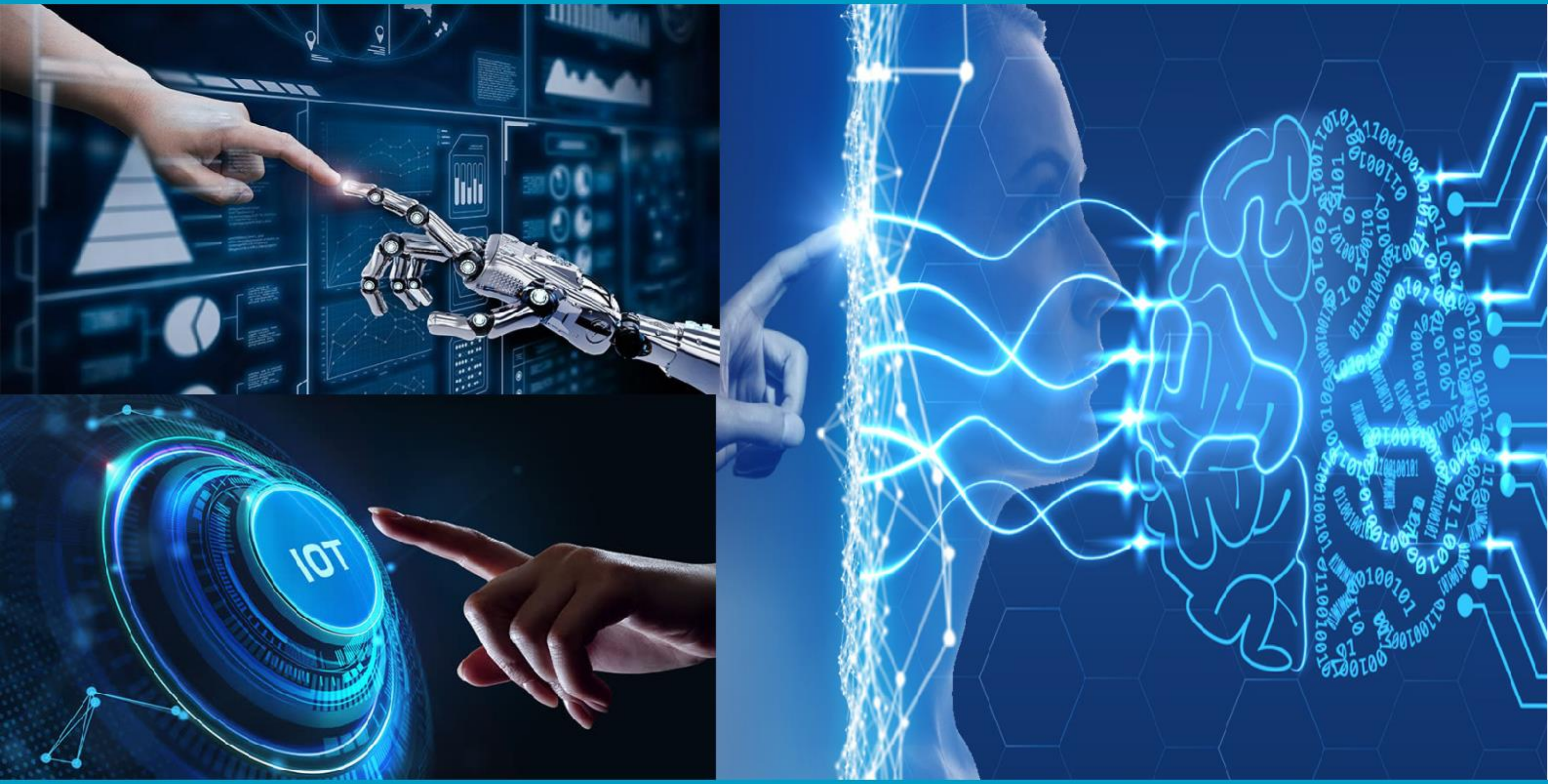
IJMRSETM

e-ISSN: 2395 - 7639



INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

Volume 11, Issue 5, May 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.802

+91 99405 72462 +9163819 07438 ijmrsetm@gmail.com www.ijmrsetm.com

International Journal of Multidisciplinary Research in Science, Engineering, Technology & Management (IJMRSETM)

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 5, May 2024 |

Network Traffic Cyber-Attacks Classification Using Supervised Machine Learning Model

Ms. Pavithra B [1], M.Dharani Kumar[2], J.Fino Franklin[3], Frino Fredy[4]

Assitant Professor, Department of Computer Science and Engineering, Velammal Institute of Technology, Panchetti, Chennai, Tamil Nadu, India[1]

UG Scholar, Department of Computer Science and Engineering, Velammal Institute of Technology, Panchetti, Chennai, Tamil Nadu, India [2 3 4]

ABSTRACT: Cyberattack classification through the utilization of supervised machine learning methods. The system is designed to categorize diverse cyber-attacks by employing a meticulously curated dataset encompassing a wide array of attack types, including but not limited to malware, phishing, and distributed denial-of-service (DDoS) attacks. Feature extraction techniques are applied to both network traffic data and behavioural attributes, facilitating the training of a robust classification model. Various supervised learning algorithms, such as decision trees, support vector machines, and neural networks, are evaluated for their efficacy in accurately predicting attack categories. The training process involves labelling historical attack instances, enabling the model to discern intricate patterns and subtle differentiators among attack types. Regular model updates and retraining with new attack data ensure its relevance in dynamically evolving threat landscapes. The system's predictive accuracy empowers cybersecurity teams to swiftly identify and respond to cyber threats, thereby bolstering overall defense strategies. Through this research, we contribute to the proactive identification and mitigation of cyber-attacks, ultimately fortifying digital security frameworks.

KEYWORDS: Cyberattack, Feature extraction, machine learning.

I. INTRODUCTION

The classification of cyber-attacks through supervised machine learning techniques is a pivotal aspect of modern cybersecurity. As the digital realm becomes progressively intricate, cyber threats grow in sophistication and frequency. Thus, the ability to swiftly and accurately categorize these threats is paramount. Supervised machine learning offers a powerful solution by leveraging labelled datasets to teach algorithms to recognize and classify different types of cyber-attacks. This classification aids organizations in responding effectively, mitigating damage, and fortifying their defences. However, challenges such as the diversity of attack methods, the adaptability of attackers, and imbalanced data make this a complex field. Nevertheless, the potential applications are extensive, spanning intrusion detection, email filtering, malware identification, and anomaly detection. Looking ahead, ongoing research will refine models to handle evolving threats, integrate them with broader security strategies, and address ethical concerns in the deployment of these technologies.

II. LITERATURE REVIEW

Literature research is the most important step in the software development process. Before creating a tool, it is important to determine the time factor, profitability, and company strengths. With these in place, the next 10 steps are to decide which operating systems and languages you can use to develop your tools. Once programmers start building tools, they need a lot of external support. This support can come from experienced programmers, books, or websites. The above evaluations will be considered in the development of the proposed system before building the system.

Wentao Zhao, Jianping Yin, Jun Long, "A Prediction Model of DoS Attack's Distribution Discrete Probability ", 2008.
This paper describes the clustering problem first, and then utilizes the genetic algorithm to implement the optimization of clustering methods. Based on the optimized clustering on the sample data, we get various categories of the relation between traffics and attack amounts, and then builds up several prediction sub-models about DoS attack. Furthermore, according to the Bayesian method, we deduce discrete probability calculation about each sub-model and then get the distribution discrete probability prediction model for DoS attack.

Xiaoyong Yuan, Pan He, Qile Zhu, " Adversarial Examples: Attacks and Defenses for Deep Learning", 2019.
With rapid progress and significant successes in a wide spectrum of applications, deep learning is being applied in many safety-critical environments. However, deep neural networks (DNNs) have been recently found vulnerable to well-designed input samples called adversarial examples. Adversarial perturbations are imperceptible to human but can easily fool DNNs in the testing/deploying stage. The vulnerability to adversarial examples becomes one of the major risks for applying DNNs in safety-critical environments. Therefore, attacks and defenses on adversarial examples draw great attention. In this paper, we review recent findings on adversarial examples for DNNs, summarize the methods for generating adversarial examples, and propose a taxonomy of these methods. Under the taxonomy, applications for adversarial examples are investigated. We further elaborate on countermeasures for adversarial examples. In addition, three major challenges in adversarial examples and the potential solutions are discussed

Preetish Ranjan, Abhishek Vaish, "Apriori Viterbi Model for Prior Detection of Socio-Technical Attacks in a Social Network", 2014.
Social network analysis is a basic mechanism to observe the behavior of a community in society. In the huge and complex social network formed using cyberspace or telecommunication technology, the identification or prediction of any kind of socio-technical attack is always difficult. This challenge creates an opportunity to explore different methodologies, concepts and algorithms used to identify these kinds of community on the basis of certain pattern, properties, structure and trend in their linkage. This paper tries to find the hidden information in huge social network by compressing it in small networks through apriori algorithm and then diagnosed using viterbi algorithm to predict the most probable pattern of conversation to be followed in the network and if this pattern matches with the existing pattern of criminals, terrorists and hijackers then it may be helpful to generate some kind of alert before crime.

Seraj Fayyad, Cristoph Meinel, "New Attack Scenario Prediction Methodology", 2013
Intrusion detection system generates significant data about malicious activities run against network. Generated data by IDS are stored in IDS database. This data represent attacks scenarios history against network. Main goal of IDS system is to enhance network defense technologies. Other techniques are also used to enhance the defense of network such as Attack graph. Network attack graph are used for many goals such as attacker next attack step prediction. In this paper we propose a real time prediction methodology for predicting most possible attack steps and attack scenarios. Proposed methodology benefits from attacks history against network and from attack graph source data. it comes without considerable computation overload such as checking of attack plans library. It provides parallel prediction for parallel attack scenarios.

Sundari S, Ananthi M, "Secure Cloud Storage with Deduplication Technique", 2015.
Utilizing cloud information clients might move information from their PC frameworks to cloud servers. Thus, the client won't have any weight of upkeep and he gets top-notch information capacity administrations. Numerous security issues are worried to distributed storage. Cloud specialist co-ops or capacity servers are not reliable. The client is worried about whether the data put away on the cloud are set up or not turns. This paper depends on homomorphic hash calculation. Further, this supports dynamic activities like supplementing, updating, erasing, and adjusting at the block level, for information elements Merkle Hash Tree is utilized which assists with tracking down the area of every powerful activity. The outsider reviewer checks the client's information for rightness and gives the precision of the information that is put away in the cloud server. The correspondence and calculation above are diminished. The deduplication [12] procedure is utilized to check whether the document that the client needs to store in distributed storage is as of now exists on a cloud server or not.

III. METHODOLOGY

The machine learning workflow begins with data collection, gathering relevant data from various sources and possibly annotating it. Following this, the raw data undergoes preprocessing to clean, normalize, and engineer features for modeling. The dataset is then split into training, validation, and testing sets for effective model training and evaluation. Feature selection and engineering further enhance the model's performance by identifying relevant features and creating new ones. Model building involves selecting algorithms, training models, and optimizing hyperparameters. Evaluation metrics such as accuracy and F1 score are used to assess model performance, followed by testing on separate datasets to gauge generalization ability. Deploying satisfactory models into production environments enables real-world predictions, while continuous monitoring and maintenance ensure sustained performance. A feedback loop facilitates model improvement based on real-world usage, while documentation and reporting track the entire process, informing stakeholders and supporting decision-making.

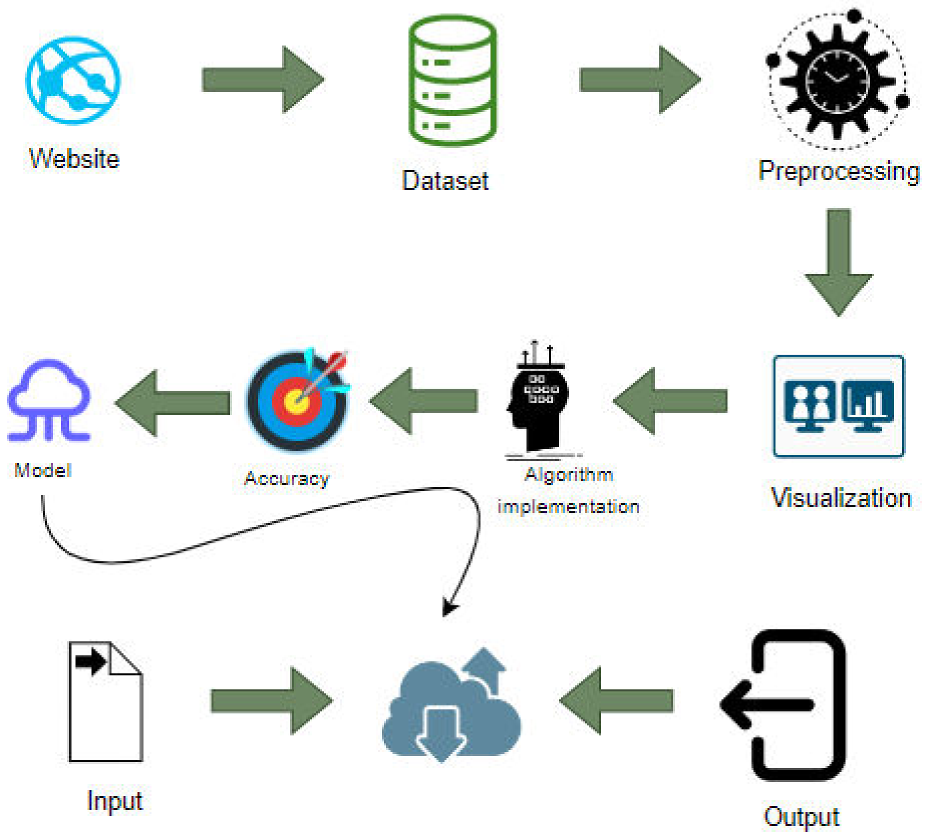


Fig1. The architecture of the proposed system

In machine learning, validation techniques play a crucial role in assessing the error rate of models, especially when dealing with datasets that may not fully represent the population. Real-world scenarios often involve working with data samples that lack complete representativeness, necessitating validation techniques for unbiased evaluation. Tasks like handling missing values, duplicates, and understanding data types are essential in data preprocessing, ensuring the data's suitability for model training and tuning. Python libraries like Pandas expedite data cleaning processes, allowing more time for exploration and modeling.

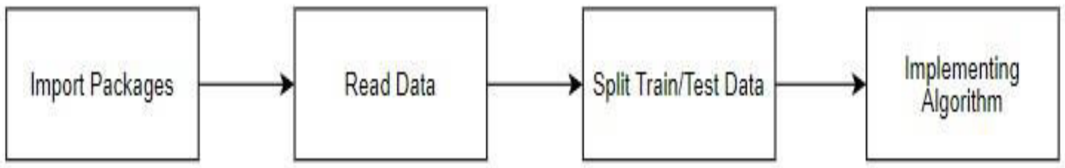


Fig2. Module Diagram

Understanding the sources of missing data aids in devising appropriate imputation strategies, while variable identification through uni-variate, bi-variate, and multi-variate analyses provides insights into data properties. Visualizing data using charts and plots aids in understanding patterns and outliers, crucial for exploratory data analysis. Algorithm implementation involves comparing the performance of multiple models, using metrics like accuracy, precision, recall, and F1-score. AdaBoost and CatBoost are notable algorithms, each offering unique strengths such as boosting weak learners and efficient handling of categorical features, respectively. Hyperparameter tuning and regularization techniques further enhance model performance, ensuring scalability and robustness to overfitting. Overall, validation, data preprocessing, visualization, algorithm implementation, and performance metrics are integral components of a comprehensive machine learning workflow, driving informed decision-making and model optimization.

IV. RESULTS AND DISCUSSION

USER REGISTER:
The register module provides a conceptual framework for entering data on that user in a way that: eases data entry & accuracy by matching the user entry to the data source (usually paper files created at the point of care), ties easily back to individual user records to connect registers to user data, and collects data elements to enable better supervision of donation programs.

USER LOGIN:

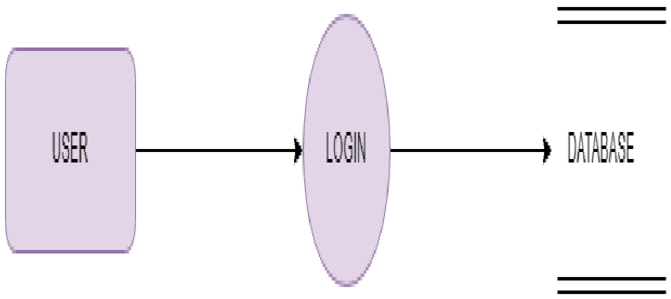


Fig3. User Login Diagram

This module in our project here symbolizes a unit of work performed within a database management system (or similar system) against a database and treated coherently and reliably independently of other transactions. A transaction generally represents any change in the database user will transfer the amount to the provider.

USER INPUT NETWORK LOGS:
Once logged in, users can input network logs retrieved from various sources. These logs contain valuable information about network activities, including traffic patterns, communication protocols, and potential security incidents.

MACHINE LEARNING ANALYSIS:
The system leverages a supervised machine learning model to analyze the network logs provided by the user. This model employs advanced algorithms to detect patterns, anomalies, and potential cyber threats within the network data.

PREDICTED ATTACK CATEGORY:
Based on the analysis of network logs, the machine learning model predicts the category of cyber attack present in the network. Common attack categories include malware infections, DDoS attacks, phishing attempts, and unauthorized access.
OUTPUT PRECAUTIONS AND DETAILS:
Upon predicting the attack category, the system generates detailed information and precautionary measures for the user. This includes guidance on how to respond to the specific type of attack, recommended security protocols, and steps to mitigate potential risks.

USER RESPONSE AND MITIGATION:
Armed with the information provided by the system, users can take proactive measures to mitigate the identified cyber threat. This may involve implementing security patches, updating firewall configurations, or enhancing network monitoring protocols.

INCIDENT REPORTING:
In addition to providing guidance on mitigation strategies, the system allows users to generate incident reports detailing the detected cyber threats. These reports capture relevant information about the attack, including timestamps, affected systems, and severity levels.

ADMINISTRATIVE TOOLS:
Administrators have access to a suite of administrative tools for managing user accounts, monitoring system performance, and overseeing incident response efforts. These tools enable administrators to maintain the integrity and security of the system infrastructure.

AUDIT LOGGING AND COMPLIANCE:
To ensure accountability and compliance with regulatory standards, the system logs all user activities and interactions. These audit logs provide a comprehensive record of system events, user actions, and security incidents for review and analysis.

CONTINUOUS IMPROVEMENT:
The system incorporates mechanisms for continuous improvement, including feedback loops, model retraining, and performance optimization. By continuously refining its algorithms and capabilities, the system enhances its effectiveness in detecting and mitigating cyber threats over time.

ALGORITHM

Selecting the appropriate algorithm for a specific machine learning task is a critical step in the model development process. It requires careful consideration of various factors, including the nature of the data, the complexity of the problem, and the desired outcomes. Here's a structured approach to algorithm selection:

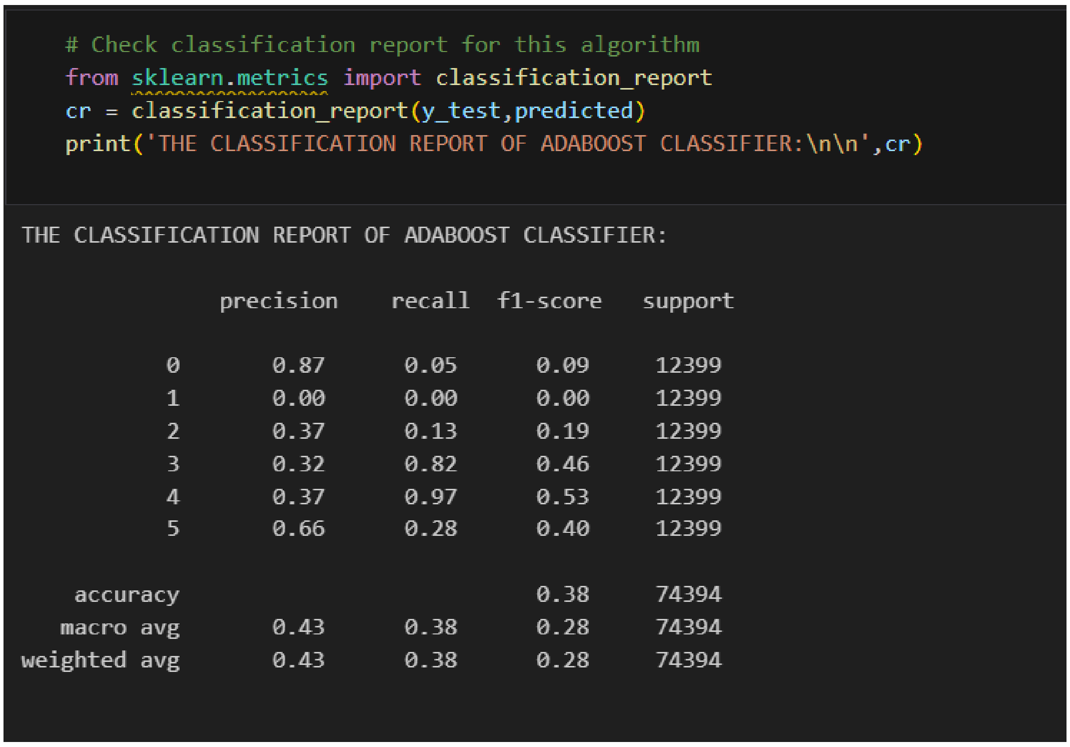
PERFORMANCE METRICS EVALUATION:
In machine learning, evaluating the performance of a model is essential to assess its effectiveness in solving the target problem. Performance metrics provide quantitative measures of how well the model performs on unseen data. Here are some commonly used performance metrics:

- **Accuracy:**
  - Accuracy measures the proportion of correctly predicted instances out of the total instances in the dataset. It is calculated as the ratio of the number of correct predictions to the total number of predictions.
- **Precision:**
  - Precision measures the proportion of true positive predictions among all positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives.
- **Recall (Sensitivity):**
  - Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly identified by the model. It is calculated as the ratio of true positives to the sum of true positives and false negatives.
- **F1 Score:**
  - The F1 score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. It is calculated as the weighted average of precision and recall, where a higher F1 score indicates better overall performance.
- **Specificity:**
  - Specificity measures the proportion of actual negative instances that were correctly identified as negative by the model. It is calculated as the ratio of true negatives to the sum of true negatives and false positives.
- **ROC Curve and AUC:**
  - The receiver operating characteristic (ROC) curve is a graphical representation of the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) at various threshold settings. The area under the ROC curve (AUC) quantifies the model's ability to distinguish between positive and negative classes, with higher values indicating better discrimination performance.
- **Confusion Matrix:**
  - A confusion matrix is a tabular representation of the model's predictions versus the actual ground truth labels. It provides insights into the true positives, true negatives, false positives, and false negatives generated by the model.
- By comprehensively evaluating these performance metrics, data scientists can gain a deeper understanding of their model's strengths and weaknesses, enabling them to make informed decisions about model selection, optimization, and deployment.

Adaboost Algorithm:
Adaboost, short for Adaptive Boosting, is a popular ensemble learning algorithm used for classification and regression tasks. It works by combining multiple weak learners (often decision trees) to create a strong classifier. Here are some key points about Adaboost:

Boosting Technique:
Adaboost is a boosting technique where each subsequent model in the ensemble focuses on the instances that were misclassified by the previous models. It assigns higher weights to the misclassified instances, effectively "boosting" their importance in the training process.

Sequential Model Training:
Adaboost builds a sequence of weak learners, where each learner is trained based on the performance of its predecessors. Weak learners are typically simple models with low predictive power, such as shallow decision trees (stumps).



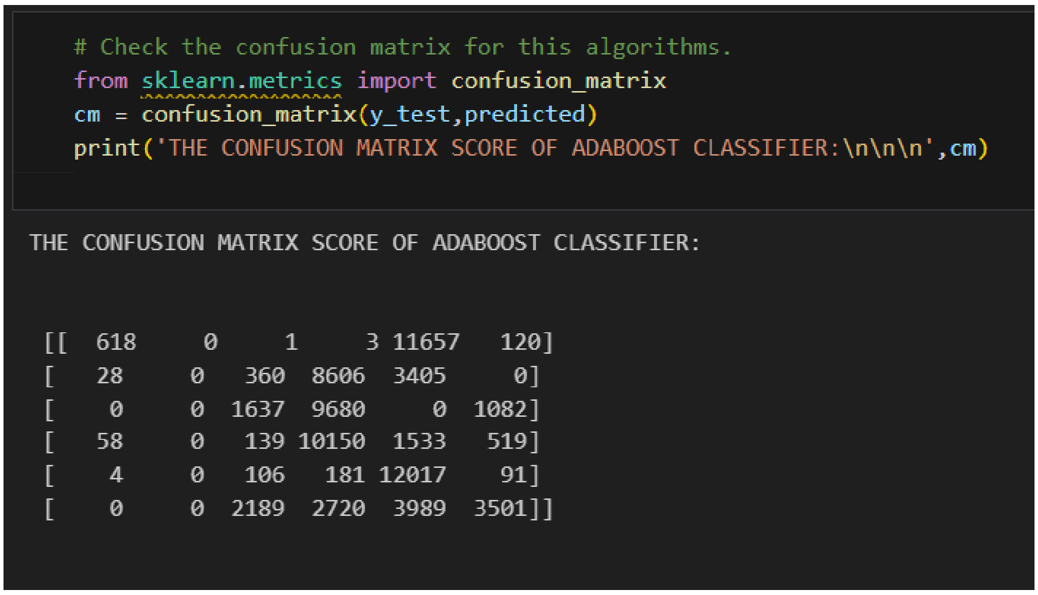
```
# Check classification report for this algorithm
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF ADABOOST CLASSIFIER:\n\n',cr)
```

```
THE CLASSIFICATION REPORT OF ADABOOST CLASSIFIER:

              precision    recall  f1-score   support

           0       0.87      0.05      0.09     12399
           1       0.00      0.00      0.00     12399
           2       0.37      0.13      0.19     12399
           3       0.32      0.82      0.46     12399
           4       0.37      0.97      0.53     12399
           5       0.66      0.28      0.40     12399

    accuracy                           0.38     74394
   macro avg       0.43      0.38      0.28     74394
weighted avg       0.43      0.38      0.28     74394
```

Fig4. AdaboostAlgorithm Classificatoin Report

Weighted Voting:
During the training phase, each weak learner contributes to the final prediction through a weighted majority vote. The weight of each learner's vote depends on its accuracy on the training data.

Model Adaptation:
Adaboost adaptively adjusts the weights of misclassified instances in subsequent iterations, focusing more on difficult-to-classify instances. This iterative process continues until a predefined number of weak learners are trained or until a desired level of accuracy is achieved.



```
# Check the confusion matrix for this algorithms.
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF ADABOOST CLASSIFIER:\n\n\n',cm)
```

```
THE CONFUSION MATRIX SCORE OF ADABOOST CLASSIFIER:


 [[  618     0     1     3 11657   120]
 [   28     0   360  8606  3405     0]
 [    0     0  1637  9680     0  1082]
 [   58     0   139 10150  1533   519]
 [    4     0   106   181 12017    91]
 [    0     0  2189  2720  3989  3501]]
```

Fig5. Adaboost Algorithm Confusion Matrix

- **Combining Weak Learners:**
  - The final prediction of the Adaboost ensemble is obtained by combining the predictions of all weak learners, weighted by their respective accuracies. This results in a strong classifier with improved generalization performance.

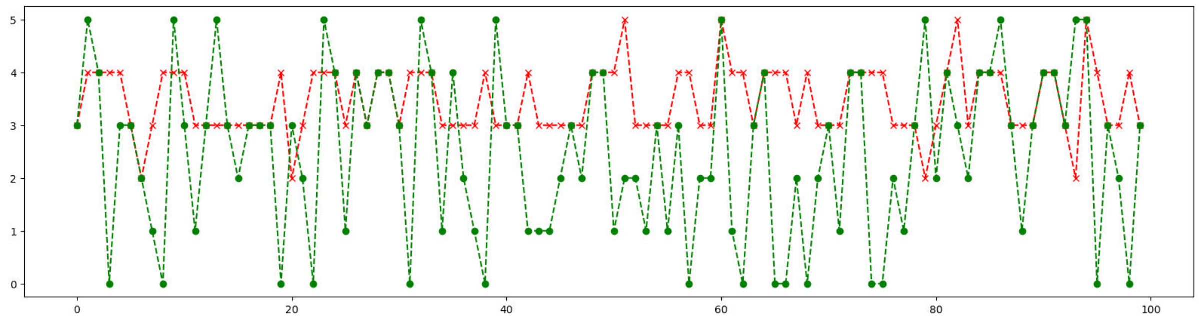
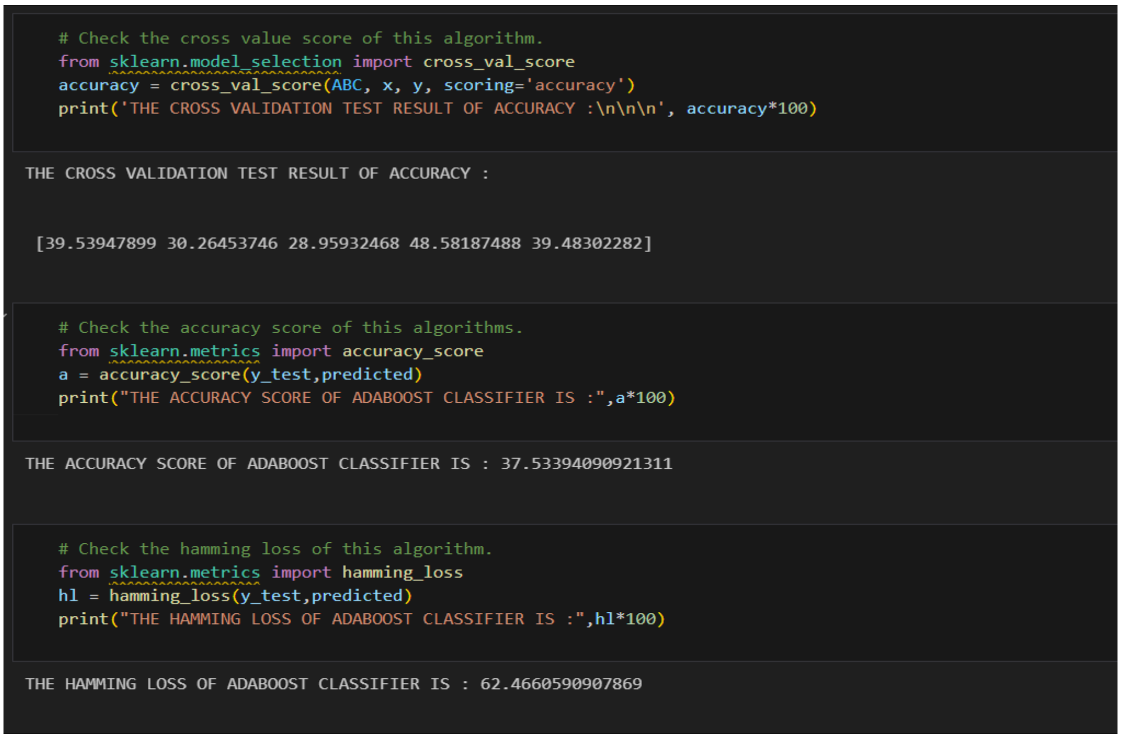


Fig6. AdaBoost worm plot

- **Handling Class Imbalance:**
  - Adaboost can effectively handle class imbalance problems by adjusting the weights of the training instances. It gives higher weights to minority class instances, ensuring that they receive more attention during model training.Overall, Adaboost is a powerful algorithm known for its ability to improve the performance of weak learners and produce accurate predictions even on complex datasets.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 5, May 2024 |



```
# Check the cross value score of this algorithm.
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(ABC, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy*100)
```

```
THE CROSS VALIDATION TEST RESULT OF ACCURACY :


 [39.53947899 30.26453746 28.95932468 48.58187488 39.48302282]
```

```
# Check the accuracy score of this algorithms.
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS :",a*100)
```

```
THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS : 37.53394090921311
```

```
# Check the hamming loss of this algorithm.
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF ADABOOST CLASSIFIER IS :",hl*100)
```

```
THE HAMMING LOSS OF ADABOOST CLASSIFIER IS : 62.4660590907869
```

Fig7. CatBoost Algorithm Accuracy Scores

Naive Bayes Algorithm:
Naive Bayes is a probabilistic machine learning algorithm based on Bayes' theorem with a strong assumption of feature independence. Despite its simplicity, Naive Bayes is widely used for classification tasks, especially in natural language processing and text classification. Here are some key points about Naive Bayes:

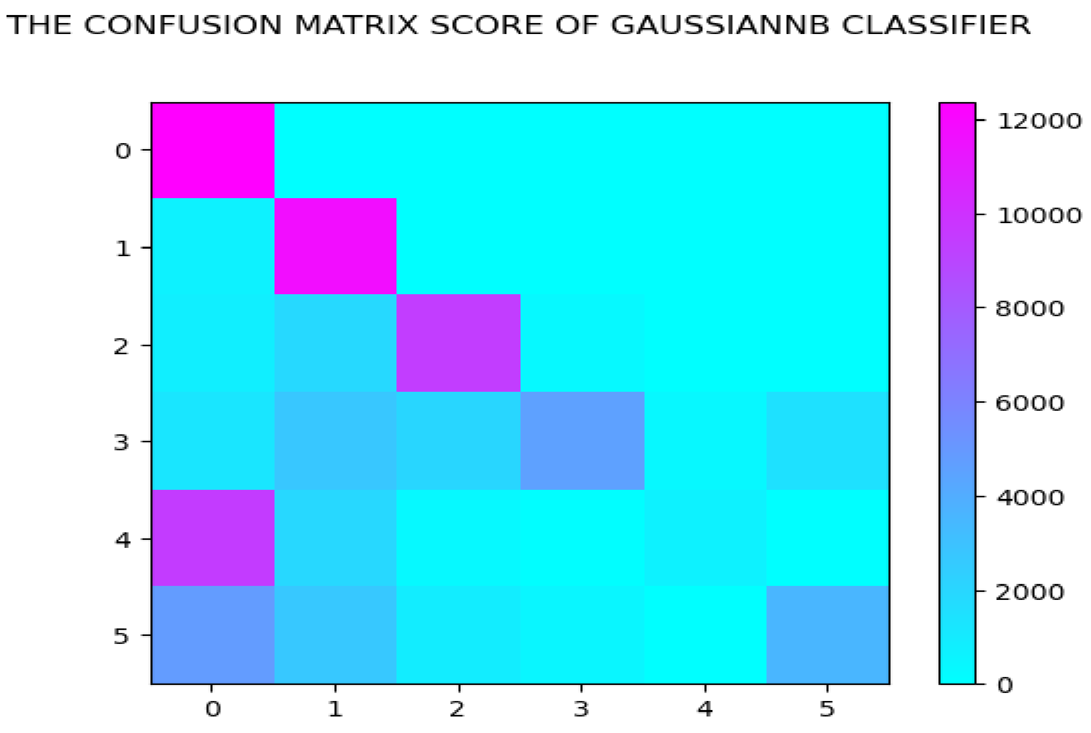


Fig8. Confusion Matrix Naive Bayes Algorithm

- **Bayesian Inference:**
    - Naive Bayes applies Bayesian inference to make predictions by calculating the probability of each class given the input features. It assumes that the features are conditionally independent given the class label.
- **Feature Independence Assumption:**
    - The "naive" assumption in Naive Bayes refers to the assumption of feature independence. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

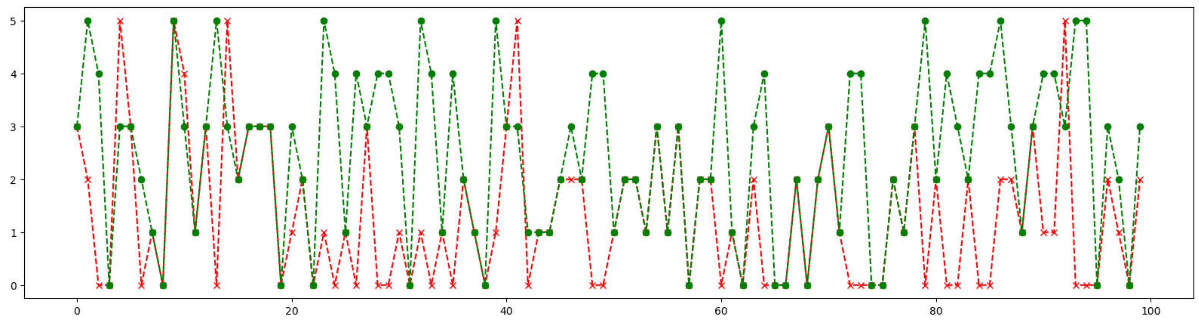
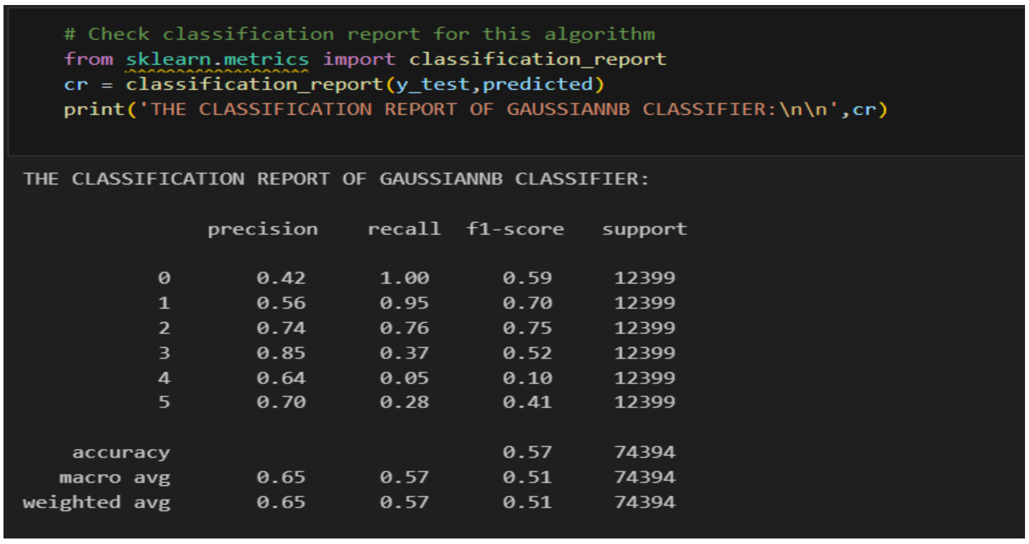


Fig9. Worm plot Naive Bayes Algorithm

1. **Simple Model:**
    a. Naive Bayes is a simple and easy-to-understand model that requires minimal training data compared to more complex algorithms. It is computationally efficient and can handle high-dimensional feature spaces effectively.
2. **Classification Rule:**
    a. The classification rule of Naive Bayes is based on the principle of maximum a posteriori (MAP) estimation, where the class with the highest posterior probability given the input features is selected as the predicted class.
3. **Types of Naive Bayes:**
    a. There are different variants of Naive Bayes classifiers, including Gaussian Naive Bayes for continuous features, Multinomial Naive Bayes for discrete features, and Bernoulli Naive Bayes for binary features.
4. **Handling Missing Values:**
    a. Naive Bayes can handle missing values in the input features by ignoring them during the probability calculation. It treats missing values as if they were not observed in the training data.
5. **Scalability and Performance:**
    a. Naive Bayes is highly scalable and performs well on large datasets with high-dimensional feature spaces. It is particularly well-suited for text classification tasks, such as spam detection and sentiment analysis.



```
# Check classification report for this algorithm
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF GAUSSIANNB CLASSIFIER:\n\n',cr)
```

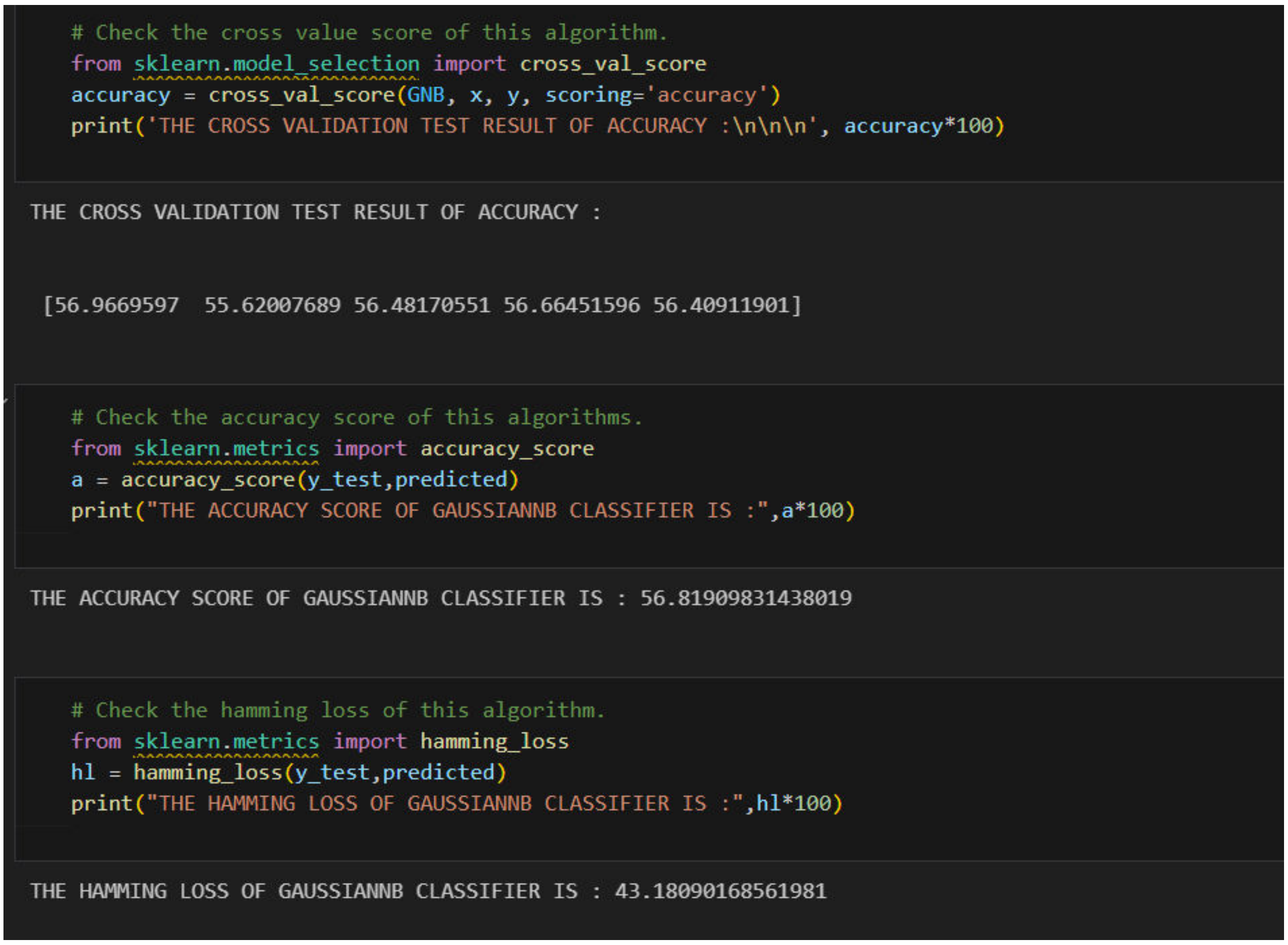
```
THE CLASSIFICATION REPORT OF GAUSSIANNB CLASSIFIER:

              precision    recall  f1-score   support

           0       0.42      1.00      0.59     12399
           1       0.56      0.95      0.70     12399
           2       0.74      0.76      0.75     12399
           3       0.85      0.37      0.52     12399
           4       0.64      0.05      0.10     12399
           5       0.70      0.28      0.41     12399

    accuracy                           0.57     74394
   macro avg       0.65      0.57      0.51     74394
weighted avg       0.65      0.57      0.51     74394
```

Fig10. Classification Report Naive Bayes Algorithm

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 5, May 2024 |

Overall, Naive Bayes is a simple yet effective algorithm that can achieve good performance with minimal computational resources, making it a popular choice for various classification tasks.



```
# Check the cross value score of this algorithm.
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(GNB, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy*100)
```

```
THE CROSS VALIDATION TEST RESULT OF ACCURACY :


 [56.9669597  55.62007689 56.48170551 56.66451596 56.40911901]
```

```
# Check the accuracy score of this algorithms.
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF GAUSSIANNB CLASSIFIER IS :",a*100)
```

```
THE ACCURACY SCORE OF GAUSSIANNB CLASSIFIER IS : 56.81909831438019
```

```
# Check the hamming loss of this algorithm.
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF GAUSSIANNB CLASSIFIER IS :",hl*100)
```

```
THE HAMMING LOSS OF GAUSSIANNB CLASSIFIER IS : 43.18090168561981
```

Fig11. Naive Bayes Algorithm Accuracy Scores

CatBoost Algorithm:
CatBoost is a gradient boosting algorithm specifically designed for handling categorical features in tabular data. Developed by Yandex, CatBoost stands for "Categorical Boosting." Here are some key points about CatBoost:

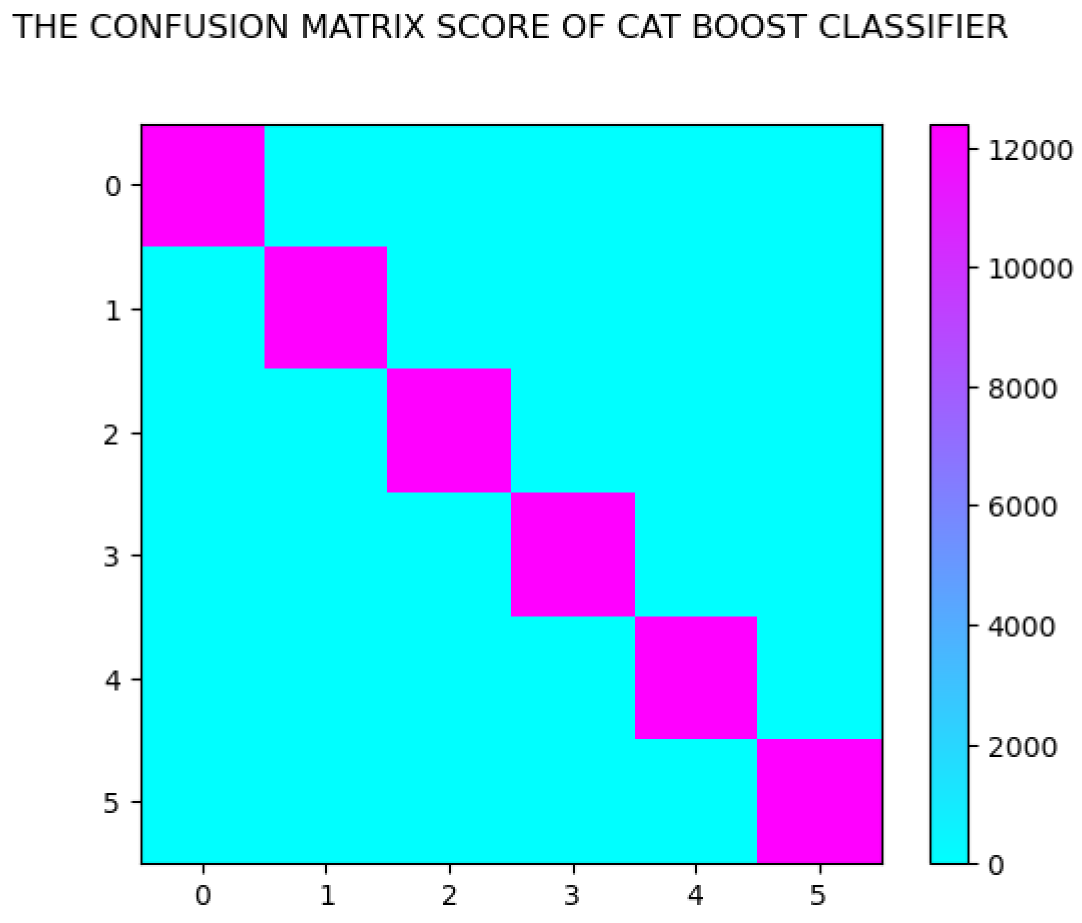


Fig12. Confusion Matrix CatBoost Algorithm

- **Gradient Boosting Ensemble:**
    - CatBoost belongs to the family of gradient boosting algorithms, which iteratively adds decision trees to an ensemble model. It minimizes a differentiable loss function by gradient descent.
- **Categorical Feature Handling:**
    - CatBoost is designed to handle categorical features directly without the need for preprocessing such as one-hot encoding. It internally converts categorical features into numerical values using techniques like target encoding and ordered boosting.

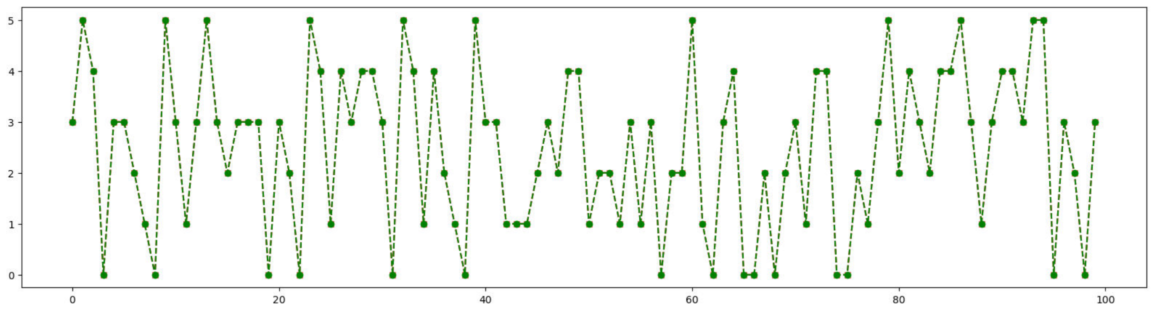
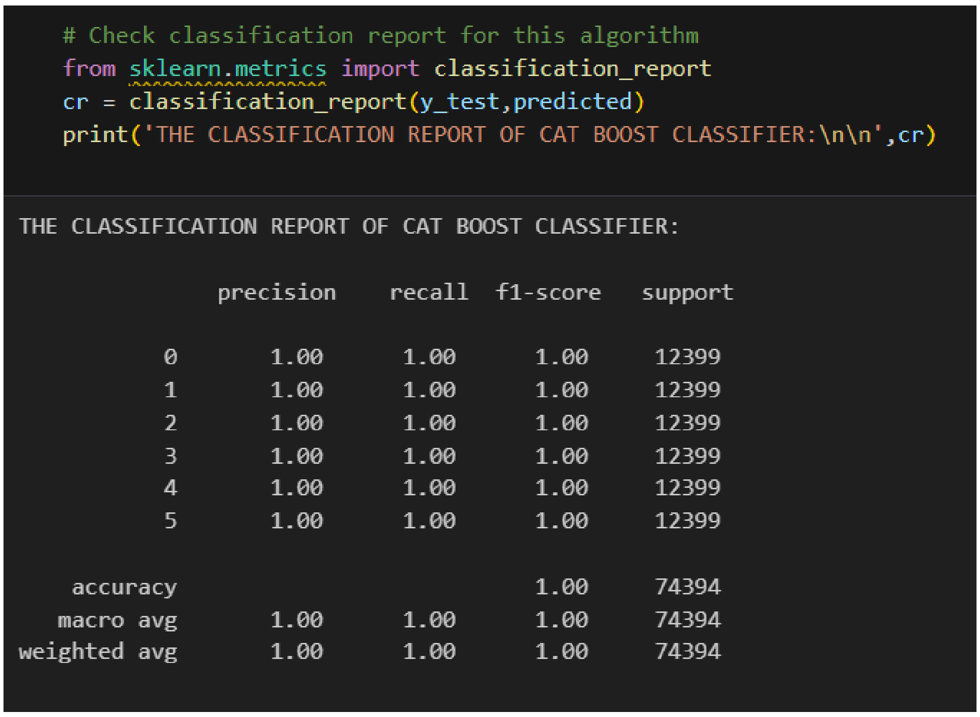


Fig13. Worm CatBoost Algorithm

- **Ordered Boosting:**
  - CatBoost utilizes an ordered boosting algorithm that sorts categorical features based on the target variable's statistics. This allows CatBoost to naturally incorporate categorical information during the training process.
- **Regularization Techniques:**
  - CatBoost includes built-in regularization techniques to prevent overfitting, such as L2 regularization on leaf values and feature combinations. These techniques help control the complexity of the model and improve generalization performance.

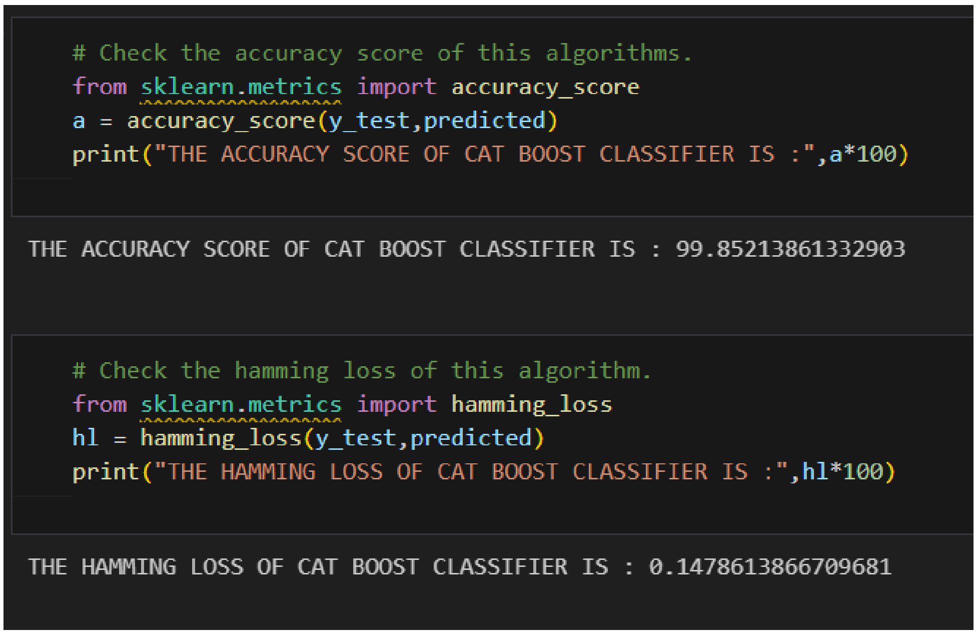


```
# Check classification report for this algorithm
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF CAT BOOST CLASSIFIER:\n\n',cr)
```

```
THE CLASSIFICATION REPORT OF CAT BOOST CLASSIFIER:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     12399
           1       1.00      1.00      1.00     12399
           2       1.00      1.00      1.00     12399
           3       1.00      1.00      1.00     12399
           4       1.00      1.00      1.00     12399
           5       1.00      1.00      1.00     12399

    accuracy                           1.00     74394
   macro avg       1.00      1.00      1.00     74394
weighted avg       1.00      1.00      1.00     74394
```

Fig14. Classification ReportCatBoost Algorithm

- **Learning Rate Shrinkage:**
  - To improve model stability and generalization, CatBoost incorporates a learning rate shrinkage strategy. It gradually reduces the contribution of each new tree to the ensemble, making the training process more controlled.
- **Handling Imbalanced Data:**
  - CatBoost provides options for handling imbalanced datasets by adjusting class weights or using custom loss functions. This makes it suitable for tasks with unequal class distributions, such as fraud detection or anomaly detection.
- **Scalability and Efficiency:**
  - CatBoost is designed for efficiency and can handle large datasets with millions of rows and thousands of features. It supports parallelization and can leverage multiple CPU cores for faster training.



```
# Check the accuracy score of this algorithms.
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF CAT BOOST CLASSIFIER IS :",a*100)
```

```
THE ACCURACY SCORE OF CAT BOOST CLASSIFIER IS : 99.85213861332903
```

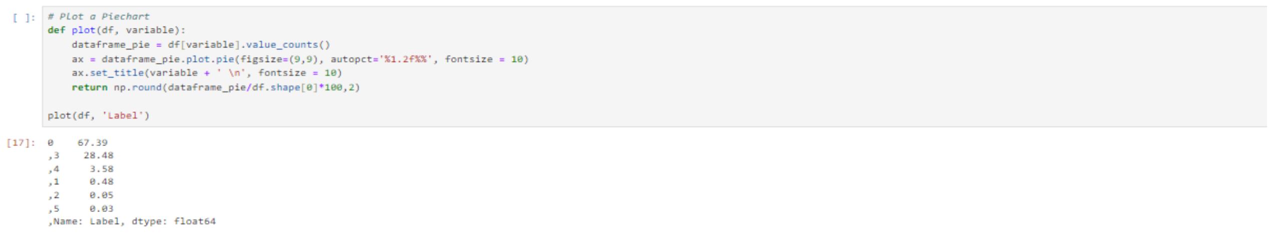
```
# Check the hamming loss of this algorithm.
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF CAT BOOST CLASSIFIER IS :",hl*100)
```

```
THE HAMMING LOSS OF CAT BOOST CLASSIFIER IS : 0.1478613866709681
```

Fig15. CatBoost Algorithm Accuracy Scores

- **Hyperparameter Tuning:**
  - CatBoost offers a wide range of hyperparameters that can be tuned to optimize model performance, including tree depth, learning rate, and regularization strength. Hyperparameter tuning can be performed using techniques like grid search or random search.

OUTPUT SCREEN SHOT:



```
[ ]: # Plot a Piechart
     def plot(df, variable):
         dataframe_pie = df[variable].value_counts()
         ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
         ax.set_title(variable + ' \n', fontsize = 10)
         return np.round(dataframe_pie/df.shape[0]*100,2)

     plot(df, 'Label')

[17]: 0    67.39
     ,3    28.48
     ,4     3.58
     ,1     0.48
     ,2     0.05
     ,5     0.03
     ,Name: Label, dtype: float64
```

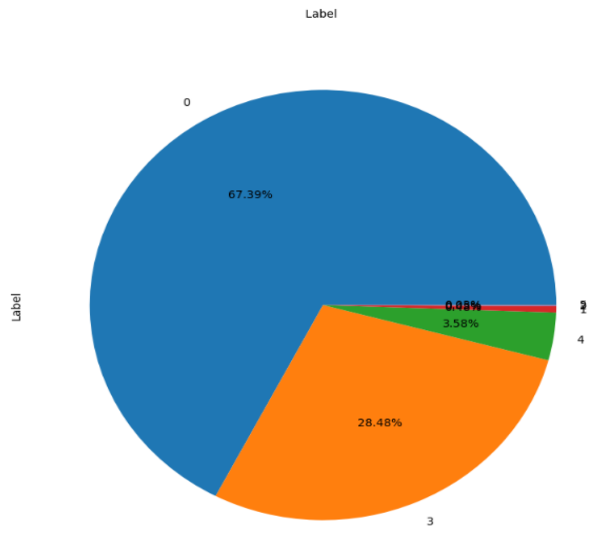


Fig16. PieChart for Visualization

```
# Plot a Histogram
df["'Pkt Len Mean'"].hist(figsize=(10,5),color='yellow',bins=25)
```

```
<AxesSubplot:>
```

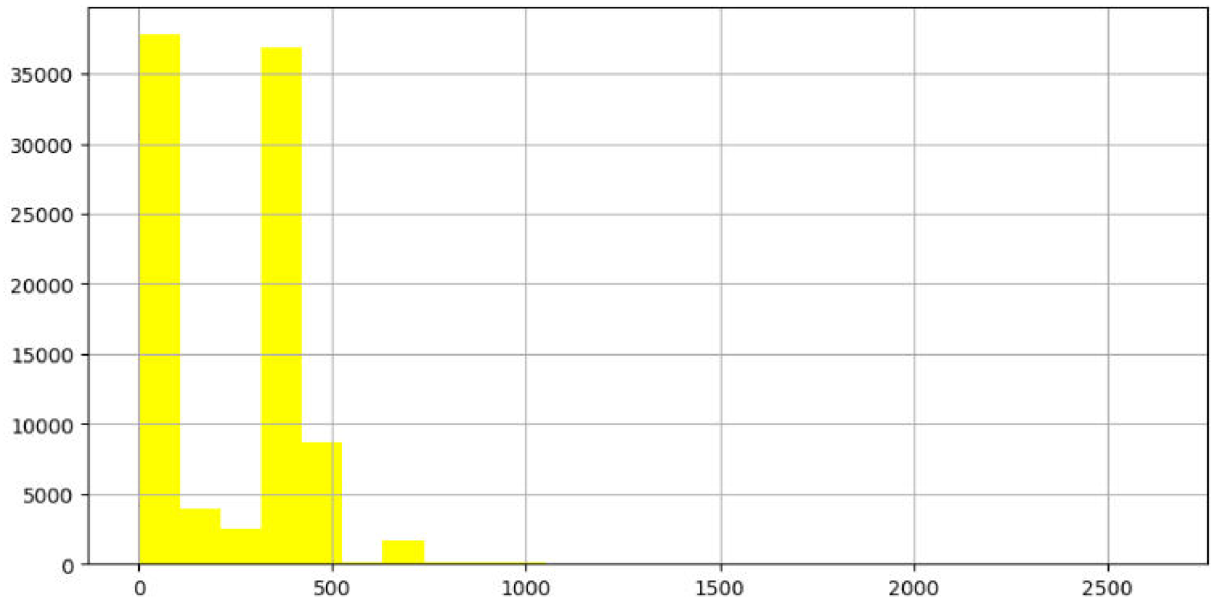


Fig17. Histogram plot for Visualization

```
# Plot a density plot
df["'Pkt Len Mean'"].plot(kind='density')
```

```
<AxesSubplot:ylabel='Density'>
```

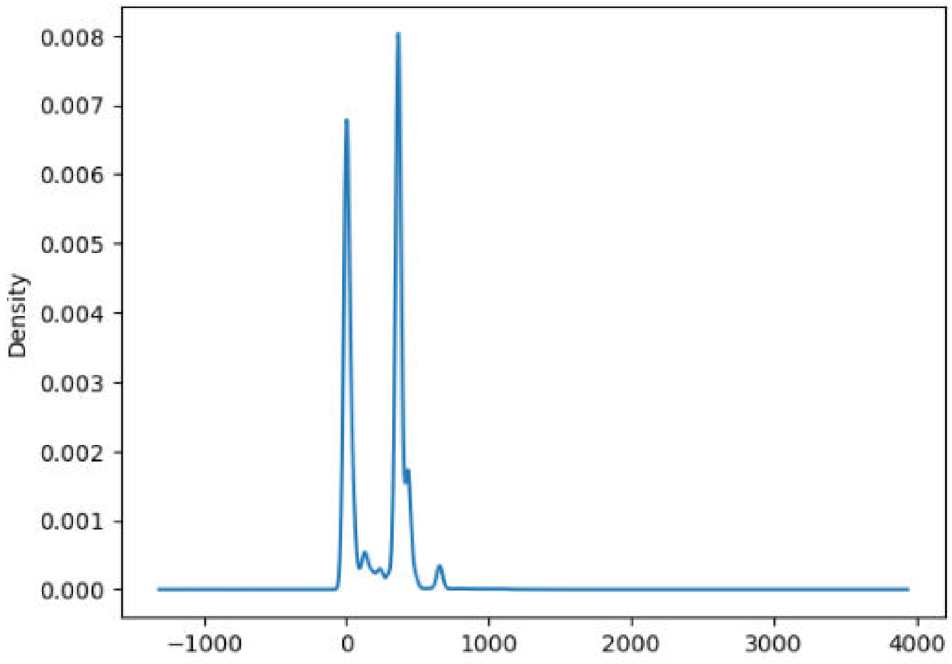


Fig18. Density plot for visualization

V. CONCLUSION

In this study, we have presented a comprehensive approach to cybersecurity threat detection using machine learning algorithms. The analytical process began with thorough data cleaning and preprocessing, including handling missing values and outliers. Exploratory analysis provided valuable insights into the dataset, aiding in feature selection and engineering.

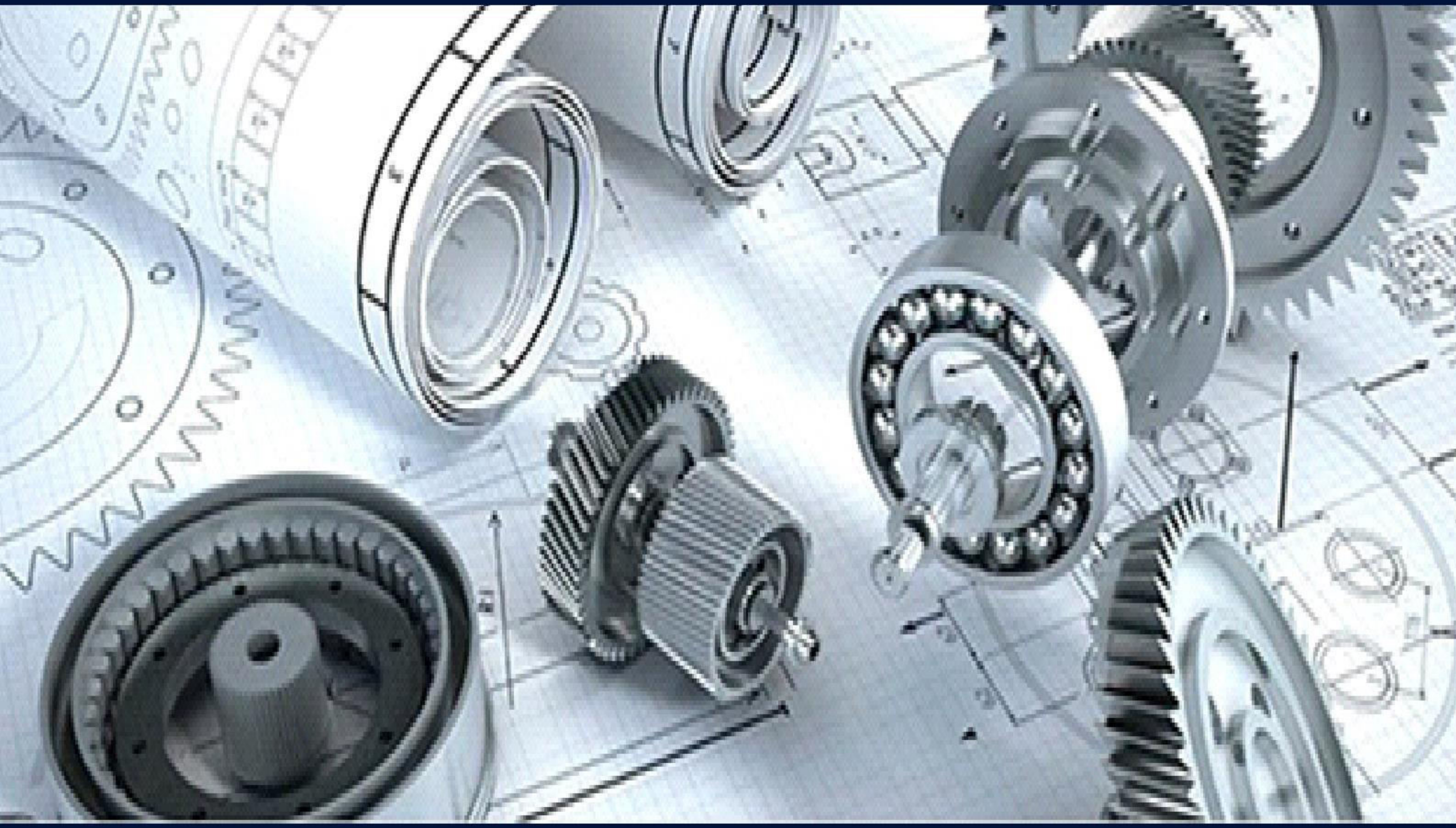
Subsequently, we built and evaluated multiple machine learning models to identify the most effective algorithm for detecting cyberattacks. Through rigorous testing and evaluation on a public test set, we determined the algorithm with the highest accuracy score, ensuring optimal performance in real-world scenarios

The selected algorithm will now be integrated into the cybersecurity application, where it will play a crucial role in identifying and categorizing various types of cyberattacks. By leveraging the power of machine learning, we aim to enhance the security posture of organizations and mitigate the risks posed by malicious actors.

Overall, our approach emphasizes the importance of data-driven decision-making in cybersecurity and highlights the potential of machine learning algorithms in bolstering defense mechanisms against evolving cyber threats. As we continue to refine and improve our methodology, we remain committed to staying at the forefront of cybersecurity innovation and safeguarding digital assets against emerging threats.

REFERENCES

[1] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.
[2] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119-139, 1997.
[3] A. G. S. Filho, F. M. J. Ferreira, and A. L. G. de Oliveira, "CatBoost: unbiased boosting with categorical features," arXiv preprint arXiv:1706.09516, 2017.
[4] T. M. Mitchell, "Machine learning," McGraw Hill, 1997.
[5] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," John Wiley & Sons, 2012.
[6] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," Springer Science & Business Media, 2009.
[7] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, "Data mining: practical machine learning tools and techniques," Morgan Kaufmann, 2016.
[8] C. M. Bishop, "Pattern recognition and machine learning," Springer Science & Business Media, 2006.
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.
[10] S. Haykin, "Neural networks and learning machines," Pearson Education, 2009.
[11] D. D. Lewis, "Evaluation of phrasal and clustered representations on a text categorization task," in Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, pp. 37-43.
[12] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in European conference on machine learning, 1998, pp. 137-142.
[13] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.
[14] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5-32, 2001.
[15] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 2012.



INTERNATIONAL JOURNAL
OF MULTIDISCIPLINARY RESEARCH
IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

+91 99405 72462 +91 63819 07438 ijmrsetm@gmail.com

www.ijmrsetm.com