EY NextWave
Data Science Competition

Marcus Gawronsky

University of Cape Town
South Africa
May 20, 2019

Introduction

This report forms part of an entry in the EY NextWave Data Science Competition 2019. It outlines the analysis and applications of predictive models trained on a dataset of user GPS data. The report provides details on how to replicate the analysis performed, as well as a criteria for evaluating the chosen model.

Reproducibility

The analysis in the report was performed using Python 3.7.3 on Ubuntu 19.04. A full list of package dependencies along with the code used to run the analysis is provided in the appendix and can be installed and executed using Anaconda 4.6.11 or using the latest Miniconda Docker Container.

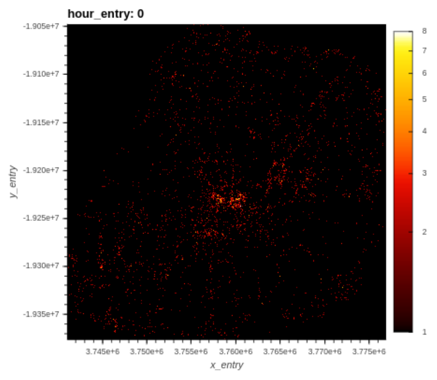
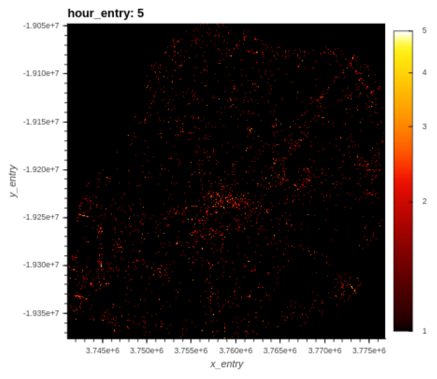
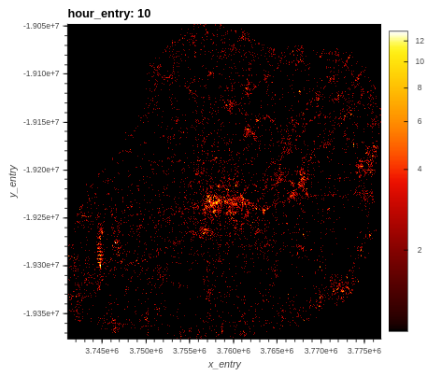
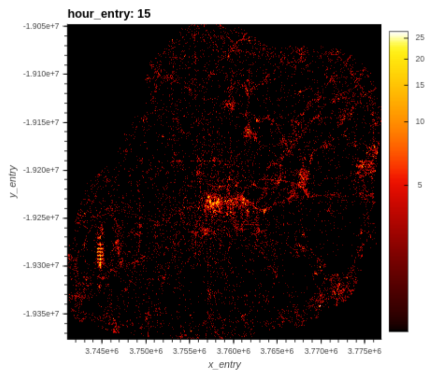
Data

The datasets provided for the competition came split into a training and test-set with 814262 and 202937 entries respectively. The table below provides a sample of the data containing datapoints trips by individuals. Each individual is given a *hash*, which is reset each day, and each trip is given a unique *trajectory_id*.

hash	**trajectory_id**	**time_entry**	**time_exit**	**vmax**	**vmin**	**vmean**
005...	traj_005a...	10:20:00	10:28:17	0.0	0.0	0.0
005...	traj_005a...	10:40:51	10:52:47	NaN	NaN	NaN
005...	traj_005a...	11:22:09	11:45:23	0.0	0.0	0.0

x_entry	**y_entry**	**x_exit**	**y_exit**
3.758e+06	-1.904e+07	3.75e+06	-1.905e+07
3.759e+06	-1.911e+07	3.75e+06	-1.913e+07
3.767e+06	-1.920e+07	3.77e+06	-1.912e+07

In the table below, one can see imaged which show the distribution of individuals, based on their GPS locations at different points accross the day. One can see that as the day progresses, we see a greater density of individuals in the city centre.

Time	Image
00:00	 hour_entry: 0 -1.905e+7 -1.910e+7 -1.915e+7 -1.920e+7 -1.925e+7 -1.930e+7 -1.935e+7 y_entry 3.745e+6 3.750e+6 3.755e+6 3.760e+6 3.765e+6 3.770e+6 3.775e+6 x_entry 8 7 6 5 4 3 2 1
05:00	 hour_entry: 5 -1.905e+7 -1.910e+7 -1.915e+7 -1.920e+7 -1.925e+7 -1.930e+7 -1.935e+7 y_entry 3.745e+6 3.750e+6 3.755e+6 3.760e+6 3.765e+6 3.770e+6 3.775e+6 x_entry 5 4 3 2 1
10:00	 hour_entry: 10 -1.905e+7 -1.910e+7 -1.915e+7 -1.920e+7 -1.925e+7 -1.930e+7 -1.935e+7 y_entry 3.745e+6 3.750e+6 3.755e+6 3.760e+6 3.765e+6 3.770e+6 3.775e+6 x_entry 12 10 8 6 4 2
15:00	 hour_entry: 15 -1.905e+7 -1.910e+7 -1.915e+7 -1.920e+7 -1.925e+7 -1.930e+7 -1.935e+7 y_entry 3.745e+6 3.750e+6 3.755e+6 3.760e+6 3.765e+6 3.770e+6 3.775e+6 x_entry 25 20 15 10 5

Given this data the aim of the challenge was two-fold:

1. To predict the final location of an individual at 15:00
2. To determine a decision-boundary for what constituted the city-centre in order to determine if individuals were in the city centre or not

Applications

There exists a large miriad of application for this data and the application of predictive modelling, as shown in the table below. While these applications have vastly different requirements in terms of accuracy, explainability and security, they can be broadly split into simulation-based application and non-simulation-based applications. The primary distinction between these two applications is that in simulation-based applications we require a global model, which given a starting point on a map can continuously predict the next location at which the person is observed. Non-simulation-based models may have different requirements and allow for models which only apply to an individual as so do not generalize to arbitary groups of users.

Simulation	**Non-simulation**
Disaster Relief	**Fraud detection (Credible credit card use)**
Infectious Disease Modelling	Policing (Testimony, finding suspects)
Urban Planning	Marketing/ Product Placement

While the simulation distinction is import, other applications may have different requirements such as explainability or a requirement in quantifying uncertainty.

Applications such as Fraud detection would use GPS coordinates of an individual given through a cellphone or records of where last a transaction was made in order to try determine if it was likely the person made the next transaction. In this application, location is not the only priority, security and the quantification of risk or uncertainty is vitally important is we may set a threshold for the the probability of a person being at an store in order to determine if the card has been duplicated. Similarly we may prefer models which have the ability to be better secured along with the users data and which may be stored and used on edge devices, such as credit card machine or EPOS devices.

Priorities

In order to best determine our ideal application and methodology, we must first determine a framework by which to consider and evaluate these use-cases. These can be broken up into four main subsections: Sustainability, Security, Resources and Predictive Requirements.

Sustainability

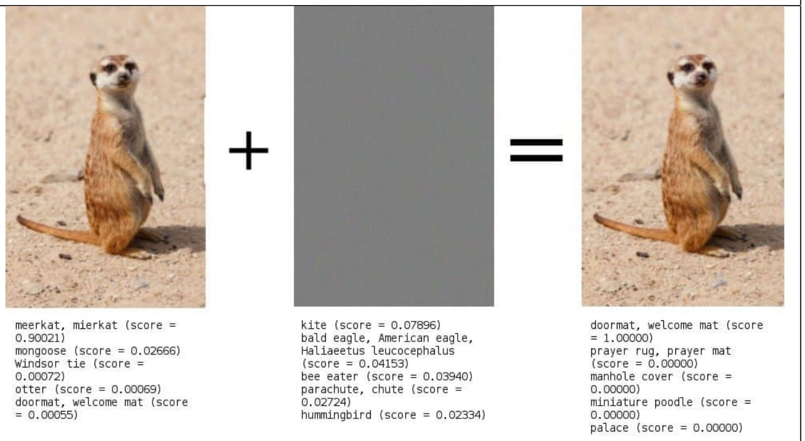
A major consideration in decising upon our applications is around sustainability. In order to identify if our application is sustainable we need to determine if there a market or need for such as product which can cover even server costs and development. We also need to determine if certain applications have requirements for greater or continuous data aquisition and whether the market allows for that.

Arguable, all the application present some oppotunity for sustainability and have different markets in which they may compete. However, while simulation-based methods require little in terms of further data aquisition, our non-simulation-based methods may require continuous data aquisition of where individuals are or have been in order to provide utility.

While this is not a limiting factor applications such as Marketing/ Product Placement may lack the sufficient incentives on the part of users to provide such data in order to optimally market to them. In the case of Google Maps, the incentives do work by providing users with a facility by which to navigate and valuable recommendation in exchange for marketing data.

Security

Security is a major consideration for Data Science and IT applications and is a concern which has been growing by legislators and individuals. In the table below a list of security concerns and examples or explanations is provided:

Type	**Example**
Fooled by Data	 meerkat, mierkat (score = 0.90021) mongoose (score = 0.02666) Windsor tie (score = 0.00072) otter (score = 0.00069) doormat, welcome mat (score = 0.00055) kite (score = 0.07896) bald eagle, American eagle, Haliaeetus leucocephalus (score = 0.04153) bee eater (score = 0.03940) parachute, chute (score = 0.02724) hummingbird (score = 0.02334) doormat, welcome mat (score = 1.00000) prayer rug, prayer mat (score = 0.00000) manhole cover (score = 0.00000) miniature poodle (score = 0.00000) palace (score = 0.00000)
Adverserial Model Drift	Agents feed false data
Model Leaks	1. Access to accurate global model (GPS spoof me to learn about you) 2. Parameterizes personal data (Leaks means I know where you live)
Data Leaks	Centralized data is accessed

A major trade-off in our application is the need for a centralized vs a distributed model. A centralized model is one in which all individuals are captured by a single model which must be centrally trained and queried by all users. While such models can scale, they necessitate data being centralized, which creates a risk or security leaks, and provide a tools which malicous agents can use in order to perform queries about the movements of other individuals. For example, if I have a model which only works for me and which I only have access to then if I put in your location it will not tell me where you will likely be next, however if I have a model which can make predictions about all individuals, I can spoof the model with your coordinates and determine where you will be. Given the concerns over our the privacy of our data and application this renders simulation-based applications less desireable, as they rely on a centralized model.

Similarly models such as random forests, which paramterize data through decision-rules (such as *if the person is at this location then they will be at this location* given us information about a persons location and) may be less preferable as they raise the security concerns about malacious agents gaining access to either our model or our data, not just the data. While this concern may be handles in some may through data obfuscation, this still remain a criterea for evaluation.

Lastly, as with many Deep Learning, concerns must be raised in certain applications about our ability to have assurances about our model, as with many cases our inability to properly reason about the model means, as with images, they adding noise to images can lead to castly inaccurate results, which may allow persons to trick models or feed it data which renders it less accurate over time without us knowing or being easily able to identify.

Resources

Across many applications an unavoidable limitation is resource constraints. Algorithms which rely on matrix factorization do no scale to large datasets easily, as they require all the data being in memory and are thus heabily io-bound. Similarly extremely Deep Neural Networks can be gigabytes in size and can require specialized and expensive hardware in order to use or may require communication with a server, limiting their application.

Scenario	**Method**	**Cheap**	**Scallable**
Embedded System	Least-square Regression	Yes	No
Large cluster	Alex-net (Deep CNN's)	No	Yes

Predictive Requirements

Different models and applications have different predictive requirements. In the table below, we detail three main requirements: Correctness, Confidence and Explainability. Correctness, here, is the accuracy of a given model which can mean common loss functions like Mean Squared-Error or Binary Cross-entropy, which may be evaluated using some form of validation, such ass using a Test-Train-Split or using Stratified K-fold Cross-validation.

Confidence asks whether our prediction gives some measure of uncertaintly of its estimate. For many classification tasts, using Binary cross-entropy a pseudo-probability is specified using a link function and using our loss function a probaility distribution is assumed, meaning we have some assumptions of the distribution of our uncertainty and some quantization in the form of a pseudo-probability. In many bayesian methods the uncertainty of our predicted output can be extensive, in which we are given a probability distribution of an output and can use a range of methods for point estimation based on the risk tolerance of the application.
Lastly, in explainability, some models allow for various levels of explainability in determining what variables in our data and what parameters led to such a prediction. Here linear models and simple decision tree provide high levels of explainabilitu and

assurance, many deep learning methods provide limited methods by which to explain a given prediction.

Example	**Correctness**	**Confidence**	**Explainability**
Siri	High	Low	Low
Stock Prices	High	High	Low
Macroeconomic Models	Low	High	High
Medical Imaging	High	High	High

Looking at the table above, applications such as Siri demand the model used in that application to be correct or users become frustrated, but the certainty of a particular prediction or the ability to explain why a particular response was given is not particularly important. Alternatively in stock prediction, quantifying uncertainty is crucial in deciding optimal asset allocation.

In our applications discussed, applications such as Fraud Detection and Disaster Relief require high levels of correctness and confidence, as we need to know crucially the confidence of a prediction in order to make an optimal decision.

Methodology

From the framework above, it is clear there are obvious downsides to the large centralized models required for simulation-based application, both in terms of the resources requirements and the security risks.

Given these constraints, we must then tackle the issue of sustainability. Looking at Fraud Detection, much of the infrastructure is currently in place for businesses with a wide distribution of customer app installs, EPOS devices and credit card machines for data collection. A clear metric exists to evaluate the value to the business and models can easily be used along with other datasets and models already present in the business.

With this application it is clear that correctness, confidence and security are the primary concerns. Using Bayesian Modelling we can specify a seperate model unique to each person, which provides us a probability distribution for their location at a point in time. Here, we can make use of some basis explansion to create polynomial and piecewise functions of time and use Bayesian Ridge Regression, with a differing prior, in order to find an optimal basis in determining the location of a user at a point in time.

For this analysis, rather than looking at the data as a series of entries and exits, we just look at the locations of individuals at points in time, using the following model:

$$y_{p,i} = \beta_{0,p} + \beta_{1,p}t_{p,i} + \beta_{2,p}t_{p,i}^2 + \beta_{3,p}t_{p,i}^3 + \beta_{4,p}B(||t_{p,i} - 10{:}00\text{am}||_+)$$

$$x_{p,i} = \beta_{0,p} + \beta_{1,p}t_{p,i} + \beta_{2,p}t_{p,i}^2 + \beta_{3,p}t_{p,i}^3 + \beta_{4,p}B(||t_{p,i} - 10{:}00\text{am}||_+)$$

$$B(t) = x^2 log(x)$$

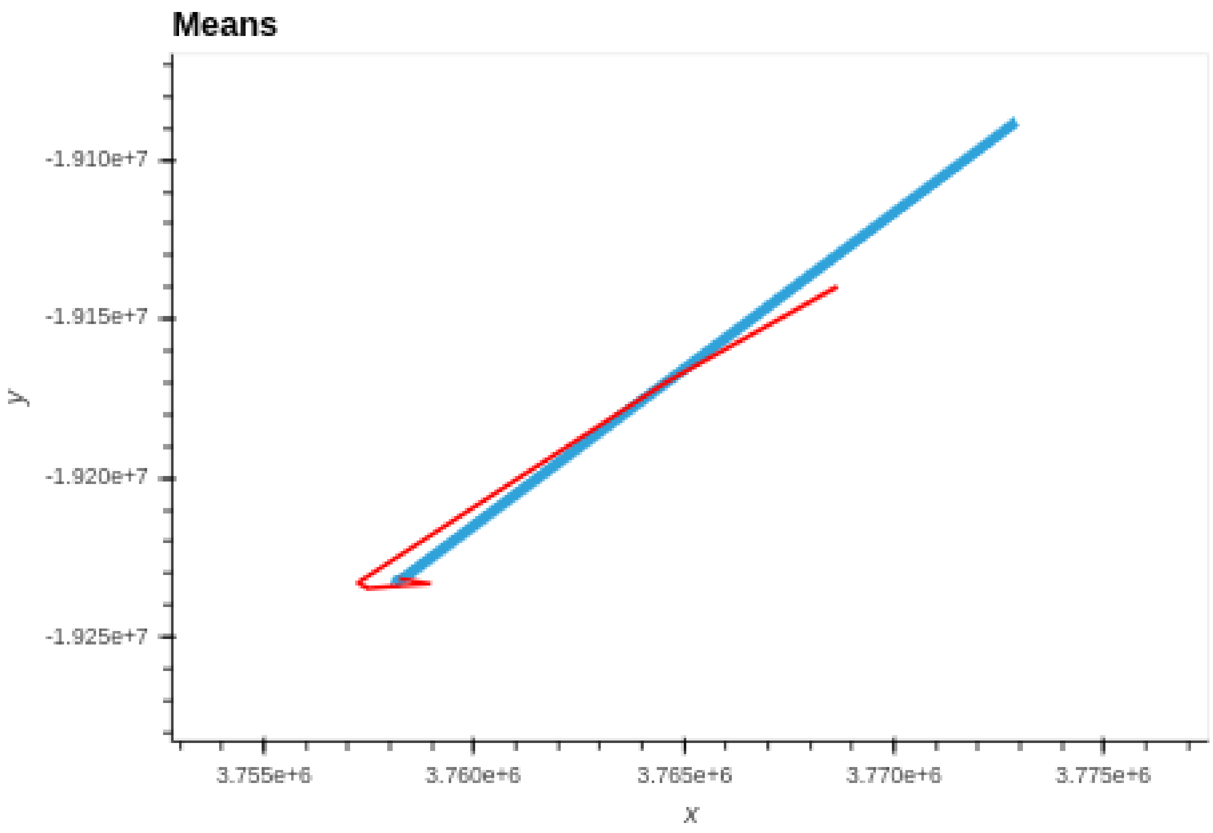
p is a person with parameter (β) and times (t)
i is a point in time (t)
$y_{p,i}$, $y_{p,i}$ are the longitude and latitude of an individual
$\beta_{k,p}$ are the parameters of each model used for each person

In order to fit our model, we used the specification foundin Tipping (2001) and MacKay (1992). This defines the distribution over our weights of:

$$Pr(\beta|\lambda) = N(\beta|0, I\lambda^{-1})$$

The priors for λ are chosen to be gamma distributions, using the conjugate prior for the precision of the Gaussian Distribution. Non-informative priors were chosen for our hyperparamters of 10^{-6}.

This model was tested over a range of Basis Functions, an example of a predicted path is shown in red below against the path of the user:



A final decision boundary for the city was chosen to be the 75th and 25th quantiles over our GPS coordinates throughout the day over the longitude and lattitude. This gave an f1 score of 0.72.

Conclusions

While model does not provide state of the art accuracy compared to other approached, this approach offers a miriad of advantages in security, scalability, explainability, compute and confidence. The abilty to analyze a moving probability distribution of the possible location of an individual over time as they move over a map, given their past movements provides a range of advantages, aiding decision-making across a range of applications.

Appendix

Anaconda Python Dependencies

```
name: EY
channels:
  - pytorch
  - conda-forge
  - intel
  - defaults
dependencies:
  - _tflow_select=2.3.0=mkl
  - absl-py=0.7.1=py37_0
  - arrow-cpp=0.13.0=py37hfe0260e_2
  - asn1crypto=0.24.0=py37_0
  - astor=0.7.1=py37_0
  - attrs=19.1.0=py37_1
  - backcall=0.1.0=py37_0
  - blas=1.0=mkl
  - bleach=3.1.0=py37_0
  - bokeh=1.0.4=py37_0
  - boost-cpp=1.68.0=h11c811c_1000
  - brotli=1.0.7=hf484d3e_1000
  - bzip2=1.0.6=h14c3975_5
  - c-ares=1.15.0=h7b6447c_1
  - ca-certificates=2019.3.9=hecc5488_0
  - cairo=1.14.12=h8948797_3
  - cartopy=0.17.0=py37hbb7e04d_1
  - certifi=2019.3.9=py37_0
  - cffi=1.12.2=py37h2e261b9_1
  - cftime=1.0.3.4=py37hdd07704_0
  - chardet=3.0.4=py37_1
  - click=7.0=py37_0
  - click-plugins=1.0.4=py37_0
  - cligj=0.5.0=py37_0
  - cloudpickle=0.8.1=py_0
  - colorcet=1.0.0=py37_0
  - cryptography=2.6.1=py37h1ba5d50_0
  - cudatoolkit=10.0.130=0
  - cudnn=7.3.1=cuda10.0_0
  - curl=7.64.0=hbc83047_2
  - cycler=0.10.0=py37_0
  - cytoolz=0.9.0.1=py37h14c3975_1
  - dask-core=1.2.0=py_0
  - datashader=0.6.9=py_0
  - datashape=0.5.4=py37_1
  - dbus=1.13.6=h746ee38_0
  - decorator=4.4.0=py37_1
  - defusedxml=0.5.0=py37_1
  - descartes=1.1.0=py37_0
  - entrypoints=0.3=py37_0
  - expat=2.2.6=he6710b0_0
  - fiona=1.8.4=py37hc38cc03_0
  - fontconfig=2.13.0=h9420a91_0
  - freetype=2.9.1=h8a8886c_1
  - freexl=1.0.5=h14c3975_0
  - gast=0.2.2=py37_0
  - gdal=2.3.3=py37hbb2a789_0
  - geopandas=0.4.1=py_0
  - geos=3.7.1=he6710b0_0
  - geoviews=1.6.2=py_0
  - geoviews-core=1.6.2=py_0
  - gflags=2.2.2=hf484d3e_1001
  - giflib=5.1.4=h14c3975_1
  - glib=2.56.2=hd408876_0
  - glog=0.3.5=hf484d3e_1001
  - gmp=6.1.2=h6c8ec71_1
  - grpcio=1.16.1=py37hf8bcb03_1
  - gst-plugins-base=1.14.0=hbbd80ab_1
  - gstreamer=1.14.0=hb453b48_1
  - h5py=2.9.0=py37h7918eee_0
  - hdf4=4.2.13=h3ca952b_2
  - hdf5=1.10.4=hb1b8bf9_0
  - heapdict=1.0.0=py37_2
  - holoviews=1.11.3=py_0
  - hvplot=0.4.0=py_0
  - icu=58.2=h9c2bf20_1
  - idna=2.8=py37_0
  - imageio=2.5.0=py37_0
  - intel-openmp=2019.3=199
  - ipykernel=5.1.0=py37h39e3cac_0
  - ipython=7.1.1=py37h39e3cac_0
  - ipython_genutils=0.2.0=py37_0
  - ipywidgets=7.4.2=py37_0
  - jedi=0.13.3=py37_0
  - jinja2=2.10.1=py37_0
  - jpeg=9c=h14c3975_1001
  - json-c=0.13.1=h1bed415_0
  - jsonschema=3.0.1=py37_0
  - jupyter=1.0.0=py37_7
  - jupyter_client=5.2.4=py37_0
  - jupyter_console=6.0.0=py37_0
  - jupyter_contrib_core=0.3.3=py_2
  - jupyter_contrib_nbextensions=0.5.1=py37_0
  - jupyter_core=4.4.0=py37_0
  - jupyter_highlight_selected_word=0.2.0=py37_1000
  - jupyter_latex_envs=1.4.4=py37_1000
  - jupyter_nbextensions_configurator=0.4.1=py37_0
```

```
- jupyterlab=0.35.4=py37hf63ae98_0
- jupyterlab_server=0.2.0=py37_0
- kealib=1.4.7=hd0c454d_6
- keras=2.2.4=0
- keras-applications=1.0.7=py_0
- keras-base=2.2.4=py37_0
- keras-preprocessing=1.0.9=py_0
- kiwisolver=1.0.1=py37hf484d3e_0
- krb5=1.16.1=h173b8e3_7
- libboost=1.67.0=h46d08c1_4
- libcurl=7.64.0=h20c2e04_2
- libdap4=3.19.1=h6ec2957_0
- libedit=3.1.20181209=hc058e9b_0
- libffi=3.2.1=hd88cf55_4
- libgcc-ng=8.2.0=hdf63c60_1
- libgdal=2.3.3=h2e7e64b_0
- libgfortran-ng=7.3.0=hdf63c60_0
- libkml=1.3.0=h590aaf7_4
- libnetcdf=4.6.1=h11d0813_2
- libpng=1.6.37=hbc83047_0
- libpq=11.2=h20c2e04_0
- libprotobuf=3.7.1=h8b12597_0
- libsodium=1.0.16=h1bed415_0
- libspatialindex=1.8.5=h20b78c2_2
- libspatialite=4.3.0a=hb08deb6_19
- libssh2=1.8.0=h1ba5d50_4
- libstdcxx-ng=8.2.0=hdf63c60_1
- libtiff=4.0.10=h9022e91_1002
- libuuid=1.0.3=h1bed415_2
- libxcb=1.13=h1bed415_1
- libxml2=2.9.9=he19cac6_0
- libxslt=1.1.33=h7d1a2b0_0
- llvmlite=0.28.0=py37hd408876_0
- locket=0.2.0=py37_1
- lxml=4.3.2=py37hefd8a0e_0
- lz4-c=1.8.3=he1b5a44_1001
- mapclassify=2.0.1=py_0
- markdown=3.0.1=py37_0
- markupsafe=1.1.1=py37h7b6447c_0
- matplotlib=3.0.3=py37h5429711_0
- mistune=0.8.4=py37h7b6447c_0
- mkl=2019.3=199
- mkl_fft=1.0.12=py37ha843d7b_0
- mkl_random=1.0.2=py37hd81dba3_0
- mock=2.0.0=py37_0
- msgpack-python=0.6.1=py37hfd86e86_1
- multipledispatch=0.6.0=py37_0
- munch=2.3.2=py37_0
- nbconvert=5.4.1=py37_3
```

```
  - nbformat=4.4.0=py37_0
  - ncurses=6.1=he6710b0_1
  - netcdf4=1.4.2=py37h808af73_0
  - networkx=2.2=py37_1
  - ninja=1.9.0=py37hfd86e86_0
  - nodejs=10.13.0=he6710b0_0
  - notebook=5.7.8=py37_0
  - numba=0.43.1=py37h962f231_0
  - numpy=1.16.3=py37h7e9f1db_0
  - numpy-base=1.16.3=py37hde5b4d6_0
  - olefile=0.46=py37_0
  - openjpeg=2.3.0=h05c96fa_1
  - openssl=1.1.1b=h14c3975_1
  - owslib=0.17.1=py_0
  - packaging=19.0=py37_0
  - pandas=0.24.2=py37he6710b0_0
  - pandoc=2.2.3.2=0
  - pandocfilters=1.4.2=py37_1
  - panel=0.4.0=0
  - param=1.8.2=py_0
  - parquet-cpp=1.5.1=2
  - parso=0.3.4=py37_0
  - partd=0.3.10=py37_1
  - pbr=5.1.3=py_0
  - pcre=8.43=he6710b0_0
  - pexpect=4.6.0=py37_0
  - pickleshare=0.7.5=py37_0
  - pillow=6.0.0=py37h34e0f95_0
  - pip=19.1=py37_0
  - pixman=0.38.0=h7b6447c_0
  - poppler=0.65.0=h581218d_1
  - poppler-data=0.4.9=0
  - proj4=5.2.0=he6710b0_1
  - prometheus_client=0.6.0=py37_0
  - prompt_toolkit=2.0.9=py37_0
  - protobuf=3.7.1=py37he6710b0_0
  - psutil=5.6.2=py37h7b6447c_0
  - psycopg2=2.7.6.1=py37h1ba5d50_0
  - ptyprocess=0.6.0=py37_0
  - pyarrow=0.13.0=py37h0978efd_0
  - pycparser=2.19=py37_0
  - pyct=0.4.6=py37_0
  - pyepsg=0.4.0=py37_0
  - pygments=2.3.1=py37_0
  - pykdtree=1.3.1=py37hdd07704_2
  - pyopenssl=19.0.0=py37_0
  - pyparsing=2.4.0=py_0
  - pyproj=1.9.6=py37h14380d9_0
  - pyqt=5.9.2=py37h05f1152_2
  - pyrsistent=0.14.11=py37h7b6447c_0
  - pyshp=2.0.1=py37_0
  - pysocks=1.6.8=py37_0
  - python=3.7.3=h0371630_0
  - python-dateutil=2.8.0=py37_0
  - pytorch-cpu=1.1.0=py3.7_cpu_0
  - pytz=2019.1=py_0
  - pyviz_comms=0.7.0=py37_0
  - pywavelets=1.0.2=py37hdd07704_0
  - pyyaml=5.1=py37h7b6447c_0
  - pyzmq=18.0.0=py37he6710b0_0
  - qt=5.9.7=h5867ecd_1
  - qtconsole=4.4.3=py37_0
  - re2=2019.04.01=he1b5a44_0
  - readline=7.0=h7b6447c_5
  - requests=2.21.0=py37_0
  - rise=5.5.0=py37_0
  - rtree=0.8.3=py37_0
  - scikit-image=0.14.2=py37he6710b0_0
  - scikit-learn=0.20.3=py37hd81dba3_0
  - scipy=1.2.1=py37h7c811a0_0
  - send2trash=1.5.0=py37_0
  - setuptools=41.0.1=py37_0
  - shapely=1.6.4=py37h86c5351_0
  - sip=4.19.8=py37hf484d3e_0
  - six=1.12.0=py37_0
  - snappy=1.1.7=hf484d3e_1002
  - sortedcontainers=2.1.0=py37_0
  - sqlalchemy=1.3.1=py37h7b6447c_0
  - sqlite=3.28.0=h7b6447c_0
  - tblib=1.3.2=py37_0
  - tensorboard=1.13.1=py37hf484d3e_0
  - tensorflow=1.13.1=mkl_py37h54b294f_0
  - tensorflow-base=1.13.1=mkl_py37h7ce6ba3_0
  - tensorflow-estimator=1.13.0=py_0
  - termcolor=1.1.0=py37_1
  - terminado=0.8.1=py37_1
  - testpath=0.3.1=py37_0
  - thrift-cpp=0.12.0=h0a07b25_1002
  - tk=8.6.8=hbc83047_0
  - toolz=0.9.0=py37_0
  - tornado=6.0.2=py37h7b6447c_0
  - traitlets=4.3.2=py37_0
  - urllib3=1.24.1=py37_0
  - wcwidth=0.1.7=py37_0
  - webencodings=0.5.1=py37_1
  - werkzeug=0.15.2=py_0
  - wheel=0.33.1=py37_0
  - widgetsnbextension=3.4.2=py37_0
  - xarray=0.11.3=py37_0
  - xerces-c=3.2.2=h780794e_0
  - xz=5.2.4=h14c3975_4
  - yaml=0.1.7=had09818_2
  - zeromq=4.3.1=he6710b0_3
  - zict=0.1.4=py37_0
  - zlib=1.2.11=h7b6447c_3
  - zstd=1.3.3=1
  - pip:
    - atomicwrites==1.3.0
    - colorama==0.4.1
    - dask==1.1.0
    - distributed==1.25.0
    - filelock==3.0.10
    - flatbuffers==1.11
    - funcsigs==1.0.2
    - jupyterlab-latex==0.4.1
    - more-itertools==7.0.0
    - pluggy==0.11.0
    - py==1.8.0
    - pytest==4.4.2
    - redis==3.2.1
    - setproctitle==1.1.10
    - typing==3.6.6
prefix: /home/marcusskky/.conda/envs/EY
```

Model

```python
import os
import abc
import datetime
from toolz.curried import *
import pandas as pd
import holoviews as hv
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures, \
                                  StandardScaler, \
                                  FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline, make_union


def Imputer(model_: abc.ABCMeta,
            data_: pd.DataFrame,
            X_:  list, y_: list)-> pd.DataFrame:

    def predict(model__: abc.ABCMeta,
                data__: pd.DataFrame,
                X__:  list, y__: list) -> pd.np.array:
```

```python
        not_nul = data_.loc[:,X___+ y___].notnull().all(1)

        model_instance = model___.fit(X = data___.loc[not_nul,X___],
                                      y = data___.loc[not_nul,y___].values.ravel())

        return model_instance.predict(X=data_.loc[:,X___])


    return data_.assign(**{ys+'_predicted': predict(model___ = model_,
                                                    data___ = data_,
                                                    X___ = X_,
                                                    y___ = list(ys))
                           for ys in y_})

def basis(x:pd.np.array, knot:float = 12*60*60)-> pd.np.array:
    x = x - knot
    x = x * (x>0.0)
    x = x**2 * pd.np.log(x, where= x > 0.0)

    return x.astype(pd.np.float64)


if __name__ == '__main__':
    path = os.path.join('Data','data_test.csv')
    data = pd.read_csv(path, index_col = 'Unnamed: 0')

    id_ = ['hash','trajectory_id','vmax','vmin','vmean']
    midnight = pd.to_datetime('00:00:00')

    views = data.pipe(partial(pd.wide_to_long,
                              stubnames = ['x','y','time'],
                              i = id_,
                              j = 'pop',
                              sep = '_',
                              suffix = '(!?entry|exit)'))\
                .reset_index()\
                .replace(['entry','exit'],[0,1])\
                .assign(time = lambda d: d.time\
                                          .pipe(pd.to_datetime)\
                                          .subtract(midnight)\
                                          .dt.total_seconds(),
                        missing = lambda d: d.x.isna().astype(pd.np.int))

    pipeline = make_pipeline(make_union(PolynomialFeatures(degree=3, include_bias=False),
                                        FunctionTransformer(basis,
                                                            validate=False)
                                          ),
                                         linear_model.BayesianRidge(fit_intercept=True)
                                          )

    imputed = views.groupby(['hash'])\
                              .apply(lambda g: Imputer(model_=pipeline, data_=g, X_=['time'], y_=['y','x']))\
                              .reset_index(drop=True)

    imputed_path = pipe(datetime.datetime.now().timestamp(),
                                          partial(round, ndigits=0),
                                          int,
                                          lambda x: os.path.join('Data',f'imputed_{x}.csv'))
    imputed.to_csv(imputed_path, index=False, sep=',')

    quantile = 0.25
    output_path = pipe(datetime.datetime.now().timestamp(),
                                          partial(round, ndigits=0),
                                          int,
                                          lambda x: os.path.join('Data',f'submission_{x}.csv'))

    imputed.assign(city = lambda d: ((d.x_predicted <d.x.quantile(0.5+quantile)) &\
                                                         (d.x_predicted >d.x.quantile(0.5-quantile)) &\
                                                         (d.y_predicted <d.y.quantile(0.5+quantile)) &\
                                                         (d.y_predicted >d.y.quantile(0.5-quantile))).astype(pd.np.int))\
                    .where(lambda x: x.missing==1)\
                    .dropna(axis=0, how='all')\
                    .loc[:,['trajectory_id','city']]\
                    .rename(columns={'trajectory_id':'id','city':'target'})\
                    .assign(target = lambda x: x.target.astype(int))\
                    .to_csv(output_path, index=False, sep=',')
```

Bibliography

Section 3.3 in Christopher M. Bishop: Pattern Recognition and Machine Learning, 2006

David J. C. MacKay, Bayesian Interpolation, 1992.

Michael E. Tipping, Sparse Bayesian Learning and the Relevance Vector Machine, 2001.