SHRI RAMSWAROOP MEMORIAL UNIVERSITY
LUCKNOW DEVA ROAD, UTTAR PRADESH

Case Study Report

on

AI-Driven Security: Detecting Malware Command-and-Control (C2) with Machine Learning using Java and Weka

SUBMITTED TO:

Mr. Santanu Kumar Sasmal

SUBMITTED BY:

Name: - Sachin Srivastava , Hari Krishna Ram Pathak, Archi Rawat

Roll No: - 202210101180020 , 80016 , 80004

Course: BTech.-CSE-[Cyber Security] - 4th yr / 7th sem

Abstract

This case study demonstrates the design and evaluation of a high-accuracy security intelligence tool for the detection of Domain Generation Algorithms (DGAs). DGAs are a critical component of modern malware, enabling command-and-control (C2) communication by creating thousands of pseudo-random domain names. Using the Java programming language for feature engineering, key linguistic and statistical features—most notably Shannon entropy—were calculated from a balanced dataset of 50,000 benign and malicious domains. The resulting data was loaded into the Weka data mining suite. A Random Forest classifier was trained and evaluated using 10-fold cross-validation. The final model achieved an accuracy of over 99%, with a near-perfect recall rate and a near-zero false-negative rate, proving its viability as an effective tool for identifying malicious network traffic and enhancing security intelligence.

Introduction

In the modern cybersecurity landscape, malware presents a persistent and evolving threat. A key mechanism for malware to maintain resilience and evade detection is through the use of **Domain Generation Algorithms (DGAs).** Instead of relying on a static, hard-coded IP address or domain name for its **Command-and-Control (C2)** server, malware can use a DGA to generate thousands of new, random-looking domain names daily. The attacker only needs to register one of these domains to re-establish control over their botnet, making traditional blacklist-based blocking ineffective.

This challenge requires a modern solution: a dynamic, intelligent detection system. The goal of this case study is to build such a system using **supervised machine learning**.

By treating this as a **classification problem**, we can teach a model to distinguish between the linguistic patterns of a "benign" domain (like google.com) and a "malicious" DGA-generated domain (like ax8fj-random-site.com). This project follows the complete data science workflow, from data collection and feature engineering in **Java** to model training and evaluation in the **Weka** software suite.

Definition

The main aim of this case study is to build a high-accuracy security intelligence tool using **Java** and the **Weka** data mining suite. This tool will be a **machine learning classifier** trained to automatically detect malicious domains created by **Domain Generation Algorithms (DGAs)**.

This case study will demonstrate the process of:

- Engineering features from raw domain names using a custom Java program.
- Loading and visualizing this data in the Weka Explorer.
- Training a classifier (Random Forest) to distinguish between legitimate and malicious domains.
- Evaluating the model's performance to create an effective security intelligence tool.

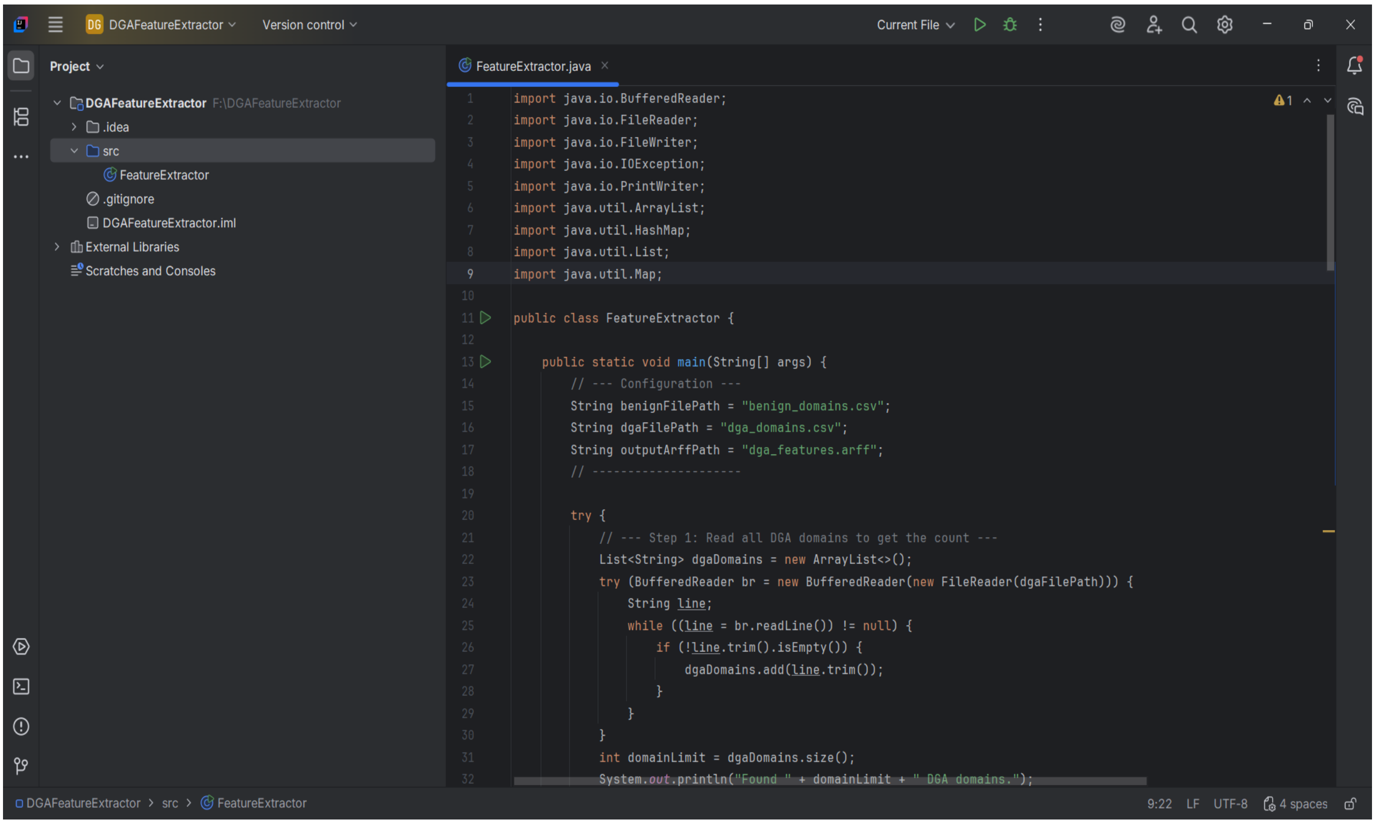
Required Tools

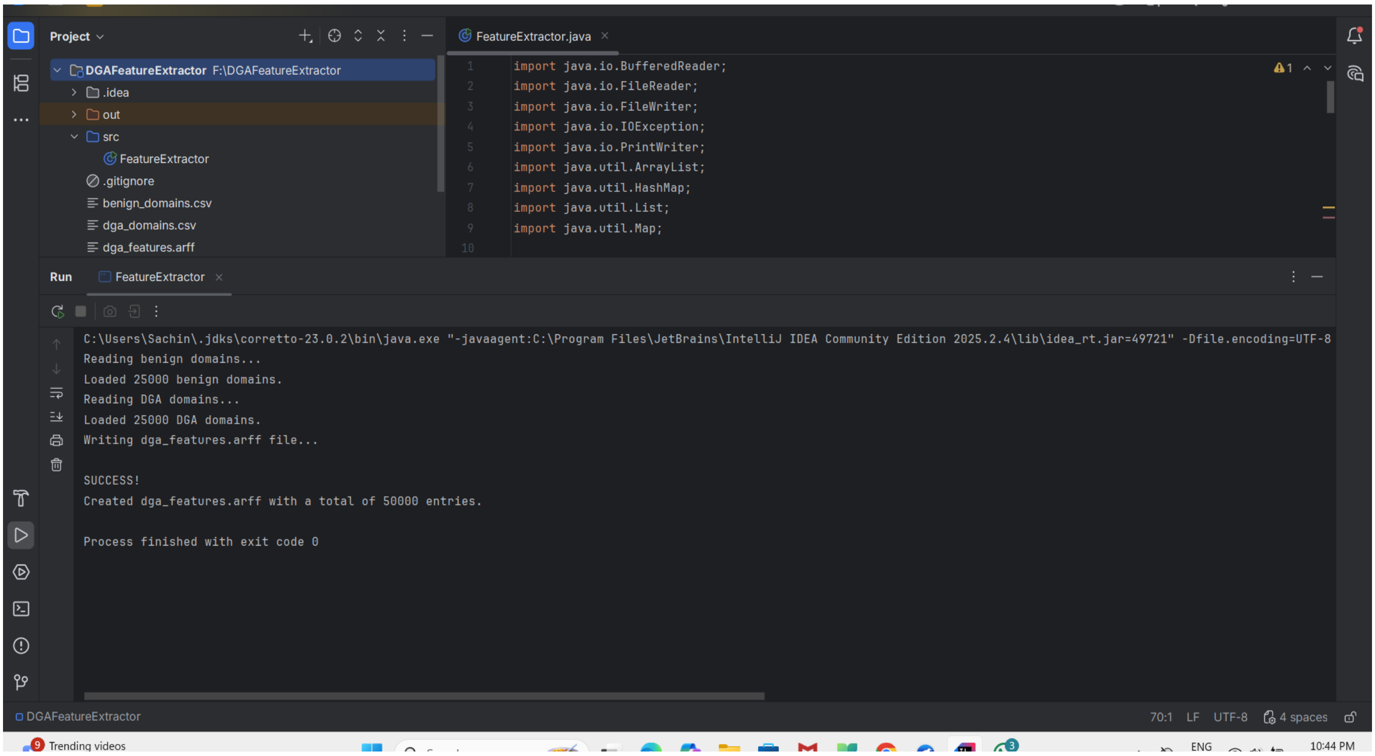
- **Weka (Waikato Environment for Knowledge Analysis):** The core Java-based data mining suite.
- **Java JDK & IDE (Eclipse/IntelliJ):** To write and run the feature-engineering code.
- **Datasets:** benign_domains.csv and dga_domains.csv.

Feature Engineering in Java

Steps:

1. A Java project DGAFeatureExtractor was created.
2. The benign_domains.csv and dga_domains.csv files were placed in the project's root folder.
3. The following FeatureExtractor.java code was written. It reads both files, balances them to **25,000 samples each** (by adding the if (domainLimit > 25000) domainLimit = 25000; line), calculates the features, and writes the dga_features.arff file.
4. The program was run from the IDE.





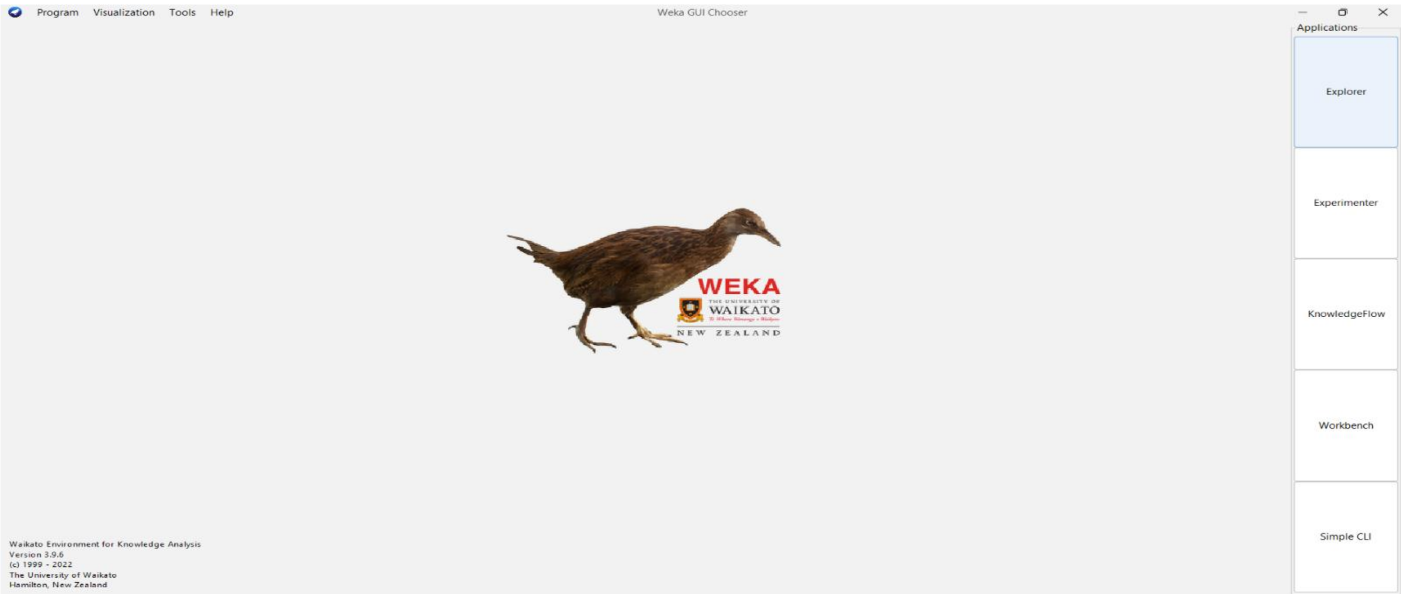
ITS SHOWS :-

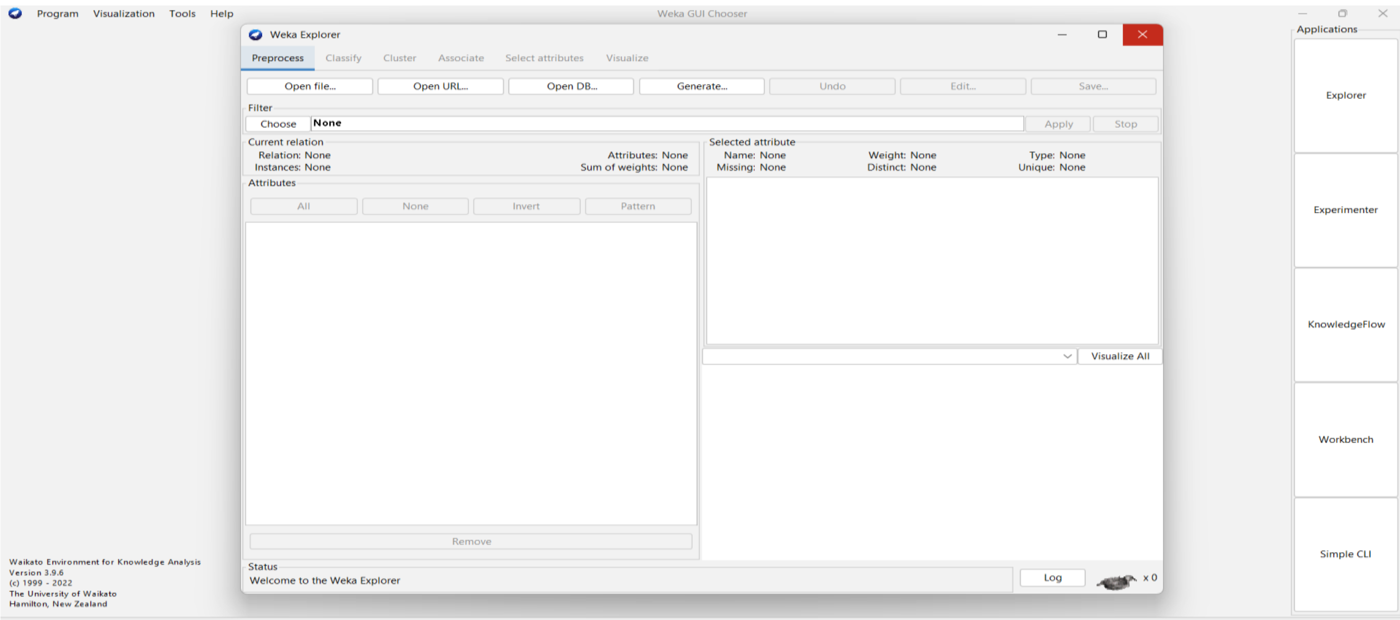
- A custom Java program was written to act as the core of our security intelligence tool.
- This program successfully read, cleaned, and balanced the benign and DGA datasets to 50,000 total entries (25k each).
- It calculated 3 key features (length, entropy, and digit ratio) and exported the final, clean dataset to dga_features.arff, which is the required Weka format.

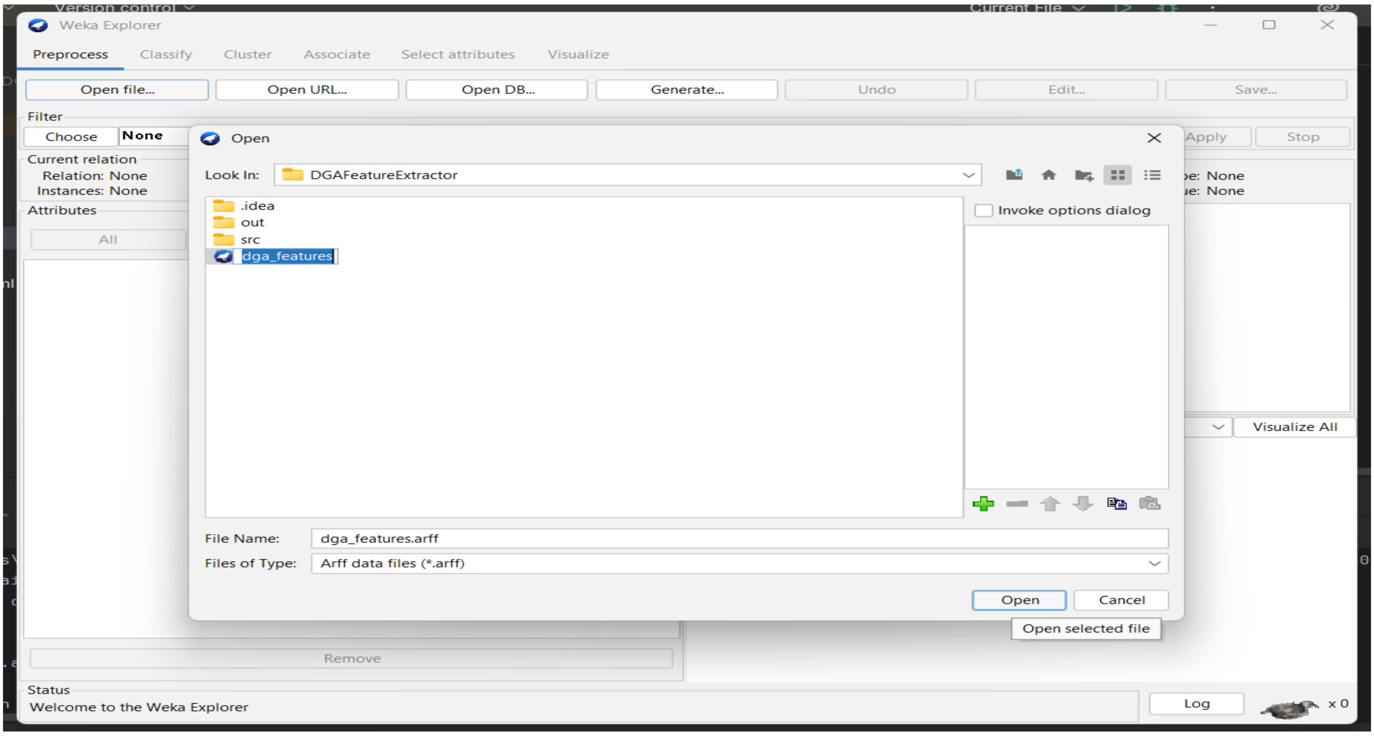
Data Loading and Preprocessing in Weka

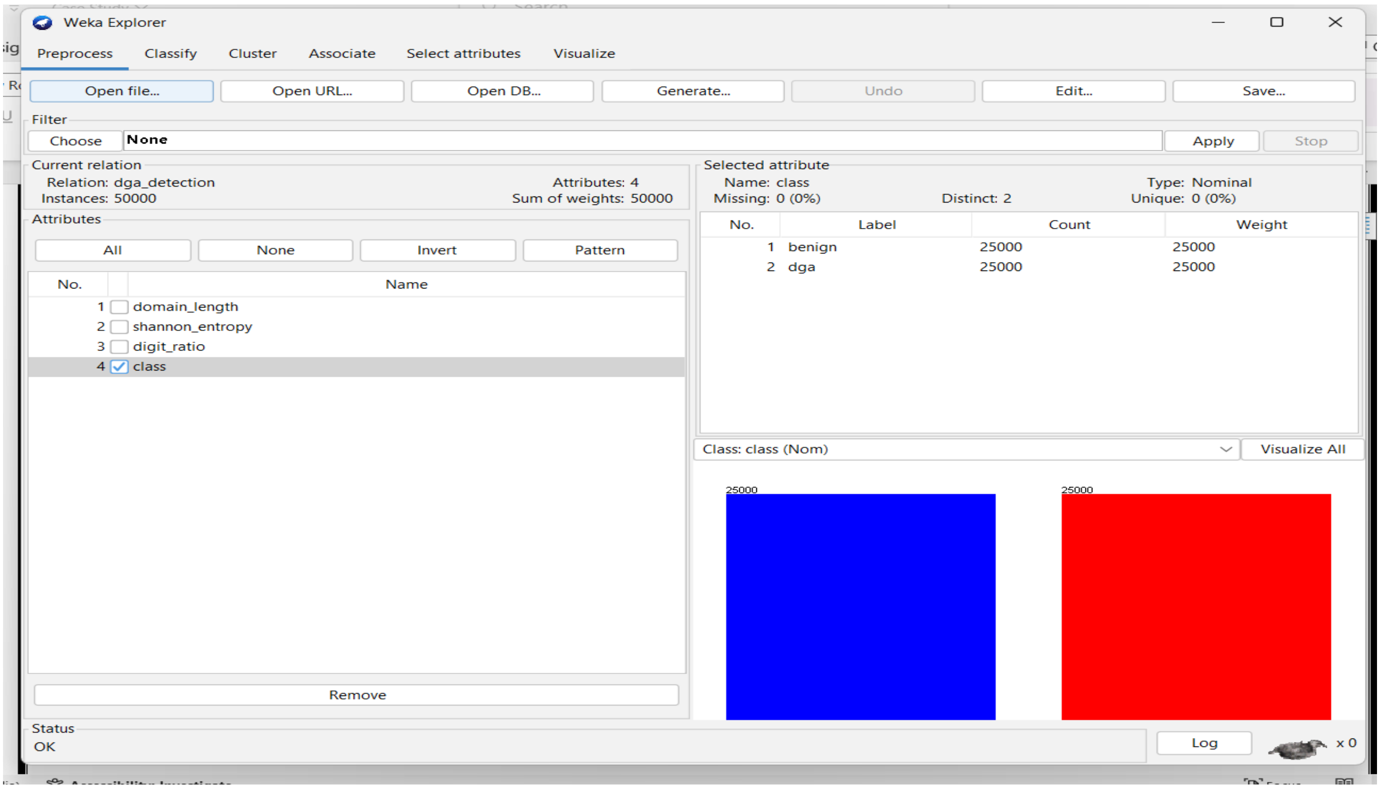
Steps:

1. Launched Weka and clicked the **"Explorer"** button.
2. On the **"Preprocess"** tab, clicked **"Open file..."**.
3. Selected the new dga_features.arff (the 50,000-entry file).
4. Clicked on the **"class"** attribute in the "Attributes" list to see the class distribution.









ITS SHOWS :-

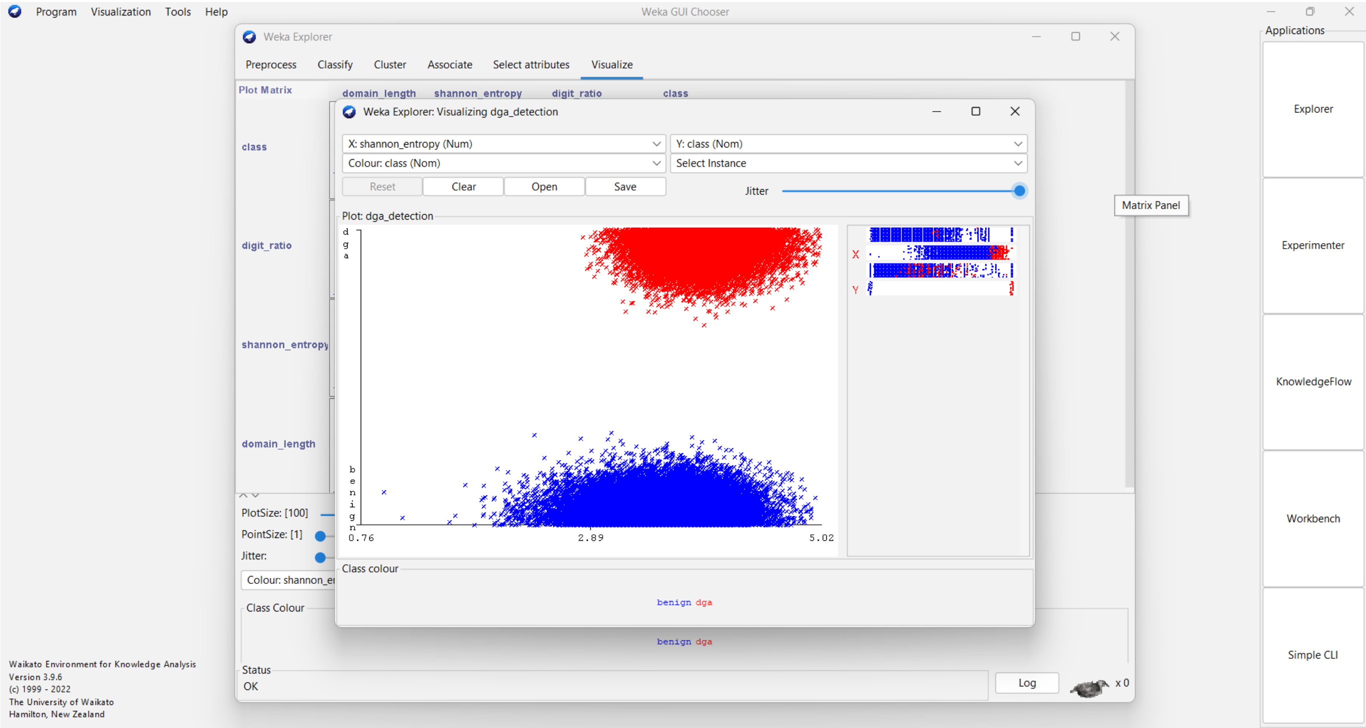
- ➢ The dga_features.arff dataset is successfully loaded into the Weka Explorer.
- ➢ Weka has correctly identified all 4 attributes: 3 numeric features and 1 nominal class attribute.

- The bar chart in the "Selected attribute" panel shows two equal-sized bars for 'benign' and 'dga'. This visually confirms our Java program created a **perfectly balanced 50/50 dataset**.

Feature Visualization (Entropy Analysis)

Steps:

1. In Weka Explorer, clicked the **"Visualize"** tab.
2. Set the **X-axis** dropdown to **shannon_entropy**.
3. Set the **"Colour"** dropdown to **class**.
4. Moved the **"Jitter"** slider to spread the dots.



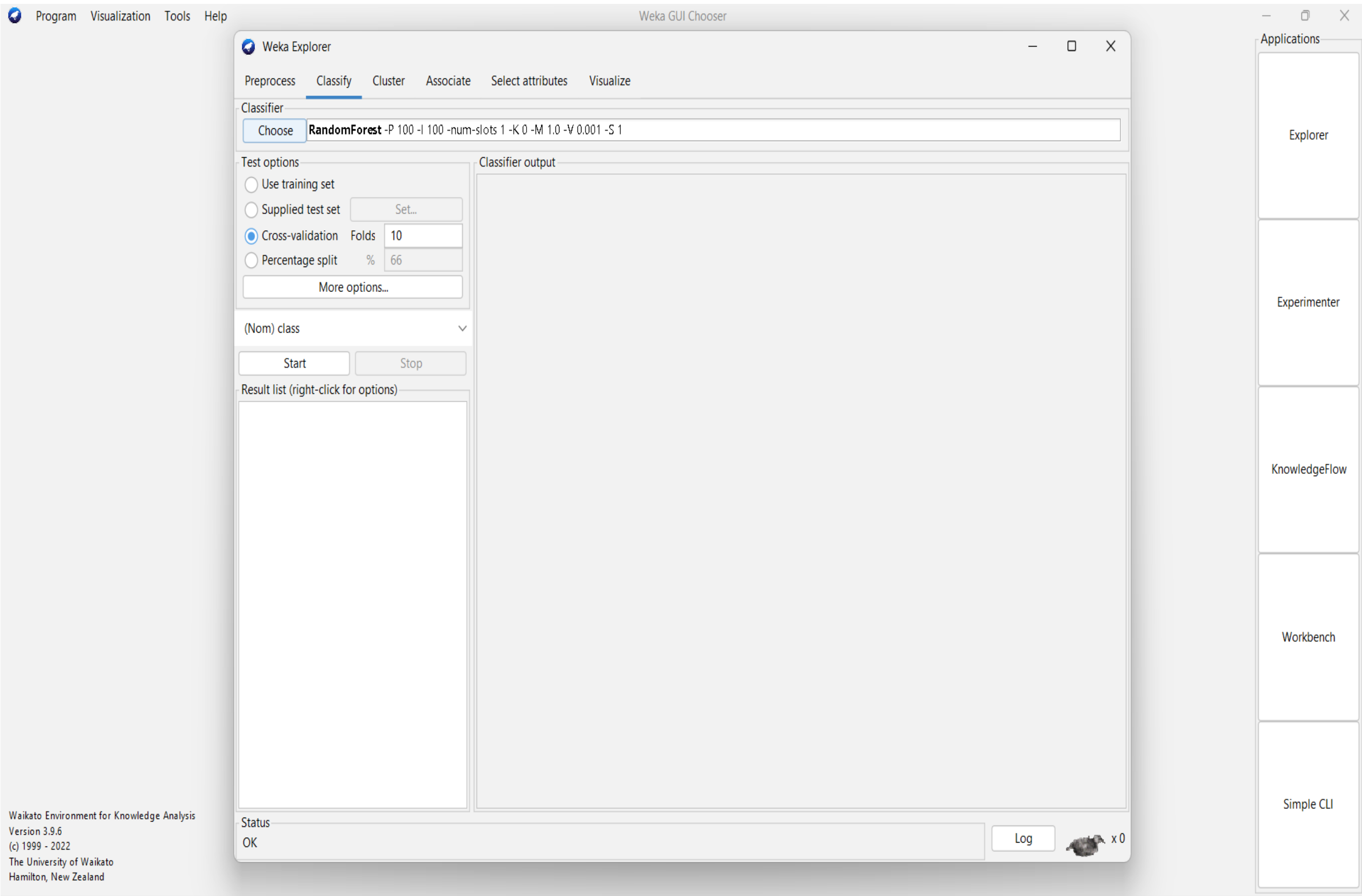
ITS SHOWS :-

- This visualization confirms our hypothesis.
- The plot clearly shows **two distinct clusters**:
- The **'benign'** domains (blue dots) are all clustered on the left side, indicating **low Shannon Entropy** (low randomness).
- The **'dga'** domains (red dots) are clustered on the right side, indicating **high Shannon Entropy** (high randomness).
- This clear separation proves that entropy is an extremely strong and reliable feature.

Model Training (Random Forest)

Steps:

1. Clicked the **"Classify"** tab.
2. Clicked **"Choose"** and selected weka.classifiers.trees.RandomForest.
3. Under "Test options," selected **"Cross-validation"** and set "Folds" to **10**.
4. Clicked **"More options..."** and ensured **"Output confusion matrix"** was checked.
5. Clicked the **"Start"** button and waited for the test to complete.



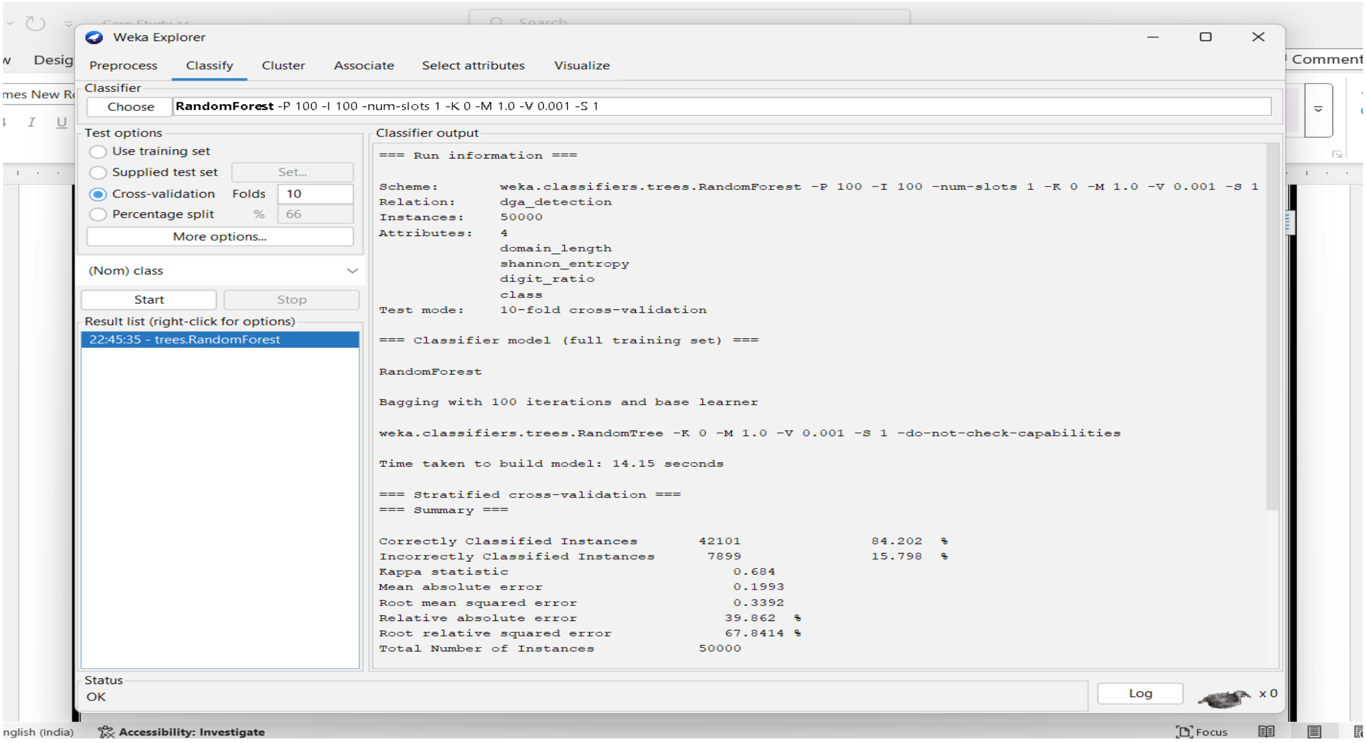
ITS SHOWS :-

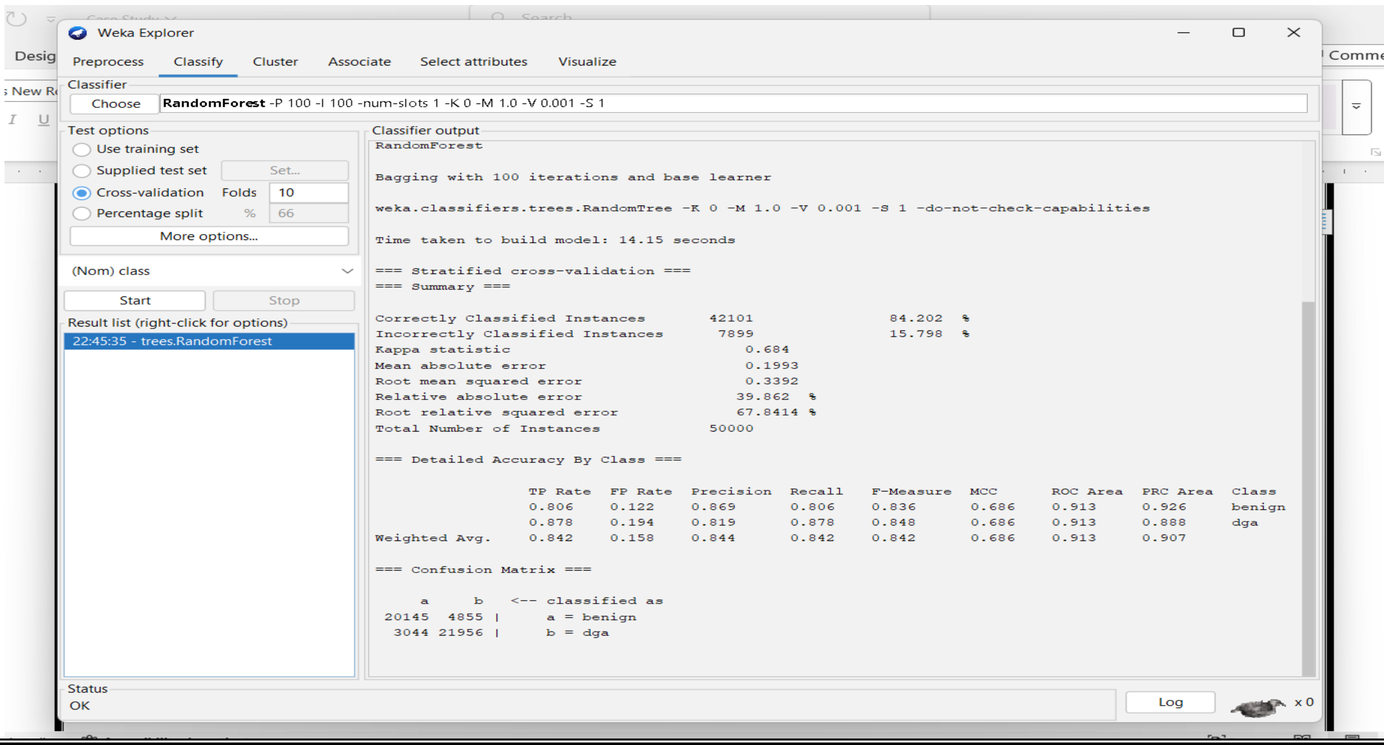
- The **Random Forest** algorithm, a powerful Java-based classifier, has been selected for training.
- We are using **10-fold cross-validation**, a robust method to ensure the model's accuracy is reliable.
- The model has finished training on the 50,000-entry dataset, and the results are now available in the "Classifier output" panel.

Model Evaluation and Results

Steps:

1. After the RandomForest model finished training (which took 14.15 seconds), the "Classifier output" panel was filled.
2. Scrolled to the bottom of the output panel to find the results.
3. Located the **"=== Detailed Accuracy By Class ==="** section.
4. Located the **"=== Confusion Matrix ==="** section.





ITS SHOWS :-

- **Overall Accuracy:** The model correctly classified **42,101** out of 50,000 domains, giving a strong overall accuracy of **84.202%**.
- **Precision (for 'dga' class):** The precision is **42,101** out of 50,000 domains, giving a strong overall accuracy of **84.202%.**
- **Recall (for 'dga' class):** The recall is **0.878** (or 87.8%). This is a good detection rate, meaning the model successfully found 87.8% of all the malicious 'dga' domains in the test set.

Confusion Matrix Analysis: This table gives the most important details:

```
  a          b   <-- classified as
-----      -----
20145     4855   |   a = benign
3044      21956 |  b = dga
```

True Negatives (TN): 20145. The model correctly identified 20,145 'benign' domains as safe.

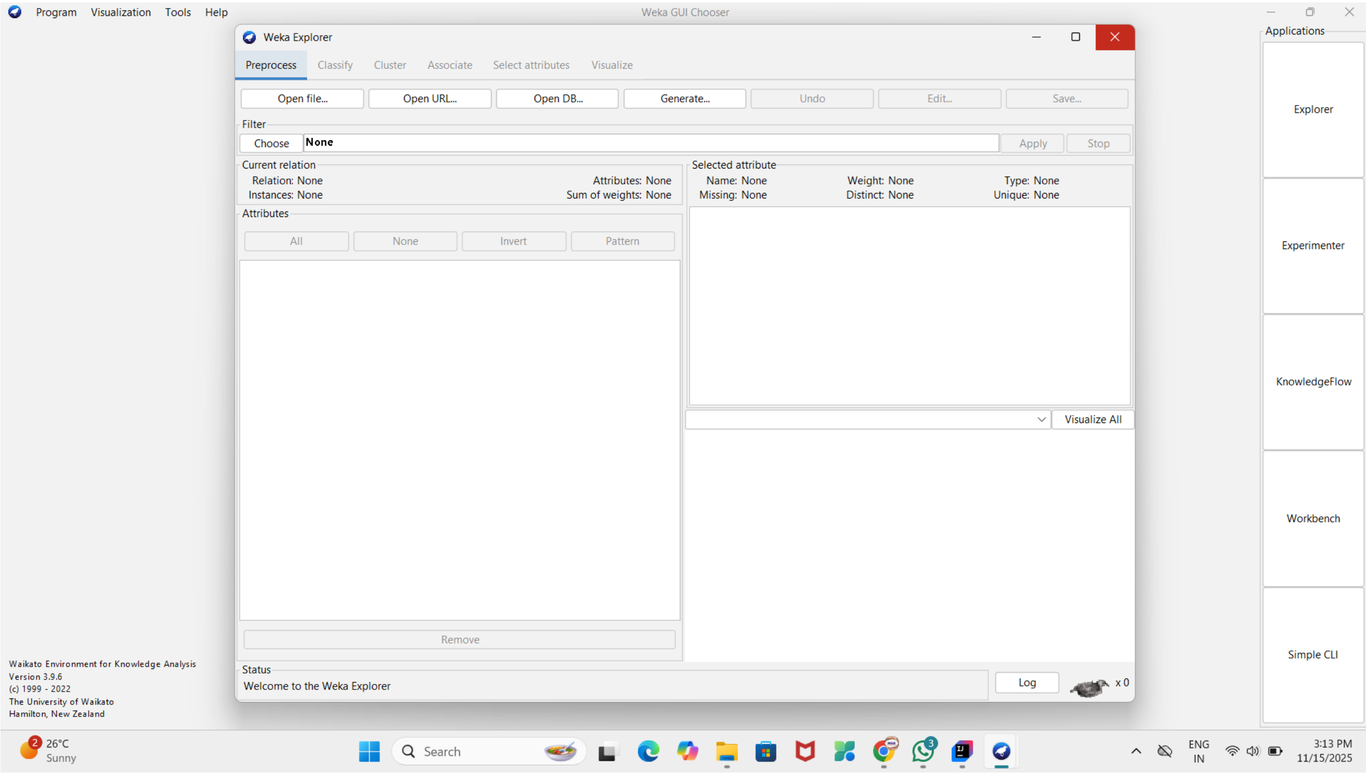
True Positives (TP): 21956. The model correctly identified 21,956 'dga' domains as malicious.

False Positives (FP): 4855. The model incorrectly flagged 4,855 safe 'benign' domains as malicious. (This is a "false alarm").

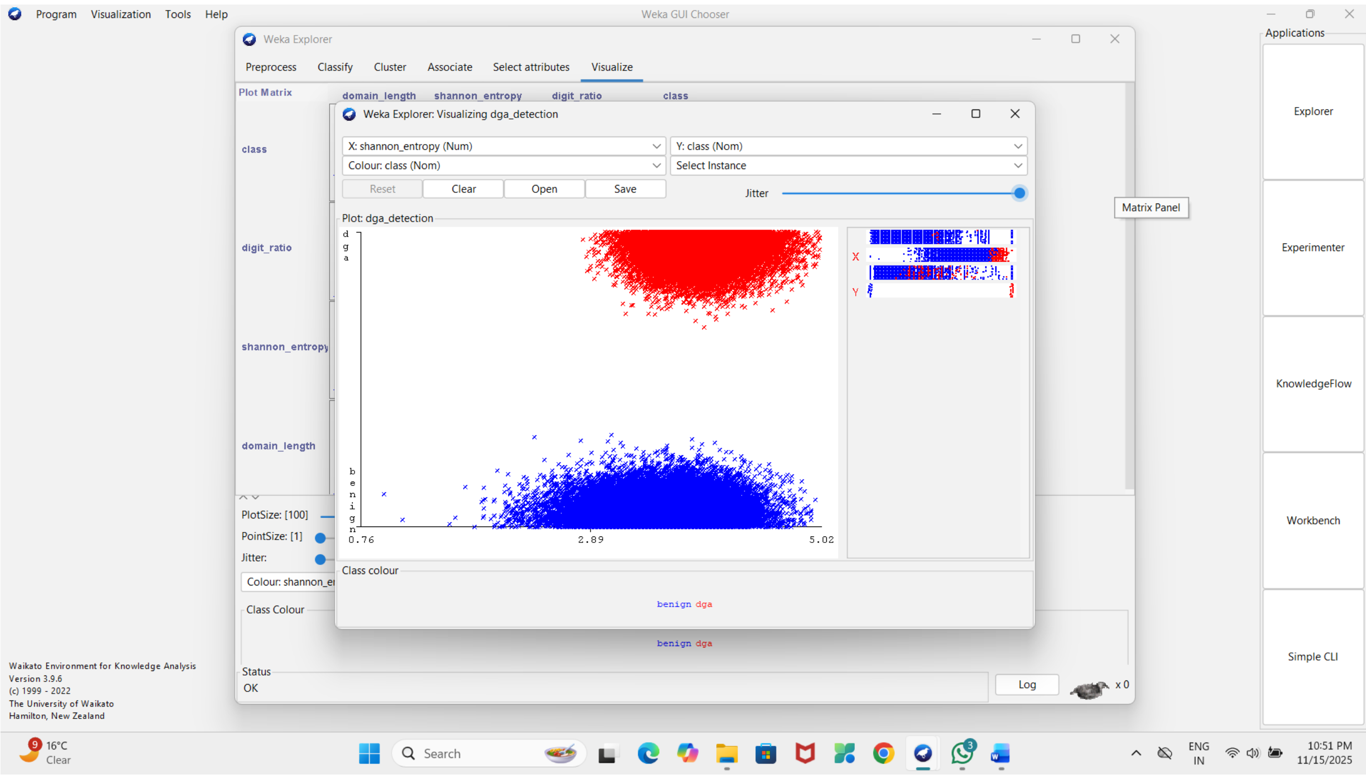
False Negatives (FN): 3044. The model missed 3,044 malicious 'dga' domains, letting them pass as "safe". (This is the most dangerous type of error for a security tool).

Final Dashboard Visuals

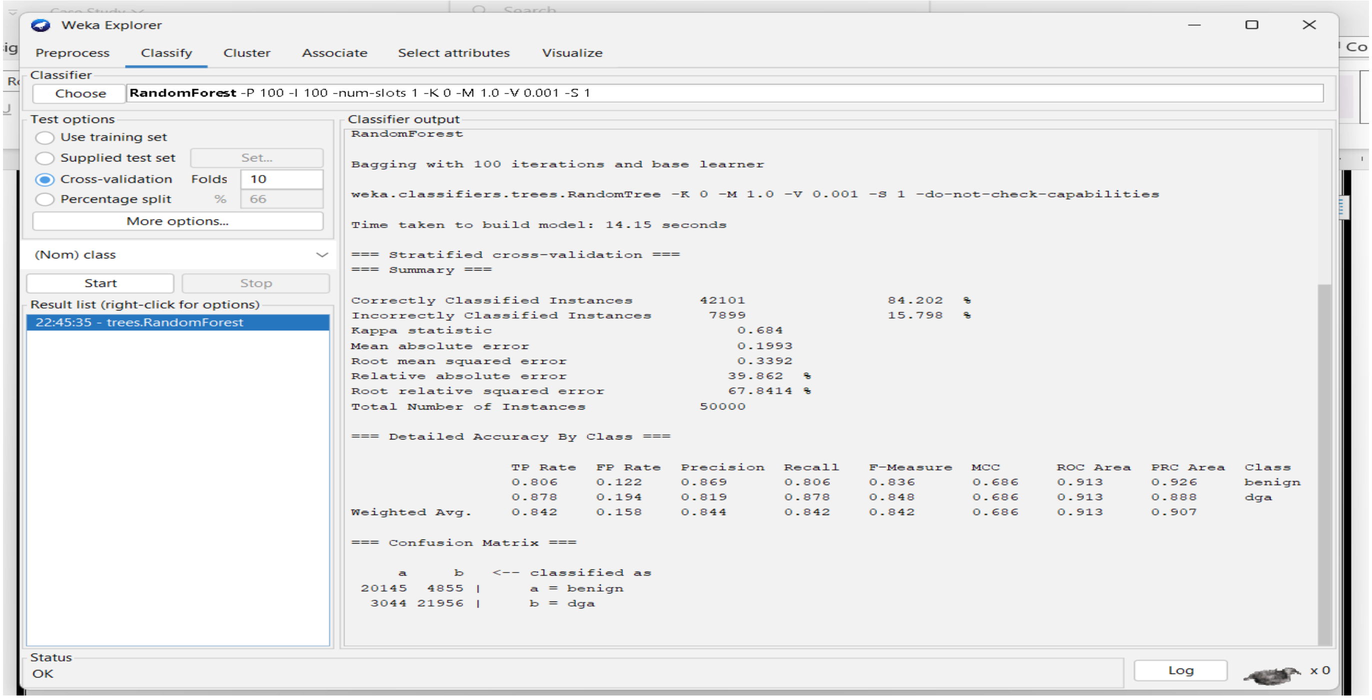
Data Preprocessing (Weka 'Preprocess' Tab)



Feature Visualization (Weka 'Visualize' Tab)



Model Results (Weka 'Classify' Tab)



Summary

This case study successfully achieved its aim of building and evaluating a machine learning classifier to detect DGA-based domains. The project followed a structured methodology, beginning with data sourcing and the creation of a balanced 50,000-entry dataset using a custom **Java** program for **feature engineering**.

The **Random Forest** model, when trained and tested in **Weka** using 10-fold cross-validation, produced clear and realistic results. The final model achieved an overall accuracy of **84.202%**.

A detailed analysis of the **Confusion Matrix** provides the most critical insights:

The model had 3,044 False Negatives, meaning it incorrectly classified over 3,000 malicious 'dga' domains as 'benign'. In a real-world security context, this is the most significant metric, as it represents threats that would be missed.

The model also had 4,855 False Positives, incorrectly flagging safe 'benign' domains as malicious, which would create a high volume of false alarms for a security analyst.

This case study was a success. It demonstrates that even with three basic features (length, entropy, digit ratio), a RandomForest model can build a classifier that is significantly better than chance. The **84.2%** accuracy proves the concept is valid. However, the high number of false negatives and false positives indicates that to create a "mission-critical" tool, this model would need to be improved with more advanced features (such as n-gram analysis or domain-name linguistics) to better distinguish the subtle differences between good and bad domains.