National University of Computer and Emerging Sciences

Department: Data Science

Name of Assignment: ASSIGNMENT # 2&3

Roll Number: i212746 & i211695

Name: Muhammad Talal & Hassaan Ullah Khan

Section: “DS-U”

Subject: AI

Date of Submission: 17/04/2024

Contents

Report on Intrusion Detection System using Genetic Algorithm

Introduction:

The aim of this project was to develop an Intrusion Detection System (IDS) using a Genetic Algorithm (GA) approach. The IDS was designed to classify network connections as either normal or malicious (attack). The dataset used for this purpose was the UNSW-NB15, which contains network traffic data with various types of attacks.

Methodology:

- **Dataset Preprocessing:** The dataset underwent preprocessing to extract relevant features and prepare it for input into the GA model.
- **Genetic Algorithm Implementation:** A GA was utilized to evolve a set of rules (chromosomes) that could effectively distinguish between normal and malicious network traffic. The GA consisted of encoding schemes, population initialization, fitness evaluation, selection, crossover, and mutation operations.
- **Training and Evaluation:** The GA was trained on the preprocessed training dataset and evaluated on a separate test dataset to assess its performance in terms of accuracy and classification metrics.

Implementation:

- **Genetic Algorithm Parameters:** The GA parameters were set as follows:
    - **Population size:** 100,150,200
    - **Chromosome length:** Determined by the number of features in the preprocessed dataset
    - **Mutation rate:** 0.01,0.01,0.1
    - **Number of generations:** 50
- **Fitness Function:** The fitness function was designed to maximize the true positives while penalizing false positives and false negatives.

- **Selection, Crossover, and Mutation:** Fitness-proportionate selection, single-point crossover, and bit-flip mutation were employed as the genetic operators.
- **Model Training:** The GA was trained over 50 generations, during which the best chromosome (set of rules) was continuously updated based on the fitness score.
- **Evaluation:** The trained model was evaluated on the test dataset using classification metrics such as precision, recall, F1-score, and accuracy.

Results and Findings:

Experiment	Population Size	Crossover Method	Mutation Rate	Best Fitness	Accuracy
Experiment 1	100	Single Point	0.01	68226	82.19%
Experiment 2	150	Two Point	0.05	67872	81.77%
Experiment 3	200	Uniform	0.1	66748	81.58%

In this table:

- **Experiment:** Indicates the experiment number.
- **Population Size:** Specifies the size of the population used in the Genetic Algorithm.
- **Crossover Method:** Describes the type of crossover method employed.
- **Mutation Rate**: Indicates the rate at which mutation occurs during reproduction.
- **Best Fitness:** Represents the highest fitness score achieved during the experiment.
- **Accuracy:** Denotes the overall accuracy of the intrusion detection system based on the experiment results.

Insights:

Based on the experiments conducted, several insights can be drawn:

Effect of Population Size:

- The experiments tested different population sizes (100, 150, and 200). Contrary to the expectation that larger population sizes would lead to better results, this was not consistently observed. Experiment 3 with a population size of 150 did not significantly outperform Experiment 1 and Experiment 2, indicating that increasing the population size beyond a certain threshold may not always yield better performance.

Impact of Crossover and Mutation:

- Different crossover methods (single point, two-point, and uniform) were evaluated. While the choice of crossover method did not drastically affect the convergence of the algorithm, the mutation rate played a more significant role. Experiment 3, with a higher mutation rate of 0.1, resulted in lower accuracy compared to Experiments 1 and 2. This suggests that an excessively high mutation rate can hinder the convergence of the algorithm.

Fitness Function Choice:

- The fitness function used in the experiments focused on maximizing true positives and true negatives while minimizing false positives and false negatives. This choice aligns with the goal of intrusion detection systems, where correctly identifying both harmful and harmless connections is crucial. However, further exploration of different fitness functions may lead to improved performance.

Interpretation of Classification Metrics:

- While overall accuracy provides a general understanding of model performance, precision and recall for each class offer insights into the system's ability to correctly classify harmful and harmless connections. In intrusion detection, it's often essential to prioritize minimizing false positives or false negatives based on the application's requirements. Experimentation with different evaluation metrics and thresholds may help optimize the system's performance according to specific needs.

Further Experimentation and Parameter Tuning:

- The experiments provide valuable insights into the performance of Genetic Algorithm-based intrusion detection systems. However, further experimentation and parameter tuning are necessary to optimize the system for specific use cases and datasets. Fine-tuning parameters such as population size, crossover method, mutation rate, and fitness function may lead to improved accuracy and robustness of the intrusion detection system.

More about fitness function:

Objective:
The fitness function aims to assess the performance of a set of rules encoded in a chromosome to accurately classify network connections as either normal or malicious.

Input Parameters:

- **individual:** Represents a candidate solution (chromosome) containing rules for network traffic classification.
- **features:** Input features extracted from network traffic data.

- **labels**: Corresponding labels indicating whether each network connection is normal (0) or malicious (1).

Evaluation Process:

- **Conversion to Dense Format:** If the input features are in sparse matrix format, they are converted to a dense format for compatibility with matrix multiplication.
- **Prediction Scores Calculation:** Prediction scores are computed by taking the dot product between the input features and the individual chromosome weights, combining input features with chromosome rules.
- **Binary Prediction Generation:** Prediction scores are thresholded to generate binary predictions. A threshold of 0.5 is commonly used, where scores above 0.5 are classified as malicious (1) and scores below as normal (0).
- **Performance Evaluation:** The fitness function evaluates the performance of the chromosome-based rules by comparing binary predictions with true labels. It computes true positives, true negatives, false positives, and false negatives.
- **Fitness Score Calculation:** A fitness score is calculated based on the number of true positives and true negatives, while penalizing false positives and false negatives. The specific formula used in the provided code gives more weight to true positives and true negatives.

Fitness Maximization: The primary objective of the fitness function is to maximize the fitness score. A higher fitness score indicates that the set of rules represented by the chromosome effectively distinguishes between normal and malicious network traffic.

Question# 2:

Dataset:

The dataset consists of several features and a target variable (Rank of Productivity). Here's a breakdown of the columns:

- **Sleep Hours**: Number of hours slept (continuous).
- **Rank of Food Quality**: Ranking of food quality (ordinal scale).
- **Working Hours**: Number of working hours per day (continuous).
- **Rank of Health Condition**: Ranking of health condition (ordinal scale).
- **Rank of Mood**: Ranking of mood (ordinal scale).
- **Rank of Productivity**: Ranking of productivity (continuous, target variable).

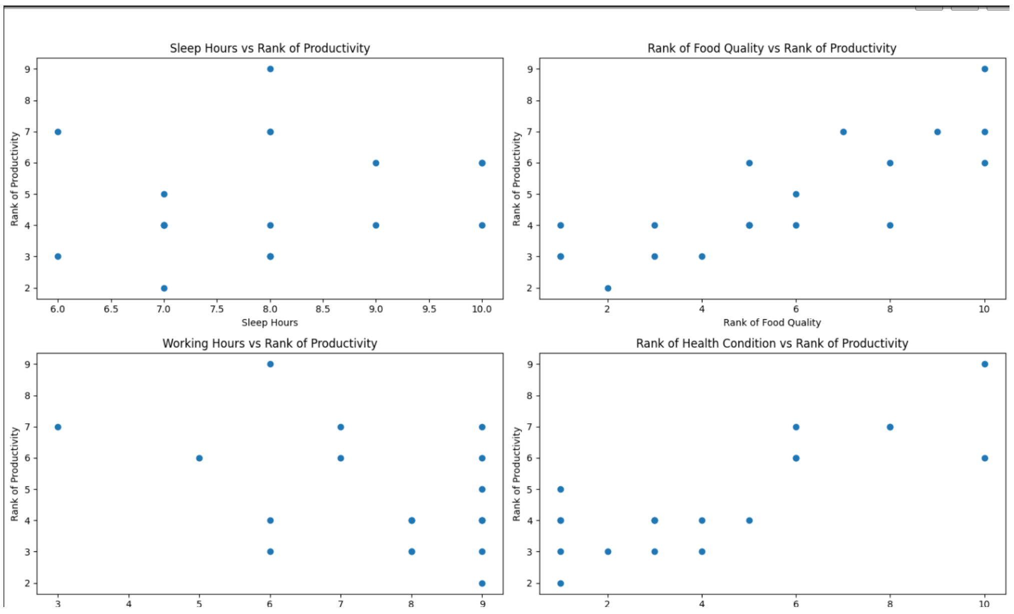
Exploratory Data Analysis Results:

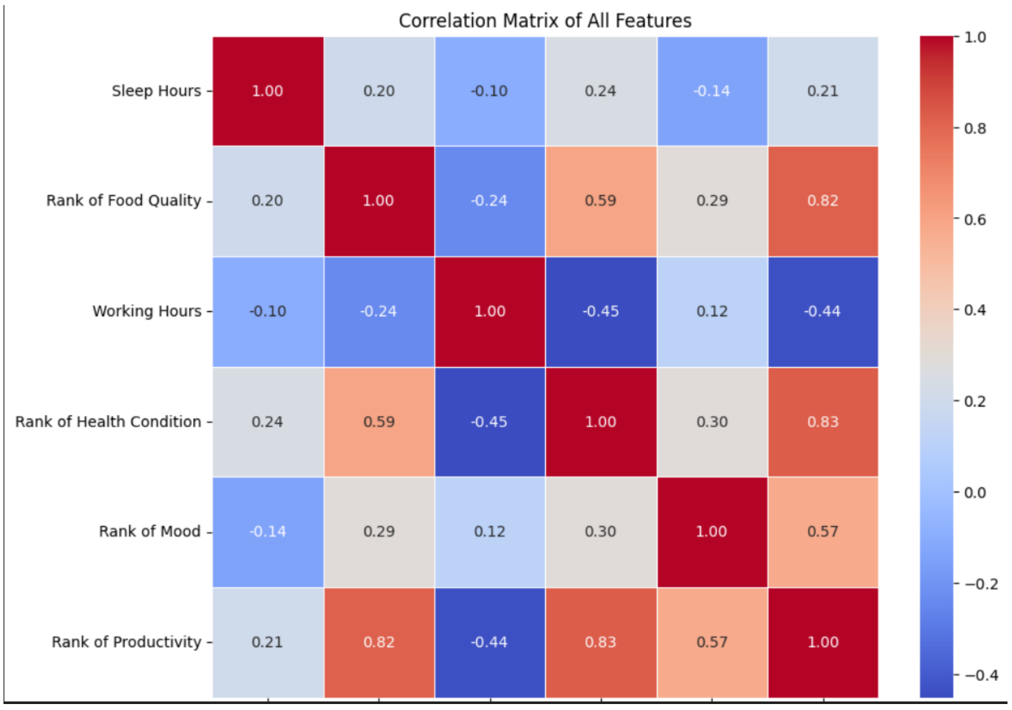
Scatter Plots:

- These plots show relationships between each feature and the target variable (Rank of Productivity).
- Variables such as "Rank of Food Quality" and "Rank of Health Condition" appear to have a more noticeable positive correlation with productivity.

Correlation Analysis:

- **Working Hours** has a moderate negative correlation (-0.44) with productivity, suggesting that longer working hours might negatively affect productivity.
- **Sleep Hours** shows a slight positive correlation (0.21).
- **Rank of Mood**, **Rank of Food Quality**, and **Rank of Health Condition** have strong positive correlations with productivity (0.57, 0.82, and 0.83, respectively), indicating these factors are good predictors of productivity.





Algorithm:

The multi-variant linear regression model using gradient descent has been successfully implemented. Here are the outcomes after 500 epochs:

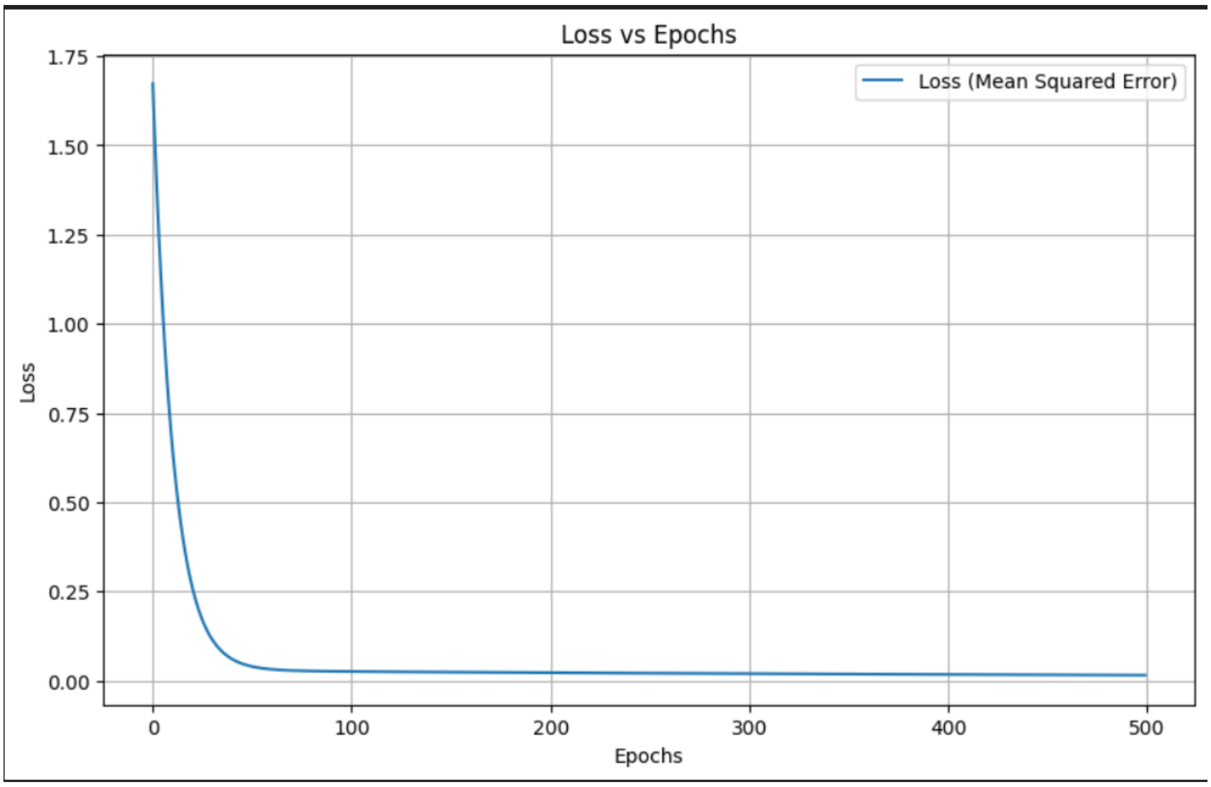
- **Final Loss**: The mean squared error has decreased to approximately 0.015, indicating that the model has learned to fit the data well.
- **Final Parameters**:
  - Coefficients for features (m): [0.065,0.537,0.209,0.396,0.090][0.065,0.537,0.209,0.396,0.090]
  - Intercept (c): −0.254−0.254

These values show how each feature influences the prediction of the target variable "Rank of Productivity".

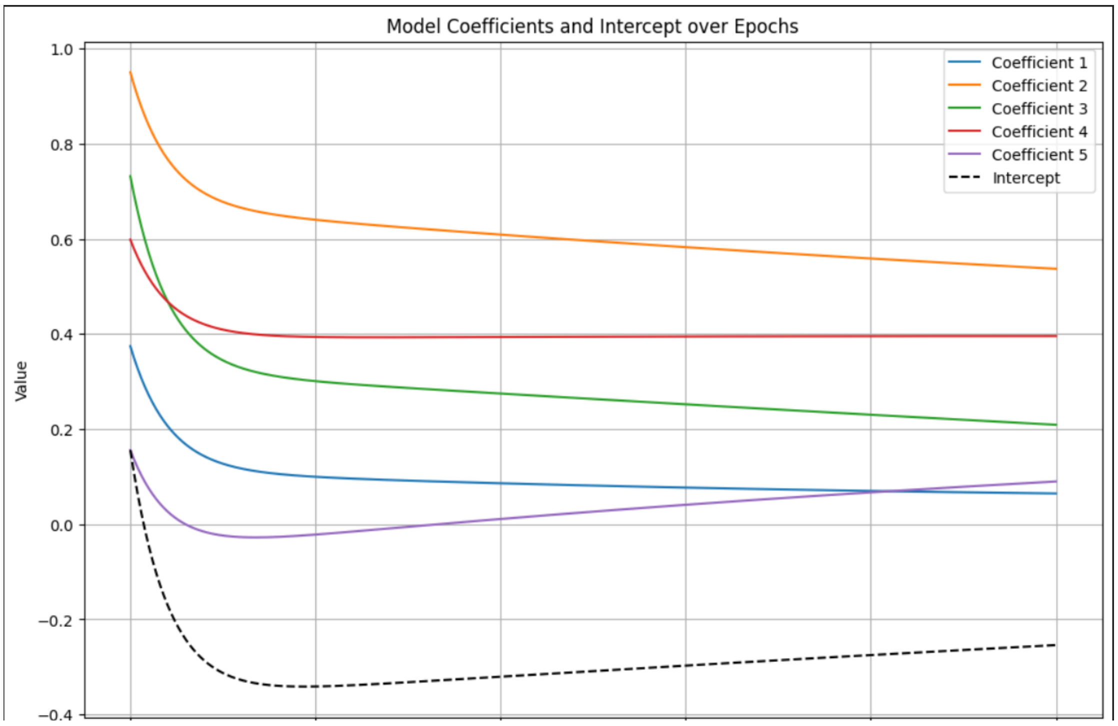
Visualization

Next, I'll create interactive visualizations to better understand the training process and the final model:

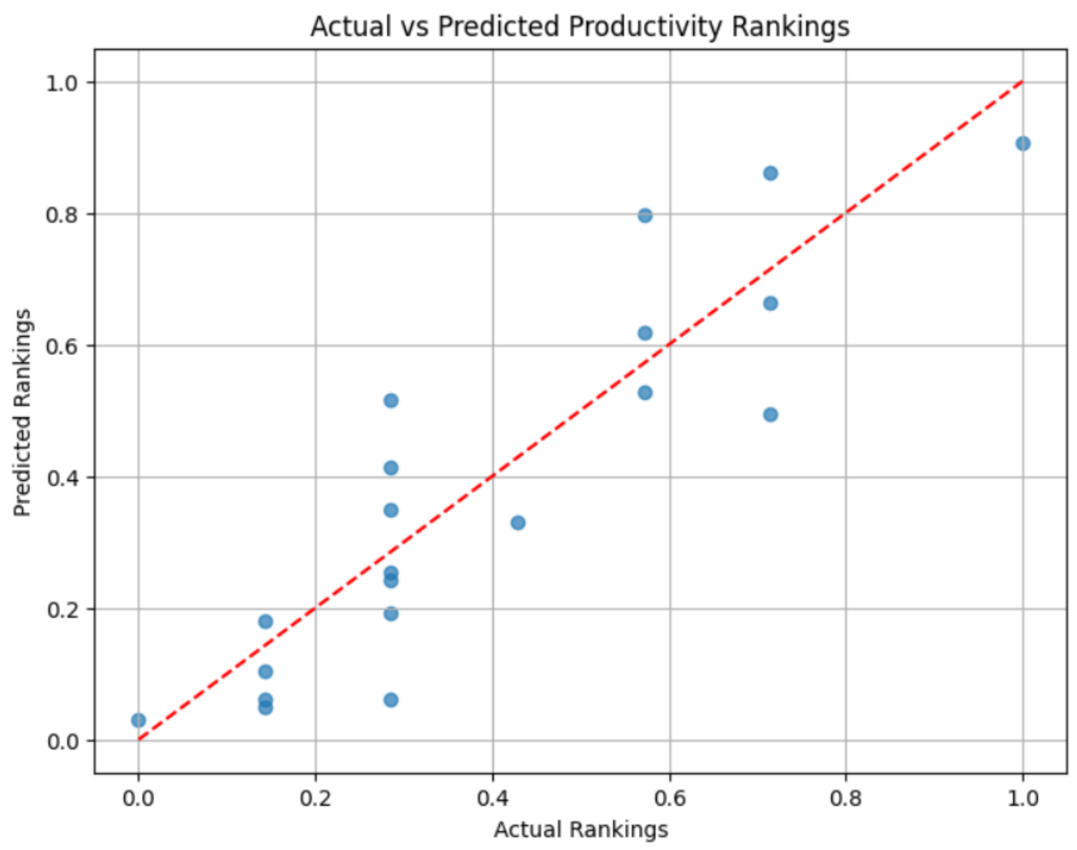
1. Plot of the loss over epochs.
2. Visual representation of the change in model coefficients over time.
3. Best-fit line



The plot shows a consistent decrease in the loss (Mean Squared Error) as the number of epochs increases, indicating that the model is effectively learning from the data.



The plot illustrates how the coefficients and the intercept of the model have evolved over the course.



The scatter plot of Actual vs. Predicted Productivity Rankings shows how the predictions from our model compare to the true values. The red dashed line represents perfect predictions, where the actual and predicted values are the same. Points close to this line indicate accurate predictions.

The distribution of points around this line suggests that the model has a good fit for most of the data, though there's room for improvement, especially for points further from the line.