Software Data Analytics

Nevena Lazarević
Selected Literature


“Perspectives on Data Science for Software Engineering”, 1st Edition, Tim Menzies, Laurie Williams, Thomas Zimmermann
Roadmap

> What is Software Data Analytics?
> Software Data
> Software Data Analytics Guidelines
Roadmap

> What is Software Data Analytics?
> Software Data
> Software Data Analytics Guidelines
What is Software Data Analytics?

“Software analytics is analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight from their data to make better decisions.”


“Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services (and software practitioners typically include software developers, testers, usability engineers, and managers, etc.).”

History of Software Data Analytics

1st generation (‘50s - ‘80s)
- Akiyama: number of defects depends on the number of LOC
- McCabe’s complexity
- Lehman’s laws

2nd generation (late ‘80s - 2000)
- Selby & Porter: decision trees can identify components that could contain errors

3rd generation (2000 - today)
- SE data science is applied more
- Ostrand et al. show that 20% of the code count for 80% of the bugs

4th generation (today)
- it becomes a daily routine
- large-scale usage
Roadmap

> What is Software Data Analytics?
> **Software Data**
> Software Data Analytics Guidelines
Software Data

> Goal - Question - Metrics methodology
> The data should reveal information that is otherwise not obvious.
> Three groups of software data:
  1. Contextual data
  2. Constraint data
  3. Development data

1. Contextual Data

This is the data related to the product or team.

Examples:
1. Project objectives (e.g. frequency of releases)
2. Audience (e.g. is software browser-based)
3. Project history (e.g. number of new features for each release)
4. Organizational structure (e.g. average number of years of experience of developers)
5. …
2. Constraint Metrics

This is the data related to the product constraints, the goals that need to be fulfilled.

Examples:

1. Quality metrics (e.g. the reliability of the code)
2. Compatibility metrics (e.g. does the software need to run on a specific platform)
3. Performance metrics (e.g. number of seconds that the user can wait for a response)
4. Legacy code (e.g. the amount of legacy code)
5. …
3. Development Metrics

This is the data related to the development process, verification and code readiness.

1. Code churn (number of lines added/deleted/changed)
2. Code velocity (the time needed to implement a new functionality)
3. Complexity (e.g. McCabe’s complexity)
4. Dependency (e.g. the depth of the dependency tree)
5. …
Roadmap

> What is Software Data Analytics?
> Software Data
> Software Data Analytics Guidelines
Software Analytics Guidelines

> Problem identification

> Collecting data

> Descriptive analysis

> Predictive analysis

> Performance evaluation

Software Analytics Guidelines

> Problem identification

> Collecting data

> Descriptive analysis

> Predictive analysis

> Performance evaluation
Problem Identification
Usually, the problems in software development process are defined on the business side. These problems need to be translated into technical questions, in order to get the answers. Clear communication between the stakeholders and the developers/team leaders is the basis for any successful software data analytics process.
Problem Identification

- The problem must be clearly defined both from the business side and from the technical side.

- The problem should be resolvable with the accessible software data.

- The process of getting to the common point (agreeing on the defined problem) is usually iterative.
Types of Software Analytics Problems

- Bug Measurements
- Development Practices
- Testing Practices
- Evaluating Quality
- Software Development Lifecycle
- Productivity
The paper lists the questions of interest for software developers split in several categories:

- Bug Measurements (e.g. when was the bug introduced)
- Development Practices (e.g. are comments really important for code comprehension)
- Testing Practices (e.g. how much time should I spend on writing tests)
- Evaluating Quality (e.g. knowing the best metrics for software quality)
- Software Development Lifecycle (e.g. how to allocate time between developing and testing)
- Productivity (e.g. how to measure a developer’s productivity)
Software Analytics Guidelines

> Problem identification
> Collecting data
> Descriptive analysis
> Predictive analysis
> Performance evaluation
Source of the Data

GitHub
Today, there are plenty of sources of software data: the code itself, mailing lists, bug reports etc. With the development of technology (computational power and internet above all) these sources have become more available than before.

There are also online repositories which contain software data, e.g. QualitasCorpus (http://qualitascorpus.com/) and Bug Prediction Dataset (http://bug.inf.usi.ch/index.php). They can be used for experiments.
Data Features

> Data resolution

> Data structure

> Data integration
Data resolution defines the level of granularity to which the data should be collected or aggregated. The very same data collected on the class level or file level may result in different information. This is known as ecological fallacy.

Data structure answers the question of whether the data is numerical or descriptive, for example. It also includes deciding on what to do when the data is missing or corrupted.

Data integration refers to combining the data coming from distinct sources. Related data can come from the code itself or mailing lists, and thus can be presented in different ways. It should then be correctly combined.
Software Data

> Static metrics

(e.g. LOC, McCabe’s complexity, Halstead metrics, Chidamber-Kemerer object-oriented metrics)

> Churn metrics

(e.g. number of commits, number of added/removed lines in a file)
Static metrics:

Lines of code (LOC) may refer to the total lines of code, the blank lines of code.

The idea behind the McCabe’s complexity a.k.a. cyclomatic complexity (https://en.wikipedia.org/wiki/Cyclomatic_complexity) is that the more complex the software becomes, the more difficult it is to maintain and test it.

Halstead metrics is related to the number of unique operators and operands.

Chidamber-Kemerer metrics are object-oriented related. They refer to the number of immediate children of a class, class coupling, depth of inheritance etc.

Churn metrics relate to the size of the code likely to be changed or deleted during the software lifetime. When combined with dependency metrics, it can reveal most likely error-prone part of the software. If the part of the software which has a high likelihood of churning has a lot of dependants, the dependent parts are also likely to change, and possibly introduce bugs.
Data Extraction Process

- Automate the process as much as possible.
- Store historical data for replication purposes.
- Reduce the level of noise to the minimum.
Software Analytics Guidelines

- Problem identification
- Collecting data
- Descriptive analysis
- Predictive analysis
- Performance evaluation
Descriptive Analysis

> Data visualisation

> Descriptive statistics
Data Visualization

> Line charts

> Scatter plots

> Bar charts and histograms

> Box plots
Descriptive Statistics

- Minimum
- Maximum
- Median
- Mean
- Variance
Statistical Tests

> Correlation tests (Pearson’s correlation coefficient and Spearman’s rank correlation coefficient)

> Difference between distributions (Kolmogorov-Smirnov test)
Both the Pearson’s and Spearman’s correlation coefficients are used to measure the level of correlation between variables. They return the value from the segment [-1, 1] where -1 indicates strong negative correlation, 1 indicates strong positive correlation and 0 indicates that there is no correlation. Spearman’s coefficient indicates the monotonic relation between variables, while the Pearson’s indicates only the linear relation.

In software data analytics, Pearson’s coefficient which has a value larger or equal to 0.75 is considered to indicate strong positive correlation.

Kolmogorov-Smirnov test is used to measure the difference between the two probability distributions.
Software Analytics Guidelines

- Problem identification
- Collecting data
- Descriptive analysis
- Predictive analysis
- Performance evaluation
Predictive Analysis

> Predictive analysis employs machine learning algorithms.

> Always start from simple algorithms.

> Measure the added value of the algorithm with its complexity.

> “No free lunch” theorem.
Machine learning (ML) algorithms are often used nowadays for predictive analysis. There are plenty of machine learning algorithms, and there is not a single one which will always provide a decent solution for all problems ("no free lunch" theorem).

It is usually a good solution to start with a simple ML algorithm, which may already provide a satisfiable answer. If it does not, its results may be used as a baseline to which to compare the results of more complex algorithms. The added level of complexity of an algorithm should always bring the corresponding level of improvement in the results.
Predictive Analysis - Algorithms

- Linear regression
- Logistic regression
- $k$ Nearest Neighbours
- Neural networks
Linear Regression

> Fitting a linear function of arguments.

\[ f(x^i) = ax^i + b \]

> It can be univariate (one argument) or multivariate (several arguments).

> It usually uses square-error cost function.

\[ cost = \frac{1}{2m} \sum_{i=1}^{m} (f(x^i) - y^i)^2 \]

> Starts with some initial \( a \) and the learning parameter \( \alpha \).

> Repeats until convergence \( a = a - \alpha cost'(a) \).
Logistic Regression
Logistic Regression

\[ x + y - 5 = 0 \]

\[ x + y - 5 > 0 \]

\[ x + y - 5 < 0 \]
Logistic Regression

\[ \text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \]

\( z \) is the boundary line

\[ \text{sigmoid}(x + y - 5) = \frac{1}{1 + e^{-(x+y-5)}} \]
Logistic Regression

\[ \text{sigmoid}(x+y-5) > 0.5 \]

\[ \text{sigmoid}(x+y-5) < 0.5 \]
Logistic Regression

> It is used to classify the data.

> It is an off-the-shelf method for binary classification problems.

> Several logistic regression algorithms can be combined for multi-class classification problems.
$k$ Nearest Neighbours
**k Nearest Neighbours**

- Can be used both for classification and prediction of a numerical value.
- Data should be represented in a vector space.
- The algorithm looks into $k$ nearest examples and predicts the output value based on the neighbours’ values.
- For classification the algorithm may use classes of neighbours.
- For regression the algorithm may use mean value of the neighbours’ values.
Neural Networks

> Sensitive to data.
> May overfit or underfit the data.
Important advices

> Ensemble methods may be very useful.

> Always perform feature normalisation.

> Apply feature selection (e.g. through Principal Component Analysis).
Software Analytics Guidelines

> Problem identification
> Collecting data
> Descriptive analysis
> Predictive analysis
> Performance evaluation
Performance evaluation

> Confusion matrix
> Recall and precision
> F-score
Source of information


> “Perspectives on Data Science for Software Engineering”, 1st Edition, Tim Menzies, Laurie Williams, Thomas Zimmermann

What you should know!

> What is software data analytics?
> What is a use case for software data analytics?
> Which metrics are used for software data analysis?
> Which are the stages of software data analysis?
> What should we pay attention on when collecting the data?
> Which data visualisation techniques can we use and what is their purpose?
> Which machine learning algorithms are mostly used in software data analytics?