ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 10 N. 3 (2021), 227-240
eISSN: 2255-2863
DOI: https://doi.org/10.14201/ADCAIJ2021103227240

An Empirical Analysis on Software Development Efforts Estimation in Machine Learning Perspective

Israr ur Rehman[a], Zulfiqar Ali[b], and Zahoor Jan[a]

[a] Department of Computer Sciences Islamia College University, Peshawar, Pakistan.
[b] Department of Computer Sciences, National University of Technology, Islamabad, Pakistan.
israr.rehman@gmail.com, zulfiqarali@nutech.edu.pk, zahoor.jan@icp.edu.pk

KEYWORDS

Machine Learning; Multilayer Perceptron; Software Efforts Estimation; Software Development

ABSTRACT

The prediction of effort estimation is a vital factor in the success of any software development project. The availability of expert systems for the software effort estimation supports in minimization of effort and cost for every software project at the same time leads to timely completion and proper resource management of the project. This article supports software project managers and decision-makers by providing a state-of-the-art empirical analysis of effort estimation methods based on machine learning approaches. In this paper five machine learning techniques; polynomial linear regression, ridge regression, decision trees, support vector regression, and Multilayer Perceptron (MLP) are investigated for software development effort estimation by using benchmark publicly available data sets. The empirical performance of machine learning methods for software effort estimation is investigated on seven standard data sets i.e. Albretch, Desharnais, COCOMO81, NASA, Kemerer, China, and Kitchenham. Furthermore, the performance of software effort estimation approaches is evaluated statistically applying the performance metrics i.e. MMRE, PRED (25), R^2-score, MMRE, Pred(25). The empirical results reveal that the decision tree-based techniques on Deshnaris, COCOMO, China, and kitchenham data sets produce more adequate results in terms of all three-performance metrics. On the Albgreenretch and NASA datasets, the ridge regression method outperformed then other techniques except the pred(25) metric where decision trees performed better.

1. Introduction

Effort estimation is the process to realistically predict the efforts and cost based on incomplete, uncertain, and noisy data to develop or maintain software (Leung and Fan, 2002). It plays a vital role in the design of project plan, budget allocation, investment analysis, and devising pricing process

Table 1: Basic COCOMO Model

Model Name	Effort(E)	Time(D)
Organic	$E = 5.2(KLOC)^{1.50}$	$D = 2.5(E)^{0.38}$
Semi-Detached	$E = 3.0(KLOC)1.12$	$D = 2.5(E)^{0.35}$
Embedded	$E = 3.6(KLOC)1.20$	$D = 2.5(E)^{0.32}$

(Huang and Chiu, 2009). Ontime delivery of software products, within the availabale budget, to meet an acceptable quality level always remains a key concern of almost all stakeholders of the software industry. The trustable and accurate estimation of software development efforts and cost can help to allocate resources appropriately and to construct an acceptable

schedule during the project planning phase. Underrating the needed software development effort and cost compromises the software quality and hence eventually results in a negative impact on the company's business reputation. An accurate estimation of software size, effort, cost, quality, and risk are the major concerns in software project management. Following are the principal challenges faced by the software estimation process (Tosun et al., 2009) 1) the nonlinear relationship between software output metrics and contributing factors; 2) the uncertain and stochastic behavior of software metrics measures; 3) the difficulty to assemble both expert knowledge and numerical project data in one model. Due to the significant importance of software efforts and cost estimation many techniques have been proposed in the literature (Albrecht and Gaffney, 1983) (Li et al., 2009). Generally, software estimation is made by different methods including Expert judgment, algorithmic effort estimation, and estimation by analogy. (Boehm et al., 1995). Expert judgment depends on the accumulated experience of professionals while algorithmic effort estimation is based on data analysis techniques to make the parameters-based effort estimation models, such as the constructive cost model (COCOMO) (Benediktsson et al., 2003). In analogy, method estimation is made by comparing the software project efforts with similar projects developed previously in history. Different software cost estimation models have been proposed to assist a project manager to make accurate and lucrative decisions (Boehm et al., 1995). Constructive Cost Model (COCOMO) (Benediktsson et al., 2003) is a common mathematical method for software effort estimation. It is based on 63 software projects which help to define mathematical equations for estimating development time, effort, and maintenance effort. Generally, a COCOMO model can be defined as:

$$E = x(KLOC)^{y} \tag{1}$$

where *E*describe the software effort in term of man per month while x and y are the constants which rely on the class of different software projects. The *KLOC* stands for Kilo Line of Code, which contains all instruction written during the implementation phase (Menzies et al., 2005). According to the COCOMO model, the software projects can be divided into three categories, based on the complexity level, such as organic, semidetached, and embedded. These models show some nonlinear attributes as explained in Table 1.

Other well known models for software effort estimation are given in Table 2. These models have been formed by analysing a huge number of delivered software projects from different organizations. (Choudhary, 2010). The soft computing techniques can be used to determine the effort estimation model (Mittal and Bhatia, 2007). (Sheta, 2006) used Neural Networks (NNs) and Fuzzy Logic (FL) to build a software effort

Table 2: Other Effort Estimation Models

Model Name	Equation
Halstead	$E = 5.2(KLOC)^{1.50}$
Walston-Felix	$E = 0.7(KLOC)^{0.91}$
Bailey-Basili	$E = 5.5 + 0.73(KLOC)^{1.16}$
Doty(for $KLOC > 9$)	$5.288(KLOC)^{1.047}$

estimation model. Similarly, Neural Network (NN) and Linear Regression (LR) were adopted to estimate the efforts in the early stages software life cycle (Kaur et al., 2010). (Wen et al., 2009) explored multiple data sets with promising results for software development effort estimation. A survey was made by (Kocaguneli et al., 2011) in which neural network is used for effort estimation models. Fuzzy logic and neural networks were used for software engineering project management in (Musílek et al., 2000). A fuzzy COCOMO model was developed in (Ryder, 1995). Nowadays, several questions have been asked about the influence of applying Soft Computing and Machine Learning methods to overcome the effort estimation issues.

Stefan Wagner and Melanie Ruhe provided the review on the impact of productivity factors in software development in (Wagner and Ruhe, 2018). Chamkaur Singh et al. proposed an efficient swarm intelligence-based approach for software maintenance effort estimation using particle swarm optimization in (Singh et al., 2019). Ali Bou Nassif et al exploited Regression and Fuzzy based models for software development effort estimation in (Nassif et al., 2019). P. Suresh Kumar et al. provided the survey on the application of neural networks and deep learning for the estimation of software efforts in (Kumar et al., 2020). Assia Najm et al. contributed by providing the review on decision tree-based software development effort estimation in (Najm et al., 2020).

In this work, we have used different machine learning (ML) techniques such as polynomial regression, ridge regression, decision trees, support vector regression, multilayer perceptron to address the problems of nonlinearity and uncertainly in software efforts and cost estimation. The main goal of this study is to investigate the validity of these techniques to predict software effort estimation as an alternative to traditional estimation models. The empirical performance of machine learning methods for software effort estimation is investigated on seven standard data sets i.e. Albretch (Albrecht and Gaffney, 1983), Desharnais (Desharnais, 1989), COCOMO81 (Boehm, 1984), (de Barcelos Tronto et al., 2008), NASA (Menzies et al., 2005), Kemerer (Li et al., 2008), China (Menzies et al., 2013) and Kitchenham (Kitchenham et al., 2002). Furthermore, the performance of machine learning-based software effort estimation approaches is evaluated statistically by applying the performance metrics i.e. MMRE, PRED (25), and R2-score. All the implementation and testing of the proposed work and the compilation of the results have been done in python. The contemporary literature shows the mainly usage of the stated approaches due to which these machine learning approaches are selected in this empirical study for effort estimation purposes.

This research paper contributes in multi-folds; by providing the review of machine learning-based software effort estimation in Section 2. Section 3 provides the performance metrics applied for the evaluation of software effort estimation approaches. In section 4, datasets are described that are used for the empirical analysis of effort estimation state-of-the-art methods. Section 5, provides the performance analysis of selected software effort estimation methods on the given data sets in terms of statistical measures. The last section of the article concludes the empirical study work under the focus.

2. Machine Learning based Software Effort Estimation

This section provides a survey of state-of-the-art machine learning-based effort estimation approaches proposed for software development.

2.1 Polynomial Liner Regression

Regression techniques predict the real-valued output. Regression is the primary and more often used type of predictive investigation of supervised and unsupervised data. Regression measures are responsible for explaining the relationship between dependent and independent variables. The dependent variable is the real-valued output and the independent variable map the one or more features of the model. The most simple type of the regression model containing two variable, one dependent variable, and one independent variable is shown by the equation

$$y = \theta_0 x + \theta_1 \tag{2}$$

where x and y are the input and output (dependent) while θ_0 and θ_1 represent the slope and intercept respectively. Polynomial regression allows us to use the mechanism of linear regression to fit very complicated and non-linear function. In polynomial regression we can define new features from the existing features which might actually get a better model. Let x_1, $x_2$and $x_3$represent different features then a typical polynomial regression can be given as:

$$y = \theta_0 x_1 + \theta_2 x_1 + \theta_2 x_3 + \theta_3 x_1 x_2 + \theta_4 x_2 x_3 + \theta_4 x_1^2 \tag{3}$$

2.2 Ridge Regression

Ridge Regression is a method responsible for analyzing the data which endure from collinearity. Collinearity, elaborate the near-linear association among all the independent variables of the model. Collinearity goes to false predictions of the coefficients. collinearity also increases the standard errors and decreases the partial t-tests which degrade the prediction capability of the model. Least squares predictions become unbiased When collinearity took place but the variances of the least square predictions are big enough so they could be away from the actual value. Ridge regression reduces the standard errors when a degree of bias is added to the estimates (Regression,).

2.3 Decision Trees

Decision Trees are the type of supervised learning responsible for the regression and classification of data. The major objective of the Decision Tree is to build a model which has the capability of prediction. Decision Trees learn decision rules The goal is to create a model that predicts the value of a target variable by learning simple decision rules deduced implicitly from the featured data.

The decision tree-like directed trees contain nodes. The node with no incoming edge is called the root node and each of the other nodes consists of only one incoming edge. Inner node is the node that has an outgoing edge and all other nodes are called leaves also called the terminal. According to a particular discrete function of the input attributes, each inner node split the instance space into multiple subspaces. Mostly every test asses a single attribute such that the instance space is divided depending on the attribute's value. When the attributes are numeric then a range is referred by the condition. every terminal is allocated to a class that represents the most suitable target value. Decision trees use if-then-else rules for the approximation of the sine curve.

From a given dataset a decision tree is automatically constructed by algorithms that work as an inducer for decision trees. the ultimate objective is to come up with an optimized decision tree having minimized generalization error. Decision tree inducers are algorithms that automatically construct a decision tree from a given dataset. Typically the goal is to find the optimal decision tree by minimizing the generalization error. There are also some other target functions like minimization of the number of nodes or minimization of the average depth, which can also be defined with the help of decision trees (Maimon and Rokach, 2005).

2.4 Support Vector Regression

Support Vector Regression (SVR) is a variant of support vector machine (SVM) used to predict or estimate a continuous output value. SVR is responsible for minimization of the empirical error and maximization of the geometric margin at the same time (Smola and Schölkopf, 2004).

Suppose we have data $(x_1, y_1),(x_2, y_2)...,(x_n, y_n)$ where $x_i \in \Re^d$ which indicate input vector and $y_i \in \Re$ indicate the output value. The aim is to define a function $f(x)$ that has only deviation from the actual output value y_i for all the data, and at the same time the function $f(x)$ is as flat as possible as shown below:

$$f(x) = \theta_0 x \ + \ \theta_1, \theta_0, \theta_1 \ \in \Re \tag{4}$$

The flatness of the function $f(x)$ means to find the smallest value of θ_0 such that:

$$\textit{Minimize} \quad \frac{1}{2}\|\theta_0\|^2 \quad s.t. \begin{cases} y_i - \theta_0 x_i + \theta_1 \leq \epsilon \\ -y_i + \theta_0 x_i + \theta_1 \leq \epsilon \end{cases} \tag{5}$$

Occasionally we want to allow some errors are allowed which are comparable in a certain respect to the "soft margin" loss function to deal with impracticable constraints of the optimization problem. The constraints I be defined as:

$$\textit{Minimize} \ \frac{1}{2}\|\theta_0\|^2 \ + \ C \sum_{i=1}^{l} \left(\xi_i + \xi_i^*\right)$$

$$s.t \begin{cases} y_i - \theta_0 x_i - \theta_1 \leq \epsilon + \xi_i \\ -y_i + \theta_0 x_i + \theta_1 \leq \epsilon + \xi_i^* \\ \xi_i + \xi_i^* \geq 0 \end{cases} \tag{6}$$

$$|\xi|_\epsilon \ = \begin{cases} 0; if \ |\xi| \leq \epsilon \\ |\xi| - \epsilon; otherwise \end{cases} \tag{7}$$

The main idea of support vector regression is to reduce the objective function considering both the norm of weight vector $\theta_0$and the losses evaluated by the variables ξ and $\xi*$.

2.5 Multilayer Perceptron

The programing paradigm that mimics the microstructure of the brain is called Neural Network which is the subfield of Artificial Intelligence. The neural network can be applied to several problems varies from simple like pattern recognition to more complex like symbolic manipulation. The Multilayer Perceptron (MLP) belongs to the class of feed-forward neural networks. MLP comprises at least three layers of nodes, an input layer, one or more hidden layers, and an output layer. The number of hidden layers varies from one to multiple. Each node of MLP represents a neuron having a non-linear activation function, excluding the input nodes. Backpropagation techniques are used by MLP for training. MLP can be applied to solve several different problems like regression, classification, interpolation, and pattern recognition (Noriega, 2005).

Back-propagation involves two steps. In the first step, it performs feeding the input layer and promulgating forward through the network to produce the predicted output. The predicted output is then mapped to the known output and the error is calculated. In a second step, the calculated error is used to adjust the weights from the hidden to the output neurons. The weights from the input to the hidden neurons are also adjusted by back-propagating the error. In MLP the value of weights represents the knowledge which is gained from the environment (Haykin, 1999).

3. Performance Metrics

The mean magnitude of relative error (MMRE), PRED (0.25) and R^2-score are used to evaluate the performance of different techniques on different datasets.

3.1 Mean Magnitude of Relative Error (MMRE)

Mean Magnitude of Relative Error (MMRE) gives the average absolute difference between the predicted values and actual targets. It can be defined as:

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} \frac{\left| Y_i - \hat{Y}_i \right|}{\hat{Y}_i} \tag{8}$$

where Y_i is the actual effort value of ith project and $\hat{Y}$i is the predicted value while n represents the number of projects. The smaller value of MMRE means good prediction and vice versa.

3.2 PRED

PRED (25) represents the percentage of predictions that fall within 25% of the actual target value.

3.3 R^2

R^2 is the extent of fluctuation of the dependent variable which is able to be predicted from the independent variable or variables. Formally, it can be define as:

$$R^2 = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO} \tag{9}$$

where *SSR* is the regression sum of error, *SSE* is the sum of square error and *SSTO* is the total sum of squares. Let Y_i, $\hat{Y}_i$ and $\bar{Y}$represent the target value, predicted value and mean of the target values respectively. Then the *SSR*, *SSE* and *SSTO* can be defined as:

$$SSR = \sum_{i=1}^{n} \left(\hat{Y}_i - \bar{Y} \right)^2 \tag{10}$$

$$SSE = \sum_{i=1}^{n} \left(Y_i - \hat{Y} \right)^2 \tag{11}$$

$$SSTO = \sum_{i=1}^{n} \left(Y_i - \bar{Y} \right)^2 \tag{12}$$

Note, that the values of R^2 lie in a range of zero and one. One means 100% accurate prediction and vice versa.

4. Datasets Description

Different dataset such as: Albrecht (Albrecht and Gaffney, 1983), Desharnais (Desharnais, 1989), COCOMO81 (Boehm, 1984), NASA (Menzies et al., 2005), Kemer (Li et al., 2008), China (Menzies et al., 2013), and Kitchenham (Kitchenham et al., 2002) are used to train and validate the results of different techniques.

4.1 Albrecht Dataset

This dataset contains the data of 24 projects where eighteen projects are written in Common Business-oriented Language, four projects are written in PL/I, and two in DMS languages respectively. This dataset contains six independent features such as input, output, query, file, function points, and the total number of lines of source code. The person-hours, in 1000h, is consider as a dependent feature (Albrecht and Gaffney, 1983).

4.2 Desharnais dataset

This dataset contains 10 input features and one output feature of 81 different projects. Four projects of this dataset contain missing values of the feature which are not included in the dataset. The independent features include Team-Exp, Manager-Exp, Year-End, Length, Transactions, Entities, Points-Adjust, Envergure, Points-NonAjust, and Language. Person hours are considered as a dependent feature which is recorded in 1000h (Desharnais, 1989).

4.3 COCOMO81 Dataset

It contains data of 63 software projects of different types which include business, scientific and system projects. There are 16 independent variables: acap, pcap, aexp, modp, tool, vexp, lexp, sced, stor, data, time, turn, virt, cplx, rely, loc, and effort. Every attribute of the project has a level of influence

on effort estimation. Therefore all attributes are categorized accordingly. The range of the levels is V-Low, Low, Normal, High, V-High, and E-High. A numerical value is associated with all of these levels (Boehm, 1984) (de Barcelos Tronto et al., 2008).

Apart from the line of code (loc) and effort the remaining features fall into three groups: 1) having a positive correlation with effort (stor, data, time, turn, virt, cplx and rely), 2) having a negative correlation with effort (acap, pcap, aexp, modp, tool, vexp, lexp),3) having U-shaped correlation with effort such as schedule constraint.

4.4 Nasa Dataset

The NASA dataset has 93 NASA projects from different centers, this dataset consists of 24 features with 15 standard COCOMO discrete attributes in the range VeryLow to ExtraHigh. these features are acap, pcap, aexp, modp, tool, vexp, lexp, sced, stor, data, time, turn, virt, cplx, rely, seven others describing the project like project name, record number, category of application, flight or ground system, which nasa center, year of development, development mode. Other features are lines of code measure, And the actual effort in person-months (Menzies et al., 2005).

4.5 Kemer Dataset

The Kemerer dataset is an organization-based data from Kemerer's work (Li et al., 2008). This dataset has data from an individual company. the data is of 15 business data processing projects. Every project has six input features i.e. programming language, hardware, duration, KSLOC, Adj-FP , and RAW-FP. LOOCV is used for the experiments because of the small size of the dataset and to enable comparison to recent results reported by (Li et al., 2008) using the same experimental framework.

4.6 China Dataset

The China dataset is a newly developed dataset used for the estimation of software development efforts. It contains 499 software projects described by 15 input features i.e. Adjusted Function points, Input, Output, Enquiry, File, Interface, Added-functions, Changed-functions, Deleted-function, PDR-AFP, PDR-UFP, NPDR-AFP, NPDR-UFP, Resource level, Development type, and two output features are Duration and Effort. All features of this dataset have numerical values. (Menzies et al., 2013).

4.7 Kitchenham Dataset

Kitchenham dataset (Kitchenham et al., 2002) consists of 145 examples and 10 features including project client code {1,2,3,4,5,6}, project type {A,C,D,P,Pr,U}, start date, duration, actual efforts, adjusted function, estimated completion date and etc. For the evaluation our techniques we used the numeric features only.

5. Results and Discussion

In this article we use seven benchmark software effort data sets to evaluate the proposed method: Albretch (Albrecht and Gaffney, 1983), Desharnais (Desharnais, 1989), COCOMO81 (Boehm, 1984; de Barcelos Tronto et al., 2008), NASA (Menzies et al., 2005), Kemerer (Li et al., 2008), China

Table 3: Results for Albretch dataset

Method	R^2		MMRE		Pred	
	Training	Testing	Training	Testing	Training	Testing
Polynomial Linear Regression	1.0	0.845536	1.279818	0.653455	100.0	66.666666
Ridge Regression	0.998136	0.893205	0.111712	0.506187	88.888888	66.666666
Decision Trees	1.0	0.688425	0.0	1.293556	72.222222	33.333333
SVR	-0.10424	0.024916	1.713777	0.886663	27.77777	16.666666
MLP	0.994552	-2.60918	0.819987	0.69916	72.222222	50.0

Table 4: Results for Desharnais dataset

Method	R^2		MMRE		Pred	
	Training	Testing	Training	Testing	Training	Testing
Polynomial Linear Regression	1.0	-2.9339	7.66318	2.76589	100.0	14.2857
Ridge Regression	0.99802	-2.13327	0.01589	2.45716	100.0	14.2857
Decision Trees	1.0	0.19974	0.0	1.09225	100.0	28.57142
SVR	-0.12242	-0.01789	0.68479	1.30638	26.66666	28.5714
MLP	0.26782	0.04261	0.96377	1.74024	25.0	28.5714

(Menzies et al., 2013) and Kitchenham (Kitchenham et al., 2002). The datasets contain data from a system called a database-oriented software system which is developed using a specific 4GL tool suite. These data sets are used in some articles to explore the performance of techniques used for the estimation of software development effort. Five machine learning techniques i.e. polynomial linear regression, ridge regression, decision trees, SVR and MLP were applied to the benchmark datasets. The performance of the proposed techniques for software development effort estimation is measure on three metrics i.e. MMRE, pred(25), and R2-score. All the proposed machine learning techniques for software development effort estimation are implemented and tested in python. Results show that All of the five applied machine learning techniques used for the estimation of software development effort outperformed the traditional prediction models. However, variation occurs in the results when we compare the results of the proposed techniques among themselves. Several factors are responsible for such variances in the performance results of the techniques like the size of the data set, the feature involved in each dataset, and the nature of the software project. The detailed results of each dataset for all five machine learning techniques again each performance metrics are shown in the tables discussed below.

Table 3 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on albretch dataset concerning three performance metrics i.e. R^2-score, MMRE, and pred(25). The outcomes show that the best technique for this dataset is ridge regression. Ridge

Regression technique outperforms polynomial linear regression, SVR, Decision trees, and MLP all in terms of R^2-score, MMRE, pred(25).

Table 4 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees and MLP applied on desharnais dataset concerning three performance metrics i.e. R^2-score, MMRE and pred(25). The results show that the best technique for this dataset

Table 5: Results for COCOMO81 dataset

Method	R^2		MMRE		Pred	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**
Polynomial Linear Regression	1.0	-6.944297	2.199594	5.147439	100.0	0.0
Ridge Regression	0.999975	-7.002276	0.017858	5.004696	100.0	0.0
Decision Trees	1.0	0.1045272	0.0	1.594110	100.0	6.25
SVR	-0.127583	-0.017450	3.184352	2.908661	6.382978	12.5
MLP	0.310911	-8.351732	19.029081	19.175637	6.382978	6.25

Table 6: Results for NASA dataset

Method	R^2		MMRE		Pred	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**
Polynomial Linear Regression	0.99997	-29968.80	0.03532	120.628	94.20289	4.16666
Ridge Regression	0.99986	0.521286	0.051951	0.9028	95.65217	29.1666
Decision Trees	0.9999	0.01302	0.002359	1.22855	100.0	29.1666
SVR	-0.08386	-0.083716	3.59744	2.25532	13.04347	8.3333
MLP	0.23409	0.12925	3.44438	2.50700	8.69565	12.5

is decision trees. Decision trees outperform polynomial linear regression, ridge regression, SVR, and MLP all in terms of R^2-score, MMRE, pred(25).

Table 5 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on COCOMO81 dataset for three performance metrics i.e. R^2-score, MMRE, and pred(25). The results show that the best technique for this dataset is decision trees. Decision trees outperform polynomial linear regression, ridge regression, SVR, and MLP all in terms of R^2-score, MMRE, pred(25). Except for pred(25) in testing for SVR which is 12.5 is better than decision trees which are 6.25.

Table 6 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on NASA dataset concerning three performance metrics i.e. R^2-score, MMRE, and pred(25). The results show that the best technique for this dataset Ridge Regression. Ridge Regression outperforms polynomial linear regression, decision trees, SVR, and MLP all in terms of R^2-score, MMRE, pred(25). Except for pred(25) wherein training decision trees perform better than ridge regression but in testing

the performance of both the methods is equal.

Table 7 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on kemer dataset concerning three performance metrics i.e. R^2-score, MMRE, and pred(25). The results show that the best technique for this dataset is MLP. MLP outperforms polynomial linear regression, Ridge Regression, decision trees, and SVR, all in terms of R^2-score, MMRE, pred(25).

Table 8 describe the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on china dataset concerning three performance metrics i.e. R^2-score, MMRE, and pred(25). The results show that the best technique for this dataset is decision trees. decision trees outperform polynomial linear regression, ridge regression, SVR, and MLP all in terms of R^2-score, MMRE, pred(25). Except for MMRE and pred(25) in testing data, polynomial, and ridge regression perform better than decision trees.

Table 9 describes the results for all techniques including polynomial linear regression, ridge regression, SVR, Decision trees, and MLP applied on Kitchenham dataset for three performance metrics i.e. R^2-score, MMRE, and pred(25). The results show that the best technique for this dataset is decision trees. decision trees outperform polynomial linear regression, ridge regression, SVR, and MLP all in

Table 7: Results for Kemer dataset

Method	R^2		MMRE		Pred	
	Training	Testing	Training	Testing	Training	Testing
Polynomial Linear Regression	1.0	-5.7338	3.35587	2.33084	100.0	0.0
Ridge Regression	0.97527	-3.8487	0.06753	2.15077	90.9090	0.0
Decision Trees	1.0	-0.33416	0.0	3.24069	100.0	0.0
SVR	-0.02717	-0.1627	0.59547	1.91245	9.09090	25.0
MLP	0.91074	-0.1644	0.11565	1.0908	81.81818	25.0

Table 8: Results for China dataset

Method	R^2		MMRE		Pred	
	Training	Testing	Training	Testing	Training	Testing
Polynomial Linear Regression	1.0	-9.6339	4.2255	0.17015	100.0	86.400
Ridge Regression	0.99939	0.54396	0.0888	0.22285	94.65	80.800
Decision Trees	1.0	0.72909	0.0	0.4577	100.0	50.3999
SVR	-0.10324	-0.0555	1.797268	1.90826	16.04278	21.6000
MLP	0.49347	0.62603	0.73499	0.68097	32.3529	32.0

Table 9: Results for Kitchenham dataset

Method	R^2		MMRE		Pred	
	Training	Testing	Training	Testing	Training	Testing
Polynomial Linear Regression	0.999	-200.9284	0.1387	6.2888	82.40740	45.9459
Ridge Regression	0.9967	0.8806	0.2816	0.4182	62.96	40.54
Decision Trees	1.0	0.3725	0.0	0.3725	100	54.0540
SVR	-0.0238	-0.14288	4.133	5.0021	3.70370	16.216
MLP	0.3016	-0.641	4.133	5.0021	3.70370	16.216

terms of R^2-score, MMRE, pred(25). Except for R^2-score in testing data, ridge regression performs better than decision trees.

6. Conclusion

Effort estimation could have a significant impact in the success of any software development project. This research paper contributes in multi-folds; by providing the review of machine learning-based software effort estimation techniques and empirical performance analysis on public benchmark data sets. In this research work, five machine learning techniques; polynomial linear regression, ridge regression, decision trees, support vector regression, and Multilayer Perceptron (MLP) are investigated for software development effort estimation by using bench mark publicly available data sets. The Albretch, Desharnais, COCOMO81, NASA, Kemerer, China, and Kitchenham databases are used for the comparative performance analysis of machine learning methods for effort estimation. Furthermore, the performance of software effort estimation approaches is evaluated statistically applying the performance metrics i.e. MMRE, PRED (25), R^2-score, MMRE, Pred(25). The experimental results show that the decision tree-based techniques on Deshnaris, COCOMO, China, and Kitchenham Data Sets provide promising results in terms of all three-performance metrics. On the Albretch and NASA Data Set, the ridge regression method outperformed then other techniques except for the pred(25) metric where decision trees performed better. With the rapid growth in the field of Big data analytics and the internet of things (IoT), the nature of software development has been changed. In the future, we need to explore a machine learning-based recommender system for predicting software development efforts. For future research, newly proposed methods based on deep learning, swarm intelligence and fuzzy logic-based effort estimation techniques can be investigated at larger size databases.

7. Acknowledgements

The authors wish to thank the Higher Education Commission of Pakistan and the Islamia College University, Peshawar, Pakistan. This work is supported in part by a grant from the Higher Education Commission of Pakistan and the Islamia College University, Peshawar, Pakistan.

8. References

Albrecht, A. J. and Gaffney, J. E., 1983. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE transactions on software engineering*, (6):639–648.

Benediktsson, O., Dalcher, D., Reed, K., and Woodman, M., 2003. COCOMO-based effort estimation for iterative and incremental software development. *Software Quality Journal*, 11(4):265–281.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R., 1995. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of software engineering*, 1(1):57–94.

Boehm, B. W., 1984. Software engineering economics. I*EEE transactions on Software Engineering*, (1), pp. 4–21..

Choudhary, K., 2010. GA based Optimization of Software Development effort estimation. *IJCST, September*.

de Barcelos Tronto, I. F., da Silva, J. D. S., and Sant'Anna, N., 2008. An investigation of artificial neural networks based prediction systems in software project management. *Journal of Systems and Software*, 81(3):356–367.

Desharnais, J.-M., 1989. Analyse statistique de la productivitie des projects informatique a partie de la technique des point des function. *Masters Thesis University of Montreal*.

Haykin, S., 1999. Neural networks a comprehensive introduction.

Huang, S.-J. and Chiu, N.-H., 2009. Applying fuzzy neural network to estimate software development effort. *Applied Intelligence*, 30(2):73–83.

Kaur, J., Singh, S., Kahlon, K. S., and Bassi, P., 2010. Neural network-a novel technique for software effort estimation. *International Journal of Computer Theory and Engineering*, 2(1):17.

Kitchenham, B., Pfleeger, S. L., McColl, B., and Eagan, S., 2002. An empirical study of maintenance and development estimation accuracy. *Journal of systems and software*, 64(1):57–77.

Kocaguneli, E., Misirli, A. T., Caglayan, B., and Bener, A., 2011. Experiences on developer participation and effort estimation. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 419–422. IEEE.

Kumar, P. S., Behera, H., Kumari, A., Nayak, J., and Naik, B., 2020. Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. *Computer Science Review*, 38:100288.

Leung, H. and Fan, Z., 2002. *Software cost estimation*. World Scientific.

Li, Q., Wang, Q., Yang, Y., and Li, M., 2008. Reducing biases in individual software effort estimations: a combining approach. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 223–232. ACM.

Li, Y.-F., Xie, M., and Goh, T. N., 2009. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82(2):241–252.

Maimon, O. and Rokach, L., 2005. Decomposition methodology for knowledge discovery and data mining. In *Data mining and knowledge discovery handbook*, pages 981–1003. Springer.

Menzies, T., Butcher, A., Cok, D., Marcus, A., Layman, L., Shull, F., Turhan, B., and Zimmermann, T., 2013. Local versus global lessons for defect prediction and effort estimation. *IEEE Transactions on software engineering*, 39(6):822–834.

Menzies, T., Port, D., Chen, Z., Hihn, J., and Stukes, S., 2005. Validation methods for calibrating software effort models. In *Proceedings of the 27th international conference on Software engineering*, pages 587–595. ACM.

Mittal, H. and Bhatia, P., 2007. A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers. *International Journal of Computer Science and Security*, 1(4):36–47.

Musílek, P., Pedrycz, W., Succi, G., and Reformat, M., 2000. Software cost estimation with fuzzy models. *ACM SIGAPP Applied Computing Review*, 8(2):24–29.

Najm, A., Marzak, A., and Zakrani, A., 2020. Systematic Review Study of Decision Trees based Software Development Effort Estimation. *Organization*, 11(7).

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

Noriega, L., 2005. Multilayer perceptron tutorial. *School of Computing.* Staffordshire University.

Ryder, J., 1995. *Fuzzy COCOMO: Software cost estimation*. Ph.D. thesis, PhD thesis, Binghamton University.

Sheta, A., 2006. Software effort estimation and stock market prediction using takagi-sugeno fuzzy models. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 171–178. IEEE.

Singh, C., Sharma, N., and Kumar, N., 2019. An Efficient Approach for Software Maintenance Effort Estimation Using Particle Swarm Optimization Technique. *International Journal of Recent Technology and Engineering (IJRTE)*, 7(6C):1–6.

Smola, A. J. and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Tosun, A., Turhan, B., and Bener, A. B., 2009. Feature weighting heuristics for analogy-based effort estimation models. *Expert Systems with Applications*, 36(7):10325–10333.

Wagner, S. and Ruhe, M., 2018. A systematic review of productivity factors in software development. *arXiv preprint arXiv:1801.06475*.

Wen, J., Li, S., and Tang, L., 2009. Improve analogy-based software effort estimation using principal components analysis and correlation weighting. In *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific*, pages 179–186. IEEE.