Classification of Risk in Software Development Projects using Support Vector Machine

M.Zavvar[1], A.Yavari[2], S.M. Mirhassannia[1], M.R.Nehi[1], A.Yanpi[1] and M.H.Zavvar[1]
[1]*Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran.*
[2]*Mazandaran University of Science and Technology.*
Zavvar.developer@gmail.com

***Abstract*—Traditionally, the lack of confidence in the system life cycle is expressed using the concept of risk. Nowadays, software development projects face various risks. However, the estimation and classification of risk, increased estimation of accuracy and reduced of uncertainty ultimately improve project outcomes. Therefore, in this paper, a Support Vector Machine (SVM) is used to model risk classification in software development projects. The proposed algorithm is compared with other methods in the literature such as Self Organizing Map (SOM) and K-Means based on measures of Classification Accuracy Rate (CAR) and Area Under Curve (AUC). According to the results, the proposed method exhibits superior CAR and AUC.**

***Index Terms*—Classification; Risk; Software; Support Vector Machine; Area under Curve.**

I. INTRODUCTION

All software projects are associated with risk. The concept of risk has been defined in various ways. For example, in [1] risk is introduced as a traditional way of expressing the lack of reliability in the system life cycle. Also, events that occur during a software project and threaten its success are known as risk [2]. It is inherented in all the projects and cannot be completely eliminated; nevertheless, it is possible to reduce its effects through effective management. Risk refers to the exposure to economic or financial loss, physical injury or damage or delay, resulting from uncertainty associated with a particular current of a job.

Generally, software project risks are divided into two groups: general and specific. While the former may happen in all software projects, the latter varies with the type of project. It is extremely difficult to identify such risks, particularly in estimating their probability of occurrence; and predicting their impact. Several factors contribute to this difficulties, such as project size, complexity, structure, content, long-term planning and volatile changes. Therefore, risk management results in reducing disaster, duplication, concentration, and balancing the required effort; furthermore, it leads to simulation of win-win conditions [3].

Software development projects are often challenged by a variety of risks. The most important factors that could lead to project failure include low efficiency, time pressure, poor quality and high cost [4]. Effective risk management is a complex task that requires a good evaluation of the underlying factors. Due to the complexity of risk factors and the interconnected nature of uncertainties associated with future resources, risk cannot be expressed with mathematical precision during the early phases of the life cycle [5]. Risk management of software projects plays an important role in achieving the desired result. Activities carried out in software projects are inherently high risk, and this results in varying degrees of functionality. Managing software risk has many benefits, including increased reliability, more accurate estimations, and preventing unnecessary effort [6]. As a result of the risk assessment, accuracy is increased while reducing the uncertainty associated with the project. Moreover, since team members are aware of the risk control measures, it is possible to avoid duplication and wasted effort [4].

Management of software projects comprises four phases: identification, evaluation, planning and controlling. According to Bohm, an estimation of risk is obtained by multiplying the probability of risk occurrence by its effect. Qualitative analysis of risk and its impact are dependent on analyst experience as well as statistical data [7].

Risk in the application depends on several small and large factors; thus, they need to be classified according to some criteria. One such classifications in the field of software risks is proposed by Wallace. Various reasons can be cited for the application of this classification including its use of up-to-date information and the fact that it reflects a consensus among the members of PMI. Also, in order to prove the competence and credibility of the framework, the SEM method is used [8]. This framework is composed of six dimensions and each dimension corresponding to risks are discussed. In Table 1, different aspects of this classification and the associated risks are mentioned.

Given the importance of the classification of risk in software development projects and the factors presented in Table 1, in this paper, we provide a method based on a Support Vector Machine (SVM) to classify risks involved in software development projects.

II. RELATED WORKS

In [9], key risks obtained from a group of information technology project managers in Hong Kong were introduced. In their study, a number of new risks were identified from the perspective of the seller. According to the authors, project managers believed that operations performed in foreign countries are associated with greater risk than those carried out domestically.

Table 1
Wallace classification of software risk factors [8]

Risk Dimensions	Risk of Software
User	Members resistance to changes Conflict between users Users with negative attitudes towards the project Users who are not committed to the project Lack of cooperation between Member
Requirements	Constant changes in system requirements System requirements are not fully diagnosed No clearly stated system requirements Anachronistic system requirements
Complexity of the project	The use of new technology High-level technical complexity Immaturity of technology Using technologies that have not been used in previous projects
Planning and Control	Absence of project management methodology that is effective and efficient Lack of sufficient monitoring in project progress Incomplete estimate of the resources required Poor project planning Project milestones are not defined in a transparent manner The project manager is not experienced enough Ineffective communication
Team	Insufficient experience with similar projects Team members are not trained enough Lack of specialized skills by team members
Organizational environment	Organizational management changes during the project Negative impact of trade policies on the project Instability in the organizational environment Organizational restructuring during the project

Furthermore, [10] focused on the experiences of IT project manager. Their reports unveiled more risk and controls that allow us to control the occurrence of risks in the future because of the different factors, which are well explained. Later, in [11], the same authors used the model to improve the quality of information technology projects in organizations. In their study, a novel chi-square test was used to control risks in software projects [12].

In [3], the authors utilized regression testing and effect size testing to improve their previous work on reducing the risk of software development projects. Moreover, in [4], they proposed new risk management method in their software projects by using stepwise regression. This study was performed by using regression analysis in order to control the comparison of each risk factor so that the effect of each factor in the implementation phase could be determined.

The authors in [13] concluded that risk management encompasses process, methodology and specific tools to assess and reduce risk factors associated with the development software life cycle.

In [14], risk management is argued to provide a solution to the standardization of risk assessment and mitigation in software development. In [15], by using the Delphi method, software development risks were extracted with the help of experts. In this study, a total of 53 risks were classified into 14 groups.

In [16], risk management was shown to consist of five phases: risk identification (planning, identifying and prioritizing), analyzing and assessing risk (risk analysis, risk assessment), risk management, risk control, communication, and documentation. In [17], a model based on fuzzy logic was proposed for evaluating the risk in software development projects. The proposed model was created using the five criteria, planning, complexity, requirements; furthermore, three membership functions were considered for input to the proposed fuzzy system. In total, 243 rules were designed. The proposed method was shown to have lower estimation error compared to previous methods in the literature.

Some researchers in [18] employed neural network (NN) and support vector machine (SVM) approaches to establish a model for risk evaluation in project development. In the model, the input is a vector of software risk factors that were obtained through interview with 30 experts, and the output is the final outcome of the project. The experiment shows the model is valid. Interestingly. In their study, the standard neural network model had lower prediction accuracy compared to SVM due to its tendency in finding local optima.

Yong et al. in [19] identified the key software risk factors responsible in achieving successful outcome and used a neural network approach to establish a model for minimizing the risks attributed to failed projects. In order to enhance model performance, principal component analysis and genetic algorithm were employed. The experimental result indicates that the software risk analysis can be improved through these methods and that the risk analysis model is effective.

III. SUPPORT VECTOR MACHINE

A Support Vector Machine is principally a linear machine whose main idea is to create a hyperplane as a level of decision-making, so that the separation between the positive and the negative samples is maximized. By using a method based on statistical learning theory, the technique achieves this optimization. More precisely, SVM is an implementation of approximation of the "structural risk minimization". The structure of the SVM training algorithm is based on a core of inner multiplication between a support vector such as x_i, and the vector x derived from the input space. The smallest subset of training data extracted by the algorithm is known as the Support Vector. Depending on

how the multiplication core of the inner is formed, different training machines with non-linear decision-making planes may be obtained [20-22].

Consider training sample $\{(x_i, d_i)\}_{i=1}^{N}$, where x_i is an input algorithm for the sample n and d_i is the corresponding final output, which is different from the input. We assume that classes displayed with a subset of $d_i = +1$ and $d_i = -1$ are linearly separated. A plane of decision-making equation such as the above plane is as follows:

$$w^T x + b = 0 \tag{1}$$

where x denotes an input vector, w represents an adjustable weight vector and b is a bias. The above equation can be written as follows:

$$\begin{aligned} w^T x_i + b \geq 0 \quad for \quad d_i = +1 \\ w^T x_i + b < 0 \quad for \quad d_i = -1 \end{aligned} \tag{2}$$

For a given weight vector w and a bias b, the distance between the above plane defined in Equation (1) and the closest data point is called the *resolution*, represented by p. The objective of SVM is to find unique plane such that the resolution p is maximized. In this situation, the decision level is considered as the optimal hyperplane. Figure 1 shows the geometry structure of the optimal hyperplane for a two-dimensional input space.

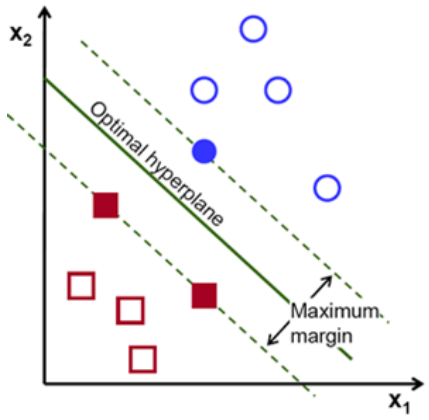

Figure 1: A schematic representation of an optimal hyperplane for linearly separable patterns [19].

Now suppose w_o and b_o are the optimal weight vector and bias, respectively. Therefore, the optimal hyperplane is defined as follows:

$$w_o^T x + b_o = 0 \tag{3}$$

The main concern in finding the parameters w_o and b_o for the optimal hyperplane with a set the training is $\tau = \{(x_i, d_i)\}$. Then, the pair (x_o, d_o) must satisfy the following conditions:

$$\begin{aligned} w_o^T x_i + b_o \geq 1 \quad for \quad d_i = +1 \\ w_o^T x_i + b_o \leq -1 \quad for \quad d_i = -1 \end{aligned} \tag{4}$$

Data points (x_i, d_i), which satisfy the condition of equality for both formulas (4) are called Support Vector and hence they are called SVM. These vectors play a prominent role in the performance of this type of training machines. Conceptually, backup vectors are data points located near the plane of making decisions, and therefore they are difficult to classify. In order to obtain the optimal hyperplane, once the Lagrange multipliers are applied and the necessary calculations are performed, the following is performed:

Given the training sample $\{(x_i, d_i)\}_{i=1}^{N}$, find the Lagrange coefficients $\{\alpha_i\}_{i=1}^{N}$ such that the following objective function is maximized:

$$Q(\alpha) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j d_i d_j x_i^T x_j \tag{5}$$

Subject to:

$$\sum_{i=1}^{N} a_i d_i = 0 \tag{6}$$

$$0 < \alpha_i \leq C \quad for \quad i = 1, 2, ..., N \tag{7}$$

where C is a positive parameter identified by the user.

IV. Methodology

As mentioned earlier, risk management during the software development life cycle is associated with a large number of benefits, because it enables increased confidence, more accurate estimate, and prevention of unnecessary efforts for project development. Therefore, considering the importance of addressing the issue of risk in software development projects and relying on the classification by Wallace, we set out to propose a method based on SVM for classification of the risks involved in software development projects.

An SVM is a statistical classification model, which initially maps non-linear data in a high-dimensional space through several kernels and then tries to find the hyperplane that separates data with the maximum margin from this hyperplane. In its general form, SVM is used for two classes; thus, it is sufficient for this purpose. A critical feature of SVM is data classification, which is performed by minimizing data error of test sets. In contrast, in other classifiers such as neural networks performance is based on minimizing the error of data of training sets. Thus, in SVM, there is no concern for being stuck in local minima. In an attempt to enhance classification accuracy in the proposed method and allow SVM input data to be focused on a specific area, the input data needs to be normalized before the procedure continues. Data normalization is applied to data points that are not in the same domain so that the values fall in the same range.

Also in the dataset, the desired output is divided into either high risk and low risk in two categories. Subsequent to the classification, for the sake of simplicity in the proposed method, "high-risk" is replaced with the value 1 and "low-risk" is replaced with the value -1. Accordingly, for a given sample, if the SVM generates an output of 1, the risk is assumed to be high, whereas an output value of -1 represents a low risk.

Also, to work with SVM, the data should be divided into two parts to be used for training and test purposes. In this paper, 70 percent of the data were used for training the SVM while the remaining were used for testing. It should be noted that, all input data for both training and testing were normalized according to the explained procedure prior to being used.

V. Results And Findings

In this section, the results of the application of SVM-based methods are discussed, and the results are compared with those of the SOM and K-Means algorithms. As stated earlier, for classified operations, the dataset was obtained from software development projects. This data set includes 530 samples 70 percent of which the data was used for training, and 30 percent for testing.

In the proposed method for SVM, the parameter C was considered equal to 100 and according to the data, a linear function was used as the kernel of the network. The linear function is shown in Equation (8).

$$K(x_i, x_j) = x_i^T . x_j \tag{8}$$

The knowledge produced in the learning stage of the model must be analyzed in the evaluation stage in order to determine its value, as well as the efficiency of the learning algorithm. These measures can be calculated for both the training data set in the learning phase and the test data set in the test phase. Also a condition for success in the data mining is the ability to interpret the obtained knowledge. One of the important criteria used to determine the effectiveness of a classifier is the CAR. In fact, this criterion is the most popular criterion of standard algorithms and public classification that shows the percentage of total records correctly classifier by the classifier. Using equation (9), the CAR is obtained [23]. Values of TP and TN are the most important values that should be maximized for two categories. In the proposed method, the value of CAR is equal to 99.5084, and in the SOM method, it is equal to 98.2467 and in method of K- Means it is equal to 97.4618.

$$CAR = \frac{TN + TP}{TN + FN + TP + FP} \tag{9}$$

In addition to the standard CAR, another important criteria used to determine the performance of classification criteria is the AUC. It represents the area under the Receiver Operating Characteristic (ROC) curve; larger values of the ROC are indicative of greater classifier marginal efficiency. UAC is calculated through Equation (10) [24]:

$$AUC = \frac{1 + TPR - FPR}{2} \tag{10}$$

where $TPR = \frac{TP}{TP + FN}$ and $FPR = \frac{FP}{FP + TN}$. TP and TN are the number of data points that have been correctly classified as positive and negative, respectively. On the other hand, FP and FN are the number of data points that has been falsely classified as positive and negative, respectively.

A ROC curve allows a visual comparison of a set of classifiers; also numerous points in the ROC space are significant. The lower left point (0, 0) indicates the strategy that will be generated in a positive classification. The opposite strategy, produced without condition of positive classification, is determined with top right point (1, 1). Point (0, 1) shows perfect grouping. More generally, considering the two points in the ROC space, one is deemed better than the other if more space is located closer to the northwest corner. Also, it should be noted that the ROC curves show a classifier behavior regardless of the distribution of categories or cost of an error; therefore, classification performance is separated from these factors [25]. Only when a classifier in the performance space clearly dominates the other categories, it can be claimed to be superior. For this reason, the area under ROC curve showing the AUC criterion can play a decisive role in introducing categories of supremacy clause. In Figure 2 the ROC curve, AUC value for the proposed approach, and methods of SOM and K-Means are shown.

Unlike other criteria for determining the efficiency classifier, AUC criterion is independent of decision-making threshold of classifier. Therefore, this measure is indicative of the reliability of the output of a classification specified for different data sets that this concept is not comparable by any other performance measures that derived the category. As shown in Figure 2, the AUC is equal to 0.9802 in the proposed method, whereas it is equal to 0.9732 and 0.9643 for SOM and K-Means, respectively. Overall, the results regarding AUC and CAR show that the proposed method outperforms other methods of risk classification in software development projects.

VI. Conclusion

In this paper, after highlighting the importance of the classification of risk in software development projects and the factors affecting it, an SVM-based method was proposed for risk classification in software development projects. Then, the classification accuracy of the proposed method was compared with that of SOM and K-Means based on the CAR and AUC. After examination, the CAR and AUC of the proposed method were equal to 99.5084 and 0.9802, respectively. The same values were 98.2467 and 0.9732 for SOM and 97.4618 and 0.9643 for K-Means. As things stand, the CAR and AUC in proposed procedure are higher than the corresponding values in SOM and K-Means. The results show that the proposed method for risk classification in software development projects, exhibits better precision and performance.

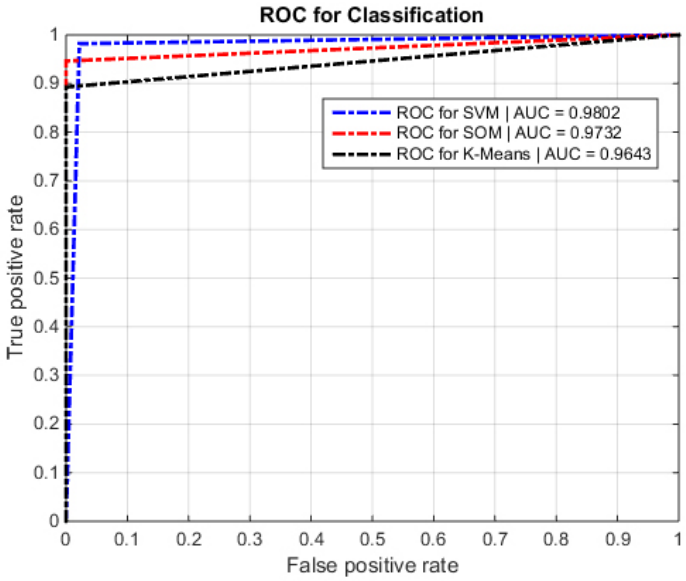

Figure 2: ROC curve AUC value of the proposed method as well as that of other methods

REFERENCES

[1] S. Cole, X. Giné, J. Tobacman, R. Townsend, P. Topalova and J. Vickery, "Barriers to household risk management: evidence from India", *American economic journal. Applied economics*, Vol.5, no.1, 2013. pp.104–135.

[2] P.L. "Bannerman, Risk and risk management in software projects: A reassessment", *Journal of Systems and Software*, Vol.81, no.12, 2008, p. 2118-2133.

[3] V. T. Covello, L. B. Lave, A. A. Moghissi and V. R. R Uppuluri, "Uncertainty in risk assessment, risk management, and decision making". *Springer Science & Business Media*, Vol. 4. 2013.

[4] M.K. Sadgrove, "The complete guide to business risk management". *Ashgate Publishing, Ltd*, 2015.

[5] P. Bolton, H. Chen and N. Wang, "Market timing, investment, and risk management". *Journal of Financial Economics*, Vol.109, no.1, 2013, pp. 40-62.

[6] J. Bessis and B. O'Kelly, "Risk management in banking", *John Wiley & Sons*, 2015.

[7] J. Lam, "Enterprise risk management: from incentives to controls", *John Wiley & Sons*, 2014.

[8] S.-J. Huang and W.-M. Han, "Exploring the relationship between software project duration and risk exposure: A cluster analysis". *Information & Management*, Vol.45, no.3, 2008, pp. 175-182.

[9] I. Rus, H. Neu and J. Münch, "A systematic methodology for developing discrete event simulation models of software development processes". In *Proceedings of the 4th International Workshop on Software Process Simulation and Modeling*, 2014.

[10] D. Ince and D. Andrews, "The Software life cycle", *Butterworth-Heinemann*, 2014.

[11] S. Islam, H. Mouratidis and E.R. Weippl, "An empirical study on the implementation and evaluation of a goal-driven software development risk management model". *Information and Software Technology*, Vol.56, no.2, 2014, pp. 117-133.

[12] P. Hopkin, "Fundamentals of risk management: understanding, evaluating and implementing effective risk management", *Kogan Page Publishers*, 2014.

[13] A. Dumont, P. Fournier, M. Abrahamowicz, M. Traoré, S. Haddad, W.D. Fraser and QUARITE research group, "Quality of care, risk management, and technology in obstetrics to reduce hospital-based maternal mortality in Senegal and Mali (QUARITE): a cluster-randomised trial". *The Lancet*, Vol.382, no.9887, 2013. pp. 146-157.

[14] D.R. Van Deventer, K. Imai and M. Mesler, "Advanced financial risk management: tools and techniques for integrated credit risk and interest rate risk management", *John Wiley & Sons*, 2013.

[15] P. Chawan, J. Patil and R. Naik, "Software risk management". *International Journal of Computer Science and Mobile Computing*, Vol.2, no.5, 2013, pp. 60-66.

[16] R. Conforti, M. La Rosa, A.H. Ter Hofstede, G. Fortino, M. de Leoni, W.M. van der Aalst and M.J. Adams, "A software framework for risk-aware business process management". in *Proceedings of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE): CEUR Workshop Proceedings*, Vol.998. 2013.

[17] A.S. Khatavakhotan and S.H. Ow, "Development of a Software Risk Management Model using Unique Features of a Proposed Audit Component". *Malaysian Journal of Computer Science*, Vol.28, no.2, 2015.

[18] Y. Hu, J. Huang, J. Chen, M. Liu and K. Xie, " Software Project Risk Management Modeling with Neural Network and Support Vector Machine Approaches". in *Third International Conference on Natural Computation*, 2007.

[19] H. Yong, C. Juhua, R. Zhenbang, M. Liu, and X. Kang, " A Neural Networks Approach for Software Risk Analysis". in *Sixth IEEE International Conference on Data Mining Workshops*, 2006.

[20] V.N. Vapnick, "The Nature of Statistical Learning Theory", *Second Edition, Springer-Verlag New York Inc*, 2000.

[21] S. Haykin, "Neural Networks: A Comprehensive Foundation. Second Edition", *Prentice-Hall Inc*, 1999.

[22] C.J. Burges, "A tutorial on support vector machines for pattern recognition". *Data mining and knowledge discovery*, Vol.2, no.2, 1998, pp. 121-167.

[23] S.-W. Lin and S.-C. Chen, "PSOLDA: A particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis", *Applied Soft Computing*, Vol.9, no.3, 2009, pp. 1008-1015.

[24] J. Davis and M. Goadrich. "The relationship between Precision-Recall and ROC curves". in *Proceedings of the 23rd international conference on Machine learning*. 2006.

[25] T. Fawcett, "An introduction to ROC analysis", *Pattern recognition letters*, Vol.27, no.8, 2006, pp. 861-874.