G12: Towards understanding Airbnb rental pricing

Sayar Ghosh Roy (sayar3), Shradha Sehgal (ssehgal4), Aditya Sinha (as146)

Abstract: We aim to understand the underlying factors that drive rental prices of Airbnb properties. We develop supervised models to predict rental prices, given features like location and typology of a property. We also present insights through various visualization schemes.

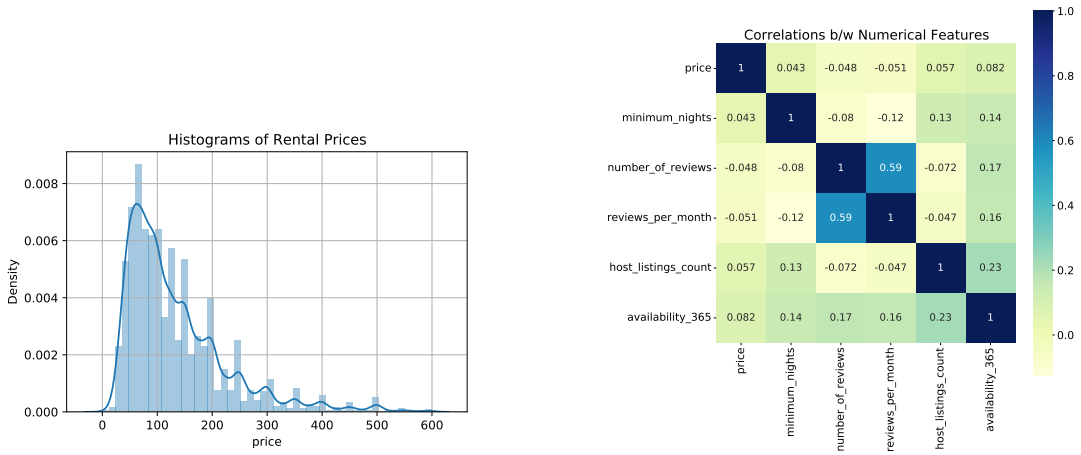
Introduction: In this work, we perform an exploratory analysis of Airbnb data and develop models to predict the price of an Airbnb property, given attributes related to its location (latitude, longitude, neighborhood, etc.) and property type (apartment, private room, etc.). Understanding the factors driving pricing of rentals is crucial for all stakeholders, be it AirBnB owners, hosts/managers, or seekers, in order to both set and find fair prices. In this report, we compare the performance of 13 supervised Machine Learning regression models for price estimation. We experiment with different encoding techniques like One-Hot Encoding and Label Encoding for categorical attributes, and study the effect of using available text features. Overall, we find that an Extra Trees regressor model or a Histogram-based Gradient Boosting regressor model leads to the best RMSE and R2 after tuning input specification settings and appropriate hyperparameters. Additionally, we observe that using TF-IDF features extracted from the available textual data and utilizing a One-Hot encoding scheme for categorical features gives a boost in R2 and a drop in RMSE for most models. We also experiment with multiple outlier removal techniques. Our code-base is publicly available here[1].

Motivation: Airbnb has served over 60M individuals looking for convenient yet affordable housing options. Present in over 34K cities, Airbnb allows homeowners to rent out their properties for short periods, thereby offering accommodation seekers an alternative to typical hotels. Given the recent surge in Airbnb usage, understanding the pricing dynamics for Airbnbs is of utmost importance for Airbnb hosts and potential guests. Our proposed data mining principles involving extensive data analysis, pre-processing, predictive modeling, clustering, etc., rely on pricing data conditioned on location and property type and, as such, would be applicable to any housing market (long-term rentals, real-estate properties for sale).

Related Work: [6, 5] utilize Machine Learning models such as Decision Trees, Random Forests, and Gradient Boosted Regression Trees for predicting rental prices of warehouses. [1, 3] attempt to study underlying price determining factors using Ordinary Least Squares (OLS) and quantile regression models for holiday-related travel. [4] use sentiment analysis to factor in the importance of customer reviews available on Airbnb. [2] propose a multi-modal setup with reviews and geographical information to create more reliable forecasting models.

The dataset for our study has been sourced from Kaggle[2]. Previous studies involving this dataset have focused on EDA[3], visualization schemes[4,5], and predictive modeling[6,7,8,9].

[1] https://github.com/sayarghoshroy/place2crash
[2] https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data
[3] https://www.kaggle.com/code/dgomonov/data-exploration-on-nyc-airbnb
[4] https://www.kaggle.com/code/subhradeep88/airbnb-analysis-eda
[5] https://www.kaggle.com/code/alvaroibrain/airbnb-data-analysis
[6] https://www.kaggle.com/code/duygut/airbnb-nyc-price-prediction
[7] https://www.kaggle.com/code/chirag9073/airbnb-analysis-visualization-and-prediction
[8] https://www.kaggle.com/code/wguesdon/nyc-airbnb-eda-visualization-regression
[9] https://www.kaggle.com/code/jrw2200/smart-pricing-with-xgb-rfr-interpretations



(a) Histogram of listing prices

(b) ρ between numerical attribute pairs

Figure 1: (a) Histogram of listing prices upto \$600: List of prices resemble a long-tailed distribution, (b) Pearson's ρ between pairs of numerical attributes and the target variable price: Attributes do not show strong correlation with price, no strongly correlated numerical attribute pairs (apart from reviews per month & number of reviews)

Methodology: We build data mining models to understand the factors that drive pricing of Airbnbs. To achieve this, we divide our work into the following broad categories.

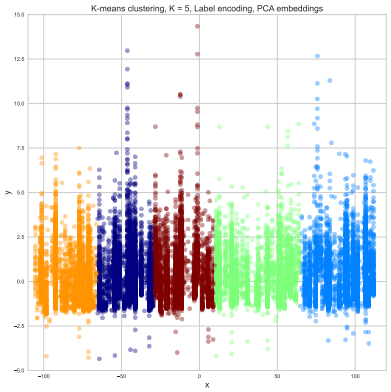
[1] **Exploratory Data Analysis**: We perform statistical data analysis to get an overview of our dataset's attributes' central tendencies, dispersions, inter-feature correlations, etc. Our dataset contains 47,906 entries with features such as neighborhood, location, room type, and price (dependent variable for predictive models), which ranges between \$0 and \$10,000 with a mean of \$152.72 and a standard deviation of \$240.15. Further statistics on our dataset can be found in Section 1.1 of the Appendix.

[2] **Visualizations and Insights**: We perform extensive visualizations to capture the distribution of features, and the interrelations among specific attributes[10]. From the histogram of listing prices (upto \$600) in Figure 1a, we observe that the set of prices resembles a long tailed distribution. We also analyzed the spread of prices w.r.t categorical attributes through various violin plots (Appendix, Section 1.2). We compute Pearson's correlation coefficient between the numerical features in Figure 1b and note that the numerical features do not show a strong correlation with the target variable (price). In addition, we do not observe any strongly correlated numerical features apart from 'reviews per month' & 'number of reviews', which is expected.

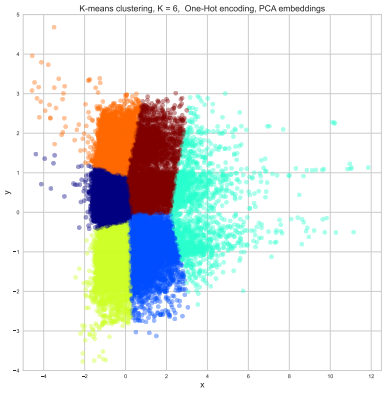
[3] **Predictive Models**: We fit multiple regression models, including Linear Regression with Ridge & Lasso regularization, Stochastic Gradient Descent (SGD) Regressor, Decision Tree, Extra Tree, Random Forest, Adaptive Boosting, Bagging, Extra Trees, Gradient Boosting, Histogram based-gradient boosting, K Nearest Neighbors, and Feed-forward Neural Networks (Multi-Layer Perceptrons), to predict the expected rent of an Airbnb listing based on our available feature set. Within our set up, we experimented with transforming categorical features ('neighbourhood group', 'neighbourhood', 'room type') into numerical attributes through two approaches — Label Encoding and One-Hot encoding. We standardized numerical features via Z-score normalization[11]. We encoded the textual field 'name' (containing a short description of the property) into Term Frequency - Inverse Document Frequency (TF-IDF) vectors.

[10] Details can be found Section 1.2 of the Appendix.

[11] We experimented with Min-Max normalization which led to very similar results.



(a) Using Label Encoding for categorical features ($K = 5$)



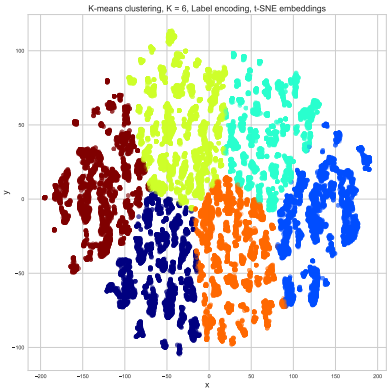
(b) Using One-Hot Encoding for categorical features ($K = 6$)

Figure 2: K-means clustering using Principal Component Analysis (PCA) for dimensionality reduction. We observe that using One-Hot encoding (OH) leads to more evenly spread out feature embeddings compared to Label Encoding (LE), whose corresponding PCA embeddings seem to occupy certain vertical zones.

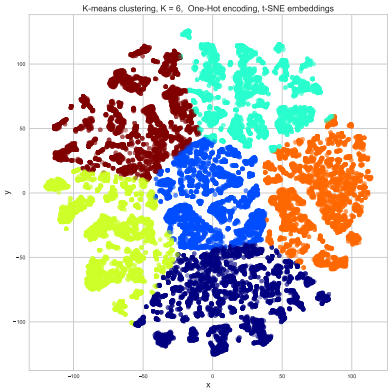
For all of our experiments, we ignored the following host-related and relative temporal features: 'host_id', 'host_name', 'last_review', 'calculated_host_listings_count'. Additionally, we dropped datapoints containing atleast a single missing (or NaN) value. For outlier removal, we experimented with pruning datapoints based on Z-score (attributes more than 3 Standard Deviations away from the mean marked as outliers) and Inter-Quartile Ranges (IQR) (attributes outside the range $[Q_1 - 1.5\times\text{IQR}, Q_3 + 1.5\times\text{IQR}]$ considered outliers). For our experiments, we isolated 10% of the data as a held-out test set. Note that we used the training set's mean and variance to normalize our test set. We performed appropriate grid searches over corresponding hyper-parameters for the different ML models and cherry-picked settings leading to the best average performance across an 11-fold cross validation. The specific hyper-parameter candidates used for grid search for the particular models can be found in Table 7, under Appendix, Section 1.3. Throughout, we evaluated model performance over two metrics – Root Mean Square Error (RMSE) & Coefficient of Determination (R2). Lastly, we re-trained our models utilizing the optimum hyper-parameter settings, on the complete train split and evaluated them upon the held out test set. A simplistic playground of our regression pipeline can be found here[12].

[4] **Clustering**: For our clustering experiments, we considered the entire set of numerical and categorical features. Within that, we considered two types of encodings for our categorical features (similar to predictive modeling experiments) — namely, Label Encoding (LE) and One-Hot encoding (OH). We experimented with Principal Component Analysis (PCA) and T-distributed stochastic neighbor embedding (t-SNE) to reduce the data dimensionality to 2 (primarily for visualization). For each of the 4 settings (Categorical features encoded via LE & OH, usage of PCA & t-SNE), we computed the optimum K for K-means clustering using the Elbow method[13]. For all settings, optimum K corresponding to the *knee* or *elbow* turned out to be 6, except for LE + PCA for which it was 5. The plots corresponding to the cluster assignments can be seen in Figures 2 and 3. For both PCA and t-SNE, we observe that using OH leads to more evenly spread out feature embeddings compared to LE which causes em-

[12] https://github.com/sayarghoshroy/place2crash/blob/main/regression.ipynb
[13] https://www.scikit-yb.org/en/latest/api/cluster/elbow.html



(a) Using Label Encoding for categorical features ($K = 6$)



(b) Using One-Hot Encoding for categorical features ($K = 6$)

Figure 3: K-means clustering using 2-dimensional T-distributed stochastic neighbor embeddings (t-SNE). Similar to Figure 2, we observe that using OH leads to more evenly spread out feature embeddings along the x-axis.

beddings to be lie in certain vertical zones. Also, t-SNE embeddings lead to better separation among individual datapoints thereby producing better distinguished clusters.

Empirical Results: The results for our predictive regression models are as shown in Table 1. Here, we report metrics corresponding to the best experimental setting (considering all possible choices for the set of input features, outlier removal methods, and specific model hyperparameters) found via K-fold cross validation. In each row, 'CV RMSE' summarizes the RMSE seen during Cross Validation. We tabulate RMSE and R2 values on the held out test set. The experimental settings leading to the best results during model selection can be found in Table 2. The grid space of candidate hyperparameters for individual models can be found in the Appendix, Section 1.3) (Table 7). In subsequent paragraphs, we further report insights from specific ablation studies to capture the contribution of text features, outlier removal methods and encoding schemes for categorical features. Overall, we find that Extra Trees and Histogram-based Gradient boosting leads to the best performance over RMSE and R2 after tuning input specification and model hyperparameters.

In Table 3, we showcase the gain in R2 that 3 models (Adaboost, Random Forest and Gradient Boosting) achieve upon employment of the Z-score based outlier removal (OR column captures the Outlier Removal method being employed). However, we also note a significant

Model	CV RMSE	Test RMSE	Test R2
Ridge	0.891 ±0.265	0.697	0.227
Lasso	0.891 ± 0.265	0.697	0.227
SGD regressor	0.897 ± 0.265	0.696	0.228
Decision Tree	1.229 ± 0.309	0.692	0.236
Extree	1.004 ± 0.278	0.712	0.192
Random Forest	0.907 ± 0.248	0.744	0.118
Adaboost	0.821 ± 0.031	2.102	0.156
Bagging	0.868 ± 0.265	0.682	0.259
Extrees	0.890 ± 0.261	**0.673**	**0.277**
Gradient Boosting (GB)	0.895 ± 0.262	0.708	0.201
Histogram-based GB	0.874 ± 0.267	**0.673**	**0.277**
KNN	0.889 ± 0.262	0.678	0.269
MLP	0.919 ± 0.146	0.684	0.255

Table 1: Best results obtained for each ML model after tuning input specification and model specific hyperparameters. Experimental setting for these models have been listed in Table 2. Model specific candidate hyperparameters are listed in Table 7 (Appendix, Section 1.3). Best results have been highlighted in bold.

Model	Hyperparameters	TF	FE	OR
Ridge	fit_intercept: False, alpha: 1.0	Yes	OH	None
Lasso	fit_intercept: False, alpha: 0.0001	Yes	OH	None
SGD regressor	loss: squared_error, penalty: l2, learning_rate: adaptive, alpha: 0.0001	Yes	OH	None
Decision Tree	splitter: best, max_leaf_nodes: 100.0	No	OH	None
Extree	max_leaf_nodes: 50.0	No	LE	None
Random Forest	n_estimators: 50, max_leaf_nodes: NaN, bootstrap: True	Yes	OH	Z-score
Adaboost	n_estimators: 100	Yes	OH	Z-score
Bagging	n_estimators: 500, bootstrap: True, bootstrap_features: True	Yes	OH	None
Extrees	n_estimators: 100, max_leaf_nodes: NaN, bootstrap: True	No	LE	None
Gradient Boosting (GB)	n_estimators: 100, max_leaf_nodes: 5.0	Yes	LE	Z-score
Histogram-based GB	max_leaf_nodes: 100.0, l2_regularization: 1.0	Yes	OH	None
KNN	n_neighbors: 60	No	OH	None
MLP	activation: ReLU, opt: Adam, n_hidden_layers: 16	No	OH	None

Table 2: Tuned input feature specification and model specific hyperparameters for individual models (based on average RMSE over 11-fold cross validation). Metrics in Table 1 are based on these settings. TF indicates whether the available textual field was utilized (as a TF-IDF based encoding) or not. FE describes the Feature Encoding scheme for available Categorical features — in that, OH stands for One-Hot encoding and LE stands for Label Encoding. OR describes the Outlier Removal method being used.

increase in RMSE compared to the no-outliers-removed setting. Overall, for most models (all except the 3 showcased in Table 3), we see that removing outliers from the training set leads to an overall decrease in performance, associated with a drop in R2 and an increase in RMSE. Metrics corresponding to employing Z-score and IQR based outlier removal for all models can be found in Table 9 (Appendix, Section 1.3). Note here that while evaluating these settings on the test set, we consider the *complete* test set including datapoints that are potential outliers.

For most models (8 out of 13), utilizing the available textual information (encoded into TF-IDF vectors) lead to improvements over R2 and RMSE. In Table 4, we highlight the specific 8 models for which we observed a performance boost upon utilizing the textual feature. The computed metrics for all models, both with and without the textual feature can be found in Table 8 (Appendix, Section 1.3).

We performed a similar ablation study to evaluate whether a specific encoding scheme for categorical features (between Label Encoding and One-Hot encoding) provides a notable improvement over another. From Table 5, we see that for *only* 4 out of 13 models, there is a drop in performance (for either R2 or RMSE) while moving from Label Encoding to One-Hot Encoding. Thus, using One-Hot Encoding generally leads to better results as compared to utilizing a simplistic Label Encoding (similar to our unsupervised clustering experiments).

Conclusion: In this study, we analyzed the Airbnb pricing data and built data mining models capable of predicting the expected rent of a short-term property rental, conditioned on its

Model	OR	CV RMSE	Test RMSE	Test R2	RMSE Drop	R2 Gain
Random Forest	None	0.907 ± 0.248	**0.744**	0.118	-0.677	0.156
	Z-score	0.694 ± 0.019	1.421	**0.274**		
Adaboost	None	3.928 ± 2.199	**1.038**	-0.716	-0.533	0.829
	Z-score	0.983 ± 0.076	1.571	**0.113**		
Gradient Boosting	None	0.895 ± 0.262	**0.708**	0.201	-0.754	0.03
	Z-score	0.715 ± 0.021	1.462	**0.231**		

Table 3: Ablation Study for Outlier Removal (OR) snapshot: Effect of introducing Z-score based outlier removal before model training. Only the above 3 models showed improvement in R2 scores after employing Z-score based outlier removal. The complete OR ablation results (including IQR based removal) can be found in Table 9 (Appendix, Section 1.3).

Model	Text Feature	CV RMSE	Test RMSE	Test R2	RMSE Drop	R2 Gain
Ridge	No	0.899 ± 0.263	0.708	0.201	0.011	0.026
	Yes	0.891 ±0.265	**0.697**	**0.227**		
Lasso	No	0.899 ± 0.263	0.709	0.2	0.012	0.027
	Yes	0.891 ± 0.265	**0.697**	**0.227**		
SGD regressor	No	0.900 ± 0.263	0.708	0.201	0.012	0.027
	Yes	0.897 ± 0.265	**0.696**	**0.228**		
Random Forest	No	0.911± 0.254	0.746	0.114	0.002	0.004
	Yes	0.907 ± 0.248	**0.744**	**0.118**		
Adaboost	No	0.769 ± 0.019	2.280	0.143	0.178	0.013
	Yes	0.821 ± 0.031	**2.102**	**0.156**		
Bagging	No	0.878 ± 0.266	0.691	0.239	0.009	0.02
	Yes	0.868± 0.265	**0.682**	**0.259**		
Gradient Boosting	No	0.907 ± 0.252	0.712	0.191	0.004	0.01
	Yes	0.895 ± 0.262	**0.708**	**0.201**		
Histogram-based Gradient Boosting	No	0.883 ± 0.267	0.684	0.255	0.011	0.022
	Yes	0.874 ± 0.267	**0.673**	**0.277**		

Table 4: Ablation Study for utilizing Textual Feature (TF) snapshot: Effect of introducing TF-IDF encoding of the available textual field as a feature. These 7 models showed some improvement in RMSE & R2 scores. The complete TF ablation results can be found in Table 8 (Appendix, Section 1.3).

location and typology information. We showcased various outcoming insights from our visualization schemes, performed necessary data pre-processing steps, experimented with unsupervised clustering algorithms, and analyzed the performance of our predictive regression models. Our results demonstrate that using a Histogram-based Gradient Boosting regressor model utilizing the available text field as a TF-IDF vector or a Extra Trees regressor model without any textual features leads to the best results after tuning other input specification settings and appropriate hyperparameters. In general, we observe that using the available textual feature, using One-Hot encoding for categorical features, and not performing any outlier removal prior to training lead to better performing models.

Model	FE	CV RMSE	Test RMSE	Test R2	RMSE Drop	R2 Gain
Ridge	LE	0.904 ± 0.26	0.715	0.186	0.018	0.041
	OH	0.891 ±0.265	**0.697**	**0.227**		
Lasso	LE	0.904 ± 0.261	0.715	0.186	0.018	0.041
	OH	0.891 ± 0.265	**0.697**	**0.227**		
SGD regressor	LE	0.919 ± 0.260	0.732	0.146	-0.036	0.082
	OH	0.897 ± 0.265	**0.696**	**0.228**		
Decision Tree	LE	1.001± 0.278	0.712	0.192	0.02	0.044
	OH	1.229 ± 0.309	**0.692**	**0.236**		
Extree	LE	1.004 ± 0.278	0.712	**0.192**	0.02	-0.037
	OH	0.910 ± 0.257	**0.692**	0.155		
Random Forest	LE	0.925± 0.253	0.771	0.052	0.027	0.066
	OH	0.907 ± 0.248	**0.744**	**0.118**		
Adaboost	LE	0.727 ± 0.009	2.280	0.143	0.178	0.013
	OH	0.821 ± 0.031	**2.102**	**0.156**		
Extrees	LE	0.890 ± 0.261	**0.673**	**0.277**	-0.768	-0.024
	OH	0.723 ± 0.019	1.441	0.253		
KNN	LE	0.900 ± 0.258	0.697	0.226	0.01	0.022
	OH	0.888 ± 0.261	**0.687**	**0.248**		
MLP	LE	0.977 ± 0.160	0.739	0.131	0.055	0.123
	OH	0.919 ± 0.146	**0.684**	**0.254**		

Table 5: Ablation Study for Categorical Feature Encoding (FE): Model performance corresponding to 2 choices for encoding available categorical features — Label Encoding (LE) and One-Hot encoding (OH). For 8 out of 13 models, using OH leads to improved RMSE and R2.

References

[1] J. L. Nicolau D. Wang. Price determinants of sharing economy based accommodation rental: A study of listings from 33 cities on airbnb.com, *International Journal on Hospitality Management*, 2017.

[2] Y. Qin FN. Peng, K. Li. Leveraging multi-modality data to airbnb price prediction, *Proceedings of Second International Conference on Economic Management and Model Engineering*, 2020.

[3] J. L. Nicolau L. Masiero and R. Law. A demand-driven analysis of tourist accommodation price: A quantile regression of room bookings, *International Journal of Hospitality Management*, 2015.

[4] H. Rezaei P. Kalehbasti, L. Nikolenko. Airbnb price prediction using machine learning and sentiment analysis, 2019.

[5] A. Ihler Y. Ma, Z. Zhang and B. Pan. Estimating warehouse rental prices using machine learning techniques, *International Journal of Computers, Communications Control*, 2018.

[6] H. Yu and J. Wu. Real estate price prediction with regression and classification, 2016.

1 Appendix

This Technical Appendix is divided into three parts:

- Section 1 presents dataset statistics.
- Section 2 showcases additional visualizations that highlight certain trends.
- Section 3 presents results of ablation studies and lists out candidate hyperparameters for predictive regression models.
- Section 4 presents further experimental details.

1.1 Dataset Details

The dataset contains the following attributes:

1. Unique ID
2. Name: A textual description of the property with an average of 36.90 characters and 6.69 tokens. The (minimum, maximum) and standard deviations of the number of characters and tokens being (1, 179), 10.51, (1, 39), 2.39, respectively.
3. Host-related attributes: Host_id and Host_name. We do not utilize these metadata features for the purpose of building regression models as pricing of a rental property should remain invariant to the host name.
4. Neighbourhood_group: 5 possible values $\in$ {Brooklyn, Manhattan, Queens, Staten Island, Bronx}. From the Table 1.1, we observe that the number of datapoints in top two neighbourhood_groups is relatively higher compared to the bottom three.
5. Neighbourhood: 221 possible neighborhoods in NYC
6. Room_Type: 3 possible values, namely {Entire home/apt, Private Room, Shared Room}.

Neighbourhood_group	Count
Manhattan	21661
Brooklyn	20104
Queens	5666
Bronx	1091
Staten Island	373

Table 6: Number of listings in each Neighborhood_group: Manhattan and Brooklyn collectively contain 41765 out of the 48895 listings (≈ 85%)

7. Quantitivate attributes such as minimum_nights, number_of_reviews, reviews_per_month, calculated_host_listings_count, availability_365 and price. Price ranges between $0 and $10,000 with a mean of $152.72 and a standard deviation of $240.15. We ignore the calculated_host_listings_count attribute for building regression models. minimum_nights varies between 1 and 1250 with mean 7.03, standard deviation 20.51. On average, we have 23 reviews for each property, with the average number of reviews per month for a property at 1.37. The mean availability of a property stands at 112.78 days each year.
8. Location: latitude and longitude coordinates for each listing. Latitude values range between 40.50 and 40.91 with a mean of 40.73 and a standard deviation of 0.05. While Longitude values range between -74.24 and -73.71 with a mean of -73.95 and a standard deviation of 0.05.

The dataset of 47,906 entries has NaN values in the following columns:

- Name: 16 rows
- host_name: 21 rows
- last_review: 10052 rows
- reviews_per_month: 10052 rows

1.2 Visualizations

To get some insights into the various types of listings available in each neighborhood, we plot the number of listings in the different neighborhood groups, with a pivot on the type of room (Figure 4). We observe a common trend across neighborhood groups: most listings seem to be either in the Private Room category, or the Entire home/apt category, while the number of listings in the Shared Room category are very small. This could indicate that home owners prefer to list out their properties as a whole as opposed to splitting them up into shared units.

We note that while there are only 5 neighbourhood groups, they are spread across 221 neighborhoods. To get a sense of the most common neighborhoods in our data, we plot the number of listings for the top-10 neighborhoods (Figure 5). We depict the number of listings on the Y-axis and neighbourhoods on the X-axis. One observation which is consistent with our prior analysis, is that the shared room type of listing is scarcely represented in the neighborhoods we visualize. Additionally, among the top-10 neighborhoods with maximum listings, only 2 neighborhood groups are represented — Manhattan and Brooklyn, which is expected since Manhattan and Brooklyn are the most *traveled-to* destinations in NYC, from a

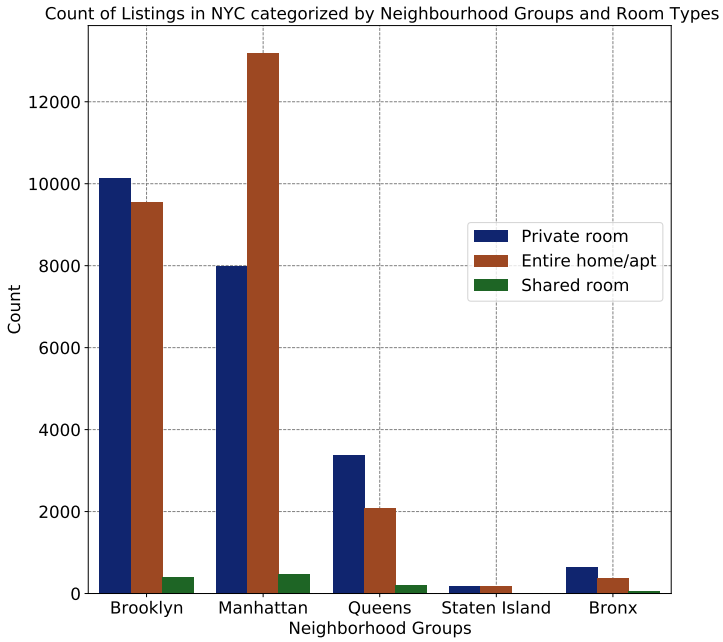


Figure 4: Number of listings categorized by Neighbourhood Group and Room Type: Most listings are in either Private Room or Entire home/apt category

tourism perspective. Further focusing at the neighbourhood-level, we observe that Bedford-Stuyvesant and Williamsburg are the most listed neighbourhoods in Brooklyn, while Harlem is the most listed in Manhattan.

We also visualize the distribution of prices in Figure 6. We observe that the prices follow a long tailed distribution. Accordingly, we filter for prices less than $800.

To observe the spread of prices conditioned on certain categorical attributes, we generate violin plots for pricing, considering neighborhood groups and room type in Figure 7 and Figure 8, respectively. Violin plots essentially present summary statistics (like box plots) — the central white dot represents the median value and the thick grey lines indicate the inter-quartile range.

From Figure 7, we observe that Manhattan has the greatest price range with a median price of $150, succeeded by Brooklyn with a $90 median price. Staten Island and Queens demonstrate very similar median prices, with Bronx being the cheapest neighbourhood group. This corroborates the perception that Manhattan has the highest standards of living and a large number of luxury property rentals.

To further analyse the spread of prices w.r.t types of rooms, we generate a similar violin plot for the room type attribute (Figure 8). Here, price distributions (conditioned upon room type) indicate that shared and private rooms have similar values for central tendencies and dispersion, and their prices do not deviate much from their central tendencies. However, houses and apartments show way more variation, reaching much higher prices, which is expected.

Given the latitude and longitude information for each listing, we can precisely plot a property on the map of New York City, and in turn, visualize the distribution of property prices. In Figure 9, we use a colormap to capture the price. Concurrent with previous observations, we see that the higher priced properties are located in the neighborhood groups of Brooklyn and Manhattan.

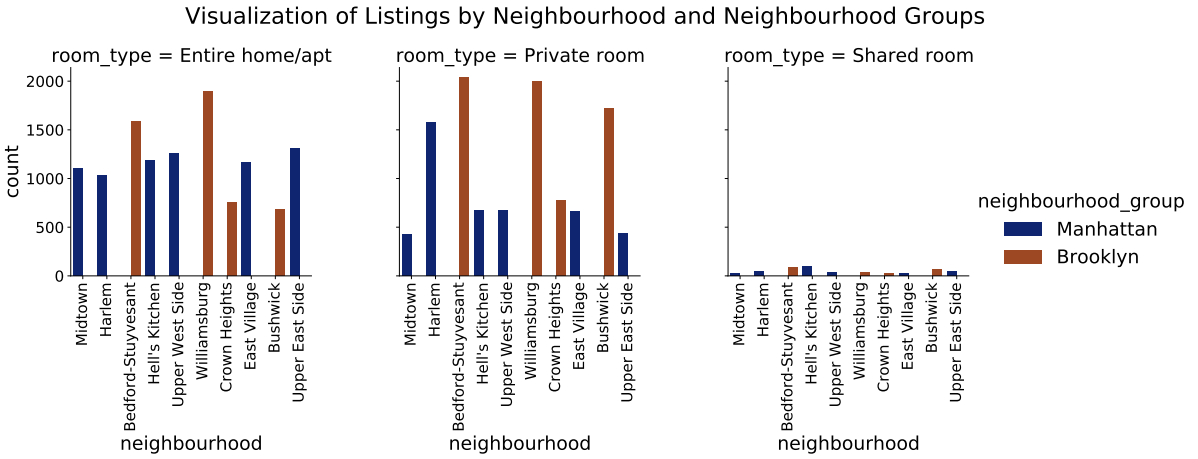


Figure 5: Number of listings based on neighbourhood groups and granular neighborhoods: Only 2 neighborhood groups (Manhattan, Brooklyn) are represented when considering the top 10 neighborhoods; compared to entire homes/apartments & private rooms, listings for shared rooms are scarce

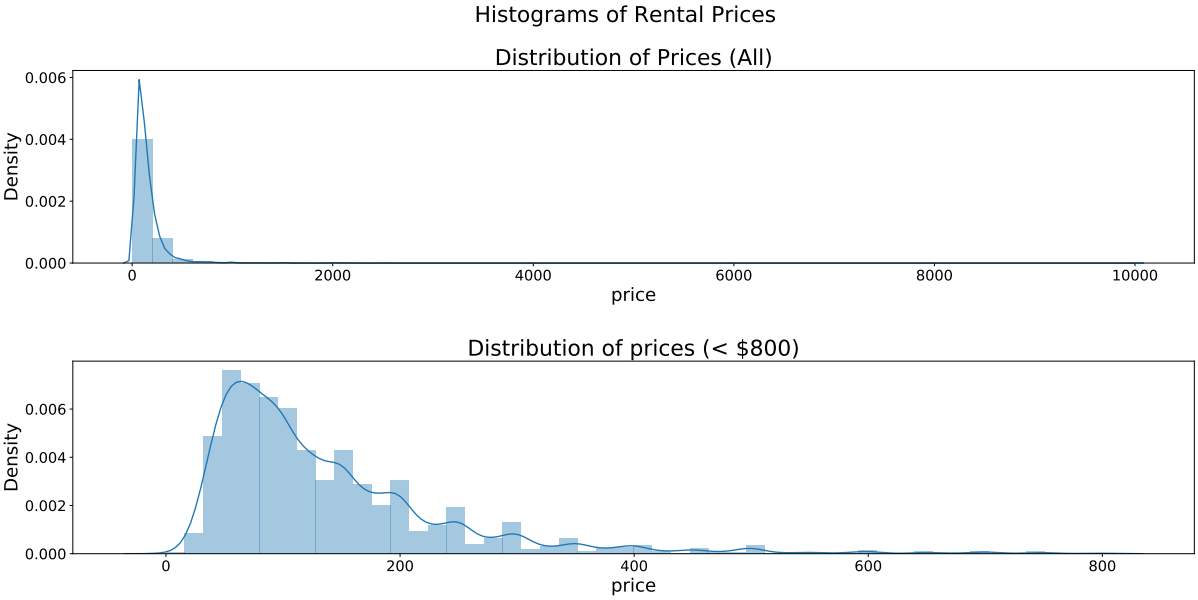


Figure 6: Histogram of price distributions: considering all possible prices (above), and prices below threshold of $800 (below); Listing prices resemble a long tailed distribution

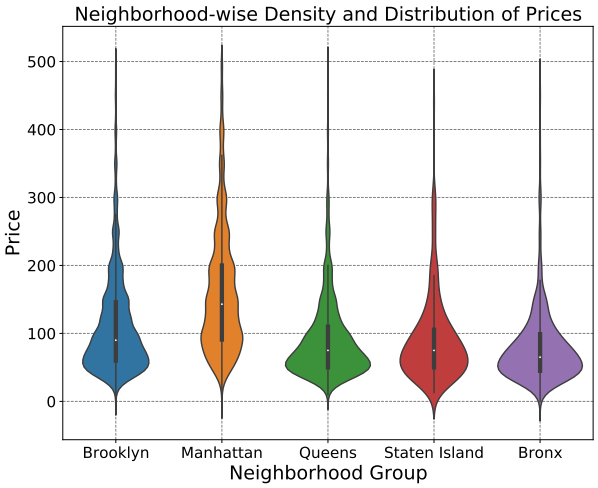


Figure 7: Neighborhood group-wise distribution of prices: Manhattan has the greatest median price (and dispersion), followed by Brooklyn; Staten Island & Queens have similar median prices; Bronx records the lowest median price

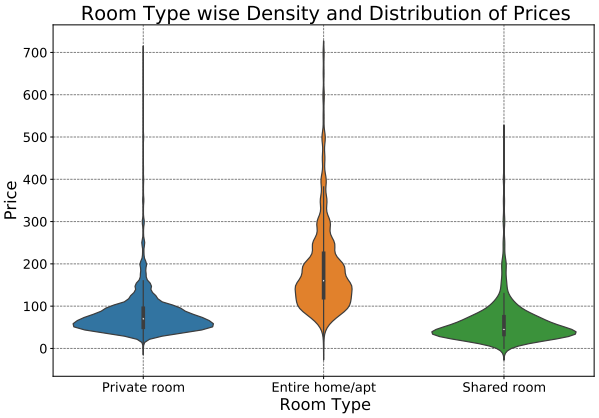


Figure 8: Room type-wise distribution of prices: Shared and Private rooms have similar medians and dispersions (prices do not deviate much from median price); prices of apartments/houses show more variation and reach much higher values

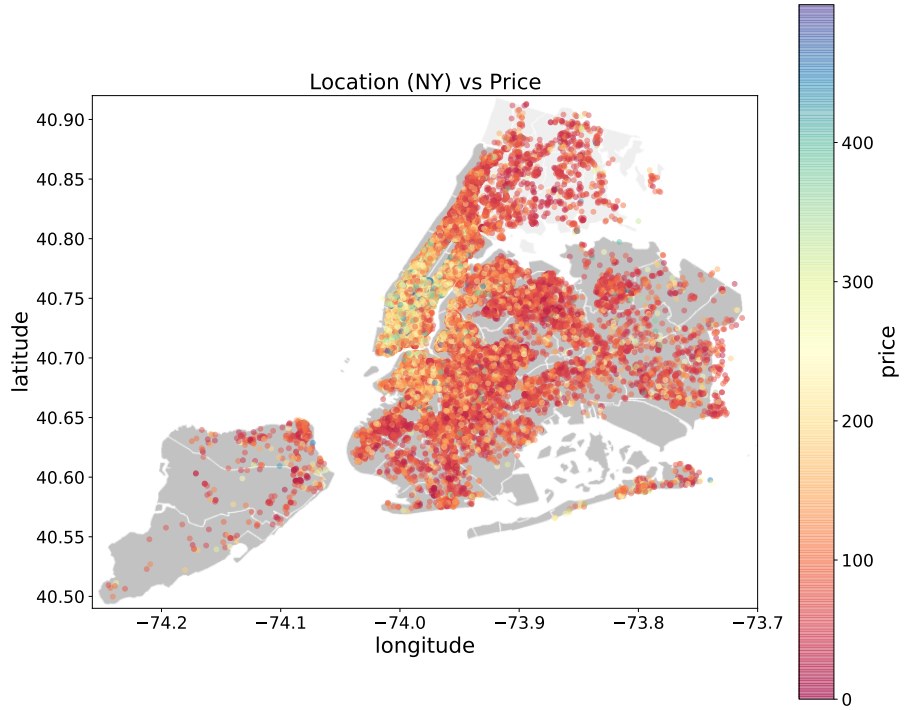


Figure 9: Geographic location on the NYC map versus Price: Brooklyn and Manhattan have an overall higher rental price compared to other neighbourhood groups

1.3 Empirical Results from Predictive Models

In this section, we report evaluation metrics for some of our ablation studies and tabulate the candidate hyperparameters for each of our ML models.

In Table 7, we list out the candidate hyperparameters that we tried out for specific ML models in our grid searches.

In Table 8, we have ablation study results for usage of the available textual field as an input feature. The column TF captures whether the textual feature was utilized or not. For most models, we see a drop in RMSE and an increase in R2 when the textual feature is introduced.

We report results for our Outlier Removal (OR) ablation study in Table 9. Here, the column OR captures the type of Outlier Removal method being employed. 'None' indicates that no outlier removal was performed. For most models, we observe that the best results on the held out test set are achieved when no outlier removal is performed. Note here that we only remove outliers from the train split prior to training. However, our test split remains invariant.

1.4 Experimental Details

We used scikit-learn's[14] implementations of clustering and regression models. We used Yellowbrick's implementation[15] of elbow-finding to pick the optimum K in our K-means clustering experiments. We encoded textual features using scikit-learn's standard TF-IDF (Term Frequency - Inverse Document Frequency) vectorizer with stopword removal turned on and generated 32-dimensional feature embeddings. For TF-IDF, we considered a maximum N-gram size of 3 with L2 norm being used for feature normalization. In our best performing

[14] https://scikit-learn.org/stable/

[15] https://www.scikit-yb.org/en/latest/index.html

Model	Candidate hyperparmeters
Lasso	fit_intercept_range = [True, False] Alpha_range = [0.0001, 0.01, 1.0]
Ridge	fit_intercept_range = [True, False] Alpha_range = [0.0001, 0.01, 1.0]
SGD Regressor	Loss_range = ['squared_error','huber'], Penalty_range = ['l1', 'l2'], Learning_rate_range = ['invscaling','adaptive'], Alpha_range = [0.0001, 0.01, 1.0]
Decision Tree	max_leaf_nodes_range = [None, 5, 10, 50, 100, 500] splitter_range = ['best', 'random']
Extra Trees	N_estimators_range = [10, 50, 100, 500] max_leaf_nodes_range = [None, 5, 10, 50, 100, 500] Bootstrap_range = [True, False]
Random Forest	N_estimators_range = [10, 50, 100, 500] max_leaf_nodes_range = [None, 5, 10, 50, 100, 500] Bootstrap_range = [True, False]
Adaboost	N_estimators_range = [10, 50, 100, 500]
Bagging	N_estimators_range = [10, 50, 100, 500] Bootstrap_range = [True, False] Bootstrap_features_range = [True, False]
Gradient Boosting (GB)	N_estimators_range = [10, 50, 100, 500] max_leaf_nodes_range = [None, 5, 10, 50, 100, 500]
Histogram-based GB	max_leaf_nodes_range = [None, 5, 10, 50, 100, 500] L2_regularization_range = [0.0, 1.0]
KNN	N_neighbors = [1, 2, 5, 10, 20, 30, 40, 50, 60, 70, 80]
MLP	activation = [ReLU, tanh, logistic]

Table 7: Candidate hyperparmeters for specific ML models.

Multi-Layer Perceptron model, the 16 hidden-layer sizes were: 256, 512, 1024, 2048, 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2.

Model	TF	CV RMSE	Test RMSE	Test R2	RMSE Drop	R2 Gain
Ridge	No	0.899 ± 0.263	0.708	**0.201**	0.011	0.026
	Yes	0.891 ±0.265	**0.697**	**0.227**		
Lasso	No	0.899 ± 0.263	0.709	0.200	0.012	0.027
	Yes	0.891 ± 0.265	**0.697**	**0.227**		
SGD regressor	No	0.900 ± 0.263	0.708	0.201	0.012	0.027
	Yes	0.897 ± 0.265	**0.696**	0.228		
Decision Tree	No	1.229± 0.309	**0.692**	**0.236**	-1.549	-0.064
	Yes	0.691 ± 0.012	2.241	0.172		
Extree	No	1.004 ± 0.278	**0.712**	**0.192**	-0.027	-0.063
	Yes	0.936 ± 0.244	0.739	0.129		
Random Forest	No	0.911± 0.254	0.746	0.114	0.002	0.004
	Yes	0.907 ± 0.248	**0.744**	**0.118**		
Adaboost	No	0.769 ± 0.019	2.280	0.143	0.178	0.013
	Yes	0.821 ± 0.031	**2.102**	**0.156**		
Bagging	No	0.878 ± 0.266	0.691	0.239	0.009	0.02
	Yes	0.868± 0.265	**0.682**	**0.259**		
Extrees	No	0.890 ± 0.261	**0.673**	**0.277**	-0.768	-0.024
	Yes	0.723 ± 0.019	1.441	0.253		
GB	No	0.907 ± 0.252	0.712	0.191	0.004	0.01
	Yes	0.895 ± 0.262	**0.708**	**0.201**		
Histogram-based GB	No	0.883 ± 0.267	0.684	0.255	0.011	0.022
	Yes	0.874 ± 0.267	**0.673**	**0.277**		
KNN	No	0.899 ± 0.258	0.687	0.247	0	0.001
	Yes	0.888 ± 0.261	**0.687**	**0.248**		
MLP	No	1.164 ± 0.178	1.379	-2.026	0.570	1.985
	Yes	1.073 ± 0.148	**0.809**	**-0.040**		

Table 8: Ablation Study for utilizing the Textual Feature (TE): We show the changes in model performances upon including the available textual field's TF-IDF encoding as a feature. For most models, including the textual feature leads to performance improvements.

Model	OR	CV RMSE	Test RMSE	Test R2
Ridge	None	0.891 ± 0.265	**0.697**	**0.226**
	z-score	0.738 ± 0.023	1.476	0.216
	IQR	0.649 ± 0.009	3.056	0.121
Lasso	None	0.891 ± 0.265	**0.697**	**0.227**
	z-score	0.738 ± 0.023	1.478	0.214
	IQR	0.649 ± 0.011	3.059	0.119
SGD Regressor	None	0.891 ± 0.265	**0.686**	**0.228**
	z-score	0.801 ± 0.206	1.477	0.215
	IQR	0.651 ± 0.011	3.058	0.121
Decision Tree	None	1.229 ± 0.309	**0.692**	**0.236**
	z-score	0.743 ± 0.022	1.475	0.217
	IQR	0.653 ± 0.008	3.068	0.114
Extree	None	1.004 ± 0.278	**0.712**	**0.192**
	z-score	0.767 ± 0.021	1.505	0.185
	IQR	0.671 ± 0.009	3.095	0.098
Random Forest	None	0.907 ± 0.248	**0.744**	0.118
	z-score	0.694 ± 0.019	1.421	**0.274**
	IQR	0.626 ± 0.011	3.046	0.126
Adaboost	None	3.928 ± 2.199	**1.038**	-0.716
	z-score	0.983 ± 0.076	1.571	**0.113**
	IQR	0.745 ± 0.023	3.115	0.087
Bagging	None	0.868± 0.265	**0.682**	**0.259**
	z-score	0.683 ± 0.021	1.438	0.256
	IQR	0.623 ± 0.009	3.073	0.111
Extrees	None	0.890 ± 0.261	**0.673**	**0.277**
	z-score	0.707 ± 0.021	1.447	0.247
	IQR	0.651 ± 0.009	3.058	0.119
Gradient Boosting (GB)	None	0.895 ± 0.262	**0.708**	0.201
	z-score	0.715 ± 0.021	1.462	**0.231**
	IQR	0.633 ± 0.008	3.062	0.117
Histogram-based GB	None	0.874 ± 0.267	**0.673**	**0.277**
	z-score	0.686 ± 0.019	1.425	0.271
	IQR	0.626 ± 0.008	3.039	0.131
KNN	None	0.888 ± 0.261	**0.687**	**0.248**
	z-score	0.731 ± 0.025	1.468	0.224
	IQR	0.661 ± 0.009	3.075	0.111
MLP	None	0.919 ± 0.146	**0.684**	**0.254**
	z-score	0.772 ± 0.019	1.466	0.227
	IQR	0.714 ± 0.006	3.084	0.104

Table 9: Ablation Study for Outlier Removal (OR): We show model performances corresponding to 3 settings: No Outlier Removal (None), OR based on Z-scores (Z-score), and OR based on IQR. For most models, not removing outliers leads to the best performance.