DCLG Fire Data Project

DCLG are responsible for the Incident Reporting System (IRS) which houses all of the fire related callout data across England and Wales. This system is currently undergoing a redesign with an external SME who are looking at questions relating to both IT architecture and front-end capabilities.

The scope of this alpha project can be split into two distinct parts:

- Look at existing data and explore the potential for new ways to visualise it. This would ideally cover both open and restricted data.

- Study the extent to which machine learning techniques can be applied to help build a predictive model at the (London) ward level. This should ideally look to highlight key demographic/geographic factors in conjunction with operationally relevant day-to-day incident level predictions.

This document aims to give an overview of the completed prototype with summarised technical details for both of the above strands. Unless otherwise stated, London and its associated boroughs/wards are used throughout in order to provide an active yet manageable test case to the methods used. An effort will be made to publish all code mentioned throughout in an attempt to provide departmental analysts/statisticians with the means to validate and - if appropriate - continue the proof-of-concept work.

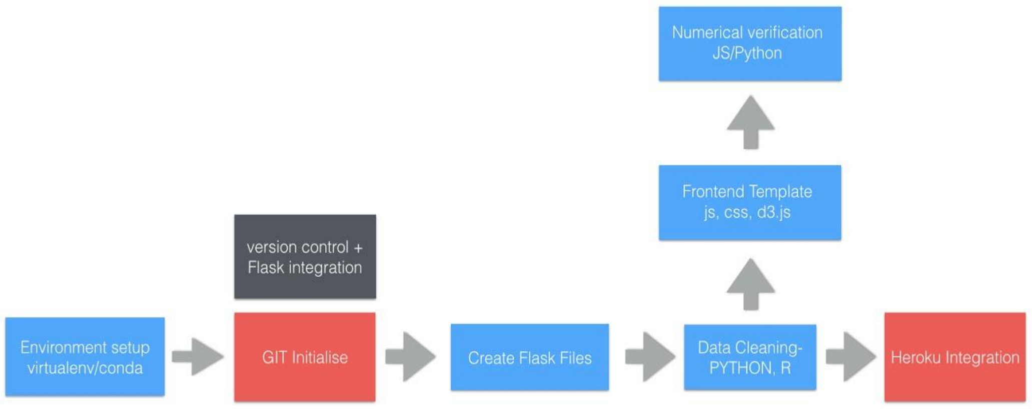
Each of the two main sections will start with a technique summary which highlights the main code libraries/languages/open source tools relevant to the piece of work. All software is free to use* and have active online support communities.

I. Data Visualisation and Interactive Queries (live)

- Data preparation/cleaning and exploratory analysis
  - Pandas (Python library)
  - R (statistical language + ggplot for static plots)
  - Matplotlib (basic plotting and visualisation for Python)

- Web development and hosting
  - Flask (Python web application framework with Jinja2 template engine)
  - Git (distributed version control software)
  - Heroku /Amazon AWS (hosting)
  - d3.js, Crossfilter and dc.js (frontend design, visualisations)
  - Twitter Bootstrap (page layout and basic CSS/Javascript)

The convenience of using London as a test case lies in the fact that a certain amount of data is already made open on www.data.gov.uk. This flat csv file gives postcode districts grouped in the form of wards and boroughs in addition to several other fields such as response time, type of alarm and number of vehicles in attendance.


Numerical verification JS/Python
Frontend Template js, css, d3.js
version control + Flask integration
Environment setup virtualenv/conda
GIT Initialise
Create Flask Files
Data Cleaning- PYTHON, R
Heroku Integration

*Heroku will charge for hosting depending on traffic volume**.**

Figure.1 Rough outline of key project steps. Starting with a clean development environment is recommended so as to keep project-specific dependencies separate from system-wide versions.

Data Cleaning and Preparation

As the initial work was concerned with the London Fire Brigade (LFB) 2012 data, minimal cleaning was required. The Pandas python library is similar to the R language in many respects with the primary (high level) data structure being the dataframe which can robustly handle different data types across multiple columns. Convenience functions are provided which allow the user to read csv, txt, xls and other file types directly allowing advanced queries to be performed efficiently. In this part of the project Pandas (in addition to Matplotlib) was used as an exploratory tool to both analyse and visualise the data before working on the javascript implementation. They also proved useful in verifying distributions resulting from the javascript filtering.

Webpage Layout

For the frontend layout Twitter Bootstrap was used which provides a backbone for the core CSS and Javascript elements. In the data explorer I have used one of the basic templates with a flexible container to house the multiple charts. The Bootstrap approach is centred around dividing a page into a discrete grid pattern which simplifies the placement of multiple **<div>** and helps make the page responsive for mobile/tablet viewing.

Crossfilter, d3.js and dc.js

The visualisation relies heavily on these three interlinked javascript libraries. Crossfilter was developed to address performance issues with querying large, multivariate datasets in browser. Several SQL-style queries can be implemented through this library which specialises in incremental filtering and reducing of data on the tens of milliseconds scale. dc.js is built on top of Crossfilter and acts as a bridge allowing users to harness the visualisation power of d3.js with the speed and efficiency of the Crossfilter method. This enables the user to link multiple chart types which automatically update according to the position of seemingly independent filters.

A strength of this approach lies in the incorporation of a choropleth showing geospatial patterns in the data. The fact that this too can update according to other user-defined filters means that outliers in the data are much easier to detect. Similarly, the compact nature of this type of tool means that complicated queries are both intuitive and dynamic. It removes the overused pattern of multiple dropdown menus to select increasingly specialised cuts of data and allows the patterns within to guide further enquiry. The importance of such responsiveness cannot be overstated and similar functionality cannot be replicated by more rudimental binary selectors (e.g. radio buttons)

Hosting and Deployment

The data explorer has a basic design with rudimental multipage support for additional content. The python based Flask framework allows simple address/URL routing and password protection to be implemented quickly and there is direct support for hosting projects on cloud based platforms. At its simplest, Flask reduces the web developers task to defining two distinct areas:

1. **Python Index File** handles page/URL routing, serves templates and executes python commands (database calls, data manipulation etc.) - SERVER SIDE

```
@app.route('/contact')
def contact():
    return render_template('contact.html', title='contact')
```

2. **Template Files** - The html rendered when a page is visited (contact.html in above example). Variables from the index file can be passed to the template and displayed on the page - CLIENT SIDE

It is recommended that interested readers consult the Flask project page for additional details and user-friendly tutorials.

Once the Flask app has been created it can be run locally on a computer which allows the developer to test and showcase early work. If server hosting is required, cloud solutions such as Heroku and Amazon AWS are fully supported and robust solutions that can handle production level code. Detailed instructions for deploying apps to these platforms is out of the scope of this summary however there exists several online tutorials which are tailored to the user's programming language of choice.

A useful side effect of using a platform such as Heroku is the necessary link to version control software Git. Each project is linked with a Git repository where changes are made and pushed to Heroku for deployment. This encourages good practice and provides a useful backup of codebase.

Limitations and Further Work

The prototype front-end currently uses a single year of LFB data which is loaded into the browser in almost raw format. This is inefficient and slows load time as the file size is approximately 18MB. This could be reduced through simple encoding of the file which contains verbose column entries (e.g. 'property type' field). Despite this file size, the data selection mechanism is still reasonably responsive once the data has loaded and it is certainly possible to extend the dataset to multiple years and/or larger geographic areas.

II. Predictive Analytics and Nonparametric Methods

The second part of this project is concerned with investigating to what extent we are able to predict total fire numbers across London's wards based solely on demographic, socioeconomic and other geographic and infrastructure data. This analysis is performed on the LFB 2012 data as this contains incident information down to the ward level which is convenient when merging other datasets with this degree of aggregation.

This section will provide an overview of the modelling techniques used in this project however it is not the aim of this work to include a thorough background on the concepts mentioned. Key representative results are presented and readers are encouraged to contact the author for additional information.

Ward Characteristics

In order to build a statistical model which is robust enough to predict total annual incidents at the ward level it is important to have a comprehensive list of features that help define (and differentiate) one area from another. A reasonable proportion of this data can be obtained from the latest 2011 census however this is not a comprehensive survey and additional information needs to be included from other sources such as ONS and Land Registry. The London datastore contains a ward-atlas dataset which compiles a list of approximately 150 indicators which I have used as a basis for model formation. Approximately 80 features have been used in the initial set with the aim of performing dimensionality reduction to reduce the number to those most effective in explaining the observed variations in incident number.

Feature Selection

Several distinct approaches to model selection were trialled in parallel during the course of this work however I will report on two of the most representative examples here which fall into the parametric and nonparametric families.

The topic of feature selection can be a somewhat contentious issue due to the potential for over reliance on automated techniques to include/remove features from a predictive model. This is a general concern in statistical work and merely restates the need to gain some prior understanding of the problem before resorting to numerical benchmarking. The two techniques I will outline here reflect vastly different approaches to the same problem.

Random Forests (RF)

An inherently nonparametric method which relies on the ability of an ensemble of decision trees to both learn patterns in data and predict new events. This class of machine learning algorithm has gained popularity for its strength as a classifier however it can also be used for regression tasks in which the dependent variable (incidents) is continuous. The term ensemble indicates that a collection of decision trees are used as 'weak learners' which are combined in an ensemble to form a 'strong learner'. The algorithm can be seen as follows:

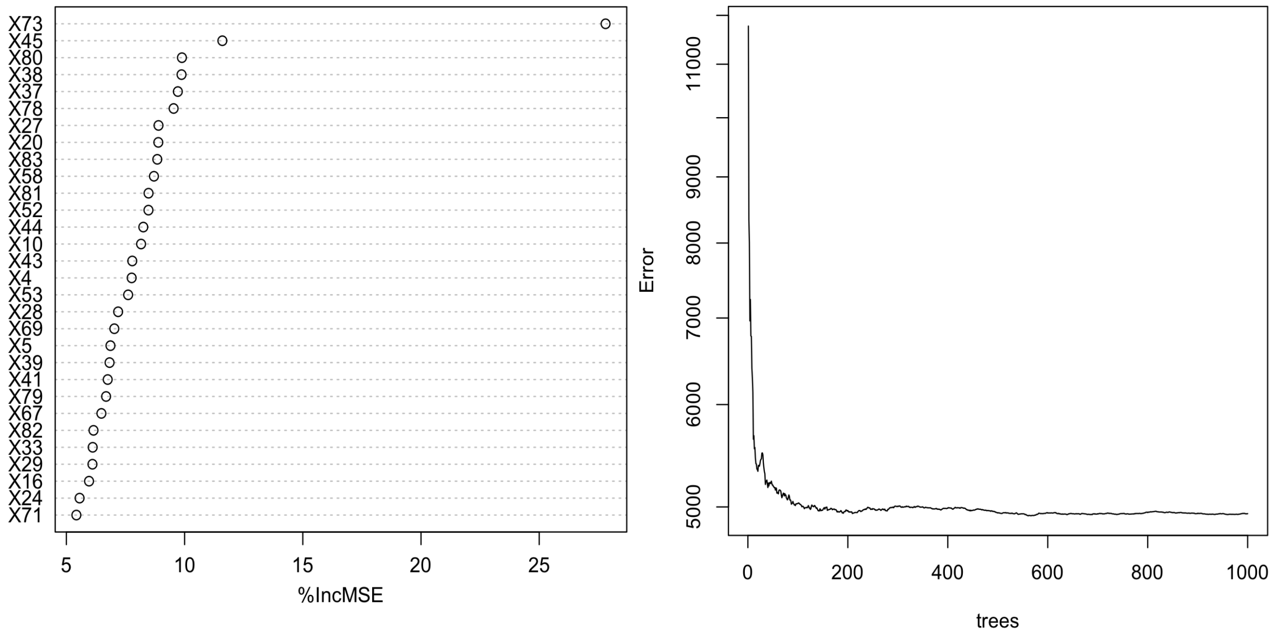


Figure 2. Two plots of the output from the random forest modelling. The left plot shows the relative variable importance (encoded in numbers for ease of display) and the right plot highlights the relationship between prediction accuracy and tree number in forest.

1. Take a random sample (60-70% with
2. replacement) of the complete dataset to create a subset.
3. At each tree node:
    - Select a random sample (size *m*) of the features/independent variables.
    - The feature and value that allow the best binary split (according to some objective function) is used to separate the data
    - At the next node, choose another m features and repeat.

the value *m* can be chosen in various ways however a typical value is $\sqrt{t}$ where t is the total number of features.

In a typical RF one may use several thousand such complete trees and then use a weighted average (or voting majority for categorical dependent variables) to determine the value of a new input value.

RFs are often viewed as a workhorse method of exploratory analysis as they are particularly robust especially when dealing with missing values, unscaled/non-centred data and multicollinearity. As with most techniques, it is not perfect; being vulnerable to overfitting noisy data and not able to predict new values outside of the training data range.

In our case we wish to use the RF model to perform feature selection/dimensionality reduction for us. Luckily, this is straightforward as it is possible to extract the key features used at each node split and produce a weighted impression of which ones tended to minimise the error on predictions. **Figure 2** shows both a typical variable importance plot in terms of impact on means squared error (MSE) for the model predictions and the relationship between tree number and prediction accuracy.

It is important to note that random forests are - as the name suggest - inherently random. Whenever they are used for these tasks it is worth running the model several times in order to assess the stability of the results and tweak parameters where necessary.

As can be seen from Figure 2. There is one feature in particular which accounts for a significant proportion of the variation from ward to ward. The features have been encoded for ease of display however the top 5 are:

1. Total Crime Rate 2011/12
2. %OwnedMortgageOrLoan
3. All Household Spaces 2011
4. %Flat_maisonette_apartment
5. Cars Per Household

The compiled dataset has a reasonably large feature space however it exhibits a great degree of collinearity. On modelling, the top five tends to have certain features interchanged such as All household spaces2011 and All dwellings2011. Replacing these with similar features results in no significant changes to the model accuracy.

NOTES

LASSO - regression - penalised - BOOTSTRAP - R^2 ~ 0.6 [18 non zero take top 5 (modulus)] Explains significant variation.

Feed top 5 into normal linear model report R2_adj very similar to R2.

Show plot of fit with y = x and the actual series.

Quote accuracy (MSE ~ 4000) R2 ~ 0.6

mention PCA

Daily variation:

Weather - no effect
Poisson process - confidence intervals - negative binomial
further work needed