Random Forest Hyperparameters Tuning for Software Development Effort Estimation and its Comparison with Regression Tree

Submitted by

Deepak Kumar
V00889149

Department of Computer Science
University of Victoria

Contents

1 Introduction **2**

2 Background **3**

3 Datasets used for experiment **4**

3.1 Desharnais 4

3.2 Albrecht 4

3.3 COCOMO81 5

4 Performance Evaluation Metrics **5**

5 Experiment Setup and Design **6**

5.1 Configuration of Random Forest Models 7

5.2 Configuration of Regression Tree Models 8

6 Result **9**

6.1 Effect of hyperparameter on RF model Accuracies 9

6.2 RF accuracies with Grid search and Randomized Search 12

6.3 Comparison between Random Forest model and Regression Tree . . . 12

7 Conclusion **13**

Acknowledgement **14**

References **15**

1 Introduction

The software industry is one of the fastest growing industry in recent years. The success of the software industry depends mostly on project management. One of the critical tasks in software project management is to find out the software development cost and efforts during the early stages of the project. A useful software effort estimation allows project managers to manage the resources, and tasks more productively. However, the lack of good software effort estimation may incur several issues like wrong budget allocation, challenging resource management, risk of delayed project delivery, poor quality of services, and team management issues.

There are several traditional methods for software development effort estimation [3] [6] [14] [2]. However, There has been growing interest to solve the effort estimation problem using machine learning techniques. Out of these machine learning techniques, regression tree based models have been focused more by researchers due to its ability of generalization and understandability. However, in paper [15], authors focused on establishing useful Random Forest model and compared the results with Regression Tree model. They performed some empirical experiments and tried to find the best Random Forest models by performing hyperparameters tuning for the values of its key parameters. I have tried to replicate the experiments done in paper [15]. Apart from these experiments, I am also performing hyperparameters tuning using some other methods called grid search and randomized search. The grid search method takes all the parameters and builds the RF models using all the exhaustive combinations of all parameter values and returns the best estimate model with the best parameters. Some times, there are a large number of parameters having many possible values. In that case, There might be a huge number of exhaustive combination of all the parameters values, which might be computation resources and time-consuming task. To resolve this, there is another method called randomized search which is used for random searching for best hyperparameters. In this method instead of searching for all combination of hyperparameters, a fixed number of parameter settings is sampled from the specified distributions.

The experiments are performed on some popular software effort estimation datasets available publicly; COCOMO81, Desharnais and Albrecht. To evaluate the models and measure the accuracy, three metrics have been used; MMRE, MdMRE, and Pred(0.25). In the following sections, I am discussing the experiments and their results.

2 Background

There has been a lots of research to compare machine learning models to find the accurate effort estimation [13] [8] [10] [9]. The ML methods that have been researched include Artificial Neural networks(ANN), Decision Trees(DT), Regression Trees(RT), Bayesian Networks(BN), Support Vector Regression(SVR).

The decision tree is one of the popular technique to build predictive models. It can be divided into two categories; Classification Tree and Regression Tree. The classification tree is used when the output class is an observation from ordinal or nominal data class. While Regression Trees are used when the output value to be predicted is from the interval domain. In an experiment [13], authors found that Regression Tree approach was more successful than COCOMO and SLIM for software effort estimation when MRE was considered as the performance metric. However, It was less successful than backpropagation based neural network. In [11], Selby and porter used ID3 algorithm to generate a large number of decision trees to classify software modules with high development efforts. To handle the uncertainty and imprecise data of software development project, authors, in [8] investigated the use of fuzzy ID3 decision trees. It is designed by combining the algorithm of ID3 and concepts of fuzzy set theory. They considered MMRE and pred as measures of prediction accuracy. They compared the result with other decision tree based models; ID3, CART and C4.5 and found fuzzy ID3 method more accurate over other mentioned methods.

As we can see, there has been a lot of research work done to improve the accuracy of SDEE using decision tree based models. However, there is another machine learning technique called Random Forest, which has not gained much focus of the SDEE research community. Random Forest is a supervised learning method which creates a forest using an ensemble of decision trees and makes it random somehow. Random Forest creates multiple decision trees and merges them to achieve more accurate and stable prediction. Papers [10] [12] has used DTF(Decision Tree Forest) and Random Forest to create predictive models for SDEE. In [10], authors, compared MLR (Multiple Linear Regression), DT (Decision Tree) and DTF (Decision Tree Forest). They used ISBSG R10 and Desharnais datasets and 10-fold cross-validation to implement the DTF. Results represented that DTF outperformed DT and MLR, when accuracy measured in terms of MMRE, MdMRE and Pred(.25). Satapathy [12] used RF to optimize the effort parameters to gain higher accuracy. He compared the results, with the multilayer perceptron, radial basis function network stochastic gradient boosting and log-linear regression techniques.

3 Datasets used for experiment

This section contains the details about the datasets being used to replicate the experiments performed in [15]. There are many traditional datasets available for SDEE publicly. I am using a few of these data sets as described below.

3.1 Desharnais

This dataset was created by Jean-Marc Desharnais in 1988 [4]. It is one of then oldest dataset available for SDEE. Hence, It has been used in many empirical studies like [10] [7] [9]. This dataset consists of data from 81 software projects from a Canadian software company. These 81 projects have been sub-grouped based on their technical environment as 46 projects of the traditional environment, 25 projects as "Improved" traditional environment and 10 projects as the micro environment. Each project has 10 features out of which 9 features(TeamExp, ManagerExp, Length, Transactions, Entities, PointsNonAdjust, Adjustment, PointsAjust, Language) are independent, and 1 feature(Effort) is dependent.

3.2 Albrecht

This dataset is created by Allan J. Albrecht [1]. It is also one of the popular datasets for SDEE and data of contains 24 projects implemented by IBM DP Services organization. The data includes the count of 4 types of external input/output elements for the application as a whole, number of "SLOC" including comments, number of function points for each project. The effort is given as the number of work hours required to design, develop and test the application. Hence, there are total seven features in this dataset out of which 5 features(Input count, Output count, Inquiry count, File count, and SLOC) are independent, and two features(Function Points and Efforts) are dependent.

3.3 COCOMO81

This dataset contains data of 63 projects [3]. All these projects contains 17 attributes out of which one is actual effort, one is line of code measured in terms of KSLOC(Kilo Source Line of Code), and 15 attributes(analysts capability, programmers capability, application experience, modern programming practices,use of software tools, virtual machine experience, language experience, schedule constraint, main memory constraint, data base size, time constraint for cpu, turnaround time, machine volatility, process complexity, required software reliability) are effort multipliers. Effort multiplier attributes are based on the software development environment and are measured on the scale of six linguistic values; 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'.

4 Performance Evaluation Metrics

In this section, I am describing the metrics used by the original paper as performance evaluation criteria. As discussed before, there are lots of traditional methods and ML models are available for SDEE. To find which model/method is better, the accuracy of the models has be calculated. The estimation models can be evaluated using several evaluation criteria. In [15], authors have considered the three most common evaluation criteria as follows:

MMRE

One of the most commonly used evaluation criteria is Mean Magnitude Relative Error (MMRE) [5]. MMRE is the mean value of Magnitude of Relative Error(MRE) that can be defined as

$$MRE = \frac{\mid E_{actual} - E_{eastimated} \mid}{E_{actual}} \tag{1}$$

Where E_{actual} is the actual effort value and $E_{eastimated}$ is the predicted value by ML model. Hence, the MMRE can be calculated as below:

$$MMRE = \frac{1}{n}\sum\nolimits_{i=1}^{n} MRE_i \tag{2}$$

Where n is the number of records in the dataset and MRE_i is Magnitude of Relative

Error for i_{th} record.

MdMRE

While being the most popular evaluation metric, MMRE has the disadvantage of being sensitive to the outliers. For this reason, another evaluation criteria is used called Median of Magnitude of Relative Errors(MdMRE) defined in equation 3.

$$MdMRE = median(MRE_i) \tag{3}$$

PRED(x)

MRE is not reliable if individual items are skewed. In that case, PRED(x) is used as the measure of estimation accuracy. PRED(X) is the percentage of predicted values that are within x percent of the actual value. It can be defined as in equation(4).

$$PRED(x) = \frac{k}{n} \tag{4}$$

where k is the number of projects where $MRE_i \leq x$ and n is the total number of observations. The most common value of x is 0.25 which is also used in this experiments The Pred(0.25) represents the percentage of projects whose MRE is less or equal to 0.25.

5 Experiment Setup and Design

In this section, I am explaining the experiment setup and configurations done to predict software development efforts using Random Forest and Regression Tree methods. Original paper [15] used the R programming language to perform the experiment. However, I have replicated the experiments using the Python programming language. I have used a few machine learning libraries (like Scikit-learn , Numpy, Pandas, etc.) written to inter-operate with the Python programming language to perform the experiments. To visualize the results, Matplotlib is used which is a data visualization library written in Python.
70% of the data is used for training the ML models and rest 30% data is used for

testing. None of the testing data is participating in the training model to avoid overfitting and to get better generalization. The Generalization in machine learning is the ability of an algorithm to perform well on new examples of data that are not part of the data that it was trained on.
The experiments are performed with 10-fold cross validation where the original sample is randomly partitioned into 10 equal size subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds are averaged to produce a single estimation. Cross-validation is performed only on 70% training set.

5.1 Configuration of Random Forest Models

Random Forest is a classifier which fits the number of decision trees on various subsamples of data and then averages it to improve the prediction accuracy. Prediction using RF model depends on various factors like number of decision trees used, the number of features used at each node to spilt, which method is being used to calculate information gain(like gini, entropy), number of samples, maximum depth of the trees. To identify the values of parameters which will result in the best RF model, the hyperparameter tuning approach has been taken rather than theory. During hyperparameter tuning, RF model is created with the different combination of the different parameter value. Then the values of parameters that yield the best prediction model are considered as best suitable values for the model.
Authors of the original paper considered two hyperparameters for experiments; the number of decision trees in the forest (n_estimators) and the number of features to consider when looking for the best split (max_features). The table 1 explains the design of experiments.

It should be noted that the original paper uses libraries in the R language and I am using Python libraries to perform experiments. Both may have different default values of the parameters other than hyperparameters considered for these experiments. Apart from experiments from the original paper [15], I have also performed grid search and randomized search to search the best model and hyperparameters. For grid search I have considered same parameters; n_estimators and max_features and setup is shown in the table 2 . However, since in randomized search, we can limit the parameter settings, I have tried to explore other parameters as well like , maximum depth of the tree (max_depth), The minimum number of samples required to split an internal node (min_samples_split), The minimum number of samples required to be at a leaf node (min_samples_leaf). The setup is shown in table 3.

Table 1: Experimental Setup for hyper parameter tuning of Random Forest Model

Datasets	**Experimental Study 1 varying max_features**		**Experimental Study 2 varying n_estimators**	
	max_features	**n_estimators**	**max_features**	**n_estimators**
Albrecht	1 to 6	500	5	100 to 1000
Desharnais	1 to 8	500	5	100 to 1000
COCOMO81	1 to 16	100	7	100 to 2000

Table 2: RF hyperparameters settings for exhaustive grid search

Datasets	**max_Features**	**n_estimators**
Albrecht	1 to 6	100 to 2000
Desharnais	1 to 8	100 to 2000
COCOMO81	1 to 10	100 to 2000

Table 3: RF hyperparameters settings for randomized search

Datasets	**max_Features**	**n_estimators**	**bootstrap**	**max_depth**	**min_samples_leaf**	**min_samples_split**
Albrecht	1 to 6	100 to 2000	True, False	10 to 100, None	1,2,3,4	1,2,5,10
Desharnais	1 to 8	100 to 2000	True, False	10 to 100, None	1,2,3,4	1,2,5,10
COCOMO81	1 to 10	100 to 2000	True, False	10 to 100, None	1,2,3,4	1,2,5,10

5.2 Configuration of Regression Tree Models

For the Regression Tree model, I have followed the same configuration as the original paper. They choose the minimum size of the node to split as 20 and the maximum depth of the tree as 30. They have chosen the parameters values such that error is minimum. The same configuration is used to create a Regression Tree model with all the datasets used in experiments.

6 Result

This section describes the results of experiments performed to build better Random Forest model for software development effort estimation and also compares this with Regression Tree models.

6.1 Effect of hyperparameter on RF model Accuracies

Experiments are performed as per the hyperparameter setting presented in Table 1. For each hyperparameter settings, a Random Forest model is built and validated on 30% test data. Here I am showing the values of different evaluation metrics(MMRE, MdMRE and PRED(25)) for each hyperparameter setting.

Albrecht Figure 1 and 2 shows the results of first study where the value of n_estimator is 500 and values of max_features varies from 1 to 6. Figure 1 shows that MMRE, MdMRE decrease and max_features increases. Figure 2 shows that Pred(25) decreases for max_features's value more than 4.

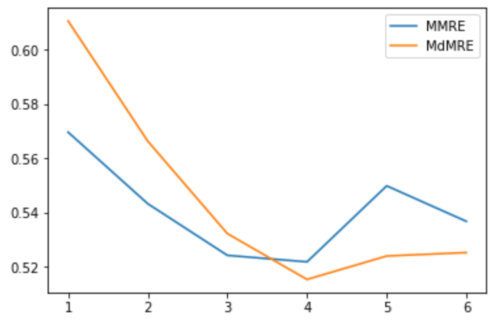

Figure 1: MMRE and MdMRE with respect to max_features for Albrecht

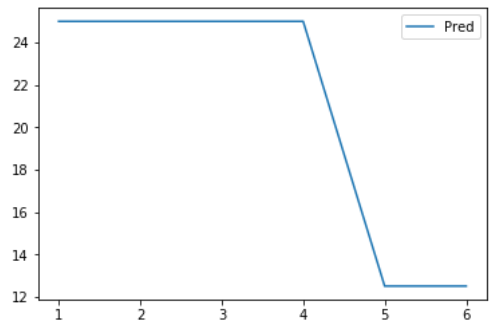

Figure 2: PRED respect to max_features for Albrecht

Figure 3 and 4 shows the results of second study where the values of n_estimator varies from 100 to 1000 and value of max_features varies from 5. Figure 3 shows that MMRE and MdMRE decreasing with slight fluctuation when n_estimator increases. However, there is no clear pattern of PRED(25) in Figure 4.

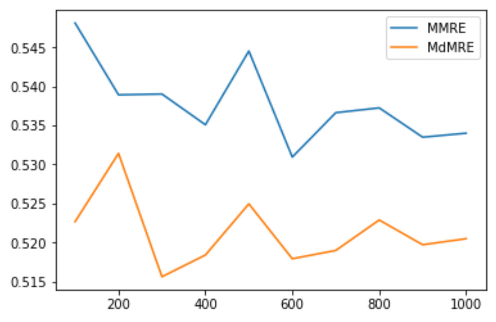

Figure 3: MMRE and MdMRE with respect to n_estimators for Albrecht

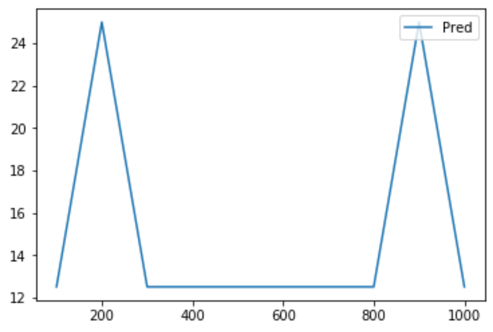

Figure 4: PRED(25) with respect to n_estimators for Albrecht

Desharnais Figure 5 and 6 displays RF model accuracies for n_estimator=500 and max_features= 1 to 8. We can see MMRE and MdMRE varying slightly with different values of max_features. However, there is a huge gap between values of MMRE and MdMRE which indicates the presence of some outlier values of MRE. Figure 6, shows that the values of PRED(25) decreased when max_features increased.

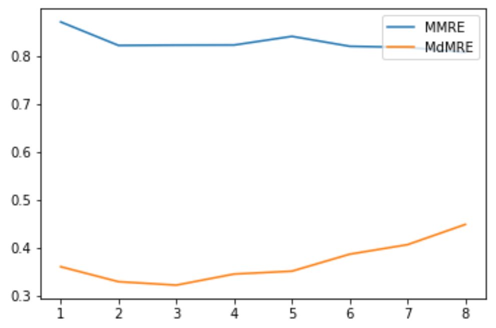

Figure 5: MMRE and MdMRE with respect to max_features for Desharnais

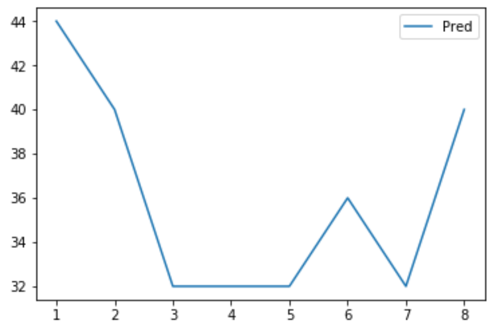

Figure 6: PRED(25) with respect to max_features for Desharnais

Figure 7 and 8 displays RF model accuracies for n_estimator varying from 100 to 100 and max_features equals to 5. Figure 7 shows that MMRE and MdMRE varying very slightly with n_estimator's values which means n_estimator has a minor impact on MMRE and MdMRE. However, there is no clear pattern for PRED(25) as shown in Figure 8

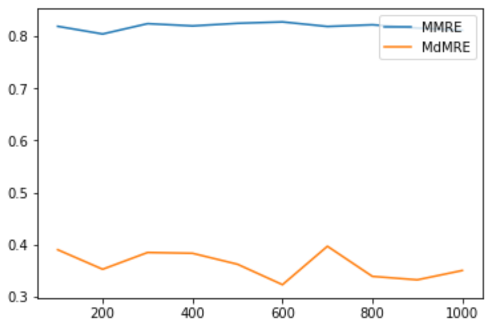

Figure 7: MMRE and MdMRE with respect to n_estimators for Desharnais

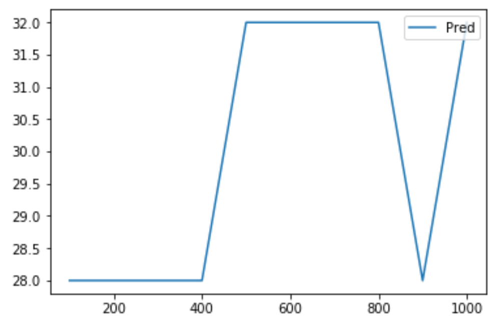

Figure 8: PRED(25) with respect to n_estimators for Desharnais

COCOMO81 Figure 9 and 10 shows the results where n_estimator=500 and max_features= 1 to 10. Figure 9 depicts that MMRE and MdMRE increase when number of features increase. There is not clear pattern for PRED(25) in Figure 10 Figure 11 and 12 shows the results for second experiment study where n_estimator

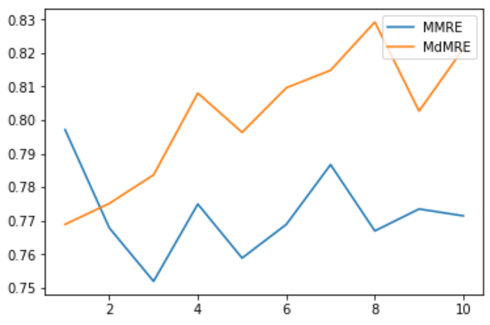

Figure 9: MMRE and MdMRE with respect to max_features for COCOMO81

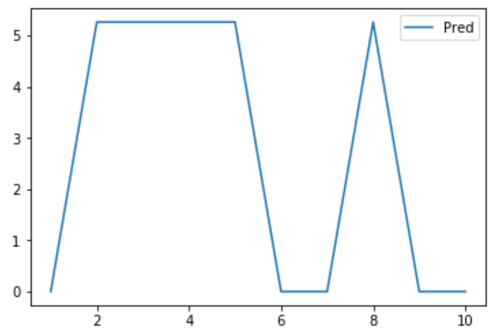

Figure 10: PRED(25) with respect to max_features for COCOMO81

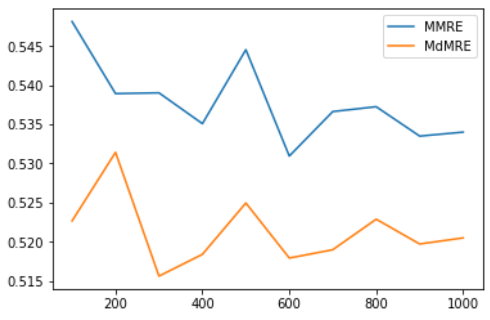

Figure 11: MMRE and MdMRE with respect to n_estimators for COCOMO81

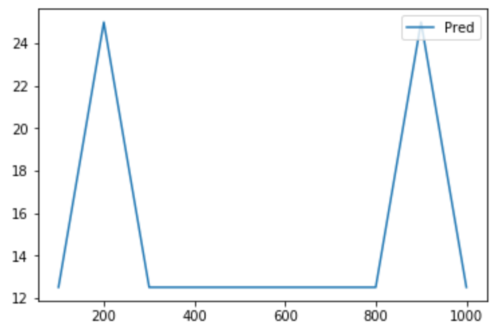

Figure 12: PRED(25) with respect to n_estimators for COCOMO81

varies from 100 to 2000 and max_features= 7. Figure 11 shows that MMRE and

MdMRE decreases with some fluctuations when n_estimator increases.

6.2 RF accuracies with Grid search and Randomized Search

Gird search exhaustively trained RF model for all possible combination of hyperparameters. Table 4 shows the values of MMRE, MdMRE and PRED(25) for the best RF model found during grid search. It also contains values of hyperparameters which resulted in the best RF model.
Table 5 contains the value of MMRE, MdMRE and PRED(25) of the best RF model found during the randomized search with the best values of hyperparameters.

Table 4: Evaluation metrics and best hyper parameters resulted from grid search method

Datasets	**MMRE**	**MdMRE**	**PRED(25)**	**Best Parameters**	
				max_features	**n_estimators**
Albrecht	0.518	0.514	25	6	200
Desharnais	0.899	0.361	40	1	400
COCOMO81	0.768	0.817	5.26	9	200

Table 5: Evaluation metrics and best hyper parameters resulted from randomized search method

Datasets	**MMRE**	**MdMRE**	**PRED(25)**	**Best Parameters**					
				max_features	**n_estimators**	**max_depth**	**min_samples_leaf**	**min_samples_split**	**bootstrap**
Albrecht	0.493	0.465	25	6	1200	30	4	10	False
Desharnais	0.951	0.361	36	1	1800	30	4	10	True
COCOMO81	0.756	0.803	5.26	9	800	70	1	2	False

6.3 Comparison between Random Forest model and Regression Tree

In the previous sections, we discussed different experiments and their results to do hyperparameters tuning and to find the best RF model. We have also trained Regression Tree(RT) models using all the datasets for SDEE and validated the models using test data. In this section, I am comparing the Random Forest model and Regression Tree model based on MMRE, MdMRE, and PRED(25). Table 6 shows

the comparison of accuracies of both the models for all datasets used in experiments. We can see that the accuracies of Random Forest model is either better or the same in all the cases.
In figure 13, we can see that for Albrecht dataset, MMRE of RF is far better than RT. In the case of Desharnais, MMRE of RF is slightly better than RT, and for COCOMO81, MMRE for both the models is almost same.
Figure 14 shows the comparison of based on MdMRE. It is same for both the model for Albrecht dataset, however for Desharnais and COCOMO81, MdMRE of RF is better than MdMRE of RT. Figure 15 shows that PRED(25) does not vary a lot with RF and RT. PRED(25) is same for both the models for Albrecht and COCOMO81 datasets. In case of Desharnais, PRED(25) of RF is better than RT.

Table 6: Comparison of evaluation metrics of Random Forest and Regression Trees for SDEE

Datasets	**Random Forests**			**Regression Trees**		
	MMRE	**MdMRE**	**PRED(25)**	**MMRE**	**MdMRE**	**PRED(25)**
Albrecht	0.493	0.465	25	0.840	0.472	25
Desharnais	0.821	0.329	40	0.877	0.504	36
COCOMO81	0.751	0.783	5.26	0.756	0.839	5.26

7 Conclusion

Accurate software development effort estimation is crucial for the success of software projects. In this paper, I have replicated the experiments done in paper [15] which is intended to study the use of the Random Forest for Software development Effort Estimation. Three popular SDEE datasets are used in this paper for experiments; Albrecht, Desharnais, and COCOMO81. The experiments are performed to find the impact of hyperparameters on estimation model. To achieve that, I did the experiments with hyperparameters settings as in the original paper. Apart from this, I also performed the exhaustive grid search and randomized search to find the best hyperparameters for Random Forest estimation model. Then, the Random Forest model is compared with the Regression Tree model using 30% holdout validation method. Both models are compared based on three evaluation metrics; MMRE, MdMRE and PRED(25). The results presented that tuning the hyperparameter improves the accuracy of the RF model and results in the best RF model. Results

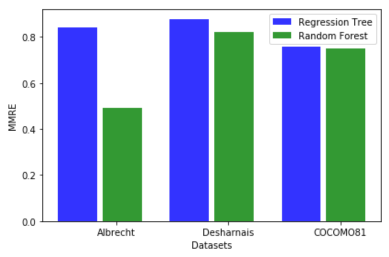

Figure 13: Comparison of MMREs for RF and RT models

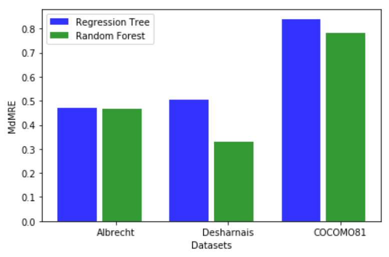

Figure 14: Comparison of MdMREs for RF and RT models

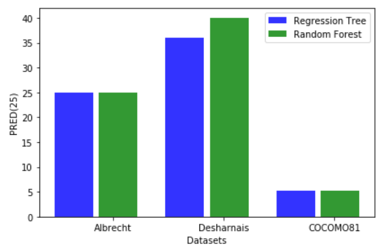

Figure 15: Comparison of PRED(25) for RF and RT models

also showed that the Random Forest model exceeds the Regression Tree model in all the evaluation criteria for all three datasets.

Acknowledgement

I would like to thanks Dr. Neil Ernst for his support and helping me to find the Datasets for SDEE.

References

[1] A.J. Albrecht and Jr. Gaffney, J.E. Software function, source lines of code, and development effort prediction. *A Software Engineering, IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.

[2] Barry Boehm, Chris Abts, and Sunita Chulani. Software development cost estimation approaches — a survey. *Annals of Software Engineering*, 10(1):177–205, Nov 2000.

[3] Barry W. Boehm. *Software Engineering Economics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1981.

[4] J.-M. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. *Master thesis, University of Montreal*, 11, 1988.

[5] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29(11):985–995, Nov 2003.

[6] F.J. Heemstra. Software cost estimation. *Information and Software Technology*, 34(10):627 – 639, 1992.

[7] A. Idri and I. Abnane. Fuzzy analogy based effort estimation: An empirical comparative study. In *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pages 114–121, Aug 2017.

[8] Ali Idri and Sanaa Elyassami. A fuzzy decision tree to estimate development effort for web applications. 2011.

[9] Ekrem Kocaguneli and Tim Menzies. Software effort models should be assessed via leave-one-out validation. *J. Syst. Softw.*, 86(7):1879–1890, July 2013.

[10] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho. A comparison between decision trees and decision tree forest models for software development effort estimation. In *2013 Third International Conference on Communications and Information Technology (ICCIT)*, pages 220–224, June 2013.

[11] A. A. Porter and R. W. Selby. Evaluating techniques for generating metric-based classification trees. *J. Syst. Softw.*, 12(3):209–218, July 1990.

[12] S. M. Satapathy, B. P. Acharya, and S. K. Rath. Early stage software effort estimation using random forest technique based on use case points. *IET Software*, 10(1):10–17, 2016.

[13] Krishnamoorthy Srinivasan and Douglas Fisher. Machine learning approaches to estimating software development effort. *IEEE Trans. Softw. Eng.*, 21(2):126–137, February 1995.

[14] Fiona Walkerden and R Jeffery. Software cost estimation: A review of models, process, and practice. *Advances in Computers*, 44:59–125, 12 1997.

[15] Abdelali Zakrani, Hain Mustapha, and Abdelwahed Namir. Software development effort estimation using random forests: An empirical study and evaluation. *International Journal of Intelligent Engineering and Systems*, 11:300–311, 12 2018.