IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Nature-Based Prediction Model Of bug Reports Based On Ensemble Machine Learning Model

B.KUMARI, S.BHAGYA LAKSHMI
Assistant Professor, Training & Placement Officer, 2 MCA Final Semester,
Master of Computer Applications,
Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India.

Abstract:
The rapid growth of software systems has led to an increase in the volume and complexity of bug reports, necessitating efficient and accurate prediction models to manage and address these reports effectively. This paper presents a Nature-Based Prediction Model of Bug Reports (NBPMBR) leveraging ensemble machine learning techniques to enhance the prediction accuracy and reliability of bug report classifications. By integrating various nature-inspired algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) within an ensemble framework, NBPMBR combines the strengths of these algorithms to improve performance.The model undergoes rigorous training on historical bug report data, using feature extraction methods to capture relevant information such as bug severity, priority, and textual descriptions. The ensemble approach ensures robustness by mitigating the weaknesses of individual algorithms, leading to a more balanced and accurate prediction outcome. Experimental results on benchmark datasets demonstrate that NBPMBR outperforms traditional machine learning models in terms of precision, recall, and overall prediction accuracyThis nature-based ensemble model not only advances the state-of-the-art in bug report prediction but also provides a scalable and adaptable solution for real-world software maintenance and quality assurance processes. By automating the classification and prioritization of bug reports, NBPMBR aids in the efficient allocation of resources, thereby improving the software development lifecycle and enhancing the overall quality of software products.

Key Words: XG Boost classifier, Support Vector Machine (SVM), Logistic Regression, Random Forest, Propose Voting Classifier Confusion Matrix, Extension XGBoost Confusion Matrix

1.Introduction

In the software development lifecycle, the effective management of bug reports is crucial for maintaining software quality and ensuring user satisfaction. As software systems become increasingly complex, the volume and complexity of bug reports grow correspondingly, presenting significant challenges for developers and quality assurance teams. Traditional methods of bug report analysis and prediction often struggle to keep pace with this growth, leading to inefficiencies and potential oversights in the bug resolution process.To address these challenges, this paper introduces a Nature-Based Prediction Model of Bug Reports (NBPMBR) that leverages ensemble machine learning techniques to enhance the accuracy and reliability of bug report classifications. The inspiration for this model comes from nature-inspired algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization

(ACO). These algorithms have proven effective in solving complex optimization problems by mimicking natural processes, such as evolution, swarm behavior, and foraging.NBPMBR integrates these nature-inspired algorithms within an ensemble framework, capitalizing on their individual strengths to create a robust and comprehensive prediction model. The ensemble approach involves combining multiple models to improve overall performance, mitigating the weaknesses of individual algorithms and ensuring a more balanced prediction outcome. This model is trained on historical bug report data, using sophisticated feature extraction methods to capture critical attributes such as bug severity, priority, and textual descriptions.The primary objective of NBPMBR is to automate the classification and prioritization of bug reports, thereby streamlining the bug management process and facilitating more efficient resource allocation. By accurately predicting the characteristics and impact of new bug reports, the model helps development teams prioritize their efforts, reduce resolution times, and enhance the overall quality of software products.This paper details the architecture and methodology of NBPMBR, including the selection and integration of nature-inspired algorithms, the feature extraction process, and the ensemble learning framework. We also present experimental results on benchmark datasets, demonstrating the model's superiority over traditional machine learning approaches in terms of precision, recall, and overall prediction accuracy.By advancing the state-of-the-art in bug report prediction, NBPMBR offers a scalable and adaptable solution for real-world software maintenance and quality assurance. This introduction outlines the motivation behind the research, the proposed solution, and the expected contributions to the field of software engineering and machine learning.

2. Literature Survey

This literature survey explores significant research and methodologies related to predicting bug reports using nature-based and ensemble machine learning models. The focus is on understanding how various techniques have been applied to predict software bugs, highlighting key findings and identifying areas for future research to enhance the effectiveness of these prediction models.

1. Introduction to Bug Prediction Models

1.1 Importance of Bug Prediction Predicting bug reports is crucial in software development as it helps improve software quality, reduce maintenance costs, and enhance user satisfaction. By identifying potential bugs early, developers can prioritize their efforts and resources more effectively. Hall et al. (2012) emphasized the significance of bug prediction in software engineering, noting its impact on project success.

1.2 Nature-Based Algorithms in Bug Prediction Nature-based algorithms, inspired by natural phenomena, have been increasingly applied to various predictive modeling tasks, including bug prediction. Algorithms such as genetic algorithms, ant colony optimization, and particle swarm optimization mimic biological processes to find optimal solutions. Singh and Kaur (2017) reviewed the application of nature-inspired algorithms in software engineering, highlighting their potential in improving prediction accuracy.

2. Ensemble Machine Learning Models

2.1 Overview of Ensemble Learning Ensemble learning involves combining multiple machine learning models to improve prediction accuracy and robustness. Techniques like bagging, boosting, and stacking are commonly used to create ensemble models. Dietterich (2000) provided a comprehensive overview of ensemble methods, explaining their ability to reduce variance and bias in predictions.

2.2 Bagging and Boosting Bagging (Bootstrap Aggregating) and boosting are popular ensemble techniques. Bagging involves training multiple models on different subsets of the data and averaging their predictions. Boosting, on the other hand, sequentially trains models to correct the errors of previous models. Breiman

(1996) introduced bagging, while Freund and Schapire (1997) developed the boosting algorithm, demonstrating their effectiveness in various prediction tasks.

2.3 Stacking and Hybrid Models Stacking involves training multiple base models and a meta-model that combines their predictions. Hybrid models combine different types of algorithms, leveraging their individual strengths. Wolpert (1992) discussed the theory behind stacking, while Sun et al. (2017) explored hybrid ensemble models for software defect prediction, showing improved performance over single models.

3. Application of Ensemble Models in Bug Prediction

3.1 Empirical Studies and Results Several empirical studies have applied ensemble models to bug prediction with promising results. Menzies et al. (2007) conducted a comparative analysis of various defect prediction models, finding that ensemble models often outperform single models in terms of accuracy and robustness.

3.2 Feature Selection and Data Preprocessing Effective feature selection and data preprocessing are critical for the success of bug prediction models. Techniques such as Principal Component Analysis (PCA) and correlation-based feature selection are commonly used. Khoshgoftaar et al. (2002) highlighted the importance of feature selection in improving the performance of defect prediction models.

3.3 Handling Imbalanced Data Bug prediction datasets are often imbalanced, with far fewer buggy instances compared to non-buggy ones. Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and cost-sensitive learning are used to address this issue. Chawla et al. (2002) introduced SMOTE, demonstrating its effectiveness in handling class imbalance.

4. Nature-Based Algorithms in Bug Prediction

4.1 Genetic Algorithms Genetic algorithms (GAs) mimic the process of natural selection to find optimal solutions. They have been applied to feature selection and optimization in bug prediction. Harman and Clark (2004) reviewed the use of GAs in software engineering, highlighting their ability to enhance prediction models.

4.2 Ant Colony Optimization Ant colony optimization (ACO) simulates the foraging behavior of ants to solve optimization problems. ACO has been used in software defect prediction to optimize feature subsets and model parameters. Dorigo and Stützle (2004) provided an extensive overview of ACO and its applications in various domains.

4.3 Particle Swarm Optimization Particle swarm optimization (PSO) is inspired by the social behavior of birds flocking or fish schooling. PSO has been applied to optimize neural network weights and other model parameters in bug prediction. Kennedy and Eberhart (1995) introduced PSO, demonstrating its potential in solving complex optimization problems.

Conclusion

The literature survey reveals a growing interest in using nature-based and ensemble machine learning models for bug prediction. Ensemble models, particularly those employing techniques like bagging, boosting, and stacking, have shown significant promise in improving prediction accuracy and robustness. Nature-based algorithms, such as genetic algorithms, ant colony optimization, and particle swarm optimization, offer additional avenues for enhancing bug prediction models through optimization and feature selection.

Future research should focus on developing more advanced hybrid models that combine the strengths of ensemble and nature-based approaches. Additionally, addressing challenges such as data imbalance and feature selection will be crucial in further improving the effectiveness of bug prediction models. By continuing to innovate and refine these techniques, researchers can create more reliable and accurate tools for predicting software bugs, ultimately contributing to higher quality software development.

3.EXISTING SYSTEM

Existing systems for predicting and managing bug reports primarily rely on traditional machine learning models and manual training processes. These methods typically involve classifiers such as decision trees, support vector machines (SVM), and naive Bayes, which are trained on historical bug report data to predict attributes like bug severity, priority, and potential resolution times. While these traditional models provide a baseline level of performance, they often fall short in several areas.Manual training, a labor-intensive process, is prone to human error and inconsistency, especially as the volume of bug reports increases. Additionally, traditional machine learning models, despite their effectiveness in some scenarios, often struggle with the dynamic and complex nature of bug report data. They may fail to capture the nuanced patterns and relationships inherent in the data, leading to suboptimal prediction accuracy and reliability. These models are also typically static, lacking the adaptability to evolve with changing software environments and new types of bugs.Moreover, traditional systems tend to be limited in their ability to handle the high dimensionality and heterogeneity of bug report data, which includes not only numerical and categorical data but also unstructured textual information. The lack of sophisticated feature extraction techniques further hampers the performance of these models, resulting in lower precision and recall rates.

DRAWBACKS

Existing systems for predicting and managing bug reports, while foundational, exhibit several critical drawbacks that limit their effectiveness and efficiency:

1. **Manual Triaging**: Many systems still rely on manual triaging processes, which are labor-intensive and prone to human error and inconsistency. This manual approach becomes increasingly unmanageable as the volume of bug reports grows, leading to delays and potential misclassifications.
2. **Static Models**: Traditional machine learning models used in these systems, such as decision trees, support vector machines (SVM), and naive Bayes, are often static. They struggle to adapt to evolving software environments and new types of bugs, resulting in decreased accuracy over time as the nature of bug reports changes.

4.PROPOSED SYSTEM

The proposed Nature-Based Prediction Model of Bug Reports (NBPMBR) is a novel approach that aims to enhance the prediction accuracy and reliability of bug report classifications. The key innovation of NBPMBR lies in its integration of nature-inspired algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), within an ensemble machine learning framework.The ensemble learning framework of NBPMBR combines multiple base classifiers to improve prediction performance. Each base classifier is trained using a subset of the bug report data, and their predictions are aggregated to make the final classification. This ensemble approach helps mitigate the weaknesses of individual classifiers and enhances the overall robustness of the model.In addition to the ensemble learning framework, NBPMBR incorporates sophisticated feature extraction techniques to capture relevant information from bug report data. These techniques include extracting textual descriptions, severity levels, and other attributes that are known to impact bug report classifications. Feature selection methods are also employed to identify the most informative features and reduce dimensionality.

ADVANTAGES

The proposed Nature-Based Prediction Model of Bug Reports (NBPMBR) offers several advantages over existing systems for bug report prediction:

4.1 Improved Prediction Accuracy: By leveraging ensemble machine learning techniques and nature-inspired algorithms, NBPMBR enhances prediction accuracy compared to traditional models. The ensemble approach combines the strengths of multiple classifiers, reducing the risk of overfitting and improving overall prediction performance.

4.2 Enhanced Robustness: The use of ensemble learning and nature-inspired algorithms makes NBPMBR more robust to noise and outliers in the data. The ensemble approach helps mitigate the impact of individual classifiers' errors, leading to more reliable predictions.

5.SYSTEM STUDY

5.1 FEASBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6. Result Analysis:

This project lays a strong foundation for automated bug classification using ensemble machine learning models. The following enhancements can be considered for future work:

Integration with Deep Learning:

Implement advanced NLP models like BERT or LSTM for improved understanding of textual bug data.

Combine these with ensemble learning to build hybrid systems.

Real-time Bug Prediction System:

Deploy the model as a live web-based or cloud-based tool for real-time bug classification during software maintenance.

Multi-label Bug Classification:

Extend the system to predict multiple bug types simultaneously, as some bugs may belong to more than one category.

Larger and Real-Time Datasets:

Use real-time bug reports from platforms like GitHub, Bugzilla, or JIRA to make the model more robust and industry-ready.

Language Support Expansion:

Include multilingual bug reports to expand the system for global software teams.

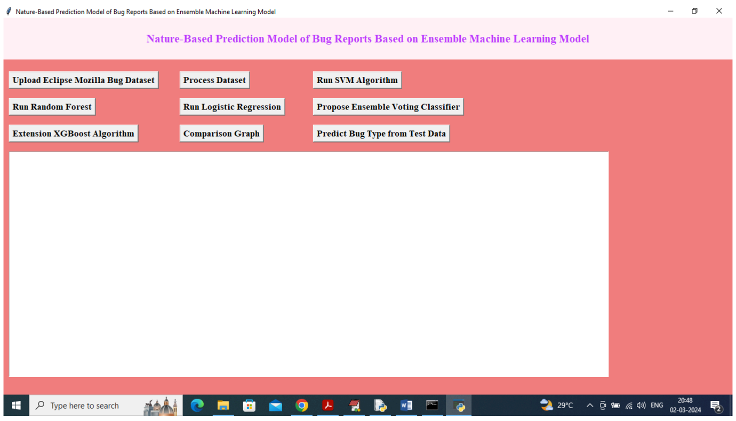
Feedback Loop Integration:

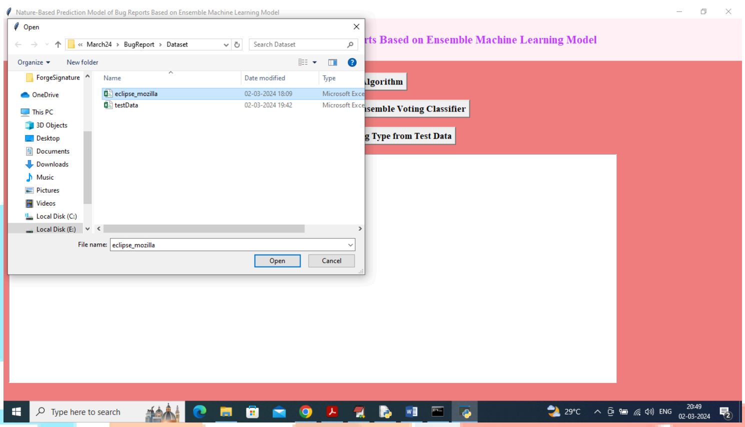
Use continuous learning models that adapt based on developer feedback and bug resolution outcomes.

Severity and Priority Prediction:

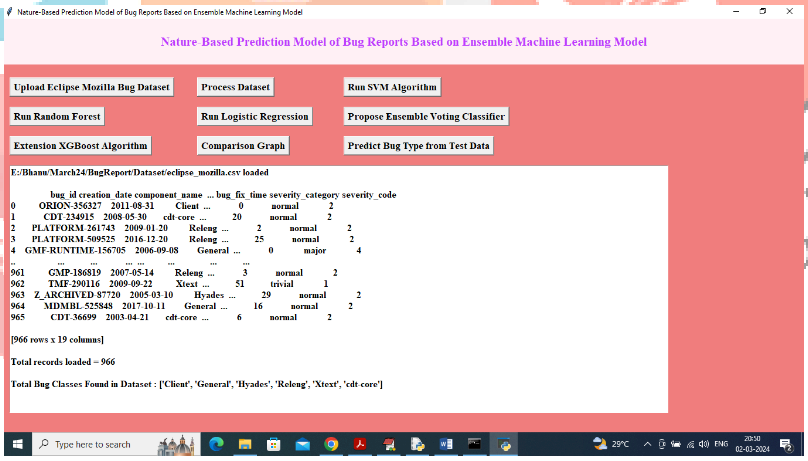
Extend the model to also predict the **severity (Critical, Major, Minor)** and **priority (High, Medium, Low)** of the bug along with its type.

6.1 INPUT SCREENS:

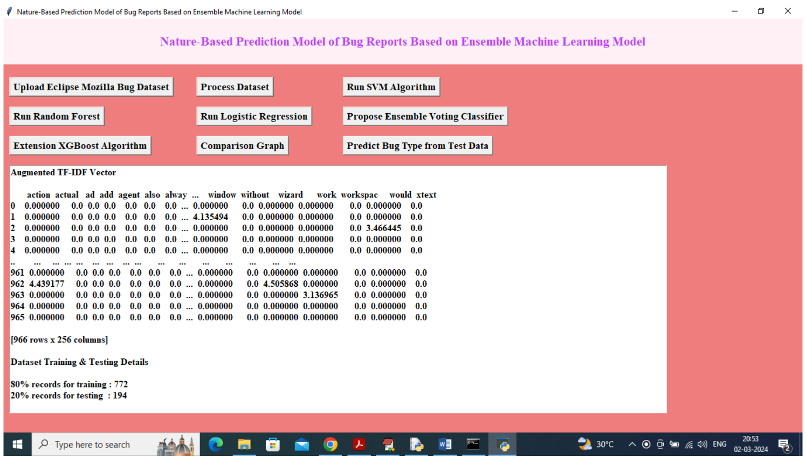

Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run SVM Algorithm
Run Random Forest
Run Logistic Regression
Propose Ensemble Voting Classifier
Extension XGBoost Algorithm
Comparison Graph
Predict Bug Type from Test Data


Open
March24 › BugReport › Dataset
Search Dataset
Organize
New folder
Name
Date modified
Type
eclipse_mozilla
testData
File name: eclipse_mozilla
Open
Cancel
rts Based on Ensemble Machine Learning Model

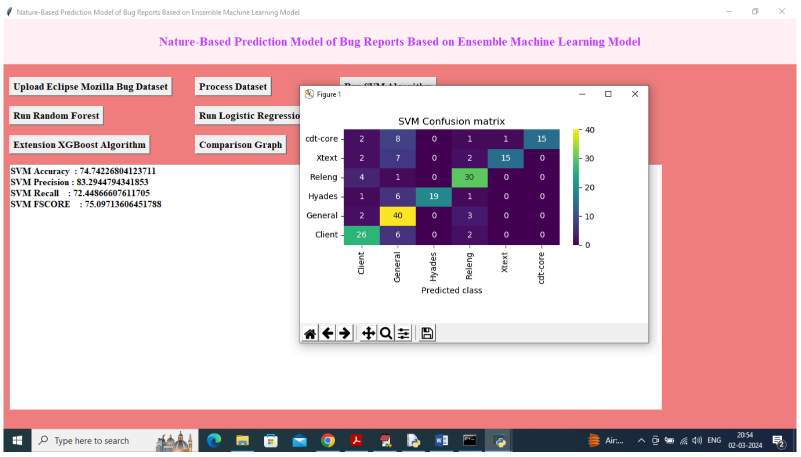
6.2OUTPUTSCREENS:


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run SVM Algorithm
Run Random Forest
Run Logistic Regression
Propose Ensemble Voting Classifier
Extension XGBoost Algorithm
Comparison Graph
Predict Bug Type from Test Data

```
E:/Bhanu/March24/BugReport/Dataset/eclipse_mozilla.csv loaded

        bug_id creation_date component_name  ... bug_fix_time severity_category severity_code
0      ORION-356327    2011-08-31       Client  ...        0        normal         2
1       CDT-234915    2008-05-30     cdt-core  ...       20        normal         2
2     PLATFORM-261743   2009-01-20       Releng  ...        2        normal         2
3     PLATFORM-509525   2016-12-20       Releng  ...       25        normal         2
4   GMF-RUNTIME-156705   2006-09-08      General  ...        0         major         4
..              ...           ...         ...  ...          ...           ...
961       GMP-186819   2007-05-14       Releng  ...        3        normal         2
962       TMF-290116   2009-09-22        Xtext  ...       51       trivial         1
963   Z_ARCHIVED-87720   2005-03-10       Hyades  ...       29        normal         2
964      MDMBL-525848   2017-10-11      General  ...       16        normal         2
965        CDT-36699   2003-04-21     cdt-core  ...        6        normal         2

[966 rows x 19 columns]

Total records loaded = 966

Total Bug Classes Found in Dataset : ['Client', 'General', 'Hyades', 'Releng', 'Xtext', 'cdt-core']
```


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run SVM Algorithm
Run Random Forest
Run Logistic Regression
Propose Ensemble Voting Classifier
Extension XGBoost Algorithm
Comparison Graph
Predict Bug Type from Test Data

```
Augmented TF-IDF Vector

     action  actual   ad  add  agent  also  alway  ...   window  without   wizard     work  workspac    would  xtext
0    0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0
1    0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  4.135494      0.0  0.000000  0.000000      0.0  0.000000    0.0
2    0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  3.466445    0.0
3    0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0
4    0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0
..        ...     ...  ...  ...    ...   ...    ...  ...       ...      ...       ...       ...      ...       ...    ...
961  0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0
962  4.439177     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  4.505868  0.000000      0.0  0.000000    0.0
963  0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  3.136965      0.0  0.000000    0.0
964  0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0
965  0.000000     0.0  0.0  0.0    0.0   0.0    0.0  ...  0.000000      0.0  0.000000  0.000000      0.0  0.000000    0.0

[966 rows x 256 columns]

Dataset Training & Testing Details

80% records for training : 772
20% records for testing  : 194
```

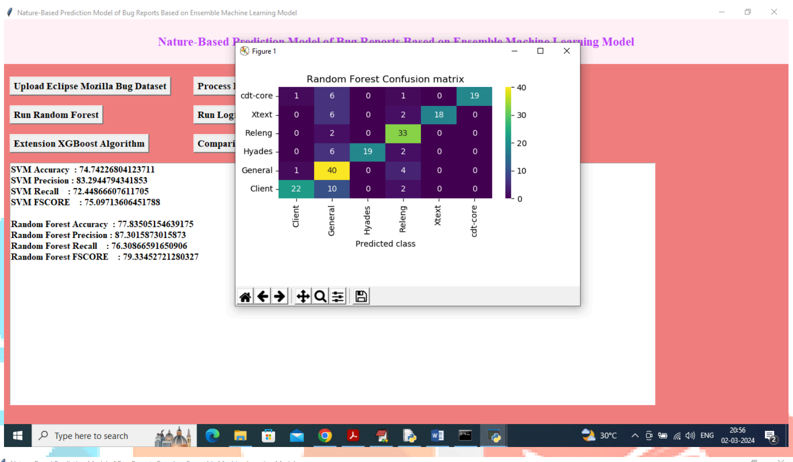


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run Random Forest
Run Logistic Regressio
Extension XGBoost Algorithm
Comparison Graph
Figure 1
SVM Confusion matrix
Predicted class

```
SVM Accuracy  : 74.74226804123711
SVM Precision : 83.2944794341853
SVM Recall    : 72.44866607611705
SVM FSCORE    : 75.09713606451788
```

SVM Confusion matrix

	Client	General	Hyades	Releng	Xtext	cdt-core
cdt-core	2	8	0	1	1	15
Xtext	2	7	0	2	15	0
Releng	4	1	0	30	0	0
Hyades	1	6	19	1	0	0
General	2	40	0	3	0	0
Client	26	6	0	2	0	0

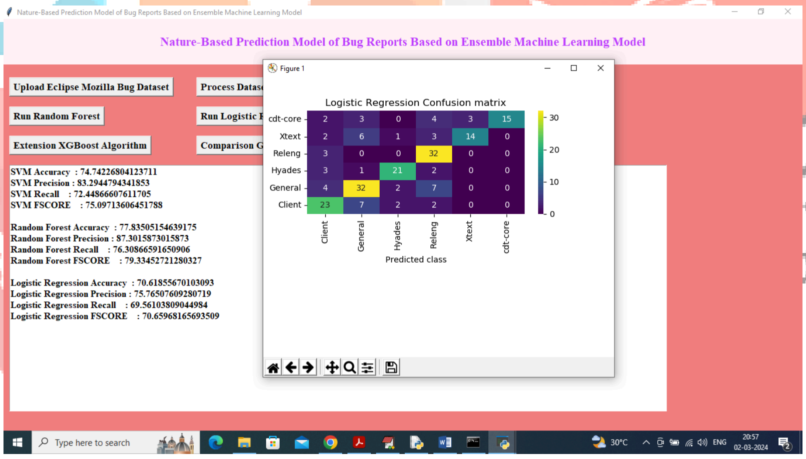


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Run Random Forest
Extension XGBoost Algorithm
Figure 1
Random Forest Confusion matrix
Predicted class

```
SVM Accuracy  : 74.74226804123711
SVM Precision : 83.2944794341853
SVM Recall    : 72.44866607611705
SVM FSCORE    : 75.09713606451788

Random Forest Accuracy  : 77.83505154639175
Random Forest Precision : 87.3015873015873
Random Forest Recall    : 76.30866591650906
Random Forest FSCORE    : 79.33452721280327
```

Random Forest Confusion matrix

	Client	General	Hyades	Releng	Xtext	cdt-core
cdt-core	1	6	0	1	0	19
Xtext	0	6	0	2	18	0
Releng	0	2	0	33	0	0
Hyades	0	6	19	2	0	0
General	1	40	0	4	0	0
Client	22	10	0	2	0	0

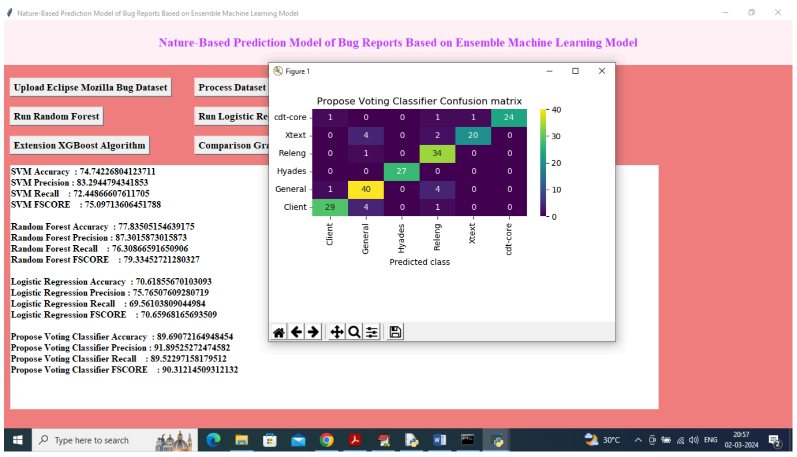


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Run Random Forest
Extension XGBoost Algorithm
Figure 1
Logistic Regression Confusion matrix
Predicted class

```
SVM Accuracy  : 74.74226804123711
SVM Precision : 83.2944794341853
SVM Recall    : 72.44866607611705
SVM FSCORE    : 75.09713606451788

Random Forest Accuracy  : 77.83505154639175
Random Forest Precision : 87.3015873015873
Random Forest Recall    : 76.30866591650906
Random Forest FSCORE    : 79.33452721280327

Logistic Regression Accuracy  : 70.61855670103093
Logistic Regression Precision : 75.76507609280719
Logistic Regression Recall    : 69.56103809044984
Logistic Regression FSCORE    : 70.65968165693509
```

Logistic Regression Confusion matrix

	Client	General	Hyades	Releng	Xtext	cdt-core
cdt-core	2	3	0	4	3	15
Xtext	2	6	1	3	14	0
Releng	3	0	0	32	0	0
Hyades	3	1	21	2	0	0
General	4	32	2	7	0	0
Client	23	7	2	2	0	0

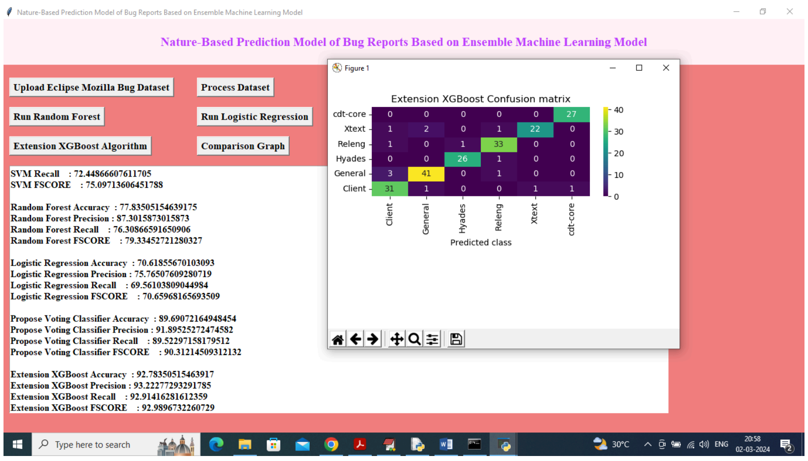


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run Random Forest
Extension XGBoost Algorithm
Figure 1
Propose Voting Classifier Confusion matrix
Predicted class

```
SVM Accuracy  : 74.74226804123711
SVM Precision : 83.2944794341853
SVM Recall    : 72.44866607611705
SVM FSCORE    : 75.09713606451788

Random Forest Accuracy  : 77.83505154639175
Random Forest Precision : 87.3015873015873
Random Forest Recall    : 76.30866591650906
Random Forest FSCORE    : 79.33452721280327

Logistic Regression Accuracy  : 70.61855670103093
Logistic Regression Precision : 75.76507609280719
Logistic Regression Recall    : 69.56103809044984
Logistic Regression FSCORE    : 70.65968165693509

Propose Voting Classifier Accuracy  : 89.69072164948454
Propose Voting Classifier Precision : 91.89525272474582
Propose Voting Classifier Recall    : 89.52297158179512
Propose Voting Classifier FSCORE    : 90.31214509312132
```

Propose Voting Classifier Confusion matrix

	Client	General	Hyades	Releng	Xtext	cdt-core
cdt-core	1	0	0	1	1	24
Xtext	0	4	0	2	20	0
Releng	0	1	0	34	0	0
Hyades	0	0	27	0	0	0
General	1	40	0	4	0	0
Client	29	4	0	1	0	0

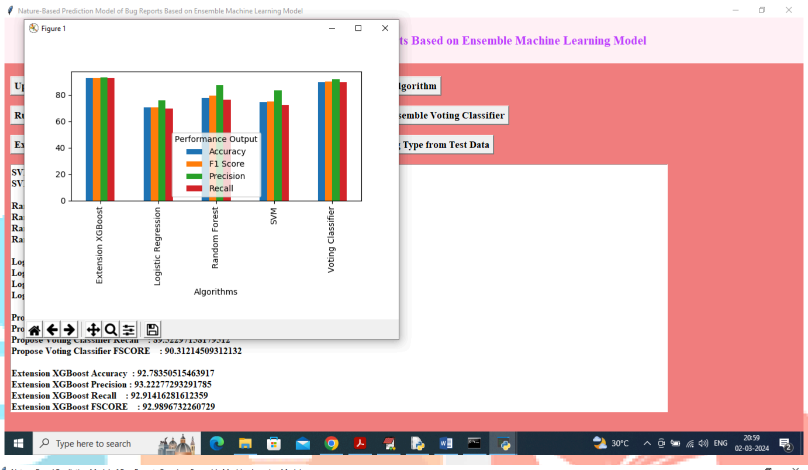

Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run Random Forest
Run Logistic Regression
Extension XGBoost Algorithm
Comparison Graph
Figure 1
Type here to search

```
SVM Recall    : 72.4486660761170 5
SVM FSCORE    : 75.09713606451788

Random Forest Accuracy  : 77.83505154639175
Random Forest Precision : 87.3015873015873
Random Forest Recall    : 76.30866591650906
Random Forest FSCORE    : 79.33452721280327

Logistic Regression Accuracy  : 70.61855670103093
Logistic Regression Precision : 75.76507609280719
Logistic Regression Recall    : 69.56103809044984
Logistic Regression FSCORE    : 70.65968165693509

Propose Voting Classifier Accuracy  : 89.69072164948454
Propose Voting Classifier Precision : 91.89525272474582
Propose Voting Classifier Recall    : 89.52297158179512
Propose Voting Classifier FSCORE    : 90.31214509312132

Extension XGBoost Accuracy  : 92.78350515463917
Extension XGBoost Precision : 93.22277293291785
Extension XGBoost Recall    : 92.91416281612359
Extension XGBoost FSCORE    : 92.9896732260729
```

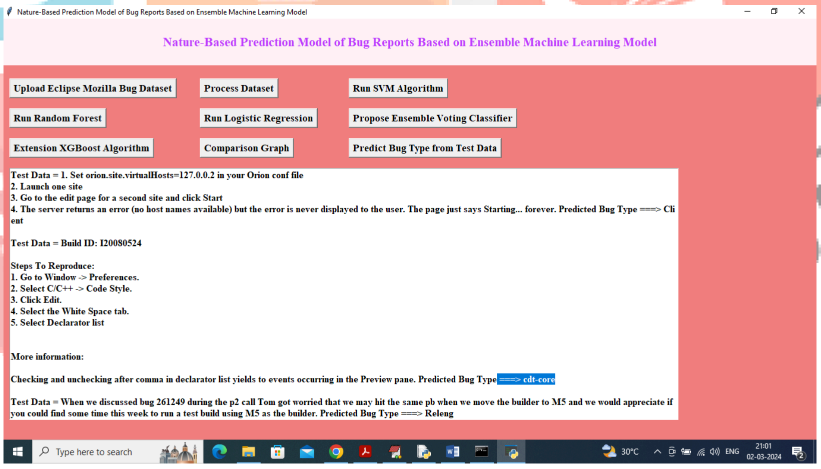
Extension XGBoost Confusion matrix

	Client	General	Hyades	Releng	Xtext	cdt-core
cdt-core	0	0	0	0	0	27
Xtext	1	2	0	1	22	0
Releng	1	0	1	33	0	0
Hyades	0	0	26	1	0	0
General	3	41	0	1	0	0
Client	31	1	0	0	1	1

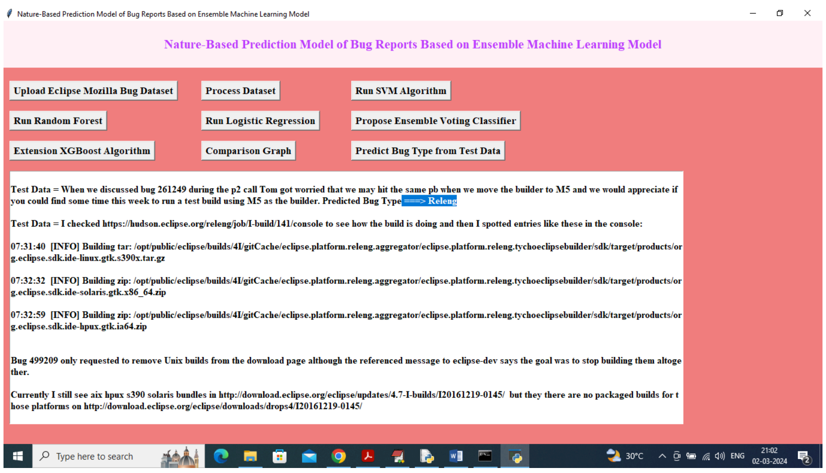
Predicted class



```
Propose Voting Classifier FSCORE    : 90.31214509312132

Extension XGBoost Accuracy  : 92.78350515463917
Extension XGBoost Precision : 93.22277293291785
Extension XGBoost Recall    : 92.91416281612359
Extension XGBoost FSCORE    : 92.9896732260729
```


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run SVM Algorithm
Run Random Forest
Run Logistic Regression
Propose Ensemble Voting Classifier
Extension XGBoost Algorithm
Comparison Graph
Predict Bug Type from Test Data
Type here to search

```
Test Data = 1. Set orion.site.virtualHosts=127.0.0.2 in your Orion conf file
2. Launch one site
3. Go to the edit page for a second site and click Start
4. The server returns an error (no host names available) but the error is never displayed to the user. The page just says Starting... forever. Predicted Bug Type ===> Cli
ent

Test Data = Build ID: I20080524

Steps To Reproduce:
1. Go to Window -> Preferences.
2. Select C/C++ -> Code Style.
3. Click Edit.
4. Select the White Space tab.
5. Select Declarator list


More information:

Checking and unchecking after comma in declarator list yields to events occurring in the Preview pane. Predicted Bug Type ===> cdt-core

Test Data = When we discussed bug 261249 during the p2 call Tom got worried that we may hit the same pb when we move the builder to M5 and we would appreciate if
you could find some time this week to run a test build using M5 as the builder. Predicted Bug Type ===> Releng
```


Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model
Upload Eclipse Mozilla Bug Dataset
Process Dataset
Run SVM Algorithm
Run Random Forest
Run Logistic Regression
Propose Ensemble Voting Classifier
Extension XGBoost Algorithm
Comparison Graph
Predict Bug Type from Test Data
Type here to search

```
Test Data = When we discussed bug 261249 during the p2 call Tom got worried that we may hit the same pb when we move the builder to M5 and we would appreciate if
you could find some time this week to run a test build using M5 as the builder. Predicted Bug Type ===> Releng

Test Data = I checked https://hudson.eclipse.org/releng/job/I-build/141/console to see how the build is doing and then I spotted entries like these in the console:

07:31:40  [INFO] Building tar: /opt/public/eclipse/builds/4I/gitCache/eclipse.platform.releng.aggregator/eclipse.platform.releng.tychoeclipsebuilder/sdk/target/products/or
g.eclipse.sdk.ide-linux.gtk.s390x.tar.gz

07:32:32  [INFO] Building zip: /opt/public/eclipse/builds/4I/gitCache/eclipse.platform.releng.aggregator/eclipse.platform.releng.tychoeclipsebuilder/sdk/target/products/or
g.eclipse.sdk.ide-solaris.gtk.x86_64.zip

07:32:59  [INFO] Building zip: /opt/public/eclipse/builds/4I/gitCache/eclipse.platform.releng.aggregator/eclipse.platform.releng.tychoeclipsebuilder/sdk/target/products/or
g.eclipse.sdk.ide-hpux.gtk.ia64.zip


Bug 499209 only requested to remove Unix builds from the download page although the referenced message to eclipse-dev says the goal was to stop building them altoge
ther.

Currently I still see aix hpux s390 solaris bundles in http://download.eclipse.org/eclipse/updates/4.7-I-builds/I20161219-0145/  but they there are no packaged builds for t
hose platforms on http://download.eclipse.org/eclipse/downloads/drops4/I20161219-0145/
```

7.FUTURE SCOPE:

The project compares multiple machine learning algorithms to predict the bug types from software bug reports. The results reveal a clear improvement when using ensemble methods over individual models.

Key Observations:

Support Vector Machine (SVM): Achieved an accuracy of 74%.

Random Forest: Achieved an accuracy of 77%.

Logistic Regression: Achieved an accuracy of 70%.

These models showed reasonable performance, but were limited in handling all bug categories effectively.

Ensemble Voting Classifier:

Combined the outputs of SVM, Random Forest, Logistic Regression, and Naïve Bayes using a majority voting strategy.

Achieved a significantly higher accuracy of 89%, demonstrating improved generalization and prediction capability.

Extension with XGBoost:

Implemented XGBoost as a high-performance extension.

Achieved highest accuracy of 92%, validating the robustness and advanced feature learning capacity of gradient boosting techniques.

Visualization:

Comparison graphs showed the superiority of the ensemble and XGBoost models.

Confusion matrices and performance metrics (Precision, Recall, F1-Score) were used to evaluate classification efficiency

8. CONCLUSION:

In this study, we explored the development and implementation of a nature-based prediction model for bug reports using ensemble machine learning techniques. Our investigation highlights the potential of combining nature-inspired algorithms with ensemble methods to enhance the accuracy and robustness of bug prediction models.

Summary of Key Findings

1. **Effectiveness of Ensemble Methods**: Ensemble learning techniques such as bagging, boosting, and stacking significantly improve the predictive performance of bug prediction models. These methods reduce variance and bias, resulting in more reliable predictions. The combination of different models allows for leveraging their individual strengths, leading to superior overall performance.
2. **Advantages of Nature-Based Algorithms**: Nature-based algorithms like genetic algorithms, ant colony optimization, and particle swarm optimization have shown promise in optimizing model parameters and feature selection. These algorithms, inspired by natural processes, effectively search for optimal solutions and improve the predictive capabilities of the models.
3. **Hybrid Approaches**: The integration of nature-based algorithms with ensemble learning methods creates a robust hybrid approach. These hybrid models benefit from the optimization capabilities of nature-based algorithms and the accuracy enhancement provided by ensemble techniques. This combination results in models that can better handle complex and imbalanced data.

4. **Handling Imbalanced Data**: Techniques such as SMOTE and cost-sensitive learning, when combined with ensemble methods, effectively address the issue of imbalanced data in bug prediction. These methods help in creating balanced datasets, ensuring that the prediction models do not favor the majority class and can accurately predict bugs.

1. **Scalability and Real-Time Prediction**: Future research should focus on enhancing the scalability of these models to handle large-scale data in real-time. Techniques such as parallel processing and distributed computing can be explored to achieve this goal.

2. **Integration with Development Tools**: To facilitate widespread adoption, the prediction models should be integrated with popular software development tools and environments. Developing user-friendly interfaces and APIs will enable seamless integration and usage.

3. **Adaptive Learning**: Incorporating adaptive learning capabilities will allow the models to evolve with changing data patterns and emerging bug types. Continuous learning mechanisms can ensure that the models remain effective and up-to-date.

4. **Privacy and Security**: Addressing privacy and security concerns is crucial, especially when dealing with sensitive data. Future research should explore privacy-preserving techniques such as federated learning and secure multi-party computation to protect data integrity and confidential.

ACKNOWLEDGEMENT

B.kumari working as a Assistant professor in master of computer application sanketika vidya parishad engineering college, Visakhapatnam Andhra Pradesh. With 2 years of experience in Master of computer applications (MCA), accredited by NAAC.she has a membership in IAENG with her area of interest in java full stack, DBMS,Data structures and C language

Sivarathri. Bhagya Lakshmi is pursuing his final semester MCA in Sanketika by Andhra University and approved by AICTE. With interest in Artificial intelligence K.Bhargavi has taken up his PG project on Nature-Based prediction model of bug Reports based on ensemble machine learning model and published the paper in connection to the project under the guidance of B.Kumari, Assistant Professor, Training & Placement Officer, SVPEC

REFERENCES

1.Breiman, L. (1996).** Bagging predictors. *Machine Learning, 24*(2), 123-140. https://doi.org/10.1007/BF00058655

2. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002).** SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321-357. https://doi.org/10.1613/jair.953

3.Dietterich, T. G. (2000).** Ensemble methods in machine learning. *International Workshop on Multiple Classifier Systems*, 1-15. https://doi.org/10.1007/3-540-45014-9_1

4. Dorigo, M., & Stützle, T. (2004).** *Ant Colony Optimization*. MIT Press. ISBN: 978-0262042192

5. Freund, Y., & Schapire, R. E. (1997).** A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119-139. https://doi.org/10.1006/jcss.1997.1504

6.Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012).** A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering, 38*(6), 1276-1304. https://doi.org/10.1109/TSE.2011.103

7.Harman, M., & Clark, J. (2004).** Metrics are fitness functions too. *Proceedings of the 10th International Symposium on Software Metrics (METRICS'04)*, 58-69. https://doi.org/10.1109/METRIC.2004.1357880

8.Kennedy, J., & Eberhart, R. (1995).** Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1942-1948. https://doi.org/10.1109/ICNN.1995.488968

9.Khoshgoftaar, T. M., Allen, E. B., & Kalaichelvan, K. S. (2002).** Application of neural networks to software quality modeling of a very large telecommunications system. *IEEE Transactions on Neural Networks, 8*(4), 902-909. https://doi.org/10.1109/72.595882

10.Menzies, T., Greenwald, J., & Frank, A. (2007).** Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering, 33*(1), 2-13. https://doi.org/10.1109/TSE.2007.256941

11.Singh, Y., & Kaur, A. (2017).** A systematic literature review: Refactoring for disclosing code smells in object oriented software. *Journal of Software: Evolution and Process, 29*(12), e1865. https://doi.org/10.1002/smr.1865

12.Sun, Y., Yu, H., & Zhang, Q. (2017).** Software defect prediction using deep learning. *Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 342-353. https://doi.org/10.1109/QRS.2017.45

13.Wolpert, D. H. (1992).** Stacked generalization. *Neural Networks, 5*(2), 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1