CS3315
Final Project Report
LT Cameron Woods
LTJG Micky Hall

I. Introduction

Our goal with this project was ultimately to build a model that would be able to accurately predict the price of an Airbnb listing, given a specific set of features. We were attracted to this problem because current and future generations will face a vastly different real estate landscape than the generations that came before us. In 1950, the median household annual income was $2,990, and the median home value was $7,400, which is a little over 247%. In 2010, the median household income had risen to $49,445, while the median home value grew to $221,800, which represents 445% of the median income.[1] The disparity between housing prices and income shows no signs of slowing, with housing prices having already recovered from the 2008 housing market crash. As purchasing a home becomes more and more infeasible, younger generations are choosing to abandon the norms of their parents and grandparents, and instead making renting a lifelong reality.[2]

This change to the housing industry is what makes price prediction for Airbnb rentals so important. With Airbnb's recent IPO, the company is now valued close to the total market cap of all U.S. publicly traded hotel chains, combined.[3] Additionally, the prevalence of large collections of Airbnb data on websites such as Kaggle.com and insideairbnb.com made it an ideal candidate for a basis to build our prediction model.

II. Hypothesis

Our hypothesis was that the price of an Airbnb rental could be predicted, within one standard deviation of the price of the listings in our dataset, based only on standard and readily available features, such as latitude and longitude, room type, and number of reviews.

III. Data Selection, cleaning, and munging

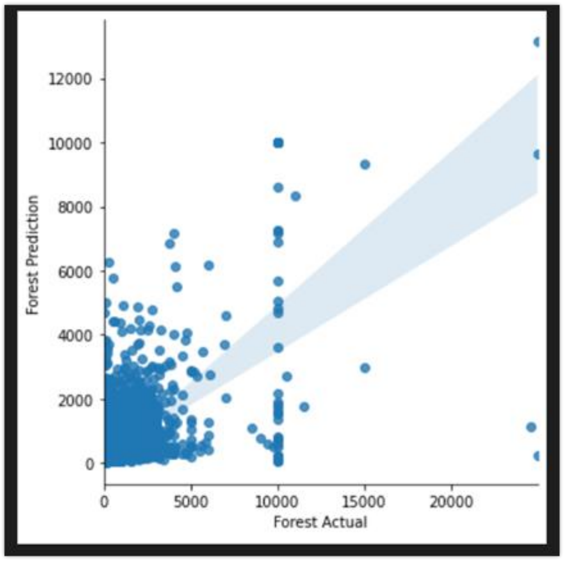
We chose the data set "U.S. Airbnb Open Data"[4] for our data set. It contains 226,029 separate Airbnb listings, with 17 unique features. This was one of the largest Airbnb data sets we could find, and it had data from Airbnb listings across the nation, while many data sets were specific to just one city or region. We wanted our model to be viable no matter what city it was being used in, so a nationwide data set was a necessity.

[1] "Why buying a house today is so much harder than in 19050", https://archive.curbed.com/2018/4/10/17219786/buying-a-house-mortgage-government-gi-bill
[2] "Millennials' share of the U.S. housing market: Small and shrinking" https://www.washingtonpost.com/business/2020/01/20/millennials-share-us-housing-market-small-shrinking/
[3] "Make it make Sense $ABNB", https://www.reddit.com/r/wallstreetbets/comments/kauoi3/make_it_make_sense_abnb/
[4] "U.S. Airbnb Open Data", https://www.kaggle.com/kritikseth/us-airbnb-open-data

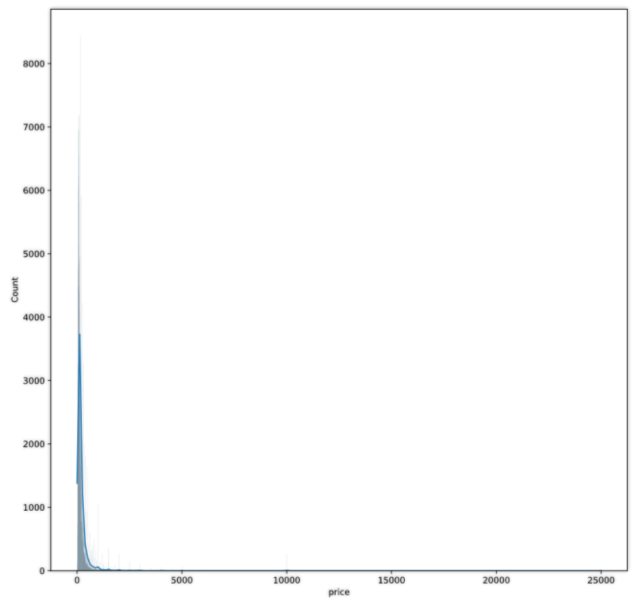
To start our data cleaning process, we immediately dropped features that we determined were redundant, strings, or were mostly missing. This meant we dropped *name, host_name, city, neighbourhood, last_review, id,* and *neighbourhood_group*. With the inclusion of latitude and longitude, we initially felt it was not necessary to have *city* or *neighbourhood*, and *neighbourhood_group* was just another representation of *neighbourhood*. This left *latitude, longitude, room_type, minimum_nights, number_of_reviews, reviews_per_month, calculated_host_listings_count,* and *availability_365* as our features.

A number of values for *reviews_per_month* were missing, however we determined that these were blank because there were no reviews for that listing at the time the data was collected, so we filled in all missing values with 0. We then "one-hot" encoded *room_type* to separate out the different possible room types Airbnb allows for a listing into their own feature columns. This ended up adding features for *Private room, Entire home/apt, Hotel room,* and *Shared room.*

This was all the data cleaning we did before running our first regressions – we wanted to see a baseline for what we were working with. As can be seen in the following chart, the result was very poor, with a calculated RMSE of 221207.383 from a linear regression, and 410.729 from a random forest regression.



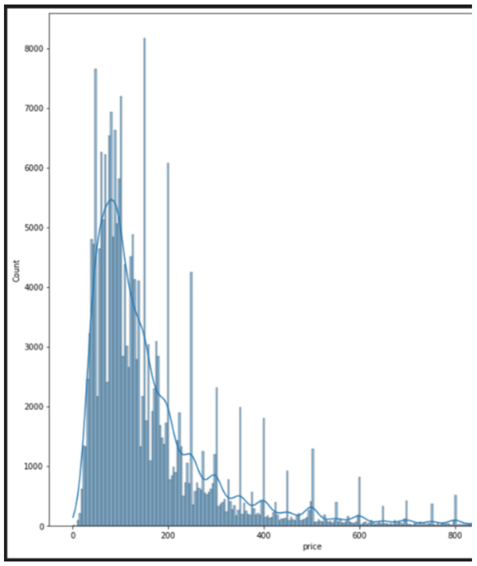
The next step was to remove outliers from our data. The following chart is the price distribution of our data set before any further cleaning was done.



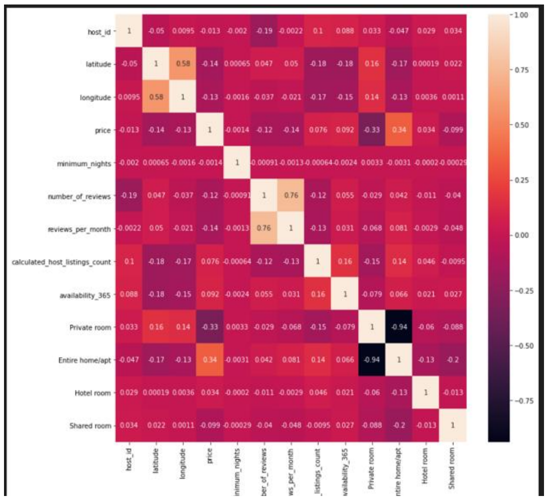
While the mean price for our data set was $219, there were a few listings with prices all the way up to $24,999. We removed just these highest values, as well as any prices that were $0 or lower, and immediately started to see much better results. The RMSE of the linear regression dropped from 221,207.383 to 504.137, and the random forest regression RMSE was down to $318.07. Technically at this point we had met our goal because the standard deviation was $502.03. However, the mean price was only $216.81. So, while yes, we were within one standard deviation, there was still a lot more work to do to get our model to be actually meaningful. So, we started to remove many more outliers.

We concluded that it was reasonable to drop all prices above $1000, because if someone is paying $1000 a night for an Airbnb, it is likely for a very specific purpose, be it a famous location or home, or a property that becomes very valuable during specific events, e.g., property in Augusta, Georgia during the Masters. Whatever the case may be, our model was intended to predict a fair price for the average user, not niche events.

Our price distribution following the removal of these outliers looked much more normal, represented below.



It is at this point that we ran a correlation heatmap for our features, to hopefully glean which remaining features were of particular value, if any. Unfortunately, the heatmap could not tell us much.



The only features with any reasonable correlation, positive or negative, were *private_room* and *entire home/apt*, which were derived from *room_type*. These correlations make sense; renting an entire home should, on average, be more expensive than just renting a room. However, the rest of the correlation table did not yield any other particularly meaningful results.

With this in mind, we decided it was time to determine what regressor our model would use, and then tune it. The last feature engineering we did at this step was drop the max price down to $600 (under the same "average user" logic) as well as added back *neighbourhood* and then label encoded it. We chose to do this because we wanted to allow for the possibility that listings from the same neighbourhood, e.g. "Manhattan" were priced higher than others, which just having latitude and longitude might not accurately reflect.

IV. Methods

We started this project by just running a linear regression and a random forest regressor, untuned, as essentially guideposts while we cleaned and munged our data. As shown previously, the random forest regression faired much better than the linear regression, so that was where we started our search for our optimal regressor.

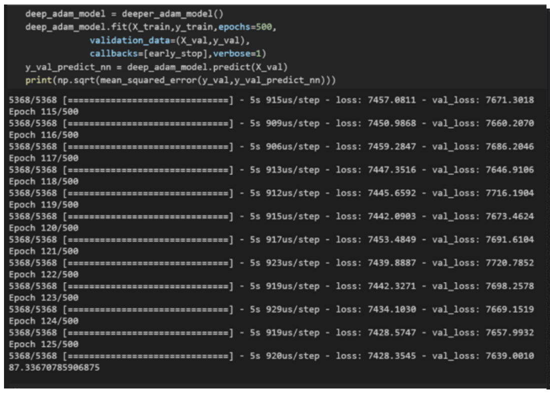
The next step was to tune the random forest regression. We ran a randomized search using cross validation, with *n_estimators, max_features, min_samples_leaf, bootstrap,* and *max_depth* as our hyperparameters to be tuned. Our random search determined these to be our optimal hyperparameters for the random forest regression.

- n_estimators: 20
- max_features: “log2”
- min_samples_leaf: 2
- bootstrap: False
- max_depth: 150

This resulted in an RMSE of just $78.95, with a standard deviation of $107.95, with our price mean at $148.66. With this, we had achieved our goal of being within one standard deviation, and the relationship between RMSE, standard deviation, and mean look much better now than they did at the start of our outlier removal. However, we still wanted to explore a number of regressors to figure out if we could get even better performance.

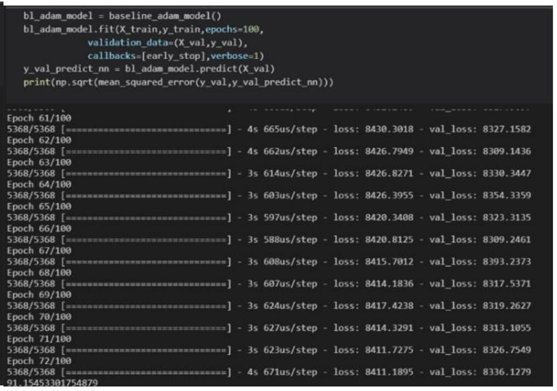
To do this, we started to tune and run a number of other regression models and compared them to our tuned random forest results to see if they could beat it. We started with an XGBoost regression, which came close with an RMSE of $79.19. We found this surprising, as we had anticipated XGBoost being the best choice based on researching similar regression problems.

Next we tried both Stochastic Gradient Descent (SGD) and Polynomial Regression. These two models did not fair as well, resulting with an RMSE of $97.17 and $94.17 respectively. At this point, we were starting to become confident that our original random forest regressor was likely to be our winning model for this project. We ultimately decided to try two neural network models with our data, having just learned about them in class. As can be seen in the results below, the deep learning model came close to beating the random forest by achieving an RMSE of $87.33, however ultimately no model we tried had a lower RMSE than the random forest.



```
deep_adam_model = deeper_adam_model()
deep_adam_model.fit(X_train,y_train,epochs=500,
          validation_data=(X_val,y_val),
          callbacks=[early_stop],verbose=1)
y_val_predict_nn = deep_adam_model.predict(X_val)
print(np.sqrt(mean_squared_error(y_val,y_val_predict_nn)))
```

```
5368/5368 [==============================] - 5s 915us/step - loss: 7457.0811 - val_loss: 7671.3018
Epoch 115/500
5368/5368 [==============================] - 5s 909us/step - loss: 7450.9868 - val_loss: 7660.2070
Epoch 116/500
5368/5368 [==============================] - 5s 906us/step - loss: 7459.2847 - val_loss: 7686.2046
Epoch 117/500
5368/5368 [==============================] - 5s 913us/step - loss: 7447.3516 - val_loss: 7646.9106
Epoch 118/500
5368/5368 [==============================] - 5s 912us/step - loss: 7445.6592 - val_loss: 7716.1904
Epoch 119/500
5368/5368 [==============================] - 5s 915us/step - loss: 7442.0903 - val_loss: 7673.4624
Epoch 120/500
5368/5368 [==============================] - 5s 917us/step - loss: 7453.4849 - val_loss: 7691.6104
Epoch 121/500
5368/5368 [==============================] - 5s 923us/step - loss: 7439.8887 - val_loss: 7720.7852
Epoch 122/500
5368/5368 [==============================] - 5s 919us/step - loss: 7442.3271 - val_loss: 7698.2578
Epoch 123/500
5368/5368 [==============================] - 5s 929us/step - loss: 7434.1030 - val_loss: 7669.1519
Epoch 124/500
5368/5368 [==============================] - 5s 919us/step - loss: 7428.5747 - val_loss: 7657.9932
Epoch 125/500
5368/5368 [==============================] - 5s 920us/step - loss: 7428.3545 - val_loss: 7639.0010
87.33670785906875
```



```
bl_adam_model = baseline_adam_model()
bl_adam_model.fit(X_train,y_train,epochs=100,
          validation_data=(X_val,y_val),
          callbacks=[early_stop],verbose=1)
y_val_predict_nn = bl_adam_model.predict(X_val)
print(np.sqrt(mean_squared_error(y_val,y_val_predict_nn)))
```

```
Epoch 61/100
5368/5368 [==============================] - 4s 665us/step - loss: 8430.3018 - val_loss: 8327.1582
Epoch 62/100
5368/5368 [==============================] - 4s 662us/step - loss: 8426.7949 - val_loss: 8309.1436
Epoch 63/100
5368/5368 [==============================] - 3s 614us/step - loss: 8426.8271 - val_loss: 8330.3447
Epoch 64/100
5368/5368 [==============================] - 3s 603us/step - loss: 8426.3955 - val_loss: 8354.3359
Epoch 65/100
5368/5368 [==============================] - 3s 597us/step - loss: 8420.3408 - val_loss: 8323.3135
Epoch 66/100
5368/5368 [==============================] - 3s 588us/step - loss: 8420.8125 - val_loss: 8309.2461
Epoch 67/100
5368/5368 [==============================] - 3s 608us/step - loss: 8415.7012 - val_loss: 8393.2373
Epoch 68/100
5368/5368 [==============================] - 3s 607us/step - loss: 8414.1836 - val_loss: 8317.5371
Epoch 69/100
5368/5368 [==============================] - 3s 624us/step - loss: 8417.4238 - val_loss: 8319.2627
Epoch 70/100
5368/5368 [==============================] - 3s 627us/step - loss: 8414.3291 - val_loss: 8313.1055
Epoch 71/100
5368/5368 [==============================] - 3s 623us/step - loss: 8411.7275 - val_loss: 8326.7549
Epoch 72/100
5368/5368 [==============================] - 4s 671us/step - loss: 8411.1895 - val_loss: 8336.1279
91.15453301754879
```

After some consideration, we decided to even further refine what we considered to be the "average" Airbnb user. We dropped all prices above $250, and we dropped all features with the exception of our "one-hot" room types, latitude, and longitude, as an exploration to see what our results would be given this extreme feature manipulation of the data. The results turned out to be very good. Our random forest regression had an RMSE of $38.98, with a mean of $111.04 and a standard deviation of $54.98. While this may seem like a drastic change to our data, in reality we only dropped 19.59% of the total data, which is less than similar projects where the bottom and top quantiles are dropped as outliers.

V. Results

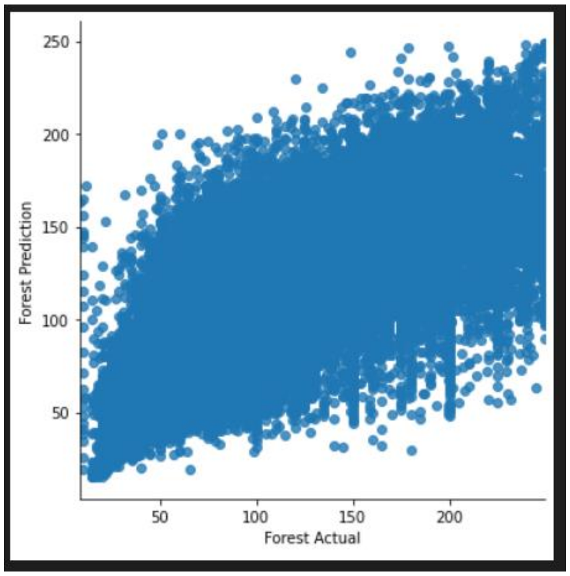
What answer was found?

At the end of our research we discovered that we could reliably predict the price of an Airbnb listing within one standard deviation from the mean. However, as we have seen, a single standard deviation can mean drastically different figures depending on how we allow our data to be structured.

In the initial run of our linear regression model on completely unclean data, our model was not able to predict the price of a listing within one standard deviation. However, this was the only model that was unable to do so throughout the entire project. Our random forest regressor performed much better, getting within our goal on its initial run of the unclean data. After we dropped only the most extreme values in the data set, which accounted for less than 1% of our total data, our model was immediately able to get within our goal using a linear regressor, without any hyperparameter tuning.

However, being able to predict the price of an Airbnb listing within a $500 margin is not very useful. So, after realizing that our goal of being within a single standard deviation was too broad, we began to refine our question so that we could produce a more satisfying and useful result for the average user. We looked at the mean value for our data set, which was roughly $220 with our unclean data, and discussed how much people would usually spend on an Airbnb, and then decided to try to get our model to be able to more closely predict the values of Airbnb listings that most people would actually rent. After dropping every listing that fell outside of what we considered to be a normal range of $0-$600, we began to see our model getting closer to the results that we wanted and expected.

This model ended up with a RMSE of $78.81 and a mean of $148.66 using a random forest regressor. We were happy with this, however we wanted to see if we could push it a little further for an even better fit if we focused only on properties that were around our original mean value of $220. After dropping everything above $250 and hyperparameter tuning our random forest regressor we were able to make predictions with a RMSE of $38.99 with a standard deviation of $54.98 and a mean value of $111.04 which we felt was a pretty good range for the predictions we were trying to make for our data set. Those results can be seen in the graph below.



What did the study find?

In the process of dropping more and more outliers and going through our feature engineering process that has been previously discussed, we discovered that the location and the type of property that is being rented are the most important determinants for the accuracy of our model. After having discovered this and creating a model that can predict the price of a model with a RMSE of only $38.99, we decided to test our model against a similar data set for Airbnb data in Seattle from four years before (2016) our initial data was recorded.

After cleaning up the new data set to a point where we could make predictions for it, we ran our model on it. On the new data set, which had a mean value of $127.97, our model had an RMSE of $89.97, which was right below the standard deviation of $90.24. We concluded that this was not a bad result for just trying to fit our model to another data set with no changes to the model. Additionally, a worse result should be expected when creating a model based on data that are constantly varying with time, such as rent prices – we know from our introduction how much housing prices can rise in a relatively short period of time. However, these results would not be good enough for implementation in a business setting and would need a great deal of work to be done for our model to be able to predict prices of rental properties over larger stretches of time.

VI. Discussion

What might the answer imply, and why does it matter?

Our results imply that it is very possible to create a model that can predict rental prices. However, the limited knowledge that we have about the properties in our data set make it hard to get any closer than a single standard deviation. This matters because if we were to continue this project and attempt to turn it into a business proposition we would need and expect a greater level of accuracy.

That greater level of accuracy would likely come from getting more features associated with the data we were given. It would also likely require some form of sentiment analysis for many of the features since rental values are based heavily on reviews and other people's opinions. If we were to continue with this project, we could see two paths leading forward. The first would be to continue using the same exact data set we were given and attempt to perform sentiment analysis on many of the features that we ended up dropping to help better fit our model. The second would be to gather more data that contained a greater number of features to look at. The second dataset from Seattle that we obtained contained 90 different features vice our 16.

If we decided to pursue the first path of continuation, then we would be looking at refining our model using some form of natural language processing. Since we dropped features such as the title of the post, the summary, and the reviews, we lost a large portion of information for each datapoint. The title could mention that the property has an ocean view, which is not expressed in our current dataset, or the review could mention that the ocean view is only visible over the city landfill which would probably have a great impact on the price of the property. We can see the inclusion of sentiment analysis having a great impact on our model.

If we decided to pursue the second path of continuation then we would need to go back and follow most of our initial steps for this project, but go into a greater hunt for important features. Even just being able to know how many bathrooms and bedrooms a property has would likely lead to a huge increase in the accuracy of our prediction. However, we would then need to spend much more time looking at each feature and doing much more data analysis then in our original project.

How does it fit in with what other researchers have found?

Our research generally fits in with what we have found with many other researchers findings. This is a well established problem that many have tackled such as Laura Lewis, a data scientist at Amazon, that ran a very similar, yet slightly less in depth project on similar data.[5] Graciela Carrillo, another data scientist working for the Scottish Department of Public and Urban Policy ran a similar project[6] and arrived at the same conclusion that we did, that predicting the average price of an Airbnb can be done with a reliable level of accuracy for the average property. They both also came to our same conclusion that the price depends most on the location of the property and the type of property that is being rented, and that after this having a multitude of features that better describe the property and the renter as a whole are helpful in making more accurate predictions.

What are the perspectives for future research?

---

[5] "Predicting Airbnb prices with machine learning and deep learning", Laura Lewis, https://towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-deep-learning-f46d44afb8a6

[6] "Predicting Airbnb prices with machine learning and location data", Graciela Carrillo, https://towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-location-data-5c1e033d0a5a

As we discussed, if we were to do a direct continuation of this research, we would likely either try to find more data sets with a greater variety of features that described the properties more in depth, or we would attempt to draw more meaning from some of the data that we dropped such as reviews, titles, and summaries. However, if we were going to branch from this research into a similar area, we could attempt to use neural networks to draw out context from the pictures that are provided for each listing. Another small adjustment that could be made to boost the accuracy of our model would be to use a voting classifier. This classifier would take a number of designated regressors of our choosing, and then average their predictions to possibly give a more accurate model.

If we had pictures for each of the listings in our original dataset, then we could attempt to build a Convolutional Neural Network to look at each and create new features for each listing. We could look for things that generally increase the asking price for Airbnb listings, such as if a kitchen is present, if there is a balcony, if there is a pool or hot tub, or if the property has a view of anything impressive. Building this CNN would be a full project in and of itself, but if it could return consistent results and the results were used to add features to our existing data set it would likely increase the accuracy of our model by a significant amount without us having to limit our price range.

Survey about the tools investigated for this assignment.

Our most used tools for this project were the LinearRegression, RandomForestRegressor, and SGDRegressor classes from the SKLearn library. We also used the XGBRegressor class from the xgboost library, the tensorflow library, and the seaborn visualization library. We used the SKLearn library to quickly figure out what our most important features were for our data by creating a model for each of the different regressors. We then graphed the results of our predictions compared to the actual labels for each type of regression using the seaborn library and then refined our model using what we learned. After we had created a well-groomed data set we wanted to create a neural network using tensorflow and hopefully have it improve on our base regressor classes. However, it turned out that our SKlearn regressors in fact consistently produced more accurate models that took much less computation power and time.

VII. Division of Labor

We found that we worked most effectively together, rather than separating the project into equal parts for us to work on separately. We frequently used Zoom to collaborate and analyze our data, deciding how we wanted to clean and munge our data set, as well as what models to use. We would then try out several different models, both of us running different models on our computers and compare the results. From there, we would compare our results, adjust our models and data, and repeat the process. We would frequently end collaboration sessions by running a regression that we expected to take a very long time, like a large grid search, to run over night so we would have the results in the morning. Due to Micky's hardware constraints, the more computationally demanding tasks were run on Cameron's computer.

VIII. Summary

In conclusion, it is possible to create a model that can, within a certain range of accuracy, predict the price of an Airbnb listing. We found that the most important thing we could do to increase the accuracy of our model was limit the price range on which we were attempting to make predictions. After implementing the removal of outliers, we found that we were able to get marginally better results by dropping features that repeated information found in other features, such as location data. However, keeping and encoding features that give you more information about the location of the property (i.e. more than just its latitude and longitude), such as which neighborhood the property is in, resulted in an increase in our accuracy. From there, tuning our regression model's hyperparameters led to an additional, if small, increase in accuracy. Ultimately, we found that while we could achieve good results on our dataset, our model failed to extrapolate to other Airbnb data sets that were not created in the same time frame. A more holistic model that can account for user sentiment as well as the continuous change in housing prices over time would likely prove more effective.