Airbnb Analysis: Leveraging Machine Learning to Predict Listing Prices

Contents

by: Karan Khubdikar, Mo Norouzi, Nicole Bidwell

Summary

In this project, our group seeks to use machine learning algorithms to predict Airbnb listing prices using various property details like geographical location, room type, and review activity. We implement rigorous methods to analyze the data and build machine learning models, including exploratory data analysis, feature engineering, cross-validation, hyperparameter optimization, and feature selection. We explore several models including Ridge, Random Forest Regression, XGBoost, and LGBM Regressor, and incorporate Recursive Feature Elimination with Cross-validation (RFECV). The highest performing model was the RFECV model which combined L1 regularization and the LGBM Regressor. It obtained a R^2 test score of **0.61** and a mean absolute error of **58.77**. Furthermore, we explored the SHAP values which provide valuable insights into feature importance and model interpretability, helping us better understand how individual features, such as location and room type, contribute to the predicted listing prices. While the best-performing RFECV reflects a reasonable level of performance, limitations and potential future improvements are outlined in the final discussion section.

Introduction

With over 8 million active listings across more than 100,000 cities and towns, Airbnb boasts an extensive network of accommodations, offering travelers a wide range of unique stays (Airbnb, 2024). As the popularity of short-term rentals remains high, understanding factors that influence listing prices becomes crucial for guests, hosts, and Airbnb. This analysis employs machine learning algorithms to predict Airbnb listing prices using property details like geographical location, room type, and review activity. By applying rigorous data analysis methods, we aim to uncover insights that not only enhance pricing strategies but also contribute to the broader understanding of market dynamics in the short-term rental sector.

Back to top

Method & Results

In this section, we provide a detailed depiction of the methodologies employed in our analysis and the key results.

Data Description

The dataset utilized for this project originates from Kaggle, available at https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data. The available information includes details about the properties on Airbnb, the corresponding geographical information, price, and the room type. It comprises 16 columns and a substantial volume of data, totaling 48895 rows. Given the large dataset size, 70% of the dataset is used for training the models while the remaining 30% is used for testing.

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399

Exploratory Data Analysis

The following exploratory data analysis techniques were used to gain insight into the data and drive the next steps in model development.

- **Summary Statistics:** Key statistics for numerical columns, like the mean, minimum, maximum, and standard deviation, were used to gain an overview of the central tendencies of the data. Observing the number of unique values within a column also helped identify the features which are unique identifiers such as `id` which do not help the model in learning any new patterns. Visualizations: -
- **Visualizations:** Observation of the distributions of numeric and categorical columns was done to identify skewness in numerical features, and detect imbalance in the categorical groups. Additionally, the pairwise correlations of numerical features were calculated to identify strong relationships between the features which are to be dealt with accordingly.

Distribution of Categorical Features

Observing two meaningful categorical columns, `room_type` and `neighbourhood_group`, we see that the listings in the dataset are primarily 'Entire Home/Apt' or 'Private Room' types, and are located in Brooklyn or Manhattan.

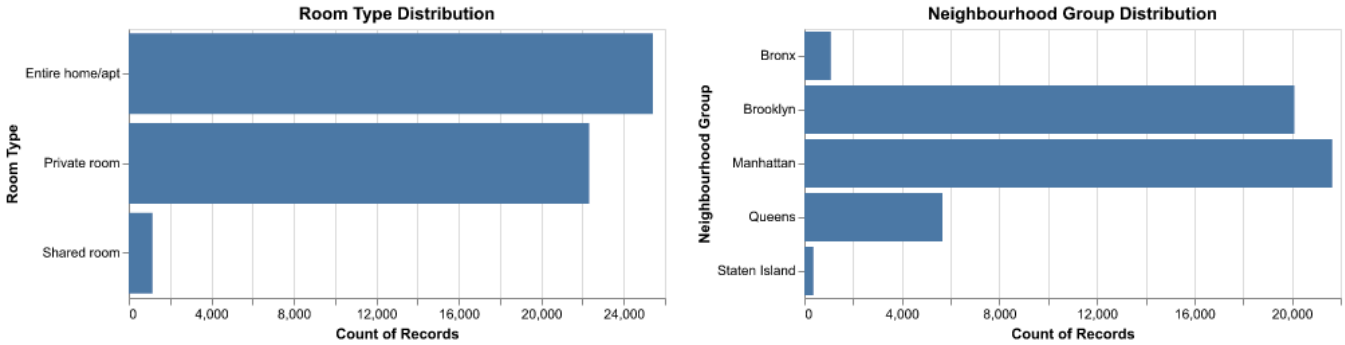

Fig. 1 Barcharts for `room_type` and `neighboorhood group` features.

Distribution of Numeric Features

The following graphs provide the distributions for the meaningful numeric features in dataset.

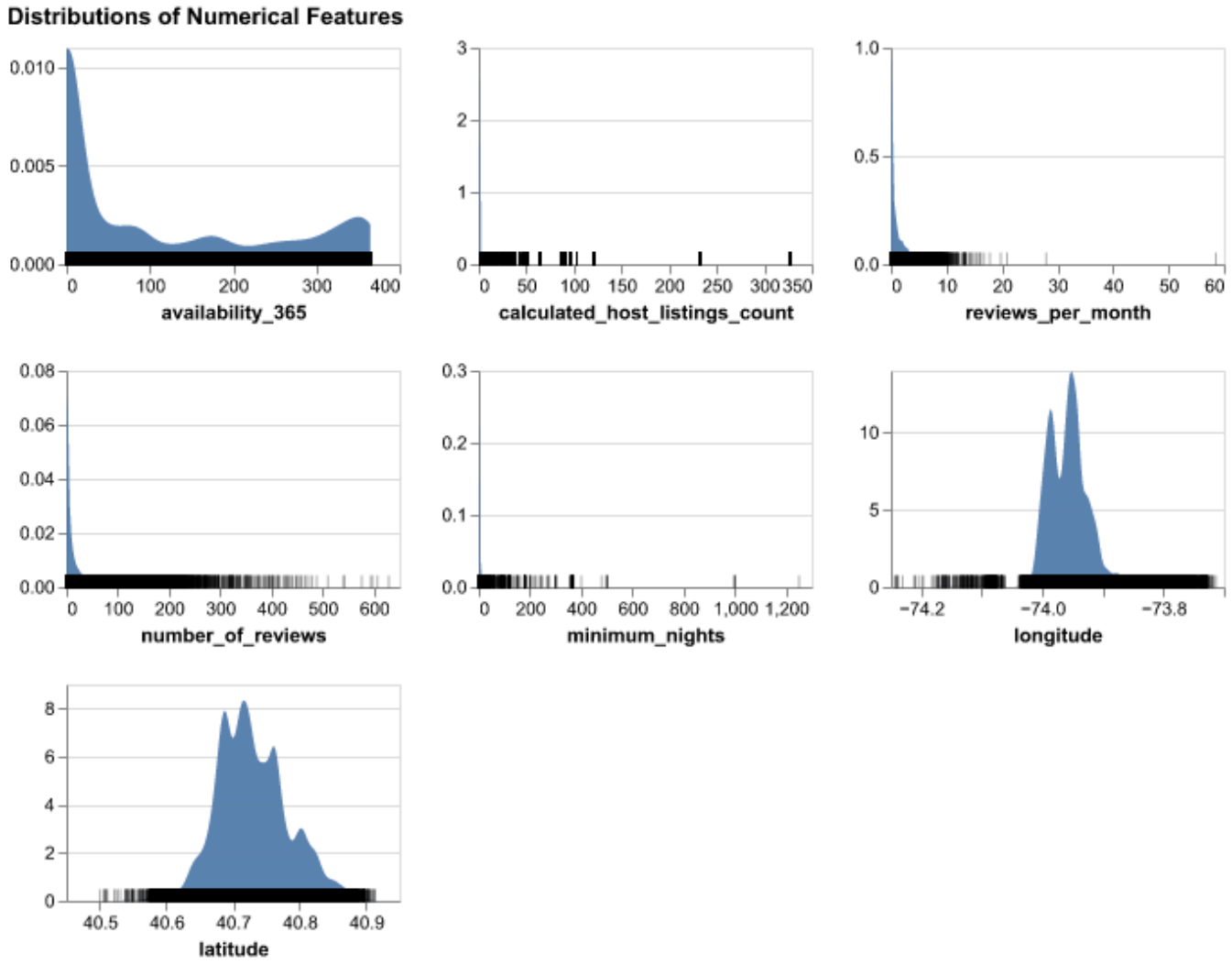

Fig. 2 Density plots portraying the distribution of numerical features in the dataset.

Many of the distributions, like `reviews_per_month` and `number_of_reviews`, are right-skewed and contain extreme outliers. Notably, `availability_365` has a significant number of values with 0. This indicates that several listings were not available to book for any day of the year, at least at the time of data collection.

Distribution of the Target Variable: Price

In this section we look in depth at the distribution of the target variable, price.

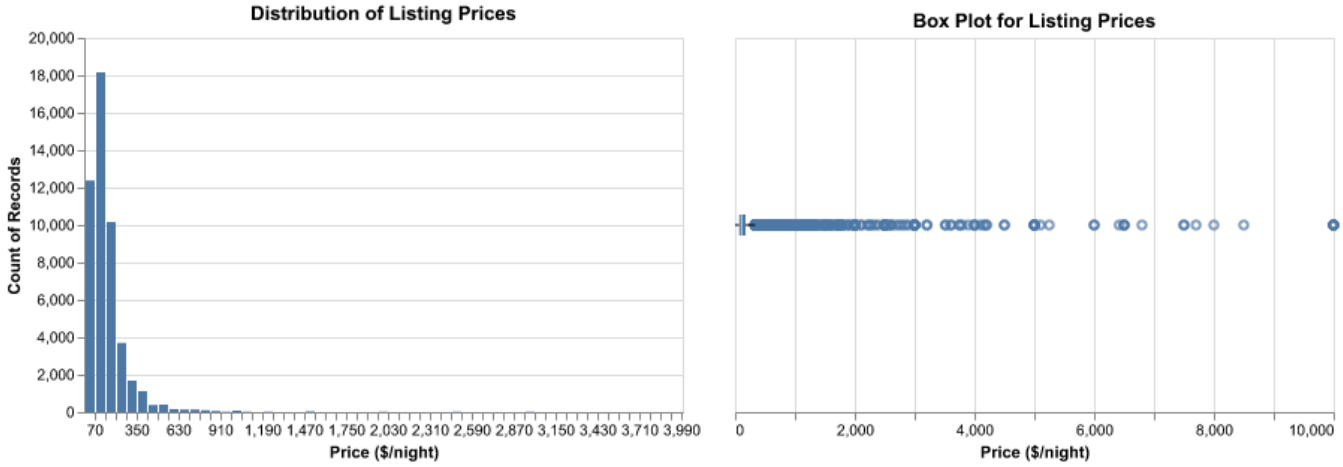

Fig. 3 The distribution of the target variable.

The distribution of price is right-skewed with extreme outliers, which can pose challenges for model prediction. Also, values with a price equal to zero may indicate an error, as a free listing price is highly unexpected. To gain more insight, we explored the price distribution, based on the features: room type, neighborhood group, and number of reviews.

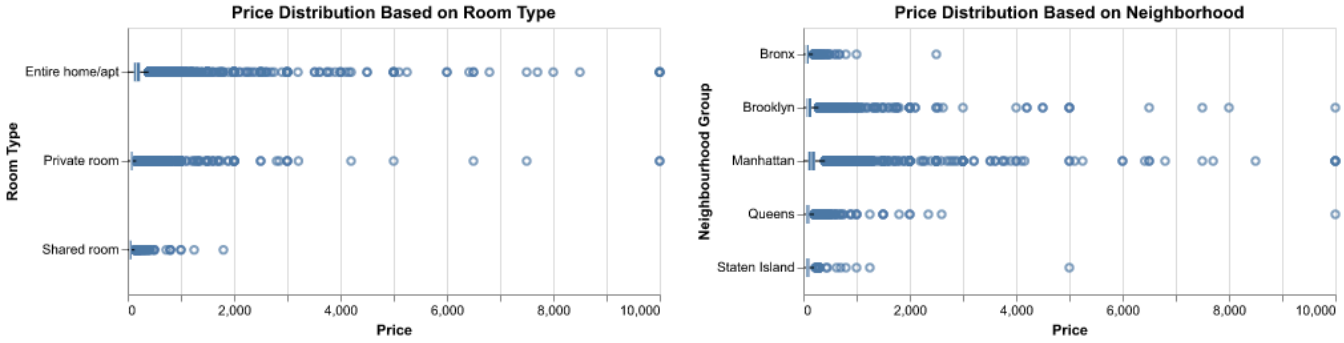

Fig. 4 The distribution of the target variable by the categories in room type and neighbourhood.

All categories display right-skewed distributions with outliers and varying degrees of spread. Notably, the 'Entire home/apt' room type has the highest median price per night of $160, and the 'Shared Room' type has the lowest at 45 per night. Furthermore, listings in Manhattan boast the highest median price at $150 per night, whereas the Bronx has the lowest at 65 per night.

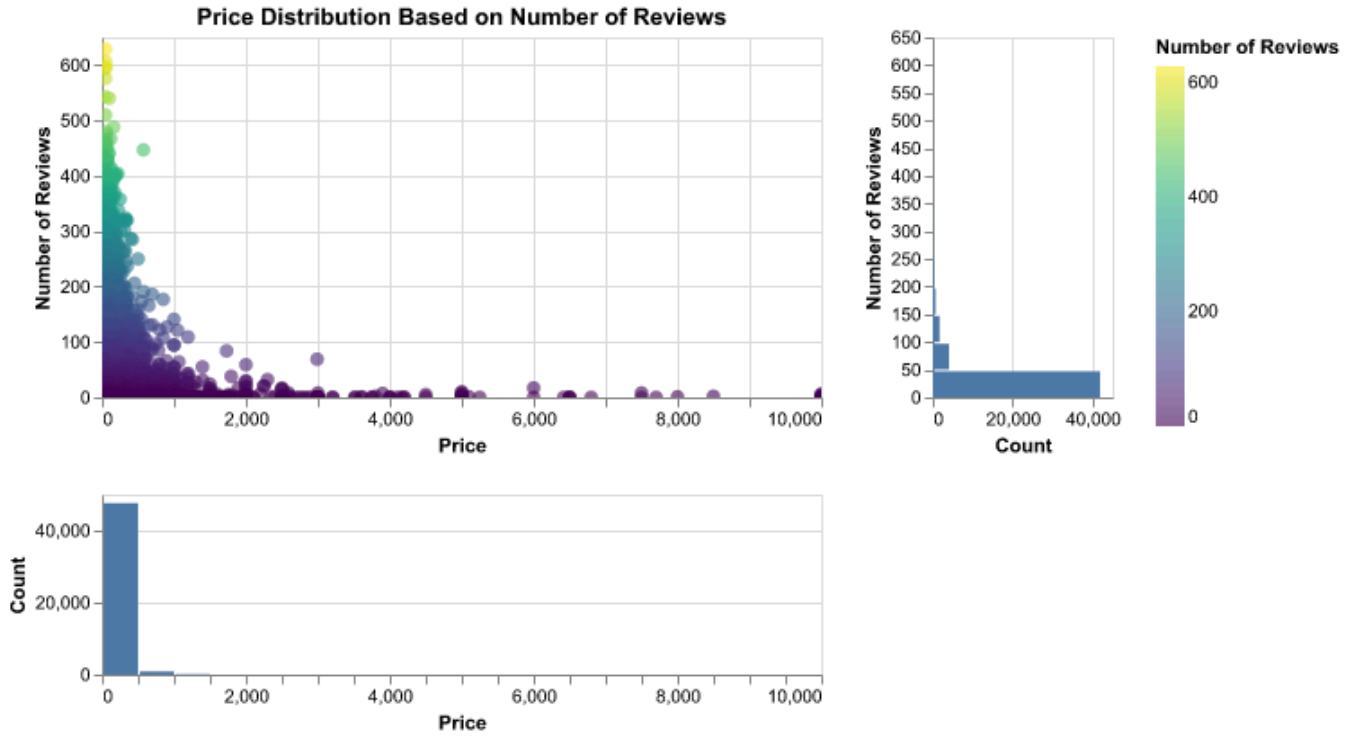

Fig. 5 The distribution of the target variable based on the number of reviews.

The scatter plot reveals a significant clustering of listings within the range of 0 to 50 reviews and 0 to 500 dollars per night. Additionally, those outlier listings with high price per night have less reviews. Likewise, there exist many listings with low price per night but a high number of reviews.

Correlation of Numeric Variables

The Pearson and Spearman correlations are calculated to gain insight into possible relationships between the numeric variables in the dataset.

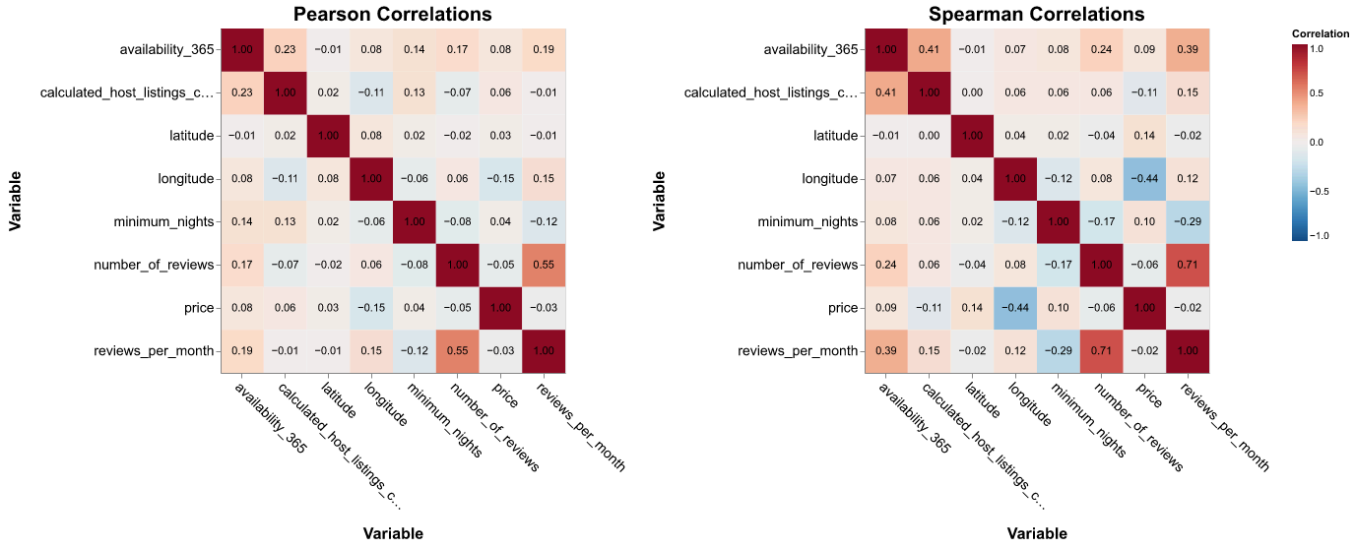

Fig. 6 The Pearson and Spearman correlations of numeric variables.

As expected we see a positive correlation for the number of reviews and reviews per month. The correlation magnitudes are not extreme for the other pairings. Interestingly, price and longitude have a -0.15 Pearson correlation and a moderate Spearman correlation of -0.44. This suggests that while there isn't a strong linear relationship, there is a moderate monotonic trend, indicating that price generally decreases as longitude increases.

Data Cleaning

The following data-cleaning steps were implemented to prepare the dataset for model development.

- **Price equal to zero:** Rows with a price equal to zero were removed, since these rows may have resulted from an error and make it more challenging for predictive models to grasp patterns in the data.
- **NaN in reviews_per_month:** For rows with NaN in `reviews_per_month`, `number_of_reviews` is zero, indicating no reviews. Thus, we replace NaN with zero.
- **NaN in last_review:** For rows with NaN in `last_reviews`, `number_of_reviews` is zero, indicating no reviews. Thus, we replaced NaN with the date 1900-01-01 to distinguish between listings that have never received a review and those that have.
- **last_review type**: Converted `last_review` to a datetime object for easier processing in later steps.

Feature Engineering

We engineered various features to enhance our models' predictive power. These engineered features are outlined below.

- `estimated_listed_months`: This feature aims to estimate how long a listing has been listed based on the number of reviews it has received and the average number of reviews it gets per month, using the formula: $\text{estimated\_listed\_months} = \frac{\text{number\_of\_reviews}}{\text{reviews\_per\_month}}$
- `availability_ratio`: This feature provides the proportion of the year that a listing is available for booking, using the formula: $\text{availability\_ratio} = \frac{\text{availability\_365}}{365}$
- `days_since_last_review`: This feature provides the number of days since the last review, utilizing information from `last_review`.
- `distance_from_city_center`: This feature provides the number of kilometers the listing is from the NYC center, using the haversine formula and information from the `latitude` and `longitude` features.

Preprocessing and Transformations

The following preprocessing and transformation steps, as justified below, were applied to prepare the dataset for model training.

Dropped Features:

- `id`, `name`, `host_id`, `host_name`: These features all have a high number of unique values, with `id` being a unique identifier.
- `last_review`: This feature was dropped since we used `days_since_last_review` instead.
- `neighbourhood`: Given the other location-based features in the dataset, along with experimentation results, dropping `neighbourhood` permitted the best results.

- `availability_365`: This feature was dropped for redundancy since we've opted to use `availability_ratio` which offers a normalized and more interpretable measure of availability.
- `number_of_reviews`: This feature was dropped for redundancy since we used `reviews_per_month` instead.

Categorical Features:

- `neighbourhood_group` and `room_type`: Both of these features are nominal values so one-hot-enconder with `handle_unknown=True` was applied.

Numeric Features:

- `latitude`, `longitude`, `minimum_nights`, `calculated_host_listings_count`, `reviews_per_month`, `estimated_listed_months`, `availability_ratio`, `days_since_last_review`, `distance_from_city_center`: Standard scalar was used on these numeric features to put them on the same scale so that no feature disproportionately influences the models due to its scale.

Skewed Target Variable:

- `price`: A log transformation on `price` was applied to reduce the skewness and make the distribution more normal, which can enhance the performance of the models.

Updated Correlations

After feature engineering and preprocessing we observe the updated Pearson and Spearman correlations, below.

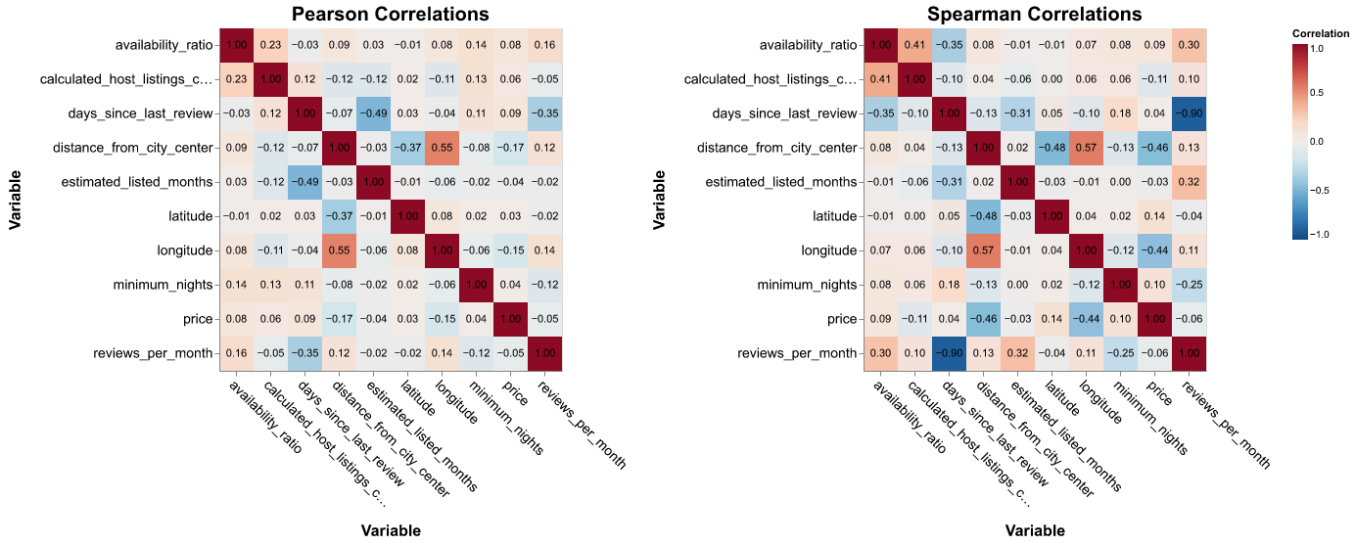

Fig. 7 The Pearson and Spearman correlations after feature engineering and preprocessing.

For the engineered features we created, there are moderate correlations for `distance_from_city_center` and `longitude`, along with `days_since_last_review` and `estimated_listed_months`. Notably, we see a strong negative Spearman correlation for `reviews_per_month` and `days_since_last_review`, suggesting that as the number of reviews per month increases, the time since the last review decreases. While these two features have a strong Spearman correlation, the Pearson correlation is only -0.34. We've opted to keep both features at this point, as they both may provide valuable information.

Model Development

The next step in our analysis was to build the model pipelines and evaluate the models. The scoring metric we used is R^2 since we compared several regression models. Additional metrics, described later, were used for feature importance and further analysis.

Baseline Model

We used the Dummy Regressor from the scikit-learn package as our baseline model. It has poor performance with a train and test (validation) R^2 score of 0.000 and 0.000, respectively.

Linear model

Ridge from scikit-learn was our first linear model which also had poor performance. The R^2 scores for the training and test (validation) sets were 0.518 and 0.517, respectively. While these results are an improvement over the baseline model and don't show signs of overfitting, there is still room for improvement.

Ensemble Models

We experimented with three ensemble models: The Random Forest Regressor, XGBoost Regressor, and LightGBM Regressor. Overall, the ensemble models showed better results compared to the linear and baseline models. A summary table with the results is provided below.

	Dummy		**Ridge**		**Random Forest**		**XGBoost**		**LGBM**	
	mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
test_score	-0.0	0.0	0.517	0.019	0.592	0.018	0.578	0.018	0.605	0.018
train_score	0.0	0.0	0.518	0.002	0.688	0.002	0.588	0.003	0.656	0.002

As observed, the top two performing models are the LightGBM Regressor, which achieved a training score of 0.658 and a test score of 0.605, and the Random Forest Regressor, with a training score of 0.695 and a test score of 0.591. The gaps in the training and test scores for these models are small, indicating minimal overfitting. XGBoost follows as the third highest-performing model, with a training score of 0.596 and a test score of 0.583. All these models significantly outperform both the Ridge Regressor and the Dummy Regressor (baseline model).

Hyperparameter Optimization

Hyperparameter optimization was then performed to fine-tune the ensemble models, aimed at enhancing the predictive accuracy. Using RandomizedSearchCV with various parameter spaces for each model, we were able to slightly improve the training and test scores for all three models. However, the LGBM fine-tuned model had the most promising results with the highest test score,**0.61**, and a relatively small discrepancy from the train score, **0.686**.

Feature selection

We decided to move forward with the LGBM Regressor and perform Recursive Feature Elimination with Cross-Validation (RFECV) and L1 regularization to optimize feature selection and improve model performance by identifying important features while reducing dimensionality. This model was at par with the hyperparameter-optimized LGMB model but showed more consistent train and test scores, as observed in the summary table of results below.

	Dummy		Ridge		Random Forest		XGBoost		LGBM		RF_Tuned		XGB_Tuned	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
test_score	-0.0	0.0	0.517	0.019	0.592	0.018	0.578	0.018	0.605	0.018	0.594	0.006	0.611	0.014
train_score	0.0	0.0	0.518	0.002	0.688	0.002	0.588	0.003	0.656	0.002	0.794	0.000	0.708	0.002

Feature Importances

Given that the top two test scores are 0.610 for the LGBM Regressor and 0.607 for the RFECV model, we proceeded with the RFECV model due to its smaller discrepancy between training and test scores. Now, we dive further into the feature importances for more insight.

The RFECV selected features, with their corresponding importance value are listed below.

Feature	Importance
longitude	881
availability_ratio	749
distance_from_city_center	674
latitude	645
minimum_nights	598
calculated_host_listings_count	517
estimated_listed_months	473
days_since_last_review	472
reviews_per_month	437
room_type_Shared room	99
room_type_Entire home/apt	74
neighbourhood_group_Manhattan	26
neighbourhood_group_Bronx	20
neighbourhood_group_Brooklyn	19
neighbourhood_group_Queens	16

Noticeably, features related to location like `longitude`, `distance_from_city_center`, and `latitude` have high importance, along with features related to availability and booking, like `availability_ratio` and `minimum_nights`. Whereas, specific neighbourhood groups are showing less importance in comparison.

Test Set Scoring

We use R^2 for consistency with our previous analysis. We also calculated the Mean Absolute Error (MAE), as it handles outliers well, and provides additional interpretability and robustness in evaluating the RFECV model.

The final R^2 test score is **0.61**. While the test MAE is **58.77**. Additionally, the MAE scores of the Linear and Dummy models are displayed below, for comparison.

Model	MAE
RFECV	58.770000
Linear	64.270000
Dummy	86.620000

The R^2 score of 0.61 tells us that the RFECV model explains 61% of the variability in the listing prices. While this is a moderate level of performance, there is 39% of the variability that is not explained. Additionally, the MAE of 58.77. tells us that the RFECV model's predictions are $58.77 away from the actual listing prices on average. This is an advancement from the Dummy and Linear model, but there is still room for improvement.

SHAP Values

We delved deeper into the analysis by looking at SHAP values to understand the influence of specific features on individual predictions. It is important to note, that due to the log transformation on our target value, the SHAP values become slightly less interpretable since the units are *log of dollars*. However, the SHAP plots serve to illustrate the relative impact of each feature on the model's predictions, providing valuable insights into feature importance and interactions.

SHAP Summary Plot

First, we consider the SHAP Summary plot which provides an overview of all SHAP values for all data points in the test set.

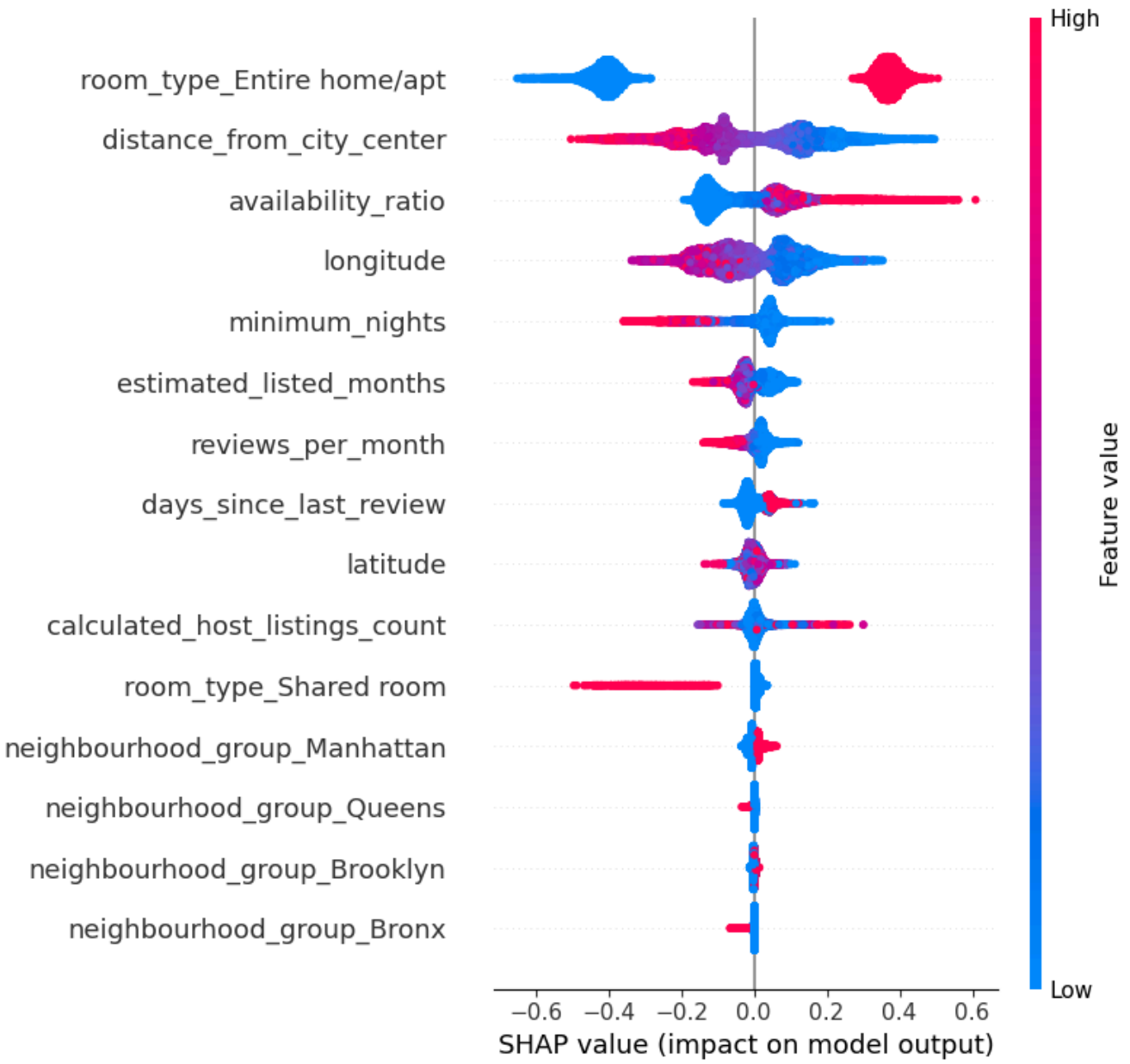

Fig. 8 The SHAP summary plot for the test set.

This plot provides evidence that characteristics like the type of room, the distance from the city center, and the availability of the listing play a clear role in predicting the listing price. For instance, the red cluster of points for `room_type_Entire home/apt` indicates that listings where users get to rent out the entire place contribute significantly to increasing the model's predicted listing price. Whereas, listings that don't rent out the entire place (indicated by the blue clusters for `room_type_Entire home/apt`) contribute by decreasing the predicted price. This interpretation is somewhat consistent with the conclusion we can gain from the clusters for `room_type_Shared room`. Likewise, the extended blue cluster for `distance_from_city_center` indicates a closer distance to the city center will contribute to higher predicted prices, and the red cluster indicates a larger distance from the city center will contribute to lower predicted prices. The remaining features can be analyzed similarly, but we see some features having less influence like `neighbourhood_group_Brooklyn`, or less clear-cut divisions, like `calculated_host_listings_count`.

SHAP Values for Individual Predictions

Next, we focus on two specific predictions (a high-value and low-value) from the test set to gain a deeper understanding of the model's predictions and the driving factors behind them.

The plot below gives insight into how different features impacted the model's prediction, resulting in a predicted value of $f(x) = 3.919$ (or $\$50.35$), which is well below the average (or expected) value of $E[f(x)] = 4.725$ (or $\$113.27$). The `room_type_entire home/apt` had the most negative impact, reducing the prediction by -0.47. Whereas, the `availability_ratio` value contributes positively by increasing the prediction by 0.3. The other features follow similarly, either increasing (those in red) or decreasing (those in blue) the prediction value.

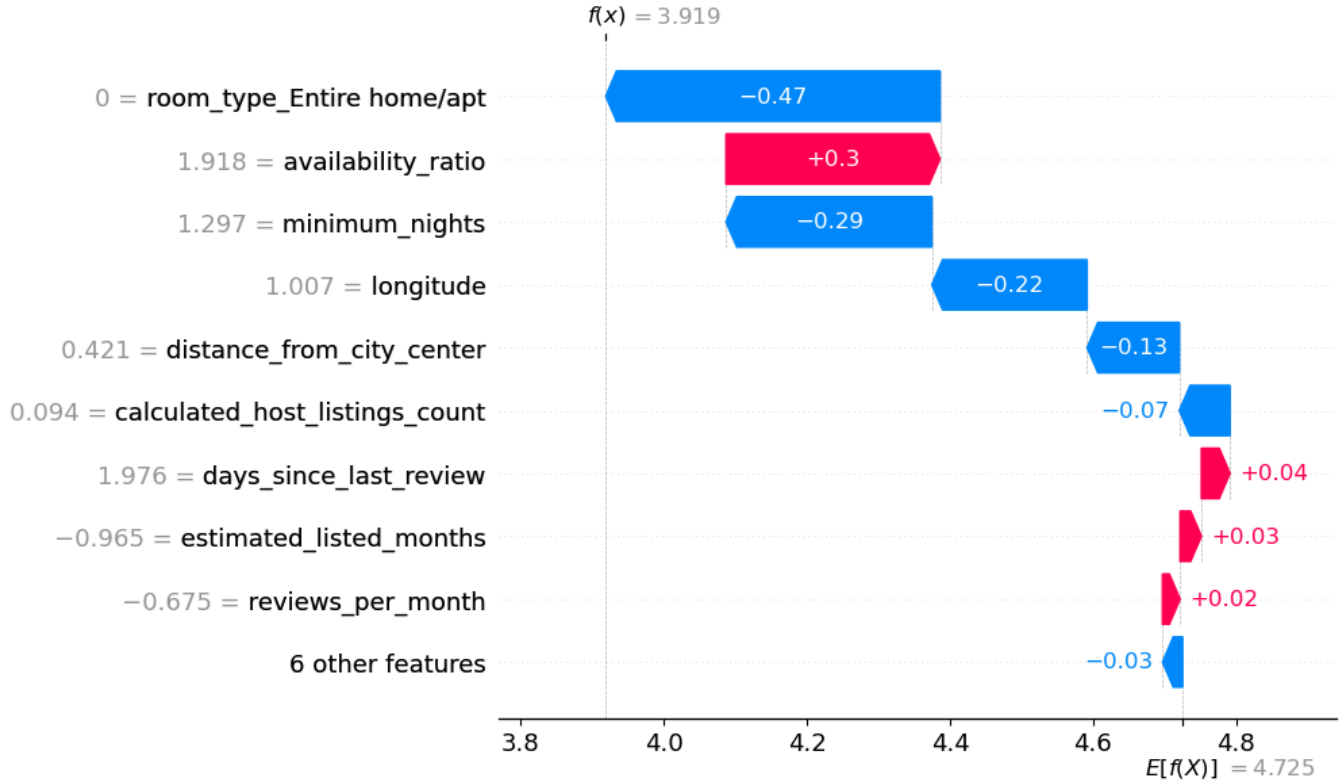

Fig. 9 The SHAP plot for a small predicted value.

In the plot below, we have an example data point with a much higher prediction of $f(x) = 5.389$ (or $\$218.98$). We see that the `room_type_entire home/apt` value has the largest positive impact, increasing the prediction by 0.41. Likewise, `distance_from_city_center` has the second largest impact, increasing the prediction by 0.21. As before, the other features had a less significant impact, either increasing or decreasing the predicted value.

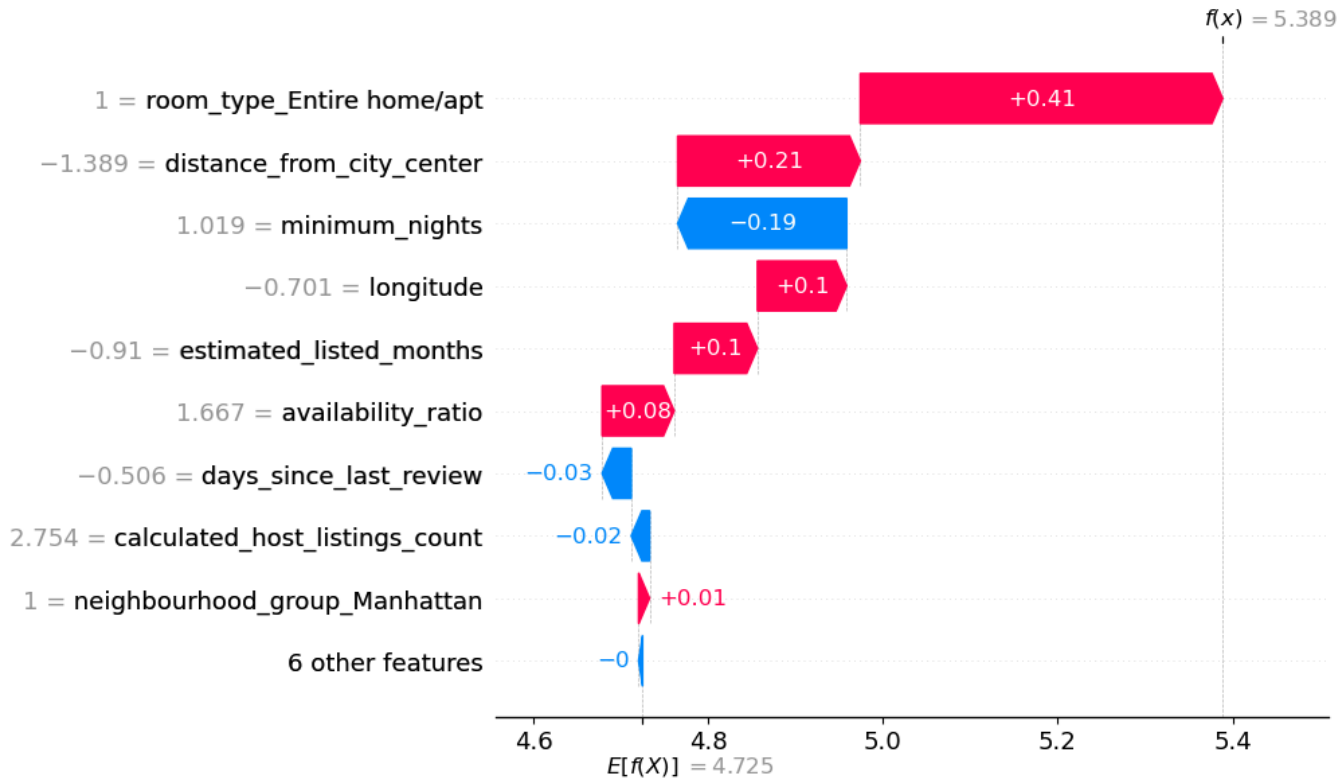

Fig. 10 The SHAP plot for a large predicted value.

Conclusions

To identify the optimal model for predicting Airbnb listing prices, this analysis explored a variety of models, beginning with a baseline Dummy Regressor and gradually increasing in complexity. Early insights from our EDA gave us a foundational understanding of the feature distributions, serving as a guide for feature engineering and model development. Engineered features, like distance from the city center, enhanced the models' abilities to capture patterns in the data—as indicated by a high feature importance value and reflected in the SHAP summary plot. After performing hyperparameter optimization and recursive feature selection we arrived at the best-performing RFECV model.

The RFECV model utilizes recursive feature elimination to optimize feature selection and reduce dimensionality while incorporating cross-validation. From the hyperparameter optimization results, we decided to implement RFECV with the top-performing LGBM Regressor, along with L1 regularization to prevent overfitting. The RFECV model achieved an R^2 test score of 0.61. This score reflects a moderate to good fit and is relatively consistent with the 0.65 train score. However, a score of 0.61 reveals that 39% of the variability in listing price can not be explained by the RFECV model, leaving room for improvement.

Additionally, the RFECV model achieved an MAE of $58.75. While this is an improvement from the dummy and linear models, it too signifies a sub-optimal level of performance, when we consider the median listing price of $106 in the test set. Despite this, insight from the feature importances and SHAP values provides quantifiable measures for the impact the different features have on the model's predictions. Overall, room type, location features (like longitude and distance from the city center), and minimum nights, appear valuable in predicting the the listing prices.

Limitations and Future-work

Despite the thorough methods implemented in this analysis, the sub-optimal model performance can be attributed to a few factors. Firstly, 36% of the listings had listed 0 available days for booking. With no other information on why these listings were unavailable or how long they have been unavailable, the model can not capture external factors that could be at play for these listings. For instance, if a listing has been unavailable for a long time its listing price could be outdated. This uncertainty on such a large portion of the data causes lots of noise, making it challenging for the models to understand patterns in the data. More information on these listings would allow for appropriate handling of these data points.

Furthermore, the outliers in the target variable (price) pose challenges to model performance. 6% of listing prices were greater than 1.5 times the interquartile range. These outliers can skew the model performance by distorting the patterns in the data. While tree-based models are generally less sensitive to outliers, the large amount of outliers can still negatively impact model performance. Comparing model performance after removing outliers or exploring additional ensemble methods, like stacking, could improve results. Lastly, it's notable that a very small portion of the listings had a price listed as 0—indicative of errors in the data. While this portion is small it raises concern with data quality.

In conclusion, while this analysis remains robust in its efforts to predict Airbnb listing prices, the presence of data limitations and outliers requires further refinement. Future work could focus on exploring advanced ensemble models, using updated data, and ensuring data quality.

References

Airbnb. (2024). *About us*. URL: https://news.airbnb.com/about-us/

Core Libraries

Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment.* Computing in Science & Engineering, 9(3), 90-95. URL: https://matplotlib.org/

Lundberg, S. M., & Lee, S. I. (2017). *A Unified Approach to Interpreting Model Predictions.* In Advances in Neural Information Processing Systems (NIPS 2017). URL: https://shap.readthedocs.io/en/latest/

McKinney, W. (2010). *Data Analysis with Python and Pandas.* URL: https://pandas.pydata.org/

Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825-2830. URL: https://scikit-learn.org/

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). *The NumPy Array: A Structure for Efficient Numerical Computation.* Computing in Science & Engineering, 13(2), 22-30. URL: https://numpy.org/

Vega, J., & Altair Development Team. (2017). *Altair: A Declarative Statistical Visualization Library for Python.* URL: https://altair-viz.github.io/