Final Report

Introduction/Background:

Since 2008, has grown drastically throughout the globe. As an online easily accessible peer-to-peer marketplace, Airbnb has facilitated 6 million+ accommodation listings and bookings in 100,000+ cities [1]. If an ownwer wished to list their property, they must be shrewd about setting a fair listing price as this directly affects consumer demand. On the customer side, a prudent decision must be made of what the best value for a property should be, with minimal knowledge of the accommodations, and whether or not they want to rent out the listing [3].

By creating a price prediction model for Airbnb listings using machine learning, we will be able to perform exploratory data analysis to help understand what features are most influential to price calculations. This will help both buyers and sellers make the best informed decision for their respective needs.

We plan to use Inside Airbnb, which includes several open-source datasets for listings in 20+ US and international cities. In particular we will work with the New York datasets. The listing datasets are extremely rich, including 20,000+ data points and up to 70 features such as host response rate, neighborhood, property type, and amenities. Listing datasets will be joined together along with neighborhood datasets found on the same site, and all of the data in aggregate will be preprocessed for our work.

Problem Definition:

Why do we want to create this prediction algorithm? The biggest reason we chose to work with Airbnb data is simply because of the benefits that it can provide. For consumers, it will help them determine whether they are getting good value for their purchase. This in turn will lead them to want to use Airbnb more and suggest it to others. For the owners, the tool will help them find a good compettive price so that they both get good profits but also not overshadowed by other listings.

How do we go about creating such a model? First, the most important part is to take our dataset of New York and identify the features/columns which seem to be most prevalent in affecting prices. Initial ideas were that size, location, and amentities would be relevant features to look at. Since there are 75 features ot begin with, many steps of pre-processing were done to ensure consistency between rows as well. Additionally, we reduced the number of features to reduce the likelihood of overfitting with our model.

From here we will be applying various ML models such as linear regression, k-means clustering, random forest, and neural net classifications.

Methods:

Data Preprocessing:

Pre-Processing:

First, we looked at all the features and their values to get a good, general idea of what was being offered and ended up dropping the following features due to either one of two reasons: all of the values of the feature were NaN, all of the values had no particular importance and would be infeasible to analyze for the purposes of this project (e.g. host_name, picture_url)

> ['listing_url', 'scrape_id', 'last_scraped', 'calendar_last_scraped', 'source', 'name', 'description', 'neighborhood_overview', 'picture_url', 'host_id', 'host_url', 'host_name', 'host_location', 'host_about', 'host_thumbnail_url', 'host_picture_url', 'host_neighbourhood', 'host_verifications', 'neighbourhood', 'neighbourhood_group_cleansed', 'property_type', 'bathrooms', 'bedrooms', 'amenities', 'calendar_updated', 'license']

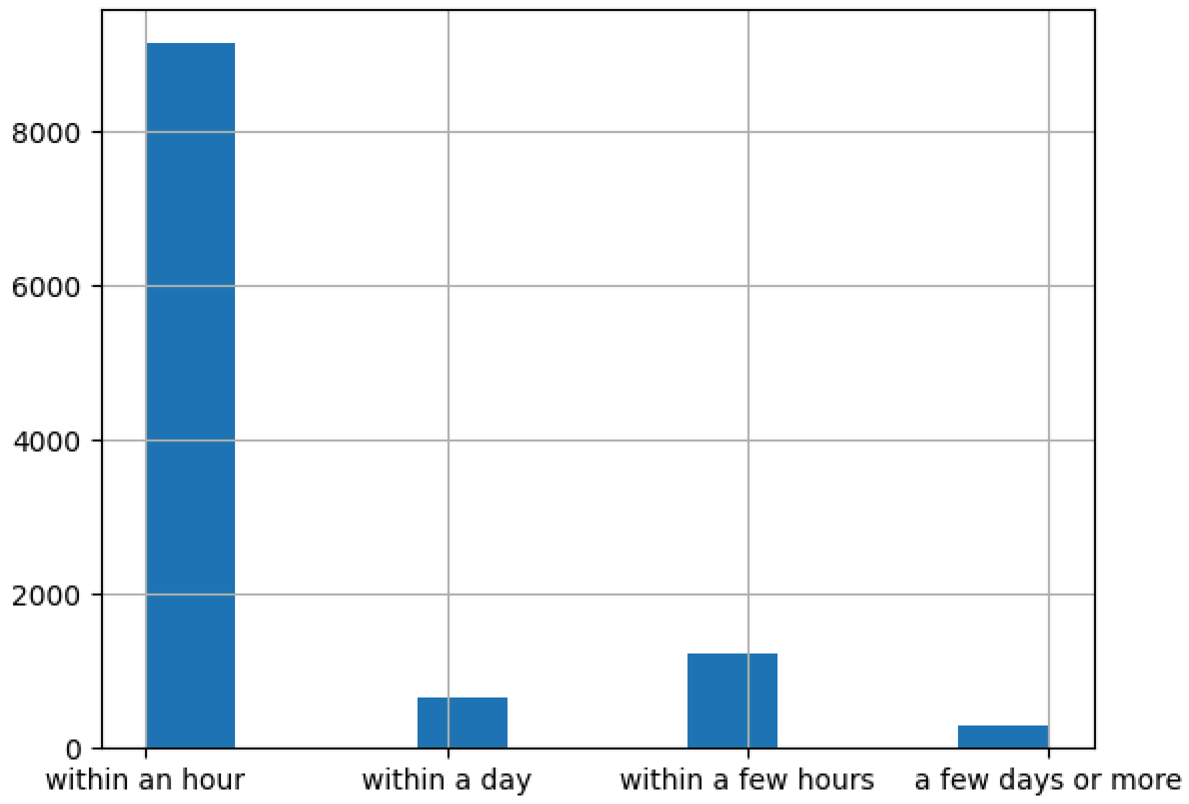
Next, we need to be able to convert dates into something numerical that could possibly be used in our future models. A date like '2023-12-16' can be converted into a numeral like '1702684800000000000'. We did this for three features: 'first_review', 'last_review', and 'host_since' using the following code:

```
df.host_since = pd.to_datetime(df.host_since)
df.host_since = pd.to_numeric(df.host_since)
```

Next, we also needed to handle percentages. This was a bit more intuitive and was done for 'host_response_rate', 'host_acceptance_rate', and 'price' which is our target value. For instance:

```
df['price'] = df['price'].fillna('0').str.replace('$', '').str.replace(',', '').astype(fl
```

Finally, as you can see from the following image, a few of our features are categorical data. Therefore, we have also utilized label encoding.

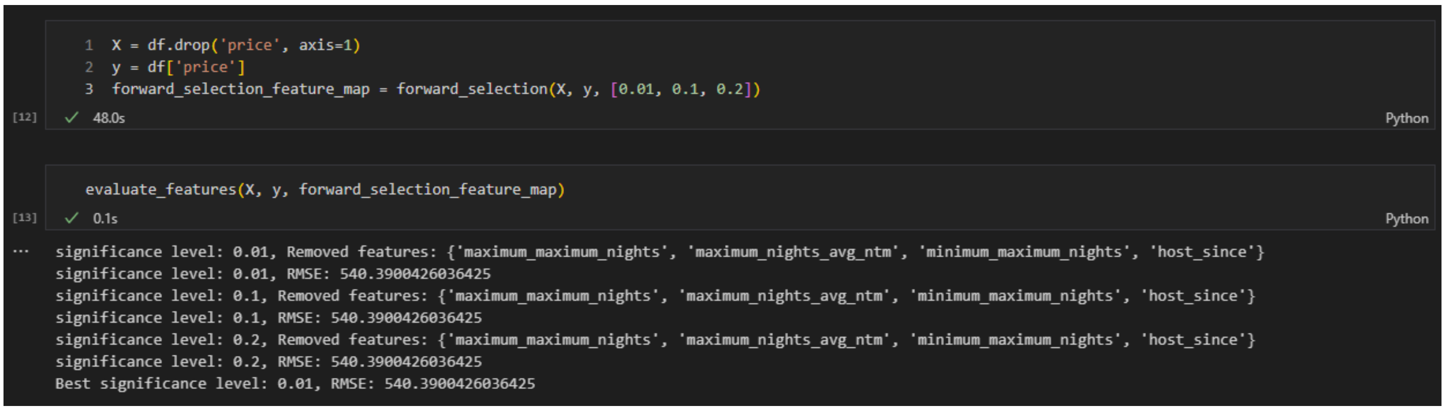


Histogram of feature: 'host_response_time'

> Label encoding was applied on the following 8 features: ['host_is_superhost', 'host_has_profile_pic', 'host_identity_verified', 'has_availability', 'instant_bookable', 'bathrooms_text', 'host_response_time', 'room_type']

Feature Selection:

Now, even with getting rid of all those features, we still want to actually reduce our features to make the lives of our ML algorithm counterparts easier and ultimately reduce the chance of overfitting or any other issues. To do this, we can use feature selection methods. We first started with using code for `forward selection` from HW3 Q6 to attempt to remove features, which didn't actually work that well.



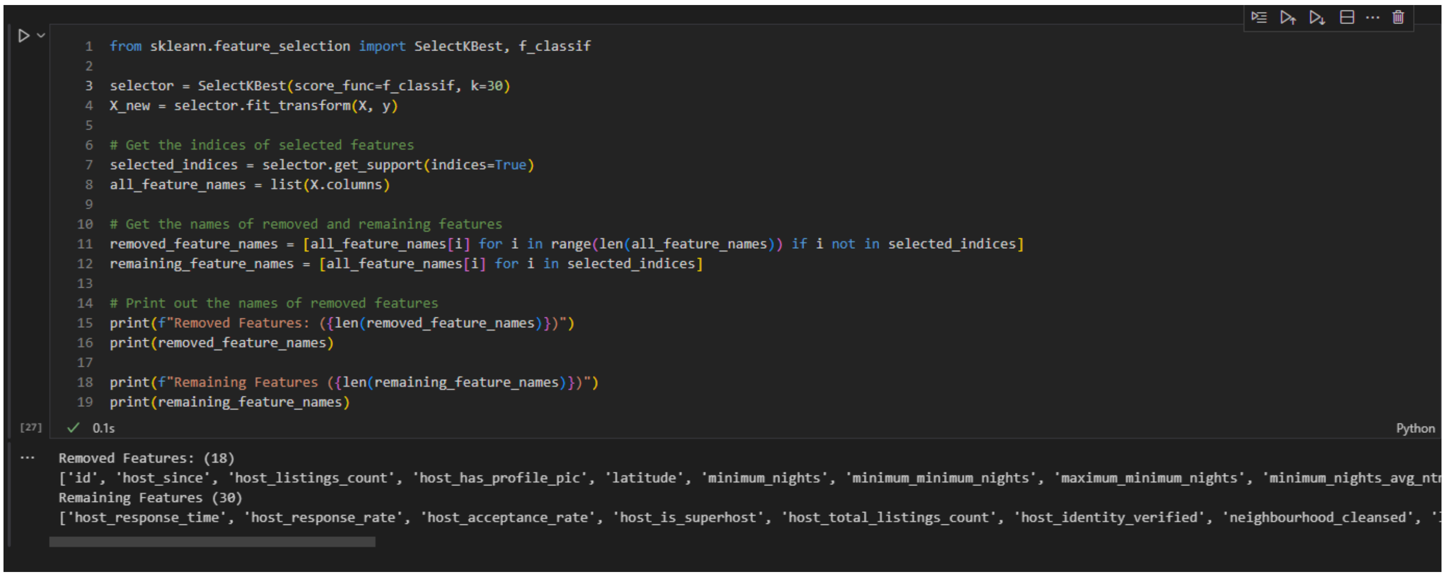
```
X = df.drop('price', axis=1)
y = df['price']
forward_selection_feature_map = forward_selection(X, y, [0.01, 0.1, 0.2])
```
[12] ✓ 48.0s Python

```
evaluate_features(X, y, forward_selection_feature_map)
```
[13] ✓ 0.1s Python

```
significance level: 0.01, Removed features: {'maximum_maximum_nights', 'maximum_nights_avg_ntm', 'minimum_maximum_nights', 'host_since'}
significance level: 0.01, RMSE: 540.3900426036425
significance level: 0.1, Removed features: {'maximum_maximum_nights', 'maximum_nights_avg_ntm', 'minimum_maximum_nights', 'host_since'}
significance level: 0.1, RMSE: 540.3900426036425
significance level: 0.2, Removed features: {'maximum_maximum_nights', 'maximum_nights_avg_ntm', 'minimum_maximum_nights', 'host_since'}
significance level: 0.2, RMSE: 540.3900426036425
Best significance level: 0.01, RMSE: 540.3900426036425
```

As seen from the image above, after a very long run-time, we were only able to remove 4 features and have a terrible RMSE - 540! This likely indicates the fact that the naive implementation couldn't actually handle the large amount of features and complexity of our own dataset as opposed to the breast cancer dataset.

So, we decided not to use an official feature selection package fromsklearn. We decided to go with the `SelectKBest` module, which is extremely easy to use and allows us to choose how many features we want out. Here are the results when we decided to select K=30 best features.



```python
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(score_func=f_classif, k=30)
X_new = selector.fit_transform(X, y)

# Get the indices of selected features
selected_indices = selector.get_support(indices=True)
all_feature_names = list(X.columns)

# Get the names of removed and remaining features
removed_feature_names = [all_feature_names[i] for i in range(len(all_feature_names)) if i not in selected_indices]
remaining_feature_names = [all_feature_names[i] for i in selected_indices]

# Print out the names of removed features
print(f"Removed Features: ({len(removed_feature_names)})")
print(removed_feature_names)

print(f"Remaining Features ({len(remaining_feature_names)})")
print(remaining_feature_names)
```

[27] ✓ 0.1s Python

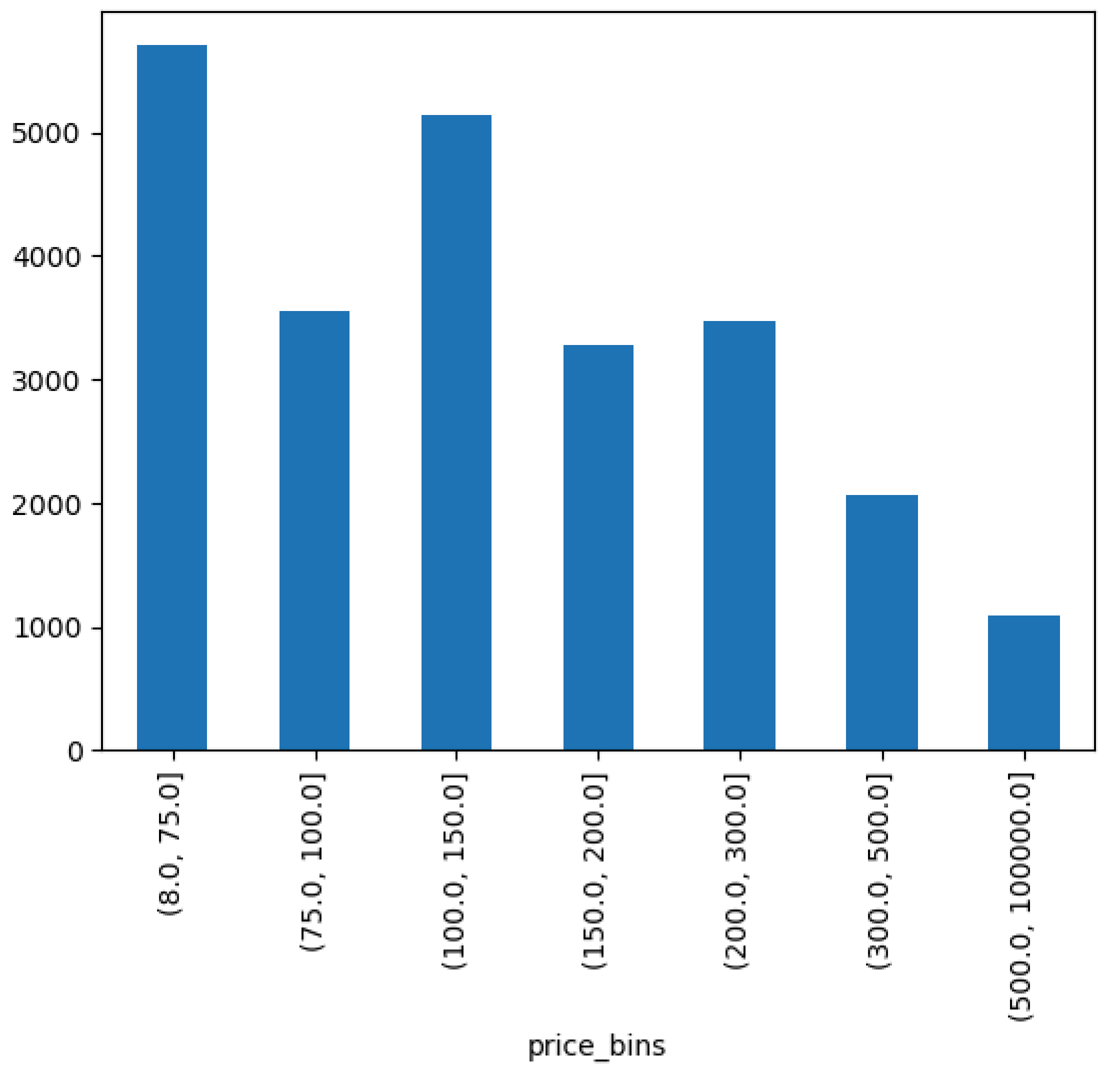
```
Removed Features: (18)
['id', 'host_since', 'host_listings_count', 'host_has_profile_pic', 'latitude', 'minimum_nights', 'minimum_minimum_nights', 'maximum_minimum_nights', 'minimum_nights_avg_nt
Remaining Features (30)
['host_response_time', 'host_response_rate', 'host_acceptance_rate', 'host_is_superhost', 'host_total_listings_count', 'host_identity_verified', 'neighbourhood_cleansed', '
```

Not including our target variable, we removed 18 features, including the features that were initially removed from our naive implementation.

Dataset Balancing:

Finally, we need to check if our dataset is actually balanced or not respect to our *target variable*: price. The intuition behind this is that if our dataset has a significant class imbalance where most AirBnB listing are within a certain price range, there might be bias towards the majority, which ultimately can negatively affect performance. If our dataset is imbalanced, we need to apply SMOTE, which handles the imbalance. Note, in this situation, we do not remove outliers.

Let's first generate a simple **histogram** to look at the distribution of prices. Note, this isn't as easy as pie - because price can differ by the dollar! Therefore, we need to start off by making some categories. Since we don't really know how to split effectively, we can use *pandas.cut* to creates bins of an arbitrary size. Again, we are measuring class *imbalance*, so it does really depend on what we are trying to measure. We don't want to oversimplify. For instance, our min price is $1 while our max price is $38,143 - we don't just want to create two bins (e.g. 1 - 10,000 to 10,000 to 38,143) because that doesn't translate to what we attempt to specifically measure. Thus, we need to make our bin edges significantly more tight. We came up with the following:

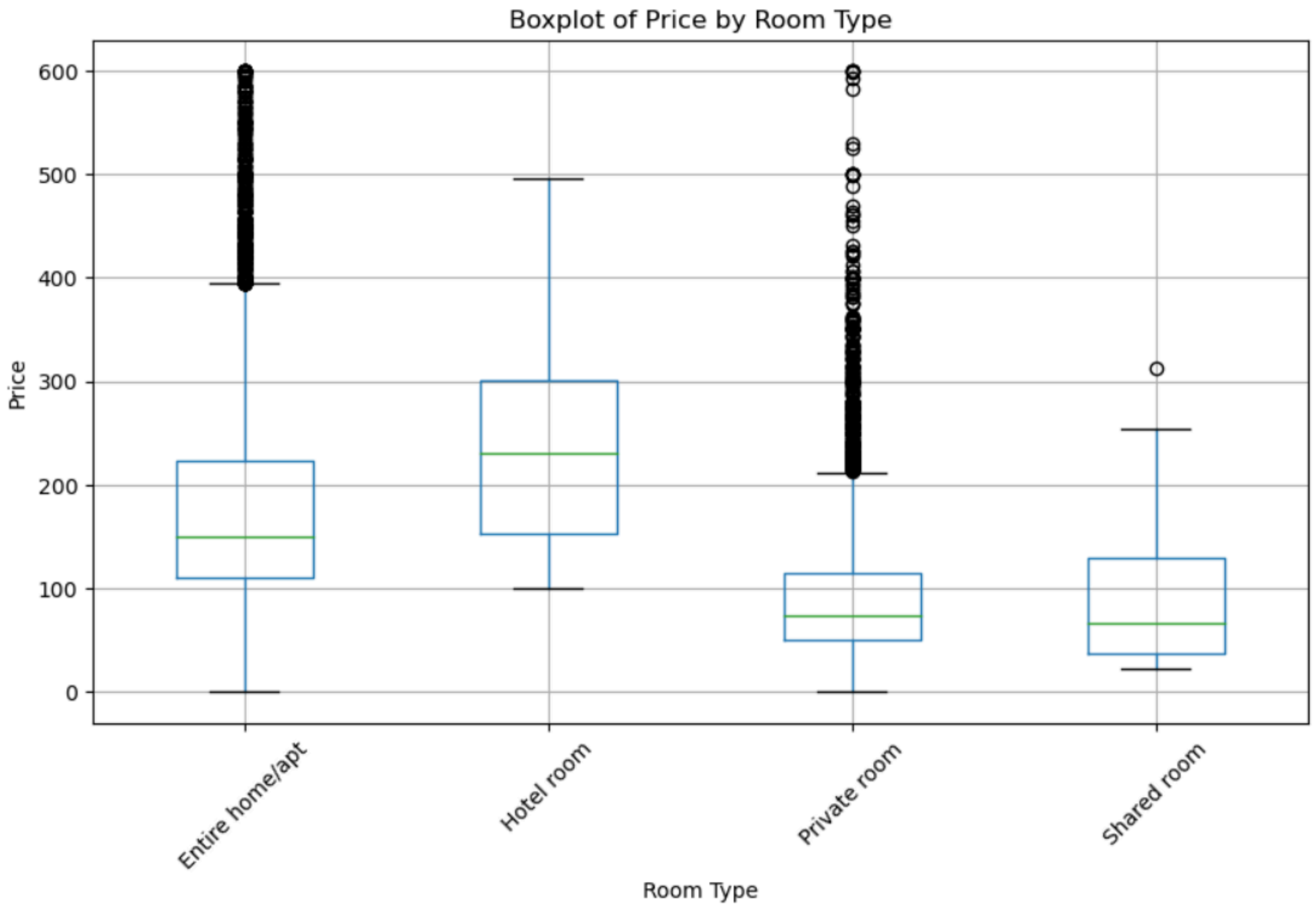


From the plot, we can see that there really isn't a *major* imbalance that requires intervention. Bin (500.0, 100,000] seems a bit severely populated but for the most part, we don't care *as much* about the prices in this range. But we can always try SMOTE and revert it later on if we find something isn't adding up.

Data Visualization

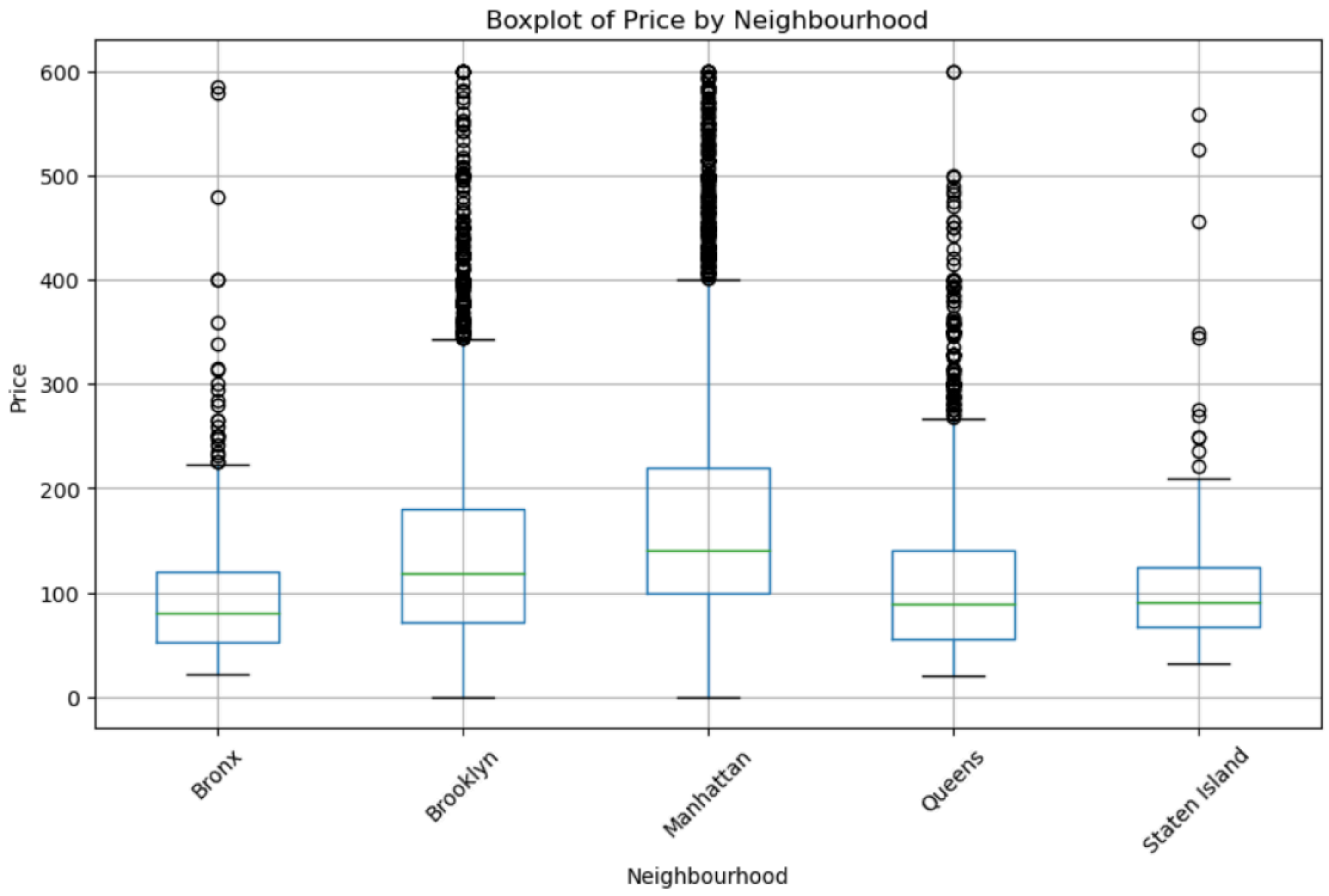
With our preprocessed data, we want to create some visualizations to showcase some of the features we believe will be the most important to our model. Those features are price by number of beds, bathrooms, bedrooms, and accommodates individually. Additioanlly, we wanted to see the distribution amongst our entries for where they are located. In other words, this will help us see which borough of NYC do we have more entries for.

The first diagram we are going to look at is what range our prices are shown to be.



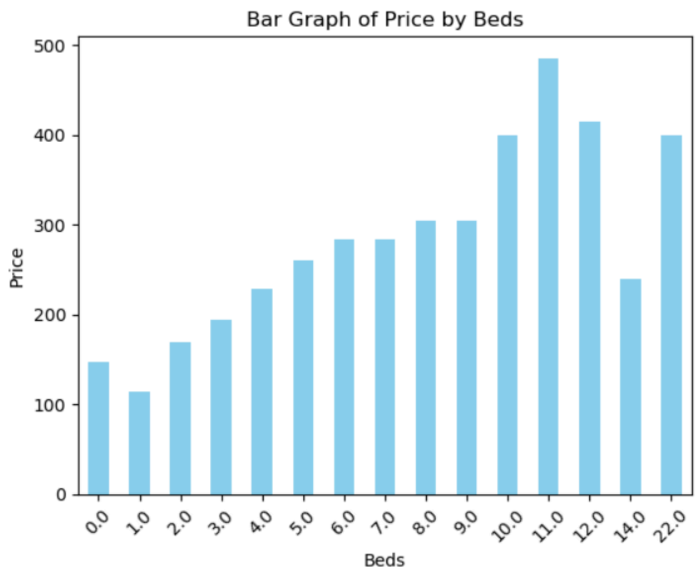
Boxplot of Price vs. Room Type

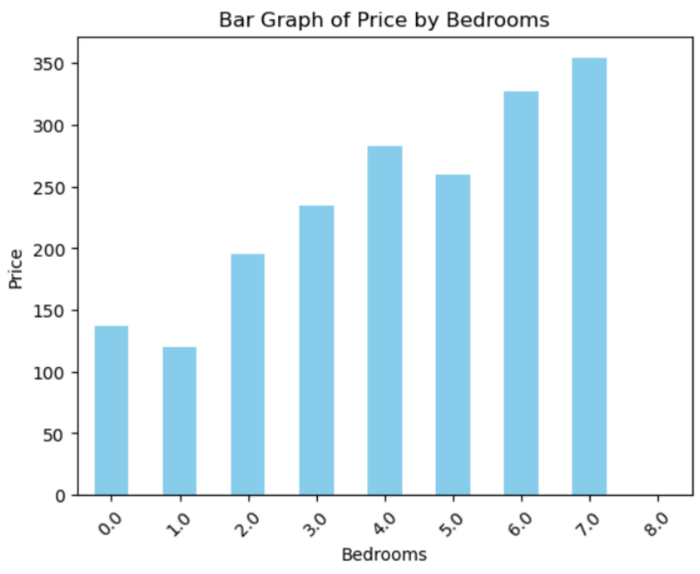
Next, we are going to take a look at the price range by the different boroughs.

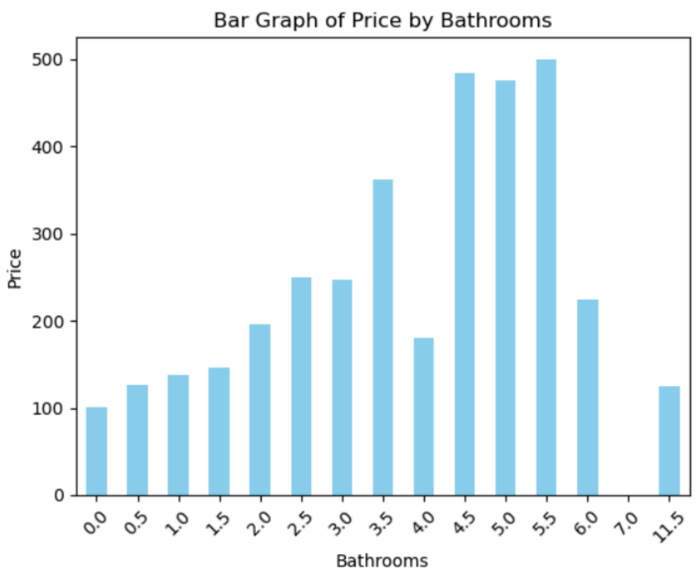


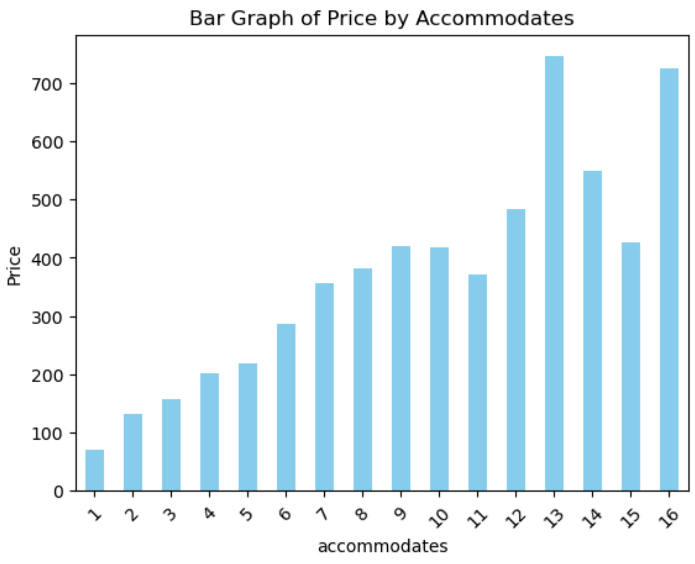
Boxplot of Price vs. Neighborhoods (Boroughs)

These following 4 bar graphs show the relationship between number of beds, bedrooms, bathrooms, and accommodates. They all share a general trend, where as the number of features increase, the average price increases.









Note: You might notice that some areas of data preprocessing, specifically concepts like data balancing and outlier removal were ommitted. We decided to separate this from the rest of the preprocessing since they could have dynamic changes to the final result. We will let you know if a result used other preprocessing techniques.

ML Algorithms:

Our modeling draws from methods used in related works for Airbnb price prediction, such as logistic regression, support-vector regression (SVR), K-means Clustering (KMC), and neural network models (Dhillon 2021). Other validated methods also include linear regression, forest classification, XGBoost, and LightGBM to produce an optimistic price prediction model on housing (Truong 2019). For this final report, we are selected **linear regression** as well as **random forest** and **neural net** classification analysis.

Linear Regression

For the [**1**] *first round* of price prediction, we used three linear modeling approaches to establish a basic correlation between features and Airbnb prices. We decided to first use linear regression techniques for their simplicity. The lower complexity of these models ensure efficiency, and they have high interpretability. These linear regression models also provide a good baseline for further evaluation.

The linear approaches we used were: linear regression, ridge regression, and lasso regression. Cross validation techniques were used with ridge and lasso regression to determine the best alpha value.

Random Forest

For the [**2**], we explored a random forest model.

We decided use random forest model due to it being able to find non-linear relationships. Random forest can also be easily used to show the feature importance. This can be very helpful to see which features have the biggest impact on the data.

The Random Forest approaches we used were: Extremely Randomized Trees Regressor, Random Forest Regression, and Random Forest Classification.

Neural Network

And finally [**3**], we implemented a neural network.

In the case of the Airbnb data, using a neural network can be beneficial if there are non-linear relationships between the features (e.g., host characteristics, neighborhood information, amenities) and the target variable (price). Neural networks can potentially capture these complex patterns and provide better predictive performance compared to linear models.

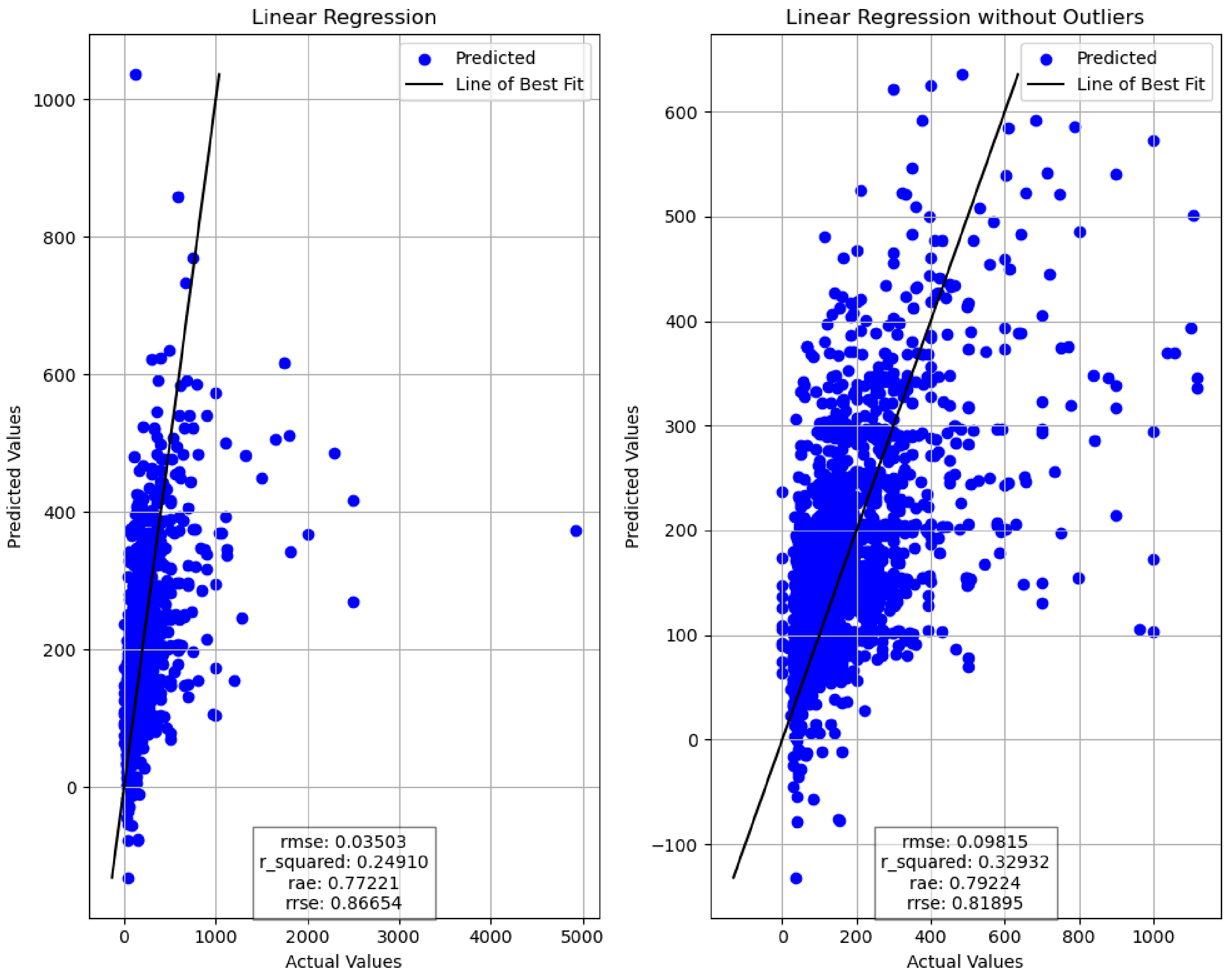
The neural network approaches used is Multi-Layer Perceptron (MLP) Regressor and for ensemble boosting Gradient Boosting Regressor is used.

Results and Discussion:

Linear Regression

Each graphic below displays a scatterplot of the actual vs. predicted prices on the left, and an adjusted scatterplot on the right with outliers (defined as points > 3 std away from mean) removed. Through each set of data is a trendline to show the best fit line for points.

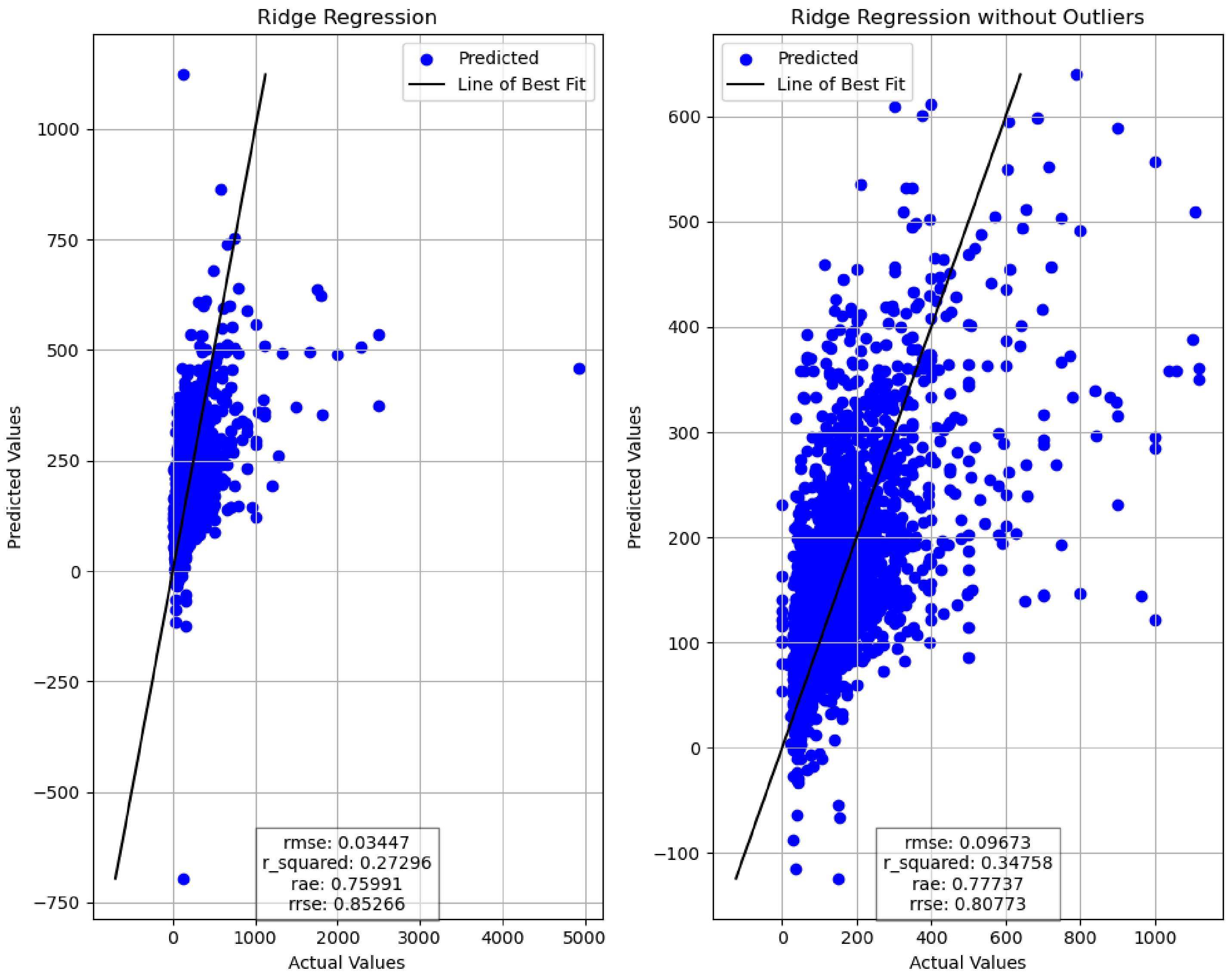
We used Recursive Feature Elimination (RFE) for feature reduction, then ran and evaluated model metrics. Our primary correlation and fit metric is R-squared, and for error we have MSE, RAE, and RRSE.



Linear Regression Model Actual vs Predicted Prices

The left plot shows an RMSE value of 0.03503, an R-squared value of 0.24910, RAE of 0.77221, and RRSE of 0.86654. This correlation is pretty low, but noticing the outliers in the graph, we decided to adjust the data once more and reevaluate.

The right plot shows an RMSE value of 0.09815, an R-squared value of 0.32932, RAE of 0.79224, and RRSE of 0.81895. Removing outliers improved the correlation by 0.08022. The RMSE and RAE increased slightly, while RRSE decreased.



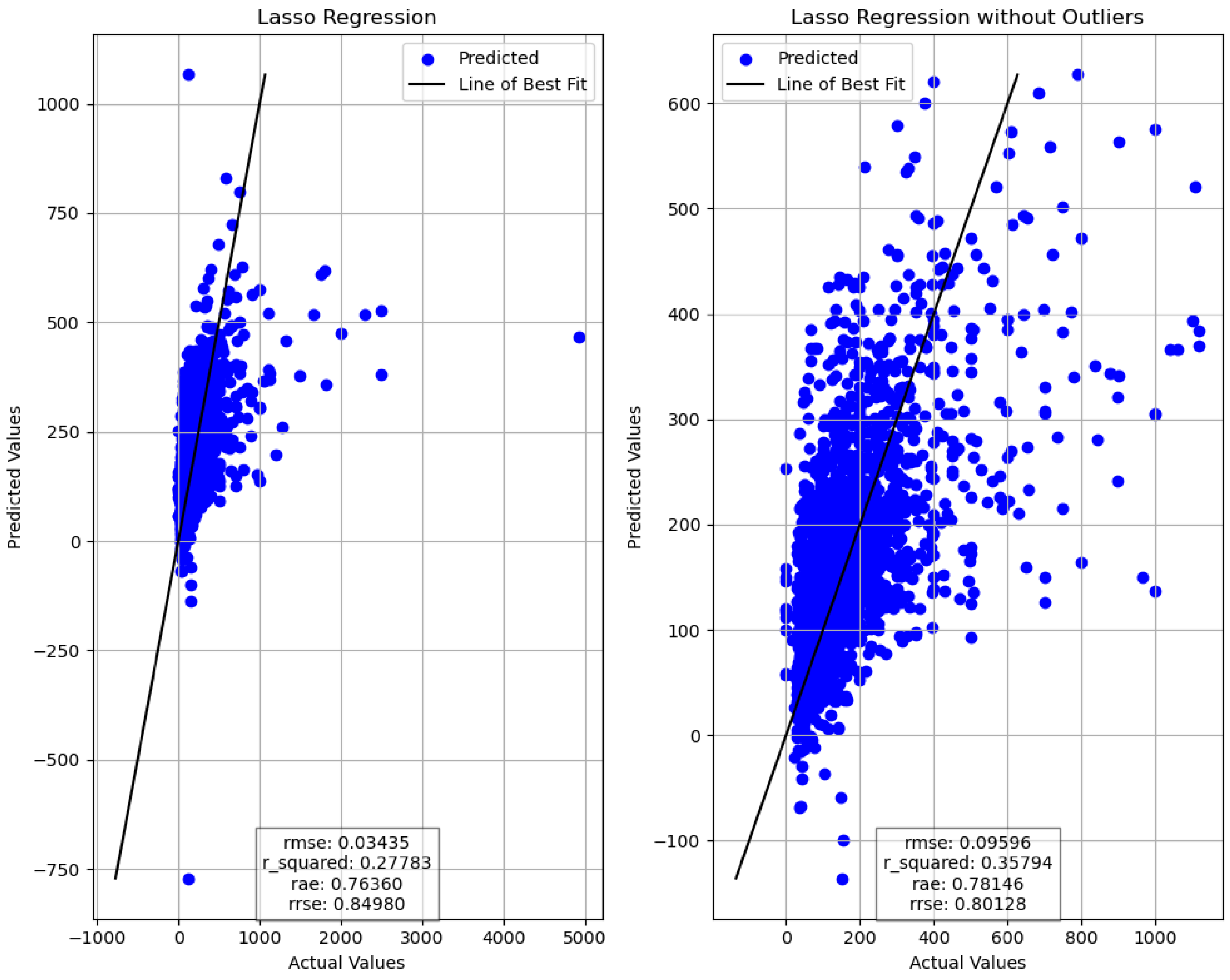
rmse: 0.03447
r_squared: 0.27296
rae: 0.75991
rrse: 0.85266

rmse: 0.09673
r_squared: 0.34758
rae: 0.77737
rrse: 0.80773

Ridge Regression Model Actual vs Predicted Prices

The left plot shows an RMSE value of 0.03447, an R-squared value of 0.27296, RAE of 0.75991, and RRSE of 0.85266. Similar to linear regression, the left plot shows a relatively low correlation between the actual and predicted prices. So once again, data was adjusted to remove outliers.

The right plot shows increased correlation with an R-squared value of 0.34758, but also increased prediction errors with an RMSE value of 0.09673, RAE of 0.77737. However, the RRSE value decreased to 0.80773, indicating an improved prediction error.



Lasso Regression Model Actual vs Predicted Prices

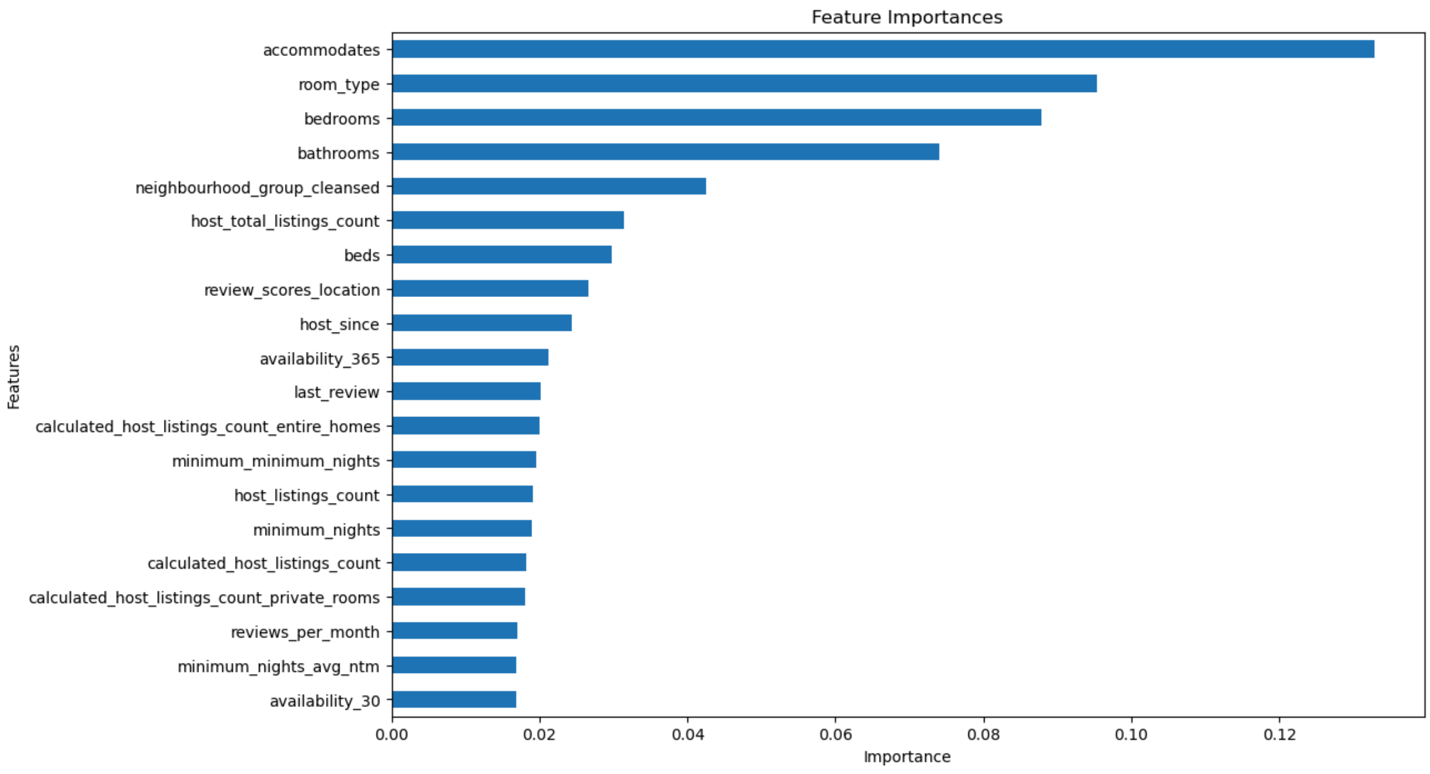
The left plot shows an RMSE value of 0.03435, an R-squared value of 0.27783, RAE of 0.76360, and RRSE of 0.84980. Again the left plot shows a relatively low correlation similar to linear and ridge regression models, indicated by the R-squared value.

After removing outliers, the correlation improved, with the R-squared value increasing to 0.35794, the highest yet. In a pattern similar to ridge regression, the RMSE and RAE values increased slightly to 0.09596 and 0.78146 respectively, while the RRSE value decreased to 0.80128 after adjusting the data.

Random Forest

Extremely Randomized Trees Regressor

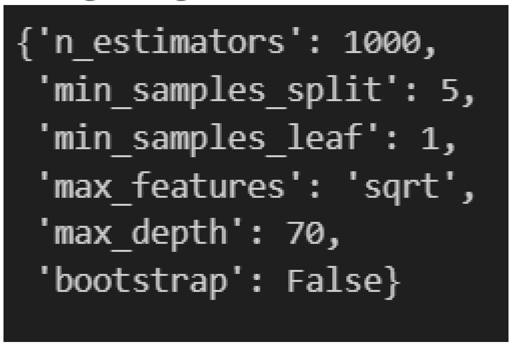
We first decided to use Extremely Randomized Trees Regressor to explore the feature importance. We plotted the results below.



Feature Importance

Random Forest Regression

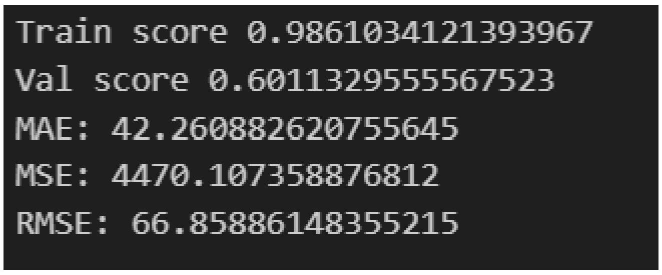
We then moved to Random Forest Regression. To improve our model we used hyperparameter tuning and got these values as the best parameters for our model:



```
{'n_estimators': 1000,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 70,
 'bootstrap': False}
```

Best Hyperparameters

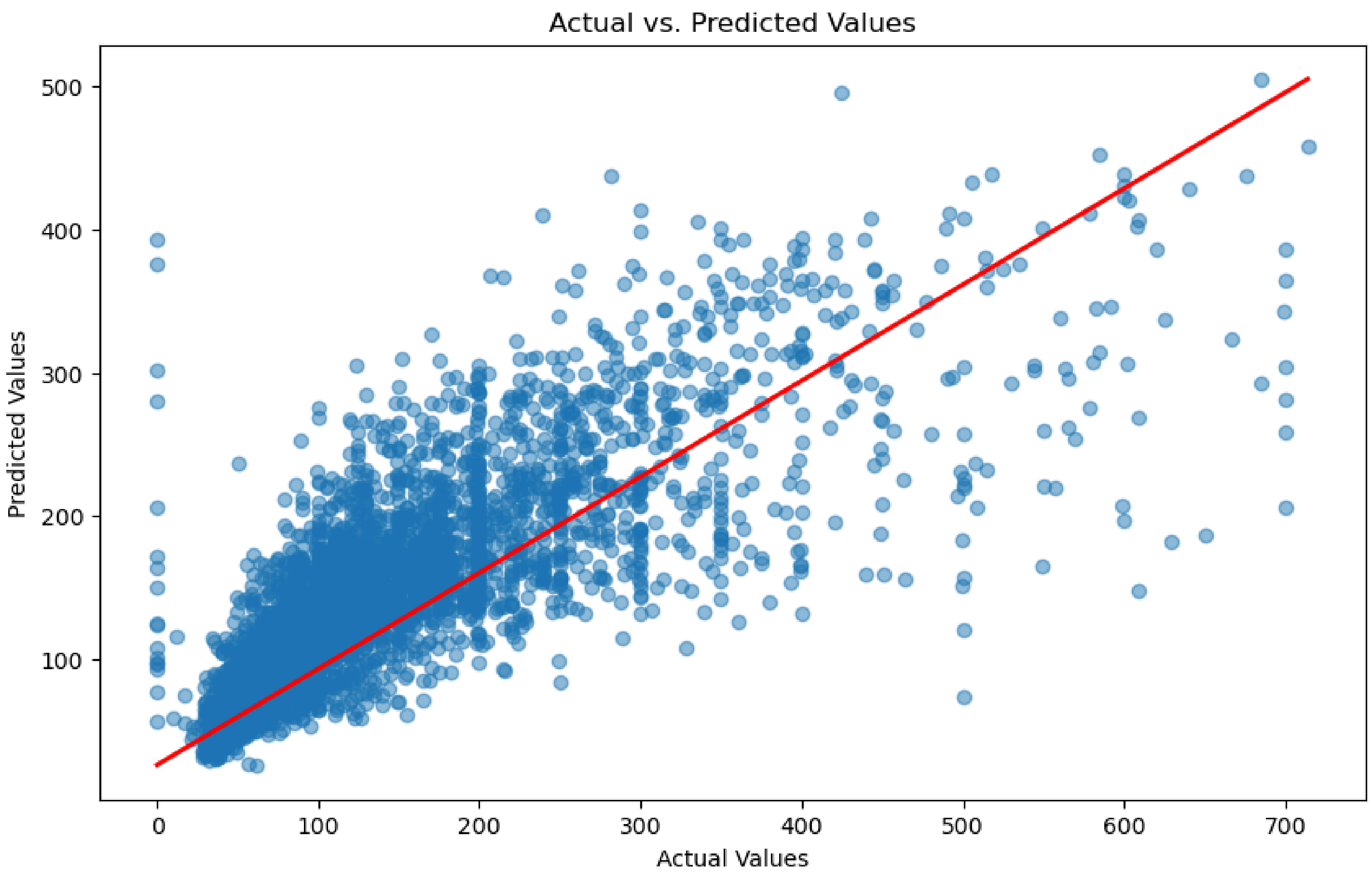
After running Random Forests Regressior we got these values for our model:



```
Train score 0.9861034121393967
Val score 0.6011329555567523
MAE: 42.260882620755645
MSE: 4470.107358876812
RMSE: 66.85886148355215
```

Random Forest: Regression Evaluation

An validation score of 60% shows that our hyperparameter tuning worked to some extent. However, there is a high MSE and RMSE values and this shows that the model can be improved more. We then compared our predicted values from our model to the actual values.

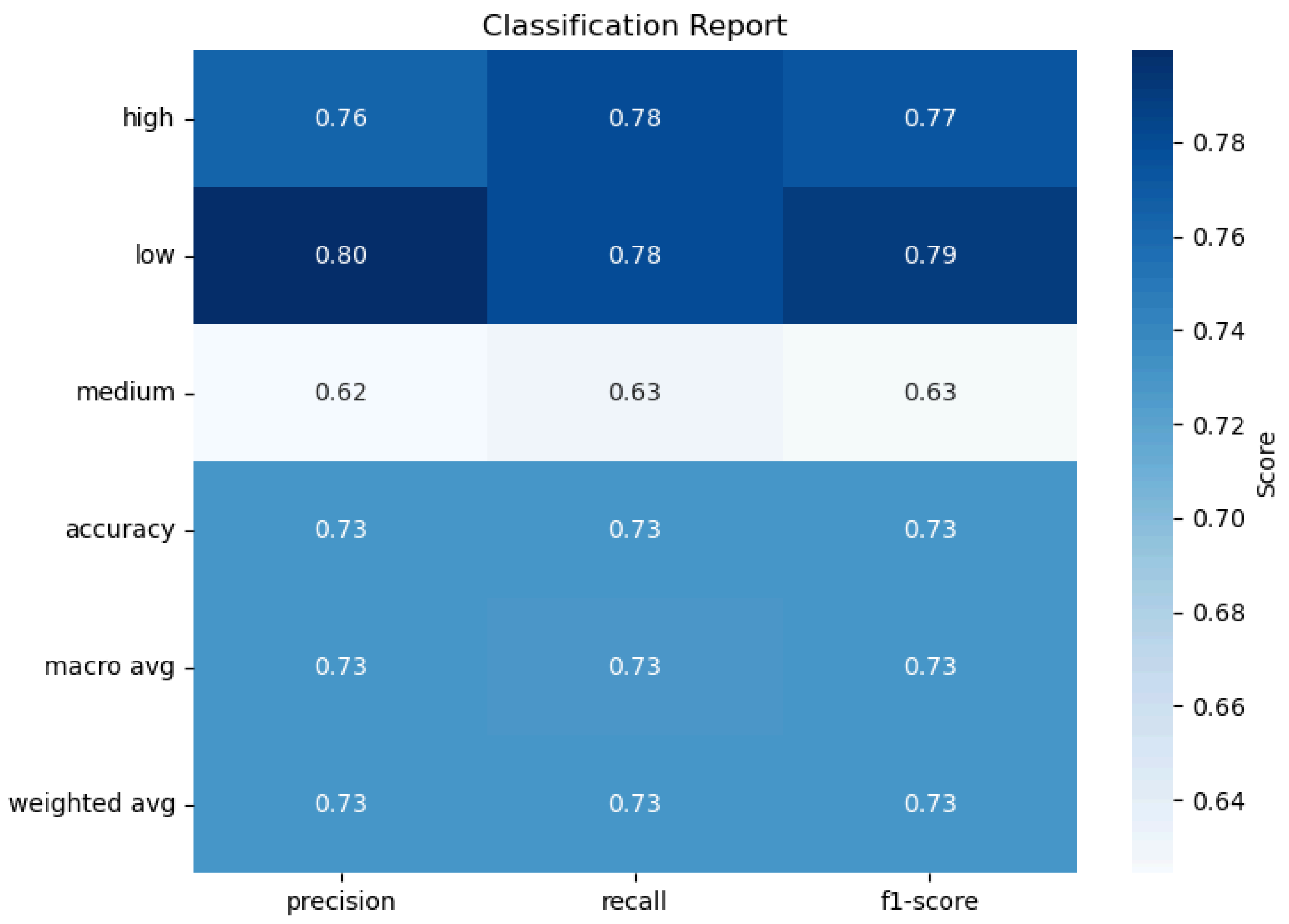


Random Forests: Actual vs. Predicted Values

Overall, there is a high density of data points that follow the trend line, however there is an abundance of values over/under the trendline, which means that there is a substantial amount of error in the model.

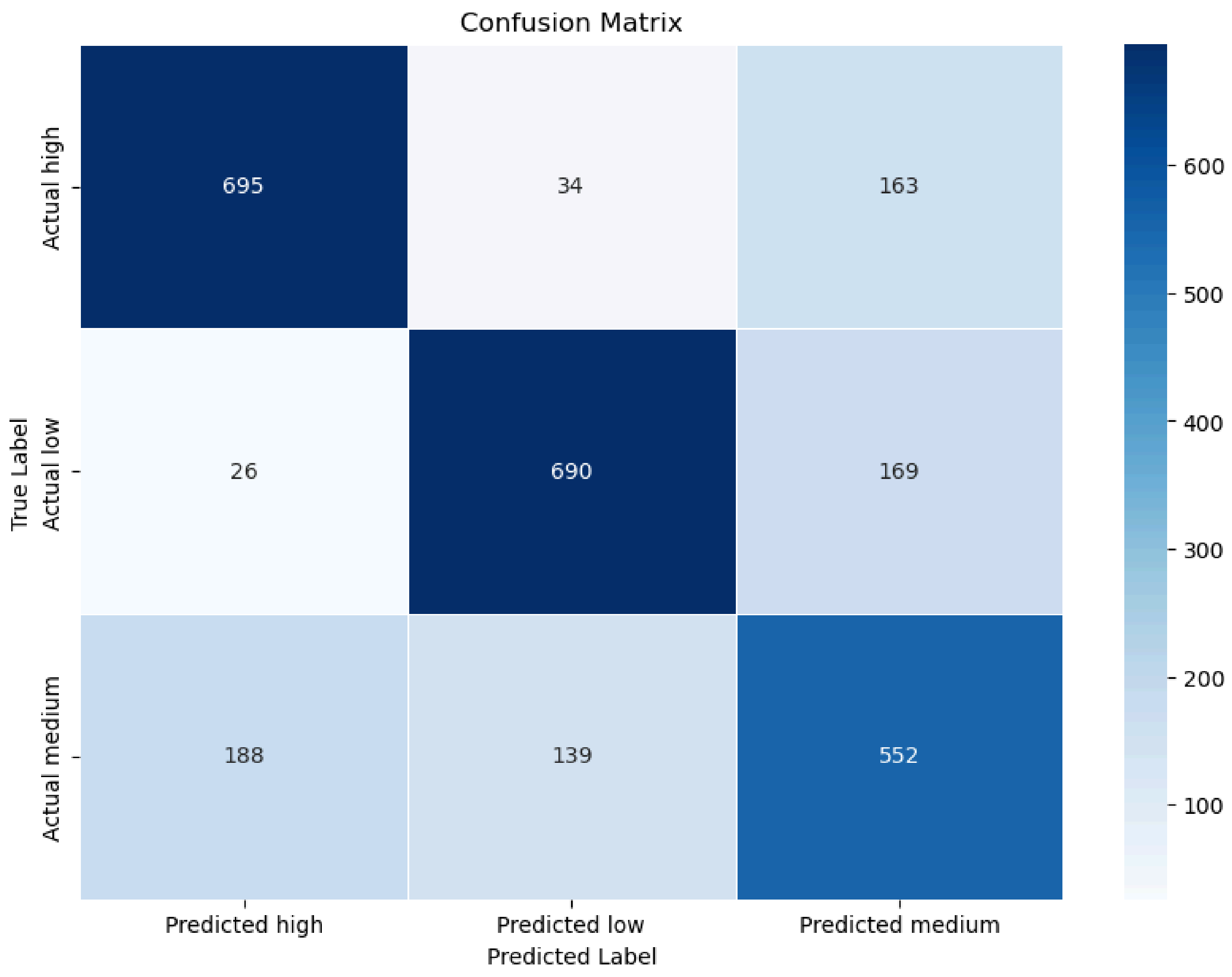
Random Forest Classification

We wanted to try classify the data between low, moderate, and high categories to see if we have better improvement. First, we need to determine what price ranges can be categorized as "low", "moderate", or "high". After running our Classifier, we got a accuracy of 72.7%. We wanted to try to improve the accuracy using hyperparameter training like the Regression. However, we were not able to improve it and got an accuracy of 72.3%. We then tried to use Bagging to improve our accuracy but we got 71.9%. We could not improve our accuracy and we believe that this is the best we could get with the current data. We then generated a classification report from the origial classification.



Random Forest: Classification Report

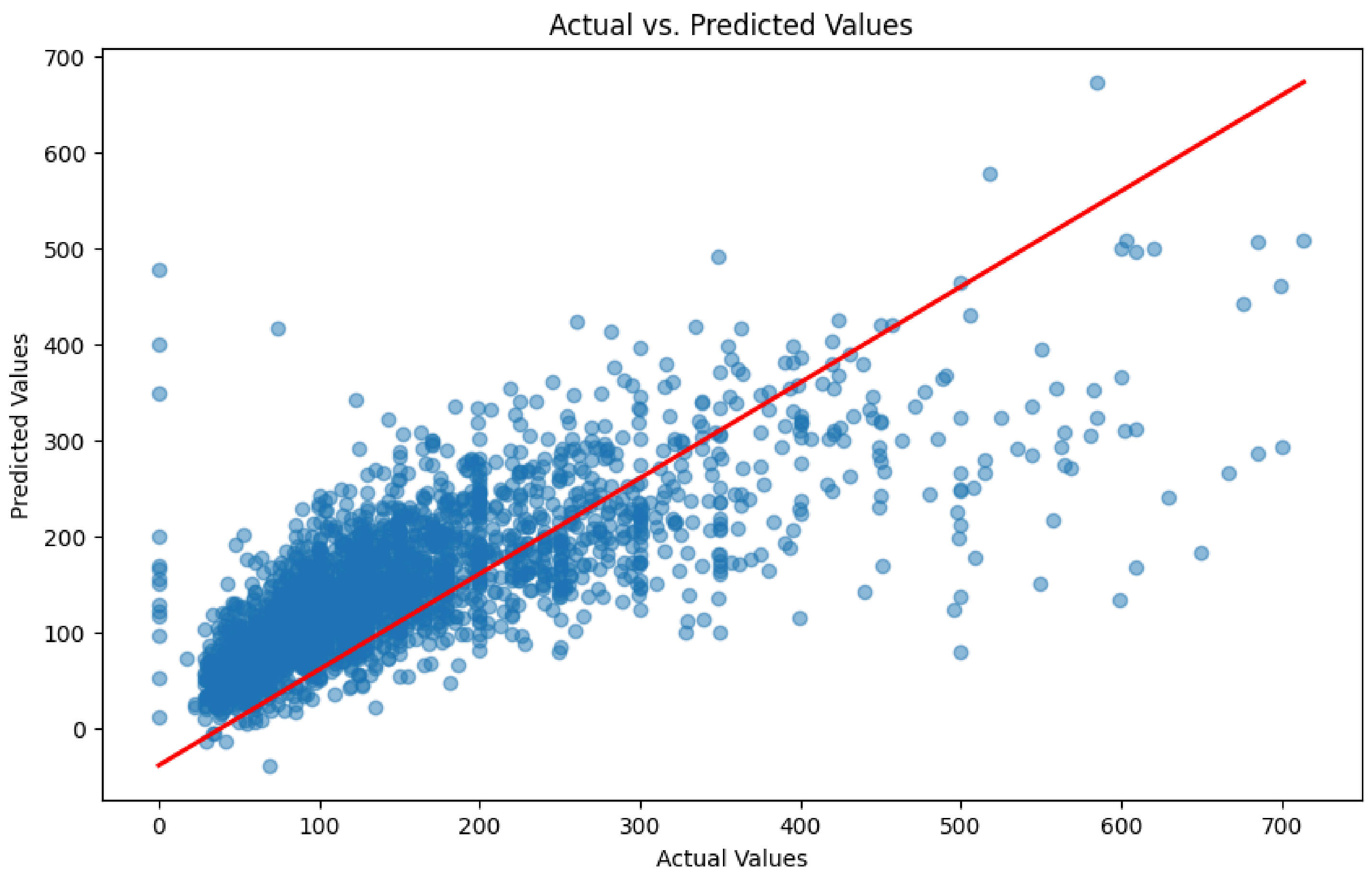
We were able to get a high f1 score for the high (0.77) and low (0.79) priced Airbnbs, however the medium priced data was lower with a value of only 0.61. We then generated a confustion matrix to show accurate each classification was.



Random Forest: Confustion Matrix

Neural Networks

We implemented a Multi-Layer Perceptron (MLP) Regressor model for the Airbnb data to predict the prices of listings. We started by preprocessing the data, including encoding categorical features, imputing missing values, and scaling the features. We then created a pipeline that incorporated feature selection using Lasso regression, polynomial feature engineering, and the MLP Regressor model. We performed hyperparameter tuning using GridSearchCV to find the best hyperparameters for the MLP Regressor. Additionally, we enabled early stopping to prevent overfitting and introduced ensemble methods like Gradient Boosting Regressor. Finally, we evaluated the performance of the models using metrics like R-squared, Mean Absolute Error (MAE), and Mean Squared Error (MSE).



MLP Model Actual vs Predicted Prices

Using MLP, we achieved the R-squared value of 0.5578599040164622 suggests that the neural network regression model explains approximately 55.79% of the variance in the data.

Conclusion

After testing and analyzing several different predictive models: 3 types of linear modeling, random forest classification, and neural networks, we have found the random forest classifier to yield the best results. The linear regression models were not well suited to the data. The random forest regressor had the highest validation accuracy of 72.74%, and the classifier was close at 60.11% validation accuracy. Our neural network had the second highest predictive capability, at a much lower accuracy of 55%. Linear regression correlation peaked at 35.794%.

To improve these results, more extensive feature engineering could occur to capture underlying data patterns more effectively. Hyperparameters could be better optimized for the model. Class imbalance could be dealt with using SMOTE.

By analyzing data from NYC airbnbs, we could gain insights into what determines demand and pricing. The results were interesting and provoke questions of further fine-tuning models or using them to create commercial price setting tools or informed buying options.

References:

[1] Alharbi ZH. A Sustainable Price Prediction Model for Airbnb Listings Using Machine Learning and Sentiment Analysis. Sustainability. 2023; 15(17):13159. https://doi.org/10.3390/su151713159

[2] J. Dhillon et al., "Analysis of Airbnb Prices using Machine Learning Techniques," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), NV, USA, 2021, pp. 0297-0303, doi: 10.1109/CCWC51732.2021.9376144.

[3] Kalehbasti, Pouya, et al. Airbnb Price Prediction Using Machine Learning and Sentiment Analysis. abs/1907.12665, Springer International Publishing, 2019.

[4] Truong, Quang, et al. “Housing Price Prediction via Improved Machine Learning Techniques.” Housing Price Prediction via Improved Machine Learning Techniques, Procedia Computer Science, 27 July 2019, www.sciencedirect.com/science/article/pii/S1877050920316318

[5] “Binning a Column with Python Pandas.” Saturn Cloud Blog, 28 Dec. 2023, saturncloud.io/blog/binning-a-column-with-python-pandas/.

Contribution Table:

Name	Proposal Contributions
Kristian Chappell	Random Forest + Visualizations
Yash Gupta	Random Forest / Neural Network
Joshua Lu	Random Forest
Freya Nair	Neural Network + Visualizations
Somtochukwu Nwagbta	Neural Network

Final Gantt Chart:

GANTT CHART

PROJECT TITLE	Airbnb Listing Predictor

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Team Composition	All	1/15/24	2/2/24	17
Project Proposal				
Introduction & Background	Yash	2/2/24	2/22/24	20
Problem Definition	Yash	2/2/24	2/22/24	20
Methods	Josh	2/2/24	2/22/24	20
Potential Results & Discussion	Kris	2/2/24	2/22/24	20
Video Creation & Recording	Freya	2/2/24	2/22/24	20
GitHub Page	Somto	2/2/24	2/22/24	20
Model 1				
Model 1 (M1) Design & Selection	All	2/26/24	2/27/24	1
M1 Data Preprocessing	Yash, Kris, Josh	2/27/24	3/1/24	4
M1 Data Visualization	Yash, Kris	3/1/24	3/7/24	6
M1 Implementation & Coding	Somto, Freya	3/1/24	3/13/24	12
M1 Results Evaluation	Freya	3/13/24	3/16/24	3
Model 2				
Model 2 (M2) Design & Selection	All	3/14/24	3/15/24	1
M2 Data Preprocessing	Yash	3/14/24	3/16/24	2
M2 Data Visualization	Kris, Josh	3/25/24	3/27/24	2
M2 Coding & Implementation	Freya, Kris, Somto	3/25/24	4/4/24	9
M2 Results Evaluation	Josh	4/4/24	4/6/24	2
Midterm Report	All	3/16/24	3/29/24	13
Model 3				
Model 3 (M3) Design & Selection	All	4/4/24	4/5/24	1
M3 Data Preprocessing	Yash	4/5/24	4/7/24	2
M3 Data Visualization	Yash	4/7/24	4/10/24	3
M3 Implementation & Coding	Josh, Freya, Somto	4/7/24	4/15/24	8
M3 Results Evaluation	Kris	4/15/24	4/17/24	2
Final Report				
M1-M3 Comparison	All	4/17/24	4/19/24	2
Video Creation & Recording	All	4/19/24	4/23/24	4
Final Report	All	4/19/24	4/23/24	4