IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

A Data-Mining Technique-Based Software Maintenance Prediction System

1. Manoj pal 2.Satvir Quantum University Roorkee, India

Abstract- In today's fast-paced industrial environment, maintenance has become an important aspect to ensure equipment reliability and optimize production processes. This paper proposes a new maintenance prediction system based on data mining technology, using Data mining and machine learning algorithms to predict equipment failures and reduce downtime. By integrating historical maintenance data, real-time sensor readings and expert knowledge, our systems can identify patterns and anomalies to predict potential failures. We use [insert metric/dataset] to evaluate the performance of the system and demonstrate its effectiveness in improving maintenance schedules, reducing costs and increasing overall equipment efficiency. The proposed system provides a scalable and adaptable solution for industries seeking to adopt data-driven maintenance strategies.

Keyword- Data mining, Maintenance prediction system, production optimization, Predictive maintenance

I.INTRODUCTION

In the rapidly developing field of industrial maintenance, it is imperative to transform from a passive strategy to a proactive strategy. The consequences of equipment failure and downtime can be devastating, resulting in huge financial losses, lost productivity and damaged reputation. Traditional maintenance methods rely on periodic intervention and time-based replacement, often failing to address the complexities of modern industrial assets. The emergence of data mining and machine learning technologies has paved the way for a paradigm shift in maintenance practices.
In the era of Industry 4.0, the pursuit of operational excellence has become a paramount objective for industries worldwide. The advent of advanced data analytics and machine learning has revolutionized the realm of maintenance prediction, enabling organizations to transition from reactive to proactive maintenance strategies. This research paper presents a novel Data-Mining Technique-Based Maintenance Prediction System, designed to harness the power of data-driven insights and forecast equipment failures with precision. By integrating historical maintenance data, real-time sensor readings, and expert knowledge, our proposed system aims to optimize maintenance scheduling, reduce downtime, and enhance overall equipment effectiveness. This research endeavors to address the pressing need for a reliable, efficient, and data-driven maintenance solution, capable of catering to the evolving demands of modern industries.

1.1 SOFTWARE QUALITY

Over the past four decades, the nature and complexity of software has transformed dramatically. The ISO/IEC 9126 standard defines software quality as a combination of six quality parameters. Functionality, reliability, availability, efficiency, maintainability and predictability are components of quality. From these main quality factors, 27 sub-quality factors are derived.
Maintainability is defined as ―the degree of ease with which a system can make changes that may affect components, functionality, and interfaces+-9 when changing functionality or fixing bugs.‖ Maintainability is a quality attribute of better design [1]. One of the most important characteristics of a

software system is its maintainability. One of the key features that may lower the price of a program is maintainability [2]. There is a misconception that high-quality software is easy to maintain, meaning that fixing bugs will take minimal time and effort, resulting in software that is both presentable and easy to maintain [3]. To date, various maintainability prediction models have been proposed to help engineers determine the maintainability of software in order to better create and enhance software systems.

1.2 SOFTWARE MAINTENANCE

The definition of software maintenance is the act of modifying a software system or component Fix bugs, enhancements or other features after delivery or adjust to new environments.
Software maintenance belongs to the category of Software Development Life Cycle. Its main purpose is to update and change software applications after they are delivered in order to fix bugs and boost efficiency. Software is a set of instruction of the real world. When the real world changes, the software requires alteration wherever possible [4].

Modern SDLCs typically accept software maintenance as a component. It refers to all changes and updates made to the software product after it has been delivered. There are many reasons why modification is needed, some of which are briefly discussed below:

✓ **Market Conditions** - Changing policies over time may create the need to adapt, such as taxes and newly imposed restrictions (such as how to keep up with bookkeeping).

✓ **Customer Demand** - Over time, users may request additional features or functionality from the software.

✓ **Modifications to Hosts** - Software must be updated to maintain flexibility in the event of any changes to the target host's hardware, operating system, or platform.

✓ **Organizational Changes** - If the client experiences any business-level changes, such as a decline in organizational strength, acquisition of another company, or entry into a new industry, the original plan may need to be modified.

1.3 SOFTWARE MAINTAINABILITY

One of the most critical factors to consider when evaluating the quality of a software product is its maintainability. The ease with which a software system or component can be changed to fix bugs, enhance performance or other functionality, or adapt to changing circumstances is called software maintainability [5].
Software Maintenance is a valuable phase in which most the costs occur during perfective maintenance. Furthermore possibly the most time-consuming activity is trying to understand the intent of the original developers. As maintainability is about trying to make maintenance as easy and cheap as possible, the aforementioned two facts shall serve as a guide. First, the software design should be as easy to understand as possible and second, it should be as easy to modify as possible. It would appear, though, that understand ability is a prerequisite for any type of maintenance and therefore it will take precedence over 13 modifiability at least in the initial analysis.
I expect that there might be conflict among the two quality attributes sometime when, for instance, the Visitor design pattern aims for certain easy modifications but there is evidence that shows that it is hard to understand Observably, it should be mentioned that whatever kind of maintenance is being performed, the skill level of the maintainers is a major factor.

1.4 MOTIVATION

There is an increasing demand for significant change to software systems to meet ever- changing user requirements and specifications. Such changes include fixing errors, adding Enhancements and

making optimizations. Software maintenance Software maintenance involves making such changes to existing software. is among the significant phases during the software development life-cycle. The largest expense in the life cycle of a software product is software maintenance.
Most of the money spent on software development goes into software maintenance. Several surveys show that software maintenance costs account for 60% to 80% of lifetime expenditures. These polls additionally demonstrate that a significant amount of maintenance costs are due to software enhancements rather than corrections.
The majority of issues related to software maintenance capable of being located to deficiencies within the software development process. Generally inside the software engineering process, the development part of the software is completed by a software programmer who might not contribute to the maintenance part. Here, the maintenance engineer, responsible for maintaining the software or application, has to get acquainted with the detailed design, relevant documentation, functioning of the source code etc. Also, the estimation of maintenance cost and effort is a very essential component of the organizing process since maintenance costs amount for a significant portion of the overall software development life-cycle cost. Also, estimate of upkeep work helps to plan adequate maintenance staff.
Software maintenance involves many management and technical issues. The more difficult issues of impact analysis, regression testing, and program comprehension are the ones that drive up the price of software maintenance.

1.5 OBJECTIVE OF RESEARCH PAPER

Software metrics can be used in the design stage of development as a means of managing the cost and effort of maintenance. Early in the development process, software metrics-based maintenance prediction can assist developers not only better understand the factors that influence program quality, but also provide guidance on design or coding. Additionally, it can supply managers with helpful data to aid in the planning of the utilization of priceless resources [6].
To address this problem quantitatively, the main aim/goal of this research is to propose the use of several machine learning algorithms aimed at predicting maintainable classes using object-oriented metrics for object-oriented systems. The proposed models are classification technique, Neural Network and Support Vector Machine. This research also focuses on a collection of object-oriented measures that are useful for gauging a technology's maintainability. Proposed models are also contrasted further.
The main objectives are as follows:

✓ To study various Data Mining Techniques.
✓ To identify Object Oriented Metrics to forecast an object-oriented system's maintenance workload.
✓ To compare and analyse Data Mining Techniques to predict and classify the upkeep undertaking of a classes of object-centric System.
✓

2. LITERATURE SURVEY

The ability of a system to adapt to changes in functionality or to correct problems with some degree of ease is called maintainability, which affects components, functionality, and interfaces. Maintainability is a quality attribute of good design. One of the most important aspects of a software system's quality is its maintainability. Software costs can be reduced by implementing key quality criteria such as maintainability. High-quality software is considered easy to maintain, meaning that fixing bugs will take minimal time and effort, resulting in software that is both presentable and easy to maintain. To help engineers better develop software by evaluating its maintainability, many predictability models have been proposed so far.
The process of making informed estimates of outcomes based on past data is called forecasting. While it's often impossible to guarantee accurate predictions, having precise results can aid planning and reduce the effort required to fix software defects. To this end, this article analyses the prediction results from the perspective of software maintainability accuracy.
Many prediction models and methods for software maintainability were created. It is necessary to be able to measure software maintainability before we can forecast its maintainability. Software maintainability has been calculated using a number of models that have been put out. The aggregated data from the maintainability models used to determine the measurement aim was displayed in Table 2.

S.No.	Model	Maintainability Measurement
1	McCall	Modularity, clarity and simplicity
2	Boehm	Testability, adaptability and understand ability
3	Sneed-Forgiveness	Average number of variables, average cyclomatic complexity (ACC), comment rate (CR)
4	Li-Henry	NOC, DIT, RFC, WMC, CBO, LCOM and NOM
5	Marcela Genero	Dimensionality and complexity of UML class diagrams
6	Aggarwal	Artificial Neural Networks: LCOM, DIT, WMC, NOC, RFC, DAC, MPC, NOM and CHANGE
7	Koten-Gray	DAC, DIT, MPC, WMC, LCOM and LOC, NOMBayesian Network
8	Zhou-Leung	Multiple regression
9	The Malhotra Framework	Group Data Processing Method (GMDH)
10	Dubey	Neural network with multilayer perceptron (MLP)

TABLE 2: Review the maintainability model

McCall's Model-In order to improve the quality of software products, Jim McCall developed the McCall model and first proposed it in 1977 [7]. He categorized three major quality criteria for software products—product operations, product transitions, and product revisions—into this model. Change capability is another definition of product revision. Product transformation refers both to the ability to adapt to changing circumstances and to the fundamental elements that make a product work. From these attributes, 11 quality factors were derived, as shown in Figure 1. McCall defines one or more criteria for each quality factor that are used to evaluate the quality of a software product. Modularity, simplicity, and simplicity are the criteria used to calculate maintainability, which is one of the quality elements of the model [8].
The concept underlying this model is that the quality factors mentioned could provide the whole picture of a software product. [9]. "Yes" or "No" answers to questions related to each other will determine quality indicators. If the "yes" and "no" answers are the same, we meet 50% of the quality requirements. The results obtained by this model may vary because the interpretation of the data depends on the opinion of one or more respondents on these questions. The problem with this model is that there is no set or standardized technique for evaluating quality parameters in this model, which makes measurement challenging [9]. This paradigm uses modularity, simplicity, and simplicity to measure maintainability.

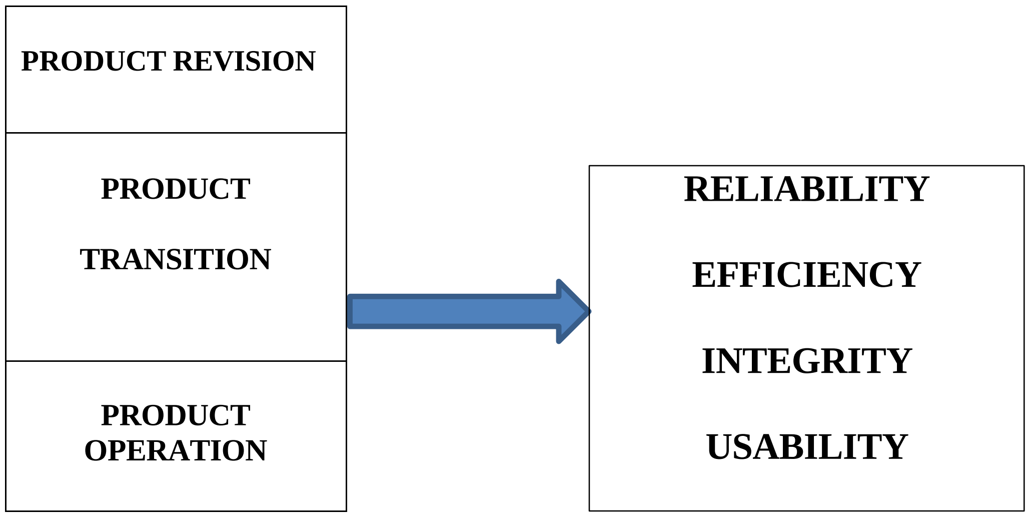


Figure 2.1: McCall's Qualitative Elements

Boehm Model-Boehm expanded on McCall's model by incorporating additional elements such as practicality and emphasizing that "in 1978, Barry Boehm popularized this high- quality model." Boehm's concept includes several additional elements of McCall's model, including practicality and a focus on software maintainability [11]. Boehm expanded on McCall's model by incorporating additional elements such as practicality and emphasizing that "in 1978, Barry Boehm popularized this model of quality." Through this approach, Boehm created a hierarchical model of software product quality that enables quantitative assessment of software quality. Additionally, he developed methods to evaluate software products based on the program's usefulness. Boehm extended McCall's model by incorporating other elements like usefulness and stressing in 1978, Barry Boehm promoted this high-quality model. As [5] "an ease of identifying what need to be changed," he defined maintainability. He identified testability, modifiability, and understands ability as sub- characteristics of maintainability. Although this paradigm is highly challenging to use in real- world situations, it is quite easy to grasp and master.

Sneed-Forgiveness Model- In 1985, Sneed and Forgiveness put up a fuzzy model. This model took into account design features [12]. Their model combines three parameters: average cyclic complexity (ACC), review ratio (CR), and average number of variables [13]. They also proposed maintainability metrics. The way they define maintainability is "a measure of the effort required to keep a system running relative to the effort required to develop it" [11]. They established a few sub-criteria for maintainability, including cohesion, readability, and quality of documentation. This model states that improved readability and thus better maintainability are associated with less CR.

Li-Henry Model- 1993, Sally Henry and Li Wei [14] developed a model that measured software maintainability using the concept of a regression model. Additionally, they identified a few software measures that can be utilized to forecast maintainability. Wake and Henry in 1988 [15], demonstrated how software metrics may be used to achieve software maintainability prediction results. Li and Henry present software metrics for maintainability prediction as a result of all those preliminary findings, despite the fact that only few metrics have been proposed thus far. DIT, NOC, RFC, WMC, CBO, LCOM, NOM, and DAC are some of these measures. Table 2 contains abbreviations for Li-Henry metrics. By measuring maintenance effort, these metrics—which are mostly utilized in the object-oriented paradigm—predict the maintainability of object-oriented software systems.

Metric	**Definition**
DIT	Depth of Inheritance Tree
NOC	Number of Direct Sub Classes
RFC	Response For Classes
WMC	Weighted Methods Per Class
CBO	Connections Between Object Classes
LCOM	Lack of Cohesion In Methods
NOM	Number of Local Methods
DAC	Data Abstraction Coupling

TABLE 2.2: LI-Henry Metrics

Marcela Genero Model-Marcela Genero [16] created a model in 2003 that included metrics for the size and structural complexity of UML class diagrams with understand ability time, modifiability time, completeness, and correctness for maintainability measures. She outlined the sub-characteristics of maintainability, modifiability, and understand ability in this model. Resolutions for software maintainability predictions can be made using the model's metrics. She employed the multivariate linear model, which is widely used and allows for a link between the counts of dependent and independent variables, to test her hypothesis. The outcomes are just what was predicted.

Aggarwal Model-This paradigm, which is based on artificial neural networks (ANNs), was put forth by K.K. Aggarwal in 2005 [17]. In this approach he uses the Li-Henry metric to predict software maintainability. This ANN-based model computes the CHANGE metric (defined as the number of row changes per class) to estimate software quality. The study by Khoshgaftaar et al. It is shown that the ANN model provides good prediction accuracy results [18], In order to produce accurate forecast results, Aggarwal presented this approach. The backpropagation algorithm was used to train and test this model, and the results of the research indicated that the model can be used to estimate the software's maintenance effort.

Koten-Gray Model-Van Koten and Gray introduced this model in 2006.The Bayesian network-based Koten-Gray model [19] is a maintainability prediction model that can be used to object-oriented software systems. The Li-Henry model metrics data [13], which were gathered from several object-oriented systems, are used to build this model. A unique kind of Bayesian network known as the NAIVE BAYES classifier is built in this model. It consists of a single node that represents a classification variable and is coupled to every other node that represents a predictor variable. He evaluated the suggested model's accuracy using the UIMS and QUES datasets, and the findings indicated that the Bayesian network-based Koten-Gray model was more accurate than models based on regression analysis. The finding demonstrated that the Koten-Grey model, which is based on Bayesian network, is more accurate than models that rely on regression analysis.

Zhou-Leung Model-Yuming Zhou and Hareton Leung [20] used Li-Henry model metrics collected from software systems to create a software maintainability prediction model through regression modeling. They used the UIMS and QUES datasets to evaluate prediction results and compared their findings with those of other prediction models. They adopted the CHANGE metric and calculated the

LOC changes during maintenance in the model to achieve maintainability predictability [21].

Malhotra Model-In 2012 Ruchika Malhotra et al. This concept was introduced. In order to evaluate the maintainability of software, they proposed this method [22]. They use various machine learning methods, such as Group Method for Data Processing (GMDH) and Genetic Algorithms (GA), to build predictive models. They evaluated the performance of the model using UIMS and QUES datasets and concluded that the One of the finest modelling strategies for estimating software maintainability is the GMDH network model.

Dubey Model-Software maintainability can be predicted by using advanced methods such as artificial neural networks. Therefore, Li-Henry model indicators including DIT, LCOM, DAC, and NOM were adopted by Dubey et al. predicting maintainability of object-oriented software using multilayer perceptron (MLP) neural network models. Their proposed model consists of three input layers, two hidden layers, and one output layer. They used the UIMS dataset to test their model and the results showed good accuracy.

3. FUNDAMENTALS OF DATA MINING TECHNIQES

INTRODUCTION

Data mining has developed into an attractive discipline in business, medical science, engineering and other professional fields also in the information technology community. In this chapter, three methods of data mining include introduced. The techniques are Classification technique, Neural Network and Support Vector machine.

3.1 DATA MINING

Data mining is the term used to describe the process to follow for extracting value from a database. According to Moxon "Data mining is the act of finding meaningful new correlations, patterns and trends by sifting through large volumes of data, using pattern recognition technologies in addition to statistical and mathematical techniques. It is a "knowledge discovery the process of obtaining previously undiscovered useful information from large amounts of data databases." alternatively it can be described as "The nontrivial extraction of implicit, previously unknown, and potentially helpful data information‖. Although knowledge discovery in databases (KDD) and data mining are sometimes used interchangeably, data mining is a step in the knowledge discovery process.

3.1.1 DATA MINING TECHNIQUES

Knowledge discovery in databases can be accomplished through a variety of methods and approaches, including decision trees, association rules, neural networks, classification, clustering, regression, artificial intelligence, genetic algorithms, nearest neighbor methods, and neural networks.

3.1.2 CLASSIFICATION TECHNIQUE

Classification is most likely the most frequently used data mining technique. It is the process of finding a set of models that describe and differentiate data classes and concepts, for the purpose of being able to utilize the model to forecast the class whose label is unknown. Classification can be accomplished through a variety of techniques, including decision trees, neural networks, Naive Bayes, SMO, and others. For their evolution, we use four classifiers in our work [28].

The two main categories of data mining techniques are supervised machine learning and unsupervised machine learning. To illustrate how learning methods work, these two learning categories are associated with various machine learning algorithms.

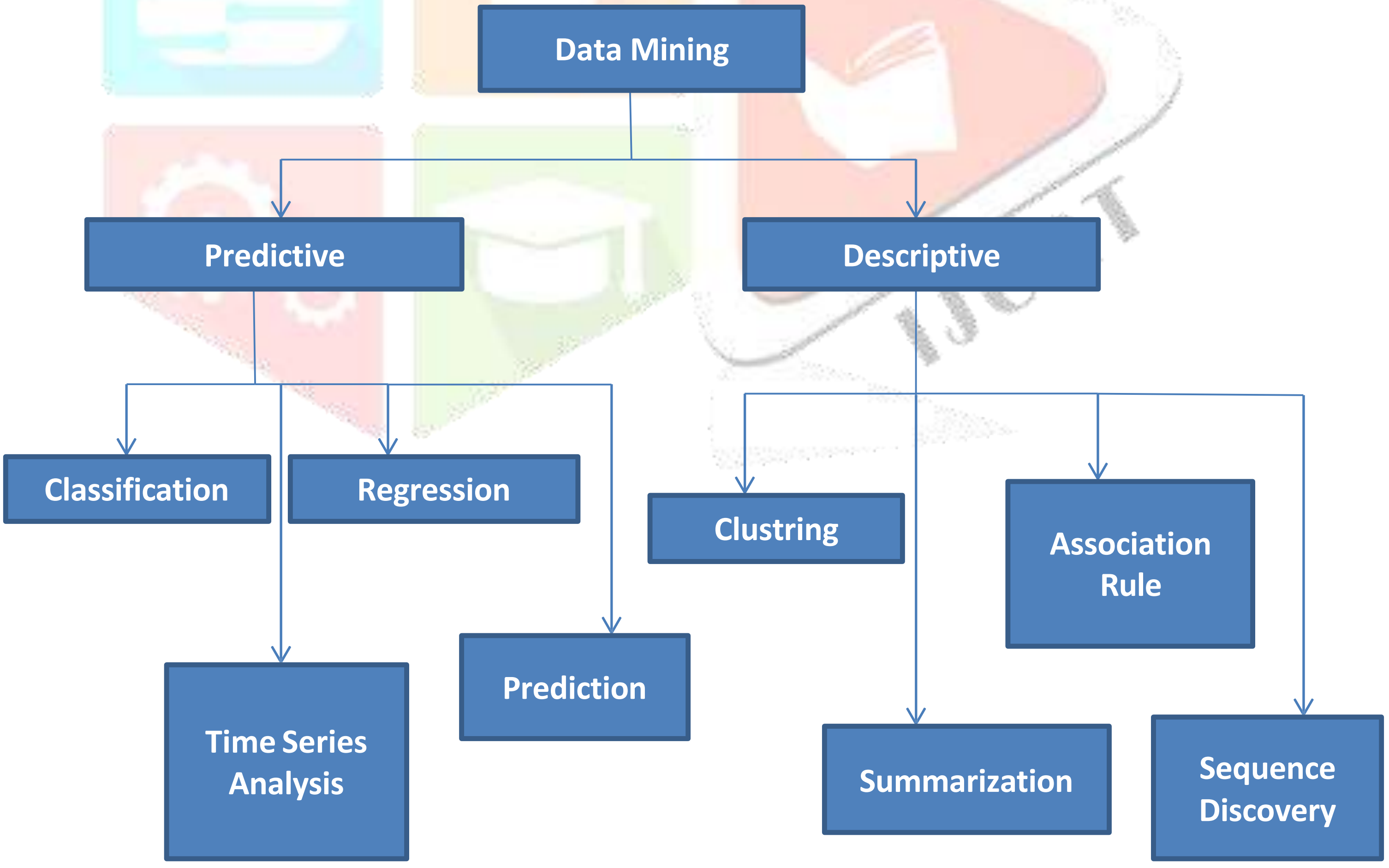


Fig 3.1-Data Mining Model and Task

✓ Predictive Technique

Supervised give to the learning algorithm several instances of input-output pairs, this algorithm learn to predict the correct output that corresponds to some inputs. Example show to learning algorithm patient cases, findings vector and a correct diagnosis for each case) then the algorithm induces a classifier that can classify a previously unseen patient to the correct diagnostic category given the findings observed in that patient.
The algorithms used in supervised learning use examples provided by external sources to generate broad hypotheses that are then used to predict the future examples. In supervised learning, an outcome variable is typically present to direct the learning process to guide the learning process.

✓ Descriptive Technique

Unsupervised learning creates models based on data without predetermined categories or examples, while supervised learning uses the presence of outcome variables to guide the learning process. Therefore, learning must rely on systematic heuristic guidance to analyse various sample data or environments without a supervisor [8]. A specific learning algorithm and its built-in constraints implicitly determine the output state.

3.1.3 DATA MINING ALGORITHMS

J48 Decision Tree

C4.5 is implemented in WEKA as J48. Information entropy is used in C4.5. The WEKA version of the C4.5 decision tree learner is called the J48 algorithm. The algorithm employs error reduction pruning and greedy techniques to generate decision trees for classification. It is the classifier we use to classify classes. It is also known as a free classifier that accepts only nominal categories. To classify instances effectively, prior knowledge is essential. This knowledge is utilized in decision tree construction from labeled training data, leveraging information entropy. The attributes used facilitate decision tree building by subdividing data into subsets, enabling the calculation of gained information through normalization. The splitting process concludes when all instances
within a subset belong to the same class, at which point a leaf node is created to represent that class. However, it's possible that no feature provides information gain, leading to the absence of a clear classification. The J48 algorithm creates decision nodes higher in the tree based on the expected class value, accommodating both discrete and continuous attributes, as well as handling missing attribute values in the training data.

NAIVE BAYES

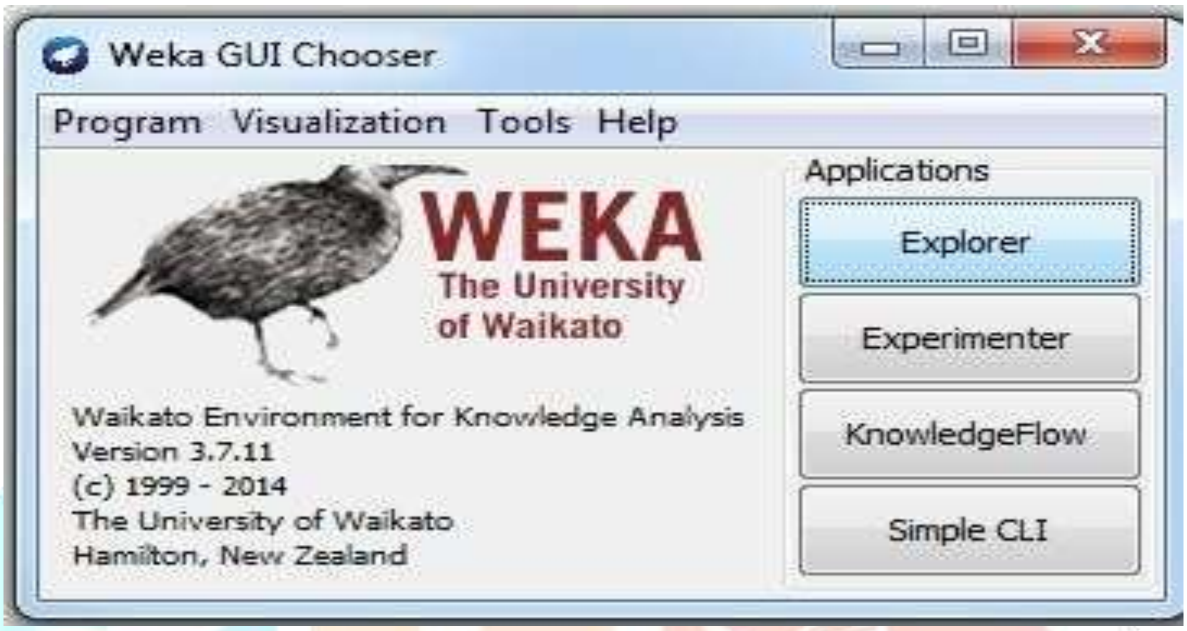
"Naive Bayes is a widely employed classification technique, renowned for its simplicity, elegance, and robustness. The name "Naive Bayes" can be broken down into "Naive" and "Bayes", where "Naive" represents the assumption of independence between events, allowing for the multiplication of probabilities and "Bayes" refers to the application of Bayes' rule. This method presupposes that, in real-world scenarios, the attributes of a class are independent. The performance of Naive Bayes is optimal when dealing with authentic datasets. Moreover, utilizing kernel density estimators to estimate probabilities within Naive Bayes can further enhance the model's performance. While various modifications have been proposed by statistical, data mining, machine learning, and pattern recognition communities to increase its flexibility, it's important to acknowledge that these modifications introduce complexity, which detracts from the technique's fundamental simplicity".

SMO (SEQUENTIAL MINIMAL OPTIMIZATION)

SMO stands for sequential minimal optimization. This algorithm was given was by John Platt. This algorithm is for training a support vector classifier. This algorithm solves the "optimization challenge emerges in the form of a quadratic programming problem." during training of SVM (support vector machine). Quadratic programming problem is defined as binary classification problem with a dataset (X1, Y1) (XN, YN) where Xi is an input vector and Yi {-1, +1} is a binary label corresponding to input vector. SMO solves this equation by breaking it into series of smallest possible sub problems, each of which is then solved analytically. An important feature of this algorithm is its replaces all missing values and nominal attributes are transformed into binary. For Better results normalization is also done.

3.2.4 WEKA TOOL

Meet WEKA, "**a powerful tool for machine learning and data mining"!** Developed by the University of Waikato in New Zealand, WEKA stands for Waikato Environment for Knowledge Learning. As open-source software, it offers a wide range of machine learning algorithms for various data mining tasks. WEKA has made a significant impact in the research community, becoming a go-to toolkit for many. Fun fact: WEKA is also the name of a New Zealand bird! Today, WEKA continues to evolve, providing a modern platform for developing and applying machine learning techniques to real-world problems. Its goal is to offer a comprehensive collection of algorithms and data pre-processing tools for researchers and users alike [7].



Waikato Environment for Knowledge Analysis
Version 3.7.11
(c) 1999 - 2014
The University of Waikato
Hamilton, New Zealand

Figure 3.4: WEKA GUI

WEKA is a powerful toolkit that offers a collection of tools for various data analysis tasks. With WEKA, you can perform data classification, regression, clustering, and generate association rules. Plus, it provides visualization tools to help you better understand your data. Built in Java, WEKA is open-source software, available for free under the GNU General Public License. This means you can use, modify, and share it without any restrictions!" [17]. The WEKA tool consists of the four applications within it.

- The Weka Explorer
- Weka Observation
- Knowledge Flow in Weka
- WEKA CLI

4. ANALYSIS OF SOFTWARE MAINTENANCE

The software systems we work with are intricate and challenging to comprehend, which can lead to errors and flaws. This complexity can result in increased costs and efforts to rectify and maintain the software [18]. The demonstration of the prediction models which classify the classes of Object Oriented System into high, medium and low maintainable classes is analysed in terms of confusion matrix and ROC Curve.

This chapter presents model evaluation techniques such as confusion matrix and ROC Curves. This chapter also presents the performance of each proposed models to predict the maintenance cost of a classes of object oriented system about confusion matrix and ROC Curves.

4.1 METHEDOLOGY

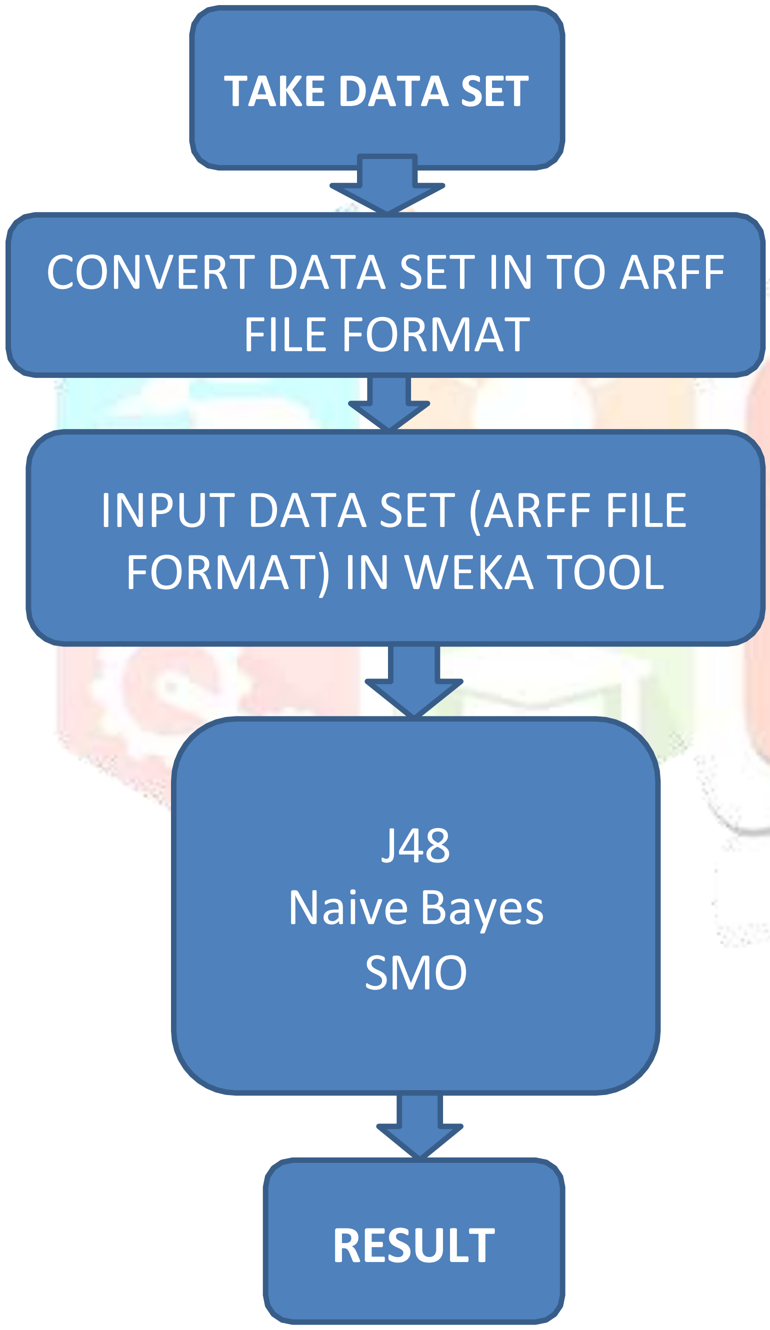

TAKE DATA SET
CONVERT DATA SET IN TO ARFF FILE FORMAT
INPUT DATA SET (ARFF FILE FORMAT) IN WEKA TOOL
J48
Naive Bayes
SMO
RESULT

Figure 4.1: Proposed Flow Chart of Methodology

Step-1:- Take the data set.

Table 5.1: Data Set Use in Analysis

UIMS DATA SET	
Attributes	11
Instances	39
Sum of Weight	39

Step-2:-Convert the data set into ARFF file format @relation UIMS @attribute DIT integer @attribute NOC integer @attribute MPC integer @attribute REF integer @attribute LCOM integer @attribute DAC integer @attribute WMC integer @attribute NOM integer @attribute SIZE2 integer @attribute SIZE1 integer
@attribute change {low, medium, high}

```
@data
% low 2-10, medium 11-49 high 50 and above
% 39 instances
%

1,3,4,14,5,1,3,7,9,37,medium
3,0,6,20,6,2,5,7,9,94,medium
2,0,1,12,4,1,1,6,7,26,low
2,0,1,2,1,0,1,1,1,4,low
1,0,1,21,12,8,0,18,26,60,low
1,8,1,7,3,1,0,4,5,12,medium
```

```
3,1,2,12,6,1,0,6,7,26,medium
2,0,1,10,7,1,9,7,8,59,medium
2,0,1,2,1,0,1,1,1,4,low
3,0,9,24,5,1,5,5,6,81,medium
2,0,1,12,4,1,1,6,7,26,low
3,0,3,17,6,3,3,12,15,74,medium
1,4,10,67,6,21,23,40,61,194,high
3,0,9,30,9,0,16,9,9,143,medium
3,0,4,12,6,0,2,6,6,79,medium
2,0,1,12,4,1,1,6,7,26,low
3,0,3,9,5,0,3,5,5,33,medium
2,0,3,17,9,2,12,9,11,83,high
2,0,3,46,26,2,30,31,33,283,medium
3,0,7,29,7,3,8,13,16,90,medium
4,0,4,10,4,0,0,4,6,61,medium
0,6,1,12,8,3,8,10,13,71,high
3,0,1,12,4,1,1,6,7,34,low
3,0,2,9,4,0,0,4,4,26,medium
2,4,6,29,6,2,9,11,13,86,medium
2,0,1,2,1,0,1,1,1,6,low
2,1,1,15,5,2,3,9,11,44,low
2,0,3,11,5,0,7,5,5,45,medium
3,0,9,25,7,3,8,10,14,131,medium
1,0,11,46,13,5,37,20,26,296,high
2,0,1,11,5,1,1,7,8,27,low
2,6,5,40,13,6,44,20,26,316,high
3,0,6,17,6,0,10,6,6,76,medium
2,0,4,54,6,2,45,32,34,338,medium
0,0,6,32,7,2,10,14,18,84,medium
3,0,9,30,9,0,16,9,9,142,medium
2,3,12,101,31,5,69,39,44,439,high
1,1,10,57,20,13,41,32,45,419,high
3,0,6,17,6,0,10,6,6,76,medium
```

Step-3:-Input data set (ARFF file format) into Weka Tool.

Step-4:- In this step select algorithm J48, Naive Bayes and SMO.

Step-5:-In this step we calculate the result.

5.2 PERFORMANCE MEASURE OF DATA SET

✓ In this experiment we have used a data set. The details of data set are shown in table 4.1.

✓ In this we used the classifiers to calculate Instance Correctly Predicted, Instance Incorrectly Predicted, Total Duration of Model Construction and Error Rate using J48, Naive Bayes and SMO algorithms. Which are shown in table No-4.2

Table 4.2: Performance Measure of UIMS Data Set

Classifier Algorithm	Instance Correctly Predicted	Instance Incorrectly Predicted	Total Duration Taken to build Model(in Seconds)	Error Rate%
J48	34	5	0.1	**12.820**
Naive Bayes	32	7	0.2	**17.948**
SMO	26	13	0.3	**33.333**

Using J48, Naive Bayes, and SMO algorithms, we calculated various performance metrics, including TPR, FPR, Precision, Recall, F-measure, and ROC value, to assess the classification models' effectiveness. These results are displayed in Table 4.3.

Table 4.3: Prediction Performance Measures

Algorithm	Class	TPR	FPR	Precision	Recall	F-Measure	ROC	Accuracy%
J48	Low	**0.9**	**0.103**	**0.75**	**0.9**	**0.818**	**0.888**	**87.179**
	Medium	**0.909**	**0.059**	**0.952**	**0.909**	**0.93**	**0.898**	
	High	**0.714**	**0.714**	**0.833**	**0.714**	**0.769**	**0.761**	
Naive Bayes	Low	**0.8**	**0.034**	**0.889**	**0.8**	**0.842**	**0.969**	**82.051**
	Medium	**0.864**	**0.176**	**0.864**	**0.864**	**0.864**	**0.85**	
	High	**0.714**	**0.094**	**0.625**	**0.667**	**0.667**	**0.882**	
SMO	Low	**0.2**	**0**	**1**	**0.2**	**0.333**	**0.724**	**66.667**
	Medium	**0.909**	**0.647**	**0.645**	**0.645**	**0.755**	**0.631**	
	High	**0.571**	**0.063**	**0.667**	**0.667**	**0.615**	**0.808**	

4.4 RESULTS AND ANALYSIS

In this analysis after experimental work we observed that the classification Accuracy Rate of J48 is higher than other two classifier. Accuracy of Classifier is shown in Table 5.3.

5. CONCLUSION AND FUTURE SCOPE

The most significant expense in software development is maintenance. However, by integrating software metrics into the design phase, developers can effectively manage and reduce maintenance costs and effort. This proactive approach enables the identification of potential issues early on, leading to more efficient and cost-effective software development. The following conclusions highlight the benefits of integrating software metrics into the development process, leading to enhanced software quality, improved resource management, and reduced maintenance costs.

✓ **The accuracy rate of J48, Naive Bayes and SMO are 87.18, 82.05 and 66.67 respectively. It is observed that the accuracy rate of J48 is higher than Naive Bayes and SMO.**

5.1 FUTURE SCOPE

✓ More similar studies on different data set for Object Oriented System using data mining technique is needed to confirm the above finding.
✓ This work may be further used for security aspects of any software system

REFERENCES:-

[1] Kan, S.H., Metrics and models in software quality engineering. 2002: Addison-Wesley Longman Publishing Co., Inc.

[2] Coleman, D., et al., Using metrics to evaluate software system maintainability. Computer, 1994.

[3] Singh, B. and S.P. Kannojia, A Model for Software Product Quality Prediction. 2012.

[4] https://www.tutorialspoint.com/software_engineering/software_maintenance_overview.ht m

[5] Malhotra R., Hug A., Software Maintainability Prediction usingMachine Learning Algorithms, An International Journal (SEIJ), Vol. 2, No. 2, SEPTEMBER 2012.

[6] Malviya, A.K. and Yadav , V.K.‖ Predicting Object-Oriented Software System maintainability at design level using K-means Clustering Techniques I manager‟s journal on Software Engineering , vol 6 no-4 April-June 2012.

[7] Saini, R., S.K. Dubey, and A. Rana, Analytical study of maintainability models for quality evaluation. Indian Journal of Computer Science and Engineering, 2011. 2(3): p. 449-454.

[8] Waghmode, M.M.L. and P.P. Jamsandekar, SOFTWARE QUALITY MODELS: A COMPARATIVE STUDY.

[9] Kitchenham, B. and S.L. Pfleeger, Software quality: The elusive target. IEEE software, 1996. 13(1).

[10] Milicic, D., Software Quality Attributes and Trade-Offs, chapter Software Quality Models and Philosophies. Blekinge Institute of Technology, 2005:

[11] Boehm, B.W., J.R. Brown, and H. Kaspar, Characteristics of software quality. 1978.

[12] Sneed, H.M. and A. Mérey, Automated software quality assurance.IEEE Transactions on Software Engineering, 1985. 11(9): p. 909-916.

[13] Bansal, S. and N. Gupta, Software Component Quality Assessment–ACritical Survey. International Research Journal of Computers and Electronics Engineering, 2013. 1(2).
[14] Li, W. and S. Henry, Object-oriented metrics that predict maintainability. Journal of systems

and software, 1993. 23(2): p. 111-122.

[15] Wake, S. and S. Henry. A model based on software quality factors which predicts maintainability. in Software Maintenance, 1988., Proceedings of the Conference on. 1988. IEEE.

[16] Genero, M., et al. Building UML class diagram maintainability prediction models based on early metrics. in Software Metrics Symposium, 2003. Proceedings. Ninth International. 2003. IEEE.

[17] Aggarwal, K., et al., Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics. Transactions on Engineering, Computing and Technology, 2006. 15: p. 285-289.

[18] Khoshgoftaar, T.M., et al., Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on, 1997. 8(4): p. 902-909.

[19] Van Koten, C. and A. Gray, an application of Bayesian network for predicting object- oriented software maintainability. Information and Software Technology, 2006. 48(1): p. 59- 67.

[20] Zhou, Y. and H. Leung, predicting object-oriented software maintainability using multivariate adaptive regression splines. Journal of systems and software, 2007. 80(8): p. 1349- 1361.

[21] Mens, T., Serebrenik, A., Cleve, A., Evolving Software Systems. 2014: Springer Science & Business Media.

[22] Malhotra, R. and A. Chug, Software Maintainability Prediction using Machine Learning Algorithms. Software Engineering: An International Journal (SEIJ), 2012. 2(2): p. 19-36.

[23] Dubey, S.K., A. Rana, and Y. Dash, Maintainability prediction of object-oriented software system by multilayer perceptron model. ACM SIGSOFT Software Engineering Notes, 2012. 37(5):

[24] http://freefeast.info/difference-between/difference-between-structured-programming-and- object-oriented-programming-structured-programming-vs-object-oriented-programming.

[25] http://www.w3computing.com/systemsanalysis/object-oriented-systems-analysis-design.

[26] Aharwal P.K. , ―Evaluation of various classification Techniques of weka using different data sets‖,journals on IJARIIE-ISSN(0)-2395-4396 Vol-2 issue 2,2016.

[27] Z hou,Y and Leung , H,‖Predicting object-oriented software maintainability using multivariate adaptive regression splines‖, The Journal of systems and software, 8(2007),1349- 1361.

[28] Li w., Henry S.,― Object-Oriented Metrics that predict Maintainability‖, J. System Software, 1993, 23:111-122.

[30] Aleitlager ,I,kuipers ,T and visser ,J ,‖ A practical Model for measuring maintainability – a preliminary report‖, sixth International Conference on the Quality of Information and commercial technology IEEE,2007.

[31] Aggarwal M, Dr. Verma K.V ,Mishra H.V., ―An Analytical Study of Object Oriented Metrics‖ in International Journal of Engineering Trends and Technology (IJETT)-Vol-6 No-2 month-2013.