A Predictive Risk Model for Software Projects' Requirement Gathering Phase

Beatrice O. Akumba[1], Samera U. Otor[1], Iorshase Agaji[2] and Barnabas T. Akumba[3]
[1]Department of Mathematics/Computer Science, Benue State University Makurdi, Nigeria
[2]Department of Mathematics/Statistics/Computer Science, Federal University of Agriculture Makurdi, Nigeria
[3]Management Sciences for Health (MSH), Abuja.

Abstract:- The initial stage of the software development lifecycle is the requirement gathering and analysis phase. Predicting risk at this phase is very crucial because cost and efforts can be saved while improving the quality and efficiency of the software to be developed. The datasets for software requirements risk prediction have been adopted in this paper to predict the risk levels across the software projects and to ascertain the attributes that contribute to the recognized risk in the software projects. A supervised machine learning technique was used to predict the risk across the projects using Naïve Bayes Classifier technique. The model was able to predict the risks across the projects and the performance metrics of the risk attributes were evaluated. The model predicted four (4) as Catastrophic, eleven (11) as High, eighteen (18) as Moderate, thirty-three (33) as Low and seven (7) as insignificant. The overall confusion matrix statistics on the risk levels prediction by the model had accuracy to be 98% with confidence interval (CI) of 95% and Kappa 97%. It was seen that with respect to the risk levels across the projects, probability and priority are the most significant variables in predicting Risk levels. Software Project Managers should put in place measures to reduce the factors (probability and priority) that increase the chances of occurrence of the identified risks.

Keywords:- *Naïve Bayes Classifier, SDLC, Risk Prediction, Software Projects, Risk Outcomes, Risk Levels.*

I. INTRODUCTION

The development of most software projects is mostly characterised by high failure rates. These failure rates are attributed to the uncertain events that occur in the Software Development Life Cycle (SDLC) process and as a result of which has led to the to the potential loss of software in most organisations. These uncertain events/occurrences are referred to as risks and they emanate from divergent risk factors that are embedded in the heterogeneous activities of the software development lifecycle. There is need for these risks to be identified timely else they become the cause of the software project failure (Salih and Ammar, 2017).

According to several surveys carried out by the Standish Group of Companies, 16% of software projects are on time and on budget, 52.7% are delivered with less functionality/performance and 31.1% are scrapped before completion by most organisations (Standish Group of Companies, 2019). These are attributed to inherent risks found in most software projects. Thus, there is need for these risk factors to be identified and mitigated through risk predictions before they become a threat to the software projects.

The requirement gathering and analysis phase is the first stage of the software development lifecycle. Cost and efforts can be salvaged if the risk at this stage is predicted. It will effectively lessen the occurrences of software project failures too.

A lot of methods have been developed for predicting risk in the SDLC ranging from models (Hu *et al.*, 2009), to several machine learning methods (Hu *et al.*, 2015) among others. The Artificial Neural Network (ANN) and Support Vector Machine (SVM) were used to predict and manage software development risks in an entire project. They formulated a model for identifying risk and used the data gathered through questionnaires that were administered to software development companies to develop the risk prediction model. (Hu *et al.*, 2009). Also, the ensemble learning techniques was used to predict risk in software projects by using classifier ensemble methods of decision trees (DT) based on bagging and SVM. The data used was also gathered through questionnaires.

Nevertheless, as most data used for risk prediction of most software projects were gotten through questionnaires by most researchers, Shaukat *et al.*, (2018) developed a dataset that accommodated most software requirements and their attributes that are required and necessary for the prediction of risks in most software projects. This is because they noted that infrequent techniques to predict risks at the requirement gathering stage/phase of the SDLC existed and as such, there were no datasets containing risk attributes for software risk prediction. They went further to developed a dataset based on the Software Requirements Specification (SRS) of new projects and used classification techniques to refine the data.

It is to this end that in this paper, the datasets by Shaukat *et al.*, (2018) has been adopted to carry out software risks predictions in the requirement gathering phase of the SDLC. A model was created and trained using machine learning techniques to predict the risks across the projects and the performance metrics of the risk attributes were evaluated with respect to the risk levels across the projects. The model was formulated and trained using the Naïve Bayes classifier and the predictions across the projects showed that the probability and priority variables were highly significant in the prediction of risks in software projects.

The rest of this paper is organised as follows: in section 2, we review some related literatures in the subject area, while section 3 describes the materials and methods used for the study. Section 4 is centered on discussions and results and section 5 is conclusion and further work.

II. LITERATURE REVIEW

Hu *et al.,* (2009) proposed that software project development process is associated with some levels of risks and are accompanied by high failure rates. In order to mitigate the risks, they proposed an intelligent model that was able to predict and manage the risks inherent in all software projects. The model developed was formal and intelligent enough to identify risk by gathering real life instances from software development companies using machine learning algorithms of Artificial Neural Networks (ANN) and Support Vector Machine (SVM). The model was able to predict risk in all the projects through the data gathered from the questionnaires that were used to collect data as there were no existing dataset to be used for the prediction. Kawamuram *et al.,* (2017) noted that, over 70% of most software-based project development ends up as failures due to risk. In order to enable successful software project rates, they recommended some software risks predictions for various Information Technology (IT) vendors and organisations by collecting data of 332 projects of different IT vendors through survey over the internet. They developed a success/failure rate risk prediction algorithm using Naïve Bayes classifier techniques on the data collected. They concluded that the prediction of the successful/failure rates of software projects helped immensely to identify the projects that organisations needed to know if they are risk prone or not in an order of priority. However, the study was not able to show the success/failure rate at each level of the SDLC process of projects.

Salih and Amar (2017) also emphasised that the growth in the complexity of software systems has made software performance predictions a difficult task. They addressed the problem using a model-based approach for resource utilization and software performance risk prediction. They employed machine learning techniques to predict the performance risk and resource utilisation.

Also, Christiansen *et al.,* (2015) in their study used multiple logistic regression to predict software development risk based on data gathered from questionnaires from some experts which analysed the factors of risk stratification and risk factors. They employed statistical integration to illustrate the risk factors which were anticipated and managed by minimizing the risk that occurred during the software development processes. A combination of factor analysis and logistic regression were used to predict the classification probability of the failure or success of software projects. Their study concluded by stressing that there are risk factors which are inherent in software development processes and must be known and addressed to enable software projects to be delivered and completed on time. Their data was gathered through questionnaires that were administered to experts because there were no dataset templates available for software projects development life cycle phases.

However, Shaukat *et al.,* (2018) recognized the requirement gathering phase of the SDLC as the most important and demanding phase amongst the other phases. They recognized that there was no explicit dataset from real life software projects that contained all the attributes of software requirements and their risks which can be used to predict risks in new software projects. They proposed a dataset for the requirement gathering phase of the SDLC which contains the requirements gotten from the Software Requirements Specification (SRS) of some software projects and their risk attributes. It also has the correlation between the requirements and risks. The dataset developed is an apt template that can be used as a tool for the prediction of risks in software projects. The dataset was subjected to three preprocessing filter techniques of Normalisation, Standardisation and Discretisation to get a better accuracy from unsupervised learning preprocessors in WEKA. The dataset is a clean and accurate template and also acts as a data source to be used by researchers for most risk decision support systems and for predicting risk at the requirement gathering phase of the SDLC.

However, this paper seeks to adopt the dataset developed by Shaukat *et al.,* (2018) to predict the risk levels and the attributes that contribute to the recognized risk in the software projects. It is worth noting that the significant difference between this paper and the work of Shaukat *et al.*, (2018) is the prediction of the risk levels across the projects in the requirement gathering phase using the dataset developed by Shaukat *et al.*, (2018) as the risk level predictions were not done in their study.

III. MATERIALS AND METHODS

Machine Learning techniques are powerful tools for software risk predictions. The supervised machine learning technique has been employed in this paper using the Naïve Bayes classifiers machine learning algorithm. Supervised learning is a learning model that is developed to make predictions based on an unforeseen input instance. A supervised learning algorithm accepts a known set of input dataset which are known responses to the data (output) to learn the classification/regression model. It is a learning algorithm that trains a model to produce a prediction for the response to new data or the test dataset. Supervised learning uses classification algorithms and regression techniques to develop predictive models. Classification task predicts discrete responses and it is recommended and applied for only data that can be categorized, tagged, or separated into specific groups or classes.

Naïve Bayes classifiers are statistical classifiers that assume that the probability of a given record in the dataset belongs to a particular class and can be predicted using the class membership probabilities of the classifier. Naïve Bayes classification is based on Baye's theorem. The Naïve Bayes classifier requires that all the variables must be categorical. It is purely probabilistic and each independent variable contributes to a decision. It produces models based on the combined probabilities of a dependent variable being associated with the different categories of the dependent variables. The Naïve Bayes Classifier used five levels of categories to represent the risk outcome levels and they are Catastrophic, High, Moderate, Low and Insignificant respectively.

R programming language was used for the implementation and testing of the risk outcome prediction model. R programming is one of the promising languages for machine learning and data science as it provides excellent visualization features which are essential to explore the data before pushing it to any automated learning and assessing the results of the learning algorithm.

A. The Dataset Used for the Risk Prediction

The dataset used for this study was derived from the Dataset on "Software Requirement Risk Prediction" from https://doi.org/10.5281/zenodo.1209601 (Shaukat *et al.*, 2018). The dataset provided 299 data instances for risk prediction at the initial phase of the software development lifecycle (requirement gathering phase). The dataset consists of twelve (12) variable attributes and one (1) target variable (outcome). The attributes of probability, magnitude of risk, impact, priority and risk levels were modified to reflect categorical values that can be used by the Naïve Bayes Classifier as the classification task predicts discrete responses and it is recommended and applied for only data that can be categorized, tagged, or separated into specific groups or classes. The risk level attribute was included to serve as the target variable.

B. The Algorithm of the Risk Prediction Model

The algorithm of the Risk Prediction Model is described using a flowchart as shown in Figure 1. The flowchart of the Risk Outcome Prediction Model using Naïve Bayes Classifier is used to show the step by step method used to achieve correct predictions on the given dataset adopted.

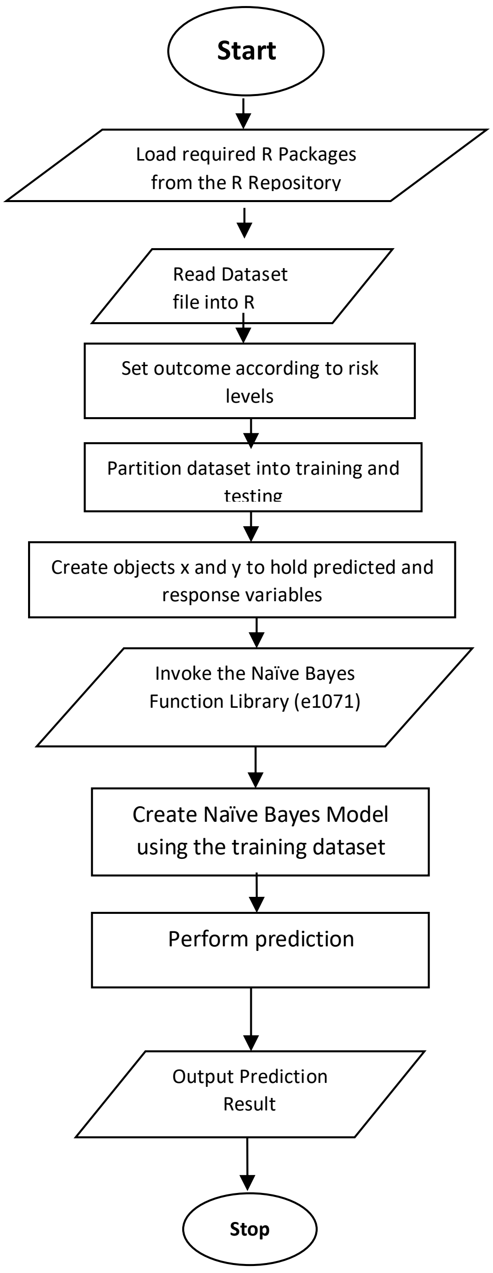

Start
Load required R Packages from the R Repository
Read Dataset file into R
Set outcome according to risk levels
Partition dataset into training and testing
Create objects x and y to hold predicted and response variables
Invoke the Naïve Bayes Function Library (e1071)
Create Naïve Bayes Model using the training dataset
Perform prediction
Output Prediction Result
Stop

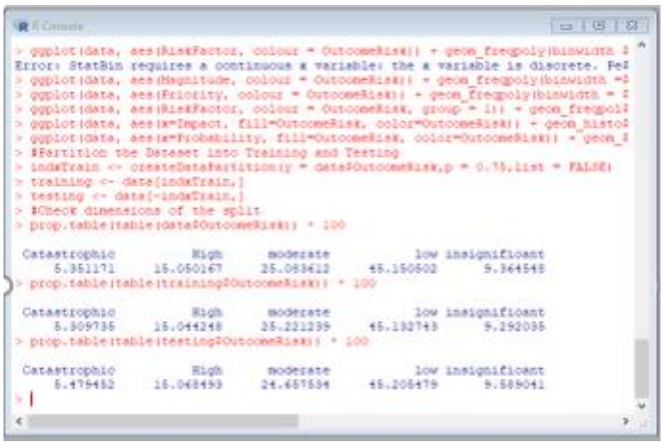
Fig 1:- The Flowchart of the Risk Outcome Prediction Model using Naïve Bayes Classifier

IV. RESULTS AND DISCUSSION

The results of this paper were analysed using R and their respective interfaces are screenshot and presented for discussions. The discussions are based on the dataset partition, the risk outcome prediction model, the risk outcome prediction model performance and analysis and the overall performance plot results are being discussed respectively with figures to depict them accordingly.

A. The Dataset Partition (Training and Testing)

The dataset used for the risk prediction model was split into two phases. They are; the training and the test instances. 75% of the dataset was used for training the Risk Outcome Prediction Model while 25% of the dataset was used for the testing and validation of the model. Figure 2 shows the split dataset and their corresponding properties in a tabular form using prop.table in R.

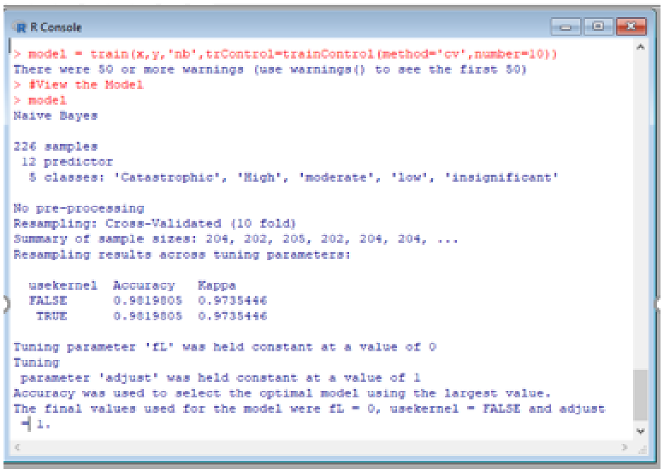


```
> ggplot(data, aes(RiskFactor, colour = OutcomeRisk)) + geom_freqpoly(binwidth $
Error: StatBin requires a continuous x variable: the x variable is discrete. Pe$
> ggplot(data, aes(Magnitude, colour = OutcomeRisk)) + geom_freqpoly(binwidth =$
> ggplot(data, aes(Priority, colour = OutcomeRisk)) + geom_freqpoly(binwidth = $
> ggplot(data, aes(RiskFactor, colour = OutcomeRisk, group = 1)) + geom_freqpol$
> ggplot(data, aes(x=Impact, fill=OutcomeRisk, color=OutcomeRisk)) + geom_histo$
> ggplot(data, aes(x=Probability, fill=OutcomeRisk, color=OutcomeRisk)) + geom_$
> #Partition the Dataset into Training and Testing
> indxTrain <- createDataPartition(y = data$OutcomeRisk,p = 0.75,list = FALSE)
> training <- data[indxTrain,]
> testing <- data[-indxTrain,]
> #Check dimensions of the split
> prop.table(table(data$OutcomeRisk)) * 100

 Catastrophic         High     moderate          low insignificant
     5.351171    15.050167    25.083612    45.150502      9.364548
> prop.table(table(training$OutcomeRisk)) * 100

 Catastrophic         High     moderate          low insignificant
     5.309735    15.044248    25.221239    45.132743      9.292035
> prop.table(table(testing$OutcomeRisk)) * 100

 Catastrophic         High     moderate          low insignificant
     5.479452    15.068493    24.657534    45.205479      9.589041
> 
```

Fig 2:- The Partitions of the Train and Test Dataset Instances and their Dimensions

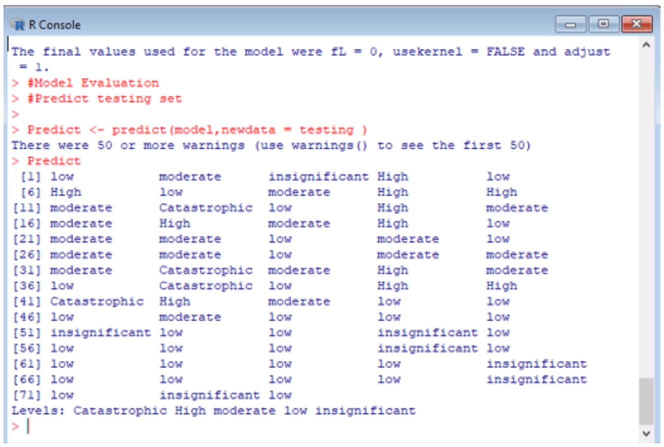
B. The Risk Outcome Prediction (RISOP) Model using Naïve Bayes Classifier

A Risk Outcome Prediction Model based on Naïve Bayes Classifier for the prediction and analysis of the risk in the software projects was created as shown in Figure 3. The model has 226 dataset samples corresponding to the 75% of the dataset that was partitioned for the training of the model. The Model has 12 predictors corresponding to the attributes. Also, Ten-fold (10-fold) Cross-Validation was applied and the accuracies of the model stood at 98% and Kappa was 97%. The model was able to perform prediction on the new data to test the risk levels across the datasets. It was able to evaluate the risk levels by indicating whether they are insignificant, low, moderate, high or catastrophic across the software projects. The models' evaluation and prediction testing set are as shown in Figures3 and 4.



```
> model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
There were 50 or more warnings (use warnings() to see the first 50)
> #View the Model
> model
Naive Bayes

226 samples
 12 predictor
  5 classes: 'Catastrophic', 'High', 'moderate', 'low', 'insignificant'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 204, 202, 205, 202, 204, 204, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9819805  0.9735446
   TRUE      0.9819805  0.9735446

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust
 = 1.
```

Fig 3:- The Risk Outcome Prediction (RISOP) Model

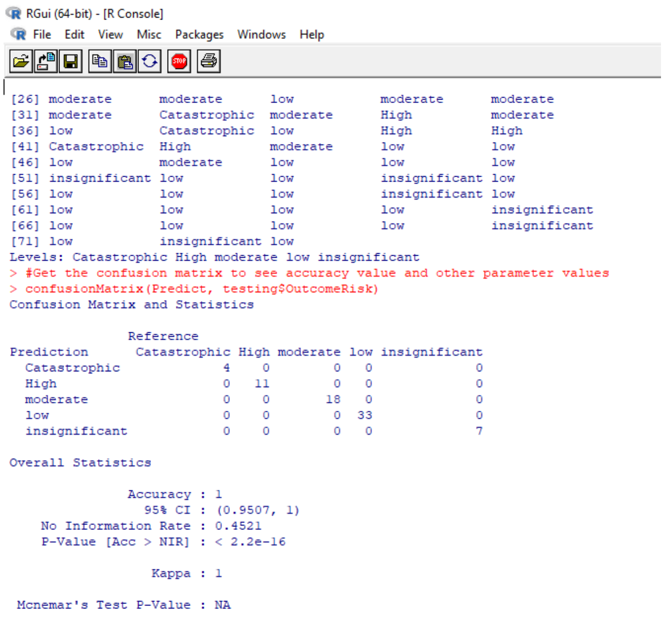


```
The final values used for the model were fL = 0, usekernel = FALSE and adjust
 = 1.
> #Model Evaluation
> #Predict testing set
>
> Predict <- predict(model,newdata = testing )
There were 50 or more warnings (use warnings() to see the first 50)
> Predict
 [1] low           moderate      insignificant High          low
 [6] High          low           moderate      High          High
[11] moderate      Catastrophic  low           High          moderate
[16] moderate      High          moderate      High          low
[21] moderate      moderate      low           moderate      low
[26] moderate      moderate      low           moderate      moderate
[31] moderate      Catastrophic  moderate      High          moderate
[36] low           Catastrophic  low           High          High
[41] Catastrophic  High          moderate      low           low
[46] low           moderate      low           low           low
[51] insignificant low           low           insignificant low
[56] low           low           low           insignificant low
[61] low           low           low           low           insignificant
[66] low           low           low           low           insignificant
[71] low           insignificant low
Levels: Catastrophic High moderate low insignificant
> |
```

Fig 4:- Model Evaluation and Prediction based on the Testing Instance

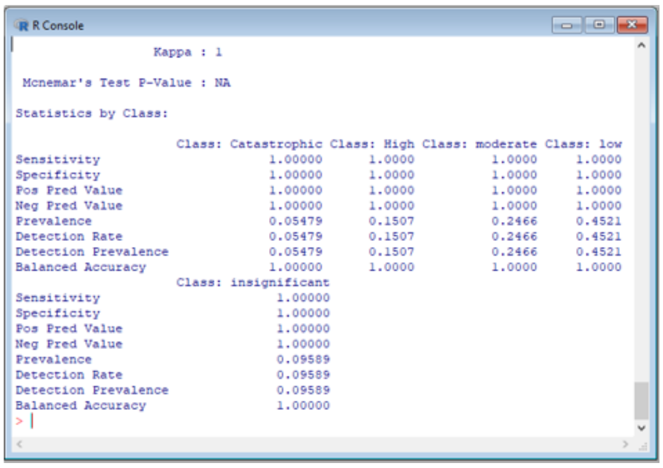
C. The Performance and Analysis of the RISOP Model

The developed RISOP Model was analysed in R. The confusion matrix was derived from the model on the predicted risk outcome levels as shown in Figure 5. The risk levels predicted indicated that four (4) were truly classified as Catastrophic, eleven (11) as High, eighteen (18) as Moderate, thirty-three (33) as Low and seven (7) as insignificant. The overall confusion matrix statistics on the risk levels prediction by the model had its accuracy to be 100% with confidence interval (CI) of 95% and Kappa 100%. The statistics by class for the risk levels with respect to sensitivity and specificity were also derived and they stood at 100% respectively. This can be deduced from Figure 6.


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

```
[26] moderate      moderate      low           moderate      moderate
[31] moderate      Catastrophic  moderate      High          moderate
[36] low           Catastrophic  low           High          High
[41] Catastrophic  High          moderate      low           low
[46] low           moderate      low           low           low
[51] insignificant low           low           insignificant low
[56] low           low           low           insignificant low
[61] low           low           low           low           insignificant
[66] low           low           low           low           insignificant
[71] low           insignificant low
Levels: Catastrophic High moderate low insignificant
> #Get the confusion matrix to see accuracy value and other parameter values
> confusionMatrix(Predict, testing$OutcomeRisk)
Confusion Matrix and Statistics

               Reference
Prediction      Catastrophic High moderate low insignificant
  Catastrophic             4    0        0   0             0
  High                     0   11        0   0             0
  moderate                 0    0       18   0             0
  low                      0    0        0  33             0
  insignificant            0    0        0   0             7

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9507, 1)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA
```

Fig 5:- The Confusion Matrix of the Risk Prediction Model



```
                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Catastrophic Class: High Class: moderate Class: low
Sensitivity                      1.00000      1.0000          1.0000     1.0000
Specificity                      1.00000      1.0000          1.0000     1.0000
Pos Pred Value                   1.00000      1.0000          1.0000     1.0000
Neg Pred Value                   1.00000      1.0000          1.0000     1.0000
Prevalence                       0.05479      0.1507          0.2466     0.4521
Detection Rate                   0.05479      0.1507          0.2466     0.4521
Detection Prevalence             0.05479      0.1507          0.2466     0.4521
Balanced Accuracy                1.00000      1.0000          1.0000     1.0000
                     Class: insignificant
Sensitivity                       1.00000
Specificity                       1.00000
Pos Pred Value                    1.00000
Neg Pred Value                    1.00000
Prevalence                        0.09589
Detection Rate                    0.09589
Detection Prevalence              0.09589
Balanced Accuracy                 1.00000
>
```

Fig 6:- The Statistics by Class for the Risk Levels

D. The Overall Variable Performance Plot of the Model

A plot of the performance matrix of the variables shows that probability and priority are the most significant variables in the prediction of the risk levels across the projects in the dataset used. The variables were seen to have high values across the projects and they indicated how much they contribute to the inherent risk in most software projects. With the performance matrix plots, it will enable the software project managers to ensure that they put in place measures to reduce the identified variables in any of their software projects. Figure 7shows the plot of the performance matrix of the variables.

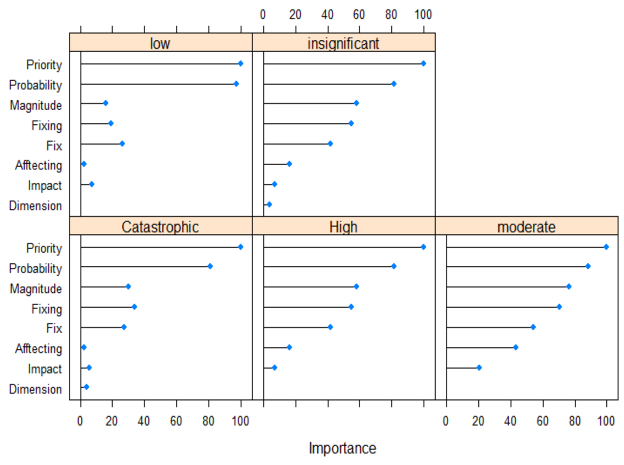


Fig 7:- Plot of the Performance Matrix of the Variables across the Project

V. CONCLUSION

The aim of this paper is to use Naïve Bayes Classifier to predict risks that occur during the requirement gathering phase of the software development lifecycle (SDLC) in some Software Projects. The requirement gathering stage is a vital stage in the SDLC. If risks can be identified and managed early, it will go a long way to improve software quality and save cost. This will also lead to timely delivery of such projects. It is to this end that a dataset for software requirement risk prediction was gotten from the requirement gathering phase of five major projects. The dataset was used to build a Naïve Bayes model and was trained to predict the risk levels based on attributes such as magnitude, impact, probability, priority and the risk dimension to determine whether they were catastrophic, high, moderate, low or insignificant. The confusion matrix derived from the model predicted four (4) correctly predicted data as Catastrophic, Eleven (11) as High, Eighteen (18) as Moderate, Thirty-three (33) as Low and Seven (7) as Insignificant. The Risk Outcome Prediction (RISOP) Model had a plot of the performance matrix of the variables to enable us know the variable that was common and highest across the software project. It was seen that probability and priority are the most significant variables in predicting risk levels across the project. If the probability of the risk occurring is high and the priority is also known in advance, they can salvage the entire project from failures. But if they are neglected, it will be disastrous as the projects will not meet their timelines and will be delivered with reduced functionalities. Therefore, Software Project Managers should put in place measures to reduce the probability of the identified variables of probability and priority of risk occurrences in their software projects to foster full functionalities of the systems delivered.

It is known that research is an ongoing process that does not terminate, and as such, we advise that further work should be done to ascertain the minimal values that each of the variables must hold before that can be said to have contributed to the risk levels across the projects. The model was able to predict risk levels across projects in the requirement gathering phase of the software development life cycle, predictions can be done on other phases of the software development lifecycle such as the design, coding, testing and maintenance phases to ascertain the attributes that lead to risk in most software projects. Also, benchmarking this model on the requirement gathering phase with the datasets gotten from the other phases of the software development lifecycle should be done to ascertain the phase with the highest risk levels in the software projects.

REFERENCES

[1]. Christiansen, T., Wuttidittachotti, P., Somchai, P. and Vallipakorn, S.A. (2015). Prediction of Risk Factors of Software Development Project by using Multiple Logistic Regression. *ARPN Journal of Engineering and Applied Sciences. 10(3),1324 -1331.*

[2]. Hu, Y., Feng, B., Mob, X., Zhang, X. Z., Ngai, E. W. T., Fan, M. and Liu, M. (2015). *Elsevier Journals on Decision Support Systems.* 72, 11-23 [online]. Retrieved on 9th August, 2019 from https://www.sciencedirect.com/science/article/pii/S0167923615000238

[3]. Hu, Y., Zhang, X., Sun, X., Liu, M. and Du, J. (2009). An Intelligent Model for Software Project Risk Prediction. International in *Conference on Information Management, Innovation Management and Industrial Engineering.*(p. 629 [online] retrieved on the 5th of May, 2019 from https://ieeexplore.ieee.org/abstract/document/5368175

[4]. Kawamura, T., Toma, T. and Takano, K. (2017). Outcome Prediction of Software Projects for Information Technology Vendors. *Conference Proceedings of the 2017 IEEE IEEM.*[online] retrieved on the 9th of May, 2019 fromhttps://ieeexplore.ieee.org/document/8290188

[5]. Salih, H. A. M. and Ammar, H.H. (2017). Model-Based Resource Utilisation and Performance Risk Prediction Using Machine Learning Techniques. *International Journal on Informatics Visualisation (JOIV). 1(3), p 101-109.*

[6]. Shaukat, Z., Naseem, R. and Zubar, M. (2018). A Dataset for Software Requirement Risk Prediction. In *IEEE International Conference on Computational Science and Engineering. (pp. 112-118)*[online] retrieved on the 9th of May, 2019 fromhttps://zenodo.org/record/1209601#.Xs5psmhKjIU

[7]. The Standish Group, “Extreme chaos”.www.standishgroup.com, 2019. [online] Retrieved on the 10th of March, 2019 from https://www.projectsmart.co.uk/white-papers/chaos-report.pdf