Data set

This project focuses on the suggested topic of housing price prediction, specifically rental prices in New York City – an issue which impacts both residents seeking housing and landlords looking to list a property at market value. Data were scraped from Realtor.com using the open-source *HomeHarvest* Python library[1], with the data-scraping script run several times to collect additional listings as they were posted. A total of 15,121 listings were returned across all scraping runs, but only 4,035 contained the minimum necessary features for price prediction (including number of bedrooms, ZIP code, and square footage) as well as the target value of price (or price range). If a price range was given rather than a single price, the midpoint of the range was treated as the listing price for prediction. The data set includes apartments across all five boroughs of NYC with a wide range of sizes and asking prices. A small number of outliers (list price over $1,000,000 per month or 10+ bathrooms) were removed.

Because the information obtained through web scraping was limited, additional features were added based on attributes of the apartment's neighborhood obtained from U.S. Census data (2023 ACS 5-year estimates) at the ZIP code level, using the *census* package. This additional data aims to capture the ways in which an apartment's neighborhood, not just attributes of the living space itself, impact pricing.

Preprocessing and feature engineering

Features included from the scraped listing data are **number of bedrooms**, **number of bathrooms**, **area in square feet**, **style of building** (apartment, condos, co-op, etc.), and year of construction (converted to **age of building** by subtracting from the current year). Based on the ZIP code, a **borough** discrete variable was added using a list of NYC zip codes by borough[2]. The scraped data set listed full bathrooms and half-bathrooms (with no tub) separately; these were converted into a single value by multiplying the number of half-bathrooms by 0.5 and adding it to the number of full bathrooms. Age of building information was sometimes missing; in these cases the average building age for the ZIP code (obtained from Census data) was used.

Features included from the Census data are **median income** and **median age** of residents, **number of vacant housing units**, **total population**, and **travel time to work**. Travel time was listed in the Census data as the estimated number of residents in each response category (e.g. "less than 5 minutes", "30 to 34 minutes"); these values were converted into the average commute time for residents based on the midpoints of each range.

Numeric features were normalized using *scikit-learn*'s StandardScaler, and discrete features (borough and building style) were one-hot encoded. Additionally, I chose to predict the log of the target value (listing price) rather than the price itself due to its long-tailed distribution – this is a common statistical strategy for more accurately modeling long-tailed data such as prices, and allows the actual price prediction to be easily obtained (and displayed in the Streamlit app) by exponentiating the value output by the model.

One additional preprocessing step was added after testing models and developing the Streamlit application. Because the number of bedrooms, number of bathrooms, and square footage are highly correlated with one another, only one of these features was given high feature weight for predictions.

[1] https://github.com/ZacharyHampton/HomeHarvest
[2] https://github.com/erikgregorywebb/nyc-housing/blob/master/Data/nyc-zip-codes.csv

While this is not an issue for prediction on the test data (where the same correlation is present), it leads to unintuitive behavior of the interactive app, where it is easy to change one of these features independent of the others. This means that users could (for example) double the area of an apartment but see only minimal impact to the predicted price. To mitigate this issue, I created two “size” variables using PCA, which captured around 88% of variance in the bedrooms, bathrooms, and area features. The first component is interpretable as overall size, as all three features contribute near-equally to its magnitude, and explains ~73% of the variance. The second component is interpretable as rooms other than bedrooms and bathrooms -- it is positively associated with square footage and negatively with bedrooms/bathrooms. This resulted in only slightly worse model performance (a difference in R-squared of 0.01-0.02 for most models) while greatly improving the interpretability of results.

Model selection process and results

Six models were tested: Multiple linear regression, ridge regression, lasso regression, random forest, gradient boosting, and XGBoost. After splitting the data into train and test sets (80% train, 20% test), all models were evaluated on the training data using 5- and 10-fold cross-validation. The ridge regression model performed very poorly in cross-validation (average R-squared of 0.287 for 10-fold CV), while the other two regression models were approximately equal in performance (average R-squared of 0.785-0.786). The random forest model slightly outperformed the gradient boosting and XGBoost models in cross-validation, though the difference was potentially negligible (R-squared = 0.868, compared to 0.858 and 0.859).

On the test dataset, the XGBoost model performed best, though again the difference was small (R-squared = 0.885, compared to 0.874 for the random forest and gradient boosting models). Similarly, its root mean square error and mean absolute error were slightly lower than those of the other models. Based on these results, the XGBoost model was selected for deployment. The fact that the test dataset R-squared value was slightly higher than the average cross-validation value is promising, as it suggests overfitting was not an issue.

Key insights and recommendations

- Supplementing the scraped data with neighborhood-level information was valuable. In particular, the average commute time was one of the most predictive features.
- Web scraping is challenging. I spent a long time attempting to scrape StreetEasy, but both straightforward HTTP requests and Selenium-based scraping were blocked by anti-bot measures. The HomeHarvest scraper was not blocked, but a considerable number of the listings returned had too much data missing to be usable for analysis.
- When features are highly correlated, it is worth combining them (using PCA or similar techniques) for a more interpretable model.

Limitations and future improvements

- The dataset includes only listings which were active at the times the scraper was run, primarily from the months of May, June, and July. This means that any seasonal patterns in rent prices would not be appropriately captured.
- More visualizations and interactive components could be added to the Streamlit application.
- Better model performance could potentially have been obtained by adjusting hyperparameters.