American Journal of Interdisciplinary Research and Development
ISSN Online: 2771-8948
Website: www.ajird.journalspark.org
Volume 04, May, 2022

NEW SOFTWARE COST ESTIMATION APPROACH BY USING MACHINE LEARNING BASED FEATURE EXTRACTION TECHNIQUES

Maryam Thabit Hussein Al-Khazraji

Dhulfiqar Mahmood Tawfeeq Al-Saada

Asst. Prof. Dr. Abdullahi Abdu Ibrahim
mrymthabthsyn57@gmail.com
Iraqi Ministry of Health

Abstract
In this study, new software cost estimation approach presented by using machine learning techniques based feature selection method. The proposed method consist from two stages, the feature selection stage which factor analysis applied to select best features and remove unaffected features from input data. In the second stage, the naïve ayes classifier applied to classify the selected features. We applied the method to the NASA software dataset, which is free dataset available online and used by researchers as metrics to test the detection methods. Then, the presented method compared with several studies presented in this field.

Keywords: Machine learning, factor analysis, naïve ayes classifier, and software cost estimation.

Introduction

The first study on the organization of software engineering undergraduate programs started in 1987 with the conference "The Conference on Software Engineering Education and Training-CSEET". These conferences, which are repeated in different periods by SEI (Software Engineering Institute), have been completed and the studies to improve the programs continue. Another important study on the preparation of training programs is the project named "The Guide to Software Engineering Body of Knowledge– SWEBOK", which was initiated in 1998 and focuses on Software Engineering standards supported by IEEE. It is of great importance to update the basic and specialist courses of software engineering programs prepared according to the SWEBOK classification in this context. Beykent University Software Engineering Department students, who started education in 2008, can turn their research and knowledge into part-time working opportunities in the industry or service sector, especially in the last years of their education (Altan, 2010a). Thus, by entering the

industrial software engineering world during their student years, they can experience software product development projects that are rapidly developing in our country, but have a huge deficit. It is clear that the contribution of the constantly updated course contents will be great for students to be the preferred engineer candidates according to their interests. Nowadays, it has been understood that software is not just about writing "program code", it has even reached a more advanced level of importance than hardware. If it is desired to make a special job or develop a "computerized system" other than computer-ready package programs, it is absolutely necessary to develop a software. The software can be a program of several lines running on a computer, or it can be a group of millions of lines of programs running on multiple computers. The higher the complexity of the programs, the more difficult it is to control their development. Therefore, curriculum development methodology is needed. Especially after the software crisis that emerged in the United States in the 1970s, the importance of software engineering and effective programming languages is better.is understood. Software issue has become a common research subject in the world that has become increasingly global and common needs. For this reason, a sub-discipline such as database, artificial intelligence, architecture, graphics has been established under the name of "Software Engineering" within the scope of the Computer Science main discipline at universities, and various disciplines and standards have been defined by international organizations in this field. started to be watched as a partner. This book emphasizes the importance of software and software engineering, approaches software engineering with the system engineering window, gives information about what software engineering is, and talks about the most modern software development techniques, methods and standards. However, in the book a specific method, a standard commercial package software or a programming language as a base Various general information is given, with emphasis only on basic rules and principles. In the thesis, first of all, computer-based systems are introduced, the general features of the systems are mentioned, it is emphasized that the software is not a stand-alone system and that development must be made with a system idea. For this purpose System engineering, methods applied in the system development process and the place of software in the system life cycle are emphasized. Only software development processes among the software and hardware development activities that make up the system. Its methods and attributes are explained, and the software life cycle is explained in detail. Throughout the entire book, development and documentation are based on standards, with sample templates for some documents. Small software is now everyone the methodology and activities for the high quality development of medium and large-scale software are constantly mentioned. Computer engineering is a very broad science with various sub-branches. This work focuses on software engineering only. Software included in the work engineering subjects should be considered as preliminary

information, and other sources should be consulted to obtain more detailed information on a specific subject such as wish specification, structural design, object oriented analysis and design, test engineering.

Contributions of The Thesis

Presented new software cost estimation system combining the factor analysis technique with naïve Bayes classifier.
The proposed system presented accurate results when compared with several studies in literature.
The system presented low execution time and increase the speed of naïve classifier when dealing with large datasets.

Research Questions

In this thesis, several points were studied and investigated to deal with IDS attacks, these points are described below.
How to detect the software cost estimation in minimum cost and accurate performance?
How to determine the classification technique to detect software cost estimation?
How to increase the accuracy by combining naïve Bayes with factor analysis?
How applied factor analysis to improve the naïve Bayes classifier?

Thesis Hypotheses

Machine learning presented computer programs in a beneficial method to achieve software cost estimation.
Effective systems with low cost developed using data mining techniques in different fields of software engineering.
Combine feature selection technique factor analysis with naïve Bayes.

Overview

Software Engineering Costs Estimation

Software engineering (industrial programming) is commonly associated with development of large and complex programs by teams of developers. Becoming and the development of this area of activity was caused by a number of problems associated with the high the cost of software, the complexity of its creation, the need management and forecasting of development processes. In the late 60s - early 70s of the last century, an event occurred that entered history as the first crisis of programming. The event was that the cost software began to approach the cost of hardware ("hardware"), and the dynamics of growth of these values made it possible to predict that by the mid-90s all humanity will develop software for computers. Then and started talking about

software engineering (or industrial technology programming, as it was called in Russia) as a kind of discipline, the goal which is to reduce the cost of programs. Since then, software engineering has developed quite rapidly. Stages The development of software engineering can be distinguished in different ways. Each stage is associated with the emergence (or awareness) of the next problem and finding ways and means solutions to this problem. The term itself - software engineering (software engineering) - was first voiced in October 1968 at a conference of the NATO Science and Technology Subcommittee (Garmisch, Germany). 50 professional software developers from 11 countries attended. Considered design, development, distribution and support issues programs. There, for the first time, the term "software engineering" was heard as a certain the discipline to be created and guided in the decision listed problems. In the early stages of the development of software engineering (even when it is has not yet been named) it was noted that the high cost of programs is associated with development of identical (or similar) code fragments in different programs. This was due to the fact that in various programs, as part of these programs, the same (or similar) tasks: solving nonlinear equations, calculating wages boards, ... Using previously written fragments when creating new programs promised a significant reduction in development time and cost. In software development projects, many errors occur in work results such as the technical concept or the software code. This finding is not surprising at first. It is interesting, however, that various empirical studies Since the mid-seventies until today we have found that the majority of the errors, namely 15% to 50%, are caused by the requirements analysis [AW05]. These are, on the one hand, requirement errors and, on the other hand, consequential errors in the design and implementation. The investigations also show that the requirement errors are often overlooked or misunderstood requirements, i.e. errors that are usually only discovered very late. The later an error is discovered, the more consequential errors can arise and the more time-consuming it becomes Correction. The cost of revisions can represent up to 41% of the total project costs [Di93]. The ramifications of these necessary revisions are familiar: software ships late and budgets are exceeded. Become however, if the revisions are not carried out, customers will receive lower quality software. Errors find their way into work results because they are caused by the process, i.e. the type and Way of performing a task. Although errors can provide relevant information about the process, it is not used. Not even if error tracking supported by a software tool is practiced(usually referred to as "bug tracking tool" or "defect tracking tool"). Corresponding tools usually do not differentiate between errors and their symptoms, e.g. B. the malfunction of the software. They also suggest collecting purely administrative information only: When did the misconduct occur? Has the error already been corrected? Who is responsible for rectification? etc. However, if you limit yourself to this administrative information, you fail to do so

because of errorsto learn. Errors are then only corrected and are then quickly forgotten again. There is no sustainable process improvement.

Concept of Software Engineering

The next stage in the increase in the cost of software was associated with the transition from development of relatively simple programs to the development of complex software systems. It should be noted that this transition was caused by the emergence of computational third generation technology (integrated circuits). With the transition to use integrated circuits, the performance of computers increased by orders of magnitude, which created prerequisites for solving complex problems. These complex tasks include:

Large amount of code (millions of lines)
A large number of links between code elements
A large number of developers (hundreds of people)
A large number of users (hundreds and thousands)
Long-time of use

For such complex programs, it turned out that the bulk of their cost falls on not to create programs, but to implement and operate them. By analogy with industrial technology began to talk about the life cycle of a software product, as a sequence of certain stages: design stage, development, testing, implementation and maintenance. Software engineering (or industrial programming technology) as a direction arose and was formed under the pressure of the growth in the cost of the software being created. The main goal of this area of expertise is to reduce the cost and time of software development. Software engineering has gone through several stages of development, during which the fundamental principles and methods of developing software products were formulated. The basic principle of software engineering is that programs are created as a result of several interrelated stages (requirements analysis, design, development, implementation, maintenance) that make up the life cycle of a software product. The fundamental design and development methods are modular, structured, and object-oriented design and programming. Despite the fact that software engineering has achieved some success, the permanent programming crisis continues. This is due to the fact that the turn of the 80s-90s is marked as the beginning of the information technology revolution caused by the explosive growth in the use of information resources: personal computers, local and global computer networks, mobile communications, e-mail, Internet, etc. To solve a problem, engineers use theories, methods and tools that are suitable for solving a given problem, but they apply them selectively and always try to find solutions, even in cases where theories or methods corresponding to the given problem do not yet exist. In this case, the engineer looks for a method or means to solve the problem, applies it and is responsible for the

result - after all, the method or means has not yet verified. A set of such engineering methods or methods, theoretically possibly not justified, but which have received repeated confirmation in practice, plays a large practical role. In software engineering, they are called best practices. Engineers work with limited resources: time, financial and organizational (equipment, machinery, people). In other words, the product must be created on time, within the allocated funds, equipment and people. Although this primarily refers to the creation of custom products (stipulated in the terms of the contract), but when creating boxed products, these restrictions are of no less importance, because here they are dictated by the conditions of market competition. Computer science deals with the theory and methods of computing and software systems, while software engineering deals with practical problems of software development. Computer science forms the theoretical foundations of software engineering and a software engineer must know computer science. So the same as an electronics engineer must know physics. Ideally, software engineering should be supported by some kind of computer science theories, but in reality this is not always So. Software engineers often use techniques that are only applicable in specific contexts and cannot be generalized to other cases, and elegant computer science theories cannot always be applied to real large systems. Finally, computer science is not the only theoretical foundation for software engineering, as it is. The range of problems facing a software engineer is much wider than just writing programs. This is also financial management, organization of work in a team, interaction with a customer, etc. Solving these problems requires fundamental knowledge that goes beyond computer science. The second significant difference is that the program is an artificial object. Those. there are no objective laws for a program to govern its behavior. For example, a civil engineer has objective laws of structural mechanics: the balance of moments and forces, the stability of mechanical systems, etc. The civil engineer can check his architectural designs against these laws and thereby ensure the success of the project. These laws are objective, they will always work. At first glance, a software engineer also has typical, proven time architectural solutions (for example, client-server architecture). But these decisions are determined by the level of development of computing technology (and the level of requirements adequate to them). With the advent of technology with fundamentally new capabilities, the software engineer will have to look for new solutions.

The Cycle of Software Development

In order to develop management methods for software engineering, one must first understand the process of software development. A key point here is the process model. A process model, also: life cycle model, is generally structured design or production process in structured activities or phases, which in turn are assigned to

corresponding methods and techniques of the organization. The task of a process model is to present the tasks and activities that generally occur in a design process in their logical order: Who does what at what time and with what result? As part of a project management an activity often ended with a milestone. The terms "phase" and "activity" are often not clearly defined, in project management a project is usually divided into phases [6, p. 10],while a software project is mostly understood as a development process that consists of individual activities. In software development, one usually differentiates between the following activities or phases. Requirements definition, often also needs assessment. This process determines what the customer actually needs. Since, in general, special IT knowledge cannot be assumed for the customer and, on the other hand, for the developer no knowledge of the application area, the main task of the requirements definition is to clarify the requirements and, in a deeper sense, to establish and maintain communication and knowledge acquisition. Some software is created supply-oriented, i.e. without an individual customer in mind. Examples of this are commercial software products such as ERP systems. Operating systems, or game software. In these cases, customer needs are anticipated. Analysis, often also requirements specification. In this phase, the results of the requirements definition are written down as precisely as possible. The specification (or the functional specification) should only contain why the software is created and what it should do (and as precisely as possible), but not how it should do it. The basic problem of the analysis is that it is supposed to fulfill two opposing goals. On the one hand, it has to be informal, intuitive and understandable for non-computer scientists so that the customer can validate: “We are creating the right product.” On the other hand, it has to be precise enough so that developers and testers always verify can: "We create the product correctly. "Maintenance. Unlike mechanical systems, software systems do not wear out over time. A program doesn't need an overhaul after 300,000 km or 10 years, it will work forever as it did on the first day. That's the problem! In order for software to remain useful, it must be modified. Modifications may become necessary because the hardware is being replaced, legal regulations have changed, requirements have increased or security gaps or errors ("bugs") have occurred. Such modifications are usually carried out by means of patches, i.e. selective changes to the software. Often the remarkable recursive effect can be observed that the newly deployed software changes the area in which it is deployed. For example, the software introduced to speed up his work processes could lead to a tax consultant winning new customers. But now he is dependent on the software, he can no longer do without it. A prototype of the software system or important system parts is often created before the design phase to validate the requirements specification. The prototype is revised until the customer or user agrees. A prototype only illustrates basic aspects of the software. Typical is the creation of prototypes of the user interface

(horizontal prototypes) to illuminate certain aspects of the user interface; In extreme cases, such prototypes can consist of paper sketches; sequences of such surface sketches (storyboards) similar to a comic strip can be created to illustrate entire processes. In the case of a function prototype (vertical prototype), on the other hand, a section of the required functionality is implemented that covers all layers of the software, for example from the user interface to the database connection to check certain requirements for the performance of master data management. Often, prototypes are realized in rapid prototyping, ie "quick and dirty" to get around to quickly provide clarity about a certain aspect and to save effort. The prototype can be thrown away during the transition to the design (disposable prototype) or it can be further developed and incorporated into the end product. Investigations indicate point out that the quality of the end product is higher with disposable prototypes [12, p. 22].the software is a project. Each of the phases can be given a time schedule for its milestone, i.e. its completion. It becomes so for project management possible to estimate costs and deadlines through the time planning and resource allocation for the individual phases. Overall, the waterfall model (in its strict interpretation) is a document-oriented process model, because for each development phase there is a clear definition of which products or result documents must be available after completion. Such result documents can be programs or documentation and serve as a Control points of the overall process. The project management and, if necessary, the customer, can use the result documents to track the progress of the process. In addition to these positive properties in terms of project management, the model also has a serious disadvantage. In order for a phase to begin, it must the previous phase will be completely done. The catalog of requirements must be complete be created before you can start designing. Conversely, the catalog of requirements is permanently frozen when the design phase has started and can no longer be changed. Often, however, the problem is only vaguely understood in the first phases, or essential detailed problems or even thinking errors are only discovered in the later phases. Grows during the design phase or implementation often only the understanding of the problem. So there are often, and especially with new and innovative systems, wrong decisions in the first phases, which are later need to be corrected. However, if the principle of the irreversibility of the phase decisions is dissolved, the project becomes more difficult to plan and an initially planned one Budget can quickly get out of hand. The waterfall model is particularly suitable when the developers (!) Have a sufficiently precise idea of the problem area, for example when an experienced software team is working on a new product that is similar to one that they have already developed. In the case of particularly innovative or simply not yet sufficiently good In contrast, the waterfall model is rather unsuitable for understood systems. The modified waterfall model was an early, quite obvious adaptation of the procedural model to these

difficulties (Figure 2.2). It enables feedback at least to the next earlier phase if errors have occurred. If this procedural model is used consistently, in the worst case the nightmare of a classic project manager can arise and already reached milestones can be revised and are to be obtained again. Every software development moves in the field of tension between this conflict of objectives between planning and budgeting on the one hand and requirement fulfillment and quality on the other. The following sections describe approaches to realizable solutions to this trade-off. In addition to standardized methods or rules of thumb in software engineering that increase the efficiency of software development, there is another factor: the software analysts, developers and managers as people, with all their strengths and weaknesses.

According to a well-known anecdote [5, p. 428ff], there was a drinks machine at one end of a large computer room for students in a university. After complaints from some students that there was too much noise coming from this corner of the room the machine removed. As a result, however, the consulting effort of the two employed tutors increased so dramatically that they could no longer process the students' inquiries. What happened? The noise of the students at the drinks machine was therefore that they could talk to each other about their computer problems and through that communication solve them on the spot. Only exceptionally difficult problems were brought up to the tutors in this way. By removing the unconventional but highly efficient advisory service of the drinks machine, intellectual productivity fell, although the opposite was intended. Achieving goals are respected and technical and social problems in software development are addressed. Agile software development is a countermovement to the software development processes, such as the Rational Unified Process, which are often viewed as heavy and bureaucratic. A completely new principle of agile software development is pair programming. The source text is created by two programmers on one computer. A programmer, the "driver", writes the code while the others, the "navigator", think about the problem, check the written code and immediately address problems that occur to him, e.g. spelling mistakes or logical errors. These can then be resolved immediately by talking to two people. The two programmers should take turns on these two roles periodically. The composition of the couples should also change regularly. Pair programming leads to programs of significantly higher quality than individual programming [5, p. 428]. Furthermore, traditional development methods follow an "assembly line strategy" and consider the developers (analysts, programmers, testers) to be interchangeable within their respective groups. Agile software development, on the other hand, focuses on individual skills and does not set any limits; every developer should gain knowledge and experience by working with others. The same team goes through all phases of a cycle from analysis to functional testing. One of the most common agile development models is eXtreme programming (XP). It is suitable for smaller projects with up to 15

developers in an environment with rapidly changing requirements. The overriding principle of eXtreme programming is simplicity, and it is called “extreme” because it pushes all good methods, called practices in the terminology of XP, to the extreme. For example, she advocates frequent and automatic tests. The evolutionary development is also carried out extremely. Firstly, the system integration takes place frequently (at least once a day) so that a fully functional system is always available, even if it does not meet all requirements in the intermediate versions; however, this means that the system can be tested at any time. Second, the development is based on short cycles (about three weeks) like the spiral model (Fig. 2.7). Third, there are customer versions (releases) of what every few months means that the customer must be involved in the process at all times (and not just in the initial phase as in the waterfall model); A customer representative who is available on site is one of the basic requirements of extreme programming. Fourth, the agile principle of pair programming is required. In addition to simplicity, effective communication and instant feedback are basic principles of XP. To enable effective communication, all developers sit in one room, and representative users and technical experts are closely involved in the development; Instead of using formal documents, communication takes place, if possible, in direct conversations and through the source text (inline documentation). Immediate feedback is achieved through consistent pair programming on the coding level and through the repeated tests on the functional level. The process model of XP is evolutionary and includes refactoring in every iteration step, by which one understands all measures by which the quality of the software and especially the design is improved without changing the functionalities. Refactoring’s are intended to improve the present software to such an extent that it is readable, understandable, becomes maintainable and expandable. A special work step is provided for this purpose, which does not contribute directly to the expansion of functionality and is thus, apparently, unproductive. A fundamental element in XP are the stories, scenarios that serve to describe the user requirements and are comparable to the use cases of UML. The aim of the stories is less the creation of a complete specification, but rather the description of a customer request, which is given a priority and used as a unit for the master planning. In the course of the development process, new stories can be added or existing ones refined. It is also important to specify the test case before implementation, so that the first function tests can be carried out automatically during the coding. The eXtreme programming places some social requirements on the developers and customers

Unsupervised Learning

Unsupervised machine learning algorithms infer patterns from a dataset deprived of orientation to known, or branded, outcomes. Dissimilar supervised machine learning,

unsupervised machine learning techniques cannot be straight realistic to a regression or a classification problem since you have no impression what the standards for the yield data strength be, making it unbearable for you to train the method the way you generally would. Unsupervised learning can in its place be applied for learning the fundamental building of the data [21].

Unsupervised machine learning purports to discover before nameless designs in data, but greatest of the period these decorations are deprived estimates of what supervised machine learning can attain. Moreover, meanwhile you don't know what the results must be, there is no way to control how precise they are, creation supervised machine learning additional applicable to real-world problems.

The superlative time to apply unsupervised machine learning is when you don't have data on target outcomes, like decisive a target marketplace for completely new creation that your commercial has not ever sold beforehand. Though, if you are annoying to become a healthier sympathetic of your current customer base, supervised learning is the best method [22].

Material and Methods

Feature Extraction Techniques

Entity extraction is a downsizing process in which the original raw dataset is reduced to more manageable groups for processing. A characteristic of this large amount of data is the large number of variables that require extensive use of the computation. This section introduces various methods of extracting functions.

Independent Component Analysis (ICA)

Independent component analysis (ICA) is a statistical and computational technique that identifies the hidden factors underlying a set of random variables, measurements, or signals. The CIA defines a generative model for multidimensional observable data, which is usually in the form of a large sample database. The model assumes that the data variables are linear mixtures of some unknown hidden variables and that the mixing system is also unknown. It is assumed that hidden variables are not external and independent of each other and are called independent components of the observed data. CA can find these standalone components, also known as a source or agent. External CLI for principal component analysis and factor analysis. However, ICA is a more robust method and can be used to find the primary factor or source when these traditional methods fail completely. The data that ICA analyzes can come from a wide variety of applications including digital images, document databases, economic indicators, psychological metrics, and more. In many cases the measurements are provided as a series of parallel signals or time series. The term blind source sharing is used to describe this problem. Typical examples are the simultaneous audio signals that are recorded by several microphones, the interfering radio signals that are

received by EEG mobile phones that have been recorded by different sensors, or the mixing of parallel time series in industrial processes [23], [24].
Isomap Isomap means isometric view. Isomap is a spectral theory-based method for reducing non-linear dimensions, which requires the acquisition of lower dimensional geodesic distances. Isomap begins building a network of neighbors. Then use the distance from the chart to the approximate geodesic distance between every pair of points. You can find lowdimensional insertions in your dataset by decomposing the eigenvalues of the geodesic distance matrix. For nonlinear variants, the Euclidean distance metric is valid only if the neighborhood structure can be approximated as linear. The Euclidean distance can be very difficult if there are holes nearby. On the other hand, if you measure the distance between the two points behind the collector, the distance between the two points will be closer. Let's understand this using a very simple 2D example [25],[26].

Latent Semantic Analysis (LSA)
Latent semantic analysis (LSA) derived from multi-dimensional vector representations based on analyzes of large bodies. However, LSA uses a fixed pop-up window (e.g. at the paragraph level) to perform a full body conjoint analysis. The factor analysis procedure (decomposition into singular values) is then applied to the exchange matrix to obtain a narrow range of indicators (generally 300-500). This size reduction ensures that words that are used in the same context have the same vectors even if they are not used in the same context. For example, "house" and "house" usually appear with "roof", but they rarely appear together in the same language. However, they have a similar vector representation in the LSA. LSA can group individual word vectors together to provide a measure of the semantics of a larger unit of text. Therefore, the meaning of a paragraph is the sum of the vector words in that paragraph. You can use this criterion to compare large blocks of text, for example to compare the meaning of sentences, paragraphs or entire documents. Figure 3.3 shows the LSA process[27].

Partial least Squares (PLS)
PLS was originally developed using ultrasonic technology in the 1960s and is now used in many chemical measuring instruments. There are actually two versions of the PLS algorithm (PLS-1 and PLS-2). This can be confirmed under experimental conditions by simultaneous regression of the PLS and PCA type. The advantage of this method is that the experimenter can create a new explanatory variable that contains the same information as the original explanatory variable. Like polymerase chain reaction (PCR), PLS is one of the most widely used spectroscopic methods for multivariate synthesis.

However, this is a stepwise process if there is no PLS regression step. Instead, PLS analyzes the spectral data and focuses on the data at the same time. For example, to determine PC parameters, only spectra are taken into account in the original PCR analysis. When using PLS, analyte concentrations are also included in calibration samples. The factors are presented in such a way that it is possible to better clarify the evolution of the contained substances with the help of the PC, which must be determined [29].

Principal Component Analysis (PCA)

It is in Hotelling (1933) that was published in PCA Psychology of Education as we know it today. Placement shows intended use with factor analysis over the years. It has since been used in many other fields, including life sciences, natural sciences, and engineering.

The main goal is to reduce the size of a multidimensional data set and clarify its interpretation by identifying fewer variables than summarizing a larger data set in some form [31]. The starting point for PCA is the matrix of correlation coefficients obtained from the original dataset. In fact, the justification of this method requires an estimation of the correlation of the measured variables on a continuous scale. In fact, for example, this variable can be completely continuous. They are separate, for example height, weight, or possibly as arithmetic variables. It is also handled well and can be called meters (i.e. Major majors). This method can be used in cases where the correlation coefficient is not calculated using a scale variable. This analysis is not permitted, but it can give you a rough idea of the structure of your data.

Auto-Encoders

Auto-encoders are an uncontrolled learning technique in which we use neural networks to teach representation. In particular, we will design the architecture of a neural network to create a network bottleneck that will lead to a concise presentation of knowledge on the source data. If the input properties were independent of each other, this compression and subsequent retrieval would be a very difficult task. However, if there is some kind of structure in the data (i.e. correlations between the input properties), this structure can be studied and then used to force the entry through a bottleneck. Network restriction [33,34].

Classification Techniques

Classification is a central topic of machine learning that relates to machine learning to group data according to specific criteria. Classification is the process by which computers group data based on specified characteristics. This is called supervised learning. There is an automatic version of the classification called grouping, which has

common functions on computers that allow you to group data when categories are not specified. A typical classification example is spam detection. To write a spam filtering program, a programmer can train a machine learning algorithm using a series of emails that resemble spam emails marked as spam. The idea is to create an algorithm that can learn the properties of spam from this tutorial so that you can filter spam when new messages arrive. In this section several classification techniques presented:

Deep Neural Network (DNN)

A Multi-Layer Perceptron with growing quantity of layers is referred to as a Deep Neural Network with the time-honored subject being referred to as Deep Learning (DL) which has a focus on the lookup of excessive "depth" hierarchical illustration getting to know systems. This cross into deep fashions used to be pushed by using the trouble regular computing device learning (ML) techniques confronted which required deliberate statistics manipulations and preprocessing to attain applicable facets to be supplied as an enter to "shallow" fashions and gain good practical performance, that resulted in a requirement of analyzing statistics with a extraordinary deal of area information in order to extract appropriate enter vector options and with exponential increase in records dimension the effort turns into exponential brought to that the range of problems that are challenging to formalize efficaciously such as consciousness of spoken phrases or recognition of faces in images. Artificial neural networks (ANNs) Inspired by the processing of information in biological systems and distributed communication nodes.

RNA is very different from a biological brain. Neural networks are usually static and symbolic, while the biological brain of most organisms is dynamic (plastic) and the like.The reason deep learning is poor quality is that it uses multiple layers in the network. The first task showed that a linear sensor cannot be a general classifier, and an invisible layer can have an infinite width of networks of admissible non-polypolic functions. Deep learning is a modern option that handles unlimited levels of limited size and allows for practical application and optimal implementation while maintaining theoretical diversity under normal conditions. In deep run, levels are also very different from biofeedback and communication-oriented models, which can be patchy and have parts "structured" for efficiency, forgiveness, and understanding.

Support Vector Machine (SVM)

The goal of SVM is to find a hyperplane in an N-dimensional space (N is the number of faces) that uniquely classifies data points. You can select several imaginable hyperplanes to discrete the two lessons of information opinions. Our goal is to find the level with the maximum reach, or H. The maximum distance between the data points of the two classes.

The edge distance optimization offers a certain gain and enables a more reliable classification of future data points [38]. In Figure 3.6 above, we notice that there are two classes of observations: the blue points and the purple points. There are tons of ways to separate these two classes as shown in the graph on the left. However, we want to find the “best” hyperplane that could maximize the margin between these two classes, which means that the distance between the hyperplane and the nearest data points on each side is the largest. Depending on which side of the hyperplane a new data point locates, we could assign a class to the new observation. It sounds simple in the example above.

However, not all data are linearly separable. In fact, in the real world, almost all the data are randomly distributed, which makes it hard to separate different classes linearly. dual instead as in Equation 4 with respect to , subject to the constraints that the gradient of L(w, b,) with respect to the primal variables w and b vanish and ≥ 0. The primal variables are eliminated from L(w, b,).

When solve x i can get and can classify a new object x using Equation 5. Note that the training vectors xi occur only in the form of dot product; there is a Lagrangian multiplier i for each training point, which reflects the importance of the data point. When the maximal margin hyper-plane is found, only points that lie closest to the hyper-plane will have i > 0 and these points are called support vectors. All other points will have i > 0.

Decision Trees (DT)

DT is a prediction model that expresses the characteristic space in a recursive partition into subspaces and forms the basis of prediction. DT is a direct root tree. In DT, the nodes with protruding edges are internal nodes. All other nodes are DT nodes or leaves. DTs are classified according to a hierarchical set of characteristic decisions. The decision made at the internal node is the distribution criterion. In DT, each leaf is assigned to a class or its probabilities. Small deviations in the drive unit lead to different units leading to different DTs. Therefore, the error contribution due to variance is important to DT. Collaborative learning, described in the next section, helps reduce errors due to deviations [40].

Naïve Bayes Classifiers

This classification method is based on the statute and requires independence between 予 測子. Simply put, it is assumed that the existence of a certain characteristic of the naïve class of Bayesian classifiers is independent of the availability of other features. Apples, for example, are red berries that are about 3 inches in diameter. Regardless of whether these signs are interrelated or dependent on other signs, these signs independently affect the likelihood that all of these fruits are apples and thus "slob".

Known as the Naive Bayesian model, it's easy to create and is especially useful for large amounts of data. The simple Bayesian approach not only overcomes simplicity but also very complex classification methods [42].

K-Nearest Neighbors (KNN)

KNN is a simple and easy-to-manage machine learning algorithm that can solve classification and regression problems. The ANN algorithm assumes that these objects are very close to each other. I mean these things are related to each other. In most cases, similar data points are located next to each other, so take a look at the image above. The ANN algorithm assumes that this assumption is a very valid and useful algorithm. KNN calculates the distance between points on a graph and

Gradient Descent

In gradient descent, Batch is the total number of samples that you will use to calculate the gamut in one iteration. So far we've assumed this is the complete package. When working with Google, newspapers typically have billions, if not hundreds of billions of examples.

In addition, Google records often contain multiple functions. So the party can be huge. It can take a long time to calculate the jackpot. A large data set with randomly selected examples can contain redundant data. Indeed, as the size of the packets increases, the repetition potential increases. Some redundancy can be useful to compensate for noise gradients, but large beams generally do not have much higher predictive value than large beams. What if we could get the correct mean gradient for a much smaller calculation? By choosing random examples from our dataset, we were able to (albeit aloud) estimate a high mean from a much smaller average. Stochastic Regression (SGD) takes this idea to the extreme: Use an example (batch size 1) to iterate. SGD works with a large number of iterations but is very powerful. The term "stochastic" indicates that the example that contains each batch is selected at random [46], [47].

Random Forests

This assumes that you are familiar with creating your own classification tree. Many tree species grow in random forests. Place the input vector in each window in the forest to classify new features based on the input vector. Each tree offers a classification and the tree can "vote" on this category. The forest chooses the rank with the highest number of votes (among all trees in the forest). To understand and use different parameters, it is helpful to know more about how they are calculated. Most of the setup is based on 2 data objects generated from a random forest. If the training set is removed from the current tree using an alternate model, approximately one-third of the cases will be excluded from the sample. This collected data is used to objectively

assess classification errors when adding trees to the forest. It is also used to get estimates of various values. After each tree is created, all tree data and calculated similarities are run for each pair of observations. If two states occupy the same endpoint, the convergence increases by one. At the end of the analysis, it is normalized by dividing by the number in the similarity tree. The technique is used to create a small and attractive data representation that replaces missing data and detects anomalies [48].

Discriminant Analysis (DA)

The discriminant analysis (DA) is mentioned often. Compare with other credit check methods. This approach is a classification. The criteria are based on instances with known categories and predictions of unknown categories based on these criteria. DA is parametric or non-parametric. Parametric AD builds a model with some assumptions about the distribution of instances (e.g. a normal distribution). However, the model is unbalanced due to the unobservable distribution. This parametric DA is not very common and DA performance is poor compared to nonlinear DA problems. In contrast, there is a non-parametric AD in the context of a credit rating. This approach examines the distribution of instances using nonparametric constructs and methods. Classification criteria Therefore the results of the nonparametric AD are more robust [50].

Optimization Algorithms

Optimization is everywhere and therefore it is a broadband paradigm. Applications We are still testing almost all technical and industrial applications. Optimize anything to minimize and maximize costs and energy consumption. Profit, production, productivity and efficiency. In fact, resources, time and money are always limited. Therefore, in practice, optimization is much more important. Optimal use of all kinds of available resources requires a paradigm shift in scientific thinking, since most real-world applications are much more complex factors and parameters that affect the behavior of the system. Modern technical design relies heavily on computer modeling. These This leads to further optimization difficulties. Growing demand for precision electronics. The growing complexity of projects and systems leads to a modeling process. More and more time. Evaluation of a project in many areas of mechanical engineering. This can take several days or even weeks. Any method that can speed up the simulation. The optimization process and time can save time and cost [52], [53].

Bayesian Optimization Algorithm (BOA) Bayesian optimization is a sequential design strategy for global optimization of black-box functions that does not assume any functional forms [54,55]. It is usually employed to optimize expensive-to-evaluate functions. Bayesian optimization is typically used on problems of the form

$maxx \in A f(x)$, where A is a set of points whose membership can easily be evaluated. Bayesian optimization is particularly advantageous for problems where $f(x)$ is difficult to evaluate, is a black box with some unknown structure, relies upon less than 20 dimensions, and where derivatives are not evaluated. Since the objective function is unknown, the Bayesian strategy consists of treating it as a random function and placing it a priori on it. The previous one captures ideas about the behavior of a function. After collecting estimates of the functions considered data, the previous ones are updated to create a background distribution for the objective function. After the submission, a data collection function is created (often called a padding criterion) that in turn determines the next polling point.

A NEW SOFTWARE cost estimation FRAMEWORK BASED ONSVM AND BOA

In this section, new framework based on Bayes naïve and factor analysis applied for software cost estimation predication. The BOA applied to optimize the structure of the factor analysis and feature Bayes naïve classifier. The main challenge of this study is how can we present the problem as objective function which can be understand by optimization algorithm BOA. The objective function of this problem

The function 3.1 is used for selecting features which applied as feature selection objective function. Furthermore, ACC applied as to increase the Acc then, our method introduced as multi objective function. This mean maximum is best which BOA remain trying the to find the best parameters right to find the maximum Acc.

References

1.B. Suresh Kumar, B.V.V. Siva Prasad, Sonali Vyas, Combining the OGA with IDS to improve the detection rate, Materials Today: Proceedings, 2020, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.09.540.

2.Gorby Kabasele Ndonda, Ramin Sadre, Network trace generation for flow-based IDS evaluation in control and automation systems, International Journal of Critical Infrastructure Protection, Volume 31, 2020, 100385, ISSN 1874-5482, https://doi.org/10.1016/j.ijcip.2020.100385.

3.Shuokang Huang, Kai Lei, IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks, Ad Hoc Networks, Volume 105, 2020, 102177, ISSN 1570-8705, https://doi.org/10.1016/j.adhoc.2020.102177.

4.Punam Bedi, Neha Gupta, Vinita Jindal, Siam-IDS: Handling class imbalance problem in Intrusion Detection Systems using Siamese Neural Network, Procedia Computer Science, Volume 171, 2020, Pages 780-789, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.04.085.

5.Ningthoujam Chidananda Singh, Avinash Sharma, Resilience of mobile ad hoc networks to security attacks and optimization of routing process, Materials Today:

Proceedings, 2020, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.09.622.
6.Alberto Urueña López, Fernando Mateo, Julio Navío-Marco, José María Martínez-Martínez, Juan Gómez-Sanchís, Joan Vila-Francés, Antonio José Serrano-López, Analysis of computer user behavior, security incidents and fraud using Self-Organizing Maps, Computers & Security, Volume 83, 2019, Pages 38-51, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2019.01.009.
7.Mohamed Amine Rguibi, Najem Moussa, Hybrid Trust Model for Worm Mitigation in P2P Networks, Journal of Information Security and Applications, Volume 43, 2018, Pages 21-36, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2018.10.005.
8.Yiping Guo, Credit risk assessment of P2P lending platform towards big data based on BP neural network, Journal of Visual Communication and Image Representation,Volume 71, 2020, 102730, ISSN 1047-3203, https://doi.org/10.1016/j.jvcir.2019.102730.
9.Massimo Cafaro, Italo Epicoco, Marco Pulimeno, Mining frequent items in unstructured P2P networks, Future Generation Computer Systems, Volume 95, 2019,Pages 1-16, ISSN 0167-739X, https://doi.org/10.1016/j.future.2018.12.030.
10.Congjun Rao, Ming Liu, Mark Goh, Jianghui Wen, 2-stage modified random forest model for credit risk assessment of P2P network lending to “Three Rurals” borrowers,Applied Soft Computing, Volume 95, 2020, 106570, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2020.106570.
11.Hui-Kai Su, Jian-Ting Pan, Kim-Joan Chen, Marek R. Ogiela, Hsing-Chung Chen, Real-Time Streaming Relay Mechanism for P2P Conferences on Hierarchical Overlay Networksb, Journal of Electronic Science and Technology, Volume 17, Issue 3, 2019, Pages 242-251, ISSN 1674-862X, https://doi.org/10.11989/JEST.1674-862X.71018158.
12.M. Imran Azim, Wayes Tushar, Tapan K. Saha, Investigating the impact of P2P trading on power losses in grid-connected networks with prosumers, Applied Energy, Volume 263, 2020, 114687, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.114687.
13.Mark Jenkins, Ivana Kockar, Impact of P2P trading on distributed generation curtailment in constrained distribution networks, Electric Power Systems Research, Volume 189, 2020, 106666, ISSN 0378-7796, https://doi.org/10.1016/j.epsr.2020.106666.
14.Rasool Azimi, Hedieh Sajedi, Mohadeseh Ghayekhloo, A distributed data clustering algorithm in P2P networks, Applied Soft Computing, Volume 51, 2017, Pages 147-167, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2016.11.045.
15.Morteza Babazadeh Shareh, Hamidreza Navidi, Hamid Haj Seyyed Javadi, Mehdi HosseinZadeh, Preventing Sybil attacks in P2P file sharing networks based on the

evolutionary game model, Information Sciences, Volume 470, 2019, Pages 94-108, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2018.08.054.
16. Sateesh Kumar Awasthi, Yatindra Nath Singh, Simplified Biased Contribution index