NYC Airbnb Data Analysis Project

By: Afif Shomali: shomali2, Krish Naik: krishn3, Humza Ahmad: humzada2

Introduction:

Airbnb designates top-rated hosts as "superhosts," granting them benefits such as greater visibility, increased guest trust, and higher earning potential. Understanding what drives superhost status and booking volumes is crucial for hosts aiming to improve their listings and for Airbnb to refine its platform algorithms. This project investigates the key factors influencing these outcomes by analyzing Airbnb listings and incorporating other factors within the NYC area.

Our Goals:

- Predict Superhost Status: Develop a model to identify factors influencing superhost designation, such as reviews, listing details, and location data.
- Forecast Number of Bookings: Build a model to predict bookings, identifying variables that help hosts attract more guests and maximize profitability.
- Feature Augmentation: Enhance the analysis by integrating data from reviews (e.g., sentiment analysis), NYC Crime (e.g., crime-based metrics), and Income (e.g., pricing and occupancy rates) to better understand location-based influences.
- Data Visualization: Create visualizations to explore trends, highlight key statistics, and uncover relationships between variables in the dataset.

By offering data-driven insights, this analysis helps hosts optimize features of their listings to achieve superhost status and attract more bookings.

Methods:

Datasets:

We sourced the Airbnb listings & reviews data from InsideAirbnb, which routinely scrapes Airbnb data from a variety of cities, we used New York City data from May to September 2024. The listings datasets had 40k rows & around 75 columns, while the reviews contained around 1.1 million reviews. Our crime dataset was pulled from NYC OpenData, a free public bank of numerous datasets from agencies all across New York City. The dataset includes over 400k complaints of valid felonies, misdemeanors, and violation crimes reported to New York City Police authorities. Each complaint had details regarding the location, type of crime, date/time, and demographics. Our last dataset was from Kaggle but was created by the Department of Finance to monitor rental incomes of over 3700 properties across the city.

Airbnb Data Preprocessing & Feature Engineering:

For the Airbnb data, we started by merging the monthly datasets into one large one. Next, we removed a few unnecessary columns which wouldn't be relevant to the modelling like listing url, host photo url, and other similar columns. Since this was raw scraped data, there were a lot of missing values, around 40 columns had missing values. For numerical variables like price, we used grouped mean replacement after grouping by borough as we thought this would be a more accurate way to reflect missing price values rather than using the global average. For string columns, like host about, we filled NA with the empty string or "Blank" so during the sentiment analysis we could just assign scores of zero (neutral sentiment) to the host with missing abouts.For some other columns we used data from other columns to fill or estimate them. An example was the number of bathrooms, we used the bathroom text column to pull out the missing values for the numeric bathrooms column. Next, we converted the datatypes of columns that were supposed to be boolean or numeric but were stored as string values.

Next we moved to feature engineering which involved converting list columns, sentiment analysis, estimating bookings and an attempt at scoring listing photos. First we converted 3 list columns: host verifications, license and amenities. The first two list columns were simple as they had 3 unique values each which were converted into boolean/dummy variable columns. Host verifications were made into whether a host was verified by email, phone and work email, while license was made into Has license, Exempt or no license. The amenities column involved a bit more work as there were 7k unique values in these lists, but a lot of these values just referred to the same general amenity. After looking at these values, we chose a subset of popular amenities to make into boolean columns. Examples include, parking, wifi, TV, pool, kitchen, and more.For each row, we searched for related keywords within each element of the amenities list to determine whether a listing contained that amenity.

Then, we feature engineered sentiment ratings for reviews as well as some of the text based columns from the listings like description, title and host about. To do this we utilized the Vader Sentiment Intensity Analyzer compound score which ranges from -1 (negative sentiment) to 1 (positive sentiment), with zero indicating a neutral sentiment. For the listings text based columns, if the column had the string value "Blank" it was automatically assigned a score of 0. For the reviews data, due to it being scraped directly from Airbnb, around 100k reviews ended up being in different languages, but Vader only works in

English, so we wanted to translate these reviews. However, we ended up running into issues with translation with the two methods we tried. We first tried to translate using the Easy-NMT library which contained a variety of pretrained models with a relative straightforward interface. However, we ran into runtime issues when trying to translate such a high number of reviews, we initially tried using parallel GPUS on kaggle but these kept crashing due to not having enough memory to store the models. After this I used a google cloud GPU with around 24gb of memory, but this needed around 24 hours to translate the reviews, and it ended up crashing after running into a language that the model did not recognize/could not translate to english. Next we tried to use the google translate API, however I ran into limit issues when sending requests to translate, and I didn't have enough GCP credits for the paid API. So instead we settled on just dropping the non english reviews using a language detection library, we don't think leaving out the non english reviews ended up being an issue since we still had 1 million reviews.

Finally, we tried to create a photo scoring metric using an API by Everypixel labs, called UGC (User generated Content) photo scoring which would rate photos on technical qualities like Noise, Contrast & Brightness. However after testing on around 1k randomly sampled listings we found essentially zero correlation between the metric and potential response variables. We considered using NIMA (Neural Image Assessment) as an alternative, but any pre-trained implementations didn't take photo urls, so we would need to download around 100GB of images, due to time constraints we didn't use photo scoring for modeling.

Crime Dataset Preprocessing & Feature Engineering:

Our first concern with the crime dataset was the sheer size of the dataset, containing over 400,000 complaints dating back to the year 1014, with a majority of rows taking place in the past two years. The large size would often slow down other processes and could not be pushed to our repo directly due to size constraints.

To reduce the size and dimensions of our dataset, we first filtered out columns we saw unfit for our goals. Seeing that our goal was to build a model effective in classifying superhosts and predicting bookings, we knew that we only required variables that primarily focused on the location of crimes. This

means we can remove variables related to jurisdiction, department, time, patrol, demographics, and other policing related features. We then followed up by removing null values. We used direct and indirect methods to do so, using functions such as dropna(), but also reclassifying certain values as null and removing those as well. Finally, we needed to trim the number of observations in our dataset by a great amount, and the best way to do so was to set a threshold for the date in which a complaint was issued. We set that threshold to be June 6th, 2024 and converted date related variables into a datetime column that can be used easily to filter our older complaints. Thus, we successfully reduced the number of observations to 182810 complaints

To ready our dataset for further EDA and training, we wanted to compare and analyze potential features for our training. We wanted to look at the types of crime that occur the most in each borough, but realized that there were over 50 descriptions of crimes in the offense_description column. To consolidate the numerous types of crime in the dataset, we created a mapping to generalize the various types of crime featured. For instance, assault, robbery, homicide would all be converted into “violent crime”, while the many forms of larceny we encountered would be converted to “property crime”. We eventually converted 53 unique descriptions into 9 specific categories. This ensured easier accessibility and visual clearance for our graphs and maps.

We then constructed our bar graph and map, both of which visualized crime in each of the boroughs. From our bar graph, we realized that the total population in each of the boroughs, specifically Staten Island, played a large role in the total amount of crime each borough saw. To resolve, we created another variable crime_rate, which was essentially the total number of crimes reported divided by the 2023 reported population of each of the boroughs. We had grouped our dataset by boroughs and now had the total crime, crime rate of each borough, and the most common type in that borough. We finally merged this dataset with our listings and rental income dataset, joining the borough. Now, for each listing in the dataset, it has the crime rate and most common crime for the borough that the listing resides in.

Income Data Preprocessing & Feature Engineering:

The income dataset did not pose much of a problem in the ways of missing values with only 76 out of the 22,000 rows containing missing values and there being no noticeable trend in these missing observations, we opted to simply drop these rows and move forward. Our next big hurdle when dealing

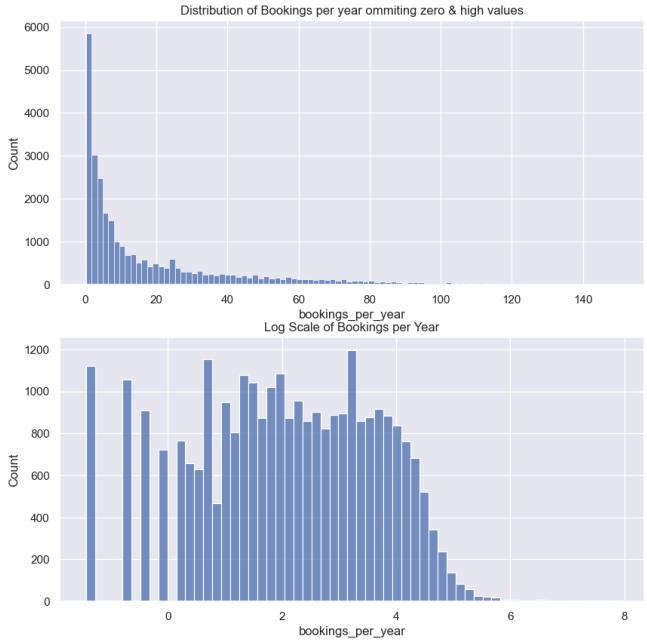
with this data was finding the associated borough for each individual observation as the borough would be an important field for merging the datasets. In its original state the data contained the associated neighborhood and Borough-Block-Lot which was in numeric for (e.g. 1-00015-7501) but given that the first number in the block lot was associated with one of the five boroughs, we created additional columns to map the borough number and then by utilizing certain neighborhoods we mapped the borough numbers with their actual borough names.

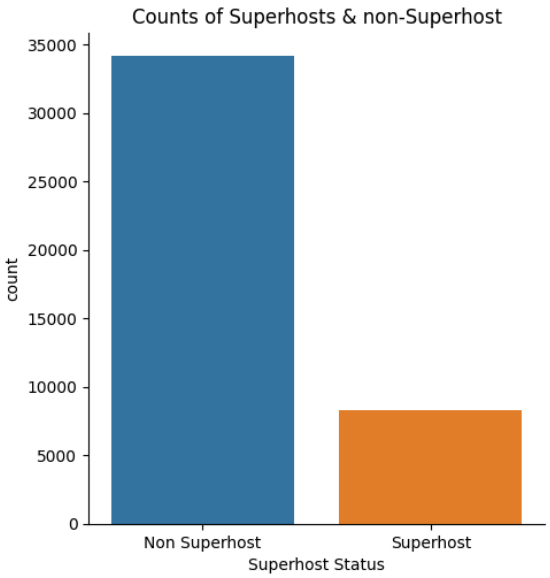
The last bit of data processing that was necessary was done on the neighborhood column, as although the borough names would be useful for filling missing values in the merged data, the neighborhood column was the key we planned to use for merging the data. The neighborhood column needed additional formatting to be merged, so we developed and used a function to change case, select only one neighborhood from observations with more than one listed (e.g. CASTLE HILL/UNIONPORT) and removed subsections markers where appropriate (e.g. upper, lower, east, etc.)

Results:

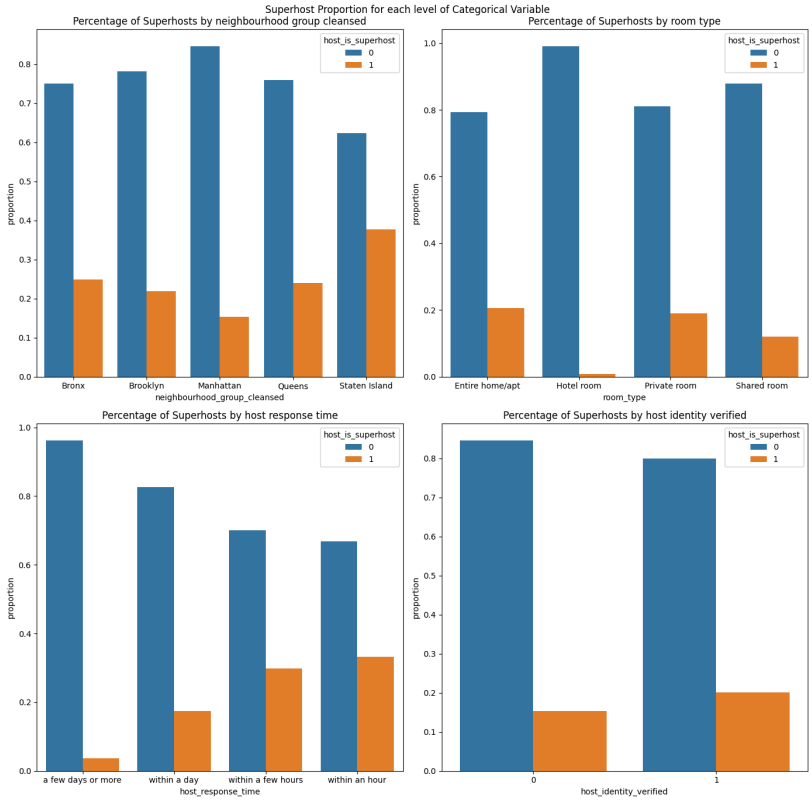
EDA Results:

For the Airbnb data, here are the distributions of the response variables:



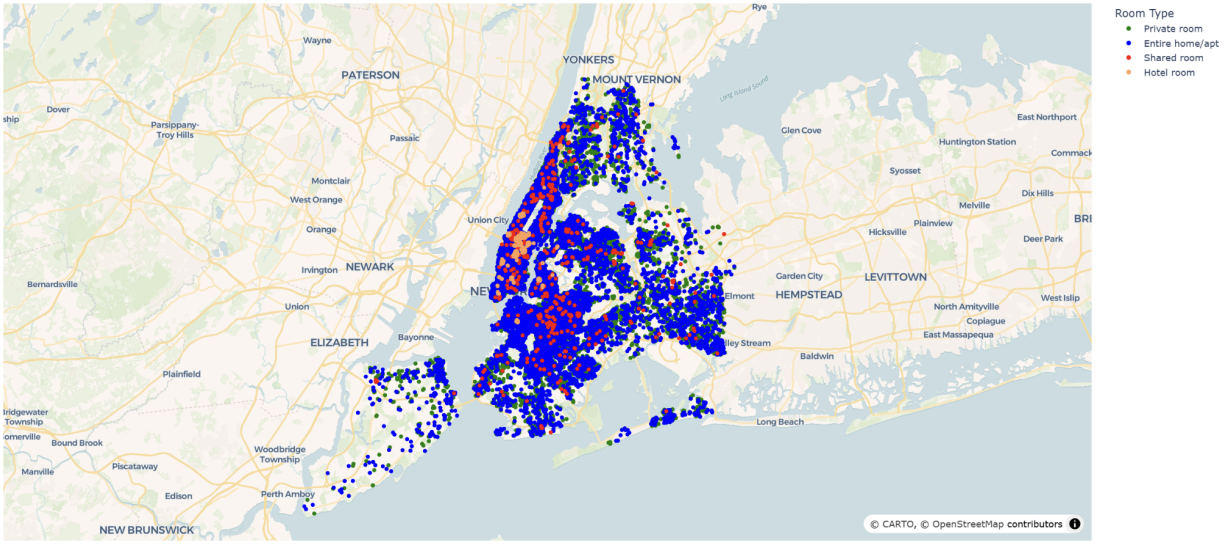


We see that the superhost variable is unbalanced which we will handle during modeling, and the numeric variable is right skewed so a log transformation is applied to make the distribution more normal. Below we have the superhost proportions for levels of different categorical variables, we see that hosts who respond faster are more likely to be superhosts and Staten Island hosts are more likely to be superhosts.

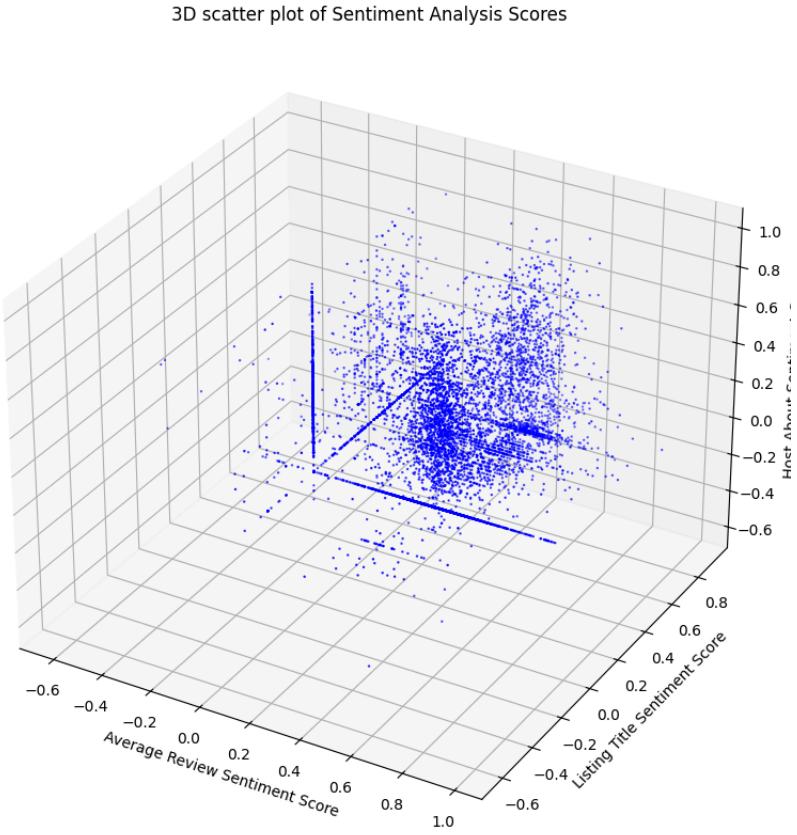


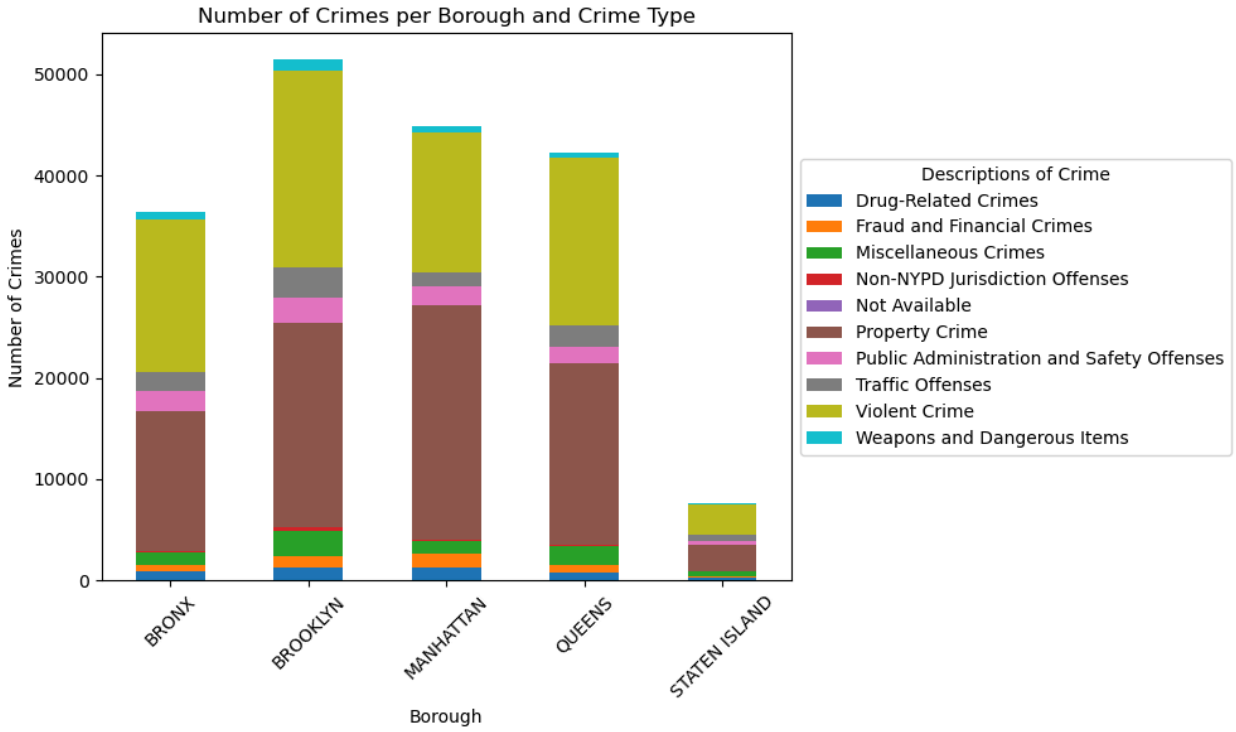
Here are the room types of the listings on a map of NYC, an interactive version is in the code:

Airbnb Listings in NYC colored by Room Type

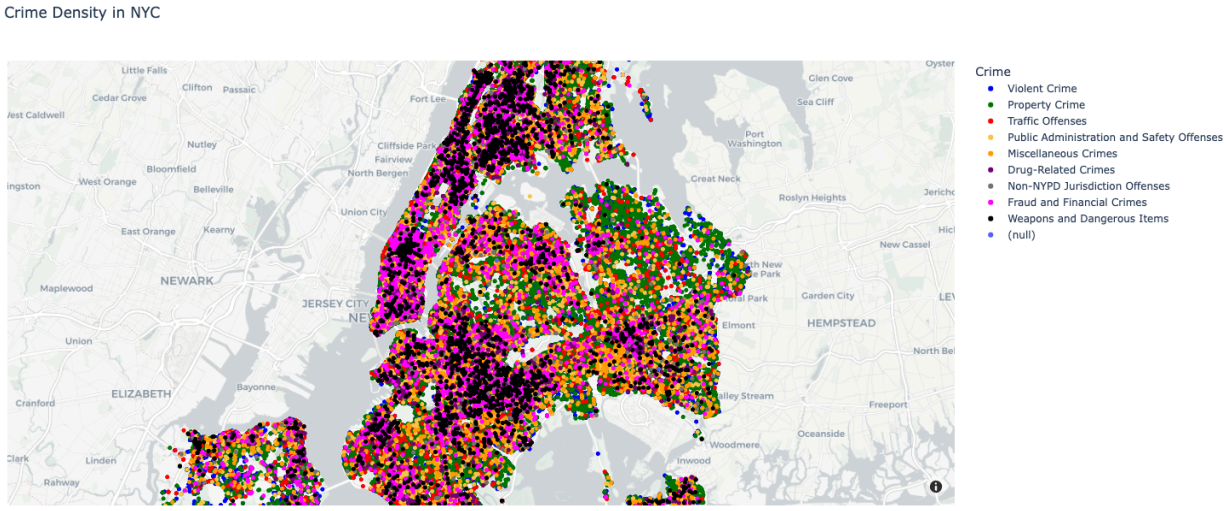


Here is a 3D scatter plot of sentiment scores, with scores for reviews, titles & host abouts:





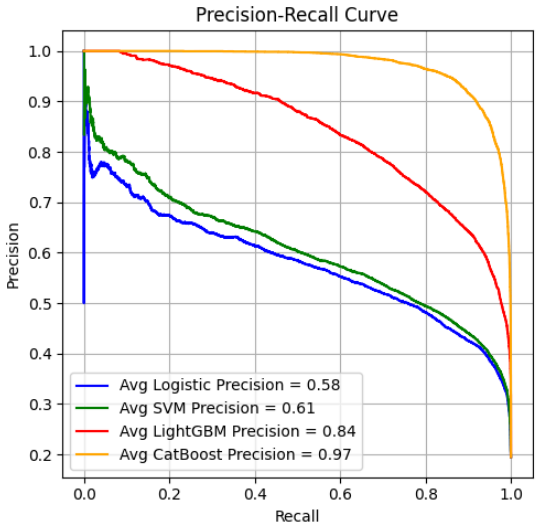
Here is a bar graph with the number of crimes in each borough split by the descriptions of each crime. Staten Island is significantly less, but this is also attributed to population, which we address.

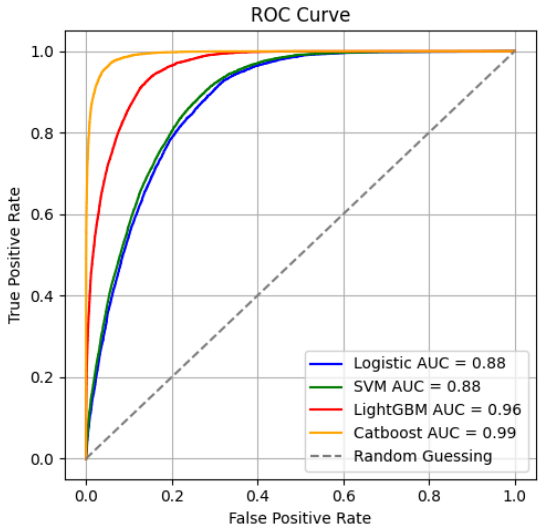


Here is a map of each of the 5 boroughs and every complaint from June 1th, 2024 to September 30th, 2024.

Superhost Classification Results:

To classify Superhost, we tested 4 different models: Logistic Regression, Linear Support Vector Classifier, Light GBM & Catboost. Our goal was to get a model with high Recall and reasonable precision. For each of these models we use Random Search Cross Validation (CV) to tune the hyperparameters. To assess model performance, we used Stratified 5 Fold CV which was to deal with the imbalance superhost variable by preserving the proportion of the target. This made sure we don't end up with a split that has just non superhosts. We parallelized the Random Search and model training when possible. Taking the best model from each of the Random Searches we then produced the Recall-Precision & ROC curves.



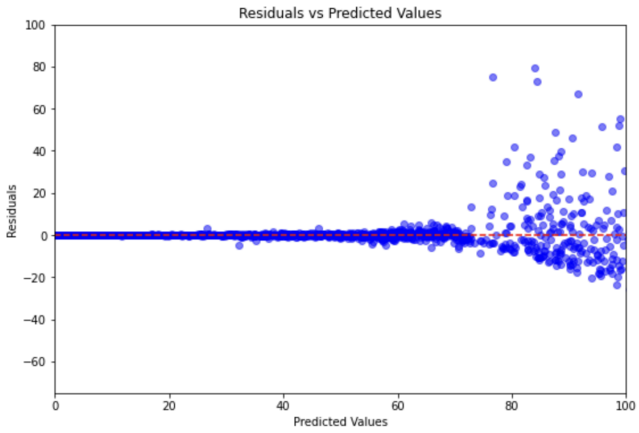


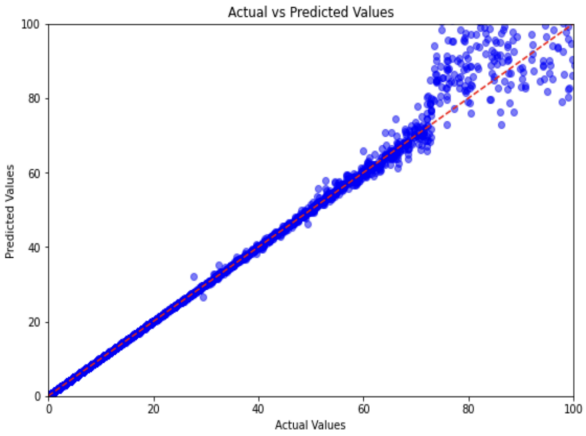
The Logistic & SVM models were pretty similar in performance while the two decision tree models performed better. We chose the Catboost model and then using the Precision-Recall values chose a threshold of 0.35 as that gave us a Recall of 0.95 and maintained a reasonable precision of 0.85.We also pull the top 5 variables that the Catboost model deemed more important when predicting superhosts: We see that host response & acceptance rate is highly important, but we also see that one of our generated features, being host about sentiment also managed to be in the top 5. Interestingly we see that the number of entire home & private room listings also are an important predictor in determining superhosts status.

	Feature	Importance
1	host_acceptance_rate	13.090171
14	calculated_host_listings_count_entire_homes	11.941963
15	calculated_host_listings_count_private_rooms	8.605590
0	host_response_rate	8.124121
33	host_about_sentiment_score	6.318699

Bookings Per Year Regression Results:

To predict the number of bookings for a particular property we opted to use a Catboost Regression model. Our aim was to get a model with a reasonable test RMSE value (Relative to the mean for context) and test R-squared value. When choosing our explanatory variables we had a wide selection from all 3 datasets, we decided to use most of the variables bar those that were manually identified as redundant or irrelevant. After variable selection, the data was split 80-20 for training and testing respectively, this was done to avoid overtraining and get model metrics on data it was not trained on. Additionally, the model was trained to predict a log transformation of the response, due to the fact that (as was shown in the EDA Results section) the response was right skewed, hence the transformation to make the distribution more normal. After finishing training the model these were the results.





As is illustrated above, although the model was trained on the log transformation of the response and it does have very accurate results for lower predictions, it does tend to struggle with larger predictions. Regardless it does achieve reasonable results, see also the test RMSE, MAE and R-squared values below.

```
Test Mean Absolute Error:  1.4365430829572727
Test Root Mean Squared Error:  13.810871000098539
Test R^2 Score:  0.8793014551058469
```

Following our results which indicate a reasonably good fit of our model, here are the top 5 variables that our model deemed the most important when predicting the number of bookings.

	Feature	Importance
48	estimated_ococcupancy_rate	92.506167
24	review_scores_rating	3.870384
23	number_of_reviews	1.204421
43	average_review_sentiment_score	0.406270
20	minimum_nights	0.376416

References:

Datasets:

NYC Airbnb Data: https://insideairbnb.com/get-the-data/

NYC Crime/Arrest Data:

- https://data.cityofnewyork.us/Public-Safety/NYC-crime/qb7u-rbmr/data (crime)
- https://data.cityofnewyork.us/Public-Safety/NYPD-Arrest-Data-Year-to-Date-/uip8-fykc/about_data (arrest)

NYC Income Data:

- https://www.kaggle.com/datasets/jinbonnie/condominium-comparable-rental-income-in-nyc (income per rental square foot in every area)

Handling missing Values:
https://medium.com/@pingsubhak/handling-missing-values-in-dataset-7-methods-that-you-need-to-know-5067d4e32b62#:~:text=1.,of%20the%20total%20891%20datapoints.

Vader Usage:
https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vaderSentiment.py#L517

Parallelization:
https://python.plainenglish.io/unlocking-parallelism-in-pandas-5dcb4badc63a

Photo Scoring:
https://labs.everypixel.com/ugc-photo-scoring

Reading Level Scoring:
https://simple.wikipedia.org/wiki/Flesch_Reading_Ease

Estimating number of bookings:
https://commissions.sfplanning.org/cpcpackets/2014-001033PCA.pdf

Handling unbalanced classes:
https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/
https://www.svds.com/learning-imbalanced-classes/

Estimating Bookings per Year:
https://commissions.sfplanning.org/cpcpackets/2014-001033PCA.pdf
https://insideairbnb.com/data-assumptions/