Machine Learning Models
Project 1
Report

Application of a classification model to identify types of listings based on listings reviews scores.

Marianna Flores 20180040
Guatemala 16 de febrero, 2021

1. Introduction

Airbnb is focused on offering users rentals with great locations, all sorts of visitor capacities, and varied amenities. Additionally, it is known for having ratings made by visitors who have already been there, therefore making the listings' experiences more reliable. The objective of this project was to create a classification model to divide properties on different categories based on their rating scores. Furthermore, this analysis also found the meaning of different variables such as capacity, number of bathrooms, price, and bedrooms related to the review score of listings.

The data consisted of 74,111 instances of different listings from cities across the US. The variables in the dataset determined characteristics of each listing, there were numerical and string variables. The data was cleaned of missing values and separated in the categories stablished for the review_scores_rating. Three algorithms were applied to classify the data. This consisted of a parametric, which means it analyses for a normal behavior, and 2 non-parametric that classifies based on the general behavior of the data.

2. Data

The dataset used for this analysis consisted of information of Airbnb listings in different cities across the US. This mainly described the components and amenities that each property offers to the app users. There were 28 variables, 18 categorical, 9 numerical and 1 boolean. It consisted of 74,111 instances, representing each listing. As the objective of this project was to categorize the review score of the listings another variable was created, the range. The main variables analyzed were the next:

- Log_price (float64): Property's price altered by Airbnb because of price changes and to protect the property's identity.
- Accommodates (float64): How many people can stay in a property.
- Bathrooms (float64): How many bathrooms are available.
- Cleaning_fee (bool): There is an additional fee or not.
- Number_of_reviews (int64): Number of reviews taken. Had too many missing values and so it was not representative.
- Review_scores_rating (float64): Scores earned by ratings made by previous users.
- Bedrooms (float64): How many bedrooms are available.

- Beds (float64): Number of beds to fit the maximum capacity previously stated.

Most of these variables had missing data so each was considered and according to the use and behavior of each, a different method was applied. The data of review_scores_rating was very inconsistent and skewed (Appendix 1) therefore it was not convenient to use imputation as it altered the general behavior. Consequently, it was decided that the best method to handle the missing data was through listwise elimination as the consequences were minimal. The other numerical variables with missing data (bathrooms, bedrooms, and beds) had few “nas” thus imputation with the mean was convenient and accurate.

The range variable was stablished based on the review_scores_rating, as mentioned before the differences in the distribution made it complicated to stablish several categories, as the data started in reviews with scores of 20 but was concentrated in scores of 90 and higher. The best approach was to apply quantile divisions so that each category would have the same number of instances, therefore it was divided in three quantiles. The quantile division was the next: first quantile (class 0: 20-93), the second (class 1: 93-98), and the last (class 2: 98-100) (Appendix 2).

3. Methods

To understand and apply the most appropriate model for the dataset it was fundamental to determine the type of distribution of the data. The review_scores_ratings did not have a normal distribution; the result of different statistical tests evidenced this behavior and the normality graph showed skewness to the left (Appendix 1). The results of the statistical tests were the next: Shapiro-Wilk test returned 0.701, D’Agostinos’s K^2 test returned 42838, and Anderson-Darling test 3633. On the other hand, the QQ Plot (Appendix 3) represented an abnormal result and a great difficulty to predicting a linear behavior in the data. A parametric algorithm was used, mostly for curiosity and to confirm it was not the best approach. It was decided that the best type of model to classify the data had to be non-parametric. Thus, the data was classified with the KNN algorithm and Random Forest Classifier, both classifiers were applied for multiclass as the ratings were split in 3 categories.

The K-Nearest Neighbors’ classifier is a simple algorithm used mainly for classification analysis and regressions. It is usually very efficient as it is based on feature similarity, which means, it identifies the elements of different categories and calculates the probability of a single point belonging to each. As largest the number of neighbors (K) the

least reliable the results get. For this project, the number of neighbors was fixed on 18, as for the size of the dataset and the 3 variables it was a convenient and dependable amount.

The next algorithm used for this project was the Random Forest Classifier, this is also used for classification and regressions. This particular algorithm is compared to a forest, as the name states, the more instances it has it is more robust. It is pretty easy to use, it selects random samples of the dataset and classifies the elements based on the others' class. Random Forest tends to have trouble overfitting and because of the data sharpness it was necessary to apply an additional parameter, max_depth, to assure the algorithm got real results. This algorithm also analyses which variables are the most important for the model so the unnecessary ones can be removed.

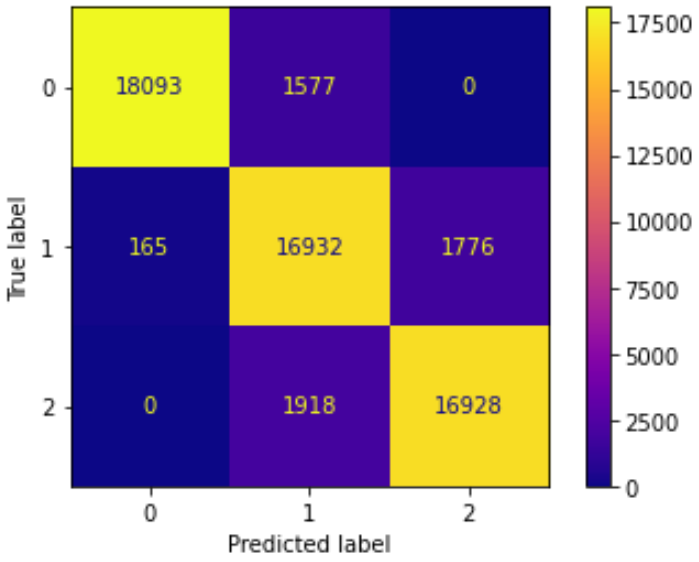
Ultimately, Cross Validation was used to evaluate all the models on their accuracy. This method evaluates the training data by creating folds and splitting to find parameters to later be tested in the test data. It was applied to the models with ten folds to confirm it was working properly. The classes and models were also measured based on the precision, recall and f1-score t. The precision represents the accuracy of the model out of the predicted positive values. The recall represents how many actual positives the model actually predicted as positives. The F1 Score shows the relationship between the precision and the recall.

4. Results and discussion

SDG Classifier

This classifier was only applied to see the result despite not being the most accurate one for the distribution. Still, the result was positive, as it was almost 80% accurate. This shows that even though the data is not normal, the algorithm did predict the model acceptably.

The accuracy presented by this model was around 79%

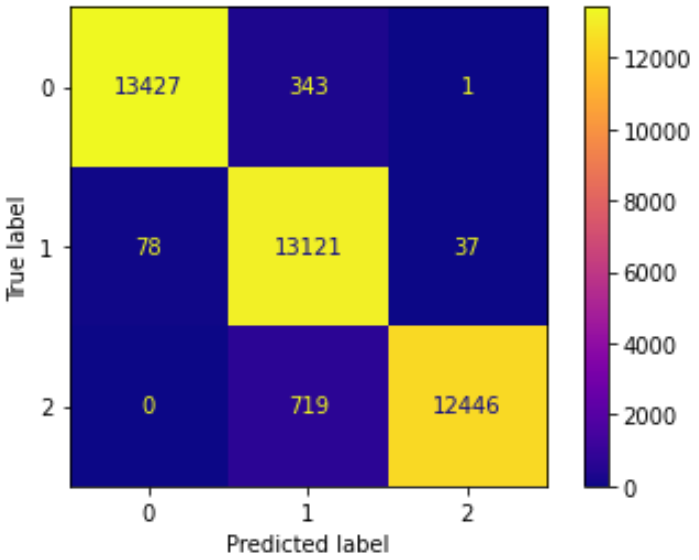


The previous picture showed the prediction ability of the model through the confusion matrix. This model did not mistake any between class 0 and class 2.

KNN Classifier

This classifier got the best results, getting the higher accuracy identifying the classes. The K – Nearest Neighbor algorithm got the best results related to the classification of review_scores_rating parameter made the

The accuracy presented by this model was around 95%



The confusion matrix for this model showed that the class with higher errors was class 1 as the higher prediction mistakes were made here. This probably happened because this class is between the other two and because of the algorithm function it could be the class more vulnerable to confusion.

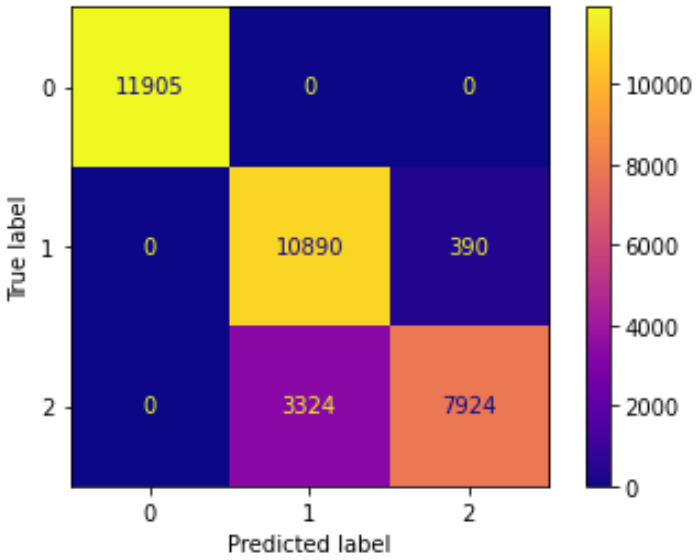
	precision	recall	f1-score	support
0	0.992	0.965	0.978	5899
1	0.907	0.988	0.945	5637
2	0.996	0.936	0.965	5681
accuracy			0.963	17217
macro avg	0.965	0.963	0.963	17217
weighted avg	0.965	0.963	0.963	17217

This model also got the highest scores for recall, precision and the f1 score.

Random Forest Classifier

This classifier got good results; it was not the most accurate one but still managed to classify the classes. It was overfitting, as mentioned before, so a maximum depth was applied. This additional parameter avoids overfittings by leaving the focus just before getting too close.

The accuracy presented by this model was around 89%.



	precision	recall	f1-score	support
0	1.000	1.000	1.000	7765
1	0.764	0.965	0.853	7593
2	0.953	0.701	0.808	7598
accuracy			0.890	22956
macro avg	0.905	0.889	0.887	22956
weighted avg	0.906	0.890	0.888	22956

This model had very high scores for class 0 (probably overfitting as it was “perfect”), the other two classes got good results, but they were worse than with KNN.

Number of reviews	0.634583
Bedrooms	0.259093
Logitude	0.030439
Log_price	0.022275
Latitude	0.018224
Accomodates	0.014647
Cleaning_fee	0.012903
Beds	0.007771
Bathrooms	0.000066

The most representative variables for this model were numbers of reviews and bedrooms according to the Random Forest variable analysis. Yet, when it was manually analyzed for the other models still the best variable to classify review_scores_rating was number_of_reviews (Appendix 6).

5. Conclusions and recomendatios

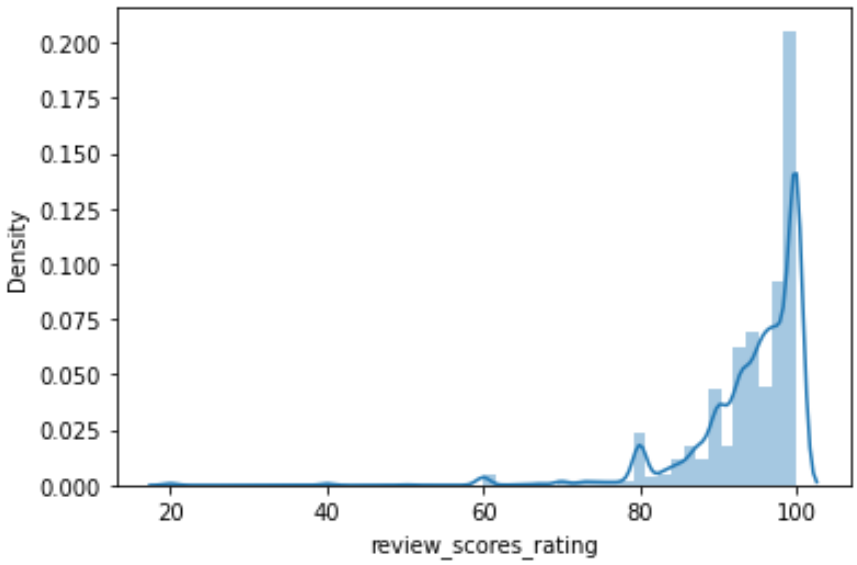
This project focused on classifying different classes of listings based on the review_scores_rating. This process helped identify the most valuable variables and model based on the distribution of the data. The irregularity of the data made it complicated to develop the models as it had to be defined the type of algorithms to apply (non-parametric). The dataset had lots of missing data and each variable had to be analyzed to determine the best approach to correct this issue. The best algorithm for this dataset was k nearest neighbors. This is a non-parametric model and fits the data better as it did not have a normal behavior. This algorithm compares the instances close by and defines which class fits best.

The classification of review_scores_rating depends mostly on the number_of_reviews variable (63.4% of the results). This relationship represents how the rating does depend on the evaluation of multiple happy users. Therefore, negative ratings usually do not have many ratings. This could mean that either there are very few bad experiences (which is pretty unlikely) or that users that had a bad experience do not come back to review the listing. Instead, customers that rate a property as very good do take some time to rate it.

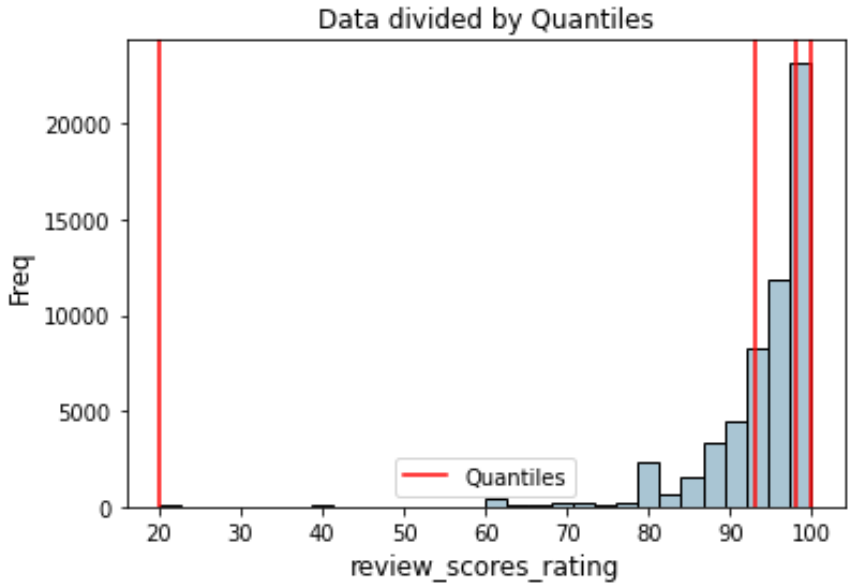
The relationship between review_scores_rating and number_of_scores guarantee new customers that a well rated listing was most probably rated by more than one user. The company is making a good job motivating happy customers to return. Still unhappy customers are not coming back, so probably certain listings are bad, and more users are falling because of the lack of reporting.

It is recommended for further analysis over the complete rating of Airbnb customers to listings and the variable related to this, to get a dataset of polls. As, after a bad experience, users are not coming back to rate the property, probably receiving a questionnaire would get more complete results. With a complete dataset, it could be possible to get a different distribution and evidently different variables related to the rating of a property. Finally, it is also recommended to do the analysis in other countries to get a more varied sample over the different cultural phenomenon and reviews from customers.

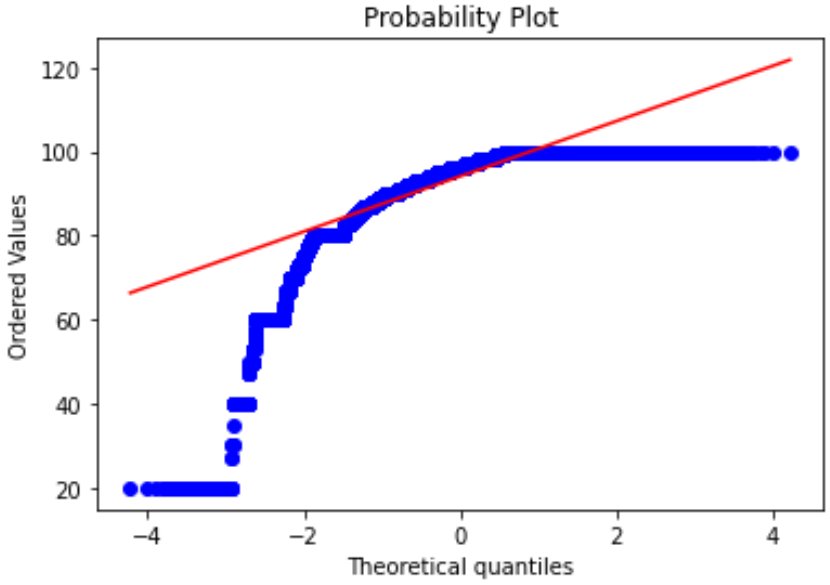
6. Appendix



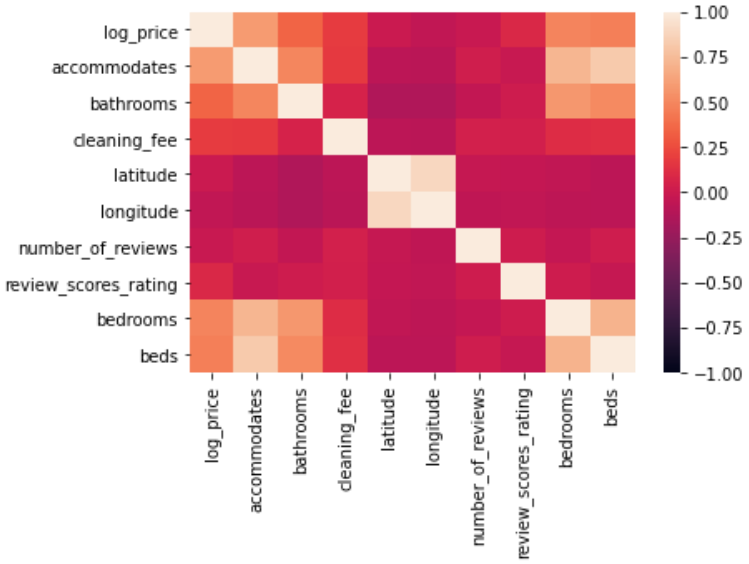
Appendix 1: The last graph illustrates the behavior of the data. It seems not to be normal as it is very skewed to the left.



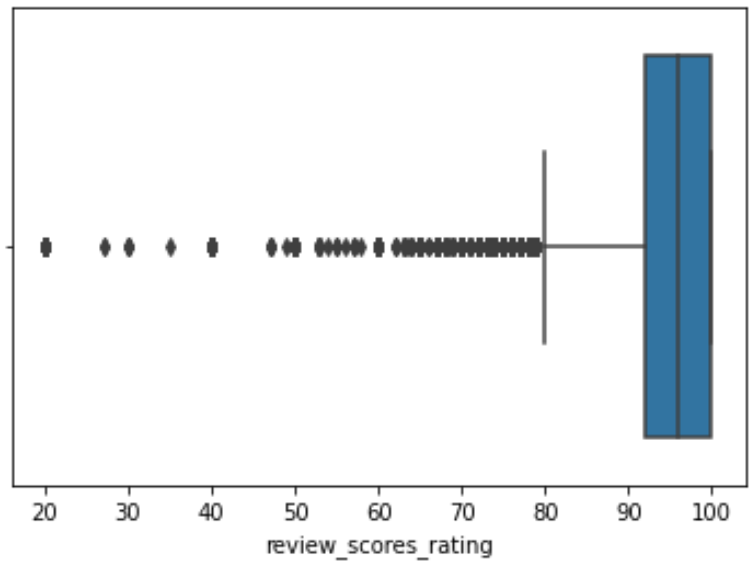
Appendix 2: As the data is not normally distributed most of the instances are located between que properties with scores around 98.



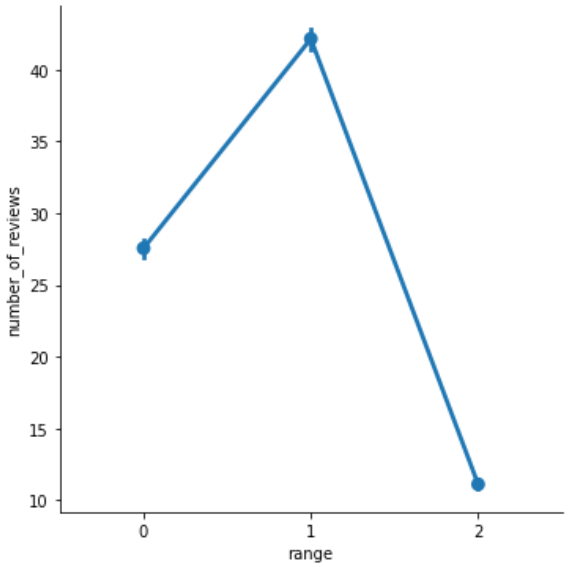
Appendix 3: The probability plot illustrates the lack of probability and precision to predict a behavior.



Appendix 4: Correlation matrix between the variables.



Appendix 5: Boxplot representing the data distribution, mostly on the highest scores.



Appendix 6: Factorplot of relationship between the classes and the number of reviews, that is the most representative variable.

Appendix 7:

Sarkar, D. (. (2019, March 27). Continuous Numeric Data. Retrieved from https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b

Appendix 8:

Pykes, K. (2020, July 28). Handling Missing Data. Retrieved from https://towardsdatascience.com/handling-missing-data-f998715fb73f

Appendix 9:

Random Forests Classifiers in Python. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/random-forests-classifier-python

Appendix 10:

KNN Classification using Scikit-learn. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Appendix 11:

Shung, K. P. (2020, April 10). Accuracy, Precision, Recall or F1? Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9