

Final Report: Predicting Future Crime Activity Utilizing Machine Learning and Flickr Metadata

James White
Cardiff University
School of Computer Science and Informatics
CM3203
Project No: 105
Supervised by: Steven Schockaert
Moderated by: Irena Spasic

May 6, 2014

Abstract. The aim of this project is to tackle the incredibly current affair of predictive policing, the practice of forecasting criminal activity prior to it happening. This process will be attempted through a combination of machine learning techniques and the utilization of tags extracted from photos posted on Flickr. The implementation of Chi-square feature selection will identify location-relevant terms, which could indicate reasons for criminal activity in a particular demographic. This report has been created to document the process of approaching, implementing and analysing the solution to this task.

1 Acknowledgements

I would like to express my appreciation and gratefulness to Dr. Steven Schockaert for his continued support and guidance throughout the length of the project. His insight was incredibly useful and played an important part in helping me understand some aspects of the research. I would also like to acknowledge Weka (Waikato Environment for Knowledge Analysis) for providing a suite of machine learning tools that were vital for my solution.

Contents

1	Acknowledgements	2
2	Introduction	6
2.1	Aims	6
2.2	Audience & Beneficiaries	6
2.3	Project Scope	6
2.4	Approach	7
2.5	Important Outcomes	8
3	Background	9
3.1	Predictive Policing	9
3.2	Weka	9
3.3	Linear Regression	10
3.4	Flickr API	10
3.5	Chi-Square Term Selection	11
4	Specification & Design	12
4.1	General Overview	12
4.2	Producing a Database from Police Data	13
4.3	Design: Flickr Tag Scraper	14
4.4	Design: Chi-Squared Calculation Program	15
5	Implementation	16
5.1	Performing Queries on the Database	16
5.2	Compiling a Data Set for Linear Regression	17
5.3	Test Options for Implementing Linear Regression	19
5.4	Identifying Regions of High and Low Crime	20
5.5	Scraping Tags via the Flickr API	22
5.5.1	Form on the HTML Page	22
5.5.2	Function to Build Request URL	23
5.5.3	Function to Search Photos	23
5.5.4	Acquiring Tags from Photos	24
5.5.5	Downloading Tag Clusters to Local Machine	25
5.6	Calculating Spatially Relevant Terms	27
5.6.1	Compiling the Input Files	27
5.6.2	Reading Tag-Clusters into Hash Maps	28
5.6.3	Sorting Hash Maps of Tags	29
5.6.4	Constructing the Chi-Squared Formula	29
6	Constraints & Unforeseen Circumstances	31
6.1	Flickr API Request Limit	31
6.2	Identifying Locations Within Police Data	31
6.3	Maximum Photos Returned Per-Page by Flickr API	32
6.4	Issues With Searching for Photos via LSOA	32

7	Analysis of Results	33
7.1	Results of Linear Regression in Weka	33
7.1.1	Estimating the Total Number of Burglaries	33
7.1.2	Combining Multiple Features	36
7.2	Tag Clusters Scraped from Flickr	38
7.2.1	Regions with Low Crime Ratio	38
7.2.2	Regions with High Crime Ratio	40
7.3	Can These Tags Aid in Making Predictions?	41
8	Strengths & Weaknesses of Current Implementation	42
8.1	Strengths	42
8.2	Weaknesses	42
9	Future Work	43
10	Project Conclusions	44
10.1	Have significant crime statistics been gathered to form the basis behind a predictive model?	44
10.2	Has a vast collection of tags been scraped from photos in specific locations via Flickr?	44
10.3	Has term selection been applied to identify spatially-relevant tags and remove redundant or irrelevant ones?	44
10.4	Has it been possible to build a regression model that can predict the number of crimes in some location?	44
10.5	Can tags from Flickr be used to improve the accuracy of a predictive model?	45
10.6	Summary	45
11	Reflection on Learning	46
11.1	Improvement to Programming Skills	46
11.2	Attention to Mathematical Theory	46
11.3	Handling Big Data	46
11.4	Decision Making & Approach	47
11.5	Overview	47
12	Glossary	48
13	Table of Abbreviations	49
14	Appendices	49
	References	50

List of Figures

1	Agile Software Development Methodology	7
2	Example Format of ARFF file	9
3	General Overview of System Operations	12
4	Database Table Column Headings	13
5	Effect of Removing Trailing LSOA Code Digits	13
6	Design Overview of Flickr Tag Scraper	14
7	Design Overview of Flickr Tag Scraper	15
8	Example Chi-Square Calculation Output	15
9	Example SQL Query Performed on Database	16
10	Example Output of SQL Query	16
11	Example Input Features for Linear Regression	17
12	Overview of the Data Preprocess Tab in Weka	18
13	Example Visualization of Data in Weka	18
14	Classification Interface for Linear Regression in Weka	19
15	Visualization of Lower Super Output Area Population Estimates	20
16	Subset of Regions with High and Low Crime Rates	21
17	Capturing User-Entered Variables	22
18	Build URL Function	23
19	Function to Search Photos	23
20	Function to Check All Photos are Scraped	24
21	Function to Get Tags From Photos	24
22	Structure of the Scraped Object	25
23	Function to Download Tags to Users Machine	26
24	Function to Download Tags to Users Macine	26
25	Example Folder Containing Monthly Tag Clusters	27
26	Reading Tags and Separating via Commas	28
27	Reading Tags and Separating via Commas	28
28	Sorting Hash Maps via Observed Frequencies	29
29	Calculation of Observed and Expected Frequencies in Java Program	29
30	Calculation of Chi-Square Formula	30
31	Example Output of Chi-Squared Calculation	30
32	Regression Model for Predicting Burglary Counts	33
33	Counts of Burglary in Cardiff	34
34	Example Calculation utilizing Linear Regression Model	34
35	Using Less Input Features for Linear Regression	35
36	Analyzing the Success of Model-Based Predictions	36
37	Success of Predictive Model Against Alternative Method	37
38	Example Redundant Tags from Chi-Squared Test	38
39	Top Tags for Regions with Low Crime Ratio	39
40	Top Tags for Regions with High Crime Ratio	40
41	Occurrence of Tag in Relation to Crimes in Blackpool	41

2 Introduction

2.1 Aims

The primary aim of this project is to make estimations regarding criminal activity in a particular demographic. This involves attempting to identify numbers of likely crime rates for locations in the UK. These numbers will be calculated via machine-learning techniques provided by Weka, such as the linear regression function. Building a model that can make such predictions requires large sets of data, for which publicly available police statistics [1] have been utilized. The second contribution to this project involves gathering tags from photos on Flickr, and analyzing the data in order to find spatially-relevant terms. This process will be achieved by implementing the Chi-square feature selection technique on a large set of tags. Having suitable data sets for this project was marked as largely significant in the aims of the initial report. The integration of social media data with historical crime statistics aims to deliver extra insight into making predictions. Some tags may help to explain reasons for a particular crime-shift or paradigm, due to their frequency in locations with contrasting crime rates. By assessing the predictive model prior to incorporating Flickr tags, it will be possible to draw conclusions at a later date, as to whether Flickr metadata can be utilized for greater precision.

2.2 Audience & Beneficiaries

Considering the surge in predictive policing, in particularly the possibility of the metropolitan police introducing such systems [2], it is hoped that this project will prove the success the process can achieve. For this purpose, policing and security companies are the primary beneficiaries being considered while investigating this system. This report is intended to act as a useful method of understanding the processes and techniques that have been implemented. For this reason, if somebody wishes to continue similar work or research, this document can provide a valuable insight to the problem. The ‘Future Work’ section of this report includes some potential ideas for furthering the possibilities of predicting crime.

2.3 Project Scope

In order to deliver the desired results from this project, there are a few essential elements that can be defined. It is vital that a database be created to store the large amount of police statistics that will be collected. This database will then be queried for specific numbers relating to crime in different locations across the UK. Also critical in this project, is a program that has the ability to gather large amounts of tags scraped from photos on Flickr. It is important these tags can be acquired from photos taken within specific locations. These tags will then be used as a basis for analyzing and identifying location-relevant terms that may suggest an explanation for crime patterns. One further program is necessary for the automation of chi-squared statistics. The values identified following implementation of the chi-squared test can be used to rank tags in terms of how discriminative they are to a location.

In order to utilize the machine learning tools provided with Weka, it is important that data sets are formatted according to the Attribute-Relation File Format (ARFF). This is the default accepted file format for the software, and is essentially an ASCII text file divided into two sections, the header segment and the data segment. Converting data sets into this format will allow the linear regression function to be applied, thus building a model that can forecast a prediction for some criminal activity.

2.4 Approach

In order to tackle this project it was important to follow a controlled approach for managing the task. I implemented the agile development methodology as a basis for my work. This methodology was applicable due to the nature of the task, and the fact my requirements from the system would frequently change depending on unforeseen constraints. Figure 1 details the general approach that was followed.

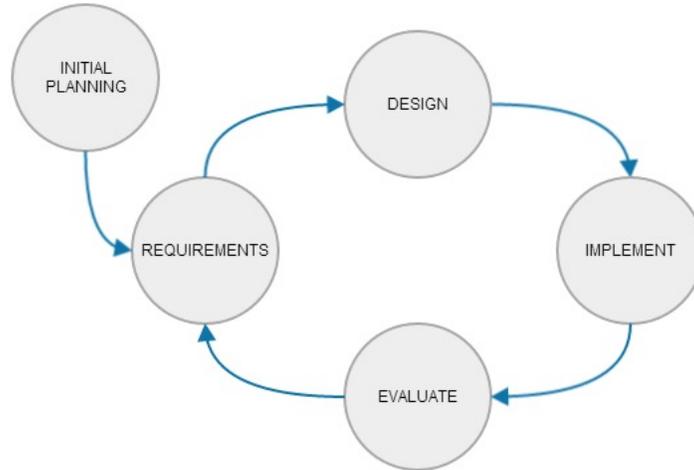


Figure 1: Agile Software Development Methodology

By compiling an initial report it was possible to outline initial planning and requirements. This acted as a precursor to any design and implementation stages, which could later enter as a round-about process whereby reviewing and evaluating results could lead to modifications at any stage. Agile software development offers flexibility for future adjustments, which was key due to the short time-length of this project [3]. It is an efficient approach that focuses on evolutionary development. Although the waterfall model was considered, it was disregarded due to the rigid structure not being suitable. Discovering a misjudgement or error in one of the latter stages of the waterfall model could have proved potentially disastrous. This is because the opportunity to make an alteration to an earlier stage is complicated if it has already been completed.

To ensure vital functionality was delivered primarily, the implementation stage focused on creating deliverables in the order they would be utilized. This meant the creation of a database that was populated with police data was tackled first, followed by the generation of a program to scrape tags from photos on Flickr, and finally the software to calculate Chi-square values and identify location-relevant tags. Once these tasks were complete and all data had been collected, it would allow for successful analysis of results. The opportunity to further evaluate and make changes to the implementation would then be possible. This is advantageous due to the strong probability of system requirements changing throughout the length of the project.

2.5 Important Outcomes

There are a number of important outcomes that will be assessed following the development of the project. In relation to the aims that this project seeks to deliver, the following questions are of most significance.

- Has sufficient data been collected as a basis for the techniques used in this project? Has this data been aggregated into a suitable database?
- Can predictions regarding counts of crime in forthcoming months be made for different locations in the UK? Are these estimations reasonable?
- Can location-relevant tags be identified through implementation of the Chi-Square term selection process?
- Do the location-relevant tags offer anything by way of improving estimations regarding future crime activity?
- Has this project been tackled with a professional approach using relevant theory and concepts where applicable? What changes could be made to a future implementation of the task?

3 Background

Research into the techniques explored in this project was essential for becoming aware of solutions that exist to similar problems. The following outlines theory and methods associated with the project, and why they are relevant and applicable.

3.1 Predictive Policing

Predictive policing is an incredibly current affair which aims to forecast criminal activity prior to the possibility of it happening. With already proven results in the US [4], it is estimated the LAPD's (Los Angeles Police Department) Foothill division have reduced crime in the first two months of 2014 by 30%, with burglary counts down by 55% respectively. Data mining techniques are utilized for large sets of historical data in an attempt to establish trends and patterns that can help identify criminal hot-spots. Machine learning is a form of artificial intelligence which focuses on trying to learn from some supplied training data. This particular approach is therefore especially useful in attempting to make future predictions based on historical data.

Most current implementations of predictive policing utilize the PredPol tool [5], which identifies 'boxes' of high risk areas to allow greater police presence in those regions. Previous crime records that detail the location and date of specific crimes are input to the system. Artificial intelligence regarding sociological routines of criminals also exists, such as the fact there are often repeat burglaries in the same area. The combination of various input data and knowledge is how predictions are reached. The PredPol tool has proven results, however it does not attempt to utilize social media to gain extra insight into potential criminal activity. This project will identify if spatially-relevant tags are a useful input feature for making estimations regarding crime.

3.2 Weka

Weka is an open-source software package available to download via the University of Waikato. The software provides a suite of machine learning algorithms and also aids in visualizing results from data mining. The functions provided with Weka can be employed directly upon a data-set, provided the data is supplied in ARFF (Attribute-Relation File Format). Below shows a simple ARFF file, whereby the attributes would be column headings in a csv file, and the data would be the rows of data that exist for the given attributes.

```
@relation BurglaryCounts2013
@attribute 'Location' Adur,Allerdale,Arun,Ashfield
@attribute BurglaryJan13 numeric
@attribute BurglaryFeb13 numeric
@attribute BurglaryMar13 numeric

@data Adur,25,24,19
Allerdale,40,30
Arun,76,63
Ashfield,98,84,86
```

Figure 2: Example Format of ARFF file

3.3 Linear Regression

The linear regression function is the key component to producing a prediction, such as the number of burglaries that will occur in one location in a particular month. The operation builds a regression model from a training data-set, that identifies a dependent variable amongst the input. The main advantage of linear regression is that the model can be utilized to calculate the value of a dependent variable that is not yet known. Various input features can be assessed in order to discover those that identify a strong relation to the predicted class. The major disadvantage of linear regression is that only linear relationships between independent and dependant variables are observed. This is an issue with some cases, such as the correlation between age and income. With increasing age it is reasonable to assume that financial income will also rise. However, it is most likely that income will then diminish when one retires. Fortunately, while crime rates are expected to fluctuate, they are not currently expected to suddenly decline.

As mentioned in the initial report, regression works ideally with numerical values. This means that the type of input features in regards to this project, would be counts of specific types of crime, taken within a specified time, in different regions of the UK. In order to try and achieve accurate predictions, multiple linear regression will be implemented whereby there are a number of independent input variables. Below details the theory of multiple linear regression.

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3.....b_nX_n$$

The formula displayed above shows the calculation for identifying Y, the predicted value for the dependent variable. b_0 identifies the intercept term, while X_1 to X_n are n independent variables, and b_1 to b_n are n predicted regression coefficients. The function operates by calculating the corresponding weights for variables and selecting those that reduce squared error on training data.

The process of obtaining data and applying linear regression has multiple stages. In order to create a model that can be implemented for the purpose of making predictions, the following steps must be taken. Initially, feature selection must be performed to obtain relevant input data, then the data must be preprocessed. Preprocessing involves ensuring quality of data, through the removal of redundant and noisy data. The data is then transformed for the purpose of applying the data-mining technique, in this case linear regression, in order to produce an interpretation.

3.4 Flickr API

The API provided by Flickr allows the extraction of information from photos that are published on the site. The API consists of a set of methods that can be utilized to make requests to an API endpoint. These requests will then receive a response that is formatted according to the calling convention that has been used. For the purpose of this project, REST and JSON formatting was implemented to make simple GET calls for specified data. In order to successfully retrieve a response it is necessary to bind a corresponding API key to all requests. The purpose of this key is so that Flickr can monitor application activity, and ensure that the service is not abused and thus ruined for all developers.

There are a number of possibilities provided via the Flickr API that made this project achievable. For example, the option to search for photos within a specified region was essential. Likewise, it was also significant to find photos that were either taken or uploaded within a

specified time frame. Lastly, the API made it possible to retrieve tags from photos whereby a user had chosen to include them.

3.5 Chi-Square Term Selection

X^2 term selection is a process that can be used to identify if there is some statistical significance between the occurrence of a tag t in some class c , and the number of times we would expect to see that tag in class c . This value is calculated in relation to other terms that appear both inside and outside of class c . By utilizing this process as a means of feature selection, it is possible to identify those terms which are most meaningful and potentially insightful of a particular location. Terms that have a high corresponding X^2 statistic are most spatially relevant to the location they have appeared in. It is hoped that identifying terms within a location with high corresponding X^2 value, could suggest relationships and reasons for particular crime in that area. For example, the persistent appearance of the tag ‘bmw’ in locations with high rates of vehicle crime. Below details the method for calculating the X^2 statistic [6].

$$X^2(t, c) = \frac{(O_{tc} - E_{tc})^2}{E_{tc}} + \frac{(O_{\bar{t}c} - E_{\bar{t}c})^2}{E_{\bar{t}c}} + \frac{(O_{t\bar{c}} - E_{t\bar{c}})^2}{E_{t\bar{c}}} + \frac{(O_{\bar{t}\bar{c}} - E_{\bar{t}\bar{c}})^2}{E_{\bar{t}\bar{c}}}$$

The values that appear in the formula above can be explained as follows in relation to this project.

- O_{tc} is the number of times tag t appears in location c 's tag cluster
- $O_{\bar{t}c}$ is the number of tags that are not t which appear in location c 's tag cluster
- $O_{t\bar{c}}$ is the number of times tag t appears in locations outside c 's tag cluster
- $O_{\bar{t}\bar{c}}$ is the number of times that tags which are not t appear outside location c 's tag cluster

E_{tc} is the occurrences of tag t we would **expect** to see in location c 's tag cluster. This value is calculated by multiplying:

- N - the total number of tags collected for all locations in the UK
- $P(t)$ - the probability of tag t appearing in any location's tag cluster. Identified by dividing the total occurrence of tag t , by N (the total number of tags).
- $P(c)$ - the probability of a tag appearing in location c 's tag cluster. Identified by dividing the total number of tags in location c , by N (the total number of tags).

Multiplying these three values will give an expected frequency of tag t in location c . We can therefore modify this formula to calculate the values detailed below necessary for the X^2 formula:

- $E_{\bar{t}c} = N * (1-P(t)) * P(c)$
- $E_{t\bar{c}} = N * P(t) * (1-P(c))$
- $E_{\bar{t}\bar{c}} = N * (1-P(t)) * (1-P(c))$

4 Specification & Design

4.1 General Overview

In order to construct a system that met the initial aspirations for this project, it was important to identify and design the key components to be created. To recognize important factors, it is necessary to consider the wider scope and general usability that the system must operate under. Considering the initial requirements that relate to predictions for different types of crime, and in different locations, it was understood that the system needed to be flexible. Essentially the opportunity to query and acquire data to personal requirements was incredibly significant. For example, if somebody was to enquire as to the number of vehicle crimes that will occur next month in Cardiff, the system should be able to gather required vehicle-crime statistics from the database, scrape a cluster of tags from photos taken in Cardiff, and aggregate this data as appropriate attributes for linear regression in Weka. As previously mentioned, in order to produce a flexible system with desired functionality, the key components were identified as:

- a database to hold vast amounts of police data and statistics.
- a program to scrape tags from photos taken within specified locations and date periods.
- a program to analyze and identify spatially relevant terms through implementation of X^2 term selection.

Below details the general overview of the system and how data is collated and utilized:

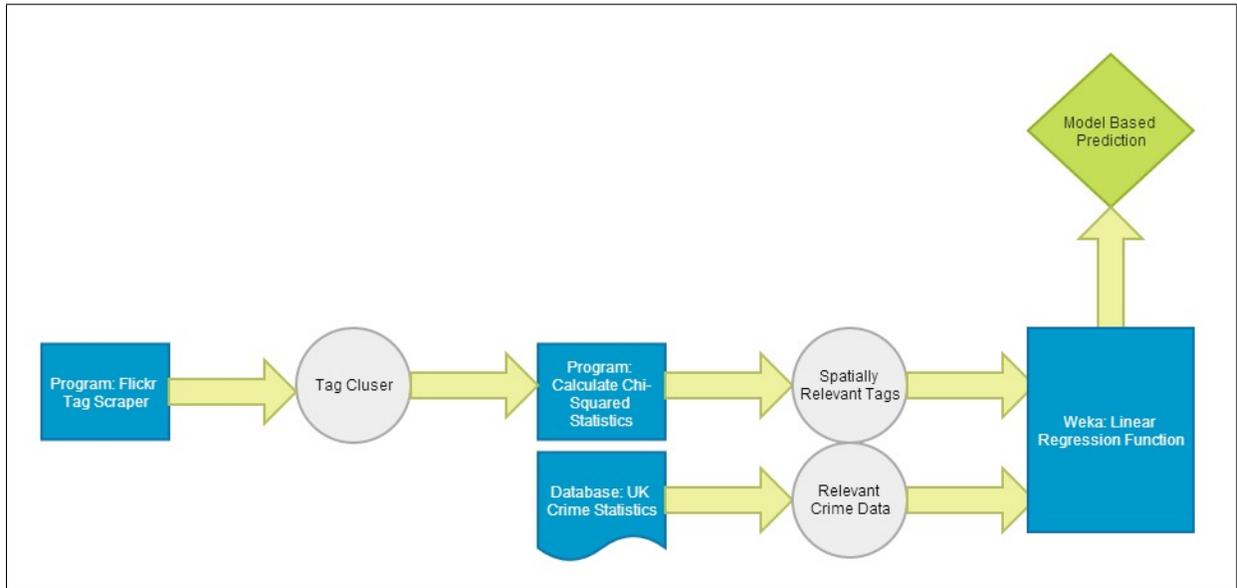


Figure 3: General Overview of System Operations

This figure details how tag-clusters for specified locations can be gathered and then analyzed to find spatially-relevant terms. In conjunction with historical statistics residing in the database, this information can then be used to reach some estimation or conclusion.

4.2 Producing a Database from Police Data

In order to be able to query a database for counts of different crime types across the UK, it had to be created and populated with relevant data. This was necessary in order to obtain numerical values regarding crime, which could then be input as features for the linear regression model. It was possible to download all police forces monthly crime records, which identify each recorded violation individually. Each record has multiple associated columns, such as the month the crime was committed in, the location, and the crime type it falls under.

The chosen program to manage the database and execute queries was SQLiteStudio (v2.1.5) due to the possibilities of importing and exporting data, while also having no limit on the size of databases or tables. Multiple tables were created to store police data on a yearly basis, e.g. 'AllCrime2013' and 'AllCrime2012.' Into both of these tables, the import function was used to transfer all 45 police forces monthly data for the corresponding years. The finite column headings in the database can be seen in the figure below:

CrimeID	Month	Reported By	Falls Within	Longitude	Latitude	Location	LSOA Code	LSOA Name	Crime Type	Last Outcome	Context
---------	-------	-------------	--------------	-----------	----------	----------	-----------	-----------	------------	--------------	---------

Figure 4: Database Table Column Headings

The decision not to remove any columns from the original police data relates back to the idea of having a flexible system. If some of the columns were removed because no initial use existed for them, it may prove problematic when wanting to improve or further add functionality to the system. Queries still perform within a reasonable time-frame, so there was no reason to remove any columns that were not of significance yet.

One aspect of the database that it was necessary to modify, was the LSOA (Lower Super Output Area) extension code that each 'LSOA name' field included. This extension code can be used to give the location of a crime greater accuracy within that area. Lower Super Output Area's are essentially bounding boxes used to define regions for the function of gathering national statistics. It caused issues when identifying unique counts of crime for each individual area. For example, wanting to know the total counts of burglary in Cardiff, not the total counts of burglary in Cardiff 003D, Cardiff 005D, Cardiff 007A...etc. The proposed solution to this issue was to execute an SQL query to remove the trailing 4-digit code in the LSOA Name field. The figure below shows the effect of the query:

```
UPDATE AllCrime2013
SET [LSOA Name] = substr([LSOA Name], 0, length([LSOA Name])-4)
```

Before	After
Cardiff 003D	Cardiff
Cardiff 005D	Cardiff
Cardiff 007A	Cardiff

Figure 5: Effect of Removing Trailing LSOA Code Digits

4.3 Design: Flickr Tag Scraper

The program described here can be found in Appendix B.

One of the most important parts to this project, was the development of software that would acquire tags taken from photos uploaded to Flickr. This would later allow for the identification of spatially-relevant terms. This would be achieved through the Flickr API, whereby JSON requests for information would be rewarded a response with relevant data.

Due to the functional requirement of being able to collect tags for various locations at different time periods, this software had to abide to the system principle of being flexible. The decision to design a web application that could be opened and customized when needed was therefore chosen. Having a graphical user-interface would allow different locations and different time periods to be set, without having to alter code each time different data was required. It also extends the future possibility of somebody using the system with basic computer skills. An overview of the software and how it operates can be seen below:

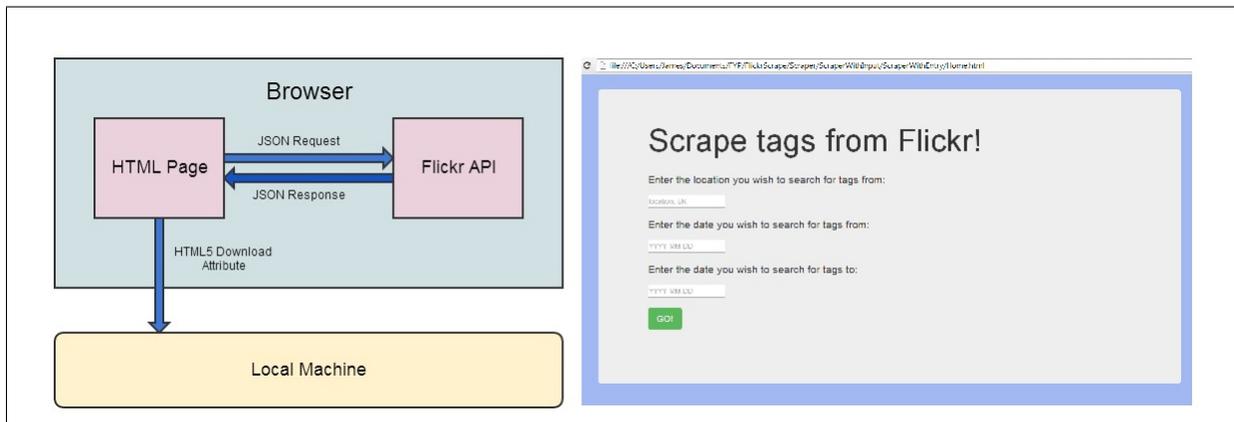


Figure 6: Design Overview of Flickr Tag Scraper

The software is designed using a combination of HTML, CSS, JavaScript, jQuery and JSON technologies. JSON is used to handle requests and responses for data via the Flickr API, while the HTML5 download attribute is used to automate a download to the users local machine. This download will consist of a cluster of tags taken and compiled into a single .txt file whereby each individual tag is separated by a comma. The tags will have been scraped from photos taken within the location entered, and within the specified date period.

Obtaining tag clusters one location at a time was not entirely efficient. In order to be able to download tag clusters for multiple locations automatically, a second version of the software exists (V2.0), whereby an array of locations can be set and individual downloads are automatically created for each location. This meant the acquisition of tags could happen more quickly. Code-wise the two versions of the software are very similar, and by operating them on different API keys it meant avoiding the abuse of the Flickr API.

The design aspect of the web-page interface is incredibly simple. Anybody would be able to use the software without extensive computing skills. This is ideal as its primary use is to gather data, therefore there is no need to create something more complicated. This also benefits the possibility of the software being used by organisations such as the police, who's employees will not necessary be extensively computer-trained.

4.4 Design: Chi-Squared Calculation Program

The program described here can be found in Appendix C.

In order to identify spatially-relevant terms from collected data, it was necessary to design a system that would automatically calculate chi-squared statistics for existing tags. With most experience using the Java programming language, this was chosen to develop the software.

The main functionality of this program would be to read in two different text files, one containing the tag cluster for a single location c , and another containing the combined tag clusters of all locations that had been scraped in the UK. This would allow for a statistical comparison between tags in one location and tags in all other locations, thus identifying terms most discriminative to a location c . An output of all tags that exist in location c and their corresponding chi-squared values would be the output. The figure below details the flow of data from input to output within the program:

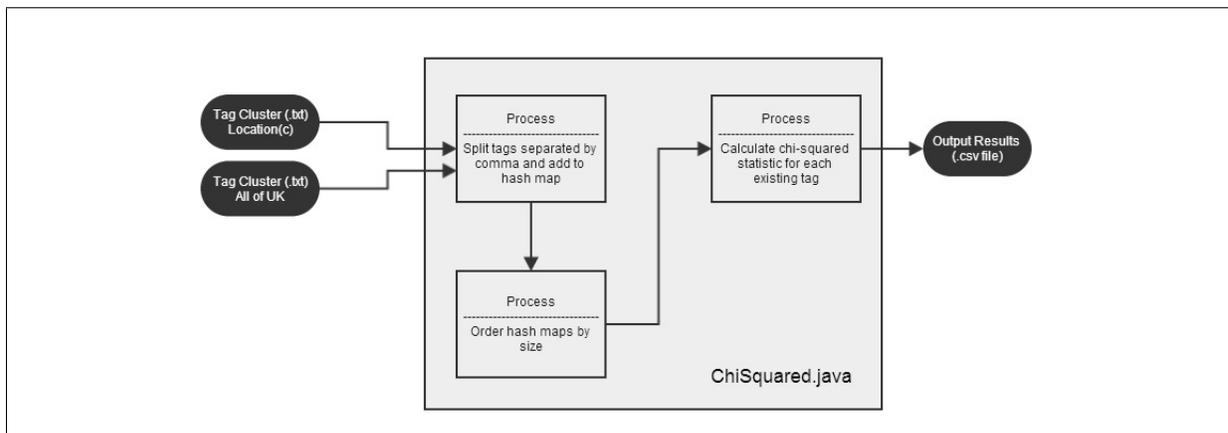


Figure 7: Design Overview of Flickr Tag Scraper

By creating two separate hash maps, one for the tags in location c , and one for all tags in the UK, it allows for calculation of corresponding observed frequencies (in location c), and expected frequencies (in reference to all tags in the UK). These values are vital in order to determine the complete chi-square formula.

The program is simple command-line style in order to generate the output file. Outputting results as comma separated values is useful for analyzing results, therefore this was an important consideration. The ability to order results via their determined X^2 value allows for instant identification of terms with maximum X^2 score. The resulting column headings in the output file are detailed in the figure below. All other values that are used in the chi-square formula are automatically generated in the java program, and will be detailed further in the implementation section.

Tag	ObservedFrequency	CountsInAllData	ExpectedFrequency	Chi-SquareTotal
sign	149	385	17.83	1012.51

Figure 8: Example Chi-Square Calculation Output

Each tag is bound to a single row with corresponding values in the relevant columns. One simple command to order the results via the ‘Chi-SquareTotal’ field and it is possible to instantly recognize those terms that are of interest, disregarding results below a certain threshold.

5 Implementation

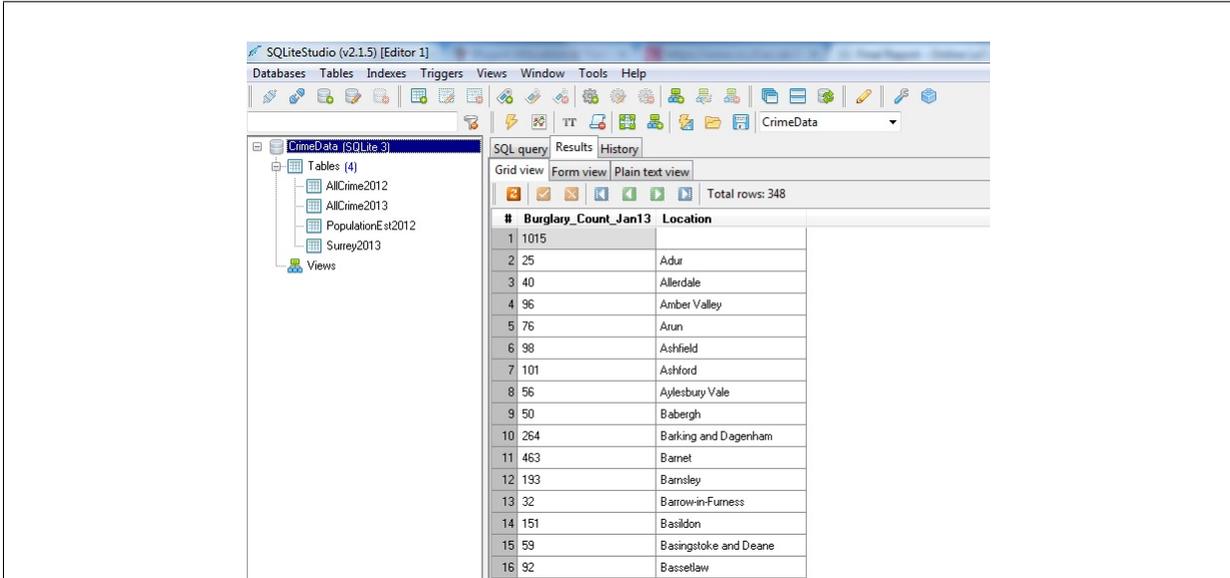
5.1 Performing Queries on the Database

In order to obtain numerical statistics that could act as attributes for linear regression, it was necessary to query the database containing police data for relevant figures. These statistics would be the basis for the machine learning processes used to create a regression model to make predictions. Various SQL queries were therefore executed to get total counts for general and specific types of crime in the 348 different locations within the UK. An example of a typical query can be seen in the figure below:

```
SELECT COUNT([Crime Type]) AS Burglary_Count_Jan13, [LSOA Name] AS Location
FROM AllCrime2013
WHERE [Crime Type] LIKE "%Burglary%"
AND Month = "2013-01"
GROUP BY [LSOA Name]
```

Figure 9: Example SQL Query Performed on Database

In the figure shown above, the query is used to gather the total counts of burglary reported in January of 2013, with the results ordered via their location. The ‘AS’ operation will change the column headings of the output, so that they relate directly to the month and crime that is being queried. By utilizing the ‘LIKE’ operation, as opposed to ‘=’ for the WHERE clause, it decreases the possibility of a false count due to a typing error. Due to the huge number of records stored in the database, the query should therefore return any records that have the phrase ‘Burglary’ included somewhere in the ‘Crime Type’ column. This should avoid a miscount if some extra character has accidentally been added to the field. An example of the output from the above query can be seen below:



#	Burglary_Count_Jan13	Location
1	1015	
2	25	Adur
3	40	Allerdale
4	96	Amber Valley
5	76	Arun
6	98	Ashfield
7	101	Ashford
8	56	Aylesbury Vale
9	50	Babergh
10	264	Barking and Dagenham
11	463	Barnet
12	193	Barnsley
13	32	Barrow-in-Furness
14	151	Basildon
15	53	Basingstoke and Deane
16	92	Bassetlaw

Figure 10: Example Output of SQL Query

The figure above shows the resultant output from the example query that was performed on the database. The output can be exported in csv format and used as a single attribute for a data set in Weka. The first record in the screen-shot appears to have a blank location. This is because some crimes documented by data.police.uk cannot be located due to the victim not knowing, or the police forces not being confident that location data is consistent or accurate. I therefore corrected this blank field with the location 'MISSING' in all data sets, or removed it and considered it redundant. As can be seen, locations are otherwise ordered in alphabetical order, with the corresponding counts of the specified crime type also associated. Occasionally an SQL query would return less than the expected 348 records, this was in circumstances where zero counts of a crime-type for a location existed. For example some places such as the 'Isles of Scilly' had incredibly low counts of burglary, and in some months absolutely no counts. This was a matter that had to be checked on each execution of a query.

5.2 Compiling a Data Set for Linear Regression

By executing queries that identify numbers of crimes committed in varying months, it was possible to create data sets that could be used to apply linear regression. For example, identifying the number of burglaries in all locations on a monthly basis through 2013. The model derived from this process could then be used to predict the number of burglaries for a forthcoming month. Applying the query in section 5.1 but changing the date for each month of the year would produce these specific input features. Using only 12 very similar queries, the data set below can be generated automatically with 12 respective features.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	LSOA Location	Burglary_Jan13	Burglary_Feb13	Burglary_Mar13	Burglary_Apr13	Burglary_May13	Burglary_Jun13	Burglary_Jul13	Burglary_Aug13	Burglary_Sep13	Burglary_Oct13	Burglary_Nov13	Burglary_Dec13
2	MISSING	1015	1054	956	1046	946	926	965	946	948	1143	1136	1034
3	Adur	25	24	19	21	41	30	28	32	25	19	28	25
4	Allerdale	40	30	53	38	55	46	71	45	54	53	42	25
5	Amber Valley	96	71	83	91	103	98	74	82	86	107	67	63
6	Arun	76	63	56	53	52	68	61	55	55	56	70	40
7	Ashfield	98	84	86	76	98	81	95	84	123	104	105	60
8	Ashford	101	50	59	51	87	64	54	71	75	67	87	75
9	Aylesbury Val	56	81	71	91	114	96	70	76	77	64	61	60
10	Babergh	50	51	48	52	41	45	61	53	47	34	44	47
11	Barking and C	264	250	245	180	171	143	158	149	130	192	197	141
12	Barnet	463	398	395	362	338	271	303	273	300	333	395	442
13	Barnsley	193	204	221	263	176	185	223	181	160	165	176	186
14	Barrow-in-Fur	32	36	29	28	30	31	31	18	28	40	38	24
15	Basildon	151	140	128	108	140	111	157	108	105	134	170	171
16	Basingstoke	59	64	81	54	49	61	63	86	52	62	79	55
17	Bassetlaw	92	88	98	87	97	81	84	104	83	95	106	81
18	Bath and Nor	78	80	48	54	73	65	73	63	59	75	56	62
19	Bedford	73	79	70	82	86	89	83	91	93	88	91	103
20	Bexley	175	161	123	157	137	129	106	127	129	135	204	220
21	Birmingham	856	778	712	731	728	703	717	792	742	794	767	763
22	Blaby	59	71	73	66	74	67	61	48	59	84	69	80
23	Blackburn wit	102	76	103	124	123	105	109	92	118	96	115	85

Figure 11: Example Input Features for Linear Regression

In the figure above, each individual location has corresponding counts of burglary for each month of 2013. This data can be used to train a model that will estimate predictions based on the features that have been provided. Analyzing the relationships and features that help make greatest predictions will be detailed in the analysis section. Currently the data is in csv format, however it needs to be converted for use with Weka. Fortunately Weka has a built-in converter to transform csv files into the ARFF format.

Following the importation of data to Weka, the explorer offers a number of basic preprocessing and visualization operations. The converter will automatically recognize column headings as attributes, and there is an option to remove any redundant columns. The software also displays statistics regarding minimum, maximum, mean and StdDev for each attribute. The figure below shows an overview of the preprocess tab.

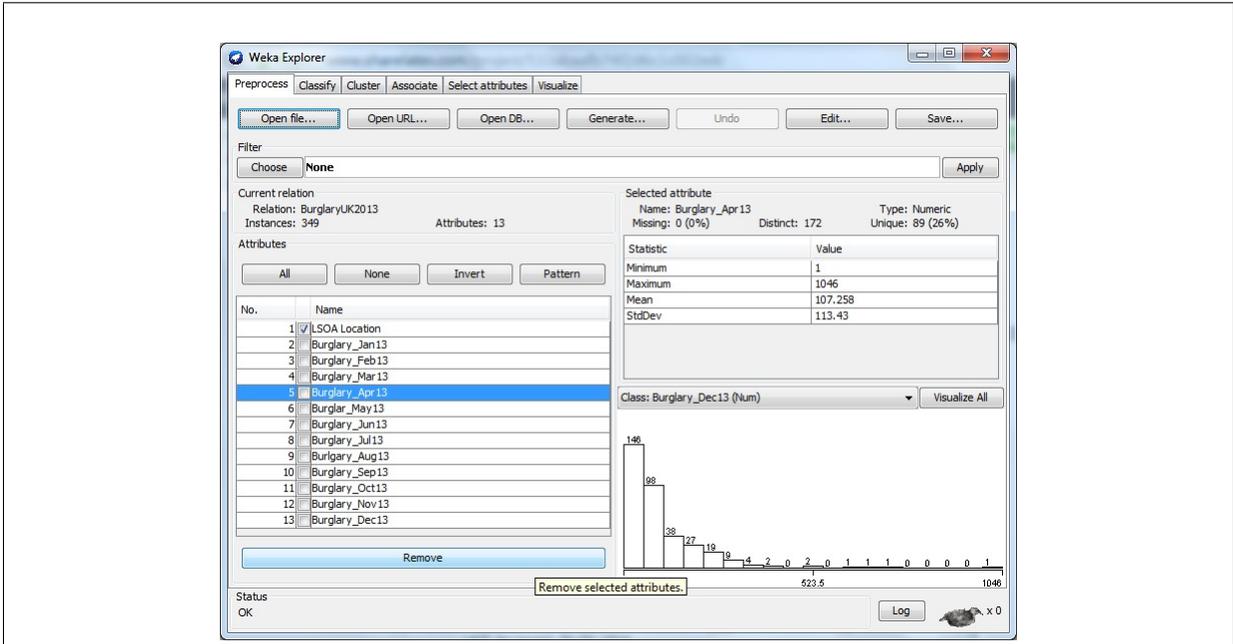


Figure 12: Overview of the Data Preprocess Tab in Weka

As can be seen in the figure above, the left tile shows the various attributes within the data, with the option to remove any if necessary. The tiles on the right show statistics regarding the column currently highlighted, and an independent visualization of the selected column. The ‘visualize all’ button can then be selected to show a graphical representation of each individual feature. The figure below shows the result of utilizing the ‘visualize all’ button.

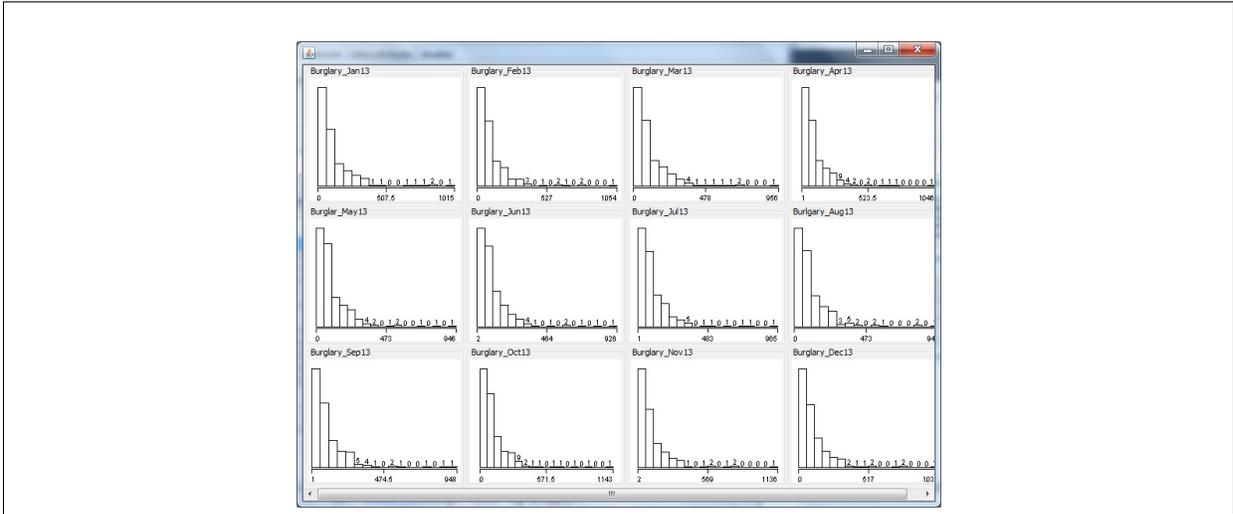


Figure 13: Example Visualization of Data in Weka

The figure above shows the visualization of attributes and corresponding data within Weka. While not particularly insightful, it is useful for noticing any striking differences between different features. It will instantly demonstrate any significant variation between crime counts in different months.

5.3 Test Options for Implementing Linear Regression

Once a set of features have been compiled and preprocessed, it is possible to perform the linear regression function in Weka. This function uses the training data provided, to build a regression model that can be evaluated in terms of its accuracy when applied to test data. There are various test options that can be implemented when executing linear regression, a brief overview of the important methods are detailed below:

- **Use Training Set:** The classifier will analyse performance of predictions made on the class of instances that were used to train it.
- **Cross-Validation:** The data is divided into the number of folds entered and a classifier is produced for each fold. The average performance is then taken from the various classifiers produced.
- **Percentage Split:** The classifier evaluates performance of predictions made on the specified % of data left out for testing.

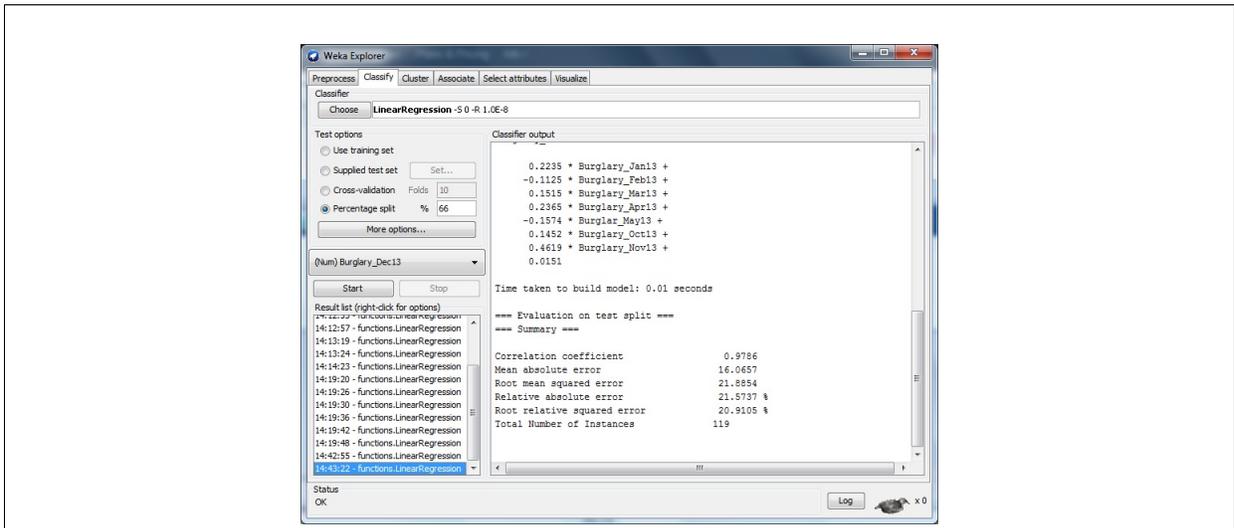


Figure 14: Classification Interface for Linear Regression in Weka

The figure above shows the interface for the classification tab in Weka. This is where the linear regression function is executed, and various options can be set regarding the test conditions. Once the necessary parameters have been set, the software produces a breakdown of the linear regression model. Also included is a summary regarding the test data, such as resultant values for the correlation coefficient, mean absolute error, root mean squared error, relative absolute error, root relative squared error, and the total number of instances. Comparing the evaluation of the classifier in terms of performance on test data can be used to identify features that help give most accurate predictions.

5.4 Identifying Regions of High and Low Crime

In order to analyze differences regarding spatially-relevant terms, it was important to recognize those LSOA locations which can be classed as having either a high or low crime rate. To reiterate, Lower Super Output Areas are essentially bounding boxes used define regions for the sake of national and neighborhood statistics. This task was implemented by obtaining population estimates for the locations that exist in the database. Below shows a visualization of all 348 LSOA's corresponding population counts. Although not each location can be uniquely identified, it gives an idea of the varying area sizes.

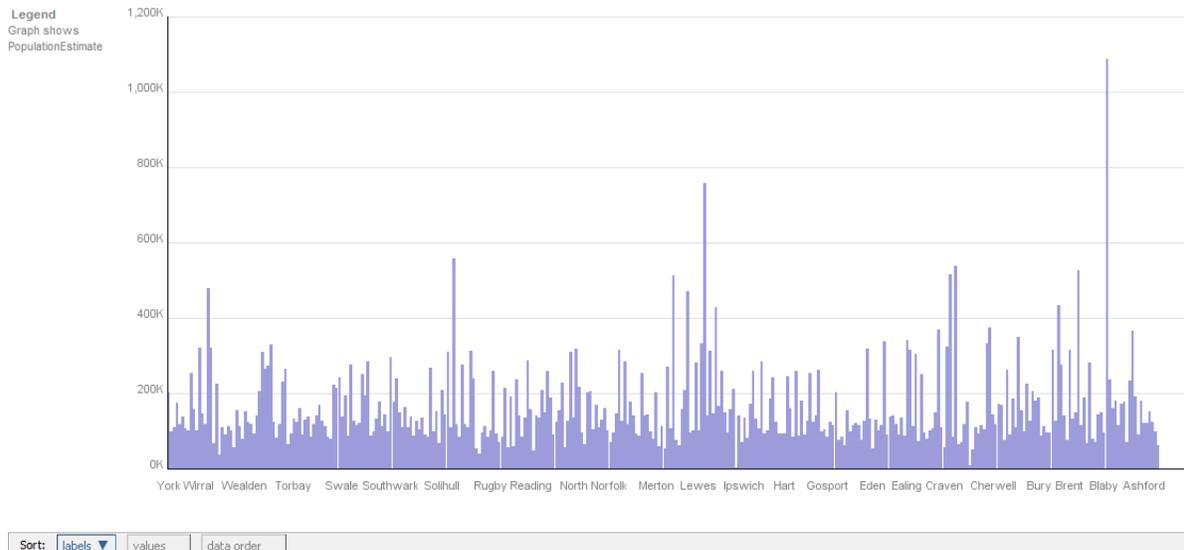


Figure 15: Visualization of Lower Super Output Area Population Estimates

These population estimates were provided by the Office for National Statistics, and allowed the calculation of total crimes committed in 2013 against the approximate population in the area. Essentially this figure identified the number of crimes roughly committed per person. Appendix A contains the entire list of locations in the database and the corresponding statistics of crime per population. After obtaining these values, a subset of locations to scrape tags for was selected. This subset would allow the gathering of tag clusters for regions with a strong variation in the levels of crime. Due to the time taken to scrape tags and the Flickr API restrictions, it was not feasible to collect sizable tag clusters for all 348 locations in the database. Instead, a total of 824,935 tags were collected across 30 different locations in the UK.

The figure below shows the resultant subset of locations within the database. The crime ratios vary from very-high to very-low and therefore give an opportunity to investigate the Flickr tagging differences. It was also important that the locations selected would provide strong numbers of tags via Flickr, something that was tackled on a trial-and-error basis. Obviously some regions have lots of photos taken within them, particularly the major cities. However a large segment of low crime regions are more remote, whereby photos are not uploaded so frequently.

#	A	B	C	D	
1	LOCATION (LSOA NAME)	Total_Crime_Count_2013	Population_Estimate	Crime_Per_Population	HIGH CRIME
2	Blackpool	29998	141976	0.211289232	
3	Middlesbrough	27830	138744	0.200585251	
4	Manchester	90900	510772	0.177965903	
5	Preston	24064	140540	0.171225274	
6	Newcastle Upon Tyne	48198	282442	0.170647425	
7	Lincoln	15977	94588	0.16891149	
8	Norwich	21789	134264	0.162284752	
9	Nottingham	48856	308735	0.158245745	
10	Liverpool	72988	469690	0.155396112	
11	Sheffield	84926	557382	0.152365882	
12	Bristol	65314	432451	0.15103214	
13	Worcester	14051	99604	0.141068632	
14	Leeds	99433	757655	0.131237833	
15	Cardiff	45601	348493	0.130851983	
16	Bradford	66845	524619	0.127416277	
17	Kettering	11371	94841	0.119895404	
18	Exeter	14222	119397	0.119115221	
19	Watford	10294	91732	0.112218201	
20	Birmingham	119314	1085417	0.109924573	
21	Redditch	9219	84419	0.109205274	
22	Cheltenham	12636	116080	0.108855961	
23	Coventry	33064	323132	0.102323509	
24	Dover	10707	111765	0.095799222	
25	Winchester	8696	117702	0.073881497	
26	Chichester	8001	114521	0.069864916	
27	Braintree	10226	148384	0.068915786	
28	Stafford	8796	131630	0.066823672	
29	Wycombe	10270	173306	0.059259345	
30	Rochford	4836	83869	0.057661353	
31	Horsham	7595	132160	0.05746822	LOW CRIME

Figure 16: Subset of Regions with High and Low Crime Rates

The initial plan was to select locations with the highest total counts of crime during 2013. This process was not useful as those areas with a large population tend to have more crime by default. By identifying the ratio which considers the population in the area, it allows for true identification of regions that could be of contrasting interest.

5.5 Scraping Tags via the Flickr API

The program used to scrape tags from photos on Flickr was crucial in order to gather tag-clusters useful for analysis. Created as a web application, the location and date input fields are customizable, allowing requests to be made in order to meet a users needs. HTTP GET requests are made in order to obtain data in the JSON format, which is then ultimately manipulated and downloaded to a users local machine. Two versions of the software exist, one in which an array of locations can be set in order to scrape multiple places for tags at once, and another version whereby a user can enter the location they wish to search for manually. Below will give further code-level detail into the way in which the program has been implemented.

5.5.1 Form on the HTML Page

The HTML homepage of the software is the interface in which a user interacts with. It is of very simple design that has been styled using the ‘Bootstrap’ front-end framework. The page has been developed to implement two input fields, a start date and an end data to search for tags within. The version in which you can manually enter a location also includes another input field for this matter. On entering the required data into the input fields, the jQuery `.click()` event is then used to trigger the scraping process via the `getTags.js` file.

To send requests to the Flickr API for data relating to the field entries on the HTML page, the values have to be obtained in the JavaScript file. Below details how the variables are created by capturing the values entered by a user.

```
25 //Execute functions when button on html is clicked
26 $("#send").click(function(event) {
27     event.preventDefault();
28     $('#tagbox').show();
29
30
31     //setup variables with input taken from entry boxes
32     var dateFrom = $('#dateFrom').val();
33     var dateTo = $('#dateTo').val();
34     var location = $('#locationEntry').val();
35
```

Figure 17: Capturing User-Entered Variables

The above code shows how on submitting the button on the HTML page, the values that have been entered in the date and location fields are passed to corresponding variables in `GetTags.js`. Alternatively, in the second version of the program, an array of locations can be set in the code in order to scrape multiple locations at once.

5.5.2 Function to Build Request URL

Due to the architecture of the Flickr API, it requires a sequence of requests in order to collect tags from photos. Due to this, it was beneficial to create a function that would provide an efficient way of constructing a request URL each time it was needed. This would avoid having to rewrite and complicate code which often used the same parameters such as the Flickr API key. The figure below shows a screen shot of code.

```
10 //setup url prefix and api_key for use with requests
11 var flickr = {
12   url: 'http://api.flickr.com/services/rest/',
13   api_key: 'be6e0e195148cbc8d2648f8dc06d50c3',
14 };
15
16 //function for constructing correct url for JSON request
17 var buildUrl = function (method, query) {
18   var url = flickr.url+"?method=flickr."+method+"&api_key="+flickr.api_key;
19   for (key in query)
20     url += '&' + key + '=' + query[key];
21   url += "&format=json&nojsoncallback=1";
22   return url;
23 };
```

Figure 18: Build URL Function

The above figure shows two important aspects that are used in the code. The variable ‘flickr’ can be used to hold common attributes that are used when constructing a URL in the program. The ‘buildURL’ function is designed to assemble a URL by storing all of the universal parameters, and combining the specific method and query where necessary. This made requests and resultant code more simple.

5.5.3 Function to Search Photos

The first step towards collecting tag-data requires searching for photos in accordance to the date and location parameters that have been set. These parameters must therefore be reflected in the search request.

```
49 //Get photos matching place id and date taken
50 var getPhotos = function(place_id, pageNum) {
51   console.log('getting photos for: ' + place_id);
52   var url = buildUrl('photos.search', {place_id: place_id, min_upload_date: dateFrom,
53     per_page: '500', max_upload_date: dateTo, safe_search: '3', page: pageNum});
54
55   console.log(url);
56   scraped[place_id] = {};
57
58   $.getJSON(url).done(function (data) {
59     console.log("Total number of photos in search: " + data.photos.total);
60     console.log("Number of pages to scrape photos for: " + data.photos.pages);
61     console.log("currently scraping page: " + data.photos.page);
62     photosByPage(data, place_id, pageNum);
63   });
64 };
```

Figure 19: Function to Search Photos

This figure details the ‘getPhotos’ function which is used to make the initial request for photos according to the desired criteria. The required location corresponds to the ‘place_id’, and

the date fields relate to 'min_upload_date' and 'max_upload_date'. The 'buildUrl' function is implemented in order to construct the request URL with the associated 'photos.search' method and corresponding query data. Also important to note is that the maximum photos per page request is 500, and by using 3 for the 'safe_search' value there will be no restricted content. Once the request has been assembled and called, the response data is captured in the JSON format. Some information such as the total number of photos in the search and the number of pages is logged by the console. The data is then passed to another function 'photosByPage'.

The purpose of the 'photosByPage' function is to ensure that if more photos are returned than can be held in one page (500), then all pages are requested in turn. It also guarantees that for each page of data, every photo is individually scraped for corresponding tags.

```
66 var photosByPage = function(data, place_id, pageNum){
67     $.each(data.photos.photo, function(key, photo){
68         //console.log("Getting individual tags");
69         getTags(place_id, photo.id, photo.secret);
70     });
71     if (data.photos.page != data.photos.pages) {
72         pageNum +=1;
73         getPhotos(place_id, pageNum);
74     }
75 };
```

Figure 20: Function to Check All Photos are Scraped

Figure 20 shows that for each photo in the response data, it's corresponding 'photo.id' and 'photo.secret' are passed to the 'getTags' function. This is necessary as a unique API call must be made in order to acquire tags for each individual photo. The 'if' clause in the above figure is used to check if the corresponding page number in the response data matches the total number of pages for the search. If these two values are not equal, then it means there is more than one page of photos in the search, therefore the page number is incremented and the 'getPhotos' function is called again with the new URL.

5.5.4 Acquiring Tags from Photos

Once photos within the specified parameters have been obtained, each individual 'photo.id' and 'photo.secret' is passed to the 'getTags' function in conjunction with the corresponding 'place_id' the photo resides in.

```
79 // gets tags for each photo
80 var getTags = function(place_id, photo_id, photo_secret) {
81     var url = buildUrl('photos.getInfo', {photo_id: photo_id, photo_secret: photo_secret});
82     var currentTags = [];
83     $.getJSON(url).done(function(data) {
84         //check to see photo exists
85         if (data.photo !== undefined){
86
87             //if photo does exist, grab tags and push to array
88             $.each(data.photo.tags.tag, function(key, tag) {
89                 currentTags.push(tag.raw);
90             });
91             scraped[place_id][photo_id] = currentTags;
92         }
93     });
```

Figure 21: Function to Get Tags From Photos

The above figure shows a screen-shot of the 'getTags' function, which implements the 'buildUrl'

function with the ‘photos.getInfo’ method to acquire tags. This is the only method that can be used to get tags from photos, therefore a single request must be made per-photo individually via the corresponding photo id. To check that the photo exists, an ‘if’ operation is utilized to ensure the photo is not undefined. This is important as some photos will be undefined if they have been deleted or are unauthorised. If the photo does exist then each individual tag (tag.raw), will be pushed into an array ‘currentTags’. This array of tags is then stored within the ‘scraped’ object. Below shows a screen-shot for the structure of the ‘scraped’ object and how it is broken down.

```

▼ Object {niTM5F9QUbNtxw: Object, NPhsmjBQV7uWhA: Object, KbEnpbtQUL32Yw: Object}
  ► KbEnpbtQUL32Yw: Object
  ▼ NPhsmjBQV7uWhA: Object
    ► 8331998541: Array[0]
    ► 8332072215: Array[3]
    ► 8332073195: Array[3]
    ► 8332302815: Array[2]
    ▼ 8332430291: Array[11]
      0: "leyland"
      1: "leylandatlantean"
      2: "alexander"
      3: "westernwelsh"
      4: "1971"
      5: "PDR1A/3"
      6: "331 Pontypridd"
      7: "Wales"
      8: "Cardiff"
      9: "Cardiff bus station"
      10: "Atlantean"
      length: 11
    ► __proto__: Array[0]
  ► 8332487573: Array[0]

```

Figure 22: Structure of the Scraped Object

The ‘scraped’ object stores a single unique object for each location (if there is multiple locations). Each location object contains individual photos and arrays for the associated tags. For example in figure 22, there are three object locations that are defined by their corresponding place id. Each place id has associated photos that contain arrays of tags. In the example above, the selected photo within one location is displaying 11 unique tags.

5.5.5 Downloading Tag Clusters to Local Machine

Following the acquisition of tags relating to photos that meet the search criteria, it was necessary to implement some way of automating a download to a users machine. This was a task that could be challenged without server-interaction, as all analysis of data was to be executed locally anyway. In order to automate the download, the .ajaxStop() operation was implemented to trigger the ‘printTags’ function. This operation means a download will be triggered when all requests have received a response and therefore no further data is being awaited.

The ‘printTags’ function is what handles encoding the tag-clusters and instantiating the download via the HTML5 download attribute. The jQuery.map() operation is used to map all contents of an object location into a single array ‘arr’. This means that if multiple locations exist in the search, contents will be mapped to an array in-turn and downloaded as separate tag clusters. The figure below shows a screen-shot of the code used in the ‘printTags’ function.

```

106 //encode tag files and download in browser
107 var printTags = function(scraped){
108   console.log("Downloading tags to machine now");
109
110   var indivLocation = {};
111   $.each(scraped, function(index, value){
112     indivLocation = value;
113     var arr = $.map(indivLocation, function(value, key){
114       return value;
115     })
116
117     var tagCluster = (JSON.stringify(arr));
118     tagCluster = tagCluster.replace(/[^A-Za-z0-9, &]/g, ' ');
119     var a = document.body.appendChild(
120       document.createElement("a")
121     );
122     a.download = "tagData.txt";
123     a.href = "data: text/plain;base64," + btoa(tagCluster);
124     a.click();
125   });
126   // location.reload();
127 };

```

Figure 23: Function to Download Tags to Users Machine

Once the contents of a location object have been mapped to an array, it is passed to the ‘tagCluster’ variable and the JSON.stringify() operation is performed. The ‘replace’ operation also ensures that any special characters that are not permitted are removed. The download attribute variable ‘a’ is then created and appended to the document. The tag cluster is then bound and encoded to ‘a’, and finally ‘a.click()’ initialises the download automatically. An example output for a tag cluster appears below.

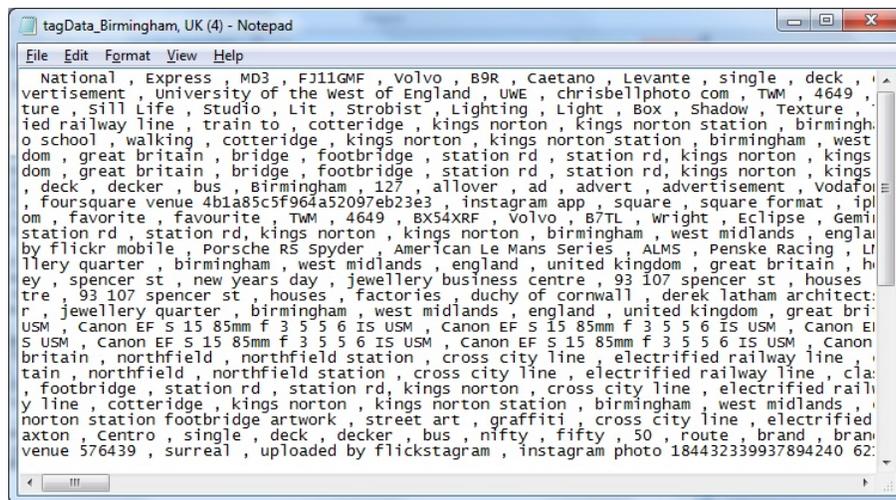


Figure 24: Function to Download Tags to Users Macine

The tag clusters are simple .txt files whereby each individual tag is separated by a comma. Clusters are downloaded in separate files depending on the location they relate too. The fact they are comma separated terms makes the files easy to manipulate in Java, and therefore made the next step of calculating Chi-Squared terms relatively simple. Any unwanted white-space that appears in the .txt files is managed in the Chi-Squared calculation program.

5.6 Calculating Spatially Relevant Terms

After a significant collection of tag clusters were gathered, it was possible to create a program that would calculate chi-squared statistics for individual tags. This would allow the identification of those terms with a high X^2 score and therefore the most spatially-relevant tags. Assessing whether these location-specific tags add any insight into helping predict crime patterns or identifying relationships would then be possible. The program has been created in the Java programming language and takes two input files, one .txt file containing tags for a single location c , and a second .txt file that contains the combined contents of all locations tag-clusters. The output is a csv file containing each unique tag from within location c and its corresponding X^2 score in relation to the rest of the UK.

5.6.1 Compiling the Input Files

Scraping tags from Flickr was done on a monthly basis, therefore clusters were saved according to the month they relate too. In order to compile an entire cluster for a single location, it was therefore necessary to merge all monthly .txt files into a single cluster. This could be done instantly with a single command-line execution in the working directory.

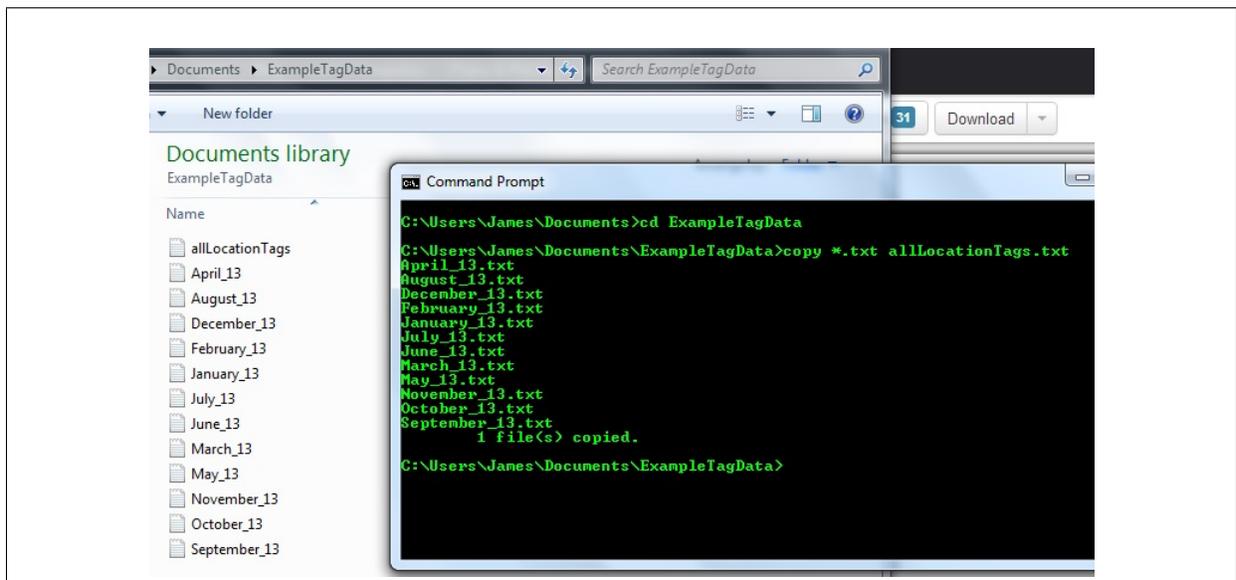


Figure 25: Example Folder Containing Monthly Tag Clusters

The figure above shows an example folder containing monthly tag-clusters for a single location, which can be merged using the 'copy *.txt newfile.txt' command to create a cluster containing all tags for location c . By implementing this command for all locations that were scraped via Flickr, it meant single tag-clusters were created for each region. The same technique was used to create a file aggregating all tags scraped from within the UK.

5.6.2 Reading Tag-Clusters into Hash Maps

The ChiSquared.java program is designed to read two input files and initially separate tags by the commas that divide them. The entire files are read in at once, using the ‘\Z’ delimiter to implicate when the end of a file is reached.

```
try {
    //Read in location c's tag cluster
    String allTags = new Scanner ( new File ("locationTags.txt" ) ).useDelimiter("\\Z").next();
    //Read in all location's combined tag cluster
    String indexTags = new Scanner ( new File ("totalTags.txt" ) ).useDelimiter("\\Z").next();

    //Split tags seperated by comma
    String[] splitTags = allTags.split(",");
    String[] splitAllTags = indexTags.split(",");
}
```

Figure 26: Reading Tags and Separating via Commas

The above figure shows how the two files (which must be in the same working directory as the Java program) are scanned into the Java program as strings. The tags are then individually split-up by breaking text where a comma exists, and adding the tag to an array of strings. It is important to note that an entity exists for both the location *c* tag-cluster and the cluster containing tags from all places in the UK. This is essential for identifying values that are implemented in the X^2 statistics calculation.

It is necessary to populate two hash-maps with data from the corresponding tag-clusters. This allows for comparisons between the number of times a tag may appear in location *c* and the number of times it appears elsewhere in the UK. This process is done by looping through the array of strings and passing the tag into the hashmap. By checking if the hashmap already contains the tag, it is possible to increment the value and therefore count the times a unique tag appears in the cluster.

```
//Loop through each dataset populating hashmap with tags & corresponding counts
for (int i = 0; i < splitTags.length - 1; i++) {
    String key = splitTags[i];
    key = (key.toLowerCase()).trim();
    if (glossary.containsKey(key)) {
        glossary.put(key, Integer.parseInt(glossary.get(key).toString()) + 1);
    } else {
        glossary.put(key, 1);
    }
}
```

Figure 27: Reading Tags and Separating via Commas

Figure 27 shows how each tag is transformed into lower case in order to avoid miscounts. By implementing the `.trim()` operation it also removes any unwanted white-space that may surround the tag. The hash map ‘glossary’ is then checked to see if it contains a key that matches the current tag. If a match is found then the value is incremented, else-wise the tag becomes a new entry in the hash map with an initial value of 1. The same process is implemented for the two individual tag-clusters.

5.6.3 Sorting Hash Maps of Tags

Once the two individual hashmaps have been populated with tags, they are then sorted in descending order in terms of observed frequencies within location c . This process is implemented by temporarily transforming the maps into linked-lists and then sorting them by using a comparator.

```
List list = new LinkedList(glossary.entrySet());
List list2 = new LinkedList(totalTags.entrySet());

//Sort List based on comparator
Collections.sort(list, new Comparator() {
    public int compare(Object instance1, Object instance2) {
        return (-1) * ((Comparable) ((Map.Entry) (instance1)).getValue())
            .compareTo(((Map.Entry) (instance2)).getValue());
    }
});
```

Figure 28: Sorting Hash Maps via Observed Frequencies

Figure 28 shows how the tags and their corresponding frequencies are sorted through the use of a comparator. The entry set for hash map ‘glossary’ becomes a linked-list ‘list’. The ‘Collections.sort’ function is then executed to implement a new comparator. Corresponding values for entries in the list can then be compared in turn by identifying them as object instances. Alternatively, by switching the ‘-1’ to a ‘+1’ would sort the frequencies of tags in ascending order. This method of sorting is loosely based on an example documented online which is referenced in the source code.

5.6.4 Constructing the Chi-Squared Formula

By implementing hash maps to count the frequency of unique tags in location c and the rest of the UK, it is then possible to use this information to calculate other numbers used in the formula. An iterator is used to cycle through each item in location c ’s tag cluster, and compare it to entries in the cluster containing all tags in the UK. If a match is found, then the value relating to the occurrence of the tag in all data is captured, and used to calculate the expected frequency of the tag.

```
ObservedFrequency = ((Integer) tagC.getValue());
tagCounter += ObservedFrequency;

//Match tag keys from location c and all data - calculate expected frequency values
Iterator it = allUK.entrySet().iterator();
while (it.hasNext()) {
    Map.Entry tagUK = (Map.Entry)it.next();
    if ((tagUK.getKey()).equals(tagC.getKey())){
        totalCounts = tagUK.getValue().toString();
        ExpectedFrequency = ((Double.parseDouble(totalCounts)*locationTagsSize)/allTagsSize);
        try {
            chiSquare = (ObservedFrequency-ExpectedFrequency)*(ObservedFrequency-ExpectedFrequency);
            chiSquare = chiSquare / ExpectedFrequency;
        }
    }
}
```

Figure 29: Calculation of Observed and Expected Frequencies in Java Program

Figure 29 shows how a tag, ‘tagC’, first has its count in location c assigned to the ‘ObservedFrequency’ variable. The iterator is then implemented to loop through the map containing all tags in the UK. If ‘tagC’ matches a key in all of the UK data, then the value for the times it appears in all locations is stored in the ‘totalCounts’ variable. This variable is vital for identifying the expected frequency of ‘tagC’, which is calculated by multiplying the times a tag appears in all data by the total number of tags in location ‘ c ’, then ultimately dividing that number by the total number of tags in all locations. The ‘ObservedFrequency’ and ‘ExpectedFrequency’

values for ‘tagC’ can then be used to implement the first part of the Chi-Squared formula, demonstrated above by the ‘chiSquare’ variable calculation.

Having calculated the initial part of the formula, the remaining parts must be determined and the total combined. The background section of this report on Chi-Squared term selection details how values in the formula relate to this project. The figure below details how these values are implemented in the code.

```
//Calculate rest of chi-square formula
//Probability of tag t occurring in any location -> total occurrence of tag / total number of tags in all data
float Pt = (Float.parseFloat(totalCounts)) / allTagsSize;
//Probability of tag occurring in this location -> total tags in location / total of all tags
float Pc = ((float) locationTagsSize) / allTagsSize;

//number of tags that aren't t that appear in location c
o1 = locationTagsSize - ObservedFrequency;
double e1 = allTagsSize * (1-Pt) * Pc;
double chi1 = ((o1 - e1) * (o1 - e1)) / e1;

//Number of times tag t appears outside location c
o2 = (Integer.parseInt(totalCounts)) - ObservedFrequency;
double e2 = allTagsSize * (Pt) * (1-Pc);
double chi2 = ((o2 - e2) * (o2 - e2)) / e2;

//Number of tags that appear outside c that are not t
o3 = (allTagsSize - locationTagsSize) - ((Integer.parseInt(totalCounts)) - ObservedFrequency);
double e3 = allTagsSize * (1-Pt) * (1-Pc);
double chi3 = ((o3 - e3) * (o3 - e3)) / e3;

//Sum the total of each of the four parts to the chi square formula
double chiSquareTotal = chiSquare + chi1 + chi2 + chi3;

//Output results with tag and corresponding values as individual rows
writer.write(tagC.getKey() + "," + ObservedFrequency + "," + totalCounts + "," + ExpectedFrequency + "," + chiSquareTotal);
writer.newLine();
```

Figure 30: Calculation of Chi-Square Formula

This figure details how variables ‘Pt’ and ‘Pc’ calculate the probability of a tag either appearing in any location or location *c*. These variables must be of float-type due to the incredibly small numbers being divided and not wanting to lose numerical accuracy. Variables o1 and e1 relate to terms $O_{\bar{t}c}$ and $E_{\bar{t}c}$ from the formula, while o2 and e2 relate to terms $O_{t\bar{c}}$ and $E_{t\bar{c}}$, and finally o3 and e3 relate to terms $O_{\bar{t}\bar{c}}$ and $E_{\bar{t}\bar{c}}$. The addition of the chiSquare variable shown in Figure 29, the chi1, chi2 and chi3 variables creates the final X^2 statistic for a tag in location ‘c’. The writer is then used to output results for each tag and its corresponding values on a row-by-row basis. An example output csv file from running the program can be seen below.

	A	B	C	D	E
1	Tags	Observed_Frequency	Counts_In_All_Data	Expected_Frequency	ChiSquareTotal
2	harry potter	867	868	20.58216466	35690.9327
3	watford	620	620	14.70154618	25546.10952
4	hogwarts	527	527	12.49631426	21711.74349
5	potter	437	440	10.43335536	17873.25348
6	harry	437	462	10.95502312	16980.99045
7	harry potter studio t	338	338	8.014713887	13921.98724
8	leavesden	330	330	7.825016516	13592.34094
9	hertfordshire	281	281	6.663120125	11573.39646
10	leavesden studios	265	265	6.283725384	10914.20131
11	studio	396	611	14.48813664	10297.87154
12	magic	245	251	5.951754987	9837.400715
13	tour	319	544	12.89942117	7445.034061

Figure 31: Example Output of Chi-Squared Calculation

This particular figure shows results of the calculation process for tags in Watford. It appears that the ‘Harry Potter Studio Tour’ which takes place in Watford is a very commonly tagged event. This particular screen-shot shows tags prior to any filtering of irrelevant or redundant data.

6 Constraints & Unforeseen Circumstances

Throughout the course of this project it was inevitable that some constraints on the planned implementation would occur. Essentially these factors are matters that were not initially known or acknowledged, and therefore had to be worked around on encountering the issue. These factors either limited my proposed solution, or required a change to some functionality.

6.1 Flickr API Request Limit

The Flickr developer guide details information regarding best practices and fair use of the service. By issuing developers with an API key it allows Flickr to monitor people using the API. According to the regulations stated on their website, a single application using one key cannot exceed 3600 queries per hour. This was a temporary setback, as it meant leaving a program to gather multiple years worth of data would exceed this limit. By breaching the request limit it could ultimately result in a temporary or permanent ban on the corresponding application key. Considering that gathering tags from a single photo on Flickr requires two requests alone, it was necessary to change the planned implementation.

As a solution to this issue, tags were gathered from photos in specified locations on a monthly basis. Gathering data for a single month would typically not exceed the API limit. Also it would result in separate files to show how the occurrence of tags fluctuates between months. By efficiently managing time between collecting data for different locations, it allowed the acquisition of a final data set containing 824,935 tags without abusing the Flickr API.

6.2 Identifying Locations Within Police Data

The data supplied by data.police.uk was vital for identifying criminal statistics across different locations in the UK. However, the data provided in its raw form needed modifying to ensure successful predictions could be made by the regression model. The data provided is approximated to street-level locations, with a further column that details the LSOA (Lower Super Output Area) that the street resides in. Technically it would be feasible to implement a project that estimates crime counts at a street-level; however there is a minor issue with this process. Despite some streets that have unusually frequent occurrences of crime, most have between zero and two crimes committed within them per month. Making estimations with regards to such low numbers is not particularly useful or insightful. Therefore, in order to build a prediction model that makes useful and potentially accurate predictions, the decision was made to work with locations at a LSOA level.

The data provided covers all areas of England, Wales and Northern Ireland. This total area is divided into 348 different Lower Super Output Areas, essentially bounding boxes that can be used to collect area statistics by government. This allows for working with 348 different areas in the UK, as opposed to just say the 62 cities that exist excluding Scotland. Each of these LSOAs is further broken down with an associated code that is provided in the police data. In order to be able to query the database for total counts of crime in these LSOAs, it was therefore essential to remove the extension code that gives further accuracy. Implementing an SQL query to remove the trailing extension code in the LSOA location column was the solution to this issue.

One other issue regarding the police data relates to some records having a blank entry in the location field. These are crimes that have been logged by police without them being able to

verify location information being entirely accurate. This may be because there is little evidence, or that the victim of a crime is unaware of where the incident occurred. Any resultant blank fields from database queries were marked as 'MISSING' or deemed redundant.

6.3 Maximum Photos Returned Per-Page by Flickr API

During the early stages of this project, when implementing the 'photos.search' method, the number of photos returned per-page was set at the maximum (500). However it was soon realised that, especially if the date-period is set to more than a few days, searches could return many more than 500 photos. This meant a check needed to be enforced, to see whether more than 1 page of photos existed for the search query. If more pages did exist, then the page number variable had to be incremented, and the GET request must be made once more. This would ensure all pages of photos were analyzed, and tags from some photos were not excluded.

6.4 Issues With Searching for Photos via LSOA

There were some small issues that arose from the decision to work with the 348 different UK locations detailed in the police data. Most of these areas were either cities such as Cardiff and Leeds, or towns such as Horsham and Kettering. These locations could be easily identified via the Flickr 'place_id' when a search was made for photos in the region. However, due to London being broken down into various LSOA's, this was an issue. Flickr would not recognize the independent LSOA's within London via their 'place_id'. Instead they were all just recognized as part of London, and not the more specific location. This meant relating photos on Flickr taken in London was difficult to pinpoint to a specific LSOA. Due to the large number of other locations in the data, the decision to focus on scraping tags for a subset of locations excluding London was made. By identifying population counts for all locations, it was still ensured that a varying contrast of high and low-crime places were chosen for analysis.

7 Analysis of Results

This section will detail example results that have been acquired from implementation of this project. It will also summarise the various data that has been collected, and the significance of any results.

7.1 Results of Linear Regression in Weka

The database was queried for multiple features regarding counts of crime in different locations in the UK. These features were then collated and used as a data-set for linear regression in Weka. The resultant classifier model could then be used to analyze the success based on test data. In order to demonstrate results, an example case for estimating the number of burglaries in a location is detailed below.

7.1.1 Estimating the Total Number of Burglaries

Referring back to the initial requirements, one produce identified was the ability to predict counts for a specific type of crime in one month. In order to do this, the database was queried for total numbers of burglaries for each month of 2012 and 2013 for all 348 locations. Each month acted as a feature within the data-set, and was passed into Weka so that linear regression could be applied. The prediction class was set to be the total counts of burglary in December 2013. By setting the test options to a 70% Percentage split, it would mean that 244 of the 348 locations would be used to train the model, and the remaining 104 instances would be left out for testing and evaluating predictions.

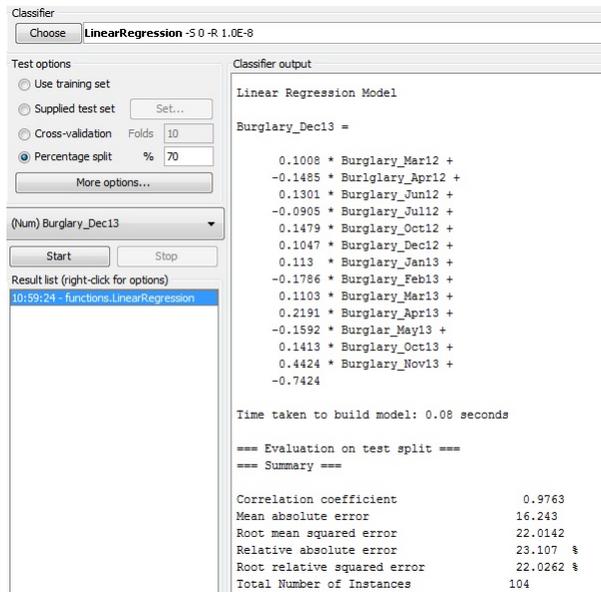


Figure 32: Regression Model for Predicting Burglary Counts

The above figure shows the resultant model built in order to predict counts of burglary in December 2013 for any location. The mean absolute error for predicting burglaries in December at any UK location is 16.243. This value is a quantity for describing the average absolute errors between the correct figure and the prediction. An example of applying the model to one location such as Cardiff can be used to demonstrate its accuracy.

	Jan-12	Feb-12	Mar-12	Apr-12	May-12	Jun-12	Jul-12	Aug-12	Sep-12	Oct-12	Nov-12	Dec-12
Cardiff	374	324	387	384	314	345	317	272	222	318	361	245
	Jan-13	Feb-13	Mar-13	Apr-13	May-13	Jun-13	Jul-13	Aug-13	Sep-13	Oct-13	Nov-13	Dec-13
Cardiff	228	306	302	260	251	236	210	233	272	277	315	259

Figure 33: Counts of Burglary in Cardiff

The figure above shows corresponding counts of burglary in Cardiff on a monthly-basis through 2012 and 2013. By using this data we can show the difference between the actual number of burglaries in December 2013, and the number that the linear regression model has predicted. Utilizing the model from figure 32, the following calculation can be shown as evidence.

$$\begin{aligned}
 & \text{Burglary_Dec13} = \\
 & 0.1008 * \text{Burglary_Mar12}(387) + \\
 & -0.1485 * \text{Burglary_Apr12}(384) + \\
 & 0.1301 * \text{Burglary_Jun12}(345) + \\
 & -0.0905 * \text{Burglary_Jul12}(317) + \\
 & 0.1479 * \text{Burglary_Oct12}(318) + \\
 & 0.1047 * \text{Burglary_Dec12}(245) + \\
 & 0.113 * \text{Burglary_Jan13}(228) + \\
 & -0.1786 * \text{Burglary_Feb13}(306) + \\
 & 0.1103 * \text{Burglary_Mar13}(302) + \\
 & 0.2191 * \text{Burglary_Apr13}(260) + \\
 & -0.1592 * \text{Burglary_May13}(251) + \\
 & 0.1413 * \text{Burglary_Oct13}(277) + \\
 & 0.4424 * \text{Burglary_Nov13}(315) + \\
 & -0.7424 \\
 & = \mathbf{270.0488}
 \end{aligned}$$

Figure 34: Example Calculation utilizing Linear Regression Model

From using the model built by the linear regression function, our prediction for counts of Burglary in Cardiff in December 2013 is 270. In comparison to the actual count of 259, this is therefore relatively accurate when considering the scope from the lowest count (July-13: 210) and the highest count (March-12: 387). Taking the average count across the 23 months leading to December 2013 will give a figure of 277.4, therefore the model-based prediction is an improvement on this method. This is one particular case however, and it must be tested on a larger scale.

Testing the linear regression function in a similar way but with less input features was also interesting to observe the differences in accuracy. It is possible that with too many input variables, the model could suffer from overfitting. Model overfitting is described [7] as “The use of models that violates parsimony - that is, that includes more terms than necessary”. It is possible that the relationship used to determine the independent variable could be captured by less features than was initially attempted. The process was therefore repeated, however this time using only counts of burglary during 2013, excluding the monthly counts from 2012.

The figure below shows the resultant regression model when using data only relating to 2013. Once again the prediction class will be counts of burglary in December. Similarly, a 70% Percentage Split has been implemented in order to evaluate the success of the model.

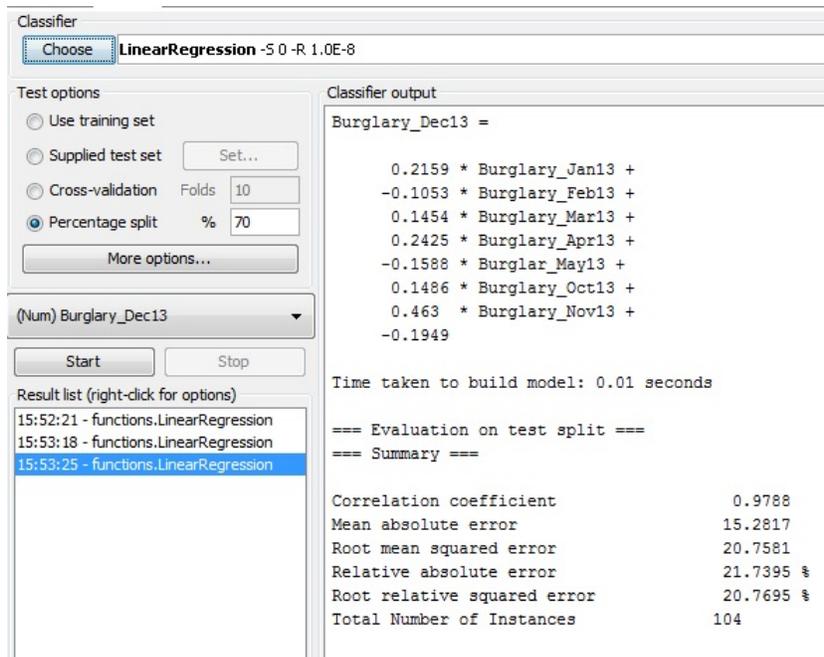


Figure 35: Using Less Input Features for Linear Regression

The function has produced a more simple model due to less input variables being provided in the training data. The evaluation on the test split shows a mean absolute error of 15.2817 and a correlation coefficient of 0.9788. The correlation coefficient describes the level of dependence between an independent variable(s) x and the dependant variable y , and a positive relationship suggests the variables share a linear relation. Essentially this means the x and y variables increase in a uniform manner in the same direction. In comparison to the test results when 23 months of features were used, this is an improvement of both terms. This is due to the decrease in mean absolute error, and strengthened correlation coefficient. It is possible to say that the previous model did suffer from overfitting, and less input variables can be used to determine a more accurate relationship between the input variables and the predicted class.

In order to analyze the overall success of the model, it was necessary to implement a significant test. An alternative method of obtaining an estimation, would be to total the counts in each month and take an average for the forthcoming month. By looking at how the predictive model performs against this alternative method in all 348 locations, it would be possible to say whether linear regression is a more useful approach to predictive policing.

In an excel spreadsheet containing corresponding counts of burglary for all locations on a monthly basis through 2013, two extra columns were created. One column would be the estimate for burglary counts in December, calculated by adding the counts from January to November and dividing by the 11 months to get an average. The other column would be the model-based prediction, calculated by implementing the model as a formula. It would then be possible to analyse which of the figures was closest to the actual number of burglaries for each location in December.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
LSOA Location	Burglary	Average Prediction	Model-Based Prediction												
Adur	25	24	19	21	41	30	28	32	25	19	28	25	29.2	19.8071	
Allerdale	40	30	53	38	55	46	71	45	54	53	42	25	52.7	40.7911	
Amber Valley	96	71	83	91	103	98	74	82	86	107	67	63	95.8	77.7557	
Arun	76	63	56	53	52	68	61	55	55	56	70	40	66.5	63.0485	
Ashfield	98	84	86	76	98	81	95	84	123	104	105	60	103.4	91.5595	
Ashford	101	50	59	51	87	64	54	71	75	67	87	75	76.6	73.7137	
Aylesbury Vale	56	81	71	91	114	96	70	76	77	64	61	60	85.7	55.4073	
Babergh	50	51	48	52	41	45	61	53	47	34	44	47	52.6	43.7326	
Barking and Dagenham	264	250	245	180	171	143	158	149	130	192	197	141	207.9	202.3381	
Barnet	463	398	395	362	338	271	303	273	300	333	395	442	383.1	381.7698	
Barnsley	193	204	221	263	176	185	223	181	160	165	176	186	214.7	193.9617	
Barrow-in-Furness	32	36	29	28	30	31	31	18	28	40	38	24	34.1	32.7037	
Basildon	151	140	128	108	140	111	157	108	105	134	170	171	145.2	138.8556	
Basingstoke and Deane	59	64	81	54	49	61	63	86	52	62	79	55	71	68.6854	

Figure 36: Analyzing the Success of Model-Based Predictions

The figure above shows cells predicted most accurately by the model marked as yellow. The formula for the currently selected cell can be seen in the sum bar, implementing the model that was produced in figure 35. In this particular case, 215 of the 348 locations was predicted more accurately using the linear regression model. This means that the predictive model produces the best results 61.8% of the time in this case. Appendix D contains the full results of this test, and the following page contains a graphical view of the data. Considering the predictive model is most accurate nearly two-thirds of the time, it is reasonable to assume it is the most precise way of estimating the number of burglaries for a forthcoming month based on test data.

7.1.2 Combining Multiple Features

In the example case detailed above, the only features used correspond to the specific crime type that is being predicted. It is possible to combine input features, such as counts of burglary and drugs offences, in order to observe if there is a knock-on effect between violations. It is plausible that there is a correlation between multiple criminal offences, and this can help improve the regression model. This is mentioned in the future work section as something that is definitely worth investigating. Although unable to find a direct relation between various offences as of yet, with more time to explore all possibilities, it is certainly feasible. Appendix E demonstrates an example data set including 25 features, such as population estimates, 12 months of counts for drug charges, and 12 months of counts for burglaries. Unfortunately, while the classifier does build an alternative model from the data, the accuracy was not an improvement to the example detailed above.

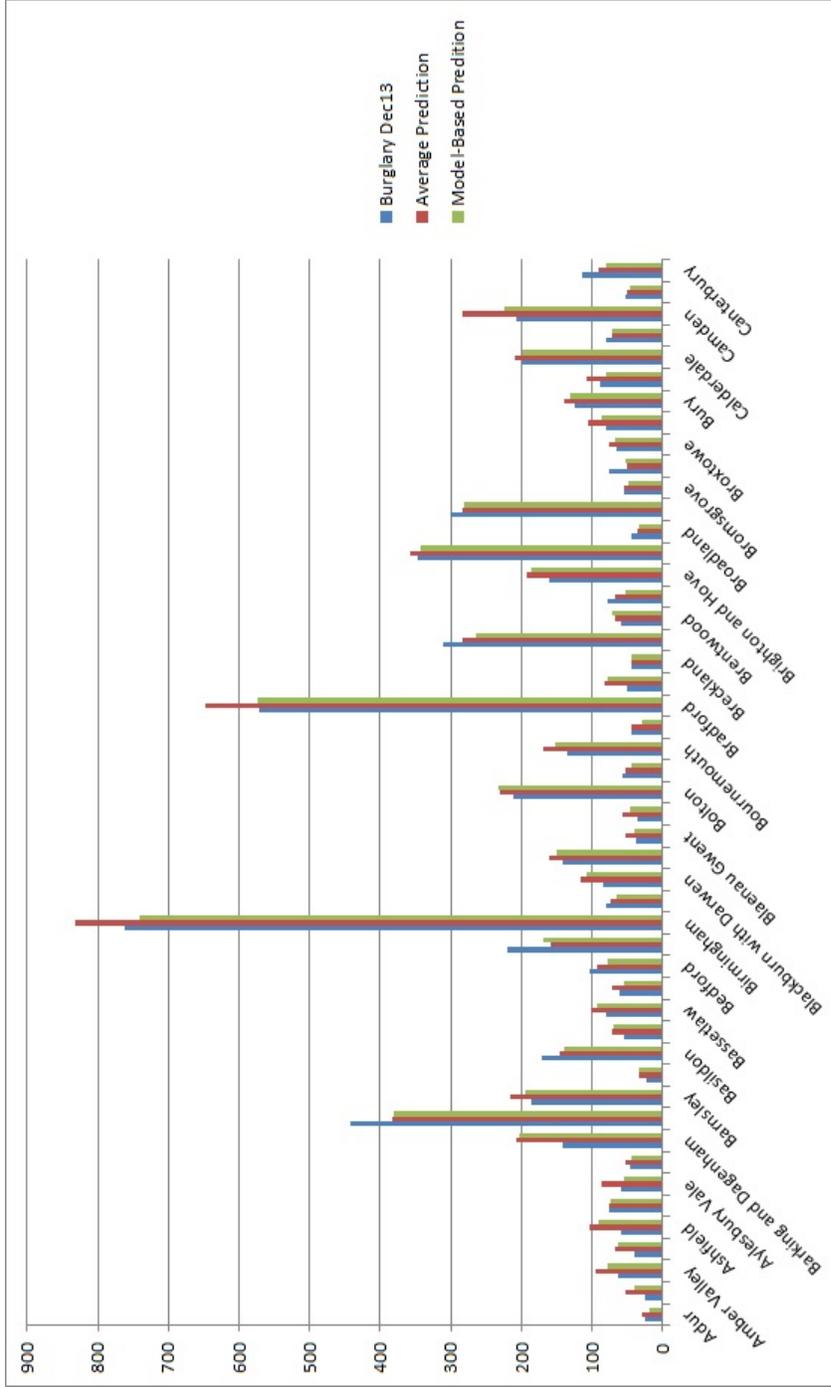


Figure 37: Success of Predictive Model Against Alternative Method

The chart shown here details a column for three elements, the actual count of burglaries in 2013, the predictive model estimation, and also the prediction made by taking an average. Unfortunately it was impossible to scale all 348 locations onto a single chart, however a larger version can be found in sheet 2 of the spreadsheet within Appendix D.

7.2 Tag Clusters Scraped from Flickr

Having created the program to scrape tags from photos taken in the UK, it was utilized to gather a number of significant tag-clusters for various locations. Due to the time taken to scrape tags, and the restrictions placed on the Flickr API, it was impossible to gather tag-clusters for all 348 defined locations. Attention was therefore focused on a subset of locations which had a highly contrasting ratio of crime per-population. The total number of tags collected for all 30 locations within the subset was 824,935. Each location had tags scraped from photos between January and December of 2013.

Each of these tag clusters was analyzed in order to identify tags with maximum chi-square statistics. On ordering the tags via their corresponding chi-squared value, it is instantly noticeable that they need to be filtered. The most common types of tag are either location-specific, such as landmarks or events, corporate or personal, such as a business name or personal photography site, or technology-related, such as the camera model that has been used to take the photo. The problem with these types of tag are that they offer very little by way of a relation to criminal activity. An example of terms appearing at the top of the chi-squared test can be seen below. The figure displays top 5 scoring tags following the chi-squared test for 5 random locations prior to any filtering.

Liverpool	Lincoln	Norwich	Chichester	Winchester
liverpool	newcastle	norwich	goodwood	winchester
merseyside	newcastle upon tyne	uploaded by flickr mobile	65	hampshire
england	go north east	flickrkiosapp filter nofilter	sony a65	winchester cathedral
albert dock	tyne	2013	alpha 65	hants
liverbird	go ahead group	norfolk	sony alpha 65	king alfred

Figure 38: Example Redundant Tags from Chi-Squared Test

As can be seen from the terms appearing above, the name of the location most regularly appears top. It is inevitable these tags appear high following the test, due to the common nature of establishing the place a photo has been taken. General location tags such as ‘united kingdom’ and ‘liverpool’ were ignored for the sake of attempting to identify terms that may be of significance to this project. Some location tags that gave more insight than purely a location, such as ‘coventry airport’, did remain for the purpose of them offering more than a single tag ‘coventry’. This is because had all locations with high crime-rates contained a frequent tag relating to their respective airport, it is possible this would indicate the airport as a hub for criminal activity. Obviously this conclusion could only be considered if the tag did not appear frequently in regions of low-crime containing airports.

7.2.1 Regions with Low Crime Ratio

Following a filter of tags deemed useless or irrelevant, it was possible to look at the top 20 tags appearing in locations, and see if there were common features amongst those regions with low crime, and those regions of high crime. It was hoped that a common tag would feature amongst those in the same class of criminal activity. Initially, it was necessary to look at tags appearing in locations of low crime rates per-population. The table below shows the top 20 tags with maximum X^2 score following the filter process.

Horsham	Winchester	Coventry	Dover	Stafford
ravelry	copyright	wyken	white cliffs	cross country
knitting	street theatre	hart	dover castle	show
natural world	winchester cathedral	boddle	scum run	motorcycle
wildbirds	prin	wilds	well child	mechanics
redpole	chesil	natural	team axles of evil	classic
greattit	british history	coventry cathedral	cougar	townscape
pixel	anglo saxon britain	standard	cliffs	showground
chaffinch	norman england	telephoto	ferry	ravilious
woodpecker	anglo saxons	is	sea	lyme
longtailedtit	historic britain	life	memorials	production
longtailed	historic england	britian	dover harbour	cattle
mascot	picture of the day	wild	war graves	lee
carfax	flower festival	coventry airport	st james cemetery	prize
nuthatch	english history	airplane pictures	world war 1	cow
rotary	anglo saxon	cvt wings	port	farmer
fullframe	creating balance	airplane photos	coast	award
cats	creating balance project	aircraft pix	ocean	under
robin	strong island	aircraft pictures	ww1	marshes
pole	jonas	general aviation	douvres	pig
back garden	cameras	tokens of love	channel	freelance

Figure 39: Top Tags for Regions with Low Crime Ratio

Figure 40 shows tags with top scoring X^2 statistic for locations that have a low crime count per population. Unfortunately there is no instantly recognisable tag amongst all locations, however there are a number of similarities. Many of the tags relate to historical matters, or wildlife and nature, sharing very similar connotations while not having exactly similar tags. The word ‘cathedral’ appears in both Winchester and Coventry, while ‘natural’ and ‘wild’ appear in both Horsham and Coventry. The general theme of nature can be displayed in all locations, with each including some tag relating to wildlife or the landscape. While this is not particularly insightful for helping make predictions, it may explain as to why these regions contain less crime. Crime more frequently occurs in busy locations, close to city centres where opportunities are more evident. The tags shown in figure 38 suggest the opposite of this, whereby tags most commonly relate to sightseeing and the living world.

It is still evident that some useless and invaluable tags appear in the lists above. I believe that the reason some redundant data such as ‘creating balance project’ appears is due to the lack of data collected. If somebody who frequently uploads to Flickr tags all of their photos with some relatively meaningless tag, it is likely to appear high following the chi-squared test. This is simply because not enough data has yet been collected to eliminate such tags completely. As the size of a data set increases, the more valuable the chi-squared test can prevail. Given extra time to spend collecting data, the more likely it would be to identify a common tag amongst all these locations.

7.2.2 Regions with High Crime Ratio

It was deemed necessary to analyze the top tags appearing in locations with a high crime ratio, in order to compare results against those regions of low crime. Once again, it was hoped a common tag or term would be frequent amongst these locations, which would not be evident within low-crime regions. The figure below shows top 20 tags for five regions that have a high crime rate per-population.

Blackpool	Manchester	Middlesbrough	Lincoln	Liverpool
seaside	refugee	drifting	prison	liverbird
silver	conference	motorsport	visitor	marathon
proof	artists	autodrome	miles	another place
obverse	event	drift	snap	world heritage site
reverse	gay village	sideways	spectacular	mersey
coin	mega	battle	rooftops	pier head
money	world aids day	santa	climb	crosby beach
coins	aids	odt	see	crosby
cratmanship	fields	beautiful woman	look	bisexual
ornate	metrolink	model photoshoot	steep hill	cancer research
one	greater	centre square	witham	action on cancer
gold	trim	percy	shot	transgender
winter gardens	whatever	model girl	campus	lesbian
precious	old trafford	bmx bikes	middle ages	beach
ballroom	christmas market	charlie chaplin	fa cup	arctic heroes challenge
dollars	city airport	town meal	steps	the prefects
macro	first cut	punch&judy	christian	arctic ball
mile	wtf	fruit&veg	houses	gay
certificate	misc	beamish	terrace	pride
circus	christmas markets	disabled	clear	help the heroes

Figure 40: Top Tags for Regions with High Crime Ratio

Figure 41 shows resultant tags following the filter process. Unfortunately no common tags are frequent amongst all locations, however there are once again some similarities. Manchester and Liverpool both contain tags relating to homosexuality, while Blackpool and Liverpool both contain a reference to the ‘seaside’ or ‘beach’. As opposed to the tag clusters for low crime regions, there is less of a common theme amongst the five locations. Each location tends to have their own theme, for example Lincoln includes tags such as ‘houses’, ‘rooftops’ and ‘terrace’ which suggest urban-life. Blackpool is a location with highest crime rate per-population (excluding London), and contains a number of tags relating to ‘money’, ‘gold’ and ‘coins’. The frequent tagging of these terms may indicate why the region is subject to particularly high criminal activity.

The issue of tags appearing because some user on Flickr frequently uses the same tag is also evident here. For example in Middlesbrough, the tags ‘beatiful women’, ‘model photoshoot’ and ‘model girl’ are more than likely tags applied by the same user who regularly uploads photos. This is one of the major issues as to why it is difficult to make any definitive conclusions at current. With larger sets of tag-clusters for each location, it would be possible to eliminate such tags from being considered.

As a further test, it was possible to look at the monthly appearance of the tag ‘seaside’ in Blackpool, to see if it fluctuated in a similar manner to any specific crimes. This would allow possible identification of whether a tag is indicative of the some crime-type. The figure below shows the most similar relationships found.

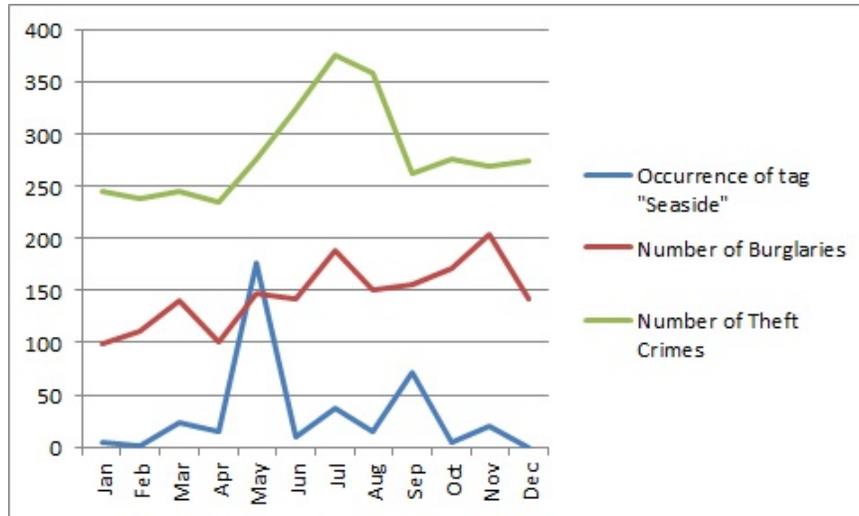


Figure 41: Occurrence of Tag in Relation to Crimes in Blackpool

Despite the obvious spike in May, where possibly a single user repeat-tagged a large number of photos, the appearance of the tag shares a similar relationship to the number of burglaries committed. This can be seen via the lines in the graph moving in roughly similar directions throughout 12 months. Both the appearance of the tag and the number of burglaries seem to increase as the year progresses, before dropping in the final months. The peak in July is very similar, before diminishing a small amount and then increasing once more. It is impossible to say from the current graph whether the appearance of this tag definitely implicates a correlation between the number of burglaries committed in Blackpool. It does however, suggest that it's certainly a possibility that is worth exploring given more time.

7.3 Can These Tags Aid in Making Predictions?

From using the data collected during this project so far, it is very difficult to conclusively express a relationship between the appearance of a tag, and it being indicative of some criminal activity. Had some tag appeared high in the chi-squared test for multiple locations with a high crime ratio, it would be very interesting to observe the frequency of the tag in those regions of low criminal activity. This is currently not feasible due to the lack of identifying a tag particularly recurring for locations with similar crime-levels. It is plausible that with larger tag-clusters for locations stretching across the UK, it would be possible to identify such tags.

8 Strengths & Weaknesses of Current Implementation

It is important to consider whether the system and components that have been created function as initially intended. Identifying the successful elements and issues with the current system is useful for a critical evaluation.

8.1 Strengths

The first main feature of this project was to apply linear regression to build a predictive model for crime in the UK. The results section details an example case for predicting the number of burglaries in any region in the UK, and in particular how to apply the model to Cardiff. In order to test whether the approach for making estimations was reasonable, it was vital to compare results to those calculated by taking an average count across the input variables. Results show that 61.8% of the time the model is more accurate when considered over all 348 locations. This is an obvious improvement and therefore a more beneficial way of making predictions. It also demonstrates multiple linear regression as an effective method of tackling the issue of predictive policing.

The flexibility that the system offers is a considerable strength. By importing all police statistics into the database, simple queries can be made to obtain data relating to any particular scenario. This may be data applying to a specific location, or relating to a particular type of criminal offence. By including other information that was not used during this project, such as longitude and latitude of crimes, it also shows potential for future related work.

The second feature of this project involved scraping tags from photos, and analyzing the collected data to find terms most discriminative to locations. Over 800,000 tags were collected in total by the program to scrape Flickr. Spatially-relevant terms can also be identified via the chi-squared term selection process. This is a strength as the desired functionality has certainly been achieved. The graphical interface that has been developed to manage scraping tags from Flickr also supports the possibility of anyone being able to operate the system.

8.2 Weaknesses

The primary weakness of the current implementation is the lack of being able to aggregate spatially-relevant terms into the predictive model. The absence of any tag appearing persistently across multiple locations made it difficult to suggest a relationship between a single tag and some crime activity. The current set of tag-data gathered is not big enough for the chi-squared test to produce intuitive results. Even following the filtering of useless tags, such as place names and camera models, an issue arises with tags that single user's frequently apply. If one Flickr user uploads 100 photos per-month and registers the same tag on each photo, this tag is destined to receive a high chi-squared statistic in the corresponding location. It is then a further issue trying to differentiate between legitimate high-scoring tags, and those that appear high for false reasons.

Another limitation of the current system, relates to the way in which tags are collected via the web application. Initially, I decided the flexibility to open a web page, specify a location and date-period was the best way of acquiring tags. While this is simple and supports my idea of anybody being able to utilize the system in the future, it is perhaps not the most efficient. Implementing a crawler that can be left collecting data for large amounts of time, would require less human interaction.

9 Future Work

Throughout the course of this project I have considered a number of possible ways that the task could be furthered and explored. These suggestions have either dawned upon me through execution of the task, or are things I would have liked to explore given a larger time frame.

In the results section of this report I detail building a regression model based on input variables. The types of features used were counts regarding the same crime-type in unique months. Exploring the accuracy of models with varying input features, such as the combination of counts of burglary, drug and theft crimes, could open a lot of opportunity. It is very possible that some crimes have a knock-on effect that lead to further violations, and this is a relationship that would be interesting to explore. While I did attempt to include various features, I was yet to uncover a direct relation between two types of offence. With more time to explore this possibility, I believe it would be feasible to identify a correlation. The possibility of looking at other features, such as unemployment rates, race groups and average income could also prove worthy future research. It is highly likely that factors such as these have relations to the number and type of offences committed in many locations.

The collection of greater tag-clusters for all locations in the UK would allow for a more significant chi-squared test. Larger amounts of data would eliminate those tags scoring highly for false reasons, such as a single user repeat-tagging. Attempting to identify a common tag or phrase concurrent in regions of similar criminal activity, will allow for a comparison between the occurrence of the tag in locations with contrasting levels of crime. The possibility of adding the frequency of a tag in particular months could be aggregated as a variable for linear regression. It is possible this feature could improve the accuracy of the predictive model. For example if locations with high criminal activity all contain the tag 'BMW', it would be possible to look at the frequency of this tag in all locations throughout different months of the year.

Another potential extension to this project, involves looking at crime shifts between neighbouring cities and towns. The current implementation does not consider the effect that crime in one city may have on one that it borders. Cluster analysis such as k-means could be explored in order to try and group together criminal activity within a certain demographic. Observing the effect of data-mining when working only with locations in a close proximity would be an interesting comparison. It is possible this could be a more effective way of creating a predictive model, as opposed to looking at the UK as a whole. By mapping committed-crimes to clusters it would also be possible to determine hot-spots of criminal activity. Identifying hot-spots of particular types of crime could be an interesting extension, for example places in South Wales particularly subject to vehicle crimes.

The obvious issue with collecting tags that are irrelevant in relation to predicting crime is evident within this project. The current implementation does not have a rigid filtering system that can disregard terms before they are collected. By applying an extension to the program to scrape tags from Flickr, it would be possible to only acquire potentially insightful tags. Writing a function to read each tag as it is collected, and identify whether it is a non-significant date, place name, or hardware-related tag would be much more efficient. No manual filtering and removing of tags would then have to be done, and the chi-squared test would only be performed on terms of interest.

10 Project Conclusions

In order to conclude as to the overall success of this project, it is important to look back at the original aspirations. Looking at the core objectives identified in the initial report, the following can be summarised.

10.1 Have significant crime statistics been gathered to form the basis behind a predictive model?

This was an initial objective marked as incredibly key in the development of this project. The publicly available data provided by data.police.uk fulfilled this requirement, and has been of great importance. By importing the data into a database it allowed for queries to be executed to acquire features for linear regression. Multiple years worth of data were collected and organised as monthly clusters, which allowed for identification of specific counts of crime in respective months.

10.2 Has a vast collection of tags been scraped from photos in specific locations via Flickr?

As a second primary goal to this project, the ability to scrape tags from photos in specified locations was marked as necessary. Achievable through the Flickr API, the creation of a web application to carry out this process has been implemented. Using this tool, a total 824,935 tags were collected for 30 determined locations. While an even larger set of data for more than 30 locations would have been preferable, the tool has been produced and I was only limited by the time period allowed to complete the project.

10.3 Has term selection been applied to identify spatially-relevant tags and remove redundant or irrelevant ones?

Through deployment of the chi-squared term selection process, it has been possible to identify tags most discriminative to the respective locations they appear in. By calculating the statistics for all tags it has been possible to rank them in order, and instantly recognize those of interest. The filtering process, although time-consuming and currently just completed manually, did remove irrelevant landmark, date, or hardware related tags. There is still an issue with tags appearing high following the chi-squared test due to a lack of data, and user's repeat-tagging some phrase or term.

10.4 Has it been possible to build a regression model that can predict the number of crimes in some location?

Through research and understanding of multiple linear regression, I have managed to demonstrate the possibility of using this process to predict crime for an unknown month. The model can be applied to any location, and the results section shows applying the model to Cardiff to determine the number of burglaries in one month. The extensive database created means that similar techniques can be used to estimate any crime-type for any specific location.

10.5 Can tags from Flickr be used to improve the accuracy of a predictive model?

From the current implementation of this project it has not been possible to conclusively say whether meta-data from Flickr is beneficial for this purpose. I could not say that this type of data is useful from my findings, however with larger data sets I can suggest it certainly is a possibility. The results of my project show that tags in locations of similar crime rate per-population, do share some common terms or themes. If an exact tag was to appear frequently in a cluster of these locations, it is possible that observing the prevalence of the tag across various months may prove useful as a feature for linear regression.

10.6 Summary

Looking at the initial aims and objectives prior to the launch of this project, I feel I have met all of the functional requirements. The development tasks have been implemented in order to obtain precisely what was desired. This includes the database of police statistics, the program to scrape tags from photos on Flickr, and the program to identify spatially-relevant terms. Only the restricting elements such as the Flickr API request limit and the time-frame of the project has stopped me acquiring the level of data that was desirable.

The system as a whole consists of multiple components, all designed to be flexible and easy-to-use. The graphical interface that can be used to collect tags from Flickr makes that process very simple. Creating a data-set from the database does not require expert computer skills, just a basic understanding of how to implement SQL queries. Looking back at the target audience for this project, I think it is fair to assume police and security companies could technically utilize this system.

The possibility to make predictions regarding crime in unforeseen months has been demonstrated through understanding and application of multiple linear regression. The success of predictive modelling has been shown, proving this is an effective method that could be used in real world situations. There are endless possibilities for identifying input features that mark strong relationships towards predicting crime. Once again, time was the only limiting agent on testing the effect of various different independent variables.

Although the project did not determine a significant use for location-relevant tags, I believe this is primarily due to a lack of collected data. Considering the tools implemented, it is possible to continue the research very easily. Being able to incorporate knowledge discovered from data-mining large clusters of tags, has shown potential to be worth exploring. Although nothing conclusive can be said regarding the impact of tags currently, I have still fulfilled the primary aim of being able to identify those most discriminative of a location.

11 Reflection on Learning

Multiple skills have been tested in this project that required a wide variety of research methods and application. The general process has involved working to strict deadlines in order to meet deliverables, while attending supervisor meetings to frequently discuss progress. Managing time effectively has been vital and encouraged me to do so. As a whole I think it has benefited my learning a huge amount and made the prospect of future research much less daunting. Breaking the overall solution into smaller and more manageable segments helped disperse the work load, taking one aspect as it came

11.1 Improvement to Programming Skills

Programming skills have been tested in multiple languages and environments in order to develop the tools necessary. The Java programming language was selected to handle the calculation of chi-squared statistics. I chose this due to having experience using it for importing and exporting files, something necessary in order to read in text files and output results in csv format. However, I feel that through utilization of hash maps in order to index tags, and the implementation of complex formulas, my Java skills have excelled even further. This can be said equally for my web-based programming skills. Although I had a good understanding of HTML and CSS techniques, I had little experience developing an application using an API. The Flickr API was excellent for grasping techniques that involve requesting data and delivering that information back to a third-party. I have also gained a strong understanding of the JSON data format and how it can be used to parse data.

11.2 Attention to Mathematical Theory

This project has also required a degree of mathematical understanding. This is a matter I did not feel particularly confident with, and was my biggest worry prior to beginning the task. The prospect of multiple linear regression and the chi-squared formula are the elements I am referring too here. However on reflection, with suitable time set aside in order to research and understand these matters, it has filled me with confidence. By abiding to my weekly plan detailed in the initial report, I was able to comprehensively investigate sources and examples, without leaving myself behind schedule at a later point.

11.3 Handling Big Data

The handling of big-data and SQL has also been demonstrated via this project. Creating a database to store large amounts of records requires some way of obtaining relevant information. Although I had experience using SQL, I did employ some new queries that I hadn't previously attempted. For example updating an entire column to remove trailing digits, executed by setting a sub-string that removed characters from the end of relative fields. This and some other advanced SQL queries have given me a much more in-depth knowledge of the possibilities the language can provide.

11.4 Decision Making & Approach

Another important way in which the project has helped with developing my personal skills, relates to making decisions in the interest of reaching desired goals. Often things take longer than expected or some constraint is realised, which requires a re-think about the method in practice. Making decisions that are in best interest of meeting original aims is a crucial process. Commitment to agile software development has strained my managerial capabilities by the consistent need to re-evaluate. I could have spent far longer collecting larger clusters of data from Flickr, however I may not have completed the tool to identify spatially-relevant tags. However with all development stages complete, the prospect of continuing the research at a later date is much more viable. My ability to make these important resolutions have been challenged by the project, and forced me to focus my attention on exactly what I set out to achieve.

11.5 Overview

This project has challenged the integration of skills and best practices. Being able to combine learning from various aspects such as development, research, planning and organisation was vital in order to achieve success. Attention to detail was crucial, while focus on end-goals was persistently considered. The project has been an enjoyable way of testing personal ability and willingness to learn.

12 Glossary

Term	Definition
Array	Data structure containing a series of elements. Most usually all elements are of the same type.
Chi-squared Test	Statistical test used to compare observed frequency of data in relation to expected frequency according to some hypothesis.
Data Scraping	The extraction of data or information, usually done by automation of a computer program.
Hashmap	Data structure that implements key-value pairs in order to store and index data.
Linear Regression	Statistical approach of modelling the correlation between a dependant variable and either a single or multiple independant variables.
Linked List	Data structure that contains an ordered sequence of elements, whereby each contains a reference to its precursor.
LSOA	Lower Super Output Area. Defined areas within the UK that can be identified for the purpose of gathering national and neighborhood statistics.
Machine Learning	The study and implementation of systems designed for machines to learn without distinct programming.
Predictive Policing	Forecasting future criminal activity using advanced data mining and analysis techniques.
Query	A request for some information, for example, relevant data from a relational database.
Tag	Metadata assigned to information, such as a photo on Flickr, to describe or identify an item. Increases the possibility of the information being found via searching.
Weka	Software designed by the University of Waikato that provides a selection of machine learning opportunities and visualization tools.

13 Table of Abbreviations

Abbreviation	Definition
ARFF	An ASCII text file that is the default format for data within Weka. The format is used to represent a number of instances that are allocated the same attributes.
API	An Application Programming Interface details how software components can communicate through a set of functions or operations.
CSV	Plain text representation of comma-separated values.
SQL	Structured Query Language is a programming language used to handle and manipulate data residing in a database.
URL	The purpose of a Uniform Resource Locator is to identify a network resource via a human readable text string.

14 Appendices

Below details what can be found in the related appendices submitted with this document.

Appendix A: Spreadsheet containing list of all 348 LSOA locations and their corresponding crime rate per-population.

Appendix B: Program containing two versions of software designed to scrape tags from Flickr.

Appendix C: Program containing software to calculate Chi-square statistics.

Appendix D: Spreadsheet containing results of test regarding predictive model against taking average count.

Appendix E: Example data set containing 25 combined features.

References

- [1] Data.police.uk, (2014).
Available: <http://data.police.uk/data/>
Last accessed 04/04/2014.
- [2] Tom Porter. (2013). "London Police to Adopt Crime-Predicting Computer Package"
Available: <http://www.ibtimes.co.uk/predictive-policing-predpol-future-crime-509891>
Last accessed 04/04/2013.
- [3] K. Beck, e. al. (2001). "Manifesto for Agile Software Development"
Available: <http://agilemanifesto.org/principles.html>
Last Accessed 04/05/2014.
- [4] Guy Smith. (2014). "How predictive policing works in the US"
Available: <http://www.bbc.co.uk/news/uk-england-london-26837251>
Last accessed 07/04/2014.
- [5] PredPol, Inc. (2014). "Looking Ahead, Not in the Rear View Mirror"
Available: <http://www.predpol.com/technology/>
Last accessed 07/04/2014.
- [6] Olivier Van Laere, Jonathan Quinn, Steven Schockaert & Bart Dhoedt. (2014).
"Spatially Aware Term Selection for Geotagging" IEEE Transactions on Knowledge and
Data Engineering. 26 (1), 222-223.
- [7] Douglas M. Hawkins. (2004). "The Problem of Overfitting"
Available: <http://www.cbs.dtu.dk/courses/27618.chemo/overfitting.pdf>
Last accessed 27/04/2014.