# Few-Shot Code Forensics: Detecting AI-Generated Code using UniXcoder Prototypical Networks

Arif Akbarul Huda (25/571619/SPA/01156)

December 5, 2025

# The Datasets

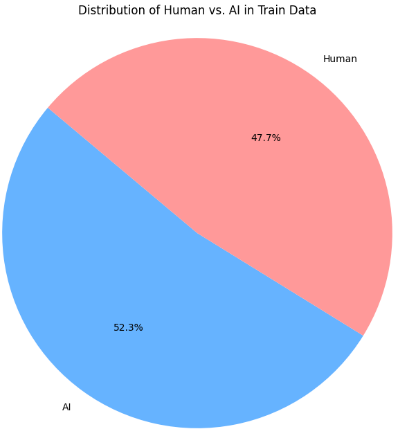
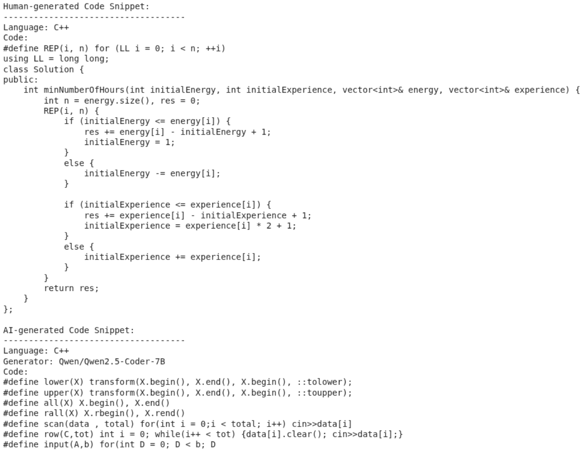


Figure: The distribution human vs ai accross training, validation, test

# The Datasets



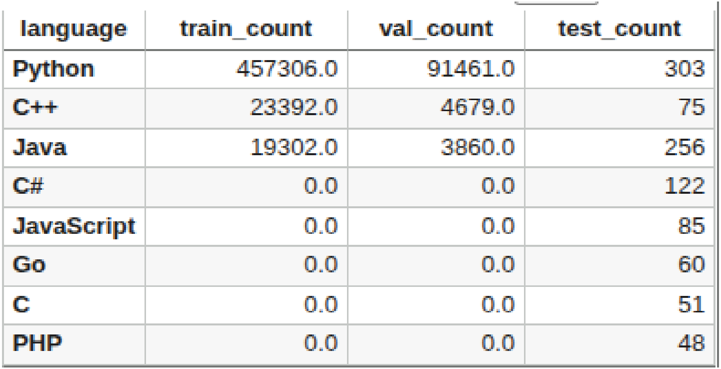
```
Human-generated Code Snippet:
------------------------------------
Language: C++
Code:
#define REP(i, n) for (LL i = 0; i < n; ++i)
using LL = long long;
class Solution {
public:
    int minNumberOfHours(int initialEnergy, int initialExperience, vector<int>& energy, vector<int>& experience) {
        int n = energy.size(), res = 0;
        REP(i, n) {
            if (initialEnergy <= energy[i]) {
                res += energy[i] - initialEnergy + 1;
                initialEnergy = 1;
            }
            else {
                initialEnergy -= energy[i];
            }

            if (initialExperience <= experience[i]) {
                res += experience[i] - initialExperience + 1;
                initialExperience = experience[i] * 2 + 1;
            }
            else {
                initialExperience += experience[i];
            }
        }
        return res;
    }
};

AI-generated Code Snippet:
------------------------------------
Language: C++
Generator: Qwen/Qwen2.5-Coder-7B
Code:
#define lower(X) transform(X.begin(), X.end(), X.begin(), ::tolower);
#define upper(X) transform(X.begin(), X.end(), X.begin(), ::toupper);
#define all(X) X.begin(), X.end()
#define rall(X) X.rbegin(), X.rend()
#define scan(data , total) for(int i = 0;i < total; i++) cin>>data[i]
#define row(C,tot) int i = 0; while(i++ < tot) {data[i].clear(); cin>>data[i];}
#define input(A,b) for(int D = 0; D < b; D
```

Figure: The example human vs ai generated code

# The Datasets



| language | train_count | val_count | test_count |
|---|---|---|---|
| Python | 457306.0 | 91461.0 | 303 |
| C++ | 23392.0 | 4679.0 | 75 |
| Java | 19302.0 | 3860.0 | 256 |
| C# | 0.0 | 0.0 | 122 |
| JavaScript | 0.0 | 0.0 | 85 |
| Go | 0.0 | 0.0 | 60 |
| C | 0.0 | 0.0 | 51 |
| PHP | 0.0 | 0.0 | 48 |

Figure: The datasets summarize across training, validation, test

# ARCHITECTURE AND TRAINING STRATEGY

## Model Architecture

- microsoft/unixcoder-base, a pre-trained cross-lingual model suitable for code representation.
- Prototypical Network (Metric-based Meta-Learning).
- Projects code samples into a metric space and classifies them based on Euclidean distance to class prototypes (Human vs. AI)

# ARCHITECTURE AND TRAINING STRATEGY

## Training Strategy - Episodic Meta-Training

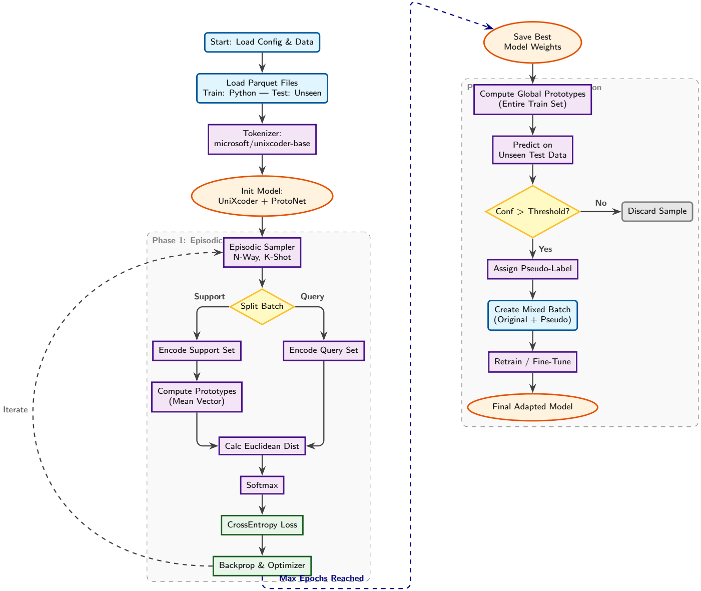
- Training simulates "N-Way, K-Shot" tasks (N=2 classes, K=5 support samples, Q= 30)
- AdamW optimizer with learning rate decay
- HIDDEN DIM = 768 , Q QUERY = 30

# ARCHITECTURE AND TRAINING STRATEGY

## Training Strategy - Test Time Adaptation

- Pseudo-Labeling
- Transductive Fine-tuning

# ML Pipeline: Meta-Training & Adaptation



# Validation Report

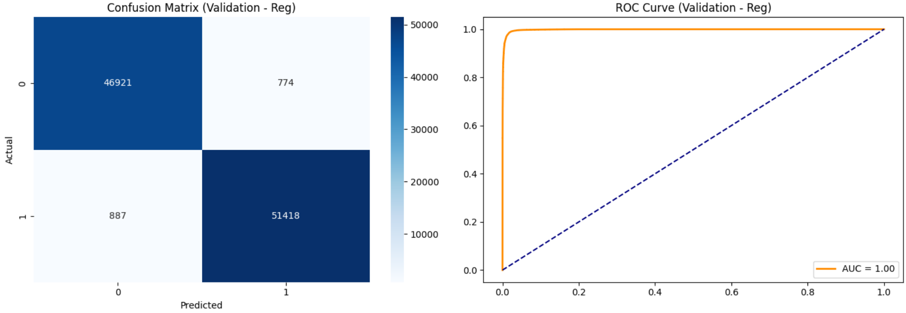


Figure: The metric on validation data

# Test Report

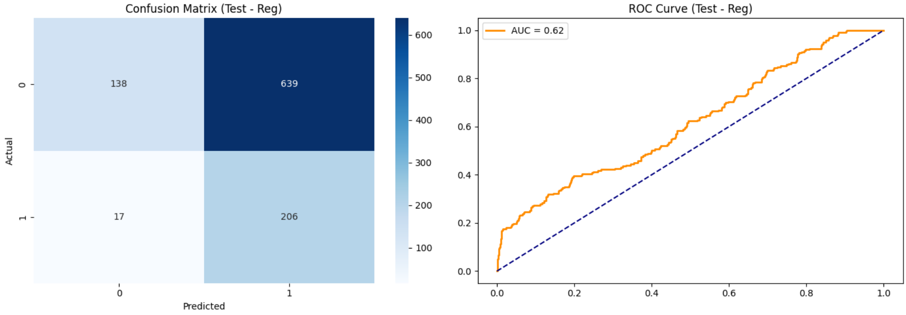


Figure: The metric on test data

# Test Report

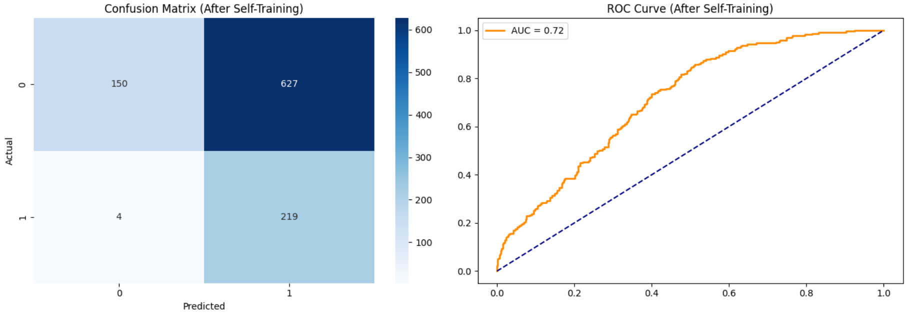


Figure: The metric on test data after self adaption

# Test Report

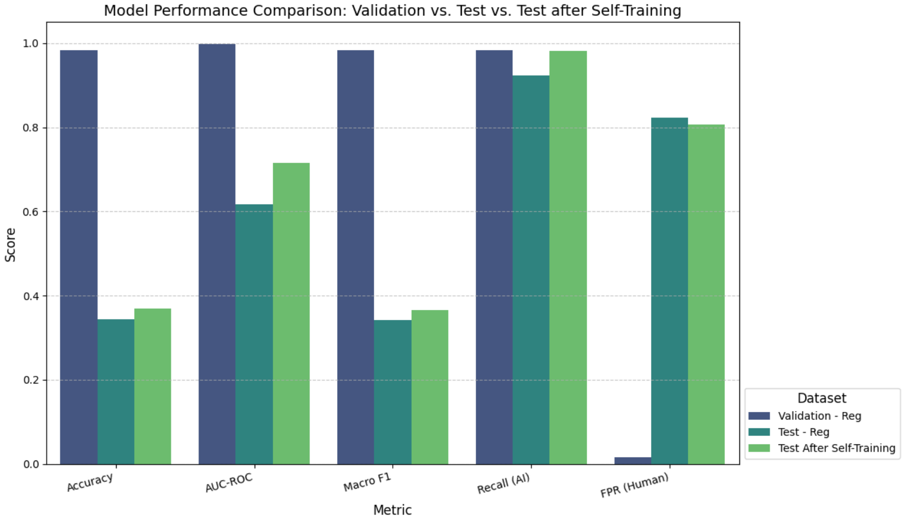


Figure: The model performance comparison