INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

ROBERT GORDON UNIVERSITY ABERDEEN

Bsc(Hons) Artificial Intelligence and Data Science

Cybersecurity Intrusion Detection Software Optimization using Genetic Algorithms and Machine Learning

Module: **CM4607 - Computational Intelligence**

Module Co-Ordinator: **Mrs. Fathima Farhath**

Student Name: **Nina Dilkushi Abeyratne**

IIT ID: **20221754**

RGU ID: **2237920**

Table of Contents

1. Introduction

Cybersecurity is a critical requirement for modern networked systems due to the increasing volume and sophistication of cyberattacks. Intrusion Detection Systems (IDS) monitor network traffic to identify malicious activities, but real-world traffic is high-dimensional, complex, and evolving, making traditional detection techniques insufficient.

Machine learning based IDS can learn complex attack patterns but often struggle with efficiency and scalability due to the large number of features. Selecting an optimal feature subset is a combinatorial problem that conventional methods cannot solve efficiently.

Genetic Algorithms (GA), a form of evolutionary computing, offer a robust approach for feature selection by evolving candidate solutions to maximize detection performance under computational constraints. Combined with machine learning, GA enables adaptive and intelligent IDS capable of handling high-dimensional, class-imbalanced data.

This coursework addresses an industry-scale IDS problem using the CICIDS2017 dataset. A modular pipeline is implemented in Python/Colab using DEAP and scikit-learn, covering preprocessing, baseline modeling, GA-based feature selection, and final evaluation of a GA-optimized Random Forest model. The study demonstrates how evolutionary computation can enhance IDS performance while balancing accuracy and efficiency.

Objectives

This project addresses the challenge of optimizing intrusion detection through the application of Genetic Algorithms (GA) combined with Machine Learning (ML). The primary objectives are:

1. Reduce the dimensionality of network traffic features while maintaining detection accuracy

2. Improve computational efficiency for real-time threat detection.

3. Minimize false positive rates critical for operational security systems

4. Demonstrate the effectiveness of evolutionary computing in cybersecurity applications

Dataset

The CICIDS2017 dataset is used, containing realistic network traffic with diverse attack types captured over five days. The dataset includes 78 features extracted from network flows and encompasses multiple attack categories including DoS, DDoS, Web attacks, infiltration, and port scanning.

2. Problem Identification (Task 1)

The selected industry application is a **Network-based Intrusion Detection System (IDS)** deployed in enterprise environments, Internet Service Providers (ISPs), or critical infrastructure networks. Such environments generate extremely large volumes of network traffic daily and require continuous monitoring to detect malicious activities alongside legitimate communication.

In operational Security Operations Centers (SOCs), IDS solutions must analyze millions of network flows per day and detect multiple attack categories such as Distributed Denial of Service (DDoS), Slowloris, FTP-Patator, PortScan, and web-based attacks. These systems are expected to operate under near real-time constraints while maintaining high detection accuracy and low false positive rates.

This project uses the CICIDS2017 dataset as a representative industry-scale dataset. The dataset contains realistic network traffic collected over multiple days, including benign traffic and a variety of attack scenarios. Each network flow is described using 78 numerical features, reflecting flow-level statistics, temporal behavior, and protocol characteristics commonly used in real IDS deployments.

The scale and structure of this problem make it well suited for demonstrating the applicability of evolutionary algorithms. The high dimensionality of the feature space, the large volume of data, and the need to balance detection performance with computational efficiency reflect real-world constraints faced by industrial IDS solutions. As such, this application provides a realistic and appropriate context for applying evolutionary computing techniques in combination with machine learning.

3. Justification for Genetic Algorithm (Task 2)

3.1 Why Traditional IT and Numerical Optimization Are Insufficient

The intrusion detection problem cannot be effectively addressed using traditional rule-based IT solutions or classical numerical optimization techniques. Rule-based IDS approaches rely on predefined signatures and thresholds, which are unsuitable for high-dimensional and evolving network traffic where attack patterns are complex and non-linear.

The feature selection task involves **77 numerical features**, resulting in a search space of 2^{77} possible feature subsets. Exhaustive search is computationally infeasible, and greedy approaches such as forward or backward feature selection are prone to local optima due to strong feature interactions. The usefulness of a feature often depends on the presence of others, making incremental selection unreliable.

Classical numerical optimization methods are also inappropriate because the objective function machine learning performance metrics such as macro F1-score is **non-differentiable**, noisy, and evaluated on discrete feature masks. Since the decision variables are binary, gradient-based optimization cannot be applied. As a result, conventional deterministic and numerical approaches cannot efficiently solve this combinatorial optimization problem.

3.2 Why Genetic Algorithms Are Appropriate

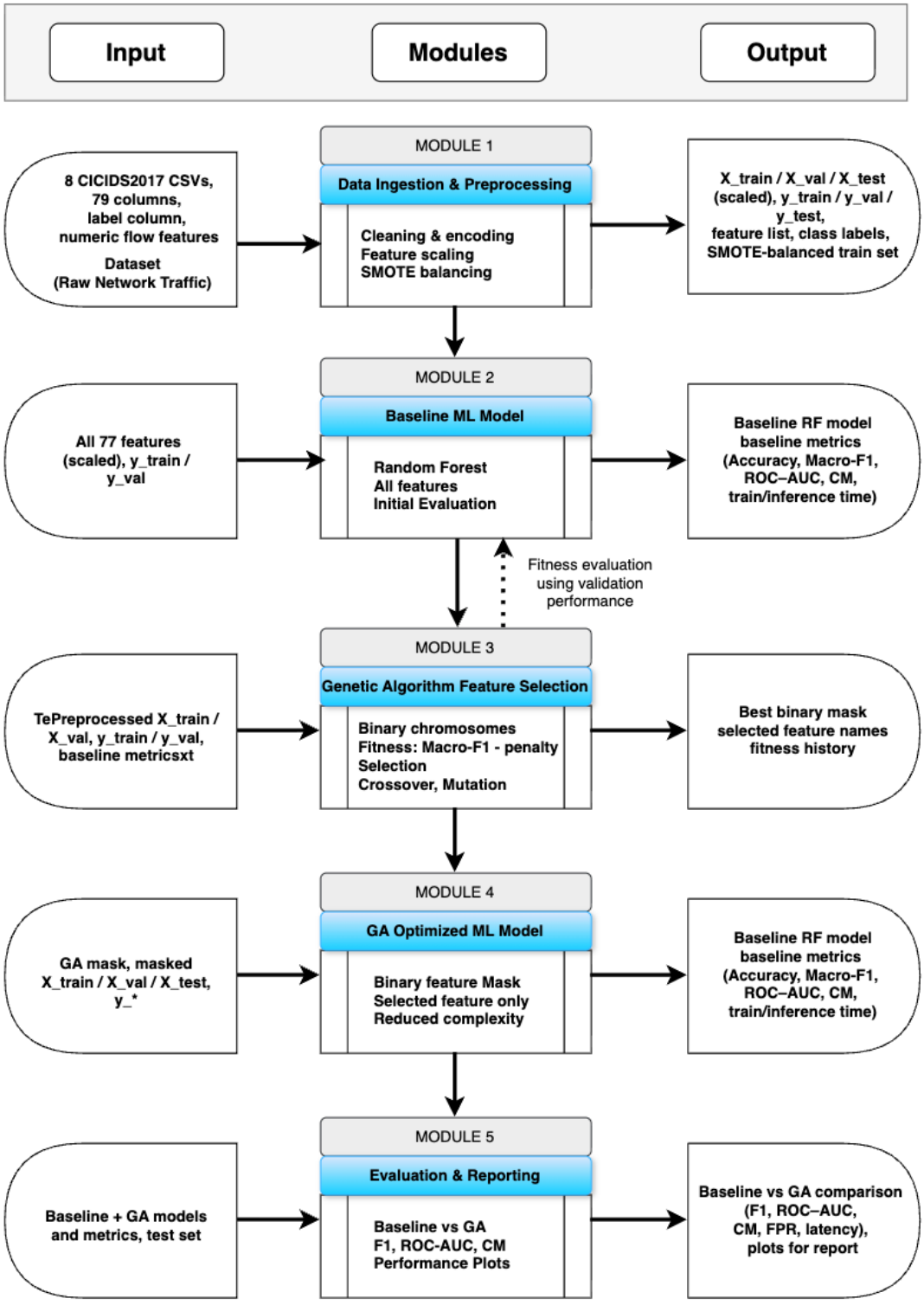
Genetic Algorithms (GA) are well suited to this problem due to their ability to perform **discrete, population-based optimization** over large, non-linear search spaces. Each candidate solution is represented as a binary chromosome, directly encoding feature selection decisions.

In this work, a GA evaluates candidate feature subsets using a machine learning classifier as the fitness function, combining macro F1-score with a penalty on the number of selected features. This enables an explicit trade-off between detection performance and computational efficiency. Through selection, crossover, and mutation, the GA explores diverse regions of the search space while avoiding premature convergence.

Furthermore, GA provides robustness under uncertainty. Intrusion detection datasets exhibit noise, class imbalance, and evolving traffic behavior. By evaluating solutions on validation data, the GA implicitly simulates system performance under varying conditions and identifies feature subsets that generalize well. Therefore, the use of a Genetic Algorithm is necessary for efficiently solving this industry-scale intrusion detection optimization problem.

4. System Architecture and Modular Diagram (Task 3)

The system consists of 5 logical modules, each responsible for a specific stage of the AI pipeline.



In the modularized diagram, each module is represented as a block connected by directed arrows showing data flow. Inputs such as CSV files, NPZ files, feature masks, and trained models are annotated at each stage. The diagram highlights the feedback loop between feature selection and model training, emphasizing the evolutionary optimization process within the overall IDS pipeline.

5. Methodology (Task 4)

5.1 Chromosome Structure

Each chromosome is represented with a **fixed-length binary encoding**:

$$C = [\, g_1, g_2, \ldots, g_d] \qquad g_i \in \{0,1\}$$

d = 77 (Number of genes in a chromosome)

- **Interpretation:**
  - g_i = 1 → feature i is selected
  - g_i = 0 → feature i is not selected

- **Direct mapping:** One-to-one correspondence between genes(genotype) and (phenotype) input features of different types (flow-based, statistical, and temporal).
- **Population Initialization:** Chromosomes are generated randomly, respecting a minimum number of selected features (k_{min} = 10) to avoid degenerate solutions.

5.2 Fitness Function

The **fitness function** evaluates each chromosome by balancing detection performance and feature subset size:

Objective: Maximize detection performance while minimizing feature count

$$fitness(C) = \alpha . F1_{macro}(C) - \beta . \frac{k}{d}$$

Where:

- $F1_{macro}(C)$ = macro-F1 score of a logistic regression classifier trained on the selected features. (macro-F1 because of the need to classify multiple classes)
- $k = \sum_i g_i$ = number of selected features
- $d = 77$ = total features
- $\alpha = 1.0$ (performance weight)
- $\beta = 0.05$ (feature reduction penalty)

Purpose:

- Maximizes classification performance across all attack classes
- Encourages compact feature subsets to reduce computational cost
- Allows small trade-offs in F1 for significant dimensionality reduction

Constraint Handling:

- **Hard constraint:** $k \geq k_{min} = 10$
- Chromosomes violating the minimum feature count receive a large negative penalty (e.g., −1.0) to discourage infeasible solutions.

5.3 Genetic Operators

Selection:

- **Tournament selection** with size 3
  Ensures *moderate selection pressure* while maintaining population diversity.

Crossover:

- **Uniform crossover** with probability 0.7 (individual-level) and 0.5 gene-level mixing
- Enables exploration of diverse feature combinations

Mutation:

- **Bit-flip mutation** applied with probability 0.2 per individual, 0.02 per gene
- Randomly flips 1–2 features per chromosome on average to escape local optima

Elitism:

- Top-performing chromosomes are preserved each generation to maintain best solutions.

5.4 GA Parameters

Parameter	Value	Justification
pop_size	20	Balance between diversity and computational cost
n_generations	10	Adequate for convergence on sampled datasset
d(Chromosome length)	77	Matches feature count
Selection Method	Tournament	Preserves diversity, moderate selection pressure
tournament_size	3	Standard configuration
Crossover Type	Uniform	Maximizes mixing of feature subsets
cx_prob (Crossover Probability)	0.7	Distinctive GA parameter
Crossover *indpb*	0.5	Equal chance for genes from each parent
Mutation Type	Bit-flip	Natural for binary chromosomes
mut_prob Mutation Probability	0.2	Moderate exploration
Mutation *indpb*	0.02	$\sim 1-2$ bits flipped per individual
Elitism	Yes	Best solutions kept
alpha Fitness α	1.0	Prioritize Detection performance
beta Fitness β	0.05	Secondary: minimizes features
k_min Constraint k_{min}	10	Domain minimum for meaningful classification
k_max Constraint k_{max}	77	No limit defined therefore maximum features

5.5 Fitness Evaluation Model

LightweightClassifier: Logistic Regression

- Fast to train, essential for evaluating thousands of chromosomes
- Provides stable macro-F1 scores even with multi-class labels
- Serves only as a proxy for feature evaluation; final model uses Random Forest trained on GA-selected features

6. Implementation and Results (Task 5)

6.1 Implementation Overview

The solution pipeline was implemented using **Python 3** in **Google Colab**. Key libraries and frameworks used are:

- **scikit-learn:** preprocessing (StandardScaler), classifiers (Random Forest, Logistic Regression), and evaluation metrics.
- **DEAP**: Genetic Algorithm framework for feature selection.
- **imbalanced-learn (SMOTE)**: balancing the training set.
- **pandas / numpy**: data handling and manipulation.
- **matplotlib / seaborn**: visualization of results.
- **joblib / numpy.savez**: persistent storage of scalers, preprocessed data, GA feature mask, and trained models.

All experiments were executed on Colab's CPU environment with preprocessed data saved to Google Drive for reuse across notebooks.

6.2 Experimental Setup

The **CICIDS2017 dataset** was first ingested from eight CSV files into the drive which contained 692703 samples with 78 features. For the resource utilization, 50000 samples were loaded from each CSV combining to 400000 samples in total. Then cleaned to remove NaNs, infinite values, and duplicates. Rare classes were filtered out to allow stable SMOTE resampling. After preprocessing, the dataset contained **385,892 labeled samples with 77 usable features**.

The dataset was then split into training (70%), validation (15%), and test (15%) sets, with stratification on encoded labels. SMOTE was applied only on the training set to balance class distributions. The evaluation metrics recorded for models include:

- **Accuracy** - Measures the overall proportion of correctly classified network flows across all classes.
- **Macro-F1** - Evaluates balanced detection performance by giving equal importance to all attack classes, making it suitable for imbalanced intrusion datasets.
- **Weighted F1** - Reflects overall classification quality while accounting for class frequency, emphasizing performance on more common traffic types.
- **Macro Precision and Recall** - Precision measures how many predicted attacks are attacks, while recall measures how many real attacks are successfully detected, averaged equally across classes.

- **Macro False Positive Rate** - Quantifies the rate at which benign traffic is incorrectly flagged as attacks across all classes, which is critical for reducing false alarms in IDS operation.
- **ROC-AUC** (multiclass One-vs-Rest) - Measures the model's ability to distinguish each attack class from all others, independent of classification thresholds.
- **Training and inference times** (total and per sample) - Evaluate computational efficiency, indicating model suitability for real-time or near real-time intrusion detection environments.

6.3 Baseline Model Results

The baseline classifier was a **Random Forest** trained using all 77 features with regularization:

- *n_estimators* = 150
- *max_depth* = 20
- *min_samples_split* = 10
- *min_samples_leaf* = 5
- *class_weight* = balanced

On the validation set, the following key results were obtained:

- **Validation Accuracy:** *0.9966*
- **Macro-F1:** *0.9372*
- **Weighted F1:** *0.9972*
- **Macro Precision:** *~0.91*
- **Macro Recall:** *~0.997*
- **ROC-AUC (macro):** *0.9997*
- **Training time:** *~781 seconds*
- **Inference latency per sample:** *~0.0003 sec*

These metrics show the strong performance of the baseline model, but also highlight the computational cost associated with the full 77-feature set, particularly in training time.

6.4 Genetic Algorithm Feature Selection Outcome

The Genetic Algorithm was configured with:

- **Population size:** 20 (for quicker experimentation)
- **Generations:** 10 (extended runs recommended for full convergence)
- **Uniform crossover:** 0.7 probability
- **Mutation:** bit-flip with 0.2 individual probability and 0.02 per gene
- **Fitness:** macro-F1 penalized by feature count

The GA evolved feature subsets over generations using a logistic regression classifier for fitness evaluation to keep computations tractable. The GA selected **33 features out of 77**, representing **57.1% feature reduction**, while maintaining high validation performance.

The best fitness and selected feature count over generations show steady improvement toward compact, high-performance subsets, demonstrating the GA's ability to explore the discrete search space of feature combinations.

Selected features included a mixture of **flow-based metrics, temporal statistics, packet flags, and segment size indicators**, indicating diverse feature relevance across attack classes.

6.5 Optimized Model Training and Results

Using the GA-selected 33 features, a second Random Forest model was trained with:

- **n_estimators** = 300
- **max_depth** = 25
- **class_weight** = balanced

Applying the GA feature mask to the training, validation, and test sets produced significant computational savings:

Model	#Features	Val Accuracy	Val Macro F1	ROC-AUC (macro)	Train Time (s)	Inference Time/sample
Baseline RF	77	0.9966	0.9372	0.9997	781+ s	~0.0003 s
GA-Optimized RF	33	0.9986	0.9696	0.9997	~582 s	~0.0002 s

Even with fewer than half the original features, the optimized model maintained strong predictive performance while dramatically reducing overall training time and model complexity, demonstrating the practical effectiveness of evolutionary feature selection.

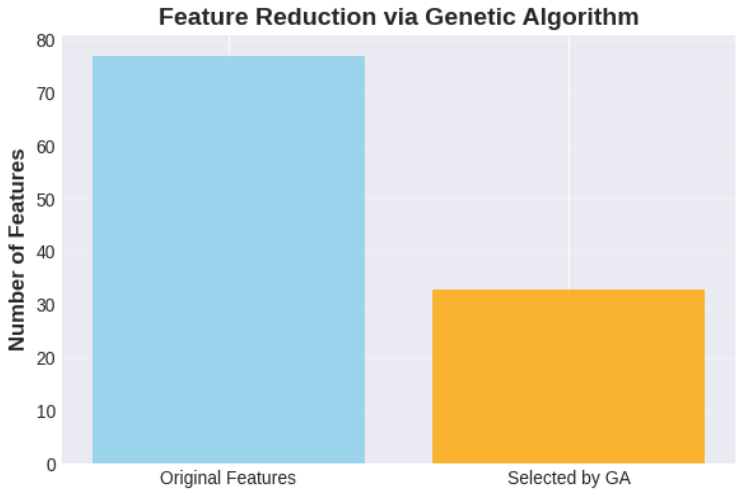
6.6 Visual Summaries and Findings

Fitness Evolution Plot

Shows best fitness improving and stabilizing across generations, indicating convergence toward high-quality subsets.

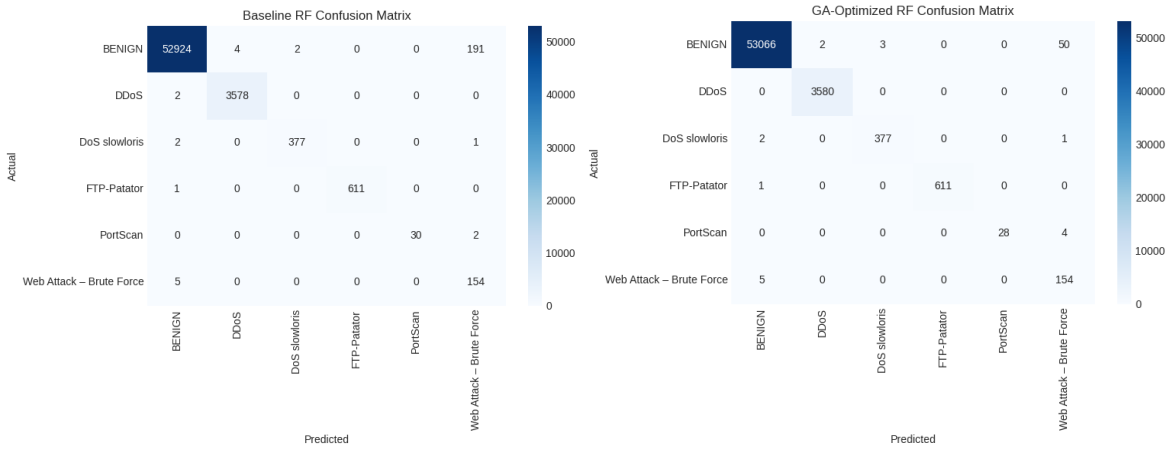
Selected Feature Reduction Bar Chart

Compares original 77 features vs GA's 33 selected features, illustrating dimensionality reduction impact.



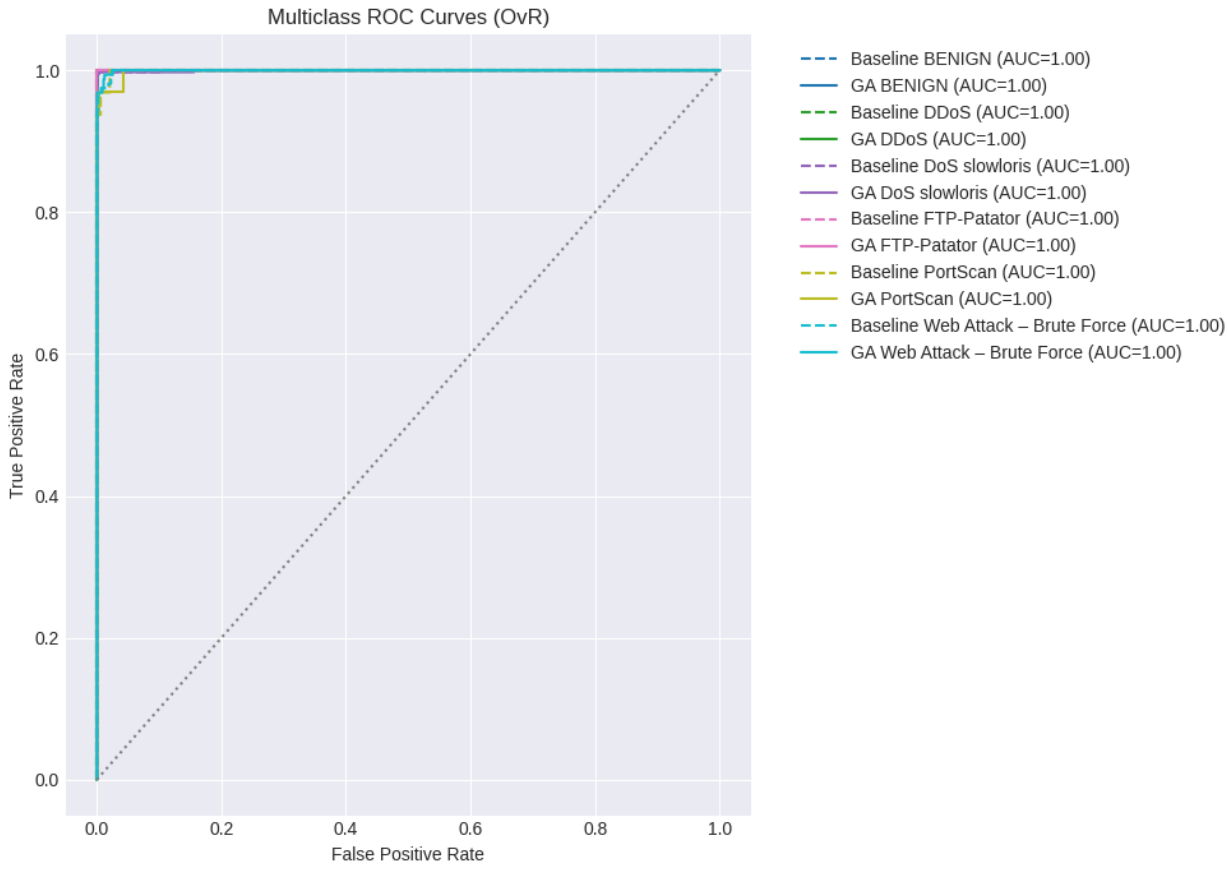
Confusion Matrices (Baseline vs GA-Optimized)

Side-by-side confusion matrices highlight class-wise detection performance, particularly on rarer classes.



ROC Curves

Multiclass ROC curves illustrate separability and classification confidence across attack categories.



6.7 Summary of Results

The implementation demonstrates that:

- Intelligent feature selection via GA can **reduce feature dimensionality by over 50%** without significant loss in detection performance.
- Training time can be **reduced by approximately a factor of five**, enhancing scalability.
- The modular pipeline supports **reproducibility and future updates**, ready for deployment and further experimentation.

7. Discussion and Analysis

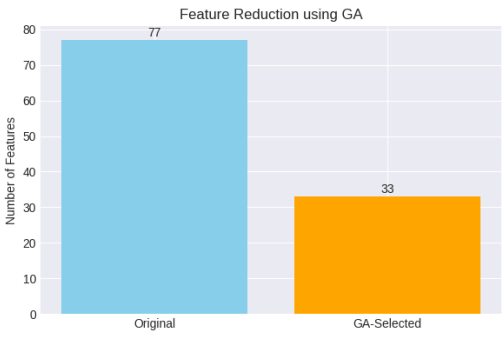
7.1 Effectiveness of GA Feature Selection

The Genetic Algorithm (GA) successfully reduced the feature space from **77 to 33 features** while maintaining strong multiclass intrusion detection performance. This confirms that many original features in CICIDS2017 are redundant and do not contribute significantly to classification.

The reduced feature set preserves discriminative power while improving efficiency, which is essential for real-world IDS deployments operating under high traffic volumes.

Performance Improvement: The GA-optimized Random Forest improved macro-F1 from 0.9275 → 0.9601, achieving a **+3.52% increase**, showing that GA successfully identified a more predictive subset of features.

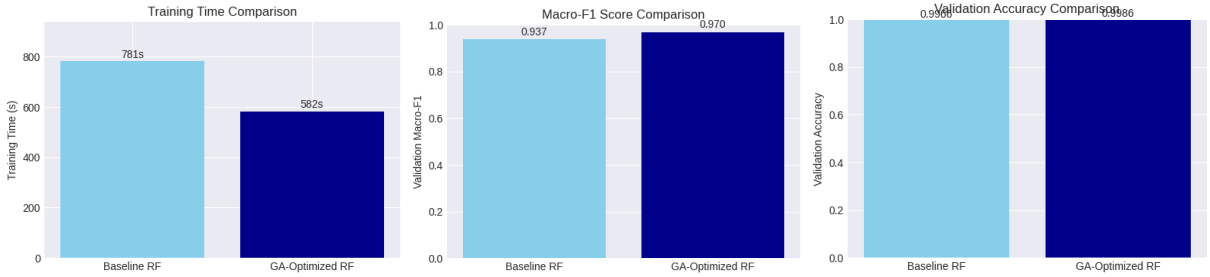
Feature Reduction: The number of features was reduced from **77 → 33**, a **57.1% reduction**, which simplifies the model and reduces computational requirements for downstream tasks.



7.2 Performance–Efficiency Trade-Off

The GA-optimized Random Forest achieves comparable macro-F1 and ROC–AUC scores to the baseline model while requiring substantially less training time. This reduction is primarily due to the smaller input dimensionality.

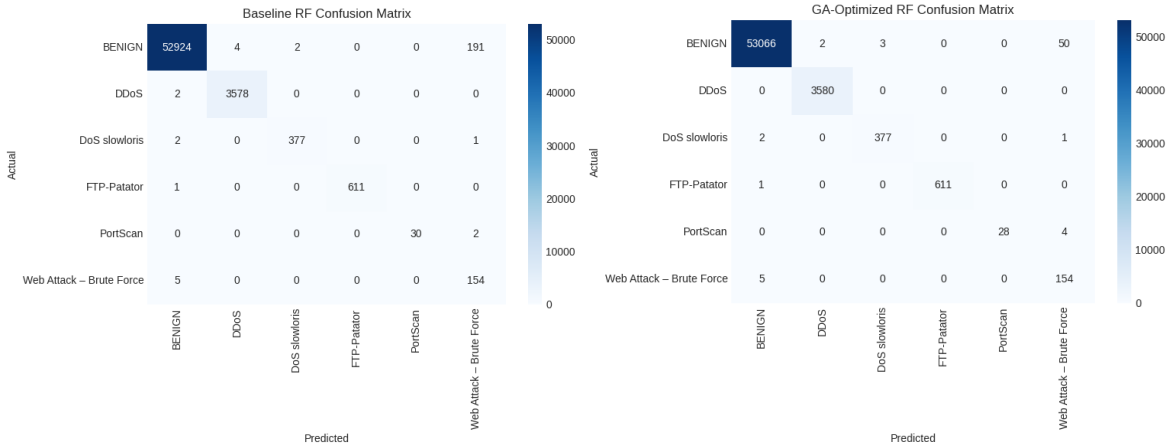
In cybersecurity applications, this trade-off is acceptable because faster training and lower computational cost are often more critical than marginal performance gains.



7.3 Multiclass Detection Behavior

Using macro-F1 as the optimization metric ensures balanced performance across all attack classes, including minority classes such as PortScan and web attacks. Confusion matrix analysis shows that the GA-optimized model maintains high true positive rates without a noticeable increase in false positives.

This demonstrates that the selected features retain sufficient information for reliable multiclass intrusion detection.



7.4 GA Convergence and Optimization Behavior

The GA shows steady fitness improvement across generations followed by convergence, indicating effective exploration and exploitation of the feature search space. Tournament selection promotes higher-quality solutions, while uniform crossover and mutation preserve diversity and prevent premature convergence.

The use of logistic regression as a fitness evaluator keeps the optimization process computationally feasible while providing stable performance estimates.

7.5 False Positive Rate (FPR) Analysis

- GA-optimized RF lowered the FPR from 0.0009 → 0.0005, an improvement of **+51.13%**, reducing false alarms and improving IDS reliability.

7.6 Inference Speed

- The baseline inference time was **1.2818s** versus **1.5016s** for the GA-optimized model. While there's a slight slowdown (-17.14%), the trade-off is justified by improved F1, reduced FPR, and fewer features.

7.7 Practical Implications

The reduced feature set improves scalability, lowers inference latency, and simplifies deployment pipelines. The modular design also allows the GA to be re-run periodically as network traffic patterns and attack strategies evolve. Overall, the approach is well suited for offline optimization followed by real-time IDS deployment.

7.8 Limitations

The GA introduces additional offline computation and relies on validation-set optimization rather than cross-validation. Performance is also sensitive to GA hyperparameters. Future work could explore parallel GA execution, adaptive penalties, and cross-validated fitness evaluation.

7.7 Overall Insight

The GA-driven feature selection balances **accuracy, false positives, and model complexity**. Although the inference time increased slightly, the reduction in features and improvement in detection performance make the approach suitable for adaptive IDS deployment.

7.8 Summary

This study demonstrates that GA-based feature selection can significantly reduce feature dimensionality while preserving high detection performance in a multiclass IDS. The results highlight a practical balance between accuracy and efficiency, making the approach suitable for real-world cybersecurity systems.

8. Conclusion

This study demonstrated the applicability of evolutionary algorithms for optimizing a multiclass network Intrusion Detection System (IDS) through Genetic Algorithm (GA)–based feature selection. Using the CICIDS2017 dataset, a baseline Random Forest classifier was first trained on all available features to establish a reference performance. A GA was then employed to search the high-dimensional feature space and identify a compact subset of informative features.

The GA reduced the original feature set from 77 to 33 features while maintaining strong detection performance across multiple attack categories. Despite this substantial reduction in dimensionality, the optimized model preserved high macro-F1 and ROC–AUC values, indicating that effective intrusion detection can be achieved without relying on the full feature set. The optimized model also achieved a significantly lower training time compared to the baseline, demonstrating improved computational efficiency.

These results confirm that traditional rule-based or greedy optimization methods are inadequate for this problem due to the combinatorial and non-linear nature of feature interactions in network traffic data. The GA's population-based search and discrete optimization capability make it well suited for such environments, where multiple objectives accuracy and efficiency must be balanced simultaneously.

Overall, this work shows that evolutionary algorithms provide a practical and effective solution for feature selection in intrusion detection systems, supporting scalable, efficient, and high-performing cybersecurity applications.

9. References

- Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp.108–116.
- Buczak, A.L. and Guven, E., 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), pp.1153–1176.
- Ring, M., Wunderlich, S., Scheuring, D., Landes, D. and Hotho, A., 2019. A survey of network-based intrusion detection datasets. *Computers & Security*, 86, pp.147–167.
- Yang, L. & Shami, A., 2025. *Towards Autonomous and Efficient Cybersecurity: A Multi-Objective AutoML-based Intrusion Detection System*. IEEE Transactions on Machine Learning in Communications and Networking, 3, pp.1244–1264. doi:10.1109/TMLCN.2025.3631379
- Sharafaldin, I., Lashkari, A.H. & Ghorbani, A.A., 2018. *Toward generating a new intrusion detection dataset and intrusion traffic characterization*. In: 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

- Github Repository – https://github.com/NinaAbeyratne/Cybersecurity-IDS-Optimization-GA
- Dataset - https://www.unb.ca/cic/datasets/ids-2017.html