US Airbnb Nightly Price Prediction

Tegan Ayers

Abstract

In recent years, Airbnb has created a new market in travel and hospitality and has grown to become a $38 billion industry (Deane, 2021). As of January 2021, there are 2.1 million Airbnb hosts worldwide (Deane, 2021). One may reason that much of the success, or profitability, of these hosts stems from having an appropriately priced listing. As such, the objective of this project was to create a model that predicts the nightly listing price of US-based Airbnb's given a set of 16 attributes, ranging from host information, location, availability and past reviews. One application of this model could be to help Airbnb hosts to determine the correct nightly price for their home based on other available listings.

The US Airbnb Open Dataset (Kritik, 2020) was obtained from Kaggle. The dataset contains 16 attributes, consisting of categorical, numerical, continuous and discrete feature types. Data was appropriately cleaned and five different supervised models (Linear Regression, K-Nearest Neighbors, Decision Tree, Random Forest and a Neural Network) were trained and evaluated on the cleaned dataset. Results showed that the Random Forest algorithm produced the best results using the default parameters within the scikit-learn implementation of the model. As such, this model was further tuned using GridSearchCV in order to find the optimal set of hyperparameters for this task. Ultimately, the model was able to produce an accuracy, as measured by the coefficient of determination (R^2), of 0.90 on the training set, but only 0.36 on the test set, suggesting significant overfitting.

There are several areas for continued improvement of this task, namely in data cleaning and feature engineering. While my results are not exceptional, I believe in doing this project I gained an intuition for the models available our supervised learning toolkit and had the opportunity to create an end-to-end data processing pipeline, which was extremely valuable learning.

Introduction

Supervised learning is the facet of machine learning which seeks to construct a hypothesis for a dataset given a training set of labeled examples. It can be applied to several datasets, and is used in several industries today, from bioinformatics to marketing and sales.

Several supervised learning algorithms exist today, with more complex algorithms continuing to be built on a few core algorithms. These algorithms can be categorized by their method of making a prediction (discriminative versus generative models), the parameters required to build a model (parametric versus non-parametric models) or the types of problems they are able to solve (regression or classification). Examples of models used for regression tasks include Linear Regression, K-Nearest Neighbors, Decision Trees, Random Forest and Neural Networks.

Not every supervised learning algorithm will perform equally well for every given dataset. Rather, a model should be chosen based on the features and properties of a particular dataset. As such, to gain an intuition for the pros and cons of some of the most popular supervised learning models, several regression models were applied to the US Airbnb Open Dataset in an effort to predict nightly listing prices of US-based Airbnbs.

Background

Descriptions of the US Airbnb Open Dataset and the regression algorithms applied to the dataset are provided below.

Dataset

The US Airbnb Open Dataset was downloaded from Kaggle. It contains 226,030 observations, 16 attributes and a single target variable, nightly listing price. Given that the predictor (i.e. nightly listing price) is a continuous variable, this was considered a regression problem.

One of the interesting facets of this dataset is the various types of attributes that are present in it. Features can be categorized as categorical or numerical, discrete or continuous, or as their measurement type: nominal, ordinal, interval or ratio. Each of these different types of variables present different preprocessing challenges and may require different handling. Table 1 shows the categorization of each of the features within the Airbnb Open Dataset. Exploratory data analysis (EDA) and the data cleaning steps taken for each feature are explained in future sections.

Algorithms

The following regression algorithms were applied to the US Airbnb Open Dataset. All algorithms were instantiated using the scikit-learn python library (Pedregosa et al, 2011).

Linear Regression

Linear Regression is a parametric method that assumes a linear relationship between the features and the target

variable. That is, the hypothesis is assumed to be of the form:

$$h(x) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots \omega_d x_d$$

When a dataset contains several observations, such as in the US Airbnb Open Dataset, stochastic gradient descent can be used for efficient training and determination of the weight values. This method utilizes the following weight update rule:

$$\omega \leftarrow \omega + \alpha(y_i - h_\omega(x_i))x_i$$

Linear regression performs well when features are linearly correlated to the target. However, in most real-world datasets, this assumption does not prove to be true.

Table 1: Features present in the US Airbnb Open Dataset

Feature Name	Discrete or Continuous	Data Type	Scale
Listing ID	Discrete	Numerical	Nominal
Listing Name	Discrete	Text	Text
Host ID	Discrete	Numerical	Nominal
Host Name	Discrete	Text	Text
Host Listings Count	Continuous	Numerical	Ratio
Neighborhood Group	Discrete	Categorical	Nominal
Neighborhood	Discrete	Categorical	Nominal
Latitude	Continuous	Numerical	Interval
Longitude	Continuous	Numerical	Interval
City	Discrete	Categorical	Nominal
Room Type	Discrete	Categorical	Nominal
Minimum Nights	Continuous	Numerical	Ratio
365 Availability	Continuous	Numerical	Ratio
Number of Reviews	Continuous	Numerical	Ratio
Last Review	Continuous	Numerical	Interval
Reviews per Month	Continuous	Numerical	Ratio

K-Nearest Neighbors (KNN)

The KNN algorithm is a non-parametric model and considered to be a "lazy learner". This means that the algorithm does not fit a model to the data during training, rather all computations occur during prediction. The default distance metric for this algorithm is Euclidean distance (Minkowski distance with p=2) and a KD-tree is utilized for efficient storage of each data point's nearest neighbors.

KNN typically does not perform well with high-dimensional or sparse datasets. This is important to keep in mind as categorical features are often one-hot encoded and in turn create sparse matrices, which may affect model performance.

Decision Tree

Decision trees are also non-parametric models, meaning they assume nothing about the underlying distribution of the data. They're typically built using the Gini Index to determine the best split, or the split that achieves the most reduction in impurity. In mathematical notation, the decision tree algorithm seeks to maximize the following:

$$\Delta_{Gini} = Gini(p) - \sum_{k=1}^{K} \frac{n_i}{n} Gini(v_i)$$

Decision trees are extremely susceptible to overfitting, especially if all leaves are allowed to contain a single sample. As such, pruning should be used to help reduce the effect of overfitting.

Decision trees typically perform well with datasets that contain different data types and can handle missing values or redundant data.

Random Forest

The Random Forest algorithm builds an ensemble of Decision Trees by sampling the observations with replacement. Each tree is created using a subset of the data and makes a prediction. These predictions are then aggregated using majority voting or averaging. Random Forests typically outperform a single Decision Tree and are less susceptible to overfitting.

Neural Network (Multi-Layer Perceptron)

A multi-layer perceptron model consists of at least one hidden layer and an output layer. They are typically trained using forward and backward propagation followed by gradient descent.

Neural networks are considered the best-in-class supervised learning techniques, however, their complex structure and several hyperparameters make them hard to train and hard to interpret, thus often getting thought of as black boxes.

While the scikit-learn implementation of the MLPRegressor was utilized for this work, Tensorflow is considered the better library for more complete and flexible neural networks.

Project Description

The scope of this work was to apply the aforementioned regression algorithms to the US Airbnb Open Dataset in order to predict nightly listing prices. Results are compared below, followed by a discussion of the pros and cons of each algorithm and a comparison to other approaches that have been tried on this dataset (published on Kaggle).

Results

First, a discussion of the EDA and data cleaning steps are described below. Then, results for each of the individual

algorithms are shown followed by an in-depth discussion and summary of the takeaways.

Exploratory Data Analysis (EDA)

EDA steps included calculating summary statistics (e.g. max, min, count, frequency) plotting the distribution of each variable, plotting relationships between variables (i.e. pairwise plots) and determining if any features could be eliminated. A few interesting findings came out of this exercise.

First, in plotting the distribution of each variable (not shown here) it became apparent that several variables were positively skewed with an extremely long tail. Most notably, "minimum_nights" had an extreme value of 100,000,000 suggesting that it was noise in the dataset and should be eliminated.

The distribution for nightly listing price also revealed a positively skewed distribution with a long tail, with a maximum listing price of $24,999 per night. This prompted further analysis to understand how many values were contained within the tail. Table 2 reveals that approximately 2% of the listings have a nightly price above $1,000. This begs the question of what should be considered an outlier, and therefore eliminated, from this analysis. It was ultimately decided to eliminate all observation with a nightly price above $5,000. Once values above $5,000 were eliminated the average nightly listing price was calculated to be $198.49.

Table 2: Nightly Listing Prices Above Specific Thresholds

Listings above $1k per night	2.03%
Listings above $5k per night	0.21%
Listings above $20k per night	0.01%

Next, pairwise plots were constructed between each continuous feature and the target variable, nightly price, in order to determine if any linear correlation existed (Figure 1). It was observed that no linear relationships exist, thus suggesting that Linear Regression will likely perform poorly on this dataset.

A Heatmap of the City variable was plotted using the Folium library in an effort to try different types of visualizations (Figure 2). While a count plot reveals similar information, the Heatmap is a nice, visually-pleasing way to present data. From the Figure, it can be ascertained that the highest density cities containing Airbnb's are NYC, Boston, the Twin Cities, Los Angeles and Las Vegas.

Finally, the number of missing values were calculated for each feature, which revealed that some features contained over 51% missing values. Results are shown in Table 3.

At the conclusion of exploratory data analysis, it was decided to eliminate 5 features. Listing ID was eliminated because it was unique to each sample and therefore did not contain any useful information. Listing Name and Host

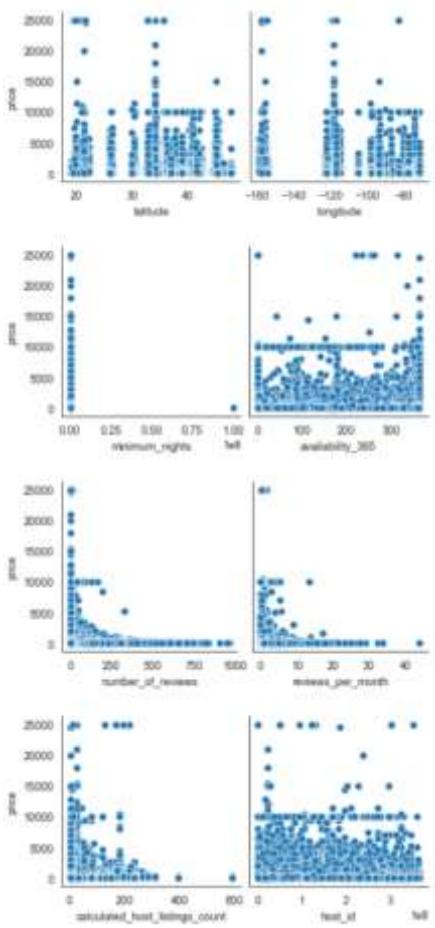

Figure 1: Pairwise plots between continuous variables and target variable, nightly price.

Name were eliminated because these were free text variables that could not be easily translated into usable features. And finally, Neighborhood Group and Neighborhood were eliminated because they contained several missing values and the information was better captured in the City variable.

Figure 2: Heatmap of City feature showing hotspots in NYC, Boston, Twin Cities, Los Angeles and Las Vegas.

Finally, EDA also helped to reveal the appropriate data cleaning steps needed for each feature, discussed in detail in the next section.

Table 3: Percentage of Missing Values per Feature

Feature Name	Percent Missing Values
Listing Name	0.01
Host Name	0.01
Last Review	21.5
Reviews per Month	21.5
Neighborhood Group	51.3

Data Cleaning

Several data cleaning steps were performed using various scikit-learn transformers and built-in pandas functions (McKinney, 2010). The data cleaning pipeline was as follows:

1. Drop features deemed unnecessary or redundant.
2. Drop extreme outliers (e.g. Minimum Nights > 200, Nightly Price > 5000).
3. One-hot encode categorical variables (i.e. Room Type and City) and drop original column.
4. Discretize Latitude and Longitude.
5. Convert dates (e.g. Last Review) to three new features (day, month, year) and drop original column.
6. Fill any remaining missing values with the mean of the column.
7. Scale all features using robust scaling.
8. Prinicipal component analysis (PCA) to ensure features are completely independent.

Prior to data cleaning the dataset consisted of 226,030 observations and 16 features. Post-cleaning the dataset consisted of 225,506 observations and 44 features.

Model Performance

Each regression model was applied to the dataset using the model's default hyperparameters and accuracy was measured using the R^2 coefficient of determination, which provides a measure of the model's prediction performance. A comparison of each model's performance is discussed below and summarized in Table 4.

Table 4: Model Performance

Model	Training Performance	Test Performance
Linear Regression	0.10	0.11
K-Nearest Neighbors	0.35	0.30
Decision Tree	0.99	-0.11
Random Forest	0.90	0.30
Multi-Layer Perceptron	0.27	0.24

Linear Regression

Linear Regression forces a linear decision boundary, however, as discussed in the EDA section, none of the features in the Airbnb Dataset were linearly correlated with the target variable. As such, both training and test sets produced poor performance, with an R^2 performance of 0.10 and 0.11, respectively. This was the expected result and the model was only fitted in order to determine the performance of a linear regressor.

K-Nearest Neighbors

The KNN model (with default k=5) performed better than Linear Regression, with an R^2 value of 0.35 for training and 0.30 for testing. Interestingly, one-hot encoding the City variable, which contained 28 unique variables, did not affect the KNN model's performance even though KNN's typically suffer with high dimensional datasets (i.e. the curse of dimensionality). Additionally, the KNN model did not exhibit too much overfitting, which would have been apparent if the training and test set performance was significantly different.

This model was also tested with a value of k = 475, as the general rule of thumb for setting the value of k is as follows:

$$k = \sqrt{n} = \sqrt{225{,}506} \approx 474.8$$

However, using this value, model performance decreased to 0.26 (test set).

Decision Tree

As expected, without any pruning the Decision Tree model grossly overfit to the training data, as shown by an R^2 value of 0.99 for the training data, while the test set performance was -0.11. Pruning of the tree, by setting the "min_samples_leaf", "max_depth", or "min_samples_split" hyperparameters, would have likely improved this model's performance.

Random Forest

The default Random Forest (n_estimators=100) showed a slight improvement in overfitting compared to the Decision Tree, with R^2 values of 0.90 and 0.30 for training and test sets, respectively. This improvement in overfitting was likely due to bagging and the use of multiple estimators.

Neural Network (Multi-Layer Perceptron)

Interestingly, the Multi-Layer Perceptron model with default hyperparameters did not perform well during either training or testing. The R^2 values of 0.27 for training and 0.24 for the test set were both lower than the default KNN model. This is likely because the MLP model has several hyperparameters and requires extensive tuning to produce a model with good performance. Given the resourcing requirements needed for efficient training, it was decided to not pursue this model further.

Model Tuning

Upon examination of the results above, I decided to try to improve the results of the Random Forest model using hyperparameter tuning (GridSearchCV). This model already had high training performance (0.90), and it was tied for the highest test performance (0.30). Additionally, the model contains several hyperparameters that could be optimized for better performance. As such, GridSearchCV was employed to determine the best value for the following hyperparameters: "n_estimators", "max_depth" and "min_samples_leaf". Table 5 summarizes the results of this exercise, however, unfortunately performance of the test set was only able to be improved to 0.32 (from 0.30).

Table 5: Best Hyperparameters for Random Forest Model

Hyperparameter	Value
n_estimators	200
max_depth	15
min_samples_leaf	3

After observing this, one last attempt to improve model performance was tried by returning to the Data Cleaning step to improve upon the features being fed into the model. It was decided to eliminate all observations with a nightly price greater than $1,000, drop two more features (Host ID and Last Review) and to try ordinal encoding, rather than one-hot encoding, Room Type and City. In doing so, test performance was increased slightly further to 0.36.

Discussion

The objective of this project was to design a data processing pipeline for the prediction of US-based Airbnb nightly listing prices while also comparing the performance of various regression models to this task. Table 4 summarizes these results.

Some models performed as expected, such as Linear Regression (performed poorly because it's not a linear problem) and the Decision Tree (performed poorly due to overfitting). Alternatively, other models, such as KNN and MLP performed differently than expected. The KNN model performed better than expected and was, surprisingly, not effected by the sparsity of the feature space due to one-hot encoding. In contrast, the MLP model, which was expected to perform well, did not in fact do so, likely to a lack of hyperparameter tuning.

Finally, the Random Forest model performed somewhat as expected, with a high training performance, but suffering from the effects of overfitting. While this was attempted to be mediated through hyperparameter tuning, it was still not enough to get performance to a place of satisfaction. Rather, by returning to the data cleaning step and continuing to refine the input features, the model was able to obtain the largest performance boost. Therefore, a large takeaway from this exercise is that data cleaning, although one of the less glamorous steps in machine learning, is perhaps the most critical step in the pipeline for obtaining a high performing model.

Related Work

As this dataset was downloaded from Kaggle, there are a few open-source notebooks that have attempted to complete the same Airbnb pricing prediction problem. I looked at a couple of these after I completed my analysis and found that none of them jumped out as being far superior to my approach, however, a couple approaches included some additional interesting preprocessing steps that I had not considered. These included:

1. Using the "City" variable to create an additional "State" variable (I did not do any feature engineering).
2. More robust elimination of outliers.
3. Converting this into a classification problem by discretizing the target variable.

Future Work & Conclusions

All of these algorithms could have benefited from further data cleaning and feature engineering, as is often the case with most machine learning problems. This became especially evident as I tried to improve the performance of the Random Forest model. That is, hyperparameter tuning improved the Random Forest model's performance slightly, but refined data cleaning is what caused the most significant performance gain.

Although none of my model's performed as well as I would have hoped, I believe I learned a lot from this project. Most importantly, I was able to develop an end-to-end data processing pipeline. I used new scikit-learn transformers, such as the Pipeline transformer, and I learned just how powerful that library can be. Finally, I also gained an intuition for the pros and cons of various supervised learning algorithms, which is one of the main reasons that I wanted to come to graduate school in the first place.

References

Deane, Steve. (2021). 2021 Airbnb Statistics: Usage, Demographics, and Revenue Growth. Stratos, available online 19 April 2021. https://bit.ly/2RFaRUu

Kritik, Seth. (2020). U.S. Airbnb Open Data. Kaggle, available online 4 March 2021. https://www.kaggle.com/kritikseth/us-airbnb-open-data

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceeding of the 9th Python in Science Conference, pp. 56-61.

Pedregosa, et al. (2011). Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.