COPERTINA QUI

Sommario

**Nessuna voce di sommario trovata.**

# Introduction

## Goals

The aim of this paper is to explain the choices and the strategies we adopted on the project and development of AirBnb Price Estimator, whose aim is to help owners to decide the most correct price for their B ‘n‘ B.

In order to accomplish it, we started from web-scraped data, we performed all the preprocessing needed for having a suitable dataset and then we built several classifiers, using different strategies, in order to determine the one that predicts best the class attribute. All these classifiers have been tested using more than one method and the analysis of the results guided us in the choice of the best classifier.

Since the class attribute is numeric, we had two possible choices:

* Discretize the attribute, choosing the most appropriate algorithm
* Keep it numeric, using regression algorithms for the classification purposes

The first approach is surely easier, but it would not be as helpful as the second one for our application purposes: suggesting a precise value to a owner will give him/her a more accurate advice rather than a range.

The regression model that generalizes best the class feature has then been chosen as the “heart” of AirBnB Price Estimator: the application asks users to input the required fields that correspond to the attributes needed by the classificatory. On these fields bases, it simply outputs to the user the suggested price for night.

## Initial Dataset

The starting dataset is composed by web-scraped data collected in a csv file. The scraping has been performed the 11th December 2020. The web source is the airbnb.us domain and concerns all the registered B ‘n’ Bs on the Metropolitan Area of New York City (NY). The scraped data has been made available by third parties.

The initial dataset is composed by 36923 instances and 74 columns. In order not to confuse the reader with useless information, the attribute list is not here reported, however on the following chapter regarding preprocessing we justify in detail the actions performed on the data. For now, the most important things to know is that:

* The dataset is composed by various mixed features regarding the estate, the host and the geographical position
* The source file has been published without respecting exactly the csv format, thus some frameworks and csv handlers are not able to parse it
* The initial data is very dirty: there are missing values, redundant attributes, lists of strings embedded in a single column, pointless features and so on. For all of these problems, a suitable solution has been provided and it is fully reported on the next chapter.

# Preprocessing

## What can be preprocessed now?

Since we have to deal with classification problems, before preprocessing data we must be sure not to apply supervised filters on the whole dataset. If we need supervised filters, we must split the dataset in training set and test set before applying them. However in our case there is no need for a supervised filters but the attribute selection, so we performed all the unsupervised preprocessing operations at the beginning, postponing the attribute selection after the training/test split.

## Data cleaning and reduction

Since the Weka framework was not able to parse correctly the source file, for the following operations we used Microsoft Excel and Apple Pages

### Removing irrelevant attributes

The first preprocessing operation is the attribute reduction, indeed we decided to remove all the features that are not domain-specific, nor useful for classification purposes. Among them we deleted:

* IDs like *Bnb ID, scrape\_id, host\_id*
* URLs like *listing\_url, picture\_url, host\_url, host\_thumbnail\_url, host\_picture\_url*
* Information on the scraping like *last\_scraped\_date, calendar\_last\_scraped*
* Useless information on the rent for classification purposes like *name, description, first\_review\_date*
* Useless information about the host like *host\_name, host\_location*(it is not the place in which the rent is, but where the host lives), *host\_about, host\_has\_profile\_pic*

### Removing redundant attributes

The next step was the reduction of attributes explicitly redundant. Not for all of these we did compute the  2 test because of the explicit correlation between the features

* *host\_neighbourhood* and *neighbourhood* w.r.t. *neighbourhood\_cleansed* are explicit redundancies; *neighbourhood\_group\_cleansed* has been demonstrated to be very highly correlated with *neighbourhood\_cleansed* (P{independence}<0.05)
* *host\_total\_listings\_count* completely equal to *host\_listing\_count*
* *host verification* w.r.t *host\_identity\_verified*
* *minimum\_minimum\_nights, minimum\_maximum\_nights, maximum\_minimum\_night,* *maximum\_maximum­\_nights, minimum\_nights\_avg* and *maximum\_nights\_avg* are redundancies of the attributes *minimum\_nights* and *maximum\_nights*
* *review\_scores\_accuracy, review\_scores\_cleanliness, review\_scores\_checkin, review\_scores\_communications, review\_scores\_location, review\_scores\_value* are rounded values that have been more precisely combined in another already-existing attribute that is *review\_scores\_rating*
* *calculated\_host\_listings\_count(+category)* are redundant attributes of *listings\_count* valid only for their respective category

### Other removed features

Eventually, the other columns that have been discarded are:

* Attributes that strongly depend from the instant in which the scraping has been performed like *has\_availability\_now, availability\_30, availability\_60, availability\_90, number\_of\_reviews\_ltm, number\_of\_reviews\_l30*
* Empty attributes like *license*
* Others, like *latitude* and *longitude* that are not more useful than the easier data on the neighborhood

## Dealing with missing fields

Once the previous steps were performed, we achieved a Weka-convertible CSV file. Thus, the following steps have been implemented using the Java Weka API.

The original dataset contains several missing values sparse in more than one attribute. The amount of missing values is enough relevant to discourage the instance deletion, although they are pretty easy to handle, indeed:

* The majority of them are on numeric attributes characterized by low variance, like the *response\_rate* (σ=26.17 on a 0-100 interval) or the *review\_score\_rating* (σ=9.52 on a 0-100 interval). In this cases, replacing the missing value with the mean does not introduces big error rates.
* Some missing values can be easily inferred with the simple analysis by the domain expert: for example the attribute *bedrooms* contains missing values, but only when the corresponding *beds* value is 1: it’s reasonable that the missing value for *bedrooms* is 1 as well.
* Supervised approaches for missing values could have guaranteed more precise results, but since we are exploiting a regression problem and we didn’t split training and test sets yet, using a supervised filter here was an error.

## Data integration and transformation

* + 1. Attribute formats transformation *(+screenshot)*
    2. Dealing with lists *(+codice)*

## Preprocessing implementation

As already mentioned before, the preliminary operations needed to make the csv readable by Weka have been performed using spreadsheet software. The resulting cleansed file still needs some modifications, i.e. :

* As reported in par. 2.3, missing values need to be managed
* As widely discussed in par. 2.4, the *amenities* must be converted in a Hot Vector
* On the *price* attribute, we get rid of the $ sign and we convert it into numeric
* The *bathroom* attribute needs its format to be changed

After the loading of the CSV file, in order to speed up the operations we parallelized them using 4 different threads: the dataset is vertically split and every thread works only on its related partition, using Java Weka API or working directly on the text according to what was the fastest approach in every single scenario; every execution flow, before ending, writes results in a CSV file. The main thread spawns a thread for each of the tasks listed before, and then waits the termination of all of them before merging the results in a single file.

In addition to the benefits of parallelism, this approach promotes the separation of concerns of the tasks: the main method defines the preprocessing pipeline, but the actual operations on data are performed by separated and independent components. All the classes implemented at this point are collected in the *com.unipi.dmaml.airbnbpriceestimator.preprocessing* package.

1. Classification
   1. Strategies *(come dividiamo train e test, quali classificatori scegliamo, quali algoritmi di attribute selection*)
      1. Train and test splitting
      2. Chosen classifiers
   2. Building classification models *(+codice)*
      1. Procedure
      2. Implementation
   3. Performance evaluation and effects of attribute selection *(+codice, screen, procedure di valutazione ed un sacco di roba)*
   4. Conclusions
2. AirBnB Price Estimator
   1. Functional Requirements
   2. *(Forse altri capitoli se avremo cose interessanti da mostrare)*