Journal of Computer Science

Original Research Paper

Software Development Effort Estimation Using Relational Database and Optimized Learning Mechanism

[1]Ravi Kumar B. N. and [2]Yeresime Suresh

[1]Department of Computer Science and Engineering, BMS Institute of Technology and Management, Bangalore, India
[2]Department of Computer Science and Engineering, Ballari Institute of Technology and Management, Ballari, India

Article history
Received: 06-01-2023
Revised: 28-02-2023
Accepted: 08-03-2023

Corresponding Author:
Ravi Kumar B. N.
Department of Computer Science and Engineering, BMS Institute of Technology and Management, Bangalore, India
Email: ravikumarbn@bmsit.in

Abstract: Accurately estimating the cost of software development is crucial for effective project planning and resource allocation. However, traditional cost estimation methods rely heavily on expert judgment and historical data, which can be time consuming and prone to errors. This study suggests a learning-based cost estimation model that leverages relational databases to improve accuracy. The proposed approach estimates project cost based on the effort required to complete software development, which is a key driver of the project cost. The proposed model is designed to address the challenges posed by the variability in open-source development, including variable team sizes, working hours, and expertise. The study collects and pre-processes data from open-source platforms and selects cost drivers and metrics based on logical rules and SQL queries. Moreover, we propose an optimized Artificial Neural Network (ANN) with augmented topology to automate the selection of neuron units, layers, and adjustment of learnable parameters according to the input variables. The proposed model is evaluated on a 100 open-source software repositories dataset and demonstrates its effectiveness in accurately estimating development cost. The system is implemented using Python and evaluated using performance parameters such as MSE, RMSE, MAE, and MMRE. Results indicate that our proposed model offers a more accurate and efficient approach to software cost estimation, especially for freelancers and outsourcing firms. The proposed model has the potential to save time and resources and improve the reliability and accuracy of software cost estimation.

Keywords: Software Project Development, Cost Estimation, Effort Estimation, Relational Database, Artificial Intelligence, Machine Learning

Introduction

Unlike any product development, the Software Development Process (SDP) also includes various activities to be performed in a defined sequence. These activities and the sequence depend on whether it is a software product or a project (Berntsson-Svensson and Aurum, 2006). Typically, in the software project process, in the early stage, a User Requirement Specification (URS) is an essential activity to understand the client's expectations (McGraw and Harbison, 2020). Further, the stages include activities like software or System Requirement Specification (SRS) and Technical Requirement Specification (TRS). Based on three requirement specifications, including URS, SRS, and TRS, the architect designs the software, and based on design, the development, coding, and debugging take place. In an organization, the overall effort varies depending on the project to project. However, there has been a continuous evolution into SDP from a simple waterfall method to scrum and agile (Bilgaiyan *et al*., 2017) to handle the uncertain dynamics of the context and reduce the effort to build software either as a project or product. Moreover, the effective Cost Estimation Model (CEM) or Effort Estimation Model (EEM) for software development provides an effective tool to manage the project or product development process seamlessly and cost effectively. The experiences gained during the past project and their data correlated with the various software development activities may provide heuristic information for estimating the efforts. However, there exists a lack of accessibility or availability of such past project information (Usman *et al*., 2014). Thus, the algorithmic-based traditional effort estimation models lack the

desirable accuracy because the complete information about the effort driver is unavailable (Shepperd and MacDonell, 2012). The accuracy of software cost estimation also depends on the volume of the data (Dolado, 2001).

The soft computing approaches nowadays are gaining popularity in building the effort estimation model even if the effort drivers are uncertain and incomplete (Suresh Kumar and Behera, 2020). Depending on the study and its goals, various Machine Learning (ML) methods and implementations are used in different research fields. Rapid development is taking place in estimating the cost of building the program. Among these factors are technological advancements, the skills of programmers working on developing projects, their expertise, and the programming languages they use. ML approaches are developed instead of purely mathematical or statistical calculations as in other approaches (Pospieszny *et al*., 2018). To develop an effective software cost prediction model, the application of ML can be an appropriate approach because it is capable of learning from the data of previous projects and adapting to significant differences that occur during the development of software projects.

The proposed research aims to suggest an effective process of the Software Development Effort Estimation (SDEE) method that takes software project details as its input and returns the estimated cost in terms of person hours required to accomplish the software development project. The data used in this study comes from Open-Source Platform (OSP) repositories, which are not linked to fixed work hours or patterns, limiting the scope of the study to freelancers and outsourcing firms. Employees working on proprietary software are typically scheduled based on a fixed schedule, whereas those working on open-source software are not. The proposed scheme offers accurate cost estimates, useful for freelancers and outsourcing firms, who often need to estimate costs for projects with limited information and resources. The proposed scheme can help these developers and organizations reduce costs by preventing overestimating resources and time, leading to more efficient project planning and resource allocation.

Methods for SDEE or Software Development Cost Estimation (SDCE) that are currently available fall primarily into three categories: (i) Algorithmic approaches, (ii) Non-algorithmic approaches, and (iii) Learning-based approaches. In SDCE, Constructive Cost Model (COCOMO) is one of the traditional methods which falls under the algorithmic category. This model was suggested by the author (Boehm, 2002). This COCOMO model has been around for quite some time and its current status demonstrates how reliable it is as a tool for assessing costs. In addition to failing early project estimates, this paradigm has a drawback. The COCOMO II model evolved as an improved version of COCOMO (Boehm *et al*., 1995). COCOMO II is better as it provides a more comprehensive and detailed approach to estimating software development costs. More detailed cost drivers can be applied to a wider range of software development projects. Similarly, the authors in the study (Khan *et al*., 2021; Keil *et al*., 2006; Menzies *et al*., 2013) have introduced some improvements and customization to the original COCOMO. However, in SDCE, determining the software metrics is challenging regarding their functional size.

The algorithmic approach includes function point analysis in its methodology (Parthasarathy, 2007). This strategy has the advantage of being able to be applied to many different languages and technologies. In function point analysis, measurements are based on two main components. A function point analysis measures software application functionality on data and transaction attributes. In a similar line of research, (Putnam, 1978) developed a multivariate cost estimation model. A major advantage of this model is that it is based on two factors, size and time, that are crucial to cost estimation and requires fewer parameters than COCOMO II. It has also been observed that the researchers have frequently explored the effectiveness of non-algorithmic approaches like expert judgment, Top-down, and bottom-up estimation. In the early stages of the SDCE's development, expert judgment was one of the conventional methods used by (Boehm *et al*., 2000).

A large part of the method relies on the expertise and experience of the expert. The expert's domain knowledge is more important than historical data. (Leung and Fan, 2002) reported on the Delphi technique, which follows the expert judgment approach. As far as understanding the impacts of incorporating a system is concerned, an expert can offer a knowledgeable and honest opinion. According to the top-down estimation approach, the project cost is estimated by taking into account the overall properties of the project. Alternatively, the bottom-up estimation method involves determining the cost of each software component and combining these elements to obtain the overall project cost estimate. In this method, smaller software components are evaluated individually to arrive at a final estimate. The literature has few recent works based on hybridizing multiple approaches in context non-algorithmic approaches. For example, (Nandal and Sangwan, 2018) combined Bat and Gravitational algorithms. (Nassif *et al*., 2019) reported a fuzzy regression model for effort estimation. The adoption of learning-based models is increasing with the availability of datasets. Neural networks, fuzzy logic, genetic algorithms, Bayesian networks, support vector regression, and analogy-based learning algorithms have all been used to estimate software costs (Wen *et al*., 2012; Gray and MacDonell, 1999). In order to improve the accuracy of estimation, researchers have proposed several Machine Learning (ML) models for software cost

estimation (Jeffery *et al*., 2000).

The accuracy of the predicted model varies when different historical project datasets or experimental designs are used (Heiat, 2002; Myrtveit and Stensrud, 1999). The authors (Monika and Sangwan, 2017) demonstrated that ANNs are the most effective model for developing estimating models. It has been discussed briefly how these models can be used to estimate software costs and their strengths and weaknesses. Many other research works explore ANN and ML approaches in SDCE. Ravi and Suresh (2022) reported an efficient and optimized neural network model to estimate the efforts in the early stages of development of the software project. The authors have adopted the COCOMO II dataset and the ANN model is optimized using a genetic algorithm. In a similar direction, the authors in the work of (Lee *et al*., 2022) presented a conceptual framework based on the ensemble of ANN and factor analysis methods to estimate the cost of executing large scale construction projects. Also, the usage of the fuzzy logic model for software effort estimation is seen in the study of (Nassif *et al*., 2019). In this study, the authors conducted a regression analysis to explore the effectiveness of different fuzzy models, namely Mamdani and Sugeno. The experimental outcomes demonstrated that Sugeno outperformed Mamdani fuzzy model when evaluated with the ISBSG dataset.

The application of the nature inspired optimization approach is used by (Ghatasheh *et al*., 2019). A firefly optimization technique optimizes the parameters required in the COCOMO based effort estimation models. The research work towards extracting the most influential cost drivers using statistical methods is conducted by (De Carvalho *et al*., 2021). This study adopts an extreme learning algorithm to predict software design efforts. Similar work can also be seen by Velarde *et al*. (2016). The variability in the cost variables is identified in the context of Source Lines of Codes (SLOC), function points analysis, and feature engineering tasks. The authors have also shown the effectiveness of data mining and ML techniques.

Hence, it can be analyzed that despite many works towards SDCE, none of the previous works have used a relational database. The previous works are limited to SDCE solutions, which only apply to scenarios with fixed working hours and expertise. Further research is needed to provide effective solutions for software projects, particularly for freelancers and outsourcing firms, to plan better and manage their projects.

Materials and Methods

The proposed system is specifically designed for freelancers and outsourcing firms working on open-source software projects, where fixed work hours or work patterns are not linked to the repositories. In such cases, it becomes difficult to estimate the cost of software development accurately. The proposed system overcomes this challenge by using a self-optimized learning mechanism and is trained on open-source project data to predict the cost of software development based on various cost drivers and metrics.

The research work reported in this study believes that using a relational database can help improve the accuracy of software effort estimation. By storing and managing data in a structured way, a relational database can provide a more detailed and reliable source of information for estimating the cost and effort involved in software development. Using an artificial neural network with optimization can further enhance the accuracy and efficiency of the cost estimation process.

The reason behind considering relational databases as a data source for implementing the proposed scheme is that it is designed to store and manage data in a structured way. This makes it ideal for storing and analyzing large amounts of data related to software development metrics. The proposed study uses information from 1800 repositories on the GitHub open-source platform to estimate a cost variable for software design effort estimation. The dataset was then used to train an Artificial Neural Network (ANN) based on the mechanism of augmented topology, which can automate the selection of adequate neuron units, layers, and adjustment of learnable parameters according to the input dataset.

The proposed SDCE method has several advantages. Firstly, it can provide accurate and fast development estimates. Secondly, it is based on a comprehensive dataset that takes into account various cost drivers related to different software metrics, developer activity, and project description. Thirdly, it can be used by freelancers and outsourcing firms to estimate the cost of open-source software development projects, which are not typically associated with fixed hours or work patterns. Additionally, the system can provide more transparency to clients by clearly outlining the costs associated with the development process.

Dataset Description

The dataset used in the proposed work is available in SQL format downloaded from the IEEE data port (IEEE DataPort, 2022). The dataset consists of information regarding software development metrics from 1800 repositories of the developers on an open-source platform, namely GitHub. However, several other open-source platforms exist, like GitLab, Bitbucket, Git bucket, source forge, phabricator, and Gitea. However, these platforms do not offer an efficient option to extract information about the developer activity and source code details; some are very new and contain fewer repositories. In contrast, GitHub serves all the purposes that are letting people see the source code of the repository and information about the developer's activity.

Figure 1 illustrates the contextual architecture of the dataset preparation procedure where software available on

the GitHub repositories consists of project description, Release, and Commits (RaC). Under RaC, software development or developer activities are mentioned and used as input for the logical engine driven by ANN. Here, the logical engine is a computing module that takes input as development activities and applies a set of logical rules and SQL queries. These rules are used to select the most optimal development metrics based on the factors such as repo size in M.B., line of codes, and stars. The top 100 repositories with more than 5 M.B. were selected using restful API to build the dataset in SQL data format.

Dataset Exploratory Analysis

The proposed research adopts a methodical approach to implementing a system for SDCE based on the given requirements description of a software project. Firstly, a web development platform, namely the WAMP server, is installed to load and read the dataset in the development environment since it is presented in the SQL data type. Afterward, exploratory analysis is carried out to understand the dataset characteristics and its property towards applying suitable pre-processing operations. A Pandas library of Python is connected to the MySQL server from the WAMP dashboard. A web address http://localhost/phpmyadmin/ is further entered in the browser, showing two tables in the database, namely release_wise_effort, and Repo_info_pv_vec, as illustrated in Fig. 2.

The dataset presented in Fig. 2 highlights the various data types associated with cost variables, each carrying a specific meaning as explained below:

- Text data type signifies that there is no limit on the number of characters that can be accommodated within the variable
- Varchar data type, on the other hand, refers to a string that can only hold a limited number of characters. For instance, varchar (100) would indicate that the string can have a maximum of 100 characters
- Int data type denotes a numerical value without a decimal point
- Float and double data types represent numerical values with a decimal point. However, double has a higher precision than float, meaning that while float can accommodate up to 8 digits after the decimal point, double can hold up to 32 digits

After conducting an initial analysis of the provided dataset, it was observed that the tables within the dataset are not linked and the unique repository names are not the same in both tables. Therefore, each table needs to be handled separately. Multiple libraries and packages are required to read the MySQL database into a pandas data frame:

- SQL alchemy library is preferred since we can avoid writing complicated statements
- Firstly, an Object Relational Model (ORM) is created as a pre-requite for the further analysis
- ORM basically creates objects or classes in Python, analogs to rows and tables in SQL
- Every table becomes a class in Python and every row becomes an object in Python
- To convert SQL tables into python classes, a package called sqlacodegen is used. Sqlacodegen means SQL alchemy code generator
- The format of the sqlacodegen command is as follows
- Sacandaga <dbtype> + <python library>://<user>:<pass>@<ip>/<dbname>

Once the above command is executed, it generates the python file necessary for reading the database and is directly converted to a panda's data frame. The attributes of the dataset named 'release wise effort' are shown in Fig. 3.

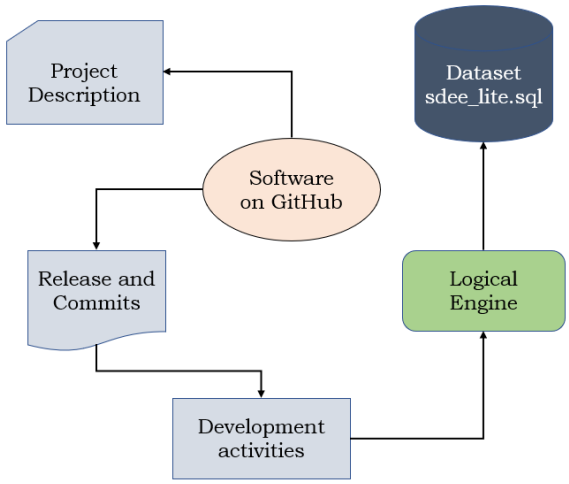


Fig. 1: Contextual illustration of how the dataset is prepared

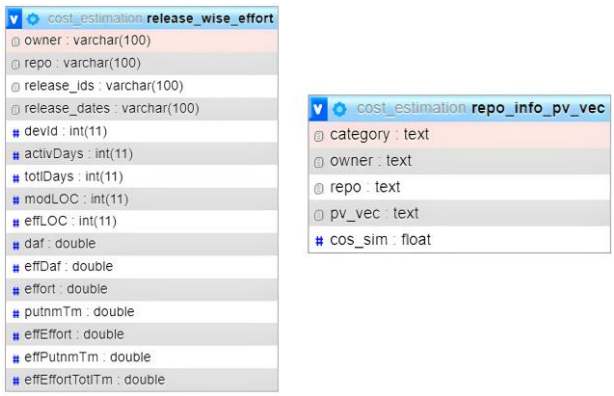


Fig. 2: Depiction of the dataset in SQL format

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7400 entries, 0 to 7399
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   owner            7400 non-null   object
 1   repo             7400 non-null   object
 2   release_ids      7400 non-null   object
 3   release_dates    7400 non-null   object
 4   devId            7400 non-null   int64
 5   activDays        7400 non-null   int64
 6   totlDays         7400 non-null   int64
 7   modLOC           7400 non-null   int64
 8   effLOC           7400 non-null   int64
 9   daf              7400 non-null   object
 10  effDaf           7400 non-null   object
 11  effort           7400 non-null   object
 12  putnmTm          7400 non-null   object
 13  effEffort        7400 non-null   object
 14  effPutnmTm       7400 non-null   object
 15  effEffortTotlTm  7400 non-null   object
dtypes: int64(5), object(11)
memory usage: 925.1+ KB
```

Fig. 3: Attributes of the dataset for release_wise_effort

```
owner
google     36
minio      52
vapor      53
instana    57
Netflix    74
Name: repo, dtype: int64
```

Fig. 4: Topic modeling of owners based on the repo

Dataset Analysis (Release_Wise_Effort)

From the above analysis, it can be observed that there are 7400 entries ranging from the index number 0 to 7399 with 16 attributes in columns. The dataset comprises 16 variables with type 5 numeric and 11 categorical variables. Further dataset analysis is done regarding the number of owners (i.e., repositories owners) and it is observed that 1094 owners are associated with different projects. In order to better understand, topic modeling is done, where the owners are grouped according to different topics analyzed from the repo column of the dataset. Figure 4 illustrates that the owner belongs to distinct project topics.

Figure 4 it can be seen that the 1094 owners belong to 5 different groups, such as google, minio, vapor, instana, and Netflix, with their corresponding number of projects in the repo column. The study then identifies each column's unique, missing, and repetitive values. The analysis shows no missing and repetitive values in the dataset. Therefore, the dataset does not require to be processed with any kind of pre-processing operation except the object data types conversion to the numerical representation for feature analysis.

Descriptive Statistics

The dataset has a total of 16 variables and not all variables are important from the modeling perspective. In this regard only 11 variables are considered as illustrated below:

- Devoid typically refers to developer I.D. or identifier, which is a unique code or number assigned to a developer or a team member working on a software project
- Active days generally refer to the number of working days taken for a software project to be completed. It may include days when the developer worked on the project and exclude days when they were not working, such as weekends or holidays
- TotlDays refers to the total number of days taken for a software project to be completed. It may include both working days and non-working days
- Mode of Operation (MoD) is a factor used to adjust the LOC measure to account for the complexity and characteristics of the software project. It takes into consideration factors such as the programming language used, the development environment, the level of documentation required, and the overall complexity of the project
- Effort per Line of Code (EffLoC): It is a measure of the average amount of effort required to develop one line of code in a software project
- Defect Adjustment Factor (DAF): It is a multiplier used to adjust the estimated effort required for a software project based on the expected number of defects or errors in the project. It is calculated based on the complexity of the project, the experience level of the development team, and other factors
- Effective Defect Adjustment Factor (EffDAF): It is the product of the DAF and a factor that accounts for the quality of the development process. This factor is usually based on the historical defect rate of the development team
- Effort PutnumTm: It is the estimated effort required for a software project, calculated using the Putnam model, which is a mathematical model for estimating software development effort based on project size and complexity
- EffEffort: It is the estimated effort required for a software project, adjusted for factors such as team experience, project complexity, and development environment
- EffPutnumTm: It is the estimated effort required for a software project, calculated using the Putnam model and adjusted for factors such as team

experience, project complexity, and development environment

- EffEffortToTalTm: It is the ratio of the estimated effort required for a software project to the total time available for the project. This metric is used to assess the feasibility of a project within a given time frame

In order to understand the distribution of the data and identify any outliers or unusual values that may need to be addressed, a descriptive statistic such as mean, standard deviation, minimum, maximum, and quartiles (such as 25, 50, and 75%). are computed for each input variable as shown in Table 1. These descriptive statistics are used to summarize and describe the characteristics of a data set, such as the central tendency, variability, and distribution of the data.

By computing descriptive statistics for these variables, significant insight can be gained into the characteristics of the data, such as the average project size, the range of project complexity, and the distribution of developer experience levels. This information can be used to develop more accurate cost estimation models, identify areas where improvements can be made, and assess the risks associated with a particular project.

It can be seen that the count for each variable is the same i.e., 7400 which also indicates there is no ambiguity or missing values. Similarly, the mean value for each variable gives an idea of the typical value for that variable across all the data points in the adopted dataset. Specifically: For variable dev id, the mean value tells the average number of developers who worked on each project. For dev id, the mean value tells the average number of developers who worked on each project. For active days, the mean value tells the average number of working days it took to complete a project. For totlDays, the mean value tells the average number of days it took to complete a project, including non-working days. For LOC, the mean value indicates the average number of lines of code in a project. For daft, the mean value tells the average development adjustment factor for a project. Like this, the mean statistics give the average analysis of each variable related to cost analysis. Next, the Standard Deviation (StD) value for each variable gives a measure of the variability or spread of the data around the mean. For dev id, the std value gives how much the number of developers who worked on each project varied from the average value. For active days, the std value exhibits how much the number of working days it took to complete a project varied from the average value. On the other hand, the minimum (min), maximum (max) and quartile values for each variable provide additional information about the distribution of the data. The minimum value for each variable gives the smallest observed value in the dataset. For example, the minimum value for the dev id would be the smallest number of developers who worked on a project. The maximum value for each variable gives the largest observed value in the dataset. For example, the maximum value for LOC would be the largest number of lines of code in a project. The quartile values (25^{th} percentile, 50^{th} percentile, and 75^{th} percentile) divide the dataset into four equal parts. The 25^{th} percentile (also known as the first quartile) represents the value below which 25% of the data falls. The 50^{th} percentile (also known as the median) represents the value below which 50% of the data falls. The 75^{th} percentile (also known as the third quartile) represents the value below which 75% of the data falls. The quartile values can be useful in understanding the distribution of the data, particularly when the data is not normally distributed. For example, a large difference between the 75^{th} percentile and the median may indicate that the data is skewed toward higher values. On the other hand, a small difference between the quartiles may indicate that the data is more evenly distributed. In general, the minimum, maximum, and quartile values can provide insight into the range and distribution of the data, which can help in identifying any outliers or unusual data points. Further, more insights are drawn using the frequency distribution of each variable by dividing the range of values into intervals, or "bins" and counting the number of observations that fall into each bin. By looking at the shape of the plot shown in Fig. 5 it can be analyzed a sense of the central tendency and variability of the data, as well as identify any skewness or outliers. As observed in devId, most of the values are in the range of 14000 to 24000. However, the active days and totlDays cannot be zero, but the zero indicates very small projects requiring negligible effort. The attributes such as 'modLoC', 'effLOC', 'daf', 'effDaf', 'effort', 'PutnmTm', 'effEfort', and 'effPutnmTM', have a similar histogram pattern, which may affect output effEffortTotlTm exponentially.

Dataset Analysis (t_repo_info_pv_vec)

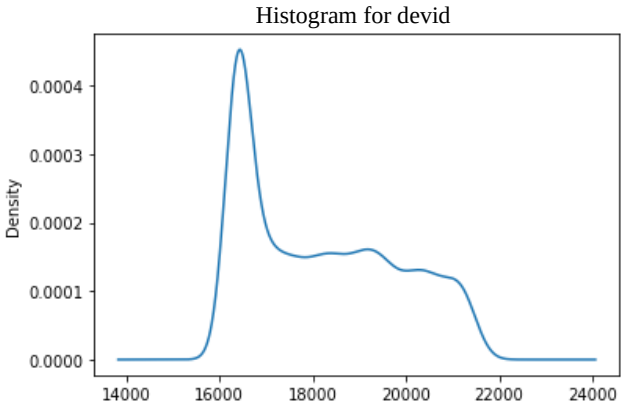
The exploratory analysis is done for a second table of the dataset, t_repo_info_pv_vec which has 5 attributes in the column header and 13042 samples in the row, Table 2.

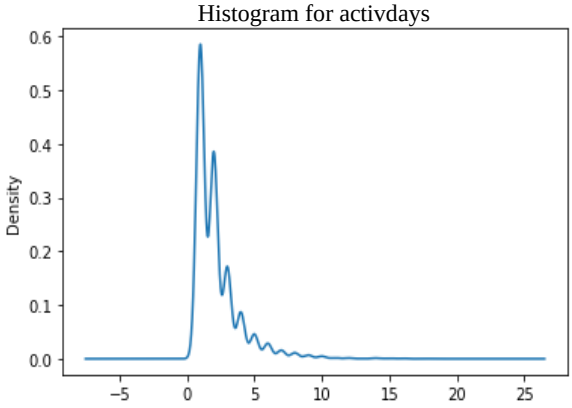
This dataset consists of project description details, which are used to extract cosine similarity (cos_sim) among the different projects using paragraph vector (pv_vec). Since the project description is presented in the form of text data and to process this text, it must be converted into numerical representation as a finite length vector.

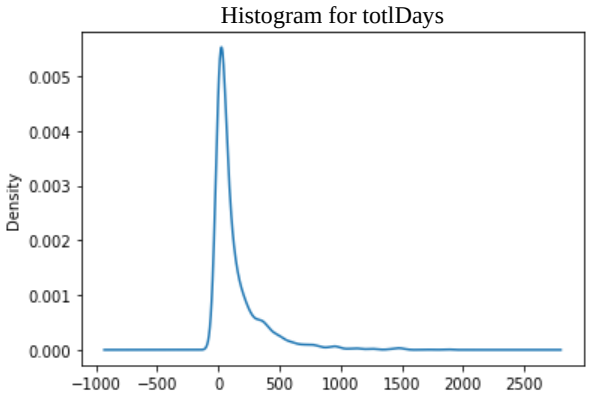
Basically, it is a feature representation of the project description, which will be provided as input by the user to the proposed cost estimation system. As software project descriptions can be represented with vectors of the same length, the cosine similarity threshold must be met to categorize two-word vectors as similar. For implementing an efficient and fast system, keeping the length of such vectors constant for every source code sample is crucial. In this way, the proposed cost estimation model is trained and built from the input of both datasets, namely release_wise_effort and t_repo_info_pv_vec.

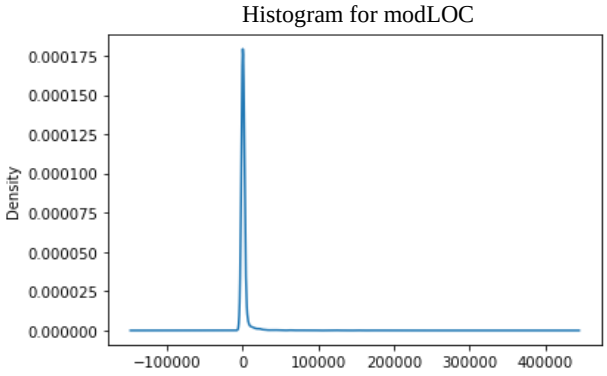
Table 1: Descriptive statistics of the dataset attributes (input variables)

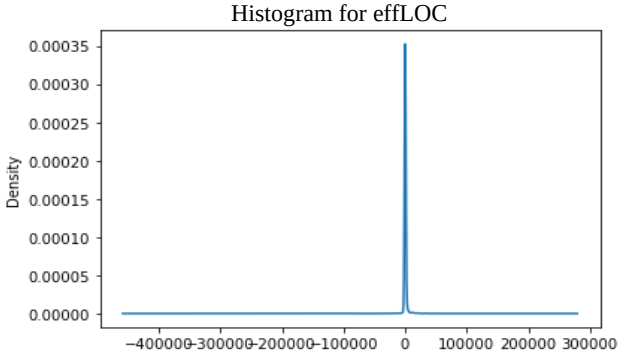
	Count	Mean	std	min	25%	50%	75%	max
devId	7400	18186.545270	1635.334306	16383.000000	16383.000000	17935.500000	1.95E+04	21503.000000
activDays	7400	2.235270	1.765530	1.000000	1.000000	2.000000	3.00E+00	18.000000
totlDays	7400	141.695676	216.275946	1.000000	16.000000	58.000000	1.74E+02	1865.000000
modLOC	7400	2107.307973	11670.106270	1.000000	20.000000	103.000000	5.62E+02	296396.000000
effLOC	7400	329.914189	6101.548859	-273940.000000	0.000000	20.000000	1.37E+02	94831.000000
daf	7400	963.241160	6272.901280	0.666667	13.000000	53.500000	2.34E+02	170795.000000
effDaf	7400	203.604147	3174.973469	-52185.000000	0.000000	11.071429	6.38E+01	94831.000000
effort	7400	11.969864	218.493036	0.000000	0.000127	0.297228	2.00E+00	12500.282390
putnmTm	7400	508.128779	1519.686391	0.153290	20.372171	79.731437	3.42E+02	31179.621150
effEffort	7400	3.030590	150.097236	-532.054300	0.000000	0.000056	6.75E-02	12481.527280
effPutnmTm	7400	116.158100	931.253909	-30563.312000	0.000000	16.434617	9.33E+01	16330.877750
effEffortTotlTm	7400	0.201634	7.184596	-124.125600	0.000000	0.000000	1.32E-07	328.067874

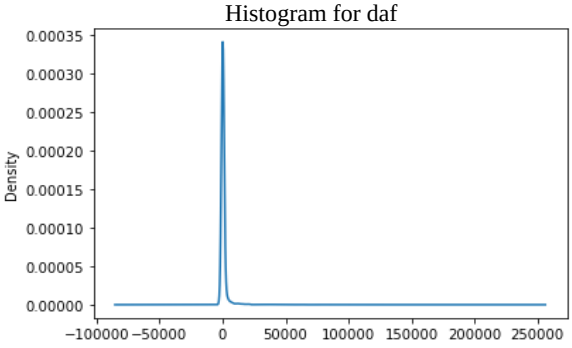












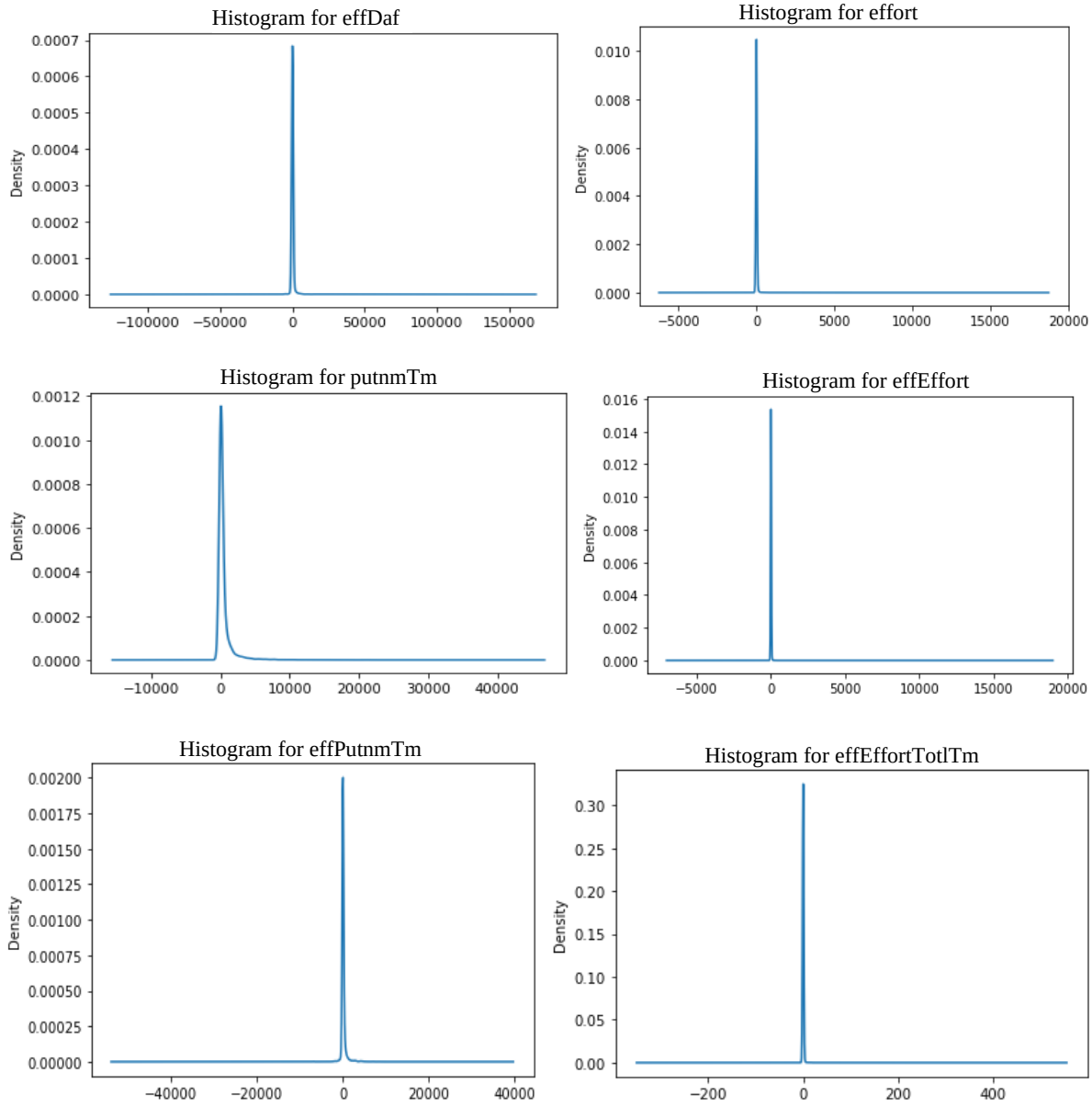


Fig. 5: Histogram plot for correlation analysis

Table 2: Visual depiction of data attributed [t_repo_info_pv_vec]

	Category	Owner	Repo	pv_vec	cos_sim
0	Configuration_libraries	Juxt	Aero	[-6.3111968e-03 7.2230858e-04 -4.5987987e-03 ...	1.000000
1	Configuration_libraries	Pd	Figgy	[0.00492741 0.00173693 -0.00801723 0.005146...	-0.122840
2	Configuration_libraries	Kelseyhightower	Envconfig	[6.8379391e-04 8.3461199e-03 -3.9898441e-03 ...	-0.129980
3	Configuration_libraries	Jeffgarland	Liaw2015	[2.9276814e-03 -7.2899782e-03 -7.2383620e-03 ...	0.240423
4	Configuration_libraries	Taneryilmaz	Libconfigini	[-0.00100779 -0.00632555 0.00106837 -0.003088...	-0.061510

Once the model is trained, it takes input from the user as project description type of programming language, operating system, software title, and features. A contextual illustration of the cost estimation system is shown in Fig. 6.

Figure 6, the proposed SDEE system takes input from the project description table (t_repo_info_pv_vec) and multiple development metrics from the table of development activities (release_wise_effort). The proposed cost estimator model is then trained using machine learning models.

Specifically, the study has implemented an ANN which is configured with a self-optimization mechanism

based on the concept of Augmented Topologies (AT) using an evolutionary algorithm. The mechanisms of AT enable an optimal adjustment of the neuron units on each layer, the number of the hidden layer, and learnable parameters. The neural network optimization is done using a genetic algorithm specifically designed for evolving ANN. Figure 7 illustrates the mechanism of augmenting neural network topology.

The algorithm starts with a population of small, simple neural networks and evolves them over time by applying genetic operations such as mutation, crossover, and selection. It also introduces new, randomly generated nodes and connections to the network, allowing it to evolve more complex topologies over time. A detailed discussion of this optimized ANN model can be found in the work of (Ravi and Suresh, 2022).

Once the model is trained, the user can give input by taking different information associated with the software projects, like a software project name or title, a short description of the project, the programming language preferred, operation system support, and other features. After receiving this input from the user, the system, based on its learning from a trained dataset containing different attributes like developer activities and project description. The algorithm then figures out the different factors associated with cost in terms of the number of developers required and time in months numerically given as follows:

- Developers: The number of developers D involved in making repository r
- Time: The cumulative time t spent in developing the repository, expressed as follows:

$$t^r = \sum_{i=1}^{j} \left(t_s^i - t_e^i\right) / \ j \tag{1}$$

where, t_s^i is the starting time of developing the i^{th} release of repository r, and t_e^i what represents the end-time of the i^{th} release? It is expressed in days, months, or years:

- Effort: Effort (e) required to develop a repository, numerically given as follows:

$$e^r \mid D^r \mid \times t^r \tag{2}$$

where, the effort is measured according to the units of $= |D^r|$ and t^r i.e., developer days or developer's month or years. The next section presents the performance analysis of the system with respect to multiple performance parameters.

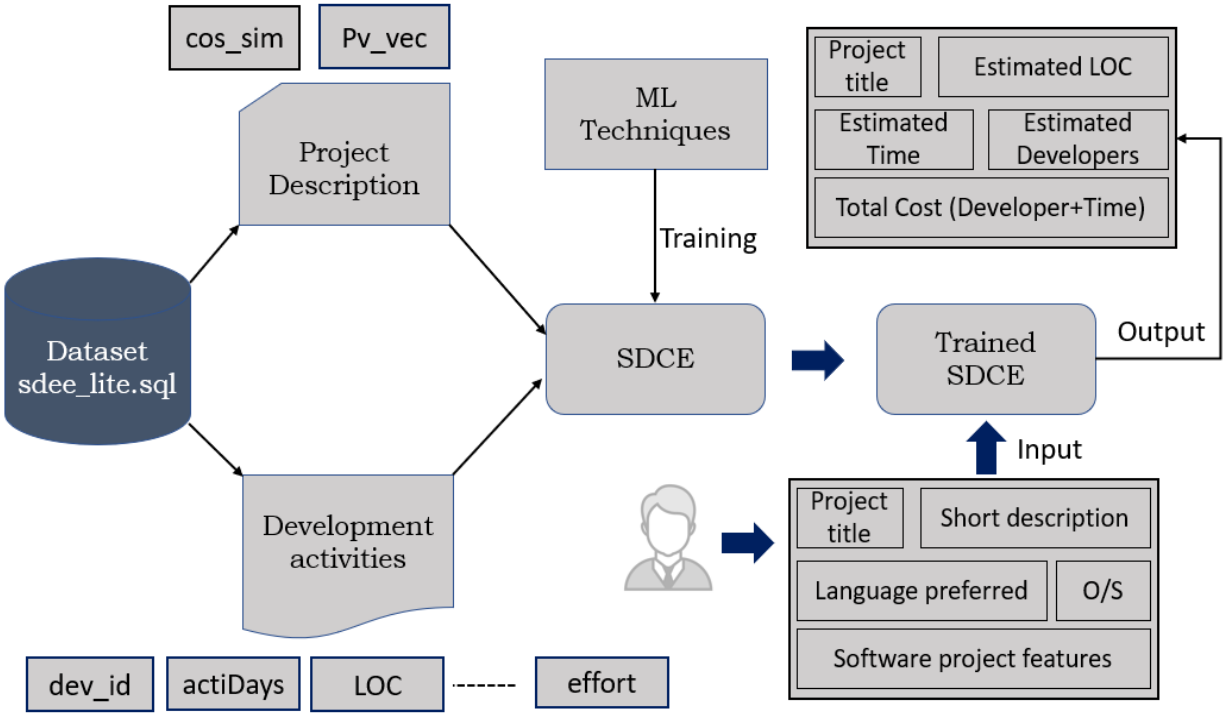


Fig. 6: The contextual architecture of the proposed system

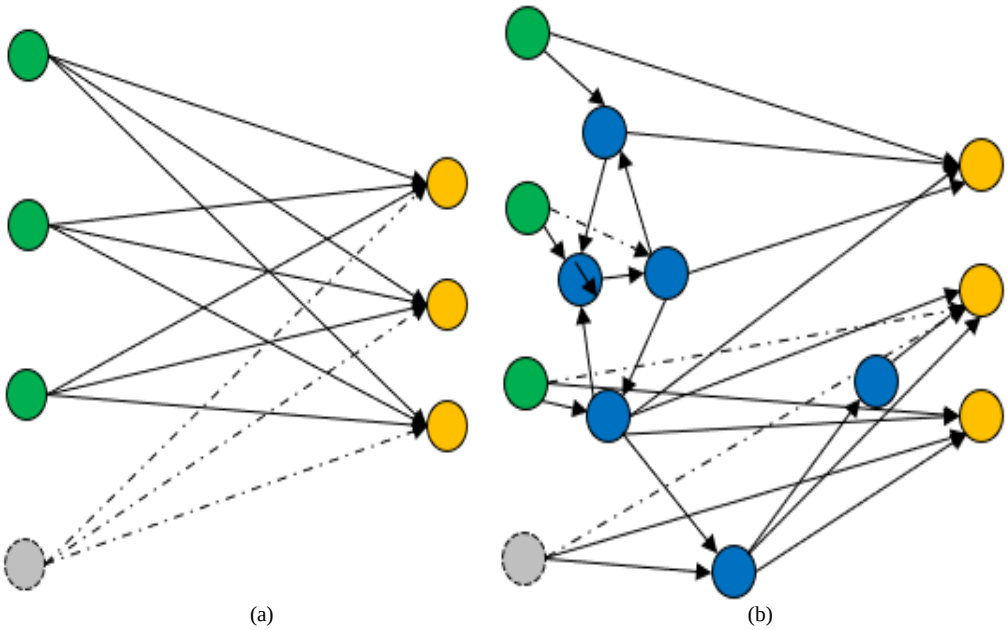


Fig. 7: ANN using Augmentation; (a) Initial architecture of ANN (b) Augmented topologies of ANN

Results and Discussion

The design and development of the proposed system are done using python programming language and execution on Anaconda. This section discusses the performance metrics followed by outcome analysis to justify the scope and effectiveness of the proposed system.

- Mean Squared Error (MSE): MSE is being calculated to determine how close a fitted line is to data points and measure the quality of the numerically given as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y - y'\right)^2 \tag{3}$$

In the above equation, y denotes actual effort and y' refers to the estimated effort for software project i and N represents the total software project:

- Root Mean Square Error (RMSE): RMSE is the square root of MSE, numerically given as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y - y'\right)^2} \tag{4}$$

- Mean Absolute Error (MAE): The MAE represents the average absolute error. The average is calculated using the absolute function and has a lower sensitivity to outliers. This can be given as follows:

$$MAE = \frac{1}{N}\cdot\sum_{i=1}^{n}\left\lfloor\left(y - y'\right)\right\rfloor \tag{5}$$

- Mean Magnitude of Relative Error (MMRE): In MMRE, the estimated effort is compared to the actual effort of software projects. The most accurate estimation process has a minimum MMRE, numerically given as follows:

$$MMRE = \frac{1}{N}\cdot\sum_{i=1}^{n}\frac{\left\lfloor\left(y - y'\right)\right\rfloor}{y} \tag{6}$$

The study also has implemented other learning classifiers such as Support Vector Regression (SVR) and Linear Regression (LR) for extensive analysis. These learning models have also been widely used in the literature.

Table 3 shows the numerical outcome obtained for learning models implemented for estimating the SDCE in terms of overall effort (number of developers and month). The outcomes are given concerning different performance metrics. The MSE measures the overall training of a learning algorithm. Essentially, it shows the difference between actual and projected data observations. The analysis of the MSE score indicates that the ANN outperforms other learning models as it has got better generalization over the input data distribution. Lower MSE shows effective training of the learning model. It can also be evident in other cases

like RMSE, MAE, and MMRE. RMSE also helps in understanding the pre-processing step of requirement re-training. The analysis in Fig. 8 showed that the proposed ANN achieved 82% improvement over LR and 29% improvement over SVR. The above numerical outcome and visual depiction of MRRE show the proposed study's effectiveness. Here also, ANN performs well compared to other techniques. ANN has got 95% of improvement over LR, which is not below 30%, which is an acceptable cost overrun ratio for software projects in general. In another case, it is also far below 80%. The overall numerical outcome shows the proposed ANN's effectiveness regarding the cost overrun ratio. The figure above shows that MSE scores for ANNs are higher in LRs and SVRs since they contain fewer trainable parameters. Accordingly, SVR and LR exhibit underfitting issues. Learning models are also evaluated using the metric RMSE.

Advantages and Scope of the Proposed System

The proposed system for software cost estimation using a relational database and optimized evolving learning method has several advantages over traditional methods. Some of the advantages are:

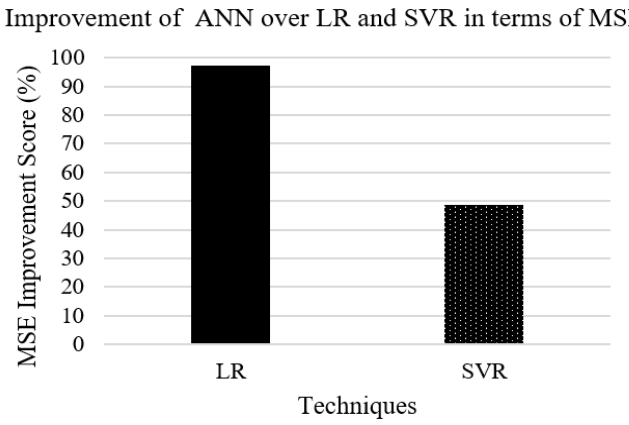
1. Improved accuracy: The proposed system uses machine learning algorithms that are trained on a large dataset to estimate the software development costs. This results in more accurate cost estimations compared to traditional methods
2. Speed: The proposed system is designed to provide fast development estimates, which can save time and resources for software development teams
3. Scalability: The proposed system can handle large datasets, making it suitable for use in enterprise level software development projects
4. Customization: The proposed system can be customized to suit the specific needs of different software development projects
5. Cost reduction: Accurate cost estimations can help organizations reduce costs by preventing overestimating resources and time

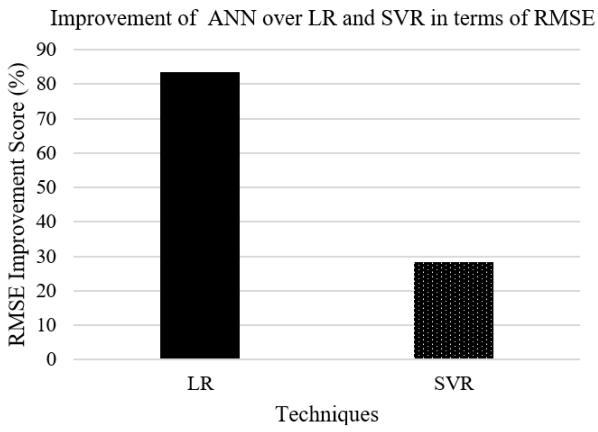
Some Real Time Use Cases

1. Software development firms: Software development firms can use the proposed system to estimate the cost and resources required for different software development projects
2. Freelance developers: Freelance developers can use the proposed system to estimate the cost of their services for different software development projects
3. Outsourcing firms: Outsourcing firms can use the proposed system to estimate the cost and resources required for different software development projects outsourced to them
4. Project management: Project managers can use the proposed system to estimate the cost and resources required for different software development projects and plan the project accordingly

Table 3: Comparative analysis

Learning model	MSE score	RMSE score	MAE score	MMRE score
LR	17930.60	133.9051	120.18	288.79
SVR	935.84	30.5915	24.09	52.11
ANN	480.82	21.9276	10.53	38.07





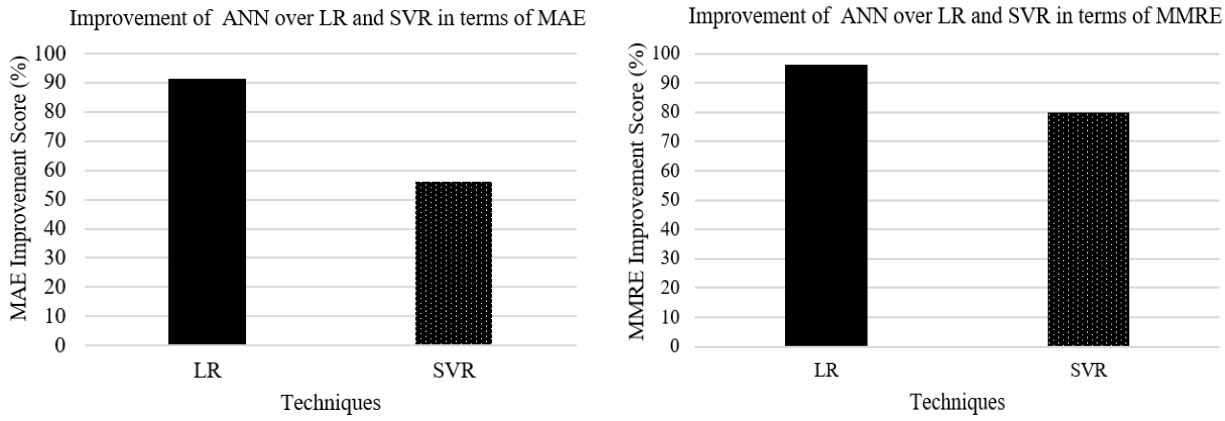


Fig. 8: Analysis of ANN improvement over SVR and L R

Conclusion

In this study, we propose an augmented learning-based effort estimation model that leverages relational databases to improve cost estimation accuracy in open-source software development. Our model addresses the challenges posed by the variability in team sizes, working hours, and expertise by collecting and pre-processing data from open-source platforms and selecting cost drivers and metrics based on logical rules and SQL queries. To automate the selection of adequate neuron units and layers and the adjustment of learnable parameters according to the input dataset, we propose an optimized form of Artificial Neural Network (ANN) based on the mechanism of augmented topology. The experimental results demonstrate that our proposed model provides a more accurate and efficient approach to software cost estimation, particularly for freelancers and outsourcing firms. Our proposed model has the potential to save time and resources and improve the reliability and accuracy of software cost estimation in open-source software development.

There are several directions for future work that can build on our proposed augmented learning-based effort estimation model. The model can be extended to incorporate data from other sources, such as enterprise software development projects, to improve cost estimation accuracy further. Secondly, our proposed model can be enhanced by incorporating more advanced machine learning techniques such as ensemble, transfer, and deep learning algorithms to improve the model's ability to learn from complex and diverse datasets. Thirdly, we plan to extend our model to include software development efforts beyond just coding, such as project management and testing, to provide a more comprehensive cost estimation model that takes into account all aspects of software development. Lastly, we can plan to explore the application of our proposed model to other domains such as healthcare and finance, where accurate cost estimation is also critical for project planning and resource allocation.

Overall, our proposed model provides a strong foundation for future research in software cost estimation and we believe that future work in this area will continue to improve the accuracy, reliability, and efficiency of cost estimation methods.

Acknowledgment

The authors of this manuscript would like to express their gratitude to the department of computer science and engineering, BMS institute of technology, for their efforts in guiding in the context of the current research work with for the constructive feedback which improved the submission. No funding was received to assist with the preparation of this manuscript. The authors have no conflicts of interest to declare relevant to this article's content.

Funding Information

The authors have not received any financial support or funding to report.

Author’s Contributions

Ravi Kumar B. N.: Work implementation and drafted the article.

Yeresime Suresh: Reviewed, guidance on ideology and implementation.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all other authors have read and approved the manuscript and no ethical issues are involved.

References

Berntsson-Svensson, R., & Aurum, A. (2006, September). Successful software project and products: An empirical investigation. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (pp. 144-153). https://doi.org/10.1145/1159733.1159757

Bilgaiyan, S., Sagnika, S., Mishra, S., & Das, M. (2017). A Systematic Review on Software Cost Estimation in Agile Software Development. *Journal of Engineering Science & Technology Review*, *10*(4). https://doi.org/10.25103/jestr.104.08

Boehm, B. W. (2002). Software engineering economics. *Software pioneers: Contributions to Software Engineering*, 641-686. https://doi.org/10.1007/978-3-642-59412-0_38

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches-A survey. *Annals of Software Engineering*, *10*(1-4), 177-205. https://doi.org/10.1023/A:1018991717352

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, *1*, 57-94. https://doi.org/10.1007/BF02249046

De Carvalho, H. D. P., Fagundes, R., & Santos, W. (2021). Extreme learning machine applied to software development effort estimation. *IEEE Access*, *9*, 92676-92687. https://doi.org/10.1109/access.2021.3091313

Dolado, J. J. (2001). On the problem of the software cost function. *Information and Software Technology*, *43*(1), 61-72. https://doi.org/10.1016/s0950-5849(00)00137-3

Ghatasheh, N., Faris, H., Aljarah, I., & Al-Sayyed, R. M. (2019). Optimizing software effort estimation models using firefly algorithm. *arXiv preprint arXiv:1903.02079*. https://doi.org/10.4236/jsea.2015.83014

Gray, A. R., & MacDonell, S. G. (1999). Software metrics data analysis-exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, *4*, 297-316. https://doi.org/10.1023/A:1009849100780

Heiat, A. (2002). Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology*, *44*(15), 911-922. https://doi.org/10.1016/S0950-5849(02)00128-3

IEEE DataPort. (2022). Software development effort estimation," [Online]. https://ieee-dataport.org/keywords/software-development-effort-estimation

Jeffery, R., Ruhe, M., & Wieczorek, I. (2000). A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data. *Information and Software Technology*, *42*(14), 1009-1016. https://doi.org/10.1016/S0950-5849(00)00153-1

Keil, P., Paulish, D. J., & Sangwan, R. S. (2006, May). Cost estimation for global software development. In *Proceedings of the 2006 International Workshop on Economics Driven Software Engineering Research* (pp. 7-10). https://doi.org/10.1145/1139113.1139117

Khan, J. A., Khan, S. U. R., Khan, T. A., & Khan, I. U. R. (2021). An amplified COCOMO-II based cost estimation model in global software development context. *IEEE Access*, *9*, 88602-88620. ahttps://doi.org/10.1109/access.2021.3089870

Lee, J. G., Lee, H. S., Park, M., & Seo, J. (2022). Early-stage cost estimation model for power generation project with limited historical data. *Engineering, Construction and Architectural Management*, *29*(7), 2599-2614. https://doi.org/10.1108/ecam-04-2020-0261

Leung, H., & Fan, Z. (2002). Software cost estimation. In *Handbook of Software Engineering and Knowledge Engineering: Volume II: Emerging Technologies* (pp. 307-324). https://doi.org/10.1142/9789812389701_0014

McGraw, K. L., & Harbison, K. (2020). *User-centered requirements: The scenario-based engineering process*. CRC Press. https://doi.org/10.1201/9781003064138

Menzies, T., Brady, A., Keung, J., Hihn, J., Williams, S., El-Rawas, O., ... & Boehm, B. (2013). Learning project management decisions: A case study with case-based reasoning versus data farming. *IEEE Transactions on Software Engineering*, *39*(12), 1698-1713. https://doi.org/10.1109/tse.2013.43

Monika, & Sangwan, O. P. (2017). Software effort estimation using machine learning techniques. 2017 *7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*. https://doi.org/10.1109/confluence.2017.7943130

Myrtveit, I., & Stensrud, E. (1999). A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering*, *25*(4), 510-525. https://ieeexplore.ieee.org/abstract/document/799947/

Nandal, D., & Sangwan, O. P. (2018). Software cost estimation by optimizing COCOMO model using hybrid BATGSA algorithm. *International Journal of Intelligent Engineering and Systems*, *11*(4), 250-263. https://doi.org/10.22266/ijies2018.0831.25

Nassif, A. B., Azzeh, M., Idri, A., & Abran, A. (2019). Software development effort estimation using regression fuzzy models. *Computational Intelligence and Neuroscience*, *2019*. https://doi.org/10.1155/2019/8367214

Parthasarathy, M. A. (2007). *Practical software estimation: Function point methods for insourced and outsourced projects*. Pearson Education India. ISBN-10: 9788131711460

Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, *137*, 184-196. https://doi.org/10.1016/j.jss.2017.11.066

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, (4), 345-361. https://doi.org/10.1109/tse.1978.231521

Ravi, K., & Suresh, Y. (2022). Effective ANN model based on neuro-evolution mechanism for realistic software estimates in the early phase of software development. *International Journal of Advanced Computer Science and Applications: IJACSA*, *13*(2). https://doi.org/10.14569/ijacsa.2022.0130223

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, *54*(8), 820-827. https://doi.org/10.1016/j.infsof.2011.12.008

Suresh Kumar, P., & Behera, H. S. (2020). Role of soft computing techniques in software effort estimation: An analytical study. In *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019* (pp. 807-831). Springer Singapore. https://doi.org/10.1007/978-981-13-9042-5_70

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014, September). Effort estimation in agile software development: A systematic literature review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering* (pp. 82-91). https://doi.org/10.1145/2639490.2639503

Velarde, H., Santiesteban, C., Garcia, A., & Casillas, J. (2016). Analyzing the effect of variables in the software development effort estimation. *IEEE Latin America Transactions*, *14*(8), 3797-3803. https://doi.org/10.1109/tla.2016.7786366

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, *54*(1), 41-59. https://doi.org/10.1016/j.infsof.2011.09.002