A Study on Machine Learning Techniques for Predicting Software Defects

Akhilesh G.V Dhavileswarapu[1], Namburi Rama Bhadra Raju[2], Rimmalapudi Sri Nihaal[3], Shanmuk Srinivas Amiripalli[4]

[1,2,3,4]*Department of CSE, GIT, GITAM UNIVERSITY, Visakhapatna, AP, India.*

---***---

Abstract - *High quality software cannot be developed without resorting to high quality o testing. Testing object oriented programs, the object oriented features need to be tested specifically. Software defect prediction is especially useful for testing of large software systems. Software defect prediction and its techniques used to at present for testing object oriented software pay very little attention to adequately test object-oriented features. Our preliminary review and proposed model show that, using proper SDP mutant score can be accomplished. Software defect prediction and its techniques used to at present for testing object oriented software pay very little attention to adequately test object-oriented features.*

Key Words: software engineering, software defect, ML Techniques, Complexity-based Oversampling Technique, software defect prediction.

1. INTRODUCTION

Until until, testing and debugging software on a regular basis was a necessary step in developing highly dependable software. A deep learning approach known as software defect prediction (SDP) has grown increasingly popular among engineers with the introduction of artificial intelligence. SDP models identify defect-prone things (such as functions and files)[3]. SDP makes testing and debugging simple and effective for developers by removing obstacles. These software defects are caused by a software system's inadequacy, which leads to unexpected conditions after the software is sent out, resulting in a situation where the software product does not meet the client's preferences, lowering the software product quality and increasing development costs. A productive software prediction model [6] may help developers and maintainers avoid such situations. The empirical findings also suggest that employing sentence-based and keyword-based prediction patterns may help pre-trained neural language models detect software defects better[2].

SDP, on the other hand, has issues with class imbalance [5]. Class imbalance is an issue that occurs when the amount of defective and non-defective occurrences in a dataset are not equal, causing prediction models to favour the majority class and disregard the minority class[3]. To address this issue, SDP employs machine learning methods that utilise previous data to build defect prediction models that can predict whether future instances will be faulty or not. Because machine language applications include several real-world simulations, machine learning methods are seen to be the ideal solution to SDP.

To cope with the issue of class imbalance in machine learning, data sampling approaches are used.

The oversampling method and the undersampling approach are two kinds of data sampling procedures. By concentrating on normalising the distribution of the datasets, both of these strategies increase the prediction model's effectiveness[3]. In SDP, however, the oversampling method has been the dominant method. There is a risk of losing crucial and useful information for the prediction model when using the under-sampling strategy. Oversampling strategies, on the other hand, add a new instance to the minority class in order to bring its value closer to or equal to that of the majority class.

Oversampling approaches such as COSTE (Complexity-based Oversampling Technique) and SMOTE (Synthetic Minority Oversampling Technique) have been developed.It has been proposed by academics and several peer studies to address the issue of class imbalance[3,15]. The COSTE technique uses higher-ranked instances more often than lower-ranked instances to generate more balanced datasets, causing the prediction model to pay more attention to the less complex instances and then averaging the higher and lower ranked instances together to generate synthetic instances. The K-nearest neighbour (KNN) method is used in the SMOTE approach to create a synthetic minority class. Another approach based on SMOTE is S.M.O.T.U.N.E.D (Synthetic Minority Oversampling TUNED), which is a neural network and algorithm used to analyse class imbalances.

2. ML TECHNIQUES AND LITERATURE SURVEY

ML approaches are regarded as an operative and operational way for locating problematic modules, in which moving components are discovered by mining hidden patterns among software measurements. ML approaches are being used by a number of academics working with healthcare datasets. The multilayer perceptron (MLP), decision tree (J48), radial basis function (RBF), random forest (RF), hidden Markov model (HMM), credal decision tree (CDT), K-nearest neighbour (KNN), average one dependency estimator (A1DE)[4], and Nave Bayes (NB)[20-25] are some of the machine learning approaches. Techniques for predicting software faults aid in the identification of software system components that are more likely to have flaws. Models may be built using defect prediction approaches to rank software modules based on the projected number of faults, defect likelihood, or classification findings [1]. Sampling Strategies, Cost-sensitive Learning Techniques, Boosting, and Cross Project Software Defect Prediction are all ML techniques addressing class imbalance problems[2]. COSTE (Complexity-based Oversampling Technique) is an unique oversampling technique that can concurrently achieve low pf and high pd [3]. Software defect prediction approaches automatically detect probable problems, resulting in considerable time, effort, and cost savings[19]. Some researchers have already used deep learning methods (e.g., CNN and DBN) to enhance software engineering processes in recent years[26-31]. Deep learning approaches have also been applied in the categorization of test reports, link prediction in online developer forums, software traceability, and other applications [6]. The majority of defect prediction algorithms rely on manually creating new discriminative features or novel combinations of features from labelled historical defect data, which are then fed into machine learning-based classifiers to detect code flaws [11]. Synthetic minority over-sampling methods (SMOTE), like informed oversampling, produce synthetic minority class samples to balance the class distribution [14]. Anomaly detection-based NIDS have widely employed machine learning approaches [32-37]. To discriminate between normal and aberrant network activity, many machine learning models have been used, including support vector machine (SVM), random forest (RF), and decision tree (DT) [13]. The table in this section of the study provides an overview of some of the research publications published in the last few years. The following table summarises the papers' techniques, limits, and benefits.

3. SOFTWARE TOOLS

Following are some of the important software defect prediction tools which are available commercially and some which are available for use for free [38-42].

NAME	DECRIPTION
Class Imbalance Learning	Class imbalance learning focuses on classifying issues with skewed distributions, which might be useful for defect prediction.
Bayesian Networks	To discover the important probabilistic correlations among software metrics and fault proneness, Bayesian networks were used.
Convolutional Neural Networks	Deep learning is used to generate effective features. We initially extract token vectors from the programmes' Abstract Syntax Trees (ASTs), which are then encoded as numerical vectors through mapping and word embedding.
Deep-Learning Model on Static Code Features	It's significant because it shows a fresh application of deep-learning models to an actual challenge encountered by software engineers.
Deep Tree-Based Model	It can learn characteristics for modelling source code automatically and use them to forecast defects, and it directly matches the Abstract Syntax Tree representation of source code.

4. REPOSITORIES FOR SOFTWARE DEFECT PREDICTION

The following table contains the information of the repositories that we found, which contains datasets for the software defect prediction.

Name of Repository	Description	URL	TYPE
Github1	Xiang Chen maintains this repository.	https://smart github.io/sdp.ht	
Kaggle	Kaggle is an online community of datascientists and machine learning practitioners. Kaggle is a subordinate of Google LLC.	https://www. gle.com/search software+defe prediction	PUBLIC
Google Datasets	Data Repository by GOOGLE	shorturl.at/pG 5	PUBLIC
Data.Gov	--	shorturl.at/st 0	PUBLIC
Promise	Software Engineering Repository	http://promis te.uottawa.ca/S epository/datas -page.html	PUBLIC
Data.World	--	shorturl.at/ju 5	PUBLIC
IIT KGP	National Digital library of india	Iitkgp.ac.in	INSTITUTIOL

AUTHOR	METHODOLOGY	LIMITATIONS	CONCLUSIONS	ADVANTAGES
Pan et al.	CodeBERT-NT, CodeBERT-PS, CodeBERT-PK, and CodeBERT-PT are some of the CodeBERT models	1. Language for programming. The PROMISE repository, which is developed in	These among the CodeBERT models proposed in this study for software defect	The use of a pre-trained CodeBERT model enhances prediction performance and saves

	proposed for software fault prediction.	Java, provided this dataset. The strategy is not, however, confined to programming languages. 2. The size of the data collection A portion of the PROMISE repository is employed in this dataset, as it has been in past studies. Some datasets, such as the NASA dataset, are inaccessible due to the project's open-source requirement.a	prediction. They tested if utilising a neural language model like CodeBERT might improve prediction performance, as well as the consequences of various prediction patterns in software defect prediction using CodeBERT models, using models in cross-version and cross-project SDP.	time, according to empirical findings.a
Feng et al.	They suggested the Comple xitybased OverSamplingTEc hnique as a new oversampli ng approach in this work (C OSTE).	The findings of this experiment, which used 23 datasets, cannot be applied to other kinds of metrics, such as process metrics. The findings' external validity is jeopardised due to their lack of generalizability to other datasets or measures.	COSTE is suggested as a viable approach to sol ve the issue of class im balance in SDP.	COSTE considerably enhances the variety of synthetic instances without reducing the effectiveness of prediction models to discover errors, according to experimental data from 23 releases of ten projects.
Feng et al.	The goal of this research is to look at the stability of oversampling approaches based on SMOTE. Furthermore, to increase the stability of SMOTE-based oversampling methods, a series of stable SMOTE-based oversampling techniques are presented.	The fault measurements used in this investigation might be a drawback. Because they only used the static code measure, it can't be used to other sorts of metrics.	Oversampling approac hes based on stable SM OTE should be viewed as an alternative for SM OTEbased oversamplin g techniques.	This comparison of SMOTE-based and stable SMOTE-based oversampling strategies reveals that the stable SMOTE-based oversampling technique outperforms the former.
Zhu et al	Using the recently released whale optimization algorithm (WOA) and simulated annealing, they developed EMWS, a feature search-based technique (SA). They also used a convolutional neural network (CNN) and a kernel extreme learning machine to create WSHCKE, a unified defect prediction predictor (KELM).	Even if we thoroughly examine the experiment method, there may still be some mistakes that go unnoticed. Another issue is the dataset's quality and widespread availability.	They created EMWS, an improved metaheuristic feature selection algorithm that selects fewer but closely related representative features for each software project, leveraging SA's strong local search capability to improve WOA's weak exploitation performance while also leveraging WOA's strong global search capability to boost SA's weak exploration.	The EMWS developed here may efficiently choose a smaller number of representative characteristics that are closely connected. WSHCKE may use CNN to combine the specified characteristics into abstract deep semantic features, improving prediction performance.
Maddipati et al.	The most significant qualities in detecting faulty	When compared to classifiers such as Neural	To tackle the issue of class imbalance, a	Detecting software components that are

	prone modules were identified using principal component analysis as an attribute selection technique.	Networks and Support Vector Machines, the Sugeno Fuzzy Inference classifier increased the area under the ROC curve by just 5%.	machine learning method is being used to predict software defects.	prone to failure
Sahingoz et al.	Error detection and error correction.	The defective class has a greater classification cost than the inert class.	The presented models provide appropriate accuracy levels for software defect prediction, resulting in higher software quality.	Not only can the programme be used on normal desktop computers, but it can also be used on tiny computing devices like sensor networks.
Esteves et al.	Uses a tree boosting technique that takes a training set of records of easy-to-compute properties of each module as input and returns whether that module is defect-prone.	The majority of these studies focus on predicting errors from a large number of software characteristics. Another major flaw in the present literature is the absence of a good explanation of what causes software to become flawed.	We give particular software elements that impact the defectiveness of chosen projects, thus the results are valuable to developers.	The less features aid model explainability, which is vital for providing information to developers on features connected to each module of the code, which is more prone to defects.
Pan et al.	By adaptively selecting groups of Inner and Danger data from the minority class, the Adaptive-SMOTE method improves the SMOTE method, resulting in the creation of a new minority class based on the selected data, preventing the category boundary from expanding and strengthening the original data's distributional characteristics..	The distribution of the original minority data is not well grasped by freshly balanced samples, which is a major shortcoming of current sampling approaches.	Adaptive-SMOTE divides the positive dataset into Danger and Inner and then oversamples the data using SMOTE depending on the original data distribution characteristics.	This method avoids the spread of positive samples and improves the original dataset's distributional features. Gaussian Oversampling is an unique sampling division approach.
Zhang et al.	SGM, a novel class imbalance processing approach that combines Synthetic Minority Over-Sampling Technique (SMOTE) with under-sampling for clustering based on Gaussian Mixture Model, is available for large-scale datasets (GMM).	The intrusion detection dataset's unbalanced class issue restricts the classifier's performance for minority classes.	SGM-CNN beats state-of-the-art intrusion detection approaches and offers an effective solution to unbalanced intrusion detection.	Ability to mine sensitive data on the distinctions between normal and malignant activity.
Singh et al.	To successfully solve unbalanced classification issues, this paper introduces synthetic minority over-sampling strategies based on class-specific extreme learning machines. By developing fresh synthetic samples using the SMOTE approach,	For data with a lot of dimensions, SMOTE isn't particularly useful. SMOTE does not take nearby instances from different classes into account when producing synthetic examples. This may lead to more class overlap and more noise.	To deal with skewed class distribution and unbalanced classification difficulties.This paper proposes and evaluates a class-specific extreme learning machine based on SMOTE.By producing	This study introduces a novel SMOTE-based class-specific extreme learning machine, a variant of the class-specific extreme learning machine (CS-ELM), that employs both minority oversampling and class-specific regularisation.

	the number of samples belonging to the minority class rises.		synthetic samples belonging to the minority class, SMOTE raises the relevance of the minority class samples for defining the decision area of the classifiers.	The synthetic minority oversampling technique was used in this work for minority oversampling (SMOTE). It emphasises the relevance of minority class samples in determining the decision region of classifiers.
Chen et al.	They introduced an adaptive robust SMOTE, dubbed RSMOTE, for class imbalance classification in this study.	It's different from most current approaches, which makes it intriguing and unreliable.	For unbalanced classification with label noise, the RSMOTE approach is presented.Unlike most other approaches, it does not depend on a particular noise filter or add any additional parameters.	RSMOTE promotes learning in three ways: it reduces bias caused by class imbalance; it self-adaptively discriminates the noisy, borderline, and safe sorts of minority samples and reinforces the boundary and safety zones separately; and it minimises bias induced by class imbalance.
Douzas et al.	To rebalance skewed datasets, the approach suggested in this paper uses the basic and common k-means clustering algorithm in combination with SMOTE oversampling.	SMOTE does not examine surrounding instances that might be from different classes while producing synthetic examples. This may lead to more class overlap and more noise.	The suggested technique accomplishes characteristics by clustering data using k-means, which allows data production to be focused on critical parts of the input space.	This paper provides a simple and efficient oversampling approach based on k-means clustering and SMOTE (synthetic minority oversampling technique), which prevents noise creation and successfully solves class imbalances.
C. Lakshmi Prabha	For the prediction of software flaws, a hybrid feature reduction approach and an artificially dependent neural network strategy are provided.	Data standardisation is required for the PCA approach employed in this investigation.Your initial features will become Principal Components when PCA is applied to the dataset. The linear combination of your original attributes makes up the main components. Original features are more legible and interpretable than PrincipalComponents.	The goal of this study is to employ data-mining methods to forecast software faults. Using feature selection strategies, it has been discovered that the period and space difficulties for defect prediction is reduced without reducing prediction accuracy.	According to the research, the suggested technique is efficient and produces an AUC of 98.70%, which is a significant improvement over various statemodels of the past.z
Rahim et al.	The assumption of independent predictor traits is Naive Bayes' primary flaw. All of the qualities in Naive Bayes are assumed to be mutually independent.	The research developed a paradigm for predicting software problems that is both efficient and trustworthy.The framework is broken down into three primary sections, each of which includes data preparation stages such as: noise reduction and	The suggested technique may achieve a 98.7% accuracy utilising the Nave Bayes algorithm, according to the results.	As a result, the suggested technique's relevance is that it would cut the cost of maintenance and reduce code complexity by anticipating flaws early in software systems, which will aid developers in removing such problems and, as a result, enhance software

		normalisation		quality prior to deployment.

PROPOSED MODEL

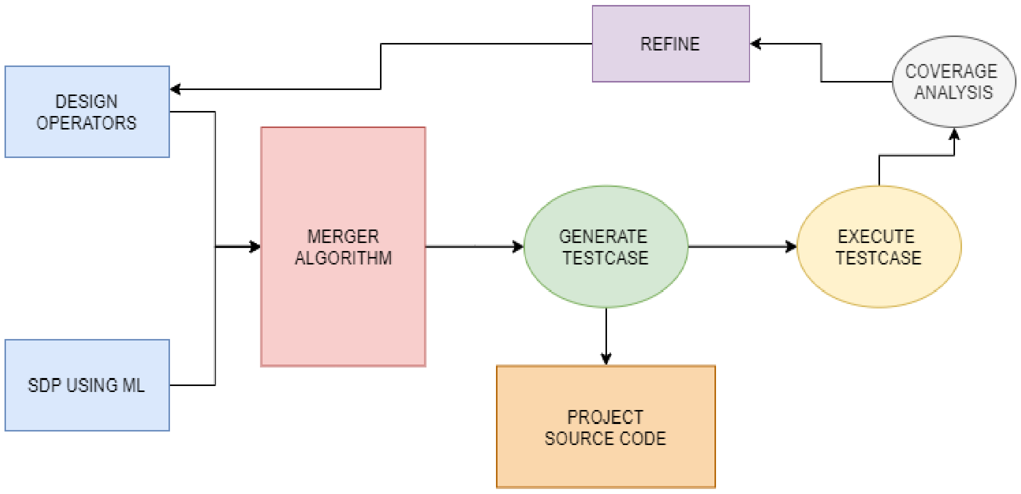

Fig -1: proposed model

5. CONCLUSIONS

Software defect prediction has gained a lot of traction, and it may help you save money and time on your project. Some of the fundamental aspects of object-oriented programming are where the biggest complications occur. The relevance of software defect prediction is steadily increasing as the quantity and complexity of contemporary software products increases. Recent programming methodologies, such as object-oriented systems, cannot, however, be thoroughly evaluated only by looking at the source code. Defect prediction using machine learning is a potential method for detecting software project flaws..

ACKNOWLEDGEMENT

The authors would like to thank prof Srinivas Prasad and the Department of Computer Science Engineering at GITAM University for their assistance.

REFERENCES

[1] Lei Qiao, Xuesong Li, QasimUmer, Ping Guo; Deep-learning based software defect prediction;Qiao, Lei, et al. "Deep learning based software defect prediction." Neurocomputing 385 (2020): 100-110.

[2] Cong, Minyan Lu, and Biao Xu. "An Empirical Study on Software Defect Prediction Using CodeBERT Model." Applied Sciences 11.11 (2021): 4793.

[3] Feng, Shuo, et al. "COSTE: Complexity-based OverSamplingTEchnique to alleviate the class imbalance problem in software defect prediction." Information and Software Technology 129 (2021): 106432.

[4] Bilal Khan,1 Rashid Naseem,2 Muhammad Arif Shah,2 Karzan Wakil,3 Atif Khan,4 M. Irfan Uddin,5 and Marwan Mahmoud6 ; "Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques." Journal of Healthcare Engineering 2021 (2021).

[5] Feng, Shuo, et al. "Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction." Information and Software Technology (2021): 106662.

[6] Zhu, Kun, et al. "Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network." Journal of Systems and Software (2021): 111026.

[7] Yao, Jingxiu, and Martin Shepperd. "The impact of using biased performance metrics on software defect prediction research." Information and Software Technology (2021): 106664.

[8] Maddipati, Satya Srinivas, and Malladi Srinivas. "Machine learning approach for classification from imbalanced software defect data using PCA & CSANFIS." Materials Today: Proceedings (2021).

[9] Cetiner, Murat, and OzgurKoraySahingoz. "A Comparative Analysis for Machine Learning based Software Defect Prediction Systems." 2020 11th

International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020.

[10] Esteves, Geanderson, et al. "Understanding machine learning software defect predictions." Automated Software Engineering 27.3 (2020): 369-392.

[11] Li, Jian, et al. "Software defect prediction via convolutional neural network." 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 2017.

[12] Pan, Tingting, et al. "Learning imbalanced datasets based on SMOTE and Gaussian distribution." Information Sciences 512 (2020): 1214-1233.

[13] Zhang, Hongpo, et al. "An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset." Computer Networks 177 (2020): 107315.

[14] Raghuwanshi, Bhagat Singh, and Sanyam Shukla. "SMOTE based class-specific extreme learning machine for imbalanced learning." Knowledge-Based Systems 187 (2020): 104814.

[15] Chen, Baiyun, et al. "RSMOTE: A self-adaptive robust SMOTE for imbalanced problems with label noise." Information Sciences 553 (2021): 397-428.

[16] Douzas, Georgios, Fernando Bacao, and Felix Last. "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE." Information Sciences 465 (2018): 1-20.

[17] Prabha, C. Lakshmi, and N. Shivakumar. "Software defect prediction using machine learning techniques." 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184). IEEE, 2020.

[18] Rahim, Aqsa, et al. "Software Defect Prediction with Naïve Bayes Classifier." 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST). IEEE, 2021.

[19] Kumar, P. Suresh, et al. "Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature." Innovations in Systems and Software Engineering (2021): 1-25.

[20] Amiripalli, S. S., Kollu, V. V. R., Jaidhan, B. J., Srinivasa Chakravarthi, L., & Raju, V. A. (2020). Performance improvement model for airlines connectivity system using network science. International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 789-792.

[21] Amiripalli, S. S., & Bobba, V. (2020). A Fibonacci based TGO methodology for survivability in ZigBee topologies. International Journal Of Scientific & Technology Research, 9(2), 878-881.

[22] Amiripalli, S. S., Bobba, V., & Potharaju, S. P. (2019). A novel trimet graph optimization (TGO) topology for wireless networks. In Cognitive Informatics and Soft Computing (pp. 75-82). Springer, Singapore.

[23] Amiripalli, S. S., & Bobba, V. (2019). Trimet graph optimization (TGO) based methodology for scalability and survivability in wireless networks. International Journal of Advanced Trends in Computer Science and Engineering, 8(6), 3454-3460.

[24] Amiripalli, S. S., Venkatarao, R., Jitendra, M. S. N. V., & Mycherla, N. M. J. (2020). Detecting emotions of student and assessing the performance by using deep learning. Int J Adv Trends Comput Sci Eng, 9(2), 1641-1645.

[25] Potharaju, S. P., Sreedevi, M., & Amiripalli, S. S. (2019). An ensemble feature selection framework of sonar targets using symmetrical uncertainty and multi-layer perceptron (su-mlp). In Cognitive Informatics and Soft Computing (pp. 247-256). Springer, Singapore.

[26] Thota, J. R., Kothuru, M., & Shanmuk Srinivas, A. Monitoring Diabetes Occurrence Probability Using Classification Technique With A UI.

[27] Jitendra, M. S., Amiripalli, S. S., Kollu, V. V. R., Chowdary, P. R., & Rao, R. V. (2020). Analysis of Airline Connectivity System using Graph Theory. International Journal of Control and Automation, 13(4), 77-84.

[28] Jitendra, M. S., Srinivasu, P. N., Shanmuk Srinivas, A., Nithya, A., & Kandulapati, S. K. (2020). Crack detection on concrete images using classification techniques in machine learning. Journal of Critical Reviews, 7(9), 1236-1241.

[29] Amiripalli, S. S., & Bobba, V. (2020). An Optimal Graph-based ZigBee Mesh for Smart Homes. Journal of Scientific and Industrial Research, Scientific Publishers, 79(4), 318-322.

[30] Naidu, J. L., Gorakala, A. C., & Amiripalli, S. S. (2020). Hash functions and its security for Snags.IRJET,7(7),3465-3471.

[31] Jitendra, M. S., Srinivas, A. S., Surendra, T., Rao, R. V., & Chowdary, P. R. (2021, October). A study on game development using unity engine. In AIP Conference Proceedings (Vol. 2375, No. 1, p. 040001). AIP Publishing LLC.

[32] Kollu, V. V., Amiripalli, S. S., Jitendra, M. S. N. V., & Kumar, T. R. (2021). A Network Science-Based Performance

Improvement Model for the Airline Industry Using NetworkX. International Journal of Sensors Wireless Communications and Control, 11(7), 768-773.

[33] Kollu, V. V., Amiripalli, S. S., & Jitendra, M. S. N. V. (2021). Design and Implementation of an Optimal N-Edge Connected Networks for IOT Based Random Mesh. International Journal of Sensors Wireless Communications and Control, 11(7), 782-788.

[34] Kanakaraju, R., Lakshmi, V., Amiripalli, S. S., Potharaju, S. P., & Chandrasekhar, R. (2021). An Image Encryption Technique Based on Logistic Sine Map and an Encrypted Image Retrieval Using DCT Frequency. In Recent Trends in Intensive Computing (pp. 1-8). IOS Press.

[35] Amiripalli, S. S., Dora, T., Addagarla, S. K., Srinivasu, P. N., & Rao, G. S. (2021). A Graph Invariant-Based TGO Model for RailTel Optical Networks. In Sixth International Conference on Intelligent Computing and Applications (pp. 69-79). Springer, Singapore.

[36] talari, s., amiripalli, s., sirisha, p., kumar, d. s., & deepika, v. k. an improved cipher based automatic theorem proving technique for encryption and decryption.

[37] Jaidhan, B. J., Madhuri, B. D., Pushpa, K., & Devi, B. L. Application of Big Data Analytics and Pattern Recognition Aggregated With Random Forest for Detecting Fraudulent Credit Card Transactions (CCFD-BPRRF).

[38] SRİNİVASU, P. N., NORWAWİ, N., AMİRİPALLİ, S. S., & DEEPALAKSHMİ, P. Secured Compression for 2D Medical Images Through the Manifold and Fuzzy Trapezoidal Correlation Function. Gazi University Journal of Science.

[39] Amiripalli, S. S., Kumar, A. K., & Tulasi, B. (2016, February). Introduction to TRIMET along with its properties and scope. In AIP Conference Proceedings (Vol. 1705, No. 1, p. 020032). AIP Publishing LLC.

[40] Amiripalli, S. S., & Bobba, V. (2019). An optimal TGO topology method for a scalable and survivable network in IOT communication technology. Wireless Personal Communications, 107(2), 1019-1040.

[41] Amiripalli, S. S., & Bobba, V. (2018). Research on network design and analysis of TGO topology. International Journal of Networking and Virtual Organisations, 19(1), 72-86.

[42] Amiripalli, S. S., & Bobba, V. (2019). Impact of trimet graph optimization topology on scalable networks. Journal of Intelligent & Fuzzy Systems, 36(3), 2431-2442.