International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6602
Article Received: 22 February 2023 Revised: 24 March 2023 Accepted: 15 April 2023

An Intelligent Framework for Estimating Software Development Projects using Machine Learning

Prateek Srivastava[1], Nidhi Srivastava[2], Rashi Agarwal[3] and Pawan Singh[4]
[1,2] Amity Institute of Information Technology, Amity University Uttar Pradesh, Lucknow Campus, Lucknow, India
[3]Department of Computer Science and Engineering, Harcourt Butler Technical University, Kanpur, India
[4]Department of Computer Science and Engineering, ASET, Amity University Uttar Pradesh, Lucknow Campus, Lucknow, India
[1] prateeksri976@gmail.com , [2] nsrivastava2@lko.amity.edu ,[3] dr.rashiagrawal@gmail.com, [4] psingh10@lko.amity.edu

Abstract: The IT industry has faced many challenges related to software effort and cost estimation. A cost assessment is conducted after software effort estimation, which benefits customers as well as developers. The purpose of this paper is to discuss various methods for the estimation of software effort and cost in the context of software engineering, such as algorithmic methods, expert judgment methods, analogy-based estimation methods, and machine learning methods, as well as their different aspects. In spite of this, estimation of the effort involved in software development are subject to uncertainty. Several methods have been developed in the literature for improving estimation accuracy, many of which involve the use of machine learning techniques. A machine learning framework is proposed in this paper to address this challenging problem. In addition to being completely independent of algorithmic models and estimation problems, this framework also features a modular architecture. It has high interpretability, learning capability, and robustness to imprecise and uncertain inputs.

Keywords: Software Engineering, Software Project Estimation, Machine Learning, Effort and Cost Estimation.

I. INTRODUCTION

Estimating software projects is a critical and challenging aspect of software development that can be extremely complicated. It is difficult to make an accurate assessment of software development during the early stages of a project. This is due to the many uncertainties associated with inputs such as changes in requirements, platform changes, size of the project, budget constraints, complexity, etc. To meet the competitive demands of today's industry, it is imperative to estimate software effort early in the development process. The procedure of software effort estimation consists of estimating the amount of effort that will be required to finish a particular software project based on the amount of time required. Several studies in the literature have used interchangeably the terms "software effort estimation" and "estimation of software costs". In contrast, the estimation of software costs is a direct result of software effort estimation [1].

There is a growing need for updated, reliable, high-quality software that is easy to use, inexpensive, and delivered in a short period of time. Hence, it is the client's or developer's responsibility to perform a cost-benefit analysis. An analysis of the estimation is converted into dollars. Since the demand for software effort estimation has increased in the industry, it has become a critical task to be performed during the early stages of development. Successful software project management relies heavily on accurate effort estimations [2]. Overestimating and underestimating depend on the allocation of resources as overestimating is the allocation of excessive resources, and underestimating is the allocation of insufficient resources. The ability to predict effort accurately allows risks to be reduced.

Among the branches of AI, machine learning is significant. It has been widely used since 1991 to estimate the development effort of software using Machine Learning. Based on the information that we have gathered from previously completed projects, machine learning allows us to perform estimations [3]. With the implementation of this methodology, experts will be less involved in estimating upcoming projects. It will be more likely that they will spend more time on those aspects of the software system that are likely to satisfy the customer. The accuracy of these predictions has, however, only been studied extensively in the last decade to compare them with various methods (algorithmic models, expert judgment, etc). With the development of machine learning techniques, the use of algorithmic models (non-machine learning techniques) has decreased.

II. RELATED WORKS

Leonardo Villalobos, et al., investigates the impact of random search hyperparameter tuning on SVR accuracy and stability in SEE. The results were compared between the grid search-tuned models and the ridge regression models. Random search is comparable to grid search based on the results of an analysis of four data sets from the ISBSG 2018 repository. Compared to tuning hyperparameters, this is an attractive solution. SVR

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6602
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 15 April 2023*

with RS-tuned hyperparameters achieved a 0.227% increase in standardized accuracy compared to SVR using default hyperparameters. In addition to SVR models, random search also achieved a 0.840 ratio. SVRs tuned using RS and GS were comparable in performance [4].

Passakorn Phannachitta, used Bayesian optimization technique to optimize software effort estimators on 13 standard benchmark datasets, all of which were fully optimized. The performance metrics and statistical tests used for comparison were robust. Adopting a combined effort adapter seems to be an effective strategy for improving analogy-dependent estimation accuracy. A traditional adaptation technique based on productivity adjustment and the Gradient Boosting Machine model performed best in the study when it was integrated with an analogy-oriented model. The model outperformed both state-of-the-art algorithmic-based techniques as well as analogy-based and machine learning-based techniques [5].

P.Suresh Kumar, et al., examine how ANN algorithms such as higher-order neural networks, basic neural networks, and deep learning networks could be applied to estimate software effort. The purpose of this paper was to compare qualitative and quantitative analyses of software effort estimation papers. A survey was also conducted on the following topics: the most widely used datasets for prediction, the most frequently used hybrid algorithms for prediction, and the most commonly used evaluation measures, namely the MMRE, MdMRE, and MRE [6].

Israr ur Rehman, et al., examined five different machine learning techniques in this paper. The study examines how machine learning methods perform when estimating software effort using seven standard data sets. In addition to MMRE and PRED (25), the R2-score is used to rate the effectiveness of the different approaches [7]. Based on all three metrics, decision tree-based techniques produce more accurate results for Desharnais, COCOMO, China, and Kitchenham. A ridge regression method performed better than other methods on both the Albrecht and NASA datasets, with decision trees beating ridge regression for pred (25).

P.Suresh Kumar, et al., proposed a robust method for analyzing regression data based on gradient-boosting regressors. Using COCOMO'81 and CHINA data sets, the performance of the regression models is compared. Various evaluation metrics are applied to evaluate regression models, like MAE, MSE, RMSE, and R2. Gradient boosting regressors performed well based on the results obtained from the two datasets, yielding accuracy of 98% and 93%, respectively. Compared to all regression models used for these datasets, the proposed method performs significantly better [8].

Amir Karimi, et al., presented a novel approach to fuzzy inference utilizing a hybrid approach combining a fuzzy inference system based on applied neural networks (ANFIS) and a methodology known as differential evolution (DE). As part of this investigation and development process, the assessment criteria for testing and comparing the ANFIS-DE algorithm with other well-known algorithms were thoroughly investigated and applied, such as GAs, SBOs, and neuro-adaptive systems. Using MMRE and PRED (0.25) criteria, the proposed method improved accuracy by up to 7% when compared to other algorithms in this study [9].

Robert Marco, et al., determined the hyperparameters of the model using Bayesian optimization, which employs the AdaBoost ensemble learning method and random forest. To build the SEE model, they used the PROMISE repository as well as the ISBSG dataset. An extensive comparison of the developed model was conducted under three-fold cross-validation with four machine-learning algorithms. This comparison shows that the RF approach based on AdaBoost ensemble learning and Bayesian optimization performs better than the previous one. It also assigns a value to feature importance, making it a promising tool for predicting software effort [10].

K Nitalaksheswaro Rao, et al., proposed a novel learning-based model called Optimized Learning-based Cost Estimation (OLCE) which can provide accurate cost predictions for both global projects and large-scale ones. Based on the benchmarked COCOMO NASA 2 dataset, the proposed system optimizes the learning method by integrating artificial neural networks with new search methodologies. As a result of the research, OLCE demonstrated approximately 73% accuracy and 50% faster response time than existing models that are said to be adopted for SCE. It can therefore be concluded that OLCE is a cost-effective and accurate method for SCE [11].

Maedeh Dashti, et al., proposed a novel method for optimizing feature weighting based on the LEM algorithm. This paper presents an approach to optimizing the weights of analogy-based estimation that relies on a learnable evolution model. To investigate this algorithm's effectiveness, two datasets were used in this study, Desharnais and Maxwell. In addition to MMRE and PRED (0.25), MdMRE criteria were used to compare the proposed method with others [12].

III. TECHNIQUES USED FOR ESTIMATION

The practice of estimation is considered one of the most challenging tasks in the software industry. The following sections summarize some methods for estimating software effort. There are three main categories of these approaches. In this section, we will discuss algorithmic approaches, non-

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6602
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 15 April 2023*

algorithmic approaches, and machine learning approaches [13].

3.1 An Algorithmic approach

Cost estimation for software projects is based on mathematical equations. Cost estimates for software projects are determined by project type, size, attributes, procedures, and team members. The use of algorithmic techniques has allowed the development of various models, including the FPBA, Putnam's model, and the COCOMO model [14].

3.1.1 Function Point Based Analysis (FPBA)

By analysing the functions that a software system provides to the user, Function Point Analysis (FPBA) calculates its complexity and size. Tools or languages used to develop the software project are used to define its functionality. As a measure of the size of a system, FPBA overrides some of the major problems associated with Lines of Code (LOC) measurements. Function points are independent of the tools, languages, or procedures that are used during processing; in other words, they do not depend on processor hardware, database management systems, programming languages, or any other technology used during processing [15]. It is also possible to estimate function points based on the specifications of the design or the requirements. This makes it possible to estimate the development effort at the beginning of the project.

3.1.2 Putnam's Model

The model is derived from Rayleigh/Norden's allocation of manpower and analysis of a number of completed projects [13].The software equation defines Putnam's model as follows:

$$S = E * Effort^{1/3} * t_d^{4/3} \quad (1)$$

Software delivery time is t_d, and E is the external factor that reproduces the competence of a developer, which can be taken from historical data using software equations. The effort is measured in person-years and the size of S is in lines of code. Putnam also established an essential relation.

$$Effort = D_0 * t_d^3 \qquad (2)$$

Manpower build-up parameter D_0 is set to 8 for newly developed software and 27 for remodelled software. Preparation and SLIM (Software Living Management) often utilize Putnam's model. Manpower planning and estimation are handled by SLIM using Putnam's model.

3.1.3 COCOMO (Constructive Cost Model)

Barry Boehm introduced the CONSTRUCTIVE COST MODEL. The model estimates project development efforts based on a group of models. Cost and time are estimated in COCOMO based on lines of code and system complexity. COCOMO also considers project attributes, hardware, assessments of the production process, and other factors. The quality of software products is defined by two parameters of the COCOMO model, such as effort calculation and development time. In order to estimate the effort required to complete a task, the number of people involved must be taken into consideration. The unit of measurement is person-month. Development time is the amount of time needed to complete a task. It is expressed in months, weeks, and days. Cost estimations at the different levels of software products can be calculated using COCOMO models of various types [16].

3.2 Non-Algorithmic Approach

This approach is used to estimate initial design experience and design requirements. Non-algorithmic models are able to estimate using previous experience and projects, similar to underestimation projects. The following are some examples of non-algorithmic approaches.

3.2.1 Expert Judgement

Cost estimation for software projects is frequently generated using expert judgment (EJ) techniques. Predicting the cost of upcoming products requires estimations based on a variety of assumptions and judgments [17]. Estimating software project costs is actually done by tapping into groups, characters, or groups of people with expert knowledge using EJ. Expert judgement relies on knowledge, experience, and motivation from experts, as well as on a discussion between analysts and experts within the area of expertise. EJ estimates the cost of software projects based on past experience. The expert utilizes his or her experience from previous projects to assist in the planned project using the EJ method. Methods based on EJ may be used to measure variances between past and upcoming programs. These methods are especially useful in the case of new programs without any previous examples.

3.2.2 Analogy Based Estimation

There are many forms of Case-Based Reasoning (CBR), and one of them is Analogy-based cost estimation (ABE). A case is defined as a partial event in both space and time, a notion of a specific set of events [18]. It is evident that the ABE method of cost estimation for software projects can be applied in most cases. This is because it relies on past information provided by comparable projects, and that comparisons are made by comparing the significant attributes and features of the projects. ABE can be used for estimation by following these steps:

1. Projects are categorized as planned.

2. An historic database is used to choose the exact comparable finalized project which has the same attributes.

It is better to use ABE in the early phases of projects when there is limited information available. It is a simple and easy method that takes less time. Due to the method's reliance on historical project records, the success rates of an organization are likely to be high.

3.2.3 Bottom Up and Top-Down Approach

A top-down or bottom-up approach may be used in EJ software development. An effort estimate could be based on the project's possessions, divided into activities, or estimated as the sum of the estimates for each activity (bottom-up) [19]. Top-down estimation is used for the entire SCE, which is distributed into various subsidiary sections of the project. Cost estimation of software projects is generally faster and easier with this method since it requires the least amount of information. Documentation, integration, configuration, and other system-level activities are considered in this method.

3.3 Machine Learning

In software development, machine learning (ML) techniques are becoming more popular. Mathematical models are a part of artificial intelligence (AI) that identify patterns in data and arrive at conclusions based on the data [20]. By using such algorithms, some information can be derived from the input (training data) that can be used to examine newly generated data (test data). Several techniques for machine learning exist, such as supervised learning, in which input-output mappings for a set of training data have prior knowledge; algorithm learning that occurs without labelled data is unsupervised learning, and reinforcement learning (reward learning approach).

3.3.1 Approaches of Machine Learning

Machine Learning technologies are used in software industry to give an effective prediction or decision support. The various Machine Learning methods are illustrated in figure 1.

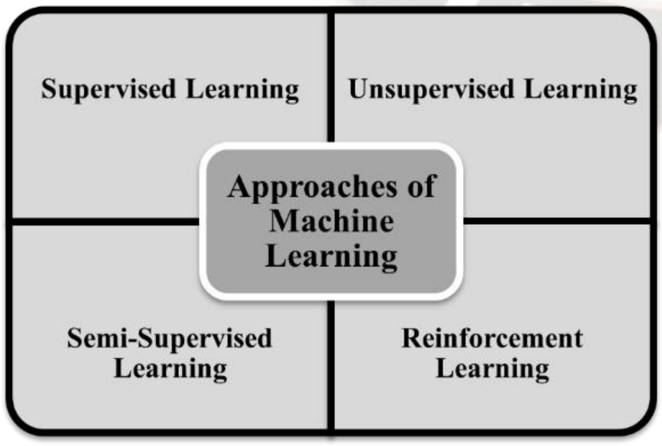

Figure 1 Different methods of Machine Learning

3.3.1.1 Supervised Learning

This type of learning requires the assistance of a supervisor or teacher. The supervisor provides the labeled data necessary to construct the model and generate test results. Learning occurs in two stages when using supervised learning algorithms. Student masters the information presented by the teacher at the beginning of the first stage. Information is received and understood by the student. At this stage, the teacher is unaware of whether a student is able to comprehend the information.

A second phase of learning occurs during this stage. In order to determine how much information has been absorbed by a student, the teacher asks a series of questions. Students are assessed based on these questions, and they are informed about their results by their teachers. Typically, this type of learning is referred to as supervised learning [21].

There are two approaches to supervised learning:

1. Classification

2. Regression

Classification

In the field of artificial intelligence, classification is a method of supervised learning. Classification algorithms use independent variables to determine input attributes. A label or dependent variable represents the target attributes. A classification model describes how input variables are related to target variables through a structure. There are two stages involved in classification learning. In the initial stage of learning, the learning algorithm takes the labelled datasets and begins to learn, and then the model is generated after the samples have been processed. A model constructed in the first stage is tested with a test or unknown sample in the second stage and an appropriate label is assigned to it. This is the process of classification.

Regression

A regression model predicts a continuous variable, such as price, unlike a classification algorithm. To put it simply, it is a numerical value. In a regression model, input x is transformed into a fitted line of the form y=f(x). There is an independent variable Y which controls the outcome of the study while a dependent variable is X.

3.3.1.2 Unsupervised learning

It is also possible to learn by self-instruction, which is one of the two types of learning. There is no supervisory or teaching component, as implied by the name. Learning is most often accomplished by self-instruction when a supervisor or teacher is not available. Using a trial-and-error method of instruction, this process is conducted [22].

There are objects provided for the program, however no labels have been defined. Based on the principles of grouping, the algorithm observes examples and recognizes patterns. Objects are grouped in such a way that they are related to one another. An example of an unsupervised algorithm would be cluster analysis and dimension reduction.

Cluster analysis

Unsupervised learning is illustrated by cluster analysis. By clustering or grouping objects, it tries to group them together. An object is clustered based on its attributes using cluster analysis. Data objects in different partitions vary from each other significantly, while some are similar.

Dimensionality reduction

An example of an unsupervised algorithm is a dimension reduction algorithm. By utilizing the variance of the data, it takes a high-dimensional dataset and outputs it in a lower-dimensional dataset. It consists of reducing the dataset to a small number of features while maintaining generality.

3.3.1.3 Semi-supervised learning

There are instances in which a dataset contains a large amount of unlabelled data and a few labelled data. Humans have difficulty performing labelling because it is very costly and time-consuming. By assigning a pseudo-label to unlabelled data, semi-supervised algorithms generate output using unlabelled data [23]. Once the datasets have been labelled and pseudo-labelled, they can then be combined.

3.3.1.4 Reinforcement learning

A reinforcement learning system mimics the way that humans learn. A reinforcement learning agent interacts with its environment in order to receive rewards [23]. This is in the same way that humans use their ears and eyes to perceive the world and act upon it. There can be many types of agents, including humans, animals, robots, or even non-human programs. Agents gain experience through rewards. Maximizing rewards is the objective of an agent. A reward could be a positive one (reward) or a negative one (punishment). It is easier to learn when the rewards are more appealing.

IV. PROPOSED ESTIMATION FRAMEWORK

It is possible to determine software development effort and cost using the proposed estimation methodology which is presented in figure 2. To estimate the effort and costs involved in developing software, this provides a structured approach to managing and planning the project.

4.1 Identification of Experimental Dataset

It is challenging to begin a research project without accurate, complete, and relevant data from experimental datasets that meet all industry standards. It is crucial to collect and prepare data carefully. We should make sure that all the data files concerned are in the most recent version before documenting a dataset.

Users commonly encounter the following problems with their data:

1. Records in the dataset cannot be uniquely identified by variables.
2. Observations that are duplicated.
3. A merge of multiple datasets may result in errors.
4. Comparing the contents of the survey questionnaire with the content of the data files produced incomplete results.
5. Data that is not labelled.
6. Missing values in variables.
7. A data file that contains unneeded or temporary variables.
8. Direct identifiers or sensitive data.

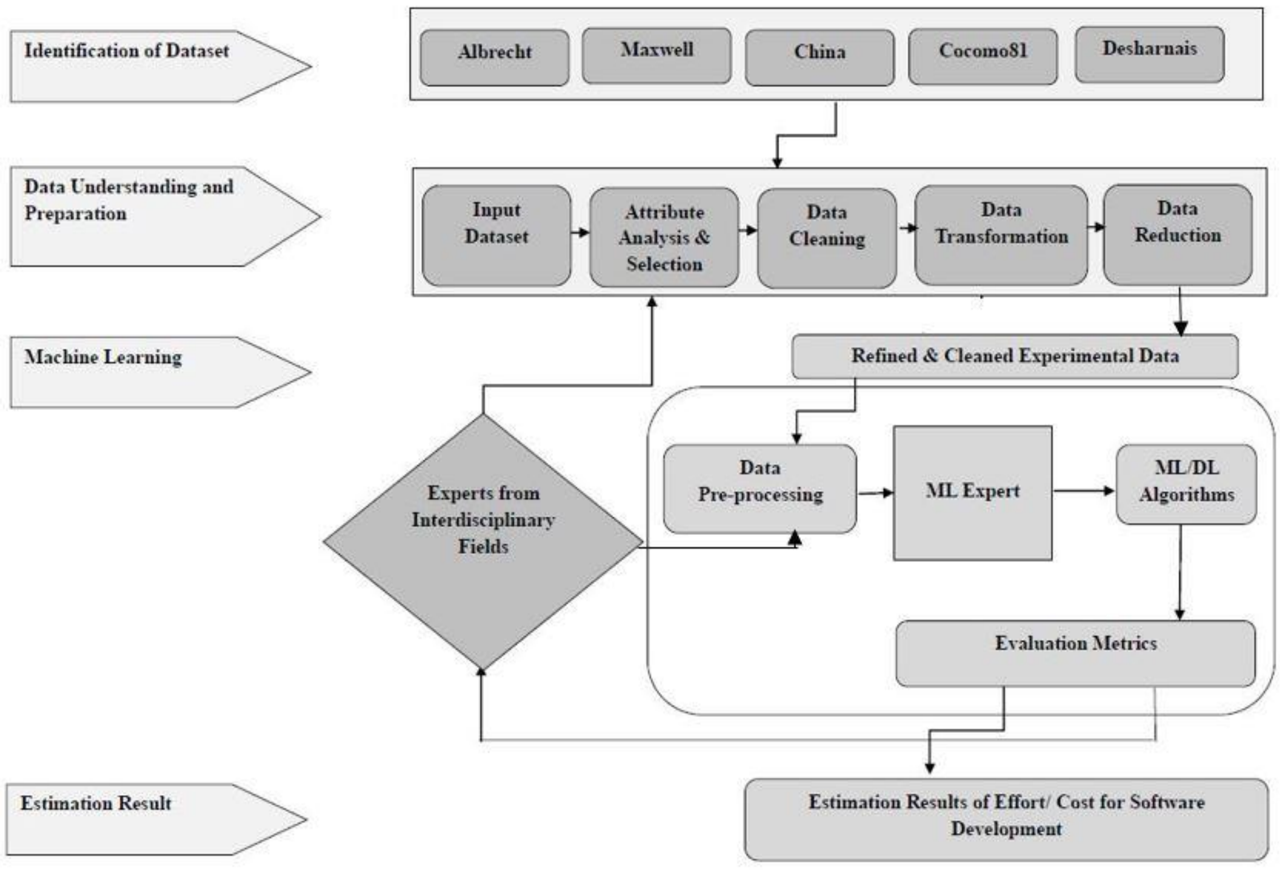
Identification of Dataset
Albrecht
Maxwell
China
Cocomo81
Desharnais
Data Understanding and Preparation
Input Dataset
Attribute Analysis & Selection
Data Cleaning
Data Transformation
Data Reduction
Machine Learning
Refined & Cleaned Experimental Data
Experts from Interdisciplinary Fields
Data Pre-processing
ML Expert
ML/DL Algorithms
Evaluation Metrics
Estimation Result
Estimation Results of Effort/ Cost for Software Development

Figure 2 Proposed Estimation Framework

Experiments should use high-quality datasets that have well-defined details, which are appropriate for software estimation. Machine learning deep learning algorithms are trained using software estimation experimental datasets. To obtain relevant and appropriate data, it is necessary to identify them from publicly available repositories [24]. Below is a list of some software estimation datasets.

Table 1 Experimental Datasets for Estimation in Software development

Datasets	Number of Attributes	Records
Albrecht	8	24
China	19	499
Cocomo81	19	63
Desharnais	12	81
Maxwell	27	62
Nasa93	24	93
ISBSG16	264	7518
Kitchenham	10	145
Kemerer	8	15
Miyazaki94	9	48
Finnish	9	38

We must choose one experimental dataset from Table 1 to proceed further.

4.2 Data Understanding and Preparation

The process of interpreting data involves reviewing data and applying different analytical techniques to draw relevant conclusions from it. To answer pertinent questions, data interpretation is used to categorize, manipulate, and summarize information. A variety of data sources may be used to gather data, and these data tend to arrive in haphazard order at the beginning of the estimation process. Interpreting data properly is of paramount importance, so it must be performed correctly. Cleansing, transforming, and reducing data are the three main steps of data preparation.

4.2.1 Selection and Cleaning

A crucial part of machine learning is data cleaning. It has a significant role to play in the creation of a model. Clean data is crucial to the success of a project. In order to make their work more efficient, data scientists tend to spend a great deal of time on this step since they believe "Good data is more useful than fancy algorithms". Clean datasets often allow us to achieve accurate results with simple algorithms as well, which can be very useful if we need to compute very large datasets.

Our model will perform better if we carefully choose the input we need and delete duplicate and redundant columns. Using appropriate data types can conserve memory by transforming numerical data into integers.Data from experiments determine how incomplete information is resolved. It may be necessary to investigate imputation, which replaces absent values with placeholders or other values based on assumptions.

4.2.2 Transformation of data

Transforming data means converting it from one scale to another. Our model must be implemented so that a significant increase in accuracy can be achieved. Transformation is therefore avoided when datasets behave as if they have already been refined. Machine learning models cannot handle data without completing certain data transformation stages. There are several steps in this process: removing string formatting, carriage returns, gaps at the beginning and end of entries, monetary symbols, and more. A phrase becomes less understandable for people if textual and other letters are deleted. However, an algorithm can digest the data more efficiently.

Transformation of data begins with identifying the data types, sources, and structures; determining how the transformations should be structured; and defining how fields will be aggregated or changed. The process involves extracting and transforming data from its original source.

4.2.3 Data Reduction

In data reduction, the original volume of the data is reduced, and it is represented as a much smaller amount than before. The integrity of data is ensured while the data is reduced through data reduction techniques. The goal of data reduction is to make it more compact. The application of sophisticated and computationally expensive algorithms is easier when the data size is smaller. Data reduction can be carried out by reducing the number of rows (records) or columns (dimensions).

To reduce data, there are several strategies available, including the following:

● Aggregation of data cubes.

● Selecting attributes from a subset.

● Choosing from a variety of options, etc.

4.3 Machine Learning

Once the experimental data has been refined and cleaned to meet the desired standards, the next step is to estimate the effort and cost associated with this evaluation. It is possible to reduce software development risk by using machine learning/deep learning in estimation [25]. This can improve software development project planning, improve project efficiency, and increase project success.

4.3.1 Data Pre-processing

During the pre-processing stage, raw input data is corrected, refined, and converted into a format that is useful for the learning process. Alternatively, the process of enhancing raw data by adjusting, cleaning, and converting it into enhanced refined forms is known as pre-processing. Machine learning / deep learning models may be adversely affected by unreliable data. It is possible that inaccurate and inconsistent results may result from poor data quality, especially if there are missing records or outliers.

Data preparation involves removing unwanted information, such as duplicate observations, observations with irrelevant information, and observations without information. We transform refined data if the set of records is biased or skewed after removing unwanted information.

There are numerous approaches and techniques available for cleaning data and preparing experimental datasets, but only those that meet our requirements can be chosen for implementation.

4.3.1.1 Machine Learning Expert

To select the appropriate ML/DL algorithm as part of machine learning methodologies, experts from interdisciplinary fields are required to provide human intervention in the process. These experts are called ML experts. There is the participation of machine learning experts in the methodologies of machine learning. Their expertise in designing ML systems as professionals provides them with a unique perspective on their potential and limitations. Data are reviewed and analysed by experts in the field of machine learning in order to determine the most effective frameworks, correlations, and characteristics. Due to the lack of experience and capabilities associated with the ML system's solution, the involvement of ML professionals is essential. The use of DL techniques does not require human intervention.

4.3.2 Machine Learning Deep Learning Techniques

As humans learn from experience, machines learn from data through machine learning algorithms. Data patterns are recognized directly by machine learning algorithms, without relying on a predetermined equation for modelling. In contrast, deep learning is used to process unlabelled or unstructured data. Therefore, it is capable of automatically detecting differences between different categories of data.

This phase is crucial to the estimation process as it incorporates ML/DL techniques. Machine learning estimates software using a variety of approaches and technologies.

Statistical inference, appraisal, and forecasting of experimental results are possibilities enabled by Deep Learning/Machine Learning techniques. Software estimation using Machine Learning and Deep Learning has been developed using several algorithms. There is a difference between Deep Learning and Machine Learning in terms of their capabilities. These algorithms perform the following actions in the following pattern:

(1) Dataset division

(2) Feature extraction and selection

(3) Selection of appropriate ML/DL algorithm.

4.3.2.1 Dataset Division

Data can be used to train machine learning models, which can then be used to generate predictions based on new data. Divide the dataset according to the following steps:

Step 1: We select a dataset for experimentation.

Step 2: Training and testing datasets are now divided, such as 80% or 75% for training and 20% or 25% for testing. As a rule, these are the standard ratios.

Step 3: There are two partitioned datasets: a training one and a testing one. We may use the training dataset to perform the next stage of feature extraction and selection. A testing dataset enables the machine learning model to be evaluated in the future after it has been trained.

Step 4: Data pre-processing is an essential and inevitable part of training dataset preparation.

4.3.2.2 Feature Extraction and Selection

As part of feature extraction, attributes are retrieved from the data. The selection of attributes is accomplished by converting relevant attributes into sets and groups, as well as determining which attributes have meaningful relationships between entities. It is possible to generate customized attributes by combining raw/provided attributes.

This process involves developing novel features from the existing ones in a dataset and discarding the original features to reduce the number of features in the data. Using regularization techniques instead can result in a variety of additional benefits, including increased accuracy during model training and a reduction in overfitting risk.

4.3.2.3 Selection of appropriate ML/DL Algorithm

A rule-based approach for selecting appropriate algorithms was developed by Machine Learning experts to analyse estimates in software development. We use machine learning techniques in this selection phase to enable prediction. However, when we require both prediction and transfer learning in the future, we can use other techniques such as deep learning. Large or complex data sets might require deep learning approaches.

In comparison to existing individual machine learning or deep learning models, ensemble techniques provide more accurate results. An ensemble approach is a method of providing better results by combining multiple models (also known as base learners). The prediction performance is improved by training different types of learning machines and combining them.

K-fold cross-validation is also possible if the available experimental attributes in the data are all required, but they are still overfitting the model. In ML/DL models, we must split data samples into groups in the training phase, so we select the K number of best-performing algorithms. By applying cross-validation to an experimental data set, we can determine accuracy and positive predictive rates.

4.3.3 Evaluation Criteria

At this stage, we will measure model performance with MAE, MMRE, RMSE, PRED and R2. Performance evaluation metrics are listed below.

4.3.3.1 Mean Absolute Error (MAE)

An effective way of measuring evaluation criteria is through MAE. The equation below indicates the average of absolute errors between actual and projected efforts [13].

$$MAE = \frac{1}{TP}\sum_{i=1}^{TP} |AE_i - PE_i| \tag{3}$$

For each test data, AEi is the original value found in the dataset. Using the created model, PEi is the resulting output. The test set has TP records.

4.3.3.2 Mean Magnitude Relative Error (MMRE)

Software prediction models are mostly compared using MMRE. We select the most appropriate model based on MMRE [13].

$$\mathrm{MMRE} = \frac{\sum_{i=1}^{N} \frac{|estimated-actual|}{actual}}{N}. \tag{4}$$

Where the test set size is equal to N.

4.3.3.3 Mean Squared Error (MSE)

MSE measures how close the regression line is to the data points by taking the projected value into account. By doing this, we avoid the unfairness inherent in residual square sums. Our estimation errors are calculated using the MSE equation, which uses both the model's actual output and its prediction

[13]. An error variation will be smaller if the MSE value is smaller.MSE can be determined mathematically using the following equation.

$$MSE = \frac{1}{n}\sum_{i-1}^{N}(y_i - \hat{y}_i)^2 \tag{5}$$

Where the test set size is equal to n.

4.3.3.4 Prediction (PRED (x))

The predictive performance of a regression model should be measured to determine if it is superior to its competitors.

(1) Use different performance indicators to compare models.

(2) Evaluate the results of the measures outlined above.Below is an equation that illustrates how to determine PRED(x) mathematically [13].

$$PRED(x) \equiv \sum_{i=1}^{N} [MRE_i \leq x])|N > 0 \tag{6}$$

There are N projects, where 'MRE' represents the percentage of projects with MREs less than or equal to x. It is possible that x could be 0.25, 0.50, 0.75, or 1.0. In a scenario where x is 0.50, PREDs (0.50) are projects with MREs under 50%.

4.3.3.5 Squared coefficient of correlation (R^2)

We will examine the performance of the model at this stage by calculating the Squared Correlation Coefficient (R2). The equation below shows the performance evaluation metrics [26].

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Actual\ Effort - Estimated\ Effort)^2}{\sum_{i=1}^{N}(Actual\ effort - Mean(Actual\ Effort))^2} \tag{7}$$

For each estimated point, the first summation is per appropriate point divided by N.

4.3.4 Experts from Interdisciplinary Fields

Interdisciplinary experts pass on the results obtained from the performance metrics to the machine learning steps. Results are assessed by experts in various fields. The ML model is tuned to predict effort and costs more accurately based on the evaluation of the findings.

4.4 Estimation Result

Studies done with experimental data are summarized in the results section, as well as how these studies supersede previous studies. Estimated results are matched with actual entities at each step of the estimation process, while proficiency is matched using evaluation criteria. As a further step, any additional improvements or threats to the study may be added for better prediction of software effort and cost estimation.

V. CONCLUSION

Project managers need to estimate efforts in order to allocate resources effectively and manage time effectively during the development process. Initially, estimation was performed using both algorithmic and non-algorithmic approaches. This study explores the possibility of predicting software project effort and costs through machine learning and deep learning approaches. When provided with accurate, large, and well-labelled data, ML models perform well, however, an inefficient dataset reduces their performance. ML model performance can be improved by using this scenario as the foundation. Data division, data pre-processing, and the algorithms used in the ML model also shape the model's performance. In ML models, the variance of performance is heavily influenced by the steps and phases. It may not be possible for the model to give a 100 percent accurate prediction; however, it can provide some useful estimations.

REFERENCES

[1] Jadhav, A. G., Kaur, M., & Akter, F. (2022). Evolution of Software Development Effort and Cost Estimation Techniques: Five Decades Study Using Automated Text Mining Approach. *Mathematical Problems in Engineering, 2022,* 1–17. https://doi.org/10.1155/2022/5782587

[2] Nassif, Ali Bou, et al. (2022). On The Value of Project Productivity for Early Effort Estimation. Science of Computer Programming, vol. 219, Elsevier BV. p. 102819. https://doi.org/10.1016/j.scico.2022.102819

[3] Hameed, S., Elsheikh, Y., & Azzeh, M. (2022). An optimized case-based software project effort estimation using genetic algorithm. Information & Software Technology, 153, 107088. https://doi.org/10.1016/j.infsof.2022.107088

[4] Villalobos-Arias, L., Quesada-López, C., Guevara-Coto, J., Martinez, A., & Jenkins, M. (2020). Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation. Predictive Models in Software Engineering. https://doi.org/10.1145/3416508. 3417121

[5] Phannachitta, P. (2020). On an optimal analogy-based software effort estimation. Information & Software Technology, 125, 106330. https://doi.org/10.1016/j.infsof. 2020.106330

[6] Kumar, P. S., Behera, H. S., K, A. K., Nayak, J., & Naik, B. (2020). Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. Computer Science Review, 38, 100288. https://doi.org/10.1016/j.cosrev.2020.100288

[7] Ali, Z., Rehman, I. U., & Jaan, Z. (2021). An Empirical Analysis on Software Development Efforts Estimation in Machine Learning Perspective. Distributed Computing and Artificial Intelligence, 10(3), 227–240. https://doi.org/ 10.14201/adcaij2021103227240

[8] Kumar, P. S., Behera, H. S., Nayak, J., & Naik, B. (2021). A pragmatic ensemble learning approach for effective software effort estimation. Innovations in Systems and Software Engineering, 18(2), 283–299. https://doi.org/ 10.1007/s11334-020-00379-y

[9] Karimi, A., & Gandomani, T. J. (2021). Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm. International Journal of Power Electronics and Drive Systems, 11(1), 707. https://doi.org/10.11591/ijece.v11i1. pp707-715

[10] Marco, R., Ahmad, S., & Ahmad, S. (2022). Bayesian Hyperparameter Optimization and Ensemble Learning for Machine Learning Models on Software Effort Estimation. International Journal of Advanced Computer Science and Applications, 13(3). https://doi.org/10.14569/ijacsa.2022. 0130351

[11] Rao KN, Bolla JV, Mummana S, et al. (2022). OLCE: Optimized Learning-based Cost Estimation for Global Software Projects. Research Square. DOI: 10.21203/rs.3.rs-2024296/v1.

[12] Dashti, M., Gandomani, T. J., Adeh, D. H., Zulzalil, H., & Sultan, A. B. (2022). LEMABE: a novel framework to improve analogy-based software cost estimation using learnable evolution model. PeerJ, 7, e800. https://doi.org/10.7717/peerj-cs.800

[13] Srivastava, P., Srivastava, N., Agarwal, R., & Singh, P. K. (2022). A Systematic Literature Review on Software Development Estimation Techniques. Advances in Intelligent Systems and Computing, 119–134. https://doi.org/10.1007/ 978-981-16-4641-6_11

[14] Sharma, A., & Chaudhary, N. (2023). Prediction of Software Effort by Using Non-Linear Power Regression for Heterogeneous Projects Based on Use case Points and Lines of code. Procedia Computer Science, 218, 1601–1611. https://doi.org/10.1016/j.procs.2023.01.138

[15] Khan, B. (2020). Software Cost Estimation: Algorithmic and Non-Algorithmic Approaches. http://ijdsaa.com/ index.php/welcome/article/view/73

[16] Software Cost Estimation – A Comparative Study of COCOMO-II and Bailey-Basili Models. (2020). IEEE Conference Publication | IEEE Xplore. https://ieeexplore. ieee.org/document/919416

[17] Srivastava, P., Srivastava, N., Agarwal, R., & Singh, P. K. (2022b). Estimation in Agile Software Development Using Artificial Intelligence. Lecture Notes in Networks and Systems, 83–93. https://doi.org/10.1007/978-981-16-8826-3_8

[18] Sylla, A., Coudert, T., & Geneste, L. (2021). A Case-Based Reasoning (CBR) approach for Engineer-To-Order systems performance evaluation. IFAC-PapersOnLine, 54(1), 717–722. https://doi.org/10.1016/j.ifacol.2021.08.182

[19] Rak, K., Car, Ž., & Lovrek, I. (2019). Effort estimation model for software development projects based on use case reuse. Journal of Software, 31(2), e2119. https://doi.org /10.1002/smr.2119

[20] Ali, A., & Gravino, C. (2019b). A systematic literature review of software effort prediction using machine learning methods. Journal of Software, 31(10). https://doi.org/ 10.1002/smr.2211

[21] Predicting Software Effort Estimation Using Machine Learning Techniques. (2018). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document /8486222

[22] Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis. (2022). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9803030

[23] Sinha, R. R. S. R. R., & Gora, R. K. (2021). Software Effort Estimation Using Machine Learning Techniques. Lecture Notes in Networks and Systems, 65–79. https://doi.org/10.1007/978-981-15-5421-6_8

[24] Mustafa, E. I., & Osman, R. (2020). SEERA: a software cost estimation dataset for constrained environments. Predictive Models in Software Engineering. https://doi.org/ 10.1145/3416508.3417119

[25] Varshini, A. G. P., Kumari, K. M., D, J., & Soundariya, S. (2021). Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation. Journal of Physics, 1767(1), 012019. https://doi.org/ 10.1088/1742-6596/1767/1/012019

[26] G, P. V. A., K, A. K., & Vijayakumar, V. (2021). Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach. Electronics, 10(10), 1195. https://doi.org/10.3390/electronics10101195