Analysis and Prediction of Open Bugs Using Machine Learning Algorithms

Sachin A S, Dr. Rajashree Shettar

Department of Computer Science and Engineering, R V College of Engineering, Mysuru Road, Bengaluru, Karnataka, India.

***Abstract*– There are many fault tracking repositories, some of them are YouTrack, Bugzilla, MantisBT and Atlassian JIRA. Atlassian JIRA repository has been used in this study, as it is extensively accepted by most of the software companies. This repository contains significant information of many projects. Each project has different kinds of issues such as bug(faults) reports, enhancement required to an existing feature, and new feature of the product and task that needs to be done. This paper focuses on analysing the previously raised bug report(history) to understand the correlation and dependability of the attributes like number of bugs created per day, their priority, number of days or hours taken to resolve etc., The data is then processed into a new format which will comply to machine learning algorithms. Different machine learning approaches such as linear regression, support vector machine, K-nearest neighbour, and generalized linear models (Ridge, Lasso and ElasticNet) will be applied on to the data to evaluate and identify the effective and efficient machine learning algorithm for prediction of possible open bugs per day. The best learning algorithm will then be used to predict open bugs per day and make better estimation of time and resource allocation to resolve the issues.**

***Index Terms* –Open bug prediction, JIRA, Machine learning, Project management, Software Quality.**

I. INTRODUCTION

These days, for accelerating software maintenance and advancement most of the companies are using the fault or bug tracking systems to organize and keep track of bugs related to software projects. Such systems are precious for developers, testers and different stakeholders, as they use the system to comfortably report different kind of faults. Faults could be of different categories such as internal tester identified defects/faults, customer identified faults, faults identified by release testing team and so on. A fault is a flaw in the software which shows an unanticipated reaction or behavior of software system [1].

The reported faults are in the form of reports. Document containing some defects or problem occurred during execution of the software is a fault report. Each of the report contains information related to faults such as severity, priority, status, fault key or identifier, created date, updated date, title, version it occurred on, customer name to which it affected, description, comments, attachments which gives more information about the problem, transition, history of activities, closed or resolved date and other fields. It also contains comprehensive elucidation of the problem in common dialect which assists researchers to understand more clearly about the problem[2]. Atlassian JIRA, Bugzilla, Mantis BT, Trac, YouTrack etc., are some of the issue tracking systems which are used in the software industries. But most extensively accepted are JIRA and Bugzilla as they provide many features which are helpful for software development like task tracking, issues, bug, features many plugins to integrate with versioning systems such as Git, mercury etc., and project management.

Consistently both commercial and open source projects experience many changes to represent new client requirements with the consideration of improving existing features, creation of new features or to fix bugs. In this paper, we focus on the analysis over the time for number of bugs open per day. The knowledge of how many bugs open per day will arrive in the next 7 days or 15 days is an extremely useful information to manager sources/staff required and for evolution of projects. Analysis of the bugs raised/created per day, closed bugs per day, may likewise give project management team indication about the quality of the project. Time series study has a two-fold application [3]. One, to examine time series for specific patterns, periodicity and stationarity. Second, predictive models are recognized to forecast and anticipate future values to determine arrival of number of open bugs.

This paper discusses on findings that are procured by analysis data from JIRA repository by querying for specific project bugs over the past five years; data-set is split into two sets for training and validation; time series of relevant attributes (e.g., date, raised bugs, and closed bugs) studied, and predictive models identified.

II. TECHNICAL BACKGROUND

A. Linear Regression (LR)

In the field of statistical learning approaches, to model the relationship between a scalar dependent variable y and one or more independent variables denoted X, a linear regression approach is applied.It is the method of finding the best-fitting straight line through the specified data points. This line is called a regression line [4].The independent variables are mapped to a function f. The assumption made by the linear regression regarding the relationship between the independent variables and the dependent variable is linear. A hypothesis is defined which is used to learn or predict the function f. The hypothesis function for a univariate model can be defined as in equation (1)

$$h(\theta) = \theta_0 + \theta_1 x \quad (1)$$

Where,x is the independent variable andθ_0,θ_1 are the weights of the model.

The equation can be extended to a multivariate model as in equation (2)

$$h(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \quad (2)$$

Where, $x_1, x_2, \ldots, x_n$ are independent variables and $\theta_0, \theta_1, \ldots, \theta_n$ are the weights of the model.

The weights are learnt using methods like least square minimization process. The hypothesis can be modified to different curves to have quadratic, cubic or polynomial linear regression.

B. Support Vector Machine (SVM)

SVM is used for challenges like classification, regression and it is a supervised approach in machine learning. Statistical learning theory provides the base for SVM. If a set of samples where each sample has been labeled as one of the two labels is used as training set, then the SVM training algorithm builds a model which will be capable of classifying any new sample to corresponding label to which it belongs.

The SVM model projects the samples of every label into a vector space. The SVM then tries to separate the projected points such that they have a maximum distance between them. When a new sample is given to the model it is projected into the vector space and the class or label to which it belongs is predicted based upon which side of the line it falls. The equation of the hyper plane is given in equation (3)

$$WX + W_0 = 0 \quad (3)$$

Where,$W_0$is the bias, W is the weight of hyper plane and X is the independent variables vector.

Based on this concept the Support Vector Regressor was developed. It attempts to compute a regression function in a high dimensional feature space when the relationship between input and output is nonlinear. The basic idea is to minimize the error bound instead of minimizing the cost function; it attempts the same by having the constraint as below in equation (4):

$$Y - W.X - W_0 \leq \varepsilon \quad (4)$$

Where, ε gives the error bound.

A SVM can not only perform on linear data but also can efficiently manage nonlinear data using a method known as kernel function which in turn maps the inputs of vector into a high dimensional feature spaces. Linear, polynomial and radial are some of the most used kernel.

C. K - Nearest Neighbors (KNN)

KNN is a simple learning algorithm and one of the simplest algorithms that can be used for both regression and classification problems [5]. A KNN is a memory based learning algorithm as the training samples must be in memory during runtime. The algorithm uses distance to find the neighbors. The number of neighbors considered is based on the hyper-parameter value k. The most used distance formula is Euclidean distance.

When a new data arrives, the target value is calculated based on its neighbor. The target value of the new data can be based on different strategies like, mean, median, maximum or minimum value of the target values of its neighbors.

D. Generalized Linear Models

In machine learning model, avoiding overfitting is one of the major aspectsof training your model. To address the issue of overfitting, there are two choices. First is reduce the number of features and second one is regularization. Regularization keeps all the features, and it will reduce the magnitude of parameters θ. When plenty of moderately useful features are available, regularization works effectively. The following sub-sections from (a) to (c) discusses the various regularization models

(a) ***Ridge Model:*** When data suffers from multicollinearity (attributes will be correlated to greater extent), this model is used. It minimizes the standard error by adding degree of bias to the regression estimates. It resolves the multicollinearity problem by using shrinkage parameter (λ), which can be computed as

$$\sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \lambda \sum_{j}^{m} \beta_j^2 \quad (5)$$

Where y_i actual value, $\hat{y_i}$ is predicted value and β gives the coefficient. It uses $L2$ regularization technique (i.e., it adds sum of squares of coefficients as factor of optimization to linear regression).

(b) ***LASSO Model:*** Least Absolute Shrinkage and Selection Operator (LASSO) is a type of linear regression which penalizes the regression coefficient. It is suitable for models which show high level of multicollinearity. This can be computed as in equation (6)

$$\sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \lambda \sum_{j}^{m} |\beta_j| \quad (6)$$

Where y_i actual value, $\hat{y_i}$ is predicted value and β gives the coefficient. It uses $L1$ regularization technique (i.e., it adds sum of absolute value of coefficients as factor of optimization to linear regression). This model is usually used when we have more number of features since it does the feature selection automatically.

(c) ***ElasticNet Model:*** It is hybrid combination of two types of regression models like the Lasso and the Ridge Regression. It uses both $L1$ and $L2$

regularization technique for training the model [6]. This can be computed as in equation (7)

$$\sum_{i=1}^{n}(y_i - \hat{y_i})^2 + \lambda_1 \sum_{j}^{m} \beta_j^2 + \lambda_2 \sum_{j}^{m} |\beta_j| \quad (7)$$

Where y_i actual value, $\hat{y_i}$ is predicted value and β gives the coefficient.

III. Empirical Study

The complete flow of this work is explained in Fig. 1.In this work raw data is extracted from the Atlassian public JIRA[1] repository. To interact with Jira remotely, Jira[2] python package is used which internally uses Jira REST API's. The Raw dataset consists of created date, priority, closed date and status for each fault available in the project. This raw dataset is then processed into a new format which will comply with machine learning algorithms. Number of open bugs per day is depends on the new number of bugs raised and number bugs that will be closed on that day and total number of bugs open on the previous day. In preprocessing for each date, the total bugs raised, closed and open till last day is calculated. Then this data is fed to different machine learning model which we discussed in section II. Then the results are compared and best model is selected for forecasting the number of open bugs per day for next 7 days or 15 days.

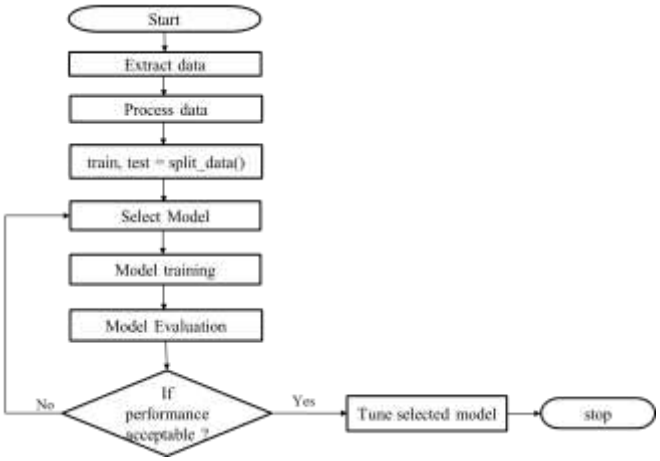

Fig. 1:Flowchart for Prediction of number of open bugs per day

A Dataset

Extracted dataset (past five-year data) consists of five attributes including the target and 1306 instances. The format of the dataset is as described in the

TABLE I given below.

TABLE I
ATTRIBUTE NAMES AND ITS TYPE USED IN DATASET

Attribute Name	Attribute Type
Date	Datetime
Raised faults	Numeric
Closed faults	Numeric
Previous open faults	Numeric
Total open faults	Numeric

B. Attribute Selection

For each machine learning model, dataset is split into two sets: training set which is 80% of the dataset and testing or validation set which is 20% of the dataset. This is done using the train_test_split [3] method from the scikit-learn [4] python library.

The models are trained by using thetraining set for predicting number of bugs raised per day and for predicting closed bugs per day. For predicting raised bugs,input variable consists of previous five days of raised bugs count and for predicting closed bugs count, input variables are raised and previous open bugs count of previous five days data. After completion of training process, these trained models are examined against the testing set and Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated. This procedure of data splitting, training and validating is iterated for 20 times. The metrics likeRMSE and MAE is averaged to evade any bias that might result from unfavorable data sampling.

C Analysis and Results

In this study, experiments are carried out for 4 different cases and each case is for various number of days to forecast. Case 1 for 1 day, case 2 for 7 days, case 3 for 15 days and case 4 for 30 days. TABLE II and TABLE III demonstrate the MAEs and RMSEs of six models. By looking at the both metrics for all the models as depicted in TABLE II and TABLE III, Lasso regression model perform best for all the 4 cases.

TABLE II
MAE VALUES OF SIX MODELS FOR FOUR CASES

Cases	LR	SVM	KNN(n=10)	Ridge	Lasso	ElasticNet
Case 1	0.0	0.01	0.1	1.279	0.054	1.035
Case 2	1.857	2.05	1.542	3.394	1.294	3.348
Case 3	2.467	3.23	1.78	6.544	1.080	3.399
Case 4	2.901	4.56	4.63	12.07	2.900	3.990

TABLE III
RMSE VALUES OF SIX MODELS FOR FOUR CASES

Cases	LR	SVM	KNN(n=10)	Ridge	Lasso	ElasticNet
Case 1	0.0	0.01	0.1	1.279	0.05	1.035
Case 2	2.171	2.42	1.542	3.394	1.46	3.655
Case 3	2.903	3.82	1.78	6.544	1.32	3.932
Case 4	3.301	5.18	4.63	12.07	3.30	6.281

For first three cases linear regression performed better, as compared to the other models except for the Lasso model. As discussed in the Technical Background section, regularization models are tuned to optimize base regression model. Test were conducted on all three-regularization model and from the

[1] https://jira.atlassian.com/issues/?jql=ORDER%20BY%20created%20ASC
[2] https://pypi.org/project/jira/
[3] http://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[4] https://pypi.org/project/scikit-learn/

results it was found that the LASSO model is better compared to other models.

D. Threats to Validity

As there will be continuous inflow of data into JIRA, the count of open bugs per day may change and pattern fed to training model may also vary. So, this may affect the target variable.

IV. Conclusion and Future Work

An automated process for extracting data from Jira database, preprocessing of raw data and machine learning model for predicting number of open bugs per day is developed and computed MAE and RMSE for each model.

In this paper, an empirical evidence on predicting number of open bugs per day by applying machine learning models such as linear regression, support vector machine, KNN, Ridge model, Lasso model and ElasticNet model is provided. According to quantitative assessment Lasso model performs better than the other models in all the cases for predicting the number of open bugs per day. In the future, Poisson regression model could be used. Neural network model to predict the open bugs count and its performance with other existing models could be used.

Acknowledgment

This work is supported by Nokia Solutions and Networks Private Limited, Bengaluru. The authors also delight edlyendorse the productive discussions and remarks of Mr. Rajesh K.

References

[1] K. Arvinder and G. J. Shubhra, "Bug report collection system (BRCS)," in *Cloud Computing, Data Science & Engineering-Confluence*, 2017.

[2] X. Gou, Xin Zhou, JiaWen Pang and ShunKun Yang, "Medical software bug prediction based on static analysis.," in *In Industrial Electronics Society*, 2017.

[3] Kenmei, Benedicte, Giuliano Antoniol and P. D. Massimiliano, "Trend analysis and issue prediction in large-scale open source systems," in *Software Maintenance and Reengineering*, 2008.

[4] D. C. Montgomery, A. P. Elizabeth and G. V. Geoffrey , Introduction to linear regression analysis, vol. 821, John Wiley & Sons, 2012.

[5] G. Rinkaj, C. Pravin and S. Yogesh, "Suitability of KNN regression in the development of interaction based software fault prediction models.," *Ieri Procedia,* vol. 6, pp. 15-21, 2014.

[6] H. Osman, M. Ghafari and O. Nierstrasz, "Automatic feature selection by regularization to improve bug prediction accuracy," in *Machine Learning Techniques for Software Quality Evaluation*, 2017.