DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.5189

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

NOVEL APPROACH FOR COST ESTIMATION OPTIMIZATION BASE FEATURE SELECTION AND CLASSIFICATION

Swati Sharma
Student of M.tech ,Department Of computer Science
Palampur(H.P) India

MunishKatoch,
Assistant Professor , Department Of computer Science
Palampur(H.P) India

Abstract- Software development effort estimation is the process of predicting the most realistic effort required to develop or maintain software. It is important to develop estimation models and appropriate techniques to avoid losses caused by poor estimation. IN text mining is use for understand user requirement, so understanding features optimize by ant colony optimization, it reduce the overlapping of features and then learn by generative classifier using Naïve Bayes.

Keywords-Optimization, Naïve Bayes, ACO, cost estimation

I. INTRODUCTION

Predictive modeling has been broadly utilized within IT enterprise environments, uses of which run from recommendation-based IT benefit conveyance and support [2], [10] to desert expectation for business software [7]. This exploration specifically goes for software effort estimation which has been examined in various software outline and development enterprises. In spite of the fact that there have been diverse estimation models grew so far, no single model has been appeared to be out performing others. Subsequently there is a requirement for new anticipating models because of dynamic changes in the environment. As of late more research consideration was attracted to factual modeling so as to address general effort estimation challenges. Authors in [8] gave a relative investigation comprising of more than twelve learning models against various benchmarks. Likewise in [5] an ensemble learning approach was proposed for effort estimation to support estimation exactness. Within Agile environments, nonetheless, there has not been much research for effort estimation utilizing supervised learning models.

Software development effort estimation is the way toward foreseeing the most reasonable effort required to create or keep up software in light of inadequate, dubious as well as loud information. Within Agile Development projects with a high level of client inclusion at each iteration, it is especially vital to have precise appraisals. It gives status visibility to the partners, contrasting arranged advance and the real advance. Effort gauges are utilized as contribution to extend plans, iteration designs, spending plans, and offering rounds.

Agile tries to limit the effect of inadequate estimation exactness by guaranteeing that the most vital functionality is created first. This is accomplished through an adaptable development process with short iterations. In Agile a client story is the unit at which software highlights are evaluated and created.

Every story is in the dialect of the client, and regularly written on a record card. The cards fill in as updates for discussions to be had about the components. The points of interest are then fleshed out later in the discussions, and passed on and archived as tests [2].

In section II the related work and the state of the art research in software effort estimation will be discussed; in section III the methodology for proposed algorithm is described, in section IV the key steps in building and validating system model are shown. Experimental results on improved estimation methods will be presented in section V. Finally conclusions are in section VI.

II. LITERATURE REVIEW

Cost estimation and software effort are essential at the beginning time of the software development life cycle for the project manager to have the capacity to effectively get ready for the software project. Sadly, the greater part of the estimation models rely upon subtle elements that will be accessible at the later phase of the development process. In [1], they proposed to use Function Point Analysis in application with dataflow outline to take care of this planning basic issue. The proposed system was approved through the graduate understudy software projects at the Chulalongkorn University Business School. The outcomes demonstrate the high capability of the appropriateness and some fascinating bits of knowledge which merit investigating. In [2], they proposed a novel model to anticipate software effort from use case graphs utilizing a cascade correlation neural network approach. The proposed demonstrate was assessed in light of the PRED and MMER criteria utilizing 214 industrial and 26 educational projects against a various linear regression display and the Use Case Point show. The outcomes demonstrate that the proposed cascade correlation neural network can be used with promising outcomes as an option way to deal with foresee software effort. An effort estimation display with more than 20 parameters is not extremely useful at early reasonable stage on the off chance that you don't have a legitimate approach for determining the information values. [3] displays a basic approach for anticipating software

development effort. The regression display uses item size and application sorts to anticipate effort. Item estimate is measured as far as the comparable source lines of code. The analysis depends on experimental information gathered from 317 extremely late projects executed inside the United States Department of Defense through the span of nine years starting in 2004. Statistical outcomes demonstrated that source lines of code and application sort are critical supporters of development effort. The condition is less difficult and more feasible to use for early gauges than customary parametric cost models. The impact of item measure on software effort should be translated alongside application area.

Agile software development process [4] speaks to a noteworthy departure from conventional, design based ways to deal with software building. Evaluating effort of agile software precisely in beginning period of software development life cycle is a noteworthy test in the software business. For enhancing the estimation exactness, different streamlining strategies are used. The Support Vector Regression (SVR) is one of these methods that aides in getting ideal assessed esteems. The primary goal of the exploration work did in this paper is to evaluate the effort of agile softwares utilizing story point approach. An endeavor has been made to advance the outcomes acquired from story point approach utilizing different SVR strategies to accomplish better forecast exactness. An execution examination of the models acquired utilizing different SVR bit strategies is likewise introduced with a specific end goal to feature execution accomplished by every technique.

The required effort of an errand can be evaluated subjectively in interviews with specialists in an association in various ways. Meeting systems managing which sort of things to ask are assessed and strategies for consolidating gauges from people into one gauge are looked at in an investigation. The outcome demonstrates that the meeting procedure is not as vital as the combination method. The gauge which is best regarding mean esteem and standard deviation of the effort depends on an equivalent weighting of every individual gauge. The analysis is performed inside the Personal Software Process (PSP) [5]. Despite many years of research, there is no accord on which software effort estimation strategies [6]-[7]create the most exact models. Prior work has detailed that, given M estimation strategies, no single technique reliably beats all others. Maybe as opposed to prescribing one estimation technique as best, it is savvier to create gauges from outfits of different estimation strategies. Strategy: Nine students were consolidated with 10 preprocessing choices to create 9 × 10 = 90 solo strategies. These were connected to 20 datasets and assessed utilizing seven mistake measures. This recognized the best n (in our case n = 13) solo techniques that demonstrated stable execution over numerous datasets and mistake measures. The best 2, 4, 8, and 13 solo strategies were then joined to produce 12 multi-methods, which were then contrasted with the performance techniques. 1) The main 10 (out of 12) multi-methods fundamentally outflanked each of the 90 solo techniques. 2) The blunder rates of the multi-methods were fundamentally not as much as the performance techniques. 3) The positioning of the best multi-method was amazingly steady. Conclusion: While there is no best single effort estimation technique, there exist best mixes of such effort estimation strategies. This examination in [7] focuses on development of effort estimation show for agile software projects. Development and use of the model is clarified in detail. The model was aligned utilizing the exact information gathered from 21 software projects. The trial comes about demonstrate that model has great estimation exactness as far as MMRE and PRED (n).In [8], analyzed how complex adaptive systems (CAS) hypothesis can be used to build our understanding of how agile software development practices can be used to build up this capacity. A mapping of agile practices to CAS standards and three measurements (item, process, and individuals) brings about a few proposals in systems development for "best practices". To give a superior understanding of lean software development methodologies and how they are connected in agile software development, in [9] they have inspected 30 encounter reports distributed in past agile software meetings in which encounters of applying lean methodologies in agile software development were accounted for. The analysis distinguished six sorts of lean application. The consequences of their investigation demonstrate that lean can be connected in agile processes in various conducts for various purposes. Lean ideas, standards and practices are frequently used for consistent agile process change, with the latest presentation being the kanban approach, presenting a persistent, flow-based substitute to time-boxed agile processes. In [10], the proposed development approach adjusts agile standards and examples keeping in mind the end goal to manufacture implanted control systems concentrating on the issues identified with the framework's imperatives and wellbeing. Solid unit testing is the establishment of the proposed procedure for guaranteeing convenience and rightness. Also, stage based plan approach is used to adjust costs and time-to-showcase in perspective of execution and functionality requirements. They presume that the proposed procedure lessens essentially the plan time and cost and also prompts better software measured quality and dependability.

III. ALGORITHM USED

A. Ant Colony Optimization

Ant colony optimization is fundamentally roused by the genuine ant settlements conduct and called artificial framework. Through the charts the Ant colony optimization calculation (ACO) is utilized for the taking care of computational problems and discovering great way. Like ant conduct, looking for way between food source and their colony to look through an ideal way comparative is the principle point of this calculation. To take care of the problem of traveling salesman problem (TSP) the principal ACO was created. Prior to the pheromones are refreshed along their food source trail on change probability bases a probability decision is made in the standard ACO. Before refreshing the pheromones along their trail to a food source in the standard ACO, which depends on the progress probability, ants settles on a probabilistic decision. For the kth ant the change probability at the time step t from city x to city y in the TSP problem:

$$PROB_{xy}^{k}(t) = \begin{cases} \dfrac{[\tau_{xy}(T)]^{\alpha}.[\eta_{xy}]^{\beta}}{\sum_{y \in I_x^k}[\tau_{xy}(T)]^{\alpha}.[\eta_{xy}]^{\beta}} & \text{if j} \epsilon I_x^k \\ 0 & Otherwise \end{cases}$$

Where

η_{xy} ←priority heuristic information,

τ_{xy} ←pheromones trail amount on the edge (x, y) at the time T,

The pheromone trail and heuristic information relative effects are identified by two factors i.e., $\alpha$and β. And the city's neighborhood set that are reasonable is denoted by I_x^k.

After a visit is finished by every ant, a constant dissipation rate at first bringing down them which refreshed the pheromone trail. Inferable from which every ant is permitted effective pheromone affidavit on curves which is its visit part as appeared in the condition underneath:

$$\tau_{xy} = (1-\rho).\tau_{xy} + \sum_{k=1}^{N} \Delta\tau_{yx}^{k}$$

Where

ρ←Pheromones rate of trail evaporation,

N←no. of ants,

The pheromone trail that is boundless aggregated is averted by the utilization of parameter ρ which empowers the awful choices to be overlooked by the calculation. The no. of cycles declining the pheromone quality related on circular segments which ants don't choose. $\Delta\tau_{yx}^{k}$, the trail substance quality per unit length which lays nervous (y,x) is given as takes after:

$$\Delta\tau_{yx}^{k} = \begin{cases} \frac{Q}{L_k} & if\ ant\ k\ in\ its\ tour\ uses\ edge\ (y,x) \\ 0 & Otherwise \end{cases}$$

Where

Q←constant that is predefined,

L_k ←length of the tour.

B. Support Vector Machine

SVM for classification in high-dimensional space builds a hyper plane. A hyper plane attaining best performance which maximizes the distances of both classes nearest training instances that is, functional margin maximization. The generalized error reduction is the main objective which makes it over fitting resistant.
A classification task is considered as: $\{u_i, v_i\}, i \in 1, 2, \dots\dots, n, v_i \in \{-1, 1\} and\ u_i \in R$
Where

u_i ←Data point,
v_i ← Corresponding label,
For separation, the hyper plane is given as:

$$x^t u + a = 0$$

Where
x← coefficient vector,
a←offset from origin
The optimization objective solving helps in obtaining ideal margin in case of separation.

$$Min\ G(x,\xi) = \frac{1}{2}\|x\|^2 + c\sum_{i=1}^{l}\xi_i$$
$$v_i(x^T u_i + a) \geq 1 - \xi_i, \xi \geq 0$$

Here
c←generalized parameter,
ξ←Positive slack variable
The Lagrangian multiplier $\alpha_i (i = 1, 2, \dots . l)$ is introduced for the reduction of optimization problem to Lagrangian dual problem (obtaining x and a). Therefore, the linear discriminant problem is given below:

$$G(u) = sgn([\sum_{i=1}^{l}\alpha_i v_i u_i^T u] + a)$$

When linear line do not separate the classes, thus mapping the original feature space to high dimensional feature space. The representation of the new decision function is:

$$G(u) = sgn([\sum_{i=1}^{l}\alpha_i v_i \phi(u_i)^T \phi(u)] + a)$$

Where
$u_i^T u$←input space in feature space depicted as $\phi(u_i)^T \phi(u)]$
Kernal function is utilized for computing $\phi(u)$and SVM (support vector machine) utilizes several kernel functions. γdefines the pre-defined parameter which controls the Gaussian kernal width.

IV. SYSTEM MODEL

The description of the system model flow:
Step1: Input the storyline dataset from with the text is extracted.
Step 2: TF-IDF features are extracted after the pre-processing of text.
Step 3: The extracted features are labeled and Ant Colony Optimization is initialized.
Step 4: Fitness function is updated and optimization of features takes place.
Step 5: Classification validation of the obtained optimized features with class in this step.
Step 6: Calculating the performance parameter: accuracy, precision and recall.

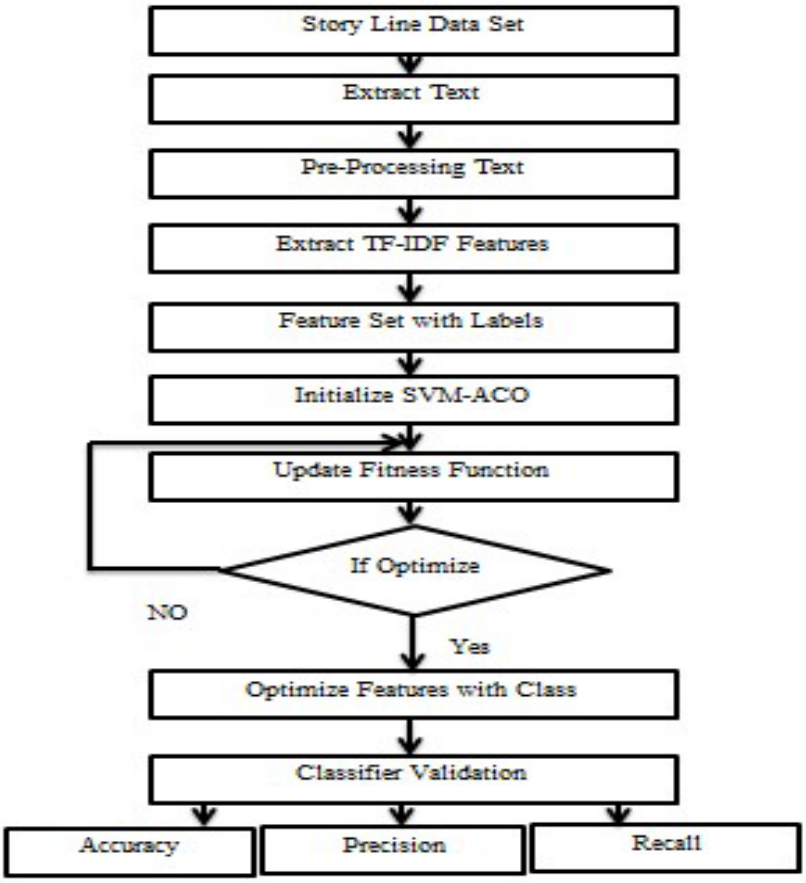

Fig 3.2: Flow Chart of Proposed Work

V. EXPERIMENTAL RESULT

Table 1: Performance comparison table between two classifiers: Nave Bayes and SVM-ACO

Parameters	**Naïve Bayes**	**Naive-ACO**
Accuracy	71.156	77.777
Precision	69.666	81.666
Recall	71.666	75.833

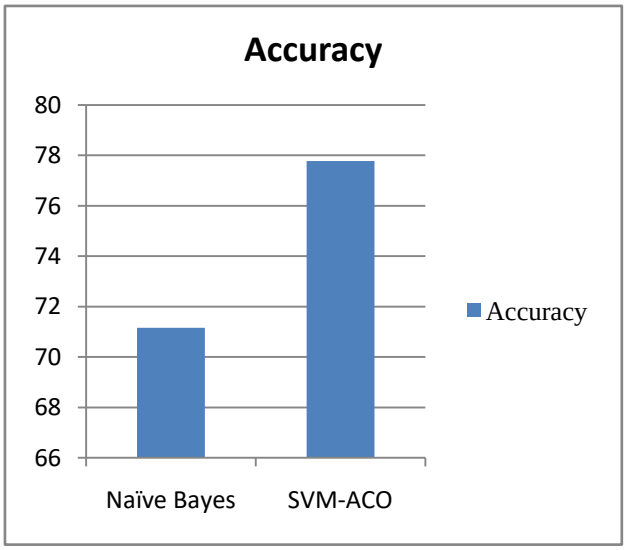

Figure1: Comparison graph of accuracy between Naive Bayes and SVM-ACO

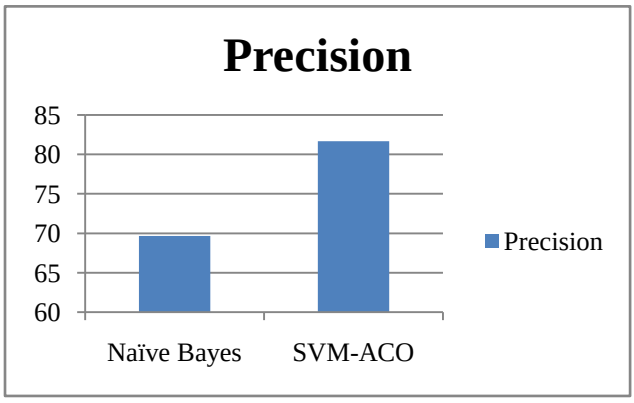

Figure 2: Comparison graph of precision between Naïve Bayes and SVM-ACO

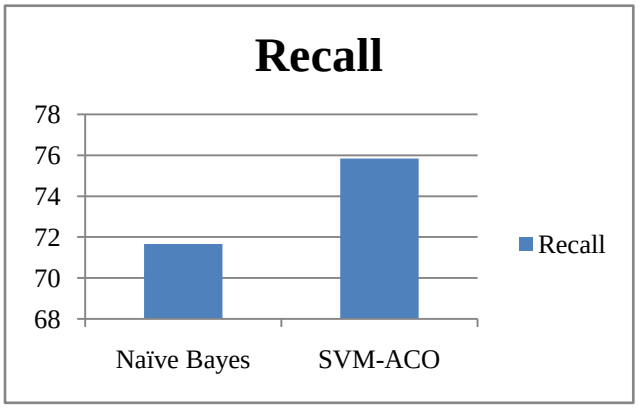

Figure 3: Comparison graph of recall between Naive Bayes and SVM-ACO

VI. CONCLUSION

Cost Estimate gets better trained on the story cards, human estimates, and actual effort data. Therefore by the later stages of the project the algorithm is more reliable than manual Planning Poker estimates and thus suitable as a tool for augmenting human effort estimation. For future work we propose research with larger datasets, and using features which were not used in this experiment (developers' demographics, story criticality, and other system and framework aspects).

VII. REFERENCES

[1] TharwonArnuphaptrairong, "Early Stage Software Effort Estimation Using Function Point Analysis: Empirical Evidence", in IMECS Vol-2,2013

[2] Nassif, Ali Bou, Luiz Fernando Capretz, and Danny Ho. "Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model." Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD"), 2012 13th ACIS International Conference on.IEEE, 2012.

[3] Rosa, Wilson, et al. "Simple empirical software effort estimation model." Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement",In ACM, 2014.

[4] Satapathy, ShashankMouli, Aditi Panda, and Santanu Kumar Rath. "Story point approach based agile software effort estimation using various svr kernel methods." (2014).

[5] Höst, Martin, and ClaesWohlin. "An experimental study of individual subjective effort estimation and combinations of the estimates." Proceedings of the 20th international conference on Software engineering.IEEE Computer Society, 1998.

[6] Kocaguneli, Ekrem, Tim Menzies, and Jacky W. Keung. "On the value of ensemble effort estimation." IEEE Transactions on Software Engineering 38.6 (2012): 1403-1416.

[7] Ziauddin, Shahid Kamal Tipu, and Shahrukh Zia. "An effort estimation model for agile software development." Advances in computer science and its applications (ACSA) 314 (2012): 314-324.

[8] Meso, Peter, and Radhika Jain. "Agile software development: adaptive systems principles and best practices." Information systems management23.3 (2006): 19-30.

[9] Wang, Xiaofeng, Kieran Conboy, and OisinCawley. "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development." Journal of Systems and Software 85.6 (2012): 1287-1299.

[10] Cordeiro, Lucas, et al. "An agile development methodology applied to embedded control software under stringent hardware constraints." ACM SIGSOFT Software EngineeringNotes 33.1(2008).