ISSN 2319-2720
Volume 7, No.2, April – June 2018
International Journal of Computing, Communications and Networking
Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn58722018.pdf
https://doi.org/10.30534/ijccn/2018/58722018

Software Predictive Classification Using Relational Association Rules and Naive Bayes Approach

Swathi K[1], Dr. Arun Biradar[2]
[1]Research Scholar-VTU& Asst. Prof. KSIT Bangalore, India, k.swathi980@gmail.com
[2]Prof. & Head Dept. of CSE- EWIT Bangalore, India, hodcsea@gmail.com

ABSTRACT

Software quality is taken into account to be of nice importance within the space of software system engineering and development. So as to extend the potency and also the quality of software system modules, software system defect predictionis employed to spot defect prone modules and this helps in achieving high software system responsibility. Software fault prediction is usually a posh space of analysis and software system practitioners and researchers have applied various ways that to predict wherever the fault is probably going to occur within the software system module and their variable degrees of success. These prediction studies ends up in fault prediction models and it permits software system personnel to target the defect free software system code, thereby leading to software system quality improvement and using the higher utility of the resources. During this Paper style Approach for software system. Defect Prediction is adopted.

Keywords: Fault, Quality, Prediction, Software.

1. INTRODUCTION

Software defect prediction is often a herculean space of analysis and software system practitioners and researchers have allotted various ways that to predict wherever the fault is probably going to occur within the software system module and their varied degrees of success. These prediction studies leads to fault prediction models and it permits software system personnel to target the defect free software system code, thereby leading to software system quality improvement and using the system quality comes into image, then software significant role. Software system is represented. This analysis work primarily concentrates on the ASCII text file of software system systems and not their functions or behavior of the system. The prediction of software system defects at AN early stage can build the corporate professionals to deliver a high quality product to the tip customers, because the value incurred for the event play a significant role.

2. RESEARCH METHODOLOGY

To improve software system quality, it's essential for software system developers to spot defective software system modules at any section of software system Development Life Cycle (SDLC). Several machine learning based mostly classification models were designed and area unit still obtaining improved to unravel the matter of defect prediction. The effectiveness of those models area unit influenced chiefly by 2 key quality information of factors – set of software system metrics won't to build the models and proportion of defect-prone instances within the software system measure data set. During this thesis chapter, a classification model is projected that could be a combination of relative association rules and ancient Naive Thomas Bayes technique. The projected classifier discovers relative association rules on the metrics information supported user outlined confidence & support throughout coaching stage and integrates with ancient Naive Thomas Bayes at testing stage to predict whether or not a software system module is flawed or non-defective.

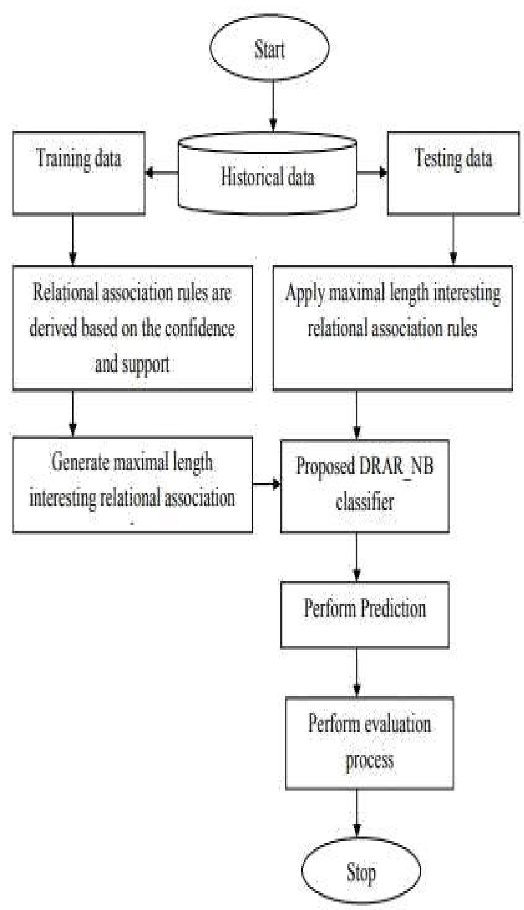

Figure 1 Proposed Design approach for software detection process

4. THE PROPOSED STEPS OF TRAINING PROCESS

Determine the set of attention-grabbing relative association rules having minimum support and confidence from the coaching dataset using DRAR formula. Discover attention-grabbing rules beginning with 2 attributes Perform discovering attention-grabbing rules till total variety of attributes are reached. Establish greatest length attention-grabbing rules in every rule set by scrutiny 1st rule set with second rule set and decide those rule. From 1st rule set that doesn‘t extend in second rule set.

Implementation of Proposed DRAR Algorithm

```
public class Class_ex1 {            public class Class_ex2() {

    public static int attr1;            private static int attr3;
    public static int attr2;            private static int attr4;

    public static void method1()        public static void method4()
    {                                   {
        attr1 = 0;                          Class_ex1.attr1 =0;
    method2();                              Class_ex1.attr2 =0;
                                        Class_ex1.method1();
    }                                   }
    public static void method2()        public static void method5()
    {                                   {
        attr2 = 0;                          attr3 = 0;
        attr1 = 0;                          attr4 = 0;
    }                                   }
    Public static void method3()        public static void method6()
    {                                   {
        attr2 = 0;                          attr3 = 0;
        attr1 = 0;                          method4();
        method1();                          method5();
        method2();
    }                                   }
}                                   }
```

Figure 2 Code example

Maximal length of fascinating relative association rules extends up to a complete range of attributes supported rules satisfying the required minimum confidence and support. Confirm defect chance for every rule by hard what number instances within the coaching dataset that happy the rule against total defective instances. The obtained worth is keep as defect chance for the rule. Confirm non-defect chance for every rule by hard what number instances within the coaching dataset that happy the rule against total non-defective instances.

While end of all rule sets

Do

While end of all rules in a rule set

Do

Step 1: Apply rule to the new instance

If rule is satisfied

Step 2: Multiply defect likelihood of the rule with existing value (initially 1).

Step 3: Multiply non-defect likelihood of the rule with existing value (initially 1).

End

Step 4: Determine prior probability of each class (defective or non-defective) as,

Step 4.1: $\frac{\textit{Count of defective instances}}{\textit{Total number of input testing instances}}$

Step 4.2: $\frac{\textit{Count of non-defective instances}}{\textit{Total number of input testing instances}}$

Step 5: Determine prior probability of predictor (rules) as,

$$\frac{\textit{Number of rules satisfied by the new instance}}{\textit{Total number of rules in a rule set}}$$

Step 6: Determine posterior probability of a defective class as,

$$\frac{\textit{Defective likelihood of the rules (from step 2)} \times \textit{prior probability of defective class (from step 4.1)}}{\textit{Prior probability of rules (from step5)}}$$

Step 7: Determine posterior probability of a non-defective class as,

$$\frac{\textit{Non-defective likelihood of the rules (from step 2)} \times \textit{prior probability of non-defective class (from step 4.2)}}{\textit{Prior probability of rules (from step5)}}$$

Step 8: If (posterior probability of defective class > posterior probability of non-defective class) then Increment score_positive by 1 otherwise Increment score_negative by 1.

End

Step 9: If score_positive > score_negative then declare new instance as defective otherwise declare new instance as non-defective.

5. CONCLUSION

The obtained worth is keep as non-defect chance for the rule develops emotional neural network ELMAN predictor for all the datasets thought-about classifier DRAR_NB is best than existing classifiers that area unit already applied for package defect prediction. The datasets thought-about for experiment area unit Eclipse and PROMISE open supply comes – Lucene, Eclipse_PDE, Eclipse_JDT, CM1, KC1 and PC1 with latest version. The experiments were conducted with and while not method metrics as options to planned classifier so as to work out the impact of method metrics on prediction model. The obtained results shows important improvement of 11

November and twenty fifth in predicting true positive instances on datasets hymenopterous insect and artiodactyls mammal once method metrics area unit combined with product metrics instead of predicting supported product metrics solely. In spite of the planned classifier, leading to higher answer relatively, it'snoted to possess stagnation throughout the prediction method, as a result to beat these occurrences succeeding.

ACKNOWLEDGEMENT

We would like to thank Dr. Arun Biradar Professor and HOD of Computer Science and Engineering, East West Institute of Technology for his constant support and his At last I also thank all faculty members of Department of Computer Science and Engineering, EWIT for their support for me to publish this paper.

REFERENCES

1. S. Finn, E. Mustafaraj, and P. T. Metaxas**The co-retweeted network and its applications for measuring the perceived political polarization**, in *Proc. WEBIST*, 2014.

2. J. Wang, W. X. Zhao, Y. He, and X. Li, **Infer user interests via link structure regularization***ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 2, 2014.

3. F. M. F. Wong, C. W. Tan, S. Sen, and M. Chiang, **Quanti-fying political leaning from tweets and retweets** in *Proc. ICWSM*, 2013.

4. I. Weber, V. R. K. Garimella, and A. Teka, **Political hashtag trends**in *Proc. ECIR*, 2013.

5. I. Weber, V. R. K. Garimella, and E. Borra, **Mining web query logs to analyze political issues** in *Proc. WebSci*, 2012.

6. F. Al Zamal, W. Liu, and D. Ruths, **Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors**, in *Proc. ICWSM*, 2012.

7. M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi, **Measuring user in uence in Twitter: The million follower fallacy** in *Proc. ICWSM*, 2010.

8. D. boyd, S. Golder, and G. Lotan, **Tweet, tweet, retweet: Conversational aspects of retweeting on Twitter** in *Proc. HICSS*, 2010.

9. A. Bacchelli, M. D'Ambros, and M. Lanza. **Are popular classes more defect prone?** In Proceedings of the *13th International Conference on Fundamental Approaches to Software Engineering, FASE'10*, pages 59–73, Berlin, Heidelberg, 2010. Springer-Verlag. https://doi.org/10.1007/978-3-642-12029-9_5

10. C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu. **Fair and Balanced? Bias in Bug- Fix Datasets.** In *ESEC/FSE'09*, 2009. 30

11. E. Arisholm, L. C. Briand, and M. Fuglerud. **Data mining techniques for building fault-proneness models in telecom java software.***In Proceedings of the The 18th IEEE International Symposium on Software Reliability*, ISSRE '07, pages 215–224, Washington, DC, USA, 2007. IEEE Computer Society.

12. V. R. Basili, L. C. Briand, and W. L. Melo. **A validation of object-oriented design metrics as quality indicators**. *IEEE Trans. Softw. Eng.*, 22:751–761, October 1996. https://doi.org/10.1109/32.544352

13. F. Akiyama. **An Example of Software System Debugging**. *In Proceedings of the International Federation of Information Processing Societies Congress*, pages 353– 359, 1971.