**Software Defect Prediction Using Fully Homomorphic Encryption With Random Forest Classifier**

Ayushmaan Pandey (Mtech Scholar), Jagdeep Kaur (Assistant Professor)

Department of Computer Science and Engineering,

Dr. B. R. Ambedkar National Institute of Technology, Jalandhar, Punjab, India

ayushmaanpandey10@gmail.com, kaurj@nitj.ac.in

**Abstract:** Software defect prediction (SDP) has been a prominent area of research in the past few years. Many techniques, such as PBSA, FILTER, LTSA, etc., have been introduced to increase the performance of SDP models. These techniques are being employed to solve various issues such as class imbalance, feature selection, reducing irrelevant and redundant features, etc. Another major issue in the field of SDP is data security. Machine learning techniques that protect privacy are used to analyze data and make predictions while preserving security and privacy. In this paper, we have put forth a safe technique for software defect prediction that combines fully homomorphic encryption with borderline-SMOTE and a random forest classifier. There were several performance metrics used, including accuracy, AUC, precision, recall, and F1-measure. The results were compared to the state-of-the-art method. The suggested method has an accuracy gain of 8–10% and a negligible accuracy loss.

**Keywords –** software defect prediction, borderline-SMOTE, fully homomorphic encryption, class imbalance

**1. Introduction**

SDP is the process of identifying potential defects or bugs in software before it is released. To determine the likelihood of future defects, it involves assessing various software features, such as code complexity, coding standards, and defect data from the past. According to a NIST report, software flaws can result in a yearly economic loss of up to $60 billion [1]. A minor data conversion bug led to the loss of the $125 million NASA Mars Climate Orbiter (MCO) spacecraft in orbit [2]. These flaws, which went unnoticed during the development process, have already had a disastrous effect on the software sector. During the software's operational phases, they developed flaws.

By identifying potential flaws early on in the SDLC, defect prediction can help organizations save time and resources. By focusing on areas of software code that are more likely to have defects, developers and testers can prioritize their testing efforts and fix issues before they become more costly and time-consuming to address. By lowering the quantity of bugs that end up in the finished product, defect prediction can also help to raise customer satisfaction and loyalty while enhancing the overall quality of software.

There are some major hurdles on the way to an efficient SDP. One of the major challenges is the issue of class imbalance (CI). It is a frequent problem in SDP, where the proportion of instances of one class, such as defective software, is much lower than that of another, such as non-defective software [3, 4]. As a result, the prediction model may have issues because it is more likely to correctly identify members of the majority class [5].

Class imbalance has been a popular topic of research in the field of SDP. Class imbalance in SDP is present due to the rarity of software defects, the unequal distribution of flaws across various software components, or biases in data collection [6]. Moreover, class size can prove insignificant for some metrics [7]. To overcome this issue, various solutions based on the concepts of oversampling minority classes, undersampling majority classes, using ensemble learning methods, etc. are explored. Previously, random oversampling, random undersampling, SMOTE, etc. have been used. Random oversampling can cause overfitting and redundancy in the dataset; on the other hand, random undersampling can drastically reduce the size of the dataset and cause the loss of important data.

Further, data security is critical in machine learning (ML), as it involves the processing and analysis of large amounts of sensitive data, such as defect density, developer’s mail, model topology, weight parameters, etc. Training an ML model for software defect prediction requires working with sensitive data related to the software of a company. Data pertaining to software contains crucial information that must be secured from adversaries and malicious users. Adversarial attacks such as "model poisoning" can compromise the integrity of ML models by injecting malicious data into the training dataset.

Ensuring data security can prevent such attacks and maintain the reliability and accuracy of ML models. In the previous work, partial homomorphic encryption (PHE) has been used with logistic regression to provide a secure method to perform SDP [24]. Since operations are limited (only one out of addition or multiplication), this may be vulnerable to privacy attacks, such as a chosen plaintext attack, which will lead to the leakage of sensitive data.

In this study, we have suggested a safe technique for software defect prediction that makes use of fully homomorphic encryption and a random forest classifier. The Borderline-SMOTE algorithm is utilized in the suggested strategy to address the problem of class imbalance. The FHE is based on the learning with errors (LWE) cryptography technique to provide security. FHE is used with the random forest classifier using the concrete-ml module that provides homomorphic machine learning solutions [8]. We have utilized the MORPH dataset for experimentation purposes [9].

The remaining sections of the paper are as follows: Section 2: Background Study, Section 3: Literature Review, Section 4: Proposed Method, Section 5: Experimental Setup, Section 6: Result Analysis, Section 7: Threats to Validity, and Section 8: Conclusion

**2. Background Study**

In this section, the various technologies used in the proposed technique are discussed. We have used borderline-SMOTE to solve the challenge of CI. For the purpose of data security, a learning with errors based FHE scheme is used to encrypt data and perform operations in an encrypted domain. The classification of data instances is performed with the help of random forest classifiers. The above-mentioned techniques are discussed in detail below.

**2.1 Fully Homomorphic Encryption**

Homomorphic encryption is the transformation of plaintext into ciphertext in such a way that the ciphertext can be worked with as if it were the plaintext. Homomorphic encryption enables complex mathematical operations on the encrypted data itself without converting it to the original data or without the need to decrypt it.

One method for achieving privacy-preserving machine learning that can guarantee the confidentiality and integrity of ML models is FHE [10]. FHE is a type of encryption that dispenses with the need to first decrypt the data and allows computations to be done directly on the ciphertext. In other words, FHE makes it possible to compute any function on encrypted data without exposing the original plaintext. FHE can provide stronger security guarantees than PHE. In PHE, certain operations may leak information about the plaintext data, which can make the encryption vulnerable to attacks. However, FHE is designed to prevent such leakage, making it more secure.

**2.2 Borderline-SMOTE**

A variation of the SMOTE algorithm known as "borderline SMOTE" concentrates on examples that are close to the line of demarcation between the minority and majority classes. Specifically, the Borderline SMOTE algorithm first identifies the examples that are located in close proximity to points that are of majority class. The points that have only majority class neighbors are considered “noise points” and the points that have both majority and minority class neighbors are considered “border points”. These examples are then used to generate synthetic examples in the same way as the original SMOTE algorithm. First, a border point is identified, and then the k-nearest neighbors are identified. Synthetic points are plotted along a line drawn between the border point and the k-nearest neighbors after the k-nearest neighbors have been chosen at random. However, rather than producing synthetic examples for all examples in the minority class, only these "borderline" examples are considered. The algorithm can produce synthetic examples that are more likely to be helpful in enhancing the performance of ML algorithms on unbalanced datasets by concentrating on the borderline examples.

**2.3 Random Forest Classifier**

A random forest classifier (RFC) is made up of many decision trees working in ensembles. In a decision tree, at each node, a decision or a split is made based on a feature in a dataset. Based on the result of the split on a particular node, a path is chosen that leads to a particular classification of an instance. In a random forest classifier, each decision tree makes a prediction about the class of an instance, and the class with the most votes becomes the decision of the model.

The fundamental idea behind RFC is that an ensemble model composed of many relatively uncorrelated decision trees will perform better than a single decision tree. Low correlation among trees will lead them to the correct decision. If some trees make the wrong decision while others make the right decision, the model as a whole will move towards the correct decision.

**3. Literature Review**

In this section, some of the previous techniques have been discussed and reviewed, which form the basis for this research. We studied numerous papers and proposed techniques to provide solutions for solving class imbalance issues such as PBSA, FILTER, LTSA, etc. Our literature review also covered the latest steps taken to provide security in SDP by applying homomorphic encryption, differential privacy, etc.

**3.1 Class imbalance**

A random forest classifier and LIME were used to create an interpretable just-in-time SDP model. By combining these methods, the author removed the unnecessary features, reducing the overall effort required for making predictions about classes [12, 35]. A novel filtering technique known as FILTER was proposed and used with SVM classifiers to tackle the problem of class imbalance [13].

A prediction-based sampling adjustment (PBSA) mechanism was developed to provide stability and reliability to SDP models. The sampling rate is adjusted with each epoch, and predictions are more stable [14]. The nested stacking of classifiers was used to improve the SDP models. Ensemble classifiers were used to provide better accuracy and predictive performance [15]. More than one layer of classifiers were stacked over each other to construct a nested stacking classifier. The initial layer consisted of boosting algorithms such as CatBoost and Adaboost, and then it was followed by a random forest, and finally a logistic regression classifier made the final decision.

Dimensionality reduction can be used to solve the problem of redundant and irrelevant features, but data loss caused by poor attributes can seriously reduce the model's precision. In this study, the local tangent space alignment technique was used for feature selection, and SVM was used for prediction. The LTSA learning method is extremely trustworthy and only requires a few parameters [16]. The popular SVM classifier is effective at making predictions and can handle data with higher dimensions. The accuracy and F-measure were improved by this method by 1–4%.

The authors combined sampling and instance recovery methods to produce an ensemble classifier. These three will create copies or fake examples of the problematic class and lessen the ratio of class inequality. Performance will eventually improve in this way. The sample will then be categorized using each learning technique, and the results will either be averaged or determined by applying a voting strategy [17].

A brand-new conditional domain adversarial adaptation (CDAA) is put forth in this work to deal with the heterogeneous problem in SDP [18]. The generator, discriminator, and classifier networks are the three underlying networks, or layers, that make up the CDAA network. Issues with feature adoption, data adoption, and class imbalance are resolved using TCLM [19]. The best feature subset for the source project is selected using cost information-based feature selection during the feature selection stage.

The authors investigated the application of resampling techniques, a popular approach for handling unbalanced data, to interpretable models. The authors investigate whether interpretable defect prediction models can be created using the original data [20]. They will also recommend a model with interpretable rules that can deal with unbalanced data right away.

When a module is predicted to be non-defective in a cost-effective problem, the penalty is higher than when a module is predicted to be defective. They suggest a hybrid approach to address problems with defect data that is cost-effective. The bagging technique was used with the artificial neurofuzzy inference system as the base classifier to address the cost-effective problem [21]. Additionally, they used the artificial neurofuzzy inference system and principle component analysis to address problems with class imbalance and high dimensionality. The study's goal is to provide background information on class imbalance issues, strategies for handling disagreements, and difficulties encountered when analyzing unbalanced data. When used in conjunction with various data-level strategies, the ensemble technique offers better results in support of experimental findings supported by one dataset. The combination of techniques will be helpful for many real-world applications, including intrusion detection, disease diagnosis, software defect prediction, etc. [22, 38]. In the CRFS process, irrelevant and collinear features are eliminated, differentially private features are selected, the best feature set is obtained, and features that can lessen dataset correlation are optimized [23].

| **Summary:** One of the major impediments to the predictive performance of SDP models is class imbalance. Various sampling and filtering techniques have been developed to counter the issue of class imbalance. PBSA and LTSA provide adjustments in sampling rates; CR-FS does correlation reduction; oversampling techniques, etc. are recently used methods. Techniques such as FILTER, PBSA, CDAA, LTSA, CRFS, etc. are some of the methods discussed above to sample data in such a way that there is a balanced ratio between the classes. |
| --- |

**3.2 Data Security**

The Paillier homomorphic encryption approximation algorithm serves as the foundation for the HOPE approach to logistic regression. The client transmits encrypted information to the server using PHE. Based on this information, the server determines the weighted matrix using the HOPE method, and an encrypted weighted matrix is then sent back to the client [24]. After the weighted matrix is decrypted at the client end, we can use logistic regression to create a prediction model for software defects. Three control groups are used in the experiment: Group 1, which deals with unencrypted data at every stage; Group 2, which deals with encrypted data sent by the client but handled as unencrypted data by the server; and Group 3, which deals with encrypted data that the server is aware of and processes using the HOPE method. The experiments were performed on the MORPH open source real-world dataset.

In this study, a differential privacy method was used for SDP model sharing. A new technique known as A-DPRF was introduced because differential privacy can be very effective even when the privacy budget is carefully chosen [25]. The FTLKD approach's methodology consists of three parts: private model construction, private data encryption, and communication [26, 27]. In the first step, encrypt sensitive data. Using Shamir sharing technology, data was homomorphically encrypted, and during subsequent processes, the data was only kept in its encrypted state. All operations are performed directly on the encrypted or cipher text data. Then, through model transfer learning and fine-tuning, train a CNN model with public data during the private model construction stage. After that, a private model is given the parameters that were obtained in this stage.

The steps in the FRLGC approach described in this work must be completed in the order required by federal learning [28]. In order to reduce dimensionality, data preprocessing uses principal component analysis (PCA). Preprocessing fixes problems with redundant data, enabling feature sharing between source projects. Then, utilizing competing DQN data, each client is trained locally. Then, for improved privacy and security, Gaussian DP is applied to each local model's parameters. The models are then combined by selecting a small number of clients and using K-means to establish their clusters.

Encryption schemes proposed over KNN are complex, and they need to use multiple encryption schemes and have frequent interactions. FKNN is proposed to solve this problem [29]. FKNN has four parts, which are as follows: 1. key generation algorithm (GENe), 2. encryption, 3. cipher text operation, and 4. decryption. There are two major subproblems in KNN: nearest neighbor selection and classification. FKNN combines FHE and KNN to perform classification over encrypted data directly for efficient and secure results.

PPML can be accomplished, for example, by combining fully homomorphic encryption (FHE) with machine learning algorithms [30, 36, 37]. One thing that needs to be controlled while performing FHE is noise in ciphertext. In this paper, a CKKS-based logistic regression scheme that protects privacy has been put forth. The leveled FHE algorithm CKKS supports the encryption of both complex and floating point numbers [31].

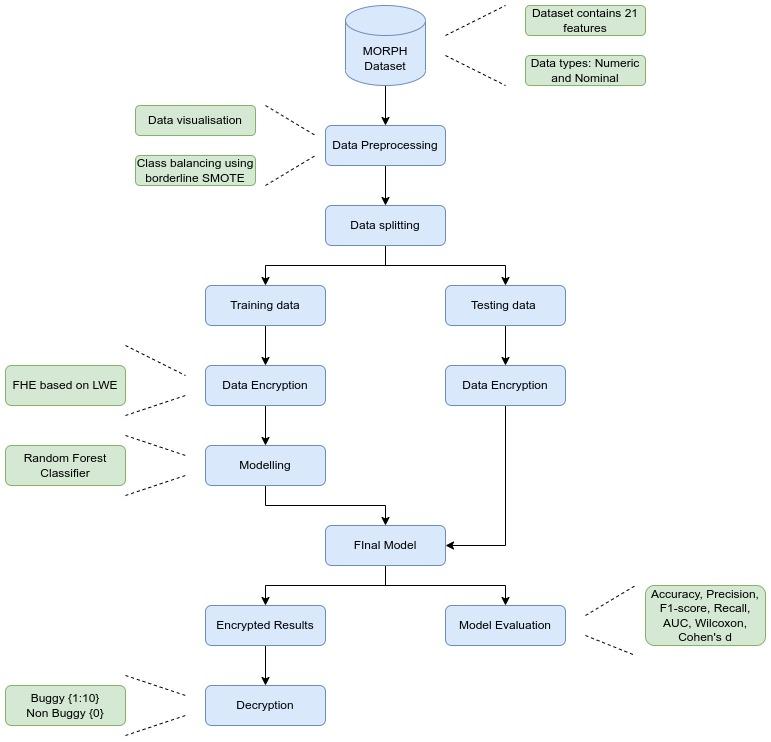
The entire operation is composed of two independent processes: privacy-preserving training and privacy protection prediction [32]. The regression training procedure is carried out in the ciphertext domain to protect data privacy. In addition to creating model parameters, a typical linear regression equation is produced. With the aid of the cipher dataset and model parameters created in the previous step, predictions are made. The returned result is encrypted, and only the user with access to the private key can decrypt it. In related fields, privacy issues are critical, and many services that use customer data must address them. The fully homomorphic encryption scheme used in the PP-NBC classification process, which evaluates operations involving encrypted data, enables service providers to handle customer data without being aware of its true value [33]. Before sending the data, the sender encrypts it using their public key. The receiver must first receive all of the data's component parts in order to analyze it. Additionally, by rotating the data vectors, it defends the classification results against outside attacks. For experimentation, the Iris dataset from the UCI ML repository was used. The authors conduct a preliminary performance analysis of the suggested method, which is implemented by the homomorphic encryption library HElib. The proposed PP-NBC performed more accurately with smoothing than it did without it. Fully homomorphic encryption provides better security as well as accuracy for classification operations.

To protect the privacy of the individual data points, LDP involves introducing noise to the data at the individual level before it is shared with the machine learning model. FML involves training a machine learning model on data that is dispersed across multiple devices or organizations. To prevent the data from being shared or accessed in an unauthorized way, techniques like secure multi-party computation or homomorphic encryption are used [34]. The ease of implementation, effectiveness at safeguarding the privacy of specific data points and user groups, and potential for bias introduction into the model are just a few of the trade-offs between these two approaches that are discussed in the article. The article offers suggestions for choosing the best privacy preservation method based on the particular requirements and limitations of a given machine learning task and discusses the potential applications and limitations of each approach.

| **Summary:** Data security is one of the major challenges for any organization. Homomorphic encryption and differential privacy are some of the concepts that can be utilized to provide security in ML models. Numerous techniques, such as HOPE, A-DPRF, FTLKD, FTLGC, etc., have been proposed that use partial homomorphic encryption and differential privacy to provide data security. Advanced homomorphic encryption techniques can be used to provide better security and performance. |
| --- |

**4. Proposed Method**

First, let’s consider the use-case scenarios for a secure data machine learning model. One of the main issues in the software industry is data security. SDP involves datasets that contain crucial and sensitive information regarding software and its defects. Moreover, these datasets also contain some private information about developers, such as their email, name, age, etc. Attackers may use this data for various malicious purposes if they can get easy access to it. Therefore, we need to perform defect prediction in a way that we can keep our data secure throughout the whole process. Privacy attacks such as, model inversion attacks, model backstepping attacks, etc. can be rendered ineffective with the use of the FHE-based SDP method.



**Fig. 1** Proposed methodology

There can be multiple ways of applying a privacy-preserving machine learning method. One approach is that a client can perform in-house ML-based software defect prediction without ever decrypting the sensitive information at any step of defect prediction through ML. Another approach can be where there is a client who requests a service and there is a service provider who will provide the service. Here, the client generates and holds the keys, performs the data preprocessing, encrypts the data, and sends it to the service provider. The service provider can perform the analysis, build and compile a model for defect prediction, and then perform the analysis. Results can be shared back to the client in an encrypted manner.

Another major issue with software defect prediction is class imbalance. As the non-buggy class is the majority here, there can be cases of overfitting of the model, which can alter the accuracy of the results. This can result in an increase in the number of false positives. In the preprocessing step, there is a need to balance the data for more accurate and reliable predictions. We will use borderline-SMOTE for providing this solution. As Borderline-SMOTE uses only the border points for creating synthetic examples of minority classes, it is well suited for SDP.

In previous works, such as the HOPE method, privacy-preserving machine learning is achieved but without solving the class imbalance issue. This results in models that are less accurate, as working with encrypted data will result in a loss of predictive performance. Since we have used borderline-SMOTE as well as fully homomorphic encryption that allows more complex mathematical operations on encrypted data, there has been an increase in the security as well as the performance of our approach.

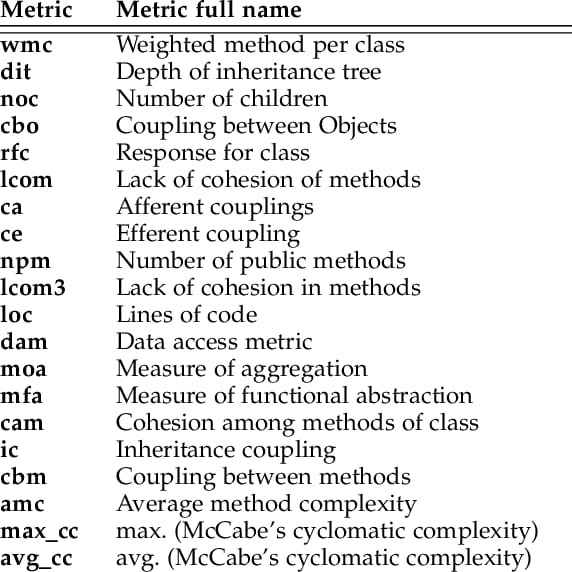
As shown in figure 1, we have first applied some data preprocessing steps to solve the issue of class imbalance. We have applied borderline-SMOTE as a technique to balance the classes, and then, with the help of a concrete-ML module, we have performed machine learning based prediction in an encrypted domain. The cryptographic technique used for encryption and decryption is FHE with learning with errors. Before encrypting , keys are generated. Then a model is built and compiled using the encrypted data. FHE allows mathematical operations on encrypted data that can be used to perform predictions.

**5. Experimental Setup**

The experiments are performed on a system with 8 GB of RAM, an Intel Core i5 processor, Ubuntu 22.04, Python 3.10.7, and Visual Studio code. Statistical testing is performed with the help of Rstudio.

**5.1 Dataset Used**

We have used the MORPH dataset for the experimentation purpose [9]. This dataset consists of nine open-source and real-world projects, such as ant-1.7, camel-1.4, velocity-1.6, etc. The name and description of each feature are presented in the figure below [11].



**Fig. 2** Description of the feature set of MORPH dataset

**5.2 Performance Measures**

The following performance measures have been used for the purpose of testing our results:

*5.2.1 Accuracy:*

This metric assesses a model's general degree of correctness. Expression to calculate accuracy: True predictions / Total predictions

*5.2.2 AUC:*

The AUC quantifies how well a classification model can distinguish between classes in a dataset. The area under the ROC curve is used to compare the TPR and FPR at various classification thresholds. High AUC models are thought to be better at differentiating between the two classes.

*5.2.3 Recall:*

Recall is a performance metric that quantifies the proportion of TP cases that classification models correctly predict to be positive. A model with a higher recall is thought to perform better at detecting positive cases. Expression to calculate recall: TP / (TP + FN)

*5.2.4 Precision:*

The proportion of positive cases that are TP when they are predicted to be positive is measured by a performance metric called precision. Positive case detection is thought to be more efficient with a model that has a high degree of precision. Depression to calculate precision: TP / (TP + FP)

*5.2.5 F-measure:*

A performance metric called the F-measure combines recall and precision into one metric. A higher value denotes better performance, and it is derived as the harmonic mean of recall and precision. The F-measure is a more balanced performance indicator because it combines recall and precision in a way that accounts for both false positive and false negative rates. Expression to calculate F-value: 2\*Prec\*Rec / (Prec + Rec)

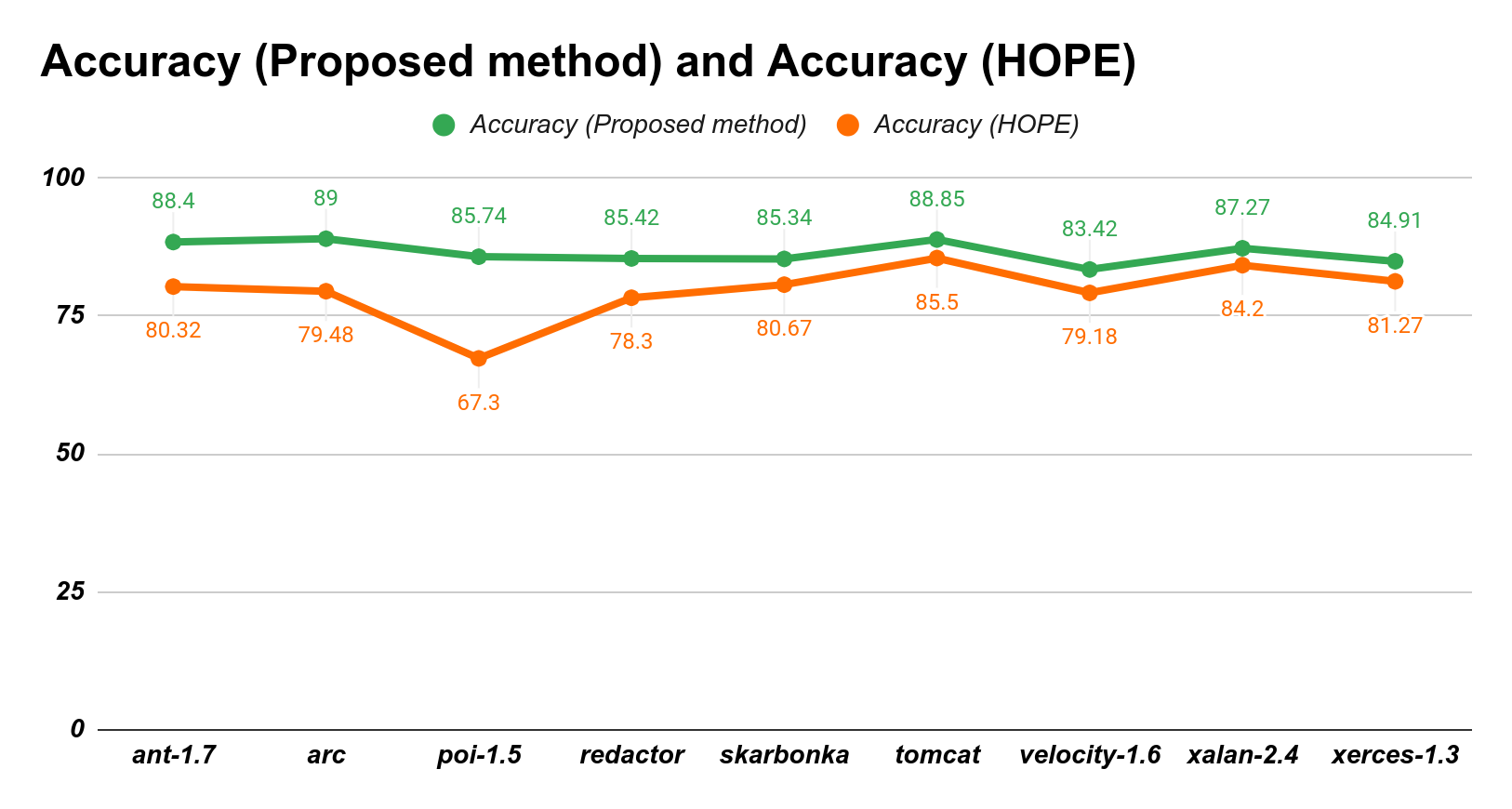
*5.2.6 ROC:*

One of the most popular methods for displaying how well a classification model performs at different thresholds is a ROC plot. It is a plot of TPR versus FPR. It can be observed that when we lower the classification threshold, the number of items classified as positive will increase. As a matter of fact, both TP and FP will increase.

**6. Result Analysis**

Security and class imbalance are some major issues for efficient software defect prediction. In this paper, we propose a technique based on borderline-SMOTE and FHE based on learning with errors that can provide better security as well as improved performance over the previous state-of-art. In the previous method (HOPE), researchers used homomorphic encryption based on the Paillier cryptosystem, which is a partial homomorphic technique. These kinds of techniques can have data leakage issues as there is only one operation performed on encrypted data.

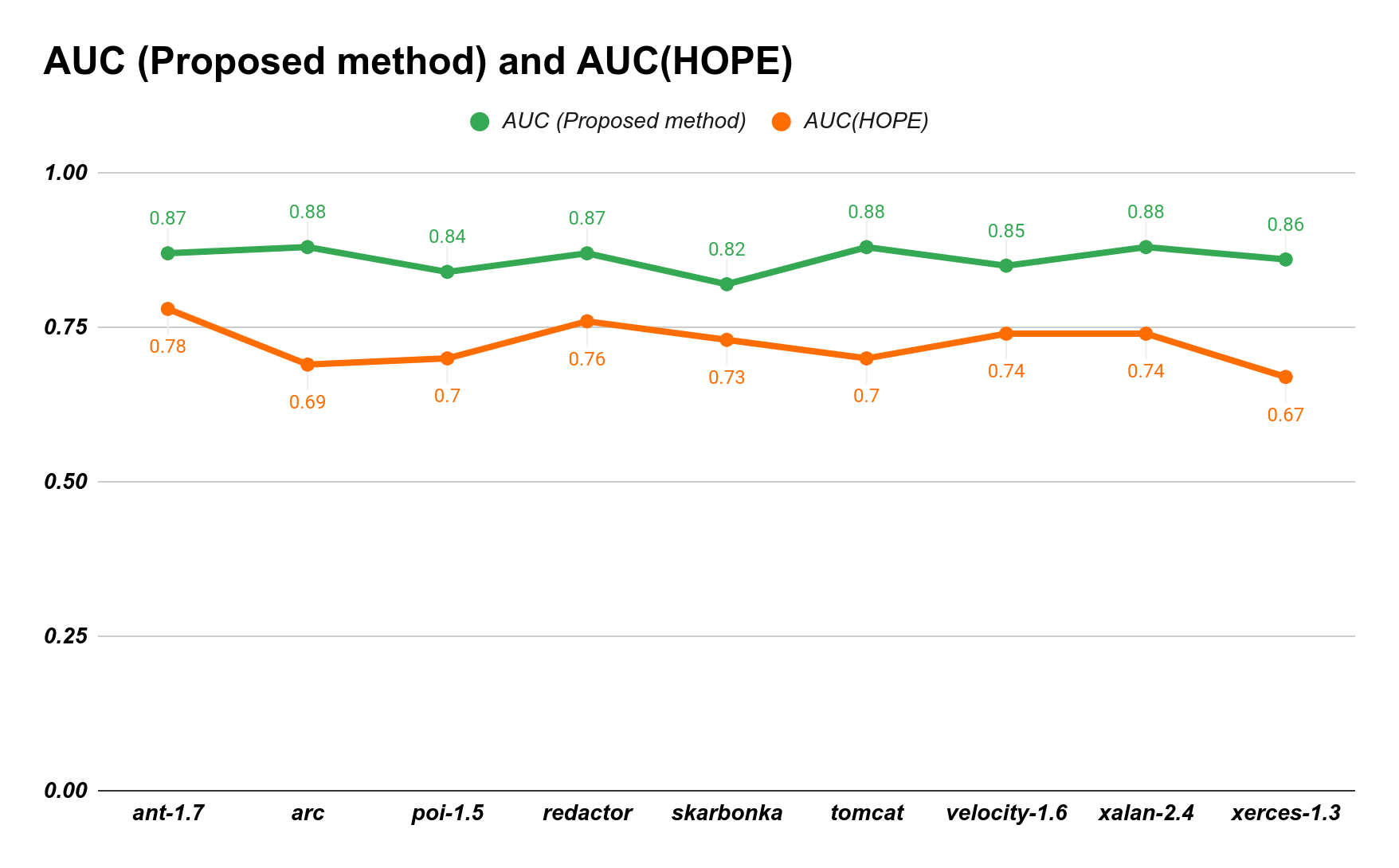
To cover the privacy issues, we have used a FHE based on the LWE problem to provide a more secure theoretical framework. Since FHE is used, more operations are allowed on encrypted data, which also increases the predictive performance of models as more complex mathematical operations are allowed in the encrypted domain.



**Fig. 3** Comparison of accuracy scores

The range of accuracy scores by the existing method is from 67.30% to 85.50%. The range of accuracy scores by the proposed method is from 83.42% to 89.00%. The state-of-the-art had an average prediction accuracy score of 79%, whereas the proposed approach had an average prediction accuracy score of 86.48%. Moreover, there was a loss of accuracy in the previous method (81% to 79%).

The range of AUC scores by the existing method is from 0.67 to 0.78. The range of AUC scores by the proposed method is from 0.82 to 0.88. The state-of-the-art had an average prediction AUC score of 0.72, whereas the proposed approach had an average prediction AUC score of 0.8611.



**Fig. 4** Comparison of AUC scores

Since the results of our suggested method can be kept remarkably similar before and after encryption, there is hardly any accuracy loss. A comparison of accuracy and AUC scores has been presented in Table 1. As shown in Figures 3 and 4, the proposed method consistently performs better than the existing HOPE method.

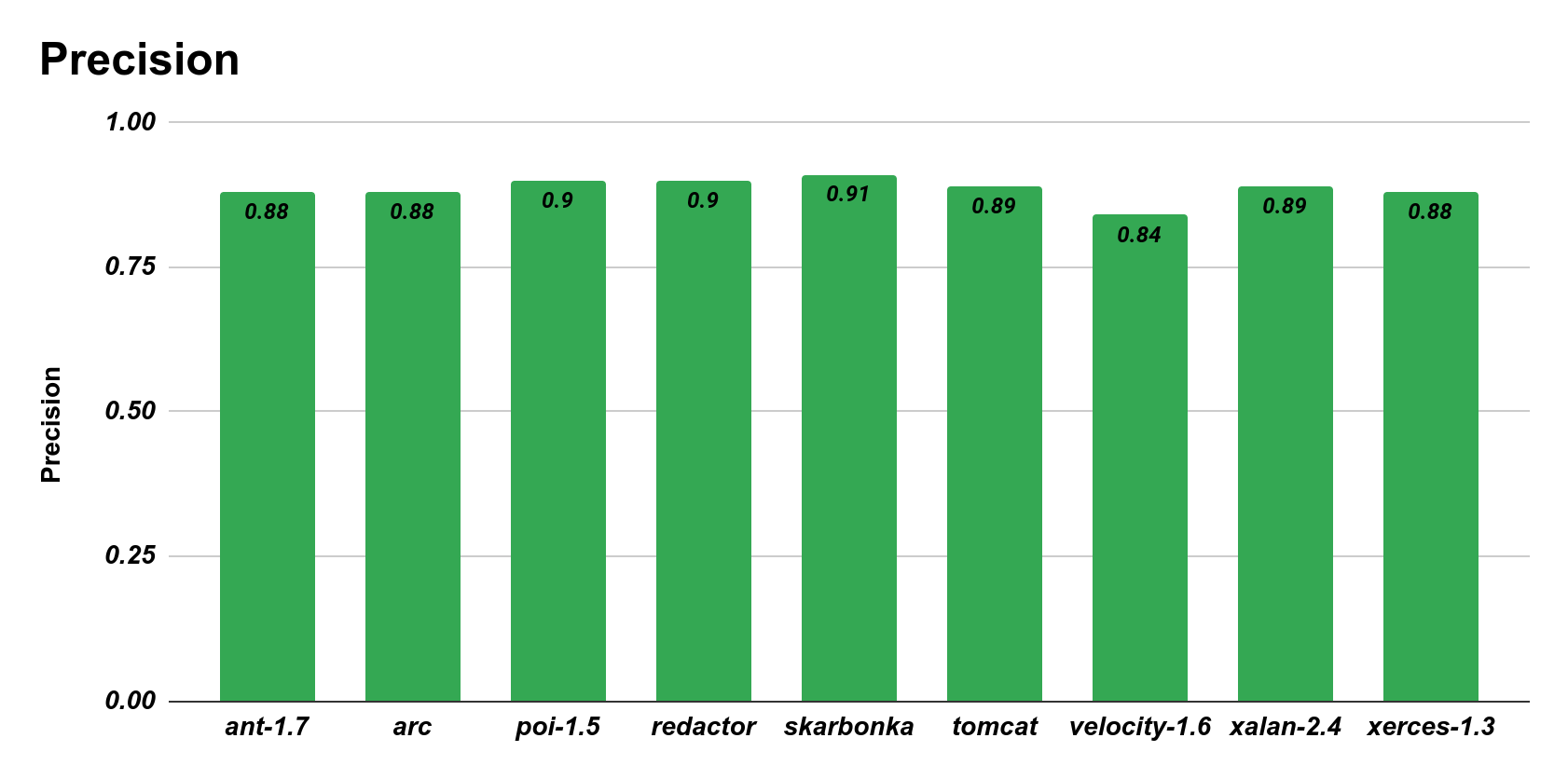
| **Name** | **Accuracy (Proposed method)** | **Accuracy (HOPE)** | **AUC (Proposed method)** | **AUC(HOPE)** |
| --- | --- | --- | --- | --- |
| ant-1.7 | **88.40** | 80.32 | **0.87** | 0.78 |
| arc | **89.00** | 79.48 | **0.88** | 0.69 |
| poi-1.5 | **85.74** | 67.30 | **0.84** | 0.70 |
| redactor | **85.42** | 78.30 | **0.87** | 0.76 |
| skarbonka | **85.34** | 80.67 | **0.82** | 0.73 |
| tomcat | **88.85** | 85.50 | **0.88** | 0.70 |
| velocity-1.6 | **83.42** | 79.18 | **0.85** | 0.74 |
| xalan-2.4 | **87.27** | 84.20 | **0.88** | 0.74 |
| xerces-1.3 | **84.91** | 81.27 | **0.86** | 0.67 |

**Table 1** Comparison of the proposed method with the state of the art (HOPE)

In order to evaluate the effectiveness of our model, we have also used additional metrics, with the results shown in Table 2.

| **Name** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| ant-1.7 | 0.88 | 0.87 | 0.87 |
| arc | 0.88 | 0.89 | 0.89 |
| poi-1.5 | 0.90 | 0.90 | 0.90 |
| redactor | 0.90 | 0.90 | 0.90 |
| skarbonka | 0.91 | 0.90 | 0.90 |
| tomcat | 0.89 | 0.89 | 0.89 |
| velocity-1.6 | 0.84 | 0.85 | 0.86 |
| xalan-2.4 | 0.89 | 0.89 | 0.89 |
| xerces-1.3 | 0.88 | 0.87 | 0.88 |

**Table 2** Performance analysis of the proposed method on the basis of precision, recall, and F1-score

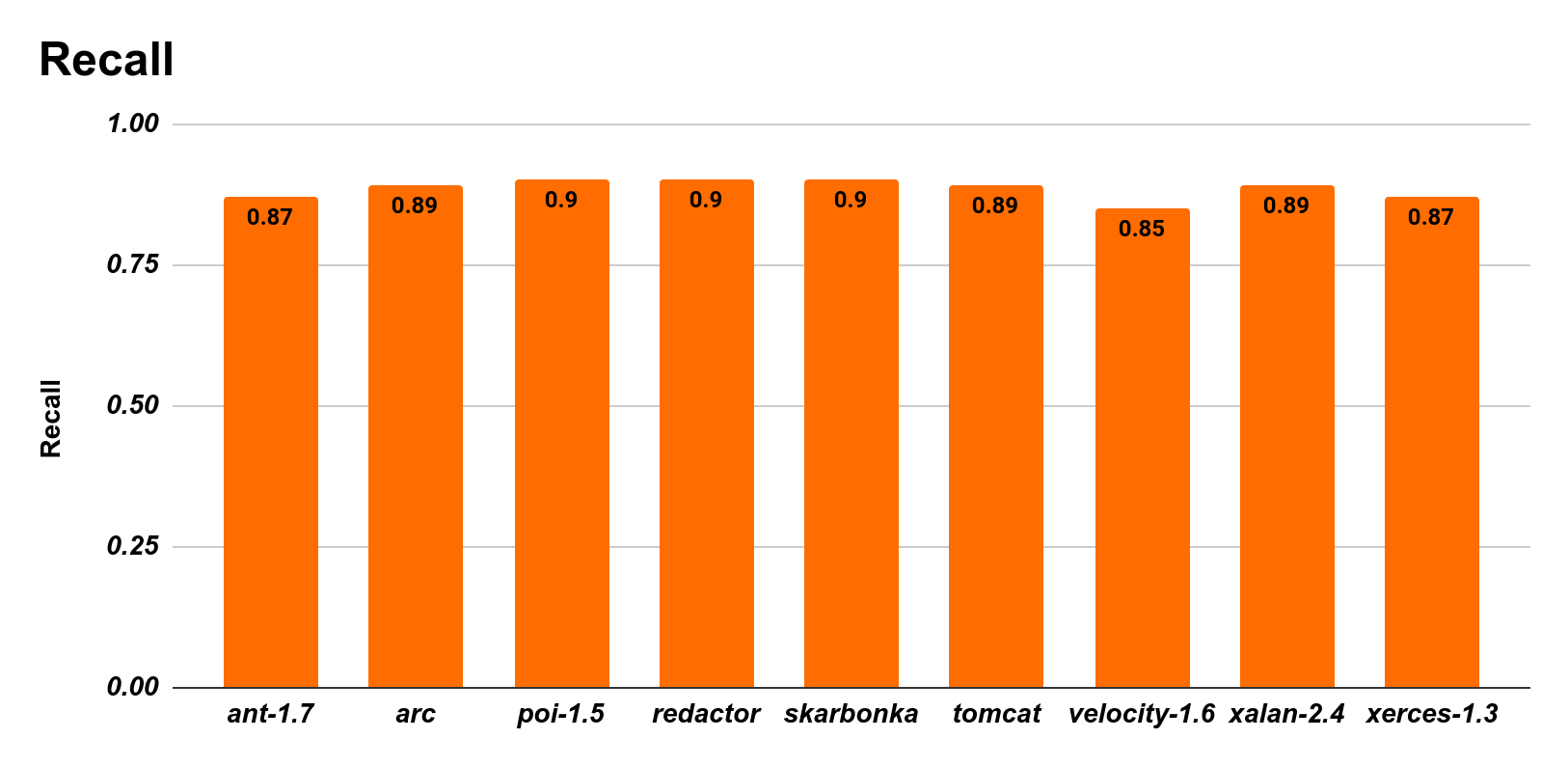


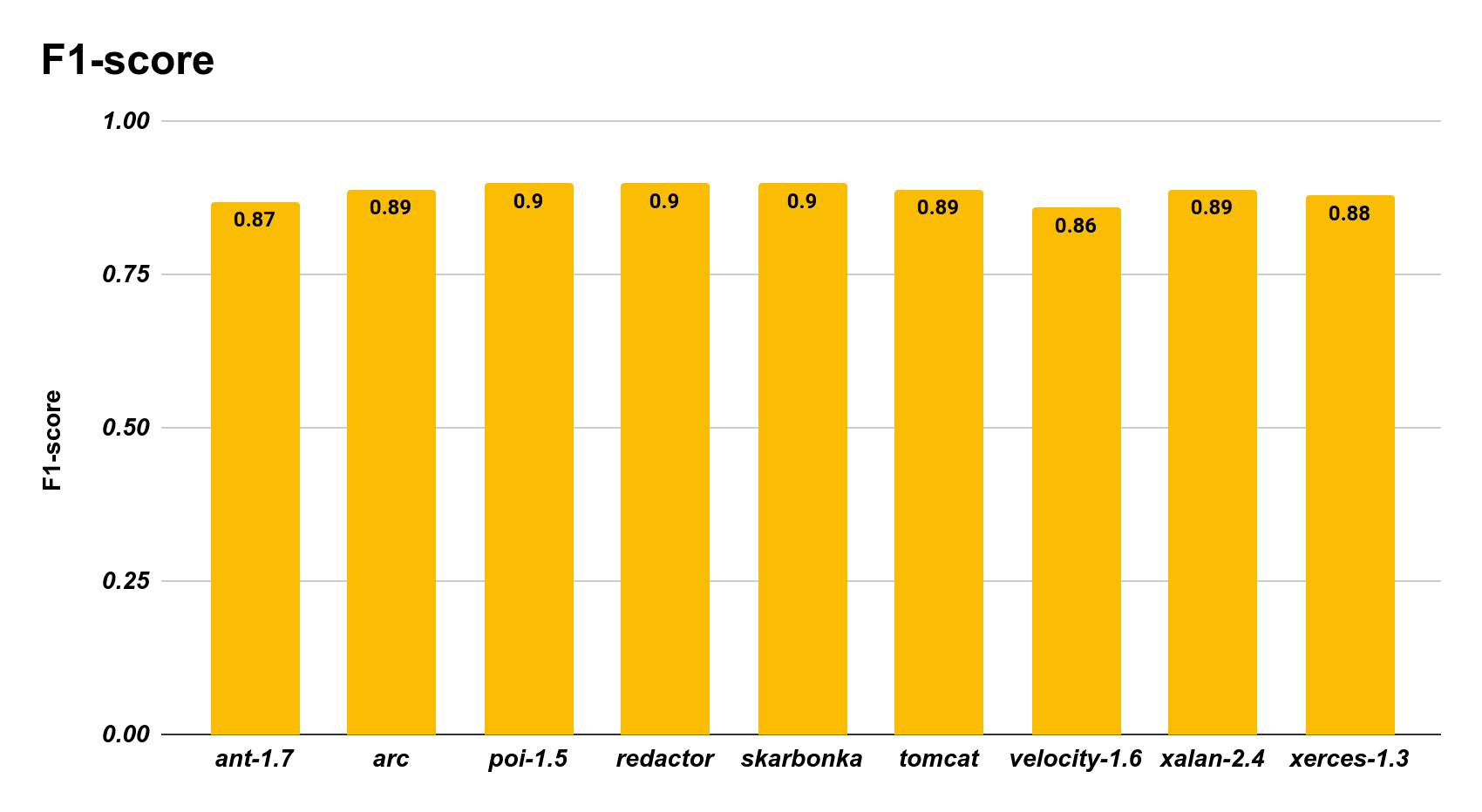
**Fig. 4** Precision of the Proposed Method

Our proposed method performs very well in terms of precision, recall, and F1-score, as shown in figures 4, 5, and 6. These metrics are very useful when data is imbalanced. An accurate classification of the true positives by our model is suggested by good performance on these metrics. On average, the proposed method on the MORPH dataset scores 0.8856, 0.8844, and 0.8867 on precision, recall, and F1-score, respectively.

**6.1 Statistical testing**

The validation of our findings can be accomplished through statistical testing, particularly through the use of hypothesis testing techniques like the Wilcoxon rank-sum test. By using Cohen's d method [39], we can determine the amount of difference in the performance of the two methods.

**Fig. 5** Recall of the Proposed method

**Fig. 6** F1- score of the Proposed Method

*6.1.1 Wilcoxon rank-sum test:*

We used the non-parametric Wilcoxon rank-sum test with a significance level of 5% to demonstrate the statistical significance of our findings. The 5 percent significance level clearly states that the null hypothesis is rejected if the p-value is less than 0.05, and if it is, it can be inferred that our suggested method is superior to the existing HOPE method.

To compare the efficacy of our suggested method and the current HOPE method, the following hypotheses are developed:

* H0: The proposed method does not outperform the one currently in use, HOPE.
* H1: The proposed approach is better than HOPE.

A comparison of accuracy scores and AUC scores is done using the Wilcoxon test. The difference between the accuracy and AUC scores is statistically significant, with a low p-value of 0.00078 and 0.00039, respectively. The alternative hypothesis will be followed because the p-value for both performance metrics, accuracy and AUC, is less than 0.05. We can comfortably state that our proposed method outperforms the existing method, HOPE.

| **Measure** | **p-value** | **Cohen’s d** |
| --- | --- | --- |
| Accuracy | 0.00078 | 1.7653 |
| AUC | 0.00039 | 4.1710 |

**Table 3** Statistical testing results on accuracy and AUC scores

*6.1.2 Cohen’s d value:*

Methods for testing hypotheses like the Wilcoxon rank-sum test are unable to provide us with information about how much the performance has improved. The term "effect size" can also be used to describe the variation in performance enhancements. The effect size is calculated using Cohen's d method. We can use the following criteria to interpret the results:

* If d-value is between 0.2 and 0.5, then effect size is small.
* If d-value is between 0.5 and 0.8, then effect size is moderate.
* If d-value is between 0.8 and 1.3, then effect size is significant.
* If d-value is over 1.3, then effect size is very large.

As shown in table 3, the effect sizes for accuracy score and AUC score were 1.7653 and 4.1710, respectively. From the above results, we can state that the effect size is very large for both the performance metrics (accuracy score and AUC score). Our suggested method has provided a very significant improvement in performance over the existing method, HOPE.

**7. Threats to validity**

In this section, threats to the internal and external validity of the proposed approach have been discussed.

**7.1 Internal validity**

The following are some threats to internal validity:

* Selection bias: There is a possibility that the selection of data used for the study is not random or representative of the entire population. This could cause bias in the results.
* Maturity: Over time, the participants or the software system under study may alter or advance, producing outcomes that differ from those seen during the study.
* Randomness: The results of the random forest classifier can be influenced by the random seeds used during training and testing, leading to variability in results.
* Model complexity: The model's complexity could be a challenge, as fully homomorphic encryption is computationally expensive, leading to long runtimes and difficulty in scaling to larger datasets.

**7.2 External Validity**

The following are some threats to external validity:

* Generalizability of results: The findings of the study may not be generalizable to other contexts outside the specific dataset and software systems used in the study.
* Unintended consequences: The use of fully homomorphic encryption for software defect prediction may have unintended consequences, such as reducing the interpretability of the model or hindering software engineers' ability to make informed decisions.
* External factors: External factors such as changes in the software development environment, new technologies, or changes in software development processes may impact the validity and applicability of the model over time.

**8. Conclusion**

In this study, we have proposed an approach that can be used to tackle the problems of class imbalance and data security. We have used borderline SMOTE to provide balance in the classes. Following that, with fully homomorphic encryption and random forest classifiers, we used concrete-ML to provide a framework for defect prediction. The results show that the applied technique can provide better prediction accuracy and security than the previous method. There was a negligible loss of prediction accuracy while performing prediction with FHE. In future work, we can reduce the time complexity of the proposed technique, as machine learning based on FHE takes a very long time to perform predictions.

**References**

1. Newman M. Software errors cost the US economy 59.5 billion annually: NIST assesses technical needs of industry to improve software-testing. 2002.
2. NASA(2015)https://www.nasa.gov/sites/default/files/files/Space\_Math\_VI\_2015.pdf
3. S. Yadav and G. P. Bhole, "Handling Imbalanced Dataset Classification in Machine Learning," 2020 IEEE Pune Section International Conference (PuneCon), 2020, pp. 38-43, doi: 10.1109/PuneCon50868.2020.9362471.
4. Bennin, K.E., Keung, J.W. & Monden, A. On the relative value of data resampling approaches for software defect prediction. Empir Software Eng 24, 602–636 (2019). https://doi.org/10.1007/s10664-018-9633-6
5. Q. Song, Y. Guo and M. Shepperd, "A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction," in IEEE Transactions on Software Engineering, vol. 45, no. 12, pp. 1253-1269, 1 Dec. 2019, doi: 10.1109/TSE.2018.2836442.
6. G. G. Cabral, L. L. Minku, E. Shihab and S. Mujahid, "Class Imbalance Evolution and Verification Latency in Just-in-Time Software Defect Prediction," 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 666-676, doi: 10.1109/ICSE.2019.00076.
7. Tahir, A., Bennin, K.E., Xiao, X. et al. Does class size matter? An in-depth assessment of the effect of class size in software defect prediction. Empir Software Eng 26, 106 (2021). https://doi.org/10.1007/s10664-021-09991-3
8. Concrete-ML: a Privacy-Preserving Machine Learning Library using Fully Homomorphic Encryption for Data Scientists, Arthur Meyre and Benoit Chevallier-Mames and Jordan Frery and Andrei Stoian and Roman Bredehoft and Luis Montero and Celia Kherfallah, 2022}, https://github.com/zama-ai/concrete-ml
9. M. Jureczko and L. Madeyski, ‘‘Towards identifying software project clusters with regard to defect prediction,’’ in Proc. 6th Int. Conf. Predictive Models Softw. Eng. (PROMISE), 2010, pp. 1–10.
10. Wood, A., Shpilrain, V., Najarian, K., Mostashari, A., Kahrobaei, D. (2018). Private-Key Fully Homomorphic Encryption for Private Classification. In: Davenport, J., Kauers, M., Labahn, G., Urban, J. (eds) Mathematical Software – ICMS 2018. ICMS 2018. Lecture Notes in Computer Science(), Vol 10931. Springer, Cham. https://doi.org/10.1007/978-3-319-96418-8\_56.
11. Okutan, Ahmet. (2018). Use of Source Code Similarity Metrics in Software Defect Prediction.
12. Wei Zheng, Tianren Shen, Xiang Chen, Peiran Deng, Interpretability application of the Just-in-Time software defect prediction model, Journal of Systems and Software, Volume 188, 2022, 111245, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2022.111245.
13. Goyal, S. Effective software defect prediction using support vector machines (SVMs). Int J Syst Assur Eng Manag 13, 681–696 (2022). https://doi.org/10.1007/s13198-021-01326-1
14. G. G. Cabral and L. L. Minku, "Towards Reliable Online Just-in-Time Software Defect Prediction," in IEEE Transactions on Software Engineering, vol. 49, no. 3, pp. 1342-1358, 1 March 2023, doi: 10.1109/TSE.2022.3175789.
15. Chen, Lq., Wang, C. & Song, Sl. Software defect prediction based on nested-stacking and heterogeneous feature selection. Complex Intell. Syst. 8, 3333–3348 (2022). https://doi.org/10.1007/s40747-022-00676-y
16. Hua Wei, Changzhen Hu, Shiyou Chen, Yuan Xue, Quanxin Zhang, Establishing a software defect prediction model via effective dimension reduction, Information Sciences, Volume 477, 2019, Pages 399-409, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2018.10.056.
17. S. Huda et al., "An Ensemble Oversampling Model for Class Imbalance Problem in Software Defect Prediction," in IEEE Access, vol. 6, pp. 24184-24195, 2018, doi: 10.1109/ACCESS.2018.2817572.
18. L. Gong, S. Jiang and L. Jiang, "Conditional Domain Adversarial Adaptation for Heterogeneous Defect Prediction," in IEEE Access, vol. 8, pp. 150738-150749, 2020, doi: 10.1109/ACCESS.2020.3017101.
19. Y. Huang and X. Xu, "Two-stage cost-sensitive local models for heterogeneous cross-project defect prediction," 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 2022, pp. 819-828, doi: 10.1109/COMPSAC54236.2022.00132.
20. Yuxiang Gao, Yi Zhu, Yu Zhao, Dealing with imbalanced data for interpretable defect prediction, Information and Software Technology, Volume 151, 2022, 107016, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.107016.
21. Satya Srinivas Maddipati and Malladi Srinivas, “An Hybrid Approach for Cost Effective Prediction of Software Defects” International Journal of Advanced Computer Science and Applications(IJACSA), 12(2), 2021. http://dx.doi.org/10.14569/IJACSA.2021.0120219
22. S. Yadav and G. P. Bhole, "Handling Imbalanced Dataset Classification in Machine Learning," 2020 IEEE Pune Section International Conference (PuneCon), Pune, India, 2020, pp. 38-43, doi: 10.1109/PuneCon50868.2020.9362471.
23. T. Zhang, T. Zhu, P. Xiong, H. Huo, Z. Tari and W. Zhou, "Correlated Differential Privacy: Feature Selection in Machine Learning," in IEEE Transactions on Industrial Informatics, vol. 16, no. 3, pp. 2115-2124, March 2020, doi: 10.1109/TII.2019.2936825.
24. C. Yu, Z. Ding and X. Chen, "HOPE: Software Defect Prediction Model Construction Method via Homomorphic Encryption," in IEEE Access, vol. 9, pp. 69405-69417, 2021, doi: 10.1109/ACCESS.2021.3078265.
25. D. Zhang, X. Chen, Z. Cui and X. Ju, "Software Defect Prediction Model Sharing Under Differential Privacy," 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 2018, pp. 1547-1554, doi: 10.1109/SmartWorld.2018.00266.
26. A. Wang, Y. Zhang and Y. Yan, "Heterogeneous Defect Prediction Based on Federated Transfer Learning via Knowledge Distillation," in IEEE Access, vol. 9, pp. 29530-29540, 2021, doi: 10.1109/ACCESS.2021.3058886.
27. Chen, X., Zhang, D., Cui, ZQ. et al. DP-Share: Privacy-Preserving Software Defect Prediction Model Sharing Through Differential Privacy. J. Comput. Sci. Technol. 34, 1020–1038 (2019). https://doi.org/10.1007/s11390-019-1958-0
28. A. Wang, Y. Zhao, G. Li, J. Zhang, H. Wu and Y. Iwahori, "Heterogeneous Defect Prediction Based on Federated Reinforcement Learning via Gradient Clustering," in IEEE Access, vol. 10, pp. 87832-87843, 2022, doi: 10.1109/ACCESS.2022.3195039.
29. Z. Guo, S. Wang, W. Jin, C. Gong and N. Lin, "A k-Nearest Neighbor Algorithm Based on Homomorphic Encryption," 2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS), Shenyang, China, 2019, pp. 15-20, doi: 10.1109/IUCC/DSCI/SmartCNS.2019.00032.
30. C. Liu et al., "Efficient and Privacy-Preserving Logistic Regression Scheme based on Leveled Fully Homomorphic Encryption," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2022, pp. 1-6, doi: 10.1109/INFOCOMWKSHPS54753.2022.9797933.
31. J. H. Cheon, D. Kim, Y. Kim and Y. Song, "Ensemble Method for Privacy-Preserving Logistic Regression Based on Homomorphic Encryption," in IEEE Access, vol. 6, pp. 46938-46948, 2018, doi: 10.1109/ACCESS.2018.2866697.
32. Bijiao Chen, Xianghan Zheng, Implementing Linear Regression with Homomorphic Encryption, Procedia Computer Science, Volume 202, 2022, Pages 324-329, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.04.044.
33. Kim, S., Omori, M., Hayashi, T., Omori, T., Wang, L., Ozawa, S. (2018). Privacy-Preserving Naive Bayes Classification Using Fully Homomorphic Encryption. In: Cheng, L., Leung, A., Ozawa, S. (eds) Neural Information Processing. ICONIP 2018. Lecture Notes in Computer Science(), vol 11304. Springer, Cham. https://doi.org/10.1007/978-3-030-04212-7\_30
34. H. Zheng, H. Hu and Z. Han, "Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning?," in IEEE Intelligent Systems, vol. 35, no. 4, pp. 5-14, 1 July-Aug. 2020, doi: 10.1109/MIS.2020.3010335.
35. Mori, T., Uchihira, N. Balancing the trade-off between accuracy and interpretability in software defect prediction. Empir Software Eng 24, 779–825 (2019). https://doi.org/10.1007/s10664-018-9638-1
36. X. Sun, P. Zhang, J. K. Liu, J. Yu and W. Xie, "Private Machine Learning Classification Based on Fully Homomorphic Encryption," in IEEE Transactions on Emerging Topics in Computing, vol. 8, no. 2, pp. 352-364, 1 April-June 2020, doi: 10.1109/TETC.2018.2794611.
37. J. -W. Lee et al., "Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network," in IEEE Access, vol. 10, pp. 30039-30054, 2022, doi: 10.1109/ACCESS.2022.3159694.
38. Dong, X., Yu, Z., Cao, W. et al. A survey on ensemble learning. Front. Comput. Sci. 14, 241–258 (2020). https://doi.org/10.1007/s11704-019-8208-z
39. Shailza Kanwar, Lalit Kumar Awasthi, Vivek Shrivastava, Candidate project selection in cross project defect prediction using hybrid method, Expert Systems with Applications, Volume 218, 2023, 119625, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.119625.