Enda McKenna CS3012 16321176

MEASURING SOFTWARE ENGINEERING

Introduction

Before we look at measuring the performance of a software engineer, it is important to understand exactly what software engineering is. 'Software engineering is the process of analysing user needs and designing, constructing, and testing end user applications that will satisfy these needs through the use of software programming languages. It is the application of engineering principles to software development. In contrast to simple programming, software engineering is used for larger and more complex software systems, which are used as critical systems for businesses and organizations' (Techopedia). Software Engineering was initially known as Software Crisis, but in the ever - evolving world of technology, software engineering became a primary component in a large number of things we are involved in, each and every day.

In alignment with this progression, data analytics and data science has also been progressing at a rapid pace. Nowadays, everything and everybody around us is analysed, and this analysis is leading to better interpretations of businesses, people, and fellow employees. Nobody is safe from the data – filled world we live in. 'Torture the data and it will confess to anything' (Coase, 2016). Every action we take is scrutinized, and often has facts and figures to back this up. As a result of this, people and employees are being analysed, and sometimes over – analysed. A main culprit of this would be a software engineer, as the majority of their work is measurable (code coverage, reusability etc.). Throughout this report, we will be looking at how the software engineering process can be measured under the following criteria : measurable data, computational platforms, algorithmic approaches and ethical issues.

Measurable Data

When gathering data, people can regularly fall into the mindset of collecting as much data as possible, as they believe this will result in the creation of as much interpretations as possible. However, it is important to ensure that the data that you are attempting to collect will serve a purpose. 'If you do not know how to ask the right question, you discover nothing' (Deming, 1990). Therefore, it is important to set out your goals, and what you want to learn, before attempting to obtain any data. There is a 5 step process that is used to determine what data you require, and what to do with this data :

1. Define Your Questions
2. Set Clear Measurement Priorities
    a. Decide what to measure
    b. Decide how to measure it
3. Collect Data
4. Analyse Data
5. Interpret Results (Dillard, 2017)

These steps are designed to keep analysts aligned with the task they have been assigned, and to prevent irrelevant analysis, as this can be time – consuming, a waste of money, and can often have a negative impact on the overall conclusion.

In terms of software engineering, the relevant data to analyse an engineer's performance can be split into two sub – categories :

- Code Data
- Non – Code Data

Code Data

This is data we can obtain from the programmer themselves, or the work they have submitted. Code data is generally a lot easier to use to measure performance than non – code data. Examples of code data include:

- Number of commits
- Frequency of commits
- Bug fixing
- Different projects involved in

Non – Code Data

This is data that can be a bit more difficult to measure. It is data that may not be possible to obtain from the programmer – with or without their assistance. However, this does not make non – code data less important, and it is a crucial piece of information in measuring the performance of a software engineer. Examples of non – code data include:

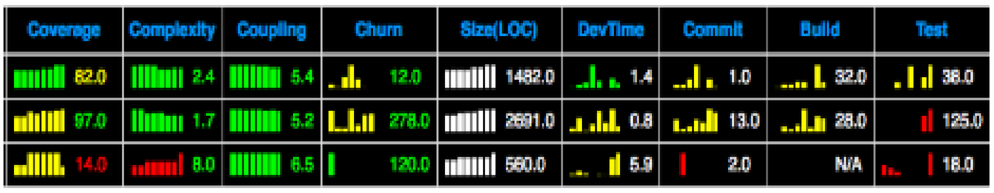
- Meetings Attended
- Emails
- Communications
- Company

As code data is easier to measure than non – code data, there are a number of different data sources which assist us in evaluating the performance of a software engineer, through automation of data collection. A process called PSP (Personal Software Process) gives developers a guideline on how to track their work, resulting in large amounts of data where automated tools came into play, such as Hackystat and Leap.

'The Personal Software Process (PSP) is a structured software development process that is intended to help software engineers better understand and improve their performance by tracking their predicted and actual development of code' (Wikipedia). This is important for software engineers when they are running tests, defining processes, and performing various other tasks. The PSP process allows developers to track their actual progress against their predicted progress, and to try and identify any areas of inefficiency. The PSP process was a huge benefit in assisting with the evaluation of the performance of a developer, however, the manual nature of PSP resulted in data quality problems, and therefore more automated entered the market and began to grow.

One of these was the Leap toolkit. Leap toolkit attempted to eradicate these data quality problems through the use of automation. Although data was still manually inputted, however, the analysis provided by Leap was greater than that of PSP. While maintain the majority of the features present in PSP, Leap also introduced a variety of complex regression models, to enhance the overall analysis. However, through the advancements in automation, certain features in Leap became very difficult to analyse.

On the other end of the spectrum was Hackystat. Hackystat is a data collection tool developed at the University of Hawaii. Hackystat takes the data gathered by PSP, and enables developers to automatically analyse this data. Hackystat contained additional information on top of the data it gathered from PSP, such as code coverage and complexity. This made Hackystat a popular choice for developers. 'Hackystat enjoys the support of many industrial and government sponsors, including Google, the National Science Foundation, SUN Microsystems, NASA, Expedia, and IBM, as well as a large number of academic institutions' (Johnson, 2009). The image below shows a Software Intensive Care Unit (ICU) display based on Hackystat (Hawaii).



Coverage	Complexity	Coupling	Churn	Size(LOC)	DevTime	Commit	Build	Test
62.0	2.4	5.4	12.0	1482.0	1.4	1.0	32.0	38.0
97.0	1.7	5.2	278.0	2691.0	0.8	13.0	28.0	125.0
14.0	8.0	6.5	120.0	560.0	5.9	2.0	N/A	18.0

These two analysis toolkits are excellent resources to extract data from developers, and to analyse **code** data. This data can be obtained from the developers themselves, or through a computer program which extracts the data. The latter is a much more accurate and efficient way of completing this, however, going back to an earlier point, choosing the relevant data is crucial here, as do you do not want too much data, and you must also protect the privacy of the developer, which we will go into further detail on later.

Non- code data is a lot more difficult to measure and analyse, but just as important. A software engineer who spends all their time committing code, but does not attend any meeting, or does not reply to any emails is not a good software engineer. Specific research, such as surveys, has been conducted on similar data, as more and more businesses are beginning to realise the importance of these skills in any profession. It has been found that 96% of executives cite lack of collaboration or ineffective communication as a main fault for workplace failures (VITRU). This shows that a software engineer could be excellent at coding and committing great pieces of work, but if they don't continue to develop softer skills, such as teamwork and communication, they are not excelling in their role in the workplace. It has also been found that the average time it takes to communicate with a colleague at work is 74 minutes (VITRU). This fact can be useful in turning non – code data into a measurable statistic. Firms can encourage their employees to better, or at least match this timeframe. This would also help to eradicate the stat mentioned earlier on workplace failures, as collaboration and communication would improve as a whole in the workplace. These figures show the importance of non – code data, and put into perspective that a good software engineer is most certainly not just an employee who is strong at coding.

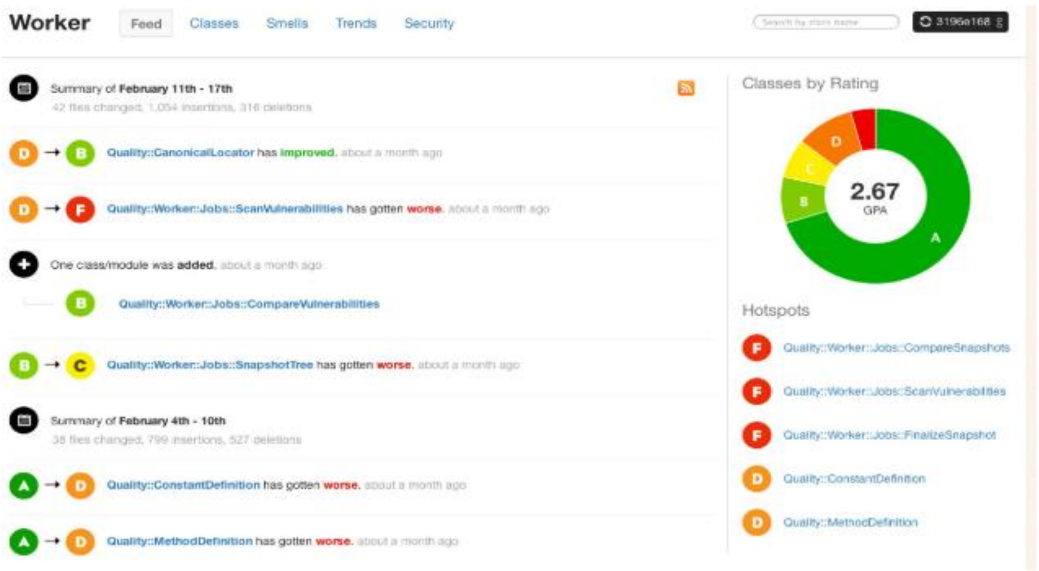
To conclude this section, there is a large volume of code data which can be collected, but it is important to know the relevance of this data, and different ways of collecting it. The non – code data also serves an equal importance in measuring a software engineers performance, however may not always be taken into account due to the fact it is difficult to measure.

Computational Platforms

Once the data has been collected, it is now time to determine where and how to compute and analyse it. This is where the profession of data analytics could come into play. Some firms may have an internal team, other firms may require external assistance. There are plenty of firms around which specialise is data analysis, however, this figure drops significantly when looking into companies who specialise in analysis of software engineers. The majority of this analysis is also computed on the *code data*, as non – code data would not be feasible to analyse in the majority of situations. Having this statistical information available makes it easier for developers to concentrate solely on their jobs. We are going to look into a number of Analytical companies in this section, and the services in which they provide.

Code Climate

'Code Climate's engineering process insights and automated code review for GitHub and GitHub Enterprise help you ship better software, faster (Code Climate).' Immediately from the company's slogan themselves, we can see that they are of course interested in analysing *code* data. Code Climate has become a popular choice for some large businesses, including Intercom and Salesforce. This is because of their approach to analysing some of the most applicable statistics of a software engineer in their day to day work. One of Code Climate's unique selling points is that it can analyse data from a wide range of programming languages, such as Java, Python, Haskell and many more. As mentioned in the slogan, Code Climate works alongside GitHub, and analyses factors such as code coverage and progress reports. One interesting thing with Code Coverage is the detail it goes into with Technical Debt. Technical Debt is the 'implied cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer' (Wikipedia). This can be a difficult concept to analyse, however Code Climate provide a detailed break - down in their analysis reports. It assesses this on the basis of amendments to files which do not improve the code, and also certain maintainability issues. The following is a summary of a report from Code Climate, showing the ease at which the analysis can be read and interpreted.



Summary of February 11th - 17th
42 files changed, 1,054 insertions, 316 deletions

D → B Quality::CanonicalLocator has improved. about a month ago

D → F Quality::Worker::Jobs::ScanVulnerabilities has gotten worse. about a month ago

One class/module was added. about a month ago

B Quality::Worker::Jobs::CompareVulnerabilities

B → C Quality::Worker::Jobs::SnapshotTree has gotten worse. about a month ago

Summary of February 4th - 10th
38 files changed, 799 insertions, 527 deletions

A → D Quality::ConstantDefinition has gotten worse. about a month ago

A → D Quality::MethodDefinition has gotten worse. about a month ago

F Quality::Worker::Jobs::CompareSnapshots
F Quality::Worker::Jobs::ScanVulnerabilities
F Quality::Worker::Jobs::FinalizeSnapshot
D Quality::ConstantDefinition
D Quality::MethodDefinition

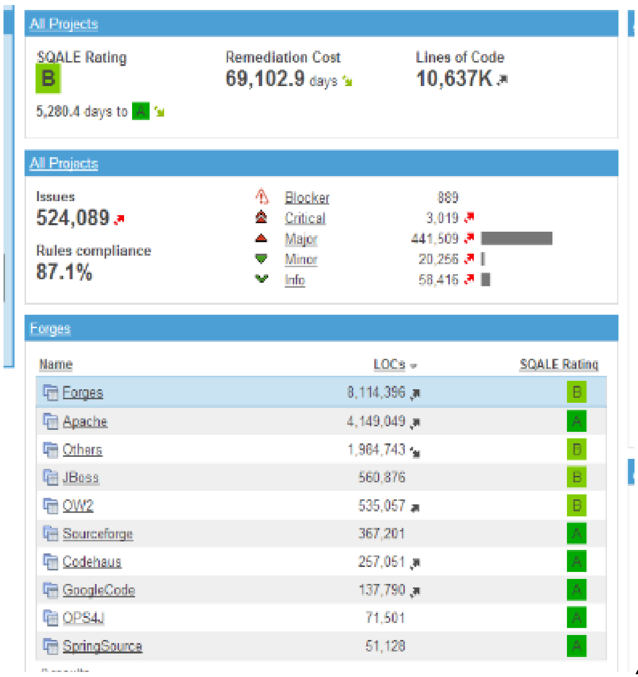
(Demeyere)

Codebeat & Codacy

Codebeat and Codacy are two companies who provide a very similar service to Code Climate. Both Codebeat and Codacy are involved in the reduction of technical debt, and then production of new, clean code. These companies both charge a cheaper price than Code Climate.

The companies mentioned above are firms who use Automated Code Review Tools. However, there are other companies who are highly involved in the analysis of software engineers, but do not use these tools. One example is an Asian company called XIAO, who were developed by Microsoft Research Asia, The prime concept behind their analysis is Code Cloning. Code Cloning is the 'process of creating an exact copy of another application program or object' (Techopedia). Code cloning allows developers to copy a piece of code, and to use it in their own project, saving the developer a lot of time. XIAO determined that Code Cloning can lead to duplication of bugs, and inconsistent bug fixing, and therefore realised that analysis on this would assist developers hugely. Inconsistent code cloning also resulted in a number of issues. XIAO then created requirements on the basis of these errors through cloning. Some of these requirements included the ability to identify near – miss code clones, scalability, and ease of deployment. This gives a larger picture of the different analysis that can be conducted on an employee's code, and much, much more.

Analysis is not a free service. The companies above charge a fee for their analysis, as this information alone could help improve the productivity of individual developers. However, some companies do provide a free version of their analytical tools. SonarQube is one of these, which is free for open source projects. SonarQube has downloadable or online code tracking options, and supports over 20 programming languages. SonarQube has a feature called a Quality Gate, which analyses new commitments of code, and determines whether or not a new project can go into production or not. SonarQube also analyses new pull requests immediately, which ensures you don't add to code that already contains issues. These are just a couple of the features provided in this software.



All Projects

SQALE Rating B — 5,280.4 days to A

Remediation Cost 69,102.9 days

Lines of Code 10,637K

All Projects

Issues 524,089

Rules compliance 87.1%

Blocker	889
Critical	3,019
Major	441,509
Minor	20,256
Info	58,416

Forges

Name	LOCs	SQALE Rating
Forges	8,114,396	B
Apache	4,149,049	A
Others	1,984,743	B
JBoss	560,876	B
OW2	535,057	B
Sourceforge	367,201	A
Codehaus	257,051	A
GoogleCode	137,790	A
OPS4J	71,501	A
SpringSource	51,128	A

Analysis Report from SonarQube (Wikipedia)

Overall, with the developments of technology and analytical tools, I think it is hugely important that companies conduct a thorough analysis on a developers performance, to try identify any areas of weakness, and to overcome these as quickly as possible. This, in hand, will improve the overall efficiency of the developers work. There are a large number of different Data Analytics companies around, some requiring subscription fees, and some which do not. In determining whether to opt for a cheaper alternative, my opinion is that you cannot put a price on a developers performance and ability to produce error – free code, so you should pay an added fee if it results in a more in – depth analysis.

Algorithmic Approaches

In relation to the data analytics companies mentioned above, we will now explore the different algorithmic approaches they use to produce their final analytical report. The main technique used by these companies is Computational Intelligence (CI). Computation Intelligence refers to the ability of a computer to learn a specific task from data or experimental observation (Wikipedia). Computational Intelligence has become increasingly useful in scenarios where mathematical approaches are not adequate to solve a complex problem. This could be due to the unpredictable nature of a problem for example. There are 5 main Principles in relation to Computational Intelligence, which we are going to explore now.

Fuzzy Logic – This deals with modelling of complex, real world problems. Fuzzy Logic can handle incompleteness, and can also ignore data in a process model. Fuzzy Logic is useful for approximate reasoning, but does not have learning abilities, unlike humans.

Neural Networks – Neural Networks consists of 3 main components – one which processes the information, one which sends the signal, and one which controls this signal being sent. Fault tolerance is one of the main assets of Neural Networks.

Evolutionary Computation – This is based on the theory of Natural Selection where only the strongest survive. This in essence ensures optimisation when attempting to solve a problem.

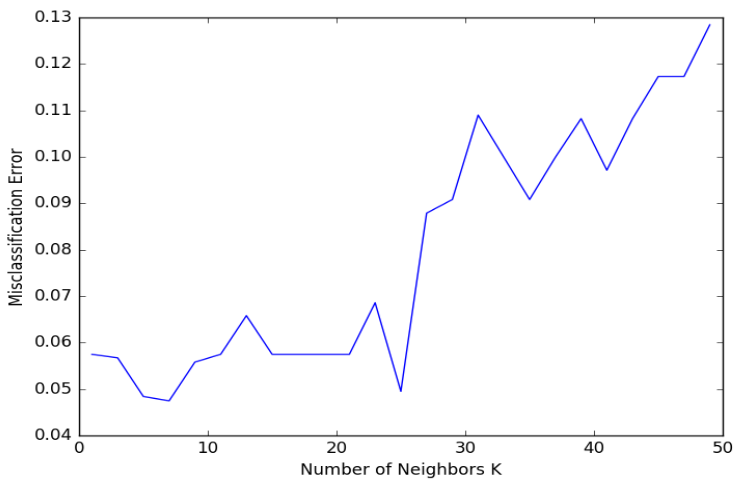
Learning Theory – Learning Theory tries to emulate as much as possible the emotions and reasonings of humans. It takes into account emotional and environmental effects when making predictions.

Probabilistic Methods – This attempts to bring out the possible solutions to a reasoning problem, based on prior knowledge.

These principles act together to optimise CI. With Computational Intelligence consistently evolving and improving, it will play a huge role in problem – solving with incomplete evidence or knowledge. Therefore, it is important to 'teach' a computer the skills it needs to do this, and allow the computer to learn. There are two methods for conducting this learning – supervised and unsupervised learning.

Supervised Learning

Supervised Learning is a type of machine learning dataset that uses a training dataset to make predictions. Included in this dataset are input and response variables. The idea is that, over time with repetitive trial and error, the computer will be thoroughly approximated to reality, that the appropriate response variable will be predicted. The larger the number of trial and errors conducted, the more accurate the model will be. An example of supervised learning is the K – Nearest neighbours algorithm. In this, the data must first be classified. Once classified, the computer takes part of the data set, and uses it as a test set. The data and variables in the set are then analysed. The supervised learning aspect comes into play here, as you have input in what each point can be. In K – Nearest Neighbours, the computer takes the distance of each data point to the known origin, and compares it with the data points distance to an unknown origin. To classify the point of unknown origin, the model finds the classification of the nearest K points, and the data point is assigned to the group with the highest number of points in the group of K (Houlding, 2018). The process is repeated, altering K to minimise the misclassification rate. The following is a misclassification error image, to a K – Nearest Neighbours problem. We can see in this example, K is minimised roughly at a value of 7.



(Zakka)

In terms of this assignment as a whole, a software engineers performance can be measured through their code coverage when supervising the computer during the learning process, as errors could potentially occur, as well as being analysed by the algorithm itself.

Unsupervised Learning

Unsupervised learning is when there are no historical output variables to coincide with inputs. The models rely solely on the underlying structure to learn more about the relevant data. The data is analysed by an algorithm to discover more about this data, however this is completed without the help of any human interaction. An example of an unsupervised algorithmic model is K Means Clustering. This involves dividing the data into a particular number (K) of categories, in which the data in each category is similar. This can be a very useful technique when wanting to analyse different data, for example the performance of a specific team. The unsupervised nature of this model has its advantages and disadvantages. One advantage it possesses is the fact that human bias cannot come into the equation at all. However, one disadvantage unsupervised learning leads to is the interpretation of a particular analysis. Data could be broken up into different clusters based on a developers behaviour, however it could be difficult for a computer to determine whether this behaviour is necessarily a good or bad thing. Interpretation of results is one of the areas that unsupervised learning falls behind on, and one that is constantly trying to be improved on by a number of organisations.

To summarise this topic, algorithms can provide a very useful analysis into measure a software engineers performance. This is even more beneficial through the increased automation, and increasing popularity in unsupervised learning. However, businesses that provide this analysis must understand that a computer cannot interpret correctly every piece of information it receives, as it may never have encountered anything similar beforehand. Therefore, it is important to ensure that supervised learning must be put in place if necessary, and if unsupervised is the preferred choice, then that the algorithm is coded perfectly to reduce the risk of errors, incorrect analysis, and misleading interpretations.

Ethics

Now that we have explored measurable data, computational platforms and algorithmic approaches, the final area of interest we must touch on is ethics. In the modern world, the vast majority of jobs are being tracked, in order to monitor performance. This could be as simple as monitoring a salespersons sales per month. This monitoring is conducted to benefit both the company and the employee in an attempt to improve performance. The concept and idea is very useful and has been proven to be quite effective. The risk attached to this lie within ethical issues, and the amount of information and data you can hold on an individual, which we are going to look into.

Boundaries

One thing companies must be aware of when processing and analysing individual data is privacy. One main privacy issue today is location services. Since the introduction of 'SnapMaps', it almost seems as though individuals are being watched 24/7. You must ask yourself whether you have any privacy any more? Another common issue is in relation to social media sites such as Facebook and Instagram. From reading an article by Josh Wolford (Wolford, 2015), he talks about the fact that he has become increasingly aware of advertisements being targeted at you, based on what you had been talking about in person a few hours, or even minutes beforehand. Wolford explores the issue, with other users also having similar problems. He mentions that, when referring to the frequency of these ads appearing, 'one's an incident, two's a coincidence, three's a pattern'. This could possibly be due to the user granting 'access' to the likes of Facebook and Instagram to the microphone on their phone. This leads to the question whether these large companies are breaking privacy boundaries? Are companies manipulating users by asking them to access their microphone, when the user has not attempted to use it to take a video for example? Or should users be more aware of what they are granting access to, and looker into the finer detail, instead of just brainlessly clicking 'accept' to everything to see?

Regulations

When it comes to the collection and storing of data, there are regulations which companies must abide with. These are an attempt to ensure that organisations don't go beyond any boundaries, such as the ones mentioned above. The main regulatory act in the European Union is the EU's General Data Protection Regulation (GDRP). This has only come into effect very recently, as it has surpassed the European Data Protection Act (DPA). Companies must now keep record of how and when they obtained personal information on an individual, and must also delete any information on them if the individual no longer wants their personal information on a company's system. The majority of individuals however are unaware of their rights as a data subject. With data analysis becoming prominent everywhere these days, and large streams of data being stored in a variety of places, it is important that people know what rights they have as a data subject.

Discriminatory Issues

Now that managers and organisations possess a large amount of data on their employees, this could lead to some issues with regards to discrimination. If a manager found out that an employee is pregnant, has a particular religious belief, or has come from a specific background, they could discriminate against this employee. This is information that a manager could use when determining which employee to give a promotion to. This shows that although the data is very beneficial and useful, it can have a negative impact when being misused. This problem could be avoided with the use of a 3rd party data company, who would be responsible for the collection and handling of personal information.

Overall, with the large amount of data companies hold on individuals, I feel it is important to ensure people are not being taken advantage of for a companies use. This is why I would be in favour of using 3rd party data companies, to protect individuals as much as possible, as today it seems as though nobody has any privacy anymore.

Conclusion

To conclude, in this report I have explored the process in how a software engineers performance can be measured -> through the collection of data, the computations and work that is then completed on this data, and the final analytical measurements that are explored. On the other hand, ethical issues were covered, particularly in obtaining and storing the data.

My personal opinion on this process is that I am in agreement with using data to analyse performance. The insights and analysis that can be obtained through this are second to none, as an employee and employer can both see, from actual facts, where the developers strengths and weaknesses lie. This can then help the employee to improve their own performance, which in hand helps the company. This is applicate not just to software engineers, but to almost every occupation. My only recommendation is that a 3rd party company deals with collecting and storing this data. This company would be much more aware of all rules in relation to data protection, and could make the employer and employee aware of any necessary regulations.

They would also only provide relevant information, significantly decreasing the risk of any discriminatory issues, and improving the individuals privacy. Overall, in my opinion, when considering risk versus reward, I think data analysis is a must in the fast – paced, technology heavy world we live in today, where the opportunity for improvement must be utilised.

Bibliography

Coase, R. (2016). *Springboard.* Retrieved from www.springboard.com

Code Climate. (n.d.). Retrieved from www.codeclimate.com

Demeyere. (n.d.). Retrieved from s3.amazonaws.com

Deming, W. E. (1990). *Analytics Hero.* Retrieved from www.analyticshero.com

Dillard, J. (2017). *Big Sky Associates.* Retrieved from www.bigskyassociates.com

Hawaii. (n.d.). Retrieved from http://csdl.ics.hawaii.edu

Houlding, B. (2018). Retrieved from www.scss.tcd.ie/Brett.Houlding/ST3011

Johnson, P. (2009). Retrieved from sonarqube-archive.com

Techopedia. (n.d.). Retrieved from www.techopedia.com

VITRU. (n.d.). Retrieved from govitru.com

Wikipedia. (n.d.). Retrieved from www.wikipedia.com

Wolford, J. (2015). Retrieved from www.webpronews

Zakka, K. (n.d.). Retrieved from kevinzakka.github.io