Designing a Machine Learning Based Software Risk Assessment Model Using Naïve Bayes Algorithm

Suresh K
Research Scholar
College of Engineering, Anna University,
Guindy, Chennai-600 025,
Tamilnadu, India.
mailsureshkrish@gmail.com

Dillibabu R
Associate Professor
Department of Industrial Engineering
College of Engineering, Guindy, Anna University,
Chennai-600 025, Tamilnadu, India.
dillibabu@annauniv.edu

Abstract— Risk management is an essential part for high quality software development processes. More probably Risks are events that could adversely affect the development of the projects or organization environment. Risk or Risk factor can damage critical factors such as budget, time or resources. Commonly it affects critical factors such as budget, time and costs. Risk assessment consists, basically, of identifying, analyzing, planning and controlling events that threat project environment. We used machine learning methods to construct a model that predicts potentially defected modules within a given set of software modules with respect to their metric data. This paper addresses the Supervised Learning mechanism and its one of the method Naive Bayesian Classification. The nature of Naive Bayesian Classification is to evaluate the parameters and it often performs better in many complex real-world situations. The data set used in the experiments is organized in two forms for learning and predicting purposes; the training set and the testing set. Based on this model framework we will analyze all the risk factors and enhance risk assessment process.

Keywords— Software Risk, Risk Assessment, Naive Bayesian Classification.

I. INTRODUCTION

In software development process 65% of project failures are accounted by management issues and 35% by technical issues. Managerial issues include problems which can be attributed to project structure, project resources, planning methodologies, customer buy-in, and inadequate risk management. Software program production often encounters issues such as, over-budget costs, delays in schedule and low quality of product. All of these issues pose a risk to the development of software systems. Program developers must perform a risk analysis before issues develop to identify the risks to their system, and create an action plan to mitigate the impact of the risks as well as resolve any issues that are unanticipated. The success of projects can be credited to the appropriate management of risks. This paper reviews the fundamentals of software risk management and the different popular risk management process models.

A. Software Risk

Risk management is a way to manage risks. In other words, it concerns all activities that are performed to reduce the uncertainties associated with certain tasks or events. In the context of projects, risk management reduces the impacts of undesirable events on a project. Risk management in any project requires undertaking decision-making activities. A brief summary of each risk management activity is described as follows:

- Identify: Identification surfaces software-related risks before they become actual problems which adversely affect the project. Before risks can be managed, they must be identified.
- Analyze: Analysis is the conversion of identified risk data into decision-making information, and provides the quantification and oversight clarity needed to guide the project manager to work on the "right" risks.
- Plan: Planning involves developing actions to mitigate individual software risks, prioritizing risk mitigation actions, and integrating these actions into an executable risk management plan.

- Track: Tracking consists of implementing the risk management plan and monitoring the status of risks and actions taken to mitigate those risks. Risk metrics and triggering events are monitored as part of the tracking function.
- Control: Control corrects for deviations from planned risk mitigation actions; and builds on project management processes to control mitigation plans, respond to triggering events, and improve risk management processes.
- Communication: Communication among the appropriate organizational entities must exist for risks to be identified, analyzed, planned for, tracked, and controlled correctly. Risk communication lies at the centre of the paradigm to emphasize both its pervasiveness and its criticality.

B. Purpose of Risk Management

Risk management involves studying a system or process thoroughly to identify concerns or potential risks, analyzing them, and developing strategies for mitigation and control of the risks. Risk mitigation does not mean altogether eliminating the activities that create the risk. It may instead result in the reduction of the risk to an acceptable level.
When identifying risks to a software system, it is important to know all the possible risks, the level of severity of each risk and all the potential consequences of each. The action steps to mitigate or control each risk are determined based on a thorough knowledge of all risks. This "preventative" approach to risk management allows software developers to finish projects within their expected timelines and budgets.
Projects with effectively managed risks also tend to produce better quality outputs, in addition to reduced costs and time.
Software risk management approaches assess risks during all the phases of software development, by integrating risk management practices along with the software development process.
As a result, in these approaches, the risk management models depend on the development process.
With increasing complexity in software development, organizations have realized the importance of risk management, because it helps in reducing the uncertainties involved in developing software, and decreasing the chances of project failures.
It is therefore imperative that software developers are aware of the risks and are knowledgeable about the nature of the risks. Risks to software systems should be considered throughout the process, rather than one part of the process.
A risk management training program for software developers should aim to teach developers to:

- identify risks (business, process and integrity),
- calculate risk probabilities for quantitative assessments and to set realistic bands for qualitative assessments,
- calculate quantitative and determine qualitative risk impacts,
- determine when applying qualitative versus quantitative assessment is appropriate,
- perform safety and hazard analysis of a software product,
- Prepare and carry out risk mitigation, monitoring and management strategies.

Risks can be identified in all parts of a process. There are several risk models available to assist with risk identification and classification. These models may focus on the management of business risks rather than product risks. However, software products are critical and therefore developers should also be aware of methods used for determining risks to the safety and integrity of their products.

2. RELATED WORK

A literature review includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. The performance of identifying risk by using the Spatial and temporal characteristics of information is done by using different methods. The dataset was collected from different CMMI level companies and also collect the risk related information. Machine learning techniques may be chiefly divided in 2 groups: classification and agglomeration techniques. Classification techniques square measure designed for classifying unknown samples mistreatment data provided by a collection of classified samples. This set is sometimes remarked as a coaching set, because, in general, it's wont to train the classification technique a way to perform its classification. Agglomeration techniques may be wont to split a collection of unknown samples into clusters. One in every of the foremost used agglomeration techniques is that the k-means technique. [5-7] the following three literature review attempts to demonstrate the fact and sight the survey parallel to the preceding survey.

Chandan Kumar and Dilip Kumar Yadav [8]. attempts to define and estimate probability value of software risk with help of the qualitative value of software risk indicator. In this paper applied the BBN approach to construct the model as well as to calculate the probability value of software risk. Estimation of software risk focuses on top ranked software risk indicator of software development risk taxonomy and their causal relationships. This model is different from existing models because the existing

model does not consider the uncertainty of software risk indicator. The model is evaluated by MMRE and BMMRE and it has been found that estimation accuracy is much better.

Wen-Ming Han, Sun-Jen Huang., [9] defines Achieving effective software risk management requires project managers to understand the nature of software risks. Thus, information about the probability of occurrence and impact of software risks on project performance can help the project managers to develop a better risk management strategy. This empirical study considered risk information on 115 software projects. The results indicate that a positive correlation does not exist between the probability of occurrence and impact among the six risk dimensions. The relationship between software risks and project performance was also examined in the high, medium, and low-performance projects. The results show that the ''requirement'' risk dimension is the principal factor affecting the project performance. Aside from this, one of the ways to improve project performance is by properly planning the development activities and reducing the project complexity. Likewise, if the project manager is unable to effectively manage the requirements of the whole project life cycle, and does not well-plan nor monitor the software risk management plan, the software projects is likely to perform poorly. The performance of the software projects could also benefit from exploring the relationship between risk components and some important attributes of software projects such as the type of software systems and project duration.

Mirko Perkusich A, Gustavo Soares A, Hyggo Almeida A, Angelo Perkusich., [10] presented a procedure based in Bayesian networks to assist in detecting the process problems in software development projects. By increasing the efficiency of software processes, increase the project's chances of success. The procedure consists of five stages: (i) Bayesian Network Construction, (ii) Bayesian Network Evaluation, (iii) Bayesian Network Data Input, (iv) Bayesian network outputs' analysis and (v) execution of corrective and preventive actions. Its goal is to expose the problems in software processes to help the team to improve the project's chances of success. For the first stage, it present a guideline to build Bayesian networks to model software development processes. This applied the procedure on Scrum-based software development projects. The goal was to develop a generic model for these projects considering Scrum's principles and rules as well as the industry's best practices. To execute the procedure's first stage (i.e., Bayesian Network Construction) we identified Scrum's key software process factors and quantified their relationships. To identify the key software process factors, we researched the literature from respected practitioners and presented the identified factors to a group of experts. To quantify the relationships between the factors, we collected data from practitioners through an online survey. And then statistically analyzed the data, defined an algorithm to build weighted expressions for each relationship, and used the expressions to define the probability functions.

Benchmarking: A particular machine learning algorithm is usually an instantiation of the model/preference/search components. The more common model functions in current machine learning practice include [4]:

- Classification: classifies a data item into one of several predefined categorical classes.
- Regression: maps a data item to a real valued prediction variable.
- Rule generation: extracts classification rules from the data.
- Discovering association rules: describes association relationship among different attributes. Summarization: provides a compact description for a subset of data.
- Dependency modeling: describes significant dependencies among variables.
- Sequence analysis: models sequential patterns, like time-series analysis. The goal is to model the states of the process generating the sequence or to extract and report deviation and trends over time. To proceed further, four techniques have been taken to compare the efficiency.

Naive Bayesian Classification: The Concept of Naïve Bayesian classifier is based on Bayes theorem with independence assumptions between predictors. Naive independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods. Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$ and $P(x|c)$. Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(c|x) . \frac{P(x|c)\,P(c)}{P(x)} \qquad (1)$$

$$\text{Where } P(c|x) . P(x_1|c) . P(x_2|c) .... P(x_n|c) . P(c)$$

P(c|x) is the posterior probability of class (target) given predictor (attribute), P(c) is the prior probability of class, P(x|c) is the

likelihood which is the probability of predictor Naive Bayesian network is a powerful tool for dealing uncertainties and widely used in risk management datasets. Bayesian network is a graphical model which encodes probabilistic relationship among variable of interest when it is used with statistical technique, the graphical model has several advantages for data analysis [12, 13].

Artificial Neural Networks (ANN or just Neural Networks) are modeled after the biological neurons in brain structures. The individual neuron models may be combined into various networks made up of many individual nodes, each with its own set of variables. These networks have an input layer, an output layer, and one or more hidden layers. The hidden layers provide connectivity between the inputs and outputs. The network may also have feedback, which will take result variables and use them as input to prior processing nodes. With the help of NN it is possible to be modeled different possible directions in the process of software development and in this way to find the potencies for risk[2]..

K Nearest Neighbor: K nearest neighbor techniques is one of the classification techniques in machine learning. It does not have any learning phase because it uses the training set every time a classification performed Nearest Neighbor search (NN) also known as proximity search, similarity search or closest point search is an optimization problem for finding closest points in metric spaces. K nearest neighbor is applied for simulating daily precipitation and other weather variables.

Decision Tree: The decision tree is one of the popular classification algorithms in current use in Machine Learning. Decision tree is a new field of machine learning which is involving the algorithmic acquisition of structured knowledge in forms such as concepts, decision trees and discrimination nets or production rules. Application of risk management techniques on risk related data for software risk management shows the success on Advanced Geospatial Decision Support System (GDSS).

Comparison of Techniques: The conclusion can be drawn from the comparison of the above stated algorithms every single method plays its vital role in different categories of Risk Management Techniques, which issues its drawback at last. In order to acknowledge basically there are two approaches used for prediction. They are Empirical method and dynamical methods. The empirical approach is based on the study of historical data of the and unpredictable variables over different software companies. In dynamical approach predictions are generated by physical models based on systems of equations that predict the evolution of the software risk in CMMI level companies.

TABLE I. COMPARISON OF TECHNIQUES

Machine Learning Techniques	Advantages	Disadvantages
Naïve Bayesian	Fast to train and classify	assumes independence of features
	Not sensitive to irrelevant features.	Too much of a black box.
Neural Network	Neural networks realized as specialized hardware systems. Useful for network learning.	Neural networks are not probabilistic.
K-Nearest Neighbor	Robust to noisy training data. Effective if the training data is large.	Need to determine the value of K Computation cost is quite high.
Decision Trees	Calculations are simple to understand and interpret. Can be combined with other decision techniques.	Can get very complex. Information gain in decision trees are biased.

3. PROPOSED SYSTEM

The system architecture shown in Fig. 1, explains the working of the proposed system. In this paper, multi-source information is integrated to achieve the purpose of accurately monitoring risk factors. Risk or Risk factor can damage critical factors such as budget, time or resources. Commonly it affects critical factors such as budget, time and costs. Risk management consists, basically, of identifying, analyzing, planning and controlling events that threat project environment. Therefore, growing demand for maximum trustable software creates a unique challenge to the software industry. Therefore, the risk and risk factors is affected by many factors, such as time, effort, architecture, configuration management, training, quality and environment. Based on the

defined risk criteria, the intensity, temporal and spatial distribution of risk can be monitored. By using this type of attributes, risk management by using the supervised machine learning methods were predicted.

Input Selection is the first process. In this, first have to browse and select the input for the process. Input of the process is dataset. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable and each row corresponds to a given member of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Normally Dataset pre-processing is the method for cleaning the dataset. Data may be incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data noisy: containing errors or outliers inconsistent: containing discrepancies in codes or names. In this elimination of this type of occurring in the dataset is going to be performed. Eliminating the unwanted value or symbols or characters in the dataset.

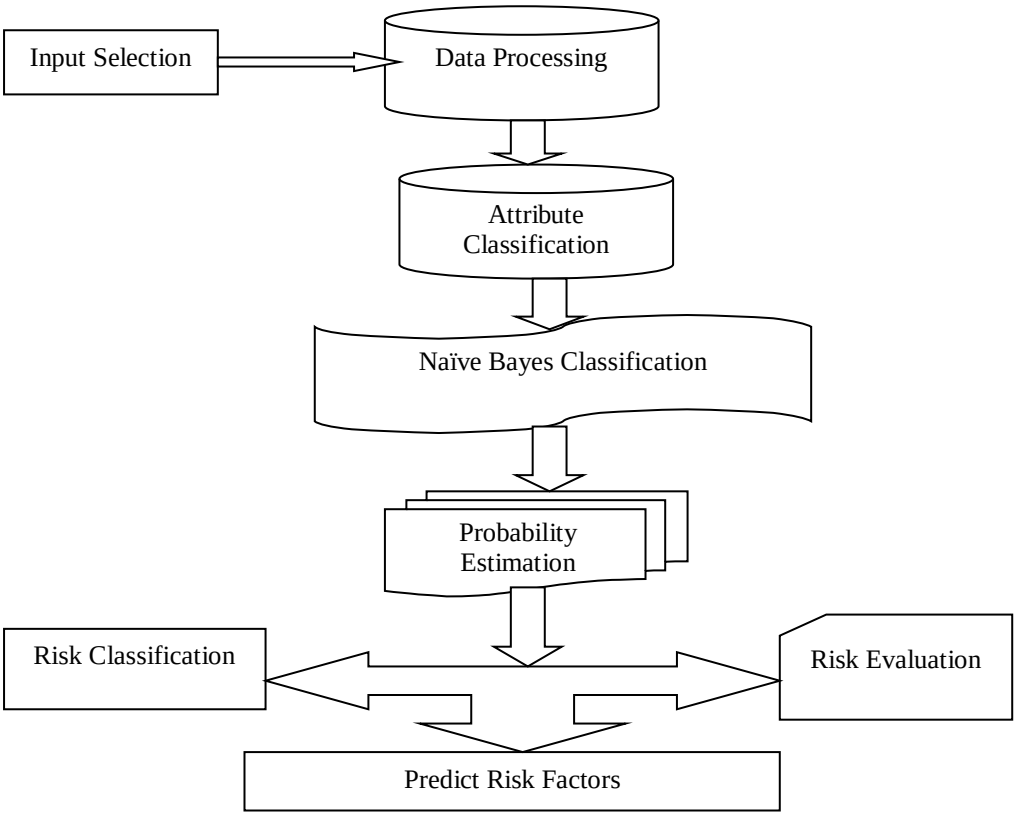

FIGURE 1: ARCHITECTURE OF PROPOSED SYSTEM

Classification is a way of categorizing the data (records) for an attribute. Attributes can use different classifications for the same data to change the nature of the display; this can be achieved based on the attributes in the dataset. Attributes in the dataset are the scarcity of precipitation, In the implementation process, implementation of the risk assessment research by using the supervised classification algorithm called Naive Bayes Classification algorithm. A Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood. Predicting the critical risk by using the naïve bayes algorithm and evaluate the performance by using the parameters of the process. Then, evaluation of the graph based on the high, mid, low, normal risk etc., is done like that. These are estimated by using the probability values in the dataset. The graph can be evaluated by using this type of parameters form probability estimations.

4. RESULT AND DISCUSSION

Craven and Shavlik et al., defines in there paper [3] listed five criteria for rule extraction and they are as follows:

Comprehensibility: The extent to which extracted representations are humanly comprehensible.

Fidelity: The extent to which extracted representations accurately model the networks from which they were extracted.

Accuracy: The ability of extracted representations to make accurate predictions on previously unseen cases.

Scalability: The ability of the method to scale to networks with large input spaces and large numbers of weighted connections.

Generality: The extent to which the method requires special training.

The table aims to come out of the techniques being used in the risk management and its allied area. Table 2, analyzes some machine learning techniques and its accuracy to predict the critical risks [14]. From this table, we understand that the Naive Bayesian has better accuracy in result than other machine learning techniques. Accuracy is determined by the formula 100-RMSE. RMSE (Root Mean Square Error) is one of the measuring techniques for predicting critical risk factors. It is measured by the differences between values predicted by a model and the values actually observed from the model [12].

TABLE II: ACCURACY OF MACHINE LEARNING TECHNIQUES

Machine Learning Techniques	**Accuracy**
Naïve Bayes	85.77%
KNN (k=30)	81.81%
Decision Trees	81.40%
Neural Networks	82.81%

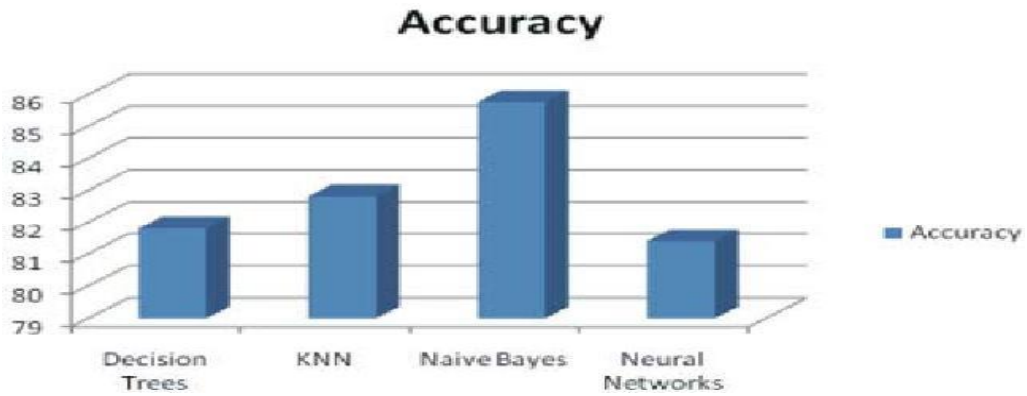

Fig 3.Graphical representation of risk analysis techniques and its accuracy

CONCLUSION

The Bayesian model attempts to characterize software risk in a more accurate and comprehensive way. Cross-validation analysis proved that the regression accuracy is very high and the risk factors can be accurately portrayed using the input variables. This model application using a variety of methods and data, there is still some work to be done in the future research because of the complex spatial and temporal characteristics of risk management. To overcome imitation, it assesses software risk by using the Spatial and temporal characteristics of information. Collecting the dataset from CMMI level companies and it can be implemented by using the Bayesian supervised machine learning algorithm. Through this the performance and effectiveness will be improved.

References

[1]. Bertolino, A., and Strigini, L., On the Use of Testability Measures for Dependability Assessment, (1996).
[2]. Bishop, M., Neural Networks for Pattern Recognition, Oxford University Press.(1995)
[3]. Boetticher, G.D., Srinivas, K., Eichmann, D, A Neural Net-Based Approach to Software Metrics, Proceedings of the Fifth International Conference on Software Engineering and Knowledge Engineering, San Francisco, (1993)
[4]. Jensen, F.V., An Introduction to Bayesian Networks, Springer. {1996)
[5]. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B.Wang. Automated support for classifying software failure reports. (2003).
[6]. Yuriy, B., and Ernst, M. D., Finding latent code errors via machine learning over program executions (2004).
[7]. Zhang, D., Applying Machine Learning Algorithms in Software Development, (2000).
[8]. A Probabilistic Software Risk Assessment and Estimation Model for Software Projects. Chandan Kumar and Dilip Kumar Yadav (2015)
[9]. An empirical analysis of risk components and performance on software projects. Wen-Ming Han, Sun-Jen Huang (2007)
[10]. A procedure to detect problems of processes in software development projects using Bayesian networks. Mirko Perkusich , Gustavo Soares , Hyggo Almeida, Angelo Perkusich. (2015)
[11] Masood Uzzafer, "A Novel Risk Assessment Model for Software Projects" 978-1-4244-9283-1/11/ 2011 IEEE.
[12] XU Ru-Zhi, NIE Pei-Yao, SAI Ying, QU Le-Hong, LEE Yun-Ting "Optimizing Software Process Based On Risk Assessment and Control" (2005)
[13]. Quinlan, J.R., 1986. Induction of decision trees. Machine Learning, (1986.)
[14]. Hudepohl, P., Khoshgoftaar, M., Mayrand, J., Integrating Metrics and Models for Software Risk Assessment, The Seventh International Symposium on Software Reliability Engineering (1996).
[15]. Mitchell, T.M., 1997. Machine Learning, McGrawHill.
[16].Neumann, D.E., 2002. An Enhanced Neural Network Technique for Software Risk Analysis, IEEE Transactions on Software Engineering, (2002).