Bolighed - powered by Python

A real life case study

About me

- Mathematician by education
- After some years in research I have since worked as a Python developer, primarily in data processing at:
  - Danish Geodata Agency, now called SDFE
  - Danish Meteorological Institute
  - And now, **Bolighed A/S**

About Bolighed

- Bolighed is a website aimed at house owners and hunters as well.
- Collects and presents information from a lot of public data sources:
  - BBR (basic information about buildings, dwellings and ownership).
  - Tinglysning (loans, entitlements etc.)
  - Energy marks
  - …
- Also a lot of “closed” non-public sources:
  - Price estimate models (machine learning)
  - Sales data
  - ...

The stack at Bolighed

Advanced setup with a *lot* of components:

- Amazon EC2
- Docker
- Redis
- Eleasticsearch
- Cloudflare
- Postgres / Postgis (databases)
- Nginx
- Tornado
- … and a whole lot more ...

Where is Python used?

The frontend is using AngularJS and Python takes care of the rest:

- Data import
- Infrastructure
  - Deployment / configuration via **ansible**
- Backend api
  - Flask
  - Django
- Data analysis
  - Numpy, scipy, pandas, matplotlib, scikit-learn, SQL via SQLAlchemy.

A deeper look into some of the use cases

Backend:

- Flask with SQLAlchemy
- Super simple and very flexible setup:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Backend

Why Python and not PHP, C, C# or Java (or Ruby)? Is performance OK??

- Much, much nicer and more maintainable than PHP!
- Very high level interface to various services / infrastructure
  - Elasticsearch, Redis, Postgres (SQLAlchemy), Datadog, Amazon EC2 / S3.
- Lot’s of caching mechanisms and load balancing in place - very few actual database calls...

Backend

We also have some apis running in Django:

- More structured than Flask + SQLAlchemy
- Includes it’s own ORM (Object-relational mapping) as a high level interface to the database.
- Lot’s of extensions, e.g. django REST framework
- Used by many *huge* web applications out there:
  - **Instagram**
  - **Pinterest**
  - ...

Django’s ORM

```
class CustomerType(models.Model):
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)
    name = models.CharField(max_length=255, unique=True)
    def __str__(self):
        return self.name

class PropertyData(models.Model):
    """
    Models any kind of property
    """
    bbr_property_data = models.ForeignKey('BBRPropertyData', null=True)
    address = models.ForeignKey('Address', null=True)
```

Django’s ORM

```
(venv_bm) Simons-MacBook-Pro:business_manager simonkokkendorff$ python manage.py shell
Python 3.6.0 (default, Dec 24 2016, 08:01:42)
Type "copyright", "credits" or "license" for more information.
...
In [1]: from business_manager.leads import models
In [2]: for obj in models.Address.objects.all().filter(street__startswith="Åsvej")[:2]:
   ...:     print(obj)
   ...:
Åsvejen 4  , 4330
Åsvejen 6  , 4330
```

- Specific database is ‘abstracted away’
- No explicit SQL queries
- However, in some cases the high level ORM is too rigid and one must resolve to plain old SQL...

Data import

We use a lot of different python libraries and protocols for fetching data from various sources:

- **Boto / boto3** for talking to Amazon EC2 and S3
- **Requests** for REST-interfaces / scraping
- **Pysimplesoap / Requests** for SOAP (XML) interfaces (sigh….)

For example there is a great API for all danish addresses at http://dawa.aws.dk/

```
In [14]: import requests
In [15]: r = requests.get("http://dawa.aws.dk/adresser", params={"vejnavn":"Fasanvej", "postnr": 8210, "husnr":15, "struktur":"mini"})
In [16]: r.json()
Out[16]:
[{'adgangsadresseid': '0a3f5096-212e-32b8-e044-0003ba298018',
  'dør': None,
  'etage': None,
  'husnr': '15',
  'id': '19910d90-1d47-41c9-e044-0003ba298018',
  'kommunekode': '0751',
  'postnr': '8210',
  'postnrnavn': 'Aarhus V',
  'status': 1,
  'supplerendebynavn': None,
  'vejkode': '2032',
  'vejnavn': 'Fasanvej',
  'x': 10.1787079932534,
  'y': 56.1647588529531}]
```

Addresses, postal districts and various other data are imported from this endpoint on a regular basis.

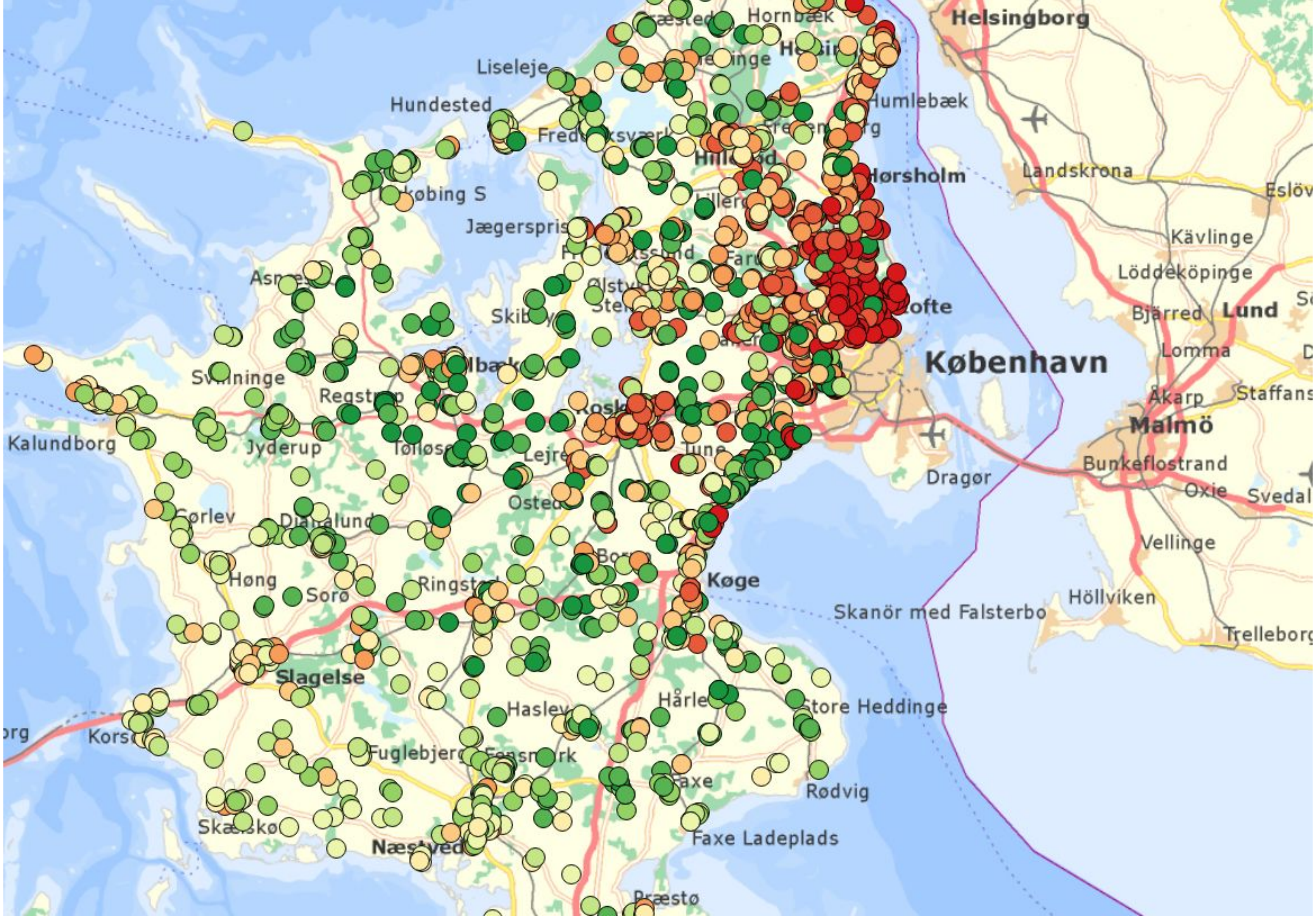
Data analysis

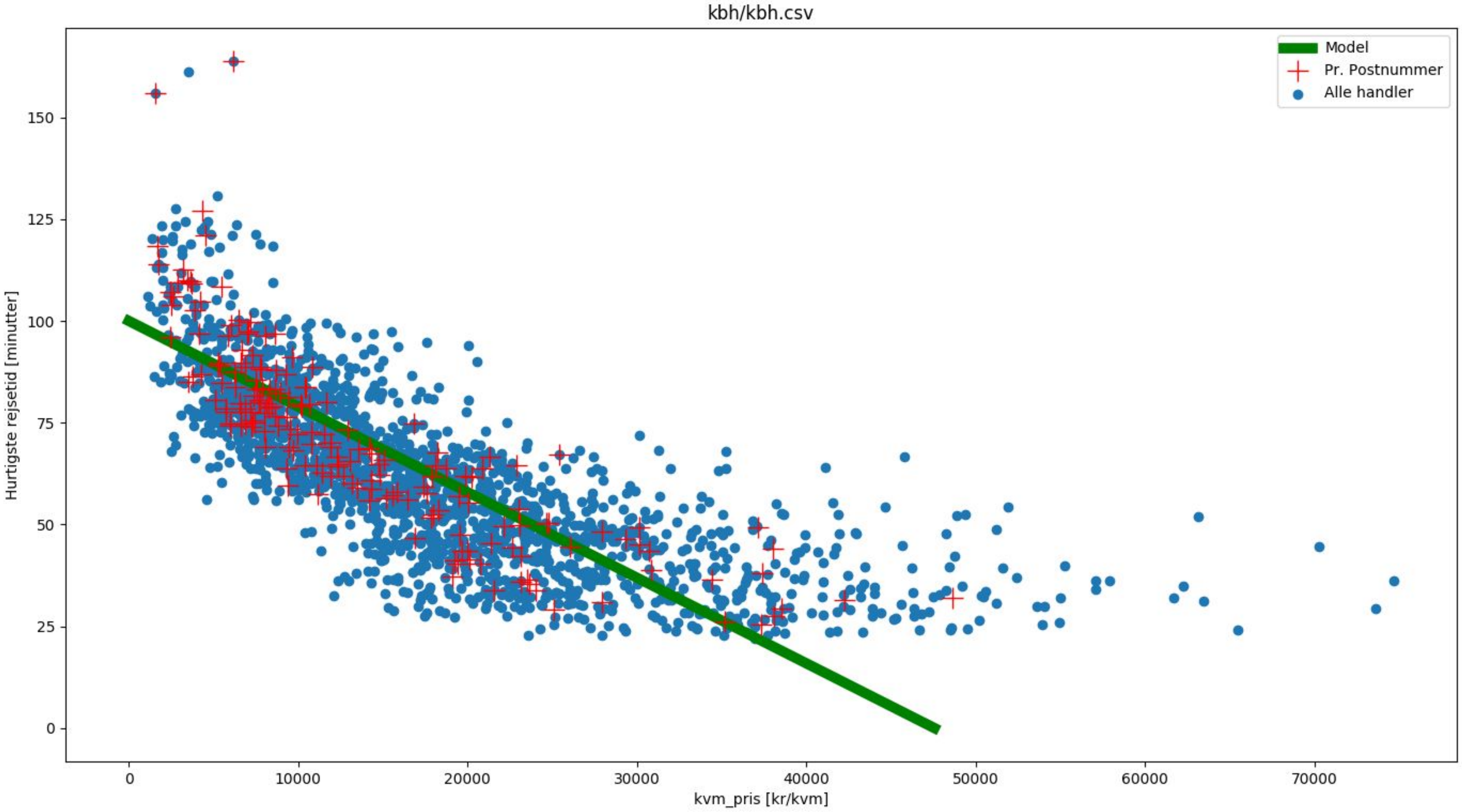
Case:

- Examine what the relation between house prices and travel time to Copenhagen is?

Plan:

- Fetch sales data + geographic location from database (Postgis) via SQLAlchemy.
- Use googlemaps Python API to query travel times to Copenhagen Central station for these locations.
- Do some analysis and plotting with numpy (linear regression, filtering) and matplotlib

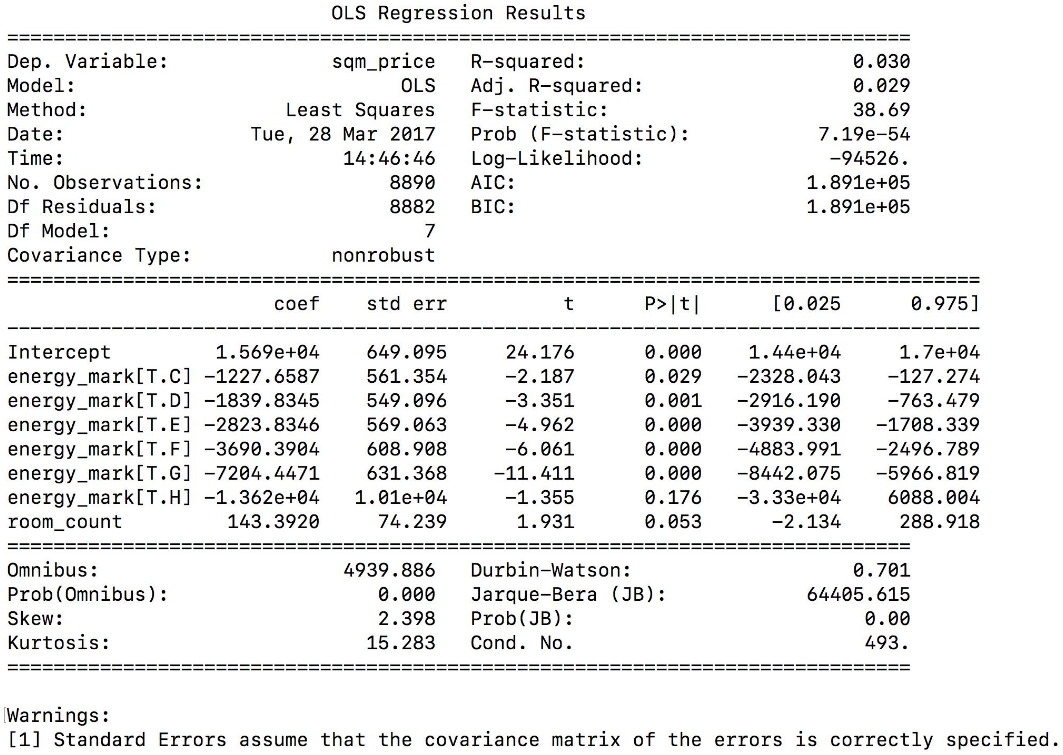

Helsingborg
Hornbæk
Græsted
Helsingør
Liseleje
Hundested
Humlebæk
Fredensborg
Frederiksværk
Hillerød
Hørsholm
Landskrona
Eslöv
Nykøbing S
Jægerspris
Kävlinge
Frederikssund
Farum
Löddeköping
Asnæs
Ølstykke
Stenløse
Skibby
Gentofte
Bjärred
Lund
Lomma
København
Svinninge
Holbæk
Regstrup
Roskilde
Åkarp
Staffanstorp
Malmö
Kalundborg
Jyderup
Tølløse
Lejre
Tune
Bunkeflostrand
Dragør
Oxie
Svedala
Osted
Gørlev
Dianalund
Vellinge
Borup
Høng
Sorø
Ringsted
Køge
Höllviken
Skanör med Falsterbo
Trelleborg
Slagelse
Haslev
Hårlev
Store Heddinge
Korsør
Nyborg
Fuglebjerg
Fensmark
Faxe
Rødvig
Skælskør
Næstved
Faxe Ladeplads
Præstø


kbh/kbh.csv
Model
Pr. Postnummer
Alle handler
Hurtigste rejsetid [minutter]
150
125
100
75
50
25
0
0
10000
20000
30000
40000
50000
60000
70000
kvm_pris [kr/kvm]

Something else that I’ve been working on...

- Mapping value increases for houses the next year:
  - https://s3.bolighed.dk/static/stories/prisprognose/index.html#7/56.188/11.646
- And something completely different - a fancy map:
  - http://gittebach.dk/case/story.html
- How does house prices depend on various parameters?
  - For example energy marks?
  - Create models using scikit-learn…
  - ...or tensorflow … or...

Work in progress… analysis with statsmodels



```
                            OLS Regression Results
==============================================================================
Dep. Variable:              sqm_price   R-squared:                       0.030
Model:                            OLS   Adj. R-squared:                  0.029
Method:                 Least Squares   F-statistic:                     38.69
Date:                Tue, 28 Mar 2017   Prob (F-statistic):           7.19e-54
Time:                        14:46:46   Log-Likelihood:                -94526.
No. Observations:                8890   AIC:                         1.891e+05
Df Residuals:                    8882   BIC:                         1.891e+05
Df Model:                           7
Covariance Type:            nonrobust
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept         1.569e+04    649.095     24.176      0.000    1.44e+04     1.7e+04
energy_mark[T.C] -1227.6587    561.354     -2.187      0.029   -2328.043    -127.274
energy_mark[T.D] -1839.8345    549.096     -3.351      0.001   -2916.190    -763.479
energy_mark[T.E] -2823.8346    569.063     -4.962      0.000   -3939.330   -1708.339
energy_mark[T.F] -3690.3904    608.908     -6.061      0.000   -4883.991   -2496.789
energy_mark[T.G] -7204.4471    631.368    -11.411      0.000   -8442.075   -5966.819
energy_mark[T.H] -1.362e+04   1.01e+04     -1.355      0.176   -3.33e+04    6088.004
room_count         143.3920     74.239      1.931      0.053      -2.134     288.918
====================================================================================
Omnibus:                     4939.886   Durbin-Watson:                   0.701
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            64405.615
Skew:                           2.398   Prob(JB):                         0.00
Kurtosis:                      15.283   Cond. No.                         493.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Thank you for your attention!

Some links:

- https://bolighed.dk/
- https://da-dk.facebook.com/bolighed/
- https://twitter.com/bolighed