Members: Ibrahim Akhtar (i211679) and Fasih Ur Rehman (i211705)

Question 1

All Changes I Made

I did not use the original source code but only took inspiration from it. I wanted to use the DEAP library and the toolboxes (which I got approved) to help with my GA. Additionally, I found the original source file too scrappy for the data of my nature as I already had to do a lot of pre-processing to standardize the data.

All further insights like the different type of data in training vs testing and irrelevant features is listed in depth in the report that follows.

Data Wrangling Efforts

Before feeding the data into the GA, I performed some preprocessing steps on both the training and testing datasets. First, I removed certain features that might be irrelevant or redundant for the classification task. Then, I standardized the dur (duration) and sload (service load) columns to ensure all features are on a similar scale. Additionally, I used one-hot encoding to transform categorical features like proto (protocol) into numerical representations.

To handle any inconsistencies between the datasets, I identified missing columns in the testing set that were present in the training set and filled them with zeros for consistency. I also ensured both datasets maintained the same column order for training and evaluation purposes. Finally, I split the datasets into features (X) and labels (y) for training and testing the model.

Designing the Genetic Algorithm

The core of this code is the GA, which I designed to select an optimal subset of features. Here's a breakdown of the key components:

Individual (chromosome): Each individual is a list of binary values (0 or 1) corresponding to each feature. A 1 indicates the feature is selected, and a 0 indicates it's excluded.
Fitness function: This function evaluates the quality of an individual (chromosome). The provided example uses the negative absolute value of the number of selected features. This might be a placeholder, and a more informative fitness function based on classification accuracy using a random forests classifier on the selected features could be designed.

Genetic Algorithm parameters:

NUM_GENERATIONS: Number of iterations the GA runs (set to 50 in the code).
NUM_FEATURES: Total number of features in the dataset.
Toolbox: This collection of functions powers the GA:
Selection: Selects individuals for reproduction based on their fitness (uses tournament selection).
Crossover: Combines genetic material from two parents to create offspring (uses two-point crossover).
Mutation: Introduces random changes to individual chromosomes (flips bits with a probability).

Experimenting with the GA

To test the GA's effectiveness, I ran three experiments with varying GA parameters:

Experiment 1: Population size: 50, Crossover probability: 0.6, Mutation probability: 0.1
Experiment 2: Population size: 100, Crossover probability: 0.8, Mutation probability: 0.2
Experiment 3: Population size: 200, Crossover probability: 1.0, Mutation probability: 0.5

For each experiment, the code reported the best individual (chromosome) and its corresponding fitness value. However, the current fitness function only considers the number of selected features, not the actual classification performance.

Insights

The first insight I deduced was that because I was using hardcoded hyperparameters, my GA was performing quite poorly and I could apply some technologies like Grid Search for better tuning. However, I was too lazy and could not apply it in time therefore settled with poor hardcoded performing GAs.

In my lazy approach, I noticed that with more population sizes and more mutation probabilities, the fitness value increases. The jump is highest between the first and second experiment but increases only 50% between the second and third meaning that a good rounded mid value for experiments will give the best fitness value.

Output Values:

Experiment: Experiment 1
Best individual: [0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
Best fitness: 122.0

Experiment: Experiment 2
Best individual: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1]
Best fitness: 139.0

Experiment: Experiment 3
Best individual: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Best fitness: 143.0

Question 2

Introduction

This report presents an analysis of a dataset containing observations on various attributes over a period, including sleep hours, rank of food quality, working hours, rank of health condition, rank of mood, and rank of productivity. Additionally, it outlines the implementation of a gradient descent algorithm to train a linear regression model to predict the rank of productivity based on the other attributes.

Dataset Description

The dataset comprises daily observations, with each row representing a single day. The attributes recorded for each day are as follows:

Sleep Hours: Number of hours slept.

Rank of Food Quality: Subjective ranking of the quality of food consumed.
Working Hours: Number of hours worked.
Rank of Health Condition: Subjective ranking of the individual's health condition.
Rank of Mood: Subjective ranking of the individual's mood.
Rank of Productivity: Subjective ranking of the individual's productivity on that day (target variable).

Exploratory Data Analysis (EDA)

Data Loading: The dataset is loaded from a CSV file.
Data Preprocessing: Features and target variable are extracted from the dataset.
Standardization: Features are standardized to have a mean of 0 and a standard deviation of 1.
Visualization: A scatter plot of the original data is generated to visualize the relationship between sleep hours and productivity.

Gradient Descent Model

Model Function: A linear regression model function is defined to predict productivity based on given features, weights, and bias.
Mean Squared Error Function: A function is defined to calculate the mean squared error between true and predicted values.
Gradient Descent Algorithm: Gradient descent algorithm is implemented to learn the optimal weights and bias for the model.
Training: The model is trained using gradient descent with a specified learning rate and a random number of epochs.
Loss Visualization: The loss (mean squared error) versus epoch is plotted to visualize the training progress.

Model Evaluation

Learned Parameters: The learned weights and bias are printed.
Predictions: Predictions are made on new data to estimate the rank of productivity.

Conclusion

In conclusion, this report provides insights into the dataset and presents a methodology for training a linear regression model using gradient descent. By following the outlined steps and recommendations, further improvements in model accuracy and predictive power can be achieved.