Research Software Engineer Campaign - Jobs Analysis

Shicheng Zhang

August 2016

M.Sc. in High Performance Computing with Data Science

The University of Edinburgh

Year of Presentation: 2016

Abstract

This M.Sc. project discusses the implementation of classifier to categorize jobs, collected from jobs.ac.uk, into software jobs and non-software jobs. A suitable definition of software jobs is identified, based on which a training set and a feature set are developed. The classifier is implemented with three different machine learning algorithm including Decision Tree, Support Vector Machine and Multinomial Naive Bayes. Through evaluations on the classification result with a Confusion Matrix, the most suitable algorithm among the three for this project is identified, namely Support Vector Machine. With the identified software jobs, some analysis is done to obtain some characteristics about software jobs in general.

Table of Contents

List of tables

List of Figures

Acknowledgements

I would like to thank my supervisors, Mario Antonioletti and Gavin J. Pringle, for their immense help.

1 Introduction

This dissertation project was established based on one of the interests of the Software Sustainability Institute (SSI)[1], which is investigating how academic software jobs, see below for a definition, contribute directly to scientific research. By jobs in this context we mean adverts for regularly paid positions or posts. Jobs scraped off a jobs website are passed through a classifier, which determines whether a job is a software job or not. Classifiers are implemented using three different algorithms in order to make a comparison of the results, so that the most accurate algorithm for the classification can be found. The extracted jobs are analysed to obtain descriptive information of those jobs.

Definition 1: Software jobs – Jobs and studentships that require candidates to write or modify scripts or codes. Jobs that require programming skills but whose job requirements do not directly involve writing software are not defined as software jobs. Other jobs, not requiring software skills, are classified as non-software jobs.

1.1 Background

1.1.1 Software Sustainability Institute

The development of computational technology has enabled scientific researchers to solve complex problems and conduct experiments and simulations that cannot be done without computers. In many cases, codes and scripts act as accelerators in the advancement of various disciplines. Researchers can benefit from well-developed programs, which are understandable, extendable and can be used by other people. The Software Sustainability Institute is funded to help researchers write better codes and scripts in their projects to reduce duplicated work and enable sustainable research. Knowledge of scientific software jobs may facilitate the SSI to progress in its goal, with answers to such questions as what the most popular programming language in the market, the SSI can get an idea of the size of its market.

[1] See: https://www.software.ac.uk, last accessed 18 August.

1.1.2 Job advert

Data Source

Jobs.ac.uk[2] is a recruitment website for jobs in academic, research and science professions where universities and organisations can promote their job adverts. A lot of the jobs, which the SSI is interested in, can be found on this website. Hence, this project has used the job data from jobs.sc.uk as data source.

Job advert component

A job advert provides information that presents a job vacancy that includes studentships. On jobs.ac.uk, each job advert is displayed on a web page and is given a job id. The job id is unique to jobs.ac.uk and can be used to identify each job. In this project this unique id is referred to as the job id. The job id is not written directly in the job advert but appears in the Uniform Resource Locator (URL)[3] address. For instance,

http://www.jobs.ac.uk/job/**ANX622**/research-associate/

Jobs that have passed their closing date are removed, each job has a lifespan on the website. At the time of writing, the above URL represented a job at the University of Edinburgh. In the URL, the ANX622 token represents the unique job id for the job in jobs.ac.uk. The actual job advert can be divided into five sections:

1. the logo of the employer,
2. a job title,
3. the job information,
4. a job description and
5. job metadata.

The job advert whose id is ANX622 is shown in Figure 1 to graphically demonstrate these elements in the job advert.

[2] See: http://www.jobs.co.uk, last accessed 18 August.

[3] See: https://en.wikipedia.org/wiki/Uniform_Resource_Locator, last accessed 18 August.

Job title – The title of the vacancy. From the job title people can get an overview of what the job is about. For instance, a job with a title of "PhD studentship in Radar Imaging" indicates that this is a PhD programme in a radar imaging discipline. A job with a title of "Lecturer in computing" suggests that it is a lectureship opportunity.

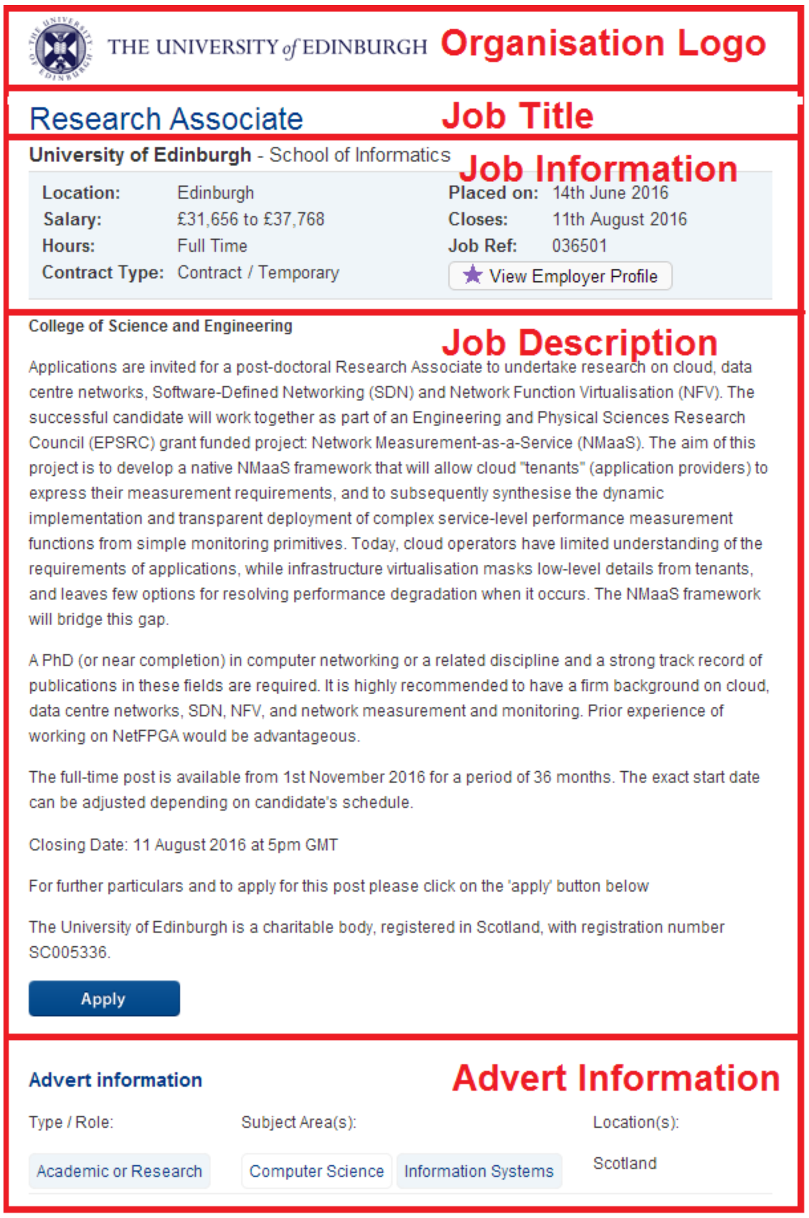
Job Information – Relevant information about the job vacancy including:

- employer – the organisation or institute that provide this job;
- location – the specific location of the job;
- salary – the income of this job;
- work type – whether this is a full time job or part time job;
- contract type – whether this job is a permanent job or temporary;
- time of publish – the date on which this job advert was published;
- Closes – the due time of application;
- job reference id – if the job is also published on the employer's website, this is the corresponding id of this job on that website.

Job description – The job description provides more detailed information about the job. Readers can get information about the employer, what is required to do the job, what the job involves and what the job benefits are, etc.

Advert information – This field contains categories determined by the jobs.ac.uk website which show:

- role – the role which the candidate will perform,
- subject areas – which subject area(s) this job comes under,
- location – a general location for the job, such as South east of England and Midland of England



THE UNIVERSITY *of* EDINBURGH

Research Associate

University of Edinburgh - School of Informatics

Location:	Edinburgh	**Placed on:**	14th June 2016
Salary:	£31,656 to £37,768	**Closes:**	11th August 2016
Hours:	Full Time	**Job Ref:**	036501
Contract Type:	Contract / Temporary		★ View Employer Profile

College of Science and Engineering

Applications are invited for a post-doctoral Research Associate to undertake research on cloud, data centre networks, Software-Defined Networking (SDN) and Network Function Virtualisation (NFV). The successful candidate will work together as part of an Engineering and Physical Sciences Research Council (EPSRC) grant funded project: Network Measurement-as-a-Service (NMaaS). The aim of this project is to develop a native NMaaS framework that will allow cloud "tenants" (application providers) to express their measurement requirements, and to subsequently synthesise the dynamic implementation and transparent deployment of complex service-level performance measurement functions from simple monitoring primitives. Today, cloud operators have limited understanding of the requirements of applications, while infrastructure virtualisation masks low-level details from tenants, and leaves few options for resolving performance degradation when it occurs. The NMaaS framework will bridge this gap.

A PhD (or near completion) in computer networking or a related discipline and a strong track record of publications in these fields are required. It is highly recommended to have a firm background on cloud, data centre networks, SDN, NFV, and network measurement and monitoring. Prior experience of working on NetFPGA would be advantageous.

The full-time post is available from 1st November 2016 for a period of 36 months. The exact start date can be adjusted depending on candidate's schedule.

Closing Date: 11 August 2016 at 5pm GMT

For further particulars and to apply for this post please click on the 'apply' button below

The University of Edinburgh is a charitable body, registered in Scotland, with registration number SC005336.

Apply

Advert information

Type / Role:	Subject Area(s):	Location(s):
Academic or Research	Computer Science, Information Systems	Scotland

Figure 1: An example of job advert.

Use of job adverts

In terms of determining whether a job is a software job or not, the job title section and the job description part are analysed to make the decision. The advert information is not used when making the decision because it is observed that jobs with tags such as "IT" and "Computer Science" do not necessarily identify software jobs according to the definition this project. Although the advert information seems to be good discriminator, we show in section 0 that this is not necessarily the case.

1.1.3 Data Description

Original Dataset

The original dataset was collected by one of my supervisor Dr Mario Antonioletti by scraping jobs from jobs.ac.uk. A scraper was implemented to collect jobs placed in the past 24 hours, which is executed at 5:00am daily. The scraper has been continually collecting jobs since September 2014. The scraper has two main contributions:

1. Download an original html fragment with the new job information.

2. Information from downloaded html fragment is extracted and organize into a comma separated values (CSV) file.

A CSV file can be perceived as a table. Each row in the file constitutes a job. Values in the different columns are separated by commas.

There are 24 columns in the original dataset including:

- JobId – the unique id that can identify this job on jobs.ac.uk;

- Title – the title for this job;

- Employer – the organisation of institute who wants to recruit people for this job advert;

- Location – the specific location of the job;

- InUK – does this job locates in UK, derived from data analysis;

- SoftwareJob – is this job a software job, derived from data analysis;

- SoftTermIn – where a term used to define a job as a software job[4] is found in the job. Values in this data can be N: none, T: title, B: body and TB: title and body. It is obtained from later analysis;

[4] The SSI determined whether a job is a software job by finding any of the following terms (case insensitive) in the job name and or description (where the "|" is taken to be an "or"): \`software developer\`|\`coding\`|\`coder\`|\`research software engineer\`|

- Salary –income for this job;
- SalaryMin – the minimum salary for this job, derived from data analysis;
- SalaryMax – the maximum salary for this job, derived from data analysis;
- Hours – this job is a full time job or part time job;
- Contract – the contract type of this job permanent or several years' contract;
- FundingFor – if this job is a studentship, what kind of student can apply for funding, e.g. UK Students, EU Students, etc.;
- QualificationType – if this job is a studentship, what degree will this studentship end up, e.g. PhD, International Doctorate, Integrated Masters / Doctorate, etc.;
- PlacedOn – the date on which this job is published;
- Closes – the closing date of application;
- JobRef – the job reference id if it is available on jobs.ac.uk;
- h1 – any level one headers in the html code;
- h2 – any level two headers in the html code;
- h3 – any level three headers in the html code;
- TypeRole – the role which candidates can perform, e.g. Technical, Clerical, Professional or Managerial, etc. ;
- SubjectArea – the subject that this job works in, e.g. Mathematics and Statistics, Mathematics, Creative Arts and Design, Design, etc.;

`software engineer`|`programming`|`programmer`|`Fortran`|`C++`|`Java`|

`JavaScript`|`Matlab`|`python|perl`.

- Location2 – the general location of this job. For instance, “South west of England” or “Midland of England”;
- Description – the detailed job description.

By 29th July 2016, there are 128345 jobs in the original dataset.

Training set

A subset of the original dataset, consisting of 1269 jobs selected at random, was manually classified by Mario Antonioletti to provide a training set. Those jobs are classified into three categories:

- “Yes” - research software jobs
- “No” - jobs are clearly not research software jobs
- “Maybe” - ambiguous jobs, they can fall in either “Yes” group or “No” group.

The criteria of distinguishing research software jobs, see definition below, and non-research software jobs is ambiguous. People have serious dispute on the definition of research software jobs. Hence, this project start with classifying jobs into software jobs and non-software jobs, whose difference is relatively clear. Under such circumstances, the training set should be modified to fit in this requirement.

Definition 2: Research software jobs - Jobs and studentships that require candidates to write scripts/codes for scientific research.

When modifying the training set, time taken by manually classifying jobs into software jobs and non-software jobs was counted. It showed that it took about less than 1 minute to make the decision that a job is obviously a software job and roughly1.5 minutes to identify an obvious/non-software jobs. However, there are a number of not so easy to classify jobs where it is difficult to make such a decision. When modifying the training set, all “maybe” jobs were supposed to be classified into software jobs and non-software jobs. These “maybe” jobs took at least 3 minutes each or longer as the job description fails to provide enough information to make a decision. For instance:

```
Job id: AKZ566

Postdoctoral Research Associate
```

```
A fixed-term postdoctoral research position is available for an
outstanding individual in the research group of Dr. Julien Michel at
the School of Chemistry of the University of Edinburgh, United
Kingdom. Research in the Michel group focusses on the development of
new computer-aided drug design methods and their applications to
structure-based ligand design problems. This appointment will be
working on the European Research Council funded grant: "Beyond
structure: an integrated computational/experimental approach to
ensemble-based drug design". This position is available from 1
September 2015 for a period of 24 months. Salary: £31,342 - £37,394
per annum Vacancy Ref: 033099 Closing Date: 22-MAY-2015 at 5pm GMT
For further particulars and to apply for this post please click on
the 'apply' button below

https://www.vacancies.ed.ac.uk/pls/corehrrecruit/erq_jobspec_version
_4.jobspec?p_id=033099 The University of Edinburgh is a charitable
body, registered in Scotland, with registration number SC005336.
```

The job description is ambiguous as it does not explicitly say what skills the candidate should have. This job mentions that the project will “focus on the development of new computer aided drug design method and their applications to structure-based ligand design problems”. We can hardly know to what extent that the process is “computer-aided” or whether it requires candidates to involve with the simulation or modelling of chemicals or doing chemistry experiments in a lab. Jobs like AKZ566 take longer to determine whether they are software jobs or not. There are a number of jobs that even a human being can hardly distinguish them based on job description, which is the biggest overcome of this project.

The training set is supposed to provide a sample of all the data that represents the original data set. The training set is processed to generate quantified training data. The classifiers used in this project can learn the rules to distinguish between software jobs and non-software jobs based on the training set data. After the modification, there are 1269 jobs in training set. 119 of them are classified as software jobs, 1150 jobs are classified as non-software jobs. As jobs in the training set are randomly selected, it can be treated as a sample of the original dataset, which means that about 12035 jobs (9.4%) in the original training set are software jobs.

Features

Features are used to discriminate software jobs are generated based on known programming languages. Each programming languages is a feature. The value of this feature in a job is the times that this particular feature is mentioned in a job. For instance, a job mentions the word “Fortran” three times. Then the value for the feature “Fortran” in this job is 3. The feature list is modified gradually. Each time the accuracy of classifier is evaluated. False positive jobs, i.e. jobs being classified as non-software jobs when they are indeed software jobs, are inspected to find out the possible reason. In one situation, an incorrect classification can be caused by one or more keywords not being included in the feature list. If this happens, the keyword is added to the feature list and the classifier is evaluated again with new feature list.

In this project, features can be single words or phrases. Single words of “application”, “development”, “software” and “engineer” may not necessarily relate to software jobs due to their different meanings in different context. However, “application development” and “software engineering” can be good indicators that jobs mention these two phrases are likely to be software jobs. Hence, phrases can be used as features. Detail operation of how to extract phrases is described in section 2.3.2. As different features have different contribution to identify software jobs, features are weighted. The weight of those features is described in section 1.3.3. The list of all features is shown in the appendix of section 6.

1.2 Overall aim and project objectives

1.2.1 Project aim

This project aims to investigate the feasibility of the following three machine learning classifiers: decision trees[5], Support Vector Machine (SVM)[6] and Naive Bayesian[7] to classify over 120 thousand academic jobs into software jobs and non-software jobs according to definition 1 in section 1. To compare the accuracy of these classifiers and the time cost of each algorithm, by using confusion matrix[8] defined later in this dissertation.

1.2.2 Project objectives

Modify and enrich training set

As this project started by classifying jobs into software jobs and non-software jobs, which is slightly different from the purpose of this training set which used research software/non-research software, the classification result had to be modified before using it to let machine learns how to classify the data into software/non-software

[5] See: https://en.wikipedia.org/wiki/Decision_tree, last accessed 18 August.

[6] See: https://en.wikipedia.org/wiki/Support_vector_machine, last accessed 18 August.

[7] See: https://en.wikipedia.org/wiki/Naive_Bayes_classifier, last accessed 18 August.

[8] See: https://en.wikipedia.org/wiki/Confusion_matrix, last accessed 18 August.

jobs. As jobs in the training set are randomly selected from the original dataset, it can be said to represent the original dataset to some extent.

Classifying jobs into software jobs and non-software jobs

In this project, the first 9% jobs in training set is used as test set. The classifier is trained by using the rest 91% jobs as training set. It will make prediction on the test set.

Evaluate classifier with confusion matrix

The training set is shuffled before the extraction of test set. The iteration of "shuffle – classification" is performed 100 times to get an averaged result. Such result is represented in confusion matrix. The time cost of each algorithm is also compared.

1.3 Approach

1.3.1 Define research-related software development jobs

As people may hold different opinions on whether a job is a software job or not, a specific definition of software jobs is needed. The definition was given after a discussion with supervisor.

1.3.2 Building a data set

The original dataset was the starting data. To give credit to this dataset, jobs from two universities were collected directly from their official vacancy websites. Jobs whose job id on universities' website is the same as reference id of a job in inherited data set is treated as duplicated jobs. Thus, duplicated jobs can be removed through the comparison between job id and reference id.

This training set mentioned before was too specific for our use – "software job" or "non-software job" which were less specific. To use such training set to classify jobs into software jobs and non-software jobs. The training set needed to be modified. Feature design

1.3.3 Data processing

The content of a job advert needs to be processed so that it can be described by features that are quantified. Keywords and phrases that indicate whether a job is a software job or not can be used as features to be able discriminate between jobs. If a job mentions words or phrases such as "Fortran", "Python" or "machine learning", this is very likely to be a software job. Such words and phrases are used as strong features as these words are discriminative in this context. If a job mentions "computer science" or "computing", it might be a software job yet it might not be one. These words and phrases are used as weak features. The list of features is generated from popular programming language and is modified afterwards. The detailed operations of feature list are described in section 2.3.5.

The number of occurrence of a feature in a job's title or description are multiplied by the weight to give the value of the feature of this job. All feature values of one job constitute a feature vector that can be used to describe the job based on the feature schema.

A feature vector of a job is the value for each feature in this job. For instance, there are 195 working features in the feature list. Hence, the feature vector of a job will be a 1 x 197 vector including the job id in the beginning and the sum value of all features in the end. For example:

AIY193,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,10,0,0,0,0,0,0,0,0.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,20,0,0,0,0,0,0,0,0,0,0,31.0

1.3.4 Classification

A classifier, a programme that can be used to classify jobs into software jobs and non-software jobs automatically, is implemented using three different machine learning algorithms. The classification is made on each job based on the mention of keywords.

1.3.5 Evaluation

The evaluation will assess the accuracy of classifier. Based on the result given by the evaluation, the classifier and the feature list can be modified to improve the accuracy.

1.3.6 Analysis of software jobs

The purpose of finding out software jobs is to obtain knowledge of potential market for the SSI. After data processing and analysis, this project intends to answer question such as:

- what the major subjects that recent software jobs research into are;
- where those software jobs located;
- what the most popular programming language are;

- and which programming language benefit salary most.

Diagrams and charts are used to help visualize these types of results.

2 Methodology

2.1 Generic methodology

Python

Python[9] is an objected oriented and interpreted programming language which is efficient for processing text. Python support various packages that can handle specific work such as natural language processing and data analysis; such as Natural Language Toolkit (NLTK)[10] and Pandas[11].

IPython Notebook

The IPython Notebooks[12], now known as Jupyter Notebook, is an interactive computational environment for Python. The IPython Notebook allows programmers to add descriptive context to code to improve the readability and make it easy to understand. It also allows programmers to do quick and small experiments and give feedback to the user.

Collections

Collections[13] in Python is a module that implements several kinds of container data type which allows python to access data faster than other built-in method. In this project when using key-value pair, a data type called defaultdict is used. Defaultdict will create a dictionary-like objects with default value. To be specific, accessing the weight of feature is implemented with defaultdict. The defaultdict creates one key value pair which key is null and value is zero. Each time, a feature and its weight are

[9] See: https://www.python.org/, last accessed 18 August.

[10] See: http://www.nltk.org/, last accessed 18 August.

[11] See: http://pandas.pydata.org/, last accessed 18 August.

[12] See: https://ipython.org/notebook.html, last accessed 18 August.

[13] See: https://docs.python.org/2/library/collections.html, last accessed 18 August.

read from file and they are used to create a dictionary-like object in the defaultdict. The key is the feature itself and the value is the weight of that feature. For instance, feature “software_engineer” has a weight of 10. In the defaultdict, an object of “software_engineer : 10” is created.

2.2 Building a data set

To give credit to the original dataset, two web scrapers are implemented to collect jobs directly from their official vacancy websites. The two universities are university of Bristol and university of Cambridge. The scrapers first access the target website and save all of html code of that page. Then they utilise a package named BeautifulSoup[14] to parse the downloaded html to extract information of the job. As html codes are organised in tags, BeautifulSoup4 is used to search for a specific tag and obtain the content or attribute of it. A reduced html code of a job on http://www.jobs.cam.ac.uk/job/ is shown to help explain:

```
<html>

<body>

<h2>Administrative Officer (Graduates)</h2>

<hr>

</header>

<section class="grid_11 prefix_2 suffix_1 alpha">

<p>The University has a community of around 8,500 graduate students.
This role supports the Head of Graduate Student Administration to
deliver a professional administration service to graduate students
and colleagues in Schools and Colleges.</p>

<p>You will lead on graduate student matters that include advising
School and College staff on regulations, policies, and procedures;
being the point of expertise for colleagues dealing with non-
standard student issues; ensuring implementation and maintenance of
clear and accurate procedures, and working with others to ensure
activities are appropriately resourced.  In addition, with oversight
of core operations, you will have responsibility for developing and
reviewing the business and so, working with others, drive forward
improvements in our business processes and service delivery.</p>
```

[14] See: https://www.crummy.com/software/BeautifulSoup, last accessed 18 August.

```
<p>The successful candidate will have a proven ability to work under
pressure with a high-volume and complex workload, be self-motivated
yet collaborative, influential, and able to foster positive working
relationships.  You will have the experience and ability to identify
and drive improvements.  You should be educated to degree level or
with equivalent experience.</p>

<p>Further information is available via the link below.</p>

<p>To apply online for this vacancy, please click on the 'Apply'
button below. This will route you to the University's Web
Recruitment System, where you will need to register an account (if
you have not already) and log in before completing the online
application form.</p>

<p>If you are unable to apply online, please contact the Academic
Secretary (email <a
href="mailto:acd.jobs@admin.cam.ac.uk">acd.jobs@admin.cam.ac.uk</a>)
.</p>

<p>Informal enquiries may be made to:

Kerri Gardiner (<a
href="mailto:kerri.gardiner@admin.cam.ac.uk">kerri.gardiner@admin.ca
m.ac.uk</a>) </p>

<p>Interviews will be held on Thursday 15 September 2016</p>

<p>Please quote reference AK09881 on your application and in any
correspondence about this vacancy.</p>

<p>The University values diversity and is committed to equality of
opportunity.</p>

<p>The University has a responsibility to ensure that all employees
are eligible to live and work in the UK.</p><h3>Further
information</h3>

<ul>

<li><a
href="/job/11185/file/FPs+A6+Graduate+Lead.pdf">Administrative
Officer (Graduates)</a></li></ul><a
href="http://hrsystems.admin.cam.ac.uk/recruit-ui/apply/AK09881"
class="campl-primary-cta">Apply online</a>

</section>

<aside id="sidebar" class="grid_4 suffix_2 omega">

<h6>Department/Location</h6>
```

```
<p><a href="/job/?unit=u00081">Academic Division</a></p>

<h6>Salary</h6>

<p>£27,328 - £32,600</p>

<h6>Reference</h6>

<p>AK09881</p>

<h6>Category</h6>

<p><a href="/job/?category=4">Academic-related</a></p>

<h6>Published</h6>

<p>16 August 2016</p>

<h6>Closing date</h6>

<p>5 September 2016</p>

</body>

</html>
```

From the piece of html code, we can find that the job description is stored in the content of tag <section> whose class is “grid_11 prefix_2 suffix_1 alpha”. Hence, we can use the find() function to locate the tag and retrieve information inside. Other information such as reference id and salary is stored in the content of tag <p> under the content of tag <aside>. It is similar to extract job description to obtain information there.

The output of a scraper is a CSV file that contains:

- Scraped_date – the date on which the job was collected;
- University_name – the mane of the university;
- Job_name – the title of the job;
- Reference_id – the unique id that can identify the job on the website;
- Division – the school or department in which the job is situated;

- Contract_type – whether this job is a permanent job or a contract of several years;
- Working_pattern – whether this job is a full time job or a part time job
- Minimum_salary – the minimum salary for this job;
- Maxmium_salary – the maximum salary for this job;
- Deadline – the application deadline;
- Job_description – the detailed description of this job.

Data collected from universities website is used to compare with original data set. To see whether the data from jobs.ac.uk is representative.

2.3 Data processing

As mentioned in section 1.1.2, job title and job descriptions are two major parts that is used in classification. The data processing in this project is to generate feature vectors for jobs from job title and job description. This processing has several major operations:

- Removing irrelevant word
- Tokenizing the content of job advert
- Generate document term matrix (DTM)[15]
- Extract feature vectors from DTM
- Several techniques are used in these operations.

2.3.1 Removing URL and email address

There are plenty of information in job title and job description that may not help us to discriminate software jobs. When processing data in future steps, removing irrelevant words can help to save memory usage and time cost. The first of data processing is to remove email address and URL.

[15] See: https://en.wikipedia.org/wiki/Document-term_matrix, last accessed 18 August.

When removing URLs and emails address in job content, regular expression[16] is used. The regular expression is to find a string of characters (also known as string) which match a particular pattern. For example, email address has a fixed format of “local-part@domain.name”. An email address[17] must contains a local part, an “@” punctuation and a domain name. Hence, the email address can be identified by using a regular expression. To be specific, regular expression of “[\w\.-]+@[\w\.-]+” is used to remove email strings.

“[\w\.-_]” represents a sequence of character that may contain letters, numbers, full dot or hyphen. Strings like “abc”, “a.b.c”, “a-b-c” and “a_b_c” can match this expression. In most cases, the local part and the domain name of email address can be covered by this expression.

“+@” means that the punctuation “@” is closely connected to local part and domain name.

Similarly, URL has its format of ABC.ABC.ABC. In this project expression of “[\w./:]+\.[\w.-]+\.[\w.-/?=&\w-]+” is used to match URL string.

In some cases, URL starts with its protocol[18], which can be “http://”, “https://” or “ftp://”. Regular expression of “[\w./:]” will match strings that contains numbers, letters, colon and slash.

Some URLs starts with domain name[19] such as “www.domain-name.com” and “jobs.ac.uk”. The expression can be represented by using “[\w.-]” that will identify strings that contains numbers, letters, dot and hyphen.

The last part of URL is the file name and parameters. For example, “index.html”, “main.php?id=123&uid=456” are used to identify the specific web page and parameters to be submitted. In this case, we use “[\w.-/?=&\w-]” to find these strings.

After email address and URLs are identified by regular expression, a python in-built function replace() is used to remove them.

[16] See: https://en.wikipedia.org/wiki/Regular_expression, last accessed 18 August.

[17] See: https://en.wikipedia.org/wiki/Email_address, last accessed 18 August.

[18] See: https://en.wikipedia.org/wiki/Application_layer, last accessed 18 August.

[19] See: https://en.wikipedia.org/wiki/Domain_name , last accessed 18 August.

2.3.2 Tokenizing job content

Tokenizing, in this project, is the operation of splitting sentences into single words. After all the punctuation is removed, words are separated by white space. Hence, white space is used to split the job title and the job description. The tokenized job title and job content is concatenated to constitute the job content.

As mentioned in the section 1.1.3, phrases can be treated as one word to be a feature. In this context phrases are referred to as “multi-word expression (MWE)”. The concept of MWE is defined in definition3.

Definition 3: MWE - “Fixed Expressions refer to specific combinations of two or more words that are typically used to express a specific concept.”[i]

In python, NLTK packages have a module named nltk.tokenize.mwe[20] which can processes tokenized text and merges multi-word expressions into single tokens. For instance, phrase “software engineering” is a combination of word “software” and word “engineering”. After tokenizing, two words become two tokens. Then those two tokens are merged into “software_engineer” such that it will be treated as one word in future processing.

2.3.3 Removing stop words

After the context is tokenized, irrelevant stop words is removed to further save memory. Stop words are some words that appear in most English context. For instance, “at”, “the”, “which” and etc. are stop words. In the job this project, python package NLTK is used to process the job description. NLTK is a package that can process human language data. In this project, we use NLTK to remove stop words. Words with particular tags are identified. The list of these tags is given below.

- CC - conjunction, coordinating such as “and”, “yet” and “for”;
- CD - numeral, cardinal such as “five”, “million” and “dozen”;
- DT – determiner such as “all”, “any” and “many”;
- FW – foreign word such as “je”, “ich” and “oui”

[20] See: http://www.nltk.org/_modules/nltk/tokenize/mwe.html, last accessed 18 August.

- IN – preposition or conjunction, subordinating such as “among”, “within” and “forward”

- MD – modal auxiliary such as “can” “may” and “must”

- PDT – pre-determiner such as “this”, “half” and “quite”

- PRP – pronoun, personal such as “she”, “himself” and “us”

- PRP$ – pronoun, possessive such as “her”, “his” and “mine”[The dollar is part of this expression]

- RBR – adverb, comparative such as “further”, “larger” and “more”

- RBS – adverb, superlative such as “best”, “biggest” and “most”

- RP – particle such as “about”, “at” and “off”

- WDT – “WH” - determiner such as “that”, “what” and “who”

- WRB – “WH” - adverb such as “how”, “however” and “whenever”

Words with tag that is included in the list above are removed.

2.3.4 Generate document term matrix

The purpose of tokenizing is to facilitate the generation of document term matrix (DTM). A DTM is a matrix that describes the frequency of words in multiple documents. Columns in DTM represent words that are mentioned at least in one document. Here, a document is the concatenation of job title and job description. Rows in DTM represent frequency of words that are used by these jobs. An example is given in Table 1.

Table 1: A schematic example of a document term matrix

	Word1	Word2	Word3	Word4	…
Document1	0	0	3	1	
Document2	1	2	0	4	
…					

The DTM in this project is generated by using a modified python package called textmining[21]. The modification made by use was keeping underline when generating DTM such that MWEs, mentioned in Section 2.2.2, can be preserved as one word.

2.3.5 Prune and weight document term matrix

After removing irrelevant words including email address, URL and stop words, there are still 69955 different words that is mentioned by all jobs. Hence, the DTM is very sparse. The DTM is pruned, columns that represent words in the keywords list are preserved – the others are pruned or removed. As features have different contributions when determining whether a job is a software job, features are assigned with different weights. Words such as "Python" and "Fortran" that are strong correlated with software jobs, are given a weight of 10. We assume that jobs that mention keywords "computer_science" and "informatics" are software jobs. They are given a weight of 5. For words such as "modelling" and "database", they can somewhat be an indicator of software jobs yet they are more relevant than other words. These words are given a weight of 1. There are some jobs that intend to recruit lecturers or chair or a department. These jobs cannot be software jobs. Hence, words such as "lecturer" and "chair" are given to a negative weight of -10 to represent that jobs mention these words cannot be software jobs. The weight operation is done by multiplying the occurrence of a keyword in a job with the weight factor. The pruned and weighted DTM becomes the matrix of feature vectors that can be used in classification.

[21] See: http://www.christianpeccei.com/textmining/, last accessed 18 August.

2.3.6 Parallelization

The whole DTM will be in the shape of 128k by 70k. In python, each integer takes up 3 bytes. The total consumption of memory of the whole DTM takes about 25GB. Commercial computer is not able to handle such data set. Also, when processing 1400 jobs, the elapsed time is around 22 seconds. Assuming the time will increase linearly as the number of jobs increases, to process all jobs it can take more than 2000 seconds. To improve processing efficiency, and make memory cost manageable, the data processing operation is parallelized by using the multiprocessing package[22]. The script of processing data is executed on Morar, which is a supercomputer with two 64-core nodes. Hence, 128347 jobs are distributed over 64 processes. Each process will process 2006 jobs except for the last process. The last process takes 1969 jobs. A partial DTM is generated and pruned and weighted according to feature list. By using the same feature list DTM generated by each process will share the same number of columns. Hence, the classifier can work on the output data of all processes.

2.4 Classification

The classifier is implemented using three different algorithms including: Decision Tree, SVM and Naive Bayes. In this project, a python package named sklearn[23] is used to implement the classifier. By passing different parameters to the classifier object, the classifier can be trained and make prediction based on different algorithms.

2.4.1 Decision Tree

Decision tree is a predictive model maps the observations about an item to conclusions about the item’s target value. Decision tree learning is often used in data analysis, machine learning and statistics. By answering questions, the outcome of decision tree is to make predictions on the value of a target variable. The figure 2 shows an example of how decision tree works.

[22] See: https://docs.python.org/2/library/multiprocessing.html, last accessed 18 August.

[23] See: http://scikit-learn.org/stable/, last accessed 18 August.

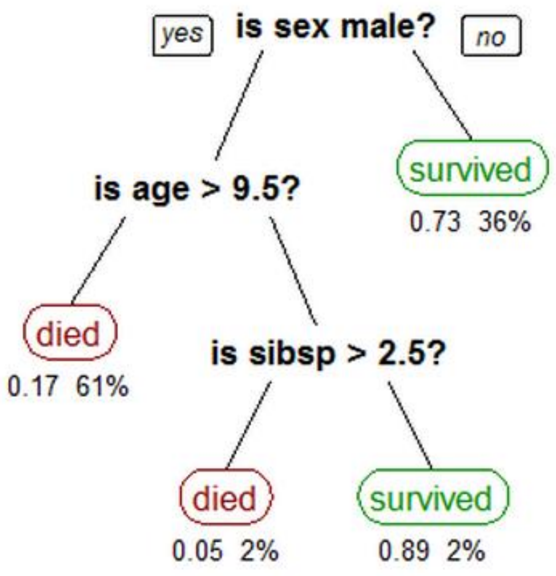


Figure 2: A decision tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard)[ii].

The end nodes constitute the leaves of the decision tree and the inner nodes make the branches. The rules, which the classifier learns, are given by features in the training set. There are 4 popular algorithms that can implement decision tree model, which goes Iterative Dichotomiser 3 (ID3), C5.0 and Classification and Regression Trees (CART). In this project, the functionality of decision tree is provided by the sklearn package which uses an optimised CART algorithm.

The advantages of using decision tree model include:[iii]

- the decision making process can be visualised which is easier to understand and interpret by a human being;
- data used by decision tree algorithm does not necessarily need to be normalised;
- efficient in processing data.

However, the disadvantages of decision tree model are obvious.

The decision tree model can be overfitting to the training set, which means the classifier can create over-complex trees that do not generalise the data well. Hence, pruning the decision tree can to some extent reduce the impact of overfitting. However, in the current version of sklearn, which is 0.17.1, the pruning of decision tree is not supported.

When data of a class dominates the training set, the decision tree classifier is more likely to classify data objects into that class. To improve the accuracy of decision tree model, the training set should be balanced before used to train classifier.

2.4.2 Support Vector Machine (SVM)

Support Vector Machine (SVM, also known as support-vector network) is supervised learning method that can be used for classification and regression analysis. One of the applications of SVM is to solve two-group classification problems. SVM implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space[iv]. The algorithm constructs a hyper plane, a subspace of one dimension less than feature space, to divide feature space into two parts which essentially becomes the boundary of two group. In SVM, the distance between the hyper plane and points in feature space, mapped from input vector, is defined as margin. Ideally, those points should be physically away from the hyper plane such that generalization error of the classification can be minimized. The goal of SVM is to identify a hyper plane whose margin is larger than any other hyper planes to act as the boundary of classification. Figure 3 demonstrates a 2 dimensional application of SVM.

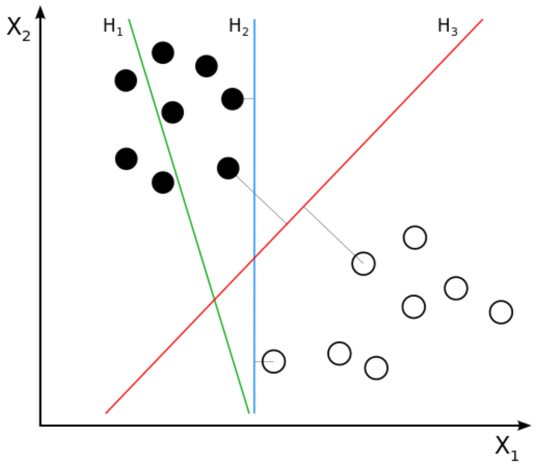


Figure 3: An example of SVM classification[v].

There are two type of data points in the figure 3, SVM is design to find a subspace, which in this case is a single line which separates these two types of data. Line H1 does not separate the two classes. H2 do separate the two classes yet the margin is smaller than H3. Hence, H3 is the used to act as the boundary.

The advantage of SVM algorithm is that:

- SVM is effective in high dimensional space;

- SVM is memory efficient due to the usage of a subset of training points in decision function;
- different kernel functions can be used and SVM support custom kernels.

The disadvantage of SVM include[vi]:

- The choice of kernel function can be difficult as different the kernel functions have huge impact on prediction accuracy. In this project, sklearn uses Radial Basis Function (RBF)[24] kernel by default.
- The speed of learning and making decisions is relatively low.

2.4.3 Naive Bayes

Naive Bayes classifier represents a family of probabilistic classifiers based on applying Bayes theorem assuming that features are independent from each other. For example, the colour and the size of fruits are independent. The Naive Bayes is a conditional probability model that calculates the posterior probability $P(c|x)$ from likelihood$P(x|c)$, class prior probability $P(c)$and predictor prior probability. The formula is shown in the formula given below.

$P(x|c)$ is the probability of event x knowing that event c happens; $P(c)$ is the probability of event c, $P(x)$ is the probability of event x, $P(c|x)$ is the probability of event c knowing that event x happen

$$P(c|x) = \frac{P(x|c) \times P(c)}{P(x)}$$

Assuming that features are independent, the calculation can be simplified into:

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times P(x_3|c) \times \ldots \times P(x_n|c) \times P(c)$$

In this project Multinomial Naive Bayes model is used. The Multinomial Naive Bayes model implements Naive Bayes algorithm for multinomially distributed data that is often used in text classification.

[24] See: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html, last accessed 18 August.

2.5 Evaluations

When evaluating the implemented classifier, several techniques are applied to find out how many correct predictions the classifier can make. The evaluation uses cross validation method and the accuracy of classifier is organized in confusion matrix.

2.5.1 Cross validation

Cross validation, also known as rotation estimation, is a kind of model validation technique to show how the prediction will generalize to the other dataset. In this project, a part of training set is used as test set. The actual classification of jobs in test set is known. The classifier is trained with jobs that are remained in the training set and then make predictions on the test set. The result of prediction is compared with the actual classification to check to what extent the classifier can make correct prediction. Before separating test set from training set, the jobs in the training set are shuffled. To make sure the result is reproducible, a random seed of 1 is given to the shuffling script. Hence, different algorithms will learn and classify the same jobs.

In this project, the first 1300 shuffled jobs are used as real training set and the rest 130 jobs are used as a test set for the trained classifier. Due to some feature is very rare in the training set, and they might be crucial to classify some jobs, the result of cross validation will be different when jobs, that can only be describe by those features, fall into test set or not. Hence, the iteration of "shuffle – validate" will be performed 100 times. The average result will be used to minimize the impact of these jobs.

2.5.2 Standard deviation

As mentioned in section 2.4.1, the accuracy of classifier may be sensitive to some particular jobs. Hence to show the variance range of averaged result, the calculation of standard deviation is:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2},$$

Where N is the number of measurement values in the population, x_i is the ith value for these Nth measurements, μ is the mean value of the population. When the value of standard deviation is large, it shows there is a lot of variation in these measurements.

2.5.3 Confusion matrix

A Confusion Matrix is a table which is often used to describe the accuracy of a classifier. As the true classification is known, the prediction made by the classifier is compared with the true result to see to what extent the classifier can make correct prediction. A sample confusion matrix is given in table 2.

Table 2: A sample confusion matrix

	Predicted negative	Predicted positive
Actual negative	True negative (TN)	False positive (FP)
Actual positive	False negative (FN)	True positive (TP)

In a typical confusion matrix, columns represent the predicted condition and rows represent actual condition. Four values:

- true negative – the number of non-software jobs are predicted as non-software jobs,
- false negative – the number of software jobs are predicted as non-software jobs,
- true positive – the number of software jobs are predicted as software jobs,
- false positive – the number of non-software jobs are predicted as software jobs,

are used to describe the accuracy of the classification. When true positive value and the true positive value are high, the classifier is making correct predictions in most of time. In this project, the classification of software job and non-software job is a start. As the software jobs might be further classified into research software jobs and non-research software jobs, the philosophy in this project is that the tolerance of the false positive predictions is larger than false negative predictions.

Some other values can be derived from confusion matrix to uncover the accuracy including:

- False positive rate (also known as fall-out) = $\frac{\sum Falsepositive}{\sum Actualnegative}$
- False negative rate (also known as miss rate) = $\frac{\sum Falsenegative}{\sum Actualpositive}$

- General accuracy (ACC) = $\frac{\sum Truepositive + \sum Truenegative}{\sum Totalpopulation}$

These values are used in the final result.

3 Experiments and evaluations result

3.1 Feature selection experiment

When choosing which part of job advert is used, we found that in some cases, advert information may not help us identify software jobs. For instance, from the job description of job "AOG309", we can hardly relate this job with writing scripts or codes.

```
AOG309

IT Infrastructure Support Specialist

As a member of the IT Unit you will support and maintain the College
ICT infrastructure and provide IT support to College staff and
students. You will be required to coordinate and work on IT projects,
collaborating with colleagues from across the College to ensure
successful delivery. Technically strong and confident in providing a
broad range of hardware and software support, possessing excellent
communication, time management and organisational skills, with a
'Can do' approach and determination to succeed and see problems
through to a solution will define the successful candidate. Key
qualities sought: Demonstrable experience of working in a 3rd line
technical support/infrastructure role. An evident 'positive' and
'can do' attitude to strengthen the colleges' culture and values of
integrity, respect, openness, fairness, high expectation and
enjoyment. Enjoys working on multiple tasks and prioritising to
achieve deadlines Excellent customer service and communication
skills Demonstrable experience of working on and coordinating
small/medium sized IT projects Strong organisational and team
working skills The closing date for receipt of completed
applications is Midday, Wednesday 3 August 2016. We are committed to
safeguarding and promote the welfare of all learners. We expect our
employees to share this commitment. Employment at the College is
subject to an Enhanced Disclosure and Barring Service - DBS,
(formerly CRB) check. Any persons applying for roles involving
'regulated activity' (as defined by the DBS), will also be subject
to a check on the child and/or adult barred list(s). "Inspiring
learners to succeed in life and in work.
```

However, this job has the subject tag of "IT". On the other hand, job "AOA926" is obviously a research software job but without any tags of "IT", "Computer Science", "Computing", "Software Engineering" or "Information Systems".

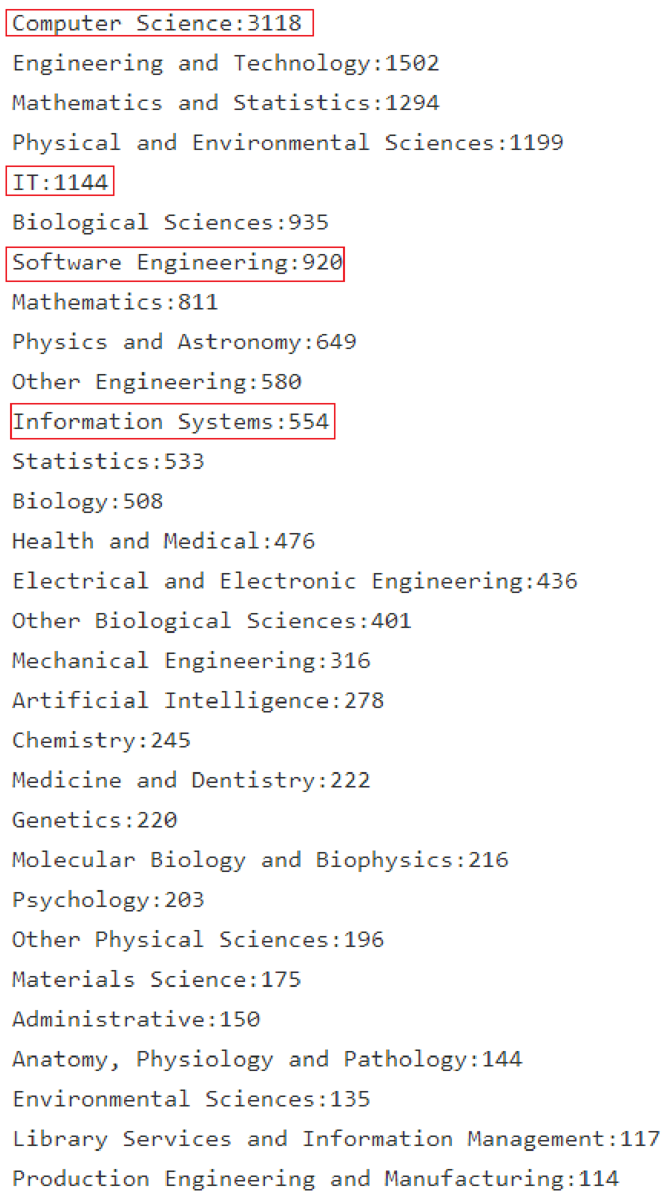
AOA926,

"PhD - The Application of Meshless Analysis Methods as an Alternative to FEA and CFD for Manufacturing Process Simulation

The School of Metallurgy and Materials is seeking a high calibre candidate for a fully funded EPSRC ICASE PhD studentship in materials modelling. The PhD is sponsored by Rolls-Royce plc and will be supervised through the Partnership for Research in Simulation of Manufacturing and Materials PRISM2 modelling group within the School of Metallurgy and Materials. PRISM2 is a research centre, at the University of Birmingham, with expertise in the modelling of materials, manufacturing and design for high technology applications in the aerospace and power generation sectors. The project will focus on **developing state-of-the-art modelling tools and capabilities to simulate the response of materials during manufacture.** Process modelling is a proven technology for improving the fundamental understanding of how components behave during manufacture, from material microstructure to bulk distortion & residual stress. Traditional numerical analysis techniques such as **Finite Element Analysis FEA)** and **Computational Fluid Dynamics (CFD)** can be applied to a wide range of manufacturing process simulation problems. However, the commercially available tools for implementing these techniques have some limitations including mesh distortion for very large strains, free surface fluid flows, granular flows, and solid body impact. The project will assess meshless analysis techniques such as Diffuse Element Methods (DEM) that offer an **alternative approach to the simulation of processes** such as complex geometry linear friction welds, powder flow in additive layer manufacture (ALM), or shot peening which are not currently practical with traditional tools. The student will require a good understanding of numerical methods and the ability to apply these to industrial processes. Contact details: Jeffery Brooks Tel: 0121 414 7836 Email: j.brooks@bham.ac.uk

This is a software job because that the when solving physics problems using computational fluid dynamics or finite element analysis, it cannot be done without the help of writing customized code to that problem.

Hence, we have the hypothesis that advert information should not be used as features. To justify this hypothesis, we took 5941 jobs that were being classified as software jobs. These jobs are being predicted as software jobs by three algorithms. More than 1% of job in the set was checked by hand and all of them are software jobs. Hence, we assume that all these jobs are software jobs.



```
Computer Science:3118
Engineering and Technology:1502
Mathematics and Statistics:1294
Physical and Environmental Sciences:1199
IT:1144
Biological Sciences:935
Software Engineering:920
Mathematics:811
Physics and Astronomy:649
Other Engineering:580
Information Systems:554
Statistics:533
Biology:508
Health and Medical:476
Electrical and Electronic Engineering:436
Other Biological Sciences:401
Mechanical Engineering:316
Artificial Intelligence:278
Chemistry:245
Medicine and Dentistry:222
Genetics:220
Molecular Biology and Biophysics:216
Psychology:203
Other Physical Sciences:196
Materials Science:175
Administrative:150
Anatomy, Physiology and Pathology:144
Environmental Sciences:135
Library Services and Information Management:117
Production Engineering and Manufacturing:114
```

Figure 4: Part of feature selection experiment result

The sum of major tags (computer science, IT, software engineering and information system) is less than the total number of “yes” jobs. Also, you should be aware of that more than one tags can be used in marking a single job. From this we conclude that many software jobs will be missed by using tags as features alone.

3.2 Classifier evaluation

The classifier is implemented with three different algorithms including Decision Tree, Support Vector Machine and Multinomial Naive Bayes. To figure out the most suitable algorithm for this project, the prediction result of classifier is evaluated through confusion matrix. Results shown in this section is based on cross validation which is the average value of 100 iterations. As mentioned in section 2.4.2, this project is tolerant with false positive jobs, which is because it intends to identify as much as software jobs as possible to. In order to achieve that goal, several experiment are carried out and the result is evaluated and compared to see if there is an improvement.

3.2.1 Result of default condition

The result of this section is used as a reference. The result of other experiment is compared with the result of this section. By default condition, several condition should be satisfied:

- Using original training set - The data set used in this section is the original data set which contains 205 software jobs out of 1427 jobs.

- Using default weight scheme – The weight of features can be 10, 5 or 1 as described in section 2.2.5.

- Using normal DTM as feature vectors – the value of a feature is the times that feature is mentioned by the document.

Table 3: Confusion Matrix of Decision Tree algorithm under default condition

Subject	Predicted negative	Predicted positive
Actual negative	108.99 +/- 3.86	2.17 +/- 1.47
Actual positive	9.02 +/- 2.58	9.82 +/- 2.54

Table 4: The Confusion Matrix of SVM algorithm under default condition

Subject	Predicted negative	Predicted positive
Actual negative	108.03 +/- 4.09	3.13 +/- 1.75
Actual positive	2.83+/- 1.61	16.26 +/- 3.48

Table 5: The Confusion Matrix of Naive Bayes algorithm under default condition

Subject	Predicted negative	Predicted positive
Actual negative	104.59 +/- 4.09	6.57 +/- 2.32
Actual positive	2.83 +/- 1.61	16.01 +/- 3.30

In this evaluation, 129 jobs are classified by the classifier. We can see that all three algorithms are able to correctly predict about 105 non-software jobs. From the three tables above we can figure out that Decision Tree algorithm has a general accuracy of 91.39%. SVM algorithm has an accuracy of 95.42%. 92.77% of decision made by Naive Bayes algorithm is correct. According to the true positive values in the three tables, we can find that SVM and Naive Bayes are almost equally good at finding out software jobs. The difference is that the result of Naive Bayes algorithm will pick up more false positive jobs which means it more incorrect predictions on non-software jobs. However, the Decision Tree algorithm is doing a less great job than the other two algorithms in terms of identifying software jobs. The reason might be that the training set is dominated by non-software jobs. Decision Tree algorithm has a disadvantage of inclining to classify objects into major class as mentioned in section 2.3.1. Hence, the next experiment is to perform classification on a balanced training set.

3.2.2 Result of using balanced training set

Under the default condition, Decision Tree algorithm has a relatively high miss rate. The hypothesis is that if we balance the training set, the general accuracy of Decision Tree will become higher.

Table 6: The Confusion Matrix from a Decision Tree algorithm using a balanced training set.

Subject	Predicted negative	Predicted positive
Actual negative	106.05 +/- 9.24	4.2 +/- 1.72
Actual positive	13.8 +/- 3.88	113.95 +/- 9.69

Table 7: The Confusion Matrix of SVM algorithm using balanced training set.

Subject	Predicted negative	Predicted positive
Actual negative	103.05 +/- 8.63	7.2 +/- 2.38
Actual positive	3.00+/- 1.87	124.75 +/- 8.28

Table 8: The Confusion Matrix of Naive Bayes algorithm using balanced training set.

Subject	Predicted negative	Predicted positive
Actual negative	27.6 +/- 5.47	82.65 +/- 8.10
Actual positive	10.15 +/- 3.57	117.60 +/- 7.87

In this evaluation process, 237 jobs are classified by the classifier. Based on the obtained result, the accuracy of Decision Tree algorithm increased to 92.43% which is improved by 1.04%. Still, SVM is the doing the best prediction whose accuracy reaches 95.75% that is not much different from default condition. However, the negative impact of putting software jobs into training set is huge on Naive Bayes. The reason might be that the classification is largely dependent on the portion of software jobs in training set. When the percentage of software jobs increases from 10% to 50%, the difference in prediction is quite large. Under these circumstances, Naive Bayes algorithm inclines to make too much positive predictions.

3.2.3 Comparison using amplified weight

The weight value of feature can be 10, 5 or 1. According to the mechanism of SVM, it intends to find out a border that with the largest margin. So the hypothesis would be if the weight is multiplied by a factor of 10, the classification result of SVM will be affected. Hence, the difference between this experiment and the default condition is the weight will be changed into 100, 50 and 10 respectively.

Table 9: The Confusion Matrix of Decision Tree algorithm using balanced training set and amplified weights.

Subject	Predicted negative	Predicted positive
Actual negative	105.64 +/- 7.49	3.88 +/- 1.88
Actual positive	14.38 +/- 3.78	114.10 +/- 7.69

Table 10: The Confusion Matrix of SVM algorithm using balanced training set and amplified weights.

Subject	Predicted negative	Predicted positive
Actual negative	100.75 +/- 7.21	8.77 +/- 2.87
Actual positive	1.58+/- 1.18	126.9 +/- 7.41

Table 11: The Confusion Matrix of Naive Bayes algorithm using balanced training set and amplified weights.

Subject	Predicted negative	Predicted positive
Actual negative	26.74 +/- 4.64	82.78 +/- 6.76
Actual positive	10.50 +/- 3.23	117.98 +/- 7.96

From the comparison of the three tables above, we can find that the amplified weight does only affect the result of SVM algorithm whose average general accuracy fall into 79.05%. However, the false negative rate also reduced from 2.36% into 1.23%. Assuming that

4 Extract information from job data

4.1 Research field

This analysis intends to investigate applications of software job with subject other than computer science. After removing tags that directly related to computer science including “computer science”, “IT”, “software engineering” and “information systems”, figure 5 plots 20 most used tags of software jobs. We say that this figure represents the application of computing technology in those fields. The more job opportunities of a subject indicates a wider application of computing technology is required by that subject.

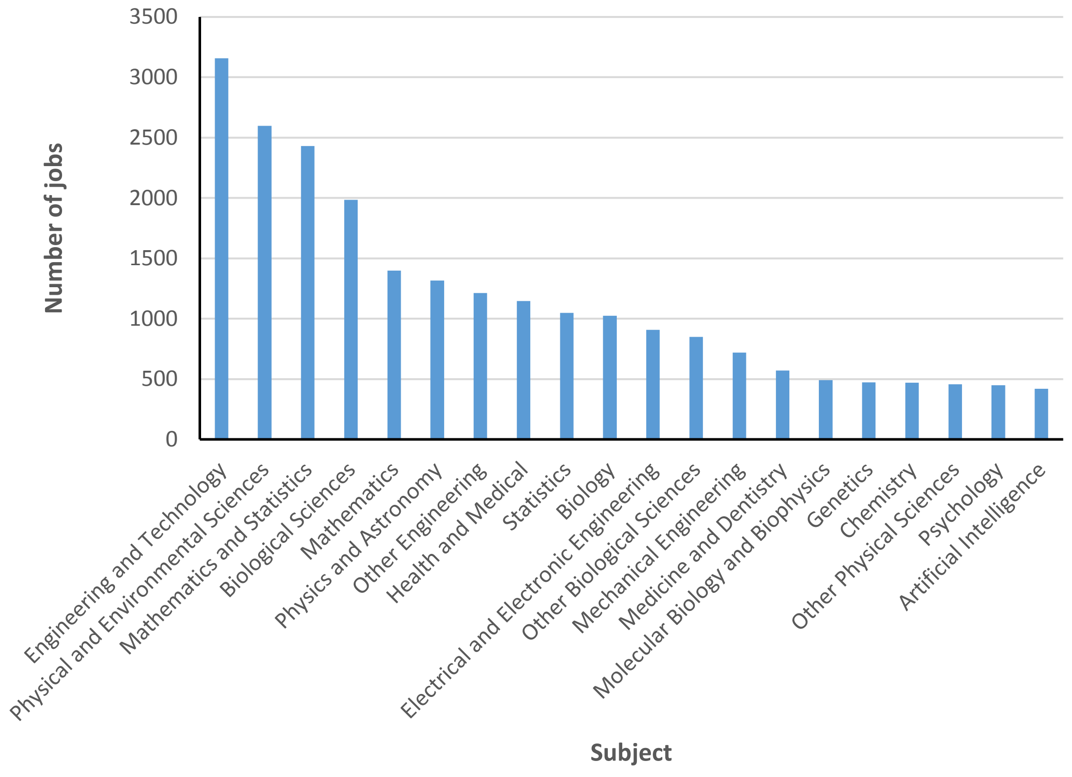


Figure 5: Popular subjects of software jobs except for computer science related.

From Figure 6, we can see that engineering, mathematics, physics and biology publish more software jobs than other subjects. What can be dig further is that whether these tags are assigned under a uniform and clear rule. For instance, we can identify “Mathematics and Statistics”, “Mathematics” and “Statistics” all appear in the list. The difference of jobs with these tags should be checked to see whether the separation makes sense.

4.2 Required programming skills

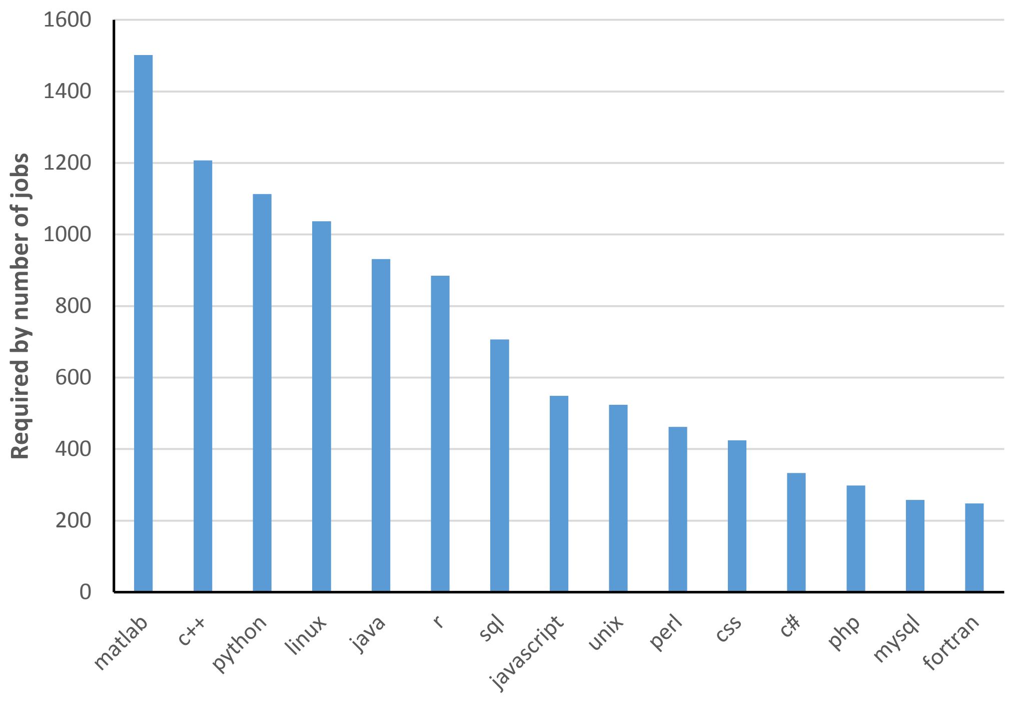


Figure 6: Popular programming languages that are required by software jobs.

From figure above, we can find that C++ is the most required programming language. Out of 11488 software jobs, there are more than 1500 jobs that explicitly require knowledge about MATLAB. About 1200 jobs require candidates to be able to programming in C++ and Python. In the meantime, shell command in Linux or Unix system is often a requirement.

4.3 Geographical distribution

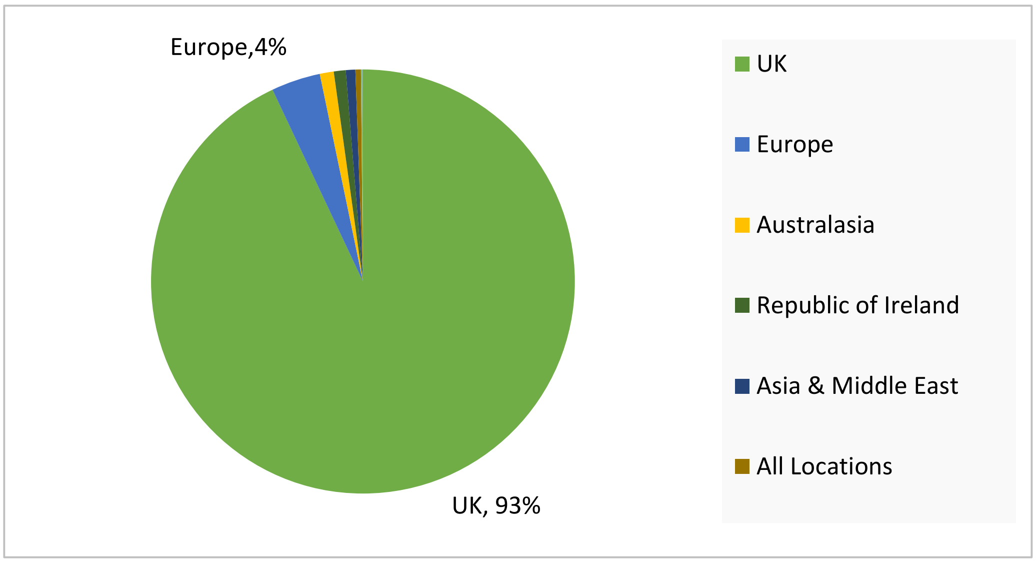


Figure 7: The world-wide geographical distribution of software jobs

According to figure 8, 93% software jobs that we found locates within in UK. This is because those who publish job advert on jobs.ac.uk are mainly UK institutes.

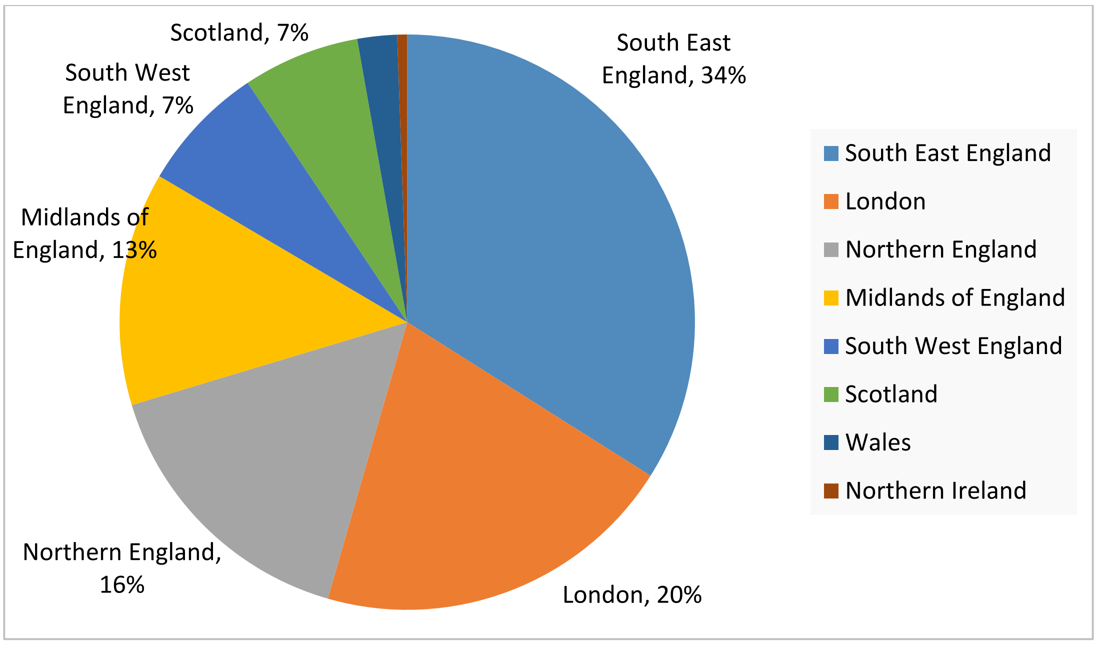


Figure 8: The nation-wide geographical distribution of software jobs

In terms of nation-wide, 34% of software jobs locates in south east of England except for London. London, as one single city, takes up 20 percent of software jobs which is higher than Northern England's 16% and Midlands of England's 13%. Scotland and South west of England share 7% of software jobs respectively. The remaining software jobs locates in Wales and Northern Ireland. The biggest market of software jobs locates in South east of England as well as London.

4.4 The relationship between salary and skills

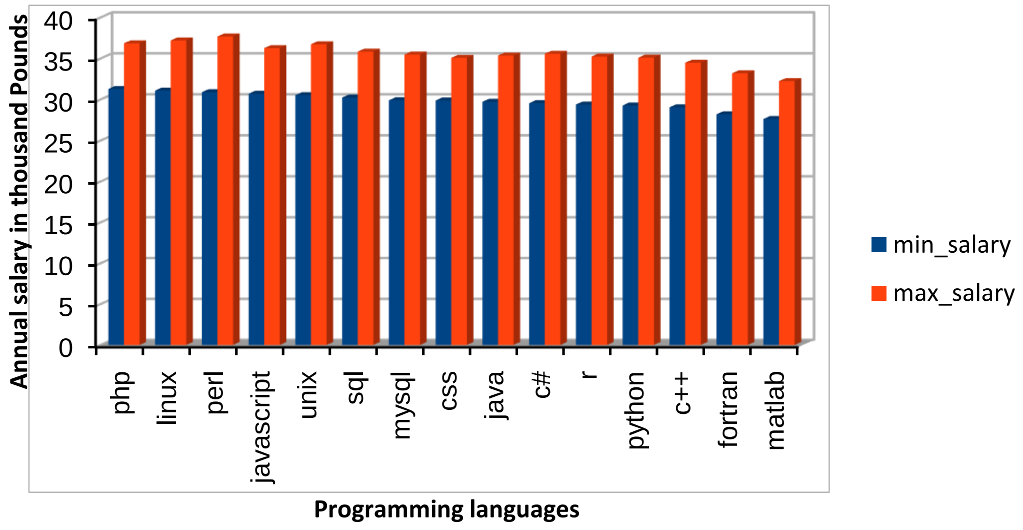


Figure 9: Top 15 minimum annual salary (£[25]) against programming languages.

Figure 10 shows the salary condition of 15 most popular programming languages which is the result shown in section 4.2. Candidates who can write PHP code may enjoys the highest minimum salary among 15 programming languages. Most successful candidates are able to make about 33k pounds a year. However, the highest average maximum salary can be found in Perl which is about 37k. Although C++ and MATLAB is often found in job requirement, developer who works in these two languages are likely to earn less than those who write Java code.

[25] Although most jobs salaries use Great Britain Pound, there are exceptions, e.g. South African Rand, Australian Dollars, etc.

5 Conclusion

5.2 Final model

From all experiment above, we can say that SVM is a more suitable algorithm for this project than Decision Tree and Naive Bayes. SVM has the best overall accuracy when is:

- the training set is not balanced,
- the weight of features is not amplified,
- the features are expressed by normal DTM.

Under this condition, the average general accuracy of SVM is able to reach 95.75%. Even under the worst condition, the miss rate would be 4.01%.

On the other hand, if we need to find out as many software jobs as possible regardless of false positive jobs, we can use amplified weighting of SVM. In such case, only about 1.23% software jobs are missed.

Which condition to apply is largely depends on the purpose of classification. If we want to investigate the characteristics of software jobs, we might need the default weight such that the false positive jobs will not affect much. If we need to further classify jobs into research software jobs and non-research software jobs, we can use amplified weight to uncover most software jobs.

5.3 Software jobs

According to the Section 3.2, the Naive Bayes algorithm makes is suitable under default condition, SVM algorithm and Decision Tree algorithm should be working with balanced training set to reach the best accuracy. We found that each algorithm can make false positive predictions that other algorithm may not. Assuming that false positive decision made by each algorithm is somewhat different, jobs that are classified as software jobs by three algorithms is very likely to be software jobs.

When applying the classifier on the whole dataset with the most suitable condition respectively, Decision Tree algorithm predicts 13276 jobs as software jobs. SVM algorithm classifies 18087 jobs as software jobs. The Naive Bayes algorithm picks up 17594 jobs as software jobs. Figure 11 shows a Venn diagram that is used to see to what extent these algorithms agree with each other.

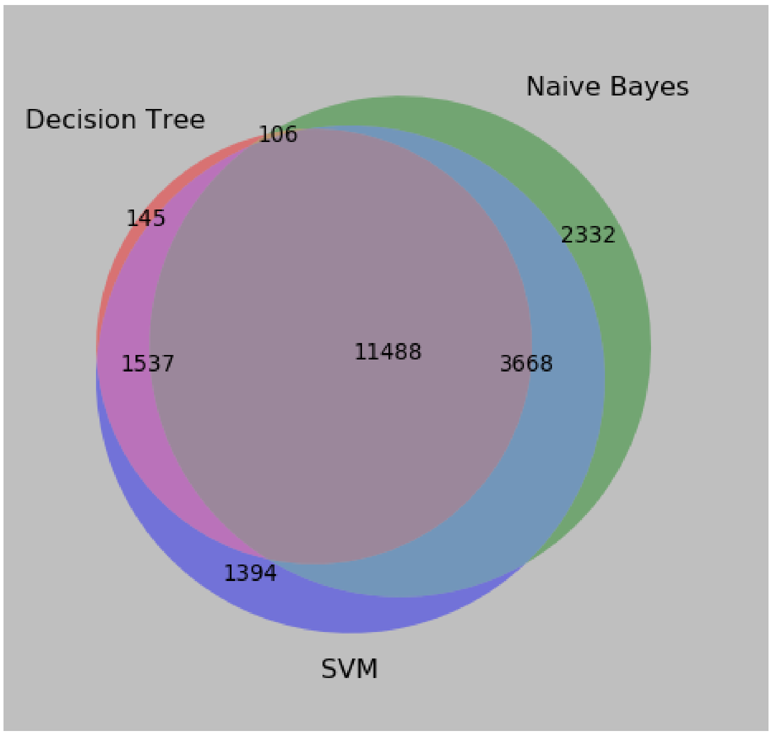


Figure 11 - Venn diagram of classification result of three algorithms.

From the Venn diagram, 11488 jobs are classified as software jobs by three algorithms. We did an inspection of 60 jobs within that set, all of them are real software jobs. Hence, this set can be somewhat treated as a set of software jobs. According to our estimation in section 1.1.3, there are about 12035 software jobs in the original dataset. 95% of software jobs are identified.

5.4 Characteristic of software jobs

Engineering, physical and environmental science and mathematics and statistics are three major subjects that require software development knowledge. More than a half of software jobs within UK locates in South east of England including London. Although, MATLAB is the most required programming language, focusing on it may not earn you more money than focusing on Perl and PHP.

5.5 Future work

This project could do the following future work:

- Separating software jobs into research software jobs and non-research software jobs - the difference here being whether the codes and scripts written are used to do scientific research or not.

- Inspecting why different predictions are being made by the different classification algorithms.

- Uncovering more features to improve the classification.

6 Appendix

6.2 Features and their weight

This is the complete feature list and their weight according to their contribution to software jobs. There are 209 features in the list, 14 of them are commented out. The use of this feature list is described in Section 2.3.5. Features in this list are case insensitive. Upper case letters will be changed into lower case when loading the weight scheme.

Keyword	Weight
programmer	10
java	10
machine_learning	10
computer_science	5
computer_sciences	5
visual_studio	10
c++	10
c#	10
jquery	10
hadoop	10
sas	5
spss	5
spark	10
python	10
fortran	10
dreamweaver	10
php	10
matlab	10
r	10
javascript	10
sql	10
mysql	10
perl	10
ruby	10
batch	10
linux	10
unix	10

#database	1
programming	10
modelling	1
simulation	1
#software	1
#development	1
developer	1
#develop	1
software_development	10
software_developer	10
software_developers	10
develop_software	10
#computing	5
computational	5
#analysis	1
computer_programming	10
computer_programs	10
computer_programmes	10
#research	1
software_engineer	10
software_engineering	10
software_engineers	10
wordpress	10
scripting	10
css	10
programming_skill	10
programming_skills	10
r_packages	10
scala	10
nosql	10
docker	10
postgresql	10
github	10
fem	10
finite_element	10
finite_element_method	10
finite_element_methods	10
image_processing_software	10
fe_model	10
fe_modelling	10
fe_models	10
fe_software	10

informatics_pipelines	10
bioinformatics_pipeline	10
bioinformatics_pipelines	10
opengl	10
computer_graphics	10
computer_code	10
geometric_modelling	10
computer_animation	10
apache	10
big_data	10
ontology	5
knowledge_discovery	10
autonomic_computing	10
ai	10
computational_skills	10
natural_language_processing	10
nlp	10
quantum_information	10
image_processing	10
ajax	10
jsp	10
xml	10
web_service	10
web_services	10
web_analytics	10
#data_manager	10
test_scripts	10
raspberrypi	10
raspberry_pi	10
arduino	10
android	10
load_balancing	10
centos	10
dns	10
high_performance_computing	10
computer_modelling	10
excel	5
nvivo	5
minitab	5
mlwin	5
stata	5
bioinformatics	10

computational_biologist	10
computational_biology	10
computational_chemistry	10
digital_signal_processing	10
stochastic_modelling	10
gillespie_algorithms	10
agent_based_modelling	10
computational_physics	10
numerical_weather_prediction	10
microsoft_windows_server	10
microsoft_sql_server	10
computational_fluid_dynamics	10
human_computer_interaction	10
numerical_modelling	10
structural_dynamics	10
scientific_programming	10
informatics	5
numerical_simulations	10
numerical_simulation	10
mathematical_biology	10
programming_experience	10
neuroinformatics	10
brain_simulation	10
many_core	10
many_cores	10
gpgpu	10
gpgpus	10
gpu	10
gpus	10
power_electronics_system_simulation	10
power_electronic_component_modelling	10
power_semiconductor_simulation	10
technical_programming	10
computational_models	10
computational_modelling	10
applied_probability	10
biophysical_modelling	10
bioinformatician	10
monte_carlo	10
numerical_analysis	10
scientific_computing	10
distinct_element_method	10

quantitative_methods	10
quantitative_methodology	10
quantitative_methodologies	10
quantitative_analysis	10
post_doctoral_statistician	10
using_r	10
quantum_spin	10
numerical_techniques	10
#postdoctoral	1
#post_doctoral	1
#post_doc	1
mathematical_modelling	5
#research_fellow	1
inverse_problems	10
inverse_problem	10
systems_modelling	10
development_of_phone_based_applications	10
biostatistics	10
causal_analysis_methods	10
mendelian_randomization	10
computational_medicine	10
computer_simulation_models	10
software_programming	10
fokker_planck	10
langevin_equations	10
supercomputer	10
solid_state_simulations	10
text_processing	10
probabilistic_models	10
deep_learning	10
probabilistic_modelling	10
lecturer	-10
lecturership	-10
chair	-10
consultant	-10
#project_management	-10
ubiquitous_computing	10
pervasive_computing	10
build_software	10
building_software	10
cfd_modelling	10
openfoam	10

applications_developer	10
application_developers	10
computer_simulation	10
high_performance_computers	10
high_performance_computer	10
computation_fluid_dynamics	10
computational_astrophysics	10
numerical_code	10
cfd_models	10
cfd_model	10
parallel_processing	10
computer_vision	10
numerical_tool	10
mathematica	10

7 Reference

[i] Prenger, S. A. (2003). Fixed expressions and the production of idioms. PhD Thesis, Radboud University Nijmegen, Nijmegen.

[ii] https://en.wikipedia.org/wiki/Decision_tree_learning, last accessed 18th August

[iii] Sklearn documentation. http://scikit-learn.org/stable/modules/tree.html, last accessed 18th August.

[iv] Vapnik, V. (1995). "Support-vector networks". 20 (3): 273–297. :

v Based on PNG version by User: Cyc from: https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes.png, last accessed 18th August.

[vi] Disadvantages of Support Vector Machines. http://www.svms.org/disadvantages.html, last accessed 18th August.