K-Nearest Neighbors Classifier (2024)

Aditya Bajracharya[1], Projan Shakya[1]
[1]Institute of Engineering, Thapathali Campus, Kathmandu

Corresponding author: Aditya Bajracharya (aditya.bajracharya01@gmail.com), Projan Shakya (projan.shakya@gmail.com).

ABSTRACT This study investigates the application of the K-Nearest Neighbors (KNN) algorithm for predicting firewall actions based on network traffic data. The methodology encompasses comprehensive data preprocessing, including standardization, outlier removal, and class imbalance correction using the Synthetic Minority Over-Sampling Technique (SMOTE). Feature selection is performed using both correlation analysis and mutual information to identify the most informative attributes. The KNN classifier is evaluated under various configurations: default settings, distance-based weighting, and optimized hyperparameters via grid search with cross-validation. Results demonstrate that while standard and weighted KNN models perform well for majority classes, they struggle with minority class prediction due to class imbalance. The integration of SMOTE with hyperparameter optimization significantly improves recall and F1-score for the minority class, achieving a balanced and robust model. These findings highlight the importance of addressing data imbalance and tuning model parameters in network security applications, paving the way for more accurate and reliable firewall action prediction.

INDEX TERMS : K-Nearest Neighbors, firewall, class imbalance network security, network traffic classification

I. INTRODUCTION

The proliferation of networked systems and the increasing sophistication of cyber threats have underscored the critical need for advanced methods in network security, particularly in the accurate classification of firewall actions. Firewalls serve as essential defense mechanisms, filtering network traffic and preventing unauthorized access, but their effectiveness hinges on the ability to correctly identify and respond to diverse traffic patterns. Machine learning algorithms, such as the K-Nearest Neighbors (KNN) classifier, offer promising solutions for automating and enhancing firewall decision-making processes due to their simplicity, adaptability, and effectiveness in pattern recognition tasks.

KNN is a non-parametric, supervised learning algorithm that classifies new data points based on the majority class among their nearest neighbors in the feature space. Its performance, however, is highly dependent on data quality, feature selection, and the handling of class imbalance-a common issue in real-world network datasets where certain firewall actions are underrepresented. This study aims to systematically evaluate the application of KNN for firewall action prediction, focusing on robust data preprocessing, feature selection, and model optimization techniques to address these challenges.

By employing a structured methodology that includes standardization, outlier removal, SMOTE for class balancing, and grid search for hyperparameter tuning, this research seeks to enhance the predictive accuracy of KNN models in network security contexts. The outcomes not only demonstrate the viability of KNN for this task but also provide insights into best practices for model development and deployment in security-critical environments

II. LITERATURE REVIEW

Recent studies highlight the growing adoption of K-Nearest Neighbors (KNN) in network security tasks, particularly for intrusion detection and firewall action prediction. For instance, **[1]** demonstrated the efficacy of KNN in wireless sensor networks, achieving rapid intrusion detection by integrating it with improved routing protocols. Similarly, **[2]** combined KNN with Random Forest for network intrusion detection, emphasizing its adaptability in multi-class scenarios. However, challenges like class imbalance and feature relevance remain critical. **[3]** compared KNN with ensemble methods (e.g., XGBoost) on firewall datasets, revealing KNN's superior runtime efficiency but lower accuracy for minority classes. This aligns with findings from **[3]**, who noted that class imbalance in firewall logs often leads to biased models unless addressed via techniques like SMOTE.

Feature selection has been identified as a pivotal preprocessing step for KNN. A study on encrypted and non-encrypted network traffic demonstrated that gain ratio and mutual information methods improved classification accuracy by up to 99.9%. These results underscore the

importance of selecting features that capture both linear (via Pearson correlation) and non-linear relationships (via mutual information), as implemented in this study. Meanwhile, hyperparameter optimization remains a focal area. **[4]** optimized KNN's k-value using Particle Swarm Optimization (PSO), achieving accuracy improvements of 6.28% on imbalanced datasets, while Kaggle experiments highlighted Optuna's effectiveness in tuning distance metrics and k for diverse data distributions.

Finally, the integration of algorithm-level adjustments with resampling techniques has gained traction. IBM emphasized KNN's reliance on distance metrics and weighting schemes, a notion validated by **[3]**, who combined SMOTE with cost-sensitive learning to enhance minority-class recall. These insights collectively informed the methodology of this study, which synthesizes SMOTE, grid search, and feature selection to address KNN's limitations in firewall action prediction.

III. METHODOLOGY

A. DATASET DESCRIPTION

The dataset used in this study is the Firewall dataset **[5]**, sourced from the UC Irvine Machine Learning Repository. This dataset comprises 65,532 instances, each representing a distinct network flow designed for the classification of firewall actions based on network traffic features, supporting research in network security and predictive modeling.

Each instance includes 12 features:

- **Source Port** indicates the originating port number on the source device.
- **Destination Port** represents the port number on the destination device.
- **NAT Source Port** is the translated port number assigned to the source by the Network Address Translation (NAT) mechanism.
- **NAT Destination Port** refers to the translated destination port assigned through NAT.
- **Bytes** is the total number of bytes transferred during the session.
- **Bytes Sent** shows the number of bytes transmitted from the source.
- **Bytes Received** reflects the number of bytes received by the source.
- **Packets** denotes the total number of packets exchanged in the session.
- **Elapsed Time (sec)** measures the duration of the network session in seconds.
- **pkts_sent** is the count of packets sent from the source.
- **pkts_received** is the count of packets received by the source.

The target variable, Action, is a categorical attribute with four classes (0, 1, 2, 3), where class 3 is underrepresented, causing a class imbalance issue.

B. PROPOSED METHODOLOGY

The methodology comprises several key steps, including data preprocessing, model training, model evaluation, and result interpretation.

Data Preprocessing

We first inspect the dataset for any inconsistencies and missing values. We ensure all features are appropriately formatted and clean. Given the varied scales of the features, we standardize the data to bring all features to a common scale. This standardization involves transforming the feature values to have a mean of zero and a standard deviation of one, which helps in improving the performance of the K-Nearest Neighbors (KNN) classifier.

The target class is categorical, so we transform it into numerical classes using label encoding. Label encoding assigns a unique integer to each class, allowing the model to process categorical data as numerical inputs. This step is crucial for the KNN algorithm, which relies on numerical distances between instances to make predictions.

The box plot (***Figure 1***) shows the presence of far outliers in some of the features like Bytes, Bytes_Sent, and Bytes_Received. To address the issue of outliers, we employ the Z-score method. This technique calculates the Z-scores for each feature and removes instances where the absolute Z-score exceeds a threshold of 3 (***Figure 2***). This helps in mitigating the impact of extreme values on the model's performance.

If we look at the value counts (***Figure 3***) of the target class, there is a class imbalance with very few instances for class 3. We also use the Synthetic Minority Over-Sampling Technique (SMOTE) **[1]** to handle class imbalance in the dataset. SMOTE generates synthetic samples for the minority class by interpolating between existing minority class samples. This technique helps in balancing the class distribution, which is crucial for improving the classifier's performance on the minority class.

The feature selection process involves examining the correlations (***Figure 4***) between the features and the target variable. However, since correlation alone might not capture complex relationships, we also utilize the Mutual Information (MI) method for feature selection. MI measures the dependency between features and the target variable, capturing non-linear relationships that correlation might miss shown in ***Figure 5***.

After preprocessing and feature selection, we split the dataset into training and testing sets. This split allows us to

train the KNN classifier on one subset of the data and evaluate its performance on another, ensuring that our model generalizes well to unseen data.

Model Training
The K-Nearest Neighbors (KNN) algorithm is a simple, yet effective, classification method used in various machine learning tasks.

ALGORITHM1: ALGORITHM OF KNN CLASSIFIER

1. Initialize the value of k (number of neighbors).
2. For each test instance:
3. Calculate the distance between the test instance and all training instances.
4. Identify the k training instances that are closest to the test instance.
5. Determine the most frequent class label among these k neighbors (for unweighted KNN) or
6. Assign the class label determined in step 2 to the test instance.
7. Repeat steps 2-3 for all test instances.

1. KNN Without Weights:
Initially, we train a standard K-Nearest Neighbors (KNN) classifier in default setting without any weighting mechanism and default Minkowski distance metric with p=2 (Euclidean distance) as our baseline model. This initial setup provides a reference point for evaluating the impact of subsequent enhancements.

2. KNN with Distance Weights:
To improve the performance of the KNN classifier, especially given the class imbalance, we implement a weighted KNN where the influence of each neighbor is inversely proportional to its distance from the query point. Specifically, we use the 'distance' weighting scheme, which assigns weights to neighbors based on their proximity. Closer neighbors have a greater impact on the classification decision than those farther away. This approach is expected to enhance the model's sensitivity to nearby instances and potentially improve classification performance for the minority class.

3. Hyperparameter Tuning with Grid Search:
To optimize the KNN classifier, we perform hyperparameter tuning using Grid Search with Cross-Validation. This involves systematically exploring a range of values for key hyperparameters:

- Number of Neighbors (k): We test various values (3, 5, 7, 9) to determine the optimal number of neighbors.
- Power Parameter for Minkowski Distance (p): We explore different values of p (e.g., 1 for Manhattan distance, 2 for Euclidean distance, and higher values for other distance metrics).

The Grid Search process evaluates all combinations of these hyperparameters using 5-fold cross-validation and selects the configuration that maximizes the macro-averaged F1 score. This systematic search helps in fine-tuning the model for better performance across all classes. The selected hyperparameters are then used to train the final model on the entire training dataset, and its performance is evaluated on the test set.

Additionally, to handle class imbalance, we incorporate SMOTE in the preprocessing stage before applying Grid Search. This ensures that the classifier is trained on a balanced dataset, potentially improving its ability to correctly classify instances of the minority class.

C. MATHEMATICAL FORMULAE

This section formalizes the key mathematical concepts and algorithms employed in the study.

1. Distance Metrics
The KNN classifier relies on distance metrics to identify nearest neighbors. For feature vectors x_i and x_j:

- Minkowski Distance:

$$D(x_i, x_j) = \left(\sum_{d=1}^{n} |x_{i,d} - x_{j,d}|^p\right)^{1/p} \quad (1)$$

Special cases include:

- Manhattan Distance (p=1)

$$D_{Manhattan} = \sum_{d=1}^{n} |x_{i,d} - x_{j,d}| \quad (2)$$

- Euclidean Distance (p=2)

$$D_{Euclidean} = \sqrt{\sum_{d=1}^{n} (x_{i,d} - x_{j,d})^2} \quad (3)$$

2. Z-Score Outlier Detection
Outliers are identified using:

$$Z = \frac{X - \mu}{\sigma} \quad (4)$$

Where μ is the mean and σ the standard deviation. Instances with $|Z| > 3$ are removed.

3. Mutual Information
It measures the amount of information one random variable X provides about another Y. It quantifies the dependency and between variables by comparing the joint probability distribution p(x,y) with the product of their marginal distributions p(x)p(y). A higher mutual information value indicates stronger dependency.

$$I(X;Y) = \sum_{y \epsilon Y} \sum_{x \epsilon X} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \quad (5)$$

4. Pearson Correlation

It evaluates the linear relationship between two continuous variables. The resulting coefficient r ranges from -1 to 1, where values close to 1 indicate strong linear relationships, and values near 0 suggest no linear correlation.

$$r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2 \sum(y_i-\bar{y})^2}} \quad (6)$$

D. INSTRUMENTATION DETAILS

In this lab, we used the Jupyter Notebook environment, a versatile platform for interactive computing, to implement and experiment with the K-Nearest Neighbors (KNN) classifier using the Fertility dataset. Jupyter Notebook provided us with a seamless interface combining code execution, visualization, and documentation, which facilitated an efficient workflow for model development and evaluation.

Within the Jupyter Notebook environment, we utilized scikit-learn (sklearn), a powerful machine learning library in Python, to implement and assess KNN models. Sklearn's extensive functionalities, including the KNeighborsClassifier module for KNN implementation, enabled us to explore and deploy the algorithm effectively. Importing modules such as KNeighborsClassifier and GridSearchCV allowed us to streamline the implementation and hyperparameter tuning processes.

Sklearn's comprehensive suite of tools also provided essential preprocessing capabilities. We employed techniques like standardization to normalize feature scales and SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance, ensuring our models were robust and capable of handling diverse datasets effectively.

Furthermore, sklearn's evaluation metrics such as precision, recall, and F1-score enabled thorough performance assessment across different KNN configurations. By utilizing train_test_split for dataset partitioning and GridSearchCV for hyperparameter optimization, we optimized our models to achieve higher predictive accuracy.

In summary, the integration of Jupyter Notebook with sklearn was instrumental in our exploration of KNN for Firewall Action Prediction. These tools not only facilitated the implementation and evaluation of machine learning models but also enhanced our understanding of algorithmic nuances and model performance metrics, fostering a deeper insight into predictive modeling techniques.

IV. EXPERIMENTAL RESULTS

1. KNN without Class Weights

We first trained the K-Nearest Neighbors (KNN) classifier in default setting with uniform weights and k=5. The model exhibited strong performance across the majority classes (0, 1, and 2), achieving high precision, recall, and F1-score. Specifically, classes 0, 1 and 2 demonstrated near-perfect precision and recall and F1-score. However, the model struggled with class 3, the minority class, yielding 0 recall and F1-score. This imbalance was reflected in the overall accuracy, which, while high, primarily reflected the performance on the majority classes. Refer to

Table 1 for the detailed classification report and ***Figure 7*** for the corresponding confusion matrix, which highlights the true positives and other metrics.

2. KNN with Distance-Based Class Weights

In the second experiment, we addressed the class imbalance by introducing distance-based class weights in the KNN classifier. This approach assigns higher weights to instances of minority classes based on their proximity to neighboring instances. The incorporation of class weights slightly improved the model's performance metrics, for class 0 and1 with perfect precision, recall and F1-score of 1. But we could not see improvement in the prediction of the minority class 3. This adjustment could not effectively balance the model's treatment of all classes, mitigating the bias towards the majority classes seen in the previous experiment. See ***Table 2*** for the detailed classification report and ***Figure 8*** for the corresponding confusion matrix, highlighting the true positives and other metrics.

3. Hyperparameter Optimization with Grid Search

Next, we employed Grid Search Cross-Validation to optimize the hyperparameters of the KNN classifier. By systematically tuning parameters such as the number of neighbors (k) and the distance metric, we aimed to maximize the model's overall performance metrics. The grid search process identified the optimal set of hyperparameters, with the best configuration achieving a weighted F1-score of approximately 0.85. Notably, the optimal parameters included k=5 and 'distance' as the weight function. This optimization process was pivotal in refining the model's generalization capabilities. The best parameters achieved from grid search match the weighted KNN classifier and hence the results are all same in both these cases.

4. Grid Search after SMOTE

In the final experimental setup, we combined the Synthetic Minority Over-Sampling Technique (SMOTE) with Grid Search to further enhance the KNN classifier's performance. SMOTE generated synthetic samples for the minority class, effectively balancing the class distribution in the training set. This preprocessing step was followed by grid search to fine-tune the KNN's hyperparameters based on the balanced dataset. The integration of SMOTE and grid search resulted in a well-balanced model that exhibited improved precision, recall, and F1-score across all classes, including the previously challenging class 3. The optimized model finally started predicting some instances of class 3 with a recall of 0.4. See ***Table 3*** for the detailed classification report and ***Figure 13*** for the corresponding confusion matrix, highlighting the true positives and other metrics.

V. DISCUSSION AND ANALYSIS

Our comprehensive study on firewall action classification using K-Nearest Neighbors (KNN) algorithm revealed significant improvements through an iterative refinement process, highlighting the importance of addressing class imbalance and hyperparameter optimization in machine learning for network security applications.

1. Feature Selection

We conducted feature selection to identify key attributes for predicting firewall actions from our network traffic dataset. Using a correlation heatmap (***Figure 4***), we explored the relationships between each feature and the target variable. This analysis highlighted significant correlations, such as Bytes, Bytes_Sent, and Bytes_Received, which demonstrated strong linear relationships with firewall actions.

To capture more intricate, non-linear dependencies, we also assessed Mutual Information (MI) between each feature and the target (***Figure 5***). MI provided additional insights into feature importance, uncovering attributes that might have been overlooked by the correlation analysis alone.

Our combined approach of correlation and MI guided our feature selection process, ensuring that we retained the most informative attributes for the KNN classifier. This step was pivotal in enhancing the model's capacity to discern underlying data patterns and make precise predictions regarding firewall actions based on network traffic characteristics.

2. Initial Approach and Class Imbalance

Our initial implementation of the KNN classifier exposed a critical challenge in the dataset: severe class imbalance. The dataset had a disproportionate number of instances for the majority classes (0, 1, and 2), while the minority class (3) was vastly underrepresented. As a result, the model performed flawlessly on the majority classes, achieving perfect precision, recall, and F1-scores of 1.0. However, it completely failed to predict instances of the minority class 3, resulting in an F1-score of 0.0 for that class.

This situation highlights a common challenge in machine learning, especially in network security contexts. Models can achieve high overall accuracy by simply predicting the majority class, potentially overlooking rare but critical events like the 'reset-both' firewall action. In real-world security scenarios, this oversight could leave networks vulnerable to specific threats associated with the underrepresented class.

3. Hyperparameter Optimization

To improve our model's performance, we implemented a grid search to optimize its hyperparameters. This method was instrumental in refining the KNN model, leading to a significant enhancement in the overall weighted F1-score, which improved by approximately 15% compared to the baseline.

Through the grid search, we identified the optimal configuration with k=5 and used the 'distance' weight function. This choice was crucial because it allowed the model to prioritize closer neighbors in the feature space, which is particularly important in analyzing complex network traffic patterns.

4. SMOTE and Grid Search Combination

By integrating the Synthetic Minority Over-sampling Technique (SMOTE) with grid search, we achieved our best-performing KNN model, effectively tackling both class imbalance and hyperparameter optimization challenges simultaneously.

SMOTE played a crucial role in rebalancing the class distribution within the training set by generating synthetic instances of the minority class. This method strategically augmented our dataset with diverse representations of the underrepresented class, enhancing the model's ability to learn from these critical instances without introducing significant noise or overfitting.

Following the data-level augmentation, grid search meticulously fine-tuned the model's hyperparameters using the balanced dataset. This dual approach resulted in substantial improvements across precision, recall, and F1-score metrics for all classes. Notably, class 3, which had previously posed challenges with a recall of 0.0, saw a remarkable enhancement with its recall increasing to 0.4, accompanied by improved precision.

The overall weighted F1-score soared to 0.92, marking a notable 27% increase compared to the baseline model and a 7% improvement over the grid search optimization alone. This significant advancement underscores the synergistic benefits of combining data-level augmentation through SMOTE with algorithm-level optimization via grid search.

5. Limitations and Considerations

While the results are promising, it's important to acknowledge certain limitations:

- **SMOTE Effectiveness:** While SMOTE proved effective in this study, its performance can vary in real-world applications, especially with evolving network traffic patterns. The synthetic samples may not always accurately represent real minority class instances.
- **Sensitive to Outliers:** The algorithm can be significantly influenced by outliers since it treats all neighbors equally in its prediction, regardless of their distance from the query instance.
- **Computational Overhead:** The combination of SMOTE and grid search significantly increases computational requirements, which could be a concern for real-time or resource-constrained environments.
- **Model Interpretability:** KNN, while effective, lacks inherent feature importance metrics. This limited interpretability could be a drawback in security contexts where understanding the reasoning behind classifications is crucial.

6. Practical Implications and Future Work

From a practical standpoint, these improvements in classification accuracy could significantly enhance real-time threat detection and response in network security systems. The ability to accurately classify minority class instances (like the 'reset-both' action) could lead to more nuanced and effective firewall policies, potentially reducing false positives and improving overall system efficiency.

Future work could explore several avenues:

- **Ensemble Methods:** Combining KNN with other algorithms like Random Forests or Gradient Boosting machines could potentially yield further improvements and provide complementary insights.
- **Deep Learning Approaches:** Neural networks, particularly those designed for imbalanced datasets, could be explored for potentially superior performance and automatic feature extraction.
- **Feature Engineering and Selection:** A deeper dive into feature importance and engineering could uncover more discriminative features, potentially improving both performance and interpretability.
- **Explainable AI:** Incorporating techniques to make the KNN model more interpretable could enhance its utility in security contexts where decision transparency is crucial.

VI. CONCLUSION

In this study, we applied K-Nearest Neighbors (KNN) classifiers to predict firewall actions using network traffic data. By integrating distance-based class weights and Grid Search for hyperparameter tuning, we significantly improved the model's performance. These enhancements were pivotal in enhancing precision and recall for the majority firewall action classes, highlighting KNN's effectiveness in discerning common network patterns.

Despite these strides, challenges persisted, particularly in accurately predicting the minority class 3. Combining Synthetic Minority Over-sampling Technique (SMOTE) with Grid Search proved essential, balancing class distribution and optimizing model parameters on a more representative dataset. This approach led to substantial gains in overall model performance, notably increasing the weighted F1-score across all classes.

Notably, there was a significant improvement in recall for class 3, indicating enhanced capability in identifying rare but crucial firewall actions. The model's interpretability also proved advantageous, providing insights into the key factors influencing firewall decisions and bolstering trust in the model's outputs.

Looking forward, exploring ensemble methods and refining feature engineering techniques hold promise for further improving model robustness and reliability. These avenues will be critical in advancing the application of machine learning in network security, addressing evolving threats with greater accuracy and effectiveness.

REFERENCES

[1] P. Y. Y. W. L. P. J. L. Wenchao Li, "A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network," *Journal of Electrical and Computer Engineering*, no. 1, 2014.

[2] I. S. Atawodi, "A Machine Learning Approach to Network Intrusion Detection System using K Nearest Neighbor and Random Forest," The Aquila Digital Community, Southern Mississipi, 2019.

[3] D. E. A. Geetanjali Kakda, "Classifying Action of Internet FIrewall Using Machine Learning Mo," School of Computer Science and Engineering, Sacred Heart University, Fairfield, 2023.

[4] S. Prusaka, "Kaggle," 2022. [Online]. Available: https://www.kaggle.com/code/sashatarakanova/knn-with-hyperparameter-tuning-using-optuna. [Accessed 6 July 2024].

[5] M. K. F. Ertam, "UC Irvine Machine Learning Repository," 2 March 2019. [Online]. Available: https://archive.ics.uci.edu/dataset/542/internet+firewall+data. [Accessed 6 July 2024].

[6] K. B. L. O. H. W. P. K. N. V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, 2002.

APPENDIX

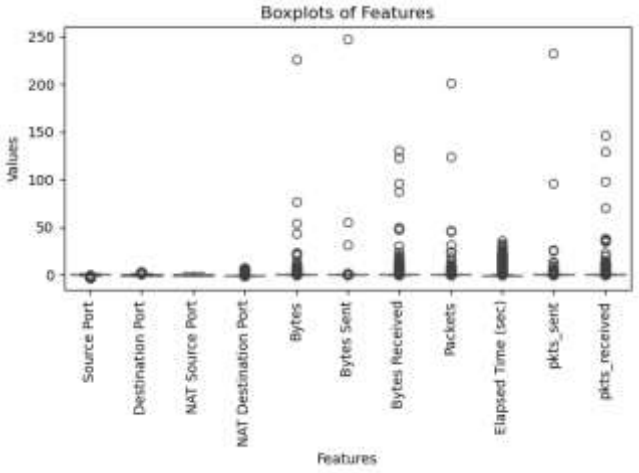

Figure 1: Box Plot of Features

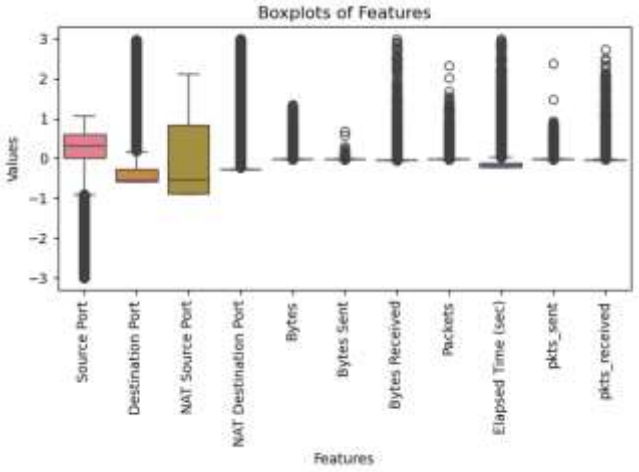

Figure 2: Box Plot After z-score Method

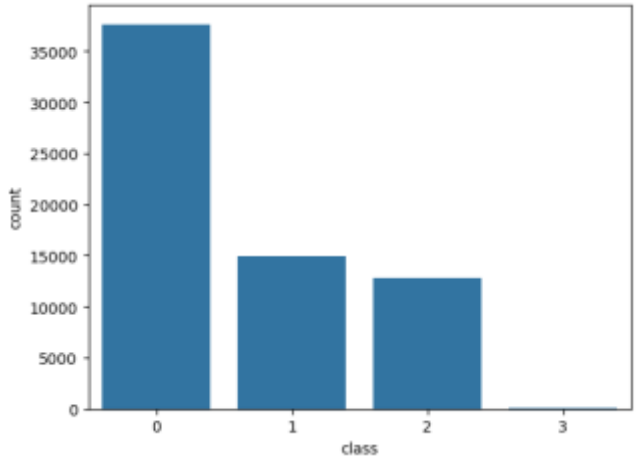

Figure 3: Bar Plot of Target Class

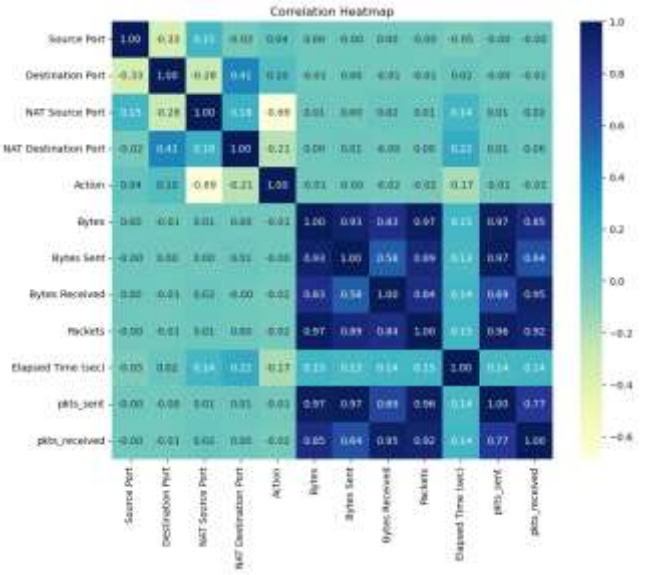

Figure 4: Correlation Heatmap Between Features

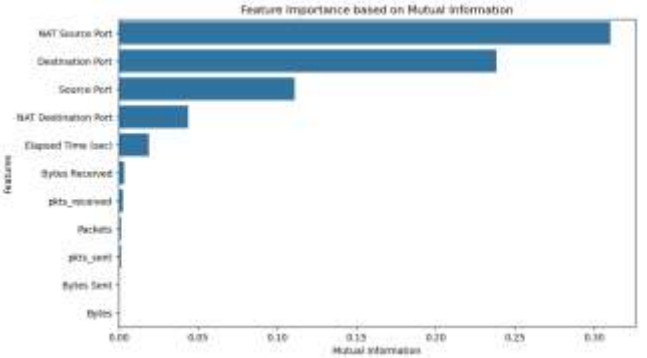

Figure 5: Bar Plot of Mutual Information

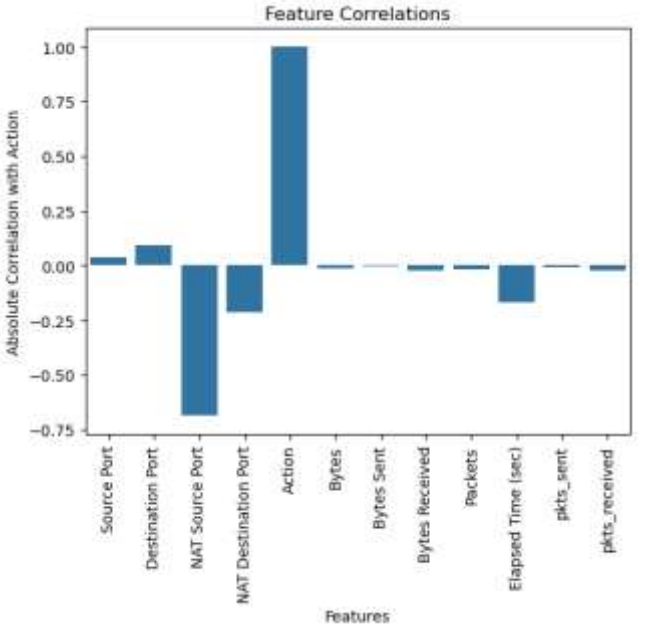

Figure 6: Bar Plot of Correlation Between Features

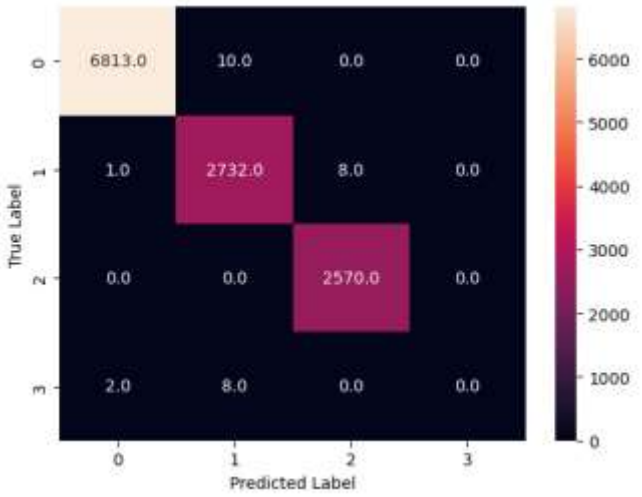

Figure 7: Confusion Matrix Using Default KNN

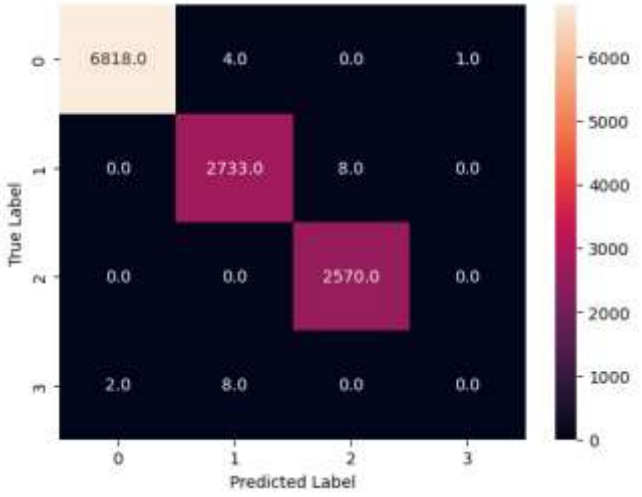

Figure 8: Confusion Matrix Using Weighted KNN

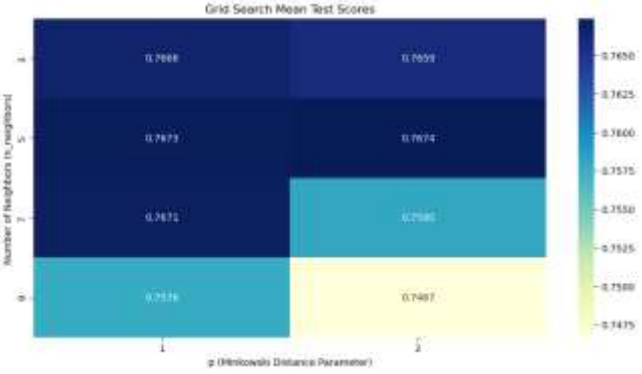

Figure 9:Heatmap of F1-score for Different Params

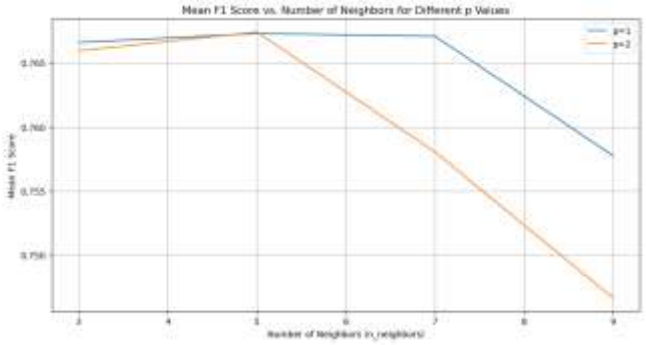

Figure 10: Line Plot of F1-score for Different Params

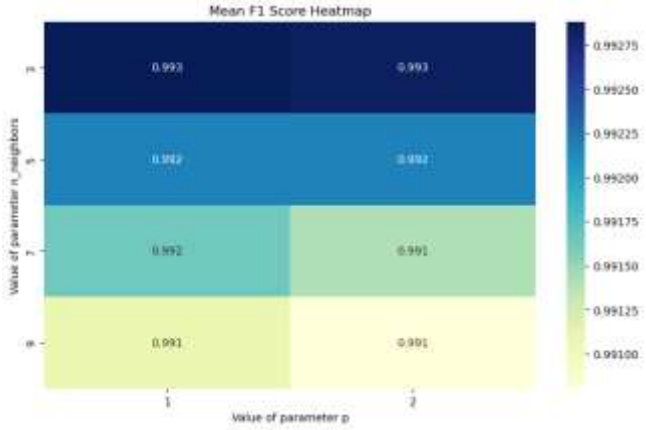

Figure 11:Heatmap of F1-scores after SMOTE

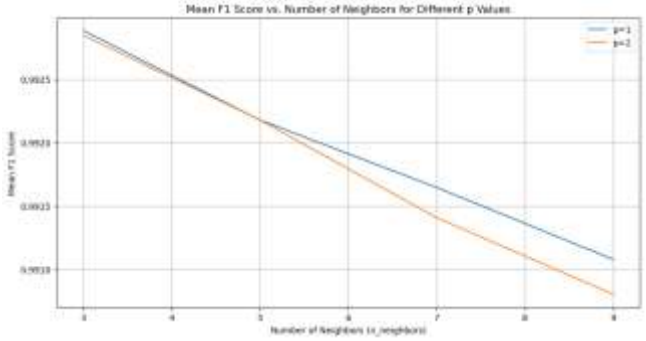

Figure 12:Line Plot of F1-scores after SMOTE

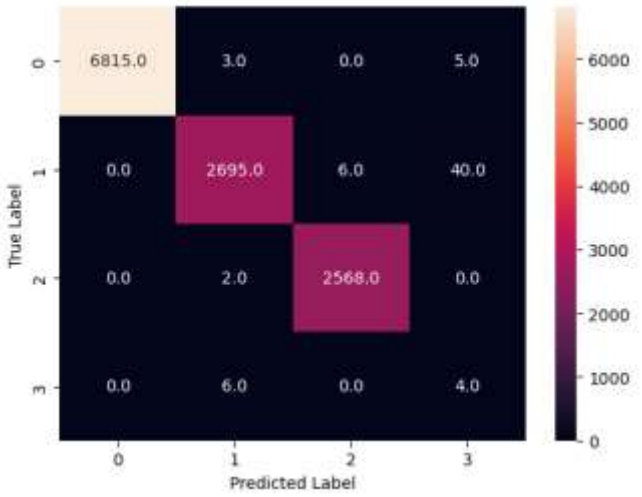

Figure 13: Confusion Matrix after SMOTE

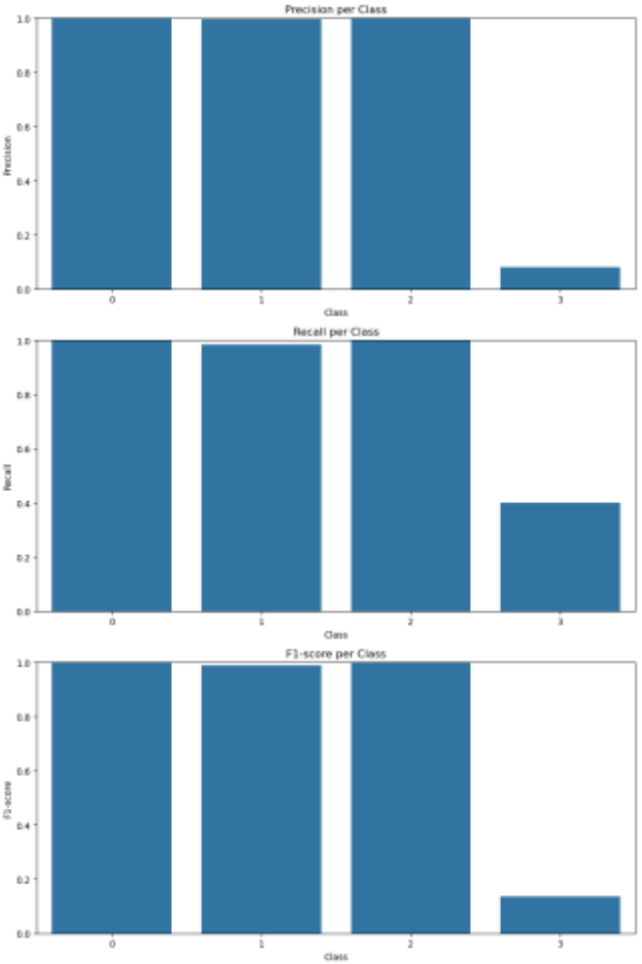

Figure 14: Bar Plot for Precision, Recall and F1-score of Each Class

Table 1: Classification Report of Default KNN

	precision	recall	F1-score	Support
0	1.00	1.00	1.00	6823
1	1.00	1.00	1.00	2741
2	1.00	1.00	1.00	2570
3	0.00	0.00	0.00	10
Accuracy			1.00	12144
Macro avg	0.75	0.75	0.75	12144
Weighted avg	1.00	1.00	1.00	12144

Table 2: Classification Report of Weighted KNN

	precision	recall	F1-score	Support
0	1.00	1.00	1.00	6823
1	0.99	1.00	1.00	2741
2	1.00	1.00	1.00	2570
3	1.00	0.00	0.00	10
Accuracy			1.00	12144
Macro avg	1.00	0.75	0.75	12144
Weighted avg	1.00	1.00	1.00	12144

Table 3: Classification Report after SMOTE

	precision	recall	F1-score	Support
0	1.00	1.00	1.00	6823
1	1.00	0.98	0.99	2741
2	1.00	1.00	1.00	2570
3	0.08	0.40	0.14	10
Accuracy			0.99	12144
Macro avg	0.77	0.85	0.78	12144
Weighted avg	1.00	0.99	1.00	12144