IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

SOFTWARE PERFORMANCE PREDICTION USING RANDOM FOREST BASED REGRESSION ANALYSIS

R. Sathya[*], Dr. P. Sudhakar
Assistant professor, Department of Computer Science & Engineering,
Annamalai University, Annamalai Nagar, India.

DOI: 10.5281/zenodo.160860

ABSTRACT

The evaluation of various software quality metrics like performance, reliability, and response time are done using quantitative techniques and it is essential for component based software applications. In this paper the performance of the software application is predicted using regression analysis. In general, the trend analysis technique is employed to predict the performance of the software system. The proposed method in this paper will help the users of the software system to predict whether it satisfies their requirements for a set of features selected by them. The performance of the software gets vary based on the features selected by the users. The features may interact with other feature and degrade the overall performance of the system. The performance prediction is carried out using the Random forest which is capable of handling thousands of input variables without deleting the variables. Also it offers an experimental method for the detection of the feature interactions.

KEYWORDS: performance prediction, random forest, categorical data, feature interaction, regression analysis.

INTRODUCTION

In software engineering the systems are constructed by integrating many reusable components which are known as features. Adding a feature in to a software system will increase the functionality of the system. This methodology will require less time to build a system and decreased time for marketing, improved quality of the product, and the products profile can be diversified. But still the success of the compositional approach mainly relies on the modularity of the reusable components. The need for modularity is made essential right from the beginning of the software engineering era, and also now in the new technological fields like synthetic biology.

In a software system made from re-usable components called features, the interaction between the features occurs when the behavior of one feature influences or disturbs the behavior of another feature. In order to develop a highly secured system the developer must analyze all the possible outcomes, both the positive and negative outcomes, of the feature interaction [1]. In the case feature interaction problem, the fact is that it is not possible to isolate a feature. It communicates and co-operates with other features. But it is possible to detect or predict desired and undesired feature interactions in a scalable fashion.

The high dynamic quality of the features in self-adaptive systems, cloud computing, and dynamic product lines exhibits a new challenge to find a solution for feature interaction problem [2]. It is noticed that even in internet applications, requirements engineering, transport systems, computational sciences and many other fields which does not come under the computer science field it is hard to solve feature interaction problem. There cannot be a common solution for feature interaction problem prevailing in many of the systems; the solution for a system will differ entirely from the solution for another system. At present many systems are providing options to the user to select the needed features based on the application scenario. Rather than focusing on the conventional requirements the users pay attention to the requirements like minimal energy consumption and max response time. To meet the non-functional requirements a prior knowledge on which combination of features will decrease the performance of the system is essential. But it is infeasible to acquire knowledge about all combination of the features. So it will be better to identify the set of features whose combination doesn't degrade the performance of the system.

In this research the focus is about predicting the performance of the system in which the set of features is selected by the user. The prediction is made based on the previous recorded performance of the system for different combination of the features [3]. This prediction will help users to identify the set of features whose combination gives maximum performance.

DATASET

The focus in this research work is essentially on the configurable systems, which gives the users an option to select features based on their requirement. But because of the feature interaction problem, for a selected list of features the performance of the corresponding software may be reduced. To prevent these difficulties this work concentrates on developing a system which will predict the performance of the system based on a regression analysis using the Random Forest algorithm. Based on the predicted performance the user may change his preferred list of features. For our experiments and analysis out of the six real-world configurable systems namely Berkley DB (C), Berkley DB(Java), Apache, SQLite, LLVM, and x264, the SQLite is considered. The dataset is benchmark dataset generated by [4] and it contains performance measurement for different configuration of the system. In each of the system the performance has been measured by utilizing a conventional benchmark factor provided by the vendor of the software or used widely in the application domain. In this research work we have focused only on industrial programs rather than self-developed programs.

RANDOM FOREST BASED REGRESSION ANALYSIS

Decision trees are considered to be one of the most important machine learning algorithms, because its performance is not degraded even when the feature values are scaled or transformed. The accuracy of the trained model is not affected when irrelevant features are added. Especially the trees that are expanded deep have the ability to learn highly irregular patterns [5]. Because of low bias and very high variance generally they over fit their training sets. Random forest is a way of averaging multiple decision trees. The individual trees are trained with different parts of the same training set, aimed at reducing the variance. This is achieved normally by increasing the bias and losing interpretability to certain limit. Random forest is an ensemble based machine learning algorithm can also be used for regression analysis. It operates by forming a multitude of decision trees at the time of training. Then the model output the mean prediction of the individual trees [6].
The training algorithm for random forest utilizes the bootstrap aggregating technique or bagging for learning. Let us considered the training set $X = x_1, ..., x_n$ which has a response $Y= y_1, ..., y_n$. The bagging is done repeatedly on the data set for B times to select a random sample with replacement of the training set and to fit trees to these samples.

```
For $b = 1, ..., B$:
    1. Sample, with replacement, $n$ training examples from $X$, $Y$; call these $X_b$, $Y_b$.
    2. Train a decision or regression tree $f_b$ on $X_b$, $Y_b$.
```

Once training is completed the model can be used to predict the response for test samples, x' by averaging the predictions of the individual regression trees on x' or by taking the majority vote in the case of decision trees.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x') \qquad (3.1)$$

The bootstrapping method of learning increases the performance of the model by reducing the variance of the model. But at the same time the bias is not increased. Training many trees with a single training set will generate strongly correlated trees. Sometimes the same tree might be produced when the training algorithm is deterministic in nature. When the samples are bootstrapped to produce different training sets, the trees will be de-correlated. The value of B, the number of samples/trees is decided based on the size and nature of the training set. To find the optimal value of B, cross-validation method is used. In some of the application the value of B is fixed by verifying the out-of-bag error.

EXPERIMENTS AND RESULTS

Initially the cross-validation algorithm is applied to split the dataset in to training and testing. The cross-validation is used for the estimation of the level of fit of a model to a dataset that is independent of the data that were used to train the model.

4.1 Repeated random sub-sampling validation

The validation is implemented by following the steps below:

1. Assign each observation to any one of the groups namely training and validation randomly.
2. Apply the training set to the model and generate a model which classifies or predicts the class or response variable for a given test sample.
3. Using the samples from the test or validation dataset check the performance of the model and store this information.
4. Repeat steps 1 to 3 many times.

This method is also known as Monte Carlo cross-validation [7], which splits the dataset into training and validation randomly. For each split, the model is fit to the training data and the predictive accuracy is measured using the test set. The results are then averaged over the splits. The advantage of this method over the k-fold cross validation is that the percentage of the split between the training and validation does not depend on the number of iteration. When this method of validation is adopted the results may vary when the analysis is repeated with random splits. This kind of variation in the results is also termed as Monte Carlo variation.

4.2 Building Random forest model

After dividing the dataset in to training and testing set using the random sub-sampling validation method, the random forest regression model is built with 500 decision trees. Random Forest does not require split sampling method to estimate the accuracy of the built model. Self-testing possible even if all data is used for training as 2-3rd of available training data is used to grow any one tree and the remaining one-third portion of training data always used to calculate out of bag error to assess model performance.

The Out of bag error [8] is calculated for random forests during training, so there is no need for a separate test set to validate result. The error is calculated internally during the training time. While the forest is built using the training data, each decision tree is tested with the 1/3rd of the samples (36.8%) which are not utilized for the training the tree. The out of bag error estimate can be considered as an internal error estimate of a random forest while it is being constructed. The Fig. 1 shows the plot of the out of bag error estimated during training phase of the regression model. The plot indicates that there is no significant reduction in the error rate after 100 decision trees.

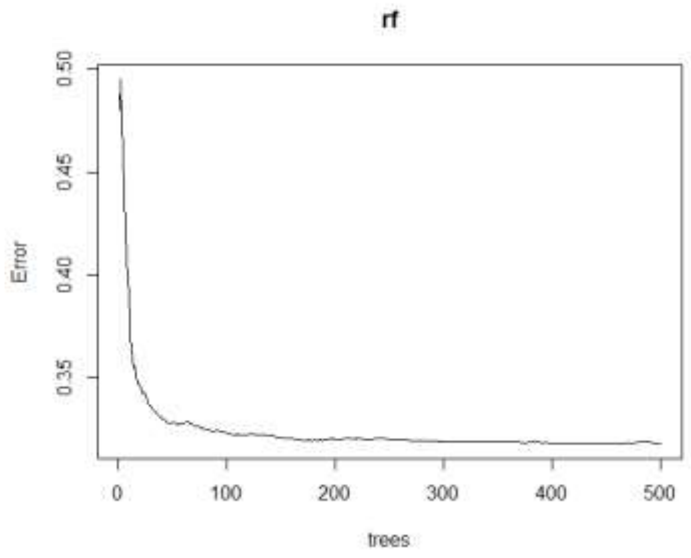

Fig. 1 Error plot for training regression model

After the error estimation then the variable importance is calculated using the two important measures namely %IncMSE and IncNodePurity.

1. %IncMSE – Using the permutation this measure is estimated from the test data. For each tree present in the random forest, the prediction error for the test samples is estimated. This procedure is repeated after permutation of the predictor variable. Over all trees the difference between the error for the normal variable and the error for the permutation of the predictor variable is averaged. Then it is normalized by the standard deviation of the differences. If the difference is higher, then the importance of the variable is also high.

$$MSE = mean((y_{actual} - y_{predicted})^{\wedge 2}) \qquad (4.1)$$

2. IncNodePurity - By averaging the overall decrease in the node impurities from splitting on the variable the node purity is calculated. It is measure by residual sum of squares. Impurity is calculated only at node at which that variable is used for that split.

The importance of the top 10 variables is presented in the Fig. 2 using a variable importance plot. Depending on the variable importance plot, the variables can be selected for other predictive modeling or machine learning technique.

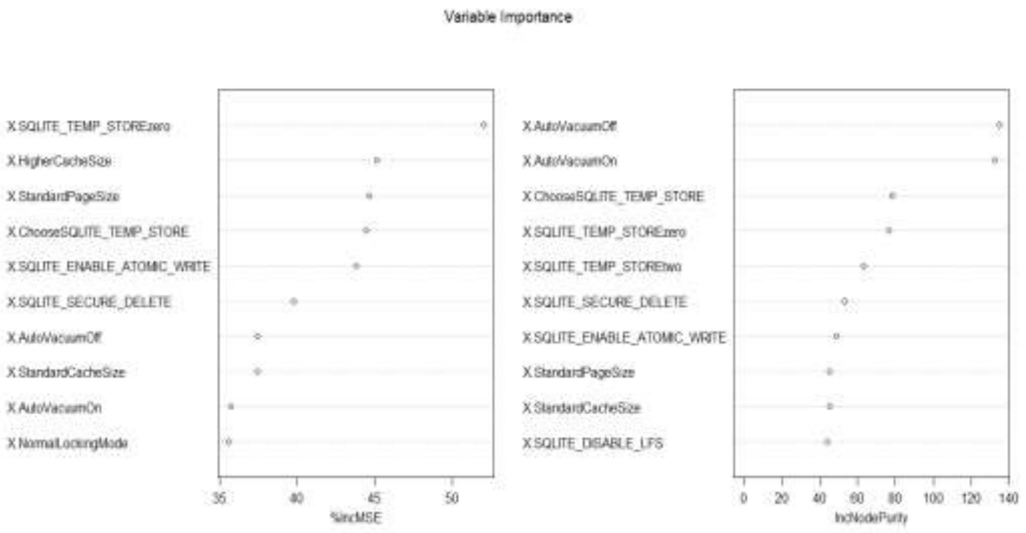

Fig. 2 Variable Importance plot (Top 10 variables)

This will give the user a clear idea about on list of features which are of high importance for the performance of the software.

Then the performance of the constructed prediction model is analyzed using the RMSE and MAE parameters. Root Mean Square Error is the difference between values response predicted by a model or an estimator and the values actually observed. The RMSE represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed out-of-sample. The RMSE serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power.

RMSE is a good measure of accuracy, but only to compare forecasting errors of different models for a particular variable and not between variables, as it is scale-dependent. In an optimal model the RMSE for the training and the test sets should be very similar. If the RMSE for the test set is much higher than that of the training set, it is likely that the training data may be over fit.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|^2} \qquad (4.2)$$

The mean absolute error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes. The mean absolute error is given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \qquad (4.3)$$

where f_i *is the predicted response and* y_i *is the actual response*.The mean absolute error is an average of the absolute errors where is the prediction and the true value.

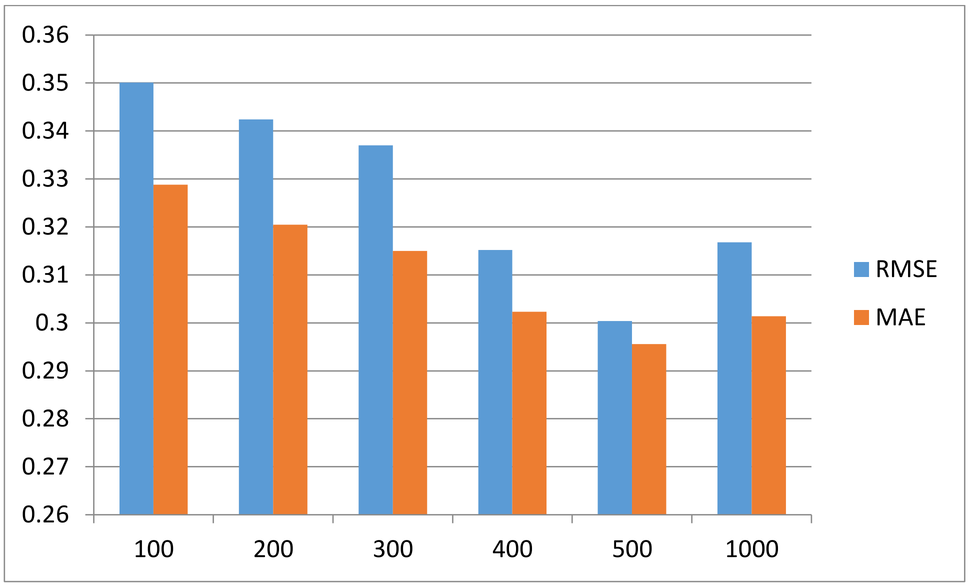

Fig. 3 Performance Analysis of Random Forest

To analyze the performance of the random forest for regression analysis, the training data is given as input to models with different node size and is presented in Fig. 3. Lower values of RMSE in the graph indicate the better fit of the model for the data. The accuracy in predicting the response can be measured with the RMSE value. It is the most important criterion for fit if the main purpose of the model is prediction. In regression analysis problem, the mean squared error is rarely utilized to denote the unbiased estimate of error variance which is equal to the residual sum of squares divided by the number of degrees of freedom.

CONCLUSION

In this paper random forest based regression analysis is preformed to predict the performance of the software system. The dataset, used for training and testing the developed prediction model, is composed of categorical data i.e. they represent the presence of a feature in the software application or not. The random forest based regression analysis is best suited for categorical kind of data. The proposed method for predicting the performance of the configurable software system exhibits high accuracy and low error rate. Also this method is capable of presenting the variable importance as a plot. The error rate during the training phase of the regression model is also very less.

REFERENCES

1. C. Y. Knaus, "Feature - Interaction design for software engineering: Boost into programming future," Interactions, 15(4), 71-74, 2008.
2. Calder, Muffy, et al. "Feature interaction: a critical review and considered forecast." Computer Networks 41.1 (2003): 115-141.
3. Balsamo, Simonetta, et al. "Model-based performance prediction in software development: A survey." IEEE Transactions on Software Engineering 30.5 (2004): 295-310.
4. Guo, Jianmei, et al. "Variability-aware performance prediction: A statistical learning approach." Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on. IEEE, 2013.
5. Criminisi, Antonio, Jamie Shotton, and Ender Konukoglu. "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning." Foundations and Trends® in Computer Graphics and Vision 7.2–3 (2012): 81-227.
6. Boulesteix, Anne-Laure, et al. "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2.6 (2012): 493-507.
7. Dubitzky, Werner; Granzow, Martin; Berrar, Daniel (2007). Fundamentals of data mining in genomics and proteomics. Springer Science & Business Media. p. 178.

8. Kodovsky, Jan, Jessica Fridrich, and Vojtěch Holub. "Ensemble classifiers for steganalysis of digital media." IEEE Transactions on Information Forensics and Security 7.2 (2012): 432-444.