

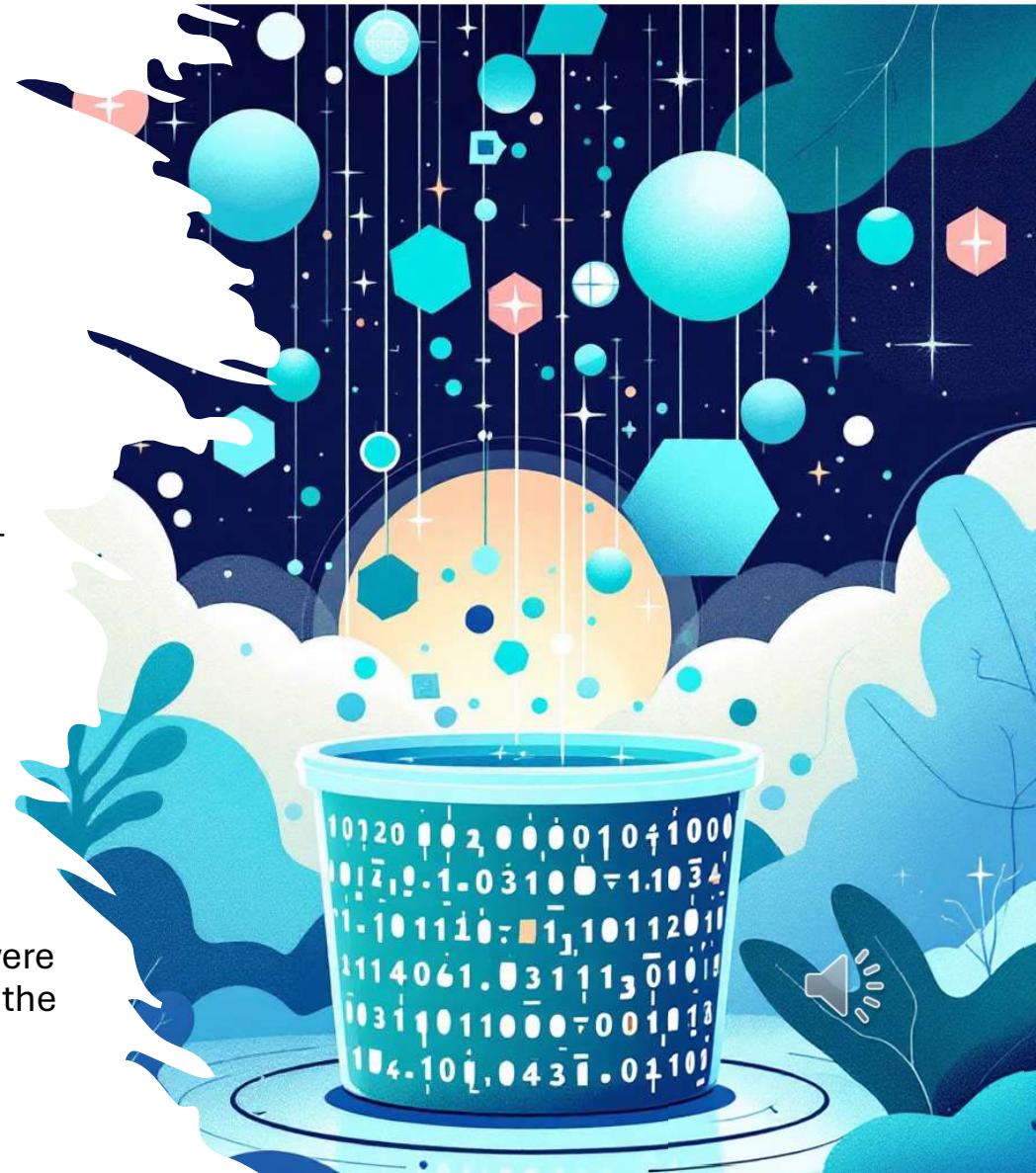
Introduction: Training a Classifier with Genetic Programming

- Choosing Genetic Programming (GP) over Grammatical Evolution (GE) was a personal choice, due to a stronger understanding of GP and its suitability for creating easily interpretable mathematical equations for binary classification.
- We will now look at the various aspects such as data pre-processing, primitive selection, fitness function design, parameter setups, results, and conclusions.



Data Pre-Processing

- 1. Column Removal Strategy:** From the 14 non-income columns, three were removed:
 - **fnlwgt:** Irrelevant as it represents final sampling weight.
 - **education:** Redundant, given the presence of 'education-num'.
 - **native-country:** Over 90% of entries were 'USA'.
- 2. Numerical Feature Scaling:** Numerical columns were scaled using **StandardScaler** to ensure uniform contribution to the model.
- 3. Categorical Feature Encoding:** Categorical columns were encoded by dropping the first column, which increased the accuracy as compared to standard one-hot encoding.



GP Primitives

Arithmetic Operations: Addition, Subtraction, Multiplication, Division, Negative, Absolute.

Mathematical Functions: Tan, Sin, Cos.

Conditional Logic: If_Else, Greater/Less Than.

Phased Out Primitives: Log, Exp, Max/Min, and And/Or

Additional Terminals: Five key terminals were introduced: 1, 0, -1, 0.5, and 2.



Fitness Function: Balancing Accuracy and Tree Complexity

- Initially, a single-objective function sampling 2000 records was used which led to underfitting. Removing sampling caused overfitting, and penalties for large tree sizes further reduced accuracy.

"The development of a multi-objective fitness function struck the perfect balance between accuracy and tree size"

- The first objective being the accuracy score and the second, the tree size. This approach significantly improved results by penalising larger trees, effectively preventing overfitting.



Parameter Setup



POPULATION_SIZE: 1000

Combined with a half-and-half initial population generation, this size produced the best results.



P_CROSSOVER: 0.65

This value, along with the mutation rate, effectively balanced exploration and exploitation in the simulation



P_MUTATION: 0.4

Higher mutation rates prevented the model from getting stuck at local optima, which increased the accuracy of the individual



MAX_GENERATIONS: 150

Provided sufficient time for convergence, especially with the higher mutation rate.



Tree Size Parameters



MIN_TREE_HEIGHT: 2

This ensured we won't get too simple solutions.



MAX_TREE_HEIGHT: 8

8 kind of hit the sweet spot between other possible values



LIMIT_TREE_HEIGHT: 10

This was one of the main factors in preventing overfitting.



Tournament Selection Size

- **Value: 20**
A higher selection size ensured increased selection pressure, leading to continuous improvement in individuals and break the 82% barrier
- Other values: 3, 5, 7, 10, 15.





Best Individual Expression

```
gt(ARG2, sin(cos(mul(mul(logical_if(logical_if(logical_if(ARG27, add(ARG41, ARG0),  
ARG29), add(tanh(2.0), ARG0), gt(ARG30, cos(ARG3))), div(sin(ARG3), lt(cos(ARG3),  
logical_if(ARG27, ARG34, ARG9))), ARG21), ARG14), mul(mul(logical_if(logical_if(ARG14,  
add(tanh(2.0), ARG0), gt(ARG30, cos(ARG3))), div(sin(ARG3), lt(cos(ARG3), logical_if(2.0,  
ARG34, ARG9))), abs(ARG5)), ARG14), add(logical_if(gt(ARG30, cos(ARG3)), div(-  
3.8860942081653924, ARG9), add(cos(ARG38), ARG1)), lt(add(ARG9, sin(add(ARG9,  
ARG3))), sin(sin(ARG0))))))))))
```



Result



- Individual with maximum fitness of 85.64%, with length as 83 and height as 10. This individual after compiling and running on the test data produced an accuracy of 78.11%.
- In the above graphs we can see the max fitness eventually converged at around 85% and that not major difference in both the runs.

