SOFTWARE DEVELOPMENT EFFORT ESTIMATION USING HARMONY SEARCH TECHNIQUE

By

S. Sriharshini

Under the supervision of

Dr. Manjubala Bisi

Department of Computer Science and Engineering
National Institute of Technology Warangal-506001,
INDIA

JUNE-2023

SOFTWARE DEVELOPMENT EFFORT ESTIMATION USING HARMONY SEARCH TECHNIQUE

Report submitted to

National Institute of Technology Warangal

towards partial fulfilment for the

award of the degree

of

Bachelor of Technology

By

S. Sriharshini

Under the supervision of

Dr. Manjubala Bisi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL-506001, INDIA

JUNE-2023

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

CERTIFICATE

It is certified that the work contained in report titled **“SOFTWARE DEVELOPMENT EFFORT ESTIMATION USING HARMONY SEARCH TECHNIQUE”** by S. Sriharshini have been carried out under my supervision and that work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr. Manjubala Bisi

Assistant Professor

Computer science and engineering

NIT Warangal

ACKNOWLEDGEMENT

We would like to thank all those who are involved in this endeavour for their kind cooperation for its successful completion. At the outset, we wish to express our sincere gratitude to all those people who have helped us to complete this project in an efficient manner. We offer our special thanks to our project guid Dr. Manjubala Bisi Mam, Department of Computer Science & Engineering, National Institute of Technology, Warangal without whose help and support throughout this project would not have been possible. We would like to thank Tsewang mam, Head, Department of computer Science and Engineering, and National Institute of Technology Warangal, who gave us the opportunity to do this project at an extreme organization. Most of all and more than ever, we would like to thank our friends who always advised and motivated us throughout the course.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Software Effort Estimation plays an instrumental role in estimating the effort required to complete a software development project. Accurate evaluation of software development effort estimation is a significant challenge for project planning and management. It involves the process of estimating the effort required in order to meet project delivery deadlines by appropriately scheduling and allocating resources during the Software Development. In this paper, we are going to do Software Effort Estimation using Harmony search. Harmony search is an effective meta-heuristic algorithm inspired by the music improvisation process, where musicians search for a pleasing harmony by adjusting their instruments' pitches. The HS algorithm and its variants have been widely used to solve binary and continuous optimization problems. Many search strategies have been exploited in implementing feature selection, to identify smaller and better subsets. In this paper, we have proposed a feature selection technique with regression models using Harmony search to estimate software development effort. We have validated our approach in four data sets for software effort estimation. We have used various regression models including Linear, Logistic regression, Random Forest regression and Ridge regression to obtain the estimated effort. We have compared the result of our proposed approach with some existing models of feature selection and regression techniques and found that our approach is able to provide better prediction performance than some existing models.

Keywords: Software development effort, Harmony Search, Feature selection, Regression model, Optimization.

CHAPTER-1
INTRODUCTION

The importance of software constantly growing across all individual fields. It would be difficult to control and plan the project without accurate estimates [3]. Hence, accurate SEE allows for effective scheduling and management of resources allocated to various software projects [4]. In real world problems FS is usually necessary due to the abundance of noisy, irrelevant, redundant, or misleading features. The main aim of feature selection (FS) is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features [1-2]. There are several techniques for Feature selection. Among them, Harmonic Search algorithm has been very successful in a wide variety of optimisation problems, presenting several advantages with respect to traditional optimisation techniques [2]. In this work, we have proposed a Harmonic search-based feature selection technique.

1.1 Software Development Effort Estimation:

The utilization of software constantly growing across all individual fields. It has become extremely important in the development, engineering, and acquisition of systems, especially for sophisticated and large-scale systems. For these kinds of systems, accurate development effort estimation is required and is regarded as a crucial component of efficient project management for software. It would be difficult to control and plan the project without accurate estimates. Proper effort estimation is essential for feasibility studies, bidding, iteration plans, project planning, estimation of cost, and budgets for a project. Furthermore, effort estimation can be useful in predicting future resource requirements [3].

Software Effort Estimation (SEE) is the process of estimating the effort needed for executing a project in order to set up effective project delivery deadlines. Accurately estimating the software effort of a new project is instrumental in the field of software engineering, since it minimizes over and underestimation, both of which, may result in the cancellation of a project due to overruns in budget and can delay project funding. Hence, accurate SEE allows for effective scheduling and management of resources allocated to various software projects [4]. Software estimation models have been far better than the traditional estimates used in the industry [3]. Feature optimization is of paramount importance in machine learning and predictive tasks/applications as it reduces the number of dimensions of high dimensional data and improves the efficiency of an algorithm by reducing computational costs [4]. For the last few decades, researchers have focused on regression models for software effort estimation, while Feature selection-based estimations are giving better results than conventional regression models [3].

The main aim of feature selection (FS) is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features. In real world problems FS is usually necessary due to the abundance of noisy, irrelevant, redundant, or misleading features [2]. It is a process of selecting an optimal subset of features from a data set so that the classifier can obtain better accuracy and/or reduce the computational burden. Nonetheless, removing irrelevant features is a challenging issue and time-consuming owing to a large search space and wrapped relationship between the features [1]. Usually, FS algorithms involve heuristic or random search strategies attempt to avoid this prohibitive complexity. However, the degree of optimality of the final feature subset is often reduced [2]. There are several FS methods. Among them, harmony search (HS) is an effective metaheuristic algorithm inspired by the music improvisation process of probing for a better state of harmony [1].

1.2 Harmonic Search Algorithm:

Traditional wrapper-based FS methods, such as sequential backward selection (SBS) and sequential forward selection (SFS), improve the performance of the learning model via sequentially adding or removing features from data set. However, they suffer from nesting effects and computationally expensive. To alleviate these problems, population-based optimization algorithms, such as particle swarm optimization (PSO), genetic algorithm (GA), genetic programming (GP), ant colony optimization (ACO), brain storm optimization (BSO) and harmony search (HS), have been used. These algorithms start with a set of randomly generated solutions, and use a fitness function to evaluate them. Then, they generate new solutions based on the individuals that performed better in the previous iteration. As a result, these algorithms reduce the computational burden as they avoid generating new individuals like the low-quality ones [1].

Among them, Harmony search is a meta-heuristic algorithm. It mimics the improvisation process of music players. Harmonic Search algorithm has been very successful in a wide variety of optimisation problems, presenting several advantages with respect to traditional optimisation techniques. It imposes only limited mathematical requirements and is not sensitive to the initial value settings. The Harmonic Search algorithm generates a new potential solution vector, after considering all existing vectors. The basic Harmonic Search algorithm has been improved by introducing methods to tune parameters dynamically [2]. Harmonic Search has been widely applied to solve real-world optimization problems, such as control system and financial management, due to its simple structure, easy to implement and less parameters [1]. Given a dataset with discretised attribute values, using random sets it is possible to find a subset (termed redact) of the original attributes that are the most informative; all other attributes can be removed from the dataset with minimal information loss [2].

Harmony search mimics the improvisation process of musicians, during which, each musician plays a note for finding a best harmony all together [2]. When applied to optimisation problems, the musicians represent the decision variables of the cost function, and HS acts as a meta heuristic algorithm which attempts to find a solution vector that optimises this function. In the process, each decision variable (musician) generates a value (note) for finding a global optimum (best harmony) [1]. The harmony search algorithm has a novel stochastic derivative (for discrete variable) based on musician's experience, rather than gradient (for continuous variable) in differential calculus. In most optimisation problems solvable by HS, the musicians are the decision variables of the function being optimised [2]. The notes played by the musicians are the values each decision variable can take. The harmony contains the notes played by all musicians, or a solution vector containing the values for each decision attribute. The harmony memory contains harmonies played by the musicians, or a storage place for solution vectors [1].

Comparison factor	**Harmony improvisation**	**Optimization process**
Targets	Aesthetic standard	Objective function
Best states	Fantastic harmony	Global optimum
Components	Pitches of instruments	Values of variables
Process unit	Each procedure	Each iteration

1.3 Objective of the Project:

- Design of a regression model to predict software development effort without using any Feature Selection. (i.e., including all Features)
- Development of a Harmonic Search approach to select the features affecting software development effort which predicts the software development effort using Feature Selection. (i.e., only selective Features are included)
- Validation of the proposed approach on four data sets and comparison of results of Harmonic Search algorithm with feature selection and results of regression models without feature selection with some existing methods.

CHAPTER-2

REQUIREMENTS

The overview of this project is to find the software development effort with and without Feature Selection using various machine learning algorithms such as Regression models and Harmonic Search algorithm for Feature Selection and compare the results of both to find which gives better results.

2.1 HARDWARE REQUIREMENTS:

The basic and important hardware requirements are listed in the table below:

S.NO	COMPONENTS	SPECIFICATIONS
1	Operating system	Windows 11
2	Ram	8GB
3	Hard disc or ssd	More than 500GB
4	Processor	Intel 5th generation

2.2 SOFTWARE REQUIREMENTS:

The basic and important software requirements are listed in the table below:

S.NO	COMPONENTS	SPECIFICATIONS
1	Software	Python 3.11.3
2	Ide	Jupyter notebook

2.3 Technical Skills:

Technologies used: Machine Learning

Languages used: Python

CHAPTER-3
METHODOLOGY

3.1 Harmonic Search Algorithm Design:

Harmonic Search Algorithm It is a population-based metaheuristic algorithm inspired from the musical process of selecting for a perfect state of harmony. It has been proposed by Geem et al. in 2001.The pitch of each musical instrument determines the aesthetic quality, just as the fitness function value determines the quality of the decision variable. In the music improvisation process, all players sound pitches within possible range together to make a good harmony. If all pitches make a good harmony, each player stores in its memory, that experience and the possibility of making a good harmony is increased next time. In optimization, the initial solution is generated randomly from decision variables within possible range. If the objective function values of these decision variables are good to make a promising solution, then the possibility to make a good solution is increased next time.

3.1.1 The Principles of Harmonic Search:

1) *Initialisation:*
   a) *Initialise Problem Domain:* The parameters of HS are assigned according to the problem, including: size of harmony memory, number of musicians, max iteration, and optionally, the harmony memory considering rate (HMCR), and the pitch adjustment rate (PAR). In any example, the number of musicians each corresponds to the decision attributes. The harmony memory size is always equal to the number of musicians, and the objective function is the function to be minimised in most of the problems, the lower the better.
   b) *Initialise Harmony Memory:* Harmony memory is filled with randomly generated solution vectors. The initial population HM contains HMS vectors is generated randomly where,

$$X_i = X_{ij} \; i = 1, 2\ldots\ldots, \text{ and } j=1, 2\ldots\ldots, n.$$

The HM matrix is fitted with HMS vectors as follow:

$$\begin{matrix} X11 & X12 & . & . & . & X1n \\ X21 & X22 & . & . & . & X2n \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ XHMS1 & XHMS2 & . & . & . & XHMSn \end{matrix}$$

2) *Improvise New Harmony:*

A new value is chosen randomly by each musician out of their note domain, and together forms a new harmony. There are two factors which affect the note choice of a musician, HMCR and PAR. HMCR, range from 0.7 to 0.95, is the rate of choosing one value from the historical notes stored in the harmony memory, pitch adjustment is applied after a note is chosen by the musician, either from the HM or all possible value range.

The whole process takes place in following steps:

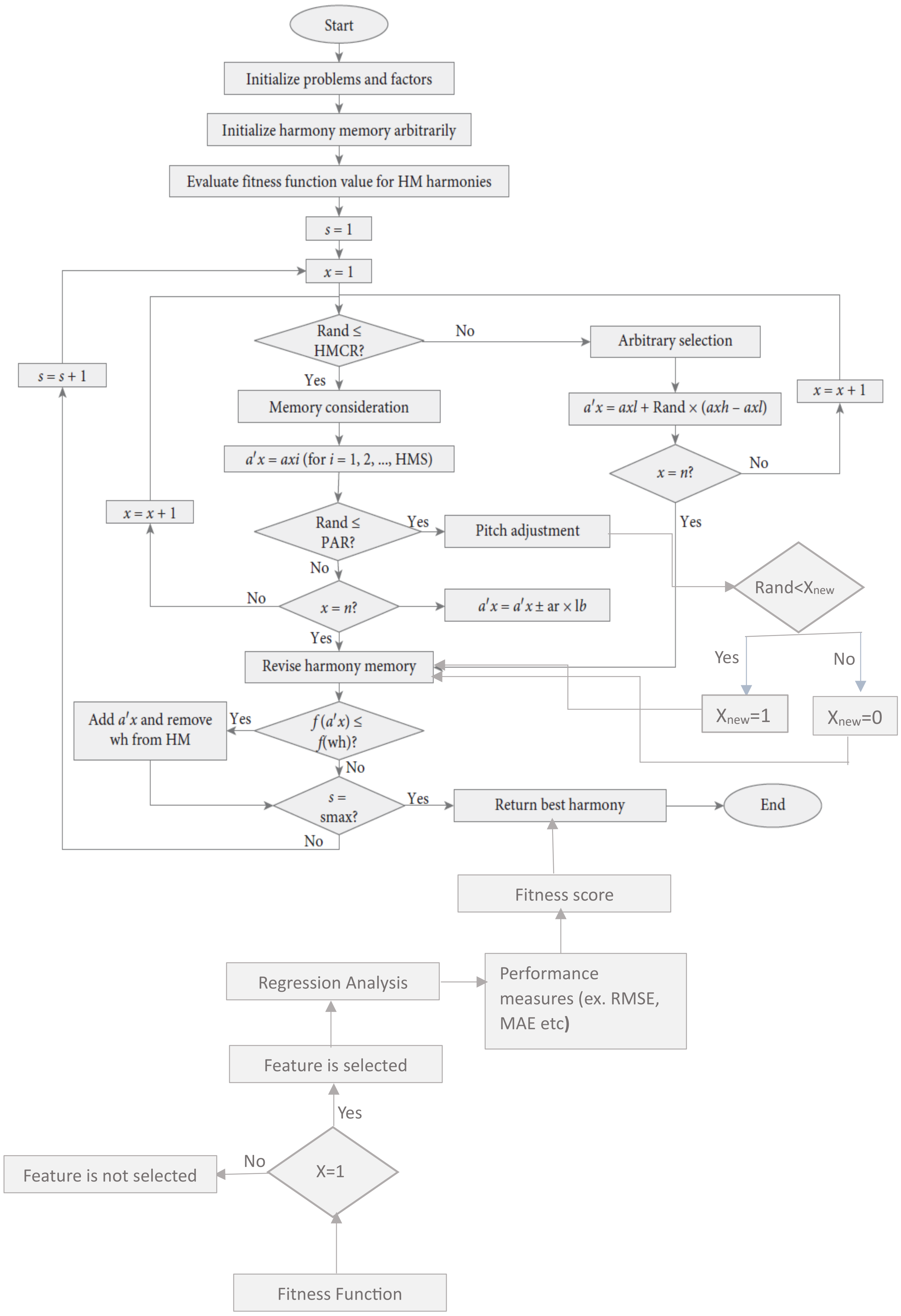
1. A new random number r1 is generated within the range [0, 1]
2. If r1 < HMCR, then the first decision variable in the new vector X_{ij}^{new} is chosen randomly from the values in the current HM as:
$$X_{ij}^{new} = X_{ij}, \text{ where } X_{ij} € \{X_{1j}, X_{2j}, \ldots\ldots\ldots X_{HMSj}\}$$
3. The obtained decision variable from the HMCR is further examined to understand if it needs to pitch adjustment or not.
4. Generate a new random number r2 within the range [0,1]
5. If r2 < PAR, then the pitch adjustment decision variable is calculated as follows:
$$X_{ij}^{new} = X_{ij} +\text{-}\ \text{rand}\ (0,1)\ .BW$$
6. Where BW is the bandwidth factor which is used to control the local search around the selected variable in the new vector.
7. For Binary Harmony Search, we may get any X_{ij}^{new} values but, we want only between (0,1) so we need to convert them using Sigmoid function which is given below:
$$f(x) = 1/1+e^{-x}$$
8. Now we need examine further so that we can get only X_{ij}^{new} as 0 and 1for that generate a random number r3 as Rand (0,1) then we get X_{ij}^{new} based on following conditions:
$$\text{If } r3 < X_{ij}^{new} \text{ then, } X_{ij}^{new} = 1$$
$$\text{If } r3 > X_{ij}^{new} \text{ then, } X_{ij}^{new} = 0$$
9. If the condition r1 < HMCR fails, the new 1st decision variable in the new vector X_{ij}^{new} is generated randomly as
$$X_{ij}^{new} = l_{ij} + (u_{ij} - l_{ij}).\text{Rand}\ (0,1)$$
Where, l is the lower and u is the upper bound for the given problem.

3) *Update Harmony Memory:*

If the new harmony is better than the worst harmony in the harmony memory, judged by the objective function, the new harmony is then included in harmony memory and the existing worst harmony is removed. The algorithm continues to iterate until the maximum number of iterations has been reached. The optimal solution is chosen based on minimal\maximal objective function score for minimization\maximization problem.

- If (Xnew < Xworst) (for minimization problem), then update HSA as Xworst = Xnew
- If (Xnew > Xworst) (for minimization problem), then update HSA as Xworst = Xnew

3.1.2 Flowchart



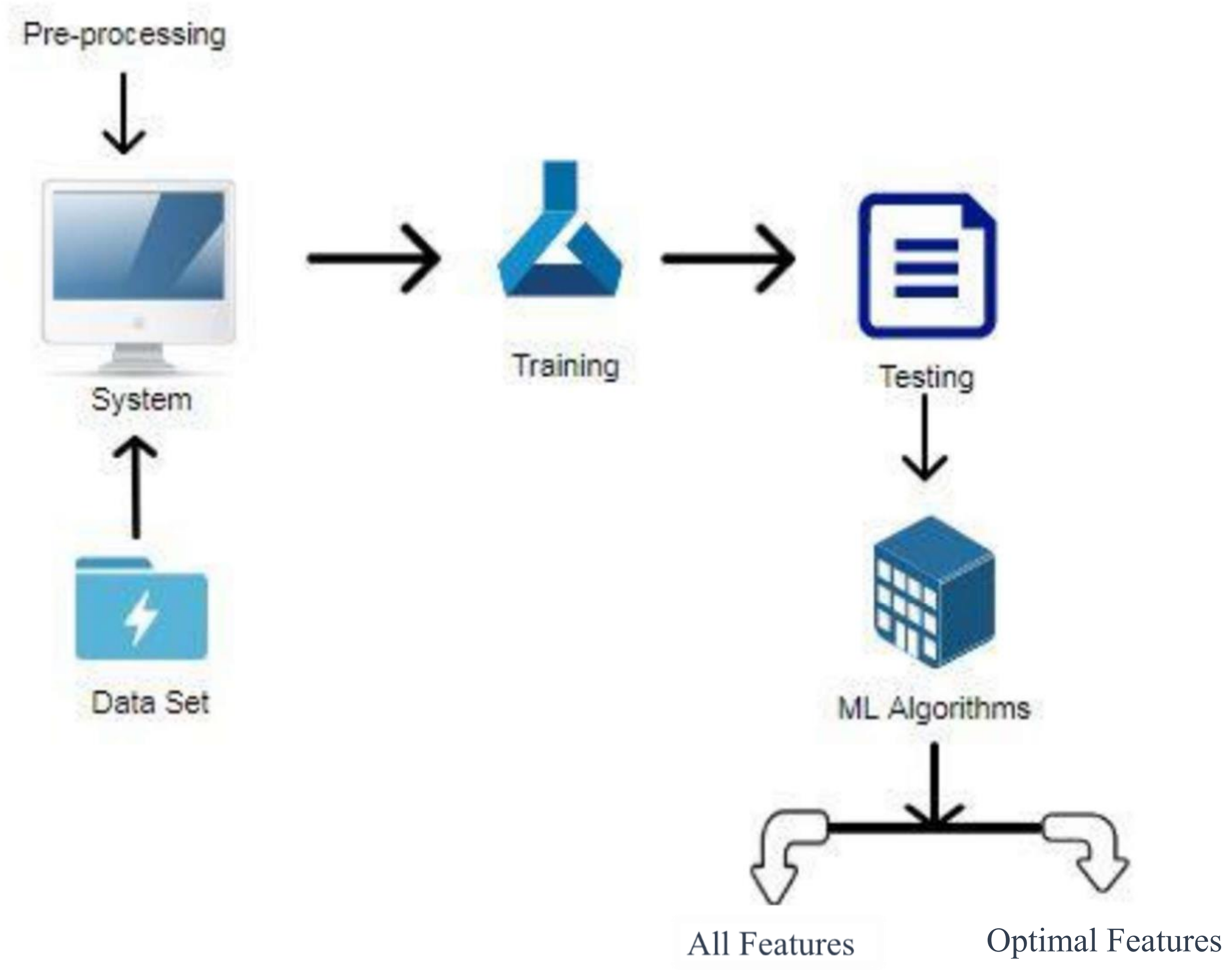
3.1.3 Algorithm

```
1.  Initialize parameters (HMS, HMCR, PAR, BW, NT):
2.  Set t = 0
3.   Initialize harmony memory:
4.   for (i = 1; i <= HMS; i++)
5.      Generate an initial harmony vector Xi
6.      Evaluate the fitness function of Xi
7. end for
8. Repeat until termination condition is met:
9. for (i = 1; i <= HMS; i++) // Generate a new solution (harmony) Xi
10.  for (j = 1; j <= n; j++) // Iterate over decision variables
11.     Generate r1, r2, r3, r4, r5 using random (0,1)
12.     if (r1 < HMCR) then
13.         $X_{ij}^{new} = X_{ij}$
14.           if (r2 < PAR) then
15.              $X_{ij}^{new} = X_{ij} \pm r3 * BW$
16.             if (r4 < $X_{ij}^{new}$) then
17.                $X_{ij}^{new} = 1$
18.             else
19.                $X_{ij}^{new} = 0$
20.         end if
21.     else
22.         $X_{ij}^{new} = l_{ij} + r5 * (u_{ij} - l_{ij})$
23.       end if
24.       if (fitness ($X_{ij}^{new}$) < fitness ($X_{worst}$)) then
25.          Update the harmony memory: $X_{worst} = X_{ij}^{new}$
26.       end if
27.    end for
28.  end for
29.  Set t = t + 1
30.  until (t < NT)
31. Produce the best solution (harmony vector) X as the optimal solution
```

Note: In this project fitness function is a Regression model which returns RMSE value as fitness score. Hence its algorithm is like machine learning model.

3.2 Model Design and Organisation:

Model design is a brief overall description of this project as shown in the figure. The first step is to import all the packages that are used in this project and then the collected dataset is loaded into it. Now the dataset is split into training and testing parts and different machine learning algorithms such as Regression models and Harmonic Search Algorithm are applied on the training part such that we can compare results calculated using all features with the results calculated using some optimal features. After following every step shown in the figure, we can understand the Importance of Feature Selection through this project.



CHAPTER-4

EXPERIMENTAL SETTING

The experimental requirements required for this project are represented under Experimental settings. As we know in this project, we are applying different machine learning algorithms such as Regression models which uses all the features without Feature selection and we are using Harmonic search algorithm with Feature selection. So, to observe and compare the results we need datasets. We are going to apply these algorithms on different datasets so we get different results from which we can compare and obtain the conclusion i.e., Feature selection gives better results or not as compared to normal machine learning models. Hence, in this project we will see four different datasets on which four different Regression models are applied. These models each give four different performance measures. After observing and comparing all the performance measure we get conclusion in this experiment.

4.1 Datasets

In this project we are going to examine four different datasets. We can observe the brief description about each dataset in the form of a table given below:

Datasets	Features	Projects	Effort units
Albrecht	07	24	Hours
Kemerer	06	15	MM (man-month)
Desharnais	10	81	Hours
Cocomo	16	63	Hours

4.2 Regression models:

In this project we are going to analyse the datasets using four different Regression models. They are Linear Regression, Ridge Regression, Random Forest Regression and SVM Regression. In this section we briefly describe each of the models used.

1. Linear Regression:

Linear regression is used for the purpose of estimation of software effort and is hence used to predict the relationship between independent variables(X) (software metrics in our case) and a single dependent variable(y) (the predicted effort). Here the independent variables(X) can be two or more than two in number. y is called the dependent variable because its value depends on the values taken by the independent variables(X). The relationship is depicted below:

$$y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$$

The dependent variable is y and the independent variables are X_1-X_n. Here β_i depicts the regression coefficient of the i th independent variable X_i.

2. Random Forest Regression:

Random Forest Regression is a machine learning algorithm that is based on the concept of ensemble learning. It combines multiple decision trees to make predictions by averaging or taking a majority vote of the individual tree predictions. In the case of regression, the algorithm predicts a continuous target variable. Randomly select a subset of the training data (with replacement) and use it to build a decision tree. Repeat this process multiple times to create a collection of decision trees. Make predictions for each decision tree. Combine the predictions from all the decision trees. For regression, this can be done by taking the average of the individual tree predictions. The equation for Random Forest Regression can be written as:

$$\hat{Y} = \frac{1}{N} \sum_{i=1}^{N} T_i(X)$$

Where, $\widehat{Y}$ is the predicted value for the target variable, N is the total number of decision trees in the ensemble, $T_i(X)$ is the prediction of the i-th decision tree for the input X. The final prediction is obtained by averaging the predictions of all the decision trees in the ensemble.

3. Ridge Regression:

Ridge regression [8] is essentially an extension of simple linear regression. It makes the use of the L2 regularization technique to prevent over fitting. Ridge regression has a penalty factor λ which penalizes a higher value of the regression coefficients. The optimal regression coefficients for Ridge regression are shown in Eq. 1.

$$\widehat{\beta}_{ridge} = \arg\min_{\beta} \sum_{n=1}^{N} \frac{1}{2} (y_n - \beta x_n)^2 + \lambda \sum_{i=1}^{p} \beta_i{}^2 \tag{1}$$

where N is the number of samples in the dataset. In the vectorized notation, Eq. 1 can be written as:

$$\widehat{\beta}_{ridge} = \left(X'X + \lambda I_p\right)^{-1} X'Y$$

where X'X is the singleton matrix of predictive variables and Ip is the identity matrix of rank p.

4. SVM (Support Vector Machine) Regression:

SVM regression is a machine learning algorithm used for predicting continuous values. It finds a hyperplane that best fits the data, minimizing errors. The equation is

$$y = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$$

where 'y' is the predicted output and 'x_1', 'x_2', ..., 'x_n' are input features. The goal is to maximize the margin around an error tolerance region. The optimization problem involves minimizing errors and a regularization parameter 'C' controls the trade-off between margin and errors. The goal is to minimize errors and maximize the margin around an error tolerance region. This is achieved by solving an optimization problem that includes a regularization parameter 'C' and slack variables ξ_i and ξ_i^*. The optimization problem can be summarized as minimizing

||w||^2 + C * $\Sigma\, \xi_i + \Sigma\, \xi_i^*$, *subject to* y_i - ($w_0 + w_1x_1$_i + w_2x_2_i + ... + w_nx_n _i) $\leq \varepsilon + \xi_i$ and ($w_0 + w_1x_1$_i + w_2x_2_i + ... + w_nx_n _i) - y_i $\leq \varepsilon + \xi_i^*$, with ξ_i and $\xi_i^* \geq 0$.

By solving this problem, the coefficients 'w0', 'w1', 'w2', ..., 'wn' are obtained, allowing for predictions by substituting the input features into the equation $y = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$

4.3 Performance measures:

In this paper I am going to compare the results of with feature selection and without feature selection by using the Performance measures. There are so many performance measures but I am considering only four. We have used Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Magnitude of Relative Error (MMRE) and Percentage of Prediction (PRED), as the performance measures to validate our proposed approach for Output, i.e., software development effort. In this section we briefly describe each of the Performance measures used.

1. Root Mean Square Error (RMSE):

RMSE stands for Root Mean Squared Error, which is a commonly used metric to evaluate the accuracy of regression models. It measures the average magnitude of the differences between predicted and actual values, providing a measure of the model's prediction error.

The equation for RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{n} * \sum (y_i - \hat{y}_i)^2}$$

To calculate RMSE, we compute the squared difference between the actual and predicted values for each data point, sum these squared differences, divide by the number of data points (n), and take the square root of the result. The RMSE value provides an estimate of the average magnitude of the prediction errors. A lower RMSE indicates better model performance, as it signifies smaller differences between predicted and actual values.

2. Mean Absolute Error (MAE):

MAE stands for Mean Absolute Error, which is another commonly used metric for evaluating the accuracy of regression models. It measures the average magnitude of the differences between predicted and actual values. The equation for MAE is as follows:

$$\text{MAE} = \frac{1}{n} * \sum |y_i - \hat{y}_i|$$

To calculate MAE, we take the absolute difference between the actual and predicted values for each data point, sum these absolute differences, and divide by the number of data points (n). Unlike RMSE, MAE does not involve squaring the differences, which means it is not as sensitive to outliers.

3. R-Squared (R^2):

R-squared (R^2) is a commonly used metric in regression analysis that represents the proportion of the variance in the target variable that can be explained by the independent variables in the model. It measures the goodness of fit of the regression model to the observed data. The R-squared value ranges between 0 and 1. Here's the equation to calculate R-squared:

$$R^2 = 1 - \frac{SSR}{SST}$$

- SSR (Sum of Squared Residuals) is the sum of the squared differences between the predicted values ($\hat{y}$) and the actual values (y) of the target variable.
- SST (Total Sum of Squares) is the sum of the squared differences between the actual values (y) and the mean of the target variable.

In essence, R-squared measures the proportion of the total variation in the target variable that is accounted for by the regression model. A value of 1 indicates that the model perfectly predicts the target variable, while a value of 0 suggests that the model fails to explain any of the variance.

4. Percentage of Prediction (PRED):

"Percentage of Prediction" (PRED) is a customized performance measure that assesses the proportion of estimated values that fall within a certain threshold (less than 0.25) of the corresponding actual values. It is used as an additional criterion to evaluate the accuracy of the predictions made by a regression model. Obtain the estimated values (predictions) and the corresponding actual values from the regression model. Calculate the absolute difference between each estimated value and its corresponding actual value. Count the number of estimated values for which the absolute difference is less than 0.25. Divide this count by the total number of estimated values. Multiply the result by 100 to obtain the percentage of predictions (PRED).

$$PRED = \frac{N\_within\_threshold}{N\_total} * 100$$

Here, N_within_threshold is the count of estimated values that fall within the specified threshold (less than 0.25) of the actual values and N_total is the total count of estimated values. The resulting PRED value represents the proportion or percentage of estimates that are within less than 0.25 of the corresponding actual values.

A higher PRED value indicates a better model performance in terms of accurately predicting values within the given threshold.

CHAPTER-5

RESULTS AND ANALYSIS

5.1 Linear Regression:

TABLE I: Results of Linear Regression model

	All Features				Optimal Features			
Dataset	**MAE**	**RMSE**	**R^2**	**PRED**	**MAE**	**RMSE**	**R^2**	**PRED**
Albert	9.782	11.010	0.903	0.108	4.530	7.370	0.9579	0.225
Kemerer	217.889	232.202	-5.563	0.021	79.778	94.467	-0.359	0.160
Desharnais	1645.909	2270.211	0.596	0.013	1531.77	1924.817	0.725	0.115
Cocomo	1383.699	1807.787	-9.575	0.028	349.91	666.462	-0.437	0.142

We can observe that from the above table the values are calculated based on Features. The left side contains with all features i.e., without Feature selection. On the right side of the table, we can observe that the model is trained with only optimal features which are identified after feature selection using Harmonic Search Algorithm. We can observe that four different datasets are trained on Linear Regression model which gives four performance measures MAE, RMSE, R^2 and PRED. From table the performance measure values are obtained better that all features. Hence, we can say that model performs better with Feature Selection than with all Features as input. Selection of only some optimal Features decreases complexity and decreases those Features which cause negative effect on target Feature i.e., Effort. There are some R^2 values negative which means our model is not able to predict values properly but we can observe that upon optimization using HSA Feature Selection we got R^2 values better which means the model can predict better before. We can clearly note the increase in PRED that is better prediction with more accuracy after optimization. Same with MAE and RMSE also decreased after optimization which means closer predictions. Hence, HSA has given better result through optimization with Feature Selection and made Linear Regression model to perform even better. Although Linear regression cannot perform better than Random Forest regression model.

5.2 Random Forest Regression:

TABLE II: Results of Random Forest Regression model

	All Features				Optimal Features			
Dataset	**MAE**	**RMSE**	**R^2**	**PRED**	**MAE**	**RMSE**	**R^2**	**PRED**
Albert	9.786	15.089	0.821	0.123	7.280	10.467	0.902	0.218
Kemerer	71.595	86.302	0.101	0.102	42.356	57.056	0.685	0.187
Desharnais	1778.533	2149.371	0.638	0.052	1603.559	2029.425	0.745	0.132
Cocomo	277.901	516.775	0.136	0.036	228.19	323.252	0.616	0.174

We have four different datasets are trained on Random Forest Regression model which gives four performance measures MAE, RMSE, R^2 and PRED. From table the performance measure values are obtained better that all features. We can say that model performs better with Feature Selection than with all Features as input. Selection of only some optimal Features decreases complexity and decreases those Features which cause negative effect on target Feature i.e., Effort. As the model is performing better compared to other models there are no R^2 values negative values which means our model can predict values properly but we can observe that upon optimization using HSA Feature Selection we got R^2 values better which means the model can predict better before. We can clearly note the increase in PRED that is better prediction with more accuracy after optimization. Same with MAE and RMSE also decreased after optimization which means closer predictions. Hence, HSA has given better result through optimization with Feature Selection and made Random Forest Regression model to perform even better. Random Forest Regression model has performed better than other models with no negative R^2 values and gave better results with more optimisation. Although for "Desharnais" dataset it gave more MAE and RMSE values compared to others which means Random Forest not suitable for "Desharnais" dataset while it performs best on other datasets.

5.3 Ridge Regression:

TABLE III: Results of Ridge Regression model

	All Features				Optimal Features			
Dataset	**MAE**	**RMSE**	**R^2**	**PRED**	**MAE**	**RMSE**	**R^2**	**PRED**
Albert	7.887	9.751	0.925	0.112	4.660	7.317	0.964	0.251
Kemerer	159.239	178.269	-2.835	0.044	65.277	81.656	0.1954	0.196
Desharnais	1635.979	1992.049	0.689	0.032	1511.33	1866.939	0.729	0.154
Cocomo	1211.641	1562.527	-6.901	0.042	342.66	646.744	-0.353	0.183

The four different datasets are trained on Ridge Regression model which gives four performance measures MAE, RMSE, R^2 and PRED. From table the performance measure values are obtained better that all features. We can say that model performs better with Feature Selection than with all Features as input. Selection of only some optimal Features decreases complexity and decreases those Features which cause negative effect on target Feature i.e., Effort. R^2 values there are some negative values which means our model is not able to predict values properly but we can observe that upon optimization using HSA Feature Selection we got R^2 values better which means the model can predict better before. We can clearly note the increase in PRED that is better prediction with more accuracy after optimization. Same with MAE and RMSE also decreased after optimization which means closer predictions. Hence, HSA has given better result through optimization with Feature Selection and made Random Forest Regression model to perform even better. Although Ridge regression cannot perform better than Random Forest regression model but it performed better when compared with Linear and SVM regression. We can see that Ridge regression has performed better on "Desharnais" dataset compared to Random Forest Regression model. Although it cannot perform better on "Cocomo" dataset with negative R^2 value. While, Random Forest Regression gave good R^2 values but has higher MAE and PRED values compared to Ridge and Linear Regression. This is a great drawback for Random Forest model even it can recognise dataset better but cannot predict with greater accuracy.

5.4 SVM (Support Vector Machine):

TABLE IV: Results of SVM Regression model

	All Features				Optimal Features			
Dataset	**MAE**	**RMSE**	**R^2**	**PRED**	**MAE**	**RMSE**	**R^2**	**PRED**
Albert	18.180	30.046	0.289	0.202	16.650	26.917	0.429	0.213
Kemerer	184.567	205.795	-4.110	0.015	171.562	195.896	-0.117	0.145
Desharnais	5723.236	4471.765	-1.567	0.033	5523.15	4350.452	-0.536	0.138
Cocomo	294.359	617.341	-0.233	0.045	291.546	616.223	-0.2302	0.156

We have four different datasets are trained on SVM Regression model which gives four performance measures MAE, RMSE, R^2 and PRED. We can clearly say that model performs better with Feature Selection than with all Features as input. Selection of only some optimal Features decreases complexity and decreases those Features which cause negative effect on target Feature i.e., Effort. R^2 values there are some negative values which means our model is not able to predict values properly but we can observe that upon optimization using HSA Feature Selection we got R^2 values better which means the model can predict better before. We can clearly note the increase in PRED that is better prediction with more accuracy after optimization. Same with MAE and RMSE also decreased after optimization which means closer predictions. SVM regression performed worse compared to all other models. It had not given that much better results like other models even after optimization which means LinearSVM is not a good model for Regression problem because even otimization cannot improve the model. Optimization on SVM model has not brought great change but we can see optimization in R^2 values. Hence, HSA has given good results through optimization with Feature Selection and made SVM Regression model to perform good but not better compared to other models.

Pairwise Relationships between Features and RMSE

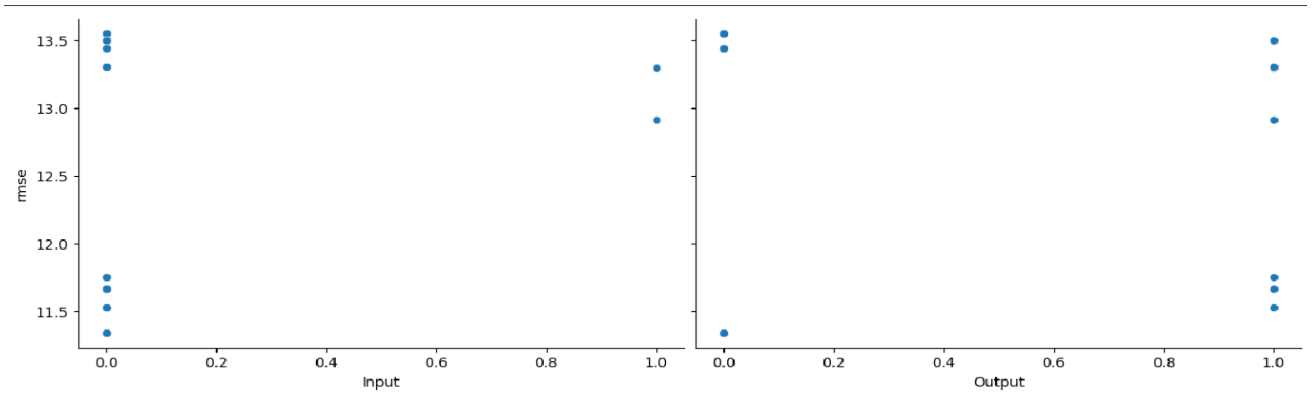


Figure 5-1

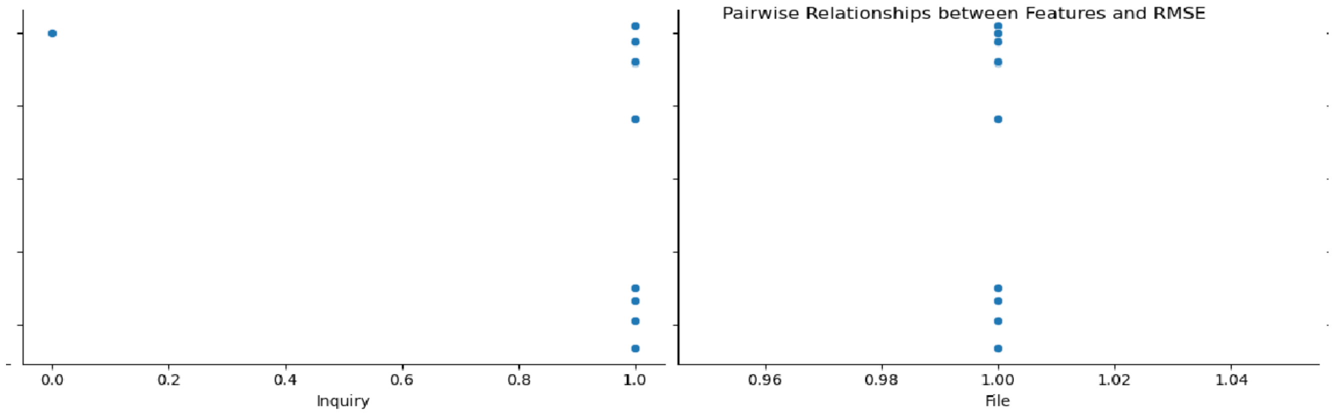


Figure 5-2

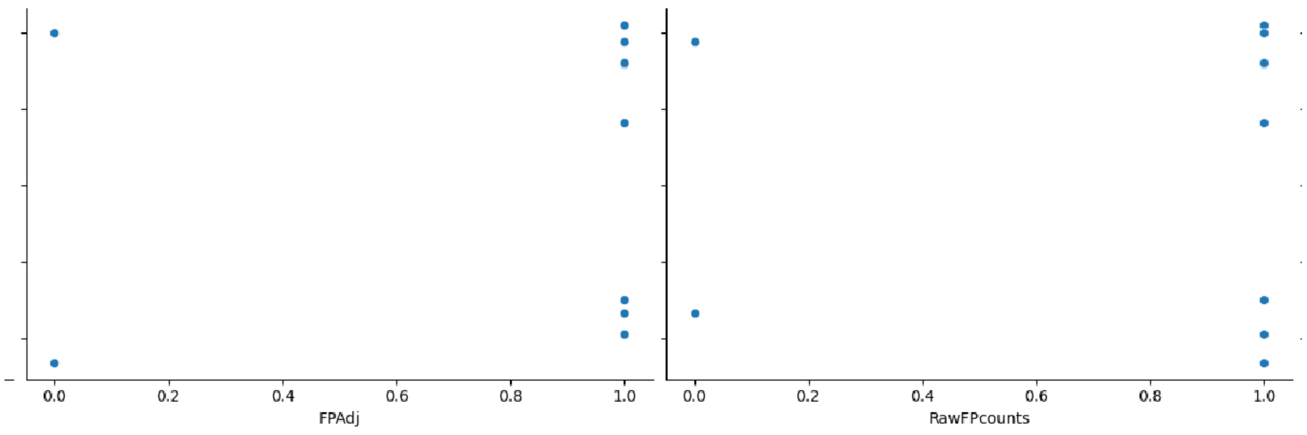


Figure 5-3

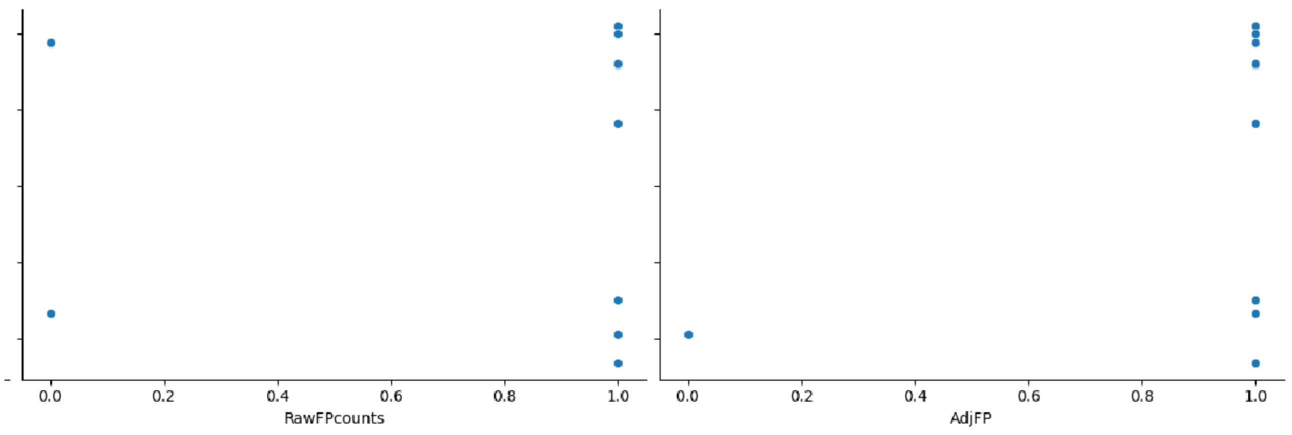


Figure 5-4

Scatter plot of Features vs RMSE

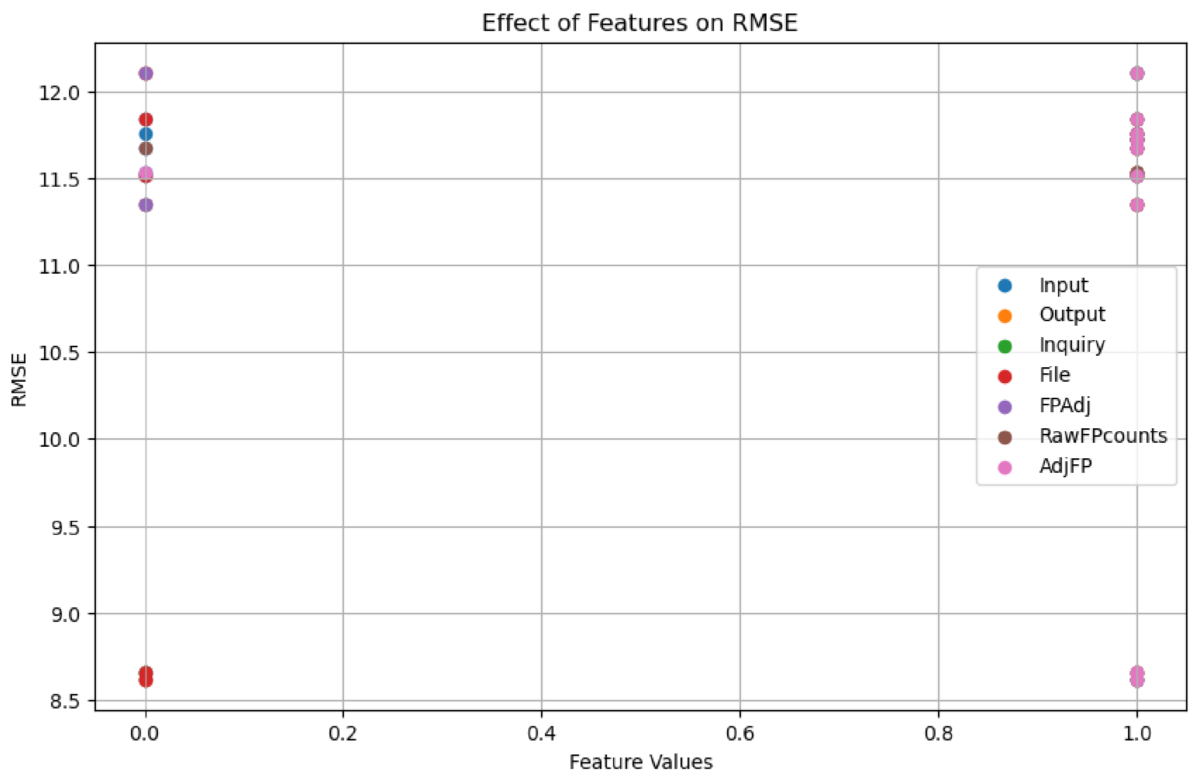


Figure 5-5

The plots shown in the above figures denote the Effect of each feature on the RMSE a performance measure value. We can observe that the plot show in *Figure 5-5* it is a scatter plot of every feature vs RMSE whereas the other plots above represent the pairwise relationships between each feature and RMSE which shows effect of each feature on RMSE. We can observe from the plots that on selecting some features frequently we are getting good RMSE values that is minimum values for a minimisation problem. But we can also observe from the same plot that upon not selecting some features we are getting optimal RMSE value whereas upon selecting the same feature we get higher RMSE (for a maximisation problem). Hence, we can say that by not considering those features which does not give better results we can get optimal results. The features which are responsible for optimal results are optimal features. Removal of features makes problem less complex and less time taking hence giving better results, hence optimises the problem. Identifying those optimal features require optimization methods. Among which Harmony Search Algorithm performs better optimization to give optimal results.

CHAPTER-6

CONCLUSION

The Project successfully demonstrated the application of machine learning techniques in predicting the Software Effort Estimation. HSA has improved the model's accuracy and reliability offer valuable insights to Software Engineers, enabling them to make informed decisions and optimize machine learning Practices for improved productivity. HS optimizes features in this model during the training phase, and these optimized weights are then used in the testing stages for model evaluation. HSA provides better PRED and R^2 values with minimized MAE and RMSE values. We conducted experiments on only four publicly available datasets and four performance measures; we are unsure how our method will perform with other commercial datasets and measures. However, we provided a detailed procedure for implementing our method. For future studies, we will consider comparing HSA with some state-of-the-art effort estimation models and will consider performing statistical comparison analysis to validate the model.

REFERENCES

[1] Gholami, Jafar, Farhad Pourpanah, and Xizhao Wang. "Feature selection based on improved binary global harmony search for data classification." *Applied Soft Computing* 93 (2020): 106402.

[2] Diao, Ren, and Qiang Shen. "Two new approaches to feature selection with harmony search." *International Conference on Fuzzy Systems*. IEEE, 2010.

[3] Manchala Pravali, and Manjubala Bisi. "Ensembling Teaching-Learning-Based Optimization Algorithm with Analogy-Based Estimation to Predict Software Development Effort." *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2022.

[4] Rathod, Kapil Ravi, et al. "Learning Automata based Feature Selection for Duplex Output Software Effort Estimation Model." *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021.