Airbnb Analysis in New York City

Boom, Boom, Boom!

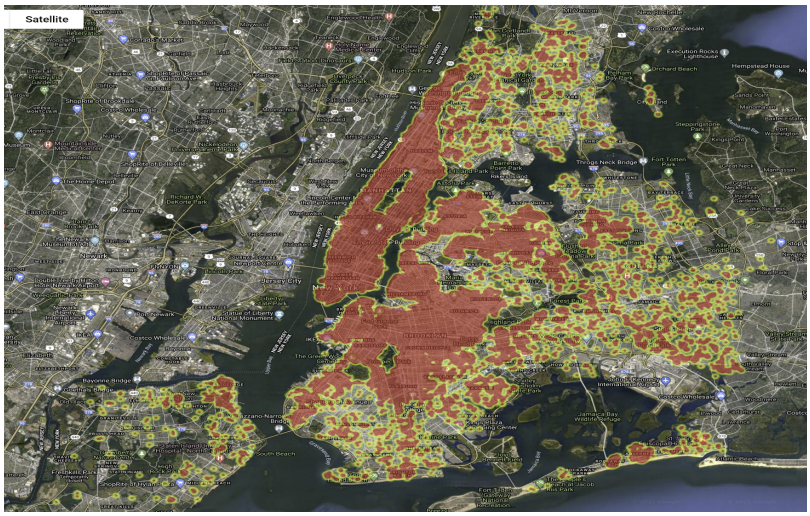
Jiyin Chen: jc5498
Ruiyi Fan: rf2759
Yufei Jin: yj2691
Lai Wei: lw2985
Fangzi You: fy2262

1 Introduction

For various features collected in our datasets for Airbnb in New York City, we are particularly interested in figuring out which features are important for clients' review scores, accomodation prices and predicting prices given the features provided. We build our analysis on techniques including Word Cloud analysis, Sentimental analysis and models such as Linear Regression, Decision Tree, Random Forest, PCA, Gradient Boosting Regression, CatBoost, Lasso, XGBoost etc. Last but not least, we provide some insights on how to improve customers' overall satisfaction and reveal the inner logic for hosts' prices based on analysis.

2 Exploratory Data Analysis

Firstly, we plot a heatmap of the distribution of Airbnb Rentals in New York city using Google API. As shown in the satellite picture, these properties are mainly distributed in Manhattan, Brooklyn and Queens where they are usually considered high density in population. Others left seem to obey the punctate and zonal distribution in the east part of New York City, Staten Island and JFK International airport.



Secondly, text mining techniques are used to give us an overall view of the reviews. We firstly merge the reviews under the same property and then divide the reviews by several ranges of rating. Word Clouds are built on several subcategories to find out the most frequent word under different ratings. Below figures from left to right respectively represent the word cloud derived from all comments, comments with rating score less 4, and comments with rating score equal to 5.

Thirdly, sentiment analysis using NRC data is employed to reveal customers' attitudes towards accommodations grouped by different scopes of rating scores.

hosts	Fear	Trust	Negative	Positive	Joy	Disgust	Anticipation	Sadness	Surprise
Rating less than 4	0.006998	0.019814	0.018959	0.031124	0.014118	0.006632	0.014443	0.008951	0.006794
4 <= Rating < 4.5	0.003719	0.025499	0.010855	0.038843	0.019334	0.002791	0.017668	0.006220	0.007914
4.5 <= Rating < 4.7	0.003120	0.026042	0.008774	0.039559	0.020717	0.001913	0.017922	0.005603	0.007693
4.7 <= Rating < 4.8	0.002785	0.027050	0.007243	0.042168	0.021884	0.001232	0.017922	0.005424	0.007784
4.8 <= Rating < 4.9	0.002571	0.028765	0.006466	0.044776	0.023614	0.001125	0.019199	0.005582	0.008591
4.9 <= Rating < 5.0	0.002369	0.028695	0.005938	0.046127	0.002444	0.000970	0.018941	0.005418	0.008587
Rating = 5	0.001652	0.030446	0.005137	0.048662	0.025696	0.000840	0.019515	0.004761	0.008484

3 Data Pre-Preprocessing

We drop none value columns such as ‘bathrooms’, ‘calender_updated’,’license’. And we drop useless columns by business sense. For example, the url columns won’t be helpful to prediction. And we drop columns which contain messy information.

Furthermore, we apply the numpy ‘where’ method to convert binary results to 0 or 1 such as ‘instant_bookable’,’host_is_superhost’,’host_identity_verified’ and ‘has_availability’. And convert date string to datetime type. In the imputing processing, we fill the missing value with the median of the column.

For dealing with object type columns, we convert those columns into dummy variables and combine the dummy variable dataframe with the original dataframe. The dummy variable data frame has 989 columns which is too large and we will need pca to reduce dimension for this dataframe. So we train the model in two ways: one with a non-dummy dataset and one with a dummy dataset after pca svd processing.

4 Machine Learning Models

4.1 Linear Regression, Tree-based methods & Gradient Boosting Regression

4.1.1 Linear Regression

From the Data Preprocessing part, we get 45 non-dummy variables filtered from 74 different columns (with dummies) in the combined dataset.

Our first intuition is to apply a naive linear regression model to the datasets. With the dummy variable, the overall performance is really bad. In this case, negative testing R- squared means that our chosen model does not follow the trend of the data and fits worse than a horizontal line. So, we eliminate the dummies, and this time our results go back to normal with R-squared of 0.15. The results are not surprising for us, by looking at the correlation graph of those variables, because there are only a few variables that have linear correlation with the dependent variable.

4.1.2 Decision tree & Random Forest

The next step is to fit some classification models to the dataset. Starting from what we learned in class, we try to fit a decision tree classification model to the non-dummy variables. Even though we ignore the dummies for this model, the result is still negative in terms of the testing R square. After taking a closer look into the result, we think the decision tree is not suitable for such a huge dataset. The wide differences between training and testing R square indicate that this classification model has a serious overfitting problem. To optimize that model, we decide to apply Random Forest techniques and Grid Search for our classification model. With randomly selected parameters, the Random Forest model already returns a better result than the decision tree. The gap between training and testing R square is narrowed by about 75%, but the leading training R square still indicates the overfitting problem. Besides, by utilizing the Grid Search method, the result is optimized, even with a small range for the parameters. Therefore, we think that Random Forest model with Grid Search technique might be one of the best models for us.

4.1.3 Gradient Boosting Regression (GBR) Model

Inspired by linear regression and the classification models, we think we also need to find a way to further filter out the variables for building the models. After researching online, we find that the GBR model might be the best solution in our case. With its specialties in showing the feature importance, we make a plot (Figure 1) for all of the 45 non-dummy variables. In terms of predicting the price, the plot shows that the date of the last review, the location ("longitude" and "latitude"), the space of listing ("accommodates"), and the number of beds are the top five features. It is not hard to understand this. The date of the last review somewhat reflects the frequency that the listing is being booked, more frequent booking means a higher price by the demand-supply logic. In addition to that, the location and the space of the listings will play more important roles than availability or review score in terms of pricing. For example, an apartment near the World Trade Center will have a higher price than some apartments in Long Island City, and a 4-bedroom, 4-bathroom apartment has a higher price than a studio. From this perspective, we can fit our model based on the result of the feature importance.

4.2 XGBoost & CatBoost Regression

Parallel to our progress of the Random Forest and GBR model, we also implement some other ensemble learning models: XGBoost and CatBoost, with CatBoost models using two separate optimizers, fmin and GridSearch.

4.2.1 Models trained by dataset with all variables including dummies

For these three additional boosting models (XGBoost regression, CatBoost regression - fmin, and CatBoost regression - Gridsearch), we first try using all variables (in which over 95% of them are dummy variables) for the model training process and find that XGB yields a training score of 0.985 and a test score of 0.35, while both of the CatBoost models have training scores between 0.60 and 0.65, with their test scores only slightly higher than 0.30. These performance results are shown in Figure 2. Sure enough from such a result we can see that XGB performs somewhat better than both CatBoost models for the set of variables with all those 900+ dummy variables included, but with the highest test score among these alternative boosting models being only 0.35, such a test-run result might not be something satisfactory enough.

The result thereby aroused our concern of skipping all those dummy variables and only training these three boosting models on the 45 non-dummy variables, which is to be discussed in the following section.

4.2.2 Models trained by non-dummy dataset

With this new round of training only on the non-dummy features, we find the training scores of both CatBoost models show a huge improvement, with both the values exceeding 0.70, and at the same time the XGB performance in terms of the training score remains as good as that in the previous training with all dummies included. However, when we come to the test scores, we don't see significant differences from those test scores in the test-runs with dummies, with XGB's test score even lower, this time being 0.30 (which became close to the other two models' test scores). These performance results are shown in Figure 3. In this way, these three boosting models, XGB, Cat-fmin, and Cat-GridSearch, are still far from being ideal models for working on price prediction machine learning modeling. Nevertheless, like all other ensemble models, we can still find out and sort the feature importances under these models to get some additional insights.

In the model runnings with dummy variables talked about in the previous section, the top-ranking important features are more likely to include a huge portion of such dummies, and sure enough some of such rankings are meaningless, since among these results some top-ranked features are dummies with respect to specific locations in the NYC, e.g. "neighbourhood_cleansed_Lower_East_Side" (shown in Figure 4), which could in no case be a decisive factor for the overall citywide price level. And in contrast, the model running we are talking about in this section with all dummies skipped provides us a significantly clearer picture of the actual variables of interest within the dataset. Similar to what's done right after the test-run of the GBR model, we make one plot for each of these three boosting models (Figures 5~7), ranking the importance with respect to all these 45 non-dummy columns.

(1) XGBoost

The XGB plot (Figure 5) shows that for price prediction, whether the host identity gets verified ("host_identity_verified"), the space of listing ("accommodates"), and the minimum number of night stay for the listing ("minimum_nights", "minimum_minimum_nights", and "maximum_minimum_nights") are the top five features. As analyzed for the GBR model, the space of the listings has a relatively huge impact on the price. And the trustworthiness of the host's identity is also something many customers may get concerned about, which could then hugely get tied to the price of that host's listing. Moreover, the minimum number of nights stay can be economically understood as some underlying quantity-like variable used by all sorts of pricing models, so there is no surprise that such variables get ranked as well upon the top few features of importance.

(2) CatBoost

The two CatBoost plots (Figures 6&7) show the same set of the top features of importance: the location ("longitude" and "latitude"), the space of listing ("accommodates"), and the availability of the listing in the future within a certain number of days ("availability_30/60/90"), which has more overlapping features compared to the GBR model. In addition to a couple of features analyzed already in previous paragraphs, (i.e., location and space of listing), the availability of the listing in the future is something in close relation to the supply side of the listings themselves, so that just like the minimum number of night stay talked in the last paragraph, as a variable with such level of importance economically, the future availability of the listing is extremely likely to be ranked for top features of importance.

Thus, though the model performance of the XGB, Cat-fmin, and Cat-GridSearch are not as nice (especially when compared to the GBR model), the important features these model results spat out continued to give us some clear views of the data's key variables.

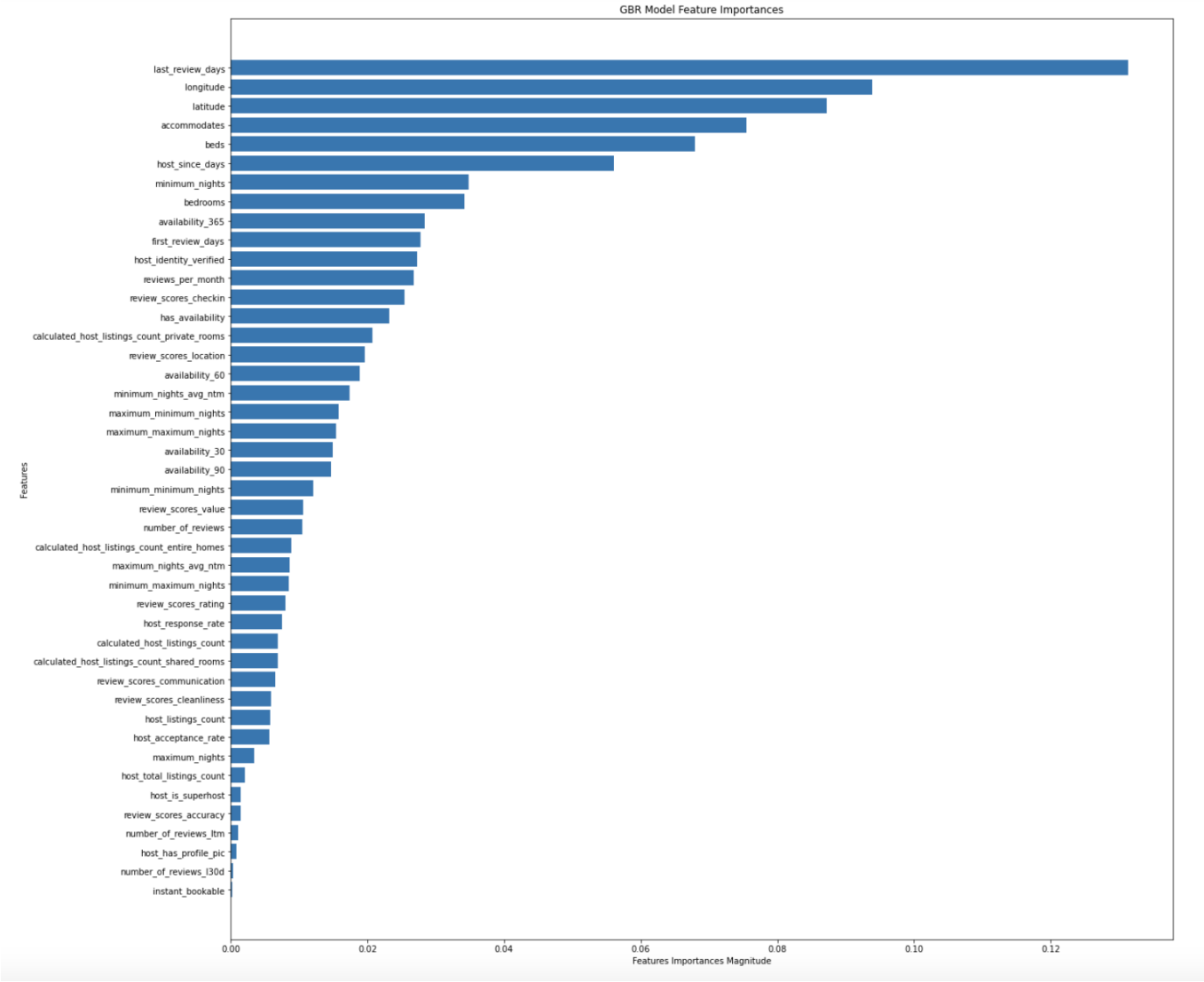
5 Summary

Based on all of our model test-runs on different types of NYC Airbnb listings datasets (dataset with dummies as well as the non-dummy dataset), we can observe that the most important features for the listing price are the location and the space of listing, all of which appear on most of the models' top-ranking features. In addition, some quantity-based features (like the minimum number of nights and number of beds), as well as some time/date-based features (like availability in the future X number of days and date of the last review), can also be considered with a certain level of importance, all of which appear on at least one of these models' top-ranking features. Note that some of these features themselves contain keywords that also appear as important ones in the word cloud (such as "location", "space" and "room") when conducting text mining based on review score ratings. In this way, we can see that the decisive factors for listing price may also more or less be decisive for the review scores. Thus, we recommend that both Airbnb's end and the hosts' end, when considering the issue of pricing, hosts, as well as Airbnb, can work on those factors that are commonly thought to affect review scores and take into account such factors to make wise decisions on pricing.

As for the possible improvements that could be done to our analysis, we think it would be a good idea to incorporate some of the dummy variables in addition to our 45 non-dummy columns, in that there may still be a few of such non-dummy variables that do not involve specific locations but instead involve whether certain rooms contain certain types of facilities, and these particular columns can be useful for our model test-runs. We may want to use the Principal Component Analysis (PCA) on these types of dummy variables and select those top features to add into the original 45 non-dummy variables' dataset. In this way, the performance of the model might be improved, and more precise conclusions can be drawn.

Appendix

Figure 1 - Feature importance for Gradient Boosting Regression with no dummy variables during training:



GBR model (non-dummy)

Figure 2 - Performance of XGBoost regression, CatBoost regression - fmin, and CatBoost regression - Gridsearch with all dummy variables included when training:

```
1 test_score
```

0.3102281473089098

```
1 train_score
```

0.6392579677162786

CatBoost-fmin (with dummy)

```
1 train_score_grid
```

0.6068849994151719

```
1 test_score_grid
```

0.3066546361931498

CatBoost-GridSearch (with dummy)

```
1 test_score_xg
```

0.35091314369951165

```
1 train_score_xg
```

0.9852002799949748

XGBoost (with dummy)

Figure 3 - Performance of XGBoost regression, CatBoost regression - fmin, and CatBoost regression - Gridsearch with no dummy variables during training:

test_score
0.3079925352527536

train_score
0.7577274680945203

CatBoost-fmin (non-dummy)

train_score_grid
0.7193465250659128

test_score_grid
0.3010907006227441

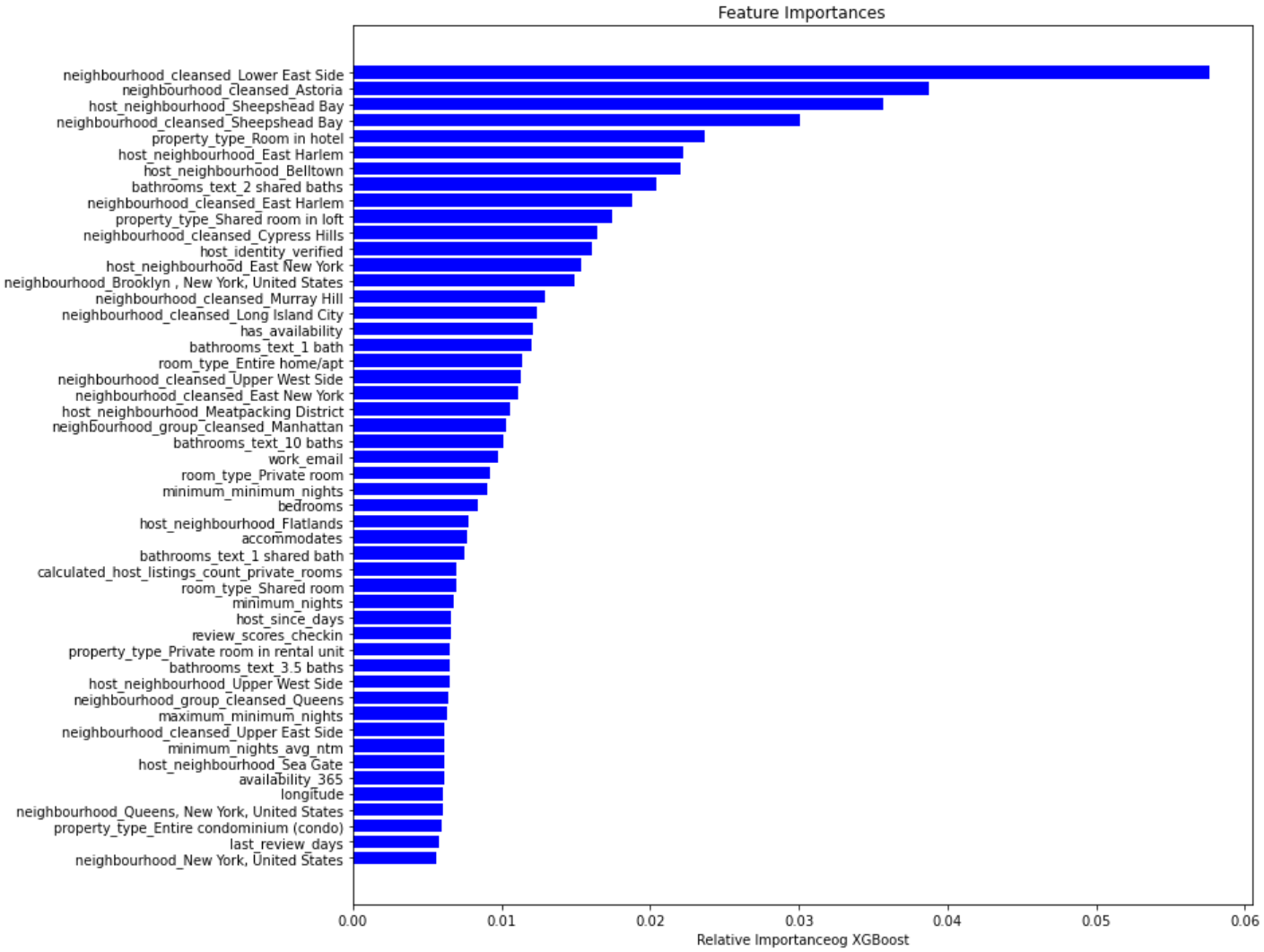
CatBoost-GridSearch (non-dummy)

test_score_xg
0.30037708067766644

train_score_xg
0.9711994006811975

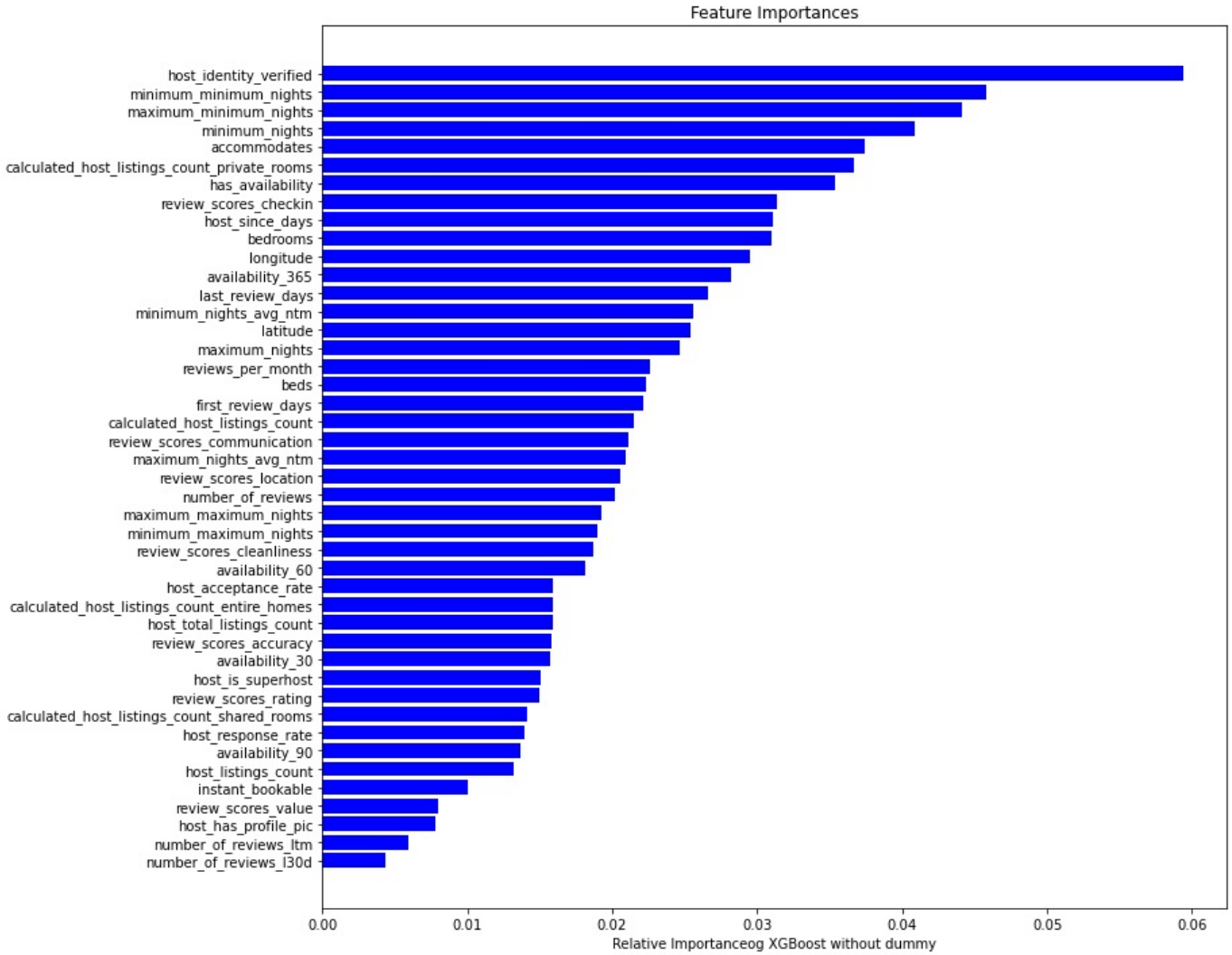
XGBoost (non-dummy)

Figure 4 - Feature importance for XGBoost regression with all variables including dummies during training:



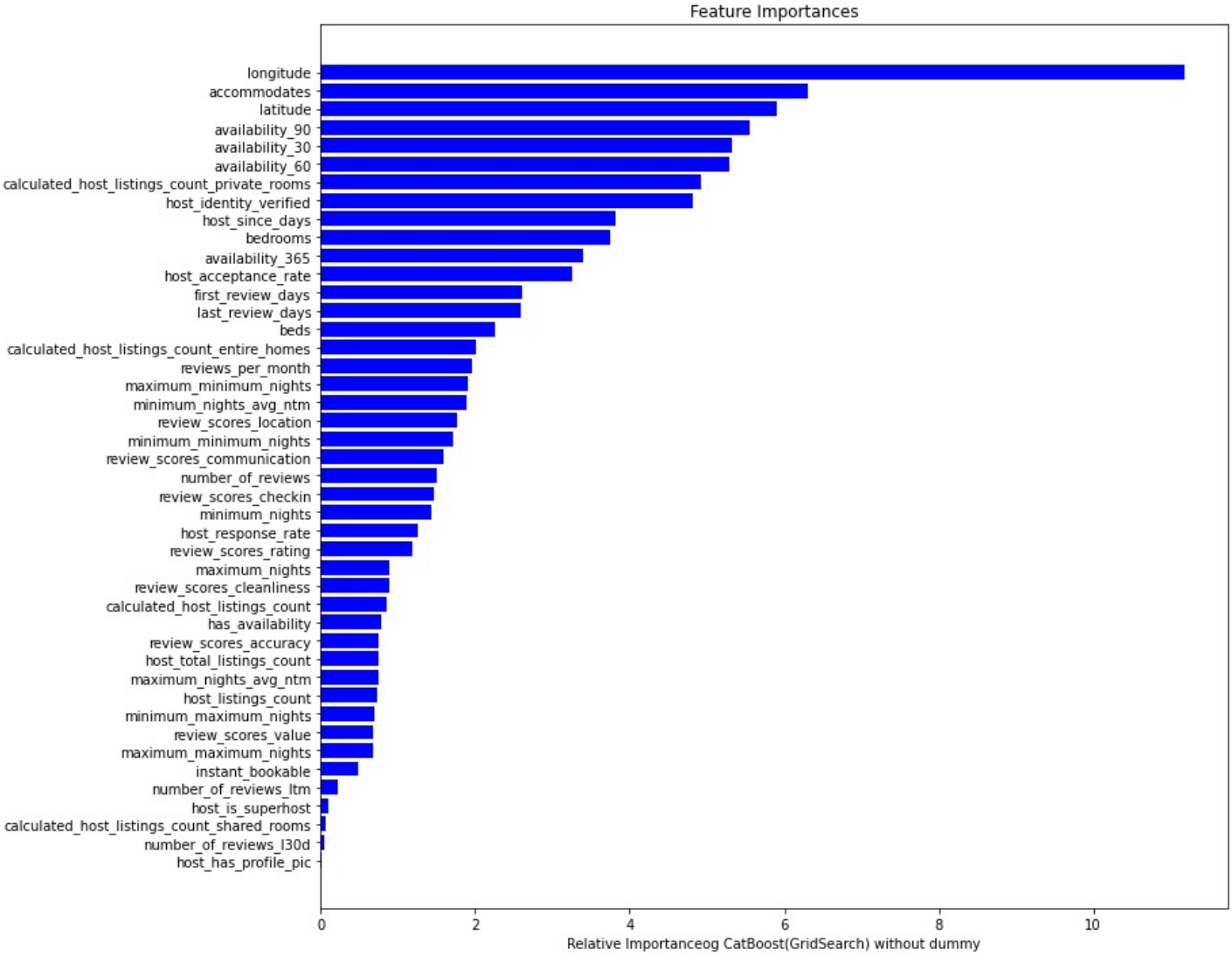
XGBoost (with dummies)

Figure 5 - Feature importance for XGBoost regression with no dummy variables during training:



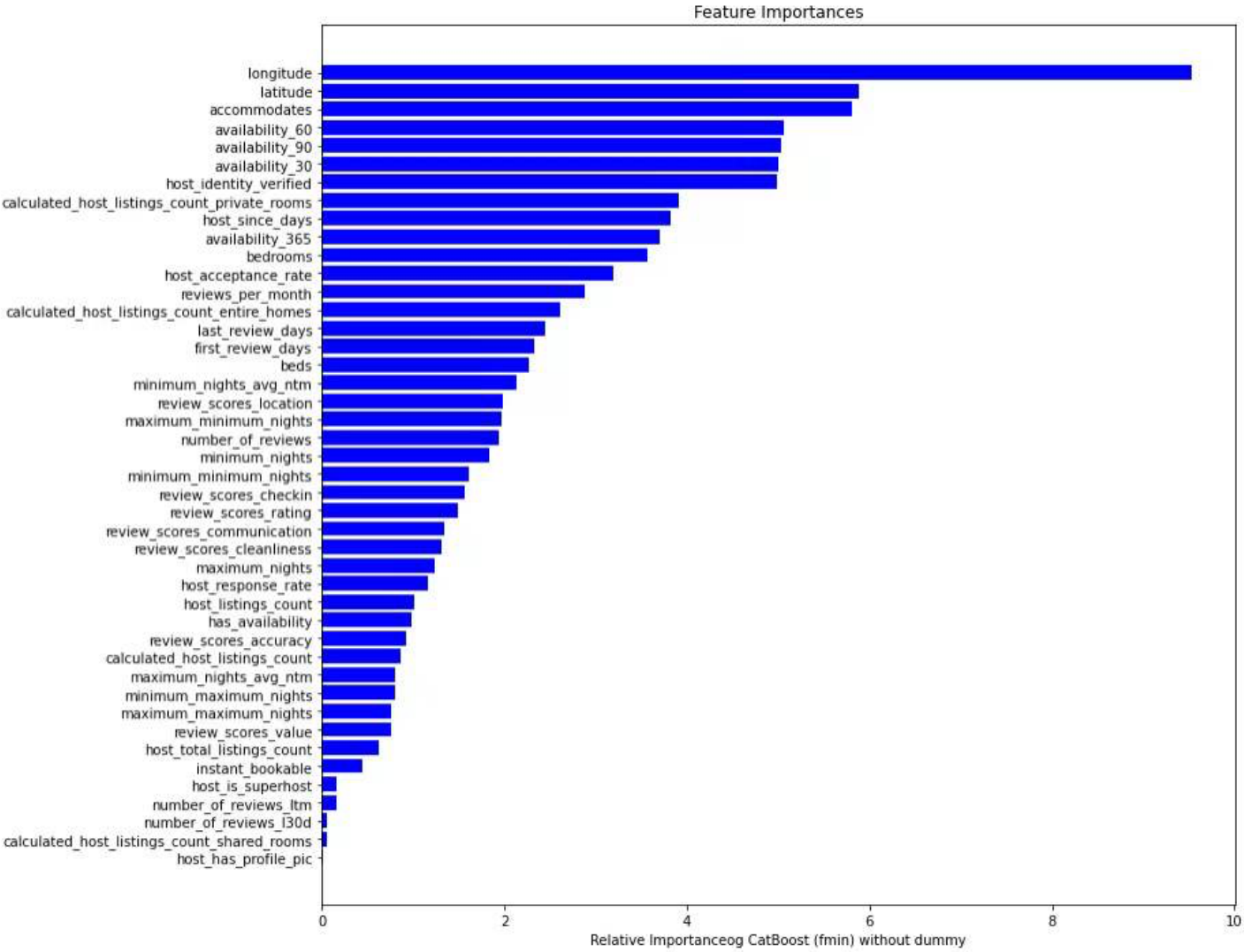
XGBoost (non-dummies)

Figure 6 - Feature importance for CatBoost regression - Gridsearch with no dummy variables during training:



CatBoost-GridSearch (non-dummy)

Figure 7 - Feature importance for CatBoost regression - fmin with no dummy variables during training:



CatBoost-fmin (non-dummy)