Final Report: MGT 6203

Optimizing House Price Prediction: A Comparative Model Analysis

Group 20

Batoulsadat Haerian

(GT email: bhaerian3@gatech.edu, GitHub: bhaerian, GT ID: 903740612)

Thiago Henrique Rizzi Donato

(GT email: tdonato3@gatech.edu, GitHub: tdonato3, GT ID: 903565456)

Usama Bin Ali

(GT email: uali8@gatech.edu, GitHub: uali8, GT ID: 903661806)

Nico Wright

(GT email: nwright63@gatech.edu, GitHub: nwright63, GT ID: 903759703)

Fall 2023

INTRODUCTION

Choice of Topic: Precisely predicting house prices in the real estate market is a challenge for both sellers and buyers, potentially leading to dissatisfaction with transactions due to the risk of financial loss from either underpricing or overpricing.[1] To enhance successful property deals, we have introduced a robust model for high-accuracy price predictions, evaluated comprehensively against other models.

Business Justification: In property transactions, precision is essential to avoid overpayment or underpayment. Accurate predictions, based on home features, empower stakeholders to make informed decisions, ensuring fair deals. Real estate professionals, providing precise information and value predictions, play a crucial role in optimizing benefits for clients. This accuracy fosters trust, efficiency, and success in the real estate market, satisfying both buyers and sellers.[1-2]

Problem Statement: In addressing the challenge of accurately predicting house prices, we employed the Light GBM (LGBM) model with the goal of minimizing prediction errors in real estate transactions. This model underwent a comparative analysis alongside Multiple Linear Regression (MLR) and Polynomial Regression Model (PRM), with a primary emphasis on minimizing the Mean Squared Error (MSE). This approach aims to offer valuable insights into market dynamics, improve trend comprehension, and facilitate strategic decision-making for stakeholders.

Literature Survey

Real estate transactions, once reliant on human interactions, nowadays leverage digital platforms. These platforms offer extensive updated data, enabling informed decisions in the real estate market more accurately. Accurate prediction of house prices based on home features is crucial for transparency, trust, and efficiency in real estate transactions. While uncertainties and errors are expected, establishing an effective predictive model mitigates transaction risks, promoting industry health and stability.[3] Numerous analytical and machine learning methods, including Stepwise Regression, MLR, Bayesian Vector AutoRegressive models, Support Vector Regression, Principal Component Analysis, Decision Trees, Artificial Neural Networks, K-Nearest Neighbors, Ridge Regression, Random Forest, and Extreme Gradient Boosting, have been employed to forecast property prices.[4-9] Each method offers different accuracy levels by emphasis on specific factors for analysis, allowing firms to choose combinations based on their goals and available resources.

In this project, we introduced the LGBM model to minimize MSE and prediction error in comparison to MLR and PRM models. Recognized for its efficiency and speed, LGBM is specifically well-suited for handling extensive and complex datasets with numerous features common in real estate data.[10] Its excellence in regression tasks, notably its effectiveness in reducing MSE, positions it as an ideal choice for achieving high accuracy in predicting house prices. LGBM is renowned for adeptly managing complex relationships within the data, particularly in scenarios involving non-linear links.[10-11] Beyond MSE, we employed other pertinent metrics such as Mean of Absolute Error (MAE), Root Mean Squared Error (RMSE, derived from the square root of MSE), and R-squared (R^2) to gauge model performance in accurately predicting house values. By emphasis on the LGBM model, these parameters were compared with MLR and PR to determine which model minimizes prediction errors for house values in the real estate dataset, enhancing overall accuracy and supporting strategic decision-making for stakeholders.

Hypothesis

Introducing a robust model for predicting house prices that outperforms other models in terms of performance, ultimately contributing to more successful property deals.

Models

In this study, prediction of property prices was performed by using following methods.

MLR Model: As a statistical method, it uses two or more independent variables to predict the outcome of a dependent variable. It determines the variation pattern of the model its contribution of each predictor in this pattern.
PR Model: This model captures the complexity and non-linear relationships between variables by fitting a non-linear regression line, which may not be possible with linear regression.
LGBM Model: It is a strong algorithm that combines weak learners into strong ones. Each new model minimizes a loss function, like MSE, using gradient descent. It iteratively refines predictions, progressively improving the ensemble's performance until a stopping point is reached. While constructing the trees, the LGBM iteratively corrects previous errors, leading to improved accuracy over successive iterations and potentially yielding more accurate predictions than other models.[10]

Data Engineering
Data preprocessing
1. **Data Collection:** A total of 40 datasets, comprising 971,520 entries and spanning 56 columns, were obtained from different webpages. The primary dataset "**Real Estate listings in the US**" was obtained from the kaggle.com* webpage which was collated from https://www.realtor.com/, a real estate listing website. The 38 complementary datasets were retrieved from the https://www.redfin.com/ webpage, a real estate brokerage which provides access to timely and trustworthy housing market data. The demographic dataset of population liivng in the housing areas obtained from the Azure Open Datasets**.

* https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset/data

** https://learn.microsoft.com/en-us/azure/open-datasets/dataset-us-population-zip?tabs=azureml-opendatasets

2. Data Understanding: Characteristics of 40 datasets are listed in Table 1. Primary dataset with 904,966 entries and 10 columns (status, bed, bath, acre_lot, city, state, zip_code, house_size, prev_sold_date, and price) is organized into 19 states (PR, VI, MA, CT, NH, VT, NJ, NY, SC, TN, RI, VA, WY, ME, GA, PA, WV, DE) and various zip codes. The complementary dataset composed of 38 datasets, each with 28 columns (STATE OR PROVINCE, ZIP CODE, BEDS, BATHS, LOT SIZE, CITY, SQUARE, FEET, PRICE, STATUS, SALE TYPE, PROPERTY TYPE, ADDRESS, LOCATION, YEAR BUILT, DAYS ON MARKET, $/SQUARE, FEET, HOA/MONTH, NEXT OPEN HOUSE START TIME, NEXT OPEN HOUSE END TIME, URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis FOR INFO ON PRICING), SOURCE, MLS#, FAVORITE, INTERESTED, LATITUDE, and LONGITUDE). The demographic dataset with 33,788 entries and 19 columns (zip, lat, lng, city, state_id, state_name, zcta, parent_zcta, population, density, county_fips, county_name, county_weights, county_names_all, county_fips_all, imprecise, military, timezone, and Unnamed: 18) organized by the states and zip codes.

Table 1: Data types before trimming the outliers

	Datasets			Total		
	Primary	Complementary	Demographic	Before merging	After merging	After filtering of columns
Number of entries	904,966	32,766	33,788	971,520	937,732	937,732
Number of columns*	10	28	19	57	46	19*
Number of states	19	38	51	51	51	51

*: The dataset comprises 19 columns, encompassing features of bed, bath, acre_lot, city, house_size, price, status, sale_type, property_type, year_built, days_on_market, square_feet, hoa_month, latitude, longitude, population, density, county_fips, imprecise and military.

3. Data cleaning: Merging the 38 complementary datasets resulted in a single dataset with housing features across 38 states. However, the data of 12 remaining states (CA, CO, DC, DE, FL, GA, HI, NJ, NV, PA, TX, and WY) were not available for download. A thorough check confirmed the absence of any duplicated entries for each of the primary, complementary and demographic datasets. Furthermore, unnecessary data or empty columns were removed from the primary (e.g., prev_sold_date), complementary (e.g., ADDRESS, NEXT OPEN HOUSE START TIME, and NEXT OPEN

HOUSE END TIME), and demographic (county_weights and parent_zcta) datasets. In each dataset, the typo errors were corrected and labels with upper-cases were converted to lowercase. Also, the non-numeric characters were removed from the zip code column values and the full names of states were abbreviated. Each dataset had a large number of missing data across most columns. To handle These missing data, numerical values were imputed with the mean of the available values and categorical values were imputed with the most frequent value within their respective columns.

Data Wrangling

1. Data integration: Combining the 40 datasets by zip code as the key created a single master dataset with 937,732 entries and 46 columns. To eliminate irrelevant data, a subset of 21 columns (state, bed, bath, acre_lot, city, house_size, price, status, sale_type, property_type, year_built, days_on_market, square_feet, hoa_month, latitude, longitude, population, density, county_fips, imprecise, military) was selected from the original dataset.

2. Data Cleaning: The data was rechecked for typo errors and abbreviated form of each state name.

3. Filtering out the outliers: The dataset had potential outliers in certain predictors (bed, bath, acre_lot, house_size, year_built, days_on_market, square_feet, hoa_month, latitude, and longitude which influence the mean. Outliers were detected by z-scores (z-score of each row = (value mean)/standard deviation), and those exceeding 3 standard deviations from the mean were flagged if they fell outside the lower and upper limits.

4. Dealing with Missing Data: The missing data had previously been converted to the mean or most frequent value during data preprocessing in each of the three datasets. However, the master dataset was checked again for the remaining missing in the integer columns (bed, bath, year_built, days_on_market, hoa_month, population and county_fips) and float columns (acre_lot, house_size, square_feet, density, latitude, and longitude).

5. Dealing with multicollinearity: The correlation matrices was used to detect potential correlations between predictors in the regression model and visualized by the heatmap plot. The extent of variance inflation in predictors was assessed by the variance inflation factors (VIF). The VIF greater than 4 indicates potential correlation between predictors.

6. Data transformation: After conversion of status, state, city, sale_type, property_type, imprecise, and military columns into the categorical type, about 66% of the dataset entries was randomly split into training set (N = 4,659) and test set (N = 2,295) for regression analysis.

Data Exploratory

1. The Data Size and Structures: The merging of 40 datasets into one, followed by the removal of redundant columns, yielded a table containing 937,732 entries and 21 columns (Table 1). Within this dataset, several columns (bed, bath, acre_lot, house_size, price, days_on_market, square_feet, hoa_month, population, and density) exhibited potential outliers with notable skewness from the mean (Appendix: Table 2, Figure 1). For instance, the highest count of 'bed' is 123, while the average is 3.3 or the maximum number of 'bath' is 198, compared to the mean of 2.5. Additionally, the largest acre_lot is 140,130, with a mean of 18.5, and the greatest days_on_market is 591, with an average of 12.5. This notable deviation of maximum values from the mean underscores the impact of extreme values on the mean, suggesting their potential classification as outliers.

2. Outlier Trimming: After identifying outliers through z-scores and trimming values exceeding 3 standard deviations from the mean, the dataset was reduced to 6,963 entries. Manual exclusion of nine outliers from the hoa_month and days_on_market columns, considering their potential effect on the mean, further reduced the dataset to 6,954 entries across 21 columns. Following the complete removal of outliers, the disparity between maximum values and the mean in each column significantly decreased (Appendix: Table 3, Figure 2). For instance, the highest number of bedrooms is 8, while the mean is 3, and the maximum number of bathrooms is 8, with a mean of 2.4. Additionally, the largest acre_lot is 58.9, with a mean of 0.4, and the maximum house_size is 8,000, while the average is 2,013.3. Overall, the presence of outliers was significantly diminished, especially

in the bedrooms, bathrooms, and population columns. The trimming process also resulted in retaining 37 states out of the initial 51. In these states, the percentage of available houses for buyers was the highest in South Carolina (8.89%) but the lowest in Illinois (0.17%). Analyzing prices, South Carolina (SC) led with the highest price of $832,511, while Missouri had the lowest price at $269,450.

3. Dealing with Multicollinearity: We computed the correlation between multiple predictors in regression models through the correlation matrix, VIF factor, and visualized by using heatmaps (Figure 3). The correlation matrix revealed significant associations among predictors such as house_size, bed, bath, property_type_numerical, and hoa_month. The house's size demonestrated a positive correlation with the number of baths (0.8) and bedrooms (0.74) but a negative correlation with the type of property (-0.51). Additionally, there was a moderate positive correlation between the number of bedrooms and baths (0.72) but a weak negative correlation with the type of property (-0.58). The type of property showed a weak and negative correlation with the number of baths (-0.42). Conversely, none of the pairwise correlations among acre_lot, year_built, days_on_market, square_feet, hoa_month, latitude, longitude, population, density, and county_fips demonstrated particularly strong associations ($r < 0.4$ in each case).

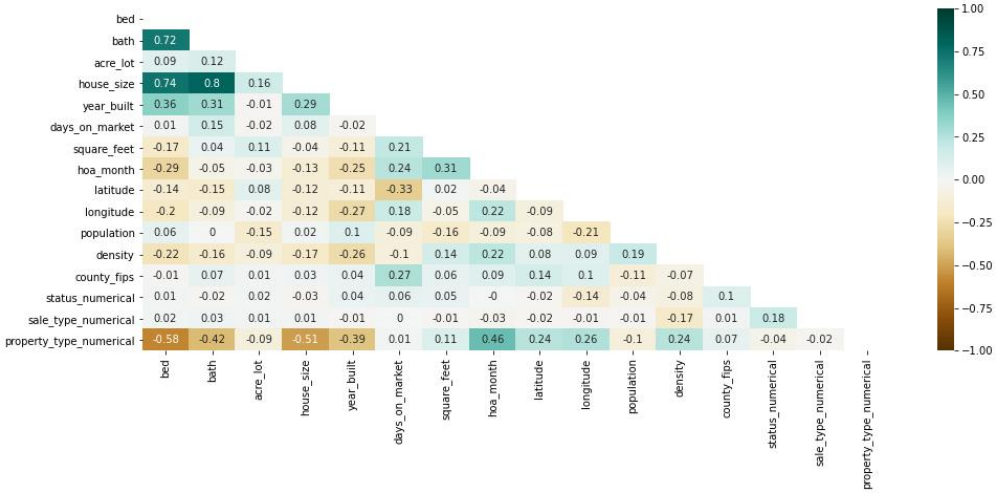

Figure 3: The correlation heatmap reveals a remarkable collinearity among certain predictors (darker colors indicate increased positive or negative correlation).

The assessed VIFs for the bed, bath, and house_size ranged between 2.1 and 3.4 showing an inflation attributed to the remarkable correlation between these predictors (Table 4).

Table 4: A summary of the VIF degree of bed and bath versus other predictors.

Variables	VIF_1	VIF_2	VIF_3	VIF_4	VIF_5
Intercept	917	923.5	924.0	922.1	923.9
bed	-	2.6	2.7	2.9	3.0
bath	2.9	-	2.1	3.2	3.3
acre_lot	1.1	1.1	1.0	1.1	1.1
house_size	3.1	2.4	-	3.4	3.4
latitude	1.2	1.2	1.2	1.1	1.2
longitude	1.2	1.2	1.2	1.2	1.2
population	1.2	1.2	1.2	1.2	1.2
density	1.2	1.2	1.2	1.2	1.2
county_fips	1.1	1.1	1.1	1.1	1.1
hoa_month	1.4	1.4	1.4	-	1.4
status_numerical	1.1	1.1	1.1	1.1	-
sale_type_numerical	1.1	1.1	1.1	1.1	2.0
property_type_numerical	1.9	2.0	2.0	1.7	1.0

VIF_1: bed vs other variables; VIF_2: bath vs other variables; VIF_3: house_sized vs other variables; VIF_4: hoa_month vs other variables; VIF_5: status vs other variables

An expansion in the size of the house correlates with an increase in the number of bedrooms and bathrooms. The status_numerical, sale_type_numerical, and property_type_numerical all exhibit VIF below 4 demonstrating very low levels of multicollinearity between these variables. In practical terms, VIF values below 4 are generally considered low, suggesting that these categorical variables do not contribute significantly to the variance inflation in the regression model, thus MLR, PMR, or LGBM models remain applicable for further investigation of MSE.

METHODOLOGY AND MODELLING

Methodology

The primary objective of this project is to precisely predict house prices based on provided features using MSE. To achieve this, a thorough analysis of the dataset using MLR, PMR, and LGBM models was performed. The modeling process map is illustrated in Figure 4.

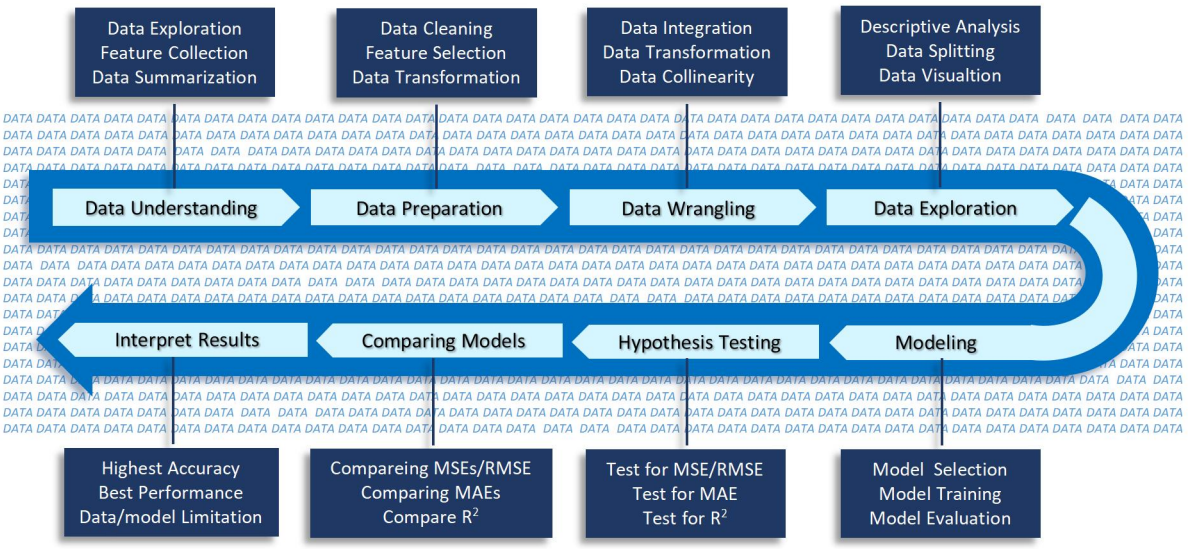

Figure 4: Modeling encompasses the stages of selecting, executing, and evaluating different models in a comparative approach.

As illustrated in Figure 4, post data preparation, the datasets undergo a wrangling step to make them suitable for analysis using compatible regression and machine learning models tailored to the structure and features of the master dataset. Following this, the modeling process includes model selection, where the chosen model is fitted with the training set (77% of the 6,954 or 4,659 entries) derived from the split main dataset. Subsequently, the model's performance is validated using the test set (33% of the 6,954 or 2,295 entries). Hypothesis testing was conducted for each model, focusing on R2, MSE, RMSE, and MAE metrics to assess its effectiveness in predicting house prices based on given features. The outcomes of the models were then compared, and the results were interpreted to evaluate and understand their overall performance.

Modeling

Addressing the research objective, two statistical models, MLR and PR, were chosen because of the presence of numerous predictors explaining house features and of them some exhibited non-linear relationships with house values. The LGBM, a machine learning algorithm was included as a main strategy due to its high accuracy in predicting house prices. It is a well-suited model for handling large and complex datasets with many observations and features, such as real estate data, which often involve numerous entries and attributes. Like the PR model, LGBM inherently captures non-linear relationships in a tree-based approach, such as relationship of house price with its features in real estate data.[10]

Regression Models

MLR: A MLR model was developed, with price as the dependent variable and state, bed, bath, acre_lot, city, house_size, state, status, sale_type, property_type, year_built, days_on_market, square_feet, hoa_month, latitude, and longitude as the independent variables. The model underwent training using the training set and was subsequently evaluated on the test set.

PR: This model was developed to improve fitting and maximize R-squared. The price as the dependent variable was estimated by the given nonlinear numeric (bed, bath, acre_lot, city, house_size, year_built, days_on_market, square_feet, hoa_month, latitude, and longitude) and categorical (city, state, sale_type, status, and property_type) predictors. The model was trained on the training set, and its performance was assessed on the test set.

Machine Learning Model

LGBM: To optimize the performance of the LGBM model, a hyperparameter tuning process was implemented. This process included splitting the training set into training and validation sets using 3 folds, while exploring different combinations of hyperparameters. The Light GBM model was assessed with different tree depths, specifically 5 or 15. Additionally, the maximum number of leaves per tree was examined with values of 1000 or 5000.

RESULTS

Prediction error was assessed using MSE, RMSE, MAE, and R^2 to evaluate the performance of the MLR, PR, and LGBM models. The designed hypotheses for the MSE, MAE, RMSE, and R^2 metrics are as follows:

Null Hypothesis (H_0): There is no significant relationship between the provided house features and its price. The predictors do not have a meaningful impact on predicting the house price.

Alternative Hypothesis (H_A): There is a significant relationship between the provided house features and its price. The predictors have a meaningful impact on predicting the house price.

In this section, R-squared (R^2) scores are computed using both OLS (Ordinary Least Squares) regression from the "sklearn.linear_model.LinearRegression" class and the r2_score function from the "sklearn.metrics" function (also applied for calculating MSE and MAE through mean_squared_error and mean_absolute_error functions). This leads to two inconsistent values for R^2, possibly because of variations in calculation formulas or approaches, handling of missing values or outliers, R^2 normalization for the number of predictors, dataset splitting into training and testing sets, and default or specified parameters. Due to marginal differences in R^2 values for each model, the OLS R^2 is utilized for individual models, while the r2_score R^2 is employed for comparing model performance (https://scikit-learn.org/).

1. Performance Analysis of Models

MLR model: In this model, the MSE, MAE, RMSE, and R^2 parameters were computed for the price as a dependent variable versus the given predictors. The model was fitted on the training set and validated on the test set. The OLS R^2 indicated a strong positive relationship between the predictors and price, signifying a good fit and accurate predictions of house prices (R^2 = 0.907). The adjusted R^2 score, which is also high and positive, suggests that the model is not overfitting the data, and the predictors are useful for predicting house prices (Adj. R^2 = 0.885). Overall, the house price is significantly associated with the house features, therefore, the given predictors have a meaningful impact on predicting the house price (F = 41.81, df = 880, p-value = 0.00). The r2_score R^2, MSE, RMSE, and MAE are 0.864, 3.352e10, 1.83e5, and 8.242e4 were computed for further comparative analysis across multiple models, respectively (Table 5).

PR Model: This model was applied because of nonlinear relationship between some predictors and prices (Appendix: Figure 5). The price was estimated as the dependent variable using nonlinear numeric (bed, bath, acre_lot, city, house_size, year_built, days_on_market, square_feet, hoa_month,

latitude, and longitude) and categorical (state, city, status, sale_type, and property_type) predictors. The model was trained on the training set, and its performance was evaluated on the test set. The OLS R^2 indicated a strong positive relationship between the price and the included predictors, signifying a well-fitted model capable of accurate predictions of house prices (R^2 = 0.914). The high and positive adjusted R^2 score suggests that the model avoided overfitting the data, and the predictors can be used for predicting house prices (Adj. R^2 = 0.894). Therefore, there is a significant correlation between house price and its features, indicating that the provided predictors play a substantial role in predicting house prices (F = 45.51, df = 886, p-value = 0.00). The computed r2_score R^2, MSE, RMSE, and MAE were 0.864, 3.352e10, 1.83e5, and 8.242e4 were 0.877, 3.046e10, 1.745e5, and 1.836e4, respectively (Table 5).

LGBM Model: This model was employed to compute the MSE, MAE, RMSE, and R^2 parameters for predicting price with the predictors. It was trained on the training set and then validated on the test set. R2_score R^2 revealed a robust positive relationship between the price and the predictors, indicating a good fit and accurate predictions of house prices (R^2 = 0.982). The MSE, RMSE, and MAE values were 3.352e10, 6.718e4, and 8.242e4, respectively (Table 5).

Table 5: Outcomes of the regression analysis employing MLR, PR, and LGBM models.

Parameters	Applied regression models to the test set		
	MLR (OLS model)	PR (OLS model)	LGBM
Method	Least Squares	Least Squares	-
No. observations	4,659	4,659	4,659
Df residuals	3,778	3,772	-
Df model	880	886	-
R-squared	0.907	0.914	-
Adj. R-squared	0.885	0.894	-
F-statistic	41.81	45.41	-
p-value	0.00	0.00	-
R-score	0.864	0.877	0.982
MSE	3.352e+10	3.045e+10	4.513e+9
RMSE ($\sqrt{MSE}$)	1.83e+05	1.75.e+05	6.72e+04
MAE	8.24+05	7.76+04	1.84+04

2. Performance Comparative Analysis of Models

Comparing the MSE, MAE, RMSE, and R^2 parameters between the MLR, PR, and LGBM models identified the enhanced model performance in the LGBM model much higher than MLR and PR. As the test set was validation with the MLR, PR, and LGBM models, the r2_score R^2 increased progressively from 0.864 to 0.877, and then reached 0.982 (Figure 6).

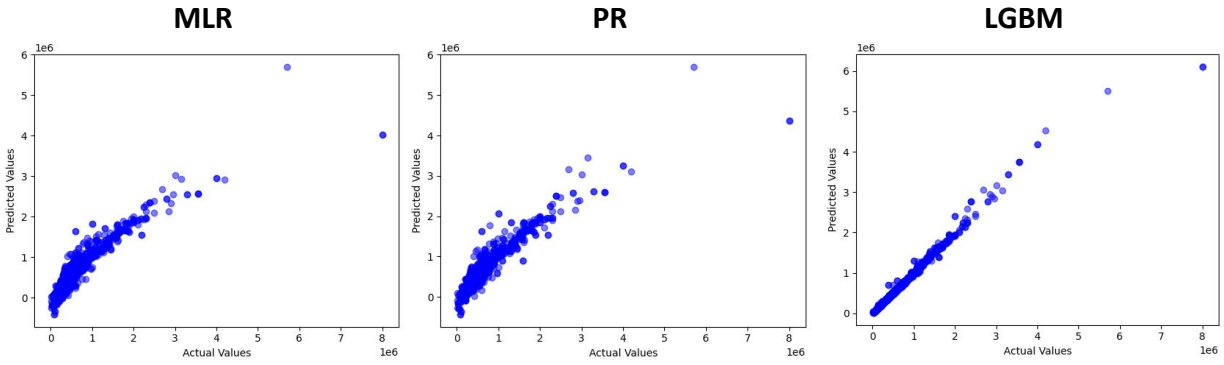

Figure 6: The scatter plot showing actual values vs predicted values in different models indicates a noticeable accuracy improvement. The plot reveals a strong positive correlation between the actual and predicted values, indicating that the models perform well.

The change in performance metrics from MLR to PR models was not very pronounced, but a substantial improvement was observed when transitioning from PR to the LGBM model. In summary,

the performance of the MLR model was lower than that of the PR model, but the LGBM model remarkably outperformed both MLR and PR models. Transitioning from the MLR model to the PR model and ultimately to the LGBM model results in a notable increase in the r2_score R^2, accompanied by a significant decrease in MSE, RMSE, and MAE (Figure 7).

CONCLUSION

In this project, we introduced a robust LGBM model for predicting house prices with higher accuracy and performance compared to MLR and PR models. The main focus was on minimizing MSE, RMSE, and MAE, while maximizing the R^2 metrics. A thorough evaluation of these parameters across the three models revealed that the LGBM model significantly outperformed MLR and PR models. The noteworthy reduction in MSE, RMSE, and MAE values suggests that the LGBM model enhances the accuracy of house price predictions more effectively than other models. This improvement can be attributed to the model's unique algorithm for correcting errors in previously built trees, thereby reducing prediction errors (Figure 8).[10,12]

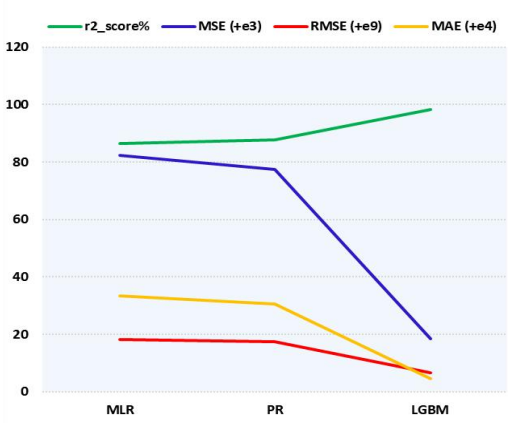

Figure 8: By transition of the models from the MLR to PR and then to LGBM, the r2_score R2, MSE, RMSE, and MAE metrics also changes. This highlighting the improvement in the model's performance when employing the LGBM model.

Additionally, we observed that the running speed of LGBM model was much higher than MLR and PR models. In conclusion, the results of this project suggest the adoption of LGBM for real estate market stakeholders, emphasizing its exceptional performance and accuracy, can contribute to successful dealings in the real estate sector.

LIMITATIONS

In this project, challenges arose with both the provided data and the LGBM model. Given the high sensitivity of the LGBM model and MSE and RMSE parameters to outliers, substantial efforts were dedicated to identifying and mitigating outliers within the datasets. However, this process led to a significant reduction in the number of observations from 937,732 to 6,954. Analyzing such a small but more homogenous dataset contradicts the nature of LGBM algorithm, which is designed for high-speed analysis of expansive datasets. Considering the high sensitivity of MSE and RMSE to outliers, MAE served as a control metric, because of its low sensitivity to outliers. Despite the extensive outlier trimming, the MAE values mirrored the outcomes of MSE and RMSE. Additionally, data for key states, such as California, Florida, and Texas were not available for download. To address these limitations, we suggest to increase the original sample size, conduct a separate analysis of trimmed outliers, incorporate datasets for the omitted states, and employ multiple machine learning models in conjunction with LGBM for more robust outcomes.

REFERENCES

1. Rosen S (1974): Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition. Journal of Political Economics. 82: 34–55.
2. Can A (1992): Specification and estimation of hedonic housing price models. Regional Science and Urban Economics. 22: 453–474.
3. Gao Q,Shi V, Pettit C, Han H (2022): Property valuation using machine learning algorithms on statistical areas in Greater Sydney. Australia Land Use Policy. 123: 106409.
4. Jafari A and Akhavian R (2019). Driving forces for the US residential housing price: a predictive analysis. Built Environment Project and Asset Management. 9: 2044-124X.
5. Zhang Q (2021). Housing Price Prediction Based on Multiple Linear Regression. Scientific Programming. 1-9.
6. Gupta R and Das S (2010). Predicting Downturns in the US Housing Market: A Bayesian Approach. The Journal of Real Estate Finance and Economics. 41:294–319.
7. Jiao YW (2017). Housing Price prediction Using Support Vector Regression. Master's Projects Master's Theses and Graduate Research. San Jose State University.
8. Khobragade AN, Maheswari N, Sivagami M (2018): Analyzing the housing rate in a real estate infomative system. A prediction analysis. International Journal of Civil Engineering and Technology. 9:1156–1164.
9. Zhang L (2023). Housing Price Prediction Using Machine Learning Algorithm. Journal of World Economy. 2:19-26.
10. Natekin A and Knoll A (2013). Gradient boosting machines, a tutorial. Frontiers in Neurorobotics. 7: 1-21.
11. Sibindi R, Mwangi RW, Waititu AG (2022). A boosting ensemble learning based hybrid light gradient boosting machine and extreme gradient boosting model for predicting house prices. Engineering Report. 5:1-19.
12. Liang Y, Wu J, Wang W, et al. (2019). Product marketing prediction based on XGboost and LightGBM algorithm. APR. 19: 150-153.

APPENDIX

Tables

Table 2: A summary of the values within certain columns prior to the removal of outliers.

Parameter	bed	bath	acre_lot	house_size	price	days_on_market	square_feet	hoa_month	population	density
count	802060	817782	671090	638549	937637	32616	26532	8433	52647	52647
mean	3.3	2.5	18.5	2186.7	861784.3	12.5	231.2	1749.3	32312.1	5997.5
std	2.0	1.9	1014.5	21990.7	2419174	55.3	268.0	57471.2	24301.6	10723.9
min	0	0.5	0	1	0	1	0	0	0	0
25%	2	1.8	0.1	1140	260000	1	142	30	13202	77.9
50%	3	2	0.3	1656	462000	3	191	100	27575	629.2
75%	4	3	1.13	2488	817900	5	266	275	46205	5844.5
max	123	198	140130	9999999	875000000	591	15000	2916786	116469	58289.6

Table 3: A summary of the values within certain columns after the removal of outliers.

Parameter	bed	bath	acre_lot	house_size	price	days_on_market	square_feet	hoa_month	population	density
count	6,954	6,954	6,954	6,954	6,954	6,954	6954	6,954	6,879	6,879
mean	3.0	2.4	0.4	2,013.4	534,913.8	8.8	268.6	208.3	28,074.9	578.5
std	1.1	0.9	1.5	1,022.2	488,550.7	22.2	144.3	279.3	17,489.9	971.2
min	0	0.5	0	271	6,500	1	5	0	35	0.2
25%	2	2	0	1,311	304,325	1	178	33	14,857	106.8
50%	3	2.5	0.1	1,796.5	417,950	3	229	109.5	25,534	273
75%	4	3	0.3	2,472	599,000	5	310	298	40,249	673
max	8	8	25.3	8,000	8,000,000	176	1,034	2,727	93,736	15,445.6

Graphs

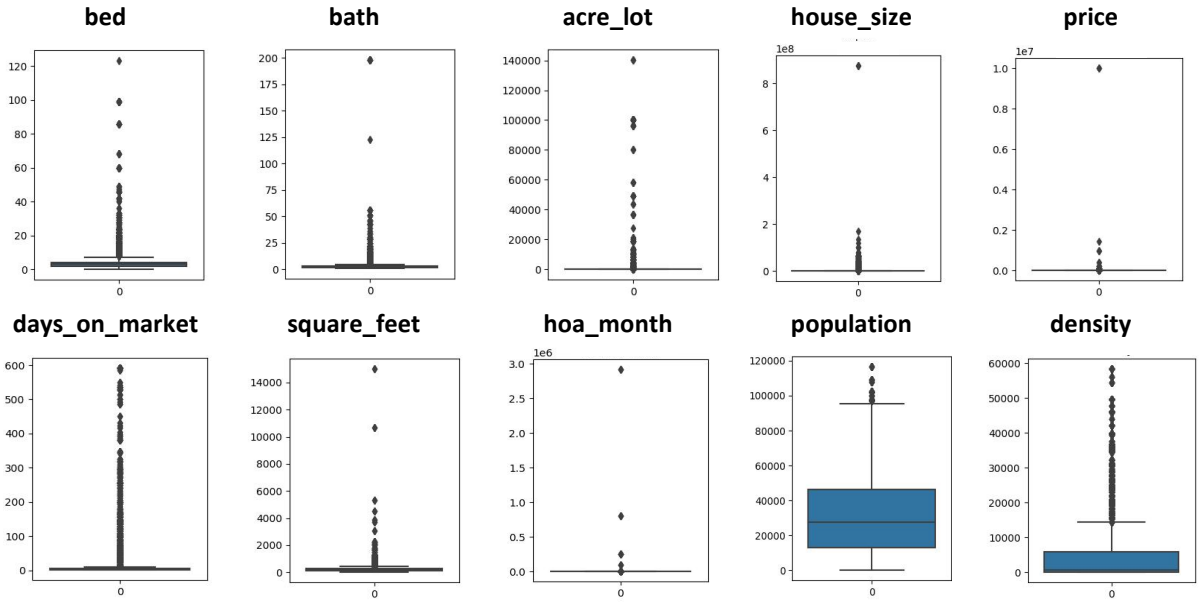

Figure 1: Box plots of four columns for checking the outliers before the trimming.

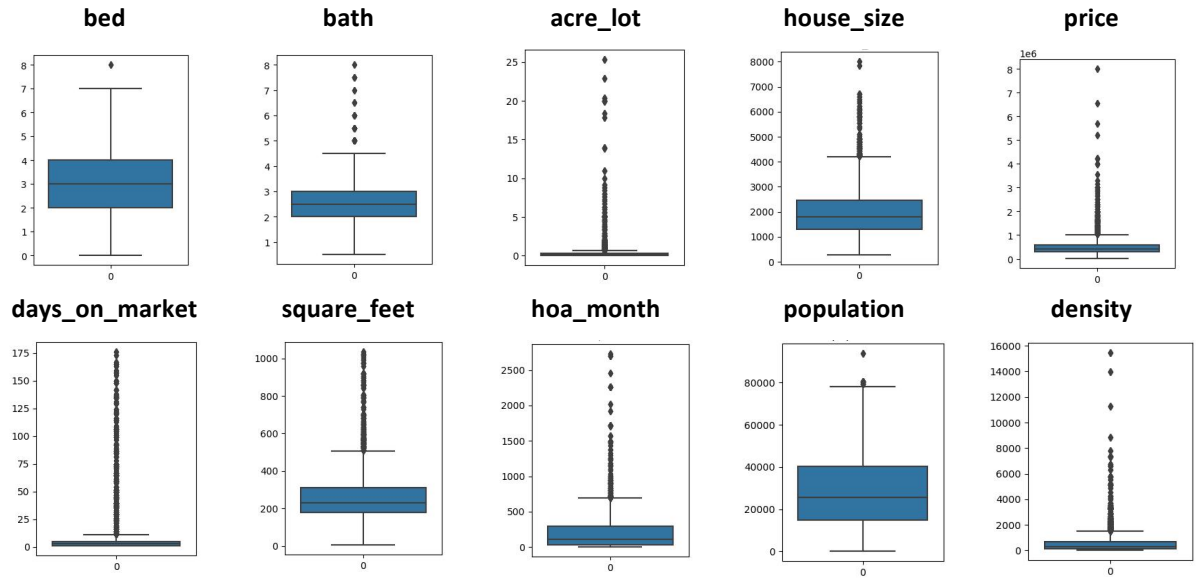

Figure 2: Box plots of four columns for checking the outliers after the trimming.

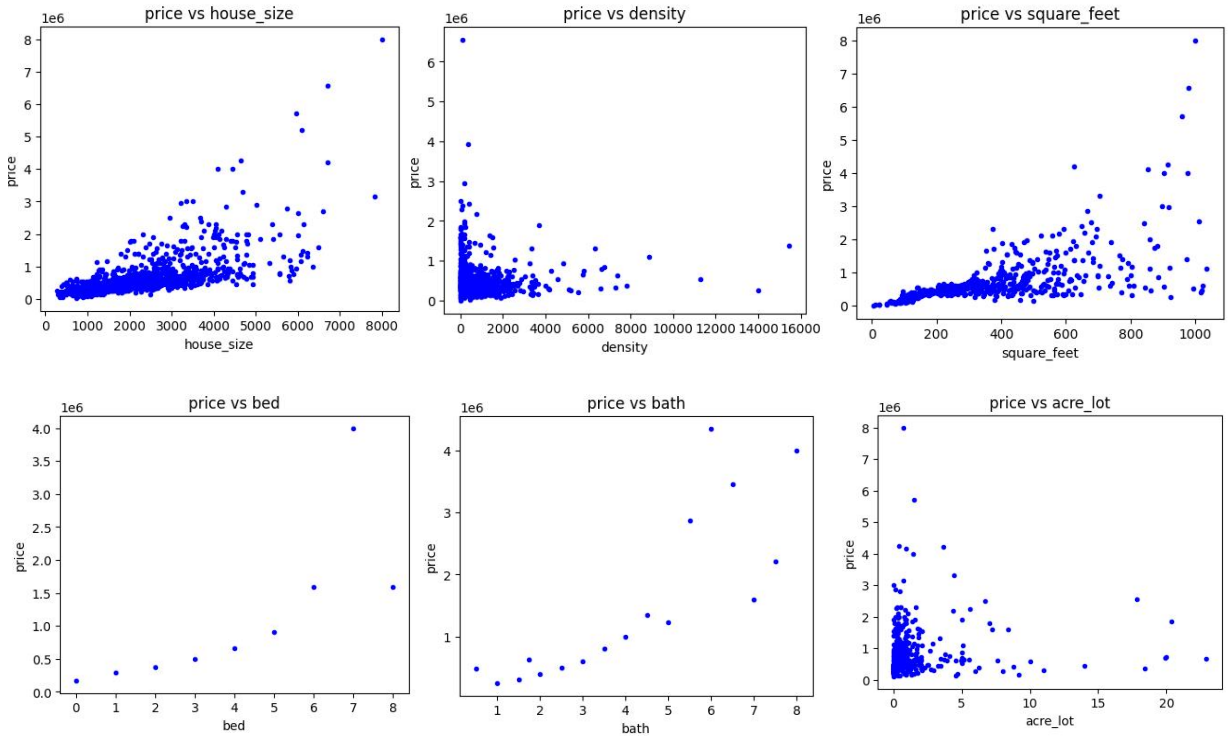

Figure 6: The non-linear relationship between the mean of price vs numeric predictors.

Python code

```
# Import libraries
import os
import zipfile
import numpy as np
import pandas as pd
import seaborn as sns
import lightgbm as lgbm
import statsmodels.api as sm
import statsmodels.formula.api as sms
import matplotlib.pyplot as plt

from patsy import dmatrices
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import PolynomialFeatures
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import (
   mean_squared_error,
   r2_score,
   mean_absolute_error,
   mean_absolute_percentage_error,
)
from sklearn.model_selection import (
   RandomizedSearchCV,
   train_test_split,
)
from statsmodels.tools.tools import add_constant
```

 # Instructions about how to load the datasets

A total of 40 datasets, comprising 971,520 entries and spanning 56 columns, were obtained from the subsequent webpages: (a) Primary dataset: The dataset titled Real Estate listings in the US was sourced from the kaggle.com* webpage, collated from https://www.realtor.com/, a real estate listing website. (b) Complementary dataset: To augment the primary dataset, 38 recommended datasets were retrieved from the https://www.redfin.com/ webpage. Redfin, a real estate brokerage, provides access to timely and trustworthy housing market data. (3) Demographic dataset: To explore the population demographics of housing areas, an additional dataset was obtained from the Azure Open Datasets.**

- https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset/data

- https://learn.microsoft.com/en-us/azure/open-datasets/dataset-us-population-zip?tabs=azureml-opendatasets

These datasets were stored in Team-20 GitHub project repository under the folder Data.

- https://github.gatech.edu/MGT-6203-Fall-2023-Canvas/Team-20

The next steps need to be followed to load the data.

1) Clone Team-20 GitHub project in your machine.

2) Unzip the .zip file "real_estate -redfin.zip".

*** The following link to the Dropbox can be also used as the second choice for reading this file: "https://www.dropbox.com/scl/fi/g394swhz6ukj53k943s95/real_estate-redfin.csv?rlkey=rwajplx5ejafldesvq8ffvmkv&dl=0".**
3) Update the file paths below according to where the repository was cloned.

```
# Real estate dataset
with zipfile.ZipFile(".../Repositories/Team-20/Data/real_estate -redfin.zip", 'r') as zip_ref:
    zip_ref.extractall(".../Repositories/Team-20/Data")

# The second choice for reading the file:
"""
url = "https://www.dropbox.com/scl/fi/g394swhz6ukj53k943s95/real_estate-redfin.csv?rlkey=rwajplx5ejafldesvq8ffvmkv&dl=0"
df2 = pd.read_csv(url)
"""

df1 = pd.read_csv(".../Repositories/Team-20/Data/real_estate -redfin.csv")
# Demographic dataset

df2 = pd.read_csv(".../Repositories/Team-20/Data/Demographic.csv", encoding="latin1")

# Homogenize ZIP code information between both datasets

df1["zip"] = df1["zip"].str.replace('[^\dA-Za-z]', '').astype(float)
df2["zip"] = df2["zip"].astype(float)

# Concatenate dataframes

df = pd.merge(df1, df2, how="left", on="zip")

# Convert states names into their corresponding codes

df['state'] = df['state'].replace(
    {
        'Virgin Islands':'VI',
        'South Carolina':'SC',
        'Tennessee':'TN',
        'Virginia':'VA',
        'Wyoming':'WY',
        'Georgia':'GA',
        'West Virginia':'WV',
    }
)
df.replace('South Carolina', 'SC', inplace=True)

# Rename variables

df = df[[
    "state",
    "bed",
    "bath",
```

```
    "acre_lot",
    "city_x",
    "house_size",
    "price",
    "status",
    "SALE TYPE",
    "PROPERTY TYPE",
    "YEAR BUILT",
    "DAYS ON MARKET",
    "$/SQUARE FEET",
    "HOA/MONTH",
    "LATITUDE",
    "LONGITUDE",
    "population",
    "density",
    "county_fips",
    "imprecise",
    "military",
]].rename(
    columns={
        "city_x": "city",
        "SALE TYPE": "sale_type",
        "PROPERTY TYPE": "property_type",
        "YEAR BUILT": "year_built",
        "DAYS ON MARKET": "days_on_market",
        "$/SQUARE FEET": "square_feet",
        "HOA/MONTH": "hoa_month",
        "LATITUDE": "latitude",
        "LONGITUDE": "longitude",
    }
)

# Prepare and filter columns

df = df[[
    'state', 'bed', 'bath', 'acre_lot', 'city', 'house_size', 'price',
    'status', 'sale_type', 'property_type', 'year_built', 'days_on_market',
    'square_feet', 'hoa_month', 'latitude', 'longitude', 'population',
    'density', 'county_fips', 'imprecise', 'military'
]]
df.dtypes

# Convert Object columns to Factor

df[[
    "status", "state", "city", "sale_type", "property_type", "imprecise", "military"
]] = df[[
    "status", "state", "city", "sale_type", "property_type", "imprecise", "military"
]].replace(np.nan,"?").astype("category")
```

 # Delete outliers

```
df3 = df.copy()
zscores_bed = (df3["bed"]-df3["bed"].mean())/df3["bed"].std()
zscores_bath = (df3["bath"]-df3["bath"].mean())/df3["bath"].std()
zscores_acre_lot = (df3["acre_lot"]-df3["acre_lot"].mean())/df3["acre_lot"].std()
zscores_house_size = (df3["house_size"]-df3["house_size"].mean())/df3["house_size"].std()

zscores_year_built = (df3["year_built"]-df3["year_built"].mean())/df3["year_built"].std()
zscores_days_on_market = (df3["days_on_market"]-
df3["days_on_market"].mean())/df3["days_on_market"].std()
zscores_square_feet = (df3["square_feet"]-df3["square_feet"].mean())/df3["square_feet"].std()
zscores_hoa = (df3["hoa_month"]-df3["hoa_month"].mean())/df3["hoa_month"].std()
zscores_lat = (df3["latitude"]-df3["latitude"].mean())/df3["latitude"].std()
zscores_lng = (df3["longitude"]-df3["longitude"].mean())/df3["longitude"].std()

df3["not_outlier_bed"] = zscores_bed.abs() < 3
df3["not_outlier_bath"] = zscores_bath.abs() < 3
df3["not_outlier_acre_lot"] = zscores_acre_lot.abs() <3
df3["not_outlier_house_size"] = zscores_house_size.abs() < 3
df3["not_outlier_year_built"] = zscores_year_built.abs() < 3
df3["not_outlier_days_on_market"] = zscores_days_on_market.abs() < 3
df3["not_outlier_square_feet"] = zscores_square_feet.abs() < 3
df3["not_outlier_hoa"] = zscores_hoa.abs() < 3
df3["not_outlier_lat"] = zscores_lat.abs() < 3
df3["not_outlier_lng"] = zscores_lng.abs() < 3

D = df3[
   df3.eval(
      'not_outlier_bed and not_outlier_bath and not_outlier_acre_lot and not_outlier_house_size
and not_outlier_year_built and not_outlier_days_on_market and not_outlier_square_feet and
not_outlier_hoa and not_outlier_lat and not_outlier_lng'
   )
].drop(
   columns=[
      "not_outlier_bed",
      "not_outlier_bath",
      "not_outlier_acre_lot",
      "not_outlier_house_size",
      "not_outlier_year_built",
      "not_outlier_days_on_market",
      "not_outlier_square_feet",
      "not_outlier_hoa",
      "not_outlier_lat",
      "not_outlier_lng",
   ]
)
D = D.drop(D.index[D['hoa_month'] >= 3500.0])
D = D.drop(D.index[D['acre_lot'] >= 50.0])
```

 # Calculate correlations between variables

```
Xs = D.loc[:, D.columns!='price']

g = sns.PairGrid(Xs, diag_sharey=False, corner=True)
g.map_lower(sns.scatterplot)
g.map_diag(sns.kdeplot)

round(Xs.corr(),2)
```

 # Plot correlation heatmap

```
plt.figure(figsize=(16, 6))
mask = np.triu(np.ones_like(round(Xs.corr(),2), dtype=bool))
heatmap = sns.heatmap(round(Xs.corr(),2), mask=mask, vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title("Predictors' Correlations Heatmap", fontdict={'fontsize':18}, pad=15);

corr = round(Xs.corr(),1)
plt.figure(figsize=(16, 6))
mask = np.triu(np.ones_like(round(Xs.corr(),2), dtype=bool))
sns.heatmap(round(Xs.corr(),2), mask=mask, vmin=-1, vmax=1, annot=True, cmap="Blues")

ans=sns.heatmap(round(Xs.corr(),1),  linewidths=1, cmap="Blues", center=0, annot=True)
figure = ans.get_figure()
figure.savefig('correlations.png', dpi=800)
```

 # Calculate VIF

```
A = D[['bed','bath','acre_lot','house_size','year_built','days_on_market','square_feet','hoa_month',
      'latitude','longitude','population','density','county_fips','status_numerical','sale_type_numerical',
     'property_type_numerical']]

y1, X1 = dmatrices('bed ~
bath+acre_lot+house_size+latitude+longitude+population+density+county_fips+hoa_month++status
_numerical+sale_type_numerical+property_type_numerical', data=A, return_type='dataframe')

vif1 = pd.DataFrame()
vif1['VIF'] = [variance_inflation_factor(X1.values, i) for i in range(X1.shape[1])]
vif1['variable'] = X1.columns

y2, X2 = dmatrices('bath ~
bed+acre_lot+house_size+latitude+longitude+population+density+county_fips+hoa_month+status_
numerical+sale_type_numerical+property_type_numerical', data=A, return_type='dataframe')

vif2 = pd.DataFrame()
vif2['VIF'] = [variance_inflation_factor(X2.values, i) for i in range(X2.shape[1])]
vif2['variable'] = X2.columns

y3, X3 = dmatrices('house_size ~
bed+acre_lot+bath+latitude+longitude+population+density+county_fips+hoa_month+status_numeri
cal+sale_type_numerical+property_type_numerical', data=A, return_type='dataframe')
```

```
vif3 = pd.DataFrame()
vif3['VIF'] = [variance_inflation_factor(X3.values, i) for i in range(X3.shape[1])]
vif3['variable'] = X3.columns

y4, X4 = dmatrices('hoa_month~
bed+acre_lot+house_size+bath+latitude+longitude+population+density+county_fips+status_numeri
cal+sale_type_numerical+property_type_numerical', data=A, return_type='dataframe')

vif4 = pd.DataFrame()
vif4['VIF'] = [variance_inflation_factor(X4.values, i) for i in range(X4.shape[1])]
vif4['variable'] = X4.columns

y5, X5 = dmatrices('status_numerical~
bed+acre_lot+house_size+bath+latitude+longitude+population+density+county_fips+hoa_month+pr
operty_type_numerical+sale_type_numerical', data=A, return_type='dataframe')

vif5 = pd.DataFrame()
vif5['VIF'] = [variance_inflation_factor(X5.values, i) for i in range(X5.shape[1])]
vif5['variable'] = X5.columns

# Control/Validation/Test sets

X = D.loc[:, D.columns!='price']
y = D["price"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
print(len(X_train), len(y_train), len(X_test), len(y_test))

# Impute missing values

imputer_y = SimpleImputer()
imputer_X_int = SimpleImputer(strategy="most_frequent")
imputer_X_float = SimpleImputer()

X_train_int_array = imputer_X_int.fit_transform(
  X_train[[
    "bed",
    "bath",
    "year_built",
    "days_on_market",
    "hoa_month",
    "population",
    "county_fips",
  ]]
)
X_test_int_array = imputer_X_int.transform(
  X_test[[
    "bed",
    "bath",
    "year_built",
    "days_on_market",
```

```
        "hoa_month",
        "population",
        "county_fips",
    ]]
)
X_train_int = pd.DataFrame(
    X_train_int_array,
    columns=[
        "bed",
        "bath",
        "year_built",
        "days_on_market",
        "hoa_month",
        "population",
        "county_fips",
    ]
)
X_test_int = pd.DataFrame(
    X_test_int_array,
    columns=[
        "bed",
        "bath",
        "year_built",
        "days_on_market",
        "hoa_month",
        "population",
        "county_fips",
    ]
)
X_train_float_array = imputer_X_float.fit_transform(
    X_train[[
        "acre_lot",
        "house_size",
        "square_feet",
        "density",
        "latitude",
        "longitude",
    ]]
)
X_test_float_array = imputer_X_float.transform(
    X_test[[
        "acre_lot",
        "house_size",
        "square_feet",
        "density",
        "latitude",
        "longitude",
    ]]
)
X_train_float = pd.DataFrame(
    X_train_float_array,
```

```
    columns=[
        "acre_lot",
        "house_size",
        "square_feet",
        "density",
        "latitude",
        "longitude",
    ]
)
X_test_float = pd.DataFrame(
    X_test_float_array,
    columns=[
        "acre_lot",
        "house_size",
        "square_feet",
        "density",
        "latitude",
        "longitude",
    ]
)

y_train = imputer_y.fit_transform(y_train.values.reshape(-1, 1))[:,0]
y_test = imputer_y.transform(y_test.values.reshape(-1, 1))[:,0]

# Append numeric and categorical columns

X_train_categorical = X_train[[
    "status",
    "state",
    "city",
    "sale_type",
    "property_type",
    "imprecise",
    "military",
]]
X_test_categorical = X_test[[
    "status",
    "state",
    "city",
    "sale_type",
    "property_type",
    "imprecise",
    "military",
]]
X_train_ = pd.concat(
    [
        X_train_int.reset_index(drop=True),
        X_train_float.reset_index(drop=True),
        X_train_categorical.reset_index(drop=True),
    ],
    axis=1
```

```
)
X_test_ = pd.concat(
   [
      X_test_int.reset_index(drop=True),
      X_test_float.reset_index(drop=True),
      X_test_categorical.reset_index(drop=True),
   ],
   axis=1
)
X_train_["price"] = y_train

# Linear regression model

ols_model_without_interaction_terms = sms.ols(
   formula="price ~ state + bed + bath + acre_lot + city + house_size + status + sale_type +
property_type + year_built + days_on_market + square_feet + hoa_month + latitude + longitude",
   data=X_train_
).fit()

y_test_pred1 = ols_model_without_interaction_terms.predict(X_test_)

# Linear regression – results

mean_squared_error(y_test, y_test_pred1)

mae = mean_absolute_error(y_test, y_test_pred1)
mse = mean_squared_error(y_test, y_test_pred1)
r2 = r2_score(y_test, y_test_pred1)

# Linear regression model with nonlinear terms

polynomial_bed = PolynomialFeatures(degree=2)
polynomial_bath = PolynomialFeatures(degree=2)
polynomial_acre_lot = PolynomialFeatures(degree=2)
polynomial_house_size = PolynomialFeatures(degree=2)
polynomial_density = PolynomialFeatures(degree=2)

# Linear regression with nonlinear terms - "bed" polynomial terms

polynomial_bed_train_ = polynomial_bed.fit_transform(X_train_[["bed"]])
polynomial_bed_test_ = polynomial_bed.transform(X_test_[["bed"]])

X_train_["polynomial_bed_0"] = polynomial_bed_train_[:,0]
X_train_["polynomial_bed_1"] = polynomial_bed_train_[:,1]
X_train_["polynomial_bed_2"] = polynomial_bed_train_[:,2]

X_test_["polynomial_bed_0"] = polynomial_bed_test_[:,0]
X_test_["polynomial_bed_1"] = polynomial_bed_test_[:,1]
X_test_["polynomial_bed_2"] = polynomial_bed_test_[:,2]
```

 # Linear regression with nonlinear terms - "bath" polynomial terms

```
polynomial_bath_train_ = polynomial_bath.fit_transform(X_train_[["bath"]])
polynomial_bath_test_ = polynomial_bath.transform(X_test_[["bath"]])

X_train_["polynomial_bath_0"] = polynomial_bath_train_[:,0]
X_train_["polynomial_bath_1"] = polynomial_bath_train_[:,1]
X_train_["polynomial_bath_2"] = polynomial_bath_train_[:,2]

X_test_["polynomial_bath_0"] = polynomial_bath_test_[:,0]
X_test_["polynomial_bath_1"] = polynomial_bath_test_[:,1]
X_test_["polynomial_bath_2"] = polynomial_bath_test_[:,2]
```

 # Linear regression with nonlinear terms - "acre lot" polynomial terms

```
polynomial_acre_lot_train_ = polynomial_acre_lot.fit_transform(X_train_[["acre_lot"]])
polynomial_acre_lot_test_ = polynomial_acre_lot.transform(X_test_[["acre_lot"]])

X_train_["polynomial_acre_lot_0"] = polynomial_acre_lot_train_[:,0]
X_train_["polynomial_acre_lot_1"] = polynomial_acre_lot_train_[:,1]
X_train_["polynomial_acre_lot_2"] = polynomial_acre_lot_train_[:,2]

X_test_["polynomial_acre_lot_0"] = polynomial_acre_lot_test_[:,0]
X_test_["polynomial_acre_lot_1"] = polynomial_acre_lot_test_[:,1]
X_test_["polynomial_acre_lot_2"] = polynomial_acre_lot_test_[:,2]
```

 # Linear regression with nonlinear terms - "house size" polynomial terms

```
polynomial_house_size_train_ = polynomial_house_size.fit_transform(X_train_[["house_size"]])
polynomial_house_size_test_ = polynomial_house_size.transform(X_test_[["house_size"]])

X_train_["polynomial_house_size_0"] = polynomial_house_size_train_[:,0]
X_train_["polynomial_house_size_1"] = polynomial_house_size_train_[:,1]
X_train_["polynomial_house_size_2"] = polynomial_house_size_train_[:,2]

X_test_["polynomial_house_size_0"] = polynomial_house_size_test_[:,0]
X_test_["polynomial_house_size_1"] = polynomial_house_size_test_[:,1]
X_test_["polynomial_house_size_2"] = polynomial_house_size_test_[:,2]
```

 # Linear regression with nonlinear terms - "population density" polynomial terms

```
polynomial_density_train_ = polynomial_density.fit_transform(X_train_[["density"]])
polynomial_density_test_ = polynomial_density.transform(X_test_[["density"]])

X_train_["polynomial_density_0"] = polynomial_density_train_[:,0]
X_train_["polynomial_density_1"] = polynomial_density_train_[:,1]
X_train_["polynomial_density_2"] = polynomial_density_train_[:,2]

X_test_["polynomial_density_0"] = polynomial_density_test_[:,0]
X_test_["polynomial_density_1"] = polynomial_density_test_[:,1]
X_test_["polynomial_density_2"] = polynomial_density_test_[:,2]
```

```
ols_model_with_nonlinear_terms = sms.ols(
   formula="price ~ polynomial_bed_0 + polynomial_bed_1 + polynomial_bed_2 +
polynomial_bath_0 + polynomial_bath_1 + polynomial_bath_2 + polynomial_acre_lot_0 +
polynomial_acre_lot_1 + polynomial_acre_lot_2 + polynomial_house_size_0 +
polynomial_house_size_1 + polynomial_house_size_2 + polynomial_density_0 +
polynomial_density_1 + polynomial_density_2 + state + city + status + sale_type + property_type +
year_built + days_on_market + square_feet + hoa_month + latitude + longitude",
   data=X_train_
).fit()

y_test_pred2 = ols_model_with_nonlinear_terms.predict(X_test_)
```

 # Linear regression with nonlinear terms – results

```
mae = mean_absolute_error(y_test, y_test_pred2)
mse = mean_squared_error(y_test, y_test_pred2)
r2 = r2_score(y_test, y_test_pred2)
```

 # LightGBM regression

```
lgbm_model_params = {
      'max_depth': (5, 15),
      'num_leaves': (1000, 5000)
}
lgbm_model_cv = RandomizedSearchCV(
   estimator=lgbm.LGBMRegressor(verbose=0),
   param_distributions=lgbm_model_params,
   cv=3,
   n_iter=3,
   verbose=0,
)
lgbm_model_cv.fit(X_train, y_train)

y_test_pred3 = lgbm_model_cv.predict(X_test)
```

 # LightGBM regression – results

```
mae = mean_absolute_error(y_test, y_test_pred3)
mse = mean_squared_error(y_test, y_test_pred3)
r2 = r2_score(y_test, y_test_pred3)
```