Andrew Vattuone
Lowest RMSE Score: 484.4782
Leaderboard Ranking: 16th
May 11, 2025

Program 2 Report

The main goal of this project was to take a collection of Airbnb listings and predict their price, based on many different features. Some of the features were categorical (like is_superhost), some numerical (like bathroom number), and some text (like description). Through this project, we were supposed to learn about various data preprocessing techniques (like one-hot encoding and TF-IDF), feature reduction, modeling, and even some data visualization techniques. After several iterations, I found a combination of several preprocessing, feature reduction, and modeling techniques that helped me to successfully predict the price of an Airbnb rental based on various inputted features.

I started out the project by using my Lab 3 code, as it used the same data as Program 2 and I had already done some preprocessing techniques on the data as part of the lab assignment. In my lab code, I had already categorized each feature column into one of three categories based on the data type (numeric, categorical, and text), and manually removed some of the columns I already knew wouldn't be useful (such as host_name) to save on runtime. I had also removed columns with a large number of missing values, although I had done this manually rather than using a code to set a threshold. Additionally, I had also cleaned all the data so that for numerical values, missing values were filled in with the mean of the corresponding training feature (I used the training mean for the test set to to prevent overfitting), for categorical features they were filled in with the word "missing", and text values were filled in with an empty string. I also found 2 features that were lists, so I turned those into strings. I had also applied scaling to the numerical features, one-hot encoding to the categorical features, and TF-IDF to the text features. At the very end, I also removed some outliers using the Interquartile Range Method and made predictions for the training values without using any labels and then compared them to the actual training labels to get the RMSE for the training data (this was just to test if it worked, not my actual prediction).

To edit this, I first made sure to write some code that automatically removed features where more than 50% of the cells had missing values. I also decided to remove outliers before I did any preprocessing the data so that they wouldn't distort other preprocessing techniques, and decided to remove the columns with largest and smallest 1% of the prices instead of using IQR to remove outliers. I also made sure to include "NA", "na", "n/a", "null", "None", and empty strings as missing values so that they would be properly accounted for once the program got to the missing values section it could deal with them properly. I decided to use PCA dimensionality reduction on all types of my features (numeric, categorical, and text) just to see how it would work out, and printed out how each feature contributed to each new PCA feature just to see what features were impactful. I didn't test without PCA but runtime would've likely been longer since it would've used more features. Afterward, I randomly split the training data so that 80% was used for training and 20% was used for validation. I then ran several models on the data, including linear, ridge, lasso, decision tree, random forest, gboost, and stochastic gradient descent, predicted the values for both the training set and the validation set, printed out their RMSEs for both training and validation, and plotted a graph of the results. I also did the same for a polynomial model with a degree of 2, but outside of the for loop since it needed extra setup. I accidentally messed up the stochastic gradient descent function inside the loop since I needed to process the data in a different way for stochastic than the other variables, but I just ignored stochastic for the time being and left the broken version in my submission since I wasn't going to consider stochastic. I then wrote a small piece of code to find the model with the smallest RMSE for the validation set, and then refitted the model to the combined training and validation set and used it to make my predictions for the test set, and wrote these predictions to

a file. Note that for my first submission I accidentally made a typo resulting in wildly inaccurate predictions that gave me an RMSE of 23323788.0121, but I fixed this error and got an RMSE of 484.4782 for my second submission (the best one happened to be random forest). I then did a third submission using predictions based on lasso just to see how much my validation vs actual RMSE differed, as my validation RMSE was only around 180 for my second submission but the actual was 484.4782. I got a higher RMSE of 524.7339. Afterward I slightly adjusted the random forest parameters like min_samples_leaf and min_samples_split to see if that led to an improvement, but that gave me a slightly higher RMSE of 490.4462.

I eventually realized that my RMSEs for the validaiton and training sets were so similar since I was accidentally fitting my validation sets on the model as if they were training sets. I quickly fixed this error and split my data into training and validation before any preprocessing was done rather than after, and treated my validation set as if it was a test set for all of my preprocessing steps. I saw that my validation RMSEs were much larger than my training RMSEs, so I realized that my model might be overfitting. Initially I tried changing some of the input variables for my models (I also removed stochastic and added XGboost), but that led to virtually no improvement in the validation RMSE. I then tried just doing PCA on numerical features and doing SVD on my categorical and text features, although the validation RMSE didn't improve, so I did not submit a new file. I thought that I had too many features, so I tried manually combining some like combining all of the review scores into a single review score. I also saw that many of my TF-IDF features were similar (like "pool" and "ocean") so I tried to cluster these together into similar groups by using so I attempted to cluster these together into related groups using KMeans.The cluster_text_features() function helped with this: it grouped together TF-IDF columns with similar prefix types (like amenities or descriptions), and used KMeans clustering to assign each word to one of a small number of clusters. It then transformed each row by summing the TF-IDF values of the words within each cluster — essentially reducing many sparse TF-IDF features into a few denser and more generalizable cluster-based features. I applied this transformation to my training, validation, and test sets using apply_cluster_mapping(). The result was a new feature set where redundant or overlapping TF-IDF features were replaced with grouped clusters like "amenities_cluster_0" or "description_cluster_2", which helped simplify the model input. I then applied PCA to both numerical and categorical features after doing this, and ran my models on the new data. I decided to take the 3 models with the lowest validation RMSEs and use VotingRegressor to create a combined ensemble VotingRegressor model to see if that led to improvements. The top 3 models I got were random forest, XGboost, and decision tree. I then retrained the ensemble model on just the training set and got an RMSE of 491.2367. I wanted to train on the combined training and validation set but I had to fix some of the validation data since it was treated as test data in the preprocessing, and the kernel kept crashing so I didn't have time to do this (although it likely would've led to improvements). I also tried a separate file without using the text clustering and instead just used PCA on the numeric and categorical and SVD on the text data, although when I did the same ensemble method I got linear regression as being better than decision tree. I got an RMSE of 493.7267 which was higher than expected, although if I had time to retrain on the combined validation and test data it might've been better. I decided to go with my original 484.4782 submission since that was the best.