DOI 10.51885/1561-4212_2025_3_268
IRSTI 28.23.13

MODERN TECHNOLOGIES AND SOFTWARE SOLUTIONS FOR WORKING WITH BAYESIAN NETWORKS

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ И ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ РАБОТЫ С БАЙЕСОВСКИМИ СЕТЯМИ

БАЙЕС ЖЕЛІЛЕРІМЕН ЖҰМЫС ІСТЕУГЕ АРНАЛҒАН ЗАМАНАУИ ТЕХНОЛОГИЯЛАР МЕН БАҒДАРЛАМАЛЫҚ ШЕШІМДЕР

A. Shayakhmetova [1]*, **A. Abdildayeva** [1], **A. Akhmetova** [1]
[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan
*Corresponding author: Assem Shayakhmetova, e-mail: asemshayakhmetova@mail.ru

Keywords:
modern algorithms, software products, software defect prediction, Bayesian network classification, machine learning.

ABSTRACT
This article examines the study of the modern market of algorithms and software products for working with Bayesian networks. One of the most important problems is the prediction of software flaws, which seems to be a necessary area in software development, because it helps creators to detect and eliminate difficulties before they turn into costly and difficult to implement errors. Early detection of software flaws focuses on saving time and money in the software development process and guarantees the nature of the final product. The purpose of this study is to analyze three algorithms of Bayesian network theory to classify whether a project is subject to defects. The selection is based on the fact that the most commonly used layout in the literature is naive Bayesian, but no Bayesian networks are used in any work. Thus, K2, Hill Climbing and TAN are used to build Bayesian networks. The results of various performance indicators used for cross-validation show that the results of systematization are comparable to a tree of conclusions and a disorderly forest, with the advantage that Bayesian algorithms show the least variability, which orients the technical software to have tremendous reliability in its forecasts, because the selection of training and testing information does not give unstable results.

Түйінді сөздер:
заманауи алгоритмдер, бағдарламалық өнімдер, бағдарламалық жасақтама ақауларын болжау, Байес желілері, классификация, Машиналық оқыту.

ТҮЙІНДЕМЕ
Бұл мақалада Байес желілерімен жұмыс істеуге арналған алгоритмдер мен бағдарламалық өнімдердің заманауи нарығын зерттеу қарастырылады. Ең маңызды мәселелердің бірі-бағдарламалық жасақтаманың кемшіліктерін болжау, бұл бағдарламалық жасақтаманы әзірлеудің қажетті саласы болып көрінеді, өйткені ол әзірлеушілерге қымбат және қиын қателіктерге айналғанға дейін қиындықтарды анықтауға және шешуге көмектеседі. Бағдарламалық жасақтама ақауларын ерте анықтау бағдарламалық жасақтаманы әзірлеу кезінде уақыт пен ақшаны үнемдейді және соңғы өнімнің сапасына кепілдік береді. Бұл зерттеудің мақсаты-жобада ақаулар бар-жоғын анықтау үшін Байес желілері теориясының үш алгоритмін талдау. Таңдау әдебиетте ең

көп қолданылатын схема аңғал Байес желісіне негізделген, бірақ ешқандай жұмыста Байес желілері қолданылмайды. Осылайша, K2, Hill Climbing және TAN Байес желілерін құру үшін қолданылады. Қиылыспалы тексеру үшін қолданылатын әр түрлі тиімділік көрсеткіштерінің нәтижелері жүйелеу нәтижелерін қорытынды ағашпен және ретсіз орманмен салыстыруға болатындығын көрсетеді, өйткені Байес алгоритмдері ең аз өзгергіштікті көрсетеді, бұл техникалық бағдарламалық жасақтаманы өз болжамдарында үлкен сенімділікке бағыттайды, өйткені оқыту мен тестілеу әдістерін таңдау олардың қаншалықты тиімді жұмыс істейтініне байланысты ақпарат тұрақсыз нәтиже бермейді.

Ключевые слова:
современные алгоритмы, программные продукты, прогнозирование дефектов программного обеспечения, байесовские сети, классификация, машинное обучение

АННОТАЦИЯ

В данной статье рассматривается изучение современного рынка алгоритмов и программных продуктов для работы с байесовскими сетями. Одной из наиболее важных проблем является прогнозирование недостатков программного обеспечения, что, по-видимому, является необходимой областью в разработке программного обеспечения, поскольку помогает разработчикам обнаруживать и устранять трудности до того, как они превратятся в дорогостоящие и трудноосуществимые ошибки. Раннее выявление дефектов программного обеспечения позволяет сэкономить время и деньги в процессе разработки программного обеспечения и гарантирует качество конечного продукта. Цель этого исследования - проанализировать три алгоритма теории байесовских сетей, чтобы определить, есть ли в проекте дефекты. Выбор основан на том факте, что наиболее часто используемой в литературе схемой является наивная байесовская сеть, но ни в одной работе байесовские сети не используются. Таким образом, K2, Hill Climbing и TAN используются для построения байесовских сетей. Результаты различных показателей эффективности, используемых для перекрестной проверки, показывают, что результаты систематизации сравнимы с деревом выводов и беспорядочным лесом, с тем преимуществом, что байесовские алгоритмы демонстрируют наименьшую вариабельность, что ориентирует техническое программное обеспечение на огромную надежность в своих прогнозах, поскольку выбор методов обучения и тестирования зависит от того, насколько эффективно они работают и информация не дает нестабильных результатов.

INTRODUCTION

The existence of software flaws seems to be a huge inconvenience and inconsistency in the development and maintenance of software, therefore having a negative impact on the property of the software. It is impossible to detect software that does not include defects, even despite the scrupulous course of software development. That's why checking software supply seems to be a crucial step in the software development lifecycle, because it is a way to avert or even repair conceivable software outages before it activates to function. However, the process associated with software testing is naturally complicated, because it requires excellent planning and the greatest number of resources (Meiliana, S.K., Karim, S., Warnars, H.L, Gaol, F.L., Abdurachman, E., Soewito, B., 2022). Software supply deficiencies dramatically affect productivity, quality, costs, and user satisfaction. Some of the mostly common results of the presence of a large number of software damages include delays in product delivery, unnecessary or sudden expenses, poor overall user experience, loss of customer confidence, and even security difficulties. All these consequences show a relaxed impact on the quality of the software.

The purpose of this study is to analyze three Bayesian network theory algorithms and apply them to early detection of software defects, which saves time and money during the development process.

The novelty of the research lies in the comparative analysis of the K2, Hill Climbing and TAN algorithms for building Bayesian networks in the problem of software defect prediction. The paper uses formal statistical methods to assess the stability and reliability of models, which increases the scientific validity of the results. Bayesian algorithms are also highly resistant to the variability of training data, which is especially important for limited and incomplete samples.

LITERATURE REVIEW

Taking into account the adverse consequences that appear when software flaws are detected at the last stages of software development, a software defect modeling site (SDP) appears, "in which a modeling modification is formed in order to predict existing software interruptions based on significant data" Therefore, the prediction of flaws is necessary to identify probably insufficient modules in the software in order to it may have been relevant to acquire an effective, an error-free software product with a not very high cost. When it is possible to detect modules subject to defects, it would be possible to allocate money and human resources to prevent unexpected costs. Developing a modification for modeling software flaws is not an easy task. Actually, artificial intelligence plays a role here, using machine learning algorithms (ML), possibly helping to develop software to predict the shortcomings of software provision at early stages. The article (Hammanouri, A., Hammad, M., Alnabhan, M., Alsarayrah, F., 2018) discusses individual classifiers based on machine learning algorithms, such as naive Bayes (NB), inference trees (DT) and artificial neural networks (ANN) for modeling software flaws.

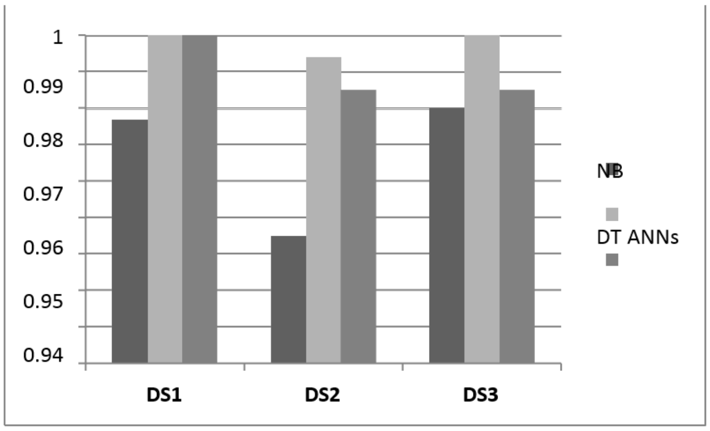

Figure 1. F-measurement values for the ML algorithms used in three data sets [2]
Note – taken from (Hammanouri, A., and all., 2018)

To compare the three classifiers in terms of memorization features and accuracy, we used the F-measure property. Figure 1 shows the F-measure values for the ML algorithms used in three data sets. As shown in the figure, DT has the most exalted property of F-measure in all data sets, followed by ANS, then the classifiers of NB.

There are some alternatives that can provide the best results in terms of accuracy, such as algorithms based on Bayesian approaches that "solve many issues in different areas; from disease prediction/patient treatment to analysis of genetic maps or expression analysis" (Misirli, A., Bener, A.B., 2014).

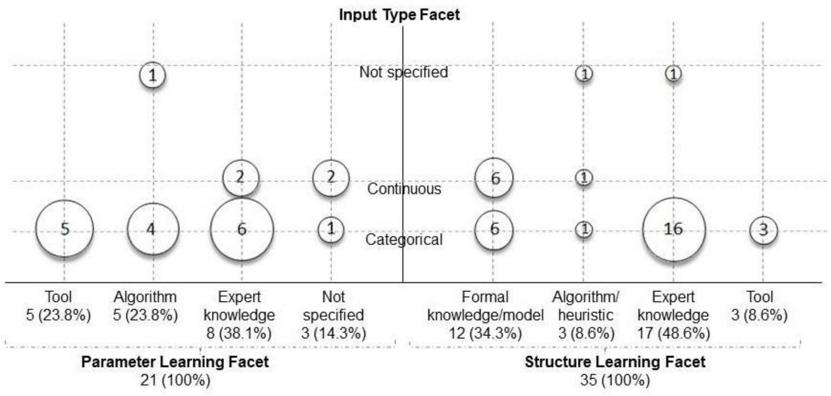

Figure 2. Bubble graph summarizing the classification results [3]
Note – taken from (Misirli, A., and all., 2014)

Figure 2 presents a bubble graph proving the result of this cartographic study on modeling software features. Filtering studies using only one way to study the structure and parameters, and plotting the relationship between the three aspects.

It is mentioned in (Herzing, K., Just, S., Zeller, A., 2013) that during a manual study of seven thousand reviews of problems from the databases of errors provided by five open source plans, it was found that 33.8% of all reports were unnaturally classified because there were no defects in them, but instead they referred to a new function, development or refactoring. Consequently, this affected the prediction of software flaws, because it depends on the properties of the data being evaluated. When the real information is incorrect, it is only possible to achieve an elevated percentage of prediction accuracy. Considering all of the above, it attempts to study the productivity and reliability of algorithms based on Bayesian networks (poorly studied algorithms), which allow a software engineer to be more convinced when making estimates and delivering a better product.

Table 1. Impact of misclassified issuerReports on mapping strategies and approaches

Measure	HTTPClient	Jackrabbit	Lucene-Java	Rhino	Tomcat5	Average
MappingBiasRate (False positive rate for mappable BUG reports)	24%	36%	21%	38%	28%	29%
DiffBugNumRate (How many files will change their defect-prone ranking?)	62%	17%	14%	52%	39%	37%
MissDefectRate (How many files wilth no original BUG have at least one classified BUG?)	1%	0.3%	0.7%	0%	38%	8%
FalseDefectRate (How many files with at least one original BUG have no classified BUG?)	70%	43%	29%	32%	21%	39%
Note – compiled by the author based on data from Herzing, K., and all. (2013)						

In (Hernández-Molinos M.J., Sánchez-García Á.J., Barrientos-Martínez R.E., 2021) 38 studies were presented on predicting software flaws for analyzing mainly used approaches and classification algorithms in this area. This study shows that the most used approaches are composite algorithms, such as Random Forest (Li R. Zhou, L. Zhang S., Liu H., Huang X., Sun Z., 2019; Aydin Z.B.G., Samli, R., 2020), followed by other algorithms, such as AdaBoost (Goyal S., 2020) and Bagging (Aljamaan H., Alazba A., 2020), among others.

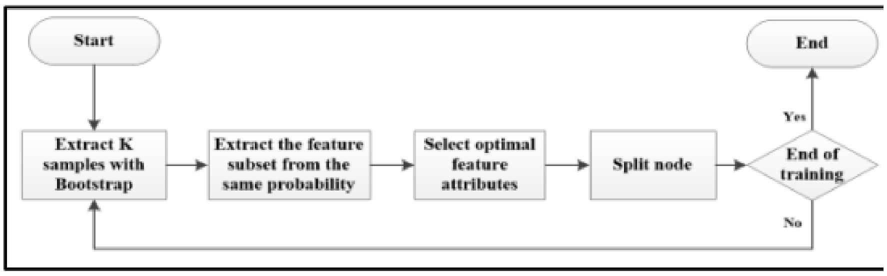
Start
End
Extract K samples with Bootstrap
Extract the feature subset from the same probability
Select optimal feature attributes
Split node
End of training
Yes
No

Figure 3. The approaches used are composite algorithms [6,7,8,9]
Note – taken from (Goyal S., 2020)

The Unexpected Forest algorithm is an ensemble algorithm that can be generated in parallel. The key idea of RF is to use the conclusion tree as a basic classifier, the concept of many conclusion trees using Bootstrap technology, and then enter the samples provided for classification, giving the final results of systematization through voting. It is possible to see that the advantages of the RF algorithm are mainly contained in the excellent strength of the noise values and the missing information about the values in the data sets of the software prediction of defects, and the RF algorithm has a bad scalability of the systematization of the modeling of disadvantages for the provided high dimensionality. Figure 4 shows the course of studying a single tree of conclusions of a disordered forest. At the same level as the latter approach, approaches based on Bayes' theorem were also conditioned. Although they all used the naive Bayes algorithm and its variants (Ge J., Liu J., Liu W. 2018; Prahba C.L., Shivahumar N., 2020).

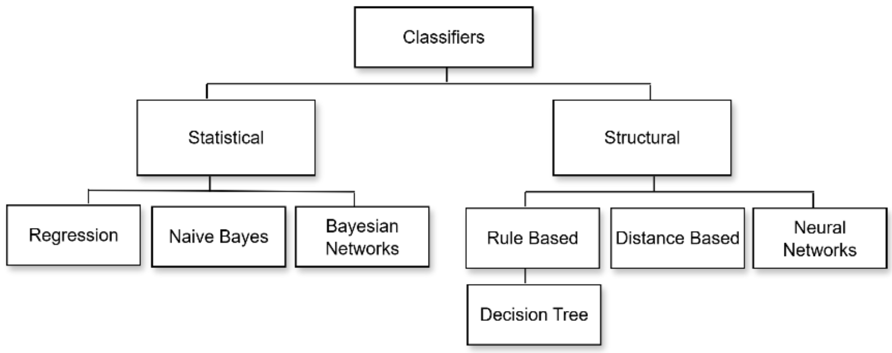

Figure 4. Naive Bayes algorithm and its variants
Note – taken from (Ge J., and all., 2018)

There are also some approaches, such as decision trees; in particular, C4. 5 is the algorithm that has been reported the most (Ge J., Liu J., Liu W., 2018). In addition, there are various more elementary classifiers, such as the method of support vectors (Ahmed M.R., Ali M.A., Ahmed N., Zamal M.F.B., Shamrat F.M.J.M., 2020), K-nearest neighbor (Zhou Y., Shan C., Sun S., Wei S., Zhang S., 2019) and logistic regression (Nehi M.M., Fakhrpoor Z., Moosavi M.R., 2018; El-Shorbagy S.A., El-Gammal W.M., Abdelmoez W.M., 2018).

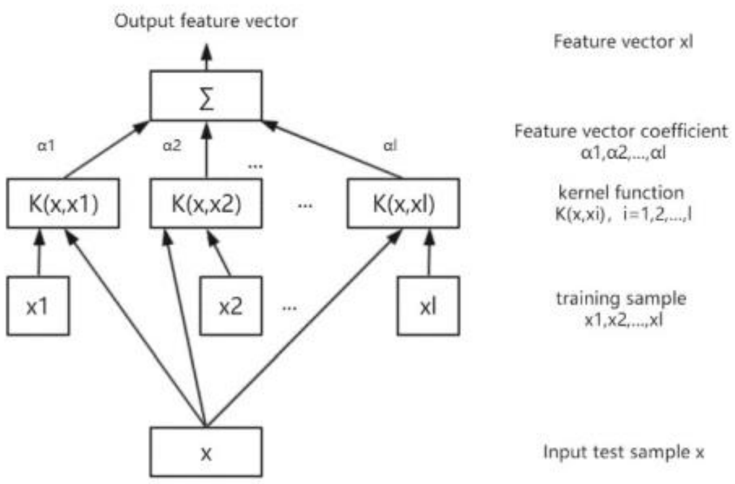
Output feature vector
Σ
α1
α2
αl
K(x,x1)
K(x,x2)
K(x,xl)
x1
x2
xl
x
Feature vector xl
Feature vector coefficient α1,α2,...,αl
kernel function K(x,xi), i=1,2,...,l
training sample x1,x2,...,xl
Input test sample x

Figure 5. K is the nearest neighbor
Note – taken from (Nehi M.M. and all., 2018)

More recently, some works have been found in which all possible approaches are equal (Bhutamapuram U.S., Sadam R., 2022; Goyal S., 2022). These comparisons are laid in the key between assembly methods, where classification is done with the support of many algorithms, and technologies with traditional approaches. There are even such works as (Goyal S., 2022), where they experiment with different versions of the algorithm, such as the main vector machine (SVM).

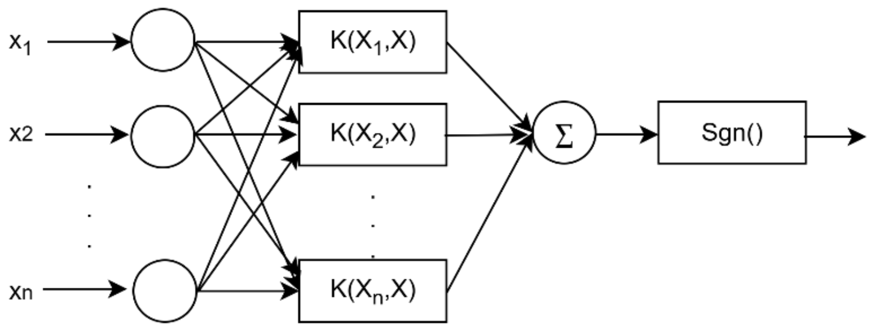

Figure 6. The principle of systematization of KPCA-SVM modification
Note – taken from (Goyal S., 2022)

The principle of systematization of the KPCA-SVM modification is shown in Fig. 6 represents the selection point after the size reduction, and K represents the kernel function.

First, a sample of a reduced dimension indicates the input function of the kernel and is compared with a location of a huge dimension. Subsequently, a suitable hyperplane of classification is found and the result is introduced.

The purpose of this study is to study the modern market of algorithms and software products for working with Bayesian networks.

MATERIALS AND METHODS

As shown in the works listed in Table 1, most of the works demonstrate an exceptionally perfect result. This is important because the machine learning algorithms used can be impressionable to training information and do everything in comparison with test data. When the test data is identical to the original training data, the best results will be obtained. According to this factor, it is necessary to notify about the performance of classification algorithms. Thus, the reliability of the classification algorithm is not only sufficient, but it is also resistant to various data sets. Cross-validation operations are simply used to create models with different training inputs from a dataset and evaluate them using a variety of test subsets of the dataset that are not used for training.

Table 1. Summary and comparison of our research with current recent research (NR = Not reported, DT = Decision Tree, EM = Ensemble Methods, NB = Naive Bayes, NN = Neural Networks, KNN = K-nearest neighbor, SVM = Support Vector Machine, LR = Logistic regression and BN = Bayesian network)

Ref	Year	Approaches Used	Bayesian Networks Use	Performance Analysis	Comparison with Other Similar Approaches or Variants	Comparison with Other Approaches
Li R. Zhou, L. Zhang S., Liu H., Huang X., Sun Z.	2019	DT, EM	NO	NO	YES	YES
Aydin Z.B.G., Samli, R.	2020	DT,EM	NO	NO	NO	YES
Aljamaan H., Alazba A.	2020	EM	NO	YES	YES	NO
Ge J., Liu J., Liu W.	2018	NB, DT, EM, SVM	NO	NO	NO	YES
Prahba C.L., Shivahumar N.	2020	NB, DT, NN	NO	YES	NO	YES
Ahmed M.R., Ali M.A., Ahmed N., Zamal M.F.B., Shamrat F.M.J.M.	2020	DT, NB, LR, EM, SVM	NO	NO	NO	YES
Zhou Y., Shan C., Sun S., Wei S., Zhang S.	2019	SVM	NO	YES	YES	NO
El-Shorbagy S.A., El-Gammal W.M., Abdelmoez W.M.	2018	EM, NN, DT	NO	NO	NO	YES
Bhutamapuram U.S., Sadam R.	2022	LR, SVM, KNN, NN, NB	NO	NO	NO	YES
Goyal S.	2022	NN,NB	NO	NO	NO	YES
Malhotra R., Meena S.	2022	LR,DT, EM	NO	NO	NO	YES
Goyal S.	2022	SVM	NO	YES	YES	NO
Our Research	2023	BN, EM, DT	YES	YES	YES	YES
Note – compiled by the author based on data from Zhou Y., and all. (2019)						

A different situation, which can be seen in Table 1, is contained in the fact that there are unequal versions of the same algorithm; although a universal version is naturally attached. Finally, when evaluating an algorithm using a conditional layout (such as a decision tree, ensemble technologies, or KNN, among others), comparisons with other approaches are not reported in many variants. Finally, the application of Bayesian networks was not found in any cited work, but the correct naive Bayes algorithm was found. We assume that the limited use of Bayesian networks in software development may be caused by a possible flaw in familiarity with this particular approach. However, this path has many advantages.

The main contribution of our study is summarized below:- In this paper, we empirically investigate the consequences of using classification algorithms based on Bayes' theorem, which are not described in the literature, when predicting software flaws, in particular, various methods of Bayesian network theory. All of the above is intended to give software engineers some strategies for modeling software flaws in their projects.- the selection of Bayesian networks is based on the fact that it is not only good for software engineers and testers to know the property of the class variable, but in addition it is essential to know the characteristic to which they should pay more attention. In addition, some algorithms, such as KNN or Random Forest, do not express variables that have the greatest impact on solving classification problems.- these studies were done in the well-known public repository PROMISE in order to work out the results renewable and comparable.- We conduct a statistical comparison of tests produced by the method of cross-testing (10-fold, as is usually done in the literature) in order to know the productivity of algorithms by comparison with a variety of initial studies and testing. These comparisons allow us to see the best and worst results, that is, how variable their indicators are.- We also compare the methods recommended in the experiment with two approaches listed in the literature: J48 (the approach of decision trees) and Random Forest (the approach of ensemble algorithms).- the comparison in this article does not cover the accuracy coefficient, since it measures only a part of well-systematized records among the total number of tests. This is important because the class values in the information sets are unbalanced, and a simple application of the correctness indicator will lead to a shift in the totals towards the class value, which is reflected mostly often. We use indicators such as recall, reliability and F1-measure, which measure all sorts of nuances of model performance.- Along with the results, a discussion was presented, allowing to coordinate the reliability and reliability of all kinds of tested methods.- this work is important because the software flaw prediction section allows the creators and managers of plans to know when the concept can be released, reducing the use of unexpected resources and improving the overall user experience by reducing the number of defects.

Table 2 shows the indicators used to evaluate the performance of classifiers and their results, among which the following are emphasized: accuracy, recall, F1-measure, thoroughness and area under the curve. This is important because the optimal productivity of the algorithm depends on the metric used, and it is essential to value it with the support of a variety of metrics, since they measure a variety of performance. In addition, some of them mitigate such problems as class imbalance or retraining. According to this factor, mainly known indicators were preferred to evaluate the results of the study. However, as shown in Table 2, numerous related works are focused exclusively on recreating the correctness of the proposal with the support of one indicator. This is somewhat suspicious, because the nature of the information means that, depending on the class, all kinds of indicators must be applied, not just percentage accuracy.

In addition, table 2 shows that many studies do not mention the model validation method. This is important because the selection of data for training and testing models affects the results obtained.

Table 2. A brief description of the indicators and verification methods reported in a recent study (NR = not reported)

Ref	Data Set	Cross Validation	Best Approach	Accuracy	Recall	F1-Measure
Li R. Zhou, L. Zhang S., Liu H., Huang X., Sun Z.	CM1	10-fold	Random Forest/Sampling	NR	NR	0.92
Li R. Zhou, L. Zhang S., Liu H., Huang X., Sun Z.	JM1	10-fold	Random Forest/Sampling	NR	NR	0.85
Li R. Zhou, L. Zhang S., Liu H., Huang X., Sun Z.	KC1	10-fold	Random Forest/Sampling	NR	NR	0.87
Aydin Z.B.G., Samli, R.	CM1	10-fold	Random Forest	0.97	0.97	0.97
Aydin Z.B.G., Samli, R.	JM1	10-fold	Random Forest	0.81	0.81	0.77
Aydin Z.B.G., Samli, R.	KC1	10-fold	Random Forest	0.83	0.85	0.83
Goyal S.	CM1	NR	Ada Boost	0.87	NR	NR
Goyal S.	JM1	NR	Ada Boost	0.89	NR	NR
Goyal S.	KC1	NR	Ada Boost	0.85	NR	NR
Zhou Y., Shan C., Sun S., Wei S., Zhang S.	CV1	10-fold	Support Vector Machine	0.88	0.89	0.82
Zhou Y., Shan C., Sun S., Wei S., Zhang S.	JM1	10-fold	Support Vector Machine	0.81	0.82	0.79
Zhou Y., Shan C., Sun S., Wei S., Zhang S.	KC1	10-fold	Support Vector Machine	0.84	0.83	0.84
Bhutamapuram U.S., Sadam R.	CM1	NR	Random Forest	0.86	NR	NR
Bhutamapuram U.S., Sadam R.	JM1	NR	Random Forest	0.34	NR	NR
Bhutamapuram U.S., Sadam R.	KC1	NR	Naïve Bayes	0.77	NR	NR
Goyal S.	CM1	NR	K-NN undersample	0.89	NR	NR
Goyal S.	JM1	NR	SVM undersample	0.93	NR	NR
Goyal S.	KC1	NR	K-NN undersample	0.96	NR	NR
Goyal S.	CM1	NR	SVM-Linear	0.79	NR	NR
Goyal S.	JM1	NR	SVM-RBF	0.88	NR	NR
Goyal S.	KC1	NR	SVM-Linear	0.83	NR	NR
Note – compiled by the author based on data from El-Shorbagy S.A., and all. (2019)						

The algorithm is obliged to make a perfect contribution to what affects the reliability, character and high value of the software. Increasing the correctness of showing software flaws implies, in turn, an increase in the productivity of developers, limiting the testing period and obtaining superiority by project managers in terms of more successful resource allocation. We focus on this approach because we emphasize individual advantages over other models. A cardinal factor in the selection of Bayesian networks is their ability to simulate cause-and-effect relationships between variables, which may help to comprehend the relationship between the functions of a software product that affect whether it is subject to defects. Secondly, according to the unanimous nature of Bayes' theorem, this composition cultivates fuzziness in the data, which can be significant when working with noisy (or discarded) or incomplete data. Thirdly, Bayesian networks are flexible and can handle all kinds of variables, starting with constant and discrete variables, which does not limit the type of data obtained with the support of software metrics. Finally, unlike other machine learning algorithms (such as K-nearest neighbor or random forest), the construction can be interpreted. This is especially important because a software engineer will be able to interpret the relationship between software attributes with the support of a graph and pay attention to those of them that reveal a negative impact on a defective product.

Bayes' theorem is a statement used to calculate the relative probability of an event. It was invented by the English mathematician and theologian Thomas Bayes. The main purpose of this theorem is to establish the possibility of an action by comparison with the possibility of another similar event. In other words, this makes it possible to know the relative possibility of an action or occurrence, conditioned as a given B, in which the direction of the possibilities of action B is analyzed when A is set (Kaur D., Sobiesk M., Patil S., Liu J., Bhagat P., Gupta A., Markuzon N., 2021).

The Bayes formula, in addition, popular, as a rule, Bayes, describes the probability of an event. There are three different probabilities in the Bayes formula, as presented in equation (1), here P(A) is the probability that imagines the a priori property of action A, P(A|B) is the probability that imagines the a priori property of action A and P(B|A) is the probability of action B based on on the information about event A.

$$P(A|B) = \frac{(P(B|A)*P(A))}{P(B)} \tag{1}$$

Bayes' theorem lies at the base of a classifier known as a Bayesian network. A Bayesian network is a graphical model that depicts unstable (usually referred to as nodes) in a set of information and probabilistic or relative relationships between them. A Bayesian network may play causal relationships with nodes; however, connections in the network (also referred to as edges) do not necessarily deliver a direct causal relationship.

On the other hand, Bayesian networks are a type of probabilistic model that applies a Bayesian solution to calculate probability. Bayesian networks are aimed at modeling relative coupling and causality through images of relative coupling using edges in a directed graph. Due to these relationships, the decision on random variables on the graph can be found and performed qualitatively with the use of coefficients.

This classifier was chosen because it represents a compact, flexible and interpretable idea of a general probability distribution. This is also important for knowledge discovery, since targeted non-periodic graphs deliver causal relationships between variables (Madden M.G., 2019). In addition, this model provides significant information about how variables are conjugated, which may be interpreted as causal relationships. Figure 7 shows the structure of the Bayesian network in the guise of an oriented non-periodic graph. This structure reproduces the relationship between variables representing relative probabilities. For example, argument C depends on variables A and B.

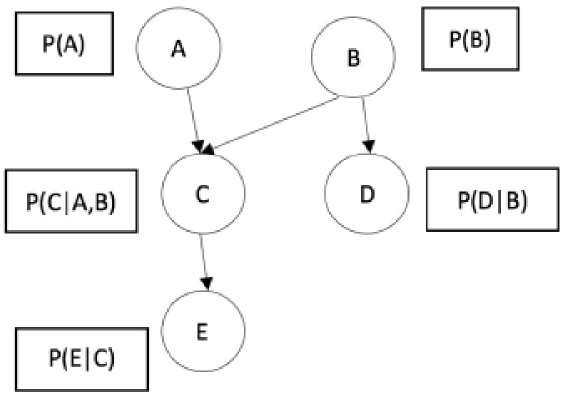

Figure 7. The structure of the Bayesian network
Note – taken from (Madden M.G., 2019)

Let's present three methods of constructing a Bayesian network.

The TAN algorithm, also commonly known as a naive Bayesian network with an extended tree, is a Bayesian network that consists of the concept of a tree of dependencies between variables that must be predicted and which, in turn, are represented as children of a variable class. Consequently, the possibility of these variables will be calculated through the use of Bayes' theorem based on the probability of a class variable (Sucar L. E., Sucar L. E., 2021). In conclusion, this assumes relative randomness between all variables set by class variables, allowing predictor variables to obey each other. Figure 2 shows how TAN bases a Bayesian network, where a class of variables has no parent elements, and objects (attributes) have a class of variables and, at most, another attribute as parent elements. Any of these variables will be calculated using the probability of a class variable based on the Bayes theorem (Sucar L. E., Sucar L. E., 2021). In conclusion, this assumes relative independence among all variables set by class variables, allowing predictor variables to depend on each other. Figure 8 shows how TAN establishes a Bayesian network, where a class of variables has no parents, and objects (attributes) have a class of variables and, at most, one more affiliation in the property.

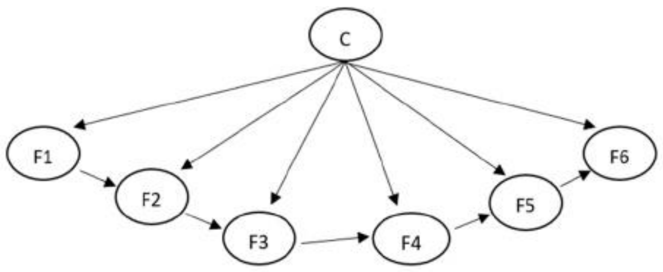

Figure 8. Bayesian network with TAN algorithm
Note – taken from (Sucar L. E., 2021)

Hill climbing is an optimization algorithm starting with a randomly generated Bayesian network (Gámez J.A., Mateo J.L., Puerta J.M., 2011). The algorithm adds or eliminates relationships for any site or object in an unexpected way, calculating the possibility of any node forming a network based on the total probability of a class variable. The algorithm selects the appropriate network with the best quality, screening out those that do not reach its level. The evaluation function, naturally used for Bayesian networks, is the value of logarithmic likelihood, which measures the possibility of the marked data taking into account the structure and parameters of the network. In other words, it measures how well the network foreshadows the data, as presented in equation (2), where S - is the estimate, G - is the network design, D - is the data, Xi is the i-th variable, and Pai is the set of parent components Xi in G.

The K2 algorithm is an approximate search that starts with the simplest possible network, a network without edges, and assumes that the sections are ordered (El-Awady A., Ponnambalam K., 2021). This algorithm applies the idea of the insatiable algorithm as the most classical structure learning algorithm (He Y.L., Zhao W.J., Xu Y., Zhu Q.X., 2021). K2 automates the flow of Bayesian network structure research, which means that huge expert knowledge in a problematic area is not required. For each variable in the problem, the algorithm adds a section with the least probability to its parent set, which leads to the greatest increase in quality, the appropriate quality of the indicator selected during the evaluation. This flow is repeated until the property is increased or an absolute Bayesian network is reached.

RESULTS AND DISCUSSION

The research of the modern market of algorithms and software products for working with Bayesian networks covers a wide range of technologies and tools used in the field of machine learning, artificial intelligence, statistics and data analytics.

Summarizing the above, the following factors can be attributed to trends and challenges in the market using Bayesian networks (Misirli, A.; Bener, A.B., 2014):

- The problems of algorithm scalability remain relevant, especially when it comes to large datasets. Algorithms for Bayesian networks must be adapted to process real-time data.
- With the increase in data volumes in various fields, the processing and training of Bayesian networks are becoming more complex tasks. This requires the creation of more efficient and faster methods for building networks.
- One of the challenges is the need to create such methods and software products that not only ensure the accuracy of forecasting, but also provide users with understandable interpretations of the result.
- Although Bayesian networks cope well with uncertainty, it is important to develop methods of working with partial or unreliable information, including methods of learning in the absence of data.
- Bayesian networks are increasingly being combined with other machine learning methods, such as neural networks, to improve the accuracy and adaptability of solutions.

Modern algorithms and software products for working with Bayesian networks are constantly evolving, offering new opportunities for effective modeling and analysis of data with uncertainty. However, scalability, performance, and interpretability remain important challenges that need to be addressed in the context of growing data volumes and increasingly complex applications. Based on this, we can safely say that more and more companies and scientific organizations are using Bayesian networks to solve complex problems in the field of AI, for example, for autonomous systems or robots, where it is important to make decisions in conditions of uncertainty (He Y.L., Zhao W.J., Xu Y., Zhu Q.X., 2021). Bayesian networks are increasingly being integrated into cloud computing and distributed systems to increase their computing power. It is expected that the development of faster and more efficient algorithms for working with large amounts of data and new technologies for training Bayesian networks will continue.

Table 1. Technical specifications of the investigated equipment

Parameter	Value	Unit of measurement
Motor power	3.5	kW
Spindle rotation speed	1500	rpm
Maximum machining diameter	250	mm
Note – compiled by the author based on data from Kulenova (2021)		

CONCLUSION

The analysis showed that the K2 and Hill Climbing algorithms provide more stable and reliable results when building Bayesian networks for predicting software defects compared to TAN. The lower variability of the estimated indicators of these methods indicates their resistance to changes in the composition of the training data, which is critical for practical use in conditions of limited or incomplete samples. Despite the fact that Random Forest demonstrates slightly higher values of quality metrics, Bayesian algorithms benefit from the consistency and reproducibility of forecasts, which makes them preferable for tasks where stability and explainability of the model are important. In addition, the revealed dependence of TAN variability on the linearity of the relationships between features highlights the need for a deeper study of the adaptation of Bayesian network structures to complex and nonlinear data dependencies. Taken together, the results confirm the promise of using Bayesian networks built using K2 and Hill Climbing algorithms for early detection of defects, which helps optimize the development process and improve software quality.

The future work suggests a study of the modern market of algorithms and software products for solving linear programming problems, as well as the development of an extended Bayesian network structure focused on the use of linear programming methods (El-Awadi A., Ponnambalam K., 2021).

Thus, in this paper, for the first time, a comprehensive comparative analysis of three algorithms for constructing Bayesian networks (K2, Hill Climbing and TAN) was carried out specifically in the context of predicting software defects, which had not previously received sufficient attention. In addition, the influence of the network structure and the nature of the relationships between features on the variability of models was identified and analyzed in detail, which opens up new directions for improving Bayesian models in software quality assurance applications.

CONFLICT OF INTEREST: The authors declare no conflict of interest.

FUNDING: This article was prepared within the framework of the grant funding project for scientific research of the Ministry Committee on Science and Higher Education of the Republic of Kazakhstan under the topic AP19679142 "Search for optimal solutions in Bayesian networks in models with linear constraints and linear functionals. Development of algorithms and programs" (2023-2025).

STATEMENT ON THE USE OF ARTIFICIAL INTELLIGENCE TECHNOLOGIES: The authors state that they did not use artificial intelligence (AI) tools in the preparation of this work.

REFERENCES

Meiliana, S. K., Karim, S., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., & Soewito, B. (2022). Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset. In Proceedings of the IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Phuket, Thailand.

Hammanouri, A., Hammad, M., Alnabhan, M., & Alsarayrah, F. (2018). Software bug prediction using machine learning approach. International Journal of Advanced Computer Science and Applications, 9, 78–83.

Misirli, A., & Bener, A. B. (2014). A mapping study on Bayesian networks for software quality prediction. In Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), Hyderabad, India.

Herzing, K., Just, S., & Zeller, A. (2013). It's not a bug, it's a feature: How misclassification impacts bug prediction. In Proceedings of 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 18–26.

Hernández-Molinos, M. J., Sánchez-García, Á. J., & Barrientos-Martínez, R. E. (2021). Classification algorithms for software defect prediction: A systematic literature review. In Proceedings of the 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT), San Diego, CA, USA, 25–29.

Li, R., Zhou, L., Zhang, S., Liu, H., Huang, X., & Sun, Z. (2019). Software defect prediction based on ensemble learning. In Proceedings of 2019 2nd International Conference on Data Science and Information Technology (DSIT), Seoul, Republic of Korea, 19–21.

Aydin, Z. B. G., & Samli, R. (2020). Performance evaluation of some machine learning algorithms in NASA defect prediction data sets. In Proceedings of the 2020 5th International Conference on Computer Science and Engineering (UBMK), Diyarbakir, Turkey, 9–11.

Goyal, S. (2020). Heterogeneous stacked ensemble classifier for software defect prediction. In Proceedings of the Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 6–8.

Aljamaan, H., & Alazba, A. (2020). Software defect prediction using tree-based ensembles. In Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), 8–9.

Ge, J., Liu, J., & Liu, W. (2018). Comparative study on defect prediction algorithms of supervised learning software based on imbalanced classification data sets. In Proceedings of the 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan, Republic of Korea, 27–29.

Prahba, C. L., & Shivahumar, N. (2020). Software defect prediction using machine learning techniques. In Proceedings of the 4th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 15–17.

Ahmed, M. R., Ali, M. A., Ahmed, N., Zamal, M. F. B., & Shamrat, F. M. J. M. (2020). The impact of software fault prediction in real-world application: An automated approach for software engineering. In Proceedings of the 6th International Conference on Computing and Data Engineering (ICCDE), Sanya, China, 4–6.

Nehi, M. M., Fakhrpoor, Z., & Moosavi, M. R. (2018). Defects in the next release; software defect prediction based on source code versions. In Proceedings of the Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 8–10.

Zhou, Y., Shan, C., Sun, S., Wei, S., & Zhang, S. (2019). Software defect prediction model based on KPCA-SVM. In Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 19–23.

El-Shorbagy, S. A., El-Gammal, W. M., & Abdelmoez, W. M. (2018). Using SMOTE and heterogeneous stacking in ensemble learning for software defect prediction. In Proceedings of the 7th International Conference on Software and Information Engineering (ICSIE), Cairo, Egypt, 2–4.

Bhutamapuram, U. S., & Sadam, R. (2022). Within-project defect prediction using bootstrap aggregation based diverse ensemble learning technique. Journal of King Saud University - Computer and Information Sciences, 34, 8675–8691. https://doi.org/10.1016/j.jksuci.2022.03.027

Goyal, S. (2022). Handling class-imbalance with KNN (neighbourhood) under-sampling for software defect prediction. Artificial Intelligence Review, 55, 2023–2064. https://doi.org/10.1007/s10462-021-10106-0

Malhotra, R., & Meena, S. (2022). Defect prediction model using transfer learning. Journal of Soft Computing, 26, 4713–4726. https://doi.org/10.1007/s00500-021-06497-7
Goyal, S. (2022). Effective software defect prediction using support vector machines (SVMs). International Journal of System Assurance Engineering and Management, 13, 681–696. https://doi.org/10.1007/s13198-021-01288-2
Kaur, D., Sobiesk, M., Patil, S., Liu, J., Bhagat, P., Gupta, A., & Markuzon, N. (2021). Application of Bayesian networks to generate synthetic health data. Journal of the American Medical Informatics Association, 28(4), 801–811. https://doi.org/10.1093/jamia/ocaa288
Madden, M. G. (2019). On the classification performance of TAN and general Bayesian networks. In Research and Development in Intelligent Systems XXV, Springer, London, UK, 3–16.
Sucar, L.E., & Sucar, L.E. (2021). Bayesian networks: Representation and inference. In Probabilistic Graphical Models: Principles and Applications, 111–151.
Gámez, J. A., Mateo, J. L., & Puerta, J. M. (2011). Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. Journal of Data Mining and Knowledge Discovery, 22, 106–148. https://doi.org/10.1007/s10618-010-0220-2
El-Awady, A., & Ponnambalam, K. (2021). Integration of simulation and Markov chains to support Bayesian networks for probabilistic failure analysis of complex systems. Reliability Engineering & System Safety, 211, 107511. https://doi.org/10.1016/j.ress.2021.107511
He, Y. L., Zhao, W. J., Xu, Y., & Zhu, Q. X. (2021). Research and improvement of K2 algorithm based on topological sorting. In Proceedings of the China Automation Congress (CAC), Beijing, China, 4623–4626.

Авторлар туралы мәліметтер
Информация об авторах
Information about authors

Шаяхметова Асем Серикбаевна – PhD, қауымдастырылған профессор, әл-Фараби атындағы Қазақ ұлттық университеті, Жасанды интеллект және BigData кафедрасының профессор м.а., Алматы қ., Қазақстан
Шаяхметова Асем Серикбаевна – PhD, ассоциированный профессор, и.о. профессора кафедры Искусственного интеллекта и Big Data Казахского национального университета им. аль-Фараби, г. Алматы, Казахстан
Shayakhmetova Assem – PhD, Associate Professor, Acting Professor of the Department of Artificial Intelligence and Big Data of Al-Farabi Kazakh National University, Almaty, Kazakhstan,
e-mail: asemshayakhmetova@mail.ru,
ORCID: https://orcid.org/0000-0002-4072-3671

Абдилдаева Асель Асылбековна – PhD, әл-Фараби атындағы Қазақ ұлттық университеті, Жасанды интеллект және BigData кафедрасының қауымдастырылған профессоры, Алматы қ., Қазақстан
Абдилдаева Асель Асылбековна – PhD, ассоциированный профессор кафедры Искусственного интеллекта и Big Data Казахского национального университета им. аль-Фараби, г. Алматы, Казахстан
Abdildayeva Assel – PhD, Associate Professor of the Department of Artificial Intelligence and Big Data of Al-Farabi Kazakh National University, Almaty, Kazakhstan
e-mail: asselabdildayeva5@gmail.com
ORCID: https://orcid.org/0000-0002-6381-9350

Ахметова Ардақ Мергембаевна – PhD, әл-Фараби атындағы Қазақ ұлттық университеті, Жасанды интеллект және BigData кафедрасының аға оқытушысы, Алматы қ., Қазақстан
Ахметова Ардақ Мергембаевна – PhD, старший преподаватель кафедры Искусственного интеллекта и Big Data Казахского национального университета им. аль-Фараби, г. Алматы, Казахстан
Akhmetova Ardak – PhD, Senior lecturer of the Department of Artificial Intelligence and Big Data, Al-Farabi Kazakh National University, Almaty, Kazakhstan.
E-mail: ardak66@mail.ru
ORCID: https://orcid.org/0009-0000-2605-7206