CS6271-EVOLUTIONARY COMPUTATION AND HUMANOID ROBOTICS

PROJECT REPORT

TOPIC: Genetic Programming for Income Classification

Submitted By

Ashik Kannampilly Janardhanan(25032615)

Saranya Shalu(25013912)

November 2025

1. Introduction

The objective of this project is to develop a classifier using Genetic Programming (GP) to predict whether an individual's income exceeds a given threshold, based on demographic and employment-related features. The task uses a modified version of the Adult dataset, provided through a private Kaggle competition for this module.

Genetic Programming was chosen because it can automatically evolve models in the form of mathematical expressions or tree-based programs, discovering both structure and parameters without requiring explicit specification. This makes it particularly suitable for capturing complex, nonlinear relationships in the data.

The project involves data preprocessing, feature encoding, model training, and evaluation, with the performance measured in terms of training and test accuracy. The results demonstrate how GP can be applied to classification tasks and the predictive power of the evolved models.

2. Data Preprocessing

The dataset provided by Kaggle was already split into training and test sets. The training set included the target variable (income), while the test set did not. Before applying Genetic Programming, the data required cleaning and transformation to ensure consistency and suitability for modeling.

Dataset Features

The dataset contains 15 features, including both numerical and categorical variables, as summarized below: Preprocessing Steps

To prepare the data for the GP model:

Feature	Type	Description
age	Numerical	Age of the individual
workclass	Categorical	Type of employment
fnlwgt	Numerical	Sampling weight
education	Categorical	Highest level of education

education-num	Numerical	Years of education
marital status	Categorical	Marital Status
occupation	Categorical	Occupation type
relationship	Categorical	Relationship status
race	Categorical	Race of the individual
sex	Categorical	Gender
capital-gain	Numerical	Capital gains
capital-loss	Numerical	Capital losses
hours-per-week	Numerical	Hours worked per week
native-country	Categorical	Country of origin
income	Binary	Target variable: 1 if > 50k, 0 otherwise

1. Handling missing values: Missing entries represented by '?' were replaced with the mode of their respective columns.

2. Encoding categorical variables: All categorical features were converted to numerical form using Label Encoding to make them compatible with GP.

3. Feature scaling: Continuous features were normalized using StandardScaler, which helps GP handle features with widely varying ranges.

4. Train-validation split: The training data was further split into training (80%) and validation (20%) sets to evaluate the model's performance during development.

These preprocessing steps ensured that the dataset was clean, standardized, and ready for training the Genetic Programming classifier.

3. Model Setup

3.1 Function Set

In Genetic Programming (GP), the function set defines the set of operations that the algorithm can use to construct candidate solutions. For this project, the function set included arithmetic, unary, and constant functions, carefully designed to ensure numerical stability.

The following functions were used:

Type	Functions	Description
Arithmetic	Add, Sub, Mul, SafeDiv	Basic arithmetic operations. safeDiv avoids division by zero errors.
Unary	Neg, safeLog, sin, cos	Unary operations including negation, logarithm, and trigonometric functions. safeLog ensures the log of non-positive numbers does not cause errors.
Constants	Random ephemeral constants	Random constants uniformly sampled between -1 and 1, used as fixed numerical values in evolved expressions.

Helper Functions for Stability

Two helper functions were defined to ensure that GP expressions could be evaluated safely:

- safeDiv(left, right) returns left / right, but returns 1 if right is zero.
- safeLog(x) returns log(abs(x)+1), avoiding undefined logarithms for zero or negative values.

These functions allow GP to handle a wide range of numeric inputs without runtime errors.

Implementation Overview

The functions and constants were added to the GP primitive set, which provided the building blocks for evolving mathematical expressions. Each feature of the dataset served as an input variable, and the GP algorithm could combine them using the function set and constants to form candidate models

3.2. GP Configuration

The following parameters were used to configure the Genetic Programming algorithm:

Parameter	Value	Purpose
Population Size	800	Number of candidate solutions in each generation. Larger populations provide more diversity.
Crossover Probability	0.9	The probability that two parent trees will exchange subtrees to create offspring. Encourages exploration of new solutions.
Mutation Probability	0.4	The probability that a random modification is applied to a tree. Introduces variability to avoid local optima.
Number of Generations	100	Number of evolutionary iterations. Determines how long the algorithm evolves solutions.
Hall of Fame Size	15	Number of best individuals preserved across generations. Ensures top solutions are not lost
Random Seed	42	Ensures reproducibility of results by initializing the random number generator consistently.
Initial Tree Depth	1-7	Depth range for randomly generated trees at the start. Controls initial solution complexity.
Mutation Tree Depth	0-3	Maximum depth of subtrees introduced during mutation. Prevents trees from growing too large.
Max Tree Limit	35	Maximum allowed depth for any tree in the population. Helps control computational cost and overfitting.

These settings guided the evolution of solutions across generations, balancing exploration and exploitation, while ensuring the results were reproducible.

3.3 Fitness Function

The fitness function evaluates how well each individual (tree or program) performs the classification task. In this project, accuracy was used as the fitness metric, measuring the proportion of correct predictions on the training data:

Accuracy = Number of correct predictions / Total number of predictions

Each GP tree was compiled into a callable function. For each data instance, the GP function produced a numerical output. The predicted class was determined as follows:

- If the output > 0, the prediction was labeled as class 1 (income > $50K).
- Otherwise, the prediction was labeled as class 0 (income ≤ $50K).

This design encourages the GP algorithm to evolve expressions that effectively separate the two income classes by maximizing accuracy.

A high-level view of the evaluation process is illustrated below:

1. Compile the GP tree into a callable function.
2. Apply the function to each row of the training data to generate predictions.
3. Convert numerical outputs to binary class labels based on the threshold of 0.
4. Compute the accuracy of these predictions against the true labels.
5. Return the accuracy as the fitness score.

This approach allowed the GP algorithm to iteratively improve individuals over generations, selecting those that most accurately classified income levels.

4. Results

The Genetic Programming (GP) model was evaluated on the training and validation sets, and predictions were submitted to the Kaggle leaderboard for test set evaluation. The observed accuracies are summarized below:

Dataset	Accuracy
Train Data	0.8427
Validation Data	0.8349
Kaggle Submission (Test Data)	0.7747

The GP model achieved a training accuracy of 0.8427 and a validation accuracy of approximately 0.8349, demonstrating that the evolved expressions effectively captured meaningful patterns in the dataset. The final evolved model was submitted to Kaggle, where its performance on the unseen test data achieved an accuracy of 0.7747, as reported by the competition leaderboard.

These results indicate that the GP approach was able to produce a reliable classifier for predicting income, with performance in line with expectations for this task.

5. Model Explainability

A key advantage of Genetic Programming (GP) is its ability to evolve interpretable symbolic expressions, allowing insight into how predictions are made.

In our best model, the evolved function combined arithmetic operations with logarithmic and trigonometric transformations of several features. The model produced a numerical output for each data instance, which was then thresholded to determine the predicted class:

- Output $> 0 \rightarrow$ high income (1)
- Output $\leq 0 \rightarrow$ low income (0)

Although the final expressions can be complex, their tree-based structure allows us to trace the contribution of individual features to the model's decisions. This interpretability is a major benefit of GP, as it provides insight into the decision-making process — something that is often difficult to achieve with black-box models such as deep neural networks.

6. Conclusion

This project demonstrated the effectiveness of Genetic Programming (GP) for a real-world binary classification task. By evolving symbolic expressions over successive generations, GP was able to achieve competitive accuracy without requiring a predefined model architecture. The evolved models are interpretable, providing insight into how individual features contribute to the prediction of income.

Potential avenues for future improvement include:

- Using alternative fitness metrics: Employing the F1 score instead of accuracy could improve performance on imbalanced datasets.
- Feature selection: Reducing the number of input features could simplify evolved trees and improve interpretability.
- Function set and selection strategies: Experimenting with different functions or incorporating elitist selection may accelerate convergence and improve final model performance.

Overall, the project illustrates that GP can produce both accurate and interpretable classifiers, making it a valuable approach for tasks where understanding model decisions is important.

7. References

- Eiben, A. E., Smith, J. E. (2015). *Introduction to Evolutionary Computing* (2nd Edition). Springer.
- Ryan, C., O'Neill, M., Collins, J. (2018). *Handbook of Grammatical Evolution*.