SOFTWARE EFFORT ESTIMATION WITH METAHEURISTIC OPTIMIZED ENSEMBLE FOR SOFTWARE CROWDSOURCING

Anum Yasmin and Wasi Haider Butt
Department of Computer & Software Engineering
National University of Sciences & Technology (NUST), Islamabad, Pakistan

ABSTRACT

Software crowdsourcing (SWCS) is rapidly growing from past decade due to its flexible work environment. Software effort estimation (SEE) is already renowned field in traditional software engineering, utilized in preliminary resource planning, budget, and time. SWCS platform can suffer from schedule, cost, and human resource uncertainty, which can be attained by estimating effort consumed on crowdsourced tasks. With more advent in SEE, ensemble effort estimation (EEE) is emerged providing unbiased results across different datasets. Recently, intelligent methods such as metaheuristic algorithms are used for ensemble by assigning optimal weights. This study aims to utilize prediction accuracy of EEE and metaheuristics on SWCS tasks to established accurate effort estimation model. In this work, ensembles are created using high predictive solo machine learning algorithms (RF, SVM, NeuralNet), whose weights are optimized with MO. TopCoder is selected as target SWCS platform, and this is first work to utilize TopCoder Design category tasks and contributes in dataset formation with relevant crowdsourced designing features. Results of proposed scheme clearly show that Metaheuristic-weight learning is giving more accurate ensembles with approximately 60% performance improvement compared to solo ML and other EEE techniques, proving it more suitable SEE technique for SWCS.

KEYWORDS

Software Effort Estimation, Crowdsourcing, Machine Learning, Ensemble Effort Estimation, Metaheuristic Optimization

1. INTRODUCTION

Software Crowdsourcing (SWCS) is new and emerging filed and known as distributed problem-solving platform for complex computations. SWCS mostly works by open-call format in competitive environment, to exploit collective crowd intelligence. Platforms enable SWCS services included oDesk, vWorker, eLance, Guru, TaskCity, and Taskcn and TopCoder. Among these platforms, TopCoder has the largest developer community and immensely utilized by researchers and practitioners (Sarı et al., 2019). The community decides whether to perform tasks by attending competitions which mainly influences by offered price of task. Currently, no SWCS platform has certain cost assignment mechanism. Decision to adopt a task can be largely impacted by effort required by the SWCS tasks, which can lead to justified task selection criteria.

SEE is already established field in traditional software engineering to estimate effort spent on software project. SEE is crucial since under and overestimation may lead unclear assumption about project’s time and resources. Due to the need and challenges of SEE, multiple techniques are developed in past including expert judgement (Jørgensen et al., 2010), algorithmic methods (Effendi et al., 2018) and machine learning (ML). Among these, ML-based SEE is popular since they handle complex, non-linear relationship between effort and software features. However, researchers agree that no single SEE technique outperformed in every context. Ensemble Effort Estimation (EEE) is explored for eliminating instability issues which predictions of more than one single SEE technique to improve accuracy. An SLR on SEE is conducted by (Idri et al., 2016) according to which, EEE outperformed in general single predictive approaches. EEE techniques are categorized into: (1) Homogeneous ensemble (HM), which work by combining more than two configurations of the same single SEE technique, and (2) Heterogeneous ensemble (HT) which combine at least two different SEE techniques via some combination rule. In SLR conducted by idri. et al HT comparatively accurate EEE. However, performance of HT ensemble majorly depends on method for combining single predictions. Weight assignment is one method of creating HT which assigns appropriate weights to predictions of single SEE techniques. Accurate weight learning is very critical in weighted HT since single SEE technique may induce its own bias in results. Metaheuristic optimization (MO) is recently investigated field in ML for catering optimization problems. In SEE, MO is utilized in dataset feature selection and hyperparameter tunning of ML algorithms. For HT ensemble weights learning, MO are rarely utilized (Palaniswamy et al., 2021; Song et al., 2019).

In this work, we are analyzing the impact of EEE for estimating effort of crowdsourced tasks. Study determines effectiveness of MO in HT weighted ensemble on our crowdsourced dataset. Ensemble is created by optimal weights obtained from MO algorithms. TopCoder is selected as target SWCS platform. Reason being, TopCoder is supporting highly skilled developers' community and tendency of researchers adopting TopCoder for analysis is high (Alelyani et al., 2017; Mao et al., 2013). Crowdsourced designing tasks on TopCoder platform are used for further analysis. Moreover, this is the first work to present any analysis on TopCoder tasks of "Design" category.

Proposed work attempts to address following research questions:

RQ1: What performance improvement is shown by ensemble models compared to solo ML model for effort estimation on SWCS dataset?

RQ2: Which EEE techniques are giving good results either homogenous (HM) ensembles or heterogenous (HT) ensembles in software effort estimation for SWCS dataset?

RQ3: What is the performance of ensemble created with metaheuristic optimization for predicting effort, and which metaheuristic technique provides best estimation?

Main contributions of the study are:

- Constructing crowdsourced designing task dataset, with highly relevant features for accurately estimating effort and establishing effort estimation scheme for SWCS designing perspective.
- Exploring the power of metaheuristic algorithms for optimal weights to heterogenous ensemble.
- Comparison of solo ML algorithms, homogenous and heterogeneous ensembles to provide most appropriate effort estimation solution for SWCS designing tasks.

Structure of the paper is as follows: Section 2 presents literature of previous SWCS and EEE work, Section 3 represents brief details of techniques used for proposed scheme implementation. Section 4 contains proposed scheme procedure, and Section 5 and Section 6 contains results and conclusions of this study.

2. LITERATURE REVIEW

Work done by (Mao et al., 2013) was the first study to address the pricing issue for TopCoder SWCS. 490 TopCoder projects with 16 cost drivers are extracted. Cost estimation is made using LogR, LReg, C4.5, CART, QUEST, KNN, NNet and SVMR. Study verified that ML estimation surpass the random guessing and parametric model (COCOMO). However, the attribute selection is not practical since some attributes are not directly participating in effort and some attributes require information about previous phase, not available for all tasks. This issue is tackled by (Alelyani et al., 2017) with context-based pricing on TopCoder with 6 pricing factors, analyzed from task descriptions and 7 classification models (LDA, LR, kNN, CART, Naive Bayes, SVM and NNet). Although the study work on limited information, but aspects related to developer's ability, task and technology complexity are not considered.

Ensemble effort estimation (EEE) is frequently explored in SEE. (Idri et al., 2016) performed SLR of 24 past EEE studies published between 2000 to 2016. SLR concluded that 17 studies used homogenous ensemble, making them most frequently utilized ensemble models. For single base learners, 12 studies choose ANNs and DTs. Overall, EEE models produce accurate results, compared to their base leaners. HT is explore by 9 studies, which yielded better performance compared to their constituent ML techniques (with MMRE=62.48%, Pred(25)=43.35%, and MdMRE=26.80%).

Work of (Elish et al., 2013) implemented various HM and HT ensembles with different hybrid optimization techniques and implementation is done on 5 datasets of PROMISE repository. Results revealed that performance of ensemble is higher than solo models.

Similarly (Hussain et al., 2021) incorporated enhanced SVR for recursive feature elimination. RF is used as ensemble and experimentation is performed on COCOMO and NASA93 datasets. Results have shown improve performance in terms of MMRE MMER, MBRE, MIBRE error measures.

Work of (Shukla et al., 2019) used ensembles of Multi-Layer Perceptron Neural Network (MLPNN), including Ridge-MLPNN, Lasso-MLPNN, Bagging-MLPNN, and AdaBoost-MLPNN to improve effort estimation. Results are evaluated on Desharnais dataset (PROMISE) concluded that ensemble MLPNN has 82.213% increased accuracy. (Palaniswamy et al., 2021) utilized Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) for tuning hyperparameters of stacking ensemble. Hyperparameters of MLP, RF, SVM and AdaBoost are tuned on ISBSG dataset and identified that stacking with hyperparameter tuning provides better accuracy.

Metaheuristics is explored frequently in past researchers for various optimization problems. However, for SEE, MO mostly utilized in hyperparameters tunning while a little work on ensemble weight optimization is tackled to get best estimation results. Known MO techniques included: GA, PSO, Artificial bee colony (ABC), Firefly algorithm (FFA) and Bat algorithm (BA) (Kaushik et al., 2019).

From the literature review, it is concluded that no previous SEE work has utilized crowdsourcing projects for SEE experimentation. Most frequently used repositories are PROMISE, ISBSG or other private projects. Also, EEE domain is not much explored in SWCS domain. This work combines prediction accuracy of EEE, more specifically metaheuristic optimized weighted ensemble (MO-WE) on SWCS tasks to established accurate effort estimation model.

3. METHODOLOGY USED

This section includes the different algorithms used to build effort estimation models on SWCS dataset. Effort is predicted using, solo ML algorithms, Homogenous ensembles and Heterogenous ensembles to assess more accurate prediction. Details of algorithms used in this study are described in Table 1.

Table 1. Description of SEE techniques used in study

Type	SEE technique	Description
ML algorithms	RF	combines several DTs with bagging, performs better with inter-correlated features, work with both numerical and categorical variables.
	SVM	deals with linearly separable data and works both for classification and regression problem.
	NeuralNet	Fast learning and is simple to implement and represent complex relationships between dependent and independent variables.
Homogenous ensemble (HM)	Bagging	ensemble algorithm works by creating separate samples of training datasets (bootstrap), model is trained for each sample, and then, results are combined (average) to find the final output.
	Gradient boosting machine (GBM)	incorporates different regularization techniques to maintain under-over fitting
Heterogenous ensemble (HT)	Majority Voting	implemented by creating two or more models with output combined by taking mean or mode.
	Stacking	multiple sub-models are created and combined with a meta-model to predict effort.
	Manually Weighted ensemble (MA-WE)	Manual weights assigned and highest weight given to best performing model in 2:1:1 ratio.
	GLM-based weighted ensemble (GLM-WE)	Assigned ensemble weights are coefficients of GLM model, trained using solo ML models' prediction.
	NeuralNet based weighted ensemble (NN-WE)	ensemble weights are learned using NeuralNet model.

RF, SVM and NeuralNet are selected as solo learners. Both categories of ensemble, HM and HT are used for experimentation. Bagging and Gradient boosting machine (GBM) are used as HM. For HT, multiple forms are used i.e., non-weight-based ensembles (Majority Voting, Stacking), weight-base ensemble (with manual, Generalized linear model (GLM) and NeuralNet based weights). In this study, MO weights learning of our solo ML models is included to form ensemble and assign optimal weights for solo ML predictions. For this, MO techniques are: GA, PSO, FFA and BA.

4. PROPOSED SCHEME

This section describes proposed scheme of this study. First, TopCoder designing task dataset is created with crowdsourced design-relevant features, along with regression analysis to ensure regression significance of features. Details of dataset are discussed in Section 4.1. After this, effort estimation is done on crowdsourced dataset (described in Section 4.2).

4.1 Dataset

This section describes the details of the dataset used in this study. As SWCS platform, TopCoder is selected. For dataset, 1500 tasks from TopCoder "Design" category are crawled, from Oct 31, 2019, to Aug 13, 2021 and sorted according to their posting time. For simplicity, only those tasks are included having 70% above submissions score with 1 winner. After applying elimination criteria, 200 tasks are finalized to be part of the dataset. Profile details of all task winners (referred as "designer" in this work) are also collected. Careful selection of TopCoder designing features is made while preparing dataset, ensuring all features are highly relevant for SEE without any dependability on previous phase. Finalized dataset can be found at 1. Total 18 input design features and one output feature as "Effort" are gathered, grouped into 5 categories (shown in Table 1). Brief description of input features is as follows:

- ***Task characteristics (TSCH)*** represent task or challenge characteristics particular to designing i.e., kind of application and design required in tasks (i.e., UI/UX or data visualization) Features of this category: APPT, DREQ, DSCH.
- ***Personnel Metrics (PERS)***, includes the details related to designer's capabilities. i.e., designer's winning and submission trend, previous performance, skills and experience. Features are: WINP, SBRT, SSCR, AVGP, TSCR, EXPR.
- ***Requirement Complexity (RCPX)*** shows business requirements complexity i.e., rounds, screens and platform constrains. Features included are: ROUN, STOC, SCRN, FEAT, ECNS.
- ***Artifact complexity (ACPX)*** includes helping material provided with task, their helping level, required documents by task and their overhead level. Features included are: ASST, DLVB.
- ***Quality parameters (QLTY)***, capturing quality aspects while delivering the output. Features included: FFIX, JUDG.

$$Effort = 91.5 + 75.5\,(\mathrm{APPT}) + 25.3\,(\mathrm{DREQ}) + 59.4\,(\mathrm{DSCH}) - 8.7\,(\mathrm{WINP}) - 9.3\,(\mathrm{SBRT}) - 5.1\,(\mathrm{SSCR}) - 4.8\,(\mathrm{AVGP}) - 1.8\,(\mathrm{TSCR}) - 24.8\,(\mathrm{EXPR}) + 15.9\,(\mathrm{ROUN}) + 2.5\,(\mathrm{STOC}) + 31.5\,(\mathrm{SCRN}) + 6.9\,(\mathrm{FEAT}) + 23.5\,(\mathrm{ECNS}) + 45.8\,(\mathrm{ASST}) + 36.1\,(\mathrm{DLVB}) + 2.7\,(\mathrm{FFIX}) + 28.3\,(\mathrm{JUDG}) \quad (1)$$

Datatypes assigned to features are Ordinal or Numerical. Ordinal features are assigned 6 levels depending upon their complexity and impact on effort as, "very low", "low", "nominal", "high", "very high" and "extra High" and values assigned to each level from 1 to 6 (1 assigned to "very low" while 6 to "extra High"). Effort is dependent feature of the dataset and measured based on task duration in hours and reflected in person-hours (PH) unit. Table 1 also showing regression analysis details for dataset features. Regression is performed on dataset with significance level ρ-value < 0.05. Overall model statistics are: R2 = 0.93, Standard Error = 0.38, F-Statistics = 128.5. Regression analysis shows that all feature are significant at ρ = 0.1 in performing regression and estimating effort. However, features like DREQ, SSCR, EXPR, FEAT, DLVB are not showing significance at ρ = 0.05. Regression equation for the dataset features is given in Eq. (1). Features of APPT, DREQ, DSCH, ROUN, STOC, SCRN, FEAT, ECNS, DLVB, FFIX and JUDG put an increasing impact (having positive β) on effort consumed on task. Similarly features like WINP, SBRT, SSCR, AVGP, TSCR, EXPR and ASST possess negative β, i.e., effort decreases by enhancing the values of these features.

[1] https://drive.google.com/drive/folders/12uBPa-KyUcZ6hj5uoSUrYkzCzynqSKXW?usp=sharing

Table 2. Descriptive statistics and regression details dataset features

Category	Features	Description	Data Type	Min	Max	Mean	β	τ-value	ρ-value
Task characteristics (TSCH)	APPT	Kind of application under design (web/android/iOS)	Ordinal	1	6	3.67	75.5	6.74	2.10e-10
	DREQ	Kind of design required (UI/ high or low fidelity etc.)	Ordinal	1	6	3.56	25.34	0.29	0.07
	DSCH	Required designing new application or adjustment in previous design	Ordinal	2	4	3.52	59.35	2.39	0.02
Personnel Metrics (PERS)	WINP	Win percentage	Numerical	0.47	100	11.12	-8.72	-2.36	0.02
	SBRT	Submission rate	Numerical	23.9	100	83.87	-9.27	-2.06	0.04
	SSCR	Screening success rate	Numerical	54.5	100	98.83	-5.06	-0.43	0.08
	AVGP	Average placement	Numerical	1	154.8	14.05	-4.80	-3.51	0.0005
	TSCR	Count of designer's acquired skills	Numerical	1	19	9.88	-1.81	-1.39	2.10e-10
	EXPR	Designer's experience in years	Numerical	0.08	13.16	7.96	-24.83	-2.49	0.07
Requirement Complexity (RCPX)	ROUN	No. of rounds in task	Ordinal	1	5	2.86	15.94	0.36	0.02
	STOC	Stock photography allowed	Ordinal	2	4	2.46	13.25	2.16	0.02
	SCRN	No. of screens required to design	Ordinal	1	5	2.55	31.50	4.05	0.04
	FEAT	No. of features required to design	Ordinal	1	5	2.47	6.97	1.25	0.09
	ECNS	Environmental constraints	Ordinal	2	6	3.88	23.51	1.67	0.0005
Artifact complexity (ACPX)	ASST	Assets provided with task for help	Ordinal	2	6	3.49	-45.76	-1.02	2.10e-10
	DLVB	Deliverable required with task	Ordinal	1	6	4.11	36.11	2.09	0.07
Quality parameters (QLTY)	FFIX	Final fixes required	Ordinal	2	4	2.95	2.66	1.41	0.018
	JUDG	Judgement criteria for design	Numerical	1	6	4.05	28.25	2.82	0.02

4.2 Effort Estimation Framework

In this section, effort estimation scheme is established for SWCS dataset, explained in Section 4.1. Figure 1 represents proposed framework. As solo models RF, SVM and NeuralNet are used to create ensemble. For weight optimization, GA, PSO, FFA and BA are used as MO. Optimal weights for each solo algorithm are obtained with lowest RMSE as fitness criteria. Optimal weights are used to created Metaheuristic optimized weighted ensemble (MO-WE) to obtain final prediction. Following steps are included in proposed framework:

1) Divide dataset into training and testing samples with 70:30 ratios.
2) For training dataset, train data separately with three solo algorithms (RF, SVM, NeuralNet).
3) Acquire predictions from three trained models on training dataset.
4) Obtain weights of predictions with MO (GA, PSO, FFA, BA) with fitness = minimum RMSE
5) On 30% remaining test data, repeat step 2-3 and acquire testing predictions.
6) Predict final outcome on testing data by creating MO-WE i.e., assign optimal weights (obtained from MO in step 4) to testing data predictions in step 5.

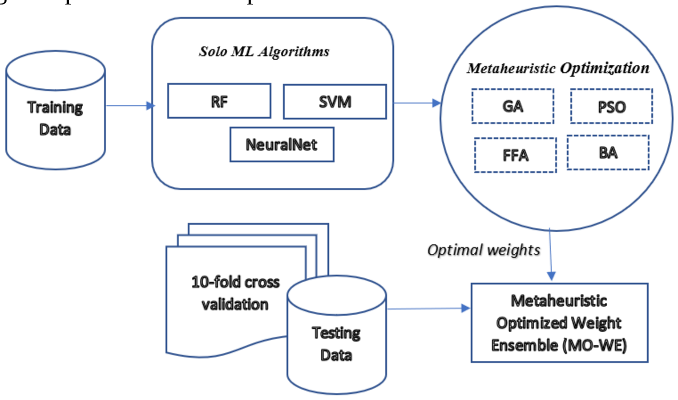

Figure 1. Overview of proposed MO-WE framework

4.3 Experimental Setup

For comparison, all models are trained on dataset (Table 2) by randomly dividing dataset into 70-30 ratio with 10-fold cross validation (10-fold-CV). All experiments are executed in RStudio. 5 performance evaluation metrices are used in this study represented in Eq. (2)- Eq. (7).

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(Effort_{actual} - Effort_{estimated})^2 \qquad (2)$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(Effort_{actual} - Effort_{estimated})^2} \qquad (3)$$

$$MRE = \left|\left(\frac{Effort_{actual} - Effor_{estimated}}{Effort_{actual}}\right)\right| \qquad (4)$$

$$MMRE = \frac{1}{T}\sum_{t=1}^{T} MRE_t \qquad (5)$$

$$MdMRE = median(MRE_t) \;\; t \in \{1 \cdots T\} \qquad (6)$$

$$PRED(d) = \frac{100}{T}\sum_{t=1}^{T}\begin{cases}1 \; if \; MRE \leq d \\ 0 \; otherwise\end{cases} \qquad (7)$$

Where t∈[1⋯T] is SWCS task from dataset T. Model having higher performance exhibit lower values of MSE, RMSE, MMRE, MdMRE, and higher PRED.

5. RESULTS AND DISCUSSION

This section presents the results of implementing SEE models (mentioned in Section 3 and Table 1) and reported in Table 3. Results of proposed MO-WE is also compared with other models. MO-WE created with GA, PSO, FFA and BA are referred as: GA-WE, PSO-WE, FFA-WE and BA-WE respectively.

Table 3. Evaluation metrices results of SEE techniques

SEE Techniques	RMSE	MMRE	MdMRE	PRED
RF	26.59	22.97	0.68	56.7
SVM	39.60	35.52	1.05	51.7
NeuralNet	37.49	25.82	0.46	71.2
Bagging	26.21	22.85	0.67	55.0
GBM	25.60	20.51	0.65	60.0
Majority Voting	34.26	26.22	0.51	63.9
Stacking	28.39	24.04	0.56	56.9
MA-WE	40.73	29.30	0.60	56.8
GLM-WE	41.09	34.87	0.77	62.1
NN-WE	36.65	31.23	0.84	61.4
GA-WE	24.16	18.63	0.41	80.0
PSO-WE	24.04	19.04	0.45	75.0
FFA-WE	22.71	17.09	0.33	80.0
BA-WE	22.60	16.98	0.30	81.7

Table 3 shows the performances of implemented SEE techniques to predict effort on our dataset using 5 performance metrices mention in Section 4.3.

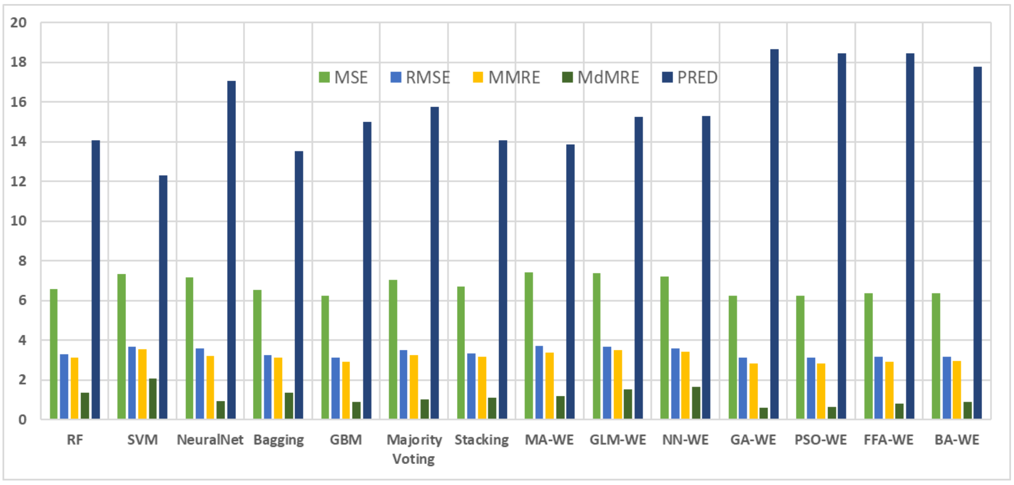

Figure 2. Performance comparison of SEE techniques

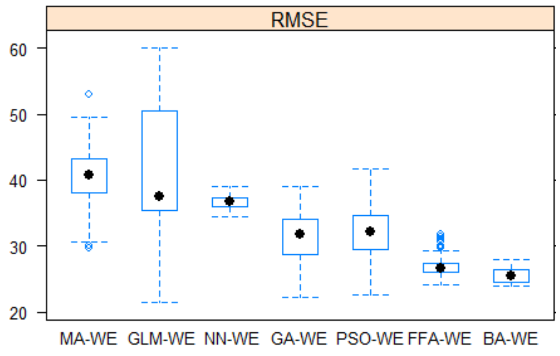

Figure 3. RMSE boxplot for weighted ensemble methods

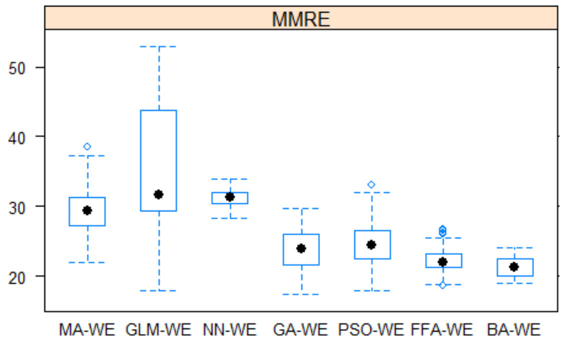

Figure 4. MMRE boxplot for weighted ensemble methods

- Solo algorithms (RF, SVM, NeuralNet) did not performance better compared to ensembles like bagging and GBM. However, compared to HT, RF is only solo algorithm performed better in RMSE and MMRE. SVM, is least performing solo algorithm. However, no solo algorithm surpassed any MO-WE (GA-WE, PSO-WE, FFA-WE, BA-WE). Overall, MO based ensembles has outperformed all models, in all performance metrices.
- Analyzing all ensembles, Bagging and GBM are better performing after MO-WE, compared to other HT ensemble.
- Among non-weight ensembles, Stacking has shown least amount of error while predicting effort. For HT, MA-WE and GLM-WE induced largest error and could not surpassed any other model.
- Overall, weighted HT ensembles (not using MO), have not shown performance improvement, compared to non-weight-based ensembles (Majority Voting, Stacking).
- MO-WE outperformed all HM and HT and solo techniques. Also, MO weight learning gives better results compared to other weight learning methods (NN-WE and GLM-WE).
- BA-WE has shown least amount of error among all models, after which FFA-WE proved better for predicting SWCS task effort, reflects, BA-WE has better weight learning ability for SWCS dataset.

Figure 2 shows performance comparison of all models using 5 metrices. Decreasing trend in error can be seen in MO-WE ensembles. PRED of MO-WE is also on increasing side, reflecting better learning capability of metaheuristics. Error bars of GLM-WE and MA-WE are higher among all, showing poor performance of algorithms on dataset. Figure 3 and Figure 4 represent boxplot of all weighted ensembles showing their RMSE and MMRE performance respectively. As it can be seen, GA-WE, PSO-WE, FFA-WE and BA-WE have lower mean, compared to MA-WE, GLM-WE, and NN-WE, giving better estimate of effort.

Furthermore, error distribution in GLM-WE is varied as it shows longer range, while FFA-WE and BA-WE show even distribution of error for effort prediction. RMSE and MMRE of GA-WE, PSO-WE, FFA-WE and BA-WE almost at center of the boxes, reflecting RMSE and MMRE for these models are symmetrically distributed around the medians. The FFA-WE and BA-WE boxes are shorter and have narrower ranges of values, which indicates that their RMSE, MMRE have less variation compared to other weighted ensembles (except NN-WE). Small number of outliers can be seen in FFA-WE and PSO-WE, while GA-WE and BA-WE possess no outliers in resultant performance values.

Results accumulated above can confirm that individual ML algorithms are not showing credible results, compared to ensemble, particularly GBM. This shows boosting mechanism can be fruitful effort prediction mechanism for datasets under-consideration, having both numerical as well as categorical features. Ensembles with averaging combination rule (i.e., Majority Voting, MA-WE) are not appropriate choice for estimating effort in SWCS dataset, since simply combining individual algorithms (without any weight learning mechanism) can negatively affect the performance. Further, non-MO weight learning i.e., GLM and NN can surpass in manual weight assignment, but due to learning limitations, optimal solution cannot be achieved. MO are good weight learning solution for our dataset. Reason is, MO are better for non-linear multi-objective problems and presence of numerical and ordinal features in dataset tends to make regression non-linear.

Here we can answer the research questions

RQ1: What performance improvement is shown by ensemble models compared to solo ML model for effort estimation on SWCS dataset?

Ans1: According to Table 3 ensemble models show considerable results compared to solo ML models. Ensembles with MO-WE show approximately 27% to 63% improvement in performance than solo RF, SVM and NeuralNet while other ensemble models show 13% improvement in overall results compared to their solo algorithms.

RQ2: Which EEE techniques are giving good results either homogenous (HM) ensembles or heterogenous (HT) ensembles in software effort estimation for SWCS dataset?

Ans2: Considering the performance of HT ensemble (without metaheuristic weights), both homogenous ensembles (Bagging and GBM) have outperformed, showing RMSE of 26.21 and 25.60 respectively. Both RMSE values are lower compared to best and worst performing non-metaheuristic HT ensemble i.e., Stacking (RMSE 28.39) and GLM-WE (RMSE 41.09) respectively. This shows overall 36% and 9% improvement in prediction performance. For Metaheuristic HT ensembles, Bagging and GBM are not showing any improvement, in fact, all of MO-WE models have shown overall 7% to 11% improved RMSE results.

RQ3: What is the performance of ensemble created with metaheuristic optimization for predicting effort, and which metaheuristic technique provides best estimation?

Ans3: MO-WE models are better than all comparing models in prediction. BA-WE is showing better performance with RMSE 22.60, after which FFA-WE is getting accurate results with RMSE 22.71. GA-WE is showing least performance among all MO-WE with RMSE 24.16. However, overall, GA-WE and PSO-WE is getting almost similar results while FFA-WE and BA-WE are giving similar performance values.

6. CONCLUSION AND FUTURE WORK

Software crowdsourcing (SWCS) is well researched field in software engineering domain but not explored much in effort estimation perspective. In this work, SWCS is analyzed particularly in crowdsourced designing perspective for analyzing effort consumed on crowdsourced task. For this, TopCoder tasks data is collected with highly effort-relevant and readily available features. Ensemble effort estimation is applied on collected dataset with three solo ML algorithms (RF, SVM, NeuralNet). Further, impact of metaheuristic optimization is explored in regard of assigning optimal weights to ensemble. This is the first work to utilized crowdsourced designing task data for analyzing effort. Results of our model reflect that for SWCS dataset, metaheuristic algorithms provide efficient mechanism in assigning optimal weights to ensemble, which significantly improve prediction accuracy in comparison with other ensemble techniques.

As future work, we intend to extend the study for other crowdsource categories i.e., development, testing and QA. Furthermore, other crowdsourced and open-sourced software development platforms are to be evaluated for metaheuristics and effort estimation perspective.

REFERENCES

Alelyani, Turki, Mao, Ke, & Yang, Ye. (2017). Context-centric pricing: early pricing models for software crowdsourcing tasks. Paper presented at the *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering*. doi:https://doi.org/10.1145/3127005.3127012

Effendi, Yutika Amelia, Sarno, Riyanarto, & Prasetyo, Joco. (2018). Implementation of Bat Algorithm for COCOMO II Optimization. Paper presented at the *2018 International Seminar on Application for Technology of Information and Communication*. doi:https://doi.org/10.1109/ISEMANTIC.2018.8549699

Elish, Mahmoud O, Helmy, Tarek, & Hussain, Muhammad Imtiaz. (2013). Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation. *Mathematical Problems in Engineering, 2013*. doi:https://doi.org/10.1155/2013/312067

Hussain, Azath, Raja, Maheswari, Vellaisamy, Pandimurugan, Krishnan, Sangeetha, & Rajendran, Lakshminarayanan. (2021). Enhanced framework for ensemble effort estimation by using recursive-based classification. *IET Software, 15*(3), 230-238. doi:https://doi.org/10.1049/sfw2.12020

Idri, Ali, Hosni, Mohamed, & Abran, Alain. (2016). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software, 118*, 151-175. doi:https://doi.org/10.1016/j.jss.2016.05.016

Jørgensen, Magne, & Halkjelsvik, Torleif (2010). The effects of request formats on judgment-based effort estimation. *Journal of Systems Software, 83*(1), 29-36. doi:https://doi.org/10.1007/s41870-019-00339-1

Kaushik, Anupama, & Singal, Niyati. (2019). A hybrid model of wavelet neural network and metaheuristic algorithm for software development effort estimation. *International Journal of Information Technology*, 1-10. doi:https://doi.org/10.1007/s41870-019-00339-1

Mao, Ke, Yang, Ye, Li, Mingshu, & Harman, Mark. (2013). Pricing crowdsourcing-based software development tasks. Paper presented at the *2013 35th International Conference on Software Engineering (ICSE)*. doi:https://doi.org/10.1109/ICSE.2013.6606679

Palaniswamy, Sampath Kumar, & Venkatesan, R (2021). Hyperparameters tuning of ensemble model for software effort estimation. *Journal of Ambient Intelligence Humanized Computing, 12*(6), 6579-6589. doi:https://doi.org/10.1007/s12652-020-02277-4

Sarı, Aslı, Tosun, Ayşe, & Alptekin, Gülfem Işıklar (2019). A systematic literature review on crowdsourcing in software engineering. *Journal of Systems Software, 153*, 200-219 doi:https://doi.org/10.1016/j.jss.2019.04.027

Shukla, Suyash, Kumar, Sandeep, & Bal, Pravas. (2019). Analyzing Effect of Ensemble Models on Multi-Layer Perceptron Network for Software Effort Estimation. Paper presented at the *IEEE World Congress on Services (SERVICES)*. doi:https://doi.org/10.1109/SERVICES.2019.00116

Song, Liyan, Minku, Leandro L, & Yao, Xin (2019). Software effort interval prediction via Bayesian inference and synthetic bootstrap resampling. *ACM Transactions on Software Engineering Methodology, 28*(1), 1-46. doi:https://doi.org/10.1145/3295700