Predictive Modeling for Apartment Selling Prices in Queens, NY

Final project for Math 342W Data Science at Queens College

Hadassah Krigsman

May 2024

Abstract

This project aims to develop a predictive model for apartment selling prices in Queens, NY, focusing on condos and co-ops valued up to \$1M. Utilizing data from the Multiple Listing Service (MLS), which provides comprehensive real estate listings, this model seeks to outperform current market estimates. The ultimate goal is to offer more accurate predictions than those provided by platforms like Zillow. Through extensive data cleaning, feature engineering, and the application of several machine learning techniques, such as OLS, regression trees, and random forests, this project provides insights into the factors influencing apartment prices in this region. Among the techniques explored, the random forest model was the most effective.

1 Introduction

The objective of this project is to predict apartment selling prices in Queens, NY, focusing specifically on condos and co-ops priced under \$1 million. In this context, a mathematical model serves as a computational tool designed to mimic the real-world process of real estate pricing by correlating various apartment features with their selling prices. The primary variable we're interested in is `sale_price`.

This paper aims to build, examine, and refine predictive models that can estimate apartment prices in Queens more accurately. Regression tree modeling, linear regression, and random forest modeling were the statistical learning algorithms used for this. By using a variety of approaches, we can provide a well-rounded analysis of how different models perform under various conditions.

2 The Data

The data for this study comes from the MLSI, harvested through Amazon's MTurk, comprising 2,330 entries initially, which focus on apartment sales in Queens from February 2016 to February 2017. After an extensive cleaning and preprocessing phase, 26 relevant columns were identified and cleaned, and missing values were imputed as needed. Following this, rows with missing `sale_price` values were removed, leaving 528 data points for use in the predictive modeling. Additionally, the filtered housing data was joined with zip code data from 2013 government resources [1] to obtain latitudinal (column name: `LAT`) and longitudinal (column name: `LONG`) coordinates for each zip code, enriching the dataset with additional geographical information.

2.1 Featurization

In the modeling process, `sale_price` was designated as the response variable. Of the 26 usable columns in the dataset, 16 were character and 10 were numeric. Significant transformations and cleaning were applied to these columns to enhance their usability in modeling. For instance, `approx_year_built` was binned into 20-year periods and converted into a categorical factor. The `kitchen_type` column was divided into three categories (efficiency, eat-in, and combo), the `fuel_type` column into four categories (gas, oil, electric, and other), and the `dining_room_type` column into four categories (combo, formal, dining area, and other).

Binary columns were created for `cats_allowed` and `dogs_allowed` where presence was coded as 1 and absence as 0, transforming these into categorical factors as well. The column `coop_condo` was also turned binary with a co-op coded as 1 and a condo as 0. Additionally, a new binary column `garage_listed`

[1] The data had been sorted into comma-separated values by Eric Hurst; the file can be found in his GitHub: `https://gist.github.com/erichurst/7882666`

was derived from garage_exists by assigning 1 if a garage was listed in any form (e.g., 'underground', 'yes') and 0 if the entry was 'none' or missing.

The season_of_sale column was crafted by formatting the date_of_sale into MM/DD/YYYY, extracting the month, and categorizing it into seasons (spring, summer, fall, and winter) to capture seasonal variations in apartment prices. The total_charges column was computed by summing the values in common_charges, maintenance_cost, and total_taxes columns, providing a consolidated financial metric.

A zip_code column was created by extracting zip codes from full_address_or_zip_code. The zip codes were then categorized according to the areas depicted in Table 1 into a column labeled area.

After removing rows where sale_price was missing, the dataset was narrowed down to 528 observations and 23 columns, of which 10 a categorical factors (cats_allowed, dogs_allowed, dining_room_type, coop_condo, fuel_type, kitchen_type, approx_year_built_binned, garage_listed, season_of_sale, and area) and 13 were numeric (community_district_num, num_bedrooms, num_floors_in_building, num_full_bathrooms, num_total_rooms, parking_charges, pct_tax_deductibl, sq_footage, walk_score, total_charges, LAT, LONG, and sale_price). This refined dataset then served as the basis for the predictive modeling.

Table 1: Zip Codes in Queens, NY

Region	**Zip Codes**
Northeast Queens	11361, 11362, 11363, 11364
North Queens	11354, 11355, 11356, 11357, 11358, 11359, 11360
Central Queens	11365, 11366, 11367
Jamaica	11412, 11423, 11432, 11433, 11434, 11435, 11436
Northwest Queens	11101, 11102, 11103, 11104, 11105, 11106
West Central Queens	11374, 11375, 11379, 11385
Southeast Queens	11004, 11005, 11411, 11413, 11422, 11426, 11427, 11428, 11429
Southwest Queens	11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421
West Queens	11368, 11369, 11370, 11372, 11373, 11377, 11378

2.1.1 Errors and Missingness

Initial data examination revealed several inconsistencies and missing values. Specifically, spelling errors required correction, particularly in categorical variables. For instance, the kitchen_type column contained four disparate values such as 'efficiency', 'efficiency kitchen', 'efficiemcy', and 'efficiency ktchen', all intending to denote the same kitchen type. These variations were consolidated into a single category 'efficiency'. Additionally, the 'dogs_allowed' column presented entries like 'yes' and 'yes89'. The latter, which was a typographical error, was standardized to 'yes' to maintain consistency across the dataset.

Beyond categorical issues, numeric strings such as those indicating financial amounts or fees were cleaned to remove non-numeric characters and converted into usable numeric formats. Missing data points were addressed using sophisticated statistical methods to impute values, ensuring the dataset's integrity and usability for modeling purposes. These corrections and imputations were crucial for maintaining the robustness of the predictive models developed from this data.

Once the data was cleaned to ensure all non-missing entries were valid and standardized, the rows where the `sale_price` (y) was missing were removed, resulting in a refined subset of 528 observations. This subset still had 9 columns with missing values.

To explore the potential impact of missingness, missingness indicator variables were introduced for the columns that underwent imputation. These indicators were used to assess whether the missing data could hold predictive value or patterns. The `missForest` package in R, a non-parametric method perfect for handling mixed types of data, was used to impute the missing values. This approach helped to maintain the integrity and completeness of the dataset for more reliable modeling.

3 Modeling

We constructed three models to predict the sale prices of Queens apartments using a subset of the dataset. This subset included imputed values and was augmented with missingness indicators for each column, focusing only on rows where `sale_price` (y) was available. This approach allowed us to effectively compare the predictive performance of each model under consistent conditions.

3.1 Regression Tree Modeling

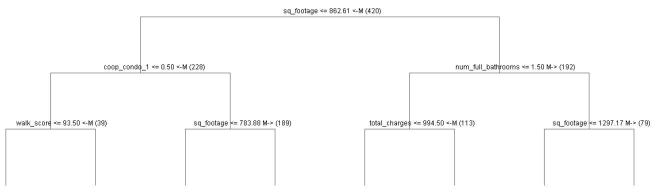
sq_footage <= 862.61 <-M (420)
coop_condo_1 <= 0.50 <-M (228)
num_full_bathrooms <= 1.50 M-> (192)
walk_score <= 93.50 <-M (39)
sq_footage <= 783.88 M-> (189)
total_charges <= 994.50 <-M (113)
sq_footage <= 1297.17 M-> (79)

Figure 1: the first three layers of the regression tree model

A regression tree with 291 nodes, 146 leaves, and a max depth of 23 was built to identify the key predictors of apartment prices. The most significant features included square footage, the amount of full bathrooms, and whether or not it was a co-op or a condo. The visualization of the tree (Figure 1) highlights these features at its nodes.

The regression tree model begins its split based on the square footage, a critical determinant in predicting apartment prices. The root node splits the data into two groups: properties with square footage less than or equal to 862.61 and those with more. This split suggests that size is a primary factor influencing price in the Queens apartment market.

For properties with square footage less than or equal to 862.61, the next split is based on the co-op/condo status, identifying whether the property is a co-op (1) or a condo (0). This distinction is crucial as it reflects different market preferences and pricing structures between these property types. Properties with a square footage above 862.61 further subdivide by the number of full bathrooms, indicating that among larger apartments, those with more bathrooms tend to command higher prices, pointing to the added value of additional amenities.

Subsequent splits in the tree look at factors such as walk score and the total charges included with each apartment. These splits give us a more detailed look at how attributes like location desirability and the overall cost of ownership fees influence sale prices

3.2 Linear Modeling

Linear modeling is a straightforward statistical approach used to predict a target variable based on its linear relationships with predictor variables. It constructs a formula that best fits a line through the data, minimizing the difference between the observed values and those predicted by the model.

For our project, we used linear regression to forecast the selling prices of apartments in Queens, NY, using various features like square footage, number of bathrooms, and location coordinates. The dataset was split into two parts: 420 observations for training the model and 108 for testing its predictions.

The complete summary of the linear model coefficients can be found in Table 2.

3.2.1 In-sample Analysis:

- **RMSE (Root Mean Squared Error):** The model's RMSE for the training data is approximately 73590.31, which indicates the typical magnitude of the errors in the price predictions. This measurement gives us an idea of how closely the predicted values on average match the actual selling prices within the training set.
- **R^2:** The model achieved an R^2 value of 0.8518 in the training set. This implies that approximately 85% of the variation in apartment sale prices is explained by the predictors used in the model.

3.2.2 Out-of-sample Performance:

- **RMSE:** On the test data, the RMSE increased to 86447.47. This indicates that the model, while still useful, tends to have higher prediction errors when applied to new, unseen data compared to the training data. This may reflect the model's sensitivity to the data it was trained on or variability in the test data.

- **R^2:** The test set's R^2 value is 0.7613618, indicating that about 76% of the variance in the actual sale prices in the test set is accounted for by the model. This reduction in R^2 compared to the training data suggests that the model's predictive power is somewhat less effective but still substantial when dealing with new data.

3.3 Random Forest Modeling

The random forest algorithm is an ensemble learning technique that builds multiple decision trees and merges them together to get more accurate and stable predictions. Each tree in a random forest splits nodes using randomly selected subsets of features and samples from the training set, which is known as bootstrap aggregation or bagging. This method helps in reducing overfitting and makes the model robust against noise in the data.

In predicting apartment prices, the random forest model can handle complex interactions between features without needing a pre-defined model structure (as in linear regression). The random forest model emerged as the most effective, given its ability to handle large data sets with high dimensionality and a mix of numerical and categorical variables.

This model has an OOB RMSE of 78341.24 indicating that, on average, the model's predictions deviate from the actual sale prices by this amount when evaluated on the unseen portions of the training set. This metric provides a good indication of the model's prediction error under practical conditions, approximating how it might perform on entirely new data.

The model's OOB R^2 of 0.8066575 is quite high, showing that the random forest model can explain about 80.67% of the variability in apartment sale prices within the unseen training data. This level of explanation power demonstrates the model's effectiveness in capturing the relationship between the predictors and the apartment prices, affirming its utility in real-world applications where predictive accuracy and reliability are crucial.

4 Performance Results

The performance of the models was evaluated using $\mathbf{R^2}$ and RMSE metrics. The random forest model displayed superior out-of-sample accuracy compared to the linear model, indicating its robustness and reliability for predictive purposes.

5 Discussion

The predictive models developed in this study provided insightful results into the apartment pricing dynamics in Queens, NY. However, the quantity and quality of the dataset used have a significant impact on the performance and reliability of predictive modeling. In this project, a substantial portion of the dataset had to be excluded from the analysis—specifically, 76% of the rows were dropped due to missing values in the `sale_price` column. This reduction in dataset size likely limited the predictive capabilities of our models.

The absence of `sale_price` data for a large number of records suggests that many potentially informative transactions were not included in the model training and validation processes. This exclusion can lead to a lack of representativeness in the dataset, potentially skewing the model's understanding of the underlying price dynamics and reducing its ability to generalize to the broader market.

Increasing the amount of available data could drastically improve the models' performance. More data would enhance the robustness of the models by providing a more comprehensive picture of the market, incorporating a wider range of property types, locations, and transaction conditions. This enhancement would likely lead to better model training, reduced overfitting, and improved accuracy and reliability of the predictions.

Furthermore, additional data could allow for more sophisticated feature engineering, such as more detailed categorizations of properties or more nuanced handling of outliers. It could also enable the use of more complex modeling techniques that may be too data-intensive or prone to overfitting with smaller datasets.

Future research should focus on gathering more complete datasets, possibly through more rigorous data collection methods or partnerships with real estate agencies or listing services. Researchers should also consider implementing advanced data imputation techniques to minimize the loss of data due to missing values. Additionally, exploring more diverse models and ensemble methods could further enhance the predictive accuracy by leveraging the strengths of multiple learning algorithms.

Overall, the potential for improving apartment price predictions in Queens, NY, is considerable, contingent on the availability and quality of the data used in the modeling processes. As such, efforts to expand and enrich the dataset should be a priority for future studies aiming to refine and advance the models discussed in this paper.

References

[1] Eric Hurst, US ZIP Code Latitude and Longitude, (2013).
GitHub repository: `https://gist.github.com/erichurst/7882666`

Code Appendix

```
---
title: "MATH 342W - Final Project"
author: "Hadassah Krigsman"
date: "2024-05-14"
output: html_document
```

```
---

### Load the Data
‘‘‘{r load data, echo=T, results=’hide’, message=FALSE, warning=FALSE}

# Load necessary libraries
pacman::p_load(skimr, dplyr, magrittr, lubridate, stringr, ggplot2, missForest)

library(tidyverse)  # A collection of R packages for data manipulation and visualization
library(readr)      # For reading CSV files

# Load the dataset
housing_data = read_csv(’housing_data_2016_2017.csv’)
zip_data = read_csv(’US-zip-codes.csv’)
skimr::skim(housing_data)

‘‘‘

### Clean the data:

‘‘‘{r clean data, echo=T, results=’hide’, message=FALSE, warning=FALSE}

# Remove unnecessary columns

housing_data = housing_data[, !(colnames(housing_data) %in% c(’HITId’, ’HITTypeId’, ’Title’,


# Function to reverse strings
reverse_string = function(s) {
  sapply(lapply(strsplit(s, NULL), rev), paste, collapse = "")
}

# Function to extract zip code by searching from the end
extract_zip_code = function(address) {
  # Reverse the address string
  reversed_address = reverse_string(address)

  # Regular expression for reversed zip code
  reversed_zip_code_pattern = "\\b\\d{5}(?:-\\d{4})?\\b"

  # Extract the reversed zip code using the pattern
  reversed_zip_code = str_extract(reversed_address, reversed_zip_code_pattern)

  # Reverse the extracted zip code back to its original form
  zip_code = ifelse(!is.na(reversed_zip_code), reverse_string(reversed_zip_code), NA)
  return(zip_code)
}

# Bin approx_year_built into 20-year periods
approx_year_bins = seq(min(housing_data$approx_year_built, na.rm = TRUE),
            max(housing_data$approx_year_built, na.rm = TRUE), by = 20)

housing_data = housing_data %>%
  mutate(approx_year_built_binned = cut(approx_year_built, breaks = approx_year_bins, incluc
  mutate(approx_year_built_binned = as.factor(approx_year_built_binned))


housing_data = housing_data %>%

  # clean specific columns
  mutate(
    kitchen_type = factor(ifelse(kitchen_type %in% c("eat in", "Eat In", "Eat in", "eatin"),
                        ifelse(kitchen_type %in% c("Combo", "combo"), "combo",
                        ifelse(grepl("^[0-9]+$", kitchen_type), NA,
                        ifelse(kitchen_type %in% c("efficiency", "efficiency kitchen", "ef
    cats_allowed = factor(ifelse(cats_allowed %in% c('yes', 'y'), 1, 0)),
    dogs_allowed = factor(ifelse(dogs_allowed %in% c('yes', 'yes89'), 1, 0)),
    coop_condo = factor(ifelse(coop_condo == 'co-op', 1, 0)),
    garage_listed = factor(ifelse(!is.na(garage_exists), 1, 0)),
    fuel_type = factor(ifelse(fuel_type %in% c("Other", "none"), "other", fuel_type)),
    dining_room_type = factor(dining_room_type),
    date_of_sale = as.Date(date_of_sale, format='%m/%d/%Y'),
    month_of_sale = month(date_of_sale),
    season_of_sale = factor(ifelse(month_of_sale %in% c(12, 1, 2), "Winter",
                      ifelse(month_of_sale %in% c(3, 4, 5), "Spring",
                      ifelse(month_of_sale %in% c(6, 7, 8), "Summer",
                      ifelse(month_of_sale %in% c(9, 10, 11), "Fall", NA))))),
    common_charges = as.numeric(gsub('[^0-9.-]', '', common_charges)),
    maintenance_cost = as.numeric(gsub('[^0-9.-]', '', maintenance_cost)),
    parking_charges = as.numeric(gsub('[^0-9.-]', '', parking_charges)),
    total_taxes = as.numeric(gsub('[^0-9.-]', '', total_taxes)),
    listing_price_to_nearest_1000 = as.numeric(gsub('[^0-9.-]', '', listing_price_to_nearest
    sale_price = as.numeric(gsub('[^0-9.-]', '', sale_price)),
    zip_code = sapply(full_address_or_zip_code, extract_zip_code))

skimr::skim(housing_data)

housing_data = housing_data %>%
  # remove unnecessary columns (either because there are too many NAs or because we just dor
  select(-num_half_bathrooms, -full_address_or_zip_code, -model_type, -date_of_sale, -month_
housing_data = housing_data %>%
  # calculate 'total_charges'
  mutate(
    common_charges = replace_na(common_charges, 0),
    maintenance_cost = replace_na(maintenance_cost, 0),
    total_taxes = replace_na(total_taxes, 0),
    total_charges = common_charges + maintenance_cost + total_taxes) %>%
  select(-common_charges, -maintenance_cost, -total_taxes) %>%
  filter(!is.na(total_charges))


# merge with the US zip code data to obtain the latitude-longitude data
housing_data = housing_data %>%
  left_join(zip_data, by = c("zip_code" = "ZIP")) %>%
  mutate(zip_code = factor(zip_code))

# create a mapping of zip codes to areas in order to collapse 'zip_code'
zipcode_to_area = c(
  "11361" = "Northeast Queens", "11362" = "Northeast Queens", "11363" = "Northeast Queens",
  "11354" = "North Queens", "11355" = "North Queens", "11356" = "North Queens", "11357" = "N
  "11365" = "Central Queens", "11366" = "Central Queens", "11367" = "Central Queens",
  "11412" = "Jamaica", "11423" = "Jamaica", "11432" = "Jamaica", "11433" = "Jamaica", "11434
  "11101" = "Northwest Queens", "11102" = "Northwest Queens", "11103" = "Northwest Queens",
  "11374" = "West Central Queens", "11375" = "West Central Queens", "11379" = "West Central
  "11004" = "Southeast Queens", "11005" = "Southeast Queens", "11411" = "Southeast Queens",
  "11414" = "Southwest Queens", "11415" = "Southwest Queens", "11416" = "Southwest Queens",
  "11368" = "West Queens", "11369" = "West Queens", "11370" = "West Queens", "11372" = "West
)
housing_data = housing_data %>%
  mutate(area = factor(zipcode_to_area[as.character(zip_code)], levels = unique(zipcode_to_a
  select(-zip_code)

# filter out all of the rows where 'sale_price' is NA
housing_data_filtered = housing_data %>%
  filter(!is.na(sale_price))

skimr::skim(housing_data_filtered)

```
### Missing Data

```{r missing data, echo=T, results='hide', message=FALSE, warning=FALSE}

# split the data
X = housing_data_filtered %>% select(-sale_price)
y = housing_data_filtered$sale_price
```

~~~
# create a matrix that represents missingness
M = apply(is.na(X), 2, as.numeric)
colnames(M) = paste("is_missing_", colnames(X), sep = "")
M = M[, colSums(M) > 0]
M = as_tibble(t(unique(t(M))))
skim(M)

# combine imputed data with the missingness matrix
X_imp = cbind(X, M)

# impute the data
pacman::p_load(missForest)
X_imp = missForest(data.frame(X_imp))$ximp

skimr::skim(X_imp)


```

### Modeling

```{r data splits for modeling, echo=T, results='hide', message=FALSE, warning=FALSE}
set.seed(8)
n_train = sample(1:nrow(X_imp), 420)
n_test = setdiff(1:nrow(X_imp), n_train)

y_train = y[n_train]
y_test = y[n_test]
X_train = X_imp[n_train, ]
X_test = X_imp[n_test, ]
```

# Regression Tree Modeling

```{r regression tree, echo=T, results='hide'}
pacman::p_load(YARF)

tree_mod = YARFCART(X_train, y_train, calculate_oob_error = FALSE)
get_tree_num_nodes_leaves_max_depths(tree_mod)

illustrate_trees(tree_mod, max_depth = 4, margin_in_px= 100, length_in_px_per_half_split = 4

#in-sample stats
y_hat_train = predict(tree_mod, X_train)
e_rt = y_train - y_hat_train # regression tree error
~~~

~~~
sd(e_rt)#in-sample rmse
1 - sd(e_rt) / sd(y_train) #in-sample r-squared

#oos stats
y_hat_test = predict(tree_mod, X_test)
e_rt_oos = y_test - y_hat_test
sd(e_rt_oos)#oos rmse
1 - sd(e_rt_oos) / sd(y_test)#oos r-squared

```

# Linear Modeling

```{r linear model, echo=T, results='hide'}
set.seed(8)
n_train = sample(1:nrow(X_imp), 420)
n_test = setdiff(1:nrow(X_imp), n_train)
y_train = y[n_train]
y_test = y[n_test]
X_train = X_imp[n_train, ]
X_test = X_imp[n_test, ]

linear_mod = lm(y_train ~ ., X_train)
summary(linear_mod)$sigma #rmse
summary(linear_mod)$r.squared #r-squared

yhat_oos = predict(linear_mod, X_test)
oos_e = y_test - yhat_oos
sd(oos_e) # oos RMSE

SST = sum((y_test - mean(y_test))^2)
1 - sum(oos_e^2) / SST # oos r-squared

```
# Random Forest Modeling

```{r}
model_rf = YARF(data.frame(X_train), y_train)
y_hat_test = predict(model_rf, data.frame(X_test))

sqrt(mean((y_hat_test - y_test)^2)) #oob rmse
1 - sum((y_test - y_hat_test)^2)/sum((y_test - mean(y))^2) #oob r-squared

```
~~~

Table 2: Summary of Linear Model Coefficients

Variable	**Estimate**	**Std. Error**	**t value**	**Pr($>\|t\|$)**
(Intercept)	-3.463e+07	2.871e+07	-1.206	0.228
cats_allowed1	1.357e+04	1.118e+04	1.214	0.225
community_district_num	3.508e+03	1.403e+03	2.501	0.013 *
coop_condo1	-1.158e+05	2.337e+04	-4.956	1.10e-06 ***
dining_room_typedining area	8.375e+03	7.564e+04	0.111	0.912
dining_room_typeformal	2.764e+04	9.993e+03	2.766	0.006 **
dining_room_typeother	1.144e+04	1.391e+04	0.822	0.411
dogs_allowed1	1.901e+04	1.265e+04	1.503	0.134
fuel_typegas	3.988e+04	3.177e+04	1.255	0.210
fuel_typeoil	5.192e+04	3.275e+04	1.585	0.114
fuel_typeother	6.413e+04	4.181e+04	1.534	0.126
kitchen_typeeat-in	-6.021e+03	1.198e+04	-0.502	0.616
kitchen_typeefficiency	-2.319e+04	1.184e+04	-1.958	0.051 .
num_bedrooms	4.931e+04	8.898e+03	5.542	5.69e-08 ***
num_floors_in_building	6.250e+03	839.1	7.448	6.72e-13 ***
num_full_bathrooms	7.578e+04	1.348e+04	5.620	3.76e-08 ***
num_total_rooms	1.652e+04	6.144e+03	2.688	0.008 **
parking_charges	431.5	133.5	3.233	0.001 **
pct_tax_deductibl	-270.5	1.193e+03	-0.227	0.821
sq_footage	11.70	13.20	0.886	0.376
walk_score	10.16	420.9	0.024	0.981
approx_year_built_binned[1930,1950)	-8.028e+04	2.235e+04	-3.592	0.0004 ***
approx_year_built_binned[1950,1970)	-8.944e+04	2.257e+04	-3.964	8.85e-05 ***
approx_year_built_binned[1970,1990)	-8.589e+04	2.922e+04	-2.939	0.0035 **
approx_year_built_binned[1990,2010]	-1.843e+04	3.241e+04	-0.569	0.570
garage_listed1	-111.9	1.186e+04	-0.009	0.992
season_of_saleSpring	-1.218e+04	1.107e+04	-1.101	0.272
season_of_saleSummer	-5.021e+03	1.107e+04	-0.454	0.650
season_of_saleWinter	1.768e+04	1.068e+04	1.655	0.099
total_charges	16.63	5.398	3.080	0.002 **
LAT	7.054e+05	3.117e+05	2.263	0.024 *
LNG	-7.997e+04	2.867e+05	-0.279	0.780
areaNorth Queens	-46.40	2.481e+04	-0.002	0.998
areaCentral Queens	-1.753e+04	2.524e+04	-0.694	0.488
areaJamaica	-6.280e+04	2.437e+04	-2.577	0.010 *
areaNorthwest Queens	-1.897e+04	6.113e+04	-0.310	0.757
areaWest Central Queens	2.989e+04	3.173e+04	0.942	0.347
areaSoutheast Queens	-2.816e+03	2.346e+04	-0.120	0.905
areaSouthwest Queens	-5.062e+04	3.087e+04	-1.640	0.102
areaWest Queens	-2.498e+03	4.037e+04	-0.062	0.951
is_missing_community_district_num	-4.445e+04	7.969e+04	-0.558	0.577
is_missing_dining_room_type	2.963e+03	9.553e+03	0.310	0.757
is_missing_fuel_type	-2.901e+04	1.831e+04	-1.585	0.114
is_missing_kitchen_type	-2.925e+04	3.313e+04	-0.883	0.378
is_missing_num_floors_in_building	1.232e+03	9.616e+03	0.128	0.898
is_missing_parking_charges	-1.219e+04	9.794e+03	-1.245	0.214
is_missing_pct_tax_deductibl	-3.279e+03	1.080e+04	-0.304	0.762
is_missing_sq_footage	-3.916e+03	8.273e+03	-0.473	0.636
is_missing_approx_year_built_binned	4.974e+04	2.094e+04	2.376	0.018 *

Residual standard error: 73590 on 371 degrees of freedom
Multiple R-squared: 0.8518, **Adjusted R-squared:** 0.8326
F-statistic: 44.42 on 48 and 371 DF, **p-value:** < 2.2e-16