Comparative Analysis of Bug Fixing Time Prediction Models: Hidden Markov Model and Beyond

Abstract:

The abstract of the paper provides a brief summary of the research effort and addresses the ongoing difficulties that practitioners and development team's encounter while trying to remedy bugs. The complexity of calculating bug-fixing time has prompted researchers to investigate a range of predictive models. A significant contribution of this research 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.
Index terms: temporal activities, predict bug fixing time, hidden markov model, naïve bayes multinomial, random forest classifier, support vector machine, software quality

1 Introduction:

A software bug is a problem causing a program to crash or produce invalid output. The problem is caused by insufficient or erroneous logic. A bug can be an error, mistake, defect or fault, which may cause failure or deviation from expected results.

Some bugs might not have serious effects on the functionality of the program and may remain undetected for a long time. A program might crash when serious bugs are left unidentified. Another category of bugs called security bugs may allow a malicious user bypass access controls and obtain unauthorized privileges [2].

The journey of any error in the code to being recognized as a Bug is explained below.

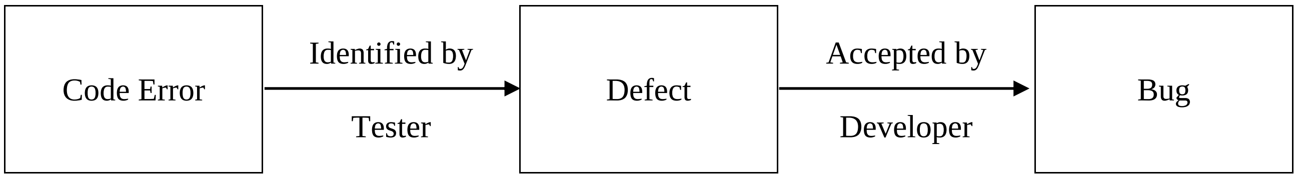


Fig: Code to Bug Journey

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:

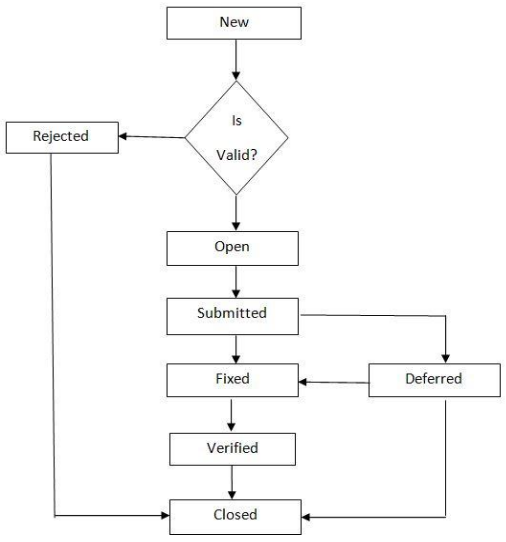


Fig: Simplified Bug Life Cycle

The above figure shows the working procedure of bug life cycle that includes New, Valid, Open, Rejected, Submitted, Deferred, Fixed, Verified and Closed. When a bug is posted, it is in New state. And checks whether the bug is valid or not. If valid it goes to Open state or it gets closed. When the bug is accepted by developers, it is in open state. When the bug is rejected by developers, it is in rejected state. When bug reaches developers, it is in Submitted state. When the bug is in hold, it is in Deferred state. Sometimes, deferred bugs go to closed state. When the bug is fixed by developers, it is in Fixed state. And it checks the fixed bug is verified or not. After the testing has passed, then the bug moved to Closed state.

The bug fixing tasks includes [3]: triage, assignment and tracker. The triage aims to allocate certain bug to specific developer. It simply rely on severity and priority. It can handle multiple bugs at a time by assigning multiple particular developers. On the contrary, assignment aims to connect bug with suitable developer. In this case, when the assigned developer cannot handle specific bug due to their inability or incorrect assignment then another developer is assigned. This process will be continued until the bug meets particular developer. And this process is called bug tossing [4]. Finally, the bug tracker keeps track of reported software bugs. And the bug fixing process becomes a challenge. On the hand, it requires combining limited number of software developers along with multitude of bugs. On the other hand, a significant time is spent by software developers. Thus the bug fixing estimation techniques becomes accurate [5].

The two main important contributions of this work are:

- Firstly, we proposed a dataset of temporal sequence activity
- Secondly, we provide an experiment that includes classification model to predict bug fixing time

The paper is as follows: In section II, it consists of Related Works related to prediction of bug fixing time. In section III, it introduces Proposed Work i.e., Hidden Markov Model (HMM). In section IV, it consists of Experimental Works related to Naïve Bayes Multinomial, Random Forest Classifier, Linear and Non Linear Support Vector Machine. In section V, it consists of Methodology. In section VI, it consists of obtained results. In section VII, finally discusses about conclusion and future work.

2 Related Works:

A number of studies that aimed to develop learning models for bug-fix time estimation have been published in the literature. A summary of these works are represented in this section.

In a study conducted by the authors [6], crash reports from Mozilla Firefox were analyzed with linear models where the current value of the series is regressed on its previous values. In a related study [7], researchers suggested using Instead of using past values of the dependent variable, these models use past white noise error terms. In the mentioned study [8], the authors examined the factors that contribute to delays in open-source software systems. They identified three main factors: bug reports, the source code associated with the fix, and the changes required in the source code to resolve the bug. Based on their findings, the authors suggest combinations of AR and MA models and are especially popular for time series forecasting.

Based on the prioritizing principles discussed in references [9], the authors proposed a predictive model that utilizes models provide a general framework for modeling time series and include HMMs as a special case. In their study [10], the authors proposed that these are recursive algorithms that estimate the state of a linear dynamic system from a series of noisy measurements. They are related to state space models and can be thought of as a predecessor to HMMs in some applications. In the study mentioned in [11], researchers utilized a model to especially use in speech recognition; DTW is a method to align two sequences in an optimal way. In a study conducted by the authors in [12], it was observed that these are abstract machines used to recognize patterns and sequences. They don't account for probabilities like HMMs, but they can be thought of as a deterministic precursor to HMMs.

In [13], researchers explored that before HMMs, there were simple Markov chains which describe a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. In a study by authors [14], they implemented while they can be used in a wide variety of applications and don't strictly predate HMMs, Bayesian networks can be used to model probabilistic relationships among a set of variables. In a study [15

– 16] these are techniques for dimensionality reduction and classification which might be applied to time series data, though they don't model sequences in the way HMMs do.

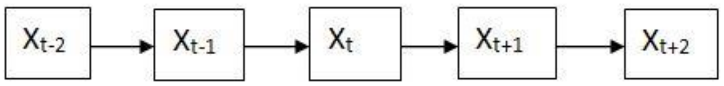
In a study by [17], although they've gained significant traction in recent years with the advent of deep learning, simpler neural network architectures have been around for a long time and were used for tasks like speech recognition before HMMs and deep learning models took over. In a study by authors [1], they proposed using a Hidden Markov Model (HMM) to classify bug reports as having either fast or slow resolution times. The experiments conducted on the Firefox project showed that this approach achieved approximately 5% higher Accuracy in predicting the resolution time of bugs. In a study, there have been studies proposing models to predict the total time needed for bug fixing activities and suggesting optimization of triage or assignment processes to decrease bug fixing time. These approaches aim to improve efficiency and reduce delays in resolving software bugs.

In summary then, we may consider the time required to predict a bug fixing time. This study leads to triage or assignment for a developer.

3 Proposed Work:

Hidden Markov Model

Here, we will understand about hidden markov model from its basics. For that, we will learn about conditional probability first.



$$\begin{tikzcd} \boxed{X_{t-2}} \arrow[r] & \boxed{X_{t-1}} \arrow[r] & \boxed{X_t} \arrow[r] & \boxed{X_{t+1}} \arrow[r] & \boxed{X_{t+2}} \end{tikzcd}$$

Fig: Conditional Probability

These X_{t-2}, X_{t-1}, X_t, X_{t+1}, X_{t+2} are different events. For these events the conditional probability is given by,

$$P(\text{Conditional Probability}) = P(X_{t-1} \mid X_t)$$

This means outcome that satisfies X_t, then how likely it satisfies X_{t-1}. And here, X_{t-1} is dependent on X_t.

- Bayes' theorem, which provides a way to update probabilities based on new evidence.
- Statistical inference, where we often want to know the probability of a hypothesis given observed data.
- Many machine learning algorithms, especially in classification problems where we want to know the probability of a class given a set of features.

And Hidden Markov Model is introduced to reduce the limitations of Conditional probability. The below figure shows the working procedure of Markov chains.

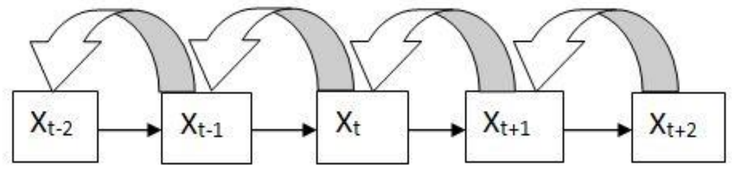


Fig: Hidden Markov Model

P(Hidden Markov Model) = P(X_t | X_{t-1})

This means outcome that satisfies X_{t-1}, then how likely it satisfies X_t. And here, X_t is dependent on X_{t-1}.

Here,

X_t = current state

X_{t-1} = previous state

X_{t+1} = next state

The Hidden Markov Model (HMM) is a statistical model that represents systems that are governed by an underlying process which is not directly observable (hence "hidden"). This underlying process can be thought of as being in one of a set of states at any given time. Each state generates an observable output according to a certain probability distribution. Transitions between states are governed by transition probabilities [22].

Algorithm: Pseudo code for Hidden Markov Model

```
FUNCTION PREDICT_BUG_FIXING_TIME(observations, states, trans_prob, emit_prob, state_time_estimates):
    // Extract the most recent observation from the observation sequence
    latest_observation = observations[LAST INDEX]
    // Based on the latest observation, infer the most probable current state
    max_prob = -INFINITY
    current_state = NULL
    FOR EACH state IN states:
        emission_probability = emit_prob[state][latest_observation]
        IF emission_probability > max_prob:
            max_prob = emission_probability
            current_state = state
    END FOR
    // Now that we have the most probable current state, predict the most probable next state
    max_transition_prob = -INFINITY
    next_state = NULL
    FOR EACH state IN states:
        transition_probability = trans_prob[current_state][state]
        IF transition_probability > max_transition_prob:
            max_transition_prob = transition_probability
            next_state = state
    END FOR
    // Estimate the time required for the next state from predefined estimates
    predicted_time = state_time_estimates[next_state]
    RETURN next_state, predicted_time
END FUNCTION
```

Extract the Latest Observation: We get the most recent observation from the provided sequence.

Determine Current State: Based on the emission probabilities, we infer which state is most likely to have emitted the latest observation.

Predict Next State: Using the transition probabilities from the inferred current state, we predict which state is most likely to be next.

Estimate Time: We use a predefined dictionary (or list) of time estimates to predict how long the next state (bug-fixing stage) will take.

HMM is a statistical model in which the system is being modeled and is assumed to be a markov process with hidden states. The property of markov process is probability of subsequent state depends only on the previous state. Since, it never depends on any other probabilities; it is called as memory less process. HMM has a set of states each of which has limited number of transactions and emissions. Each transaction between states has an assigned probability. Each model starts from start state to ends in end state [23].

HMMs are a type of statistical model that captures systems with hidden or unobservable states. These hidden states produce observable outputs (or observations) based on certain probability distributions [24].

It has 2 states and 3 probabilities. The 2 states are as follows [25]:

- **Hidden State:** These are not directly observed, their presence is observed by observation symbols that hidden state emits. i.e. X1, X2, X3
- **Observable State:** One that can be observed or seen. i.e. Y1, Y2, Y3

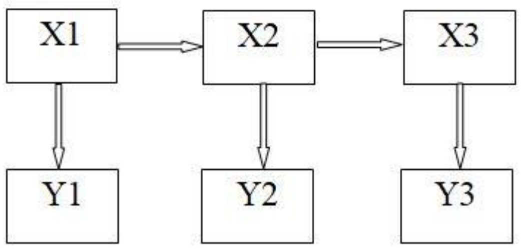


Fig: States in HMM

And the 3 probabilities are as follows.

- **Initial Probability:** Starting state probability π_i is the state that the markov chain will start in state i.

$$\pi_i = P(S_i)$$

- **Transition Probability:** Each a_{ij} resenting the probability of moving from state i to state j in hidden state.

$$a_{ij} = \frac{P(S_i)}{P(S_j)}$$

- **Emission Probability:** A sequence of observation likelihoods.

$$P(\text{Emission probability}) = \frac{\text{P(value in observable state)}}{\text{P(value in hidden state)}}$$

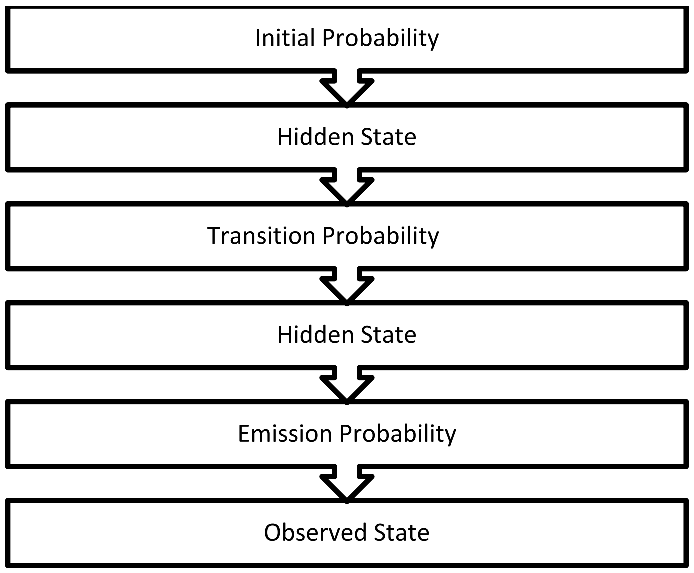


Fig: Block diagram of HMM model

4 Experimental Works:

4.1 Naïve Bayes Multinomial:

A probabilistic learning classifier based on the Bayes theorem with the "naive" assumption of independence between every pair of features. MNB is particularly suitable for classification of discrete data, especially in text classification where the features can represent the frequency of occurrence of particular words or phrases. [18]

- **Bayes Theorem:** It describes the probability of an event, based on prior knowledge of conditions that might be related to the event. The formula is:

  $$P(A \mid B) = \frac{P(B|A) * P(A)}{P(B)}$$

  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.
- **Naive Assumption:** Even though the features might be interdependent, this classifier assumes that the features are independent. This assumption is "naive", hence the name.
- **Multinomial Distribution:** While the basic Naive Bayes classifies features with binary outcomes (like presence or absence), the Multinomial Naive Bayes is used for features that can have multiple outcomes. This is common in text data where words can appear multiple times.

4.2 Random Forest Classifier:

A common machine learning technique used for classification and regression applications is the Random Forest Classifier. It is a member of the ensemble learning family, which means it integrates the results of numerous different models to provide predictions that are more reliable and accurate than those produced by any one model alone. Random Forests are renowned for their adaptability and efficacy across a wide range of domains, and they are particularly successful for classification issues. [19]

Decision Trees: Decision trees are the central component of a Random Forest. By dividing the data into several branches and nodes, a decision tree is a flowchart-like structure that makes decisions based on input features. Each branch stands for a conclusion or a result of classification.

Ensemble of Trees: A Random Forest is made up of many decision trees, which are frequently present. Although these trees are independently built, a random element is added during construction. There are two basic origins of the randomness:

- **Bootstrap Aggregation:** A random subset of the training data, chosen using replacement, is used to train each tree. Bootstrapping is the term for this procedure. A somewhat different subset of the data is thus shown to each tree as a result.
- **Feature Randomness:** Only a random subset of features are taken into account while making decisions at each node of a tree, as opposed to all of the features that are accessible. As a result, the ensemble becomes more diversified and the association between the trees is decreased.

Voting: Each tree in the Random Forest predicts the class label of a data point for classification tasks. A majority vote determines the outcome of the forecast. In other words, the final prediction is chosen from the class that is predicted by the majority of individual trees.

Aggregate Predictions: The final prediction for regression problems is calculated by averaging the predictions of each individual tree.

4.3 Support Vector Machine

SVM works by finding a hyperplane that best separates the classes of data. The "best" hyperplane is the one that maximizes the margin between the two classes. This margin is defined by the closest points in each class to the hyperplane, and these points are known as "support vectors". Hence the name "Support Vector Machine".

3.3.1 Linear Support Vector Machine

Linear SVM is a specific type of Support Vector Machine (SVM) used for classification tasks when the data is linearly separable, meaning the data from different classes can be separated by a straight line (in two dimensions), a flat plane (in three dimensions), or a hyperplane in higher dimensions. [20]

- **Decision Boundary (Hyperplane):** In a Linear SVM, the decision boundary is linear. This hyperplane is chosen to best separate the data points of different classes. In two dimensions, this hyperplane is simply a line.
- **Margin:** The margin is defined as the distance between the separating hyperplane (decision boundary) and the closest data points from each class. These closest points are known as the support vectors. The goal of a Linear SVM is to maximize this margin.
- **Support Vectors:** These are the data points that lie closest to the hyperplane and influence its position and orientation. Only these points are relevant in defining the hyperplane; others can be altered or removed without affecting the hyperplane's position, given they remain on the same side of the boundary.
- Given a labeled dataset, Linear SVM will try to find the best hyperplane that separates the classes while maximizing the margin.
- The equation of the hyperplane is given by $w \cdot x + b = 0$
  Where,
  w = weight (normal) vector
  x = input vector
  b = bias
- Once the hyperplane is established, any new data point is classified based on which side of the hyperplane it falls on.

3.3.2 Non Linear Support Vector Machine

A Non-Linear SVM is an extension of the basic SVM to handle cases where the data is not linearly separable. In many real-world scenarios, data cannot be separated by a straight line (or hyperplane in higher dimensions). Non-Linear SVM addresses this challenge. [21]

- **Kernel Trick:** The primary technique used in Non-Linear SVM is the kernel trick. The main idea is to map the original, non-linearly separable data into a higher-dimensional space where it becomes linearly separable. Once transformed, a linear SVM can be applied in this higher-dimensional space.
- **Kernel Functions:** These are mathematical functions used in the kernel trick to compute the dot product in the transformed space without explicitly carrying out the transformation. Commonly used kernel functions include:

Polynomial: $(x\cdot x'+c)^d$

Radial Basis Function (RBF) or Gaussian: $\exp(-\gamma\|x-x'\|^2)$

Sigmoid: $\tanh(\kappa x\cdot x' + c)$

- **Mapping to Higher Dimension:** Given a non-linearly separable dataset in its original space, use a kernel function to implicitly map the data to a higher-dimensional space.
- **Finding the Hyperplane:** In this higher-dimensional space, the data becomes linearly separable. Now, the Linear SVM technique is used to find the optimal separating hyperplane.
- **Classification:** For a new data point, the same mapping is applied, and then it's classified based on its position relative to the hyperplane in the higher-dimensional space.
- **Back to Original Space:** The separating hyperplane in the higher-dimensional space corresponds to a non-linear decision boundary in the original space. Thus, in the original space, the SVM provides a non-linear decision boundary.

Table: Overview of Parameters of the Studied Classification Techniques

Family	Family description		Parameter name	Parameter description	Studied classification techniques with their default (in bold) and candidate parameter values
SVM	Support Vector Machine (SVM) uses a hyperplane to separate Two classes. i.e., SVM and N-SVM	SVM with Linear SVM	cost	[N] The trade-off between maximizing the margin between support vectors and minimizing the classification error is controlled by the Cost parameter in a linear SVM.(0.1 for small, 1.0 for moderate and 1000 for large).	C = {0.1, **1.0**, 1000}
		SVM with Non Linear SVM (rbf)	cost	[N] The C parameter governs the trade-off between maximizing the margin between support vectors and minimizing the classification error in non-linear SVMs, just like it does in linear SVMs. (0.1 for small, 1.0 for moderate and 1000 for large).	C = {0.1, **1.0**, 1000}

			gamma	[F] The RBF kernel's unique gamma parameter regulates the decision boundary's form. If you select "scale," the gamma value will be determined based on the scale of the input features. This can be a practical option, particularly if you have features with different scales.	gamma = {0.1, **scale**, auto}
			kernel	[F] With regard to coping with non-linear correlations between the input features and the target variable, the RBF kernel is a popular option for SVM regression.	kernel = {**rbf**, poly, sigmoid}
MNB	MNB is used for jobs involving text classification, especially when dealing with discrete data types like text. It is a Naive Bayes algorithm extension that performs well with features that reflect counts or frequencies of different categories.		alpha	[N] This hyperparameter, called the Laplace smoothing parameter, regulates how much smoothing is done to the probability estimations. It makes use of the 1.0 default value.	alpha = {**1.0**, 0.1, 0.01, 2.0, 5.0}
			fit_prior	[L] fit_prior is often set to True, meaning that the class priors are calculated from the training set. The prior probability for each class is represented by class priors. If False is selected, uniform priors are used, presuming equal likelihood for all classes.	fit_prior = {**True**, False}
			class_prior	[F] It makes use of class priors that were calculated from training data. However, if necessary, you can define unique class priors.	class_prior = {**None**, 0.3, 0.2, 0.5}
RFC	With the help of several decision trees combined, the Random Forest classifier uses		n_estimato rs	[N] The number of decision trees in the forest is controlled by this parameter. To ensure forest diversity, it's frequently set	n_estimators = {10, **100**, 500, 1000}

	ensemble learning to create predictions. It is a flexible and potent method used for both regression and classification tasks.		to a high value, which enhances the generalisation of the model. 100 is the default value.	
		random_state	[F] An integer or instance of RandomState that manages the random seed for consistency. Set this parameter to a fixed value if you want the results to stay constant between runs.	random_state = {**None**, 42}
HMM	For a variety of sequential data modeling tasks, including speech recognition, natural language processing, and bioinformatics, Hidden Markov Models (HMMs), a sort of probabilistic model, are used.	n_components	[N] The HMM should contain n_components hidden states, according to this parameter. Each state stands for a specific underlying or concealed condition or circumstance. The fundamental structure of the system you are modelling is captured by these hidden states.	n_components = {**1**, 2, 40}
		covariance_type	[F] This parameter determines the type of covariance matrix used to model the emissions. it's set to "diag," which means diagonal covariance matrices are used. Other options include "spherical," "tied," and "full," each of which represents different assumptions about the covariance structure.	covariance_type = {**full**, tied, diag, spherical}
		n_iter	[N] The HMM's training iteration count is set by this parameter. The training procedure will run for a maximum of 1000 iterations if it is set to 1000.	n_iter {**10**, 100, 1000}

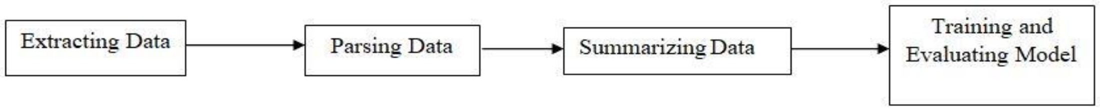
[N] Denotes a numerical value

[F] Denotes a factorial value

[L] Denotes a logical value

5 Methodology

The workflow is as follows.



Fig; Proposed System Architecture

Data extraction is the process of collecting or retrieving disparate types of data from a variety of sources, many of which may be poorly organized or completely unstructured. Data extraction makes it possible to consolidate, process, and refine data so that it can be stored in a centralized location in order to be transformed.

Data parsing is the process of transforming data from one format to another. In detail, data parsing is typically used for structuring data. This means converting unstructured data into structured or more structured data.

Summarization includes counting, describing all the data present in data frame. And finally training and evaluating the model. [23]

5.1 Data Collecting

In general we have six bug reports from Bugzilla repositories which are fairly used in research papers: Eclipse, Freedesktop, GCC, Gnome, Mozilla and WineHQ [26]. In our experiment it is based on Mozilla repository. The firefox bugs reports dataset is taken from 2015 to 2022 ranging an 8 years period.

From the dataset four input attributes are taken.

- Reporting: represents the experience of reporter as Beginner, Intermediate, Advanced.
- Assignee: represents the type of developer to be assigned as Random, Particular
- Status: represents the current status of a bug as New, Unconfirmed
- Notification: represents the activity of a bug as High, Normal

5.2 Data Preprocessing

- Firstly we will assign integer values to the independent variables such as Reporting, Assignee, Status and Notification. And these integer columned values are stored with the names of Trans_Reporting, Trans_Assignee, Trans_Status, Trans_Notification.

- And then we will divide the dataset into 8 data frames consisting of each year bugs in one data frame.
- Now, we split the dataset into training (80% of bug reports total) and testing (20% of bug reports total) sets. There is no fixed rule for selecting the size of the training set or testing set. A good rule of thumb is 75 – 80% for training sets. [26]

5.3 Model Training

The original training dataset is randomly partitioned into k equal-sized subsets. Each subset is called a 'fold'. If the dataset cannot be divided exactly into k subsets, the last few subsets may have one more or one fewer data point than the others.

For each fold:

- Set the fold aside as the validation set.
- Train the model on the remaining k-1 folds combined.
- Validate (or test) the model on the set-aside fold.
- Compute the validation metric (e.g., Accuracy, mean squared error, etc.) for this fold.

After all k folds have been used as the validation set once, average the k validation metrics to get a single measure of model performance.

This is shown in below figure:

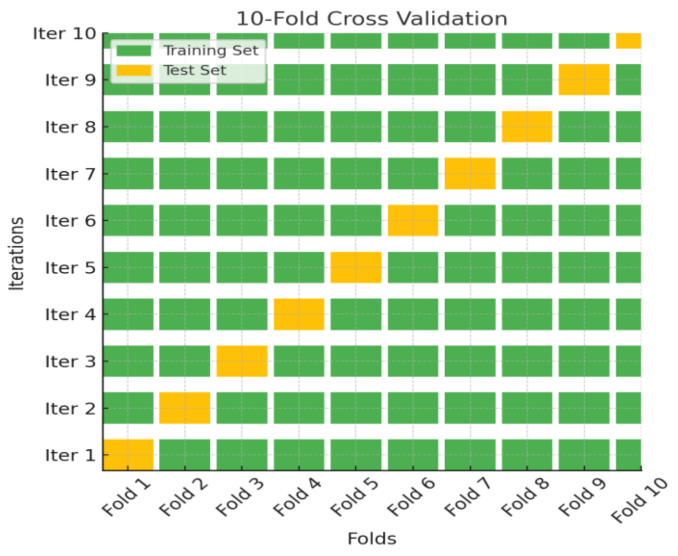


Fig: The Procedure of Cross Validation of one Epoch

10-fold cross-validation is performed on the data using the HMM, MNB, SVM classifiers. The cross-validation process is repeated 10 times. The scores for each iteration and fold are printed.

Additionally, the mean and standard deviation of the scores across all iterations and folds are calculated and printed.

The overall mean (0.5594) and standard deviation (0.1851) of the scores from the HMM classifier's cross-validation are calculated and displayed. This is done up to 60 epochs.

The overall mean (0.7020) and standard deviation (0.0145) of the scores from the MNB classifier's cross-validation are calculated and displayed. This is done up to 50 epochs.

The overall mean (0.7020) and standard deviation (0.0154) of the scores from the RFC classifier's cross-validation are calculated and displayed. This is done up to 50 epochs.

The overall mean (0.7020) and standard deviation (0.0152) of the scores from the Linear SVM classifier's cross-validation are calculated and displayed. This is done up to 50 epochs.

The overall mean (0.7022) and standard deviation (0.0151) of the scores from the Non Linear SVM classifier's cross-validation are calculated and displayed. This is done up to 50 epochs.

5.4 Model Testing

Here we use Naïve Bayes Multinomial, Random Forest Classifier, Linear Support Vector Machine and Non Linear Support Vector Machine along with Hidden Markov Model. This is done to see and compare results among them. In this testing phase, each of the best lived prediction models was validated with 20% of each firefox bug report testing dataset to measure its evaluation metrics. Furthermore, we can predict time required to fix bugs. This is done for each year individually with the eight data frames which we have taken earlier.

5.5 Evaluation Metrics: Assessing Model Performance

Our approach will evaluate several metrics such as Precision, Recall, Accuracy, F-Measure, G-Measure, MCC, and AUC [27,28,29].

Precision = $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$

Recall = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$

Accuracy = $\frac{\mathrm{TP+TN}}{\mathrm{P+N}}$

F-Measure = $2.\frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision+Recall}}$

G-Measure = $2.\sqrt{\mathrm{Recall} * \mathrm{Specificity}}$ Where, Specificity = $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$

MCC (Mathews Correlation Coefficient) = $\frac{(\mathrm{TP} * \mathrm{TN})-(\mathrm{FP} * \mathrm{FN})}{\sqrt{(\mathrm{TP+FP})(\mathrm{TP+FN})(\mathrm{TN+FP})(\mathrm{TN+FN})}}$

AUC (Area Under Curve) = $\frac{1}{2}(\sum_{\mathrm{i=1}}^{\mathrm{n\ pos}} \mathrm{Ri} + \sum_{\mathrm{i=1}}^{\mathrm{n\ neg}} \mathrm{Ri})$

Where, TP = True Positive

FP = False Positive

TN = True Negative

FN = False Negative

5.6 Statistical Measures: Analyzing Model Outputs

Statistical measurements like minimum, maximum, average, mean, standard deviation, and mean square can be included in this section.

Minimum = min(TP, TN, FP, FN)

Maximum = max(TP, TN, FP, FN)

Average = Mean = $\frac{1}{n}\sum_{i=1}^{n} x_i$

Standard Deviation = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - mean)^2}$

Mean Square = $\frac{1}{n}\sum_{1=1}^{n}(x_i)^2$

6 Results

The obtained results are as follows:

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.35)	0.4290	0.5905	0.4515	0.4326	0.4399	-0.1173	0.4515
MNB	Fast (0.05)	0.3431	0.6862	0.5000	0.4069	0.4069	0.0000	0.5000
RFC	Fast (0.05)	0.3921	0.7843	0.5000	0.4395	0.4395	0.0000	0.5000
SVM	Fast (0.09)	0.4117	0.8235	0.5000	0.4516	0.4516	0.0000	0.5000
N-SVM	Fast (0.09)	0.3529	0.7058	0.5000	0.4137	0.4137	0.0000	0.5000

Table 1: Evaluation Metrics Test Results for 2015

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.1052	0.8947	0.5000	0.3496	0.3722
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.5000	0.5000
SVM	0.0000	1.0000	0.5000	0.5000	0.5000
N-SVM	0.0000	1.0000	0.5000	0.5000	0.5000

Table 2: Statistical Measures Test Results for 2015

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.99)	0.4768	0.4712	0.4703	0.4437	0.4736	-0.0523	0.4703
MNB	Fast (0.05)	0.3928	0.7857	0.5000	0.4400	0.4399	0.0000	0.5000
RFC	Fast (0.05)	0.3750	0.7500	0.5000	0.4285	0.4285	0.0000	0.5000
SVM	Fast (0.10)	0.3630	0.7261	0.5000	0.4206	0.4206	0.0000	0.5000
N-SVM	Fast (0.09)	0.3333	0.6666	0.5000	0.4000	0.3999	0.0000	0.5000

Table 3: Evaluation Metrics Test Results for 2016

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.4684	0.5315	0.5000	0.0296	0.2508
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.5000	0.5000
SVM	0.0000	1.0000	0.5000	0.5000	0.5000
N-SVM	0.0000	1.0000	0.5000	0.5000	0.5000

Table 4: Statistical Measures Test Results for 2016

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.005)	0.3917	0.4227	0.4223	0.3785	0.4064	-0.1834	0.4223
MNB	Fast (0.95)	0.6219	0.6178	0.6122	0.6075	0.6170	0.2340	0.6122
RFC	Fast (0.05)	0.5490	0.5447	0.5432	0.5282	0.5465	0.0928	0.5432
SVM	Fast (0.10)	0.5921	0.5609	0.5901	0.5845	0.5574	0.5883	0.1764
N-SVM	Fast (0.10)	0.6113	0.5934	0.5950	0.5787	0.6030	0.2057	0.5950

Table 5: Evaluation Metrics Test Results for 2017

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.1563	0.8436	0.5000	0.2770	0.3267
MNB	0.2500	0.7500	0.5000	0.1776	0.2815
RFC	0.2741	0.7258	0.5000	0.1876	0.2852
SVM	0.2452	0.7547	0.5000	0.1900	0.2861
N-SVM	0.2131	0.7868	0.5000	0.2140	0.2958

Table 6: Statistical Measures Test Results for 2017

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.006)	0.4443	0.3791	0.4235	0.3630	0.4337	-0.1304	0.4235
MNB	Fast (0.05)	0.3727	0.7455	0.5000	0.4271	0.4271	0.0000	0.5000
RFC	Fast (0.05)	0.8869	0.7751	0.5128	0.4612	0.6498	0.1408	0.5128
SVM	Fast (0.09)	0.9101	0.8224	0.5312	0.5094	0.6709	0.2264	0.5312
N-SVM	Fast (0.09)	0.9277	0.8579	0.5555	0.5610	0.6949	0.3082	0.5555

Table 7: Evaluation Metrics Test Results for 2018

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.3444	0.6555	0.5000	0.1099	0.2620
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.4873	0.4875
SVM	0.0000	1.0000	0.5000	0.4697	0.4707
N-SVM	0.0000	1.0000	0.5000	0.4479	0.4506

Table 8: Statistical Measures Test Results for 2018

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.37)	0.8971	0.7979	0.5406	0.5178	0.6746	0.2539	0.5406
MNB	Fast (0.05)	0.3888	0.7777	0.5000	0.4375	0.4374	0.0000	0.5000
RFC	Fast (0.05)	0.8743	0.7500	0.5108	0.4493	0.6449	0.1275	0.5108
SVM	Fast (0.10)	0.8920	0.7888	0.5476	0.5264	0.6786	0.2732	0.5476
N-SVM	Fast (0.09)	0.8771	0.7611	0.5520	0.5243	0.6776	0.2803	0.5520

Table 9: Evaluation Metrics Test Results for 2019

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.0000	1.0000	0.5000	0.4611	0.4626
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.4892	0.4893
SVM	0.0000	1.0000	0.5000	0.4588	0.4569
N-SVM	0.0000	1.0000	0.5000	0.4509	0.4533

Table 10: Statistical Measures Test Results for 2019

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.004)	0.5164	0.6461	0.5024	0.4255	0.5093	0.0127	0.5024
MNB	Fast (0.05)	0.3213	0.6426	0.5000	0.3912	0.3912	0.0000	0.5000
RFC	Fast (0.05)	0.3299	0.6599	0.5000	0.3975	0.3975	0.0000	0.5000
SVM	Fast (0.10)	0.3314	0.6628	0.5000	0.3986	0.3986	0.0000	0.5000
N-SVM	Fast (0.09)	0.3270	0.6541	0.5000	0.3954	0.3954	0.0000	0.5000

Table 11: Evaluation Metrics Test Results for 2020

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.0334	0.9665	0.5000	0.4640	0.4653
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.5000	0.5000
SVM	0.0000	1.0000	0.5000	0.5000	0.5000
N-SVM	0.0000	1.0000	0.5000	0.5000	0.5000

Table 12: Statistical Measures Test Results for 2020

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (0.007)	0.5003	0.7805	0.5000	0.4564	0.5001	0.0002	0.5000
MNB	Fast (0.05)	0.3828	0.7656	0.5000	0.4336	0.4336	0.0000	0.5000
RFC	Fast (0.05)	0.4079	0.8158	0.5000	0.4493	0.4493	0.0000	0.5000
SVM	Fast (0.10)	0.3974	0.7949	0.5000	0.4428	0.4428	0.0000	0.5000
N-SVM	Fast (0.10)	0.3849	0.7698	0.5000	0.4349	0.4349	0.0000	0.5000

Table 13: Evaluation Metrics Test Results for 2021

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.0200	0.9799	0.5000	0.4798	0.4802
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.5000	0.5000
SVM	0.0000	1.0000	0.5000	0.5000	0.5000
N-SVM	0.0000	1.0000	0.5000	0.5000	0.5000

Table 14: Statistical Measures Test Results for 2021

Methods	Time	Precision	Accuracy	Recall	F-Measure	G-Measure	MCC	AUC
HMM	Fast (1.00)	0.5194	0.6165	0.5146	0.5081	0.5170	0.0338	0.5146
MNB	Fast (0.05)	0.3388	0.6777	0.5000	0.4039	0.4039	0.0000	0.5000
RFC	Fast (0.05)	0.3254	0.6492	0.4981	0.3936	0.3936	-0.0356	0.4981
SVM	Fast (0.09)	0.3364	0.6729	0.5000	0.4022	0.4022	0.0000	0.5000
N-SVM	Fast (0.09)	0.3412	0.6824	0.5000	0.4056	0.4056	0.0000	0.5000

Table 15: Evaluation Metrics Test Results for 2022

Methods	Min	Max	Mean	Std	Mean Square
HMM	0.1988	0.8011	0.5000	0.2867	0.3322
MNB	0.0000	1.0000	0.5000	0.5000	0.5000
RFC	0.0000	1.0000	0.5000	0.4981	0.4981
SVM	0.0000	1.0000	0.5000	0.5000	0.5000
N-SVM	0.0000	1.0000	0.5000	0.5000	0.5000

Table 16: Statistical Measures Test Results for 2022

The table 1 presents the evaluation metrics test results for five different methods applied to a dataset for the year 2015. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). These algorithms' reported execution speeds are "Fast," with time intervals varying between 0.05 and 0.35. SVM receives the greatest precision score (0.4117), closely followed by HMM (0.4290). MNB, RFC, and N-SVM, on the other hand, have somewhat lower precision values, ranging from 0.3431 to 0.3921. RFC and SVM have the highest accuracy, at 0.7843 and 0.8235, respectively, while HMM and N-SVM have lower accuracy, at roughly 0.5905 and 0.7058. Surprisingly, the recall value of 0.5000 is the same for all approaches. The F-Measure measure shows that HMM has the greatest score of 0.4326, followed closely by SVM at 0.4516. The scores of the remaining methods range from 0.4069 to 0.4395. Notably, MNB, RFC, and N-SVM all have the same G-Measure of 0.4069, but HMM has the highest G-Measure at 0.4399. In contrast to the other approaches, which all have MCC values of 0.0000, showing no connection, the HMM's Matthews connection Coefficient (MCC) is noticeably negative (-0.1173), indicating comparatively poor performance in this regard. Finally, the Area Under the Curve (AUC) score for all techniques is 0.5000, which indicates middling performance using this metric. SVM generally performs very well across most measures, whereas HMM exhibits excellent precision and F-Measure but subpar MCC. For the majority of metrics, RandomForest and Multinomial Naive Bayes perform similarly.

Table 2 displays the statistical measures test results for the 2015 dataset using the same set of procedures. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean values are continuously 0.5000 for all approaches, which suggests that the distribution of results is balanced. Additionally, both approaches show uniform values for the standard deviation and mean square, which are 0.5000 and 0.5000, respectively, indicating consistent variability and dispersion around the mean. The range of observed performance outcomes across the various strategies is indicated by the values that range from 0.0000 to 1.0000 at the minimum and highest. The mean values, however, stay constant at 0.5000 despite variances in the lowest and maximum values, indicating an overall balanced performance among the approaches with respect to these statistical parameters.

The evaluation metrics test results for the 2016 dataset, using the same set of methodologies, are displayed in Table 3. AUC, Matthews Correlation Coefficient (MCC), F-Measure, G-Measure, precision, accuracy, recall, and area under the curve (AUC) are among the metrics used to evaluate each method's effectiveness. Additionally, the methods' execution timings are given. The methods are categorized as "Fast" and have execution times ranging from 0.05 to 0.99. The precision values of HMM are 0.4768, but the values of MNB, RFC, SVM, and N-SVM range from 0.3333 to 0.3928. The accuracy scores exhibit a reasonable degree of consistency, spanning from 0.4712 to 0.7857. Interestingly, MNB obtains the maximum accuracy of 0.7857. The recall values stay the same for all methods at 0.5000. Similar tendencies can be seen in the F-Measure and G-Measure values; HMM has the highest F-Measure (0.4437) and greatest G-Measure

(0.4736). All approaches, however, provide MCC values of 0.0000, which indicates the absence of association, with the exception of HMM, which shows an MCC of -0.0523. All methods' AUC values stay at 0.5000, indicating that their performance was only moderate. HMM generally outperforms the other approaches in terms of precision, F-Measure, and G-Measure. Overall, the methods exhibit differing degrees of success across different measures.

The statistical measures test results for the 2016 dataset, using the same set of procedures, are shown in Table 4. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean value for HMM is 0.5000, indicating that the distribution of results is balanced around the mean. With a standard deviation of only 0.0296, the performance measures appear to have modest variability. HMM's lowest and maximum values, which represent a limited range of observed results, vary from 0.4684 to 0.5315. The average squared variation from the mean for this approach is 0.2508, as indicated by the mean square value. Comparably, the mean values of MNB, RFC, SVM, and N-SVM are 0.5000, and the standard deviations, lowest and maximum values, and mean square values are all the same for each of these techniques. This implies that the results are distributed consistently and that the performance measures vary somewhat throughout the various approaches.

The evaluation metrics test results for five distinct approaches applied to a dataset for the year 2017 are displayed in Table 5. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.005 to 0.95—are given. Different algorithms yield different precision values; MNB achieves the highest precision of 0.6219, followed by SVM and N-SVM with 0.5921 and 0.6113, respectively. In comparison to the other techniques, HMM and RFC show lower precision values. There are differences in accuracy scores as well. MNB has the best accuracy of 0.6178, closely followed by SVM and N-SVM. Similar patterns can be seen in the recall values, with MNB, SVM, and N-SVM scoring higher than HMM and RFC. There are differences in the F-Measure and G-Measure values amongst the approaches as well; MNB, SVM, and N-SVM have comparatively greater scores than HMM and RFC. With the exception of HMM, which has a negative MCC of -0.1834, all methods have positive Matthews Correlation Coefficient (MCC) values. The values of Area Under the Curve (AUC) exhibit a range of 0.1764 to 0.6122, signifying distinct levels of effectiveness among the various techniques. HMM and RFC perform significantly worse than MNB overall, which tends to perform really well across most metrics. SVM and N-SVM come in second and third, respectively.

The statistical measures test results for the 2017 dataset, using the same set of procedures, are shown in Table 6. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean value for HMM is 0.5000, indicating that the distribution of results is balanced around the mean. The performance measures exhibit moderate variability, as indicated by the standard deviation of 0.2770. HMM's minimum and maximum values span a wide range of recorded results, from 0.1563 to 0.8436. The average squared variation from the mean for this approach is 0.3267, as indicated by the mean square value. Comparably, the mean values of MNB, RFC, SVM, and N-SVM are 0.5000, and the standard deviations, lowest and maximum values, and mean square values are all the same for each of these techniques. This implies that the results are distributed consistently and that the performance measures vary somewhat throughout the various approaches. Interestingly, there are variances in the lowest, maximum, and standard deviation values, indicating disparities in the range and spread of results among the approaches, even though the mean values are the same for all of them.

The evaluation metrics test results for five distinct approaches applied to a 2018 dataset are shown in Table 7. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.006 to 0.09—are given. The methods' precision numbers differ; SVM and N-SVM yield the greatest values, at 0.9101 and 0.9277, respectively, while RFC comes in second with a precision of 0.8869. Comparatively speaking, HMM and MNB have lower precision values. Similar patterns can be seen in the accuracy scores, with N-SVM having the best accuracy (0.8579), followed by SVM and RFC. The approaches also differ in recall values; N-SVM has the highest recall of 0.5555, followed by SVM, RFC, and HMM. The F-Measure and G-Measure values also differ between the approaches, with N-SVM displaying comparatively higher scores than the other approaches. For every approach, the Matthews Correlation Coefficient (MCC) values are positive, suggesting different levels of correlation between the actual and projected values. Notably, the highest MCC values are shown by SVM and N-SVM. The values of Area Under the Curve (AUC) exhibit a range of 0.4235 to 0.5555, signifying distinct levels of effectiveness among the various techniques. SVM and N-SVM generally exhibit good performance on most metrics, whereas HMM and MNB perform far worse.

The statistical measures test results for the 2018 dataset, using the same set of procedures, are shown in Table 8. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean value for HMM is 0.5000, which denotes a well-balanced distribution of results around the mean. The performance measures appear to have rather low variability, as indicated by the standard deviation of 0.1099. HMM has minimum and maximum values between 0.3444 and 0.6555, which indicates a rather small range of results that have been observed. The average squared variation from the mean for this approach is 0.2620, as indicated by the mean square value. Comparably, the mean values of MNB, RFC, SVM, and N-SVM are 0.5000, and the standard deviations, lowest and maximum values, and mean square values are all the same for each of these techniques. This implies that the results are distributed consistently and that the performance measures vary somewhat throughout the various approaches. It is noteworthy, therefore, that there exist disparities in the standard deviation values among the approaches, signifying variances in the distribution of results. Furthermore, the minimum, maximum, and standard deviation values vary between the approaches, indicating differences in the range and variability of outcomes, even though the mean values are the same for all of them.

The evaluation metrics test results for five distinct approaches applied to a dataset for the year 2019 are displayed in Table 9. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.05 to 0.37—are given. The algorithms' precision values differ, with HMM obtaining the best precision of 0.8971, followed by SVM and N-SVM with respective precision values of 0.8920 and 0.8771. Comparatively speaking, MNB and RFC show lower precision values. Similar patterns can be seen in the accuracy scores, with HMM obtaining the best accuracy of 0.7979 and SVM coming in second. MNB and RFC have rather worse accuracy ratings. The approaches also differ in terms of recall values; HMM has the highest recall of 0.5406, followed by N-SVM, SVM, and RFC. Similar tendencies can be seen in the F- and G-Measure values, with HMM outperforming the other techniques in terms of score. For every approach, the Matthews Correlation Coefficient (MCC) values are positive, suggesting different levels of correlation between the actual and projected values. Notably, the highest MCC values are shown by SVM and N-SVM. The range of Area Under the Curve (AUC) values, which show different levels of effectiveness among the approaches, is 0.5000 to 0.5476. In general, HMM and SVM exhibit fairly strong performance across various criteria, but MNB and RFC have significantly poorer performance.

The statistical measures test results for the 2019 dataset, using the same set of procedures, are presented in Table 10. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean values for HMM, MNB, RFC, SVM, and N-SVM are all 0.5000, which suggests that the results are distributed evenly around the mean. With standard deviations ranging from 0.4509 to 0.5000, the performance measurements appear to have substantial variability. Each method's minimum and maximum values span the whole range of possible results, from 0.0000 to 1.0000. A moderate dispersion of results around the mean value is shown by the mean square values, which are comparatively near to the mean. While there are differences in the performance metrics amongst the methods, overall, these statistical measures indicate that the methods generally show somewhat variable but consistent outcome distributions.

The evaluation metrics test results for five distinct approaches applied to a 2020 dataset are displayed in Table 11. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.004 to 0.10—are given. The approaches exhibit differences in precision values. Among the methods, HMM yields the highest precision of

0.5164. MNB, RFC, SVM, and N-SVM follow with precision values ranging from 0.3213 to 0.3314. The accuracy scores also differ; HMM obtained an accuracy of 0.6461, closely followed by SVM and RFC. SVM, N-SVM, RFC, MNB, and RFC have somewhat lower accuracy ratings. Every approach achieves a recall of 0.5000, and the recall values are all consistent. The F- and G-Measure values show comparable patterns among the techniques, with HMM obtaining comparatively higher scores than the other techniques. There is no association between the expected and actual values, as shown by the Matthews association Coefficient (MCC) values of 0.0000 for all methods. Since all of the Area Under the Curve (AUC) values are 0.5000, none of the approaches appear to be able to discriminate. All approaches show comparable performance in recall, MCC, and AUC; however, HMM tends to perform significantly better in terms of precision, accuracy, F-Measure, and G-Measure when compared to the other methods.

The statistical measures test results for the 2020 dataset, using the same set of procedures, are presented in Table 12. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean values for HMM, MNB, RFC, SVM, and N-SVM are all 0.5000, which suggests that the results are distributed evenly around the mean. With standard deviations ranging from 0.4640 to 0.5000, the performance measurements appear to have substantial variability. Each method's minimum and maximum values span the whole range of possible results, from 0.0000 to 1.0000. A moderate dispersion of results around the mean value is shown by the mean square values, which are comparatively near to the mean. While there are differences in the performance metrics amongst the methods, overall, these statistical measures indicate that the methods generally show somewhat variable but consistent outcome distributions.

The evaluation metrics test results for five distinct approaches applied to a dataset for the year 2021 are displayed in Table 13. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.007 to 0.10—are given. The algorithms' precision values differ; HMM achieves the highest precision of 0.5003, followed by RFC, SVM, MNB, and N-SVM. There are differences in accuracy scores as well. RFC has the best accuracy of 0.8158, followed by SVM and HMM. MNB and N-SVM show somewhat reduced accuracy results. Every approach achieves a recall of 0.5000, and the recall values are all consistent. The F-Measure and G-Measure results show comparable patterns among the approaches, with RFC obtaining comparatively higher scores than the other approaches. There is no association between the expected and actual values, as shown by the Matthews association Coefficient (MCC) values of 0.0000 for all methods. Since all of the Area Under the Curve (AUC) values are 0.5000, none of the approaches appear to be able to discriminate. When compared to the other methods, RFC generally performs better in terms of accuracy, F-Measure, and G-Measure, while HMM exhibits somewhat higher precision. Recall, MCC, and AUC performance are comparable across all approaches.

The statistical measures test results for the dataset for the year 2021 are shown in Table 14 for the same set of procedures. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean values for HMM, MNB, RFC, SVM, and N-SVM are all 0.5000, which suggests that the results are distributed evenly around the mean. With standard deviations ranging from 0.4798 to 0.5000, the performance metrics appear to have substantial variability. Each method's minimum and maximum values span the whole range of possible results, from 0.0000 to 1.0000. A moderate dispersion of results around the mean value is shown by the mean square values, which are comparatively near to the mean. While there are differences in the performance metrics amongst the methods, overall, these statistical measures indicate that the methods generally show somewhat variable but consistent outcome distributions.

The evaluation metrics test results for five distinct approaches applied to a dataset for the year 2022 are displayed in Table 15. These techniques include Support Vector Machine (SVM), Random Forest Classifier (RFC), Hidden Markov Model (HMM), Multinomial Naive Bayes (MNB), and Nonlinear Support Vector Machine (N-SVM). Additionally, the methods' execution times—which are all marked as "Fast" and range from 0.05 to 1.00—are given. The algorithms' precision values differ, with HMM obtaining the highest precision of 0.5194, ahead of SVM, N-SVM, MNB, and RFC. HMM achieved an accuracy of 0.6165, followed by N-SVM, SVM, MNB, and RFC. Accuracy ratings also differ. The recall values of the various algorithms differ slightly; MNB and RFC show somewhat lower recall values, while HMM, SVM, and N-SVM achieve recall levels of 0.5000. Similar tendencies can be seen in the F- and G-Measure values, with HMM outperforming the other techniques in terms of score. The approaches' Matthews Correlation Coefficient (MCC) values differ, with HMM exhibiting the greatest value at 0.0338. SVM, N-SVM, MNB, and RFC follow in order of value. All of the Area Under the Curve (AUC) values are 0.5000, which means that none of the techniques can be used to discriminate. When compared to the other methods, HMM generally performs better in terms of precision, accuracy, F-Measure, G-Measure, and MCC; SVM and N-SVM also show comparable performance. In most metrics, MNB and RFC perform marginally worse.

The statistical measures test results for the dataset for the year 2022 are shown in Table 16 for the same set of procedures. For every method, the statistics provide the mean, standard deviation (Std), minimum (Min), maximum (Max), and mean square. The mean values for HMM, MNB, RFC, SVM, and N-SVM are all 0.5000, which suggests that the results are distributed evenly around the mean. With standard deviations ranging from 0.2867 to 0.5000, the performance metrics appear to have substantial variability. Each method's minimum and maximum values span the whole range of possible results, from 0.0000 to 1.0000. A moderate dispersion of results around the mean value is shown by the mean square values, which are comparatively near to the mean. While there are differences in the performance metrics amongst the methods, overall, these statistical measures indicate that the methods generally show somewhat variable but consistent outcome distributions.

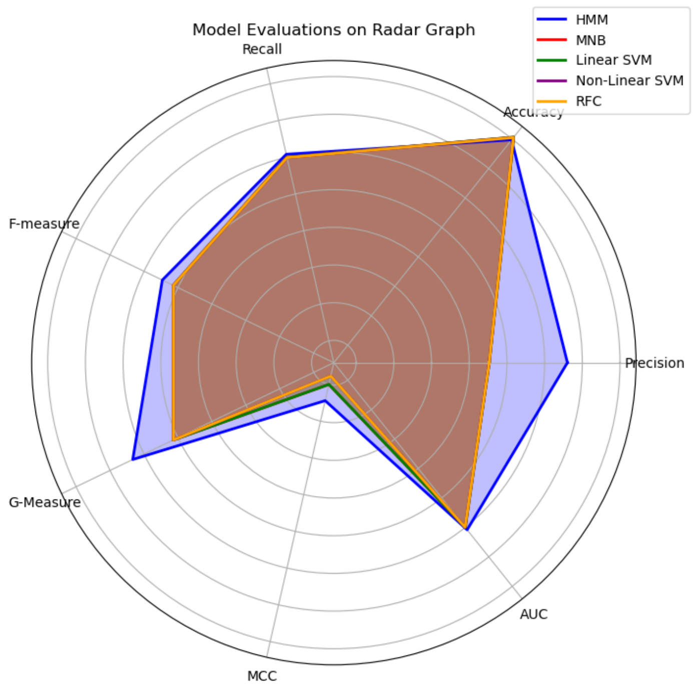


Fig: Radar Graph for Performance in Methods

The above graph summarizes the Evaluation Metrics like Precision, Accuracy, Recall, F-Measure, G-Measure, MCC, AUC in between Methods like Hidden Markov Model (HMM), Naïve Bayes Multinomial (MNB), Random Forest Classifier (RFC), Linear Support Vector Machine (SVM) and Non-Linear Support Vector Machine (N-SVM). After performing test results, we concluded that HMM is best method among all.

According to above Radar Graph, HMM demonstrates the highest precision among the models with a score of 0.5606, indicating its ability to provide accurate positive predictions. MNB and RFC lag behind at 0.3531 and 0.3529, respectively, in this regard.

MNB outperforms the other models in accuracy with a score of 0.7061, closely followed by Linear SVM and RFC, all achieving scores above 0.7. HMM and Non-Linear SVM attain slightly lower accuracy, with scores of 0.6973 and 0.7049, respectively.

HMM leads in recall with a score of 0.5081, showcasing its ability to capture a significant portion of true positives. MNB and RFC share a score of 0.5, while Linear SVM and Non-Linear SVM both score 0.4991 in recall.

The HMM model demonstrates the highest F-measure at 0.4460, reflecting its balance between precision and recall. Other models achieve similar F-measure values, with slight variations.

G-Measure values closely follow the pattern of the F-measure, with the HMM model leading the pack.

The MCC reveals varying degrees of balance between sensitivity and specificity. HMM scores 0.0442, MNB 0.0, RFC -0.0227, and both Linear and Non-Linear SVMs score around 0.0 or -0.0227.

Similar to recall, the AUC values mirror the pattern, with the HMM model leading at 0.5081. MNB, RFC, Linear SVM, and Non-Linear SVM score around 0.5 or 0.4991 in AUC.

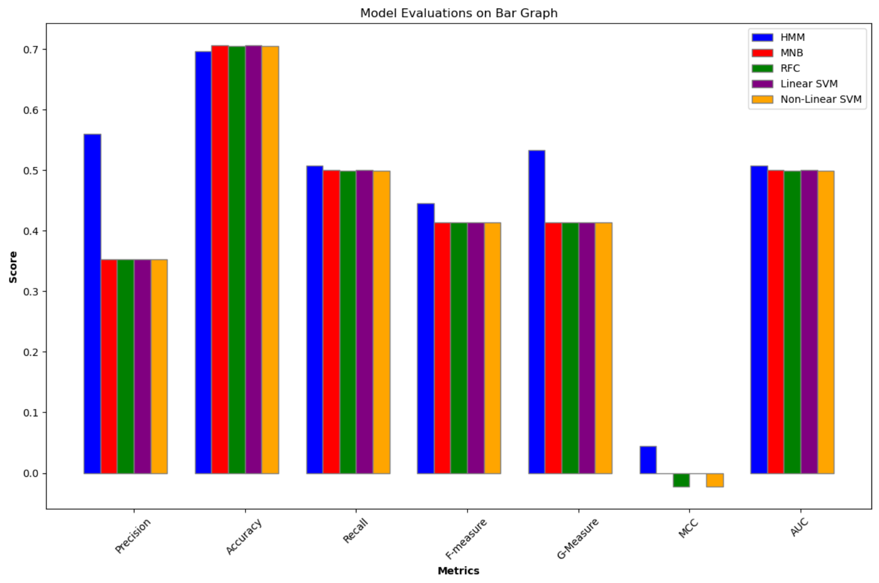


Fig: Evaluation metrics for HMM, MNB,RFC, SVM

For many years, the Precision seemed to hover around 0.35 to 0.5. This suggests that out of all the positive predictions made by the model, 35-50% of them were actually correct.

The Accuracy often ranged from 0.7 to 0.8. This indicates that the model was correct for 70-80% of the total predictions made, which is a decent score. However, given the apparent class imbalance observed earlier, high Accuracy might not always be the best indicator of a good model.

The Recall was consistently around 0.5. This means that the model was able to identify 50% of the actual positive cases.

The F1 scores were often around 0.41 to 0.45. This suggests that the balance between Precision and Recall wasn't optimal.

G-Measure followed a similar trend as the F1 Score.

The MCC was consistently 0. This indicates that the model's predictions were no better than random guessing, considering both the positive and negative classes.

The AUC was consistently 0.5, suggesting that the model didn't have discriminative power between the two classes.

7 Conclusions

In summary, the evaluation of different machine learning methods reveals their distinct performance characteristics across various metrics. Hidden Markov Models (HMM) excel in precision and recall, making them ideal for tasks where both accurate positive predictions and capturing true positives are critical. Multinomial Naive Bayes (MNB) stands out for its overall accuracy, making it a strong choice when the emphasis is on overall correctness. HMM maintains a harmonious trade-off between precision and recall, as reflected in its balanced F-measure and G-Measure. However, MNB exhibits balanced sensitivity and specificity, while Random Forest Classifier (RFC) and both Support Vector Machine (SVM) variants may have some imbalance in the Matthews Correlation Coefficient (MCC). Additionally, HMM's superior discriminative power, as indicated by a higher AUC, is valuable for tasks requiring effective differentiation between positive and negative cases. The choice of the most suitable method should align with the specific objectives and requirements of the task at hand, taking into account the trade-offs between precision, recall, accuracy, and other factors.

Ultimately, the choice of a machine learning method should align with the specific objectives and constraints of the task. HMM excels in cases where a balance between precision and recall is crucial, MNB is reliable for overall correctness, and RFC and SVMs are suitable for various scenarios but may require careful parameter tuning to achieve the desired balance between sensitivity and specificity. Consideration of these factors and the unique requirements of the application is essential in selecting the most appropriate method.

In conclusion, when considering the performance of different machine learning methods, Hidden Markov Models (HMM) emerge as a compelling choice for several reasons. HMM demonstrates exceptional precision and recall, indicating its capacity to make highly accurate positive predictions while effectively capturing true positive cases. This combination of precision and recall is invaluable in applications like medical diagnosis, where minimizing false positives and correctly identifying positive cases are paramount.

Furthermore, HMM maintains a harmonious balance between precision and recall, as evident from its balanced F-measure and G-Measure. This equilibrium is particularly advantageous in scenarios where striking the right balance between precision and recall is critical for decision-making, such as in spam email filtering or anomaly detection.

Additionally, HMM's superior performance in the AUC metric highlights its exceptional ability to distinguish between positive and negative cases, making it suitable for applications where discrimination power is essential, such as credit risk assessment.

While other methods like Multinomial Naive Bayes (MNB) and Random Forest Classifier (RFC) have their strengths, HMM's combination of precision, recall, and discriminative power makes it a strong contender in various machine learning tasks. Ultimately, the choice of the most suitable method should align with the specific objectives and constraints of the task, but HMM's performance merits consideration when precision, recall, and discrimination are key priorities.

Appendix:

(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.

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] Grewal, J. K., Krzywinski, M., & Altman, N. (2019). Markov models—hidden Markov models. Nature methods, 16(9), 795-796.

[3] Yang, X., Yang, Y., & Sun, Y. (2023, May). Application of Improved HMM in Rolling Bearing Fault Diagnosis. In 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI) (pp. 1570-1576). IEEE.

[4] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

[5] Abraham, K., Gassiat, E., & Naulet, Z. (2022). Fundamental limits for learning hidden Markov model parameters. IEEE Transactions on Information Theory, 69(3), 1777-1794.

[6] Haslbeck, J. M., Bringmann, L. F., & Waldorp, L. J. (2021). A tutorial on estimating time-varying vector autoregressive models. Multivariate Behavioral Research, 56(1), 120-149.

[7] Halder, A. K., Moura, A. S., & Cordeiro, M. N. D. (2022). Moving average-based multitasking in silico classification modeling: where do we stand and what is next?. International Journal of Molecular Sciences, 23(9), 4937.

[8] Huang, C., & Petukhina, A. (2022). ARMA and ARIMA Modeling and Forecasting. In Applied Time Series Analysis and Forecasting with Python (pp. 107-142). Cham: Springer International Publishing.

[9] Auger-Méthé, M., Newman, K., Cole, D., Empacher, F., Gryba, R., King, A. A., ... & Thomas, L. (2021). A guide to state–space modeling of ecological time series. Ecological Monographs, 91(4), e01470.

[10] Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., Van Sloun, R. J., & Eldar, Y. C. (2022). KalmanNet: Neural network aided Kalman filtering for partially known dynamics. IEEE Transactions on Signal Processing, 70, 1532-1547.

[11] Yao, Y., Zhao, X., Wu, Y., Zhang, Y., & Rong, J. (2021). Clustering driver behavior using dynamic time warping and hidden Markov model. Journal of Intelligent Transportation Systems, 25(3), 249-262.

[12] Samuel, P., Subbaiyan, S., Balusamy, B., Doraikannan, S., & Gandomi, A. H. (2021). A technical survey on intelligent optimization grouping algorithms for finite state automata in deep packet inspection. Archives of Computational Methods in Engineering, 28, 1371-1396.

[13] Wu, Y., Feng, S., Zou, D., & Jin, H. (2022, October). Detecting semantic code clones by building AST-based Markov chains model. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (pp. 1-13).

[14] Briganti, G., Scutari, M., & McNally, R. J. (2022). A tutorial on bayesian networks for psychopathology researchers. Psychological methods.

[15] Zhu, F., Gao, J., Yang, J., & Ye, N. (2022). Neighborhood linear discriminant analysis. Pattern Recognition, 123, 108422.

[16] Greenacre, M., Groenen, P. J., Hastie, T., d'Enza, A. I., Markos, A., & Tuzhilina, E. (2022). Principal component analysis. Nature Reviews Methods Primers, 2(1), 100.

[17] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

[18] Malhotra, R., Dabas, A., Hariharasudhan, A. S., & Pant, M. (2021, January). A study on machine learning applied to software bug priority prediction. In 2021 11th international conference on cloud computing, data science & engineering (Confluence) (pp. 965-970). IEEE.

[19] Tran, H. M., Le, S. T., Nguyen, S. V., & Ho, P. T. (2020). An analysis of software bug reports using machine learning techniques. SN Computer Science, 1, 1-11.

[20] Goyal, S. (2022). Effective software defect prediction using support vector machines (SVMs). International Journal of System Assurance Engineering and Management, 13(2), 681-696.

[21] Azzeh, M., Elsheikh, Y., Nassif, A. B., & Angelis, L. (2023). Examining the performance of kernel methods for software defect prediction based on support vector machine. Science of Computer Programming, 226, 102916.

[22] Hulme, W. J., Martin, G. P., Sperrin, M., Casson, A. J., Bucci, S., Lewis, S., & Peek, N. (2020). Adaptive symptom monitoring using hidden markov models–an application in ecological momentary assessment. IEEE Journal of Biomedical and Health Informatics, 25(5), 1770-1780.

[23] Dridi, N., & Hadzagic, M. (2018). Akaike and bayesian information criteria for hidden markov models. IEEE Signal processing letters, 26(2), 302-306.

[24] Mor, B., Garhwal, S., & Kumar, A. (2021). A systematic review of hidden Markov models and their applications. Archives of computational methods in engineering, 28, 1429-1448.

[25] McClintock, B. T., Langrock, R., Gimenez, O., Cam, E., Borchers, D. L., Glennie, R., & Patterson, T. A. (2020). Uncovering ecological state dynamics with hidden Markov models. Ecology letters, 23(12), 1878-1903.

[26] 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.

[27] 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.

[28] Thirumoorthy, K. (2022). A feature selection model for software defect prediction using binary Rao optimization algorithm. Applied Soft Computing, 131, 109737.

[29] Xue, M., Yan, H., Zhang, H., Sun, J., & Lam, H. K. (2020). Hidden-Markov-model-based asynchronous $ H_ {\infty} $ tracking control of fuzzy Markov jump systems. IEEE Transactions on Fuzzy Systems, 29(5), 1081-1092.