Bayesian Logistic Regression for software defect prediction

Jinu M Sunil
BITS Pilani Hyderabad Campus
Hyderabad, India
f20130423@hyderabad.bits-pilani.ac.in

Lov Kumar
BITS Pilani Hyderabad Campus
Hyderabad, India
lovkumar@hyderabad.bits-pilani.ac.in

N L Bhanu Murthy
BITS Pilani Hyderabad Campus
Hyderabad, India
bhanu@hyderabad.bits-pilani.ac.in

***Abstract*—Timely identification of bugs plays an important role in delivering quality software. Defect prediction models help to detect or rank the defect prone files so that the project management team can allocate resources diligently or may seek help from external sources to enable rigorous quality assurance activities on defect prone files. Though defect prediction models have been built using several machine learning algorithms, Bayesian approach of these models is not explored. We propose Bayesian logistic regression with non-informative and informative priors to build defect prediction models. We seek to study if there are any advantages of using Bayesian logistic regression over logistic regression and the role of priors in the performance of Bayesian logistic regression. A comparative study of the performance of Bayesian logistic regression with other widely known classifies is also presented.**

***Index Terms*—Bayesian regression, Informative priors**

I. Introduction

Delivering quality software is one of the most important goals of any IT vendor to survive in the highly competitive market. Bugs[1] or defects surfaced during any time in the evolution of a software, especially in the post-production phase, will be detrimental to the software quality. In practice, defects get uncovered during post-production phase in spite of expending good number of man hours for quality assurance activities like different kinds of testing and code reviews. Researchers and practitioners proposed the methodologies to curtail these defects. Software Defect Prediction is one such technique to predict defects, preferably before rolling to production environment, where in the deeper assurance activities like code reviews and testing by Subject Matter Experts (SMEs) can be performed on these defect prone files. The defect prediction models have been developed using machine learning algorithms like logistic regression, Naive Bayes classifier and Random forest etc.

Logistic Regression is a probabilistic discriminative model where in probabilities of positive and negative class are modeled by sigmoid function.

$$P(Y = 1|w, x) = 1/(1 + e^{-w^T x}) \tag{1}$$

It gives the probability that a class/file is defective, given the feature vector of a file x and parameter vector w. Suppose there are n observations where the i^{th} observation is a tuple (x_i, y_i) , x_i is a m dimensional feature vector and y_i is boolean valued representing wether the file is defective (1) or not (0). $D = \{(x_1, y_1), (x_2, y_2)..(x_n, y_n)\}$ is the set of all training instances.

The following equality is obtained from bayes theorem.

$$P(w|D) = P(D|w)P(w)/P(D) \tag{2}$$

where $P(w|D)$ is posterior distribution, $P(D|w)$ is likelihood and $P(w)$ is the prior distribution.

$P(D)$ is a constant for a given D and hence

$$P(w|D) \propto P(D|w)P(w) \tag{3}$$

[1]bug and defect are used synonymously throughout this paper

In logistic regression, it is assumed that w follows uniform distribution and hence any tuple in R^{m+1} (m is the size of input vector) is a candidate solution and hence

$$P(w|D) \propto P(D|w) \tag{4}$$

$y_i's$ are conditonally independent hence we can write equation 4 as:

$$P(w|D) \propto \prod_1^n P(y_i|x_i, w) \tag{5}$$

Let $p_i = P(y_i = 1|x_i, w) = \frac{1}{1+e^{-w^T x_i}}$

$$\therefore P(w|D) \propto \prod_1^n P(y_i|x_i, w) = \prod_1^n p_i^{y_i} * (1 - p_i)^{1-y_i} \tag{6}$$

The optimal w for logistic regression is determined as the maximum likelihood estimator (MLE) of the given training data.

$$w_{MLE} = argmax_w(\prod_1^n p_i^{y_i} * (1 - p_i)^{1-y_i}) \tag{7}$$

There are some drawbacks of logistic regression when applied to defect prediction problem and they are listed below.

1) MLE's are asymptotically unbiased. i.e $E(w_{mle}) \approx w_{true}$ as n (number of instances/samples) becomes very large. It has been shown that regression coefficients are biased for small and moderate sample sizes [1]. Long et al. [2] offers a rough heuristic about appropriate sample sizes: it is risky to use MLE with samples smaller than 100, while samples larger than 600 seem adequate. For 44 datasets that are used in our study, the average number of files in a project are 400.
2) Logistic regression works well when both non-event (no bug) and event (bug) occur in the same ratio and the bias is substantial for small and medium samples with skewed ratio [3]. For defect prediction problem, generally the files are skewed towards non-event (non-defective) and the number of files in each of the project considered in our study is not high.

Logistic regression as stated above assumes uniform priors, but in practice priors can follow any distribution. In Bayesian Logistic regression a prior for w is ascertained from previous data or expert opnion. Consider the case where the prior is a normal distribution $w \approx N(\mu, C)$ with mean μ and covariance C. The posterior distribution is.

$$P(w|D) \propto P(D|w) * P(w) \tag{8}$$

$$P(w|D) \propto (\prod_1^n p_i^{y_i} * (1-p_i)^{1-y_i}) * \frac{exp(\frac{-(w-u)^T C^{-1}(w-u)}{2})}{\sqrt{\det(2\pi C)}} \tag{9}$$

It has been shown that Bayesian Logistic Regression improvises prediction accuracy of learning models [4], [5]. Though there are

DOI reference number: 10.18293/SEKE2018-181

several learning models developed to predict defects, the Bayesian perspective of the models have not been studied so far. We study Bayesian Logistic Regression and attempt to answer the following research questions:

- **RQ1**: Is there any significant difference between Logistic Regression and Bayesian Logistic Regression?
- **RQ2**: Study informative and non-informative priors of w and test whether there is any significant difference between the posterior probabilities of the classifiers developed using these different priors?
- **RQ3**: Is there significant difference between the Bayesian Logistic Regression and other widely used classifiers for defect prediction problem?

The rest of the paper is organized as follows. In section II related work is discussed. The metrics and datasets used in this study are discussed in section III. A description of the bayesian logistic regression model is given in section IV. The experimental setup is described in section V. The results are discussed in section VI. Threats to validity of the experiments are discussed in section VII. Conclusions are presented at the end.

II. Related Work

Bug Prediction models have been built using various metrics of a software. Chidamber and Kermer (CK), Halstead metrics are well known metrics used in defect prediction. However it has been shown that lines of code (LOC) have strong correlation with proneness to defect. Zhang et al. [6] concluded that LOC increases bug proneness. CK metrics have been useful in detecting bugs ([7], [8]). Thomas et al. [9] explored the relationship between code complexity and defect proneness and arrived at the conclusion that complexity metrics are very useful in defect prediction and also code bases that undergo a lot of evolution are prone to defects. These metrics are used in classification algorithms.

Bug detection algorithms are well studied. L. Guo et al. [10] applied random forests to predict fault prone modules and found that random forests were better when applied to large datasets. Neural networks for bug prediction was explored by Rajni Jindal et al. [11] they used ROC to interpret the results obtained. Czibula et al. [12] used unsupervised algorithms for bug prediction and their rule mining based approach resulted in as good results as other algorithms. Relevant to this study Rakesh et al. [13] experimented with a bayesian approach to predict number of bugs in a software and showed that incorporating prior information gave beter results.

A study of benchmarking various algorithms was done by Stefan et al. [14]. They conducted a freidman neymni with AUC-ROC as their measure and showed that random forest outperformed all other models but not significantly. Also around 17 models including random forest, logistic regression, naive bayes and multi layer perceptron were shown to have no significant difference.

III. Metrics and Dataset

TABLE II gives a description of software projects considered. Only software projects with three or more versions are considered. The data of these 12 projects is extracted from the Promise repository [15]. The average number of classes over all projects and versions is 400. The metrics considered for this study are given in TABLE I. Jurecko and Madeyski [21] give a detailed description of all the metrics.

TABLE I
METRICS

Author	Metric
Chidamber and Kemerer [16]	Weighted methods per class (WMC)
	Depth of Inheritance Tree (DIT)
	Depth of Inheritance Tree (DIT)
	Coupling between object classes (CBO)
	Response for a Class (RFC)
	Lack of cohesion in methods (LCOM)
Bansiy and Davis [17]	Number of Public Methods (NPM)
	Data Access Metric (DAM)
	Measure of Aggregation (MOA)
	Measure of Functional Abstraction (MFA)
	Cohesion Among Methods of Class (CAM)
Tang et al. [18]	Inheritance Coupling (IC)
	Coupling Between Methods (CBM)
	Average Method Complexity (AMC)
Martin [19]	Afferent couplings (Ca)
	Efferent couplings (Ce)
McCabe [20]	cyclomatic complexity (CC)

IV. Methodology

As discussed in the first section the prior $p(w)$ need not necessarily be a unifrom distribution but can be any distribution. There are two kinds of priors informative and non-informative priors. Informative priors can be derived from historical data or domain expertise whereas Non-informative priors do not rely on previous data but can be approximated by making use of training data.

In this study we make use of a Non-informative prior−Jeffery prior and also propose couple of informative priors that follow a multivariate normal distribution.

A. *Choice of Priors*

- **Bayes-1 prior** - This is an **informative** prior. The prior of the current version is the posterior of the previous version of the same software. For example consider the project ivy as described in table II. Ivy has 3 versions, initially assume an isotropic normal distribution $w \approx N(0, \alpha^{-1}I)$ as prior, and obtain posterior with ivy 1.1 as the data

$$p(w|D_{ivy1.1}) = p(D_{ivy1.1}|w) * N(0, \alpha^{-1}I)$$

. A normal approximation of the above posterior distribution will be used as prior for ivy 1.4

$$p(w|D_{ivy1.4}) = p(D_{ivy1.4}|w) * p(w|D_{ivy1.1})$$

.

- **Bayes-2 prior** - This is also an **informative** prior and is best explained with an example. Consider again ivy described in TABLE II,

1) Choose 66% instances of ivy 1.1 randomly let it be $D_{ivy1.1} = 66\%\ of\ ivy1.1$. Train a logistic regression with this data, the resulting parameters are $w_1 = argmax_w(P(D_{ivy1.1}|w))$
2) Choose 66% instances of ivy 1.4 randomly let it be $D_{ivy1.4} = 66\%\ of\ ivy1.4$. Train a logistic regression with this data, the resulting parameters are $w_2 = argmax_w(P(D_{ivy1.4}|w))$

TABLE II
12 PROMISE PROJECTS

Project	Version	No. of clases	% of Bugs
Ant	1.3	125	16
	1.4	178	22.4
	1.5	293	10.9
	1.6	351	26.2
	1.7	745	22.2
Camel	1	339	3.8
	1.2	608	35.5
	1.4	872	16.6
	1.6	965	19.48
Ivy	1.1	111	56.7
	1.4	241	6.6
	2	352	11.3
Jedit	3.2	272	33.09
	4	306	24.53
	4.1	312	25.31
	4.2	367	13.08
	4.3	492	2.23
Log-4j	1	135	25.18
	1.1	109	33.94
	1.2	205	92.19
Lucene	2	195	46.67
	2.2	247	58.3
	2.4	340	59.7
Poi	1.5	237	59.49
	2	314	11.78
	2.5	385	64.41
	3	442	63.57
Synapse	1	157	10.19
	1.1	222	27.03
	1.2	256	33.59
Velocity	1.4	196	77
	1.5	214	66.35
	1.6	229	34.06
Xalan	2.4	723	15.21
	2.5	803	48.19
	2.6	885	46.44
	2.7	909	98.79
Xerces	1.2	440	16.14
	1.3	453	15.23
	1.4	588	74.32
PC	1	735	8.3
	2	1493	1.2
	3	1099	12.5
	4	1379	12.9

3) Repeat the above 2 steps around 30 times, giving 60 parameter vectors−30 from ivy 1.1 and 30 from ivy 1.4.
4) Find the best fit multivariate normal for these 60 points
5) Use this multivariate normal as prior for ivy 2.0

- **Jefferey Prior**-The jefferey prior is a **'non-informative'** prior (uniform priors are also non-informative). $p(w) \propto \sqrt{det(I(w))}$, where $I(w)$ is the fisher information matrix. The fisher information matrix in the case of logistic regression is $I_{i,j}(w) = -E[\frac{\delta^2 ln(Likelihood)}{\delta w_i \delta w_j}]$ (w_i and w_j are the i^{th}, j^{th} component of vector w. $E[]$ is expected value). The posterior distribution becomes.

$$P(w|D) \propto P(D|w)\sqrt{det(I(w))} \tag{10}$$

It was shown by Firth [22] that this choice of prior reduces first order bias in the case of logistic regression.

B. Sampling from the Posterior

In logistic regression we get a point estimate w_{MLE} and in the bayesian approach a set of samples are drawn from the posterior $P(w|D)$. For any testing example, average of all probabilities would be taken into consideration to classify the testing instance. Suppose k samples are drawn $w_1, w_2...w_k$ from $P(w|D)$ then the probability that a file with attribute x is defective would be.

$$P(y = 1|x, D) = \frac{1}{k}\sum_{1}^{k} 1/(1 + e^{-w_i^T x}) \tag{11}$$

We discuss below the two ways of drawing samples from posterior $p(w|D)$.

Laplace Approximation: The posterior is approximated to a multivariate normal distribution with mean w_{max} and covariance matrix K where $w_{max} = argmax_w(p(w|D))$ and $K = H^{-1}$, H is the hessian of $p(w|D)$ computed at w_{max}. Samples are drawn from this approximation. Laplace approximation works well when the pdf is unimodal otherwise it might be a bad approximation [23] and hence we have not used it in this study.

MCMC: The other way of drawing samples from the posterior is the famous Markov Chain Monte Carlo (*MCMC*) which involves constructing a markov chain with desired distribution as its equilibrium distribution. In this study we use gibbs sampling which is a MCMC method, to draw samples from the posterior.

Among these prior choices using jeffrey priors did not improve performance the other two priors improved performance over simple logistic regression significantly.

V. EXPERIMENTS

A. Cost Model

Normalized Expected Misclassification Cost (NEMC) is used to compare performance of the models. Type 1 error is predicting a non-defective file as defective and Type 2 error is predicting a defective file as non-defective. For the defect prediction problem it is evident that type 2 errors are costlier than type 1 errors. The cost ratio β is defined as.

$$\beta = \frac{cost\ of\ Type\ 2\ error}{cost\ of\ Type\ 1\ error} \tag{12}$$

False Positive Rate and False Negative Rate are defined as.

$$E_1 = \frac{FP}{TN + FP} \tag{13}$$

$$E_2 = \frac{FN}{TP + FN} \tag{14}$$

where FP-false positive, TN-true negative, FN-false negative and TP-true positive

$$NEMC = \beta * (E_2 * P_{df}) + (E_1 * P_{ndf}) \tag{15}$$

Where P_{df} and P_{ndf} are prior probabilities of defective and non-defective files in the dataset respectively. We refer the reader to Khoshgoftaar et al. [24] for a detailed derivation of NEMC.

B. Implementaion settings

To answer RQ1 and RQ2, we implement logistic regression, Bayesian logistic regression with Bayes-1 Prior, Bayes-2 Prior and Jeffery Prior. We also implement Random Forest, Nave Bayes and SVM for comparative study of these classifiers with Bayesian logistic regression to answer RQ3. All models are implemented in R-programming [25] and details are discussed below:

- Bayes 1 and Bayes 2: Logit function of the Bayes Logit package [26] is used to implement both the bayesian logistic regression methods. The function requires Data, prior mean and

prior variance as input. Logit function returns samples from the posterior − 1000 points were sampled after a burn-in of 1000.
- Jeffrey Prior (JP): logistf [27] package is used to implement logistic regression with jeffrey prior. The package computes the fisher matrix and also retuns a set of samples from the posterior.
- Random forest (RF): randomforest [28] function is used to implement random forest, the ntree variable was set to 100 after experimentation.
- Naive bayes (NB): naiveBayes [28] function is used to implement naive bayes algorithm, this function requires data to be un-normalized.
- Logistic regression (LR): glmnet [29] is used to implement logistic regression algorithm, the regularization parameter was computed using cross-validation.
- svm, l-svm: svm [28] function is used to implement both the svm's. l-svm kernel is linear whereas the kernel for svm is radial.

C. Training and Testing

Each model is trained on 66% of the last version of each of the software projects mentioned in TABLE II and tested on the remaining 34% of the same project. For the bayesian methods the priors come from the previous version of a project. Consider the ivy project for elaboration of the training-testing process.

- Train RF, LR, NB, svm and l-svm on 66% of ivy 2.0 (last version of ivy)
- Train JP, Bayes 1 and Bayes 2 on 66% of ivy 2.0 and the priors are approximated from ivy 1.1 and ivy 1.4.
- Test all the models on the remaining 34% of ivy 2.0, Compute NEMC on the testing data of ivy 2.0 for all models.

NEMC is used to compare performance of classifiers as discussed in the previous section. NEMC for each model is averaged over several runs (R). For example consider random forest (RF) and ivy 2.0

for $i\ in\ 1 \rightarrow R$ **do**
  $D_{train} = 66\%\ of\ ivy_{2.0}$
  $D_{test} = ivy_{2.0} - D_{train}$
  train RF on D_{train}
  compute NEMC for RF over D_{test}, let it be np_i
end for
$avg\ NEMC\ for\ RF = np_{RF} = \frac{\sum_1^R np_i}{R}$

The above algorithm is used to compute average NEMC for all 8 models.

D. Statistical Tests for comparing classifiers

Classifiers are compared based on average NEMC and statistical tests used in our study are discussed below:

- **Two sample test − Wilcoxon signed rank test and rank test**
  In both cases the null hypothesis is

  $$H_0 : The\ two\ models\ have\ the\ same\ performance$$

  and the alternative

  $$H_a : The\ two\ models\ have\ different\ performance$$

  H_0 is rejected if $p_{value} < 0.05$. For example, consider TABLE III to compare Bayes 2 and Logistic Regression (LR) at $\beta = 5$. Here two samples are NEMC values of Bayes-2 (column 2) and LR (column 3)

- **K sample test − Friedman test**
  Here the null hypothesis is

  $$H_0 :\ all\ classifiers\ perform\ alike$$

  and the alternative

  $$H_1 :\ atleast\ 2\ classifiers\ differ\ in\ performance$$

  Friedman test assigns a mean rank to each classifier. And if the null hypothesis is rejected a **Nemenyi** test is used to compare all classifiers. The mean rank of two classifiers have to differ by a critical difference (CD) for them to be considered significantly different.

  $$CD = q_{a;\infty;L}\sqrt{\frac{L(L+1)}{12K}} \tag{16}$$

  where L is the number of classifiers and K is the number of projects.

VI. Results and Discussion

Type 2 error is much costlier than Type 1 Error for defect prediction problem and the ratio of two costs (β) is an important parameter in the performance measure NEMC. The cost ratio (β) varies with the type of project and organization. We consider the cost ratio to be 5, 10, 20, 40 and conduct experiments to answer RQ1 through RQ3.
For each of the project and cost ratio, learning models are developed using logistic regression and Bayesian logistic regression with Bayes-1 Prior, Bayes-2 Prior and Jeffery prior. The average NEMC over 1000 runs for each of the project are tabulated in Table III, IV, V and VI respectively. Wilcoxon signed rank test and rank test is performed to check whether there is any significant difference between logistic regression and Bayesian logistic regression (RQ1).
For Bayesian logistic regression with Bayes-1 prior and Bayes-2 prior, the null hypothesis for Wilcoxon signed rank test (WSR) and rank test (RT) is rejected as p_{values} are considerably less than 0.05. Hence the Bayesian logistic regression with informative priors is significantly better than logistic regression. The p_{values} for different values of β are tabulated in Table VIII. The average number of data points, 400 for our study, is low and could lead to biased estimates and this might be one reason for lower performance of logistic regression. Also, the imbalance of defective files and non-defective files could be another reason.
However it is found out that there is no significant difference between the logistic regression and Bayesian logistic regression with Jeffery prior. Jeffery prior is non-informative prior like uniform prior and it is not generated from historical data and this could be one of the reasons for downplay of Bayesian logistic regression with Jeffery prior. Supporting the argument, it is also observed that there is a significant difference between the performances of Bayesian logistic regression with informative priors and that of non-informative priors (RQ2).
After observing significantly better performance of Bayesian Logistic regression (with informative priors) as compared to logistic regression, the obvious next step is to rank its performance with other widely used classifiers like Random Forest, Nave Bayes, SVM etc (RQ3). We have not considered Bayesian logistic regression with Jeffery prior in this comparative study as it is not significantly different from Logistic Regression (LR). We have conducted Freedman test to check whether there is any significant difference between these classifiers - Bayes 1, Bayes 2, RF, LR, NB, L-SVM, SVM. It is observed that there is significant difference between the performances of these classifiers. Neymni test has been conducted to rank these classifiers and the results are reported in Table VII. Random Forest

is ranked higher than all classifiers and this result is in tune with other comparative studies [14], [30], [31]. The random forest (RF) is an ensemble classifier and for reasons mentioned in [14], it ranks higher than other classifiers. Bayesian logistic regression with Bayes-2 prior and Bayes-1 prior rank 2^{nd} and 3^{rd} respectively but with no significant difference from Random Forest (RF).

TABLE III
NORMALIZED PENALTY AT $\beta = 5$

$\beta = 5$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	0.6505	0.7140	0.8952	0.7919	1.9743	**0.597**	0.6954
ANT	0.7197	0.6946	0.8072	0.7737	**0.609**	0.6569	0.8243
CAMEL	0.9143	0.8947	0.9551	0.9962	**0.8192**	0.8540	0.9916
XALAN	0.0268	0.0360	0.0560	0.0114	0.9149	**0.0102**	0.0110
JEDIT	0.1403	0.1423	0.1525	0.1308	0.2900	0.1301	**0.1276**
PC	0.0050	**0.0047**	0.0055	0.0057	0.0056	0.0058	0.0070
IVY	0.7132	0.7169	0.7107	0.7310	**0.6004**	0.7160	0.8256
LOG4J	0.2809	0.4614	0.5059	0.1459	2.3786	0.0971	**0.0732**
LUCENE	0.9301	**0.7724**	1.1920	1.0087	1.8144	0.7732	0.9008
SYANAPSE	0.9720	0.9738	1.1104	1.0479	0.9182	**0.812**	1.0251
VELOCITY	0.9461	0.9618	1.1267	1.0334	1.2760	**0.8941**	1.2530
XERCES	0.3294	0.3415	0.5492	0.3436	1.5594	**0.1671**	0.4893

TABLE IV
NORMALIZED PENALTY AT $\beta = 10$

$\beta = 10$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	1.1383	1.3174	1.6982	1.4682	3.9051	**1.0939**	1.2691
ANT	1.3936	1.3215	1.5855	1.5138	**1.12**	1.2534	1.6114
CAMEL	1.8034	1.7376	1.8962	1.9867	**1.5753**	1.6650	1.9760
XALAN	0.0451	0.0654	0.1027	0.0118	1.8270	0.0151	**0.011**
JEDIT	0.2590	0.2625	0.2732	0.2553	0.3729	**0.2532**	0.2551
PC	0.0099	**0.0093**	0.0110	0.0114	0.0111	0.0116	0.0140
IVY	1.3894	1.3707	1.3932	1.4399	**1.1048**	1.4087	1.6490
LOG4J	0.4954	0.8631	0.9555	0.2199	4.7415	0.1307	**0.0732**
LUCENE	1.6974	**1.3707**	2.2785	1.8765	3.5797	1.3730	1.6320
SYANAPSE	1.8489	1.8511	2.1542	2.0187	1.7318	**1.5253**	1.9739
VELOCITY	1.7829	1.8181	2.1713	1.9712	2.4785	**1.6755**	2.4385
XERCES	0.6008	0.6251	1.0521	0.6393	3.0898	**0.2866**	0.9008

TABLE V
NORMALIZED PENALTY AT $\beta = 20$

$\beta = 20$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	2.1137	2.5242	3.3042	2.8209	7.7667	**2.0876**	2.4164
ANT	2.7414	2.5753	3.1420	2.9941	**2.1419**	2.4464	3.1856
CAMEL	3.5816	3.4233	3.7783	3.9676	**3.0873**	3.2871	3.9448
XALAN	0.0819	0.1243	0.1963	0.0127	3.6511	0.0250	**0.011**
JEDIT	**0.4964**	0.5028	0.5146	0.5043	0.5388	0.4994	0.5102
PC	0.0197	**0.0186**	0.0219	0.0229	0.0220	0.0231	0.0281
IVY	2.7418	2.6782	2.7581	2.8575	**2.1136**	2.7941	3.2958
LOG4J	0.9246	1.6665	1.8548	0.3678	9.4673	0.1981	**0.0732**
LUCENE	3.2318	**2.5673**	4.4516	3.6120	7.1101	2.5724	3.0943
SYANAPSE	3.6026	3.6058	4.2419	3.9603	3.3592	**2.9517**	3.8717
VELOCITY	3.4565	3.5309	4.2605	3.8468	4.8836	**3.2383**	4.8097
XERCES	1.1436	1.1924	2.0578	1.2307	6.1507	**0.5256**	1.7239

VII. THREATS TO VALIDITY

The various threats to validity that may impact the analysis of the proposed approach, the experimental study and conclusions are presented here.

A. Internal Validity

NEMC as performance measure has been adopted by previous studies. Performance measures of similar kind which give more weightage to one kind of error is common in defect prediction [30],

TABLE VI
NORMALIZED PENALTY AT $\beta = 40$

$\beta = 40$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	**4.0646**	4.9378	6.5161	5.5262	15.4899	4.0750	4.7109
ANT	5.4371	5.0829	6.2550	5.9547	**4.1858**	4.8324	6.3339
CAMEL	7.1380	6.7948	7.5424	7.9295	**6.1114**	6.5313	7.8823
XALAN	0.1553	0.2419	0.3834	0.0143	7.2993	0.0448	**0.011**
JEDIT	0.9713	0.9835	0.9974	1.0021	**0.8706**	0.9917	1.0204
PC	0.0394	**0.0371**	0.0437	0.0457	0.0439	0.0462	0.0561
IVY	5.4465	5.2932	5.4879	5.6929	**4.1313**	5.5648	6.5895
LOG4J	1.7829	3.2732	3.6534	0.6637	18.9188	0.3327	**0.0732**
LUCENE	6.3008	**4.9606**	8.7976	7.0829	14.1711	4.9713	6.0189
SYANAPSE	7.1101	7.1153	8.4172	7.8437	6.6138	**5.8047**	7.6672
VELOCITY	6.8036	6.9563	8.4389	7.5979	9.6938	**6.3639**	9.5520
XERCES	2.2292	2.3269	4.0693	2.4136	12.2726	**1.0037**	3.3701

TABLE VII
FRIEDMAN NEMENYI **CD=2.62**

$\beta = 5$	Mean rank
RF	2.083
Bayes2	3.167
Bayes1	3.500
SVM	4.417
L-SVM	4.667
NB	4.833
LR	5.333
$\beta = 10$	Mean rank
RF	2.250
Bayes2	3.167
Bayes1	3.250
SVM	4.417
L-SVM	4.583
NB	4.833
LR	5.500
$\beta = 20$	Mean rank
RF	2.333
Bayes2	2.917
Bayes1	3.083
L-SVM	4.667
SVM	4.667
NB	4.833
LR	5.500
$\beta = 40$	Mean rank
RF	2.583
Bayes2	2.917
Bayes1	3.083
NB	4.333
L-SVM	4.833
SVM	4.833
LR	5.417

TABLE VIII
WILCOXON SIGNED RANK TEST (WSR) AND RANK TEST (RT)

$\beta = 5$	RT	WSR
Bayes2-LR	0.006	0.001
$\beta = 10$	RT	WSR
Bayes2-LR	0.0004	0.0005
$\beta = 20$	RT	WSR
Bayes2- LR	0.0004	0.0005
$\beta = 40$	RT	WSR
Bayes2-LR	0.0003	0.00004

$\beta = 5$	RT	WSR
Bayes1-LR	0.005	0.001
$\beta = 10$	RT	WSR
Bayes1-LR	0.0005	0.0005
$\beta = 20$	RT	WSR
Bayes1- LR	0.0005	0.0005
$\beta = 40$	RT	WSR
Bayes1-LR	0.0004	0.00004

[32]. The choice of the cost factor β was made under the assumption that a defect uncovered in production is more costly than testing for bugs in a defect free class. Different choices of cost factor β could yield different results, it is up to the project team to set a suitable value for β based on available resources and time constraints.

B. External Validity

The results presented holds good for the 12 Promise projects. The experiment may lead to different results if another dataset is used. Further the metrics used are a combination of several metrics proposed by different authors, these metrics have been adopted in several defect prediction studies. Different set of metrics could lead to different results.

VIII. Conclusion

In this paper, we have formulated the Bayesian logistic regression with three different priors for defect prediction problem. We study the performance of Bayesian approach for logistic regression by making use of several versions of 12 promise projects. The performance of Bayesian logistic Regression with informative priors is significantly better than logistic regression. The performance of Bayesian logistic regression is very much dependent on the choice of prior. Bayesian logistic regression with informative priors outperform their counterpart, Bayesian logistic regression with non-informative prior. The comparative study of all classifiers reveal that Random Forest ranks first and Bayesian logistic regression with Bayes-2 prior, Bayesian logistic regression with Bayes-1 prior ranks second and third as compared with other widely used classifiers.

References

[1] J. Van Houwelingen and S. Le Cessie, "Predictive value of statistical models," *Statistics in medicine*, vol. 9, no. 11, pp. 1303–1325, 1990.

[2] J. Scott Long, "Regression models for categorical and limited dependent variables," *Advanced quantitative techniques in the social sciences*, vol. 7, 1997.

[3] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[4] E. M. Schulz, D. Betebenner, and M. Ahn, "Hierarchical logistic regression in course placement," *Journal of Educational Measurement*, vol. 41, no. 3, pp. 271–286, 2004.

[5] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.

[6] A. G. Koru, D. Zhang, K. El Emam, and H. Liu, "An investigation into the functional form of the size-defect relationship for software modules," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 293–304, 2009.

[7] R. Subramanyam and M. S. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects," *IEEE Transactions on software engineering*, vol. 29, no. 4, pp. 297–310, 2003.

[8] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on software engineering*, vol. 22, no. 10, pp. 751–761, 1996.

[9] T. Zimmermann, N. Nagappan, and A. Zeller, "Predicting bugs from history.," *Software Evolution*, vol. 4, no. 1, pp. 69–88, 2008.

[10] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, pp. 417–428, IEEE, 2004.

[11] R. Jindal, R. Malhotra, and A. Jain, "Software defect prediction using neural networks," in *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference on*, pp. 1–6, IEEE, 2014.

[12] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Information Sciences*, vol. 264, pp. 260–278, 2014.

[13] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and W. Meding, "Analyzing defect inflow distribution and applying bayesian inference method for software defect prediction in large software projects," *Journal of Systems and Software*, vol. 117, pp. 229–244, 2016.

[14] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[15] "The promise repository of empirical software engineering data," 2015.

[16] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[17] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on software engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[18] M.-H. Tang, M.-H. Kao, and M.-H. Chen, "An empirical study on object-oriented metrics," in *Software Metrics Symposium, 1999. Proceedings. Sixth International*, pp. 242–249, IEEE, 1999.

[19] R. Martin, "Oo design quality metrics," *An analysis of dependencies*, vol. 12, pp. 151–170, 1994.

[20] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[21] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, p. 9, ACM, 2010.

[22] D. Firth, "Bias reduction of maximum likelihood estimates," *Biometrika*, vol. 80, no. 1, pp. 27–38, 1993.

[23] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[24] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.

[25] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[26] N. G. Polson, J. G. Scott, and J. Windle, "Bayesian inference for logistic models using polya-gamma latent variables." Most recent version: Feb. 2013., 2013.

[27] G. Heinze and M. Ploner, *logistf: Firth's Bias-Reduced Logistic Regression*, 2016. R package version 1.22.

[28] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[29] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[30] K. Muthukumaran, A. Dasgupta, S. Abhidnya, and L. B. M. Neti, "On the effectiveness of cost sensitive neural networks for software defect prediction," in *International Conference on Soft Computing and Pattern Recognition*, pp. 557–570, Springer, 2016.

[31] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using bayesian network classifiers," *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp. 237–257, 2013.

[32] K. Muthukumaran, A. Choudhary, and N. B. Murthy, "Mining github for novel change metrics to predict buggy files in software systems," in *Computational Intelligence and Networks (CINE), 2015 International Conference on*, pp. 15–20, IEEE, 2015.