ITEA 2 Office

High Tech Campus 69-3
5656 AG Eindhoven
The Netherlands

Tel	: +31 88 003 6136
Fax	: +31 88 003 6130
Email	: info@itea2.org
Web	: www.itea2.org

ITEA 2 is a EUREKA strategic ICT cluster programme

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D4.2: Industrial Data Analysis Algorithms

MEASURE

Edited by Sana Mallouli and Wissam Mallouli on 01/12/2017
Edited by Sana Mallouli and Wissam Mallouli on 15/05/2018
Edited by SELAMI BAGRIYANIK on 16/05/2018
Edited by Sana Mallouli and Wissam Mallouli on 16/05/2018
Edited by Sana Mallouli on 23/05/2018
Edited by Alin Stefanescu on 25/05/2018
Reviewed by Jérôme Rocheteau on 28/05/2018
Reviewed by Stephane Maag on 01/06/2018
Edited by Sana Mallouli on 04/06/2018
Edited by Wissam Mallouli on 04/06/2018
Edited by Alin Stefanescu on 06/06/2018
Edited by Sana Mallouli and Wissam Mallouli on 06/06/2018
Edited by Alessandra Bagnato on 06/06/2018

Executive summary

The objective of WP4 is to design and develop a platform to analyse the large amount of measurement data generated by the use cases carried out in WP5. The data analysis platform will implement analytics algorithms, to correlate the different phases of software development and perform the tracking of metrics and their value. The platform will also connect and assure interoperability among the tools, selected or developed in WP3, and will evaluate their performance and define actions for improvement and possible countermeasures.

This deliverable D4.2 describes the data analysis algorithms and supporting tools that are required to identify correlations within the large amount of data which can point out problems and lead to improvements for the developed systems or the development processes. Correlations covering all aspects of the system like functional behaviour, energy and timing separately or together are identified as well as correlations covering different stages of development.

The deliverable is organized as follows: Section 2 presents a state of analysis algorithms used for large amount of data. Statistical analysis algorithms and machine learning techniques are presented in detail. The section 3 presents the algorithms used in the MEASURE project. Constraints based analysis is used in quality guard tool developed by Softeam. Rule based correlation is used in MINT tool developed by IMT and relying MMT-Correlator developed by MTI. Different clustering algorithms provided the Elki analysis tool are also described by ICAM. Machine learning algorithms (e.g. SVM) used in Metrics Suggester tool are describe by IMT and finally, Stracker algorithms are presented by the University of Bucharest.

Table of Contents

1. Introduction

1.1. Role of this deliverable

This deliverable D4.2 describes the data analysis algorithms and supporting tools that are required to identify correlations within the large amount of data which can point out problems and lead to improvements for the developed systems or the development processes. Correlations covering all aspects of the system like functional behaviour, energy and timing separately or together are identified as well as correlations covering different stages of development.

1.2. Structure of this document

The deliverable is organized as follows: Section 2 presents a state of analysis algorithms used for large amount of data. Statistical analysis algorithms and machine learning techniques are presented in detail. The section 3 presents the algorithms used in the MEASURE project. Constraints based analysis is used in quality guard tool developed by Softeam. Rule-based correlation is used in MINT tool developed by IMT and relying MMT-Correlator developed by MTI. Different clustering algorithms provided the Elki analysis tool are also described by ICAM. Machine learning algorithms (e.g. SVM) used in Metrics Suggester tool are describe by IMT and finaly, Stracker algorithms are presented by the University of Bucharest.

1.3. Relationship with others MEASURE deliverables

This deliverable is linked to D4.1 where we describe how the developed algorithms and tools are integrated in the analysis platform of MEASURE.

1.4. Contributors

MTI: Is the coordinator of this deliverable. MTI worked on the state of art of analysis algorithms and participated in the description of rule-based correlation used in MINT tool.

IMT: Contributed by the algorithms used in Metrics suggester tool and also participated in the the rule-based correlation algorithms used in MINT tool

SOFTEAM: Contributed to the constraints-based analysis is used in quality guard tool

ICAM: Described the different clustering algorithms provided the Elki analysis tool

University of Bucharest: Contributed with the algorithms used in the Stracker tool

2. State of the art of analysis algorithms

2.1. Statistical analysis

Statistical analysis is used extensively in science, from physics to the social sciences including software engineering. As well as testing hypotheses, statistics can provide an approximation for an unknown that is difficult or impossible to measure. For example, the field of quantum field theory, while providing success in the theoretical side of things, has proved challenging for empirical experimentation and measurement. Some social science topics, like the study of consciousness or choice, are practically impossible to measure; statistical analysis can shed light on what would be the most likely or the least likely scenario. In the following, we present the following five fundamentals[1] – and learn to avoid their pitfalls – before advancing to more sophisticated techniques presented in section 2.2.

2.1.1. Mean

The arithmetic mean, more commonly known as “the average,” is the sum of a list of numbers divided by the number of items on the list. The mean is useful in determining the overall trend of a data set or providing a rapid snapshot of the analysed data. Another advantage of the mean is that it’s very easy and quick to calculate.

Pitfall:

Taken alone, the mean is a dangerous tool. In some data sets, the mean is also closely related to the mode and the median (two other measurements near the average). However, in a data set with a high number of outliers or a skewed distribution, the mean simply doesn’t provide the accuracy you need for a nuanced decision.

2.1.2. Standard Deviation

The standard deviation, often represented with the Greek letter sigma, is the measure of a spread of data around the mean. A high standard deviation signifies that data is spread more widely from the mean, where a low standard deviation signals that more data align with the mean. In a portfolio of data analysis methods, the standard deviation is useful for quickly determining dispersion of data points.

Pitfall:

Just like the mean, the standard deviation is deceptive if taken alone. For example, if the data have a very strange pattern such as a non-normal curve or a large number of outliers, then the standard deviation won’t give you all the information you need.

2.1.3. Regression

Regression models the relationships between dependent and explanatory variables, which are usually charted on a scatterplot. The regression line also designates whether those relationships are strong or weak. Regression is commonly taught in high school or college statistics courses with applications for science or business in determining trends over time.

Pitfall:

Regression is not very nuanced. Sometimes, the outliers on a scatterplot (and the reasons for them) matter significantly. For example, an outlying data point may represent the input from the most critical supplier or the highest selling product. The nature of a regression line, however, tempts you to ignore these outliers.

[1] *https://www.bigskyassociates.com/blog/bid/356764/5-Most-Important-Methods-For-Statistical-Data-Analysis*

2.1.4. Sample Size Determination

When measuring a large data set or population, like a workforce, you don’t always need to collect information from every member of that population – a sample does the job just as well. The trick is to determine the right size for a sample to be accurate. Using proportion and standard deviation methods, you are able to accurately determine the right sample size you need to make the data collection statistically significant.

Pitfall:

When studying a new, untested variable in a population, proportion equations might need to rely on certain assumptions. However, these assumptions might be completely inaccurate. This error is then passed along to sample size determination and then onto the rest of statistical data analysis

2.1.5. Hypothesis Testing

Also, commonly called t testing, hypothesis testing assesses if a certain premise is actually true for the data set or population. In data analysis and statistics, you consider the result of a hypothesis test statistically significant if the results couldn’t have happened by random chance. Hypothesis tests are used in everything from science and research to business and economic

Pitfall:

To be rigorous, hypothesis tests need to watch out for common errors. For example, the placebo effect occurs when participants falsely expect a certain result and then perceive (or actually attain) that result. Another common error is the Hawthorne effect (or observer effect), which happens when participants skew results because they know they are being studied.

2.2. Machine learning

2.2.1. An introduction to machine learning

“Machine learning is programming computers to optimize a performance criterion using example data or past experience.” [24]

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people [20].

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Machine learning is closely related to (and often overlaps with), which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

2.2.2. Machine Learning Process

Machine learning algorithms are only a very small part of using machine learning in practice as a data analyst or data scientist. In practice, the process often looks like:

1. Start Loop
    1. **Understand the domain, prior knowledge and goals**. Talk to domain experts. Often the goals are very unclear. You often have more things to try then you can possibly implement.
    2. **Data integration, selection, cleaning and pre-processing**. This is often the most time-consuming part. It is important to have high quality data. The more data you have, the more it sucks because the data is dirty. Garbage in, Garbage out.
    3. **Learning models**. The fun part. This part is very mature. The tools are general.
    4. **Interpreting results**. Sometimes it does not matter how the model works as long it delivers results. Other domains require that the model is understandable. You will be challenged by human experts.
    5. **Consolidating and deploying discovered knowledge**. The majority of projects that are successful in the lab are not used in practice. It is very hard to get something used.
2. End Loop

It is not a one-shot process, it is a cycle. You need to run the loop until you get a result that you can use in practice. Also, the data can change, requiring a new loop.

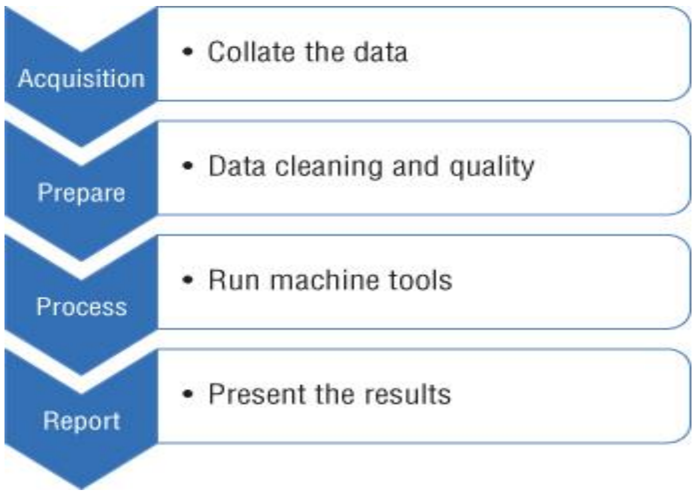


Figure 1. Machine learning Process

2.2.3. Machine Learning: types of learning

There are so many algorithms available that it can feel overwhelming when algorithm names are thrown around and you are expected to just know what they are and where they fit.

Two ways to think about and categorize the algorithms:

- The first is a grouping of algorithms by the **learning style**.
- The second is a grouping of algorithms by **similarity** in form or function (like grouping similar animals together).

Algorithms Grouped by Learning Style

There are different ways an algorithm can model a problem based on its interaction with the experience or environment or whatever we want to call the input data.

Generally, the field of machine learning is divided into three subdomains: supervised learning, unsupervised learning, and reinforcement learning. Briefly, supervised learning requires training with labeled data which has inputs and desired outputs. In contrast with the supervised learning, unsupervised learning does not require labeled training data and the environment only provides inputs without desired targets. Reinforcement learning enables learning from feedback received through interactions with an external environment. Based on these three essential learning paradigms, a lot of theory mechanisms and application services have been proposed for dealing with data tasks. This taxonomy or way of organizing machine learning algorithms is useful because it forces you to think about the roles of the input data and the model preparation process and select one that is the most appropriate for a specific problem in order to get the best result [21], [22].

Let's take a look at these different learning styles of learning:

Supervised Learning

Supervised learning is the task of inferring a function from **labeled training data**. The consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Example problems are classification and regression.

Example algorithms include Logistic Regression and the Back Propagation Neural Network.

Unsupervised Learning

Unsupervised machine learning is the task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). In other words, Input data is not labeled and does not have a known result. Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm which is one way of distinguishing unsupervised learning from and .

A model is prepared by deducing structures present in the input data. This may be to extract general rules. It may be through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity.

- Example problems are clustering, dimensionality reduction and association rule learning.
- Example algorithms include: the Apriori algorithm and k-Means.

Approaches to unsupervised learning include:

- Clustering
  - k-means
  - mixture models
  - hierarchical clustering,
- Anomaly detection
- Neural Networks
  - Hebbian Learning
  - Generative Adversarial Networks

- Approaches for learning latent variable models such as
    - Expectation–maximization algorithm (EM)
    - Method of moments
    - Blind signal separation techniques

Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning inspired by behaviourist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. In other words, RL enables learning from feedback received through interactions with an external environment. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, the field where reinforcement learning methods are studied is called approximate dynamic programming.

Table 1. A survey of machine learning for big data processing [23]

Learning types	**Data processing tasks**	**Distinction norm**	**Learning algorithms**
Supervised learning	Classification/ Regression/ Estimation	Computational classifiers	Support vector machine
		Statistical classifiers	Naïve Bayes
			Hidden Markov model
			Bayesian networks
		Connectionist classifiers	Neural networks
Unsupervised learning	Clustering/Prediction	Parametric	K-means
			Gaussian mixture model
		Nonparametric	Dirichlet process mixture model
			X-means
Reinforcement learning	Decision-making	Model-free	Q-learning
			R-learning
		Model-based	TD learning
			Sarsa learning

Algorithms Grouped By Similarity

Algorithms are often grouped by similarity in terms of their function (how they work). For example, tree-based methods, and neural network inspired methods.

This is a useful grouping method, but it is not perfect. There are still algorithms that could just as easily fit into multiple categories like Learning Vector Quantization that is both a neural network inspired method and an instance-based method. There are also categories that have the same name that describe the problem and the class of algorithm such as Regression and Clustering.

We could handle these cases by listing algorithms twice or by selecting the group that subjectively is the “best” fit. I like this latter approach of not duplicating algorithms to keep things simple.

2.2.4. Machine Learning: Algorithms

In this section, we list many of the popular machine learning algorithms. The list is not exhaustive in either the groups or the algorithms, but we think it is representative and will be useful to you to get an idea of the lay of the land [19].

Logistic regression

Logistic regression, which is borrowed from the field of classical statistics, is one of the simpler machine learning algorithms. This machine learning technique is commonly used for binary classification problems, meaning those in which there are two possible outcomes that are influenced by one or more explanatory variables. The algorithm estimates the probability of an outcome given a set of observed variables. Where logistic regression differs from other methods is in its interpretability. Since this algorithm is derived from the highly interpretable linear regression algorithm, the influence of each data feature can be interpreted without much effort. As a result, logistic regression is often favored when interpretability and inference is paramount. This versatile algorithm is used to determine the outcome of binary events such as customer churn, marketing click-throughs, or fraud detection.

Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model.

The most popular regression algorithms are:

- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

Instance-based Algorithms

Instance-based learning model is a decision problem with instances or examples of training data that are deemed important or required to the model.

Such methods typically build up a database of example data and compare new data to the database using a similarity measure in order to find the best match and make a prediction. For this reason, instance-based methods are also called winner-take-all methods and memory-based learning. Focus is put on the representation of the stored instances and similarity measures used between instances.

The most popular instance-based algorithms are:

- k-Nearest Neighbor (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organizing Map (SOM)
- Locally Weighted Learning (LWL)

Regularization Algorithms

An extension made to another method (typically regression methods) that penalizes models based on their complexity, favoring simpler models that are also better at generalizing.

The most popular regularization algorithms are:

- Ridge Regression

- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net
- Least-Angle Regression (LARS)

Decision Tree Algorithms (Random Forest)

Decision trees use directed graphs to model decision making; each node on the graph represents a question about the data and the branches stemming from each node represent the possible answers to that question. Compounding hundreds or even thousands of these decision trees is an "ensemble" method called a **random forest**. Decision trees are trained on data for classification and regression problems. They are often fast and accurate and a big favorite in machine learning. Though highly accurate, random forests are often dubbed black box models because they are complex to the point that they can be difficult to interpret.

The most popular decision tree algorithms are:

- Classification and Regression Tree (CART)
- Iterative Dichotomiser 3 (ID3)
- C4.5 and C5.0 (different versions of a powerful approach)
- Chi-squared Automatic Interaction Detection (CHAID)
- Decision Stump
- M5
- Conditional Decision Trees

Bayesian Algorithms

The Bayesian algorithm is a set of rules for using evidence (data) to change your beliefs. In simple terms, a Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple.

Bayesian methods are those that explicitly apply Bayes' Theorem for problems such as classification and regression. The most popular Bayesian algorithms are:

- Naive Bayes
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Averaged One-Dependence Estimators (AODE)
- Bayesian Belief Network (BBN)
- Bayesian Network (BN)

Clustering Algorithms

Clustering is a type of unsupervised learning, which is used when working with data that does not have defined categories or groups (unlabeled data). The goal of is to find distinct groups in the data based on inherent similarities between them rather than predetermined labels. In k-means, "k" represents the total number of unique groups the algorithm will create. Each example is assigned to one group or another based on similarity to other examples across a set of characteristics called features. K-means clustering is useful for business applications like customer segmentation, inventory categorization, and anomaly detection.

The most popular clustering algorithms are:

- k-Means

- k-Medians
- Expectation Maximisation (EM)
- Hierarchical Clustering

Association Rule Learning Algorithms

Association rule learning methods extract rules that best explain observed relationships between variables in data. These rules can discover important and commercially useful associations in large multidimensional datasets that can be exploited by an organization.

The most popular association rule learning algorithms are:

- Apriori algorithm
- Eclat algorithm

Artificial Neural Network Algorithms

Artificial neural networks (ANNs) are statistical models directly inspired by, and partially modeled on biological neural networks. The goal of artificial neural network machine learning algorithms is to mimic the way the human brain organizes and understands information in order to arrive at various predictions. In artificial neural networks, information is passed through an input layer, a hidden layer, and an output layer. Each layer can contain one or more neurons. The input and output layers can be comprised of raw features and predictions, respectively. The hidden layer in between consists of many highly interconnected neurons capable of complex meta-feature engineering. As the neural network “learns” the data, the connections between these neurons are fine-tuned until the network yields highly accurate predictions. This biological approach to computation allows neural networks to excel at some of the most challenging, high-dimensional problems in artificial intelligence, such as speech and object recognition, image segmentation, and natural language processing. Like random forests, neural networks are difficult — if not impossible — to interpret without the use of tools like , an open source model interpretation package[2] . This means that data scientists will often defer to simpler machine learning algorithms unless their analysis demands superior accuracy [20].

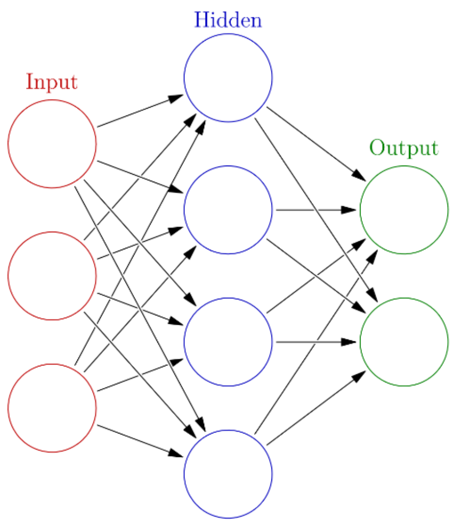


Figure 2. Artificial neural networks model

One thing worth noting is that while ANNs are extremely powerful, they can also be very complex and are considered black box algorithms, which means that their inner-workings are very difficult to understand and explain. Choosing whether to employ ANNs to solve problems should therefore be chosen with that in mind.

[2] *Section 3.3 describes in detail some of these open source model interpretation package*

The most popular artificial neural network algorithms are:

- Perceptron
- Back-Propagation
- Hopfield Network
- Radial Basis Function Network (RBFN)

Dimensionality Reduction Algorithms

Like clustering methods, dimensionality reduction seeks and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarize or describe data using less information.

This can be useful to visualize dimensional data or to simplify data which can then be used in a supervised learning method. Many of these methods can be adapted for use in classification and regression.

- Principal Component Analysis (PCA)
- Principal Component Regression (PCR)
- Partial Least Squares Regression (PLSR)
- Sammon Mapping
- Multidimensional Scaling (MDS)
- Projection Pursuit
- Linear Discriminant Analysis (LDA)
- Mixture Discriminant Analysis (MDA)
- Quadratic Discriminant Analysis (QDA)
- Flexible Discriminant Analysis (FDA)

Ensemble Algorithms

Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.

Much effort is put into what types of weak learners to combine and the ways in which to combine them. This is a very powerful class of techniques and as such is very popular.

- Boosting
- Bootstrapped Aggregation (Bagging)
- AdaBoost
- Stacked Generalization (blending)
- Gradient Boosting Machines (GBM)
- Gradient Boosted Regression Trees (GBRT)
- Random Forest

2.2.5. Machine Learning: Advanced learning methods

In this subsection, we introduce a few recent learning methods that may be either promising or much needed for solving the big data problems. The outstanding characteristic of these methods is to focus on the idea of learning, rather than just a single algorithm [22].

Representation Learning

Datasets with high-dimensional features have become increasingly common nowadays, which challenge the current learning algorithms to extract and organize the discriminative information from the data. Fortunately, representation learning [25], a promising solution to learn the meaningful and useful representations of the data that make it easier to extract useful information when building classifiers or other predictors, has been presented and achieved impressive performance on many dimensionality reduction tasks. Representation learning aims to achieve that a reasonably sized learned representation can capture a huge number of possible input configurations, which can greatly facilitate improvements in both computational efficiency and statistical efficiency.

There are mainly three subtopics on representation learning: feature selection, feature extraction, and distance metric learning. In order to give impetus to the multidomain learning ability of representation learning, automatic representation learning, biased representation learning, cross-domain representation learning, and some other related techniques have been proposed in recent years. The rapid increase in the scientific activity on representation learning has been accompanied and nourished by a remarkable string of empirical successes in real-world applications, such as speech recognition, natural language processing, and intelligent vehicle systems.

Deep learning

Nowadays, there is no doubt that deep learning is one of the hottest research trends in machine learning field. In contrast to most traditional learning techniques, which are considered using shallow-structured learning architectures, deep learning mainly uses supervised and/or unsupervised strategies in deep architectures to automatically learn hierarchical representations [26]. Deep architectures can often capture more complicated, hierarchically launched statistical patterns of inputs for achieving to be adaptive to new areas than traditional learning methods and often outperform state of the art achieved by hand-made features. Deep belief networks (DBNs) and convolutional neural networks (CNNs) are two mainstream deep learning approaches and research directions proposed over the past decade, which have been well established in the deep learning field and shown great promise for future work.

Due to the state-of-the-art performance of deep learning, it has attracted much attention from the academic community in recent years such as speech recognition, computer vision, language processing, and information retrieval [28]. As the data keeps getting bigger, deep learning is coming to play a pivotal role in providing predictive analytics solutions for large-scale data sets, particularly with the increased processing power and the advances in graphics processors. For example, IBM's brain-like computer and Microsoft's real-time language translation in Bing voice search have used techniques like deep learning to leverage big data for competitive advantage.

More details about deep learning are presented in section 2.3.

Distributed and parallel learning

There is often exciting information hidden in the unprecedented volumes of data. Learning from these massive data is expected to bring significant science and engineering advances which can facilitate the development of more intelligent systems. However, a bottleneck preventing such a big blessing is the inability of learning algorithms to use all the data to learn within a reasonable time. In this context, distributed learning seems to be a promising research since allocating the learning process among several workstations is a natural way of scaling up learning algorithms. Different from the classical learning framework, in which one requires the collection of that data in a database for central processing, in the framework of distributed learning, the learning is carried out in a distributed manner.

In the past years, several popular distributed machine learning algorithms have been proposed, including decision rules, stacked generalization, meta-learning, and distributed boosting. With the advantage of distributed computing for managing big volumes of data, distributed learning avoids the necessity of gathering data into a single workstation for central processing, saving time and energy. It is expected that more widespread applications of the distributed learning are on the way. Similar to distributed learning, another popular learning technique for scaling up traditional learning algorithms is parallel machine learning. With the power of multicore processors and cloud computing platforms, parallel and distributed computing systems have recently become widely accessible. A more detailed description about distributed and parallel learning can be found in [27].

Transfer learning

A major assumption in many traditional machine learning algorithms is that the training and test data are drawn from the same feature space and have the same distribution. However, with the data explosion from variety of sources, great heterogeneity of the collected data destroys the hypothesis. To tackle this issue, transfer learning has been proposed to allow the domains, tasks, and distributions to be different, which can extract knowledge from one or more source tasks and apply the knowledge to a target task [29]. The advantage of transfer learning is that it can intelligently apply knowledge learned previously to solve new problems faster.

Based on different situations between the source and target domains and tasks, transfer learning is categorized into three subsettings: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. In terms of inductive transfer learning, the source and target tasks are different, no matter when the source and target domains are the same or not. Transductive transfer learning, in contrast, the target domain is different from the source domain, while the source and target tasks are the same. Finally, in the unsupervised transfer learning setting, the target task is different from but related to the source task. Furthermore, approaches to transfer learning in the above three different settings can be classified into four contexts based on “What to transfer,” such as the instance transfer approach, the feature representation transfer approach, the parameter transfer approach, and the relational knowledge transfer approach. Recently, transfer learning techniques have been applied successfully in many real-world data processing applications, such as cross-domain text classification, constructing informative priors, and large-scale document classification.

Active learning

In many real-world applications, we have to face such a situation: data may be abundant but labels are scarce or expensive to obtain. Frequently, learning from massive amounts of unlabeled data is difficult and time-consuming. Active learning attempts to address this issue by selecting a subset of most critical instances for labeling [30]. In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data. It can obtain satisfactory classification performance with fewer labeled samples via query strategies than those of conventional passive learning.

There are three main active learning scenarios, comprising membership query synthesis, stream-based selective sampling and pool-based sampling. Popular active learning approaches can be found in. They have been studied extensively in the field of machine learning and applied to many data processing problems such as image classification and biological DNA identification.

Kernel-based learning

Over the last decade, kernel-based learning has established itself as a very powerful technique to increase the computational capability based on a breakthrough in the design of efficient nonlinear learning algorithms [31]. The outstanding advantage of kernel methods is their elegant property of implicitly mapping samples from the original space into a potentially infinite-dimensional feature space, in which inner products can be calculated directly via a kernel function.

The data in the input space can be projected onto different feature spaces with different mappings. The diversity of feature spaces gives us more choices to gain better performance, while in practice, the choice itself of a proper mapping for any given real-world problem may generally be nontrivial. Fortunately, the kernel trick provides an elegant mathematical means to construct powerful nonlinear variants of most well-known statistical linear techniques, without knowing the mapping explicitly.

2.2.6. Machine Learning Software

The application of machine learning to diverse areas of computing is gaining popularity rapidly, not only because of cheap and powerful hardware, but also because of the increasing availability of free and open source software, which enable machine learning to be implemented easily. Machine learning practitioners and researchers, being a part of the software engineering team, continuously build sophisticated products, integrating intelligent algorithms with the final product to make software work more reliably, quickly and without hassles.

There is a wide range of open source machine learning frameworks available in the market, which enable machine learning engineers to build, implement and maintain machine learning systems, generate new projects and create new impactful machine learning systems. Below is a list of frameworks for machine learning engineers:

- **Apache Singa** is a general distributed deep learning platform for training big deep learning models over large datasets. It is designed with an intuitive programming model based on the layer abstraction. A variety of popular deep learning models are supported, namely feed-forward models including convolutional neural networks (CNN), energy models like restricted Boltzmann machine (RBM), and recurrent neural networks (RNN). Many built-in layers are provided for users.

- **Amazon Machine Learning** is a service that makes it easy for developers of all skill levels to use machine learning technology. Amazon Machine Learning provides visualization tools and wizards that guide you through the process of creating machine learning (ML) models without having to learn complex ML algorithms and technology. It connects to data stored in Amazon S3, Redshift, or RDS, and can run binary classification, multiclass categorization, or regression on said data to create a model (Website: https://aws.amazon.com/machine-learning/). Its key features are:

    - Supports multiple data sources within its system.
    - Allows users to create a data source object from data residing in Amazon Redshift – the data warehouse Platform as a Service.
    - Allows users to create a data source object from data stored in the MySQL database.
    - Supports three types of models: binary classification, multi-class classification and regression.

- **Azure ML Studio** allows Microsoft Azure users to create and train models, then turn them into APIs that can be consumed by other services. Users get up to 10GB of storage per account for model data, although you can also connect your own Azure storage to the service for larger models. A wide range of algorithms are available, courtesy of both Microsoft and third parties. You don't even need an account to try out the service; you can log in anonymously and use Azure ML Studio for up to eight hours.

- **Apache Mahout** Apache Mahout, being a free and open source project of the Apache Software Foundation, has a goal to develop free distributed or scalable machine learning algorithms for diverse areas like collaborative filtering, clustering and classification. Mahout provides Java libraries and Java collections for various kinds of mathematical operations.
Apache Mahout is implemented on top of Apache Hadoop using the MapReduce paradigm. Once Big Data is stored on the Hadoop Distributed File System (HDFS), Mahout provides the data science tools to automatically find meaningful patterns in these Big Data sets, turning this into 'big information' quickly and easily.

    - Building a recommendation engine: Mahout provides tools for building a recommendation engine via the Taste library– a fast and flexible engine for CF.
    - Clustering with Mahout: Several clustering algorithms are supported by Mahout, like Canopy, k-Means, Mean-Shift, Dirichlet, etc.
    - Categorizing content with Mahout: Mahout uses the simple Map-Reduce-enabled naïve Bayes classifier.

- **Caffe** is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center and by community contributors. created the project during his PhD at UC Berkeley. Caffe is released under the . Models and optimization are defined by configuration without hard-coding & user can switch between CPU and GPU. Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process over 60M images per day with a single NVIDIA K40 GPU.

- **H2O** makes it possible for anyone to easily apply math and predictive analytics to solve today's most challenging business problems. It intelligently combines unique features not currently found in other machine learning platforms including: Best of Breed Open Source Technology, Easy-to-use WebUI and Familiar Interfaces, Data Agnostic Support for all Common Database and File Types. With H2O, you can work with your existing languages and tools. Further, you can extend the platform seamlessly into your Hadoop environments.

- **Massive Online Analysis (MOA)** is the most popular open source framework for data stream mining, with a very active growing community. It includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, concept drift detection and recommender systems) and tools for evaluation. Related to the WEKA project, MOA is also written in Java, while scaling to more demanding problems.

- **MLlib (Spark)** is Apache Spark's machine learning library. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.

- **Mlpack**, a C++-based machine learning library originally rolled out in 2011 and designed for "scalability, speed, and ease-of-use," according to the library's creators. Implementing mlpack can be done through a cache of command-line executables for quick-and-dirty, "black box" operations, or with a C++ API for more sophisticated work. Mlpack provides these algorithms as simple command-line programs and C++ classes which can then be integrated into larger-scale machine learning solutions.

- **Pattern** is a web mining module for the Python programming language. It has tools for data mining (Google, Twitter and Wikipedia API, a web crawler, a HTML DOM parser), natural language processing (part-of-speech taggers, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, clustering, SVM), network analysis and <canvas> visualization.

- **Scikit-Learn** leverages Python's breadth by building on top of several existing Python packages — NumPy, SciPy, and matplotlib — for math and science work. The resulting libraries can be used either for interactive "workbench" applications or be embedded into other software and reused. The kit is available under a BSD license, so it's fully open and reusable. Scikit-learn includes tools for many of the standard (such as clustering, classification, regression, etc.). And since scikit-learn is developed by a large community of developers and machine-learning experts, promising new techniques tend to be included in fairly short order.

- **Shogun** is among the oldest, most venerable of machine learning libraries Shogun was initiated by Soeren Sonnenburg and Gunnar Raetsch in 1999 and is currently under rapid development by a large team of programmers. This free and open source toolbox written in C++ provides algorithms and data structures for machine learning problems. Shogun Toolbox provides the use of a toolbox via a unified interface from C++, Python, Octave, R, Java, Lua and C++; and can run on Windows, Linux and even MacOS. Shogun is designed for unified large-scale learning for a broad range of feature types and learning settings, like classification, regression, dimensionality reduction, clustering, etc. It contains a number of exclusive state-of-art algorithms, such as a wealth of efficient SVM implementations, multiple kernel learning, kernel hypothesis testing, Krylov methods, etc. Shogun supports bindings to other machine learning libraries like LibSVM, LibLinear, SVMLight, LibOCAS, libqp, VowpalWabbit, Tapkee, SLEP, GPML and many more. Its features include one-time classification, multi-class classification, regression, structured output learning, pre-processing, built-in model selection strategies, visualization and test frameworks; and semi-supervised, multi-task and large-scale learning. The latest version is 4.1.0. available on**:** *http://www.shogun-toolbox.org/*

- **TensorFlow** is an open source software library for numerical computation using data flow graphs. TensorFlow implements what are called data flow graphs, where batches of data ("tensors") can be processed by a series of algorithms described by a graph. The movements of the data through the system are called "flows" — hence, the name. Graphs can be assembled with C++ or Python and can be processed on CPUs or GPUs.

- **Theano** is a Python library that lets you to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (numpy.ndarray). Using Theano it is possible to attain speeds rivaling hand-crafted C implementations for problems involving large amounts of data. It was written at the lab to support rapid development of efficient machine learning algorithms. Theano is released under a BSD license.

- **Torch** is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation. The goal of Torch is to have maximum flexibility and speed in building your scientific algorithms while making the process extremely simple. Torch comes with a in machine learning, computer vision, signal processing, parallel processing, image, video, audio and networking among others, and builds on top of the Lua community.

- **Veles** is a distributed platform for deep-learning applications, and it's written in C++, although it uses Python to perform automation and coordination between nodes. Datasets can be analyzed and automatically normalized before being fed to the cluster, and a REST API allows the trained model to be used in production immediately. It focuses on performance and flexibility. It has little hard-coded entities and enables training of all the widely recognized topologies, such as fully connected nets, convolutional nets, recurrent nets etc.

- **Oryx 2** is a realization of Lambda architecture built on Apache Spark and Apache Kafka for real-time large-scale machine learning. It is designed for building applications and includes packaged, end-to-end applications for collaborative filtering, classification, regression and clustering. Oryx 2 comprises the following three tiers.

    - General Lambda architecture tier: Provides batch, speed and serving layers, which are not specific to machine learning.
    - Specialization on top which, in turn, provides machine learning abstraction to hyperparameter selection, etc.
    - End-to-end implementation of the same standard machine learning algorithms as an application (ALS, random decision forests, k-means) on top.

  Oryx 2 consists of the following layers of Lambda architecture as well as connecting elements.

    - Batch layer: Used for computing new results from historical data and previous results.
    - Speed layer: Produces and publishes incremental model updates from a stream of new data.
    - Serving layer: Receives models and updates, and implements a synchronous API, exposing query operations on results.
    - Data transport layer: Moves data between layers and takes input from external sources.

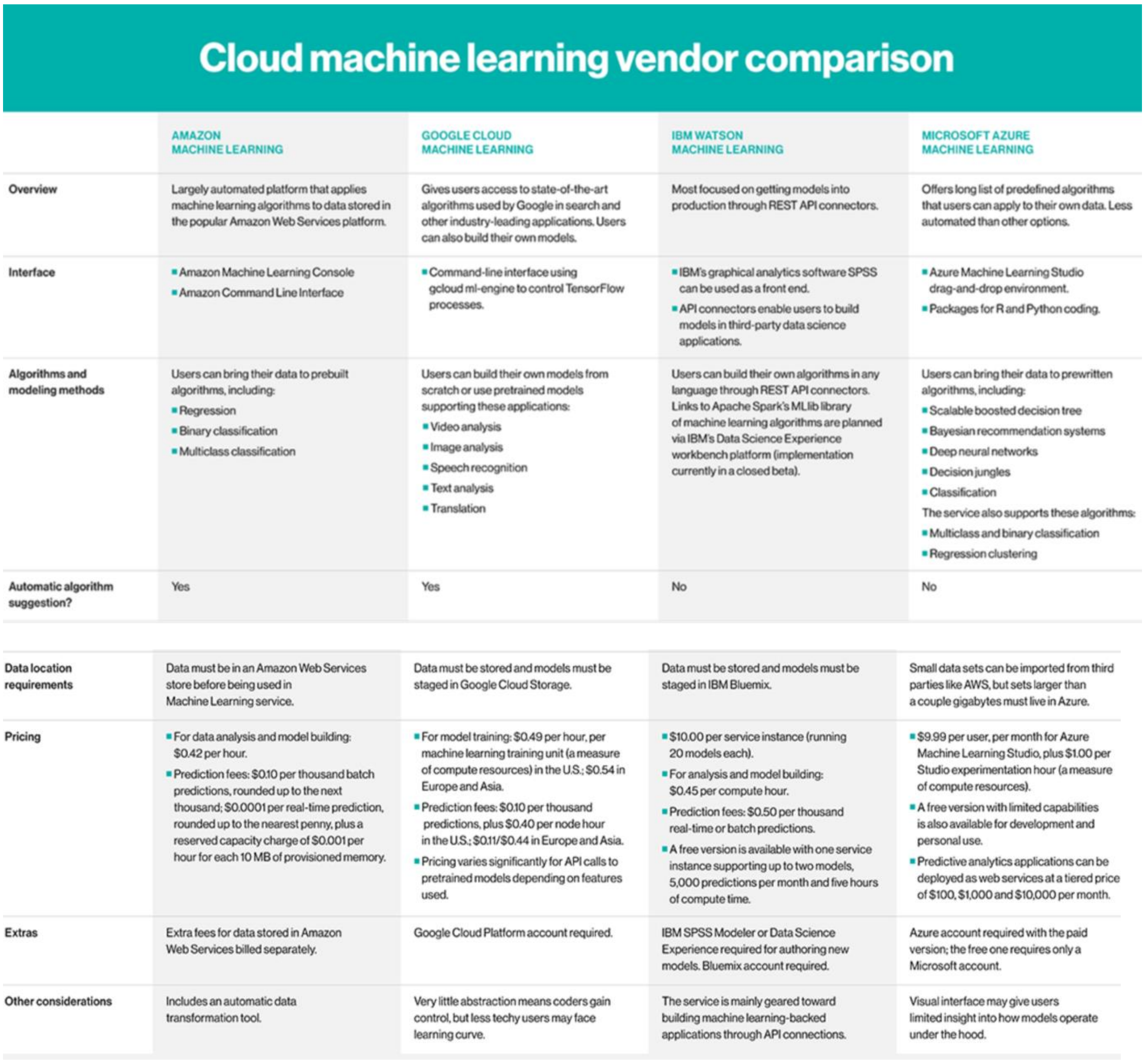
- **Weka** is a rich suite of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from a Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka has also an Experimenter interface to benchmark machine learning model performances in a statistically significant manner, as well. It's also possible to use Weka for Big Data. Weka is an open source software issued under the GNU General Public License. Weka website is https://www.cs.waikato.ac.nz/ml/weka/ Free Weka online courses can be accessed on following URLs:

- https://www.cs.waikato.ac.nz/ml/weka/
- https://www.youtube.com/user/WekaMOOC

2.2.7. Machine learning platforms comparison

Several vendors have beefed up their offerings in recent months and now offer simple, cloud-based platforms for and developing models that can quickly be put into production. But these machine learning platforms all come with their own downsides. There is a significant risk of vendor lock-in with each. They generally require users to bring their data to the broader cloud platform Once all of an enterprise's data is located in one vendor's cloud, it's difficult for that business to use another vendor's services or to use open source tools for other tasks. Table 2 present a comparison of some Cloud Machine learning vendor [32].

Table 2. Cloud Machine Learning Vendor Comparison



Cloud machine learning vendor comparison

	AMAZON MACHINE LEARNING	GOOGLE CLOUD MACHINE LEARNING	IBM WATSON MACHINE LEARNING	MICROSOFT AZURE MACHINE LEARNING
Overview	Largely automated platform that applies machine learning algorithms to data stored in the popular Amazon Web Services platform.	Gives users access to state-of-the-art algorithms used by Google in search and other industry-leading applications. Users can also build their own models.	Most focused on getting models into production through REST API connectors.	Offers long list of predefined algorithms that users can apply to their own data. Less automated than other options.
Interface	▪ Amazon Machine Learning Console ▪ Amazon Command Line Interface	▪ Command-line interface using gcloud ml-engine to control TensorFlow processes.	▪ IBM's graphical analytics software SPSS can be used as a front end. ▪ API connectors enable users to build models in third-party data science applications.	▪ Azure Machine Learning Studio drag-and-drop environment. ▪ Packages for R and Python coding.
Algorithms and modeling methods	Users can bring their data to prebuilt algorithms, including: ▪ Regression ▪ Binary classification ▪ Multiclass classification	Users can build their own models from scratch or use pretrained models supporting these applications: ▪ Video analysis ▪ Image analysis ▪ Speech recognition ▪ Text analysis ▪ Translation	Users can build their own algorithms in any language through REST API connectors. Links to Apache Spark's MLlib library of machine learning algorithms are planned via IBM's Data Science Experience workbench platform (implementation currently in a closed beta).	Users can bring their data to prewritten algorithms, including: ▪ Scalable boosted decision tree ▪ Bayesian recommendation systems ▪ Deep neural networks ▪ Decision jungles ▪ Classification The service also supports these algorithms: ▪ Multiclass and binary classification ▪ Regression clustering
Automatic algorithm suggestion?	Yes	Yes	No	No
Data location requirements	Data must be in an Amazon Web Services store before being used in Machine Learning service.	Data must be stored and models must be staged in Google Cloud Storage.	Data must be stored and models must be staged in IBM Bluemix.	Small data sets can be imported from third parties like AWS, but sets larger than a couple gigabytes must live in Azure.
Pricing	▪ For data analysis and model building: $0.42 per hour. ▪ Prediction fees: $0.10 per thousand batch predictions, rounded up to the next thousand; $0.0001 per real-time prediction, rounded up to the nearest penny, plus a reserved capacity charge of $0.001 per hour for each 10 MB of provisioned memory.	▪ For model training: $0.49 per hour, per machine learning training unit (a measure of compute resources) in the U.S.; $0.54 in Europe and Asia. ▪ Prediction fees: $0.10 per thousand predictions, plus $0.40 per node hour in the U.S.; $0.11/$0.44 in Europe and Asia. ▪ Pricing varies significantly for API calls to pretrained models depending on features used.	▪ $10.00 per service instance (running 20 models each). ▪ For analysis and model building: $0.45 per compute hour. ▪ Prediction fees: $0.50 per thousand real-time or batch predictions. ▪ A free version is available with one service instance supporting up to two models, 5,000 predictions per month and five hours of compute time.	▪ $9.99 per user, per month for Azure Machine Learning Studio, plus $1.00 per Studio experimentation hour (a measure of compute resources). ▪ A free version with limited capabilities is also available for development and personal use. ▪ Predictive analytics applications can be deployed as web services at a tiered price of $100, $1,000 and $10,000 per month.
Extras	Extra fees for data stored in Amazon Web Services billed separately.	Google Cloud Platform account required.	IBM SPSS Modeler or Data Science Experience required for authoring new models. Bluemix account required.	Azure account required with the paid version; the free one requires only a Microsoft account.
Other considerations	Includes an automatic data transformation tool.	Very little abstraction means coders gain control, but less techy users may face learning curve.	The service is mainly geared toward building machine learning-backed applications through API connections.	Visual interface may give users limited insight into how models operate under the hood.

2.3. Deep learning

2.3.1. An introduction to Deep learning

Deep learning is the fastest growing field and the new big trend in machine learning. It can revolutionize the way we see Artificial Intelligence.

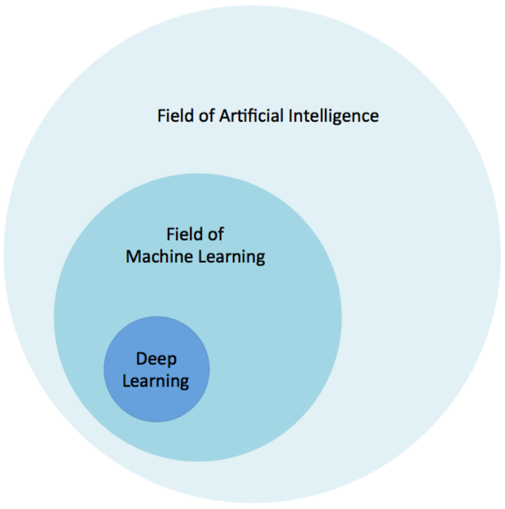


Figure 3. The relationship between AI and deep learning

Deep learning, while sounding flashy, is really just a term to describe certain types of neural networks and related algorithms that consume often very raw input data. They process this data through many layers of nonlinear transformations of the input data in order to calculate a target output.

Deep learning has been used successfully in many applications and is considered to be one of the most cutting-edge machine learning and AI techniques at the time of this writing. The associated algorithms are often used for *supervised*, *unsupervised*, and *semi-supervised* learning problems.

For neural network-based deep learning models, the number of layers is greater than in so-called *shallow learning* algorithms. Shallow algorithms tend to be less complex and require more up-front knowledge of optimal features to use, which typically involves feature selection and engineering.

Following are some of the facets in this evolution of neural networks:

- More neurons than previous networks
- More complex ways of connecting layers/neurons in NNs
- Explosion in the amount of computing power available to train
- Automatic feature extraction

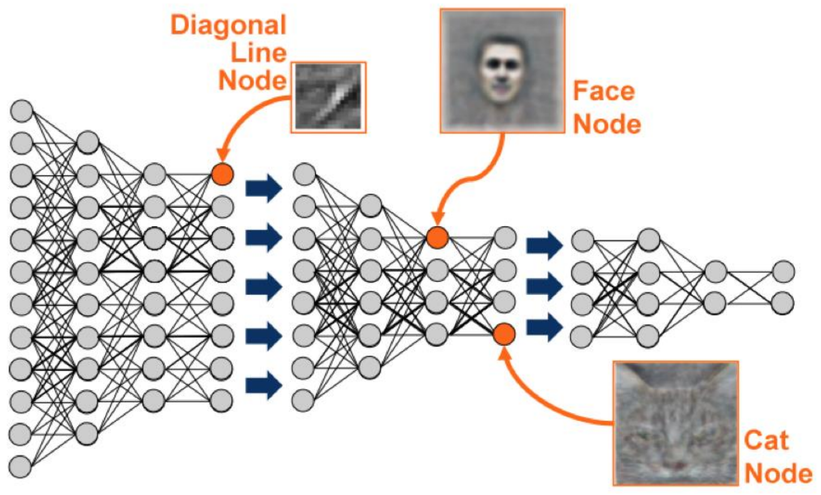


Figure 4. Neural network-based deep learning models [33]

In addition to statistical techniques, neural networks and deep learning leverage concepts and techniques from signal processing as well, including nonlinear processing and/or transformations.

Many phenomena observed in the physical universe are actually best modeled with nonlinear transformations. This is true as well for transformations between inputs and the target output in machine learning and AI solutions.

As mentioned, input data is transformed throughout the layers of a deep learning neural network by artificial neurons or processing units. The chain of transformations that occur from input to output is known as the credit assignment path, or CAP.

2.3.2. Deep Learning Algorithms

There are many different deep-learning model architectures and learning algorithms, some of the more notable ones include:

- Feed-forward neural networks
- Recurrent neural network
- Multi-layer perceptrons (MLP)
- Convolutional neural networks
- Recursive neural networks
- Deep belief networks
- Convolutional deep belief networks
- Self-Organizing Maps
- Deep Boltzmann machines
- Stacked de-noising auto-encoders

It's worth pointing out that due to the relative increase in complexity, deep learning and neural network algorithms can be prone to overfitting. In addition, increased model and algorithmic complexity can result in very significant computational resource and time requirements.

It's also important to consider that solutions may represent local minima as opposed to a global optimal solution. This is due to the complex nature of these models when combined with optimization techniques such as gradient descent.

Given all of this, proper care must be taken when leveraging artificial intelligence algorithms to solve problems, including the selection, implementation, and performance assessment of algorithms themselves.

2.3.3. Comparison of deep learning software

The following table compares some of the most popular software frameworks, libraries and computer programs for deep learning. (Source Wikipedia).

Table 3. Deep learning software

Software	Creator	Software license	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++	Wolfram Language	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, macOS, Windows, Android, iOS	C, Lua	Lua, LuaJIT, C, utility library for C++/OpenCL	Yes	Third party implementations	Yes	Through Twitter's Autograd	Yes	Yes	Yes	Yes	Yes
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development	Yes	Yes	Through Lasagne's model zoo	Yes	Yes	Yes	Yes
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, macOS, Windows	C++, Python	Python (Keras), C/C++, Java, Go, R	No	On roadmapbut already with SYCL support	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PaddlePaddle	Baidu PaddlePaddle team	Apache 2.0	Yes	Linux, macOS, Android, Raspberry Pi	C++, Go	C/C++, Python	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?
Neural Designer	Artelnics	Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
MXNet	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, macOS, Windows, AWS, Android, iOS, JavaScript	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Microsoft Cognitive Toolkit	Microsoft Research	MIT license	Yes	Windows, Linux (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line, BrainScript (.NET on roadmap)	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes
MATLAB + Neural Network Toolbox	MathWorks	Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	Train with Parallel Computing Toolbox & generate CUDA code with GPU Coder	No	Yes	Yes	Yes	No	With Parallel Computing Toolbox
MatConvNet	Andrea Vedaldi, Karel Lenc	BSD license	Yes	Windows, Linux (macOS via Docker on roadmap)	C++	MATLAB, C++,	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Keras	François Chollet	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano or MXNet as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	On roadmap	Yes	Computational Graph	Yes	Yes	Yes	Yes	Yes
Caffe2	Facebook	Apache 2.0	Yes	Linux, macOS, Windows	C++, Python	Python, MATLAB	Yes	Under development	Yes	Yes	Yes	Yes	Yes	No	Yes
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, macOS, Windows	C++	Python, MATLAB	Yes	Under development	Yes	Yes	Yes	Yes	Yes	No	?
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java	No	Yes	Yes	?	Yes	Yes	Yes	Yes	Yes

2.3.4. Other Deep Learning tools

- **Adnn** – Javascript neural networks (*https://github.com/dritchie/adnn*)
- **Blocks** – Theano framework for building and training neural networks (*https://github.com/mila-udem/blocks*)
- **Caffe2** – Deep learning framework built on Caffe, developed by Facebook in cooperation with NVIDIA, Qualcomm, Intel, Amazon, and Microsoft. (*https://caffe2.ai/)*
- **CaffeOnSpark** – Scalable deep learning package running Caffe on Spark and Hadoop clusters with peer-to-peer communication *(https://github.com/yahoo/CaffeOnSpark)*
- **CNNLab** – Deep learning framework using GPU and FPGA-based accelerators *(https://arxiv.org/abs/1606.06234)*
- **ConvNetJS** – Javascript library for training deep learning models entirely in a web browser (*https://cs.stanford.edu/people/karpathy/convnetjs/)*
- **Cortex** – Theano-based deep learning toolbox for neuroimaging (*https://github.com/rdevon/cortex)*
- **cuDNN** – Optimized deep learning computation primitives implemented in CUDA *(https://developer.nvidia.com/cudnn)*
- **CURRENNT** – CUDA-accelerated toolkit for deep Long Short-Term Memory (LSTM) RNN architectures supporting large data sets not fitting into main memory. *(https://sourceforge.net/projects/currennt/)*
- **Darknet** – Darknet is an open source neural network framework written in C and CUDA, and supports CPU and GPU computation. *(https://pjreddie.com/darknet/)*
- **DeepCL** – OpenCL library to train deep convolutional networks, with APIs for C++, Python and the command line *(https://github.com/hughperkins/DeepCL)*
- **DeepLearningKit** – Open source deep learning framework for iOS, OS X and tvOS *(https://memkite.com/deeplearningkit/)*
- **DeepLearnToolbox** – Matlab/Octave toolbox for deep learning *(https://github.com/rasmusbergpalm/DeepLearnToolbox)*
- **DeepX** – Software accelerator for deep learning execution aimed towards mobile devices *(http://niclane.org/pubs/deepx_ipsn.pdf)*
- **deepy** – Extensible deep learning framework based on Theano *(https://github.com/zomux/deepy)*
- **DSSTNE** (Deep Scalable Sparse Tensor Network Engine) – Amazon developed library for building deep learning models (*https://github.com/amzn/amazon-dsstne*)
- **Faster RNNLM (HS/NCE) toolkit** – An rnnlm implementation for training on huge datasets and very large vocabularies and usage in real-world ASR and MT problems (*https://github.com/yandex/faster-rnnlm*)
- **GNU Gneural Network** – GNU package which implements a programmable neural network (*https://www.gnu.org/software/gneuralnetwork/)*
- **IDLF** – Intel® Deep Learning Framework; supports OpenCL (*https://github.com/01org/idlf*)
- **Intel Math Kernel Library** (Intel MKL), library of optimized math routines, including optimized deep learning computation primitives (*https://en.wikipedia.org/wiki/Math_Kernel_Library*)
- **Keras** – Deep Learning library for Theano and TensorFlow (*https://keras.io/)*
- **Lasagne** – Lightweight library to build and train neural networks in Theano (*http://lasagne.readthedocs.io/en/latest/)*

- **Leaf** – "The Hacker's Machine Learning Engine"; supports OpenCL (official development suspended) (*https://github.com/autumnai/leaf*)
- **LightNet** – MATLAB-based environment for deep learning (*https://arxiv.org/abs/1605.02766*)
- **MatConvNet** – CNNs for MATLAB (*http://www.vlfeat.org/matconvnet/)*
- **MaTEx** – Distributed TensorFlow with MPI by PNNL (*https://github.com/matex-org/matex*)
- **Mocha** – Deep learning framework for Julia, inspired by Caffe (*https://github.com/pluskid/Mocha.jl*)
- **neon** – Nervana's Python based Deep Learning framework (*https://github.com/NervanaSystems/neon*)
- **Neural Network Toolbox** – MATLAB toolbox for neural network creation, training and simulation *(https://fr.mathworks.com/products/neural-network.html)*
- **PaddlePaddle** – "PArallel Distributed Deep LEarning", deep learning platform (*https://github.com/PaddlePaddle/paddle*)
- **Purine** – Bi-graph based deep learning framework (*https://github.com/purine/purine2*)
- **Pylearn2** – Machine learning library mainly built on top of Theano (*http://deeplearning.net/software/pylearn2/*)
- **Pytorch** - Python based implementation of Torch API, allows for dynamic graph construction *(https://pytorch.org/)*
- **scikit-neuralnetwork** – Multi-layer perceptrons as a wrapper for Pylearn2 (*https://scikit-neuralnetwork.readthedocs.io/en/latest/*)
- **sklearn-theano** – Scikit-learn compatible tools using Theano (*https://github.com/sklearn-theano/sklearn-theano*)
- **Tensor Builder** – Lightweight extensible library for easy creation of deep neural networks using functions from "any Tensor-based library" (requires TensorFlow) through an API based on the Builder Pattern (https://github.com/cgarciae/tensorbuilder)
- **TensorGraph** – Framework for building any models based on TensorFlow *(https://github.com/hycis/TensorGraphX*)
- **TensorFire** – Neural networks framework for the web browser, accelerated by WebGL *(https://tenso.rs/)*
- **TF Learn** (Scikit Flow) – Simplified interface for TensorFlow (*https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/learn/python/learn*)
- **TF-Slim** – High level library to define complex models in TensorFlow *(https://ai.googleblog.com/2016/08/tf-slim-high-level-library-to-define.html*)
- **TFLearn** – Deep learning library featuring a higher-level API for TensorFlow (*http://tflearn.org/)*
- **Theano-Lights** – Deep learning research framework based on Theano (*https://github.com/Ivaylo-Popov/Theano-Lights*)
- **tiny-dnn** – Header only, dependency-free deep learning framework in C++11 (*https://github.com/nyanp/tiny-dnn*)
- **torchnet** – Torch framework providing a set of abstractions aiming at encouraging code re-use as well as encouraging modular programming (*https://github.com/torchnet/torchnet*)
- **Veles** – Distributed machine learning platform by Samsung (*https://github.com/Samsung/veles*)

3. MEASURE analysis algorithms

3.1. Constraints based analysis (QUALITY GUARD)

This paragraph describes the Quality Guard Tool, the Rule Based Analysis Tool developed by Softeam within the MEASURE Project.
The Quality Guard Tool is a whole web application which allows to provide advanced data analysis functionalities to data collected by the measure platform including advanced dashboard cards.

The tool is meant to help everyday work of Quality Engineers and Project Managers. In particular thanks to the tool, *QA engineers* will be able to define constraints based on measure thresholds related to a specific project and *Project Managers* will be able to verify the state of each constraint periodically or to get the incidents history of all constraints defined by the Quality Guard Tool.

Within the MEASURE Quality Guard Tool, a *quality rule* is defined on a Measure Project to check that a numeric measure stays on delimited range. A Guard Condition defines a binary condition to compare the measure field value with threshold values and a Violation defines a period during which one of the condition defined in the Quality Guard is not respected.
In the Quality Guard Tool, conditions and constraints can be based on simple or cross measures expressions and the expression language support the logic operators: “AND”, “OR” and the comparison operators: “>”,”<”.
More in detail, each guard is defined by a quality guard name field, a description field, a measure instance and measure field name, a guard operator field (Superior or Inferior), a warning value field and an error value field.
Constraint violations are evaluated by the tool once a new measurement is collected by the platform and periodically. i.e. average values of measurements collected in a specific interval.

The Quality Guard Tool web application is integrated to the platform as component as described in D4.1. The MEASURE platform provides the REST API which allow the Quality Guard analysis tool to access to the platform data and the analysis tools provide specific web pages which are directly embedded to the Measure Platform web application.
The main view provided by the Quality Guard Analysis tool allows to visualize the state of each constraints defined by the tool. For each constraint, the Quality Guard Analysis tool allows to visualize a history of the last incidents. The tool can also show Dashboard card which lists the last constraints violations that occurred in the monitored project. The dashboard cards provided by the Quality Guard Analysis tool allow to get an overview of the state of either the quality guards or the constraints violations occurred in each project. Screenshots and usage details of the tool are provided in the deliverable D4.1.

3.2. Rule based correlation (MINT)

To improve software quality, it is necessary to introduce new metrics with the required detail and increased expressive power, in order to provide valuable information to the different actors of software development. For this reason, we define an approach based on metrics that contribute to improve software quality development. This approach focuses on the combination, reuse and correlation of metrics. It suggests to the user indications of how to reuse metrics and provide recommendations after the application of metrics correlation. The approach has been implemented in a tool called MINT. The main idea if the MINT approach is to identify and design correlations between metrics that contribute to the improvement of the development process and help developers to take decisions about it. The proposed correlations cover all aspects of the system like functional behavior, security, green computing and timing. For instance, we have defined correlations covering different phases of development. Also, correlation of two metrics from the same development phase or from different phases, this last to calculate the same metric at different times. Techniques to correlate metrics are provided and recommendations are given as an outcome to the developer and project manager. Recommendations will affect their actions and decisions.

MINT (Metrics Intelligence Tool) is a software solution designed to correlate metrics from different software development life cycle in order to provide valuable recommendations to different actors impacting the software development process. MINT considers the different measurements collected by the MEASURE platform as events occurring at runtime. The correlation is designed as extended finite state machines (EFSMs) allowing to perform Complex Event Processing (CEP) [16] in order to determine the possible actions that can be taken to improve the diverse stages of the software life cycle and thus the global software quality and cost.

3.2.1. Background

a) Metrics correlation: The correlation can be defined as a mutual relationship or association between metrics (or the values of its application). Metrics correlation can be the basis for the reuse of metrics; it can help to predict one value from another; it can indicate a causal relation between metrics and can establish relations between different metrics and increase the ability to measure. Examples of correlation are: to correlate two metrics from the same development phase; to correlate the same metric at different times; to correlate a metric (a set of metrics) from phase X regarding metrics of phase Y. As an outcome, recommendations and a selection of metrics will be proposed to the developer to improve the software development. MINT is based on correlation techniques.

b) Complex Events Processing: Complex event processing (CEP) technology addresses exactly the need of matching continuously incoming events against a pattern. Input events from data streams are processed immediately and if an event sequence is matching a pattern, the result is emitted straight away. CEP works very efficiently and in real-time, as there are no overheads for data storing. CEP is used in many areas that include for instance manufacturing processes, ICT security, etc. and is adapted in this work for software quality assessment process.

c) Extended Finite State Machine: In order to formally model the correlation process, the Extended Finite State Machine (EFSM) formalism is used. This formal description allows to represent the correlation between metrics as well as the constraints and computations needed to retrieve a meaningful recommendation related to software quality assessment.

Definition 1. An Extended Finite State Machine M is a 6-tuple M = < S; s_0; I;O; $\vec{x}$; Tr > where S is a finite set of states, s_0 is the initial state, I is a finite set of input symbols (eventually with parameters), O is a finite set of output symbols (eventually with parameters), $\vec{x}$ is a vector denoting a finite set of variables, and Tr is a finite set of transitions. A transition tr is a 6-tuple tr = < s_i; s_f ; i; o; P;A > where s_i and sf are the initial and final state of the transition, i and o are the input and the output, P is the predicate (a boolean expression), and A is an ordered set (sequence) of actions.

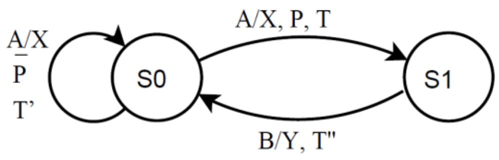


Figure 5. Example of a simple EFSM with two states.

We illustrate the notion of EFSM through a simple example described in Fig. 5. The ESFM is composed of two states S0, S1 and three transitions that are labeled with two inputs A and B, two outputs X and Y, one predicate P and three tasks T, T', and T". The EFSM operates as follows: starting from state S0, when the input A occurs, the predicate P is tested. If the condition holds, the machine performs the task T, triggers the output X and passes to state S1. If P is not satisfied, the same output X is triggered but the action T0 is performed and the state loops on itself. Once the machine is in state S0, it can come back to state S1 if receiving input B. If so, task T" is performed and output Y is triggered.

3.2.2. Writing correlation processes

a. Correlation process inputs and outputs:

The basic idea behind the MINT approach is to specify a set of correlation rules based on the knowledge of an expert of the software development process. These rules can rely on one or different sets of metrics (seen as inputs) and allow to provide different recommendations (seen as outputs) to different kinds of actors:

- Actors from the DevOps team: Analysts, designers, modelers, architects, developers, testers, operators, security experts, etc.
- Actors from the management plan: product manager, project manager, responsible of human resources, responsible of financial issues etc.

The automatic generation of such rules or their continuous refinement based on some artificial intelligence techniques is an ongoing work.

b. Example of correlation processes:

The correlation processes rely on different measurements that are computed and collected by external tools. Some examples of correlations are presented in the Figure 6.

Software Modularity

The assessment of the software modularity relies on two metrics provided by the SonarQube tool that are the class complexity and the maintainability rating. The class complexity measure (also called cognitive complexity) computes the cognitive weight of a Java Architecture. The cognitive weight represents the complexity of a code architecture in terms of maintainability and code understanding. The maintainability rating is the ratio of time (according to the total time to develop the software) needed to update or modify the software.

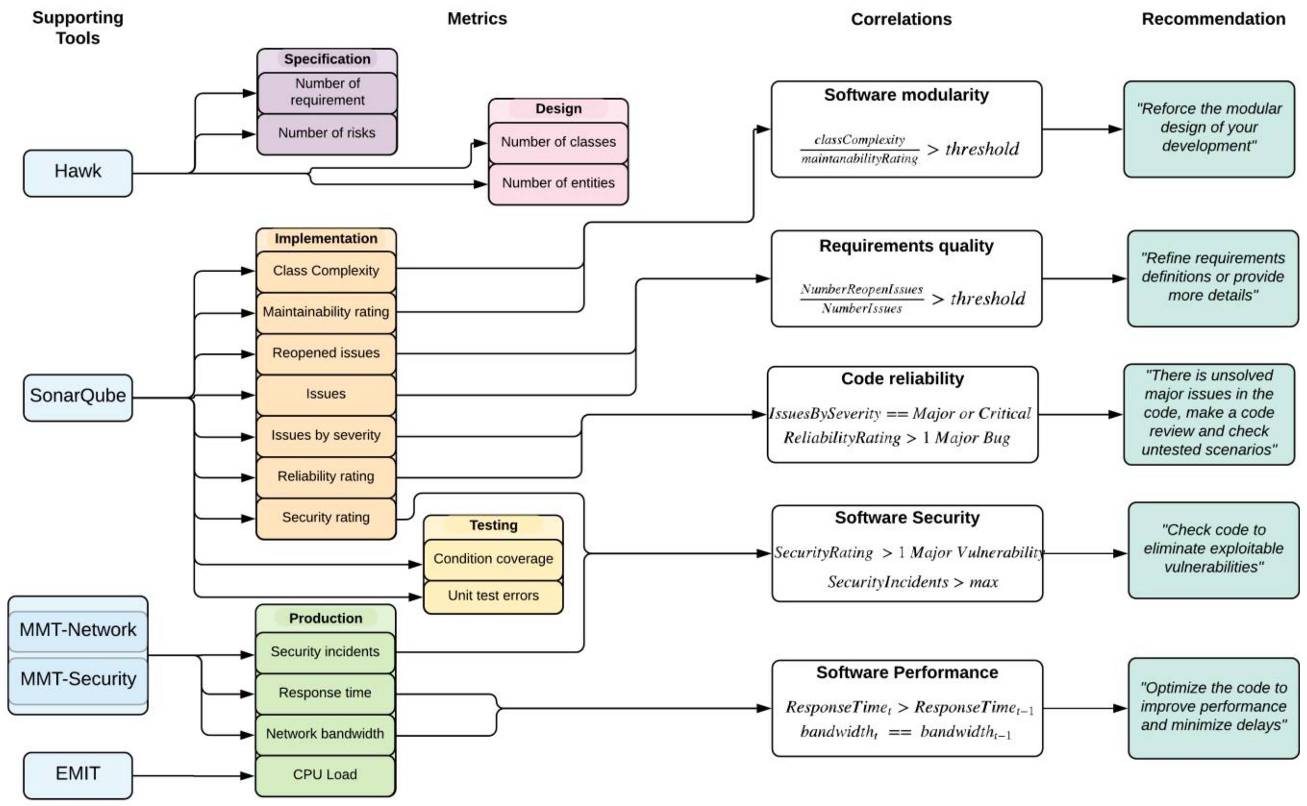

Supporting Tools
Metrics
Correlations
Recommendation
Specification
Number of requirement
Number of risks
Design
Number of classes
Number of entities
Hawk
Implementation
Class Complexity
Maintainability rating
Reopened issues
Issues
Issues by severity
Reliability rating
Security rating
SonarQube
Testing
Condition coverage
Unit test errors
MMT-Network
MMT-Security
Production
Security incidents
Response time
Network bandwidth
CPU Load
EMIT
Software modularity
classComplexity / maintanabilityRating > threshold
Requirements quality
NumberReopenIssues / NumberIssues > threshold
Code reliability
IssuesBySeverity == Major or Critical
ReliabilityRating > 1 Major Bug
Software Security
SecurityRating > 1 Major Vulnerability
SecurityIncidents > max
Software Performance
ResponseTime_t > ResponseTime_t−1
bandwidth_t == bandwidth_t−1
"Reforce the modular design of your development"
"Refine requirements definitions or provide more details"
"There is unsolved major issues in the code, make a code review and check untested scenarios"
"Check code to eliminate exploitable vulnerabilities"
"Optimize the code to improve performance and minimize delays"

Figure 6. Example of Correlation processes.

Based on these definitions and considering that a modular code can be more understandable and maintainable, we can correlate the two metrics and compute the ratio R = class complexity/maintainability rating. If this ratio is more than a specific threshold set by an expert, the recommendation “Reinforce the modular design of your development” will be provided to the software architect and developers.

In the initial state, we can either receive the input related the class complexity denote cc or the maintainability rating denoted mr. The process accesses respectively to the states “cc received” or “mr received”. If we receive the same measurement related to the same metric, we update its value and loop on the state. Otherwise, if we receive the complementary metric, we compute the ratio R = class complexity/maintainability rating. If this ratio is less than the defined threshold, we come back to the initial state otherwise, we raise the recommendation.

Timers are used to come back to the initial state if the measurements are too old. For sake of place, only this EFSM is presented in Fig. 7. All the others follow the same principles.

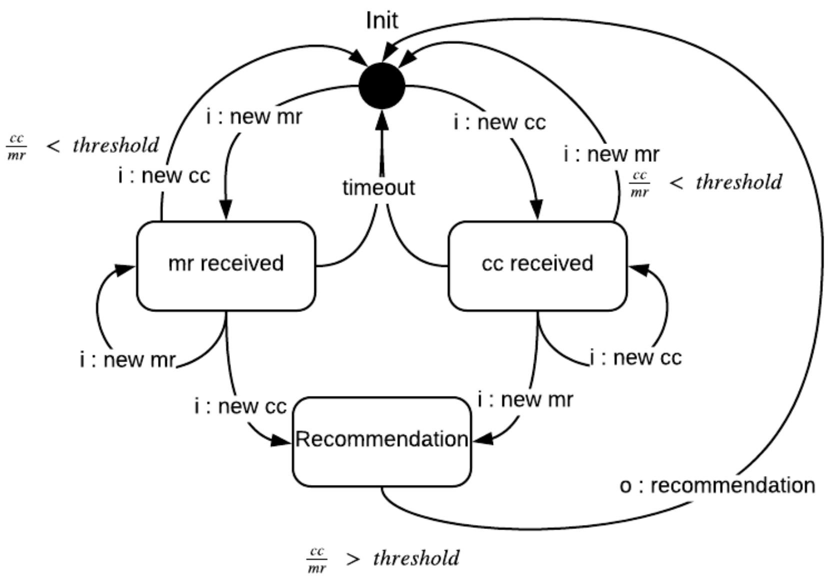


Figure 7. Software Modularity Correlation processes.

Requirements quality

The assessment of the requirements quality can rely on two metrics provided by the SonarQube tool that are the total number of issues and the total number of reopened issues. These numbers are collected during the implementation phase and we can consider that the fact that we reopen an issue many times during the development process can be related to an ambiguous definition of the requirement that needs to be implemented. If we have a ratio R = number of reopened issues/number of issues that is more than a specific threshold, we can consider that the requirements are not well defined and that the development needs more refinement about them. The recommendation “Refine requirement definitions or provide more details” will be provided to the requirements analyst.

Code reliability

The assessment of the code reliability relies on two metrics provided by the SonarQube tool that are the number of issues categorized by severity and the reliability rating. The issues in SonarQube are presented with severity being blocker, critical, major, minor or info and the reliability rating are from A to E: A is to say that the software is 100% reliable and E is to say that there is at least a blocker bug that needs to be fixed. Based on these definitions and considering that a reliable code should be at last free of major or critical issues, we can check that there is no major, critical nor blocker issues and the reliability rating is < C corresponding to 1 Major bug. If this condition is not satisfied, the recommendation “There is unsolved major issues in the code, make a code review and check untested scenarios” will be provided to the software developers and testers.

<u>Software security</u>

The assessment of the software security relies on two metrics, one provided by the SonarQube tool that is the security rating and the other is provided by MMT that is the number of security incidents. The security rating in SonarQube provide an insight of the detected vulnerabilities in the code and are presented with severity being blocker, critical, major, minor or no vulnerability. The number of the security incidents provided by MMT reports on successful attacks during operation. The evaluation of security demonstrates that if an attack is successful this means that the vulnerability in the code was at least major because an attacker was able to exploit it to perform its malicious activity. Based on these definitions and considering that a reliable code should be at last free of major vulnerabilities, we can check if there is a major vulnerability and that the number of attacks at runtime are more than a threshold. If this condition is satisfied, the recommendation “Check code to eliminate exploitable vulnerabilities” will be provided to the software developers and security experts.

<u>Software Performance</u>

The assessment of the software performance relies on two metrics provided by the MMT tool that are the response time and the bandwidth usage. The response time denotes the delay that can be caused by the software, hardware or networking part that is computed during operation. This delay is in general the same for a constant bandwidth (an equivalent number of users and concurrent sessions). Based on this finding, we can correlate the two metrics and compute that the response time is not increasing for during time for the same bandwidth usage. If this response time is increasing, the recommendation “Optimize the code to improve performance and minimize delays” will be provided.

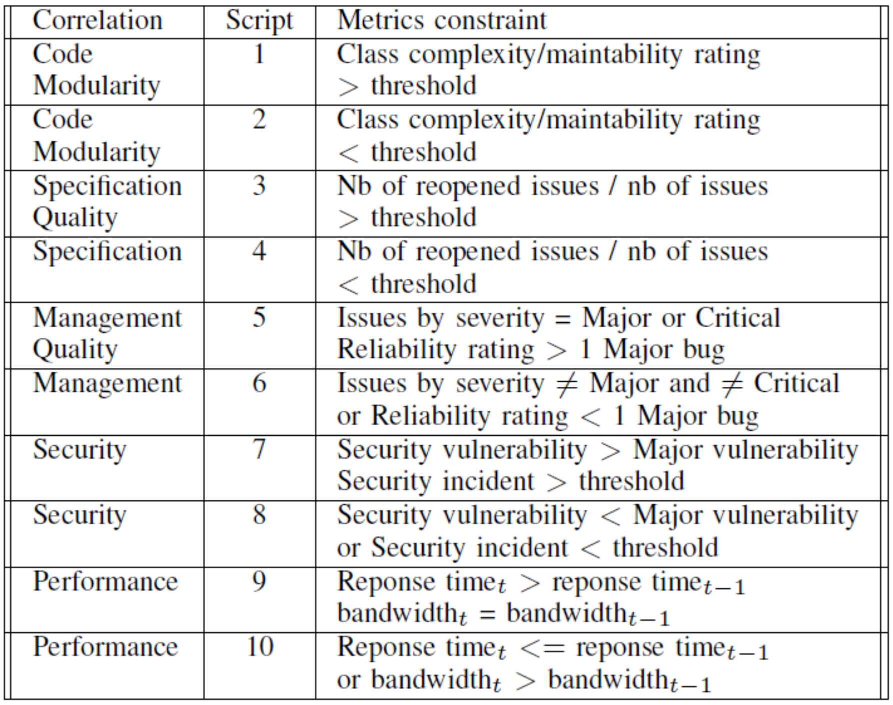
3.2.3. MINT Experiment

To test the efficiency of the MINT tool, we created ten scripts enabling to generate different values for the ten metrics that are relevant for the correlation processes defined in the Figure 6. For each correlation, we created 2 scripts: one that meets the condition that satisfies the recommendation and another that does not satisfy it. The 10 scripts are summarized in next Table 4.

Each script pushes the metric values into an event bus that feeds the 7 correlation processes defined in Section 3.2.2.b. The results correspond to the desired recommendations and the Figure 7 displays an example of recommendation provided by the MINT tool.

This experiment demonstrated the efficiency of the tool. More work is planned to apply this tool to real datasets provided by real users in the context of the software development process.

Table 4. Experiments scripts



Correlation	Script	Metrics constraint
Code Modularity	1	Class complexity/maintability rating $>$ threshold
Code Modularity	2	Class complexity/maintability rating $<$ threshold
Specification Quality	3	Nb of reopened issues / nb of issues $>$ threshold
Specification	4	Nb of reopened issues / nb of issues $<$ threshold
Management Quality	5	Issues by severity = Major or Critical Reliability rating $>$ 1 Major bug
Management	6	Issues by severity $\neq$ Major and $\neq$ Critical or Reliability rating $<$ 1 Major bug
Security	7	Security vulnerability $>$ Major vulnerability Security incident $>$ threshold
Security	8	Security vulnerability $<$ Major vulnerability or Security incident $<$ threshold
Performance	9	Reponse time_t $>$ reponse time_{t-1} $\text{bandwidth}_t = \text{bandwidth}_{t-1}$
Performance	10	Reponse time_t $<=$ reponse time_{t-1} or $\text{bandwidth}_t > \text{bandwidth}_{t-1}$

3.3. Clustering Algorithms provided by the ELKI Analysis Tool

Cluster analysis or clustering[3] is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. The appropriate clustering algorithm and parameter settings depend on data sets such that is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure.

3.3.1. DBScan

DBScan means Density-based spatial clustering of applications with noise (DBSCAN). it is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature. This algorithm was awarded the test of time award (an award given to algorithms which have received substantial attention in theory and practice) at the leading data mining conference. The DBSCAN algorithm can be abstracted into the following steps:

1. Find the neighbors of every point, and identify the core points with more than a given threshold.
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.

[3] *https://elki-project.github.io/*

3. Assign each non-core point to a nearby cluster if the cluster is a neighbor.

A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory. The original DBSCAN algorithm does not require this by performing these steps for one point at a time.

Advantages

- DBSCAN does not require one to specify the number of clusters in the data a priori.
- DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster.
- DBSCAN is designed for use with databases that can accelerate region queries.

Disadvantages

- DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data are processed. For most data sets and domains, this situation fortunately does not arise often and has little impact on the clustering result.
- The quality of DBSCAN depends on the distance measure. The most common distance metric used is Euclidean distance. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value for ε. This effect, however, is also present in any other algorithm based on Euclidean distance.
- DBSCAN cannot cluster data sets well with large differences in densities, since the threshold combination cannot then be chosen appropriately for all clusters. If the data and scale are not well understood, choosing a meaningful distance threshold can be difficult.

3.3.2. KMeans

KMeans (or k-means) clustering is a popular method for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian Mixture Modeling. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

Three key features of k-means which make it efficient are often regarded as its biggest drawbacks:

1. Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.
2. The number of clusters k is an input parameter: an inappropriate choice of k may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
3. Convergence to a local minimum may produce counterintuitive wrong results.

3.3.3. EM

The Expectation–Maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

The EM algorithm is used to find (local) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either missing values exist among the data, or the model can be formulated more simply by assuming the existence of further unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component to which each data point belongs.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values, the parameters and the latent variables, and simultaneously solving the resulting equations. In statistical models with latent variables, this is usually impossible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.

The EM algorithm proceeds from the observation that there is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work, but it can be proven that in this context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point. In general, multiple maxima may occur, with no guarantee that the global maximum will be found. Some likelihoods also have singularities in them, i.e., nonsensical maxima. For example, one of the solutions that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

3.3.4. SLink and Hierarchical Clustering

Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

Metric

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. For example, in a 2-dimensional space, the distance between the point (1,0) and the origin (0,0) is always 1 according to the usual norms, but the distance between the point (1,1) and the origin (0,0) can be 2 under Manhattan distance, or 1 under maximum distance. Some commonly used metrics for hierarchical clustering are:

- Euclidean distance
- Squared Euclidean distance

- Manhattan distance
- Maximum distance

For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used.

Linkage criteria

The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations. Some commonly used linkage criteria between two sets of observations A and B are:

- Maximum or complete-linkage clustering
- Minimum or single-linkage clustering
- Mean or average linkage clustering
- Centroid linkage clustering
- Minimum energy clustering

3.4. Algorithms in Metrics Suggester

The goal of this tool consists on suggesting relevant and efficient measurement plans at runtime using a machine learning algorithm. For that purpose, measurements are performed continuously, and data analysis periodically processed according to cycles (e.g., each 1s, 1mn, half-day, etc.) well-defined by the expert. Besides, it considers a defined set of features, metrics and software classes to give an insight to the measured software quality characteristics. A continuous analysis of the measure metrics and their significance is performed to be matched to the most representative class. Finally, a change in the measurement plan is suggested to only take into consideration the relevant metrics to the project under analysis during the period of analysis. This approach utilizes several concepts that are described in the following. Besides, a tool, namely Metrics Suggester, has been developed by IMT and integrated.

3.4.1. Basics

a) Support Vector Machine: A support vector machine (SVM) [17] is a linear classifier defined by a separating hyperplane that determines the decision surface for the classification. Given a training set (supervised learning), the SVM algorithm finds a hyperplane to classify new data. Consider a binary classification problem, with a training dataset composed of pairs (x1; y1)…..(x_l; y_l), where each vector x_i ∈ Rn and y_i ∈ {-1; +1}. The SVM classifier model is a hyperplane that separates the training data in two sets corresponding to the desired classes. Equation (1) defines a separating hyperplane where w ∈ R^n and b ∈ R are parameters that control the function.

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0 \qquad (1)$$

Function f gives the signed distance between a point x and the separating hyperplane. A point x is assigned to the positive class if f(x) ≥ 0, and otherwise to the negative class. The SVM algorithm computes a hyperplane that maximizes the distance between the data points on either side, this distance is called margin. SVMs can be modeled as the solution of the optimization problem given by (2), this problem maximizes the margin between training points.

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\| \boldsymbol{w} \|^2 \\ \text{subject to:} \quad & y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1, i = 1, \ldots, l \end{aligned} \qquad (2)$$

All training examples label -1 are on one side of the hyperplane and all training examples label 1 are on the other side. Not all the samples of the training data are used to the determine the hyperplane, only a subset of the training samples contribute to the definition of the classifier. The data points used in the algorithm to maximize the margin are called support vectors.

b) Features & Classes: The set of measurements that is classified using SVM is defined as a vector of features. Each feature is a field of a vector and a measurement of one specific measure. Each field is unique. S_o, a feature is a measurement composing a vector for our classification. Further, the vectors are classified into classes according to the feature values. Each class refers to a measured software property, such as the maintainability or reliability. The features composing a vector are the measurements which give information on the classes. Some of them can give information on several classes or only one. The features are chosen according to the metrics defined in the starting measurement plan.

3.4.2. The Mapping System

In order to suggest relevant and effective measurement plans, a mapping system is defined between classes and metrics, and between metrics and features.

It aims at allowing an automate suggestion procedure. This mapping is performed by the experts of the measured system. According to the type of interest (in terms of numbers of vector contained) of the classes highlighted by the SVM classification, some metrics will be added or removed from the measurement plan. Thus, new features will be gathered and others will no longer be.

a) Classes-Metrics: A relationship between a class and some metrics is needed to measure specific targeted software properties. The classes are used for the classification of the vectors according to their features values. As above mentioned, our classification method is to classify a vector in the class corresponding to the property whose the values of the vector show a type of interest.

b) Features-Metrics: The features values inform about the properties (classes) of interest. There are features which give information on only one property and others which can give information on several different properties (complex metrics). Some of the measures can be used by different metrics. Thus, the features associated with a metric are the features corresponding to the measures which composed the metric.

In order to ensure the sustainability of measurement cycles by having at each cycle an information on all measured properties, a set of metrics should always be gathered. This set is called mandatory features. To select the mandatory features, we use the RFE technique, explained below, based on SVM.

c) The Feature Selection: The goal of the Feature Selection (FS) process is to select the relevant features of the raised classes. Its objective is to determine a subset of features that collectively have good predictive power. With FS, we aim at highlighting the features that are important for classification process. The feature selection method is Recursive Feature Elimination (REF) [18]. RFE performs backward elimination that consists of starting with all the features and test the elimination of each variable until no more features can be eliminated. RFE begins with a classifier that was trained with all the features that are weighted. Then, the feature with the absolute smallest weight is eliminated from the feature set. This process is done recursively until the desired number of features is achieved. The number of features is determined by using RFE and cross validation together. In this process each subset of features is evaluated with trained classifier to obtain the best number of features. The result of the process is a classifier trained with a subset of features that achieve the best score in the cross validation. The classifier used during the RFE process is the classifier used during the classification process.

3.4.3. Measurement Plan Suggestion

Based on the classification, matching and FS, two sets of classes are notified: the one with the most vectors called Biggest and the other set constituted of all the other classes called Others. The Biggest means that the corresponding property is the most interested element while the Others means that the corresponding properties are not the elements of interest. Thereby, our Suggestion procedure is applied for the property corresponding to the Biggest. Indeed, the Biggest property needs a further measurement, while the Others one no longer need it. Basically, based on the procedures Analysis and Selection, we raise unnecessary features for the classification that should be removed from the measurement plan. Through this method, the measurement load is increased only on needs and decreasing due to less interested properties. This suggestion approach allows to reach a lighter, complete and relevant measurement plan at each cycle of the software project management.

3.4.4. Suggester Experiment

The suggestion process is evaluated by analyzing the new measurement plans (MP) based on the results of the classification process. These results are used in the feature selection process to identify the class of interest. The objective is to highlight the effects of using the proposed measurement plans and its impact on the classification of new data and on the amount of data collected by this plan. The measurement data used are the measurement results provided by our industrial platform.

Setup

We herein considered the following measurement plan which determined by our expert. An initial MP can be defined by 15 features, 15 metrics and 4 software quality properties. Each metric is composed of only one feature and the mapping between metrics and classes is the following: (i) Maintainability (Class 1): cognitive Complexity, Maintainability Index, Code Size, Nb of issues, (ii) System Performance (Class 2): Computational Cost, Infrastructure Cost, Communication Cost and Tasks, (iii) Performance (Class 3): Response Time, Running Time and I/O Errors, (iv) Functionality (Class 4): Usability, Precision, Stability Response Time and Illegal Operations.

Using the previously described plan, we considered the class with the most predicted instances during each cycle. A huge set of 16,000,000 unclassified vectors were processed. This data set was divided into 32 subsets each containing 500,000 vectors. For each period of the suggestion process, only one subset was used as input.

The initial measurement plan used during the experiment consisted of the following 5 metrics: Maintainability Index, Response Time, Running Time, Usability, Computational Cost. These metrics where selected by the expert as an example of a measurement plan with a small number of metrics that has links to all software quality properties. During the suggestion process a number was assigned to each metric. In our experiments the number of each is shown in Table 5. Table 6 lists the plans and in which cycle they were used.

Table 5. Each metric and its assigned index during the suggestion process.

Index	Metric
1	Cognitive Complexity
2	Maintainability Index
3	Code Size
4	Nb of issues
5	Response Time
6	Running Time
7	Usability
8	Computational Cost
9	Infrastructure Cost
10	Communication Cost
11	Tasks
12	I/O Errors
13	Precision
14	Stability Response Time
15	Illegal Operations

Table 6. Measurement plans used during the suggestion process and the cycles where they were used. Metrics of the plans are represented by the indexes describe in table 5.

	Metrics	Cycles
MP1	2, 5, 6, 7, 8	1
MP2	4, 5, 6, 12	2, 4, 17, 22, 23, 24
MP3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15	3, 5, 18
MP4	8, 9, 10, 11	6, 30
MP5	7, 8, 9, 10, 11	7, 8, 9
MP6	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15	10
MP7	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	11, 19, 20
MP8	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	12, 21
MP9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	13, 14, 15, 16
MP10	3, 4, 5, 6, 8, 9, 10, 11, 12	25
MP11	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	26, 32
MP12	1, 2, 3, 4, 5, 6, 8, 9, 10, 11	27
MP13	1, 3, 4, 5, 6, 8, 9, 10, 11, 12	28
MP14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	29
MP15	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	31

Results

During the suggestion process, 15 metrics (Table 5) were available to suggest new MP. Fig. 6 shows how the classification of the vectors was distributed during the cycles and the percentage of the vectors assigned to each class. From these metrics, 15 unique measurement plans were used in the suggestion process.

MP1 was only used at the beginning of the process, this was the plan suggested by the expert. We note that MP2 was the most used plan during the process (6 times). This plan is composed by the metrics linked to the Performance property and was suggested when the classification of vector to class 3 overwhelmed the other classes. This tells us that if we focus on the Performance property then the metrics in MP2 are sufficient.

MP3 was suggested when the four classes were present in the classification results and class 4 was the class of interest. The tool suggests to take into consideration more than the linked metrics to the class, it seems that these features help to the classification of class 4. MP4 was suggested when the input vectors were only classified to class 2, this MP2 consists of the metrics linked to that class. This happens when the input vectors are classified to only one class, the same can be observed in cycle 1 but with class 3. MP5 has only one more metric than MP4, Usability. It is also a MP focused on System Performance property. MP11 was also suggested when class 2 overwhelmed the number of classifications during the classification phase.

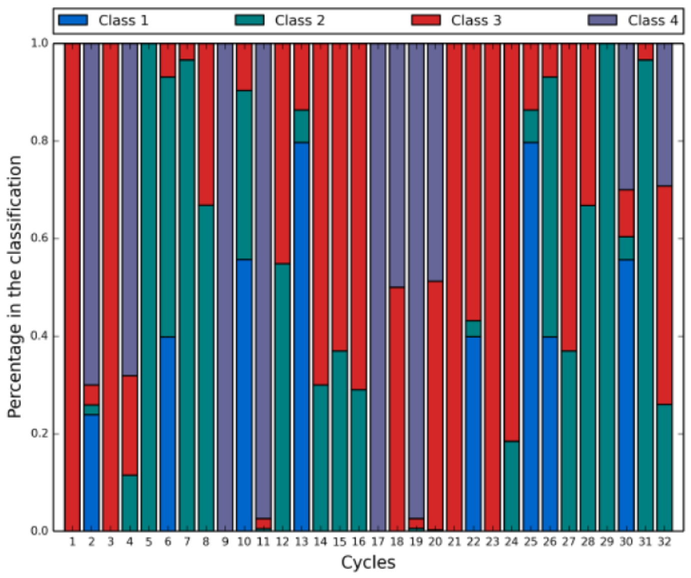


Figure 8. Classification results of each cycle. The results show the percentage in the predictions of each cycles for the 4 classes.

MP7, MP8 and MP9 are very similar measurement plans. These plans have the highest number of metrics, MP7 15 metrics and MP8&9 14 metrics. These plans are suggested when the classification results usually have more than 2 classes. This is because the classes do not share any metric between them. A measurement plan with the majority of the metrics is expected to classify well the majority of the classes. MP10, MP12, MP13, MP14 and MP15 where suggested in the same case as the previously mentioned plans but these plans where only suggested one time during the process.

3.5. STRACKER – algorithms and tools

STRACKER from UnBuc tool has three main capabilities: metrics prediction, metrics correlation, and metrics forecasting.

3.5.1 Metrics prediction

The first problem we solve with STRACKER is the following: As input: We have values for *k* metrics - metric_1, ..., metric_*k*. As output: We predict the value of another metric M that is correlated to the k metrics

We tested several machine learning algorithms on a large training set in order to obtain a model which will be used to predict the value of the output metric. As training set, we used open datatasets are from tera-PROMISE http://openscience.us/repo/, which contains historical values for many software projects. In particular, we had 33 projects, with many intermediate versions for each project and many classes for each version. From that we extracted 86 000+ rows for 21 features

As a use case, we showed how we can predict existence of bugs in the code (one feature) based on the other 20 features.

In the table below, we show our experiments with different machine learning algorithms (left column) and their accuracies. We can notice that the neural network algorithm performed best.

Table 7. Accuracies of different machine learning algorithms

Decision Tree	81.05%
Random Forest	82.46%
SVM	83.62%
Logistic Regression	83.88%
KNN	84.04%
Gradient Boost	84.23%
Neural Network	**85%+**

The models for all the algorithms except Neural Network were created using the Scikit-learn library described below. The model with Neural Network was created using Keras over Tensorflow, both of them also described below.

The bug prediction model is not yet included in the tool integrated in the platform, because we do not have yet all the metrics implemented in the MEASURE platform. However, we have now the framework available to experiment with various sets of metrics from the MEASURE platform, especially those from SonarQube. With our initial experiments we obtained the best results with neural networks (and they can be improved further by fine tuning), but for good accuracy it is very important to have a big dataset of measures and those are not yet available in MEASURE platform (we are working now on importing such data needed for training in the platform).

3.5.2 Metrics correlation

The second problem we solve is the following. As input: We have values for 2 metrics metric_1, metric_2. Output: We check their correlation (the method works with several metrics, but we exemplified it on two in this deliverable.

To get correlation between metrics we used regression algorithms. Below is an example of Linear Regression.

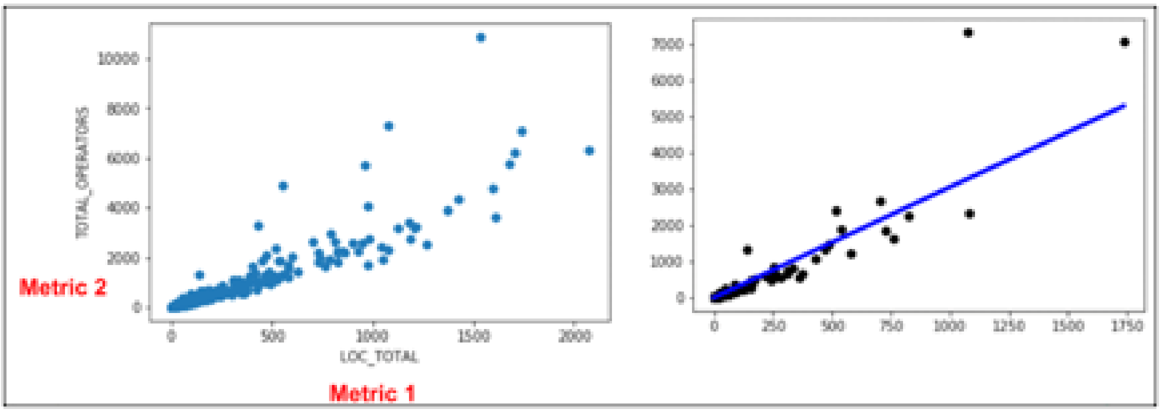


Figure 9. Example of linear regression

3.5.3 Metrics forecasting

The third and final problem we implemented in STRACKER is the following: As input: We have (historical) values for one metric. Output: We forecast the future values of the metric

This module predicts possible values of a metric in the future using time series prediction algorithms) To generate these values, we used ARIMA technique (*autoregressive integrated moving average*, described below).

Here we have an example of ARIMA prediction

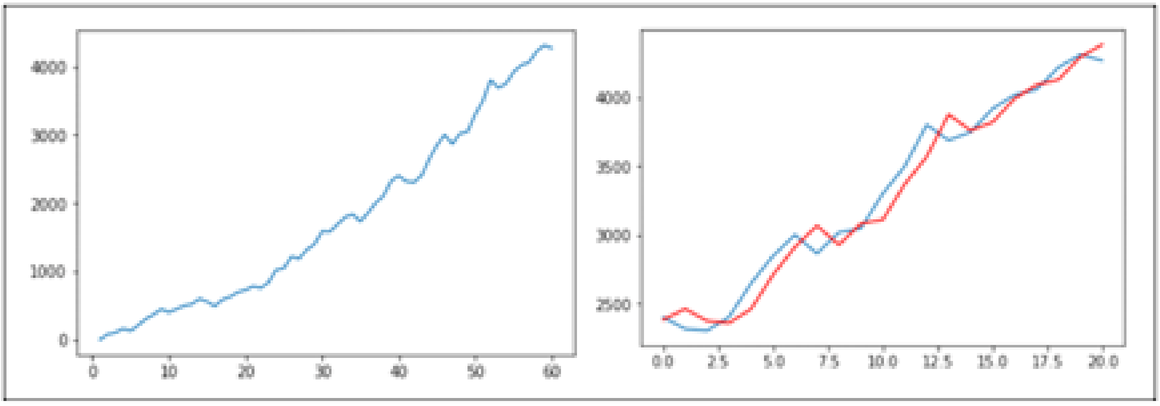


Figure 10. Example or ARIMA prediction

In the left side we have an image with graphical representation of a series values for a metric. In the right side we have a zoom in on the last 33% of the values. There, we compare the actual values with blue (of the last 33% of the ones in the left), with the predicted values depicted by the red line. The prediction in red is based in the based on the 66% of the historical data. As we can see, the predicted values are quite accurate.

Below we describe shortly the **algorithms used by STRACKER:**

Neural networks are data processing elements, inspired by the structure of the neural networks of the human brain. They try to simulate the learning process of people and are used in various fields, including in the field of classification. A detailed description was given at the beginning of this deliverable.

Linear regression is a very simple machine learning algorithm that calculates the relationship between two or more independent variables based on a linear equation. It is a supervised learning algorithm, a regression algorithm (as the name says) that is also frequently used in statistics and analysis field. A detailed description was given at the beginning of this deliverable.

Logistic regression is a very popular machine learning algorithm used for binary classification. The algorithm is easy to implement using Scikit-learn library.

KNN is an algorithm used for classification or regression. For classification, an object is classified by the majority of values of its neighbours. For example, if we have a binary classification (with 0 and 1), if there are more neighbors with 1 then the object will have the value 1, and 0 otherwise.

Time series forecasting: Forecasting is the process of predicting future data values based on historical data values and is an important part of the Machine Learning domain. ARIMA (*autoregressive integrated moving average*) is a technique used for forecasting that uses historical data to make predictions of future data. It depends on 3 parameters, whose values can change from model to model. p - refers to past values; d - refers to data stationarity (nonseasonal differences); q - refers to prediction errors. The optimal values for the three parameters differ from one model to another, depending on the set data values. Programming languages like Python or R are very popular for working with ARIMA models.

Also, we mention the t**ools used for the implementation and experimentation:**

Scikit-Learn is a very popular library for Python language that contains many Machine Learning algorithms. In addition to the wide variety of algorithms, it also allows many data processing methods (working with libraries like numpy and pandas) and algorithm testing (model accuracy score).

Tensorflow is a library used for Machine Learning, especially for Deep Learning, for working with neural networks. It supports multiple programming languages, multiple operating systems and can break across multiple CPUs or GPUs. Tensorflow also supports Keras as an interface to facilitate user experience.

Keras is a library that allows working with neural networks and runs over libraries such as Theano or Tensorflow. It is more like a framework or interface for libraries over which it runs and facilitates user interaction with the Deep Learning domain, providing simple but diverse methods for creating and managing neural networks.It is very suitable for those who start working in the field of Deep Learning (you can simply add new layers, fit a model or get accuracy etc. with few lines of code), but it is also widespread among experienced users because it is very flexible; you can quickly create both simple and complex neural networks.

3.6. Other algorithms

Other algorithms will be used by the Turkish consortium. Contributions are still at the stage of data preparation/model validation and integration to Measure platform has not been commenced yet.

- ***Random Forest (RF):*** This is an ensemble learning method consisting of a set of decision tree classifiers. Each tree in the forest is triggered by an independently created random number vector.
- ***Naïve Bayes (NB):*** This method uses Bayes' rule to do the classification by computing class probabilities and using observed attribute values. The method is called “naïve” since it has two basic assumptions: attributes are conditionally independent, and no hidden factor impacts on the prediction process.
- ***REPTree:*** This is a fast decision tree algorithm that generates a decision tree using information gain method to split. Missing values are managed as in C4.5 algorithm.
- ***J48:*** It is a Java implementation of a slightly different version of C4.5.
- ***LMT:*** Logistic Model Trees are standard decision trees which use logistic regression functions at their leaves.
- ***Multilayer Perceptron (MLP):*** MLP is a feed-forward artificial neural network which uses back propagation training algorithm. It is a system of interconnected nodes or neurons which maps an input

vector into an output vector to maintain a nonlinear relation. The neurons are connected via weights and output signals.

- ***Support Vector Machines (SVM) and Sequential Minimal Optimization (SMO):*** In linear case, an SVM is a hyperplane that set a boundary between some positive instances and negative instances. It can also be further extended to non-linear cases. Training an SVM requires quadratic programming (QP) optimization problem solving which is a very time and memory consuming operation and SMO is a substantial improvement on the original training algorithm.
- ***K-nearest Neighbour Classifier (IBk):*** It classifies a data point based on its k most similar other data points.
- ***ZeroR***: It predicts the majority class of nominal test data while it predicts the average value if numeric class is the case. it is used as a baseline for the performance of machine learning algorithms.
- ***OneR:*** This method classifies instances based on a one rule which is extracted from a single attribute.

4. References

[1] Natalia Kushik, Jorge López, Ana Cavalli, Nina Yevtushenko: "Improving Protocol Passive Testing through 'Gedanken' Experiments with Finite State Machines", in the proceedings of the IEEE International Conference on Software Quality, Reliability and Security, QRS, Vienna, Austria, 2016.

[2] Rainer Gerhards: "The Syslog protocol", Request for Comments (RFC) 5424, DOI: 10.17487/rfc5424, 2015.

[3] https://www.iso.org/standard/35733.html

[4] K. P. Bennett and A. Demiriz, "Semi-Supervised Support Vector Machines," Advances in Neural Information Processing Systems, pp. 368– 374, 1999.

[5] S. Dahab, S. Maag, A. Bagnato, and M. A. A. da Silva, "A learning based approach for green software measurements," in Proc. of the 3rd International Workshop on Measurement and Metrics for Green and Sustainable Software Systems, MeGSuS 2016, co-located with the 10th International Symposium on ESEM 2016, 2016, pp. 13–22

[6] S. Dahab, S. Maag, X. Che,"A Software Measurement Framework guided by Support Vector Machines." The 5th International Workshop on Network Management and Monitoring, NetMM 2017, Accepted to be Published, Taiwan

[7] Hawk: towards a scalable model indexing architecture Konstantinos Barmpis, Dimitris Kolovos University of York, Heslington, York, UK Proceeding BigMDE '13 Proceedings of the Workshop on Scalability in Model Driven Engineering Article No. 6

[8] Integration of a graph-based model indexer in commercial modelling tools - Antonio García-Domínguez, Konstantinos Barmpis, Dimitrios S. Kolovos, Marcos Aurélio Almeida da Silva, Antonin Abherve, Alessandra Bagnato MoDELS2016 DOI: 10.1145/2976767.2976809

[9] MONDO Project Web Site

[10] MONDO Deliverable 5.5 D5.5 Model Indexing Framework Final Version

[11] Hawk Model Indexer in GitHub,

[12] Modelio Modeling Tool, www.modelio.org

[13] T. Stoenescu, A. Stefanescu, S. Predut, F. Ipate. RIVER: A Binary Analysis Framework using Symbolic Execution and Reversible x86 Instructions. In Proc. of 21st International Symposium on Formal Methods (FM'16), LNCS 9995, pp. 779-785, Springer, 2016.

[14] T. Stoenescu, A. Stefanescu, S. Predut, F. Ipate. Binary Analysis based on Symbolic Execution and Reversible x86 Instructions, To appear in Fundamenta Informaticae 153, 2017.

[15] C. Paduraru, M. Melemciuc, A. Stefanescu. A distributed implementation using Apache Spark of a genetic algorithm applied to test data generation. To appear in Proc. of PDEIM'17, workshop of GECCO'17, ACM 2017.

[16] A. Grez, C. Riveros, and M. Ugarte, "Foundations of complex event processing," CoRR, vol. abs/1709.05369, 2017. [Online]. Available:

[17] V. N. Vapnik and V. Vapnik, Statistical learning theory. Wiley New York, 1998, vol. 1.

[18] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in Science and Information Conference (SAI), 2014. IEEE, 2014, pp. 372–378.

[19] A Tour of Machine Learning Algorithms

https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

[20] Artificial Intelligence, Deep Learning, and Neural Networks, Explained

https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html

[21] Introduction to Machine Learning

https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning#machine-learning-methods

[22] A survey of machine learning for big data processing

https://link.springer.com/article/10.1186/s13634-016-0355-x

[23] A survey of machine learning for big data processing

https://link.springer.com/article/10.1186/s13634-016-0355-x

[24] Introduction to Machine Learning

http://cs.du.edu/~mitchell/mario_books/Introduction_to_Machine_Learning_-_2e_-_Ethem_Alpaydin.pdf

[25] Y Bengio, A Courville, P Vincent, Representation learning: a review and new perspectives. IEEE Trans Pattern Anal **35**(8), 1798–1828 (2012)

[26] D Yu, L Deng, Deep learning and its applications to signal and information processing. IEEE Signal Proc Mag **28**(1), 145–154 (2011)

[27] R Bekkerman, M Bilenko, J Langford, *Scaling up machine learning: parallel and distributed approaches* (Cambridge University Press, Oxford, 2011)

[28] D Yu, L Deng, Deep learning and its applications to signal and information processing. IEEE Signal Proc Mag **28**(1), 145–154 (2011)

[29] SJ Pan, Q Yang, A survey on transfer learning. IEEE Trans Knowl Data Eng **22**(10), 1345–1359 (2010)

[30] Y Fu, B Li, X Zhu, C Zhang, Active learning without knowing individual instance labels: a pairwise label homogeneity query approach. IEEE Trans Knowl Data Eng **26**(4), 808–822 (2014)

[31] G Ding, Q Wu, YD Yao, J Wang, Y Chen, Kernel-based learning for statistical signal processing in cognitive radio networks. IEEE Signal Proc Mag **30**(4), 126–136 (2013)

[32] Machine learning platforms comparison: Amazon, Azure, Google, IBM

http://searchbusinessanalytics.techtarget.com/feature/Machine-learning-platforms-comparison-Amazon-Azure-Google-IBM

[33] KDnuggets, By Matthew Mayo https://www.kdnuggets.com/2015/10/top-arxiv-deep-learning-papers-explained.html