Improved Data Segmentation Architecture for Early Size Estimation using Machine Learning

Manisha[1]*, Rahul Rishi[2]
CSE, UIET, MDU, India

Sonia Sharma[3]
ECE, UIET, MDU, India

***Abstract*—Software size estimation plays an important role in project management. According to the report given by Standish Chaos, about 65% of software projects are beyond companies budget or overdue; which could have been saved if an early estimation was imposed. Though the software size can't be measured directly, but it is related to effort and hence a low effort will lead to low size. The calculation of effort depends upon how the data is organized or segmented. This research paper focuses on the improvement of data segmentation in order to reduce the effort and parallel the size. In order to improve the segmentation architecture, the project data is divided based on the similarity indexes which the projects have in between their attributes. Three similarity measures were used namely Cosine Similarity (CS), Soft Cosine Similarity (SC) and a hybrid similarity index which combines CS and SC. Based on these similarity indexes, the project data is divided into groups by K-means algorithm. In order to estimate the size, the co-relation between the formed groups are calculated. To calculate the correlation, Mean Square Error (MSE), Square Error (SE), and Standard Deviation (STD) is calculated and the normalized parameters are used to evaluate the software size.**

***Keywords*—Cosine similarity; hybrid similarity; machine learning; size estimation and soft cosine similarity**

I. Introduction

Software size estimation has always been a favorite area for the researchers as the wrong estimation of effort and size will lead to high computation risk and project failure. If the company does not earn from a project, even if the project is complete, it is termed as failure. Therefore, to overcome this problem, an early stage prediction model is designed based of the past project experience [1]. Underestimation and overestimation both leads to high project risk completion. In case, if size of the software project is estimated at the earliest stage, then, project handling team can provide better plan. Conventional software development mainly goes through four levels namely requirements elicitation, design, coding and testing [2]. After designing early size estimation model, most of the systems have better response. If the project estimation is terminated just after first level, then it affects directly the project management. Therefore, early size estimation of project is a crucial task not only for conventional software development bust also for agile development, as it makes the projects more manageable. Software size metrics play an important role in the success of this task [3]. In the conventional approach, Lines of Code (LOC), evaluation has been a way to evaluate the size of the projects [4]. Function Point (FP) sizing also evolved as a method to acquire the details of the project but being a object oriented concept again, the usage of FP will also require code oriented metrics [5], [6]. Early size estimation requires the attributes which does not involve post coding methods. Any estimation method, requires a learning model and as it is described earlier, code oriented metrics will be only attained once the project is complete. In order to train the system, the Ground Truth (GT) of the data is the foremost requirement. GT refers to the class label of the data. As for example, suppose a project had 5000 LOC ,10 team members,1 team lead and 50 hours of development and the effort from the team in order to develop the project was "high", then the GT for Project Attribute(PA)→{5000 ,10,1,50} is high. Unfortunately, these type of metrics can be only evaluated after the development and does not suit early size estimation. In the early age of development of early size estimation [15-20], authors introduced learning methods using machine learning and its extension. [7] used machine learning oriented neural networks to train the system based on the feedback from the engagements of the company but the generation of GT through machine learning failed here.

A. Machine Learning

The machine learning architecture needs a training data algorithm with its ground truth value. The material methods which involves statistics are called machine learning. Common machine learning algorithms are as follows.

1) K-means.
2) Regression.
3) Linear Propagation Model.

This research article uses K-means clustering algorithm with the enhancement of base architecture. The k-means clustering algorithm selects a random centroid out of the provided set of attributes. If there are three attributes, then the centroid of the clusters will also have three attributes. The adjustment in the centroid takes place when the co-relation between the attributes and centroids varies and goes to the adjustment into another cluster. In order to calculate the distance between centroid and attribute values, Euclidian distance is calculated. The data is always divided into 'k' number of clusters and in order to decide 'k', there are multiple ways of advisory. This research work does not focuses on the calculation of 'k' as there are two sections for the entire data namely 'high size' and 'low size' which is further subdivided into two sub-segments namely 'moderate' and 'high' . The size before the development can't be directly evaluated and hence this research article refers to the statement that high effort will lead to high size and low co-relation will lead to high effort.

*Corresponding Author. mvmanishavatsa@gmail.com

$$Effort \propto cost\,||size \quad (1)$$

The rest of the paper is organized as follows: Section 2 represents the related work in the field of size estimation. Section 3 presents the proposed work including technique used and process of work. The result and discussions are presented in Section 4. Conclusion is presented in Section 5.

II. Related Work

Ahn et al. (2003) have presented a metric analyses based methodology tool to estimate effort at early stage of software project. Using this model, a project manager can analyzed, software system like as analyzed by FP analysis. In addition, relation between UML point and effort has also been observed [8]. Baskeles et al. (2007) have presented ML based software estimation model to resolve the problems related to cost and schedule extension. The test has been conducted on the collected data obtained from software industry in Turkey. Principle Component Analysis (PCA) has been applied as data dimension reduction approach. ML technique has been trained by 50 and 40 projects, and test later using 10 and 20 projects respectively. From the test results it has been concluded that the parametric models are not sufficient for software effort estimation [9]. Nassif et al. (2013) have presented a new logarithmic based linear regression model, which works on the basis of use case point model (UCP). This approach is used case diagram to determine software effort. To regulate the productivity factor of regression model Fuzzy logic is used as Fuzzy Interface System (FIS). Based on software size and project group productivity, neural network is design and used to estimate software effort. At last comparison between ANN and log-linear regression based model has been presented. The results show that NN outperformed compared to regression model while estimating small project, but for large scale projects regression algorithm performed better [10]. Satapathy et al. (2016) have used Random Forest (RF) as ML approach to enhance the performance of effort prediction model. UCP approach has been used for estimating project effort, the parameters of which has been improved using RF approach. Among various ML techniques like NN, log-linear regression scheme, stochastic approach, and radial function method, RF surpass among all [11]. Sharma et al. (2017) have presented a review on various ML techniques used for software effort and size estimation. LOC and FP are the two main metrics used for effort estimation. There are various validation methods that can be considered in the expansion of research to confirm the consequences of software effort assessments. The main validation methods are Cross Assessments, Jacknife method and Iterative method. In addition, research trends have shown that assessment methods need to be explored and improved. In addition, other ML approaches, such as size metrics and regression trees, can also be utilized using real-life data sets [12]. Spikol et al. (2018) have presented ML based effort estimation model using International Software Benchmarking Standards Group (ISBSG) dataset. For cross validation, three ML techniques SVM, ANN, and generalized Linear Model has been used [13]. Lavazza et al. (2019) have empirically studied the COSMIC early size estimation scheme using historical dataset. Average Functional Process along with the Equal Size Bands methods have been used for the prediction of effort estimation. Results shows that sometimes functional process provides better results that can be accepted but sometimes the error is so large, which is not to be acceptable for better performance of any project. On the other hand, band method provides better results [14]. Nassif et al. (2019) have compared three FIS systems namely, Mamdani, Sugeno with defined output, and Sugeno with linear output. The performance of the system has been analyzed based on standard sized accuracy, error, effect size and statistical tests. The designed model is trained using ISBSG dataset, and results show that FIS are very sensitive to outliers. Authors also concluded that when fuzzy logic is used in combination to regression model, the system outperforms [15]. Silhavy et al. (2021) had proposed a method for size estimation based on the use case and the number of actors. In the process, stepwise regression was performed that minimized the number of errors adjoining the size estimation of the software system. This is also added that the predictions were independent of the use case points to avoid inaccurate estimations [16]. Daud and Malik (2021) had aimed at quantitative analysis of the effect of precise software size estimations when the project transits to the designing phase. To achieve this Analysis to Design Adjustment Factors were introduced in the proposed work that demonstrated improved accuracy of the proposed model [17]. Suresh (2022) had integrated Artificial Neural Netowrk (ANN) to generate precise estimations in the initial stages of development cycle. It was observed that the work offered optimal and automated predictions based on training and classification using ANN architecture. The work exhibited better and more accurate software effort estimations in comparison to the existing works [18].

A. Problem Formulation

It becomes easy to the train a system when the GT of the data is available but, if there is no GT, the system remains uncertain in order to make the system understand what the data represents for. In case of post size estimation, the GT is easy as the developers speaks themselves and the outcome defines the GT but, in case of early size estimation, GT is not present. This articles focuses on the development of automatic GT for improved segmented section in order to quantify low size and high size based on the co-relation between the groups as shown in Fig. 1.

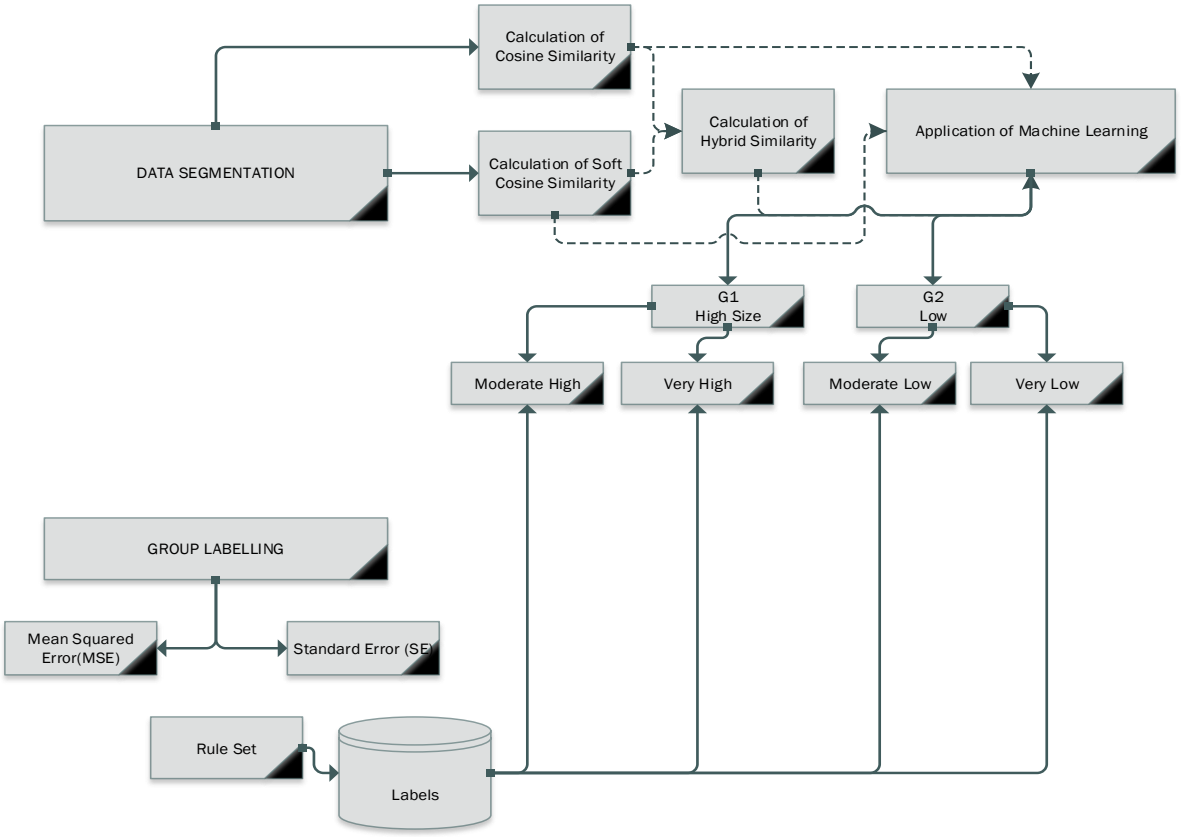

Fig. 1. The Proposed Work Model.

III. Proposed Work

The proposed work is divided into two segments, namely, segment generation and group labelling. The architecture of the proposed work is illustrated using Fig. 2. The data segmentation involves the application of Cosine Similarity (CS), Soft Cosine Similarity (SC) and hybrid similarity which is a summed value of CS and SC. The formation of groups utilizing K-means is also a part of this section. The K-means algorithms divides the entire attribute set in two groups and their labelling is done by the group labelling section of the proposed algorithm.

A. Calculate Similarity Indices

Here data, segmentation is used to obtained ground truth value. Here, data is fragmented based on their size into two parts high fragments and low fragments after finding the centroid for each row in the dataset. The available dataset contains irrelevant information, therefore need to be rectify before used for analysis process. Hence, need to be isolated from the available data for any project for size estimation. Therefore clustering using K-means play an important role for cluster formation. Before applying clustering, we need to find out similarity indices, which are obtained using three similarity measures (i) Cosine Similarity (CS), (ii) Soft Cosine (SC) measure, (iii) Hybrid similarity measure.

1) Cosine similarity: It is similarity measurement approach used to find out the similarity metric between two entities based on their size using the concept of the cosine angle between entities in an n-dimensional space. Exactly, to calculate the cosine similarity between two entities in an n-dimensional space, the given formula is used:

$$Cos\ (\theta) = \frac{\vec{M}.\vec{N}}{\|\vec{M}\|\|\vec{N}\|} \quad (2)$$

Hare, 'M' and 'N' are two entities may be array or vectors; cosine equation gives the angle between them using the written equation:

$$\vec{M}.\vec{N} = \sum_{k=1}^{n} M_k \times N_k \quad (3)$$

$$= M_1 \times N_1 + M_2 \times N_2 + \cdots + M_n \times N_n$$

$$\|\vec{M}\| = \sqrt{{M_1}^2 + {M_2}^2 + \cdots + {M_n}^2}$$

$$\|\vec{N}\| = \sqrt{{N_1}^2 + {N_2}^2 + \cdots + {N_n}^2}$$

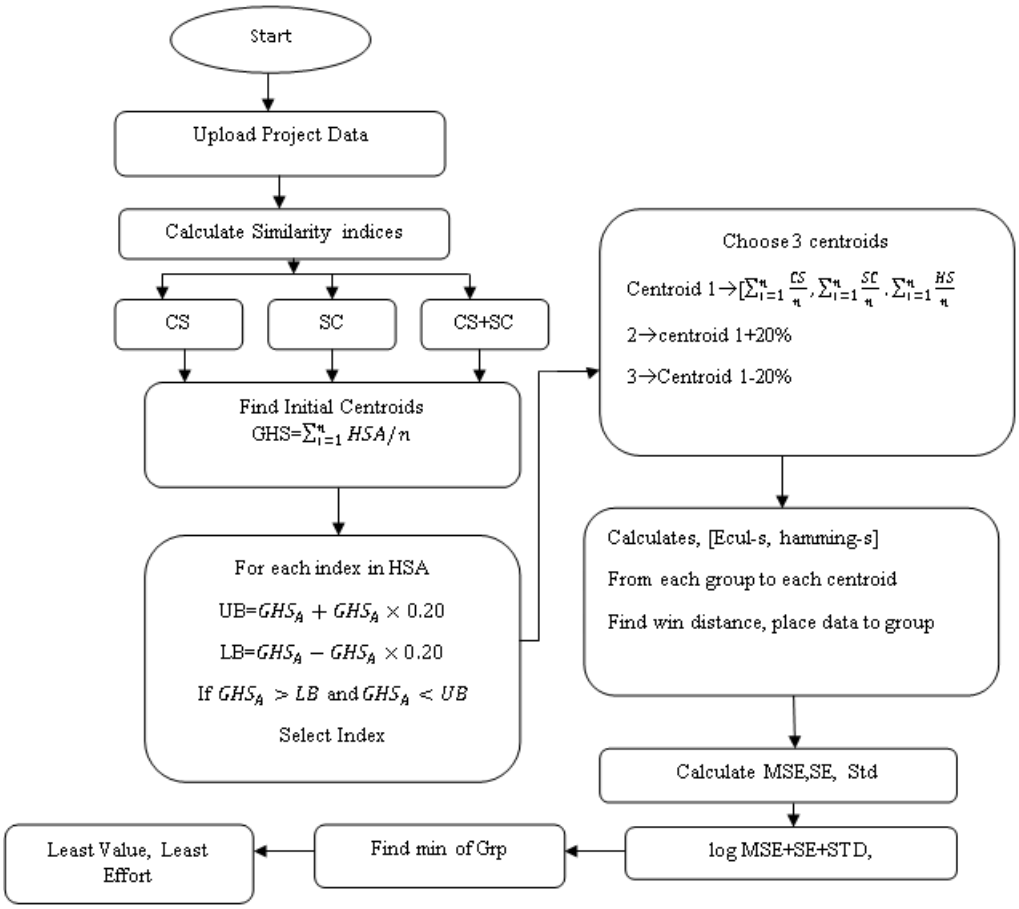

Fig. 2. Proposed Work.

We consider an example to calculate the similarity between two sample data such as M and N, where

M = [1, 1, 0]

N = [1, 0, 1]

$$\vec{M}.\vec{N} = 1 \times 1 + 1 \times 0 + 0 \times 1 = 1 + 0 + 0 = 1 \quad (4)$$

$$\|\vec{M}\| = \sqrt{1^2 + 1^2 + 0^2} = \sqrt{2} \quad (5)$$

$$\|\vec{N}\| = \sqrt{1^2 + 0^2 + 1^2} = \sqrt{2} \quad (6)$$

So, the value of cosine similarity is;

$$Cos\ (\theta) = \frac{1}{\sqrt{2}\times\sqrt{2}} = \frac{1}{2} = 0.5 \quad (7)$$

$$\theta = cos^{-1}(0.5) = 60^{\circ} \quad (8)$$

Cosine similarity is used to measure the closeness between the data elements. It is a measurement of similarity between two non-zero arrays or vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any angle in the interval (0, π) radians. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. These bounds apply for any number of dimensions, and the cosine similarity is most commonly used in high-dimensional positive spaces. Some important points related to the cosine similarity is written as:

1) Always applicable for two non-zero arrays or vectors.

2) The cosine similarity considers the Vector Space Model (VSM) features as independent or completely different.

3) The time complexity of cosine similarity is non quadratic.

4) Cosine similarity use the concept of Term Frequency–Inverse Document Frequency (TF-IDF) to calculate the similarity between documents

5) Cosine Similarity uses the Euclidean Distance to measure the closeness between two entities.

The designed algorithm for CS is written.

Algorithm: Cosine Similarity

Input: PD →Uploaded project data to calculate similarity
Output: SIM_{COS}→ Cosine Similarity for PD
SIM_{COS}= [()] // to store cosine similarities values
Sim-count = 0
For m = 1 → Length (PD)
Current-PD = PD (m)
For n = m+1 →Length (PD)
Reference-PD = PD (n)
Calculate the cos similarity using given equation

$$\overrightarrow{\text{Current} - \text{PD}} . \overrightarrow{\text{Reference} - \text{PD}} = \sum_{k=1}^{n} \text{Current} - \text{PD}_k \times \text{Reference} - \text{PD}_k$$

$$\left\|\overrightarrow{\text{Current} - \text{PD}}\right\| = \sqrt{\text{Current} - \text{PD}_1{}^2 + \text{Current} - \text{PD}_2{}^2 + \cdots + \text{Current} - \text{PD}_n{}^2}$$

$$\left\|\overrightarrow{\text{Reference} - \text{PD}}\right\| = \sqrt{\text{Reference} - \text{PD}_1{}^2 + \text{Reference} - \text{PD}_2{}^2 + \cdots + \text{Reference} - \text{PD}_n{}^2}$$

$$Cos\ (\theta) = \frac{\overrightarrow{\text{Current} - \text{PD}} . \overrightarrow{\text{Reference} - \text{PD}}}{\left\|\overrightarrow{\text{Current} - \text{PD}}\right\| \left\|\overrightarrow{\text{Reference} - \text{PD}}\right\|}$$

SIM_{COS}[Sim-count]=Cos(**θ**)
Increment in index of array, Sim-count = Sim-count + 1
End – For
End – For
Return: SIM_{COS}as an output in terms of Cosine Similarity for PD
End – Algorithm

2) Soft cosine similarity: Soft cosine or the soft similarity approach is used to calculate the similarity between two entities similar to the cosine similarity approach. Cosine similarity has the VSM (Vector Space Model)features as completely different or independent whereas the soft cosine measures considers the VSM similarity features that assists in the generalization of cosine and soft cosine with similarity idea. For the computation of soft cosine, S as matrix is utilized for the indication of similarity among features. It could be measured by Levenshtein distance and Wordnet similarity. Soft cosine similarity finds the most common between two document sets.

$$Soft\ Cosine\ = Cos\left(\frac{n}{i=1}|X \cap Y|\right) \quad (9)$$

Where, X and Y are the two entities in n dimensional space and the some important points related to the soft cosine similarity is written as:

1) A soft cosine similarity between two vectors considers similarities between pairs of features.

2) The soft cosine similarity measures the similarity of features in Vector Space Model (VSM) as dependent.

3) The time complexity of Soft Cosine Similarity is quadratic, which makes it perfectly applicable to real-world tasks but it can be reduced to sub quadratic.

4) Soft Cosine similarity uses the concept of n-grams which is a contiguous sequence of n items.

5) Soft Cosine Similarity uses the Levenshtein Distance.

The algorithm for soft cosine similarity index is written below.

Algorithm: Soft Cosine Similarity

Input:PD → Uploaded project data to calculate similarity
Output: $SIM_{SOFTCOS}$→Soft Cosine Similarity for PD
$SIM_{SOFTCOS}$ = [()] // to store soft cosine similarities values
Sim-count = 0
For m = 1 → Length (PD)
Current-PD = PD (m)
For n = m+1 →Length (PD)
Reference-PD = PD (n)

$$\boldsymbol{SIM_{SOFTCOS}} = \boldsymbol{Cos}\left(\frac{\boldsymbol{n}}{\boldsymbol{i=1}}|\text{Current} - \text{PD} \cap \text{Reference} - \text{PD}|\right)$$

$SIM_{SOFTCOS}$[Sim-count]=$SIM_{SOFTCOS}$
Increment in index of array, Sim-count = Sim-count + 1
End – For
End – For
Return:$SIM_{SOFTCOS}$ as an output in terms of Soft Cosine Similarity for PD
End – Algorithm

B. Hybrid Similarity

When CS and SC measures are used in combination, then the measure form is known as hybrid similarity indices. By comparing average of each row with remaining rows, an average value is obtained. Suppose we have 100 rows in the project, therefore, to find similarity either applying CS, SC, or hybrid similarity average value obtained for each row (1st row) is compared to the remaining 99 rows. This process will provide an average value, which is used to find initial centroid of data that is given by equation (10).

$$GHS_A = \sum_{i=1}^{n} HS/n \quad (10)$$

Where, GHS_Arepresents Gross Hybrid Similarity, HS indicates hybrid similarity value.

For each index in HS, determined upper bound (UB), and Lower Bound (LB) values by multiplying 20 % of data to the obtained GHS average, adding and subtracting the obtained value to GHS. The added value is indicated by UB, and, subtracted value is represented by LB. Both UB and LB values are represented by equation (11), and equation (12), respectively.

$$UB=GHS_A + GHS_A \times 0.20 \quad (11)$$

$$LB=GHS_A - GHS_A \times 0.20 \quad (12)$$

If the obtained average values of is greater than LB or less than UB ($GHS_A > LB$ and$GHS_A < UB$), then index value is selected.

The algorithm for hybrid similarity measure is written.

Algorithm: Hybrid Similarity (Soft Cosine Similarity)

Input:PD → Uploaded project data to calculate similarity
Output: SIM_H→HybridSimilarity for PD
SIM_H = [()] // to store cosine similarities values
Sim-count = 0
For m = 1 → Length (PD)
Current-PD = PD (m)
For n = m+1 →Length (PD)
Reference-PD = PD (n)
Calculate the cos similarity using given equation

$$\overrightarrow{\text{Current} - \text{PD}}.\overrightarrow{\text{Reference} - \text{PD}} = \sum_{k=1}^{n} \text{Current} - \text{PD}_k \times \text{Reference} - \text{PD}_k$$

$$\left\|\overrightarrow{\text{Current} - \text{PD}}\right\| = \sqrt{\text{Current} - \text{PD}_1{}^2 + \text{Current} - \text{PD}_2{}^2 + \cdots + \text{Current} - \text{PD}_n{}^2}$$

$$\left\|\overrightarrow{\text{Reference} - \text{PD}}\right\| = \sqrt{\text{Reference} - \text{PD}_1{}^2 + \text{Reference} - \text{PD}_2{}^2 + \cdots + \text{Reference} - \text{PD}_n{}^2}$$

$$Cos\,(\theta) = \frac{\overrightarrow{\text{Current} - \text{PD}}.\overrightarrow{\text{Reference} - \text{PD}}}{\left\|\overrightarrow{\text{Current} - \text{PD}}\right\|\left\|\overrightarrow{\text{Reference} - \text{PD}}\right\|}$$

$$SIM_{SOFTCOS} = Cos\left(\frac{n}{i=1}|\text{Current} - \text{PD} \cap \text{Reference} - \text{PD}|\right)$$

SIM_H[Sim-count]={Cos(**θ**) + $SIM_{SOFTCOS}$}
Increment in index of array, Sim-count = Sim-count + 1
End – For
End – For
Return:SIM_H as an output in terms of HybridSimilarity for PD
End – Algorithm

Now, apply K means as ML approach for centroid selection.

ML is a branch of artificial intelligence (AI). In more detail, it is a method of data analysis that allows a machine / robot or any analysis system to learn independently by solving a number of similar problems. Machine learning aims to develop computer programs that can access information and use it to learn. The training process begins with a specific set of information, such as examples, direct experience, or instruction, to look for examples in the data and make better decisions in the future based on the examples we provide. The main purpose is to ensure that computers automatically learn without human intervention and outside help, and to regulate actions accordingly.

ML algorithms are often classified as supervised, semi-supervised or un-supervised. In general, these algorithms can apply past data to new data, using the noted examples to predict future events. Starting with the analysis of a known training database, the learning algorithm creates a resultant function to predict the output values. In doing so, the learning algorithm can also compare the result with specific "correct" samples of the latest data and find errors to change the development model accordingly. In semi-supervised ML approach small amount of data (labeled) is merged to the huge dataset (unlabeled). It lies between supervised (labeled data), and un-supervised learning (un-labeled data). In contrast, unsupervised ML algorithms are used when the information used for training is not classified or marked in any way. The system examines the data and can draw conclusions from the datasets.

Improving machine learning algorithms is used to interact with the environment - trial and error search is the most important learning characteristic of this method, which allows programs to automatically detect ideal behavior in a specific context to maximize its performance.

K-means is a clustering approach used to partitioned N number of data into k number of groups known an clusters. The process is followed as:

- Initially, select K centroid, which is the center point of each fragmented data.
- Allocate data point to the closest centroid.

- Reallocate centroid value as the calculated average value for each group.
- Reallocate the data points to the adjoining centroid.
- Repeat until the data points remain in the same group.

After dividing data into different groups on the bases of boundary conditions as mentioned above, three centroids have been selected for the divided data. The process of centroid selection is as follows:

Consider an initial set of k means (centroids) $(C_1, C_2, \dots \dots \dots C_n)$ in clusters $(S_1, S_2, \quad S_n)$. At the first stage, the centroids of the clusters are selected randomly or according to a certain rule (for example, choose the centroids that maximize the initial distances between the clusters). We refer observations to those clusters whose mean (centroid) is closest to them. Each observation belongs to only one cluster, even if it can be assigned to two or more clusters. Then the centroid of each i-th cluster is recalculated according to the equation (13).

$$C_j = \frac{1}{S_j}\sum_{x(j)\epsilon S_i} X(j) \quad (13)$$

Thus, the k-means algorithm consists in recalculating the centroid at each step for each cluster obtained in the previous step. The algorithm stops when the value C_j does not change the maximum and minimum value. The three centroids are formed and named as centroid 1, centroid 2, and centroid 3, each represented by equation (14), equation (15), and equation (16) respectively.

$$\text{Centroid 1}=[\sum_{i=1}^{n}\frac{CS}{n}, \sum_{i=1}^{n}\frac{SC}{n}. \sum_{i=1}^{n}\frac{HS}{n} \quad (14)$$

$$\text{Centroid 2}=\text{centroid 1}+20\% \quad (15)$$

$$\text{Centroid 3}\rightarrow\text{Centroid 1-20\%} \quad (16)$$

The designed algorithm for K-means is written.

Algorithm: K-means

Input:PD → Uploaded project data for clustering
Output: C-Data and C→ Clustered Data and their centroids
Initialize an estimated Centroid C = C1, C2 and C3
Where, $C1 = \frac{SIM_{COS}}{Count\ of\ PD} + \frac{SIM_{SOFTCOS}}{Count\ of\ PD} + \frac{SIM_H}{Count\ of\ PD}$

$$C2 = C1 + 20\%\ of\ C1$$
$$C3 = C1 - 20\%\ of\ C1$$

Calculate size of PD in terms of [Row, Col.] = size (PD)
For i→1 to all Row
For j→1 to all column
If PD (i,j)==C1
C-Data 1=PD (i,j)
Else if PD (i,j)==C2
C-Data 2=PD (i,j)
Else
C-Data 3=PD (i,j)
End
Adjust Centroid C using their mean
C = Average (C-Data 1, C-Data 2 and C-Data 3)
End – For
End – For
Return: C-Data and C as a Clustered Data and their centroids respectively
End – Algorithm

After selecting centroid, next step is to calculate Euclidian distance and hamming distance as metrics. The k-means method is a method of cluster analysis, the purpose of which is to divide m observations (from spaceP^n) into k clusters, with each observation referring to the cluster to the center (centroid) of which it is closest. Euclidean distance is used as a measure of proximity, which is given by equation (17).

$$p(x,y) = \sqrt{(x_1 - y_1)^2} + (x_2 - y_2)^2 + \cdots \dots \dots \dots (x_n - y_n)^2 \quad (17)$$

$$=\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Where x, y represent the Euclidean vector.

C. The Group Labelling

As shown in Fig. 1, once the group is formed, the correlation between the group elements is calculated using Mean Square Error (MSE), Standard Error (SE) and Standard Deviation (STD) for both the groups. In order to neutralize, the evaluated parameters for every group, log scale is considered which joins summative of MSE, SE and STD. A ruleset for labelling the group is designed as follows.

Algorithm Labelling

Input: Groups , Evaluated Parameters
Foreach grp in Groups
Calculate $\frac{\sum_{i=1}^{n1} MSE}{n}, \frac{\sum_{i=1}^{n1} STD}{n}, \frac{\sum_{i=1}^{n1} SE}{n}$

$$Judgement_{Level}.Append = \log(\frac{\sum_{i=1}^{n1} MSE}{n} + \frac{\sum_{i=1}^{n1} STD}{n} + \frac{\sum_{i=1}^{n1} SE}{n})$$

End For
Find min$(Judgement_{Level})$ and label it low effort as the co-relation is high and viceversa.

IV. Analysis and Discussion

In size and effort estimation in software project, there are a number of measures on the basis of which evaluations has performed. In this research, we have used three metrics to measure the performance of designed size estimation of software project as given by equation (18), equation (19), and equation (20). These are as follows:

Mean Square Error (MSE):

$$\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i^2 \quad (18)$$

Standard Error (SE):

$$\text{SE}=\sum_{i=1}^{n}\varepsilon_i^2 \quad (19)$$

Standard Deviation (STD):

$$\text{STD}=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2} \quad (20)$$

In this research, we have applied data fragmentation approach in addition to K-means approach. The results evaluated based on above mentioned parameters is listed in Table I.

TABLE I. EVALUATED PARAMETERS

Attributes	MSE	SE	STD
Project	0.0572	0.1247	0.472
TeamExp	0.0269	0.1345	0.136
ManagerExp	0.0378	0.2365	0.257
YearEnd	0.0472	0.3254	0.369
Transactions	0.0952	0.1025	0.158
Length	0.0360	0.2456	0.354
Effort	0.0761	0.6582	0.648
Entities	0.0871	0.7251	0.756
PointsNonAdjust	0.0675	0.6981	0.597
Adjustment	0.0752	0.5368	0.694
PointsAjust	0.0697	0.4865	0.486
Language	0.0315	0.3785	0.697

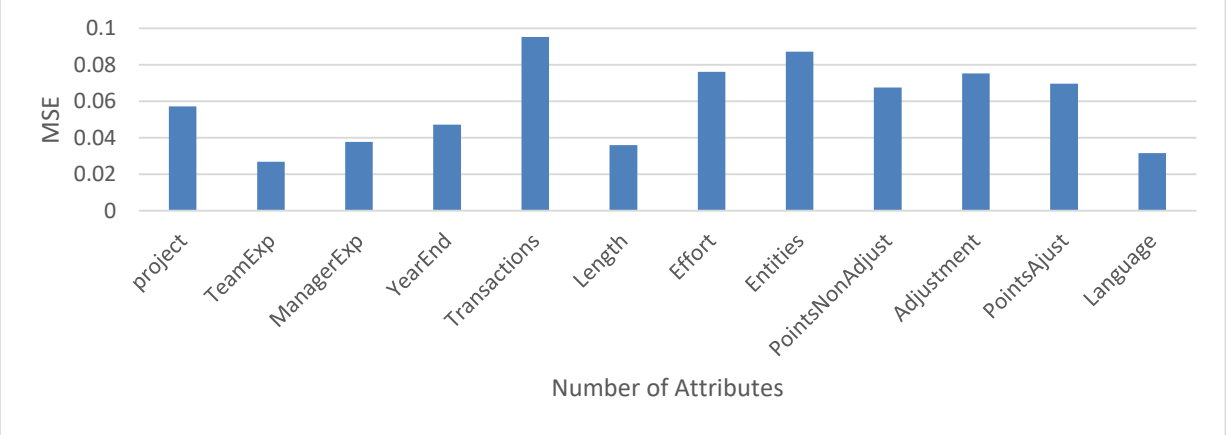

Fig. 3. MSE.

Fig. 3 represents the MSE value analysed for 12 different attributes. X-axis and y-axis represents the number of attributes taken for software project and evaluated MSE value. It is observed that for transaction attribute, designed system show highest error, whereas for team expert error is analysed as minimum. The average mean square error obtained for the designed system is observed as 0.05895, which is very small, and the system can be considered as accepted one for the early size estimation of project using ML technique.

SE is an estimate of the standard deviation of its sample distribution that roughly shows how much the value of a analyzed project rows can differ from its mean. The standard error of estimation is a value equal to the square root of the root mean square error (MSE) regression .MSE, in turn, is equal to the sum of the squares of the differences between the observed parameter (x) and the regression-estimated values ($\check{x}$) calculated from all observations and referred to their number n as represented by equation (19). The standard error value has been measured to find out the degree of deviation of the values obtained using the regression, from the actually observed, and thus assess the accuracy of the corresponding model. The deviation has been represented in Fig. 4, in which the graph shows maximum and minimum SE value of 0.7251, 0.1025 for attribute Entities and transaction respectively. Overall, the average SE analyzed for the proposed work is observed as 0.3877.

The standard deviation is a statistical characteristic of the distribution of a random variable, showing the average degree of dispersion of the values of the quantity relative to the mathematical expectation, denoted by greek σ (sigma).The standard deviation is measured in units of the random variable itself and is used to calculate the standard error of the arithmetic mean, when constructing confidence intervals , when statistically testing hypotheses , when measuring the linear relationship between random variables. It is defined as the square root of the variance of a random variable. Fig. 5 represents the examined STD for different attributes individually. The results show that for entities attribute, the designed early size estimation system using ML approach shows maximum STD, whereas, for 0.158 and 0.136 respectively.

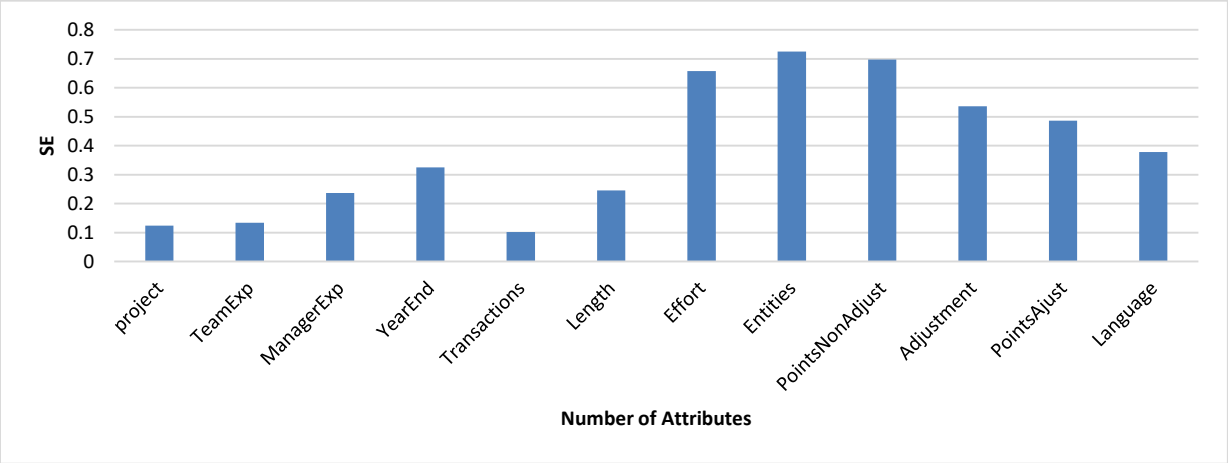

Fig. 4. SE.

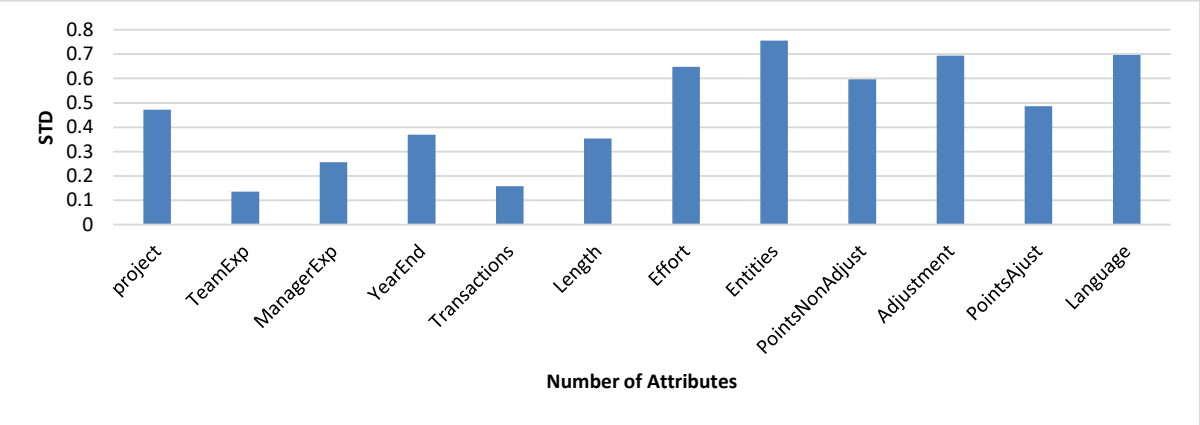

Fig. 5. STD.

V. Conclusion

This paper represents a novel segmentation method in order to labels the projects as high size and low size. In order to do so, three similarity indexes namely CS, SC and a hybrid similarity which is a combination of CS and SC is designed. The value of 'k' which is total number of groups that can be formed from a supplied set of data is set to be 2 for "high" and "low" size. Once the data is divided, the proposed work model goes for group labelling which is based on machine learning parameters namely MSE, STD and SE. As the parameters are different in nature, a log scale neutralization is performed. A rule architecture was developed to judge the labels and over 500 projects were labelled for high and low software size. A detailed evaluation of parameters is presented in section 4 and the current segmentation architecture will help in establishing Q-Learning mechanism for future project size estimations.

References

[1] E. N. Regolin, G. A. De Souza, A. R Pozo, & S. R. Vergilio, "Exploring machine learning techniques for software size estimation," In *23rd International Conference of the Chilean Computer Science Society, 2003. SCCC 2003. Proceedings*, pp. 130-136, November 2003. IEEE.

[2] C. Zhang, S. Tong, W. Mo, Y. Zhou, Y. Xia, B. Shen, "Esse: an early software size estimation method based on auto-extracted requirements features," *In Proceedings of the 8th Asia-Pacific Symposium on Internet ware*, pp. 112-115, 2016.

[3] D. Azar, "A genetic algorithm for improving accuracy of software quality predictive models: a search-based software engineering approach," *International Journal of Computational Intelligence and Applications*, vol. 9, no. 2, pp.125-36, 2010.

[4] M. Kaur, S.K. Sehra, "Particle swarm optimization based effort estimation using Function Point analysis," *In 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 140-145, 2014.

[5] L. J. Lazić, M. Petrović, P. Spalević, S. Serbia, "Comparative Study on Applicability of Four Software Size Estimation Models Based on Lines of Code," *In Proceedings of the 6th WSEAS EUROPEAN COMPUTING CONFERENCE (ECC'12), Prague, Czeh Republic*, pp. 71-80, 2012.

[6] P.R. PVGD, C. V. Hari, "Software Effort Estimation Using Particle Swarm Optimization with Inertia Weight," *Global Journal of Computer Science and Technology*, 1969.

[7] B. Baskeles, B. Turhan, A. Bener, "Software effort estimation using machine learning methods," *In 2007 22nd international symposium on computer and information sciences*, pp. 1-6, 2017.

[8] Y. Ahn, J. Suh, S. Kim, H. Kim, "The software maintenance project effort estimation model based on function points," *Journal of Software maintenance and evolution: Research and practice*, vol 15, no.2, pp.71-85, 2003.

[9] B. Baskeles, B. Turhan, A. Bener, "Software effort estimation using machine learning methods," *In 2007 22nd international symposium on computer and information sciences*, pp. 1-6, 2007.

[10] A.B. Nassif, D. Ho, L.F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software*, vol. 86, no. 1, pp.144-60, 2013.

[11] S.M. Satapathy, B.P. Acharya, S.K. Rath, "Earl stage software effort estimation using random forest technique based on use case points," *IET Software*, vol.10, no. 1, pp. 10-7, 2016.

[12] P. Sharma, J. Singh, "Systematic literature review on software effort estimation using machine learning approaches," *In 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, pp. 43-47, 2017.

[13] D. Spikol, E. Ruffaldi, G. Dabisias, M. Cukurova, "Supervised machine learning in multimodal learning analytics for estimating success in project-based learning," *Journal of Computer Assisted Learning*, vol. 34, no. 4, pp. 366-77, 2018.

[14] L. Lavazza, S. Morasca, "Empirical evaluation and proposals for bands-based COSMIC early estimation methods," *Information and Software Technology*, vol.109, pp.108-25, 2019.

[15] A.B. Nassif, M. Azzeh, A.Idri, A. Abran, "Software development effort estimation using regression fuzzy models," *Computational Intelligence and neuroscience*, pp.1-17, 2019.

[16] R. Silhavy, P. Silhavy, & Z. Prokopova, "Using Actors and Use Cases for Software Size Estimation," *Electronics 2021*, vol. 10, p. 592, 2021.

[17] M. Daud, & A. A. Malik, "Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (ADAFs)," *IEEE Access*, vol. 9, pp. 81986-81999, 2021.

[18] Y. Suresh, "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*, vol.13, no.2, 2022.

[19] Manisha, Rahul Rishi, "Early Size Estimation using Machine Learning," 8th International Conference on Computing for Sustainable Global Development (INDIACom), 2021, pp. 757-762.

[20] Manisha and Rahul Rishi, "An Enhanced Metaheuristic Based Cuckoo Search Algorithm for Software Size Estimation," 4th International Conference on Recent Developments in Control, Automation & Power Engineering (RDCAPE), 2021, pp. 526-520, doi: 10.1109/RDCAPE52977.2021.9633575.