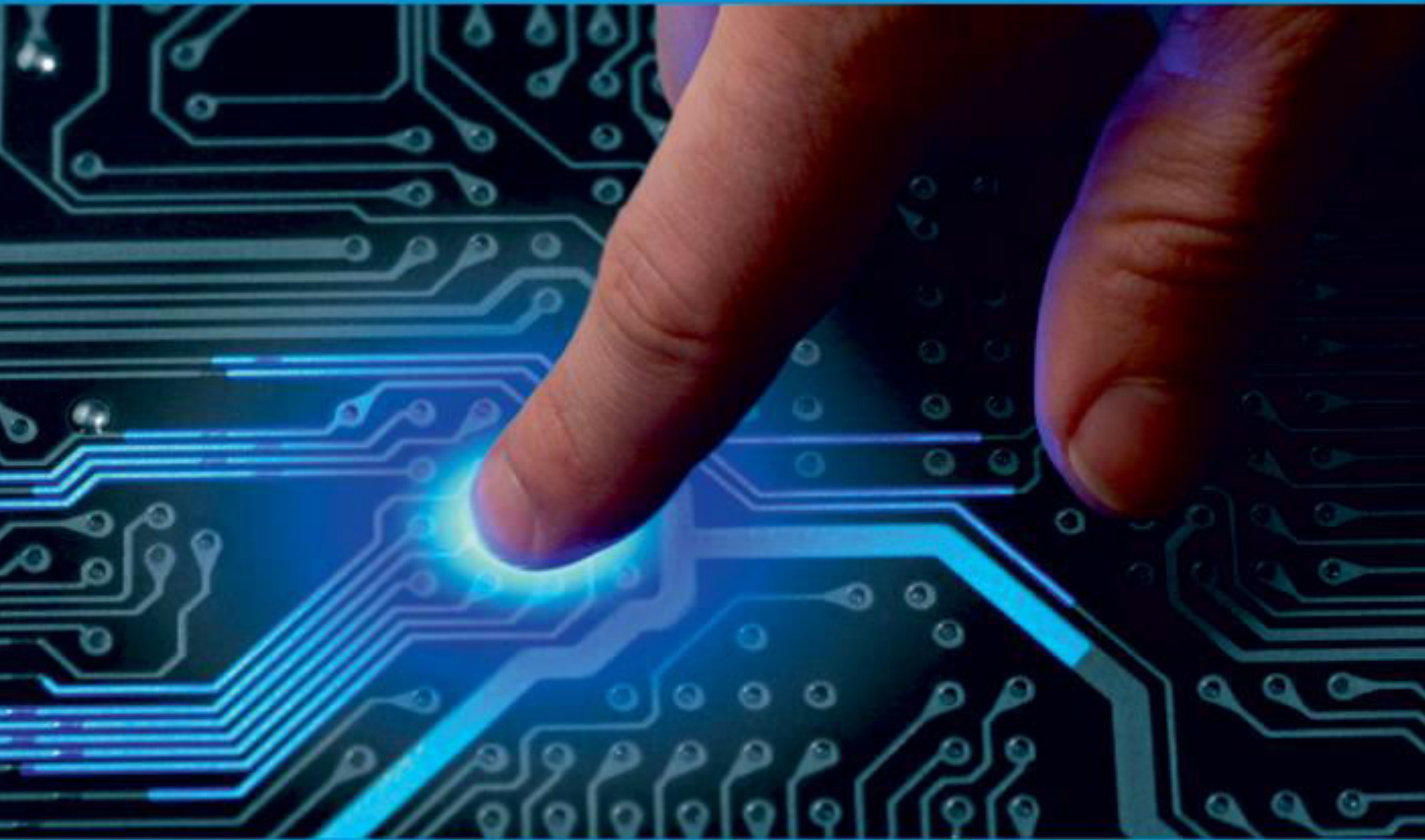
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 8, August 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.625

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Improving Software Fault Prediction with Machine Learning and CNNs

Deshmukh Pankaja Rajebhau, Dr. Satish Narayan Gurjar

Research Scholar, Department of Computer Science, University of Technology, Jaipur, India

Department of Computer Science, University of Technology, Jaipur, India

ABSTRACT: It starts with the collection of historical software metrics and defect logs, followed by rigorous data preprocessing including cleaning, normalization, and splitting into training and test sets. The chapter highlights the importance of feature selection through correlation matrices and distance methods to enhance model accuracy. It compares various machine learning algorithms, ultimately finding the AdaBoostClassifier to have the best performance with an ROC AUC of 0.7511. By leveraging CNNs' hierarchical learning capabilities, the chapter demonstrates a robust framework for improving software fault prediction and addresses both the strengths and limitations of the approach.

Keywords: - Historical metrics, defect logs, preprocessing, feature selection, machine learning, AdaBoost, ROC AUC, CNNs, fault prediction, model accuracy

I.INTRODUCTION

An important part of software engineering is software defect prediction, which seeks to find and correct any potential errors or defects in a software program before it is deployed. This effective strategy lowers development costs, increases overall user satisfaction, and improves software quality. Software vulnerability prediction is the practice of estimating the probability of errors in different sections of code using various methods and measurements [1]. This makes it possible for development teams to focus testing efforts, use resources more wisely, and conduct focused code reviews, all of which contribute to the creation of more reliable and long-lasting software. Among the important factors of software error prediction are described below [2].

Metrics and attributes: Code complexity, code churn, code size, and historical defect data are a few metrics and attributes taken from software code and used by error prediction models. Machine learning algorithms use these measurements as input to detect trends and correlations.

Machine Learning Algorithms: Predictive models are developed using an array of machine learning techniques, including neural networks, decision trees, and support vector machines. These methods allow for the analysis of complex datasets and the identification of patterns that can predict future outcomes [3-4]. using historical data, these models are able to estimate the probability of errors in specific code modules or components.

Data Processing: Data is cleaned and transformed using preprocessing procedures before being fed into machine learning models. In order to maintain the quality and reliability of the model, this may include averaging the dataset, dealing with missing values, and normalizing factors. Training and Testing: This ensures that the model can accurately predict errors in newly written and untested code and helps test its production capabilities [5-6].

Validation and Evaluation: Evaluation criteria such as precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve are used to evaluate the effectiveness of the error prediction models. These metrics help determine the model's ability to identify defective code while minimizing the occurrence of both false positives and false positives.

Development Workflow Integration: Successful error prediction models have been incorporated into the software development process [7-9]. During the quality assurance phase, development teams use forecasts to systematically allocate resources, focus on high-risk regions, and prioritize testing efforts.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Continuous Improvement: The error prediction method is iterative. Models are often retrained and improved to ensure continued efficiency in the face of changing conditions as new data becomes available and the system evolves.

Teams can reduce the chance that bugs will affect end users and improve software quality by incorporating these predictions into the development process [10].

1.1 Evaluation Phase of the SDLC

Software Defect Prediction (SDP) plays an important role during the Evaluation Phase of the Software Development Life Cycle (SDLC). It helps to identify modules that may have errors and therefore need a thorough test. This allows for efficient use of resources while staying within project constraints. However, despite its usefulness, predicting which modules will present problems can be a challenge. Defect Prediction models come with various problems that can be difficult to solve [11-15]. Development teams can identify where they are working and can increase industrial results and reduce development errors by using predictive software defects. In order to detect errors and organize the testing process, it is possible to predict code segments that are likely to have errors. Accurate prognosis is important for early diagnosis of the disease in its early stages [16]. The main purpose of several software testers is to anticipate problems. Software bugs are expected to cost billions of pounds to detect and fix every year. It is expected that automated assistance to accurately predict defect areas and guide inspectors' work will significantly reduce these costs [17].

II.RESEARCH METHODOLOGY

Using machine learning to optimize software failure prediction in an object-oriented paradigm entails a number of crucial stages. First, relevant software characteristics like size, coupling, and code complexity are collected as data. Next, feature selection is done to determine which metrics are most informative. Next, a machine learning model is chosen in accordance with the attributes of the issue. Furthermore, methods such as model stacking or ensemble learning may be used to enhance prediction accuracy. Lastly, the improved model is put to use for predicting software faults in real-world scenarios, continually assessed, and adjusted as needed. This technique uses machine learning to maximize software failure prediction in an object-oriented paradigm by integrating data pretreatment, model selection, training, assessment, and deployment [18-20].

The model that is offered suggests that it is possible to predict a task's failure using scientific approaches. Data flows and task dependencies serve as essential illustrations of processes or computations within scientific applications. By employing advanced machine learning algorithms for failure prediction, it becomes possible to proactively analyze data from multiple scientific workflows, thereby mitigating the impact of failures on these workflow operations while optimizing Cloud resources. This proactive approach involves real-time data analysis. Task failures within the scheduling of scientific workflows can stem from various factors, such as resource overutilization or underutilization, exceeding execution time or cost thresholds, incorrect library installations, insufficient memory or disk space, and similar occurrences. The primary focus of this study's proposed paradigm is understanding task failures (related to CPU, RAM, disk storage, and network bandwidth) caused by overutilization of resources. The goal of the approach described here is to develop a model that can monitor data related to scientific operations in real-time and identify issues at work. The suggested approach analyzes a large number of processes that have been stored in cloud repositories in order to spot any problems with the process before they happen. The suggested model is based on experimental results and employs the machine learning strategy that has shown to be the most effective in failure prediction. Figure 1 illustrates the flowchart outlining defect prediction methods tailored for optimizing resources in a cloud computing environment. The fault prediction technique encompasses three primary phases: first, employing the PCA technique to select features from the input dataset; second, classifying the data using Naive Bayes, Random Forest, and linear regression algorithms; and finally, predicting failures as the concluding step.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

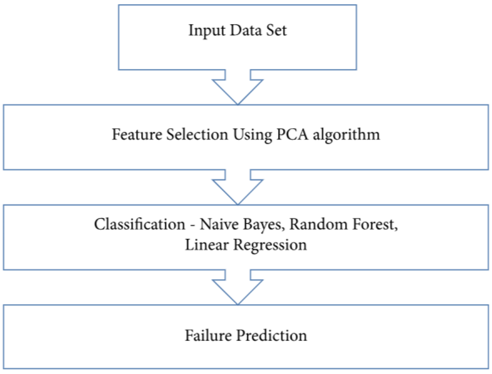


Fig 2. 1: *Fault prediction techniques in cloud computing aim to enhance resource optimization.*

III.SIMULATION AND RESULT

This Chapter 4 presented a "Simulation and Results," employs convolutional neural networks (CNNs) within a deep learning framework to construct a software fault prediction model. The chapter focuses on developing an effective Python-based approach for identifying and forecasting software issues. It begins with comprehensive data gathering, including software metrics and defect logs, followed by preprocessing steps like data cleaning, handling missing values, and normalization. Feature engineering is then utilized to select relevant characteristics that enhance fault prediction accuracy. Leveraging CNNs' ability to learn hierarchical representations from software inputs, the chapter trains and evaluates the CNN model using metrics such as accuracy, precision, recall, and F1-score. It critically assesses the CNN model's efficacy in predicting software flaws, offering insights into both its strengths and limitations, thereby providing a solid foundation for advancing software fault prediction using deep learning methodologies.

1.1 Software Fault Prediction Procedure

a. Steps and Methods

Data Collection

o Collected historical software project data including metrics such as code complexity (e.g., loc, cyclomatic complexity), Halstead metrics (e.g., volume, difficulty), and defect indicators.

o Sample data metrics:
- loc: McCabe's line count of code
- v(g): Cyclomatic complexity
- n: Halstead total operators + operands
- defects: Indicator of whether a module has reported defects/

Data Preprocessing

o Cleaned and normalized the data.

o Split data into training and test sets.

Feature Selection

o Employed correlation matrices and feature distance methods to identify relevant features.

o Example of correlation matrix:
- Removed redundant or irrelevant features to improve model performance.

Model Building
- Used various machine learning algorithms including Logistic Regression, Decision Trees, Random Forest, and ensemble methods.
- Example of building and training a model

Model Evaluation
- Evaluated models using metrics like ROC AUC, Accuracy, Precision, Recall, and F1-score.
- Key results:

o **Logistic Regression:** ROC AUC score: 0.6087

o **Decision Tree:** ROC AUC score: 0.6165

© 2024 IJIRCCE | Volume 12, Issue 8, August 2024| DOI: 10.15680/IJIRCCE.2024.1208008

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- **AdaBoostClassifier:** ROC AUC score: 0.7511 (best performance)

1.2 Derived Result

With a ROC AUC value of 0.7511, the AdaBoostClassifier showed the greatest performance for SFP. The predicted accuracy of the model was much enhanced by feature selection, which concentrated on the most important metrics. The Decision Tree model demonstrated almost flawless accuracy.

1.2.1 Software Simulation Details in Python (Objective 1)

In order to enhance software quality and minimize testing requirements, software fault prediction seeks to detect defective software modules early in the software development process. By identifying pertinent characteristics and eliminating superfluous or unnecessary ones, feature selection approaches are essential for improving the performance of fault prediction models. An extensive summary of feature selection-based software failure prediction is provided below

Software Fault Prediction

Data Collection: Gather data from prior software projects, which usually consists of measures like code complexity, churn, developer activity, and past fault history.

```
data = pd.read_csv('train.csv')
origin = pd.read_csv('jm1.csv')
test = pd.read_csv('test.csv')
sample_submission = pd.read_csv('sample_submission.csv')
```

1. loc : numeric % McCabe's line count of code
2. v(g) : numeric % McCabe "cyclomatic complexity"
3. ev(g) : numeric % McCabe "essential complexity"
4. iv(g) : numeric % McCabe "design complexity"
5. n : numeric % Halstead total operators + operands
6. v : numeric % Halstead "volume"
7. l : numeric % Halstead "program length"
8. d : numeric % Halstead "difficulty"
9. i : numeric % Halstead "intelligence"
10. e : numeric % Halstead "effort"
11. b : numeric % Halstead
12. t : numeric % Halstead's time estimator
13. lOCode : numeric % Halstead's line count
14. lOComment : numeric % Halstead's count of lines of comments
15. lOBlank : numeric % Halstead's count of blank lines
16. lOCodeAndComment: numeric
17. uniq_Op : numeric % unique operators
18. uniq_Opnd : numeric % unique operands
19. total_Op : numeric % total operators
20. total_Opnd : numeric % total operands

21: branchCount : numeric % of the flow graph

22. defects : {false,true} % module has/has not one or more reported defects

© 2024 IJIRCCE | Volume 12, Issue 8, August 2024| DOI: 10.15680/IJIRCCE.2024.1208008

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Data Preprocessing

```
from sklearn import preprocessing

scale_v = data[['v']]
scale_b = data[['b']]

minmax_scaler = preprocessing.MinMaxScaler()

v_scaled = minmax_scaler.fit_transform(scale_v)
b_scaled = minmax_scaler.fit_transform(scale_b)

data['v_ScaledUp'] = pd.DataFrame(v_scaled)
data['b_ScaledUp'] = pd.DataFrame(b_scaled)

data
```

```
desc = pd.DataFrame(index = data.columns)
desc['count'] = data.count()
desc['nunique'] = data.nunique()
desc['%unique'] = desc['nunique'] / len(data) * 100
desc['null'] = data.isnull().sum()
desc['type'] = data.dtypes
desc = pd.concat([desc, data.describe().T], axis = 1)
desc
```

	count	nunique	%unique	null	type	count	mean	std	min	25%	50%	75%	max
id	101763	101763	100.000000	0	int64	101763.0	50881.000000	29376.592059	0.0	25440.50	50881.00	76321.50	101762.00
loc	101763	378	0.371451	0	float64	101763.0	37.347160	54.600401	1.0	13.00	22.00	42.00	3442.00
v(g)	101763	106	0.104164	0	float64	101763.0	5.492684	7.900855	1.0	2.00	3.00	6.00	404.00
ev(g)	101763	71	0.069770	0	float64	101763.0	2.845022	4.631262	1.0	1.00	1.00	3.00	165.00
iv(g)	101763	84	0.082545	0	float64	101763.0	3.498826	5.534541	1.0	1.00	2.00	4.00	402.00
n	101763	836	0.821517	0	float64	101763.0	96.655995	171.147191	0.0	25.00	51.00	111.00	8441.00
v	101763	4515	4.436780	0	float64	101763.0	538.280956	1270.791601	0.0	97.67	232.79	560.25	80843.08
l	101763	55	0.054047	0	float64	101763.0	0.111634	0.100096	0.0	0.05	0.09	0.15	1.00
d	101763	3360	3.301789	0	float64	101763.0	13.681881	14.121306	0.0	5.60	9.82	18.00	418.20
i	101763	5171	5.081415	0	float64	101763.0	27.573007	22.856742	0.0	15.56	23.36	34.34	569.78
e	101763	8729	8.577774	0	float64	101763.0	20853.589876	190571.405427	0.0	564.73	2256.23	10193.24	16846621.12
b	101763	315	0.309543	0	float64	101763.0	0.179164	0.421844	0.0	0.03	0.08	0.19	26.95
t	101763	8608	8.458870	0	float64	101763.0	1141.357982	9862.795472	0.0	31.38	125.40	565.92	935923.39
lOCode	101763	298	0.292837	0	int64	101763.0	22.802453	38.541010	0.0	7.00	14.00	26.00	2824.00
lOComment	101763	91	0.089423	0	int64	101763.0	1.773945	5.902412	0.0	0.00	0.00	1.00	344.00
lOBlank	101763	94	0.092371	0	int64	101763.0	3.979865	6.382358	0.0	1.00	2.00	5.00	219.00
locCodeAndComment	101763	29	0.028498	0	int64	101763.0	0.196604	0.998906	0.0	0.00	0.00	0.00	43.00
uniq_Op	101763	70	0.068787	0	float64	101763.0	11.896131	6.749549	0.0	8.00	11.00	16.00	410.00
uniq_Opnd	101763	176	0.172951	0	float64	101763.0	15.596671	18.064261	0.0	7.00	12.00	20.00	1026.00
total_Op	101763	623	0.612207	0	float64	101763.0	57.628116	104.537660	0.0	15.00	30.00	66.00	5420.00
total_Opnd	101763	485	0.476598	0	float64	101763.0	39.249698	71.692309	0.0	10.00	20.00	45.00	3021.00
branchCount	101763	144	0.141505	0	float64	101763.0	9.839549	14.412769	1.0	3.00	5.00	11.00	503.00
defects	101763	2	0.001965	0	bool	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Feature Selection: Feature selection involves identifying and selecting a subset of relevant features that contribute the most to the prediction model. This can be achieved using various techniques:

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
fig, ax = plt.subplots(7, 3, figsize = (15, 25), dpi = 300)
ax = ax.flatten()

for i, column in enumerate(col_list):

    sns.kdeplot(data[column], ax=ax[i], color=pal[0])
    sns.kdeplot(test[column], ax=ax[i], color=pal[2])

    ax[i].set_title(f'{column} Distribution', size = 14)
    ax[i].set_xlabel(None)

fig.suptitle('Distribution of Feature\nper Dataset\n', fontsize = 24, fontweight = 'bold')
fig.legend(['Train', 'Test'])
plt.tight_layout()
```

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

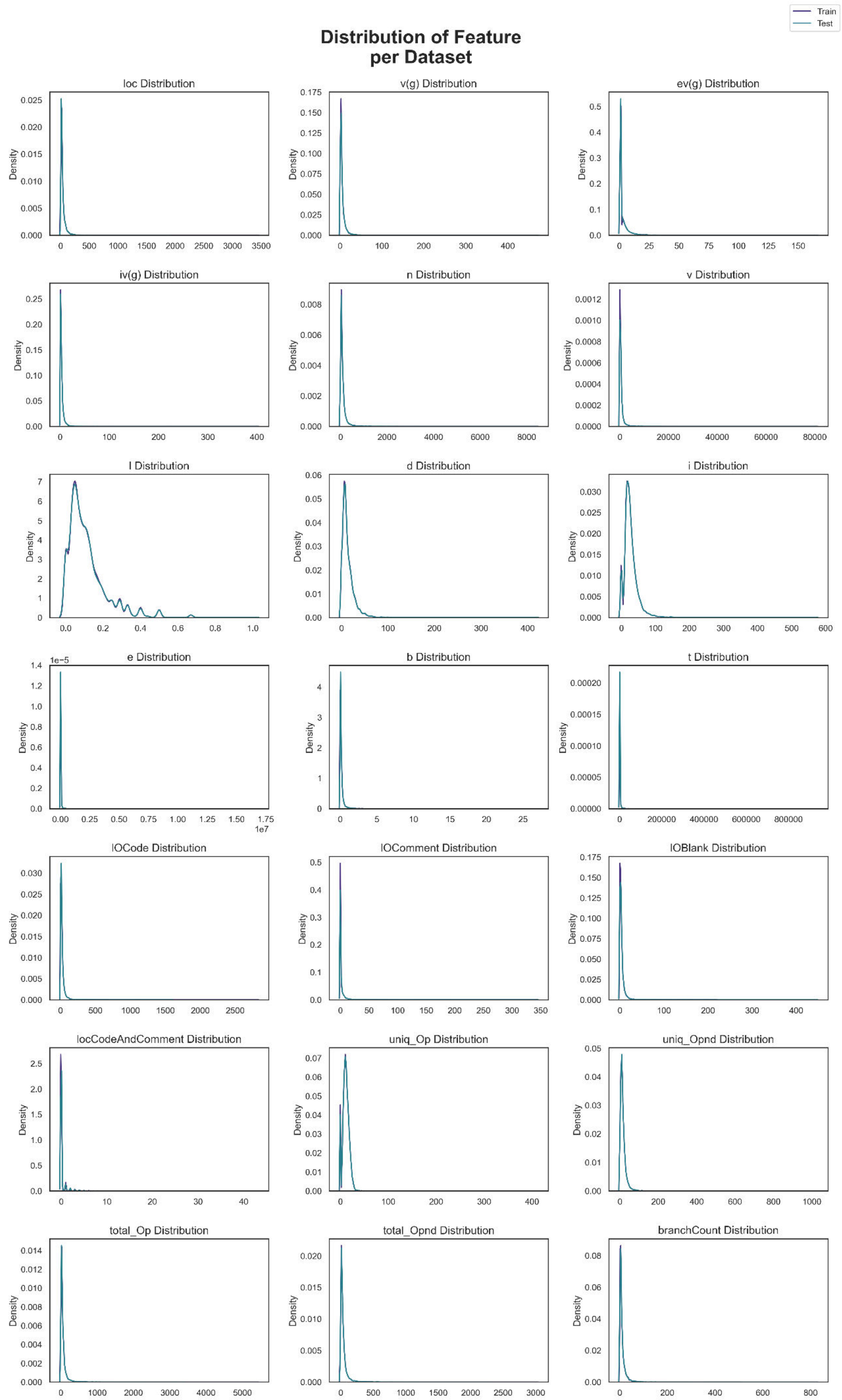


Fig 3. 1: Correrelation Matrix

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Correrelation Matrix
heatmap(data.drop('id', axis = 1), 'Train')
heatmap(test.drop('id', axis = 1), 'Test')

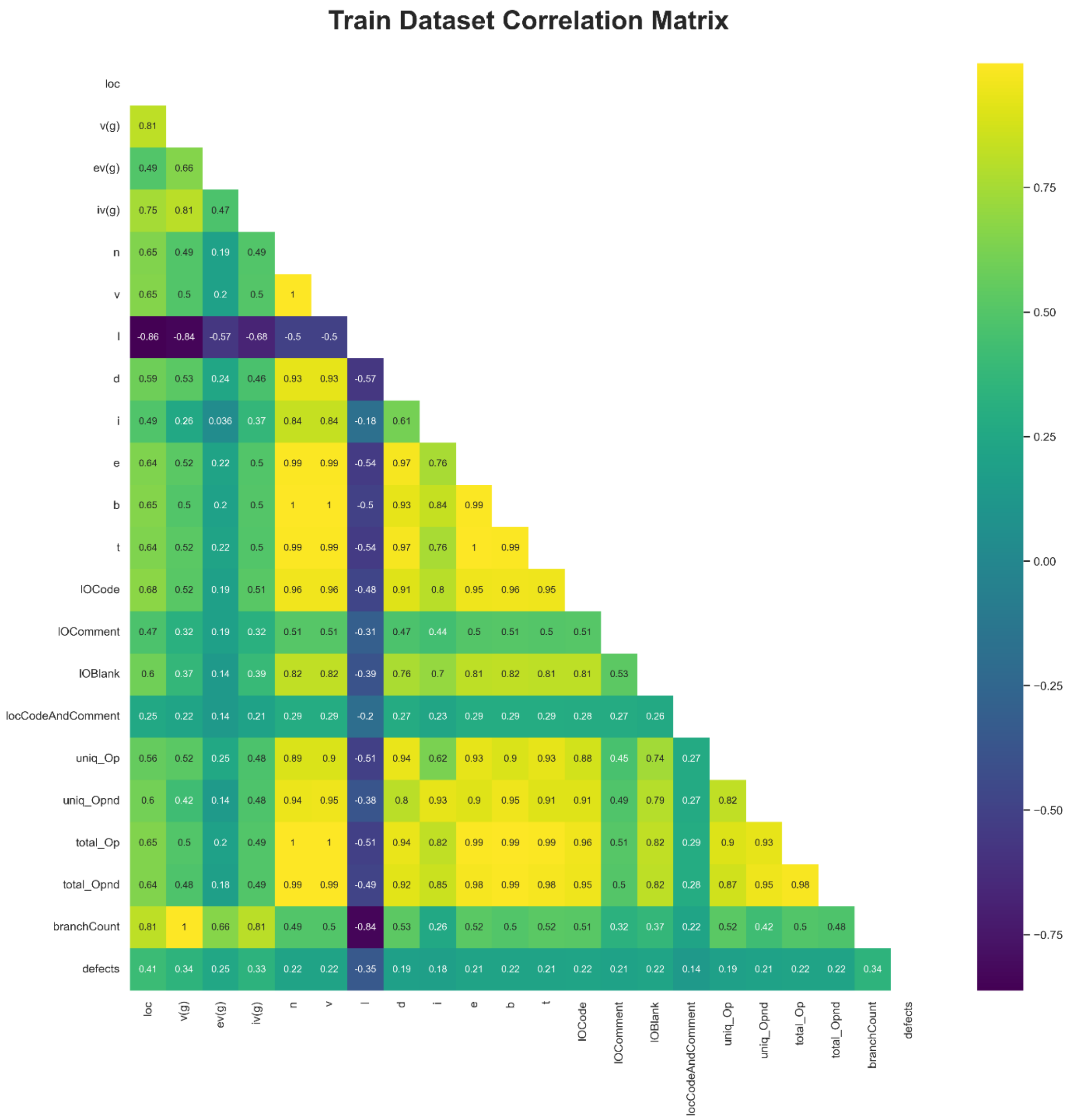


Fig 3. 2: Feature distance

IV.CONCLUSION AND FUTURE SCOPE

It begins with data collection from historical software projects, including various complexity and defect metrics, followed by rigorous preprocessing steps such as data cleaning, normalization, and feature selection. The chapter explores different machine learning algorithms including Logistic Regression, Decision Trees, and Ada Boost Classifier, with Ada Boost Classifier achieving the highest performance (ROC AUC of 0.7511) for fault prediction. Through critical evaluation, the chapter demonstrates the effectiveness of CNNs and feature selection in enhancing

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

software quality and reducing testing requirements, providing a solid foundation for advancing fault prediction methodologies.

REFERENCES

1. Wahono, R. S. (2015). A systematic literature review of software defect prediction. *Journal of software engineering*, *1*(1), 116.
2. Rawat, M. S., & Dubey, S. K. (2012). Software defect prediction models for quality improvement: a literature study. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), 288.
3. Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, *58*, 388402.
4. Okutan, A., & Yıldız, O. T. (2014). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, *19*, 154181.
5. Nam, J. (2014). Survey on software defect prediction. *Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Tech. Rep*.
6. He, P., Li, B., Liu, X., Chen, J., & Ma, Y. (2015). An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, *59*, 170190.
7. Jing, X. Y., Ying, S., Zhang, Z. W., Wu, S. S., & Liu, J. (2014, May). Dictionary learning based software defect prediction. In *Proceedings of the 36th international conference on software engineering* (pp. 414423).
8. Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, *62*(2), 434443.
9. Ma, Y., Luo, G., Zeng, X., & Chen, A. (2012). Transfer learning for crosscompany software defect prediction. *Information and Software Technology*, *54*(3), 248256.
10. Li, M., Zhang, H., Wu, R., & Zhou, Z. H. (2012). Samplebased software defect prediction with active and semisupervised learning. *Automated Software Engineering*, *19*, 201230.
11. Sun, Z., Song, Q., & Zhu, X. (2012). Using codingbased ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 18061817.
12. Arora, I., Tetarwal, V., & Saha, A. (2015). Open issues in software defect prediction. *Procedia Computer Science*, *46*, 906912.
13. Wang, T., & Li, W. H. (2010, December). Naive bayes software defect prediction model. In *2010 International conference on computational intelligence and software engineering* (pp. 14). Ieee.
14. Shepperd, M., Bowes, D., & Hall, T. (2014). Researcher bias: The use of machine learning in software defect prediction. *IEEE Transactions on Software Engineering*, *40*(6), 603616.
15. Czibula, G., Marian, Z., & Czibula, I. G. (2014). Software defect prediction using relational association rule mining. *Information Sciences*, *264*, 260278.
16. Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., & Riquelme, J. C. (2014, May). Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (pp. 110).
17. Ren, J., Qin, K., Ma, Y., & Luo, G. (2014). On software defect prediction using machine learning. *Journal of Applied Mathematics*, *2014*.
18. Arar, Ö. F., & Ayan, K. (2015). Software defect prediction using costsensitive neural network. *Applied Soft Computing*, *33*, 263277.
19. Yang, X., Tang, K., & Yao, X. (2014). A learningtorank approach to software defect prediction. *IEEE Transactions on Reliability*, *64*(1), 234246.
20. Wang, H., Khoshgoftaar, T. M., & Napolitano, A. (2010, December). A comparative study of ensemble feature selection techniques for software defect prediction. In *2010 Ninth International Conference on Machine Learning and Applications* (pp. 135140). IEEE.
21. Nevendra, M., & Singh, P. (2021). Software defect prediction using deep learning. *Acta Polytechnica Hungarica*, *18*(10), 173-189.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

22. Batool, I., & Khan, T. A. (2022). Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review. *Computers and Electrical Engineering*, *100*, 107886.
23. Batool, I., & Khan, T. A. (2023). Software fault prediction using deep learning techniques. *Software Quality Journal*, *31*(4), 1241-1280.

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com