Capstone 1 – Machine Learning

Building a Price Model for Airbnb Listings in Austin, TX

Zach Palamara

Project Goal

At the core, the fundamental goal of this project was to develop a model to accurately predict the nightly price of an Airbnb listing using features of the listing itself. In other words, this simply boils down to predicting the value of the dependent variable (price), by using those listing features, which serve as the dependent variables. Given this problem statement, I decided to use a linear regression model since its inherent properties would be able to accomplish this.

Model Preparation

One-Hot Encoding

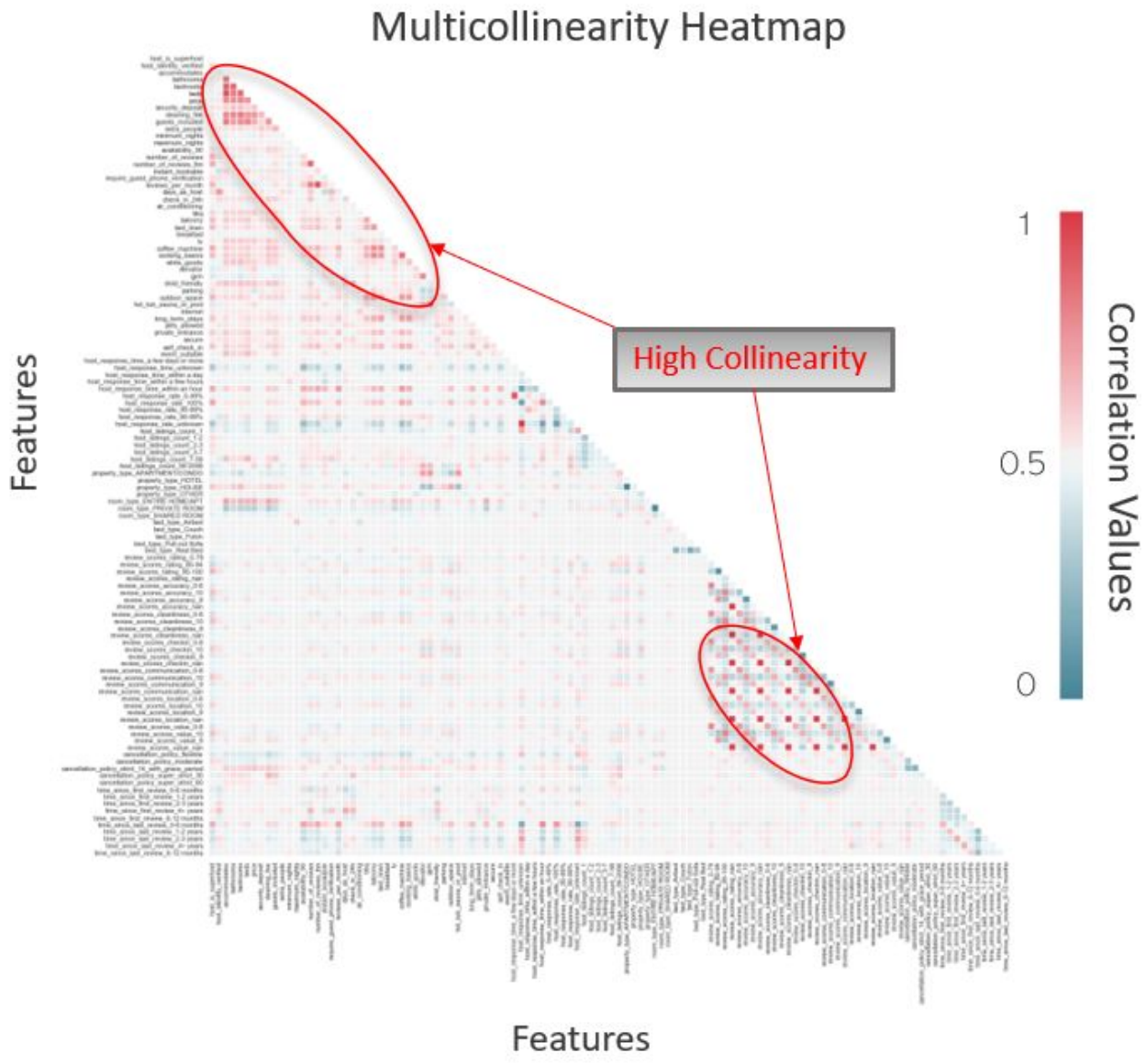
Before applying any regression models to the data, I had to massage the features of my data. The first step in this process was to use a common method called “One-Hot Encoding”, which allows categorical features to be expressed as integers. For example, instead of having a single feature such as “room_type” that contains string values such as “private_room” or “entire_home/apt”, One-Hot Encoding transforms each one of those unique values into separate columns containing either a “1” or a “0”. The result of implementing this technique can be seen below.

One-Hot Encoded Dataframe

id	host_is_superhost	host_identity_verified	accommodates	bathrooms	bedrooms	beds
2265	1.0	1.0	4	2.0	2.0	2.0
5245	1.0	1.0	2	1.0	1.0	2.0
5456	1.0	1.0	3	1.0	1.0	2.0
5769	1.0	1.0	2	1.0	1.0	1.0
6413	1.0	0.0	2	1.0	0.0	1.0

Multicollinearity

The next step I took was to assess the features for multicollinearity, which highlights features that have very strong linear relationships with each other. I was able to accomplish this by using a function from the Seaborn library. Using this function, I was able to generate a visual heat map that clearly showed areas of multicollinearity in the data.



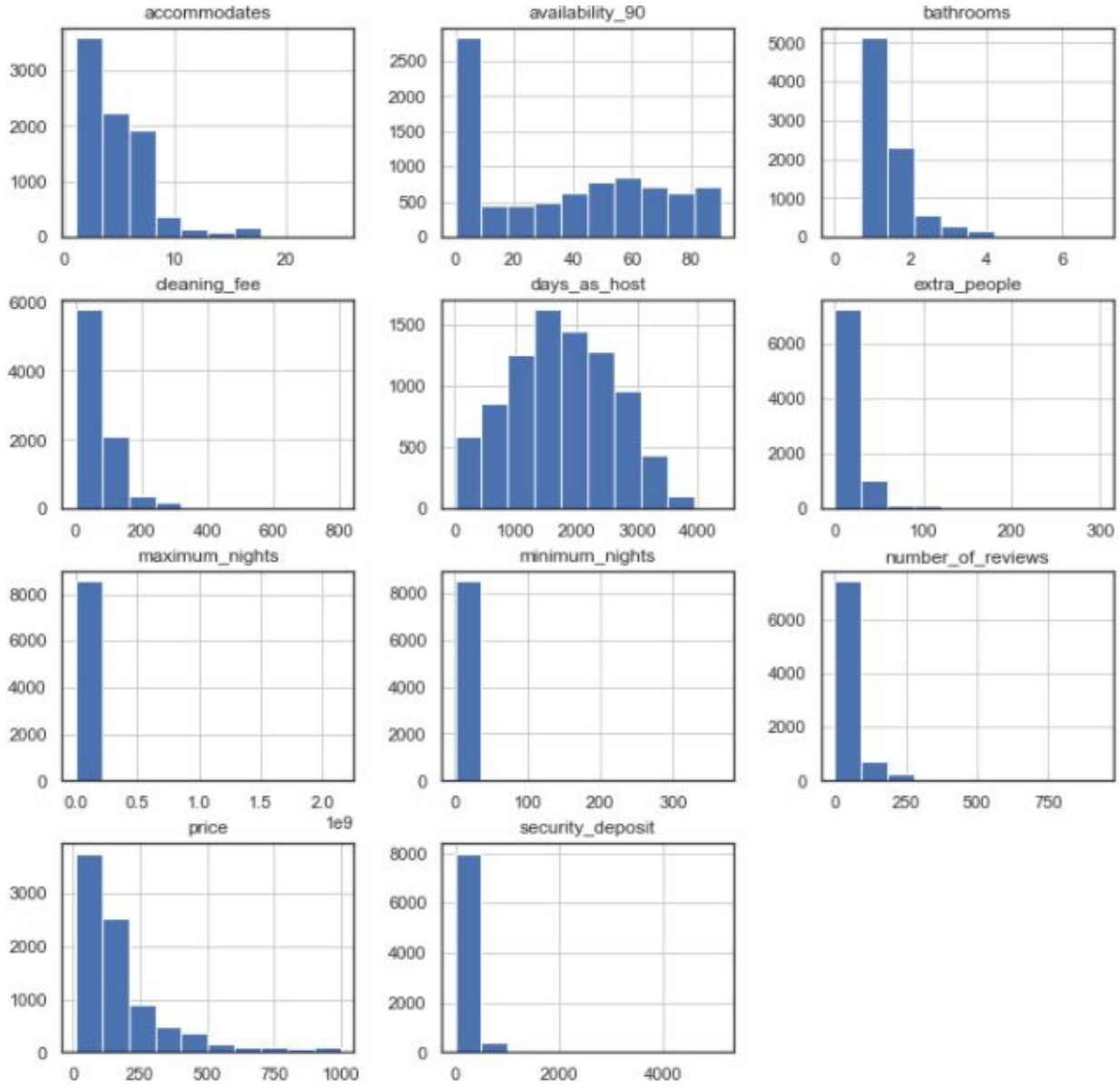
The dark red areas on the heat map consist of the beds, bedrooms and guests features. Since these features are all highly correlated and relatively similar in nature with the

number of people that a listing accommodates, I decided to drop them from the data set.

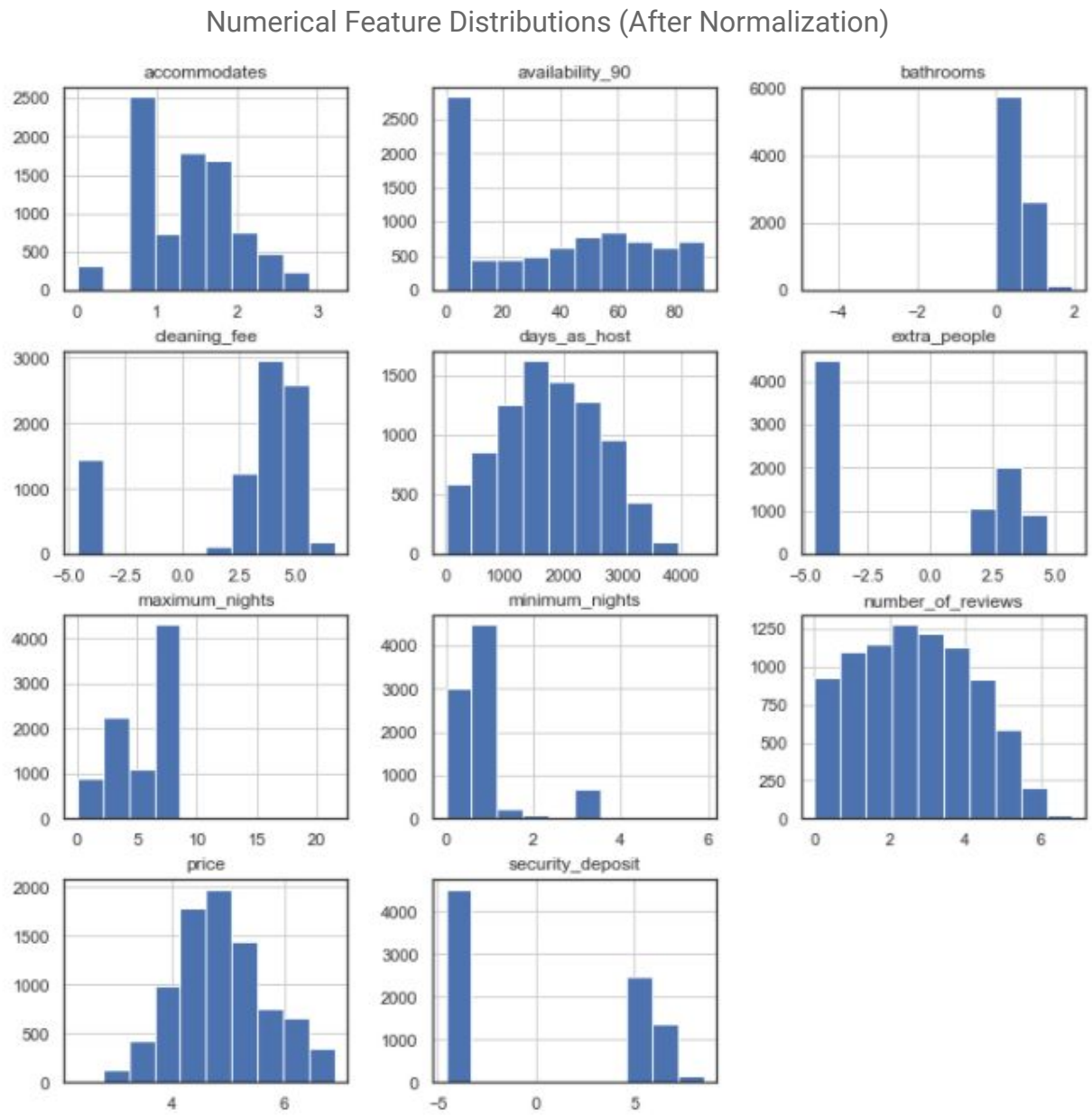
Normalization and Standardization

As a last step in the model preparation process, I decided to take a look at some of the numerical features to see if there were any features that needed to be normalized.

Numerical Feature Distributions (Before Normalization)



Since a lot of these features seem to be positively skewed, I decided to do a log transformation on the following features: *'price','accommodates', 'bathrooms', 'security_deposit', 'cleaning_fee', 'extra_people', 'minimum_nights', 'maximum_nights', 'number_of_reviews'*. Most of the resulting distributions appear much more normal and should provide better results in the modeling phase.



Lastly, I scaled these numerical features so they would have equal weights in the machine learning model. I accomplished this by using the StandardScaler in SKLearn.

Machine Learning

Model Implementation

After completing steps for model preparation, I began setting up regression models. My initial model was constructed using a gradient boosted regression classifier from the SKLearn ensemble. I also decided to test a gradient boosted regression model using the XGBoost framework. The results of these two models are shown below.

Model Performance

Metric	Model	
	XGBoost Regression	SKLearn Gradient Boosted Regression
Train MSE	19.61%	19.28%
Test MSE	21.11%	21.64%
Train R^2	0.7228	0.7274
Test R^2	0.6943	0.6867
Computational Time (sec)	0.9	11.8

Feature Importance

Feature Importance	Model	
	XGBoost Regression	SKLearn Gradient Boosted Regression
1	room_type_entire home/apt	accomodates
2	accomodates	property_type_other
3	bathrooms	security_deposit
4	cleaning_fee	bathrooms
5	air_conditioning	require_guest_verification
6	parking	extra_people
7	cancellation_policy_strict_14_ with_grace_period	reviews_per_month
8	host_listings_count_58-2056	child_friendly
9	security_deposit	cleaning_fee
10	property_type_other	white_goods

Interestingly, the feature importance is actually quite different between these two models, but the most important feature across appears to be the number of people a listing and the number of bathrooms a listing accomodates. There are however, some other important ancillary

features that could help boost the price of the listing, such as having parking, air conditioning, the entire home to yourself, being child friendly or having linens supplied in addition to some other features.

Conclusion

Considering the data presented above, it appears that both models had nearly the same results. The SKLearn model had slightly better training results, while the XGBoost model had better results on the test set. This indicates that the SKLearn model might have been slightly overfitting the data. However, the biggest standout metric was computational time. The XGBoost model was more than 13x faster in computational speed, which could have huge implications as the data set grows.