Data cleaning in Python: some examples from cleaning Airbnb data

How to deal with messy location and text data in Python

Laura Lewis Follow
May 16, 2019 · 7 min read ★

Source: airbnbeazy.com

Project background

Airbnb fascinates me. I previously worked for a year and a half at an Airbnb property management company, as head of the team responsible for pricing, revenue and analysis. One thing I find particularly interesting is how to figure out what price to charge for a listing on the site. Although 'it's a two bedroom in Manchester' will get you

reasonably far, there are actually a huge number of factors that can influence a listing's price.

As part of a bigger project on using deep learning to predict Airbnb prices, I found myself thrown back into the murky world of property data. Geospatial data can be very complex and messy — and user-entered geospatial data doubly so. This post will explain how I went about sourcing and preparing the data for my project, including some thoughts on dealing with UK geographic data (it's surprisingly complicated) and extracting relevant information from long text strings.

The dataset

The dataset used for this project comes from Insideairbnb.com, an anti-Airbnb lobby group that scrapes Airbnb listings, reviews and calendar data from multiple cities around the world. The dataset was scraped on 9 April 2019 and contains information on all London Airbnb listings that were live on the site on that date (about 80,000).

Cleaning and preparing the data

For the full exciting details (disclosure: level of excitement may vary) of data cleaning, feel free to check out my GitHub repo. In the interests of brevity, I'll discuss three particular areas of data pre-processing that might be of interest.

Features that weren't included (but I would have liked to include)

The original dataset contained 106 features, including quite a few text columns of all the different description fields that you can fill in for an Airbnb listing. Due to time constraints I did not do any natural language processing (NLP) in this model, so all these features were dropped. However, an interesting avenue of future development for this model would be to augment it with NLP — perhaps for sentiment analysis, or looking for keywords, or some sort of fancy Word2Vec type situation that looks for similar listing descriptions and uses this to help guess price based on similar listings.

Another potential direction of future work could include reviews. Insideairbnb.com also scrapes reviews, which can be matched to listings with their listing IDs. Although most guests tend to give most listings high ratings, more nuanced ratings could perhaps be derived from the reviews themselves.

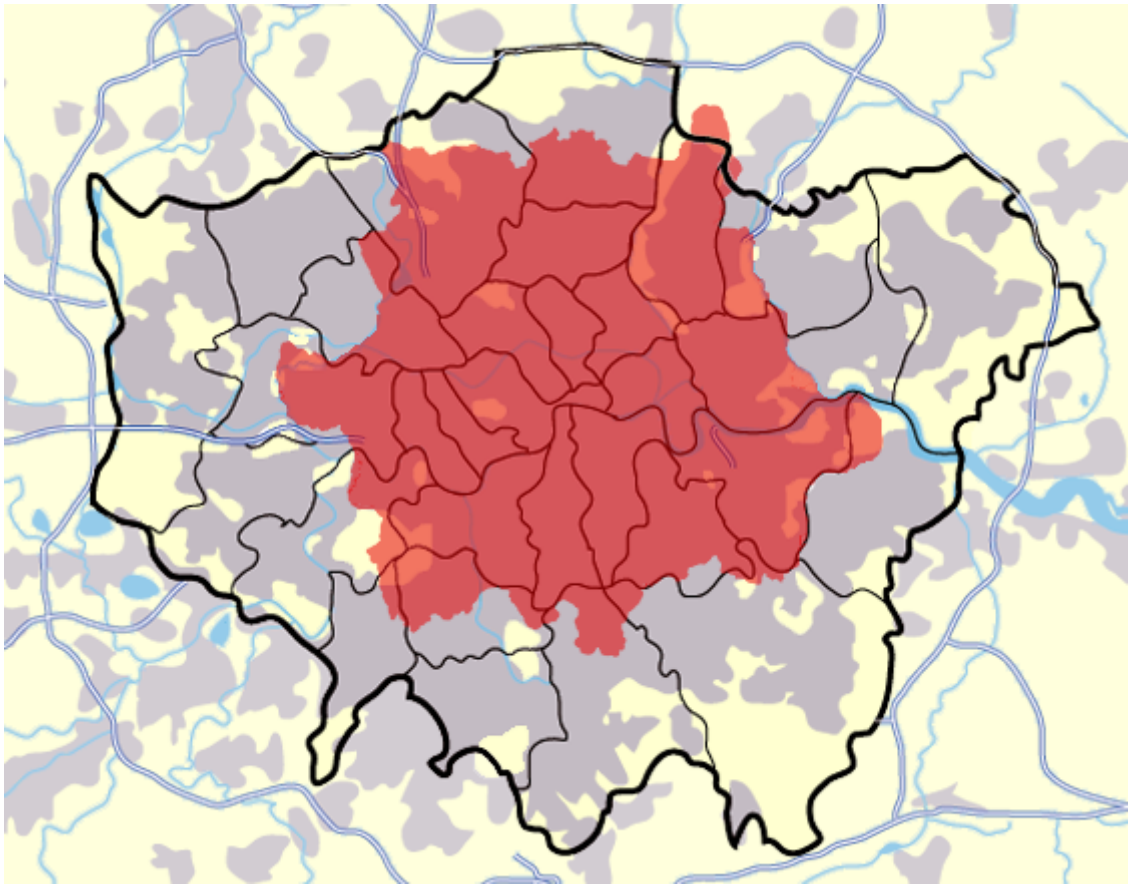
Dealing with London geography (TLDR: London was not mapped with data scientists in mind)

Postcodes in the UK are complicated and messy. They can be various lengths, and consist of letters and numbers in various orders. The first half of a postcode is called the outcode or postcode district, and refers to areas as shown here:

London postcode districts. Source: https://en.wikipedia.org/wiki/London_postal_district

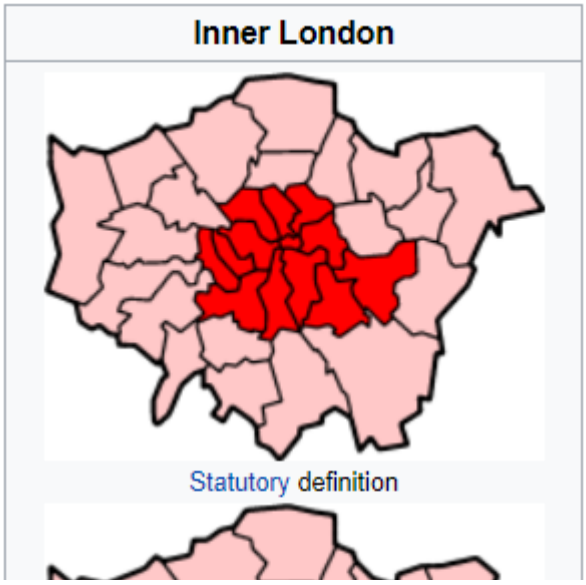
Just to make things more complicated, the main geographic division of London is into 32 boroughs plus the City of London (technically a Corporation rather than a borough due

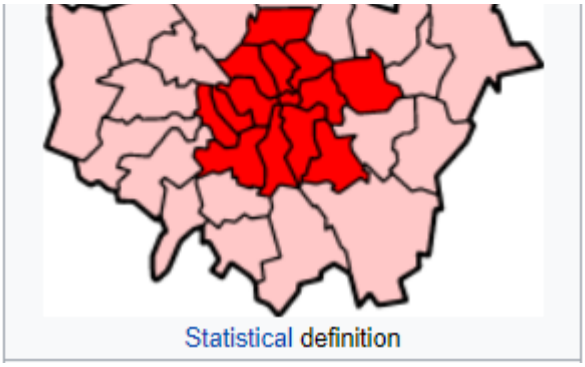
to some quirks of 12th Century English history), which do not align with postcode districts (because that would be too easy, right?):



London postcode districts (red), layered over London boroughs (black lines). Source: https://en.wikipedia.org/wiki/London_postal_district

There also aren't any easy ways of classifying London areas on a less granular level. In fact, there is not even agreement on what counts as 'inner London':





What even is London? Source: https://en.wikipedia.org/wiki/Inner_London

And to make matters even worse, it turns out Airbnb allows hosts to enter postcodes in a free text entry box, precluding any easy separation of parts of postcodes, and allowing hosts to write all kinds of nonsense (my favourite is just the word ‘no’).

In the end, after discarding a bunch of regex experimentation with postcodes, I settled on using borough as the unit of geography. Location is very important for Airbnb listings, and so I was not entirely happy about having to use borough. It is not on a particularly fine-grained level, and does not always express well whether a property is in central London or out in the sticks — which makes a huge difference to price. For example, the famous Shard skyscraper is in Southwark, but so is Dulwich , where the tube doesn’t even reach (disclaimer: Dulwich is actually lovely, but is probably less well known to tourists in London).

Both in Southwark, but possibly with different Airbnb prices? Left: the Shard (source: https://www.visitbritainshop.com/world/the-view-from-the-shard/). Right: Dulwich high street (source: https://de.wikipedia.org/wiki/Dulwich_(London)).

I did also experiment with using latitude and longitude instead of borough in order to get more fine-grained results — but as a future blog post will show, it was not entirely successful.

Amenities (so very many amenities)

In the dataset from Insiderairbnb.com, amenities were stored as one big block of text—here's one example:

```
# Example of amenities listed
df.amenities[:1].values
```

```
array(['{TV,"Cable TV",Wifi,Kitchen,"Paid parking off premises","Smoking allowed","Free street parking","Buzzer/wireless int
ercom",Heating,"Family/kid friendly",Washer,Dryer,"Smoke detector","Carbon monoxide detector","Fire extinguisher",Essential
s,Shampoo,"Lock on bedroom door",Hangers,"Hair dryer",Iron,"Laptop friendly workspace","Outlet covers",Bathtub,"Children's b
ooks and toys","Babysitter recommendations",Crib,"Pack 'n Play/travel crib","Room-darkening shades","Children's dinnerwar
e","Hot water","Bed linens","Extra pillows and blankets","Ethernet connection","Coffee maker",Refrigerator,"Dishes and silve
rware","Cooking basics",Oven,Stove,"Patio or balcony","Luggage dropoff allowed","Long term stays allowed","Step-free acces
s","Wide doorway","Wide clearance to bed","Accessible-height bed","Step-free access","Wide doorway","Bathtub with bath chai
r","Accessible-height toilet","Host greets you","Handheld shower head","Roll-in shower"}'],
      dtype=object)
```

In order to figure out what the various options were and which listings had them, I first made a giant string of all the amenities values, tidied it up a bit, split out the individual amenities separated by commas, and created a set of the resultant list (fortunately the dataset was small enough to allow this, but I would have needed a more efficient way to do this with a much larger dataset):

```
amenities_list = list(df.amenities)
amenities_list_string = " ".join(amenities_list)
amenities_list_string = amenities_list_string.replace('{', '')
amenities_list_string = amenities_list_string.replace('}', ',')
amenities_list_string = amenities_list_string.replace('"', '')
amenities_set = [x.strip() for x in amenities_list_string.split(',')]
amenities_set = set(amenities_set)
amenities_set
```

amenities-set.py hosted with ❤ by GitHub view raw

And here's a list of all the amenities it is possible to have:

```
'24-hour check-in',
'Accessible-height bed',
'Accessible-height toilet',
'Air conditioning',
'Air purifier',
'Alfresco bathtub',
'Amazon Echo',
'Apple TV',
'BBQ grill',
'Baby bath',
'Baby monitor',
'Babysitter recommendations',
'Balcony',
'Bath towel',
'Bathroom essentials',
'Bathtub',
'Bathtub with bath chair',
'Beach essentials',
'Beach view',
'Beachfront',
'Bed linens',
'Bedroom comforts',
'Bidet',
'Body soap',
'Breakfast',
'Breakfast bar',
'Breakfast table',
'Building staff',
'Buzzer/wireless intercom',
'Cable TV',
'Carbon monoxide detector',
'Cat(s)',
'Ceiling fan',
'Ceiling hoist',
'Central air conditioning',
'Changing table',
"Chef's kitchen",
'Children’s books and toys',
'Children’s dinnerware',
'Cleaning before checkout',
'Coffee maker',
'Convection oven',
'Cooking basics',
'Crib',
'DVD player',
'Day bed',
'Dining area',
'Disabled parking spot',
'Dishes and silverware',
'Dishwasher',
'Dog(s)',
'Doorman',
'Double oven',
```

```
'Dryer',
'EV charger',
'Electric profiling bed',
'Elevator',
'En suite bathroom',
'Espresso machine',
'Essentials',
'Ethernet connection',
'Exercise equipment',
'Extra pillows and blankets',
'Family/kid friendly',
'Fax machine',
'Fire extinguisher',
'Fire pit',
'Fireplace guards',
'Firm mattress',
'First aid kit',
'Fixed grab bars for shower',
'Fixed grab bars for toilet',
'Flat path to front door',
'Formal dining area',
'Free parking on premises',
'Free street parking',
'Full kitchen',
'Game console',
'Garden or backyard',
'Gas oven',
'Ground floor access',
'Gym',
'HBO GO',
'Hair dryer',
'Hammock',
'Handheld shower head',
'Hangers',
'Heat lamps',
'Heated floors',
'Heated towel rack',
'Heating',
'High chair',
'High-resolution computer monitor',
'Host greets you',
'Hot tub',
'Hot water',
'Hot water kettle',
'Indoor fireplace',
'Internet',
'Iron',
'Ironing Board',
'Jetted tub',
'Keypad',
'Kitchen',
'Kitchenette',
'Lake access',
'Laptop friendly workspace',
```

```
'Lock on bedroom door',
'Lockbox',
'Long term stays allowed',
'Luggage dropoff allowed',
'Memory foam mattress',
'Microwave',
'Mini fridge',
'Mobile hoist',
'Mountain view',
'Mudroom',
'Murphy bed',
'Netflix',
'Office',
'Other',
'Other pet(s)',
'Outdoor kitchen',
'Outdoor parking',
'Outdoor seating',
'Outlet covers',
'Oven',
'Pack ’n Play/travel crib',
'Paid parking off premises',
'Paid parking on premises',
'Patio or balcony',
'Pets allowed',
'Pets live on this property',
'Pillow-top mattress',
'Pocket wifi',
'Pool',
'Pool cover',
'Pool with pool hoist',
'Printer',
'Private bathroom',
'Private entrance',
'Private gym',
'Private hot tub',
'Private living room',
'Private pool',
'Projector and screen',
'Propane barbeque',
'Rain shower',
'Refrigerator',
'Roll-in shower',
'Room-darkening shades',
'Safe',
'Safety card',
'Sauna',
'Security system',
'Self check-in',
'Shampoo',
'Shared gym',
'Shared hot tub',
'Shared pool',
'Shower chair',
```

```
'Single level home',
'Ski-in/Ski-out',
'Smart TV',
'Smart lock',
'Smoke detector',
'Smoking allowed',
'Soaking tub',
'Sound system',
'Stair gates',
'Stand alone steam shower',
'Standing valet',
'Steam oven',
'Step-free access',
'Stove',
'Suitable for events',
'Sun loungers',
'TV',
'Table corner guards',
'Tennis court',
'Terrace',
'Toilet paper',
'Touchless faucets',
'Walk-in shower',
'Warming drawer',
'Washer',
'Washer / Dryer',
'Waterfront',
'Well-lit path to entrance',
'Wheelchair accessible',
'Wide clearance to bed',
'Wide clearance to shower',
'Wide doorway',
'Wide entryway',
'Wide hallway clearance',
'Wifi',
'Window guards',
'Wine cooler',
'toilet',
```

In the list above, some amenities are more important than others (e.g. a balcony is more likely to increase price than a fax machine), and some are likely to be fairly uncommon (e.g. 'Electric profiling bed'). Based on previous experience in the industry, and furtherresearch into which amenities are considered by guests to be more important, a selection of the more important amenities were extracted. These were then selected from for inclusion in the final model depending on how sparse the data was. For example, if it turns out that almost all properties have/do not have a particular amenity,

that feature will not be very useful in differentiating between listings or helping explain differences in prices.

The whole convoluted code for this can be found on GitHub, but this is the final section where I removed columns where over 90% of the listings either had or did not have a particular amenity:

```
# Produces a list of amenity features where one category (true or false) contains fewer than 10% o
infrequent_amenities = []
for col in df.iloc[:,41:].columns:
    if df[col].sum() < len(df)/10:
        infrequent_amenities.append(col)
print(infrequent_amenities)

# Dropping infrequent amenity features
df.drop(infrequent_amenities, axis=1, inplace=True)
```

infrequent-amenities.py hosted with ❤️ by GitHub view raw

These are the amenities that I ended up keeping:

- Balcony
- Bed linen
- Breakfast
- TV
- Coffee machine
- Basic cooking equipment
- White goods (specifically a washer, dryer and/or dishwasher)
- Child-friendly
- Parking
- Outdoor space

- Greeted by host
- Internet
- Long term stays allowed
- Pets allowed
- Private entrance
- Safe or security system
- Self check-in

Summary

After these (and many other) cleaning and pre-processing steps, the Airbnb was in suitable form to begin exploration and modelling, and you can read more about this in my next post on data exploration, and another post I wrote about building a predictive model.

. . .

If you found this post interesting or helpful, please let me know via the medium of claps and/or comments, and you can follow me in order to be notified about future posts. Thanks for reading!

Airbnb Python Data Science Data