Airbnb Review Score Prediction Model

Cormac Madden

January 5, 2023

1 Feasibility of Predicting a Listing's Rating

This report evaluates the feasibility of predicting for a given Airbnb listing the following ratings: overall rating, location, accuracy, cleanliness, check-in, value and communication ratings.

1.1 Feature Variables and Approach to Each Rating

This first section gives an overview of the approach and feasibility of predicting 5 of the 7 ratings. The following section is the implementation of the remaining 2 ratings "overall_rating" and "communication_rating".

For each of the ratings a combination of several approaches could be taken.

- Just the listings variables could be used to predict the ratings. The relevant variables would be different for each rating and which ones are significant could be analysed by using regression coefficients.
- The comments for each listing is could also be used as feature variables. Once again the significant words could be determined and would depend on the rating being predicted.
- Variables from the listings could be transformed or combined with one anther to create new feature variables.
- External pre-trained NLP models could be used to provide additional information on the comments, that could be then used to train our model.

1.1.1 Location Rating

The variables I would expect to be most significant when predicting the rating of the location are the coordinates, the neighbourhood and the reviews that include terms such as "area, location, surroundings, view etc". Figure 1 shows a scatter plot that shows the ratings of the listings based on their coordinates to evaluate if this may be a useful feature variable. There is some indication that the ratings are quite high in areas such as the south of the city center and along the sea front, whereas north and more west of the city has more dark orange spots.

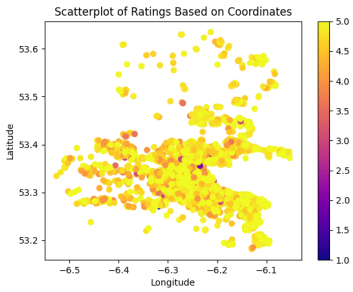

Figure 1: Scatter-plot of Ratings From their Coordinates

1.1.2 Accuracy Rating

Accuracy is a difficult rating to predict. Feature variables that may be useful are the length of the description and number of amenities but those variables may just be correlated to accuracy with little causation. Its worth being aware that a model based on bad feature variables may lead to undesirable user behaviour to try and game the algorithm. For example, users over-filling their listing description in order to improve their accuracy rating. Another approach to predicting the accuracy of the listing would be to use a pre-trained classification model on text contradictions to detect contradictions between the description and the comments.

1.1.3 Cleanliness Rating

There are very few variables in the listing dataset the would have an obvious correlation to cleanliness. Using the review comments from the review dataset would probably be the best approach in this case.

1.1.4 Check-in Rating

Check in rating could be interpreted as the experience and convenience of checking in, or the check-in times that the listing has available, ie. is 24hr check-in available. It can be difficult to predict a variable when the target variable has come from an ambiguous source ie. "garbage in garbage out".

The presence of certain amenities_variables such as: "Host greets you", "Keypad", "Private entrance", "Free parking on premises", or the description including phrases relating to check-in such as "24HR check-in" or "late check-in" could be used to create a useful check-in rating. But it would probably differ greatly from the user generated check-in rating.

1.1.5 Value Rating

Value rating would have been a very interesting rating to try to predict as it is used on many accommodation websites such as booking.com. All the four approaches mentioned at the top of this section could be used to try predict this rating. I think this type of rating would be quite feasible to create but I think it would be limited if it was trained using the user rating of value as the output. Users have a lot less information that a ML model to correctly determine how "good value" a listing is, especially if they don't travel very often or know the area.

2 Implementation

Although it would have been nice to use an example of both a classification and regression approach, I felt that because the desired outcome was a continuous rating, a regression approach would be more appropriate. The problem could be tackled as a classification problem by converting the rating to discrete values of 4.5, 4.6, 4,6... etc.

The two approaches I will be taking to predicting the ratings will be to use Natural Language Processing (NLP) on the reviews to predict the overall rating, and to use regression to predict the communication rating using several feature variables.

For both approaches my aim was to replicate and predict the exiting rating data as close as possible. It could be argued that an approach that created a new rating that did not follow the same skew as the original data might be more informative to the end users. However the potential ambiguity of that rating may frustrate listing owners, and I thought it would be best to stick to the brief of predicting the existing rating. I also decided for both these approaches not to use rating data from one category to predict another as I assume the goal of predicting these ratings is to not have to ask the user give any ratings.

2.1 Approach 1: Predicting Overall Rating using NLP

2.1.1 Feature Extraction

To predict the overall rating I decided to focus on just using the comments from the ratings dataset. Although the overall rating should really take into account as many variables as possible, I thought it would be interesting to see how well the comments alone would work. My hypothesis being that there

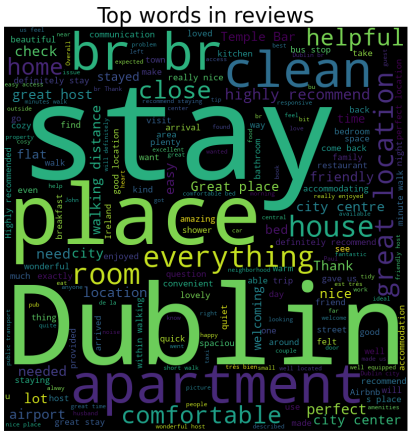

Figure 2: Word Cloud of Comments

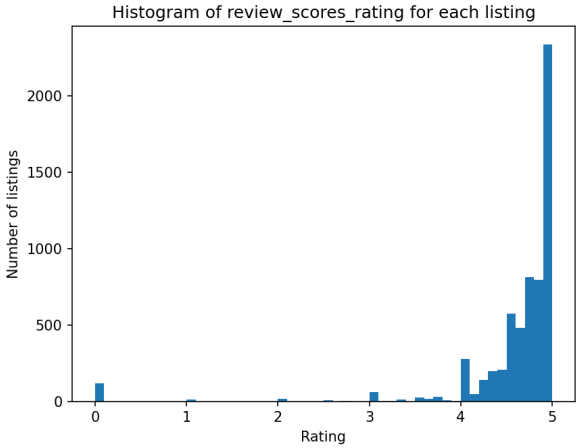

Figure 3: Histogram of Review Scores

should be a strong correlation between the words used in reviews that gave positive comments and high ratings and vise versa.

To clean the data I removed any listings that had either no comments or no overall review score. I joined all the comments for each listing into one string and merged that data-frame of comments with the data-frame of ratings. The raw text data cannot be used as training input to the model as it expects numerical feature vectors. To extract features from the data I used bag of words model which includes uses both tokenization and occurrence counting. To implement these techniques I used the "TfidfVectorizer" class provided by scikit-learn. I set the vectorizer to remove all capital letters, all punctuation and extracted all words of at least two letters, which removes all words with only one letter such as "a" and "I". I preserved some context by also extracting 3-grams of words (Treating three and two adjacent words as one new word). Each extracted word is then given a unique integer index. The vectorizer then performs occurrence counting which simply counts the number of occurrences of each word.

Using the hyper-parameters min_diff and max_diff I removed the words that appeared too frequently or infrequently. The aim is to remove filler words or typos. As can be seen from the word cloud, it would remove words such as stay, place, Dublin and br. We used cross-validation to determine the best values for each. Figure X shows the results of cross validation on both parameters. Based on these results. I decided to use a min_diff value of 1000 and a max_diff value of 5000.

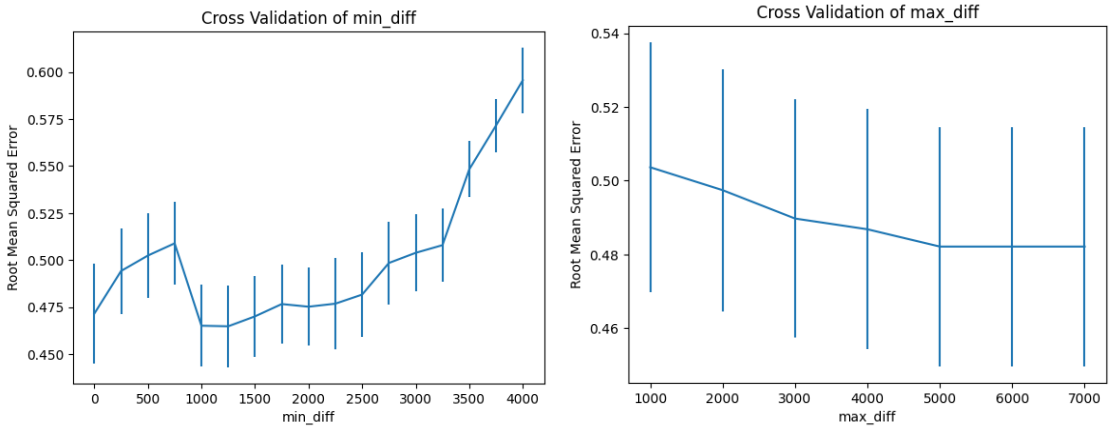

Figure 4: Cross Validation of min_diff and max_diff respectively

2.1.2 Machine Learning Methodology

For the machine learning model I first used a dummy regression model to get a baseline metric. I then tested Linear Regression, Random Forest Regression, Ridge Regression and Lasso Regression. For each of these I adjusted the hyper-parameters so that the model performed optimally. For Ridge regression a very low C value of 1, performed best, whereas for Lasso Regression a much higher C value of 200 was optimal. For Random Forest Regression 20 seemed like an appropriate number of estimators to use and 25 seemed like an appropriate depth to search at.

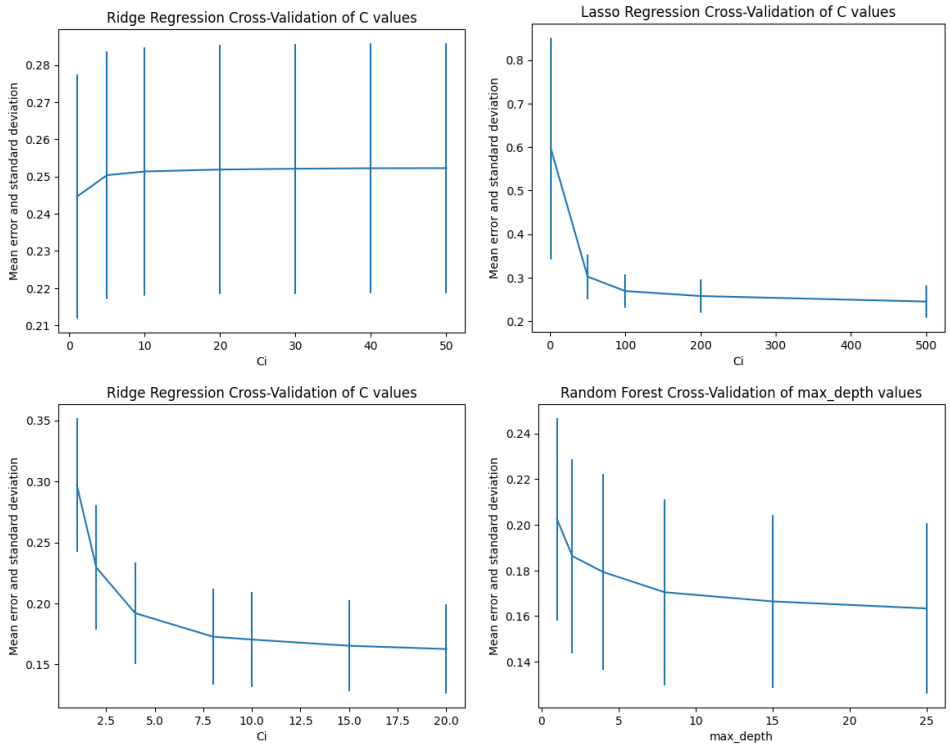

Figure 5: Cross Validation of Hyper-Parameters for Ridge, Lasso and Random Forest Regression

2.1.3 Evaluation / Visualisation

Using the optimal hyper-parameters found in the previous section, table 1 shows how each model performs on the data. The root mean squared (RMSD) and mean squared (MSD) show the variation of the data over the prediction. The R^2 score is the coefficient of determination and describes the variation in the dependent variable (overall rating) that is predicable from the independent variables (The feature variables). All of the models were able to outperform the dummy model which is good.

Various Regression Models					
	RMSE	Std	R^2 Score	T1	T2
Dummy Regression (mean)	0.78	0.044	-0.0004	0.5s	0.1s
Linear Regression	0.68	0.02	0.24	0.2s	0.6s
Ridge (C=1)	0.51	0.013	0.55	0.8s	0.1s
Lasso (C=500)	0.51	0.014	0.57	2.1s	2.5s
Random Forest (estimators=20,depth=25)	0.43	0.046	0.009	34.9s	34.8s

Table 1: A Table of the metrics from each Regression Model.

The Random Forest model performed the best, this is because it is typically better when it comes to data that does not follow a normal curve like in this case. It did require much longer to train and predict which becomes significant when scaled. It also performed quire poorly on the R^2 score. In

conclusion I still don't think a RMSE of 0.43 is good enough to use this model considering the majority of the data falls within the ratings 4-5.

2.2 Approach 2: Predicting Communication Rating using Regression

2.2.1 Feature Extraction

In predicting the communication rating I decided the following features would be appropriate to test: host_response_time, host_response_rate, host_is_superhost, host_listings_count, price and property_type. Many of these variables needed to be converted to numerical values in order to be used in the machine learning model. For example the price needed to be converted from a strings with dollar signs and commas to a floats. Categorical data such as the property_type or is_superhost needed to be converted to a matrix of 1 and 0 values. This was done using the pandas get_dummies function. Once I created a data-frame that consisted of all these features in numerical form, I then split them into test and training data and began testing the data on various models.

2.2.2 Machine Learning Methodology

The models I tested the data on were the same in the first approach Linear Regression, Random Forest Regression, Lasso, Ridge and a Dummy Regression Model. I attempted to optimize the hyper-parameters using Cross-Validation but found that there was little accuracy increase to be gained. The Ridge and Lasso models remained very consistent around 0.35 RMSE, whereas the Random Forest model saw some initial improvement with an increase in estimators. I thought this was due to the skewed data, but after standardizing the data with sklearn's standardization function the results remain similar (Fig 6).

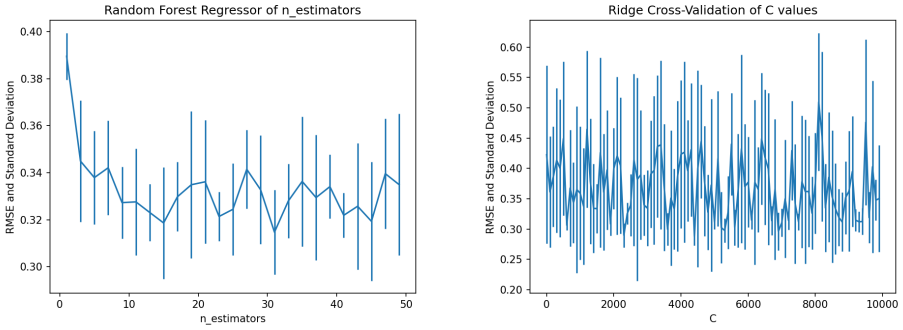

Figure 6: Cross-Validation of Hyper-parameters for Random Forest Regressor (left) and Ridge (right)

2.2.3 Evaluation / Visualisation

For performance evaluation KFolds is used with a 5 times split. This means the data is trained on 80% of the data and then tested on 20% of the data, and this process is repeated 5 times. This allows us to calculate the standard error of our predictions.

Various Regression Models			
	RMSE	Std	R^2 Score
Dummy Regression (mean)	0.2934	0.08	-0.007
Linear Regression	0.2890	0.06	0.012
Ridge (C=100)	0.29	0.08	0.01
Lasso (C=100)	0.32	0.01	-0.6
Random Forest (estimators=20,depth=25)	0.3283	0.08	-0.27

Table 2: A Table of the metrics from each Regression Model.

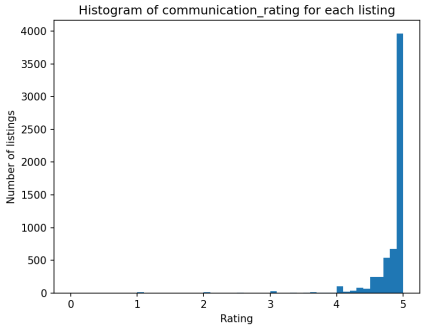

Figure 7: Histogram of communication ratings

The results in this case are more accurate than the overall rating, however all the models performed similarly if not worse than the dummy model, which is useless. After looking at the outcome, I decided to go back and look at the histogram for communication and discovered why the RMSE was much lower. The communication ratings are even more closely compact than the overall ratings (Figure 7).

I am still somewhat surprised the ML models still didn't outperform the dummy model, but I do think if the comments were used alongside these variables there would be a slightly better chance of getting a prediction closer to reality. I think the difference between it and the dummy model would still be negligible, and I would not recommend using this data to predict listing's communication rating.

3 Questions and Answers

3.1 Two examples of situations when logistic regression would give inaccurate predictions.

Logistic Regression gives inaccurate predictions when the data is not linearly separable or when the data is highly imbalanced.

Logistic regression is based on the assumption that the data is linearly separable, meaning that a linear decision boundary can be drawn to separate the different classes in the data. If the data is not linearly separable, then logistic regression may not be able to accurately predict the class of a given sample.

Logistic regression is sensitive to class imbalance, meaning that it may perform poorly if one class is significantly more prevalent than the other. This can lead to biased predictions in favor of the more prevalent class.

3.2 Advantages and disadvantages of a kNN classifier vs an MLP neural net classifier

Advantages of k-Nearest Neighbors (kNN) classifier

- Simple and easy to implement: kNN is a simple and straightforward algorithm that is easy to implement and understand. It does not require any complex mathematical calculations or assumptions about the data.
- No need for training: kNN does not require any training phase, unlike other machine learning algorithms. It simply uses the data as is to make predictions, which can be useful when working with small or poorly labeled datasets.
- Can handle multi-class classification: kNN can easily handle multi-class classification problems, where the data is divided into more than two classes.

Disadvantages of kNN:

- kNN is sensitive to irrelevant features, meaning that it may give more weight to features that are not actually relevant to the classification problem. This can lead to poor performance and inaccurate predictions.
- kNN is can be computationally expensive when working with large datasets because it requires a distance calculation between each sample and every other sample in the dataset
- kNN is sensitive to the scale of the features in the dataset. Therefore, it is important to scale the features before using kNN, which can be an additional step in the preprocessing phase.

Advantages of MLP:

- Can learn non-linear relationships: MLP neural networks are able to learn and model non-linear relationships between the input and output variables, making them suitable for complex classification problems.
- MLP neural networks can handle high-dimensional data, meaning that they can work well with datasets that have a large number of features.
- MLP neural networks can learn multiple layers of abstraction, allowing them to capture complex patterns in the data.

Disadvantages of MLP neural networks:

- MLP neural networks generally require a large amount of data to be effective, which can be a disadvantage if the dataset is small.
- MLP neural networks can be sensitive to the initial values of the weights and biases, which can affect the performance of the model.
- MLP neural networks require careful preprocessing of the data, including scaling and normalization of the features.
- MLP neural networks can be difficult to interpret, as they learn complex patterns in the data that may not be easily understood by humans.

3.3 In k-fold cross-validation a dataset is re-sampled multiple times. What is the idea behind this re-sampling i.e. why does re-sampling allow us to evaluate the generalisation performance of a machine learning model. Why are k = 5 or k = 10 often suggested as good choices?

The idea behind k-fold cross-validation is to split the data-set into k folds, and then train the model k times, each time using a different fold as the test set and the remaining folds as the training set. The performance of the model is then evaluated by averaging the performance metrics across all k folds.

Re-sampling the data allows us to evaluate the generalization performance of the model because it simulates the process of training the model on different data-sets and then testing it on unseen data. This is important because it helps to ensure that the model is not over-fitting to the training data, which would lead to poor generalization to new data.

The value of k is typically chosen to be a small integer, such as 5 or 10, because this allows for a balance between bias and variance in the model's evaluation. A smaller value of k will result in a higher bias, as there is less data available for training, while a larger value of k will result in a higher variance, as the model is being evaluated on more data. A value of k = 5 or k = 10 is often suggested as a good choice because it strikes a balance between these two extremes.

3.4 Discuss how lagged output values can be used to construct features for time series data.

Lagged output values can be used to construct features for time series data by shifting the output values back in time and using them as input features for a model. For example, if the output value at time t is y(t), then the input features for the model at time t could include y(t-1), y(t-2), y(t-3), and so on. This can be useful because it allows the model to take into account the previous values of the output when making a prediction.

4 Project GitHub Repository

Link to GitHub: https://github.com/cormacmadden/Twitter_Censorship_Prediction