MEASURING THE SOFTWARE ENGINEERING PROCESS

INTRODUCTION

The last number of decades has seen a rapid growth in technology and software engineering, both of which are continuing to advance into new and unexplored territories. Along with this, a rapid growth in data has been observed as a by-product of the exponential increase in the volume by which people use technology and their respective advancements. Software engineering and the development of these platforms are no exception.

This rapid growth of data has led to the need for data analysation on a daily basis. Everything is scrutinised and critically evaluated in the attempts to maximise efficiency and improve quality in every aspect of our lives. Every single process is measured and assessed, problems are identified as early as possible, and actions are outlined to resolve and mitigate any future issues for users. These processes occur in all environments; ranging from personal goals that an individual may set for themselves, to the formal setting of the workplace. However, the process which will be discussed in this report, is that of the software engineering process.

When the words 'measurement' and 'assessments' are used, a reasonable connection one might make, is something consisting of a numerical foundation. With the technology available at present, an exorbitant amount of analysis can be carried out at the click of a button. The same analytical techniques can be devised, and repeatedly used to analyse an incomprehensible number of processes. Some say that about 90% of all data in the world today has been created since the start of the millennium. Therefore, the onus of sorting, analysing and computing data falls on machines and software. While, we as users are the ones which ultimately come to our own conclusions from the computation of data, technology is the medium which facilitates and allows anyone at all to study data relating to an immense amount of topics.

We have the ability to analyse processes, behaviour and data in order to ensure the smooth and consistent growth of companies, the free and orderly operations within governments and the structural integrity of the legal system, among others. This practice is more than applicable in terms of software engineering.

In this report, I will discuss a number of issues relating to the measurement of the software engineering process.

Structure:

1. The ways which software engineering practices can be measured and assessed in terms of measurable data.
2. Offer an insight into computational platforms available.
3. The algorithmic approaches implemented in analysation.
4. The ethics involved in the analysation of data.

THE SOFTWARE ENGINEERING PROCESS

Software engineering is a widely used term for the process of developing a software. It is the concept of conceiving and applying engineering rules in order to create a software. The process generally consists of five steps: 1) requirement analysis, 2) software design, 3) unit testing, 4) system testing and 5) maintenance.

1. MEASUREMENT OF SOFTWARE ENGINEERING IN TERMS OF MEASURABLE DATA

Before looking at different types of data, we must first carefully consider the exact kind of data that will prove beneficial. Collecting data leads to challenges as one must ensure that the data will "provide useful information for project, process, and quality management and, at the same time, that the data collection process will not be a burden on development teams".

In general, the aim is to minimise cost and time, and maximise efficiency and productivity. However, to achieve this aim, we must first ask ourselves a number of questions. What data do we require? What should we take into consideration? What data is relevant to measure and assess a software engineer's performance?

RELEVANT DATA

In terms of data relating to the software development process, a wide range of data is available concerning the work of the software engineer.

Relevant data may include:

Data relating to the software engineer's tangible projects:

- Number of commits
- Number of contributions
- Frequency of contributions
- Individual performance
- Number of team projects
- Team performance
- Technical debt
- Bug fixing

Non project related data:

- Repository dates
- Company joining data
- Meetings attended
- Communications with team members
- Communications with managers
- Communications with clients
- Gender
- Job satisfaction and purpose

With the completion of the data collection, comes an important component in data analytics known as taxonomy. A taxonomy provides the framework for your data, enabling you to analyse the performance of a couple of dimensions that matter most to your data. By implementing a taxonomic framework, your data will consistently be categorised by a number of classes. This classification of data can be achieves through various computational platforms such as SQL (structured query language) which will store data in a large database.

DATA COLLECTION

Manual data collection is unreliable; often being both time consuming and costly. It often results in missing data or data that is error filled which will affect the analysation process.

Human error will also often lead to unreliable data, so a solution is the automation of the process. The automation of data collection improves both the data quality and number of hours that an employee must spend in their acquisition of data. The Personal Software Process (PSP) is a process whereby a software engineer is provided with a framework as to how they should track their work. This results in a large amount of data where automated tools such as Hackystat and PROM are useful.

In theory, it allows software engineers to track development, identify areas for improvement, and ultimately assess and measure performance.

DATA COLLECTION TOOLS

Hackystat is a data collection tool that enables developers to automatically process and analyse data that has been gathered in the PSP. The tool is focused on ensuring the privacy of the individual.

PROM (Pro Metrics) is an automated tool for collecting and analysing software metrics and PSP data. It uses plug-ins that continuously collect data from the tools used by the developers.

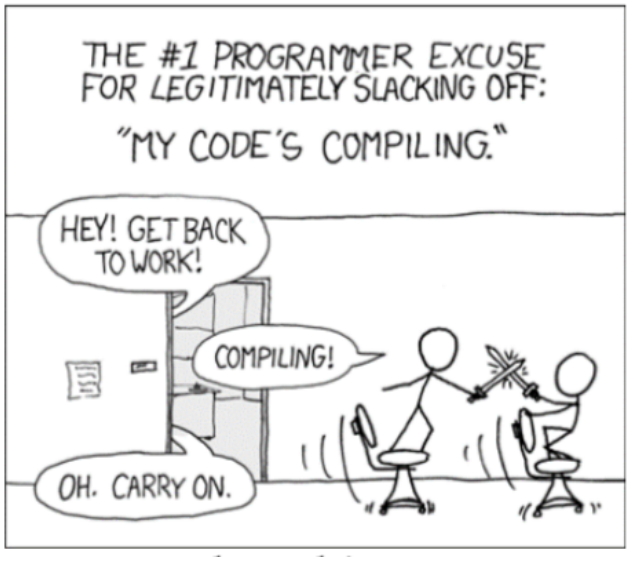
PROM analyses the whole development process including non-code related topics, providing a comprehensive overview of all team-members whereas Hackystat is solely focused on code related activities and individual software engineers.

MEASUREMENT AND ASSESSMENT OF SOFTWARE ENGINEERING PRACTICES

Once the data has been collected, in order to accurately assess and measure performance, the three main measurable branches of analytics; those of time, size and quality, come into account. We cannot say that one software engineer is better than another because he completed a project faster, if we do not know the size of the respective projects – just like we cannot say that one software engineer is better because he has more lines of code, if we do not know how much of the code is actually efficient.

I believe that in measuring the proficiency of software engineering practices, one should also take into account softer data such as job satisfaction.

TIME



In measuring productivity, we must consider the amount of time spent on each process of a project. Attainable deadlines must be set for each individual stage of a project in order to instil a sense of determination in employees and increase productivity. As discussed by Humphrey in "The Personal Software Process", the calculation of project time depends on three factors: the time which the work started, the time the work finished and interruption time. Interruption time is completely random in that it follows no discernible pattern and cannot be predicted. While it follows no set pattern, omitting it from our time calculations would reduce the accuracy of the analysis.

In Lowe's article, '9 metrics that can make a difference to today's software development teams' , he highlights the importance of planning by further splitting time into four separate measurable components. These four measures include lead time (the time it takes to go from an idea to delivered software), cycle time (the time it takes to make changes to software systems), team velocity ('units' of software the team typically completes in an iteration) and open/close rates (the production issues reported and closed within a set period).

Detailed time records need to be kept in order for assessment to take place. Due to the fact that time records are solely reliant on the input of the software engineer, there may be resulting drawbacks. For example, the engineer may overstate the task's complexity in order to always finish a head of the deadline. For this reason, these time logs should be reviewed in conjunction with the size/difficulty of the task completed in this time.

SIZE

If you were told that Donald Trump won a race in 9.25 minutes and George W. Bush did 9.11, the first question you would need to ask, is whether or not their respective distances were equal.

This brings into perspective the critical link between time and size. However, this poses complexities as there are no physical dimensions for software.

Lines of code (LOC) is the principal measure of size, as outlined in "The Personal Software Process". The LOC size of code is the sum of added, modified or reused code, minus any deleted code.

Even if the software engineer takes both time efficiency and task complexity/size into account, these measures are often not enough in reliably measuring performance metrics. Often, these two should be combined with code quality.

QUALITY

Our final and most important measurable performance metric is code quality. This relates to anything that could detract from the program's ability in meeting user needs.

Firstly, **defect density** refers to the number of defects per new and changed LOC in a software engineer's program. In an ideal world, the software would never fail, however, this is statistically improbable. It is important to look at the defect density, rather than the number of defects for the reason that larger code is likely to have a much larger number of defects.

Running in tandem with defect density, is the second measure of quality, the **application crash rate**. This refers to the number of times an application fails divided by the number of times it is used. Along with the application crash rate, the recovery time for the application is also important.

It is said that a typical software engineer writes somewhere between 150 to 200 lines of code per hour. Due to the fact that they are writing at such a quick rate, they may miss numerous defects. The third measure of quality is the **review rate**. The review rate is the rate at which engineers find all or most of their program's defects.

SOFTER DATA

It is my opinion that to successfully measure a software engineer's performance, softer data must also be considered, however this is often harder to assess. This softer data may include the job satisfaction of the software engineer. The traits of satisfied employees include autonomy, mastery (a strive for accomplishment and self-improvement) and a sense of purpose.

2. COMPUTATIONAL PLATFORMS AVAILABLE

ANALYTICS AS A SERVICE

For companies to analyse their data, they require analytics processes. These analytical processes are labour-intensive and costly and will often require the purchase of different kinds of hardware.

Analytics as a service (Aaas) is the provision of analytics software and operations through web-delivered technologies, and could bypass the company having to develop analytical processes of their own. Analytics as a service also have the ability to reduce compliance risks as workers can access data sets through an online portal, without having to duplicate it. When the software engineer is finished with datasets, they simply disconnect from a virtual access point and no further clean-up is required.

Aaas results in reduced costs and compliance risks while increasing user productivity. By using Aaas, employees have the ability to make more informed decisions and deliver more significant outcomes.

UNIT TESTING PLATFORMS

Parasoft is an integrated Development Testing solution that focuses on the productivity of development teams and the overall software quality. Their aim is to provide end-to-end functional testing for complex applications. Parasoft automatically analyses code and generates an extensible and reusable high-coverage test suite. This test suite exposes functional problems and defects that cause programs to crash. Functional tests are created via 'tracing'; a technique which mimics the application behaviour.

McCabe IQ follows the same approach as Parasoft. Their analysation of code is done through the scrutinization of test activities in a path-orientated approach. Their Test Team edition assesses the thoroughness of a program's testing and establishes the resources to ensure a well-tested and reliable application.

CODE COVERAGE AND CODE QUALITY PLATFORMS

A good software engineer will ensure that they are testing as much of their code as possible.

Code Climate is a platform which "incorporates fully-configurable test coverage and maintainability data throughout the development workflow, making quality improvement explicit, continuous, and ubiquitous". Code Climate is in use by over 100,000 projects of a variety of different languages and analyse over 2 billion lines of code per day. Code Climate enables organisations and engineering teams to take control of the quality of code they produce by introducing "fully configurable test coverage and maintainability data throughout the development workflow". In software development, a huge concern is technical debt. This reflects the marginal cost of reworking code by choosing an easier solution now, rather than using a more sustainable solution in order to minimise the development time. Code Climate is beneficial in that it aims to minimise the chances of technical debt by frequently identifying changed files that add no increased value to an overall project. It is also extremely useful for the software engineer in that it is able to be seamlessly integrated with GitHub, directly providing line by line technical debt and code coverage. Other features of the platform include a week-by-week progress tracker which analyses each GitHub contributor's performance and a trend section which analyses the program's performance over time.

Code Climate is used by companies such as Harvest, Litmus and Chargify.

Parasoft also specialise in code coverage analysis whereby test traceability is ensured. Their service measures code coverage to identify untested code and untested functionality and also, tests traceability to understand the impact of changes to code.

Parasoft is used by companies such as Comcast, Apple, AIG, Samsung, Coca-Cola and Lufthansa.

CODE QUALITY

Codebeat is a platform integrated with GitHub, GitLab and Bitbucket which aims to increase code quality between members of a development team by giving instant feedback on a developer's code. It provides an automated code review, helping the developer prioritise issues and identify quick solutions.

Codacy checks "code style, security, duplication, complexity and coverage on every change while tracking code quality throughout".

RISKS OF ANALYTICS AS A SERVICE

While there is undoubtedly many benefits to employing Analytics as a Service to measure software, they are not without their risks. For example, there are concerns with the safety and privacy of data that is stored remotely. In using Aaas, there is a trade-off between communication/collaboration and security. A firm must ensure that the service chosen is entirely secure and reliable. Developers and researches must weigh the benefits of various platforms and decide whether they prefer easily-obtained analytical methods or richer analytics with privacy and overhead concerns.

Aaas allows easy installation and integration, often reducing costs for customers, but the results can provide little illumination of software processes and products. While these platforms may score the software engineer's programs well on a technical basis, they will not determine if the programs are accurately answering the initial and overall purpose of the assessment, nor do they support behavioural, client-side data.

PLATFORM OVERVIEW

Businesses looking to assess the software engineering process may perhaps invest in a hybrid of these platforms to ensure their specific needs are being met.

3. ALGORITHMIC APPROACHES AVAILABLE

BRUTE FORCE VS. HEURISTIC

One way of approaching a software engineer process is by means of a brute force algorithm. A brute force algorithm will exhaust every single possibility until a satisfactory result is found. Brute force algorithms consider every possible occurrence but are the most costly and time inefficient of all algorithms.

An alternative approach to brute force algorithms are heuristic algorithms. Heuristic algorithms find solutions faster, but may sacrifice accuracy in doing so. Heuristic algorithms find a viable process, however, they do not ensure that the viable solution found is the most optimal of all solutions. In using a heuristic algorithm, there is a trade-off between time efficiency and accuracy.

Using mathematical modelling, there are other algorithms which can be built, however, a high degree of expertise is often required for this.

MACHINE LEARNING ALGORITHMS

In machine learning, the machine has the ability to think intelligently; to learn, apply reasoning and use self-direction in finding a solution. Systems have been created whereby computational algorithms can absorb new information, put this information together and find new solutions dynamically. The majority of success in machine learning algorithms is generally from supervised learning systems.

Supervised learning is a type of machine learning algorithm which uses a training dataset, that includes input data and response values, in order to make predictions. The algorithm then creates a model which predicts values for new data. The larger the test dataset, the more accurate the model will be.

Unsupervised learning is where the test file only has input data and no corresponding output values. These unlabelled datasets are sorted according to similarities or differences by the machine. The algorithm models the underlying structure and attempts to establish patterns in the software engineer's behaviour.

SUPERVISED MACHINE LEARNING

LOGISTIC REGRESSION

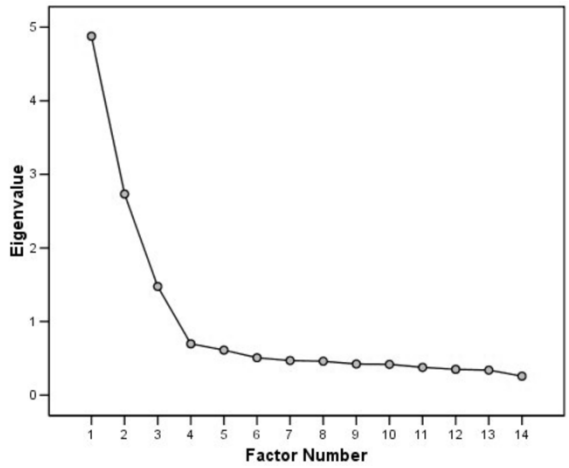
Logistic regression is a predictive modelling algorithm that is used when the Y variable is binary categorical, meaning that it can only take two values like 0 or 1. Logistic regression is also known as a binary classification problem. The goal is to determine a mathematical equation that can be used to predict the probability of the two events. Once the equation is established, it can be used to predict Y for each input variable.

UNSUPERVISED MACHINE LEARNING

PRINCIPAL COMPONENT ANALYSIS (PCA)

For datasets with multiple variables, it can be extremely difficult to understand and analyse inherent associations.

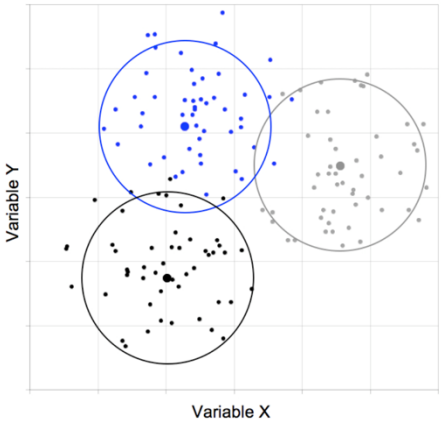
Principal component analysis (PCA) is an unsupervised machine learning algorithm that can be thought of as a dimension reduction technique, or a method for identifying associations among variables. The aim of PCA is to describe the variation in a set of correlated variables through linear combinations of the original data. A set of correlated variables is described in terms of a new set of uncorrelated variables, called principal components. Most of the dataset's variability will be explained by the first few principal components, allowing us to grasp the data surrounding the software engineer's performance from a more compact perspective. If we were to assume that all of the collected data has a strong impact on the software engineer's performance and productivity levels, we could be modelling inherent noise.



A scree plot is a line segment plot which shows the fraction of total variance in the data as explained by each principal component. Using the scree plot for visualisation purposes, and choosing a percentage of data that we want accounted for (for example, 90%), we can determine the number of principal components which should be used.

The scree plot that after 4 principal component, the plot begins to smoothen and level off, meaning that by including additional principal components, little additional variance is explained.

CLUSTER ANALYSIS



Cluster analysis is an unsupervised machine learning algorithms which aims to distinguish group structure within a dataset. The objective with cluster analysis is to cluster the observations in such a way so that there is little dissimilarity within groups, but large dissimilarity between groups.

To perform a cluster analysis, we input the selected data from Principal Component Analysis into a clustering algorithm. The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one clustering algorithm over another. A popular clustering algorithm used is a k-means cluster analysis.

Once the preferred cluster algorithm is chosen, the computer will predict and make comments on the performance of the software engineer. The algorithm would ideally cluster the data into groups ranging from excellent performance to bad performance and from here.

4. ETHICS

Throughout this report, my main focus has been on the logistics of collecting data and how this data is analysed. However, data ethics is extremely topical in the last number of years, as data manipulation and security concerns have become more prominent in global media. The ability to collect and analyse data is moving much faster than any legal and ethical guidelines can manage. In most cases, businesses are collecting data for the one purpose – to increase business profits. However, the methods which data is collected may appear as obtrusive and unethical by users if they feel that what they are doing is being watched or scrutinised. The growing concern is that perhaps there is no concrete privacy line when it comes to data.

In looking at Facebook as an example, specifically tailored ads for each user are shown on the newsfeed. According to a 2016 study, Facebook is able to identify users who plan on buying a car, what groceries a user buys, the types vacation a user goes on, the user's relationship details, etc. This leads to us asking the question: is anything actually private anymore?

It is extremely important to abide by regulations surrounding data privacy, and ensure consumers are aware of their data will be used.

DATA SOVEREIGNTY

The wide-spread adoption of cloud computing services, as well as new approaches to data storage, have broken down traditional geopolitical barriers more than ever before. In response to this, many countries have introduced compliance requirements by amending their current laws or enacting new legislation.

Data sovereignty is the concept that data in digital form is subject to the laws of the country in which the data resides. The main concern with data sovereignty is maintaining privacy regulations and keeping foreign countries from being able to subpoena data.

Verifying that data exists only at allowed locations can be difficult. It requires the cloud customer to trust that their cloud provider is completely honest and open about where their servers are hosted and adhere strictly to service level agreements.

DATA OWNERSHIP

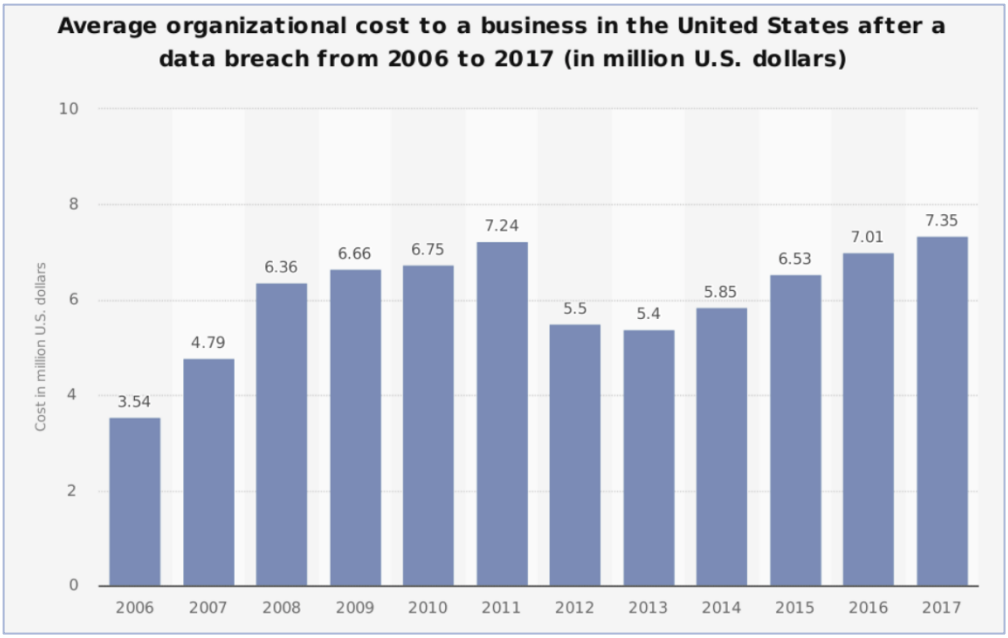
An important question being asked is who actually owns the data online?

In returning to Facebook as an example; if a user makes a post, do they retain the ownership or do Facebook claim the rights? Laws in the past have been governed on the basis that assets are physical and non-replicable objects.

Data ownership is primarily a data governance process that details an organisation's legal ownership of enterprise-wide data. The data owner has the ability to create, edit, modify, share and restrict access to the data. Data ownership defines the data owner's ability to surrender all of these privileges to a third party. The data owner claims the possession and copyrights to such data to ensure their control and ability to take legal action if their ownership is illegitimately breached.

DATA BREACH EXAMPLES

Studying data breaches within companies is a good way to tie these ethical issues in at a corporate level, thus hopefully mitigating the chances of them happening again. Companies must stay ethical and ensure there are no breaches of data when analysing and measuring the software engineering process.



FACEBOOK

Attackers exploited a vulnerability in the code of Facebook's "View As" tool, a feature that shows users what their profile looks like to other people, in September 2018, exposing up to 90 million Facebook user accounts.

Facebook responded by fixing the vulnerability, informing law enforcement and resetting the access tokens of all the affected accounts.

The breach sent Facebook stock tumbling but the real cost is yet to come. Under the terms of GDPR, the company faces a maximum fine of up to 4 percent of its global annual revenue from the prior year, which works out at $1.63 billion.

GOOGLE+

Earlier this year, Google discovered a vulnerability with Google+ which made it possible for third-party app developers to access data from the friends of the app users. According to documents reviewed by the Wall Street Journal, Google not only exposed this data but then chose not to disclose it. In response, parent company Alphabet decided to shut down Google+ for good.

GDPR

Introduced in May of this year, the General Data Protection Regulation (GDPR) standardises data protection law across all EU countries and imposes strict laws on the control and the processing of data. It also extends the protection of personal data by giving control back to EU residents. There are many essential items in the regulation, including increased fines, breach notifications, opt-in consent and responsibility for data transfer outside the EU. As a result, impact to businesses is huge and will permanently change the way customer data is collected, stored and used.

CONCLUSION

The software engineering process has developed over the past number of years due to the rapid surge in technology that has facilitated its growth. For the growth of software engineering to continue, it is necessary to measure and assess the software engineering process in concluding what exactly makes a software engineer productive and beneficial.

In this report, I discussed the manner in which the software engineering process can be measured and assessed in terms of measurable data. Specific processes such as the PSP guide outlines how software engineers should track and record their work, while analytical tools such as Hackystat and PROM automate the process providing analysis of the data for accurate measurement of a software engineer's performance and productivity levels.

I also offered an insight into the computational platforms and algorithmic approaches available in implementing accurate analysation. Computational platforms such as Code Climate and Parasoft facilitate the accuracy and ease with which the software engineering process can be measured. Supervised algorithms such as logistic regression and unsupervised algorithms, such as principal component analysis and hierarchical clustering enable the production of powerful predictive models. With respects to the software engineering process, these algorithms can improve the engineer's performance by looking at data directly related to their work, and also by looking at non-work related data.

Lastly, I discussed the ethics that are involved in terms of analysation of data. Data analytics has advanced at such a velocity that conclusions could potentially be drawn on a person's sex or nationally based on how they code their data. This feeds into how ethics plays a part in the analysis today.

In my opinion, as long as companies adhere to regulations and ensure they are being ethical in their analysis, their measurements and assessments of the software engineering performance will ultimately lead to improvements for both the business and their employees.

BIBLIOGRAPHY

1. Code Climate. (n.d.). Retrieved from https://docs.codeclimate.com/
2. Parasoft. (2018). Retrieved from Parasoft: https://www.parasoft.com/
3. Code Beat. (2018). Retrieved from https://codebeat.co/
4. Codacy. (n.d.). Retrieved from https://www.codacy.com/
5. Humphrey, W. S. (2000, November). The Personal Software Process. Pittsburgh, PA: Carnegie Mellon Softwaer Engineering Institute. Retrieved from The Personal Software Process.
6. Johnson, P. M. (n.d.). Searching under the Streetlight for Useful Software Analytics. Manoa: University of Hawaii.
7. https://www.techrepublic.com/article/5-ethics-principles-big-data-analysts-must-follow/
8. Martin, E. R. (2014, March 27). The Ethics Of Big Data. Retrieved from Forbes: https://www.forbes.com/sites/emc/2014/03/27/the-ethics-of-big-data/#c8717fa6852e
9. Mathiprakasam, M. (2014, September 16). The Road to Analytics As A Service. Retrieved from Forbes: https://www.forbes.com/sites/oracle/2014/09/26/the-road-to-analytics-as-a-service/#7d44a9093622
10. Roie, S. (2017, July 25). Analytics as a Service. Retrieved from Birst: https://www.birst.com/business- insights/analytics-as-a-service/
11. Atos. (2013, March). Data Analytics as a Service: unleashing the power of Cloud and Big Data. https://atos.net/wp-content/uploads/2017/10/01032013-AscentWhitePaper-DataAnalyticsAsAService.pdf
12. https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/
13. Image Source: https://www.empirical-methods.hslu.ch/decisiontree/interdependency-analysis/grouping-of-objects/3170-2/
14. Rouse, M. (2013, March). Data Sovereignty. https://whatis.techtarget.com/definition/data-sovereignty
15. Fresser, D. (n.d.). Data Sovereignty: What it is and why it matters. https://rgtechnologies.com.au/resources/data-sovereignty/
16. Techopedia. (n.d.). Data Ownership: https://www.techopedia.com/definition/29059/data-ownership
17. Techworld. (2018, November 5). The UK's most infamous data breaches. https://www.techworld.com/security/uks-most-infamous-data-breaches-3604586/
18. Forbes. (2018, February 14). What Is General Data Protection Regulation? https://www.forbes.com/sites/quora/2018/02/14/what-is-general-data-protection-regulation/#4433b00e62dd