Predictive Models for Bug Fixing Time: An In-depth Analysis with Hidden Markov Model

By

Gurrala Madhu Mohan Vamsi
21VV1F0012

Under the esteemed guidance of
Dr. B. Tirimula Rao, M.Tech, Ph.D.
Assistant Professor & HOD
Department of Information Technology

Abstract

A significant contribution of this disseration is the suggested Hidden Markov Model (HMM)-based temporal sequence activity model. HMMs are a good option for estimating the time needed to correct bugs because they are effective at modeling sequential data. HMMs are not the only strategy being looked at, though. This study investigates the effectiveness of Multinomial Naive Bayes (MNB), Random Forest Classifier (RFC), Linear Support Vector Machine (SVM), and Non-linear Support Vector Machine (NSVM) in addition to HMMs. These algorithms provide several viewpoints on bug-fixing time prediction, each with particular advantages and disadvantages. Extensive experiments were carried out utilizing bug reports from the Firefox bugs dataset to assess the performance of these models. The outcomes of these tests offer insightful information on the viability and relative efficacy of each model.

Introduction

- Every reported bug follows a lifecycle till closure. A bug life cycle illustrates the journey of a bug from the time it is created to the time it is fixed and closed. A generic bug lifecycle is explained below:

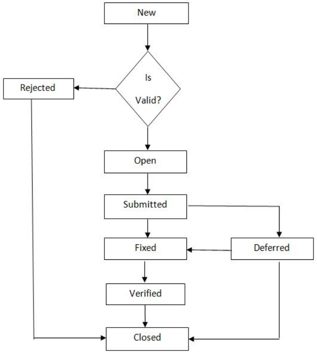
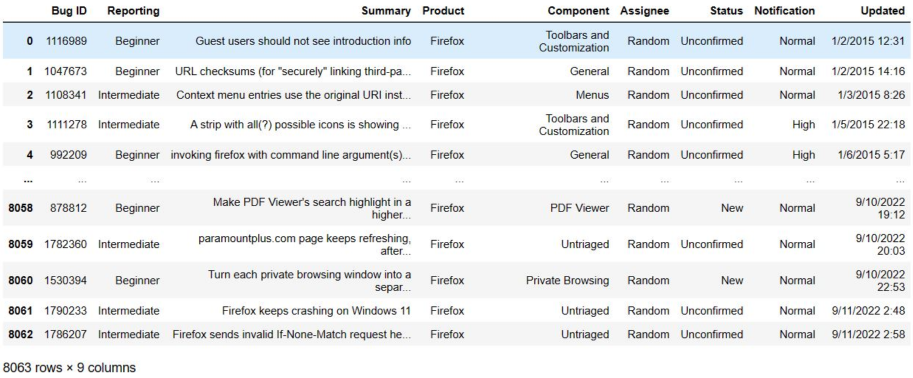


Figure: Simplified Bug Life Cycle

Dataset

- (https://www.kaggle.com/datasets/azizullah444/firefox-bugs) This dataset was obtained from the Kaggle website and describes the Firefox bugs. Firefox bugs data were selected for a 8 year period ranging from 2015 to 2022. It contains information about 8063 bugs.
- In general we have six bug reports from Bugzilla repositories which are fairly used namely Eclipse, Freedesktop, GCC, Gnome, Mozilla and WineHQ. In our project, it is based on Mozilla repository.
- **Input Attributes**
  1. **Reporting** - Beginner, Intermediate, Advanced
  2. **Assignee** - Random, Particular
  3. **Status** - New, Unconfirmed
  4. **Notification** - Normal, High

Dataset Contd..



	Bug ID	Reporting	Summary	Product	Component	Assignee	Status	Notification	Updated
0	1116989	Beginner	Guest users should not see introduction info	Firefox	Toolbars and Customization	Random	Unconfirmed	Normal	1/2/2015 12:31
1	1047673	Beginner	URL checksums (for "securely" linking third-pa...	Firefox	General	Random	Unconfirmed	Normal	1/2/2015 14:16
2	1108341	Intermediate	Context menu entries use the original URI inst...	Firefox	Menus	Random	Unconfirmed	Normal	1/3/2015 8:26
3	1111278	Intermediate	A strip with all(?) possible icons is showing ...	Firefox	Toolbars and Customization	Random	Unconfirmed	High	1/5/2015 22:18
4	992209	Beginner	invoking firefox with command line argument(s)...	Firefox	General	Random	Unconfirmed	High	1/6/2015 5:17
...	...	...	...	...	...	...	...	...	...
8058	878812	Beginner	Make PDF Viewer's search highlight in a higher...	Firefox	PDF Viewer	Random	New	Normal	9/10/2022 19:12
8059	1782360	Intermediate	paramountplus.com page keeps refreshing, after...	Firefox	Untriaged	Random	Unconfirmed	Normal	9/10/2022 20:03
8060	1530394	Beginner	Turn each private browsing window into a separ...	Firefox	Private Browsing	Random	New	Normal	9/10/2022 22:53
8061	1790233	Intermediate	Firefox keeps crashing on Windows 11	Firefox	Untriaged	Random	Unconfirmed	Normal	9/11/2022 2:48
8062	1786207	Intermediate	Firefox sends invalid If-None-Match request he...	Firefox	Untriaged	Random	Unconfirmed	Normal	9/11/2022 2:58

8063 rows × 9 columns

Figure: Firefox Bugs Dataset

Architecture

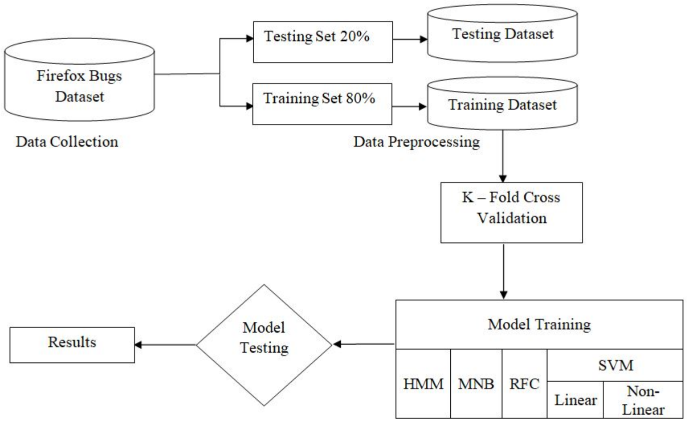


Figure: Architecture

Hidden Markov Model

- A Hidden Markov Model (HMM) is a statistical model used in various fields such as speech recognition, natural language processing, bioinformatics, and many others for modeling sequences of data.
- Hidden Markov Models (HMMs) can be used in bug prediction and defect forecasting to model the evolution of software defects over time. HMMs can help in understanding and predicting the occurrence of software bugs by considering various factors and their hidden states.

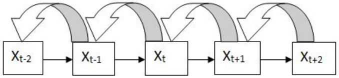


Figure: HMM Model

$$P(Hidden\ Markov\ Model) = \frac{P(X_t)}{P(X_{t-1})} \tag{1}$$

Hidden Markov Model Contd..

It has 2 states and 3 probabilities. The 2 states are as follows:

1. **Hidden State:** These are not directly observed, their presence is observed by observation symbols that hidden state emits.
2. **Observable State:** One that can be observed or seen.

And the 3 probabilities are as follows:

1. **Initial Probability:** Starting state probability π_i is the state that the markov chain will start in state i.
$$\pi_i = P(S_i) \tag{2}$$
2. **Transition Probability:** Each a_{ij} resenting the probability of moving from state i to state j.
$$a_{ij} = \frac{P(S_i)}{P(S_j)} \tag{3}$$
3. **Emission Probability:** A sequence of observation likelihoods.

$$P(Emission\ Matrix) = \frac{P(Value\ in\ Observable\ state)}{P(Value\ in\ Hidden\ state)} \tag{4}$$

Naive Bayes Multinomial

- Naive Bayes with a Multinomial distribution, often referred to as Multinomial Naive Bayes, is a probabilistic classification algorithm that is particularly well-suited for text classification and document classification tasks.
- Multinomial Naive Bayes can be applied to bug prediction over time, especially when you want to classify the likelihood of defects or bugs occurring in software based on historical data and various features.
- Bayes Theorem : Based on knowledge of conditions that might be associated to the event, it describes the likelihood of an event. The equation is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5}$$

Where,
P(A | B) = posterior probability of class A given predictor B
P(A) = prior probability of class
P(B | A) = likelihood which is the probability of predictor given class
P(B) = prior probability of the predictor

Random Forest Classifier

- A Random Forest Classifier is an ensemble learning method for classification tasks in machine learning. It is a versatile and powerful algorithm that combines the predictions of multiple decision trees to improve the accuracy and robustness of the classification.
- Random Forest Classifier can be a valuable tool for bug prediction over time, as it can effectively handle complex relationships within software development data and provide robust predictions.
- Random Forest Classifier is a powerful tool for bug prediction over time, as it can capture non-linear relationships and interactions between features. It is robust and less prone to overfitting compared to individual decision trees, making it suitable for real-world software development data.

Linear Support Vector Machine

- A Linear Support Vector Machine (Linear SVM) is a supervised machine learning algorithm that is primarily used for classification tasks. It is a member of the broader class of Support Vector Machines (SVMs) and is specifically designed for linearly separable data, where the classes can be separated by a hyperplane.
- Linear SVMs are particularly well-suited for scenarios where you have features representing historical data and want to predict future bug occurrences in different time intervals.
- Linear SVMs offer a straightforward approach to bug prediction over time, where the emphasis is on linear separation of bug rate classes in a multidimensional feature space. If the data is not linearly separable, you might explore non-linear kernels or other classification algorithms better suited to your specific data characteristics.

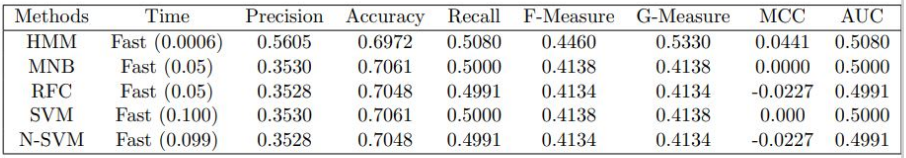
Non Linear Support Vector Machine

- A Non-Linear Support Vector Machine (Non-Linear SVM) is a machine learning algorithm that extends the capabilities of a linear SVM to handle non-linearly separable data. It accomplishes this by transforming the feature space into a higher-dimensional space, making it possible to find a hyperplane that separates the data into different classes.
- Build a Non-Linear SVM classifier. This classifier uses a kernel function, such as the Radial Basis Function (RBF) kernel, to implicitly map the feature space into a higher-dimensional space where non-linear relationships can be captured.
- Non-Linear SVMs are particularly valuable when the relationship between features and bug rates is complex and non-linear, as they can capture these intricate patterns effectively. The choice of the appropriate kernel and its parameters is crucial for achieving good bug prediction results.

Results

- Our project's primary objective is to categorise bug reports into slow and fast. "slow" means that it will take a while to fix the bug. "fast" means that fixing the bug won't take long. This was discovered through an examination of the bug report sequence using HMM, MNB, RFC, Linear SVM and Non Linear SVM models that had been put into practice. These models are also tested using a number of measures, including recision, Accuracy, Recall, F-Measure, G-Measure, MCC, and AUC.

Table 8.1: Test Results



Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.0006)	0.5605	0.6972	0.5080	0.4460	0.5330	0.0441	0.5080
MNB	Fast (0.05)	0.3530	0.7061	0.5000	0.4138	0.4138	0.0000	0.5000
RFC	Fast (0.05)	0.3528	0.7048	0.4991	0.4134	0.4134	-0.0227	0.4991
SVM	Fast (0.100)	0.3530	0.7061	0.5000	0.4138	0.4138	0.000	0.5000
N-SVM	Fast (0.099)	0.3528	0.7048	0.4991	0.4134	0.4134	-0.0227	0.4991

Comparative Performance

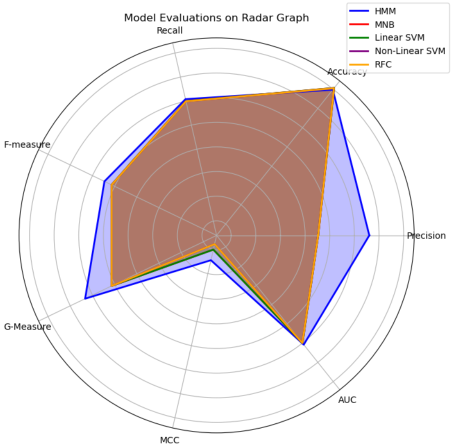


Figure: Radar Graph for Performance in Methods

Comparative Metrics

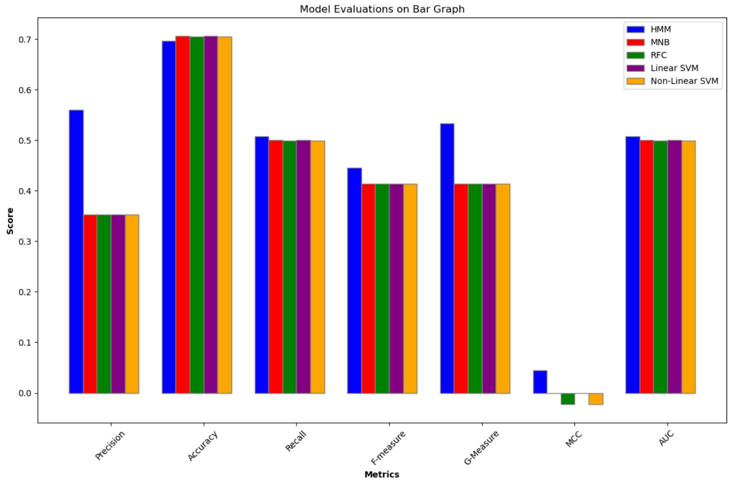


Figure: Evaluation metrics for HMM, MNB,RFC, SVM

Conclusion

In a comparative evaluation of various machine learning methods, Hidden Markov Models (HMM) emerged as a standout choice due to their exceptional precision and recall, making them particularly well-suited for tasks requiring accurate positive predictions and true positive capture. HMM's balance between precision and recall, evident in its F-measure and G-Measure, is advantageous in applications where striking that balance is critical. Furthermore, HMM's superior discriminative power, reflected in a higher AUC, makes it suitable for tasks that demand effective differentiation between positive and negative cases, such as credit risk assessment. While other methods like Multinomial Naive Bayes and Random Forest Classifier have their strengths, HMM's blend of precision, recall, and discrimination sets it apart, making it a strong contender for various machine learning applications, with the choice of method ultimately aligning with the specific task requirements and objectives.

References

1. Pombo, N., Teixeira, R. (2020, October). Contribution of temporal sequence activities to predict bug fixing time. In 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT) (pp. 1-6). IEEE.
2. Gomes, L., da Silva Torres, R., Côrtes, M. L. (2023). BERT-and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: a comparative study. Information and Software Technology, 160, 107217.
3. Thirumoorthy, K. (2022). A feature selection model for software defect prediction using binary Rao optimization algorithm. Applied Soft Computing, 131, 109737.
4. Zhu, K., Ying, S., Ding, W., Zhang, N., Zhu, D. (2022). IVKMP: A robust data-driven heterogeneous defect model based on deep representation optimization learning. Information Sciences, 583, 332-363

Thank you!