International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

A Comparative Analysis of Regression Models for Software Effort Estimation

Md. Tanziar Rahman[1*], Md. Motaharul Islam[2], Ummay Salma Shorna[3]

[1]Department of Software Engineering, Nuclear Power Plant Company Bangladesh Limited, Bangladesh*
[2]Department of Computer Science & Engineering, United International University, Bangladesh
[3]Department of Statistics, Central Bank of Bangladesh

Email: [2]motaharul@cse.uiu.ac.bd, [3]ushorna@gmail.com
Corresponding Email: [1]tanzrahman3@gmail.com*

Received: 26 May 2023 **Accepted:** 14 August 2023 **Published:** 01 October 2023

Abstract**: Software Effort Estimation is the utmost task in software engineering and project management. This is important to estimate cost properly and the number of people required for a project to be developed. Many techniques have been used to estimate cost, time, schedule and required manpower for software development industries. Nowadays software is developed in a more complex way and its success depends on efficient estimation techniques. In this research, we have compared five regression algorithms on different projects to estimate software effort. The main advantage of these models is they can be used in the early stages of the software life cycle and that can be helpful to project managers to conduct effort estimation efficiently before starting the project. It avoids project overestimation and late delivery. Software size, productivity, complexity and requirement stability are the input vectors for these regression models. The estimated efforts have been calculated using Ridge Regression, Lasso Regression, Elastic Net, Random Forest and Support Vector Regression. We have compared unitedly these models for the first time as software effort estimators. R-squared Score, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are calculated for these regression models. Ridge, Lasso and Elastic Net show comparatively better results among others.

Keywords**: Software Effort Estimation, Ridge, Lasso, Elastic Net, Random Forest, Support Vector Regression.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

1. INTRODUCTION

Nowadays software effort estimation is a very exigent phase in the software development industry. In Software Development Life Cycle (SDLC), it is very essential to calculate estimated effort at the beginning of the cycle. For the planning and strategies of software development, it is required to calculate effort consciously cause overestimation and underestimation both are major deterrents in this industry. In many cases in the middle of project development, the estimation is redefined. Then both the time and cost are wasted only for lack of consciousness and technical knowledge. A good estimation can provide a quality product in due time with predefined manpower.

There are several models for software effort estimation. These include algorithmic models, expert judgment models, estimation by analogy models and soft computing models [1].
Algorithmic models are calculated by mathematical formulas which are linked with effort drivers to produce an estimation of the project and these are the most popular models in the literature. Usually, the main effort driver used in these models is software size (Function Point, Source Lines of Code) [2] which needs to be calibrated to local circumstances.

Consultation with experts regarding the effort calculation based on their experiences is related to the Expert Judgement model. In this case, the experts share their experiences and use their gathered knowledge to estimate effort. Through these models, the final estimation report can be reached in a reasonable period and this is the main advantage of this model.

If the proposed project is compared with some previous projects and its effort is estimated by observing these previous projects then it will be included as Estimation by Analogy model. All required information related to the previous projects is documented. In this case, they should have some expertise also who can contribute to comparing the proposed projects with some developed projects. This model will not be applicable for any start-up or the companies who haven't completed a good number of historical projects that might be helpful for new proposed projects.

Machine learning, neural network, fuzzy logic, genetic algorithm and hybrid models like neuro-genetic, neuro-fuzzy etc are included in the Soft Computing model. Soft Computing models are applied in two main situations.

- It can be applied as standalone models that take several inputs such as software size, complexity, requirement stability and then generate an output like software effort. These input vectors play a vital role in the development phase. Some papers like [3], [4] are available for just determining these factors for making the development activities easy.
- They can be used to determine some parameters or weights of COCOMO or any other algorithmic model's parameter and weights of function point model.

In this paper, we have implemented five regression algorithms (i.e. Ridge Regression, Lasso Regression, Elastic Net, Random Forest and Support Vector Regression) to estimate software

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

effort. We have compared these regression models based on estimated effort and some errors formula. We have used these models because in previous many machine learning models and other models have been used to estimate effort but these models are not amalgamated. Ridge, Lasso and Elastic Net regression models are very new as software effort estimation models. We observe Random Forest and Support Vector Regression are often used and that's why we have compared these two models with the other three regression models which are not used previously. We have used ISBSG release 11 as the training and testing dataset. Ridge, Lasso and Elastic Net show relatively better accuracy among them.

The main contributions of this paper are as follows:

- We predict software effort by using the five most important regression models which are not compared together in the previous analysis.
- This paper provides acceptable accuracy like R-squared score greater than 97% for Ridge and Elastic Net regression models.
- Though effort estimation depends on various parameters, we use four important parameters that can reduce complexity while estimating.
- We have compared these models with three types of accuracy measures. In our case, Elastic Net generates a very admissible R-squared score and minimum 20% better accuracy in Mean Absolute Error (MAE) but not optimal for Mean Squared Error (MSE). So, we conclude that not a single regression model is the best for all types of accuracy measures. So single regression can't be an optimal estimator if we consider multiple accuracy evaluation techniques.

Related Works

In paper [5], different machine learning algorithms have been implemented. Artificial Neural Network, Genetic Algorithm, Fuzzy Logic and other hybrid models have been used here but they don't get always reliable results for any specific algorithm. So, they conclude that no specific algorithm should not be preferred.

Paper [6] compares different algorithms (such as Multilayer Perceptron, Radial Basis Function, Support Vector Machine) on some datasets from Turkey's software industry and they also conclude that one model cannot produce the best results. In [7], they introduce a machine learning approach that has been used as a text processing model which gives a good estimation result as well. Paper [8] – [10] estimate effort by applying different Regression models including linear and non-linear approaches.

Software effort estimation can be applicable for both agile and non-agile methodologies, from [11] we can get the comparative analysis between these two development models. Various Machine learning algorithms are implemented in [12] – [14] for productivity analysis and based on different features.

The effort has been determined and analysed in [15], [16] considering different sized software and company as well. For web-based project estimation [17] Radial Basis Function (RBF) kernel technique provides better result for Support Vector Regression than others. In this paper

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

[18], Extreme Learning Machine (ELM) is used as the replacement of Linear Least Squares Regression. But only for small projects ELM has been used there.

In [19], [20], it is discussed 4 types of software effort prediction algorithm (MLP, GRNN, RBNN & CCNA) and Conclude that RBNN & CCNA give better results according to the dataset. It has been applied RBNN and Regression model [21] to estimate effort under 93 projects dataset of NASA and concludes that RBFNN provides less MRE than regression.
The paper [22] has described the fault and effort prediction of a project. They have used Rule Extraction, Support Vector Machine and also Radial Basis Neural Network for efficient effort and fault prediction of a given project.

In [23], has determined the widths of Radial Basis Neural Network performs a vital role for better result. The wrong choice of widths gives poor results. In that paper, two methods have been described for determining widths.

This paper [24] has been extended the UCP (Use Case Points) model by classifying actors into seven groups. The weight proposed for actors varies between 0.5 and 3.5. Moreover, the authors proposed four types of use cases and assigned new weights for each use case. Paper [25], authors propose RBFNN model for software effort estimation. The model is trained based on the k-mean clustering algorithm and is evaluated using the COCOMO 81 dataset. It takes too much time to converge when weights are calculated.

In our previous work [26], We have compared Radial Basis Function Neural Network, Extreme Learning Machine and Decision Tree based on estimated effort in different sized Software.

2. METHODOLOGY

We are familiarized with Linear and Logistic Regression which are implemented without Regularization. But we should use Regularization for getting a better result. In this paper, we have used Ridge, Lasso, Elastic Net Regression which are basic regression models with Regularization but still they are not used frequently. The overall idea of regression remains the same in these models.
Ridge and Lasso work by penalizing the magnitude of coefficients of features minimizing the error between actual and predicted observations. These are called ‘Regularization’ techniques. The key difference is in how they assign penalty to the coefficients are regularization and minimization objectives.

Ridge Regression performs L2 regularization and Lasso regression performs L1 regularization whereas Elastic Net uses both L1 and L2 regularization. We have also used another two important Regression algorithms, Random Forest and Support Vector Regression which are also good models for prediction analysis. These algorithms are described below in brief.
We have included pseudocodes of the regression models which have been used to estimate effort. The models need some hyperparameters to be implemented. The hyperparameters may have different values. These parameters are tuned changing the value and we have finalized the

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

value based on the implemented result of the models. Our included pseudocodes show the final value of these parameters like α (alpha) for Ridge and Lasso regression, n estimators for Random Forest regression. So, to apprise regarding the value of the hyperparameters and for understanding the implementation steps of the regression models, we have added pseudocodes. The following sub-sections are described the models and pseudocodes as well. Section 3.1, 3.2, 3.3, 3,4 and 3.5 discuss Ridge, Lasso, Elastic Net, Random Forest and Support Vector Regression respectively.

A. Ridge Regression

We mentioned before, Ridge Regression performs 'L2 regularization'. Thus, Ridge Regression optimizes the following: Objective = LR objective + α ∗ (sum of square of coefficients)

Here, LR = Linear Regression and α is the parameter that is used to balance the amount of emphasis given for minimizing LR objective vs minimizing the sum of the square of coefficients. α can have different values: When α = 0, the objective of Ridge Regression becomes the same as Linear Regression; when $\alpha = \infty$, the coefficients will be zero because of increasing the value of α, the coefficients move towards zero; when $0 < \alpha < \infty$, the coefficients will be somewhere between 0 and ones for Simple Linear Regression. That's the way α impacts the magnitude of coefficients. Any non-zero value of α gives a value that is less than Simple Linear Regression. Algorithm 1 shows the pseudocode of Ridge Regression.

Algorithm 1 RIDGE Regression

```
procedure RIDGE
LOAD TRAINING_DATA
for each dataItem in TRAINING_DATA do
TRAIN the model
ridgeRegression ← RIDGE (alpha = 1 )
Fit the model
ridgeRegression.fit(dataItem)
end for
LOAD TESTING_DATA
for each dataItem in TESTING_DATA do
PREDICT the result
effort predict ← ridgeRegression. PREDICT (dataItem)
end for
end procedure
```

B. Lasso Regression

LASSO stands for Least Absolute Shrinkage and Selection Operator. It performs 'L1 regularization' that means it adds a factor which is sum of absolute value of coefficients in the optimization objective. Thus, lasso regression optimizes as follows: Objective = LR Objective + α ∗ (sum of absolute value of coefficients)

Here, α works similarly to Ridge. In this case, α also can take various values like Ridge such that $\alpha = 0, \infty$ and any value between 0 and ∞. For these values α acts as same as Ridge. The pseudocode of Lasso implementation has been given below in Algorithm 2.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

Algorithm 2. LASSO Regression

1. procedure LASSO
2. LOAD TRAINING_DATA
3. for each dataItem in TRAINING_DATA do
4. TRAIN the model
5. lassoRegression ← LASSO (alpha = 1)
6. Fit the model
7. lassoRegression.fit(dataItem)
8. end for
9. LOAD TESTING_DATA
10. for each dataItem in TESTING_DATA do
11. PREDICT the result
12. effort predict ← lassoRegression. PREDICT (dataItem)
13. end for
14. end procedure

C. Elastic Net Regression

Elastic Net Regression is another Regression algorithm that is also a modification of Linear Regression. It is known Linear Regression is implemented without Regularization and it suffers from overfitting problem and it is not suitable for collinear data. Apart from Ridge and Lasso Regression, Elastic Net is developed which is included both L2 and L1 Regularization. As Ridge and Lasso have some limitations, we include this model in this paper to get the benefits of both Ridge and Lasso at the same time. The cost function for Elastic Net Regression is given below: Objective = LR Objective + α ∗ (sum of square of coefficients) + α ∗ (sum of absolute value of coefficients). Elastic Net is implemented using Algorithm 3.

Algorithm 3. ELASTIC NET Regression

1. procedure ELASTIC NET
2. LOAD TRAINING_DATA
3. for each dataItem in TRAINING_DATA do
4. TRAIN the model
5. elasticNetRegression ← ELASTICNET (alpha = 1)
6. Fit the model
7. elasticNetRegression.fit(dataItem)
8. end for
9. LOAD TESTING_DATA
10. for each dataItem in TESTING_DATA do
11. PREDICT the result
12. effort predict ← elasticNetRegression. PREDICT (dataItem)
13. end for
14. end procedure

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

D. Random Forest Regression

Random Forest is said as an ensemble of decision trees because it is formed by many trees which are constructed in a 'random' way. This algorithm has some benefits. It takes less training time as compared to different algorithms. It can predict accurately even for large datasets. Sometimes it maintains accuracy when a portion of the dataset is missed. Random Forest Algorithm generally gives better prediction result for Classification than Regression. But it is used in the Software Effort Estimation field as a good regressor.

Some points of Random Forest:

- Each tree is built from a different sample of rows and a different sample of features that are selected for splitting each node.
- Each tree makes its individual prediction.
- The individual predictions are then averaged to produce a single estimated result.

While implementing Random Forest Regression, the hyperparameter 'n_estimators' is tuned. We have checked from 1 to 1000 as 'n estimators' value and got optimal value 5 for 'n estimators' parameter. For this reason, the 'n estimators' value is selected here as 5. Algorithm 4 shows the implementation steps of Random Forest Model.

Algorithm 4. Random Forest Regression

1. procedure RANDOM FOREST
2. LOAD TRAINING_DATA
3. for each dataItem in TRAINING_DATA do
4. TRAIN the model
5. randomForestRegression ← RandomForestRegressor (n_estimators =5, random_state = 0)
6. Fit the model
7. randomForestRegression.fit(dataItem)
8. end for
9. LOAD TESTING_DATA
10. for each dataItem in TESTING_DATA do
11. PREDICT the result
12. effort predict ← randomForestRegression. PREDICT (dataItem)
13. end for
14. end procedure

E. Support Vector Regression

Support Vector Machine is another machine learning algorithm. It is used for both Classification and Regression problems. For regression, it is said as Support Vector Regression (SVR). SVR works with the same principles as Support Vector Machine (SVM) which is used widely for Classification. SVR works with continuous data whereas SVM works with categorical data. The main purpose of both algorithms is to reduce the error rate based on a predefined threshold. SVR is implemented with different types of kernels such as Linear,

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

Polynomial, Gaussian. We have implemented SVR with Linear Kernel cause our dataset contains a linear relationship. SVR model is implemented using Algorithm 5.

Algorithm 5 Support Vector Regression

1. procedure SVR
2. LOAD TRAINING_DATA
3. for each dataItem in TRAINING_DATA do
4. TRAIN the model
5. svrRegression ← SVR (kernel = 'linear')
6. Fit the model
7. svrRegression.fit(dataItem)
8. end for
9. LOAD TESTING_DATA
10. for each dataItem in TESTING_DATA do
11. PREDICT the result
12. effort predict ← svrRegression. PREDICT (dataItem)
13. end for
14. end procedure

Implementation

We have implemented the five most popular Regression algorithms to estimate software effort. We have used Python as our programming language. For comparison of the different errors and effort-size relationship in graphical representation, we use matplotlib library of python.

A. Dataset

In this paper, to analyse effort estimation we have used ISBSG Release 11 [1] dataset. We have used the same dataset for every Regression algorithm. We divide the dataset into training and testing datasets. We have selected 70% data for training purposes and 30% for testing purposes. Each data contains four attributes including 'Software Size' which is one of the important inputs in the software effort estimation field. Both the training and testing datasets include different-sized software from very small to large-sized software. In this dataset, software size is measured in 'UCP' instead of 'KLOC'. These both are measuring units of software size. We include data for training and testing which have software size that varies from 5 UCP to almost 4000 UCP. Besides the software size, the dataset contains other attributes as requirement stability, complexity and productivity.

B. Flow of Models

The algorithms which are used in this paper follow the common steps of flow. It contains six steps from input to getting output i.e. estimated effort. The model flow is given below in Fig. 1:

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

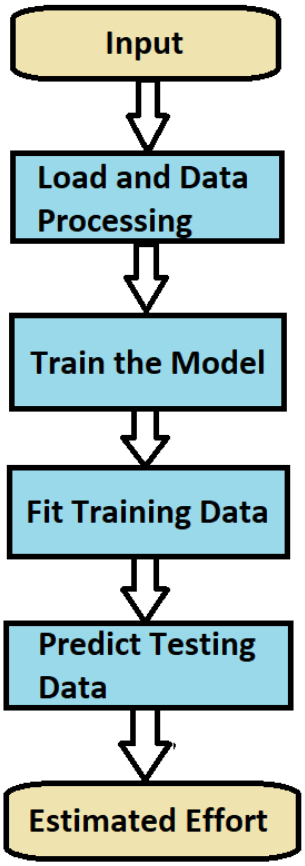

Fig. 1. Flow of Models

C. Comparison of Estimated Effort

In this section, the estimated efforts have been compared and we have shown the comparison result through graphical representation. The X-axis of the graph represents Software Size which has been measured by UCP and the Y-axis of the graph represents Effort which is measured in person-hours. We have compared estimated effort with actual effort for each regression model. In the graph, red scattered points represent actual effort and the blue plot represents estimated effort. The effort is estimated considering all inputs but only Software Size is included in the graphs. The graphs have been displayed in the next part.

D. The Relationship Chart between Effort and Software Size

The following Fig. 2, Fig. 3, Fig. 4, Fig. 5 and Fig. 6 have been delineated to make comprehensible regarding the comparison of estimated and actual data of all regression models. The feasible models are summarized by analysing the graphs.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

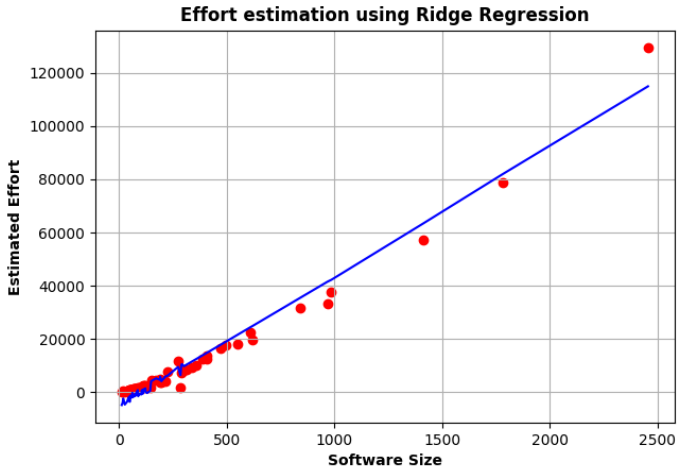

Fig. 2. Comparison graph between Actual and Estimated Effort using Ridge Regression

Fig. 2 shows the comparison between actual effort and estimated effort using Ridge Regression based on Software Size. This Regression model maps inputs to output with good accuracy. The regression line has been fitted with actual data and produces good efficacy to estimate result. After training, we have tested data using Software size from 13 to almost 2500 UCP. The average Software Size for testing is 290 UCP. Ridge Regression calculates 97.23% R-squared score, 11282187.21 for Mean Squared Error and 2676.68 for Mean Absolute Error. It scores the best R-squared and lowest Mean Squared Error.

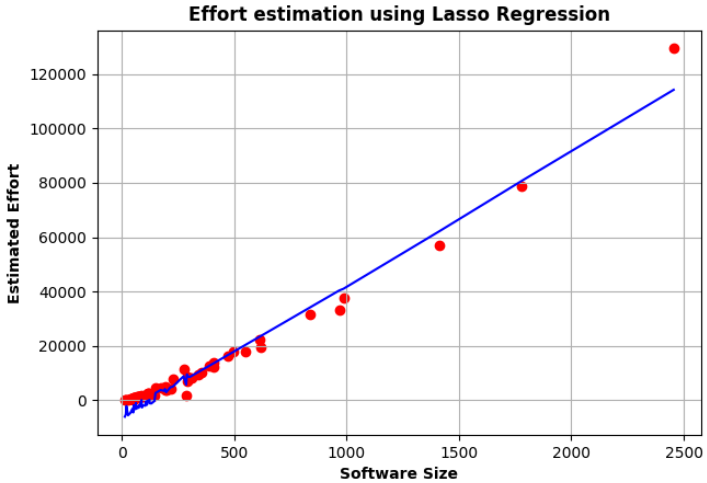

Fig. 3. Comparison graph between Actual and Estimated Effort using Lasso Regression

The result of Lasso Regression is described graphically in Fig. 3. This graph shows us the almost same result as Ridge Regression. The proper fit proves the model as a good estimator. Its R-squared score is 96.61%, Mean Squared Error is 13800678.03 and Mean Absolute Error is 2982.57.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

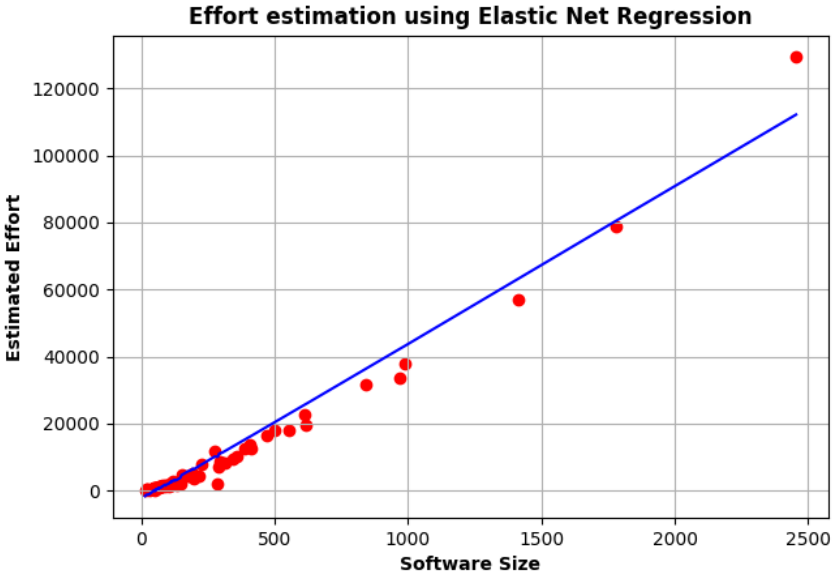

Fig. 4. Comparison graph between Actual and Estimated Effort using Elastic Net Regression

From Fig. 4 we get the estimated result of Elastic Net Regression. Elastic Net Regression algorithm provides more accurate result than Ridge and Lasso Regression. Its Regression line is closer to actual data. For different types of error measures, we see Elastic Net efficacy is more desirable. The R-squared score is 97.1%, Mean Squared Error is 11794075.88 and its Mean Absolute Error is 2110.17. It calculates the lowest Mean Absolute Error among other models.

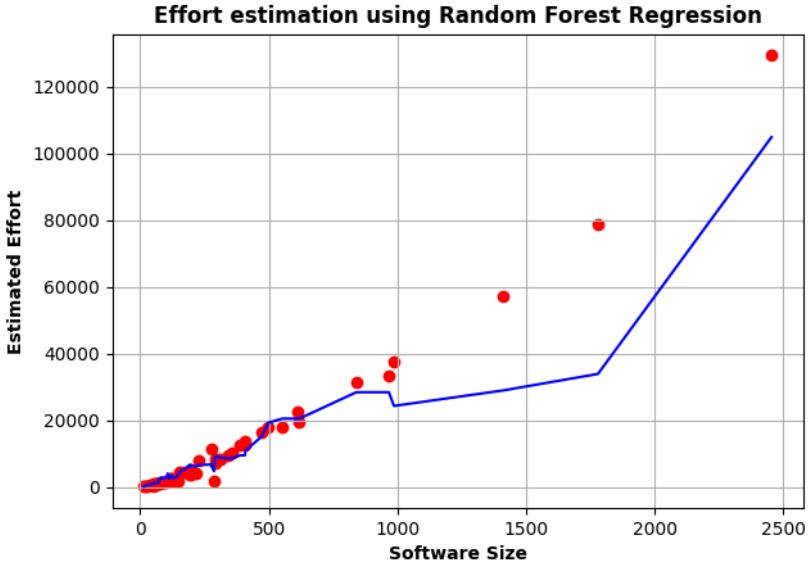

Fig. 5. Comparison graph between Actual and Estimated Effort using Random Forest Regression

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

Fig. 5 has been drawn using Random Forest Algorithm. We observe the deviation of the Regression line from actual data. It determines comparatively somewhat low accuracy than other Regression models. For Ransom Forest R-squared score is 85.87%, Mean Squared Error is 57555514.78 and the Mean Absolute Error is 2836.27

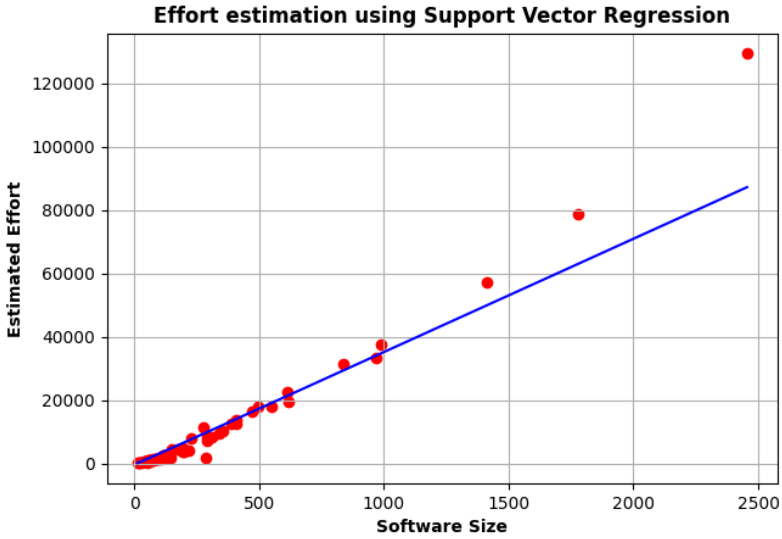

Fig. 6. Comparison graph between Actual and Estimated Effort using Support Vector Regression

From Fig. 6 we observe the estimated effort-size relationship of another popular Regression model i.e. Support Vector Regression. Support Vector Regression provides a better result than Random Forest but not good as Ridge and Elastic Net. It provides 91.43% R-squared score and determines Mean Squared Error and Mean Absolute Error are 34914047.1 and 2307.58 respectively.

E. Error Calculation

We have evaluated these algorithms using three types of error measures as metrics. We have used R-squared, Mean Absolute Error (MAE) and Mean Squared Error (MSE) to find out the feasible algorithms regarding effort estimation among these five regression algorithms.

1) R-squared: R-Squared is a measure of statistical that indicates how much dependent variables are varied by the independent variable(s) in a regression model. R-squared values range from 0 to 1 and are commonly stated as percentages from 0 to 100. An R-squared of 100% means that all dependent variables are completely explained by the independent variable(s). The formula of R-squared is given below:

$$\text{R\_squared} = 1 - \left(\frac{\text{UV}}{\text{TV}}\right) \quad (1)$$

Where, UV – Unexplained Variation, TV – Total Variation

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

2) Mean Absolute Error: The Mean Absolute Error (MAE) is the average of all absolute errors. The smaller the Mean Absolute Error (MAE), the regression line will be closer to the best fit. Depending on testing data, the Mean Absolute Error (MAE) can be large or small. The formula is as follows:

$$\text{MAE} = \frac{1}{n} \sum_1^N (|\ E_e - A_e|) \qquad (2)$$

Where, n = the number of errors, $|\ E_e - A_e|$= the absolute error between Estimated and Actual Effort.

3) Mean Squared Error: The Mean Squared Error (MSE) is another statistical measure that tells how close a regression line is to a set of points. It does this by taking the distances from test data to the regression line. These distances are errors of individual data. Then these errors are squared and negative signs are removed. The lower the MSE, the better the prediction result.

As it is squared formulae, MSE is also large like MAE depending on testing data. MSE is calculated as:

$$\text{MSE} = \frac{1}{n} \sum_1^N (E_e - A_e)^2 \qquad (3)$$

Where, n = the number of errors, $E_e - A_e$ = difference between Estimated and Actual Effort.

The graphical views of the error calculation for the implemented algorithms are depicted and provided below.

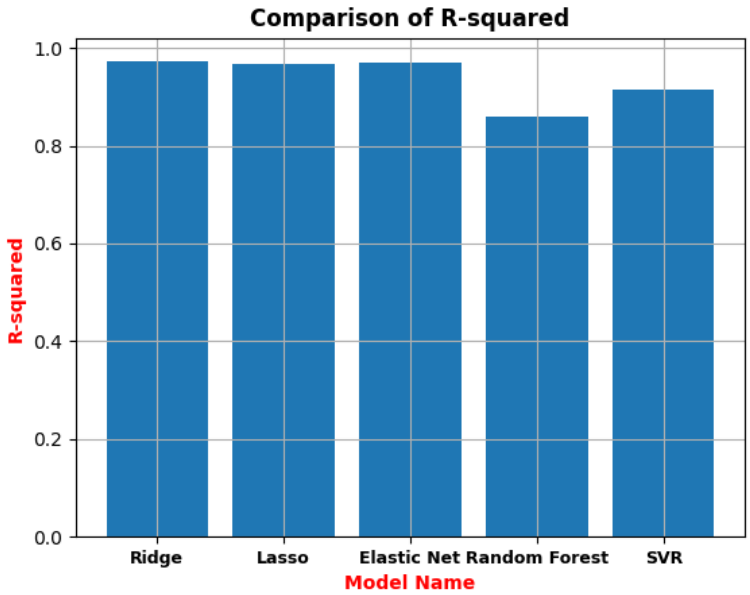

Fig. 7. Comparison by R-squared scores

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

Fig.7 shows the comparative result of used regression models by R-squared scores. The graph describes that Ridge, Lasso and Elastic Net calculate about the same and better result than Random Forest and SVR.

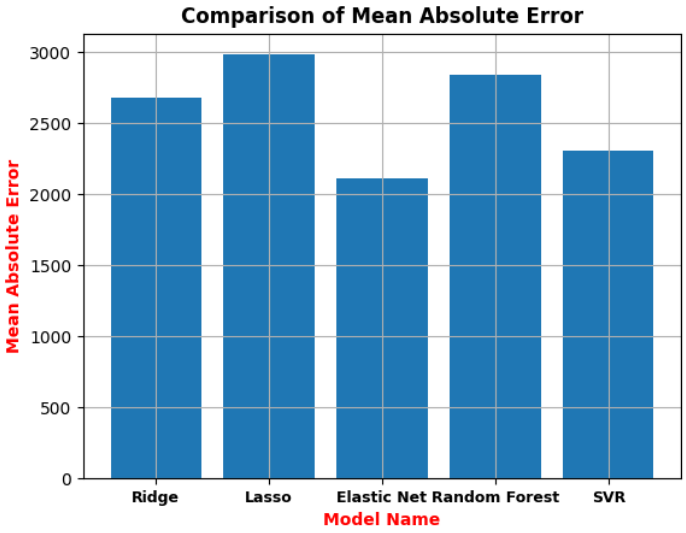

Fig. 8. Comparison by Mean Absolute Error

It is clearly visible from Fig.8 that for Mean Absolute Error (MAE) Elastic Net can estimate feasible result for effort estimation cause its MAE is relatively very good than others.

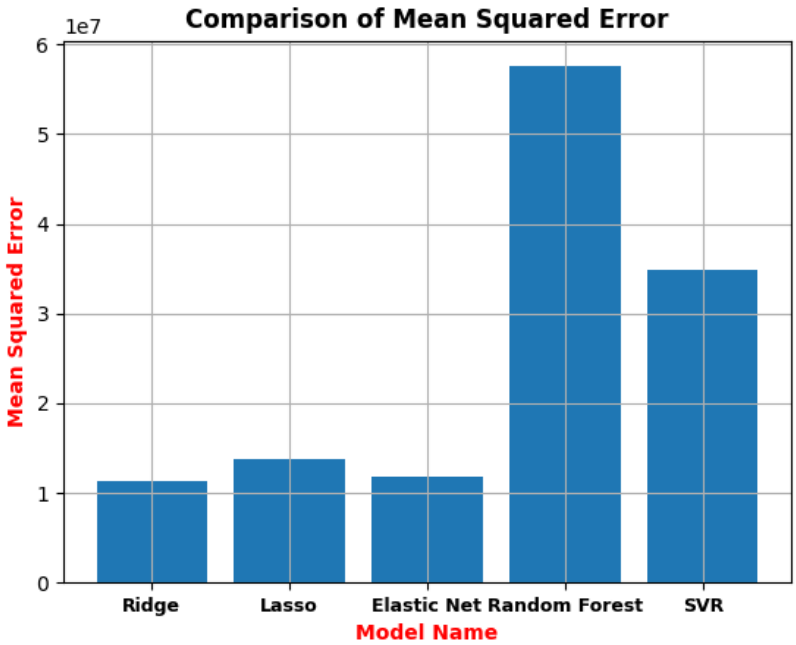

Fig. 9. Comparison by Mean Squared Error

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

To evaluate by MSE, we observe from Fig. 9 Ridge and Elastic Net both can be good estimators cause they calculate less MSE than other regression models used in this paper.

3. CONCLUSIONS

Software Effort Estimation is considered in software industries with great importance. It is emergent to estimate effort accurately earlier to avoid last time stress. This paper summarizes that a single algorithm cannot estimate optimal effort for all cases in software industries. To find the feasible algorithms, we have compared five regression models. The regression models have been evaluated by measuring three types of error formulas. It is observed that Elastic Net can estimate more accurate result than other models. Apart from Elastic Net, Ridge Regression can be a good estimator in the effort estimation field. In the future, hybrid models with multiple regression algorithms can be proposed and the implementation of these models might be explored with multiple datasets.

4. REFERENCES

1. A. B. Nassif, “Software size and effort estimation from use case diagrams using regression and soft computing models,” Electronic Thesis and Dissertation Repository, 2012.
2. S. Srichandan, “A new approach of Software Effort Estimation Using Radial Basis Function Neural Networks,” International Journal on Advanced Computer Theory and Engineering, vol. 1, issue. 1, pp. 113-120, 2012.
3. N. Govil, ”Analyzing Software Complexities by Applying Data Structure Metrics on Different Programming Languages”, 5th International Conference on Communication and Electronics Systems (ICCES), pp. 833-838, 2020.
4. N. Govil, ”Applying Halstead Software Science on Different Programming Languages for Analyzing Software Complexity”, 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), pp. 939-943, 2020.
5. Monika, O. P. Sangwan, “Software effort estimation using machine learning techniques,” 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, pp. 92-98, 2017.
6. B. Baskeles, B. Turhan, A. Bener, “Software effort estimation using machine learning methods,” 22nd international symposium on computer and information sciences, 2007.
7. V. S. Ionescu, “An approach to software development effort estimation using machine learning,” 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 197-203, 2017.
8. A. B. Nassif, M. AbuTalib and L. F. Capretz, ”Software Effort Estimation from Use Case Diagrams Using Nonlinear Regression Analysis,” IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1-4, 2020.
9. A. Sharma and N. Chaudhary, ”Linear Regression Model for Agile Software Development Effort Estimation,” 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp. 1-4, 2020.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

10. R. Silhavy, P. Silhavy and Z. Prokopova, ”Analysis and selection of a regression model for the Use Case Points method using a stepwise approach”, Journal of Systems and Software, vol. 125, pp. 1-14, 2017.
11. A. Kaushik, D. K. Tayal and K. Yadav, ”A Comparative Analysis on Effort Estimation for Agile and Non-agile Software Projects Using DBN-ALO”, Arabian Journal for Science and Engineering, pp. 1-14, 2019.
12. M. Azzeh and A. B. Nassif, ”Project productivity evaluation in early software effort estimation”, J. Softw. Evol. Process, vol. 30, no. 12, pp. e2110, 2018.
13. K. Korenaga, A. Monden and Z. Y¨ucel, ”Data Smoothing for Software Effort Estimation,” 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 501-506, 2019.
14. M. Hammad and A. Alqaddoumi, ”Features-Level Software Effort Estimation Using Machine Learning Algorithms,” International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pp. 1-3, 2018.
15. G. Srivastava, Y. More and J. Sam, ”Effort Estimation Model for an Enterprise Software Upgrade,” International Conference for Emerging Technology (INCET), pp. 1-6, 2020.
16. T. Vera, S. F. Ochoa and D. Perovich, ”Development Effort Estimation Practices in Small Software Companies: An Exploratory Study,” 39th International Conference of the Chilean Computer Science Society (SCCC), pp. 1-8, 2020.
17. S. M. Satapathy, S. K. Rath, “Effort estimation of web-based applications using machine learning techniques,” International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 973-979, 2016.
18. S. K. Pillai, M. K. Jeyakumar, “Extreme learning machine for software development effort estimation of small programs,” International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], pp. 1698-1703, 2014.
19. L. F. Capterz, M. Azzeh, D. Ho, A. B. Nassif, “Neural network models for software development effort estimation: a comparative study,” Neural Computing and Applications, 2015.
20. S. Tarannum, M. Suaib and A. Muttalib, “Neural Network: A better Approach for Software Effort Estimation,” International Journal of Computer Application, vol. 130-No.8, pp. 21-24, 2015.
21. A. M. Bautista, A. Castellanos, T. S. Feliu,“Software Effort Estimation using Radial Basis Function Neural Networks,” International Information Theories and Applications, vol. 21, 2014.
22. P. Reddy P.V.G.D, K. R. Sudha, R. Sree and S. Ramesh, “Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks,” Journal of Computing, vol. 2, pp. 87-92, 2010.
23. D. S. Broomhead and D. Lowe, “Multivariable Functional Interpolation and Adaptive Networks,” Complex Systems, vol. 2, pp. 321-355, 1988.
24. F. Wang, X. Yang, X. Zhu and L. Chen, “Extended use case points method for software cost estimation,” in International Conference on Computational Intelligence and Software Engineering, 2009.

International Journal of Information Technology and Computer Engineering
ISSN: 2455-5290
Vol: 03, No. 06, Oct-Nov 2023
http://journal.hmjournals.com/index.php/IJITC
DOI: https://doi.org/10.55529/ijitc.36.26.42

25. H. B. K. Tan, Y. Zhao and H. Zhang, "Conceptual data model-based software size estimation for information systems," ACM Transactions on Software Engineering and Methodology, vol. 19, pp. 4:1-4:37, oct, 2009.
26. M. T. Rahman and M. M. Islam, "A Comparison of Machine Learning Algorithms to Estimate Effort in Varying Sized Software," IEEE Region 10 Symposium (TENSYMP), pp. 137-142, 2019.