How have I (datascientist) been searching for a rent in Edinburgh

PETER ALACS, BDP

PYDATA – 25TH MARCH 2019

My Landlord is selling her flat and I have to move out (the pic is an illustration)

Competitive market

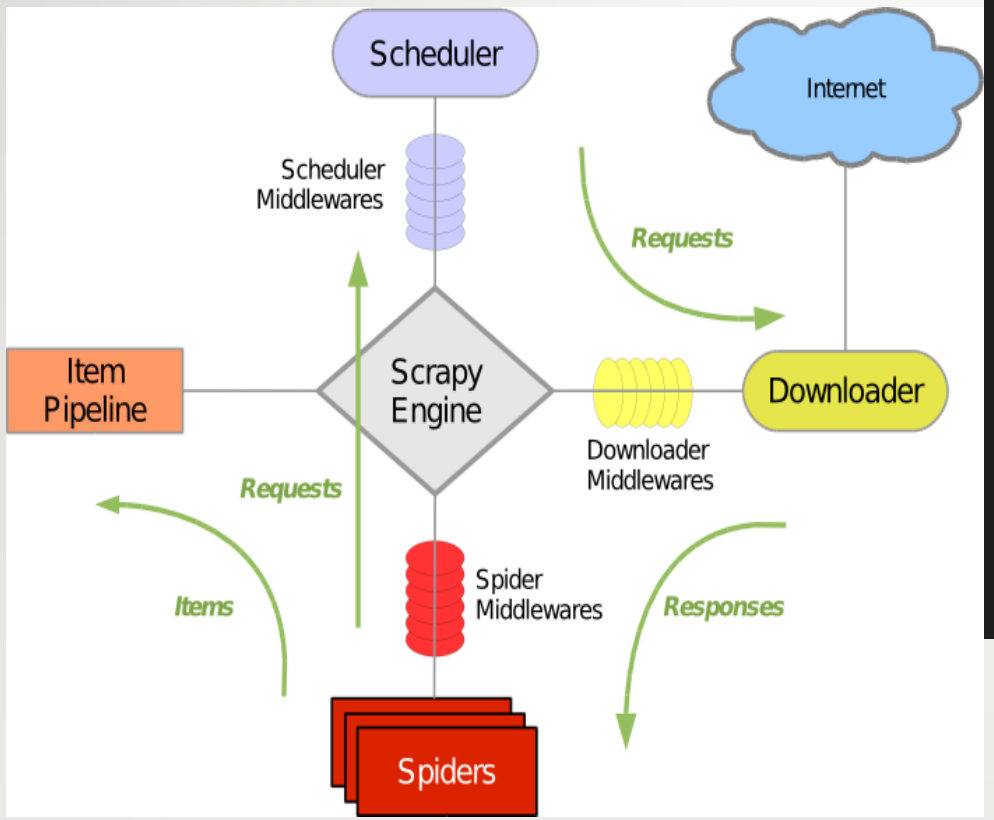
I have to do multiple viewings / applications

I need an efficient way to find my new place

How to spice up flat-hunting? (for geeks only)

- The data:
  - Coming from multiple (4+) agencies
  - Possible augmentations?
- The model:
  - Is the price reasonable?
  - Could I construct a score that would rank my queries according to my preferences?
- The process:
  - I need a good overview of the opportunities and the upcomings

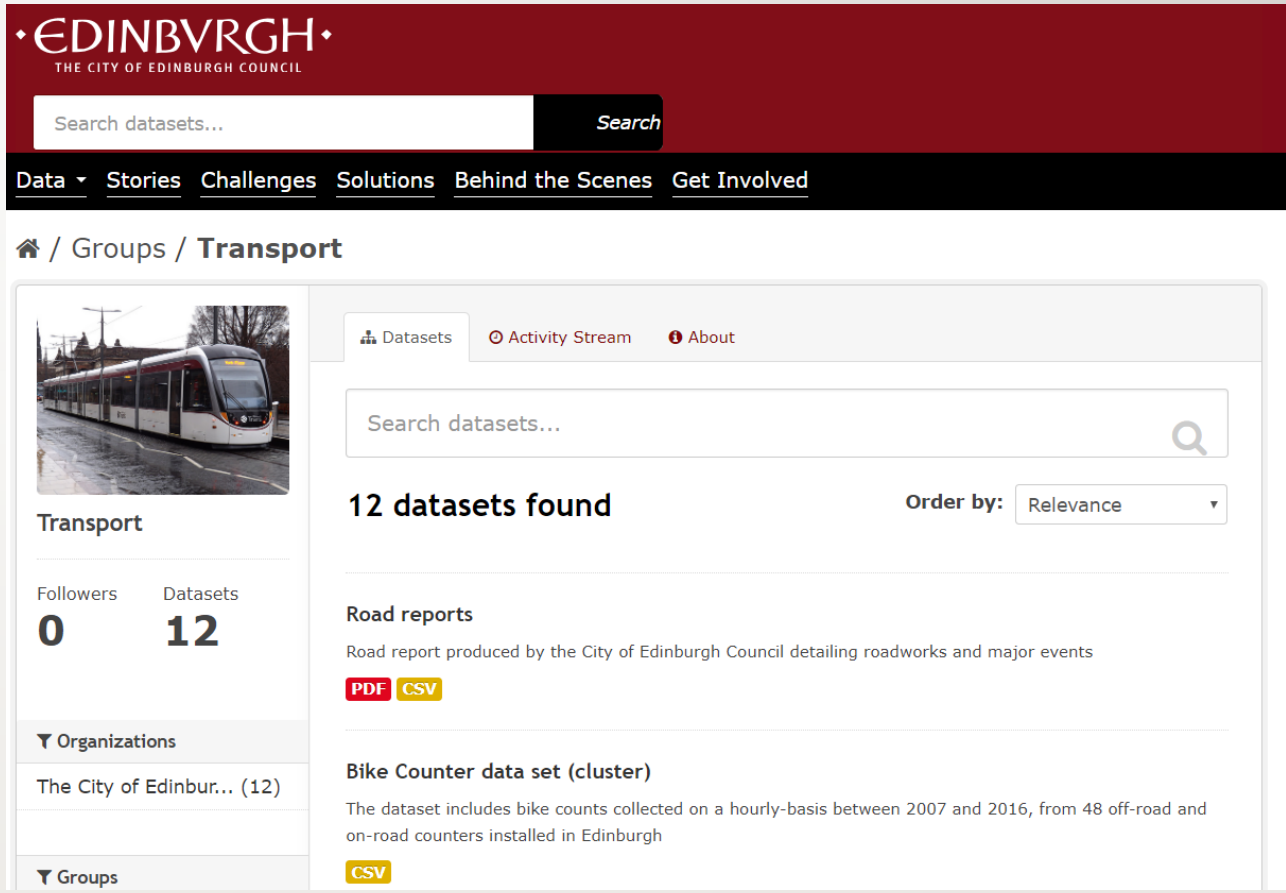
The agencies' DATA: go Scrapy!




```python
import scrapy
from scrapy.loader import ItemLoader
from flapp.items import [redacted]FlatItem

class [redacted]Spider(scrapy.Spider):
    name = "[redacted]eflats"

    def start_requests(self):
        urls = [
            'https://[redacted]/latest-properties/'
        ]
        for url in urls:
            yield scrapy.Request(url,callback=self.parse)

    def parse(self,response):

        flatlinks = response.xpath('//h3[@class="entry-title"]/a/@href').extract()[:3]
        for link in flatlinks:
            yield scrapy.Request(link,callback=self.parse1)

    def parse1(self,response):
        l = ItemLoader(item=SouthSideFlatItem(source=self.name,link=response.request.url),response=response)
        l.add_xpath('price','//span[@class="page-price"]/text()')
        l.add_xpath('address_street','//span[@class="item-street"]/text()')
        l.add_xpath('address_pcode','//span[@class="item-pcode"]/text()')
        l.add_xpath('available','//div[contains(@class,"date-available")]/text()')
        l.add_xpath('descr_short','//div[contains(@class,"tab-content")]/h2[@class="entry-title"]/text()')
        l.add_xpath('description','//div[contains(@class,"tab-content")]/p')

        return l.load_item()
```

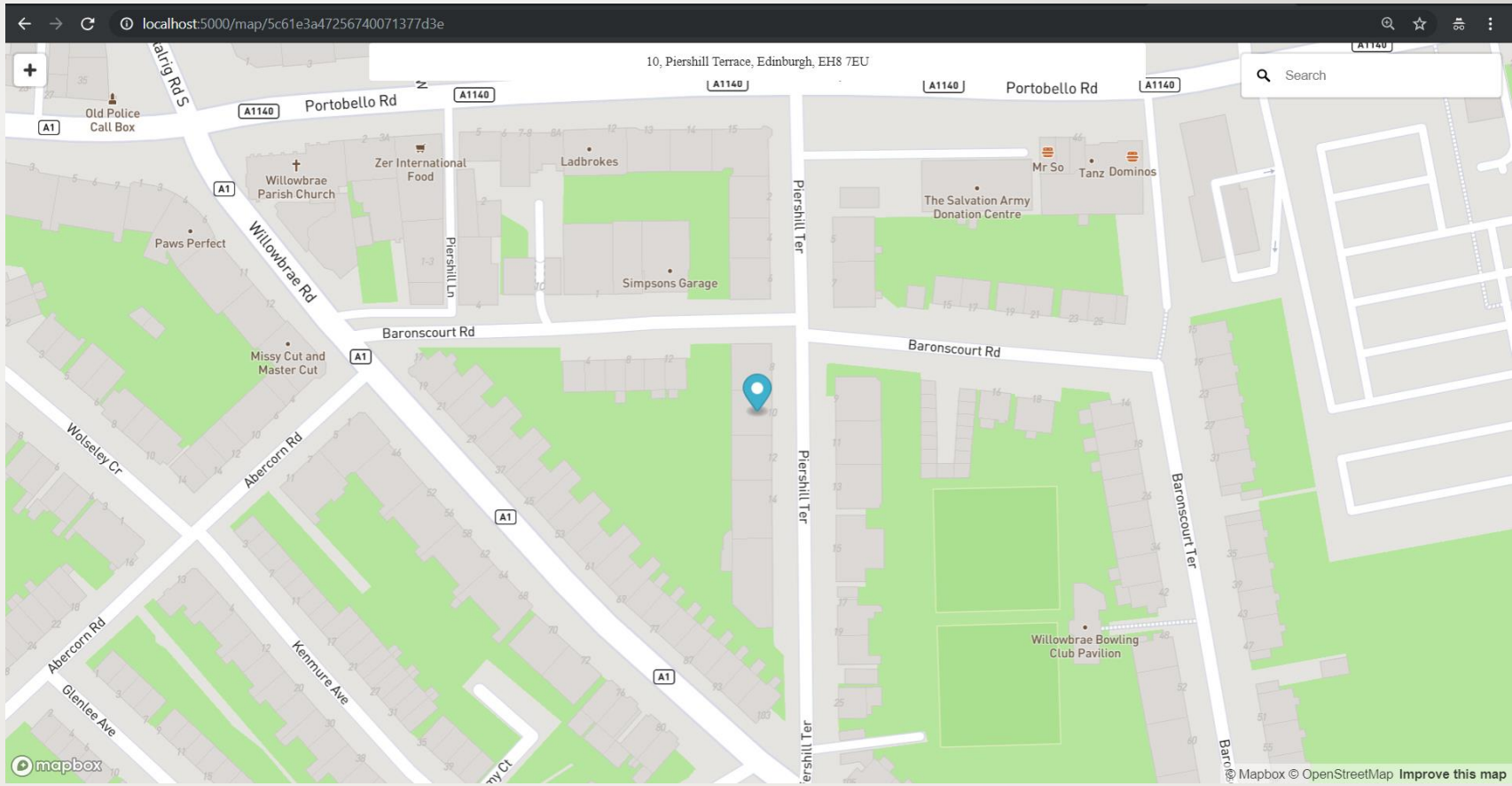
Open DATA

- UK government strategy
- https://edinburghopendata.info/
  - (not updated regularly)
  - Transportation: accidents
  - Planning (construction projects)
  - Public toilets
  - Etc…



Maps

- Mapbox for visualizing
- Google Maps API for geocoding and data augmentation (e.g. distance from the centrum)



The DATAbase: MongoDB

- NoSQL, Document-based
- Fits perfectly to ever changing / forming modelling needs, unknown data types (new agencies)
- MongoEngine ODR

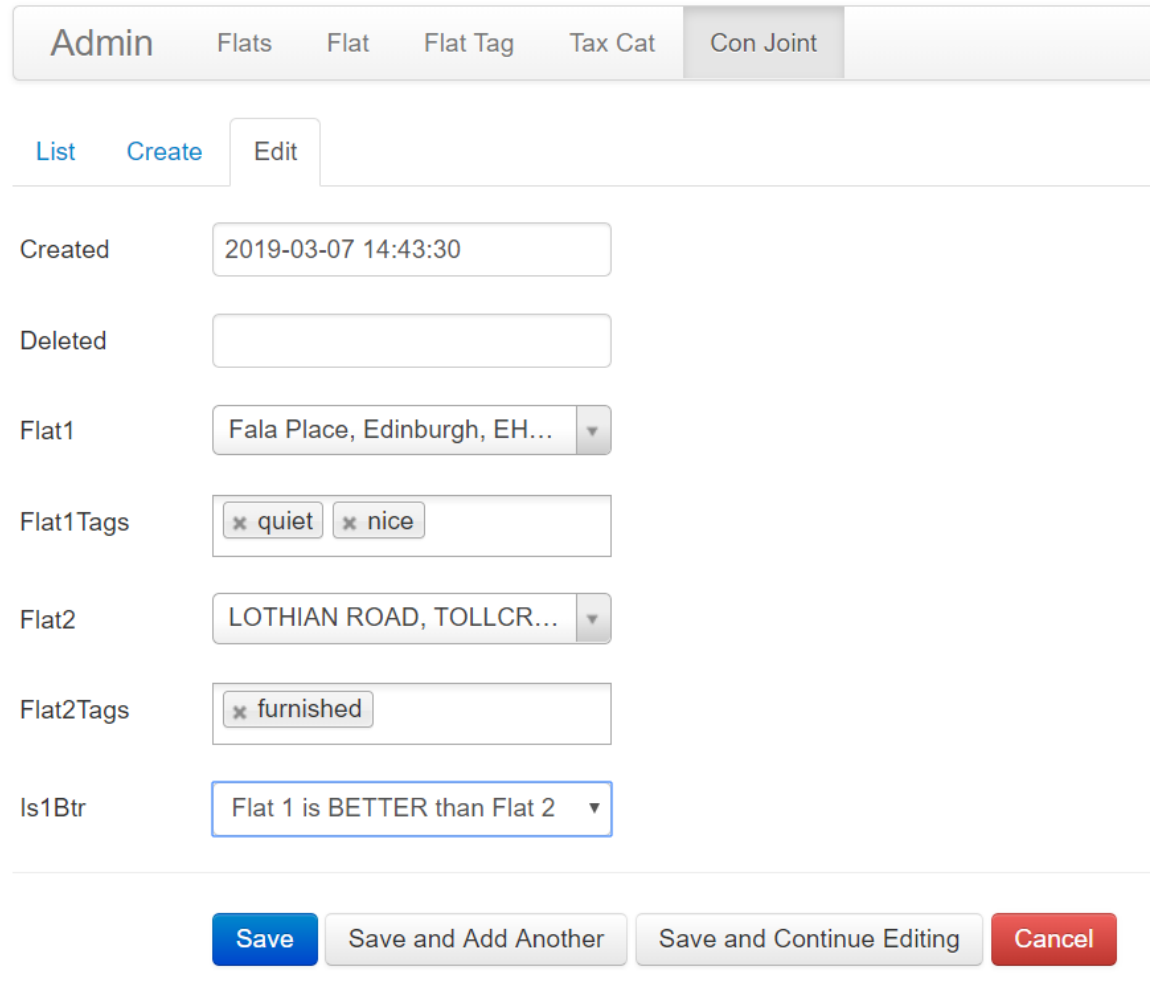
```
class TaxCat(db.Document):
    cat = db.StringField()
    tax = db.DecimalField()
    def __unicode__(self):
        return f"{self.cat} (£{self.tax/12:.2f})"

class Viewing(db.EmbeddedDocument):
    created = db.DateTimeField(default = datetime.utcnow)
    viewingat = db.DateTimeField()
    contact = db.StringField()
    result = db.StringField()

    def __unicode__(self):
        return self.viewingat

class AddressLocation(db.EmbeddedDocument):
    address = db.StringField()
    lng = db.FloatField()
    lat = db.FloatField()
```

The process: Flask

- Lightweight:
  - less code in the beginning,
  - Expandable: all components available, but need to manage it manually
- Supported by MongoEngine (e.g. admin-views: all creation, update, list views are free)

The model: Contjoint Analysis

- Using the available Flask views compare 2 flats at once
  - Add tags for reasoning
  - Pick one as 'BETTER'
- Tags are converted to latent traits
- Price = Price Acknowledged + Extra



Results

- Fun: a lot in the weekends with developing
- Views: some
- Rejected: 1
- Accepted: 0

Morals

- The three great virtues of a programmer: laziness, impatience, and hubris. (Lary Wall)

Thank You!