American Journal of Multidisciplinary Research & Development (AJMRD)
Volume 04, Issue 02 (February - 2022), PP 35-41
ISSN: 2360-821X
www.ajmrd.com

Research Paper **Open Access**

Software project risk assessment using machine learning approaches

Mohammad Ibraigheeth[1],Aws Ismail Abu eid[2]
[1]Faculty of Science, Computer Science Department, Northern Boarder University,1321 Arar, Saudia Arabia.
[2]Faculty of Science, Computer Science Department, Northern Boarder University,1321 Arar, Saudia Arabia.

Abstract: Both Data Science and Software Engineering require programming strengths. While Big Data is moreconcerned with data collection and analysis, Software Engineering is more concerned with developing, functionality, and features.
To develop software projects, one of the major demands is a high system functionality to get over the complexsystem requirements. Risk assessment is one of the important attributes that directly affect the success of thesoftware systems. The ability
to assess the software system risks by identifying the expected failures early can help organizations in makingdecisions about possible solutions and improvements. Inaccurate risk analysis could lead the performance towardfailure and thus, could reflect the system reliability. This paper focuses on software project risk assessment byperforming early failure prediction. Different
machine learning approaches will be applied. The research intends to develop evidence-based risk assessmentmodel that use historical failure data from several past software projects as a training data to effectively assesssoftware project risks. The model is developed using machine learning. For developing the model, six approacheswere selected in an attempt to achieve diversity, namely, logistic regression, Naïve Bayes, support vector machines, decision trees, neural networks and adaptive neuro-fuzzy inference systems. This research contributes to the field ofsoftware system development as it develops software project risk assessment models that can be generally appliedon any software project during any phase of software development process.

Keywords: (Big Data, Data Science ,Failure factors, Failure probability, Risk assessment , Software Failure ,SoftwareRisk).

I. INTRODUCTION

The academic and professional literature on 'big data' places a strong emphasis on opportunities. while, the risk posed by repurposing data, consolidating data from multiple sources, applying analytical tools to the resulting collections, drawing inferences, and acting on them have received far less attention. [1]. Also, Big data analysis necessitates an increase in the level of professional risk [2].

Inaccurate risk analysis could lead the performance toward failure and thus, could reflect the system reliability [3]. Software reliability can be defined as “the probability of failure-free software operation for a specified period of time in a specified environment [4]. Therefore, the software reliability is strongly affected by different type of failures [5].

Software failures prediction could be performed during different phases of the software development life cycle. The results of prediction can be used by project managers to evaluate the current project state and can help in avoiding probable risks. Failures prediction plays an important role in guiding software managersto recognize strategies that can steer project from failure to success.

To assess the probable risk, this research proposes failure prediction model developed based the most factors that affect the software project and can lead to projectfailure. A list of failure factors are used as predictors in the prediction models implementation. The proposed models use historical data gathered from several past software projects. This data is used to fit and validate the models to efficiently predict potential failures. These models generate theexpected project outcome (fail or success). Also, these models generate theprobability of the project failure, and therefore the decision makers can use the results of these models to decide proceeding,

proceeding with improvement, or not preceding the project and replace it with other successful and robust software product.

The prediction algorithms, in general follow machine learning and statistical approaches [6]. These approaches can learn from data without depending on rule-based systems. Statistical approaches aim to find mathematical relationship between variables. These approaches can be used to develop models that would be able to solve various types of problems based on some information and constrains. In the first step, the models is extracted by applying machine learning algorithm on "training" data, and then in the second step, measure the created model accuracy and performance by applying it on predefined "testing" data

The models are developed using different machine learning techniques to compare their performance and achieve greater prediction accuracy.Machine learning algorithms are used to predict the target class for each data instance in the dataset according to certain constrains. The result of these models is the excepted software project outcome; they returns either 1 (if the model result is "Failure") or 0 (if the model result is "Success").Also, the model output can be a value ranging from 0 to 1 representing failure probability.The developed models are capable of classifying data based on observed information in training set, and then can be used in predicting class labels for newly available information, and making decisions regarding new situation.

In this paper, the prediction model is developed using six different machine learning techniques;logistic regression (LR), Naïve Bayes (NB), support vector machines (SVM), decision trees (DT), neural networks (NN) and adaptive neuro-fuzzy inference systems (ANFIS).The performance of these techniquesare evaluated using a number of performance measures such asaccuracy, sensitivity and F-measure.

II. RELATED WORK

Software risk and failure assessment could be performed during various phasesof the software development process [7]. Numerous techniques have been used to address and manage the software risks. Yavari et al. [8] proposed a method to assess software risk using fuzzy logic.Hu Y et al. [9] proposed a framework for risk analysis based on risk causality using Bayesian networks. Each of these techniques has its own advantages. For example, regression analysis is suitable for risk prediction as it can find the relationships between variables. Applying decision trees is fast and simple while neural network is suitable when the relationships between the system variables are non-linear. Applying Bayesian network with considering causality dependencies can perform better prediction.

Sangaiah et al. developed [10] an assessment framework to rank software project risks by using Fuzzy Multi-Criteria Decision Making approachesThey proposed methodology to provide an evaluation mechanism to assess software risk factors.

Bilal et al. [11] utilized a questionnaire-based approach to highlight the potential risks from 163 companies working in software industry in Pakistan.They aimed to prioritized risks to guide companies to prioritize the risks they face while working on small and medium software projectsin order to increase the chances of project success.The results concluded that the level of severity of the majority of risks faced by the software industry in Pakistan is large and high.

Suresh &Dillibabu [12] addresses Naive Bayesian classificationmechanism to analyze risk factors and develop risk assessment process. The developed model attempts to characterize risk. They used cross-validation analysis to prove that the prediction performance is high and the factors of software risk can be accurately identified.

Most of proposed models were developed based on specific metrics and they verified using certain case studies. Consequently, those approaches could not be applicable for all software projects. Therefore, it is a necessity to develop general prediction models that can be applied during any development phase for any software project

III. METHODOLOGY

This paper aims to assesssoftware project risks by performing software project failure prediction early or during any phase of project development process.

In the first stage of this research, a common failure factors are identified. These factors were identified by analyzing different studies related to software project failure and success.Six core failure factors that are identified in previous study [13] are selected to be the model's predictors. These factors are presented in TABLE I. Then, the model is fitted using failure dataset. The dataset is constructed based on availablesoftware project reports, case studiesand surveys related to previously developed software projects.Theconstructed dataset will be used to fit and verify the implemented model.

TABLE I
COMMON SOFTWARE PROJECT FAILURE FACTORS

ID	Failure Factor
X1	Unrealistic objectives.
X2	Staff technical problems
X3	Lack of users involvement
X4	Instability of requirements
X5	Problems in the used technology
X6	Management Problems in the project

In the second stage, the risk assessment (Failure prediction)model is implemented using six machine learning methods: logistic regression (LR), Naïve Bayes (NB), support vector machines (SVM), decision trees (DT), neural networks (NN) and adaptive neuro-fuzzy inference systems (ANFIS).These methods are selected because they have shown their ability to provide an adequate prediction performance. The elements of the developed models are the elements of machine learning method used. For example the element of the model are the elements of neural network (or decision tree …) structure. The failure factors (TABLE I) are the inputs of the model, the extracted dataset is used to train and verify the implemented model, and the failure probability (or project outcome fail/success) is the model output.

We started developing the risk assessment model using six different approaches: Logistic Regression (LR), Naïve Bayes(NB), Support Vector Machine (SVM), Decision Trees (DT), Neural Networks (NN), and Adaptive Neuro Fuzzy Inference System (ANFIS). In this research, the model is the process of project failure prediction which implemented using different machine learning techniques. MATLAB is used to develop and run the models that return a binomial values: 1 (if the result is "Failure") or 0 (if the result is "Success"). Also, the model output can be a failure probability which is ranging from 0 to 1.

IV. DEVELOPING THE MODEL USING MACHINE LEARNING METHODS

A. Developing the Model using Logistic Regression (LR)

LR uses mathematical expression to evaluate the "logit" function.The probability of failure π is founded by taking the inverse of the logit function l:

$$\boldsymbol{\pi = logistic(l) = \frac{e^{l}}{1+e^{l}}} \quad (1)$$

B. Developing the Model using Naïve Bayes (NB)

Naïve Bayes classifier calculates the P(C | X) which is the posterior probability of the class C given the failure factor X , from the prior probability of the class P(C), class prior probability P(X), and the probability of the failure factor X of given class C P(X | C) which is called "Likelihood".

$$\boldsymbol{P(C|X) = \frac{P(X|C)P(C)}{P(X)}} \quad (2)$$

The classification problem of Naïve Bays model can be written as: for each class C, find maximum of α P(C) $\prod_{i=1}^{6}$ P(X_i| C) which can be written as:

$$\boldsymbol{\hat{c} = argmax_C P(C) \prod_{i=1}^{6} P(X_i|C)} \quad (3)$$

For developing the model , we used "NaiveBayes.fit" MATLAB function to build and train the naïve bays classifier, where ObjBayes = NaiveBayes.fit(X,Y) returns NB model (ObjBayes), trained by the failure factors (X1, X2,… X6) in array X and class labels (Failure/Success) in array Y. The naive Bayes model training is based on calculating P(X | Y) which is the probability of failure factor X given class Y. The training function NaiveBayes.fit function provides support for Gausian (normal) distribution as default distribution. This function can support other distributions such as Multivariate multinomial distribution (mvmn) and Multinomial distribution (mn). Posterior (ObjBayes,X) function used to estimate the posterior probability of each class in training data X. The posterior probability is a value between 0 and 1.

C. Developing the Model using Support Vector Machine (SVM):

Given a training dataset containing n samples:

$(x_1^{\rightarrow}, y_1)$, $(x_2^{\rightarrow}, y_2)$, $(x_3^{\rightarrow}, y_3)$, . . . $(x_n^{\rightarrow}, y_n)$

where the data sample $x_i^{\rightarrow}$ is a one dimensional vector containing six failure factors values, and yi is the class to which the sample $x_i^{\rightarrow}$ belongs. The goal of the SVM is to find the optimal hyperplane that accurately separates the data samples. Once the hyperplane is determined, it can be used to make predictions.

In case of linear SVM, the optimal hyperplane can be written as the set of samples $\boldsymbol{x}^{\rightarrow}$ satisfying:

$$\boldsymbol{\omega}^{\rightarrow}.\boldsymbol{x}^{\rightarrow} - \boldsymbol{b} = \mathbf{0} \quad (4)$$

where $\boldsymbol{\omega}^{\rightarrow}$ is the norm vector to the hyperplane and $\frac{b}{|\omega^{\rightarrow}|}$ is the offset of the hyperplane from the origin.

Consider the vector x, the length of this vector (norm) is written as ||x|| can be calculated (for vector points x1, x2,… xn):

$$||\mathbf{x}|| = \sqrt{\boldsymbol{x_1^2} + \boldsymbol{x_2^2} + \cdots + \boldsymbol{x_n^2}} \quad (5)$$

In this research, the SVM model is generated and trained using the following MATLAB function:

SVMMdl = fitcsvm(X,Y,'KernelFunction','rbf');

This function returns a support vector machine classifier SVMMdl trained using the training data contained in the array X. Y is the name of the output variable that contains the class value (1 for failed, and 0 for success). The failure factors data is mapped using radial basis function (rbf).

The following two lines are used to estimate the SVM posterior probability

ScoreSVMModel = fitSVMPosterior(SVMMdl);

[label, probability] = predict(ScoreSVMModel,x);

The SVM posterior probability is a value between 0 and 1 for any data instant.

D. Developing the Model using Decision Trees (DT)

In this research, the decision tree (DT) model is generated and trained using the following MATLAB function:

treeMD = ClassificationTree.fit(x,y);

This function returns a DT classifier treeMD trained using the set of failure factors contained in the array X. Y is the name of the output variable that contains the class value (1 for failed, and 0 for success).

The "predict" method can be applied to predict responses of testing data (xT):

predTesting=tree.predict(xT);

E. Developing the Model using Artificial Neural Networks (ANN)

The following MATLAB function used to establish ANN with 10 hidden layers:

net = feedforwardnet(10,'traingd');

where traingd means that the network is trained with back propagation gradient descent algorithm that updates network weights .

The network is used to map the model inputs (Failure Factors) and outputs (Failure). The network consists of ten of layers. The model input is connected to the first layer, and each of the subsequent layer is connected to the previous layer. The last layer produces the model output.

The following lines used to train the network with training data XN,YN, and predict the labels yNP1 of testing data xTN

net = train(net,XN,YN);

yNP1 = (net(xTN));

The result of prediction is a value between 0 and 1 for any data instant. The default decision threshold 0.5 also is selected as a classification threshold, with failure predicted for any result above 0.5.

F. Developing the Model using Adaptive Neuro Fuzzy Inference System (ANFIS)

According to ANFIS, a combination of backpropagation and least square estimation (LSE) is employed. The idea of using the Backpropagation is to learn premise parameters while using LSE is meant to locate the parameters in the rules' consequents. Two passes could be detected in the learning procedure. Firstly, the forward pass, where node outputs head forward, and consequent parameters {pi, qi, ri} are roughly calculated by least squares procedure, whereas the premise parameters stay fixed. However, in the backward pass, the fault flags are generated backwards, and backpropagation is practiced to adjust the premise parameters {ai, bi, ci}, while sequential parameters remain rigid. This mixture of least-squares and backpropagation techniques are utilized for training FIS membership function parameters to create a given set of input/output data.

In this research, an adaptive network with six inputs is implemented. The identified software project failure factors are used as inputs (predictors). The model predicts the result of the expected class of the software project: failed or success. The Neuro-Fuzzy Design app in MATLAB is used to develop the ANFIS based on the input/output training data set. The membership functions can be tuned automatically using this training data.

The following lines to generate the ANFIS model (fis) and train it with training data DT:

fis = genfis1(data, nMFs,InputMF);

fis = anfis(DT, fis);

To evaluate the output of the six inputs of the testing data in DT we used:

anfis_output = round(evalfis(inT(:,1:6), fis));

The result of prediction (anfis_output) is a value between 0 and 1 for any data instant. The default decision threshold 0.5 also is selected as a classification threshold, with failure predicted for any result above 0.5.

V. EXPERIMENTAL RESULTS

In this research stage, six different prediction techniques were implemented and their performances were compared in terms of their ability to predict software project failures. These techniques were chosen to achieve diversity.

The experiments included calculating seven performance measures that are: sensitivity or recall, specificity, precision, negative predictive value, accuracy, F- measure, kappa coefficient and Area Under Curve (AUC) value. Sensitivity (Recall) also known as True Positive Rate (TPR) is the percentage of correctly identified failed projects (positive class) out of all actual failed projects, true positive/ (true positive + false negative) Specificity also known as True Negative Rate (TNR) is the percentage of actual successful projects (negative classes) that are identified as successful out of all the projects that actually successful, true negative/ (true negative + false positive). Precision value is the probability of correctly predicting the failed projects (positive class), true positive/ (true positive + false positive). Negative predictive value is the probability of correctly predicting the successful projects, true negative/ (true negative + false negative). Accuracy is one of the most widely used performance measures and it is defined as the total number of correct predictions over the overall predictions, (true positive + true negative)/ (true positive + false negative+ true negative + false negative). F-measure combines both sensitivity and precision measures in single measure that specifies how the classification model perfectly capturing sensitivity and precision. κappa takes into account the agreement possibility occurring by chance. The area under curve (AUC) is the area enclosed by the receiver operating curve (ROC) which measures the classification performance at different thresholds, the classifier with AUC =1 is a perfect classifier.

Table II shows summary of performance measure values over training and testing datasets for all prediction models. These values were calculated based on resulted confusion matrices and ROC curves for each model. For overall performance comparison, Table II also presents the average of all performance measure values for each model. According to average performance measure values, LR, SVM and ANFIS provide highest average prediction performance on both testing and training dataset. On the other hand, the NB model has the lowest average prediction performance. The remaining models appear fairly accurate and produce acceptable performance results.

In addition, the performance measures of the models were plotted over testing data. This plot is presented in Fig.1.. According to average performance measure values, LR, SVM and ANFIS provide highest average prediction performance on the testing dataset. On the other hand, the NB model has the lowest average prediction performance. The remaining models appear fairly accurate and produce acceptable performance results.

TABLE II
PREDICTIVE MODELS PERFORMANCE MEASURES OVER TESTING DATASETS

Measure	LR	NB	SVM	DT	ANN	ANFIS
Sensitivity	0. 8 5	0. 7 5	0. 7 7	0. 7 9	0. 6 9	0 .8 1
Specificity	0. 7 7	0. 7 3	0. 9 1	0. 7 8	0. 9 6	0. 9 6
Negative predictive value	0. 8 9	0. 7 9	0. 9 0	0. 8 1	0. 9 6	0. 9 6
Accuracy	0. 7 1	0. 6 8	0. 8 3	0. 7 5	0. 6 7	0. 8 2
Precision	0. 8 3	0. 7 4	0. 8 4	0. 7 8	0. 8 0	0. 8 8
F- measure	0. 8 7	0. 7 7	0. 8 3	0. 8 0	0. 8 0	0. 8 8
Kappa	0. 7 9	0. 4 8	0. 6 9	0. 7 4	0. 6 2	0. 7 7
AUC	0. 9 4	0. 9 0	0. 8 4	0. 8 5	0. 7 2	0. 6 2
Average	0. 8 1	0. 7 2	0. 8 2	0. 7 8	0. 7 7	0. 8 3

.

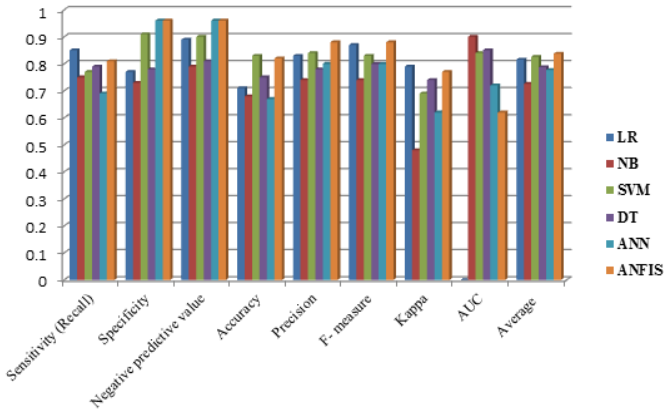

Fig.1: Risk prediction models performance measures over testing dataset

VI. CONCLUSION

In this paper, machine learning methodswere used to predict the system risks by predicting the target class (fail or success) for each data instance (software project) in the dataset according to certain constrains. They are capable of classifying data based on observed information in training set, and then can be used in predicting class labels for newly available information, and making decisions regarding new situation, and assign each case to one of predefined categorical labels. Developing the classification procedure based on training data set is termed as supervised learning.

In this paper, the risk predictive model is developed using six of existing machine learning techniques namely, logistic regression (LR), Naïve Bayes (NB), support vector machines (SVM), decision trees (DT), artificial neural networks (ANN), adaptive neuro- fuzzy inference systems (ANFIS). We suppose that the elements of the developed models were the elements of technique used. For example the element of the model are the elements of neural network (or decision tree …) structure. The identified failure factors were used as the model inputs, the collected failure data was used to train and validate the model, and the project outcome (failed/success) or failure probability is the model output.

We examined the performance of these techniques using different measures such as confusion matrix, sensitivity, accuracy and AUC. The performance results showed that LR, SVM and ANFIS provide highest average prediction performance. On the other hand, the NB model has the lowest average prediction performance. The remaining models appear fairly accurate and produce acceptable performance results. In the future research, the predictive model will be developed based on ensemble learning.

REFERENCES

[1] Clarke, R. (2016). Big data, big risks. Information Systems Journal, 26(1), 77-90.

[2] Big Data Analysis in an Audit Environment-Challenges and Opportunities.2020 Available at SSRN 3688095.

[3] Ibraigheeth, M., & Fadzli, S. A. (2018). Software reliability prediction in various software development stages. Journal of Theoretical and Applied Information Technology, 96(7).

[4] Arora, Y. (2017). Approaches for Enhancing Reliability of Software Product. International Journal of Computer Applications, 161(5).

[5] Singh, N., Verma, P., & Kumar, A. (2016). Software Reliability Models. Available at SSRN 2836387.

[6] Nikam, S. S. (2015). A comparative study of classification techniques in data mining algorithms. Oriental journal of computer science & technology, 8(1), 13-19.

[7] Ibraigheeth, M. A., &Fadzli, S. A. (2020, October). Software project failures prediction using logistic regression modeling. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-5). IEEE.

[8] Yavari, A., Golbaghi, M., &Momeni, H. (2013).DAssessment of Effective Risk in Software Projects based on Wallace's Classification Using Fuzzy Logic. International Journal of Information Engineering and Electronic Business, 5(4), 58.

[9] Hu, Y., Zhang, X., Ngai, E. W. T., Cai, R., & Liu, M. (2013). Software project risk analysis using Bayesian networks with causality constraints. Decision Support Systems, 56, 439-449.

[10] Sangaiah, A. K., Samuel, O. W., Li, X., Abdel-Basset, M., & Wang, H. (2018). Towards an efficient risk assessment in software projects–Fuzzy reinforcement paradigm. Computers & Electrical Engineering, 71, 833-846.

[11] Bilal, M. U. H. A. M. M. A. D., Gani, A. B. D. U. L. L. A. H., Liaqat, M. I. S. B. A. H., Bashir, N. A. U. M. A. N., & Malik, N. A. D. I. A. (2020). Risk assessment across life cycle phases for small and medium software projects. Journal of Engineering Science and Technology, 15(1), 572-588.

[12] Suresh, K., &Dillibabu, R. (2018). Designing a machine learning based software risk assessment model using Naïve Bayes algorithm. TAGA J, 14, 3141-3147.

[13] Ibraigheeth, M., &Fadzli, S. A. (2019). Core factors for software projects success. JOIV: International Journal on Informatics Visualization, 3(1), 69-74.