force students to complete each step on a tight deadline which I found fascinating. Also, he was available for weekly meetings in a very flexible timeframe.

In the first term, "AMOD 5210-Foundations of Modelling" course discussed our potential project and how it could be done. However, it is needed to be recalled again, in my opinion. If there are some courses for students to learn about the prerequisite material of their potential project the term before the final term, time and energy will be saved for that project.

I have coded my project with R, and I believe my R skill level increased from intermediate to professional by writing more than 5000 lines of code. Overall, I am grateful for this journey and would like to thank Prof. Riegert and Trent University for sharing this experience with me.