Software Cost Estimation using Stacked Ensemble Classifier and Feature Selection

Mustafa Hammad
Department of Software Engineering
Mutah University
Al-Karak, Jordan

***Abstract*—Predicting the cost of the development effort is essential for successful projects. This helps software project managers to allocate resources, and determine budget or delivery date. This paper evaluates a set of machine learning algorithms and techniques in predicting the development cost of software projects. A feature selection algorithm is utilized to enhance the accuracy of the prediction process. A set of evaluations are presented based on basic classifiers and stacked ensemble classifiers with and without the feature selection approach. The evaluation study uses a dataset from 76 university students' software projects. Results show that using a stacked ensemble classifier and feature selection technique can increase the accuracy of software cost prediction models.**

***Keywords*—Software project management; effort estimation; prediction model; machine learning**

I. Introduction

The process of developing software has evolved into a fundamental function of modern society as a result of the quick development of software in our days. However, a crucial step in the lifecycle of software development is software effort estimation. The goal of a software development task is to deliver the product on time and within budget. The planning process for any software project must therefore include early software cost estimation.

Predicting the amount of work required to create a software system is a part of software effort estimation. It is expressed in terms of the number of working hours or the number of hours needed to construct the software. Software testing, maintenance, requirements engineering, and other software operations are all included in the broad category of software effort estimation.

To produce software projects, various software development lifecycle models call for varying amounts of work at each stage. Software effort estimation is regarded as one of the most significant problems in software engineering. It affects the cost of the project and is a problem that many engineers and project managers encounter. A major issue that could harm software companies is the accuracy of the development effort estimation.

Several scholars as [1] have suggested various models to predict the effort of software development. Several researches have been done to determine the early software effort estimate to determine the significance [2]. Software companies need to understand the work required to develop projects in addition to how they should proceed about accomplishing this.

In this paper, the software work is estimated based on project attributes using four machine learning techniques. This study's primary objective is to assess software effort estimation models created using machine learning techniques.

Before deciding to use a software component as a reusable asset, software engineers must analyze the software component. Assessing reuse potentials can be aided by predicting successful reuse. Datasets are used to train and test predictive models. Datasets, however, occasionally include attributes that are not useful. The performance of the model may suffer as a result of these characteristics. Therefore, choosing the key attributes improves the performance of the model and yields a more accurate output.

In this paper, an empirical study utilized a dataset to investigate and extract the essential features that lead to a successful reuse experience. Usually, the performance of a prediction model can be improved with an ideal subset of useful features. Feature selection algorithm selects a portion of the original dataset's most useful qualities. This subset can improve the prediction model's effectiveness and efficiency. Additionally, it avoids data overfitting. In this paper, six feature selection algorithms were utilized and evaluated to enhance accuracy. These algorithms are; Classifier Attribute Evaluation, Correlation Attribute Evaluation, InfoGain Subset Evaluation, Wrapper Subset Evaluation, Classifier Subset Evaluation, and CfsSubset Evaluation.

There are many benefits of using feature selection techniques. For instance, it reduces the training time, helps visualize the data, and optimizes the storage requirements. In this paper, the primary purpose of using feature selection techniques is to improve the prediction model's performance by removing the irrelevant attributes [3].

The organization of this paper is as follows; the next section discusses the literature that concerning software effort prediction. The proposed evaluation model is presented in Section III followed by a brief description about the used dataset in the evaluation process. The experimental results are presented in Section V. Finally, Section VI concludes the paper and highlights the future work.

II. Related Work

Many approaches have been presented in the literature about estimating the development's efforts of software projects. Most of these approaches utilize the machine learning and artificial intelligence techniques to predict the effort.

Rankovic et al. [4] proposed two different architectures of Artificial Neural Networks (ANN) for predicting software effort. They used exponent-scale factors, cost factors, and software size as control variables from COCOMO models. BaniMustafa [5] predicted the effort estimation by applying machine learning techniques and data mining. He applied Naïve Bayes, Logistic Regression and Random Forests. Priya Varshini et al. [6] proposed stacking using random forest for effort estimation. They used ensemble techniques to create and combine multiple models termed base-level classifiers. The works in [7, 8] applied Artificial Neural Network (ANN), Support Vector Machines (SVM), K-star, and Linear Regression to estimate software effort based on project features. Mahdie et al. [9] provided a detailed review about the application of Machine Learning in software project management which includes effort estimation. Another systematic performance evaluation study for software effort estimation accuracy prediction of ML techniques is presented in [2].

A different prediction technique has been presented by Nassif et al. [10]. They proposed an approach called regression fuzzy logic that is based on fuzzy logic models and regression analysis. Also, Fadhil et al, [11] used swarm intelligence techniques from the AI field. They applied two models based on dolphin algorithm and the hybrid dolphin and bat algorithm.

Rai et al. [12] proposed a hybrid model, based on team size using Support Vector Regression (SVR) and constructive cost model (COCOMO) approaches. Van Hai et al. [13] proposed a model called effort estimation using machine learning applied to the clusters (EEAC). The goal of the model is to evaluate the influence of data clustering on software development effort estimation.

III. Proposed Software Cost Prediction Model

This work proposes an evaluation framework to evaluate the effectiveness of using basic machine learning and ensemble classifiers, as well as, feature selection algorithms to build a software cost estimation model. Fig. 1 depicts the proposed prediction framework. As shown in Fig. 1, the first prediction model is set with the basic standalone machine learning algorithms, while the second model is built using the stacked ensemble approach. Both models are evaluated using the full dataset and with a set of the selected features. The purpose of introducing feature selection to the proposed prediction models is to extract the most relevant features from the dataset. Since, the redundant and irrelevant features increase the data dimensionality without adding new information to the dataset. This could negatively affect the performance of the prediction models.

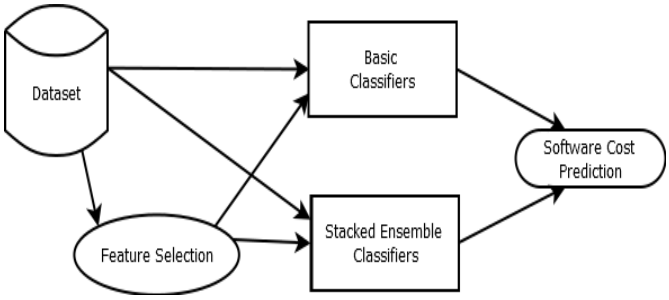

Fig. 1. The proposed software cost prediction framework.

Four machine learning algorithms were applied in the prediction process. These ML algorithms are:

- The Artificial Neural Networks (ANN): ANN system considers unsupervised learning as one of the training algorithms in a command to build an unlabeled data [14]. It consists of an input layer, hidden layers, and an output layer.
- K-Star: K-star is a machine learning algorithm that uses the entropic distance from the information theory to measure the similarities among the data elements and cases. [15].
- Support Vector Machine (SVM): SVM uses Sequential Minimal Optimization (SMO) algorithm, which transfers all nominal attributes and null values into binary ones. Then, the algorithms try to identify a margin that divides the data into different classes [16].
- Random Forest (RF): RF works by building multiple decision trees and then combining their results to make predictions. Each tree is trained on a randomly selected subset of the training data and a randomly selected subset of the features. By doing this, the algorithm can reduce errors and improve accuracy. To make a prediction, the algorithm takes in a set of features and passes them down each of the decision trees in the forest. Then it combines the results of all the trees to arrive at a final prediction [17].

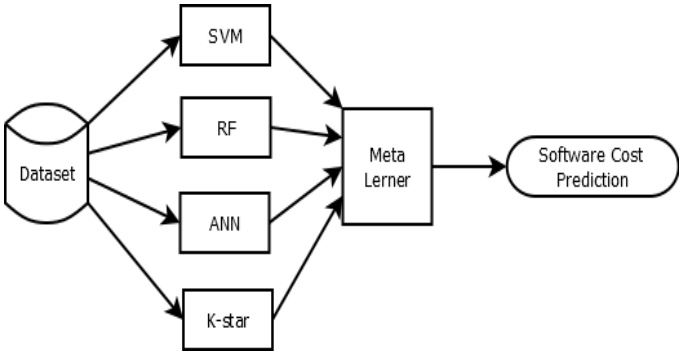

Fig. 2. The proposed ensemble stacked cost prediction classifier.

Ensemble learning is a method that combines more than one learning classifiers. Combining machine learning classifiers is primarily done to reduce risks and errors associated with employing a single classifier [18]. Additionally, ensemble learning enhances prediction

performance by balancing out the shortcomings of a single classifier. Additionally, issues with training the classifier may arise due to the size of the dataset. For instance, using a single classifier on a sizable dataset is impracticable. The dataset should therefore be divided into subgroups, with each subset being used to train a different classifier. Additionally, ensemble learning might be able to alleviate issues brought about by utilizing a tiny dataset [19]. In ensemble learning, different classifiers can be added using Stacking, Bagging, and Voting approaches. In this paper, stacking ensemble approach is used to build different software cost prediction model.

Stacking ensemble classifier is one of the most used approach in ensemble learning to combines multiple classification algorithms. Fig. 2 shows the basic levels of the used stacking ensemble classifiers. The learning process of stacking learning consists of two levels. The first level combines different machine learning classifiers $C = \{C_i(s, f), i = 1, \dots, I\}$, which are called base classifiers. Each base classifier utilizes the training set f. In the second learning level, which is the meta-learning process, a single machine learning classifier μ uses the outputs on the base classifiers as an input to generate the ensemble learning final prediction [20], Therefore:

$$j_{stack}(s) = \mu(s, C) \qquad (1)$$

where $j_{stack}(s)$ is the final stacking prediction of the input s. As shown in Fig. 2, the used four basic ML classifiers are combined in the first learning level of the proposed ensemble classifier. For the meta learning level, each basic ML classifier is used as based meta-learner classifier to create different stacked ensemble prediction model. The goal is to evaluate all possible combinations of these basic ML classifiers. As a results, the generated ensemble classifiers are:

- ANN-based Stacked classifier
- K-star-based Stacked classifier
- SVM-based Stacked classifier
- RF-based Stacked classifier

All previous basic and ensemble software cost prediction models are evaluated using the full features of software projects, as well as, a set of selected features. In this paper the wrapper method algorithm is used by applying the Classifier Subset Evaluation (CSE) technique [21].

CSE approach is a popular feature selection technique that evaluates the performance of a specific classifier for each subset of features. Fig. 3 shows the basic steps for the used CSE approach. It starts by creating all possible subsets of software features. Then, randomly select a set of features to evaluate it using the target software cost prediction model. This process is repeated until achieving the optimal accuracy for this model. Once the best feature subset is recognized, it is used for the final model training process.

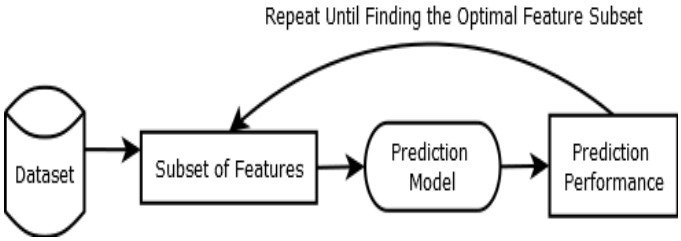

Fig. 3. The used wrapper CSE feature selection approach.

IV. Used Dataset

Due to the lack of software expense data, gathering public software effort data is a difficult undertaking. This is owing to the fact that software companies generally keep software cost data private. Usp05-tf, a publicly accessible dataset for empirical software engineering data, is made available online through the promise online repository (http://tunedit.org/repo/PROMISE/EffortPrediction), which was used in this study. Data from 76 university students' software projects are included in this dataset. Every project has 13 attributes. A list of these attributes is shown in Table I.

TABLE I. The 13 Attributes of the Studied Projects

#	Project attributes	Description	Values
1	IntComplx	The complexity of the internal project calculations	1 (lowest) to 5 (highest)
2	DataFile	Total number of accessed data files	Positive integer
3	DataEn	The number of entry data items	Positive integer
4	DataOut	The number of output data items	Positive integer
5	Lang	The used programming language	C++, Java, HTML, etc.
6	UFP	Unadjusted Function Point Count	Positive integer
7	Tools	The used platforms and tools	VJ++, Delphi, Junit, etc
8	ToolExpr	The experience level of the developer team	Range of number of months, e.g. [3, 7]
9	AppExpr	The applications experience level	1(lowest) to 5 (highest)
10	DBMS	The used database system	SQLServer, Oracle, MySQL, etc.
11	TeamSize	The size of the developer team	Range of min-max number of developers, e.g. [3, 6]
12	AppType	The used system architecture	B/S, C/S, Centered, etc.
13	Effort	The actual effort (in hours) expended on implementation tasks by all participating developers	Positive float

The features listed in Table I are not all numerical. This is crucial to verifying the machine learning approach's capacity for learning. The historical data is utilized as a learning tool to anticipate the work required for future software systems.

V. Experimental Results

The evaluation criteria and the results of the experiments are discussed in the following two sections.

A. Evaluation Criteria

In order to assess the performance of the prediction algorithms, five statistical criteria were used. The following is a description of these criteria:

- Statistics Kappa (KS)
- Mean Absolute Error (MAE)
- Root Mean Square (RMSE)
- Error in Relative Absolute (RAE)
- Root Relative Squared Error (RRSE)

Statistics Kappa (KS) is the degree of agreement between the classifier's output and the actual classification is measured by KS. KS values vary from 0 to 1. The closer to 1 KS gets, the better. The following equation is used to compute KS:

$$KS = PA - PC / (1 - PC)$$

where PC is the percentage of agreement by chance and PA is the percentage of actual agreement.

Mean Absolute Error (MAE) calculates how well the actual classification matches the anticipated classification. The value of MAE is calculated using the formula below:

$$MAE = \sum_{n=1}^{m} | a_n - a`_n |$$

where n = 1 through m, m is the total number of instances, a_n is the actual reuse output, and $a`_n$ is the predicated reuse output. The performance of the prediction model is good if the MAE value is low.

Root Mean Square (RMSE) is difference between data that was expected and actual data is measured by the quadratic mean known as RMSE. The accuracy of the model is inversely correlated with the RMSE. High precision is indicated by a low RMSE score. The error value can be calculated using the formula below:

$$RMSE = \sqrt{\frac{1}{m} \sum_{n=1}^{m} (a_n - a`_n)^2}$$

When n ranges from 1 to m, a_n represents the observed values, and $a`_n$ represents the anticipated values.

Error in Relative Absolute (RAE) is calculated by dividing the total absolute error by the total absolute error of the trivial model, RAE normalizes the total absolute error. RAE is obtained by the following equation:

$$RAE = \sum_{n=1}^{m} | a_n - a`_n | / \sum_{n=1}^{m} | a_n - a``_n |$$

where a_n is the predicted value, $a`_n$ is the target value, $a``_n$ is the mean of a_n, m is the number of instances, and n = 1 through m.

Root Relative Squared Error (RRSE) is calculated by dividing the total relative error of the naive model by the square root of the total relative error. Equation used to calculate RRSE is as follows:

$$RRSE = \sqrt{\sum_{n=1}^{m} (a_n - a`_n)^2 - \sum_{n=1}^{m} (a_n - a``_n)^2}$$

where a_n is the predicted value, $a`_n$ is the target value, $a``_n$ is the mean of a_n, m is the number of instances, and n = {1,2,...,m}.

B. Experimental Results

The experiments were conducted using WEKA 3, a machine learning software platform written in Java (https://www.cs.waikato.ac.nz/ml/weka/). The ten folds cross-validation training technique was used to evaluate all software cost prediction models. In this technique, the training and testing the prediction models is repeated in iterations. Moreover, the dataset is divided equally into ten subsets called folds. In each iteration, the prediction model uses nine folds for training and one-fold for testing. This process is ended when the classifier tests all the dataset.

Table II shows the selected features of software projects after applying the feature selection approach for each base classifier. Number of the selected feature for K-star, RF, ANN, and SVM are 10, 5, 9, 13 respectively. The smallest number of selected features was produced with the RF classifier while the biggest number happed with SVM classifier.

TABLE II. The Selected Features for each Classifier

Classifier	Selected Features
K-star	ID, IntComplx, DataFile, DataEn, DataOut, UFP, Lang, ToolExpr, AppExpr, TeamSize
RF	ID, IntComplx, DataOut, Tools, TeamSize
ANN	ID, IntComplx, UFP, Lang, Tools, ToolExpr, AppExpr, TeamSize, Method
SVM	ID, IntComplx, DataFile, DataEn, DataOut, UFP, Lang, Tools, ToolExpr, AppExpr, TeamSize, DBMS, AppType

Table III presents the selected features of software projects using the feature selection approach for each stacking classifier. As shown in the table, the feature selection approach is more effective in reducing number of selected features needed to learn the stacking classifier than basic classifier. This can be helpful to software project engineers. It can point out a smaller set of key project features which can impact on the project's cost. In this study, number of the selected feature for stacking based K-star, RF, Stacking based ANN, and Stacking based SVM classifiers are 5, 6, 5, 9 respectively.

Fig. 4 shows the MAE values for the four basic classifiers in the used dataset before and after using the feature selection approach. As shown in the figure, the value of MAE is reduced when using the feature selection algorithm. The lowest MAE is obtained when using the RF classifier, were the MAE is

reduced from 2.5025 to 2.3154 after applying the RF on the selected feature instead of using the full features.

TABLE III. THE SELECTED FEATURES FOR EACH STACKING CLASSIFIER

Stacking classifier	Selected Features
Stacking based K-star	DataFile, DataOut, Lang, TeamSize, AppType,
RF	IntComplx, DataEn, DataOut, ToolExpr, TeamSize, Method
Stacking based ANN	IntComplx, DataEn, UFP, Lang, TeamSize
Stacking based SVM	ID, IntComplx, DataFile, DataOut, Lang, ToolExpr, TeamSize, Method, AppType

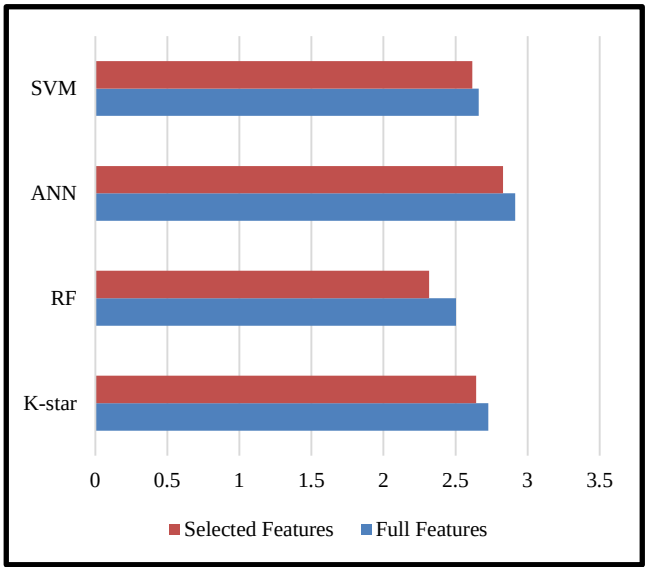

Fig. 4. MAE values before and after using feature selection.

Fig. 5 shows the values of RMSE for the four classifier on the full features and on the selected features. Same as MAE values, the RMSE values were reduced after selecting a subset of the software projects' features. The lowest RMSE value is produced when using RF classifier. For RF classifier the RMSE value was 4.8546 and 4.4979 with and without the feature selection algorithm respectively.

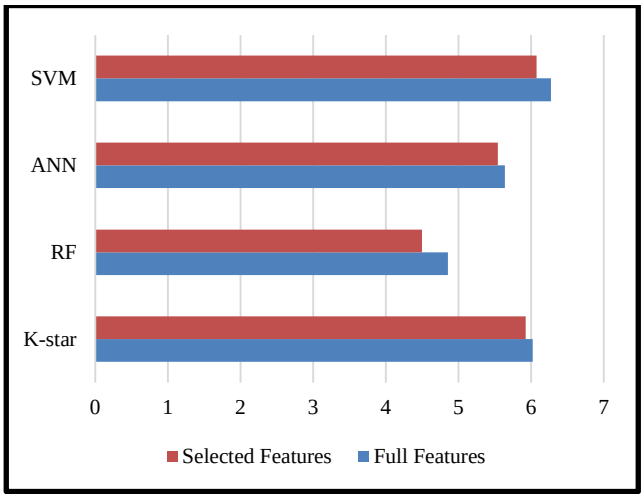

Fig. 5. RMSE values before and after using feature selection.

From pervious results, we can conclude the using the feature selection approach can increase the accuracy of software cost prediction model. Moreover, to evaluate other criteria, Table IV presents the KS, RAE, and RRSE evaluations before and after using feature selection method for the four classifiers. As shown in the table, RF is the best classifier among all others with highest KS and lower RAE and RRSE. However, feature selection approach is able to enhance the performance of all classifiers for all evaluation criteria.

TABLE IV. KS, RAE, AND RRSE EVALUATION RESULTS BEFORE AND AFTER USING FEATURE SELECTION

	KS– Full features	KS– selected features	RAE – Full features	RAE – Selected features	RRSE– Full features	RRSE– Selected features
SVM	0.7504	0.8098	44.3547 %	43.6098 %	64.6744 %	63.5698 %
ANN	0.7981	0.8098	48.8383 %	46.9337 %	64.6744 %	63.5698 %
RF	0.8441	0.8826	41.7323 %	38.6118 %	55.6778 %	51.5873 %
K-star	0.7797	0.7823	45.4795 %	44.0878 %	69.0658 %	67.9704 %

The second part of this experiment is set to evaluate the effectiveness of using stacked classifier on predicting the software cost. Fig. 6 and 7 show the MAE and RMSE evaluation values for the four possible combinations of stacked classifiers over the full features and the selected features. Stacked classifier with SVM is based classifier has the lowest MAE and RMSE with values of 2.4391 and 4.3705 respectively. After applying the feature selection approach, the MAE and RMSE values were reduced to 2.1801 and 4.0779 respectively. On the other hand, the ANN based stacked classifier has the highest MAE and RMSE values with 4.6594 and 8.1786 respectively.

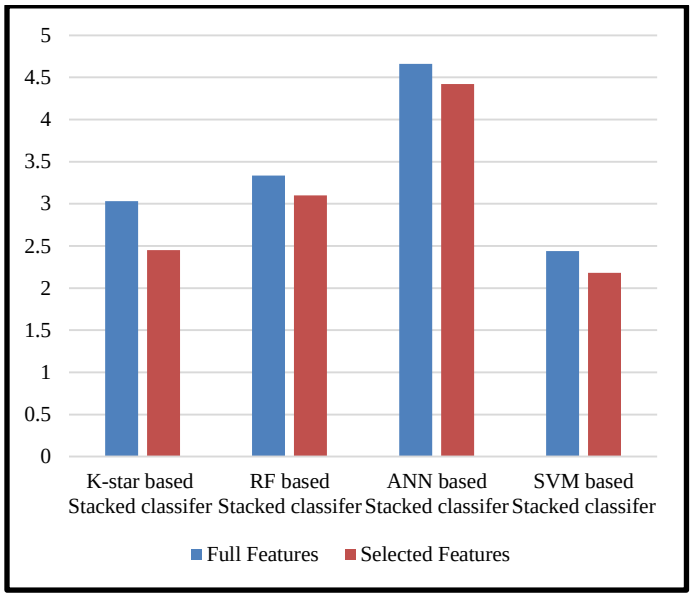

Fig. 6. MAE evaluation of stacked classifiers for full and selected features.

Results in Fig. 6 and 7 show that the using the feature selection approach can increase the accuracy of software cost prediction model based on stacked classifier. Moreover, to evaluate other criteria, Table V presents the KS, RAE, and RRSE evaluations before and after using feature selection method for the four stacked classifiers. As shown in the table, the stacked based RF classifier is the best classifier among all others with highest KS and lower RAE and RRSE. Moreover, feature selection approach is able to enhance the performance of all stacked classifiers for all evaluation criteria.

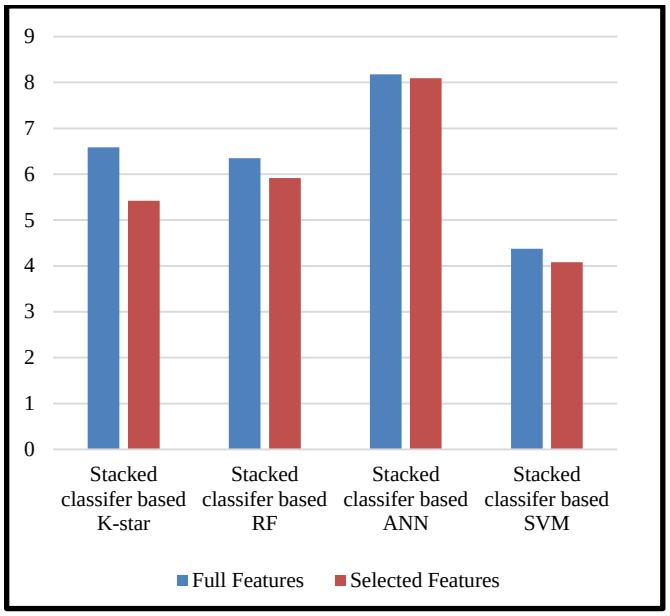

Fig. 7. MAE evaluation of stacked classifiers for full and selected features.

TABLE V. KS, RAE, AND RRSE EVALUATIONS BEFORE AND AFTER USING FEATURE SELECTION

	KS – Full features	KS– selected features	RAE – Full features	RAE – Selected features	RRSE– Full features	RRSE– Selected features
Stacked based SVM	0.7504	0.8098	44.3547 %	43.6098 %	64.6744 %	63.5698 %
Stacked based ANN	0.5168	0.5506	77.7009 %	80.4054 %	64.6744 %	63.5698 %
Stacked based RF	0.8441	0.8826	41.7323 %	38.6118 %	55.6778 %	51.5873 %
Stacked based K-star	0.6635	0.7823	50.559 %	40.8892 %	75.5606 %	62.1436 %

VI. CONCLUSIONS AND FUTURE WORK

An evaluation model has been presented for effort estimation. The model utilizes a set of machine learning algorithms and techniques to predict the effort. The model was evaluated using basic standalone machine learning algorithms and using the stacked ensemble ML approach. The evaluation is done on full features and a set of selected features that have been previously extracted using feature selection technique. Results showed that a stacked ensemble classifier with feature selection technique achieved higher accuracy for software cost prediction. Our future work aims to expand the evaluation process by including deep learning techniques. Another issue under investigation is the utilization of more project attributes to enhance the prediction results.

REFERENCES

[1] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, Andrzej Kobylinski, (2018) An effective approach for software project effort and duration estimation with machine learning algorithms, Journal of Systems and Software, vol. 137, 2018, pp. 184-196.

[2] Mahmood, Y., Kama, N., Azmi, A., Khan, A.S. and Ali, M., 2022. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. Software: Practice and experience, 52(1), pp.39-65.

[3] X. Deng, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," Information Sciences 340, 2016.

[4] Rankovic, N., Rankovic, D., Ivanovic, M. and Lazic, L., 2021. A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays. IEEE Access, 9, pp.26926-26936.

[5] BaniMustafa, A., 2018, July. Predicting software effort estimation using machine learning techniques. In 2018 8th International Conference on Computer Science and Information Technology (CSIT) (pp. 249-256).

[6] AG, Priya Varshini, and Vijayakumar Varadarajan. "Estimating software development efforts using a random forest-based stacked ensemble approach." Electronics 10, no. 10 (2021): 1195.

[7] Hammad, M. and Alqaddoumi, A., 2018, November. Features-level software effort estimation using machine learning algorithms. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 1-3).

[8] M. M. Al Asheeri and M. Hammad, "Machine Learning Models for Software Cost Estimation," 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakhier, Bahrain, 2019, pp. 1-6, doi: 10.1109/3ICT.2019.8910327

[9] Mahdi, M.N., Mohamed Zabil, M.H., Ahmad, A.R., Ismail, R., Yusoff, Y., Cheng, L.K., Azmi, M.S.B.M., Natiq, H. and Happala Naidu, H., 2021. Software project management using machine learning technique—A Review. Applied Sciences, 11(11), p.5183.

[10] Nassif, A.B., Azzeh, M., Idri, A. and Abran, A., 2019. Software development effort estimation using regression fuzzy models. *Computational intelligence and neuroscience*, *2019*.

[11] Fadhil, A.A., Alsarraj, R.G. and Altaie, A.M., 2020. Software cost estimation based on dolphin algorithm. IEEE Access, 8, pp.75279-75287.

[12] Rai, P., Verma, D.K. and Kumar, S., 2021. A hybrid model for prediction of software effort based on team size. *IET Software*, *15*(6), pp.365-375.

[13] Van Hai, V., Nhung, H.L.T.K., Prokopova, Z., Silhavy, R. and Silhavy, P., 2022. Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis. IEEE Access, 10, pp.83249-83264.

[14] Dike, H.U., Zhou, Y., Deveerasetty, K.K. and Wu, Q., 2018, October. Unsupervised learning based on artificial neural network: A review. In 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS) (pp. 322-327).

[15] Cleary, J.G.; Trigg, L.E. K*: An instance-based learner using an entropic distance measure. In Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, USA, 9–12 July 1995; pp. 108–114

[16] Jan Luts, Fabian Ojeda, Raf Van de Plas, Bart De Moor,Sabine Van Huffel, and Johan AK Suykens (2010). A tutorialon support vector machine-based methods for classifica-tion problems in chemometrics. Analytica Chimica Acta,665(2):129–145.

[17] Breiman, L., 2001. Random forests. Machine learning, 45, pp.5-32.

[18] O. Sagi and L. Rokach, "Ensemble learning: A survey," WIREs Data Mining Knowl Discov, vol. 8, no. 4, Jul. 2018, doi: 10.1002/widm.1249.

[19] R. Polikar, "Ensemble based systems in decision making," IEEE Circuits and systems magazine, vol. 6, no. 3, 2006.

[20] Sagi, O. and Rokach, L., 2018. Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), p.e1249.

[21] Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. Artificial intelligence, 97(1-2), 273-324.