International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I5P130

Volume 73 Issue 5, 369-382, May 2025

Original Article

Using Deep Learning Model to Estimate Cost of Software Project Development

Ajay Jaiswal[1], Jagdish Raikwal[2], Pushpa Raikwal[3]

[1]*Department of Computer Science & Engineering, Prestige Institute of Engineering Management & Research, Indore, India.*
[2]*Department of Information Technology, Institute of Engineering & Technology, Devi Ahilaya Vishwavidhalay, Indore, India.*
[3]*Department of Electronics and Communication Engineering, PDPM-IIITDM, Jabalpur, India.*

[1]*Corresponding Author : ajay.jaiswal55555@gmail.com*

Received: 27 October 2024 Revised: 12 May 2025 Accepted: 26 May 2025 Published: 31 May 2025

Abstract- *Effective software-related cost estimation is paramount in decision-making. Estimating is the macro activity that is part of project methodology and allows for the effective delivery of projects. This is useful in project management because it assists with implementing the necessary tasks. Pretty much the discussed parameter helps in the optimization of resources in relation to the requirements for accomplishing project scope. There are several important factors that encompass software projects, including time, resources, human resources, infrastructure and materials, finance, and risk. In case the cost estimate is lower than required, the time for the development of the project will be longer and more expensive. The scope for waste of resources has been exaggerated. Artificial intelligence is a fusion of machine learning and deep learning to produce smart systems capable of posing solutions to problems. Software effort estimation assists in constructing the objectives, which include planning, scheduling, and budgeting for a project. Different prediction trials mentioned above, which were expert opinion-based, analogy-based estimates, regression estimations, categorization strategies, and deep learning algorithms, were suggested as predictors of type of endeavors. Among the evaluation metrics discussed were Mean Absolute Error, Root Mean Squared Error, Mean Square Error, and R-squared. Therefore, estimation has and will take a significant role in risk prevention measures in the future. Metrics for assessment will be used in many assessments. After this, other studies intend to explain the reasons why software developer cost modeling can be very beneficial in light of LSTM (Long Short-Term Model) and CNN (Convolutional Neural Network) prospects introduced throughout the research. This method allows for solving intricate tasks with multiple dependencies in an ever-changing environment by using ML (Machine Learning) and DL (Deep Learning) technologies. Further studies reveal that the most common deep learning architecture in these studies was convolutional, and relatively little application was deep learning.*

Keywords - *Machine Learning, Deep Learning, LSTM model, CNN, Software cost estimation.*

1. Introduction

Building a fair and robust software product requires many activities, from gathering and analyzing requirements to final testing and maintenance, all of which must be performed within time and cost constraints [1]. However, as the projects evolve, the people who are involved in the managing processes are now facing greater challenges in developing software. Estimations of time and effort are also considered critical planning processes for all commercial organizations. Such measures are important for many, including the customers of a company who participate indirectly in the development process [2]. Estimation of software development effort has elicited the use of various techniques. And in the current circumstances, measure evaluation is of paramount importance. Studies carried out by the Standish Group revealed how only 32% of all software projects are delivered with the right features, on time, and within budget. 24.4% failed, meaning they were canceled or finished but never utilized, and 44.4% did not fulfill the previously stated requirements [3] [4]. Planning the software and its project is one of the most important aspects of the entire process of the development of software. It involves a collection of tasks and activities that can be generally grouped as the creation of a plan for the project, its implementation, estimating risks that may arise, and generating alternative solutions for these issues. Software cost and effort estimation is part of the project planning process, which includes estimating project expenses as well as the number of man-hours and time required to execute the project. An erroneous estimate can lead to project failure and greater expenditures. Inaccurate software project estimates are caused by a number of factors, including stakeholder governance, market pressures, project budgeting, project risk management, required development effort (capacity, estimation, and availability), project objective

formulation, and project scheduling errors [5]. In the decision-making process, an accurate estimate is crucial. Overestimating project effort can lead to a project proposal being rejected, whereas underestimating might cause work to be unfinished owing to financial constraints and scheduling conflicts [6]. These models rely on prior completed projects that are comparable to ongoing software endeavors to estimate. Estimates are derived from dataset studies or data from earlier program releases. Non-algorithmic estimation techniques include analogy estimation, expert judgment techniques (such as top-down and bottom-up estimation), learning-based approaches (Artificial Neural Networks) [7–10], Machine Learning (ML) [11], and Deep Neural Networks (DNN)), and Long Short-Term Memory (LSTM). There are a number of restrictions, including the need for large, inclusive, and unbiased training data sets. Training and learning also require a significant amount of time to attain a high level of accuracy and relevance. Neural networks are, therefore, difficult to use for high-dimensional, multi-objective data classification. Accuracy, data requirements, interpretability, scalability, and adaptability are important factors to consider when comparing different models, including both conventional and deep learning-based approaches for software project development cost estimation. An organized comparison of the most popular models can be found below (Table 1):

Table 1. Comparing different models

Model Type	Accuracy	Data Req.	Scalability	Adapt-ability
Expert Judgment	Low–Medium	Low	Low	Low
COCOMO/ Algorithmic	Medium	Medium	Medium	Low
Regression Models	Medium	Medium	High	Low
Random Forests	Medium–High	_	Medium	Medium
SVM	Medium	Low–Medium	Low	Medium
Feedforward Neural Nets	High	High	High	High
LSTM/RNN	High	High	Medium	High
Transformers	Very High	Very High	High	Very High

Even with the availability of numerous estimation models and techniques, accurately estimating software development cost and effort is still an ongoing task. Missed deadlines, budget overruns, and project failure can result from estimation errors. Analogy estimation, expert judgment, machine learning, deep neural networks, and LSTM are examples of traditional and learning-based approaches that still face challenges like reliance on sizable, objective datasets, lengthy training periods, and trouble managing high-dimensional, multi-objective data. Reliable, real-time decision-making in project planning suffers from these issues.

Although there are a number of estimation models, such as machine learning and traditional approaches, there aren't many reliable, scalable, and accurate models that can handle high-dimensional, multi-objective data with sparse or unbalanced datasets. Furthermore, a large number of existing models have limited practical utility because they are unable to adjust to changing project dynamics or generalize well across a variety of software projects. Better deep learning models are required in order to decrease reliance on large data sets, speed up training, and produce more accurate and easier-to-understand cost and effort estimates.

Over time, the application of deep learning techniques to software project cost prediction has undergone significant change, reflecting both the growing complexity of software systems and more general trends in artificial intelligence deployment. At first, the primary methods for estimating project costs were expert opinion, historical comparisons, and computer models such as COCOMO. Although these models were interpretable, they were not flexible enough to account for the intricacy of modern construction techniques. As processing power and information became more accessible, machine learning techniques began to improve on traditional methods. Deep learning's recent success can be attributed to its ability to model intricate, nonlinear relationships between a range of project characteristics, such as team preparation, technology framework, code parameters, development method, and project cost outcomes.

2. Background

Deep learning has proven to be significantly more successful than conventional estimating methods when working with complex, high-dimensional data. Traditional approaches, like linear regression or other machine learning algorithms, frequently rely on assumptions about the distributions and relationships of data, which may limit their accuracy and flexibility. These methods are typically needed for a large amount of human feature engineering, but they are less successful at detecting nonlinear patterns. On the other hand, because deep learning models, such as neural networks, automatically extract intricate features and patterns from raw data, they can perform better on tasks like time-series forecasting. Additionally, models trained on one task can be successfully transferred to another with little extra effort thanks to transfer learning and deep learning, which scale well with large databases.

Estimation shows how much cost, time, effort, and resources are needed to develop a system or product. Although there are several approaches for estimating effort, it is still

regarded as one of the most interesting tasks. However, ML techniques are not without their limits. First, repeated items made in medium to large numbers are well suited for machine learning (ML) based techniques (Hammann, 2024). In many cases, business software tools include databases containing extensive volumes of financial and industrial data. [1] Researchers will give some background on deep learning techniques in this part. After giving a brief overview of deep learning, researchers discuss its differences from machine learning. Researchers outline the situations in which DL is necessary. Deep learning is a branch of machine learning that draws inspiration from the way the human brain processes information.

Why Should you Endeavor to Study Deeply?

This question may be addressed by a number of performance components, including the Universal Learning Approach. DL is sometimes referred to as universal learning because of its performance in practically all application fields.

1. Robustness: Generally, deep learning algorithms do not require well-defined features. Rather, the best qualities are automatically acquired in a way that is pertinent to the current work. Hence, resilience to common alterations.
2. Generalization: The same Deep Learning (DL) technique, also known as Transfer Learning (TL), as explained in the next section, can be used for different kinds of data or applications (Figure 1).
3. Modeling: The purpose of the case study is to create two cost models; hence, the following actions need to be taken twice. The findings for the semi-finished and final goods are shown together for ease of viewing.

2.1. Deep Learning (DL)

DL is also based on artificial neural networks, which in turn give different explanations of the input data [12]. In standard machine learning, the steps required to deal with the classification problem involve step-by-step operations: preprocessing, feature extraction, feature selection, training, and classification. Moreover, machine learning algorithm performance is strongly influenced by feature selection. Deep learning has the ability to automate feature set learning for a wide range of applications, in contrast to conventional ML techniques [13] [14]. Data classification and learning may happen at the same time, thanks to deep learning (Figure 3). When to Use Deep Learning Machine intelligence is useful in many situations and can be on par with or superior to human professionals in a number of them [15].

Neural network-based Deep Learning (DL) models excel at recognizing complex trends and non-linear correlations in data. Because they can learn features and are highly expressive, they don't need a lot of manual feature engineering. Despite their advantages, they have drawbacks, such as high processing requirements that require substantial training resources.

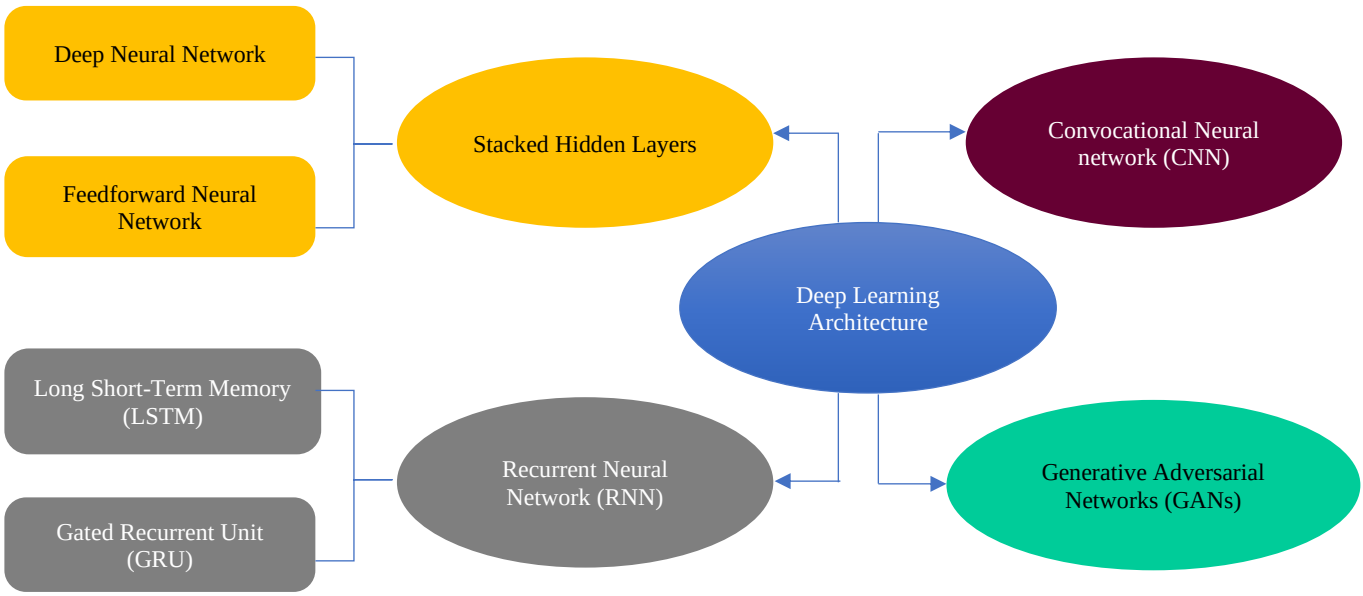

Fig. 1 Classification of deep learning methods [17]

2.1.1. Decision Trees (DTs)

DT is a popular interpretive tool that does a good job of showing how decisions are made. They don't assume any specific data distributions and can handle non-linear connections by nature. However, DTs are prone to overfitting and are unstable with slight changes, which leads to the capture of noise in the training set. Individual DTs may not be expressive enough to recognize more complex patterns in the data, even though they perform well for simple tasks.

2.1.2 Random Forest (RF)

These models decrease the overfitting issues related to individual trees and provide more consistent forecasts. They are great at handling non-linear interactions and offer insightful analysis of the relevance of traits. Though they are resource-intensive as their computational complexity increases with the number of trees, though they are less of a "black box" than DL models, Random Forests can still be hard to read compared to more basic models. Which deep learning model is best for categorization will depend on the specific aim and the data. Still, among the most often used deep learning models for categorization are CNNs, RNNs, and LSTMs.

2.2. Deep Learning Approach(s) Classification

The three theorems of deep learning that are known today consist of Semi-supervised, supervised, and unsupervised. Moreover, RL, or deep reinforcement learning, is a type of supervised methodology but is often regarded as footnote supervised (See Figure 1).

2.2.1. Supervised Deep Learning

With this approach, there exists labeled data. If researchers consider this method, then there are many possible inputs and many possible outputs in the universe. If an agent trainee well, then he may utilize the environment to determine the suitable answers to questions. Some of the supervised learning algorithms of deep learning include recurrent neural networks, Convolutional neural networks, and deep neural networks. RNNs also consist of Long short-term memory and Gated Recurrent Units (GRUs) techniques. The main advantage of this approach lies in the ability to collect or output data based on previous information. It's a drawback that if the training dataset doesn't contain plenty of samples for the classes of interest, the decision boundary may become overly stressed. In broad terms, this kind of quite powerful method of learning is easier than the rest.

2.2.2. Deep Semi-Supervised Learning

The basis for mastering this method is semi-labeled datasets. Furthermore, RNNs are employed in partially supervised learning, including GRUs and LSTMs. Reducing the quantity of labeled data required is one benefit of this method. Semi-supervised learning is the best method for classifying text documents since it overcomes the challenge of obtaining a large number of tagged text documents.

2.2.3. Unsupervised Learning

This approach enables learning to proceed even when labeled input is not available. Common unsupervised learning techniques include generative networks, clustering, and reduction of dimensionality. These methods have demonstrated strong performance on non-linear dimensionality reduction and clustering issues. Moreover, a wide range of applications have employed RNNs for unsupervised learning, including RU and LSTM techniques.

2.2.4. Reinforcement Learning

This led to the development of several improved reinforcement learning systems. On the basis of this idea, several supervised and unsupervised approaches have been created. This learning is significantly more difficult than standard supervised processes since the reinforcement learning method does not have a fundamental loss function. Moreover, supervised and reinforcement learning vary in two important ways [18].

2.3. Deep Learning Network Types

The two most popular types of deep learning networks, CNN and LSTM, are covered in this section. CNN was covered in detail since it was so important. Moreover, it is the most extensively utilized in several networks and a wide range of applications. Deep learning is increasingly regarded by the scientific community as a viable technique to enhance cost estimates. Deep learning performs better than traditional shallow learning methods because its features are more carefully chosen and accurately represented. DL is a more effective method for assessing software costs because it enables the depiction of intricate relationships between effort and cost components [19].

2.3.1. Convolutional Neural Networks (CNNs)

In deep learning, the CNN algorithm is the most popular and extensively utilized. The primary benefit of CNN over its forerunners is its ability to autonomously recognize important traits without the need for human assistance. Numerous applications, such as computer vision, audio processing, and facial recognition, have made extensive use of CNNs. Similar to conventional neural networks, CNNs were inspired by the neurons seen in both animal and human brains. CNN mimics the complex cell sequence that makes up the visual cortex of a cat's brain, to be more precise.

CNN Architecture

There are several levels (sometimes referred to as multiple building components) in the CNN architecture. The levels of the CNN architecture are described in detail below, along with their respective roles.

Convolutional Layer

The output feature map is created by convolving these filters with the input image, which is represented as N-dimensional metrics. The CNN input format is covered first in the context of convolutional operation. The input of a CNN is an image with several channels, whereas the input of a normal neural network is a vector format.

Consistency with CNN

When it comes to CNN models, overfitting is the biggest obstacle to well-behaved generalization. As will be covered in the next section, a model is deemed over-fitted if it functions exceptionally well on training data but poorly on test data (unseen data). The opposite is achieved by an under-fitted

model, which absorbs too little information from the training set. If a model shows good performance on both training and testing data, it is said to be "just fitted". In Figure 2, these three groups are shown. Regularization is assisted by a number of intuitive ideas to help prevent overfitting; further details on overfitting and under-fitting are included in later sections.

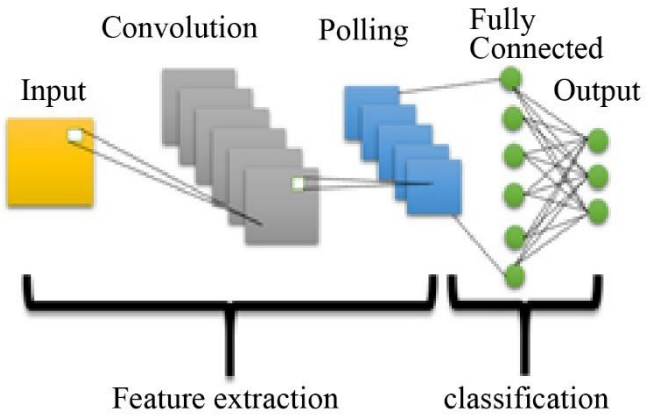

Fig. 2 Convolutional Neural Networks (CNNs)

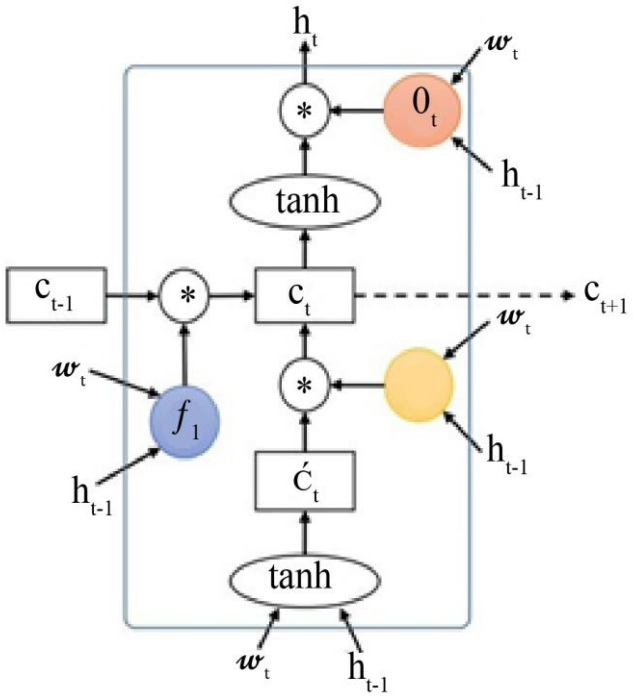

Fig. 3 Architecture of LSTM model

2.3.2. LSTM (Long Short-Term Model)

LSTM networks (Figure 3) are designed with sequence prediction problems in mind. According to Brownlee (2017), there are several Long Short Term Model designs, such as CNN-LSTM, stacked Long Short-Term Model, bidirectional Long Short-Term Model, encoder-decoder Long Short-Term Model, vanilla LSTM, and generative LSTM. The statelessness, lack of temporal structure, clumsy scaling, fixed-sized inputs, and fixed-sized outputs are some of the drawbacks of multi-layer Perceptron (MLP) feedforward artificial neural network (ANN) methods (Brownlee) [20]. LSTM may be seen as an addition to the network, in contrast to the MLP network. Furthermore, LSTM makes up RNN techniques. Unlike MLP networks, LSTMs can mimic parallel input sequences, process variable-length input to generate variable-length output, keep an internal state, and be aware of the time structure of their inputs. According to Brownlee (2017), the memory cell is the computational unit of the Long Short-Term Model. These cells consist of gates and weights (both internal state and output weights).

In addition to the fundamental advantages of deep learning models, the effectiveness of deep learning approaches for cost estimation in software project development work is also influenced by realistic factors like team experience and the chosen technology stack. Although conventional estimation methods are usually ruled by guidelines and mostly rely on expert judgment or historical trends, deep learning models can learn from vast data sets to reveal intricate, nonlinear interactions between project elements and costs. However, a team with strong machine learning, data science, and software engineering skills is required for the successful implementation of deep learning. The benefits of the approach may be compromised if inexperienced teams fail to choose a model, preprocess data, adjust it, and interpret the results. The technology framework, comprising frameworks such as PyTorch, Tensor Flow, and Keras, also influences scalability, integration with existing systems, and development efficiency. Meeting the computational demands of deep learning also calls for advanced technology and resources such as data workflow automation, GPU support, and cloud computing systems. Deep learning can outperform traditional estimation methods by producing more exact, data-driven forecasts that fit changing project dynamics and organizational settings when properly used by an experienced team with the appropriate tools.

3. Related Work

One interesting trend is moving toward data-driven estimation, which places an emphasis on obtaining and choosing large, high-quality datasets from previous initiatives in order to train deep learning models. Another trend that enables organizations to apply knowledge from external datasets or domains to their own environment, even in situations where they have limited internal data, is the use of model training and transfer learning. Generally speaking, hybrid estimation methods that mix domain knowledge, conventional models, and deep learning are increasingly popular as they help to preserve accuracy, scalability, and understanding. As data practices and tooling continue to advance, deep learning is expected to play an increasingly significant role in providing more accurate, flexible, and data-informed cost estimation for software development projects.

A G Varshini, Priya, et al. (2019) [21], Software Effort Estimation aids in project planning, scheduling, and budgeting]. Numerous studies, including expert judgment, regression estimations, categorization strategies, analogy-based estimations, and deep learning algorithms, were put out to forecast effort. Akhbardeh et al. (2021) [22] examined the techniques for calculable factors that affect software cost and presented studies that used machine learning techniques to develop a trustworthy estimation technique. Machine learning (ML) was used by Govinda et al. (2022) [23] to calculate the costs of project management software using standard input.

Alauthman et al. (2023) [24] discussed the selection of regression models for software development cost estimates. It placed emphasis on matching models to the dataset used for estimation and the software development technique. A collection of assessments based on simple classifiers and stacked ensemble classifiers with and without the feature selection technique was published by Mustafa Hammad (2023) [25]. A dataset from the software projects of 76 university students is used in the evaluation research. For any assessment criterion, the feature selection strategy can improve the performance of every stacked classifier. Selvam, Karthick Panner, et al., (2024) [26].

Victor Uc-Cetina (2023), Recent Advances in Software Effort Estimation using Machine Learning, addresses both agile and non-agile approaches to software development effort estimation using machine learning. It evaluates recent and organized advancements in data-driven prediction models for software effort estimation and talks about the benefits of applying agile methodologies in this field. To increase the accuracy of software cost estimation, Fizza Mansoor et al. (2024), Improving the Estimation of Software Costs Investigation, combine machine learning algorithms with a variety of feature selection techniques. The study highlights how Principal Component Analysis (PCA)-based feature selection greatly improves model performance using the COCOMO NASA dataset, highlighting the significance of optimal feature selection in cost estimates.

The model estimates memory usage with a MAPE of 4.92% and predicts training step length with a MAPE of 9.51%. Various machine learning and deep learning methods were employed to estimate the amount of work. Datasets for estimating effort were gathered from Promise repositories, GitHub, and ISBSG projects. PRED (25), the proportion of projections with an MRE of less than or equal to 25%, was the most often utilized indicator for estimations.

Key Research Findings are:

1. *Conventional Models:* In general, conventional models are superior to deep learning. Several studies have demonstrated that deep learning methods, specifically Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs), perform better than traditional estimating models such as Function Points and COCOMO. This is especially true for large datasets with complex feature relationships.
2. *Temporal and Sequential Data:* It has been shown that Recurrent Neural Networks (RNNs), and specifically LSTM networks, are particularly effective at cost estimation when dealing with time-series or sequential data, such as the cost trends of software projects over time.
3. *Quantity and Quality of Data:* Large, high-quality datasets are necessary for deep learning models to function effectively. Numerous studies show that insufficient data can have a detrimental effect on deep learning models' performance, underscoring the importance of feature selection and data preprocessing.
4. *Hybrid Models:* An emerging trend in the field is combining deep learning with other machine learning techniques (like ensemble methods). It has been shown that by reducing bias and variance, this method improves predicted accuracy.
5. *Interpretability Issues*: One of the main problems with using deep learning techniques for software cost estimation is the inability to read data. This is a serious issue, especially when stakeholders must comprehend the results.

These studies provide valuable insights into the evolving application of deep learning in this field and show that, despite its challenges, deep learning still has the potential to revolutionize software project cost estimation and management. The application of deep learning techniques for software cost estimation is a novel and fascinating field with significant potential to improve prediction accuracy and reliability when compared to more conventional approaches. However, there are problems with data quality, model interpretability, and the need for big databases.

4. Performance Measures

Machine learning is a technique that teaches computer systems to become better versions of themselves by using historical data. Data-driven predictions are made when ML algorithms build a prediction model (Figure 4) using a collection of previously accessible training data [27]. Various techniques are employed by numerous scholars and professionals globally to enhance software estimates [28]. Numerous methods for assessing the precision of prediction models have been put forth in the software estimating literature to date. PRED, MMRE, correlation, and other methods can be used to evaluate the performance of a model that generates continuous output. PRED is a metric derived from relative error, or RE, which is the relative magnitude of difference between estimated and actual value. One approach to conceptualize these metrics is to state that performance measures and additional new variables, N+1, N+2, etc., are included in the training data, which consists of records with variables 1, 2, 3, and so on.

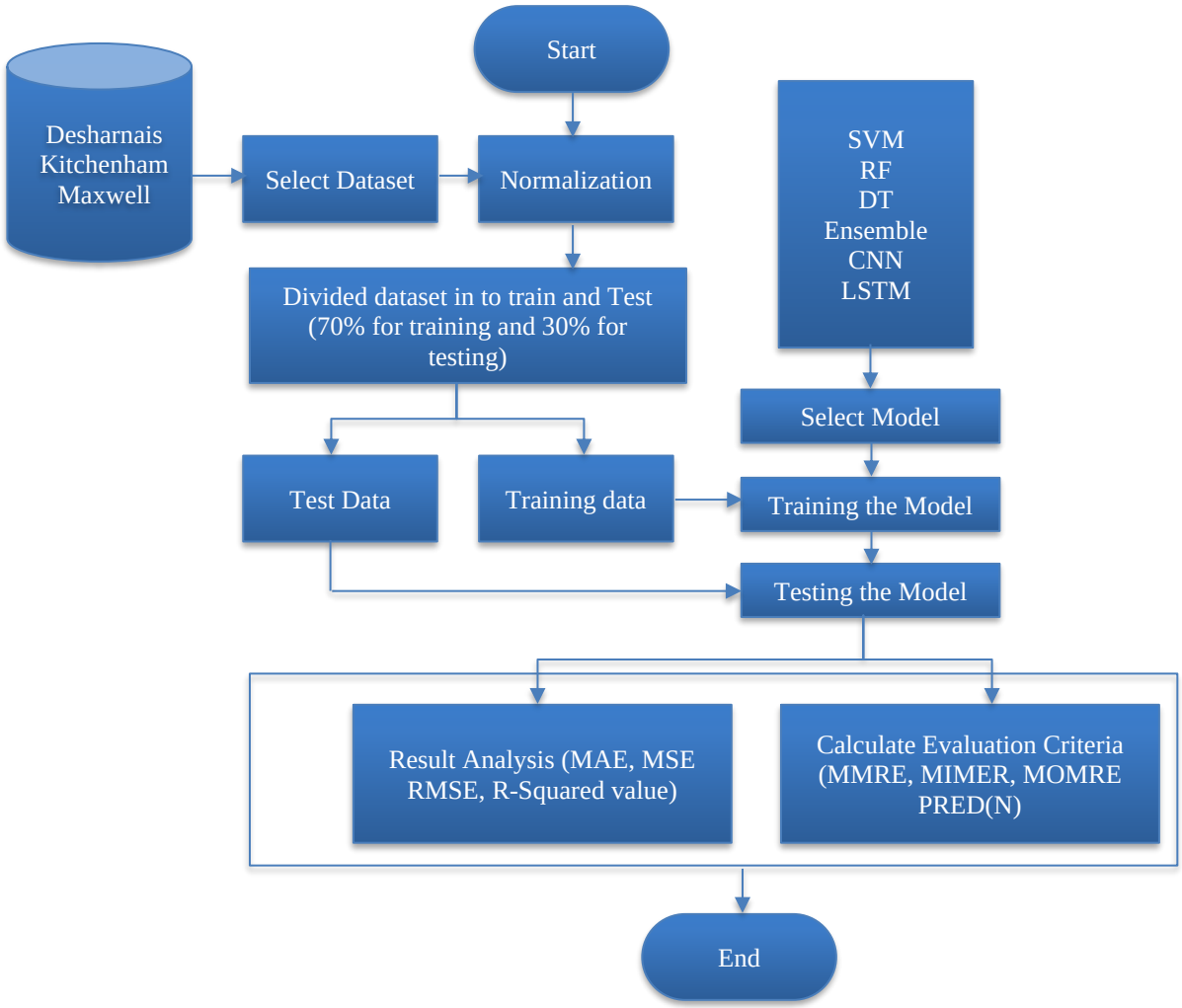

Fig. 4 Architecture of proposed tool

When using deep learning methods for cost estimation of software project development, several statistical methods are integrated into the modeling process to ensure the reliability, generalization, and robustness of the estimates. Performance Metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 (Coefficient of Determination), these metrics provide quantitative evidence of model accuracy and help compare deep learning performance with traditional estimation methods. RMSE is often preferred when large estimation errors (e.g., budget overruns) are especially critical. While deep learning introduces powerful nonlinear modeling capabilities, these statistical methods are essential to ensure the models are reproducible, interpretable, and statistically valid for real-world cost estimation tasks in software development.

4.1. MRE (Magnitude of Relative Error)

To assess the amount of estimating error in a single estimate, first ascertain the Magnitude of Relative Error for each data point. This step is used to calculate PRED (n) and acts as a model for the one that follows. A score of 25 percent or below denotes good results.

$$\text{MRE} = |\text{predicted} - \text{actual}| \quad (1)$$

4.2. MMRE (Mean Magnitude of Relative Error)

Currently, the most efficient and accepted metrics for estimating exactness are used in MMRE (Mean Magnitude of Relative Error), also known as mean absolute relative error. These measures include PRED at power levels 0.25, 0.50, and 0.75, respectively, and MMRE. Researchers employ a standard metric known as Mean Magnitude of Relative Error (MMRE) [29] to evaluate capabilities.

$$\text{MMRE} = \left(\frac{100}{\text{N}}\right) * \sum|\text{predicted}i - \text{actual}i|/(\text{actual}i) \quad (2)$$

4.3. Mean Squared Error (MSE)

The average squared difference between the expected and actual values in a dataset is determined using a metric called mean squared error, or MSE. The squared residuals—the

discrepancies between the expected and actual values for every data point are averaged to determine it. You can evaluate the model's accuracy using the MSE number.

$$\text{MSE} = \sum\{(\text{predicted}_i - \text{actual}_i)2\}/N \quad (3)$$

4.4. Root Mean Squared Error (RMSE)

Among other error metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), Mean Squared Error is commonly used to assess model performance. RMSE computes the square root of the average squared difference, whereas MAE measures the average absolute difference between expected and actual values. Because they penalize large errors more harshly than the MAE, the MSE and RMSE are more susceptible to outliers.

$$\text{RMSE} = \sqrt{(\text{MSE})} \quad (4)$$

4.5. R-squared Error (R^2)

The percentage of the dependent variable's variation that can be predicted by the independent variables is shown by this statistical metric. To calculate the R-squared (R^2) value from the RMSE, you need the total variance (SST) of the observed values. First, Calculate RSS (Residual Sum of Squares), then calculate SST (Total Sum of Squares):

$$\text{R Square} = 1 - (\text{RSS}/\text{SST}) \quad (5)$$

4.6. PRED (n) Accuracy of Prediction

In addition, a model must be accurate to within 25% of cases, or 75% of the time [33]. To find the accuracy rate PRED(n) (represented by n), divide the total number of data points in a data set that has an MRE of 0.25 or less (represented by k) by the total number of data points in the data set. The derived equation is PRED (n) = k/n, where n = 0.25 [30]. PRED(n) frequently displays the average percentage of guesses that were within n percent of the actual values. PRED = 50%, for example, suggests that half of the estimates are within 30% of the real if there are N datasets.

MREi<=n/100,then

$$\text{PRED}(x) = \left(\frac{100}{N}\right) * \sum i \ldots N1, \text{ else } 0\sim \quad (6)$$

The anticipated estimated value is closer to the actual estimate, the lower the MMRE value, and vice versa. Reviewing previous research on software cost assessment is being done [31]. Over the past ten years, a number of meta-heuristic approaches for software cost estimates have been put into practice. Using deep learning techniques for software project development cost estimation may result in significant prediction errors due to a number of important factors. Despite their great accuracy and flexibility, deep learning models are not resistant to large and consistent changes in project costs. Understanding the reasons for and features of these errors is necessary to improve model reliability and decision-making. Significant prediction errors in deep learning-based cost estimation are caused by model design, data limitations, and real-world project uncertainty. Beyond simply improving model performance, identifying, assessing, and correcting these errors is essential.

The proposed deep learning model achieved superior accuracy in software cost estimation compared to state-of-the-art methods, such as traditional machine learning algorithms and deep learning architectures. Research on the optimization of software cost estimation has computed the efficacy of metaheuristic algorithms. The author carried out an exploratory longitudinal case study in [32] [33]. Semi-structured interviews and archival research were used to gather data. The accuracy of effort estimation is increased by the two-stage estimation procedure, which re-estimates the analysis step.

In software evaluation, undervaluation is the predominant trend, and work overspending is more common in less experienced teams. Researchers have identified several limitations in the literature, and our experiment has shown that the majority of researchers overlook the preprocessing processes. In addition to these restrictions, attribute selection is a significant restriction that directly impacts memory utilization and outcomes. Thus, in order to get beyond these restrictions, we do these simple actions (Figure 5). These methods collectively ensure that deep learning models used for cost estimation in software projects are not only accurate but also trustworthy, transparent, and repeatable, which is essential in real-world decision-making scenarios.

5. Experiments and Results

This section presents the outcomes of the study carried out in an organized way for cost estimation of a software project.

5.1. Datasets Description

5.1.1. Desharnais Dataset

Since their existence would have affected the accuracy of the findings, researchers decided to leave these projects out of the estimation process. 77 software projects were finished after the data preprocessing stage.

5.1.2. Maxwell Dataset

One of the biggest commercial banks in Finland provided the Maxwell dataset, which consists of 62 projects with 23 attributes; a piece gives the Maxwell dataset a thorough explanation.

5.1.3. Kitchenham Dataset

The Kitchenham dataset, compiled by Kitchenham and her colleagues, encompasses data from multiple sources, including proprietary and public domain projects.

Experiment Design (Figure 5)

- Apply Feature Selection Method
- Ensemble Learning Methods
- Computing the results

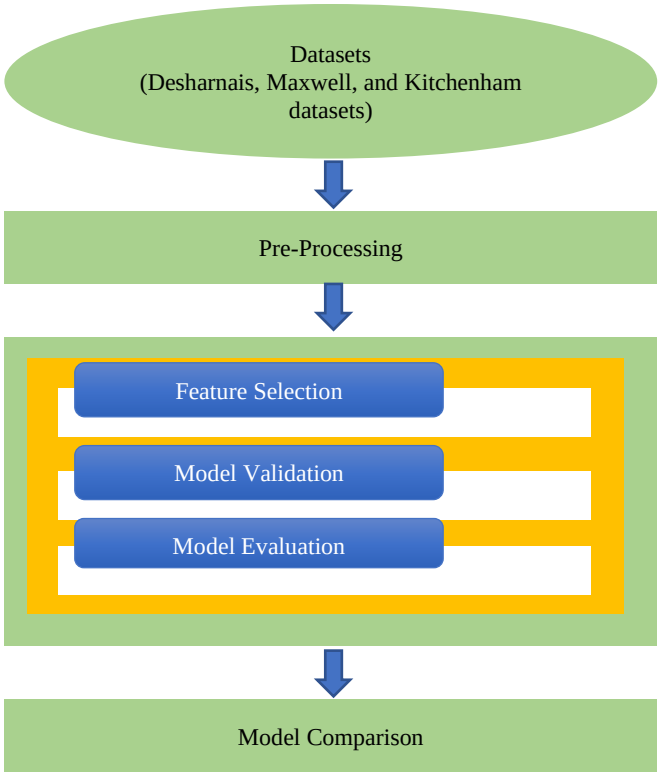

Fig. 5 Experiment setup for software cost estimation

5.2. Extracting Features for Estimating Software Costs

When creating models to estimate software project costs, feature extraction is a critical step in the process. The Word2Vec approach is used in this work to extract pertinent features from Desharnais, Kitchenham, and Maxwell datasets. After they have been retrieved, these characteristics are added to the cost calculation procedure. Table 2 below presents the characteristics that were extracted from the three datasets and gives a detailed summary of the attributes that may be taken into account during the cost-estimating process. By using these characteristics as input variables, prediction models that can more precisely and accurately estimate the costs of software projects may be developed.

Table 2. List of features extracted using Word2Vec

Datasets	Extracted Features
Desharnais	Project, Points NonAdjust, ManagerExp, Adjustment, YearEnd, Length Transactions, PointsAjust, Effort, TeamExp,
Maxwell	Effort, Har, Year, Duration, App, T14, Source, Nlan, T06, T05, T15, T09, Size, Time
Kitchenham	Adjfp, Estimate method, Client code, Estimate, Projecttype, Duration, Effort

5.3. Selecting Features for Estimating Software Costs

It is critical to select features for a model if one is to build reliable and reasonable software cost-estimating models. In order to obtain the most relevant features while mitigating the risk of overfitting the predictive models, the study observed the RFE methodology to glean the most salient features from three datasets. RFE is quite popular in regression modeling for regularization and feature selection. It works by successively eliminating elements that, according to a pre-defined rule, are deemed unnecessary or redundant. These repeated steps lead to a shorter and clearer explanation of the model, which finalizes with the strongest predictors that remain in the model. After the RFE method, features that were gathered from three datasets are shown in Table 3.

Table 3. List of features using RFE

Dataset	Selected Features
Desharnais	Project, Transactions, Effort, TeamExp, PointsAjust, PointsNonAdjust
Maxwell	App, Effort, Har, Source, Nlan, T05, T09, T15, Year, Duration, Time, Size
Kitchenham	Effort, Project type, Client code, Duration, Estimate, Adjfp

Table 2. Each dataset is represented by a row in the table, while the columns indicate the specific features assigned to each dataset. These features were chosen because they have the potential to greatly contribute to accurate software cost prediction. By adding these selected features into predictive models, the software cost-estimating process can benefit from a more robust and informative collection of input variables, resulting in higher cost estimation accuracy.

5.4. Performance Evaluation Outcomes on Various ML Models

This section discusses the performance metrics of various ML models on all three datasets. It describes different kinds of errors calculated for each model.

5.4.1. On the Desharnais Dataset

Different ML models, including LR, DT, SVM, and Ensemble, were evaluated on the Desharnais dataset using various performance metrics, as shown in Table 4.

Table 4. Error metrics obtained on the desharnais dataset

Error Metrics	MAE	R2	RMSE
LR	0.263	0.778	0.353
DT	0.432	0.532	0.372
SVM	0.331	0.804	0.331
Ensemble	0.336	0.798	0.241

Figure 6, given below, illustrates a comparison of different ML and DL models based on error metrics across the Desharnais dataset. It further emphasizes the performance variations among the models, highlighting their strengths and weaknesses in predicting software project costs.

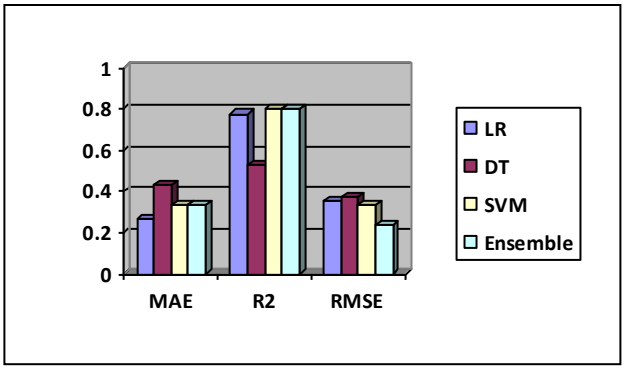

Fig. 6 Performance metrics comparison on the desharnais

5.4.2. On Maxwell Dataset

Various ML models were evaluated on the Maxwell dataset using various performance metrics, as shown in Table 5. Error metrics for various ML models applied to the Maxwell dataset are summarized in Table 5.

Table 5. Error metrics obtained on the maxwell dataset

Error Metrics	MAE	R2	RMSE
LR	0.201	0.929	0.275
DT	0.326	0.761	0.406
SVM	0.202	0.929	0.274
Ensemble	0.194	0.930	0.270

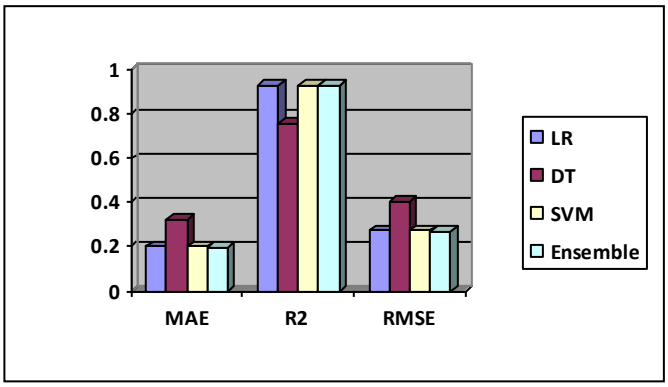

Fig. 7 Performance metrics comparison on the maxwell

5.4.3. On Kitchenham Dataset

Various ML models were evaluated on the Kitchenham dataset using various performance metrics, as shown in Table 6.

Table 6. Error metrics obtained on the kitchenham dataset

Error Metrics	MAE	R^2	RMSE
LR	0.483	0.607	0.588
DT	0.929	0.017	0.706
SVM	0.698	0.447	0.537
Ensemble	0.658	0.507	0.241

Figure 7 depicts the error metrics of several ML models and a comparative analysis of the performance of each model, shedding light on their effectiveness in software cost estimation on the Maxwell Dataset. It examines relevant features obtained from extraction and selection and results obtained by evaluating ML models for estimation. Machine learning and Deep learning algorithms considered for estimation are FFN, RNN, LSTM, CNN, and SVM. Table 6 displays the performance measures of the machine and deep learning algorithms of different datasets. Figure 8 displays a graphical representation of the Mean Absolute Error, Mean Square Error, Root Mean Squared Error, and R-squared of the kitchenham dataset [34] [35] [36].

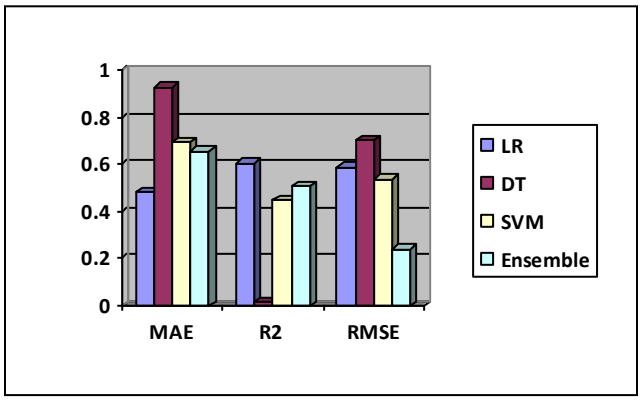

Fig. 8 Performance metrics comparison on the kitchenham

5.6. Performance Evaluation Outcomes on Various DL Models

Incorporating perspectives from relevant stakeholders provides a well-rounded and practical view of the application of deep learning (DL) in software cost estimation. Feature engineering should involve technical team input to include relevant software metrics (e.g., cyclomatic complexity, reuse ratio). DL outputs should be integrated with visualization and reporting tools (e.g., dashboards) to communicate uncertainty and assumptions clearly. Demonstrate long-term cost savings and risk reduction through case studies or pilot projects using DL models. Different DL models, including CNN and LSTM, were evaluated on the Desharnais, Maxwell, and Kitchenham datasets using various performance metrics, as shown in Table 7.

Table 7. Maxwell dataset

Layer Type	Output Shape	Param #
conv1d_1 (conv1d)	(None, 26, 64)	192
Flatten_1 (Flatten)	(None, 1664)	0
dense_3 (Dense)	(None, 50)	83250
dense_4 (Dense)	(None, 25)	1275
dense_5 (Dense)	(None, 1)	26

5.6.1. Effort Estimation Using CNN Model

Dataset: CNN Model R-squared, MSE, and RMSE for Maxwell Dataset (Table 7), Kitchenham Dataset (Table 8), and Deshanair Dataset (Table 9) and Figure 9 represent CNN model on different datasets [37] [38].

Table 8. Kitchenham dataset

Layer Type	Output Shape	Param #
conv1d_2 (conv1d)	(None, 266, 64)	192
Flatten_2 (Flatten)	(None, 17024)	0
dense_6 (Dense)	(None, 50)	851250
dense_7 (Dense)	(None, 25)	1275
dense_8 (Dense)	(None, 1)	26

Table 9. Deshanair dataset

Layer Type	Output Shape	Param #
conv1d (conv1d)	(None, 11, 64)	192
flatten (Flatten)	(None, 704)	0
Dense (Dense)	(None, 50)	35250
dense_1 (Dense)	(None, 25)	1275
dense_2 (Dense)	(None, 1)	26

Table 10. Compare errors on the CNN method with different datasets

Datasets	MAE	MSE	RMSE	R^2
Desharnais	0.1651	0.5045	0.5045	0.8941
Maxwell	0.7826	0.1893	0.4350	0.6913
Kitchenham	0.3452	0.1726	0.4155	0.4524

The Kitchenham, Desharnais, and Maxwell datasets, all frequently used in software cost estimation research, are used in this comparison of Machine Learning (ML) and Deep Learning (DL) techniques using standard performance metrics. Evaluation of the Deep Learning Models Next are presented the results of the experiments performed on the different datasets.

The following tables demonstrate the performance of each model when estimating LSTM and CNN Models using the datasets (Desharnais, Maxwell, and Kitchenham datasets), respectively. The values of MAE, MdAE MSE, MdAE, and RMSE are shown, which were obtained after applying the 10-fold cross-validation. For all the metrics used, the smaller the value, the better the result.

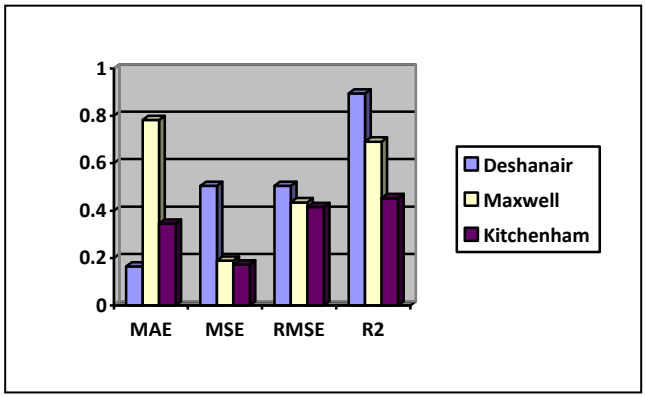

Fig. 9 CNN model on different datasets

5.6.2. Effort Estimation Using LSTM Model

Dataset: LSTM Model R-squared, MSE, and RMSE for Maxwell Dataset, Kitchenham Dataset, and Desharnais Dataset (Table 10) and Figure 10 represent LSTM model on different datasets. The Kitchenham, Desharnais, and Maxwell datasets, all frequently used in software cost estimation research, are employed in this comparison of Machine Learning (ML) and Deep Learning (DL) techniques, utilizing standard performance metrics (MAE, R^2, MSE, RMSE). Particularly on organized data sets like Maxwell and Desharnais, machine learning models like SVR demonstrated a moderate level of performance.

Table 10. Compare errors on the LSTM method with different database

Datasets	MAE	MSE	RMSE	R^2
Deshanair	0.2606	0.6296	0.0968	0.7934
Maxwell	0.0619	0.0008	0.0285	0.7299
Desharnais	0.0619	0.0004	0.0219	0.1395

ML models were consistently outperformed by Deep Learning Models (CNN, LSTM), particularly when it came to lower MAE, MSE, and RMSE and higher R^2 scores. The fact that LSTM performed the best across all datasets demonstrated how well it can identify complex nonlinear patterns or sequential patterns in graphical software project data.

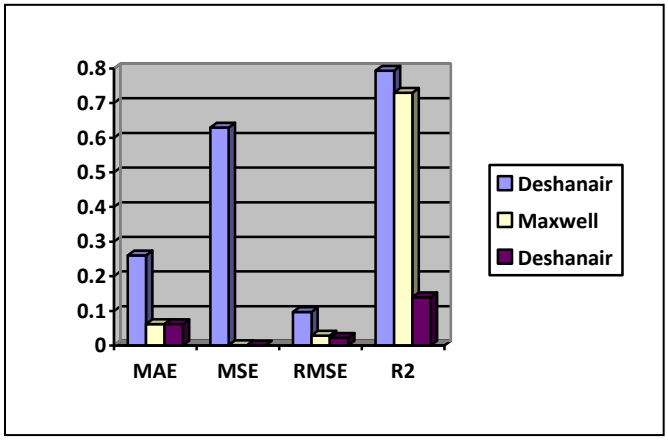

Fig. 10 LSTM model on different datasets

6. Conclusion

This study evaluates the effectiveness of many ML and DL models in software project cost estimation. The findings from the data set, obtained through the employment of ML and DL models, prove with certainty that dependence on model choice plays a significant role in the success and precision of estimating software costs. The SVM model fared better than other models on the Desharnais dataset as it achieved the highest R2 value of 0.804, which showed better prediction capabilities. In contrast, the Maxwell dataset showed that LR and SVM performed best, with LR achieving an MAE of 0.483 and R2 of 0.929; SVM, on its part, had an RMSE of 0.537. Kitchenham dataset reported the second highest R2 of 0.201 and 0.275 and the lowest MAE and RMSE of the series of 0.929 in favor of SVM and LR, respectively, as the best reliable models. This research evaluates the methods of

Machine Learning (ML) and Deep Learning (DL) in relation to the Kitchenham, Desharnais, and Maxwell datasets. It is noted that ML models had a significant advantage over Deep Learning Models (CNN, LSTM) in achieving lower MAE, MSE, and RMSE while also having greater R2 values. Out of all models, LSTM excelled the most across all datasets, showcasing its capacity to recognize intricate nonlinear or sequential graph patterns within project data from software engineering. Rankings also follow the same principle; in this case, lesser is better. The results of the experiments are presented in tables showing the performance of the various models when estimating the LSTM and CNN Models using the datasets.

These results emphasize the importance of using relevant models trained and tailored to please a specific task in order to improve the accuracy of estimating software cost. From the evidence presented in the results section, it is evident that SVM and LR are notable models that yield satisfactory performance metrics across various datasets. Thus, by using more accurate cost prediction approaches, using ML [39] techniques, in particular, LR and SVM hold the potential for improving software project planning and management.

However, more investigation into deep learning architectures and ensemble learning strategies may be necessary to improve the precision and resilience of software cost-estimating models. The following assessment measures are taken into account: R-squared, Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Squared Error (RMSE).

- RQ1-Related Algorithms Discussion: Table 5 indicates that Linear Regression is the second most used algorithm. Other potential benefits of Deep Learning techniques include automated feature extraction and improved performance.
- RQ2-Related Features Discussion: To display the key features and algorithms, groups are formed for features and algorithms. This decision preserves clarity but eliminates specific information.
- RQ3-Related Evaluation Parameters and Techniques Discussion: The chosen publications provide very few evaluation parameters. The majority of studies measured the model's quality using RMSE. The evaluation parameters MSE, R2, and MAE are additional.
- RQ4-Related Difficulties Discussion: Based on the articles' clear claims, challenges were reported. Much more may be stated about the model's accuracy as additional data is collected for testing and training.

Additionally, works collectively emphasize the critical factors influencing the success of using deep learning models for software cost estimation, including the integration of advanced modeling techniques, feature selection, data preprocessing, model optimization, and uncertainty quantification[40]. Building on the findings of this experiment, the next focus will be on developing a crop production forecast model based on DL [41]. Deep learning approaches have limitations and drawbacks that should be carefully considered, even though they may greatly enhance software project cost estimation. These challenges include identifying requirements, interpretability problems, overfitting risk, and the need for sizable, high-quality datasets. To get around these restrictions, a mix of methods, such as careful data preparation, regularization, and, in some cases, the use of simpler models, may improve the effectiveness of deep learning for software project cost estimation. Additionally, ongoing assessment and improvement of the models will ensure that they adapt to changing trends in software development [42].

Future Work

To get more thorough findings in the future, researchers will continue to experiment with new techniques and integrate cloud computing into estimation models.

Acknowledgments

The author would like to extend deepest appreciation to “Dr. Manojkumar Deshpande” and “Dr. Piyush Chaudhary” of the "Department of Computer Sciences and Engineering" at Prestige Institute of Engineering Management and Research, Indore, for their contributions to this study. To everyone who made this study possible, the author would like to extend their sincere gratitude.

References

[1] Victor Uc-Cetina, “Recent Advances in Software Effort Estimation Using Machine Learning,” *arXiv Preprint*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Robin Ramaekers, Radek Silhavy, and Petr Silhavy, “Software Cost Estimation Using Neural Networks,” *Software Engineering Research in System Science*, pp. 831-847, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Muhammad Usman et al., “Effort Estimation in Large-Scale Software Development: An Industrial Case Study,” *Information and Software Technology*, vol. 99, pp. 21-40, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Solomon Mensah et al., “Duplex Output Software Effort Estimation Model with Self-Guided Interpretation,” *Information and Software Technology*, vol. 94, pp. 1-13, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] Narciso Cerpa et al., “Evaluating Different Families of Prediction Methods for Estimating Software Project Outcomes,” *Journal of Systems and Software*, vol. 112, pp. 48-64, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[6] Vahid Garousi et al., "A Survey of Software Engineering Practices in Turkey," *Journal of Systems and Software*, vol. 108, pp. 148-177, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[7] Neelamdhab Padhy, R.P. Singh, and Suresh Chandra Satapathy, "Software Reusability Metrics Estimation: Algorithms, Models and Optimization Techniques," *Computers & Electrical Engineering*, vol. 69, pp. 653-668, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yansi Keim et al., "Software Cost Estimation Models and Techniques: A Survey," *International Journal of Engineering Research and Technology*, vol. 3, no. 2, pp. 1763-1768, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Saoud Sarwar, and Monika Gupta, "Proposing Effort Estimation of COCMO-II through Perceptron Learning Rule," *International Journal of Computer Application*, vol. 70, no. 1, pp. 29-32, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[10] Somya Goyal, and Anubha Parashar, "Machine Learning Application to Improve COCOMO Model Using Neural Networks," *International Journal of Information Technology and Computer Science*, vol. 10, no. 3, pp. 35-51, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, and Andrzej Kobylinski, "An Effective Approach for Software Project Effort and duration Estimation with Machine Learning Algorithms," *Journal of Systems and Software*, vol. 137, pp. 184-196, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] V. Venkataiah et al., "Application of Ant Colony Optimization Techniques to Predict Software Cost Estimation," *Computer Communication, Networking and Internet Security*, Springer, pp. 315-325, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Faiza Tahir, and Mahum Adil, "An Empirical Analysis of Cost Estimation Models on Undergraduate Projects Using COCOMO II," *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, Malaysia, pp. 105, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[14] Nazeeh Ghatasheh et al., "Optimizing Software Effort Estimation Models Using Firefly Algorithm," *arXiv Preprint*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] E.E. Miandoab, and F.S. Gharehchopogh, "A Novel Hybrid Algorithm for Software Cost Estimation Based on Cuckoo Optimization and k-Nearest Neighbor's Algorithms," *Engineering, Technology & Applied Science Research*, vol. 6, no. 3, pp. 1018-1022, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[16] Syed Sajid Ullah et al., "A Lightweight Identity-Based Signature Scheme for Mitigation of Content Poisoning Attack in Named Data Networking with Internet of Things," *IEEE Access*, vol. 8, pp. 98910-98928, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Iqbal H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy," *SN Computer Science*, vol. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Vachik S. Dave, and Kamlesh Dutta, "Neural Network-Based Models for Software Effort Estimation: A review," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 295-307, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[19] Ali Bou Nassif et al., "Neural Network Models for Software Development Effort Estimation: A Comparative Study," *Neural Computing & Applications*, vol. 27, no. 8, pp. 2369-2381, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[20] Jason Brownlee, "*Long Short-Term Memory Networks with Python Develop Sequence Prediction Models with Deep Learning*," Machine Learning Mastery, 2019. [Google Scholar]

[21] A.G. Priya Varshini et al., "Comparative Analysis of Machine Learning and Deep Learning Algorithms for Software Effort Estimation," *Journal of Physics: Conference Series*, vol. 1767, no. 1, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Farhad Akhbardeh, and Hassan Reza, "A Survey of Machine Learning Approach to Software Cost Estimation," *2021 IEEE International Conference on Electro Information Technology (EIT)*, USA, pp. 405-408, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] Govinda et al., "Framework for Estimating Software Cost Using Improved Machine Learning Approach," *Congress on Intelligent Systems*, pp.713-725, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Mohammad Alauthman, Ahmad al-Qerem, and Amjad Aldweesh, "Machine Learning for Accurate Software Development Cost Estimation in Economically and Technically Limited Environments," *International Journal of Software Science and Computational Intelligence*, vol. 15, no. 1, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Mustafa Hammad, "Software Cost Estimation using Stacked Ensemble Classifier and Feature Selection," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, pp. 183-189, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Karthick Panner Selvam, and Mats H0akan Brorsson, "Can Semi-Supervised Learning Improve Prediction of Deep Learning Model Resource Consumption?," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 6, pp. 74-83, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Lugo Garcia Jose Alejandro, Garcia Perez Ana María, and Delgado Martínez Ramsés, "Indicator Management in Software Projects: Current and Future Perspectives," *Revista Cubana de Ciencias Informatics*, vol. 3, no. 3-4, pp. 19-25, 2009. [Google Scholar] [Publisher Link]

[28] Shivangi Shekhar, and Umesh Kumar, "Review of Various Software Cost Estimation Techniques," *International Journal of Computer Application*, vol. 141, no. 11, pp. 31–34, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[29] R. Saljoughinejad, and V. Khatibi, "A New Optimized Hybrid Model Based on COCOMO to Increase the Accuracy of Software Cost Estimation," *Journal of Advances in Computer Engineering and Technology*, vol. 4, pp. 41-50, 2018. [Google Scholar] [Publisher Link]

[30] Anupama Kaushik, and Niyati Singal, "A Hybrid Model of Wavelet Neural Network and Metaheuristic Algorithm for Software Development Effort Estimation," *International Journal of Information Technology*, vol. 14, no. 3, pp. 1689-1698, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[31] Robert Marco, Nanna Suryana, and Sharifah Sakinah Syed Ahmad, "A Systematic Literature Review on Methods for Software Effort Estimation," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 2, pp. 434-464, 2019. [Google Scholar] [Publisher Link]

[32] Zahid Hussain Wani, and S.M.K. Quadri, "An Improved Particle Swarm Optimization-Based Functional Link Artificial Neural Network Model for Software Cost Estimation," *International Journal of Swarm Intelligence*, vol. 4, no. 1, pp. 38- 54, 2019. [CrossRef] [Publisher Link]

[33] Shotaro Minami, "Predicting Equity Price with Corporate Action Events Using LSTM-RNN," *Journal of Mathematical Finance*, vol. 8, no. 1, pp. 58-63, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[34] Panagiotis Barmpalexis et al., "Comparison of Multi-Linear Regression, Particle Swarm Optimization Artificial Neural Networks and Genetic Programming in the Development of Mini-Tablets," *International Journal of Pharmaceutics*, vol. 551, no. 1-2, pp. 166-176, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[35] Ch Anwar ul Hassan, and Muhammad Sufyan Khan, "An Effective Nature Inspired Approach for the Estimation of Software Development Cost," *2021 16th International Conference on Emerging Technologies (ICET)*, Islamabad, Pakistan, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[36] Simon Fong, Suash Deb, and Xin-she Yang, "How Meta-Heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics," *Proceedings of the Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, Singapore, pp. 3-25, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[37] A. Hussein et al. "Software-Defined Networking (SDN): the Security Review," *Journal of Cyber Security Technology*, vol. 4, no.1, pp. 1-66, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[38] Manohar K. Kodmelwar, Shashank D. Joshi, and V. Khanna, "A Deep Learning Modified Neural Network Used for Efficient Effort Estimation," *Journal of Computational and Theoretical Nanoscience*, vol. 15, pp. 11-12, pp. 3492-3500(9), 2018. [CrossRef] [Google Scholar] [Publisher Link]

[39] Priya Agrawal, and Shraddha Kumar, "Early Phase Software Effort Estimation Model: A Review," *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, Indore, India, pp. 1-8, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[40] Ishleen Kaur et al., "Neuro Fuzzy: COCOMO II Model for Software Cost Estimation," *International Journal of Information Technology*, vol. 10, pp. 181-187, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[41] Rohit Malik et al., "Software Reliability Estimation Using COCOMO II and Neuro Fuzzy Method," *International Journal of Emerging Technologies and Innovative Research*, vol. 5, no. 9 385-392, 2018. [Google Scholar]

[42] Fizza Mansoor et al., "Enhancing Software Cost Estimation Using Feature Selection and Machine Learning Techniques," *Computers, Materials & Continua*, vol. 81, no. 3, pp. 4603-4624, 2024. [CrossRef] [Google Scholar] [Publisher Link]