Utilizing machine learning to assist in indicating AirBNB listing costs in Seattle.

Lim Hur

***Abstract*—** An AirBNB refers to Air, Bed and Breakfast, which is also commonly known as a listing property, rented from a homeowner for a period of time. AirBnB units are commonly known for its affordable accommodation costs as compared to other alternatives such as hotels. With Airbnb providing the freedom of giving prices, hosts can competitively select their listing prices.

Typically, the process of renting an Airbnb unit is as follows: a customer would pre- book a unit of their personal choice, and indicate their requirements, like room count, etc and pay for the duration of stay. From the perspective of an owner, the owner must provide a listing price based on the apartment they are renting.

Given this, confusion may arise for many customers when trying to determine if the listing is well worth for its cost or not. Likewise, from the perspective of an owner, the idea of balancing profitability and competitiveness can be challenging.

For a very long time, Seattle has one of the strongest AirBnB short term rental markets in the world, generating very high revenue. Seattle had the highest growth rate revenue of 118% in 2015 [2].

In this paper, the prediction task is to leverage on labelled data to develop an accurate model to indicate the price of an AirBnB listing in Seattle. Through extensive Feature Engineering techniques, utilizing unsupervised clustering techniques, that group data items into cluster, such that data in a cluster is 'similar to each other' [3].

I aim to build a highly accurate model to predict the listing cost of an AirBnB. Moreover, I would like to provide more insights on listings aspects that would be of high relevance to the costs.

I. Introduction

Machine learning is defined as computer algorithms, with the aim of imitating human intelligence by learning from its surroundings. By leveraging on data, they are considered the 'working horse' in this era. [1]. Machine learning techniques have been applied to various industries such as manufacturing, entertainment, healthcare, etc. In this paper I will be covering supervised learning technique known as regression.

II. Related works

Previously, in 2015, there were works published related to predicting prices for a new listings. A project was done to investigate the price of Airbnb listings in San Francisco, United States. The dataset used was collected from Airbnb. They utilized techniques and models such as upsampling, downsampling as well as Random Forests to aid their predictions and balancing the dataset. They have found out that a balanced dataset while using Random Forests have produced an impressive result. However, an imbalanced dataset, with high frequency of some samples was skewing predictions. [4]

Another example of related works would be the prediction of housing prices through sentiment analysis, after extensively testing models like K-Means Clustering, SVR, Neural Networks, and after extracting features from reviews from customers, which ultimately improved the performance. It was discovered that SVR outperformed other models, with an MSE of 0.147 and R^2 of 69%. This was done for New York City[5].

III. Dataset

Our dataset is initially provided from insideairbnb, providing detailed listings information. The initial dataset had 92 attributes, however this would be extremely difficult for us to perform data analysis on, so I had decided to drop irrelevant columns that I would not be using in the model and analysis such as url links, image links, IDs, and repeated attributes, or NULL attributes.

The final dataset has a total of 1393570 rows and 26 columns, after dropping irrelevant columns. We have an exhaustive lists of listing aspects pertaining to listings, such as bedrooms, amenities, and even review scores.

Table 1. Attributes of dataset

listing_id	unique identifier
date	date of listing
price_x	base price
host_response_time	average response time by host
host_response_rate	host response rate
host_is_superhost	if host is superhost
host_listings_count	host listings count
host_verifications	verification methods
host_identity_verified	if host is verified
neighbourhood_group_cleansed	neighbourhood group of listing
zipcode	zipcode of listing
latitude	latitude of listing
longitude	longitude of listing
property_type	listing type
room_type	room type

accommodates	no of max guests

bathrooms	*no of bathrooms*
bedrooms	*no of bedrooms*
beds	*no of beds*
bed_type	*bed type*
amenities	*amenities*
guests_included	*charge for extra guests per person per night*
minimum_nights	*min nights*
maximum_nights	*max nights*
number_of_reviews	*no of review in all years*
review_scores_rating	*rating score*

3.1 Cleansing before Data Analysis

We will perform some general cleansing before EDA.

Perform some general cleansing so as to facilitate EDA:

```
def general_cleansing(df):
    # Drop date column after extracting time data.
    df['Month'] = df.apply(lambda i: int(i['date'].split('-')[1]),axis=1)
    df['Year'] = df.apply(lambda o: int(o['date'].split('-')[0]),axis=1)
    df = df.drop(columns = ['date'])

    df['price'] = df['price_x'].astype(str)
    df['price'] = df['price'].str.replace("[$, ]", "").astype("float")
    df = df.drop(columns = ['price_x'])
    df['host_response_rate_no'] = df['host_response_rate'].astype(str)
    df['host_response_rate_no'] = df['host_response_rate_no'].str.replace("%", "").astype("float")
    return df
df = general_cleansing(df)
```

```
def OHE(df, column_name):
    d = {}
    for unique_value in df[column_name].unique():
        for value in unique_value.replace('[', '').replace("'", '').replace("]", '').replace('"', '').replace('{', '').replace('}', '').split(','):
            if value in d:
                d[value] = d[value] + 1
            else:
                d[value] = 0
    values_sorted = sorted(d.items(), key=lambda pair: pair[1], reverse = True)

    for value in values_sorted[: 10]:
        df[column_name + value[0]] = df.apply(lambda row: extract(row, column_name, value[0]),axis=1)
```

Some of our attributes had interesting values in the form of a python list, I had to extract the items from the list, and transforming them into a new column for each list item, in a way 'One Hot Encoding' it.

3.2 Exploratory Data Analysis

The term Exploratory Data Analysis was introduced by John W. Tukey shows how graphical techniques can be used to explore data. Through EDA, we can discover underlying structures, detect outliers and anomalies in the dataset, and most importantly, discover suitable models for our scenario. [6]

3.2.1 Target Variable Analysis

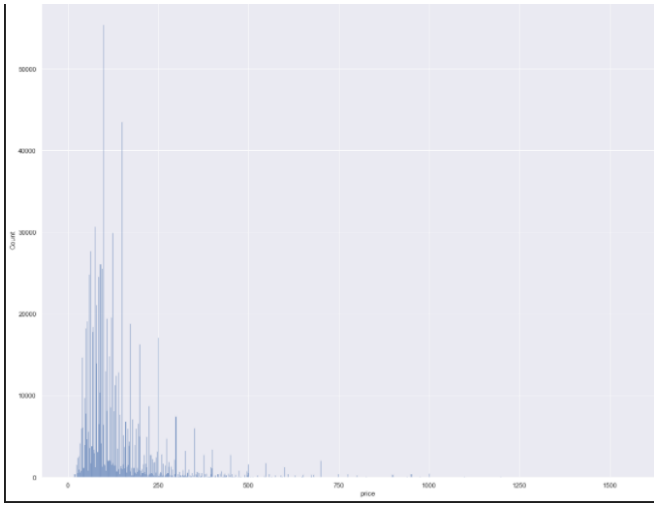


Figure 1.

In figure 1, we note that our target variable price is highly right skewed, skewness, which mean the distortion from normal distribution, can pose a problem for our model. As such, we can attempt to log transform to make highly skewed distributions less skewed, this can help us have easier time discover patterns, as well as make the data more interpretable. [7].

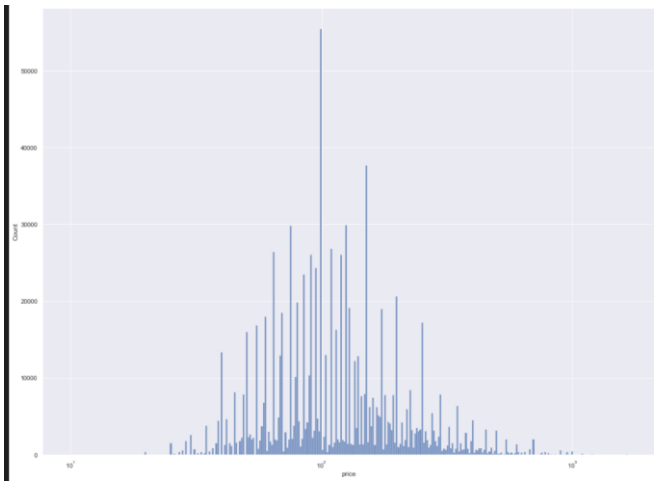


Figure 2.

We can see from figure 2, after log transformation, our data is more normalized.

3.2.2

PHIK correlation

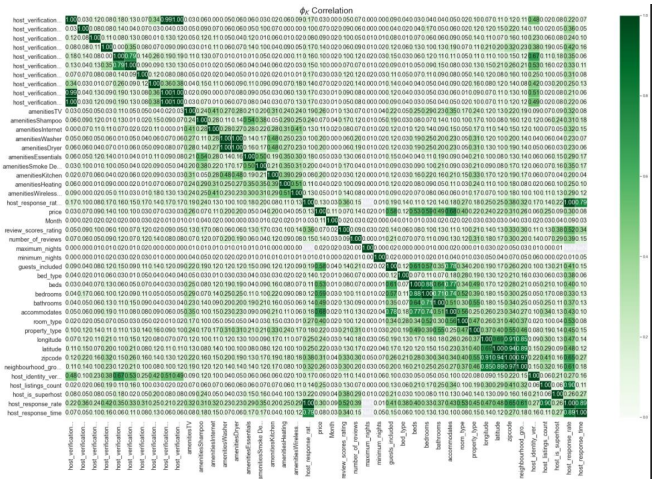


Figure 3.

ϕ_K correlation is implemented through python *phik* library. It is a new correlation coefficient that works well between numerical and categroical variables, which is of high relevance in this dataset. 0 denotes low correlation, while 1 denotes perfect correlation. A heatmap is generated as seen in figure 3. [8]

- We immediately notice that there are several features that are highly correlated to each other which may introduce the problem of multicollinearity, which will reduce the accuracy of coefficients, and may make our model unstable.
- ['bedrooms(.59)', 'bathrooms (.49)', 'accommodates (.68)'] are highly correlated to target variable, this makes sense as An Airbnb listing may cost higher given a listing with higher bathrooms.
- Maximum_nights have the lowest correlation, it is likely due to the fact this does not affect the price at all.
- Several variables have little correlation between each other, hence we note that feature selection can benefit us to pick the most powerful features.

3.2.3 Exploring prices by latitude and longitude

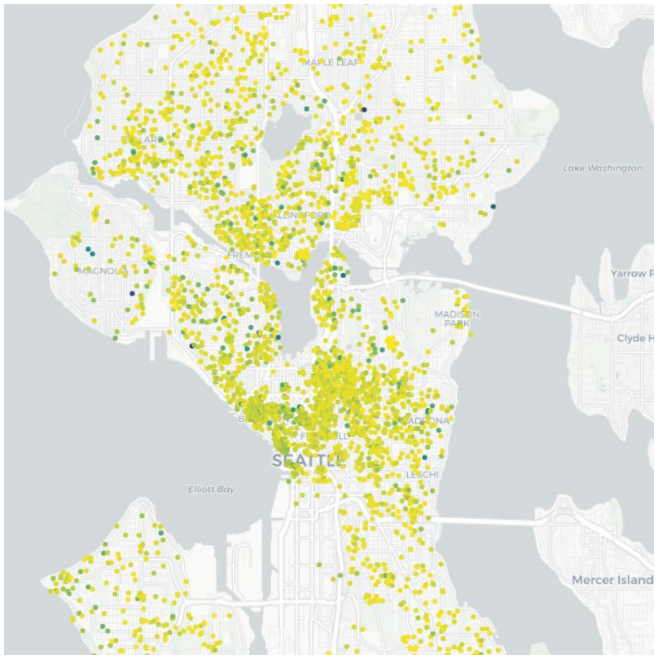


Figure 4.

We note that in seattle, the Airbnb prices tend to be more expensive in the central regions. Apart from this, south seattle also has some pricey listings, such as near the water front of the west region, Diving deeper, we can see that prices near Queen Anne are more expensive, which may be due to the stylish shops near the area.

3.2.3 Exploring some continuous features

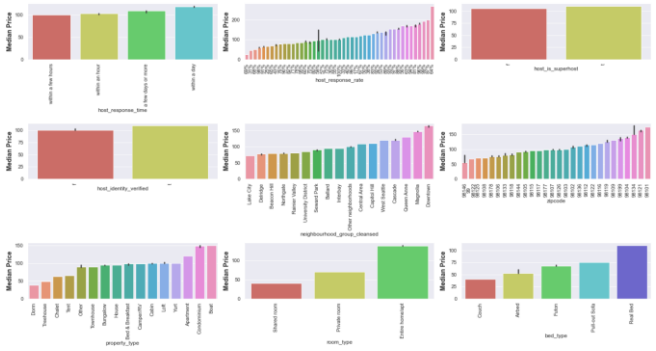


Figure 5

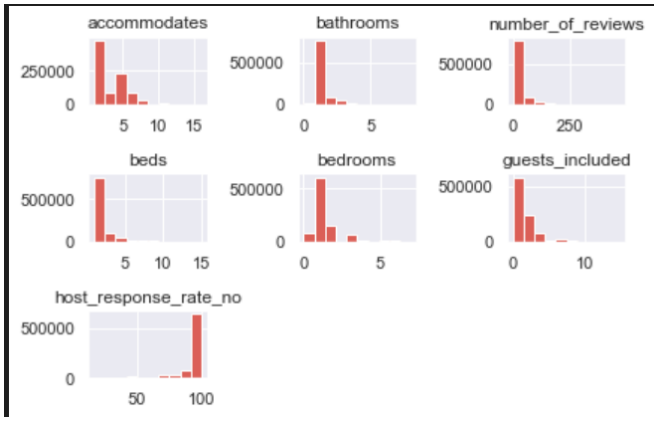
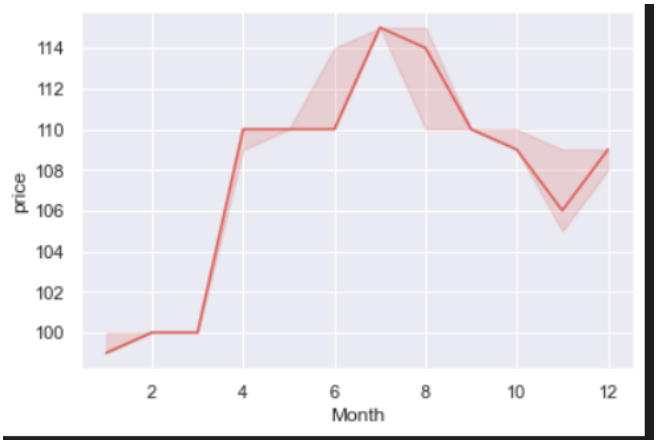


Figure 6

From Figure 5, Boat and condominium property type costs the highest, Similarly, having the entire apartment costs more than living in a shared/private room. This would make sense as factors such as privacy, can be a main reason for the price difference. Downtown neighborhood are the most expensive place to get a listing unit. Generally, there Is no trend seen from host response rate and price of listing.

From figure 6, we can see that some features have highly skewed distributions, such as accomodates, and guests_included. Similarly, inspecting the x axis shows that features are of different scales, hence we need to introduce feature scaling later on when we model.

3.2.4 Time series features



We can see that the price of listing varies throughout the months, and generally shows an increasing trend. Hence, this would be a valid feature in our model. Prices raises significantly during summer, may be due to high demand and lower supply. An increase in december (winter holiday periods) is also obsesrved. Airbnb prices tend to increase during seasons.

IV. Preprocessing

Before Modelling, we will preprocess the data. We will drop listing_id, longitude, latitude, and features. Furthermore, we note that there are presence of NULL values in this dataset. This would not be good as most models do not accept nulls. To counter this, we will perform further preprocessing through SKLearn's Pipeline, utilizing its IterativeImputer(), which is a multivariate method to impute data. [9]. Pipeline also helps to prevent data leakage.

I went to partition my data into 80:20 ratio of train:test, Since this is a very large dataset, 20% would be a good amount.

```
X,y = df.drop(columns = 'price'),df[['price']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)
```

V. Feature engineering

Use of Sklearn's Agglomerative Clustering technique to cluster **zip code** into 6 clusters. Thereby, dropping the original zip code column. Since they are nominal in nature, we can one hot encode them later. Through this, we can further represent our location data in our model.

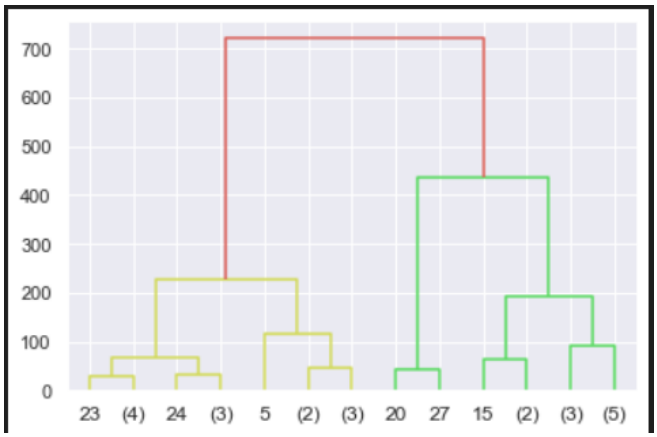


Figure 7.

From figure 7, we can make a partition at y=100, making a total of 6 clusters. Hopefully, through this the model can ultimately take into account the geo location.

5.1 Preprocessing categorical feeatures in pipeline

```
ohe_cols = ['neighbourhood_group_cleansed', 'property_type', 'room_type', 'bed_type', 'host_response_time', 'clust']
mask = X_train.columns[np.where((pd.DataFrame(X_train.dtypes == 'object')[0]== True) )  ]
cat_cols = [i for i in mask if i not in ohe_cols]
numeric_cols = [i for i in list(X_train.columns) if i not in list(cat_cols) and not i in list(ohe_cols)]
preprocess = Pipeline([
    ("features", FeatureUnion([
        ('ohe', make_pipeline(Columns(names=ohe_cols), OneHotEncoder(drop='first', handle_unknown='ignore', sparse=False)  )),
        ('numeric', make_pipeline(Columns(names=numeric_cols),RobustScaler() )),
        ('categorical', make_pipeline(Columns(names=cat_cols),CustomLabelEncoder(cat_cols)))
    ])
])
```

We will separate the onehot encoded features, binary features as well as numerical features. Firstly, as in in Scikit learn's FeatureUnion class, I will use one hot encoder to encode features that have no clear order among its categories. Moreover, since some of the features have high number of unique values, we may have a problem of high cardinality, which would bring in a problem of extra dimensions. Hence, I further deem feature selection to be extra useful for us.

Next, for binary features, I have applied a label encoder, which encodes it to ones and zeros. I deep this would be more relevant than one hot encoding as, binary features have only 2 values.

5.2 Feature scaling

For numerical columns, I did not want features with higher magnitude to 'control' the model, hence I applied a robust scaler. Moreover, Since some of our features are right skewed, robust scaler can handle outliers well.

$$X_{scale} = \frac{x_i - x_{50}}{x_{75} - x_{25}}$$

To perform scaling, we subtract the median, and then dividing by the interquatile range, this is the formula for robust scaling. Use of sklearn's robust scaler to help me to perform this.

Finally, I chained all steps into the pipeline, ready for Cross validation.

5.2.1 EVALUATION METRICS

We will be using several evaluation metrics to select our model. Our primary metric will still be using RMSE, root mean squared error. I will be also including other metrics like MAPE, Mean absolute percentage error and R2 as well, just not as much. (from notebook)

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}$$

RMSE is easier to interpret, and it will penalize the model when it makes large errors in predictions, as the residuals are squared. Such will help us to be certain of the utility of the final model, making sure it does not predict observations that are very far from actual observations.

VI. DISCUSSION

6.1 Model Selection

We will be adopting several techniques for model selection, namely K Fold cross validation, and scrutinizing of learning curves of models to determine the best model through its biases and variance. Moreover, we can also tell if the model is overfitting or underfitting.

KFold CV: by setting k to an integer, example k=5, we will partition the training set into 5 different folds, 4 of them will be used for model training, while the last fold will be used for evaluation/prediction, and computation of metrics score. This process is repeated 5 times, with a set previously from the train set, swapped as the test set for each split, all 5 folds will be in place of the 'test/CV' set once. We will average the scores after cross validation

```
plt.figure(figsize=(16,9))
scores_final = model_evaluation([dummy, lasso,ridreg,svm,knr,dectreereg,gbr],X_train,y_train)
```

Model	***CV RMSE***	***CV MAPE***	CV R^2
Dummy Regressor	***109.3***	***0.49***	***0.075***
Ridge	*65.063*	*0.32*	*0.618725*
Decision Tree	*15.809*	*0.033*	*0.9802*
Lasso	*67.313825*	*0.32*	*0.5919*
Gradient Boosting	*52.8706*	*0.26*	*0.748275*
KNN	*16.3*	*0.03*	*0.98*
SVR	*65.0415*	*0.26*	*0.57*

After cross validation, Dummy classifier performed the worst with an unimpressive RMSE of 109, severely underfitting the data with high bias, which is awful considering that Air Bnb prices are measured per day. This will be a reference point.

We note that the best performing model is decision tree regressor, possessing a low bias and slightly low variance. Moreover, it is also less computationally expensive. Hence, it was definitely a better choice than ensembles like gradient boosting, which possess high bias, slighlty underfitting the data. Distanced based models like KNN performed quite well, with an impressive RMSE, however they pale in comparison to tree models like decision trees. Support vector machines however, they posses high bias, as well as slightly overfitting the training set.

Linear models performed poorly with high 60s RMSE, underfitting the data, suggesting high bias, which may not generalize well to new observations.

Our final model is decision tree regressor, which possesses the best metric score and does not severely underfit or overfit, suggesting that it will work well to new examples.

6.2 Model Improvement

Since we have seen that the target variable is quite skewed, we will attempt to log transform it using sklearn's TransformedTargetRegressor.

```
model = Pipeline([
    ('Prep',preprocess),
    ('Impute', IterativeImputer(random_state=42)),
    ('model', TransformedTargetRegressor(DecisionTreeRegressor(random_state=42),func=np.log,inverse_func=np.expm1))
])
```

This had helped our model in terms of lowering the variance.

6.3 FEATURE SELECTION

Feature selection brings about a lot of benefits, especially with datasets with large number of attributes, and it can even improve accuracy scores, decrease the training time, to reduce overfitting, and to avoid the curse of dimensionality [11]

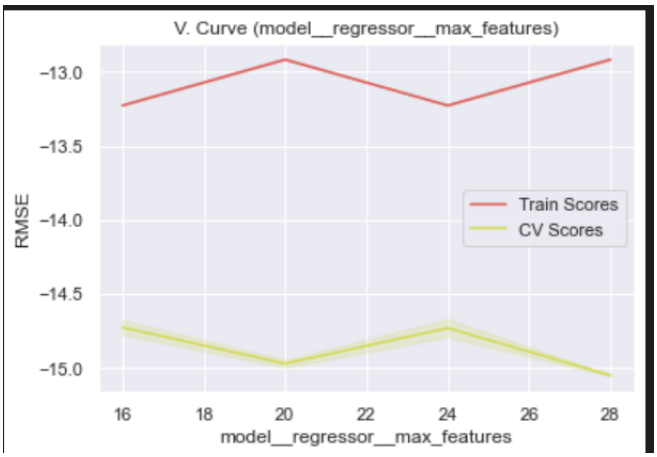
6.4 Recursive Feature Elimination.

Since we have a slight problem of having large number of features, this would introduce the problem of extra dimension spaces. Hence, we will attempt to use a wrapper type algorithm that select features that are the most relevant in predicting target variable.

```
model_rfe = Pipeline([
    ('Prep',preprocess),
    ('Impute', IterativeImputer(random_state=42)),
    ('RFE', RFE(DecisionTreeRegressor(random_state=42), n_features_to_select=0.5)),
    ('Model', TransformedTargetRegressor(DecisionTreeRegressor(random_state=42),func=np.log,inverse_func=np.exp))
])
```

RFE although did not really drastically improve the score, it has helped us attain a simpler model.

Before embarking on hyperparameter tuning, we will be plotting validation curve for max_depth to accurately determine the good range to gridsearch for. Max depth controls the depth of the tree, hence this can also help to control the overfitting as well as lowering the variance even more. We can see that further increasing may result in overfitting, hence, we will try to keep our hyperparameter tuning in the similar range as this curve.



VII. Hyperparameter Tuning

Lastly, we will attempt to tune some hyperparameters for Decision Trees, namely max features and max depth. Max depth can help reduce the overfitting risk, and finally we will evaluate if it will help us. Hyperparameter tuning is done via halving Gridsearchcv , which is much faster than GridSearchCV. HalvingGridsearchCV can find good parameters in less time as well. [10]

```
from sklearn.experimental import enable_halving_search_cv  # noqa
from sklearn.model_selection import HalvingGridSearchCV
result = HalvingGridSearchCV(model,param_copy,cv=2,random_state=42,scoring=rmse,n_jobs=-1)
```

The final model on an independent unseen test set, and we can see that our model performs well with a low RMSE, and hence we can conclude that our model can generalize well to new observations, and is reliable to use as an indicator of Airbnb listing costs. RMSE: 14.5

```
1  preds = final_model.predict(X_test)
2  print(mean_absolute_error(y_test,preds))
3  print(mean_absolute_percentage_error(y_test,preds))
4  print(mean_squared_error(y_test,preds,squared=False))
5  print(r2_score(y_test,preds))
✓ 5.1s
4.900814315739521
0.03189636677221292
14.577159630117269
0.9804598912183169
```

Using a residual plot we can see that, some points are scattered randomly from the y=0 line, we can conclude that the model is very approriate in our scenario.

Our mean residuals is shown above, and since the residuals are positive, means the model has slightly underpredicted the price of listings.

Conclusion

In conclusion, we have managed to develop a regression model, with the aid of feature engineering techniques, model improvement approaches. Moreover, we can rest assure that our data is able to generalize well to new data, given that we have managed to overcome the bias variance trade off. Although this model is more suited for the context of Seattle and the results may be different for each city, but the current approach still can be used for other cities' data.

Acknowledgements

Thank you, Dr Wilson and other tutors, for implementing a Part C (technical paper) into this assignment, I found this study extremely insightful.

References

[1] Issam El Naqa, Martin J. Murphy. What is Machine Learning?, 2015 ISBN : 978-3-319-18304-6 URL: https://link.springer.com/chapter/10.1007/978-3-319-18305-3_1

[2] Lane J.,(2020). The Sharing Economy checks in: An analysis of AirBnB in the United States. http://rss.hsyndicate.com/file/152006265.pdf

[3] Nizar Grira, Michel Crucianu, Nozha Boujemaa INRIA Rocquencourt, B.P.105 https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.3581&rep=rep1&type=pdf

[4] Paridhi, C. Aniket, J. Rahul, B. Unravelling Airbnb predicting price for New Listing https://arxiv.org/ftp/arxiv/papers/1805/1805.12101.pdf

[5] Pouya Rezazadeh Kalehbasti, Liubov Nikolenko, Hoormazd Rezaei Machine Learning and Knowledge Extraction, 2021, Volume 12844 ISBN : 978-3-030-84059-4 https://link.springer.com/chapter/10.1007/978-3-030-84060-0_11

[6] Hinterberger, H. (2009). Exploratory Data Analysis. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA.

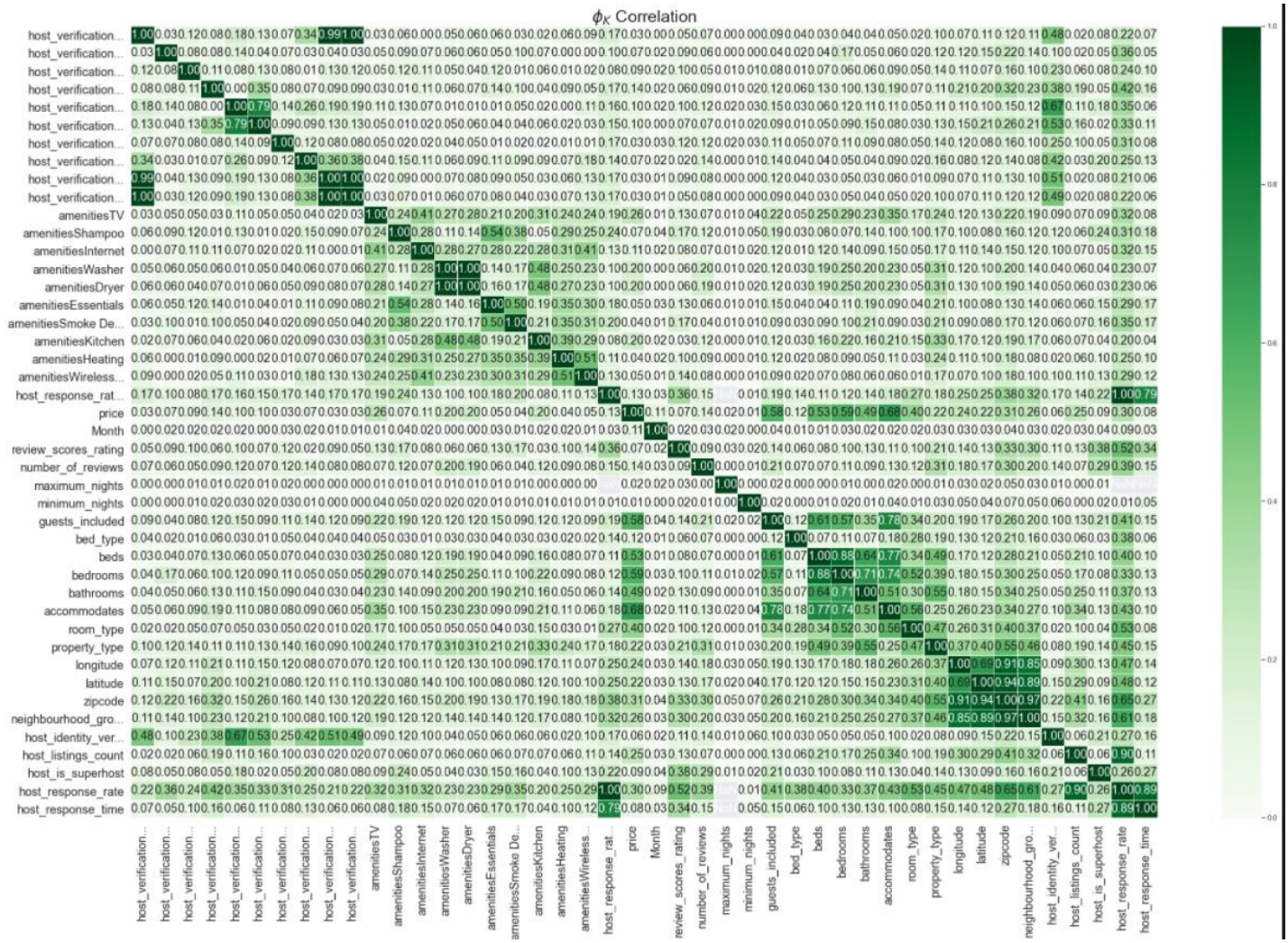
https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_1384
[7] Lane,D. Log Transformations.
https://onlinestatbook.com/2/transformations/log.html
[8] M. BAAK, R. KOOPMAN, H. SNOEK, S. KLOUS. (2019). A NEW CORRELATION COEFFICIENT BETWEEN CATEGORICAL, ORDINAL AND INTERVAL VARIABLES WITH PEARSON CHARACTERISTICS. https://arxiv.org/abs/1811.11440

[9] Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011. Scikit-learn: Machine Learning in Python,

[10] Glide L. (2021). Faster hyperparameter tuning with scikit-learn's HalvingGridSearchCV. https://towardsdatascience.com/faster-hyperparameter-tuning-with-scikit-learn-71aa76d06f12

[11] Chen, RC., Dewi, C., Huang, SW. *et al.* Selecting critical features for data classification based on machine learning methods. *J Big Data* **7,** 52 (2020). https://doi.org/10.1186/s40537-020-00327-4

Bigger version of figure 3.


ϕ_K Correlation
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
host_verification...
amenitiesTV
amenitiesShampoo
amenitiesInternet
amenitiesWasher
amenitiesDryer
amenitiesEssentials
amenitiesSmoke De...
amenitiesKitchen
amenitiesHeating
amenitiesWireless...
host_response_rat...
price
Month
review_scores_rating
number_of_reviews
maximum_nights
minimum_nights
guests_included
bed_type
beds
bedrooms
bathrooms
accommodates
room_type
property_type
longitude
latitude
zipcode
neighbourhood_gro...
host_identity_ver...
host_listings_count
host_is_superhost
host_response_rate
host_response_time