Predicting Airbnb prices in Washington D.C.

Daniel Park

DATA 606

Spring 2020

**Intoduction**

The project I have decided to work on this semester is to predict Airbnb prices in Washington D.C. using different machine learning tools. Airbnb is an internet marketplace where people can rent out their property for a fixed rate per day. The issue with this is that Airbnb doesn’t provide an algorithm for its hosts that can predict the price they should put for their property. It is the host’s responsibility to determine the worth of their property. For this project, I received data from a website from insideairbnb.com, a website (not affiliated with Airbnb) that provides data of different properties based on city. They have data from cities all over the world. For my project I decided to work on the city of Washington D.C. because of how close it is to me and to UMBC. One of data file is called listings.csv, which is the most important one. This file provides the ID, accommodation, bathroom, daily rate, location, etc. For my project, I will be cleaning out the data, provide an exploratory data analysis, and apply five different models into my data.

**Existing Approaches**

After looking through similar projects, I noticed that there are many similarities. Just like me, each author used insideairbnb.com for the source of their dataset. Insideairbnb.com is an anti-Airbnb lobby group that scrapes Airbnb data from multiple cities around the world. The dataset that is given by insideairbnb consists of 84 columns with thousands of rows. The dataset for Washington D.C. has 9,153 rows. That is a lot of data to process. To make things easier, the authors transformed and cleaned the data by removing most of the columns and keeping the features they feel were important. Some features that all the authors kept are accommodates, bedrooms, bathrooms, beds, price, minimum\_nights, maximum\_nights, and number\_of\_reviews. These were proven to be important features that determined prices throughout the projects. From what I’ve been seeing, the preliminary exploratory data analysis is very similar to each other. Some authors created graphs based on the data they initially found, like number of beds, accommodates, etc. One author (1) went ahead and created visualizations based on zip codes which I think is very interesting and clever. Another author (4) started with very basic machine learning models and applied them to only one feature (number of bedrooms vs price). These machine learning models involved KNN (K-nearest neighbor) and using RMSE (root mean square error) to evaluate the model. I plan on implementing simple machine learning models myself and putting in visuals to allow the audience to better understand what I am trying to accomplish for this project. More details will be present in part three of the project.

**Exploratory Data Analysis**

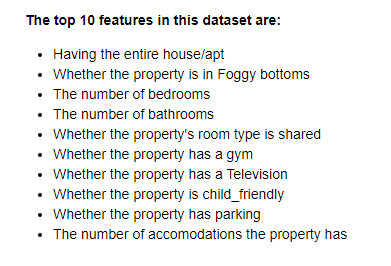
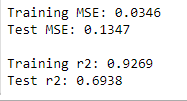
After looking into my data even more, I was able do a full data exploratory analysis on my data. I did this by graphing all my numerical columns to see what kind of trends were present. It was no surprise to me that almost all of them showed a skewed pattern, mainly because most of the properties share the same number of a certain feature (like bathroom, bedroom, beds, accommodations, etc. By looking at the graph, it would be hard to apply models to such data because of their skewed patterns. In order to make it more distributed, I would have to apply log transformation to each graph, which is exactly what I did. Once I did that, I applied One Hot Encoding, which basically converts categorical variables into a format that makes it easier for ML algorithms to read, which will be explained in my presentation. However, there were so many different types of variables in property\_type and amenities that I had to format them even more in order to make it easier to read and have more accurate results. Once all of the data was cleaned out, I created a Multi Collinearity Heatmap to see what kind of relationships other features have towards each other. This is important to understand because when two features show a strong relationship with each other then they are considered either positively or negatively correlated. This will evidently cause a skew in the overall output. By looking at the heatmap, I see that there are multiple strong relationships between features like bedrooms and price, price and guests\_included, etc.

**Implementation**

In my project, I applied five models: XGBoost, a three layer neural network, a three layer neural network with L1 regularization, a three layer neural network with L2 regularization, and a three layer neural network with Dropout regularization. First, we’ll start with XGBoost.

**XGBoost**

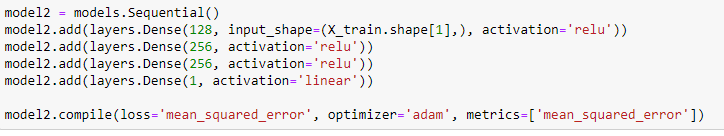
XGBoost stands for eXtreme Gradient Boosting. It is a ML algorithm that uses decision trees to make predictions. In order to start, I had to apply One Hot Encoding to my dataset using the pd.get\_dummies() function. Once that was set, I set my X variable to my dataframe that did not involve the ‘price’ column, and my y variable as the price column itself. Once that was set, I split my data into testing and training, where the test size was 0.2. After running XGBoost I got a training MSE (Mean squared error) of .0346 and a test MSE of .1347.. For r2 (measurement of how close the data is to the fitted regression line), I got .9269 for the Training and .6938 for the test, which are promising numbers for the first model. Once the results came in, I went ahead and acquired the feature importance of the dataset. The purpose of feature importance is to show how valuable each feature was to construct the model that created the decision. I was able to create a chart that showed the feature importance of every single feature in my dataset. Through that I was able to find the top 10 features that gave the most importance. However, by changing the random\_state variable, the answers may change.

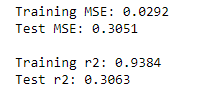
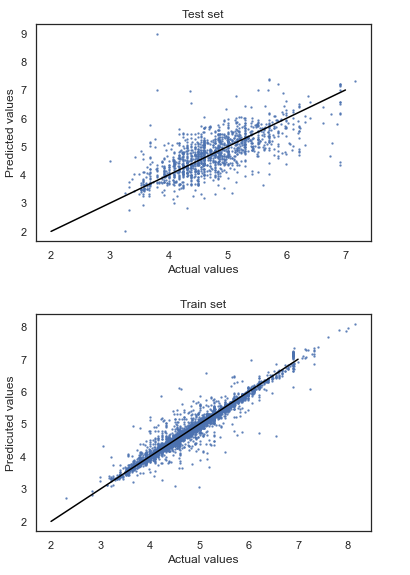
There’s no surprise that number of bathrooms, bedrooms, and having the entire house/apt to yourself is on the list because that is what people usually look at first when they look for property, especially if there are multiple people joining the Airbnb.

**Three Layer Neural Network**

The second model I decided to work on is a simple three layer neural network.



I used the code above to set up the neural network, which allowed me to fit the model. I used 100 epochs and a batch size of 256. For those who aren’t familiar with epochs, one epoch is when an entire dataset is passed through the neural network once. We need to use multiple epochs to get a more accurate result. By running through the data, we were able to find results. ReLu stands for rectified linear unit, which is used to output data if it is positive, otherwise it will be zero.

The Training r2 for this model is better than XGBoost. However we see a significant decrease for the r2 value, which shows that this model wasn’t effective at all for predicting prices.

**Three Layer Neural Network with L1 Regularization**

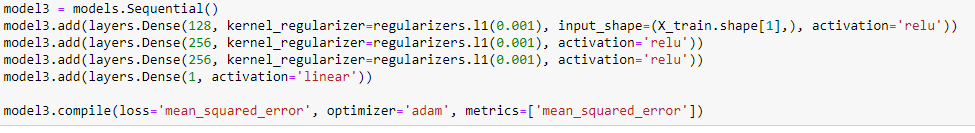
Next, I will be using the same three layer neural network using the L1 Regularization. The L1 Regularization is a common type of regularization. It updates the general cost function by addint another term known as the regularization term [5].

Cost function = Loss (say, binary cross entropy) + Regularization term

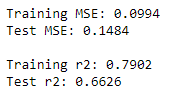
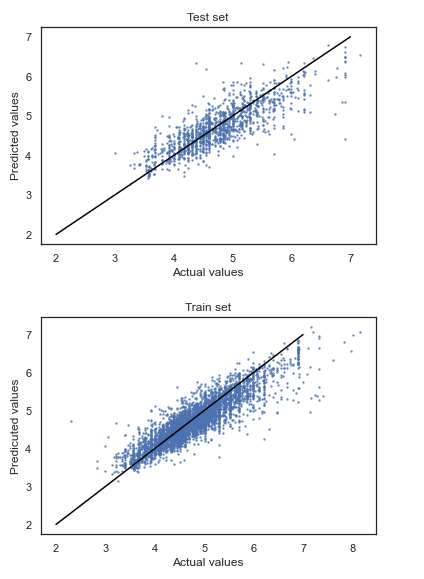
For L1 regularization, this function is used:



This function is used to penalize the absolute value of the weights [5].



Once again, I used 100 epochs and a batch size of 256 and fitted the model.

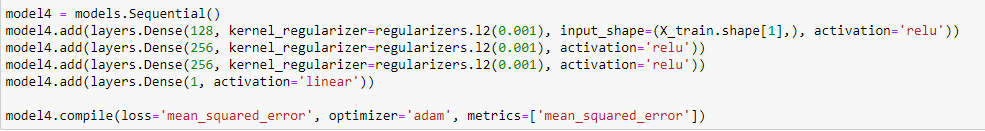
Compared to the normal neural network, the training r2 went down, but the test r2 went up significantly. This shows the L1 regularization was able to successfully explain more of the price we are trying to estimate.

**Three Layer Neural Network with L2 Regularization**

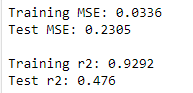
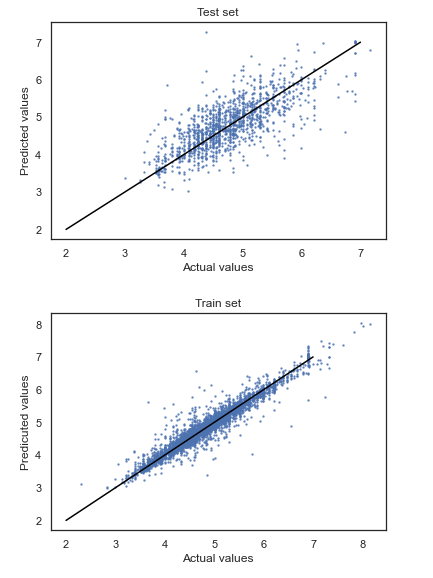
My fourth model will be the same neural network, except I will be applying L2 regularization. For L2 regularization, the function is used:



Here, lambda is the regularization parameter. It is the hyperparameter whose value is optimized for better results. L2 regularization is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero) [5].



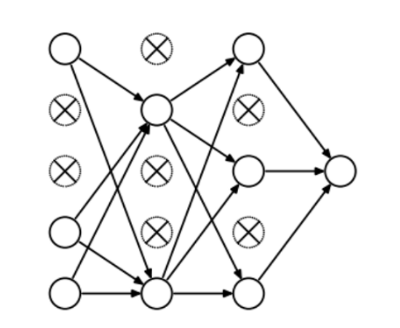
Once again, I used 100 epochs and a batch size of 256 and fitted the model.

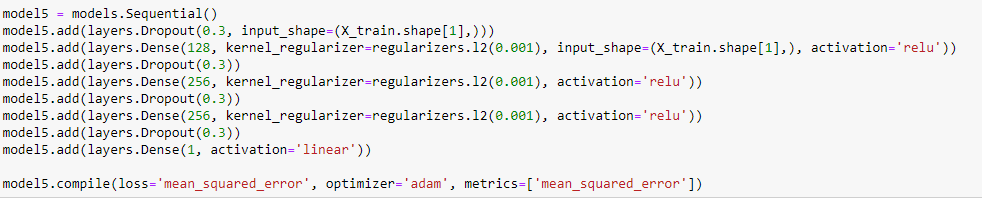
Compared to the third model, the training r2 went up once again. However, the test r2 went down again by a significant amount.

**Three Layer Neural Network with Dropout Regularization**

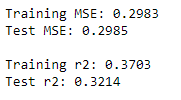
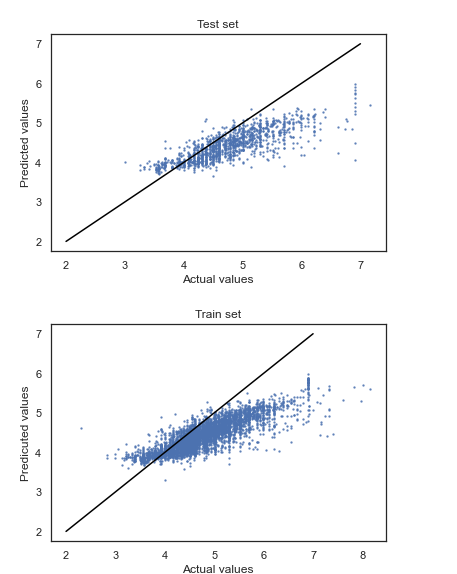
The final model I will be using is a three layer neural network with the dropout regularization method. The dropout technique is very interesting, and it is my personal favorite out of the five models I used. The dropout regularization technique will randomly select nodes and remove them along with all their incoming and outgoing conenctions [5].



For this model, I only chose to dropout only 30% of the nodes.



Once again, I used 100 epochs and a batch size of 256 and fitted the model.

Unforunately, the training r2 and test r2 aren’t good at all, so this model was deemed to be useless in my project.

**Conclusion**

By looking at the four variables (Training MSE, Test MSE, Training r2, and Test r2), here are the results, listed from best to worst.

|  |  |  |  |
| --- | --- | --- | --- |
| **Best training MSE** | **Best test MSE** | **Best training r2** | **Best test r2** |
| **Model 2** | **Model 1** | **Model 2** | **Model 1** |
| **Model 4** | **Model 3** | **Model 4** | **Model 3** |
| **Model 1** | **Model 4** | **Model 1** | **Model 4** |
| **Model 3** | **Model 2** | **Model 3** | **Model 5** |
| **Model 5** | **Model 5** | **Model 5** | **Model 2** |

Based on the five models I have worked on, it seems that XGBoost worked the best. Even though XGBoost was the highest out of all of them, the validation set still had a r2 of .6938. This means that the best model that we used (XGBoost) was only able to explain about 69% of the price we're trying to estimate. The other 31% could be from data that I removed in the beginning (The columns I viewed as unnecessary for the project).

**References**

1. *Airbnb Pricing Predictions*, airbnb-pricing-prediction.herokuapp.com/index.html.
2. Brownlee, Jason. “A Gentle Introduction to XGBoost for Applied Machine Learning.” *Machine Learning Mastery*, 21 Apr. 2020, machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.
3. Carrillo, Graciela. “Predicting Airbnb Prices with Machine Learning and Location Data.” *Medium*, Towards Data Science, 29 Jan. 2020, towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-location-data-5c1e033d0a5a.
4. DevlinData, Josh. “Python Machine Learning Tutorial: Predicting Airbnb Prices.” *Dataquest*, 10 July 2019, [www.dataquest.io/blog/machine-learning-tutorial/](http://www.dataquest.io/blog/machine-learning-tutorial/).
5. Jain, Shubham. “An Overview of Regularization Techniques in Deep Learning (with Python Code).” *Analytics Vidhya*, 19 Apr. 2018, www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/.
6. Lewis, Laura. “Predicting Airbnb Prices with Machine Learning and Deep Learning.” *Medium*, Towards Data Science, 22 May 2019, towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-deep-learning-f46d44afb8a6.