Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 24 Issue 1 Version 1.0 Year 2024
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Automated Lead Time Estimation for Anomaly Detection using a Machine Learning Algorithm

By Dr. Shivakumar Nagarajan, Ms. Divya T & Dr. Prasanna S

Abstract- The increasing complexity of modern business operations demands efficient and accurate lead time estimation to enhance decision-making processes. This study proposes a novel approach to automate lead time estimation using machine learning algorithms. Traditional lead time estimation methods often rely on manual calculations and historical averages, leading to inaccuracies and inefficiencies. In contrast, machine learning algorithms leverage historical data, contextual factors, and patterns to predict lead times dynamically. The automation of lead time estimation not only improves accuracy but also facilitates real-time decision-making. The system continuously learns from new data, adapting its predictions to changing business environments. A user-friendly interface is developed to allow easy input of relevant data and to visualize the lead time prediction.

Keywords: particle swam optimization, k-means clustering algorithm, neural network, analogy-based estimation.

GJCST-D Classification: LCC Code: HD30.215

Automated Lead Time Estimation for Anomaly Detection using a Machine Learning Algorithm

Dr. Shivakumar Nagarajan [α], Ms. Divya T [σ] & Dr. Prasanna S [ρ]

Abstract- The increasing complexity of modern business operations demands efficient and accurate lead time estimation to enhance decision-making processes. This study proposes a novel approach to automate lead time estimation using machine learning algorithms. Traditional lead time estimation methods often rely on manual calculations and historical averages, leading to inaccuracies and inefficiencies. In contrast, machine learning algorithms leverage historical data, contextual factors, and patterns to predict lead times dynamically. The automation of lead time estimation not only improves accuracy but also facilitates real-time decision-making. The system continuously learns from new data, adapting its predictions to changing business environments. A user-friendly interface is developed to allow easy input of relevant data and to visualize the lead time prediction. In this project design an automated time estimation is calculated for the usage of two algorithms and to get the accuracy for the maximum amount of iterations to be fitted by the PSO algorithm and then use the K-means clustering for the grouping the classes. From the PSO algorithm get the best features and then apply the Neural Network and Analogy Based Estimation for encrypt the data and then apply model to get the accuracy and time estimation from initialization to the end of the prediction and compare the two model for the accuracy and time comparison and also get the best features.

Keywords: particle swam optimization, k-means clustering algorithm, neural network, analogy-based estimation.

I. Introduction

In the fast-paced world of software engineering, accurate estimation of lead time is a critical factor that can significantly impact project planning, resource allocation, and overall project success. Lead time refers to the time taken from the inception of a software development task to its completion, encompassing various stages such as design, coding, testing, and deployment. Traditional methods of lead time estimation often suffer from subjectivity, imprecision, and a lack of adaptability to changing project dynamics. However, the rise of data-driven approaches and machine learning techniques has opened new possibilities for automating and improving lead time estimation.

The dynamic field of software engineering encompasses the methodical design, development, testing, and maintenance of software systems. It involves using a methodical approach to software development, which includes defining requirements, coming up with solutions, writing code, and subjecting the finished result to rigorous testing. Together, software developers use a range of methodologies, including as Agile and DevOps, to continuously build and enhance software while reacting to shifting input and requirements. The methodology includes selecting appropriate programming languages and design patterns as well as managing version control, assuring security, and maintaining documentation. A combination of technical know-how, teamwork, project management, and a commitment to ethical considerations are required to produce reliable and efficient software solutions that address real-world problems the software industry

The K-Means clustering algorithm is one such powerful technique that is gaining prominence in the software engineering domain. K-Means is an unsupervised machine learning approach that uses similarity to divide data into separate clusters. Its capacity to detect hidden patterns and structures in datasets makes it a great candidate for lead time estimating automation. K-Means can accurately forecast lead times for new activities by exploiting prior project data to find intrinsic correlations between development tasks and their related lead times.

This paper presents an in-depth exploration of the K-Means clustering algorithm and its application to automate lead time estimation in software engineering projects. We will delve into the underlying principles of K-Means, its strengths, limitations, and how it can be fine-tuned to suit the unique characteristics of software development processes. Furthermore, we will discuss the challenges associated with implementing K-Means for lead time estimation and propose strategies to overcome them.

The objective of this research is to equip software development teams, project managers, and stakeholders with a data-driven, efficient, and reliable lead time estimation tool that can facilitate better decision-making, resource allocation, and project planning. By harnessing the power of K-Means clustering, we aim to enhance the accuracy and efficiency of lead time estimation, ultimately leading to improved project management and successful software development endeavours.

Software lead time estimation, a critical aspect of project management, employs various techniques to predict the time needed for software development projects. Among these techniques, analogy-based

Author α: Department of Computer science and Engineering, Thiagarajar college of Engineering. e-mail: shiva@tce.edu

methods, neural networks, and Particle Swarm Optimization (PSO) play distinctive roles.

Analogy-based methods leverage historical data from similar projects to estimate lead times. By comparing known project outcomes and characteristics to those of the current project, these methods extrapolate estimates. While intuitive and easily accessible, their accuracy relies heavily on the quality and relevance of historical data.

Neural networks, a machine learning approach, demonstrate their strength in lead time estimation by learning complex patterns from historical data. These interconnected layers of nodes process input parameters to generate accurate predictions. Their ability to capture nonlinear relationships and adapt to diverse project variables makes them robust tools, particularly when adequate data is available for training. Particle Swarm Optimization, inspired by the social behaviour of birds flocking or fish schooling, employs optimization to estimate lead times. It simulates a population of particles traversing a solution space to find optimal estimates. Through iterative refinement, PSO converges towards accurate predictions by adjusting particle positions based on local and global information.

II. Related Works

In 2008, Kim[1] et al. published the first study on JIT-SDP, where they described a set of input features for JIT-SDP models. Other research looked into input features that could be useful for JIT-SDP, like the day of the week or the time of day a software change was produced, and input features to make it possible to identify software changes that are difficult to fix. According to Shihab[2]et al., the amount of lines of code contributed, the proportion of bug fixes to total file changes, the number of bug reports associated with a commit, and the developer experience are all reliable indications of software modifications that cause defects. Kamei[3] et al. carried out a substantial empirical investigation to look into a variety of factors extracted from commits and bug reports as input features for JIT-SDP models. They considered 14 features grouped into five dimensions of diffusion, size, purpose, history and experience, and showed such features to be good indicators of defect-inducing software changes for yielding high predictive performance on both open source and commercial projects. Many subsequent studies have adopted these features.

In 2022, Elvan Kula[4]; Eric Greuter[5]; Arie van Deursen[6]; Georgios Gousios[7] wrote late delivery of software projects and cost overruns have been common problems in the software industry for decades. Both problems are manifestations of deficiencies in effort estimation during project planning. With software projects being complex socio-technical systems, a large pool of factors can affect effort estimation and on-time delivery. To identify the most relevant factors and their interactions affecting schedule deviations in large-scale agile software development, we conducted a mixed-methods case study at ING: two rounds of surveys revealed a multitude of organizational, people, process, project and technical factors which were then quantified and statistically modelled using software repository data from 185 teams. We find that factors such as requirements refinement, task dependencies, organizational alignment and organizational politics are perceived to have the greatest impact on on-time delivery, whereas proxy measures such as project size, number of dependencies, historical delivery performance and team familiarity can help explain a large degree of schedule deviations. We also discover hierarchical interactions among factors: organizational factors are perceived to interact with people factors, which in turn impact technical factors. We compose our findings in the form of a conceptual framework representing influential factors and their relationships to on-time delivery. Our results can help practitioners identify and manage delay risks in agile settings, can inform the design of automated tools to predict schedule overruns and can contribute towards the development of a relational theory of software project management.

In 2021, Lei Zou[8], Zidong Wang[9], Qing-Long Han[10], Donghua Zhou[11] wrote the full information estimation (FIE) problem is addressed for discrete time-varying systems (TVSs) subject to the effects of a Round-Robin (RR) protocol. A shared communication network is adopted for data transmissions between sensor nodes and the state estimator. In order to avoid data collisions in signal transmission, only one sensor node could have access to the network and communicate with the state estimator per time instant. The so-called RR protocol, which is also known as the token ring protocol, is employed to orchestrate the access sequence of sensor nodes, under which the chosen sensor node communicating with the state estimator could be modelled by a periodic function. A novel recursive FIE scheme is developed by defining a modified cost function and using a so-called "backward-propagation-constraints." The modified cost function represents a special global estimation performance. The solution of the proposed FIE scheme is achieved by solving a minimization problem. Then, the recursive manner of such a solution is studied for the purpose of online applications. For the purpose of ensuring the estimation performance, sufficient conditions are obtained to derive the upper bound of the norm of the state estimation error (SEE). Finally, two illustrative examples are proposed to demonstrate the effectiveness of the developed estimation algorithm.

III. Proposed Methodology

a) *Particle Swam Optimization (PSO)*

PSO was developed by Eberhart[4] and Kennedy[5] in 1995. It is swarm intelligence meta-Heuristic inspired by the group behaviour of birds flocks or fish schools. It is simple, fast and easy to understand. Similarly, to Genetic Algorithm (GAs), it is a population-based method, that is, it represents the state of the algorithm by a population, which is iteratively modified until a termination criterion is satisfied. In PSO, the potential solution called particle which searches the whole space by previous best position (pbest) and best position of the swarm (gbest). The fitness function can be evaluated by the following equation.

$$f_n = \sum_{j=1}^{k} \sum_{i=1}^{n} || x_i^j - c_j ||^2 \quad (1)$$

Initially, position and velocity of the particles are expressed as follows

$$X_i = \text{LB} + \text{rand (UB-LB)} \quad (2)$$

$$V_i = \frac{LB + rand\ (UB - LB)}{\Delta t} \quad (3)$$

The velocity and positions values are updated during each iteration. Velocity update equation is given below

$$V_{i+1} = wV_i + c_1\,\text{rand}\frac{pbest_i - X_i}{\Delta t} + c_2\,\text{rand}\frac{pbest_i - X_i}{\Delta t} \quad (4)$$

Where, Xi is current position, Vi+1 is the velocity of next iteration, Vi is current velocity, Δt is the time interval, rand is a uniformly distributed random variable that can take any value between 0 and 1. Pbest_i is the location of the particle that experiences the best fitness, gbest_i is the location of the particle that experiences a global best fitness value, c1 and c2 are two positive acceleration constants responsible for degree of information, w represents inertia weight which is usually linearly decreasing during the iterations.

Updating position is the last step in each iteration, it is updated using velocity vector. Position update equation is given below

$$X_{i+1} = X_i + V_{i+1}\Delta t \quad (5)$$

Where, X_{i+1} is the next position, X_i is the current position, V_{i+1} is the next velocity, Δt is the time interval. Update the best fitness values at each generation, based on below equation

$$P_i(\text{t+1}) = \begin{cases} P_i(t) & f(X_i(\text{t+1})) \leq f(X_i(\text{t})) \\ X_i(t+1) & f(X_i(\text{t+1})) > f(X_i(\text{t})) \end{cases} \quad (6)$$

Where, f denotes the fitness function Eq. (1), Pi (t) stands for best fitness value and the coordination where the values are computed, Xi(t) is the current position, t denotes the generation step.

Update the velocity, position and fitness computations are repeated until the termination criteria is met.

b) *K-Means Clustering Algorithm(KMC)*

This K-means clustering is the best unsupervised learning algorithm. Using a certain number of clusters, it is simple and easy to classify a given data set. Initially, select the k centroids, then calculate the distance between each cluster centre and each object. Repeat the steps until no more changes are done. The objective function of k-means clustering is given as

$$f_n = \sum_{j=1}^{k} \sum_{i=1}^{n} || x_i^j - c_j ||^2 \quad (7)$$

Where, n is the number of data points, k is the number of clusters, $||x_i^j - c_j||^2$ is the distance between each cluster centre and each object. The steps of k-means clustering is shown below

1. Place K points into the space which denotes initial group centroids.
2. The group closer to the centroid is assigned with a cluster.
3. When all the objects are assigned to the groups, the positions of K centroids are reallocated.
4. Repeat Steps 2 and 3 until the centroids no longer move.

In this paper, combination of PSO and k-means clustering is used. PSO works efficiently for global search but its local search ability is poor. K-means clustering which produces local optimal solution due to initial partition but it does not work well for global search ability. Hence the proposed method overcome the limitation of K-means clustering, PSO offers global search methodology. Initially PSO is performed to find the location of cluster centroids. These locations of cluster centroids are given as input for the k-means clustering and provide optimal clustering solution. The steps for the proposed method is given below

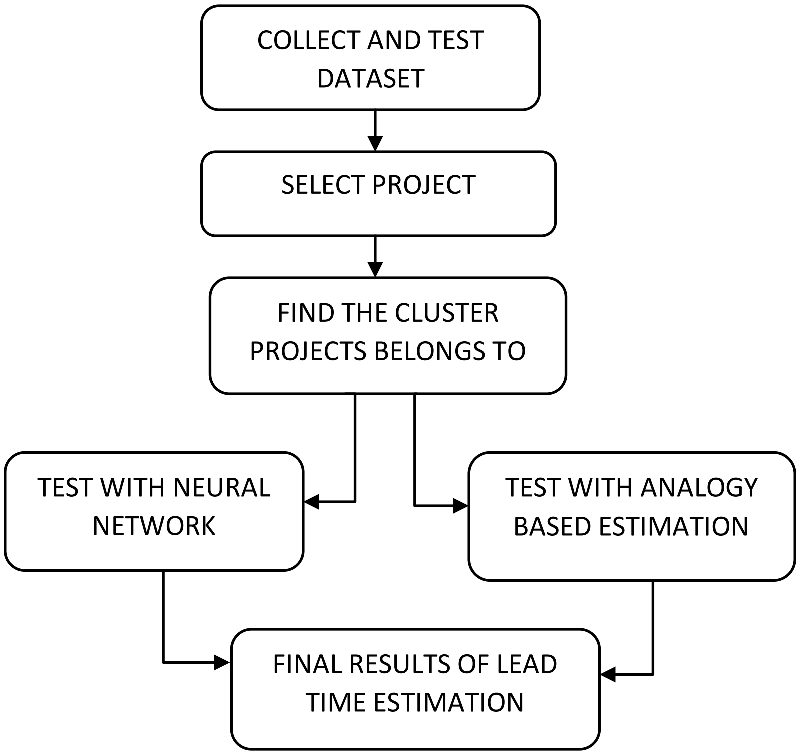
Step 1: Randomly generate particles and particles are grouped which creates population.

Step 2: Position and velocity of particles are initialized using Equation (2) and (3).

Step 3: Compute the fitness value using Eq. (6)

Step 4: Position, velocity, gbest and pbest values of the particles are updated using Equation (4) and (5).

Step 5: Repeat step (3) and (4) until one of following termination conditions is satisfied. 5(a). The maximum number of iterations is exceeded. 5(b). The average change in centroid vectors is less than a predefined value.


COLLECT AND TEST DATASET
SELECT PROJECT
FIND THE CLUSTER PROJECTS BELONGS TO
TEST WITH NEURAL NETWORK
TEST WITH ANALOGY BASED ESTIMATION
FINAL RESULTS OF LEAD TIME ESTIMATION

Step 6: Input the number of clusters which is to be generated.

Step 7: Compute the cluster centroids using the best positions of k of PSO.

Step 8: Compute the distance between the particles and centroids.

Step 9: Cluster centroids of k-means clustering can be recomputed using Eq. (7).

Step 10: Repeat the steps (8) and (9) until the centroids no longer move.

The combination of PSO and k-means provides better clustering result compared to result of each individual technique. PSO is used to find the optimal solution, the output of PSO is given as input to the k-means to obtain the clustering result. Check if the number of projects in clusters is greater than or equal to 15. For analogy method the number of projects present in a cluster should be less than fifteen. There are two reasons for choosing based on 15 projects. First, each project in the dataset usually has at least 10 features to train the neural network properly. Second, when using combination of PSO and K-means clustering the minimum requirement is 15. Anything less than 15 will become useless for training Neural Network. Testing stage consists of following steps. The Euclidean distance between a particular project and cluster centre is measured. The it is done for all the projects, the one with minimum is chosen. If it is marked in the training stage as ABE, then ABE is used to estimate the lead time. Otherwise NN is applied to predict the lead time. Repeat the steps until all test projects are applied to the hybrid model. For each test project the MRE is calculated and the final result is computed based on MMRE and PRED.

IV. Analogy based Method

The most common example of analogy-based project estimation is case-based reasoning, where identical projects from the lesson learned are identified and used for estimation. They can be used in combination with collective expert opinion and formal models, Software lead time estimation by using an analogy-based tech commonly involves the following steps.

1. Measuring the values of identified metrics of the software project for which estimation is being performed (target project).
2. Finding a similar project from the repository.
3. Using the estimated effort values of the selected projects to use as an initial estimate for the target project.
4. Comparison of metric's value for the target project and selected project.
5. Adjustment of lead time estimates in view of the comparison performed in the previous step.

V. Neural Network

Neural networks are a type of machine learning model inspired by the structure and function of the human brain. They are mathematical models composed of interconnected nodes, or neurons, that process information through a series of weighted connections. These connections allow neural networks to learn patterns and make predictions based on the input data. The artificial neural network is a powerful tool in the field of machine learning, as it can adapt and learn from data in a way that resembles human brain processing (Singh & Shrimali, 2019). It is important to understand the structure and functionality of neural networks in order to effectively utilize them for various applications in machine learning. Additionally, neural networks have layers, including an input layer, hidden layer(s), and an output layer. These layers help to organize and process the data, allowing for more accurate predictions or estimations. Furthermore, neural networks have been successfully applied in various fields including image recognition, natural language processing, and financial forecasting. By understanding the structure and functionality of neural networks, researchers and practitioners can effectively utilize these powerful machine learning models for various applications. In today's rapidly changing world, the significance of accurate weather forecasts cannot be overstated. In the context of neural networks, understanding their structure and functionality is crucial for effectively utilizing them in machine learning applications. Additionally, the widespread applications of neural networks in the control field, such as system identification and controller design, highlight their versatility and potential for solving complex problems. Overall, understanding the structure and functionality of neural networks is essential for harnessing their potential in machine learning applications and solving a variety of complex problems. In today's rapidly changing world, the significance of accurate weather forecasts cannot be overstated. Neural networks, with their ability to learn from data and make predictions, have wide-ranging applications in machine learning.

VI. Exprimental Design

a) Evaluation Criteria

The research community realizes that MMRE, Pred(x) can be influenced by the presence of outliers, therefore, we used the following performance metrics to assess and compare the accuracy.

b) Magnitude of Relative Error

The error ratio between actual and predicted effort for each project instance in the dataset is computed using MRE. To calculate the fitness function, first the MRE is calculated for every project i.

$$\text{MRE} = \left| \frac{Actual\ \ effort - Estimated\ \ Effort}{Actual\ \ Effort} \right| \tag{8}$$

c) Mean Magnitude of Relative Error

MMRE measure is used for assessing software estimation technique performance. The values of MRE is calculated for each software project instance. MMRE computes the average over N number of project instances in the data-set.

$$\text{MMRE} = \frac{1}{N}\sum_{i=1}^{N} MRE \tag{9}$$

d) Percentage of Prediction

PRED(x) represents the percentage of MRE that is less than or equal to the value x/100 among all projects.

$$\text{PRED(X)} = \frac{100}{N}\begin{cases} 1 & if\ MRE \le \frac{X}{100} \\ 0 & otherwise \end{cases} \tag{10}$$

where A is the number of projects with MRE less than or equal to X. N is the number of projects in test set. Usually the ideal value of X is 25% in software development effort estimation and compared with various methods.

Table No. 1: Time Estimation Table

Algorithm	Time Estimation
Neural Network	3.73
ABE algorithm	0.78

Table No. 2: Accuracy Table

Algorithm	Accuracy
Neural Network	95.79
ABE algorithm	97.68

Table No. 3: Loss comparison Table

Algorithm	Loss
Neural Network	4.21
ABE algorithm	2.32

e) Lead Time Estimation

Lead time estimation is the process of predicting the amount of time it will take to complete a specific task, project, or deliver a product or service. It is a critical aspect of project management and supply chain management and is used in various industries, including manufacturing, logistics, and software development. Accurate lead time estimation helps organizations plan and allocate resources effectively, manage customer expectations, and avoid delays and disruptions in their operations.

f) Leave-One-Out

In this method a project is selected from the dataset as the target project. The ABE method is applied to estimate the effort of the target project by using all the remaining projects in dataset. Then the target project combined with the dataset and another project is selected as a target project. This procedure repeated until all the projects in dataset are estimated.

Finally, the MRE and MMRE values are computed for each project

g) Dataset Description

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections.

The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset.

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well- defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Each connection is labelled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Attacks fall into four main categories:

- *DOS:* denial-of-service, e.g. sync flood;
- *R2L:* unauthorized access from a remote machine, e.g. guessing password;
- *U2R:* unauthorized access to local super user (root) privileges, e.g., various "buffer overflow" attacks;
- *Probing:* surveillance and other probing, e.g., port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the "signature" of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

h) Numerical Results

In this paper, for training stage two clusters are considered. When Applying the PSO and k-means clustering to the intrusion detector learning dataset, the best situation is with two clusters. When it is more than two some clusters with one or two projects appeared not perfect for estimation. On the other hand, splitting projects in clusters size less than two decreases the performance of NN training.

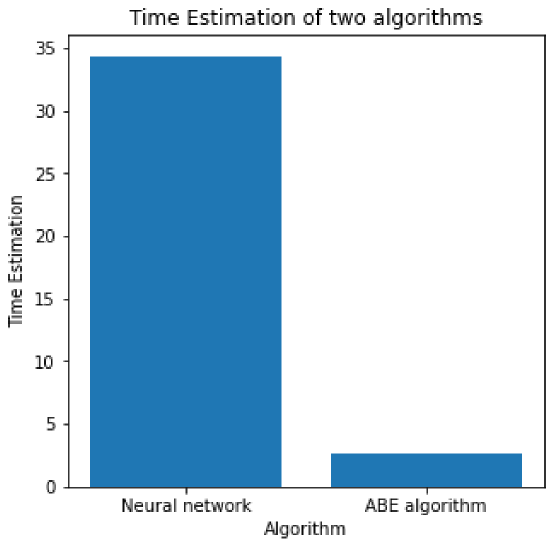


Fig. 1: Time Estimation of NN and ABE algorithm

Results on all Data. Initially, the selected features from PSO is 19 are grouped into five clusters by k-Means clustering algorithm. Based on the number of projects in a cluster either ABE or NN is identified displayed the clusters formed from PSO and K-Means clustering algorithm. The identified method for cluster. Identifying a cluster for a project is done according to the maximum degree of membership function. First clusters are NN and second cluster are marked as ABE.

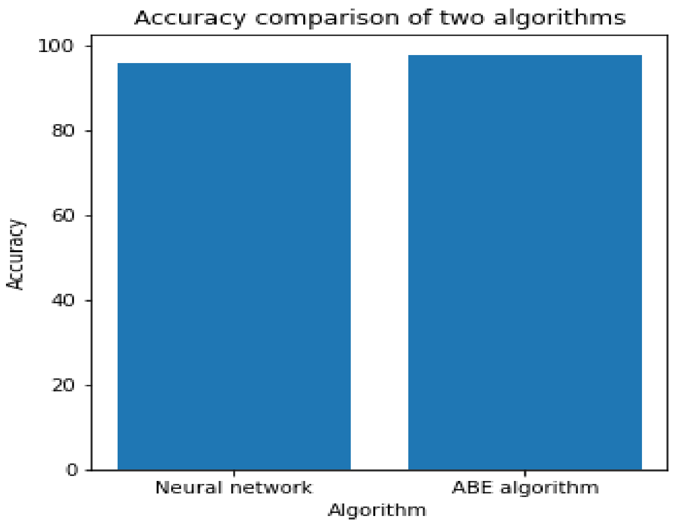


Fig. 2: Accuracy comparison of NN and ABE algorithm

VII. Result

Finally, from the Fig-1 and Fig-2, by comparing the Neural Network and Analogy Based Estimation, the accuracy of NN: *95.79%* and time taken for Neural Network: *34.34 seconds.* Accuracy of ABE: *97.68%* and time taken by Analogue Based Estimation: *2.66921 seconds.*

The performance of *ABE is better than NN*, with less time taken and improved accuracy.

VIII. Conclusion

This paper proposes a hybrid scheme for clustering based on PSO and K-means algorithm. K-means algorithm paves way to predetermine the number of clusters by the end users which results in an improper clustering. Hence PSO algorithm is integrated with K-means algorithm to improve the better clustering result. This result is more efficient when applied to neural networks and ABE.

IX. Future Work

To perform the model compiling for the minimum accuracy. Time taken to done for the pre-processing the dataset is very low, when compared with the other techniques. Our future work is to implement other techniques for effort estimation for this method.

References Références Referencias

1. A. Sinha, P. Malo, and K. Deb, ''A review on bilevel optimization: From classical to evolutionary approaches and applications,'' IEEE Trans. Evol. Comput., vol. 22, no. 2, pp. 276–295, Apr. 2018.
2. Ahmad Setiadi; Wahyutama Fitri Hidayat; Ahmad Sinnun; AdeSetiawan; Muhammad Faisal, "Analyse the Datasets of Software Effort Estimation With Particle Swarm Optimization".
3. Alvin Jian Jia Tan; Chun Yong Chong; Aldeida Aleti, "Closing the Loop for Software Remodularisation - REARRANGE: An Effort Estimation Approach for Software Clustering-based Remodularisation", 12 July 2023.
4. B. Colson, P. Marcotte, and G. Savard, ''An overview of bilevel optimization,'' Ann. Oper. Res., vol. 153, no. 1, pp. 235–256, Jun. 2007.
5. D. Vesset, C. Gopal, C. W. Olofson, D. Schubmehl, S. Bond, and M. Fleming, ''Worldwide big data and analytics software forecast", 2018.
6. D. Aussel and C. S. Lalitha, ''Generalized Nash equilibrium problems,'' in Bilevel Programming and MPEC. Singapore: Springer, 2018.
7. Elvan Kula; Eric Greuter; Arie van Deursen; Georgios Gousios, "Factors Affecting On-Time Delivery in Large-Scale Agile Software Development". 2021.
8. F. Vandenbergh and A. Engelbrecht, ''A study of particle swarm optimization particle trajectories,'' Inf. Sci., vol. 176, no. 8, pp. 937–971, Apr. 2006.
9. G. M. Wang, Z. P. Wan, and X. J. Wan, ''Bibliography on bilevel programming,'' (in Chinese), Adv. Math., vol. 36, no. 5, pp. 513–529, 2007.
10. Gregor Molan; Gregor Dolinar; Jovan Bojkovski; Radu Prodan; Andrea Borghesi, "Model for Quantitative Estimation of Functionality Influence on the Final Value of a Software Product" 2021.
11. H. Liu, L. Gao, and Q. Pan, ''A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem,'' Expert Syst. Appl., vol. 38, no. 4, pp. 4348–4360, Apr-2011.
12. H. L. T. K. Nhung, V. Van Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, ''Parametric software effort estimation based on optimizing correction factors and multiple linear regression,'' 2022.
13. H. Mühlenbein and G. Paaß, ''From recombination of genes to the estimation of distributions I. Binary parameters,'' in Proc. 4th Int. Conf. Parallel Problem Solving Nature, Berlin, Germany, Sep. 1996, pp. 178–187.
14. J. Agor and O. Y. Özaltın, ''Feature selection for classification models via bilevel optimization,'' Comput. Oper. Res., vol. 106, pp. 156–168, Jun. 2019.
15. J. Wang, ''A novel discrete particle swarm optimization based on estimation of distribution,'' in Proc. ICIC LNAI, vol. 4682, 2007, pp. 791–802
16. Koszty´ an, Z. T., Jakab, R., Nov´ ak, G., & Hegedus, ˝c "Survive IT! survival analysis of IT project planning approaches. Operations Research Perspectives".
17. Mona Najafi Sarpiri; Mohammadreza Soltan Aghaei; Taghi Javdani Gandomani," Towards Better Software Development Effort Estimation with Analogy-based Approach and Nature-based Algorithms". 2021.
18. M. Iqbal and M. Montes de Oca, ''An estimation of distribution particle swarm optimization algorithm,'' in Proc. ANTS, in Lecture Notes in Computer Science, vol. 4150, 2006, pp. 72–83.
19. P. Phannachitta, "On an optimal analogy-based software effort estimation", Sep 2020.
20. P.-L. Poirion, S. Toubaline, C. D'Ambrosio, and L. Liberti, ''Algorithms and applications for a class of bilevel MILPs,'' Discrete Appl. Math., vol. 272, pp. 75–89, Jan. 2020.
21. R. V. Kulkarni and G. K. Venayagamoorthy, ''An estimation of distribution improved particle swarm optimization algorithm,'' in Proc. 3rd Int. Conf. Intell. Sensors, Sensor Netw. Inf., Melbourne, VIC, Australia, 2007, pp. 539–544.

22. Shivakumar Nagarajan; Balaji Narayanan, "K-Means Clustering algorithm to compute Software Effort Esatimation", October-2011.
23. T. Zhang and X. Li, ''The backpropagation artificial neural network based on elite particle swam optimization algorithm for stochastic linear bilevel programming problem,'' Math. Problems Eng., vol. 2018, pp. 1–9, Oct. 2018.
24. Taghi Javdani Gandomani; Maedeh Dashti; Mina Zaiei Nafchi," Hybrid Genetic-Environmental Adaptation Algorithm to Improve Parameters of COCOMO for Software Cost Estimation, 14 March 2022.
25. U. Farooq, M. Ahmed, S. Hussain, F. Hussain, A. Naseem, and K. Aslam, ''Block chain-based software process improvement (BBSPI): An approach for SMEs to perform process improvement,'' 2022.
26. V. V. Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, ''A new approach to calibrating functional complexity weight in software development effort estimation,'' Jan. 2022.
27. X. Deng, ''Complexity issues in bilevel linear programming,'' in Multilevel Optimization: Algorithms and Applications, A. Migdalas, P. M. Pardalos, and P. Varbrand, Eds. Norwell, MA, USA: Kluwer, 1998, pp. 149–164.
28. Y. Jiang, X. Li, C. Huang, and X. Wu, ''Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem,'' Appl. Math. Comput., vol. 219, no. 9, pp. 4332–4339, Jan. 2013.