Project Report
CS 396: Data Science Pipeline
Team 10: Vittorio Iocco
06 December 2021

Introduction and Motivation

The goal of this project is to create an Airbnb suggestion tool that suggests the listing price for a new Airbnb rental location. This is a useful tool for new Airbnb hosts who do not understand the market. Without tools such as this, new hosts run the risk of missing profit either by undervaluing their property or overvaluing their property and causing vacancy. Our analysis will be for all of Ireland.

Dataset

The dataset for this analysis is the listings.csv.gz file located under the Ireland section of the Inside Airbnb database located at http://insideairbnb.com/get-the-data.html. This file contains descriptive information about the listing and the host in tabular format. There are 81 initial features from the dataset. Some of these features are features that traditionally are correlated with price, such as the number of bedrooms, the number of bathrooms, the property type, the property location, and property amenities. Others are "dummy" metadata like the listing URL. Using the former variables, the team can make suggested prices for users, potentially adjusting the price for the percent of time the user wants their location occupied.

In terms of the size of the dataset, before any cleaning or modification, the data contains 26,176 unique property listings. After all of the cleaning and modification discussed in future sections there are 25,261 entries meaning that the model uses 96.50% of the initial data.

EDA

The goal of the team's Exploratory Data Analysis is to verify the feasibility of generating a suggestion tool that suggests the listing price for new Airbnb rentals in Ireland using the data. There are three major aspects of the data the team analyzes to make this decision: the distribution of the daily prices of listings, the distribution of location of the listings throughout Ireland, and the staticity of data collection of the listings.

Price Distribution

The team aims to create a model for the price of a unit; however, any such model must consider

the distribution of daily prices since models will in general be more accurate at dense areas of the distribution and less accurate for sparse areas of the daily price distribution. With this goal in mind, the team created three boxplots. The leftmost boxplot uses all the price data, the middle boxplot uses only daily prices under $20,000 and the rightmost boxplot uses only daily prices under $1,000. It is evident that the vast majority of the listings are priced for under $1,000 daily (Figure 1). Outlier analysis on the data shows that prices over $403 dollars are extreme outliers. Thus, the model should only be used to predict prices in the range [$0,$403]. This interval contains 96.75% of the data, verifying this is a beneficial decision.

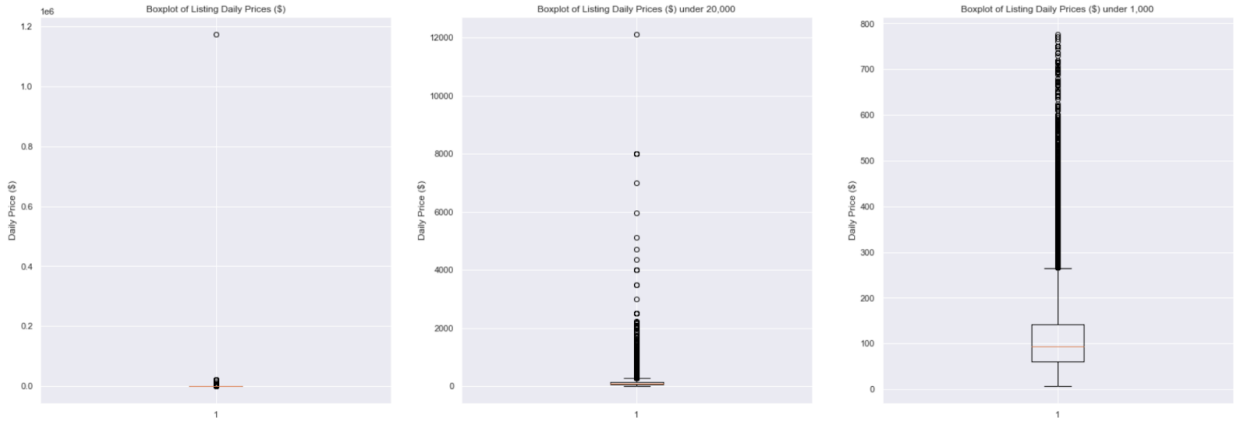

Figure 1: Boxplots of The Daily Prices of Airbnb Listings

Data Staticity

Another important aspect to consider for the feasibility of the project is whether or not the listing information was scraped over a small time period. Since prices fluctuate based on the time of year (e.g. high season vs low season) it must be ensured that the listings used in this analysis were scraped in a short time period. As detailed in Figure 2, all listings were scraped within two weeks of each other. Thus, the assumption that the data is static is correct and the model is feasible from this perspective.

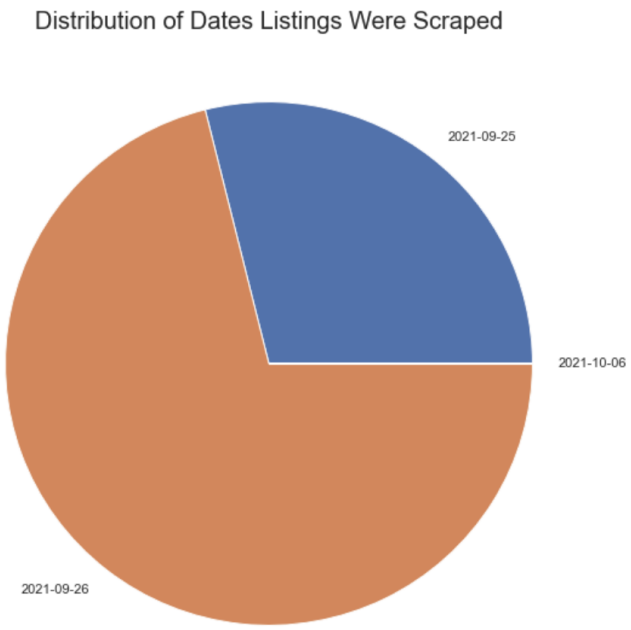

Figure 2: Distribution of Scrape Dates For Airbnb Listings

Location Distribution

The final feasibility consideration is the location distribution of properties. The team initially set out to generalize the model over all of Ireland. This would be a poor generalization if the distribution of properties were highly skewed towards some areas of the country with little information for other areas of the country. While the distribution of listings does tend to be skewed towards touristy areas such as the remote west and south of Ireland as well as Dublin, the distribution is otherwise pretty even and there appear to be a decent amount of listings throughout the country (Figure 3). Thus, it is feasible to assume the model generalizes across the country.

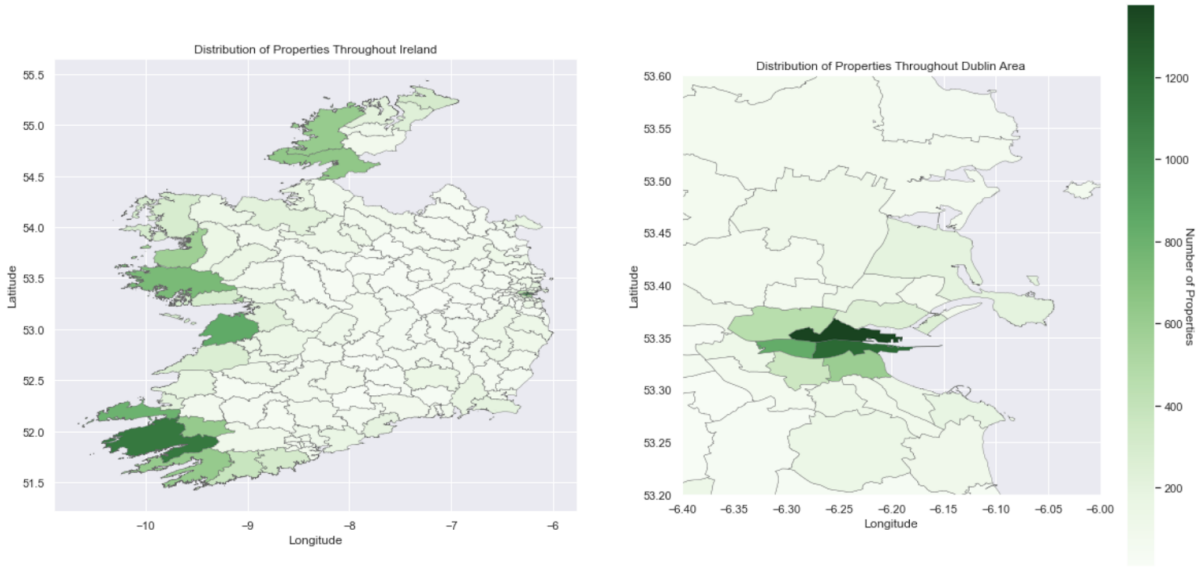

Figure 3: Choropleth of Listing Density By Irish Regions

The data passes all of the major feasibility tests and thus it is determined that the model is feasible to generate given the data.

Data Cleaning & Feature Selection

Data Cleaning

Once it was determined that the model would be feasible, the team began to clean the data and engage in feature selection/engineering. The initial cleaning involved entity resolution of place names. Some of the common resolutions were:

1. Missing 'County' Descriptor: Kildare, Kildare, Ireland vs. Kildare, County Kildare, Ireland
2. Abbreviations: Co. Mayo vs. County Mayo
3. Missing Punctuation: Cork, County Cork, Ireland vs. Cork, County Cork Ireland

Additionally, data type cleaning was performed. Specifically, the team performed:

1. Canonization of Bathroom Information (bathrooms_text)
    a. Creation of a column tracking whether the bathroom is shared or private (bathroom_type) and retaining the number of bathrooms in bathrooms_text
2. Standardization of Entries:
    a. Removing $ from price column entries
3. Conversion of column types from string to boolean and int/float:
    a. To Numeric: price, host_id, scrape_id, id, region_id, region_parent_id, bedrooms, beds, accommodates, latitude, longitude, host_total_listings_count, review_scores_value, review_scores_location, review_scores_checkin, review_scores_cleanliness, review_scores_rating, review_scores_accuracy

b. To Boolean: host_identity_verified, host_has_profile_pic, host_is_superhost

All of this cleaning was pre-processing aimed at making the data more accurate and amenable to manipulation. This initial cleaning was done on OpenRefine. The full record of this cleaning is available in JSON format in src/listings_complex_cleaning.json and can be reverse engineered in OpenRefine. After this initial round of cleaning, the team embarked on feature selection.

Feature Selection

As previously stated, the listings.csv.gz has 81 distinct features out-of-the-box. However, some initial analysis of these columns revealed that 50 of these columns were not useful to the problem. Each of these 50 features met one of the following:

1. It contains duplicate information
    a. e.g. region_name, region_parent_name, and region_parent_parent_name are redundant versions of region_id, region_parent_id, and region_parent_parent_id
2. It represents information not available to a new listing
    a. e.g. reviews_per_month, availability_30
3. It is bogus or not related to the individual property itself
    a. e.g. calculated_host_listings_count, id, host_picture_url
4. It has a high amount of missing information
    a. e.g. host_neighbourhood, host_about, neighbourhood

Figure 4 shows the features eliminated because of a high number of null entries. Figure 5 shows the features eliminated due to being duplicate information and which column they duplicate.

Feature	Proportion Null
license	1.00
bathrooms	1.00
calendar_updated	1.00
requires_license	1.00
host_neighbourhood	0.54
host_about	0.53
neighborhood_overview	0.39
neighbourhood	0.39

Figure 4: Features Eliminated by High Proportion of Null Values

Feature	Duplicate of _?
property_type	room_type
region_name	region_id
region_parent_name	region_parent_id
region_parent_parent_name	region_parent_parent_id

Figure 5: Features Eliminated by Duplication

The rest of the features that were eliminated here were eliminated due to irrelevance or lack of availability for new Airbnb listings. This is a long list, so for a full list of the out-of-the-box features and the features eliminated in each category see the code in src/ML.html.

At the end of this feature selection, 30 features remained. They are divided into numerical and categorical features and listed in Figure 6.

Numeric Columns
bathrooms_text
latitude
minimum_nights
maximum_maximum_nights
minimum_nights_avg_ntm
host_acceptance_rate
accommodates
longitude
minimum_minimum_nights
minimum_maximum_nights
region_id
maximum_nights
beds
bedrooms
maximum_minimum_nights
host_response_rate
region_parent_id
maximum_nights_avg_ntm

Categorical Columns
host_identity_verified
instant_bookable
room_type
bathroom_type
neighborhood_overview
region_parent_parent_id
name
host_has_profile_pic
description
host_response_time
amenities
host_is_superhost

Figure 6: List of Base Features Used in Model

Please note that these are not the final features used in the model, as new features were created from these using feature engineering. However, this is conceptually the list of features used in the model.

Feature Engineering

Numerical Features

The feature engineering for the features in Figure 6 was pretty simple. Since the data was already numeric, the only hiccup was the null values in each feature. So, the numeric features with a large number of nulls were dropped (host_response_rate and host_acceptance_rate). For the other features (bedrooms, beds, and bathrooms_text) with a low number of null values, I used Mean Value Amputation on the Mean of the remaining entries (Figure 7).

Feature	Number of Nulls	Proportion of Nulls
host_response_rate	8376	0.32
host_acceptance_rate	7414	0.28
bedrooms	760	0.03
beds	142	0.01
bathrooms_text	52	0.00
region_id	0	0.00
region_parent_id	0	0.00
maximum_minimum_nights	0	0.00
maximum_nights	0	0.00
minimum_maximum_nights	0	0.00
latitude	0	0.00
minimum_minimum_nights	0	0.00
longitude	0	0.00
accommodates	0	0.00
minimum_nights_avg_ntm	0	0.00
maximum_maximum_nights	0	0.00
minimum_nights	0	0.00
maximum_nights_avg_ntm	0	0.00

Figure 7: Numerical Features and Null Counts and Proportions

Categorical Features

There were four major categories of categorical feature numerization:

1. Numerical Features ‘In Disguise’
    a. region_parent_parent_id
2. Plain Text Features
    a. name
    b. neighborhood_overview

    c. description
3. One-Hot Encodable Categorical Features
    a. instant_bookable
    b. bathroom_type
    c. host_response_time
    d. room_type
    e. host_has_profile_pic
    f. host_is_superhost
    g. host_identity_verified
4. Remaining Non-One-Hot Encodable Categorical Features
    a. amenities

The region_parent_parent_id feature was simply a string of the from ‘IE###’ and so removing ‘IE’ converted it into an integer. For the plain text features, the team performed sentiment analysis and converted each respective column from the text into the polarity of the text. Once these two steps were complete, the team decided which columns were one-hot-encodable by looking at the number of unique values of each remaining categorical feature (Figure 8). Every remaining categorical feature except for amenities had a low number of unique values and was thus one-hot encoded.

column name	number of unique values
amenities	23695
host_response_time	5
room_type	4
host_is_superhost	3
host_identity_verified	3
bathroom_type	3
host_has_profile_pic	3
instant_bookable	2

Figure 8: Number of Unique Values of One-Hot Encodable Columns

The amenities feature was the most complicated to numerize. The problem was that there were

too many unique lists (23,695) and unique amenities (1,648) to one-hot encode. So, the following technique was followed:

1. Find unique set of amenities
2. Run KNN algorithm over this set with n_clusters = [10, 50, 100, 200]
3. Replace list of amenities (list of strings) with list of clusters (list of numbers)
4. One-Hot Encode
5. Save a version of the data for ever value of n_clusters

At the end of feature engineering, the 30 numerical and categorical features turned into 50 numerical features through the methods discussed above. Figure 10 provides a list of the names of these features. Please keep in mind that conceptually the initial 30 features are being used. Additionally, at the end of feature engineering, there were 25,261 samples and thus after all the feature engineering and data cleaning, 96.50% of the data was retained.

```
host_response_time_a few days or more
host_response_time_within a day
host_response_time_within a few hours
host_response_time_within an hour
instant_bookable_f
instant_bookable_t
latitude
longitude
maximum_maximum_nights
maximum_minimum_nights
maximum_nights
maximum_nights_avg_ntm
minimum_maximum_nights
minimum_minimum_nights
minimum_nights
minimum_nights_avg_ntm
name
neighborhood_overview
region_id
region_parent_id
region_parent_parent_id
room_type_Entire home/apt
room_type_Hotel room
room_type_Private room
room_type_Shared room
accommodates
bathroom_type_Private
bathroom_type_Shared
bathroom_type_Unknown
bedrooms
contains_amenity_cluster_0
contains_amenity_cluster_1
contains_amenity_cluster_2
contains_amenity_cluster_3
contains_amenity_cluster_4
contains_amenity_cluster_5
contains_amenity_cluster_6
contains_amenity_cluster_7
contains_amenity_cluster_8
contains_amenity_cluster_9
description
host_acceptance_rate
host_has_profile_pic_0
host_has_profile_pic_1
host_identity_verified_0
host_identity_verified_1
host_is_superhost_0
host_is_superhost_1
host_response_rate
host_response_time_Unknown
```

Figure 10: List of Final Features After Features Engineering

Data Modeling

At this point, the features were finalized and I began testing some initial models. Since the goal of the project is to predict the price of a new listing, the team used regression and the coefficient of determination (r^2) metric. In order to prevent overfitting, the team used 4 fold cross validation in all of the model evaluations. The initial modeling tested three regressors -- the Gradient Boosting Regressor, the Random Forest Regressor, and the KNN Regressor -- against the four versions of data: amenities one-hot encoding generated from KNN with 10, 50, 100, and 200 clusters. The Gradient Boosting Regressor performed highest and so the team selected the Gradient Boosting Regressor. While the highest performance was with 50 clusters, the team

chose the 10 clusters because adding the 40 additional one-hot-encoded features only increased the r^2 performance by around .01 (Figure 11).

	10	50	100	200
Gradient Boosting Regressor	0.528965	0.531649	0.528893	0.528362
Random Forest Regressor	0.520582	0.525988	0.521807	0.526131
KNN Regressor	0.324982	0.325273	0.330194	0.329080

Figure 11: Initial Performance of Select Models

From here, grid search over the Gradient Boosting Regressor with parameters of learning_rate = [.05, .075, .1, .3], n_estimators = [100, 150, 200], and other parameters set to default. The best learning_rate, on average, was .1. Within models with a learning_rate=.1, the highest performing model used 200 estimators. However, from 100 to 200 estimators, the r^2 ratio only increased by around .02. This suggests overfitting and thus 100 was selected for the number of estimators.

Summary of findings/Potential implications

Using these Gradient Boosting Regressor with learning_rate=.1, n_estimators=100, and the other parameters set to default values. The number of samples was 25,261 which is 96.50% of the original number of samples. The testing r^2 value over 4 fold cross validation was .5529. This is not a super high value, but considering that the MSE was 2064.2814 and thus the "average distance" to the correct price from the prediction is around $40, this performance is not terrible (Figure 12). Additionally, this lack of performance can potentially be explained by noticing that features such as the aesthetic and quality of furniture of a building are not included in the data yet are often important in pricing. Thus, a future model could perform sentiment analysis on the image data and add this as a feature to improve accuracy.

```
Training MSE: 1585.0463
Testing MSE: 2064.2814

Training r2: 0.6575
Testing r2: 0.5529
```

Figure 12: r^2 Performance of Gradient Boosting Regressor(learning_rate=.1, n_estimators=100) over 4 Fold Cross Validation