Final Report

Introduction & Background

Literature Review

Airbnb pricing is influenced by location, listing characteristics, reviews, and automated strategies. One study shows that property type, location, and host characteristics significantly impact rental rates (Gibbs et al., 2018). Another indicates that negative reviews affect cheaper listings more and diminish for properties farther from city centers (Lin et al., 2024). Additionally, a study explains that automated pricing can enhance revenue in competitive markets like New York City (Fan, 2022). Airbnb's strategy uses binary classification and customized regression for optimal pricing, requiring tailored approaches for each listing (Ye et al., 2018).

Dataset Description

This dataset contains around 49,000 Airbnb listings in NYC, detailing price, neighborhood, property type, reviews, availability. Manhattan and Brooklyn dominate, potentially introducing geographic bias. "Entire home/apt" and "Private room" are the most common room types, which may also skew the model.

Dataset

View Dataset

Problem Definition

Problem

Our project aims to predict Airbnb prices in the growing rental economy, especially in high-demand metropolitan areas like NYC.

Motivation

Airbnb prices in New York vary greatly on neighborhood differences, rental types, and other features. Many hosts rely on competitive pricing strategies, often underselling properties, missing potential revenue. Guests struggle to find listings that provide good value for their needs. Given the fluctuations in prices, a way to predict optimal pricing would benefit customers by providing dynamic pricing suggestions that balance pricing strategies and customer preferences.

Methods

Data Preprocessing Methods

In terms of data cleaning and preprocessing, we performed several steps to ensure that our data had strong integrity and was ready to be inputted into our models. Firstly, we counted the number of NaN values for each feature, and here we identified that the host_name and name features have 16 and 21 null values respectively. We decided to remove these two columns from our dataset because their values would not have a significance in our prediction. Furthermore, we decided to convert the last_review column into an ordinal value because it would allow the dates to be processed by the linear and ridge regression models more efficiently. We then filled all of the NaN values with the earliest date in this column. Additionally, we set a cap on prices at the 99th percentile in order to remove extreme outliers in the price column.

Lastly, we noticed that our data is skewed, so we decided to apply log normalization to each column to achieve a more balanced distribution. We standardized our numerical features using a Standard Scaler to ensure consistent variance and centered data. For categorical features, we applied one-hot encoding, transforming them into binary columns that our models could process directly. To reduce dimensionality for high-cardinality categorical features like neighborhood and neighborhood_group, we kept only the top three most frequent categories and grouped the rest under an 'Other' category.

Supervised Learning Models

Model 1 - Linear Regression

We decided to utilize linear regression and ridge regression for our models. Linear regression aims to establish a linear relationship between features by minimizing the sum of squared differences between actual and predicted values. The model inherently assumes a linear relationship between variables and seeks the best-fitting line, even if it involves large coefficients. While this approach can be sensitive to outliers, it retains the complexity of the original data since it does not adjust for bias, as there is no regularization term.

Model 2 - Ridge Regression

To build off following the linear regression model we also decided to incorporate the ridge regression model in our data visualization. Ridge regression is a type of linear regression, but it includes a regularization term that helps control large coefficient values, allowing us to overall reduce overfitting by simplifying our model model and avoiding large deviation. This means that while ridge regression, like linear regression, aims to minimize the sum of squared errors, it also adjusts for outliers or extreme values in the data. This regularization helps to create a more generalized model, but it may also sacrifice some of the data's complexity or nuanced relationships.

After considering the two models we ensured to take into consideration the pros and cons and compare these two similar methods with the difference being that one considers overfitting risks and accounts to try and counter that possibility.

Model 3 - Random Forest

Unlike linear regression and ridge regression, random forests allow us to use a nonlinear model to predict AirBnb prices. Decision trees work by recursively splitting the data into buckets according to a feature and threshold on that feature. Multiple decision trees are then randomly chosen subset of data points with a randomly chosen subset of features to use. Linear regressions and ridge regression have risks of underfitting, because they can not represent more complicated nonlinear functions. On the other hand, decision trees have the risk of overfitting when there are lots of features because there may not be a clear way of interpolating prices on combinations of features it has not seen before. Random forests are an example of bootstrap aggregating. Multiple decision trees are trained together and the output of the random forest is the average of the results of the decision tree. There may be random fluctuations in the output of any single decision on data it has not seen before, but hopefully averaging over many different decision trees will lead to more stable outputs with less variance.

Model 4 - Support Vector Regression (SVR)

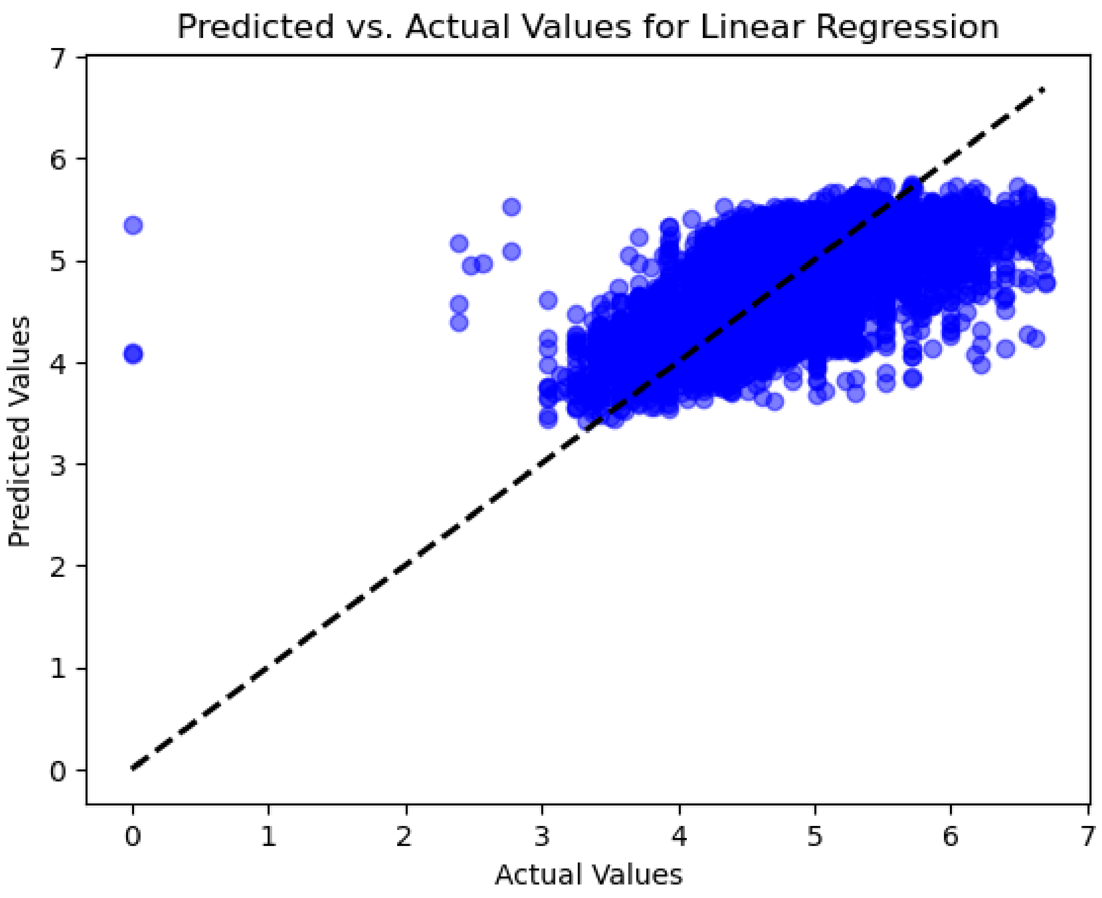
SVR is a machine learning algorithm that builds on the principles of support vector machines. This algorithm works by finding a function, generally linear, that predicts values within a certain margin of tolerance (epsilon) without penalty. Unlike linear regression, SVR is given more flexibility, via hyperparameters, on how it trains to fit points. One can choose epsilon, a hyperparameter which governs the distance to the hyperplane where it may tolerate any point without preference. One can also choose C, a hyperparameter between 0 and 1 which governs the tradeoff between a more simpler function or a more complex function that matches the data closely, but may potentially overfit. More generally versions of SVR can even model nonlinear functions if one chooses the kernel function to be gaussian, radial basis functions, or any nonlinear kernel function. But, even with a linear kernel, one can now capture subtle nonlinear relationships that are within epsilon of a linear function in a SVR. And unlike random forests, SVRs are simpler models that can generalize better without as large of a risk of overfitting. They are best on data that is close to linear with some nonlinear subtle relationships.

Results

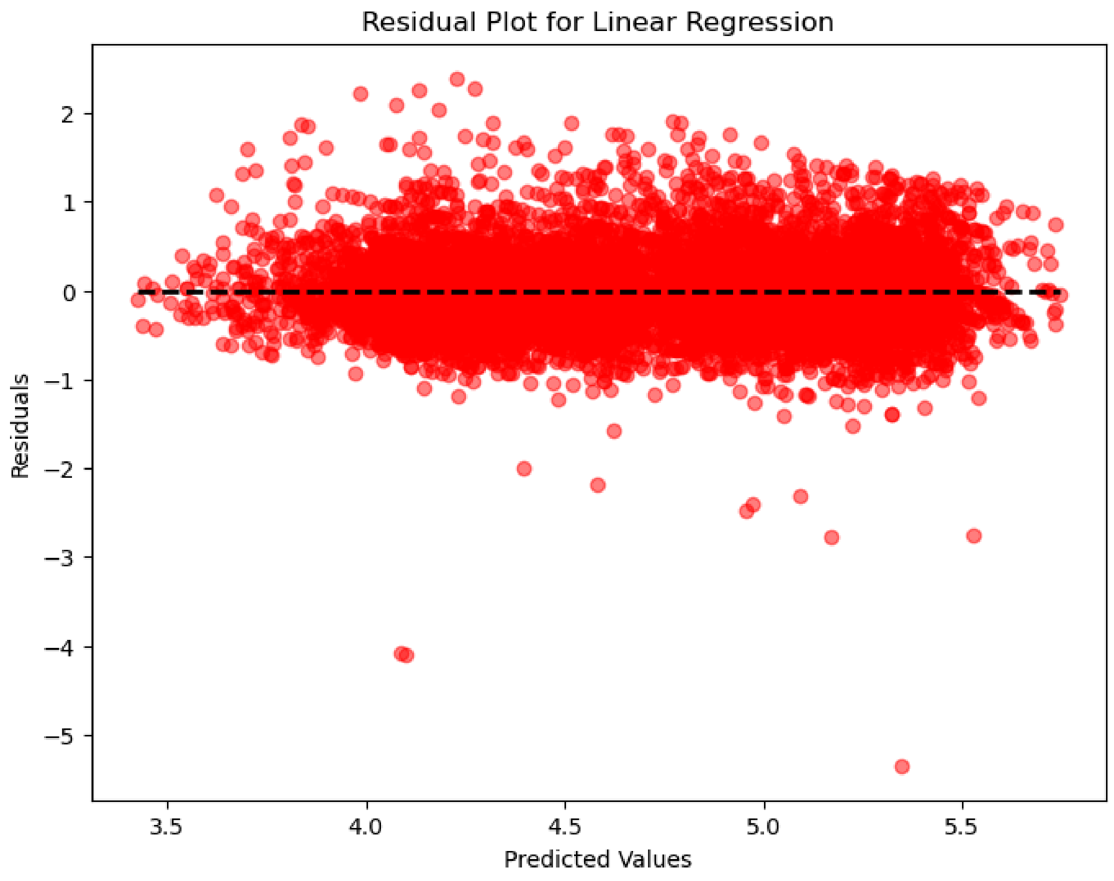
Model 1 - Linear Regression

The results from running our linear regression model are shown below. It is clear that from the residual plots, there is a deviation from the predicted to the actual values. This is measured in terms of the deviation from the diagonal line in plot 1, as we can see that the dots on both sides of the line clearly deviate from the center diagonal dotted line, indicating that there is error within the model’s predictions. Additionally, there are a few outliers within the model’s predictions. Observing the lower end of the actual values, from (0-3) we can clearly see that there is a divergence from the center diagonal, as the predicted values tend to be much higher for these points. Moving to the higher end of the actual values, some of the points deviate significantly from the center diagonal as well. This indicates that the linear regression model is not able to capture the relationships between features when the actual values are lower in this dataset, and that there is more complexity involved with the prediction that is not being reflected in a linear relationship between inputs and predictions. With the second residual plot, it’s clear that the majority of data points fall between -1 and 1 in a random scatter around the 0 line. This indicates that the spread of residuals is consistent across the range of predicted values.

We utilized several different metrics to evaluate the performance of the linear regression model. These included the mean squared error, mean average error, and root mean squared error. Our mean squared error, the average of the squared differences between predicted and actual values, was 0.197, which is quite low, indicating that the predictions generated by the linear regression model were generally close to the actual data points. This indicates that there aren't many significant outliers or large errors in the predictions. Additionally, our mean average error (MAE), which is the average of the absolute differences between the predicted and actual values, was 0.331. The mean average error does not penalize errors in predictions as heavily as the MSE or RMSE, and with a lower MAE, our model’s predictions are in alignment with their actual values. Lastly, we utilized the root mean squared error (RMSE), to determine the error in terms of the units of the inputs, while still penalizing errors but not as heavily as the MSE. Our RMSE was 0.4437, which is reasonable after log normalizing the data for the prices.



linear regression residual plot 1



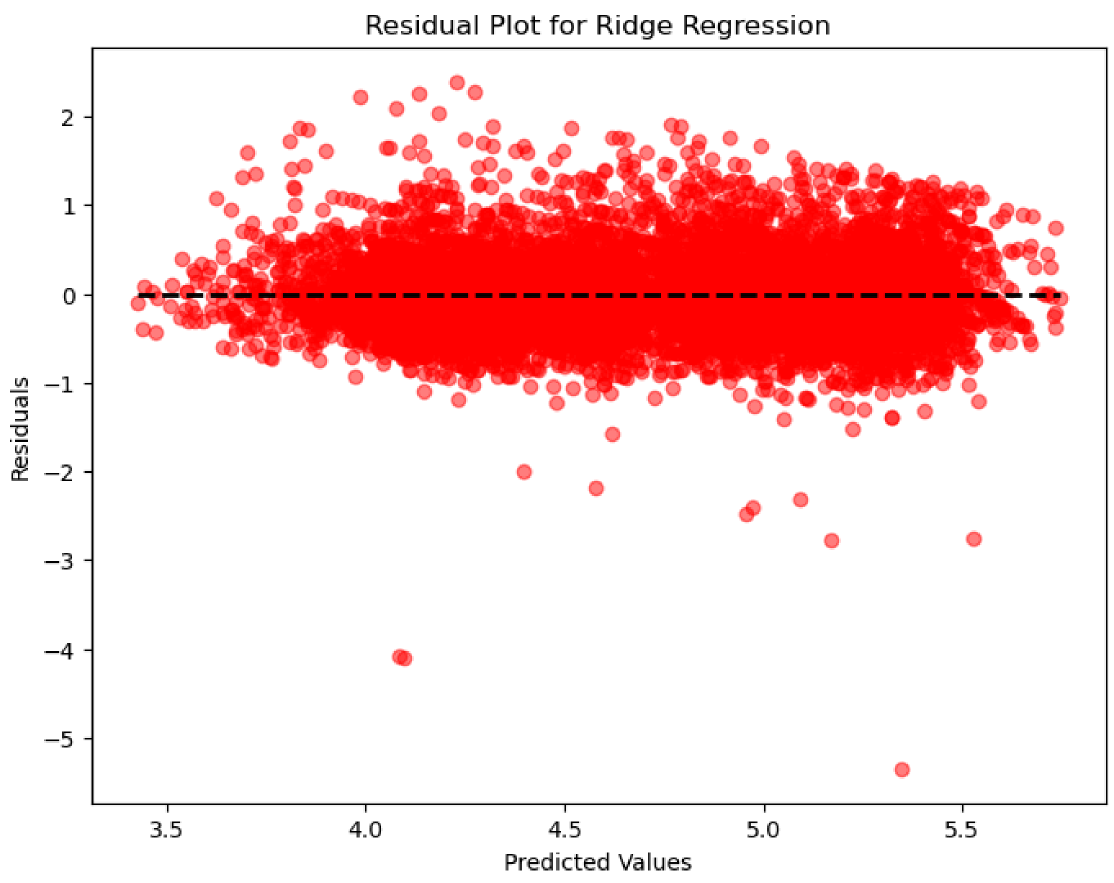
linear regression residual plot 2

Model 2 - Ridge Regression

The results from our visualization in the Ridge regression are shown below, the main concerns we found with both of these linear models is the assumption of linear relations between features even when there is a high chance that these features are non linear. This indicates that linear and ridge regression may not be the best model because prices are influenced by other more complex factors which can't be best represented through a linear model. Ideally, if the model were perfectly accurate, all points would lie along the dashed line to represent a perfect relation between predictions and true values. However, we see a significant deviation, especially when the actual values increase, indicating that our model is not able to predict accurately across the range. Given the deviation seen in the residual plots, the MSE would likely be higher due to outliers and errors at higher values.

The residual plot for the ridge regression model shows the error differences between the predicted and actual values which helps model and visualize the potential limitation in the linear assumption that Ridge regression imposes. When the underlying relationship is nonlinear or complex, linear models fail to capture the full dynamics and relationships between the features, as seen in our graphs.

The Mean Squared Error for our ridge regression remained nearly constant at approximately 0.19657 at lower less significant alpha values e.g. 0.01, 0.1, 1.0, 10.0 suggesting that low alpha values do not strongly influence our model's performance. However, as alpha increased, there was also a slight increase in MSE, with alpha = 100 giving us an MSE of 0.19665. This suggests that as we apply stronger regularization, the model may underfit the data, leading to minor increase in prediction error. At the highest alpha of 1000, the MSE rose to 0.20128, indicating that we have regularization that penalizes the coefficients too heavily, limiting the model's ability to capture the underlying patterns in the data and therefore ignoring correlation that are non linear. This increase in error at high alpha values reinforces the tradeoff in ridge regression between model flexibility and regularization strength.



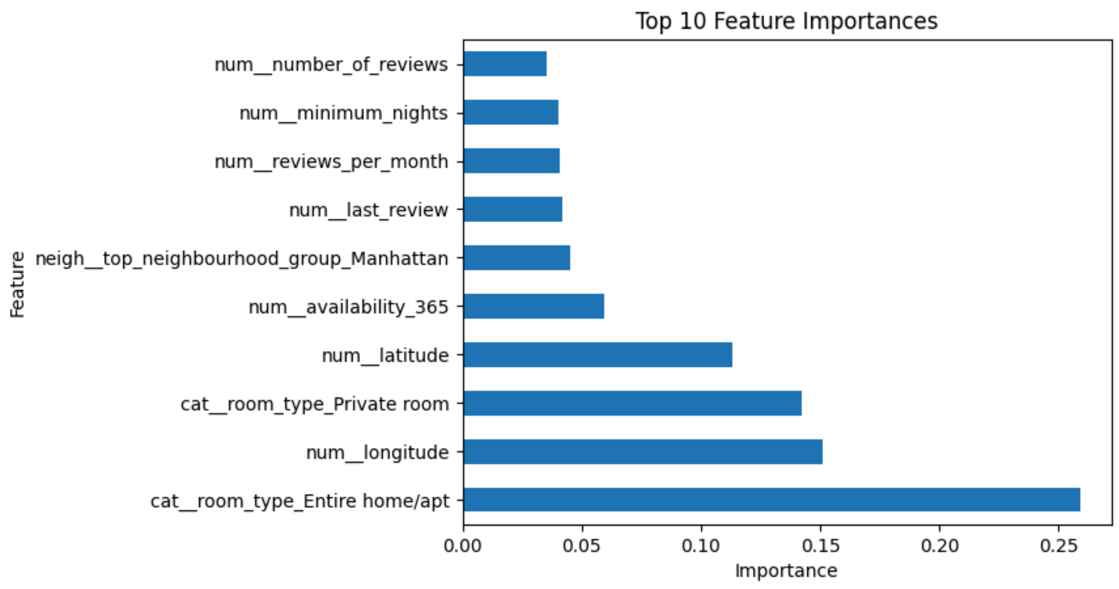
ridge regression residual plot 2

One thing we were surprised by was the similarities between the linear and ridge model that we implemented. We believe this is because they are both limited linear models which may not be best suited for a complex data set like ours. To improve performance, we will try and look into exploring more complex models that can capture nonlinear patterns such as decision trees, random forests, or neural networks.

Model 3 - Random Forest

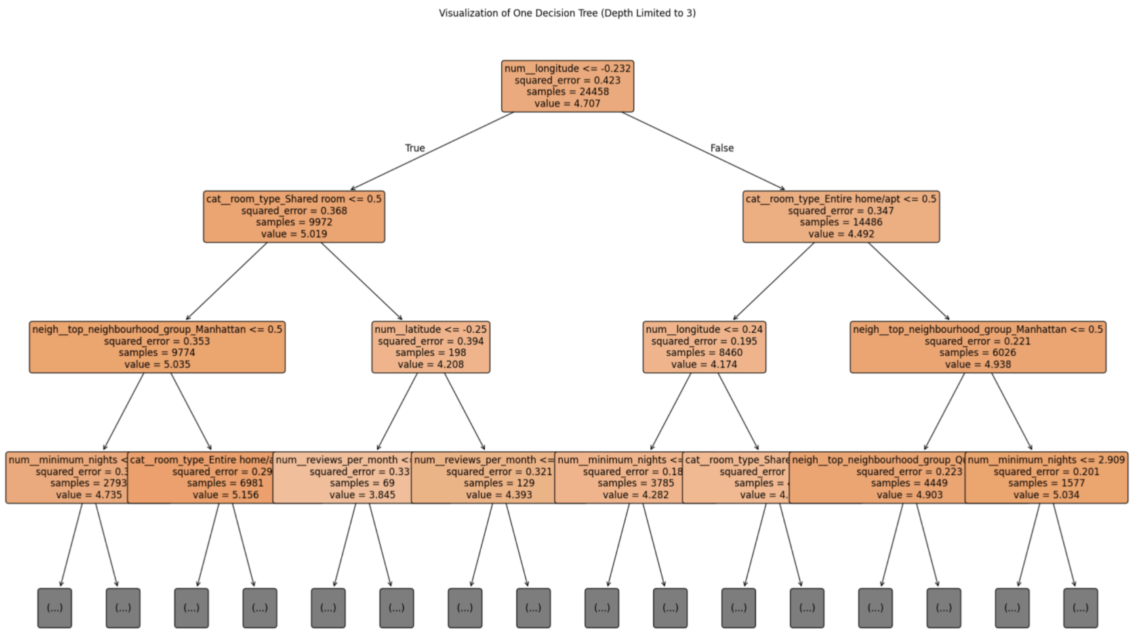
One benefit of using random forests is that they can help us distinguish between important and unimportant features. Features that reduce the “impurity” more, a measure of how uniform or spread out the labels are, are features that give more information. There are many ways to measure “impurity” and the resulting drop in “impurity” from each feature. We measured the drop in the Gini impurity, which is default in the python implementation we used to measure how important features ares

From the results below, we see that features that distinguish between entire home Airbnb listings and room Airbnb listings carry the most information about price. This is an important feature because full house airbnb hostings are likely much more expensive. Other features like room types, and location (longitude and latitude) also help predict price. This makes sense because there are more expensive and less expensive locations in New York, and generally location can sometimes have a strong causal influence on the price of a home/apartment.



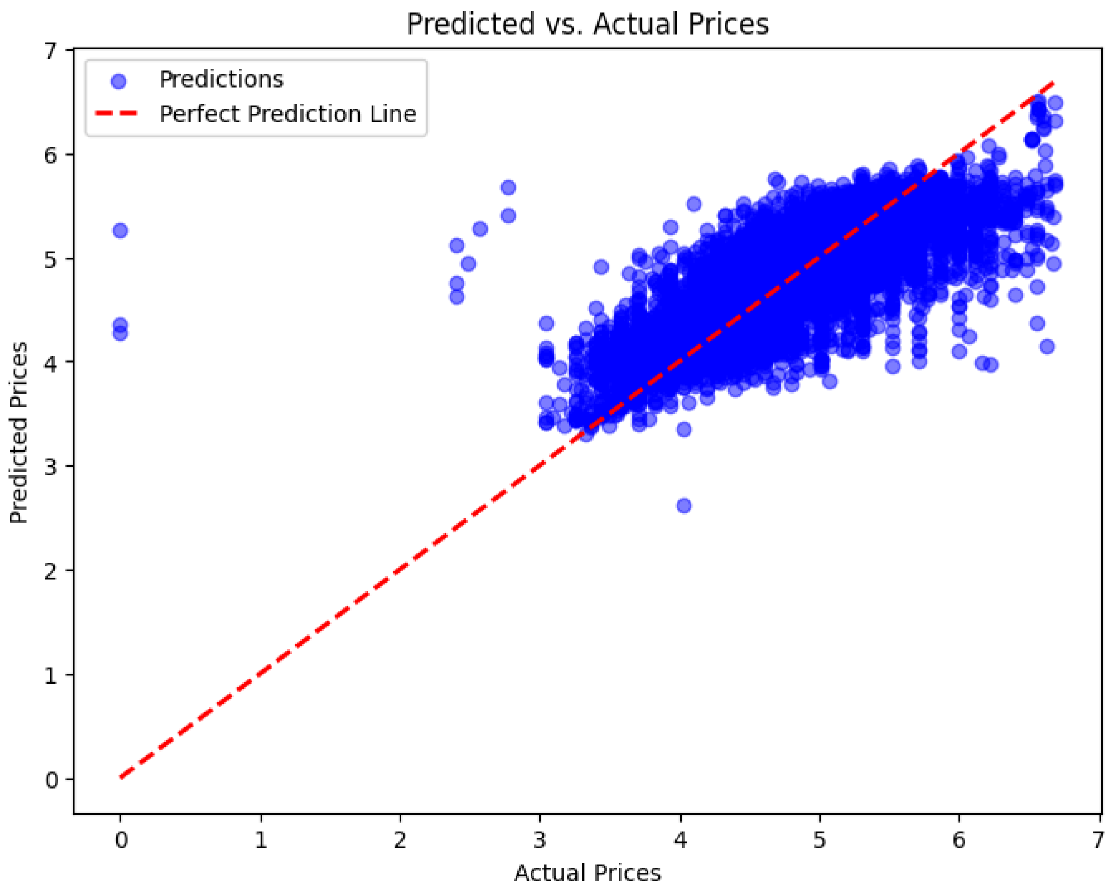
Feature Importance in Random Forest

Here is small cutout of one of the decision trees in our random forest. Since it could be very large, only the first 3 levels are shown.



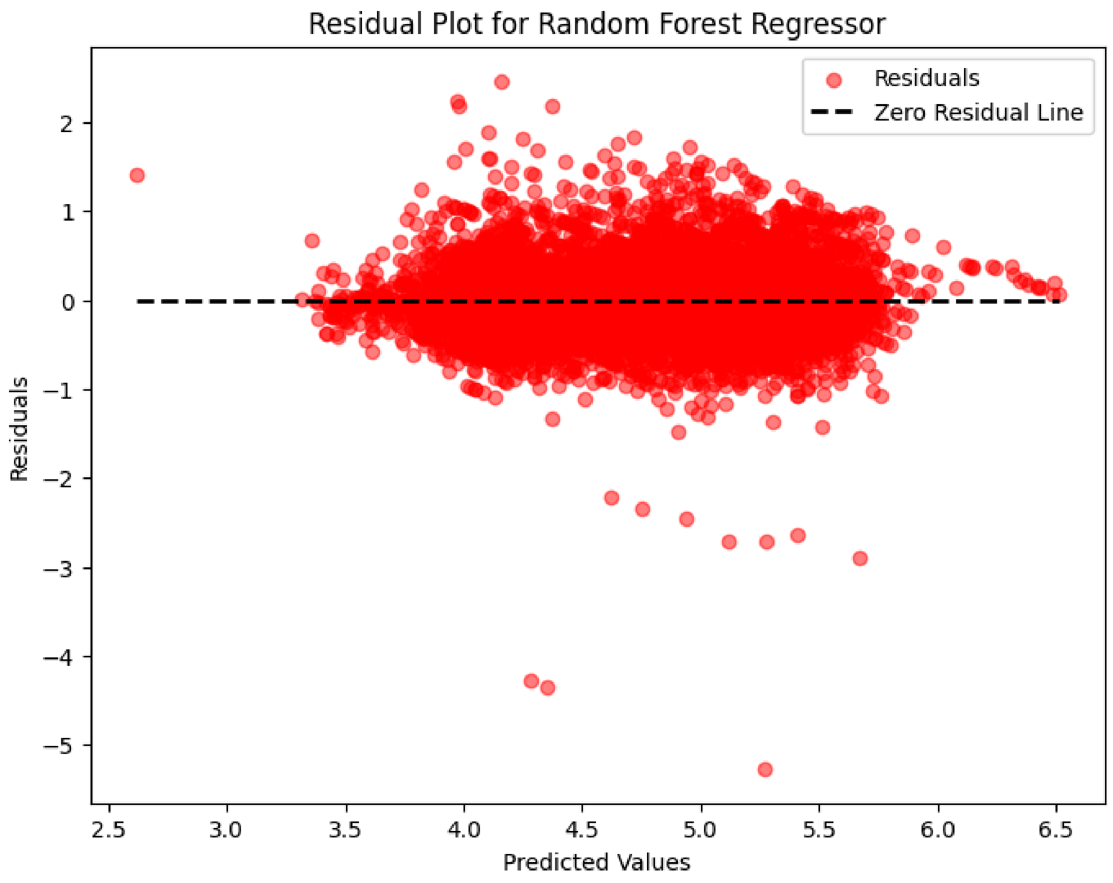
A Decision Tree in our Random Forest

There are differences between predicted and actual prices. Similarly to ridge regression and linear regression, the plot below shows that there is slightly less variance in predicted prices (in y-coord) than in the actual prices (in x-coord), meaning our random forest model generally predicts prices slightly closer to mean, the average AirBnB listing price.



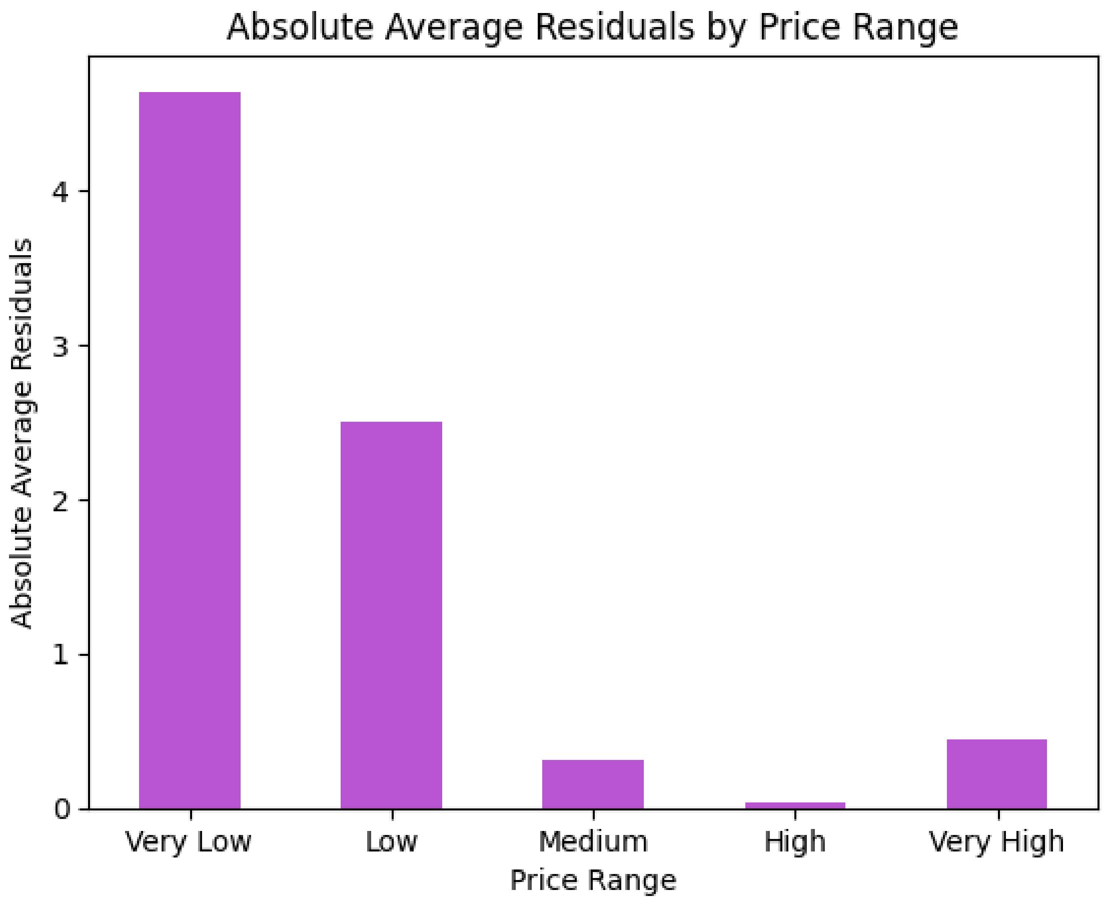
Residual Plot 1

The residual plots below show that there is still a similar amount of discrepancy between predicted prices in random forests just like linear regression and ridge regression. This discrepancy might mean that even decision trees are unable to capture the true relationship when predicting prices of AirBnb listings. Similar to how linear and ridge regression can not truly represent nonlinear relationships, random forests might struggle to represent wavy or smooth functions, because each decision tree can be thought of as splitting the data with successive flat hyperplanes. One might need to split the data with a large number of hyperplanes to represent wavy and smooth functions, which may be infeasible. The inefficiency or representing smooth wavy functions with decision tres may be on reason for the diferene between predicted and actual prices. Alternatively, the difference between predicted prices and actual prices could be the result of inherent noise in prices that no model can predict.



Residual Plot 2

A further analysis shows that our random forests model tends to make errors more often on the least expensive two quintiles of AirBnBs and the most expensive quintile of expensive AirBnB listings. This might be because our random forest model predicts AirBnB prices closer to the mean AirBnB price, and maybe requires more information that we provide to decide whether an AirBnB listing will be unusually inexpensive or expensive.



Residual Bar Graph by Price Quintiles

Random forests come with many hyperparameters that control how likely the decision trees are to overfit/underfit and how long it takes to train. The main one is the number of decision trees to use. The more decision trees used, the more robust our model will be against overfitting, but it will take longer to train.

Other hyperparameters control tree size, such as max_depth, min_samples_split, and min_samples_leaf. A node can only split if it meets three conditions: its depth is less than max_depth, it has more than min_samples_split samples, and each child node has more than min_samples_leaf samples. Smaller trees help reduce overfitting and prevent the model from being too sensitive to individual data points. Another hyperparameter, max_features, selects the best feature from a random subset of features for each split, reducing sensitivity to any particular feature. Thus, hyperparameters like max_depth, min_samples_split, and min_samples_leaf limit tree size and sensitivity to individual samples, while max_features reduces sensitivity to specific features.

We use a random search to go through the grid of possible hyperparameters, and we use 3-fold cross validation to check how good each combination of hyperparameters are. The hyperparameters chosen were 500 decision trees, a max_depth of 30, a min_sample_split of 2, a min_sample_leaf of 2, and a max_features of sqrt. This might indicate that our random forest is relatively robust and safe from overfitting, since it picks a relatively small min_sample_split and min_sample_leaf and a relatively large max_depth.

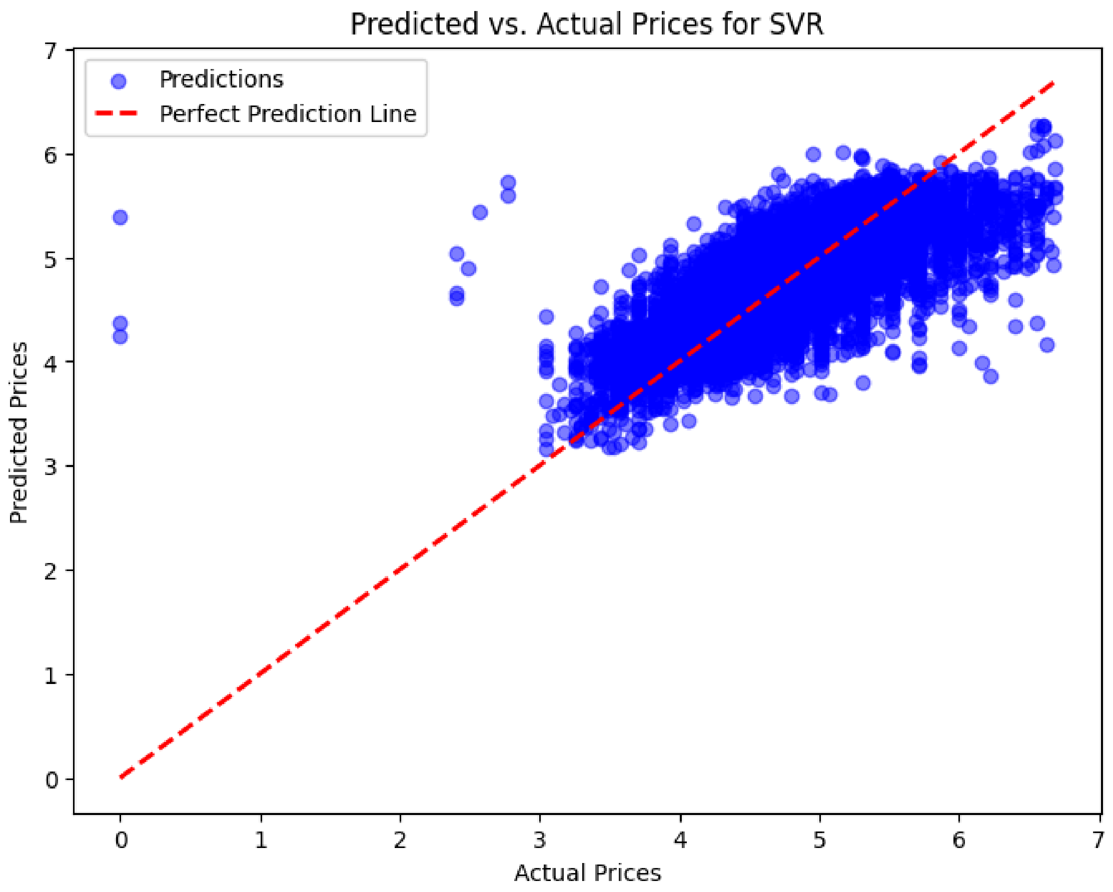
Using random forests, we get a mean squared error (MSE) of approximately 0.156, a mean absolute error (MAE) of approximately 0.294, and a root mean squared error (RMSE) of approximately 0.403. This is better than both linear regression and ridge regression. This might be because random forests can help capture some nonlinear relationships that a linear model might not be able to.

Model 4 - Support Vector Regression (SVR)

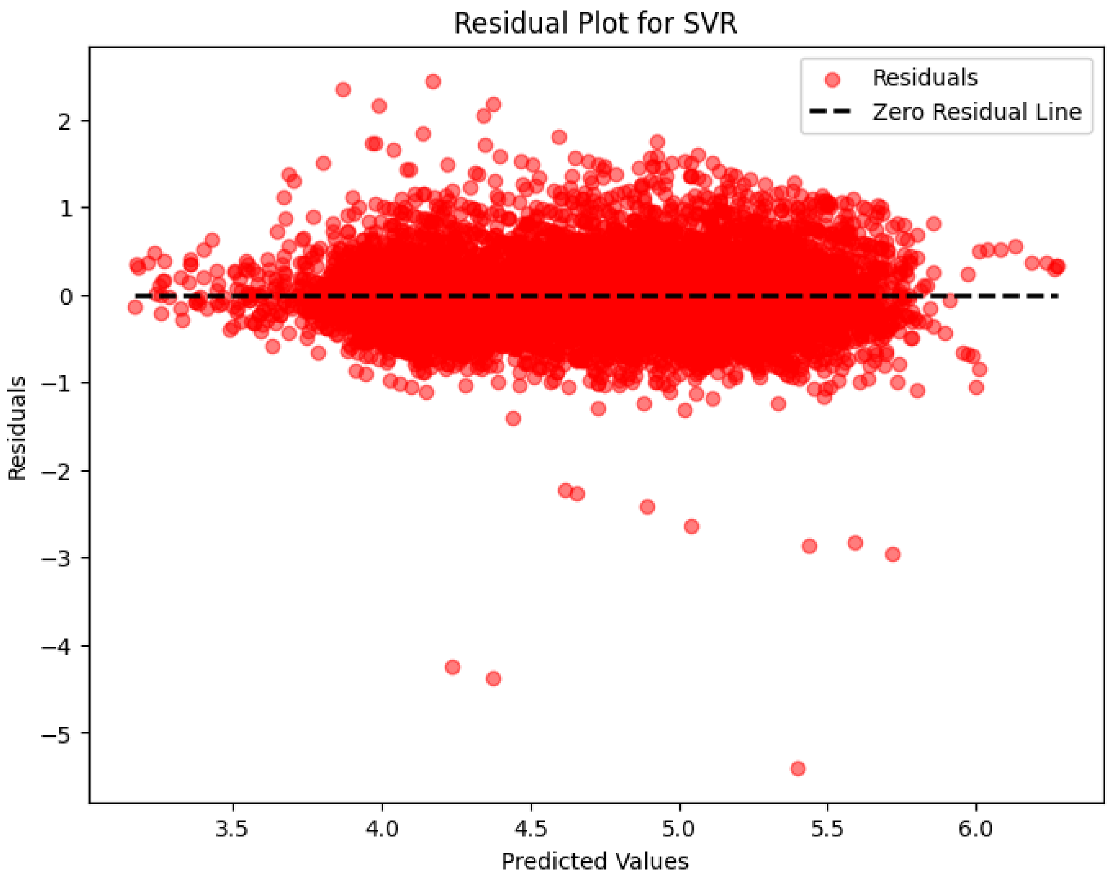
One issue with linear regression and ridge regression is that they might underfit because they struggle to represent nonlinear relationships. Similarly random forests might overfit without lots of data because they can sometimes struggle to interpolate to combinations of features not seen before. SVR is more of a balance. They are simpler than random forests because they have fewer parameters and are harder to overfit. However, unlike linear and ridge regression they can still represent nonlinear relationships.

SVR works similar to SVM, which works by finding a hyperplane that can separate linearly separable data. With SVR, we can map the input into a higher dimensional space with nonlinear kernels (by adding features like x^2 and such). By finding a linear separation in the new transformed higher dimensional space, we have found what would have been a non-linear separation in our original space. With this technique SVR can represent certain predefined nonlinear relationships, while also remaining relatively simple and resilient to overfitting.

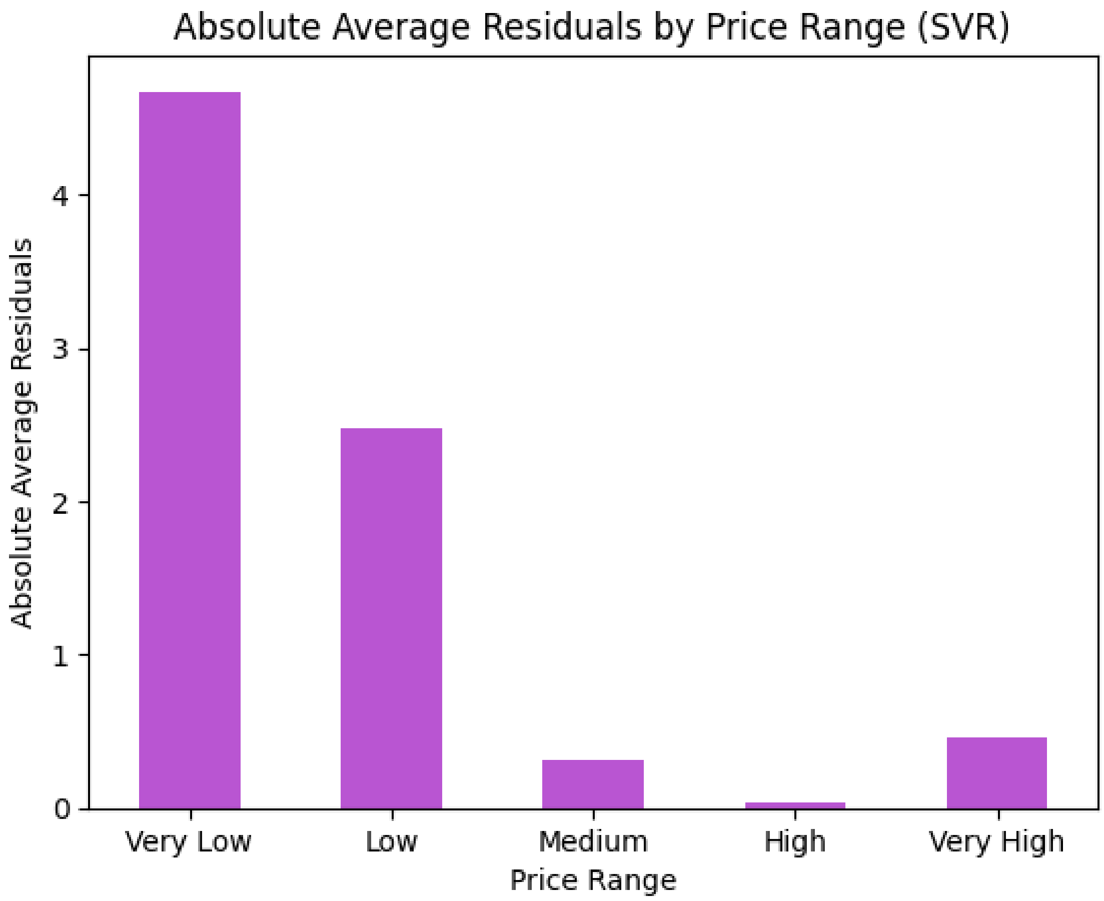
The residual plots for SVR below came out similar to those for linear regression, ridge regression, and random forest. There seems to be slightly less variance in predicted prices than actual prices because, by inspection, the y-coordinate of our data points vary slightly less than the x-coordinate of our data points. Further analysis supports this because our error seems to be larger in the bottom two quintiles and the top quintile of AirBnB prices. It seems that our model prefers to predict prices closer to the mean AirBnB price and is not confident enough to make very expensive and very inexpensive predictions. The fact that this is true for all models we have tested might imply that this is an issue with our dataset rather than our model. Perhaps there are not enough features to properly tell when an AirBnB listing is going to be very expensive or inexpensive. Or, very expensive and inexpensive AirBnB listings could be the result of random noise that no model could predict well.



Residual Graph 1



Residual Graph 2



Residual Bar Graph by Price Quintiles

SVR comes with a few hyperparameters to tune in order to get a good model. The main hyperparameters are the kernel, epsilon, and C. The kernel we choose dictates how we transform our inputs into a higher dimensional space. This is an important consideration because, while SVR has the ability to model nonlinear relationships, we must choose what kind of nonlinear relationships preemptively by choosing the Kernel. Popular choices are the polynomial kernel, radial basis kernel, and linear kernel. The epsilon we choose dictates how large of a region surrounding our predicted function points can lie in without penalty. This helps control our sensitivity to the random noise of points nearby the predicted function. The hyperparameter C in the objective function controls the trade-off between penalizing overly complex models (with large weights) and penalizing models that do not fit the data well. In other words, epsilon controls the model's sensitivity to points near the predicted function (i.e., how much deviation is tolerated), while C controls the overall sensitivity of the model to all points, balancing the need for a good fit with the desire to keep the model simple.

We perform a random search through a grid of possible hyperparameters to choose the best ones. We use 3-fold cross validation to find the mean squared error, mean absolute error, and root mean squared error to test how good our choices of hyperparameters are. The best ones appear to be radial basis kernels, with an epsilon of 0.35 and a C of 10. The choice of kernel basis might imply that the nonlinearity in our data occurs from the distance between data points from each other rather than an overarching polynomial trend.

We get an MSE of 0.165, worse than the MSE of 0.156 of random forests, but better than the MSE of 0.197 of linear regression and MSE of 0.201 of ridge regression. We get an MAE of 0.300 and an RMSE of 0.406 which is worse than random forests but better than linear regression and ridge regression. The fact that our metrics are much better for random forests and svr rather than linear regression and ridge regression means that there are likely nonlinearities in our data that cannot be captured by linear models. But, surprisingly, these nonlinearities are better captured by random forests rather than SVR. Perhaps our dataset is large enough, with roughly ~40,000 entries, that there is no issue of overfitting for our random forests model.

Comparison Between Models

Overall

Random forests did the best of all our models tested. It had the lowest mean squared error, mean absolute error, and root mean squared error. Random forests were able to model the nonlinear relationships that linear models (like linear regression and ridge regression) could not. Random forests can sometimes struggle with overfitting data. However, in this case that did not happen, which we can tell because our random search through hyperparameters found that hyperparameters favored larger trees that examined lots of features, which are the most prone to overfitting. This is likely because the data had some small nonlinearities, but nothing overly wavy and smooth. There were over 40 thousand data points, but only 17 features, which helps prevent overfitting.

Between Random Forest and Linear Models

Random forests can capture nonlinearities in the data that linear models can not. It takes longer to train random forests and they have more hyperparameters to tune. Random forests may not be as easily interpretable, one of the big advantages to linear models, but they performed much better in predicting AirBnB prices.

Between SVR and Linear Models

SVR also performed much better than linear models. SVR can represent some nonlinear relationships with nonlinear kernels, which is likely why it performed better than linear models. Just like Random Forest, SVR requires hyperparameter tuning, and they are harder to interpret than linear models.

Between Random Forest and SVR

Both SVR and random forests can model nonlinear relationships. Random forests can model nonlinear relationships, but it does so in a piecewise way. Geometrically, it looks like the composition of many splits in the data via a hyperplane. So, very smooth and wavy functions might require lots of straight hyperplane splits to model. SVR can also model nonlinear relationships, but one has to specify the kind of nonlinear relationship beforehand via the kernel. For this problem, random forests performed better. Most likely, the relationship in the data was too complicated to easily be represented by the nonlinear kernel, but also not overly wavy or smooth. Also, lots of our features were either categorical (transformed using one hot encoding)l, or numerical like longitude and latitude. There is no easy physical interpretation to adding and multiplying these numbers, so that may be why our function was not as nicely represented by an analytic function in SVR.

Next Steps

We can try doing a more exhaustive hyperparameter search for random forest, or search for yet another better model besides these four. For all our models, it appeared that our predictions were worse for the most inexpensive and expensive AirBnB listing. This suggests a limitation of our data. Perhaps the best idea moving forward is to get data with more extensive features. More information might be required to predict the most inexpensive and expensive AirBnB listing. Additionally, more than just predicting the best possible price. Software that automatically prices AirBnB might have societal implications. Making AirBnB's too expensive or too inexpensive might have more personall and economic effects on people that could also be studied for a more holistic strategy.

References

Gibbs, C., Guttentag, D., Gretzel, U., Morton, J., & Goodwill, A. (2018). Pricing in the sharing economy: a hedonic pricing model applied to Airbnb listings. Journal of Travel & Tourism Marketing, 35(1), 46–56. https://doi.org/10.1080/10548408.2017.1308292

Fan, S. (2022). Costly Price Adjustment and Automated Pricing: The Case of Airbnb. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4077985

Lin, W., & Yang, F. (2024). The price of short-term housing: A study of Airbnb on 26 regions in the United States. Journal of Housing Economics, 65, N.PAG. https://doi.org/10.1016/j.jhe.2024.102005

Ye, S., & Rajaram, S. (2018). Customized Regression Model for Airbnb Dynamic Pricing. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 932–940. https://doi.org/10.1145/3219819.3219830

Gantt Chart

View Gantt Chart

Contribution Chart

Name	Contributions
Sahithya	Worked on the Data preprocessing section and Linear Regression.
Sahit	Worked on Methods Section, focused on Random Forest and SVR.
Aakrishtaa	Focused on interpreting results/metrics and video
Nandita	Focused on interpreting visualizatons/metrics and video
Anshul	Worked on Methods Section, focused on implementing and running Random Forest a

Video Presentation

View Video