

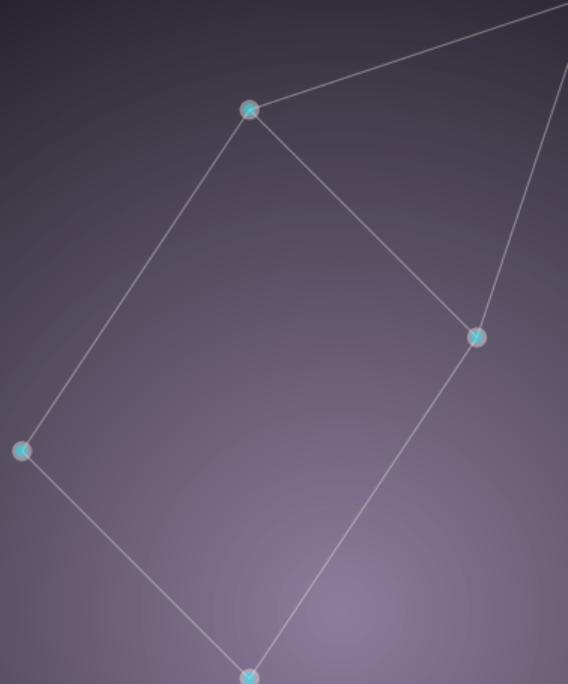
GenAI Code Detection & Attribution

Speech & Language Processing Exam

Giovanni Giuseppe Iacuzzo

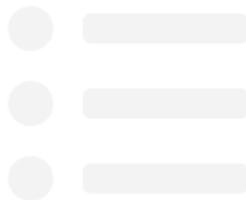
Università degli Studi di Enna “Kore”

AI & Cybersecurity Student ·  [giovannilacuzzo](#)



Roadmap

- 1 Introduzione e Obiettivi
- 2 Architecture dei modelli
- 3 Analisi dei Dati
- 4 Risultati del Task
- 5 Conclusioni



Il Nuovo Paradigma: GenAI Code



The Rise of LLMs

L'evoluzione di modelli come **GPT-4, Llama-3 e StarCoder** ha reso il codice sintetico indistinguibile da quello umano. Questo fenomeno introduce rischi critici per la **sicurezza software**, il **copyright** e l'integrità dei sistemi.



Human Code

(Creativity, Bugs)



GenAI Code

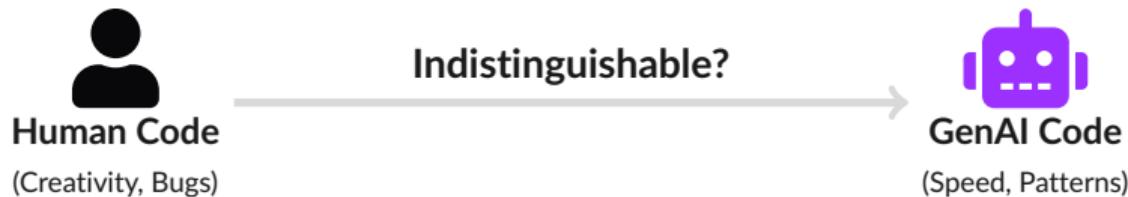
(Speed, Patterns)

Il Nuovo Paradigma: GenAI Code



The Rise of LLMs

L'evoluzione di modelli come **GPT-4, Llama-3 e StarCoder** ha reso il codice sintetico indistinguibile da quello umano. Questo fenomeno introduce rischi critici per la **sicurezza software**, il **copyright** e l'integrità dei sistemi.

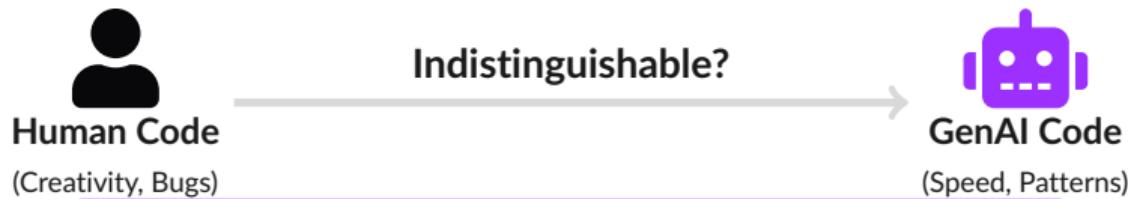


Il Nuovo Paradigma: GenAI Code



The Rise of LLMs

L'evoluzione di modelli come **GPT-4, Llama-3 e StarCoder** ha reso il codice sintetico indistinguibile da quello umano. Questo fenomeno introduce rischi critici per la **sicurezza software**, il **copyright** e l'integrità dei sistemi.



SemEval-2026 Task 13

Detect · Attribute · Analyze

Obiettivi della Competizione

La Task 13 risponde a tre esigenze fondamentali dell'industria moderna:



Sicurezza

Prevenire vulnerabilità
introdotte da codice generato
(Hallucinations) in sistemi
critici.

Obiettivi della Competizione

La Task 13 risponde a tre esigenze fondamentali dell'industria moderna:



Sicurezza

Prevenire vulnerabilità introdotte da codice generato (Hallucinations) in sistemi critici.



Attribuzione

Identificare il modello specifico (es. Llama vs OpenAI) per licensing e proprietà intellettuale.

Obiettivi della Competizione

La Task 13 risponde a tre esigenze fondamentali dell'industria moderna:



Sicurezza

Prevenire vulnerabilità introdotte da codice generato (Hallucinations) in sistemi critici.



Attribuzione

Identificare il modello specifico (es. Llama vs OpenAI) per licensing e proprietà intellettuale.



Analisi Irida

Gestire scenari complessi: codice misto Uomo-AI, Refactoring e Codice Avversario.

Struttura del Progetto: I 3 Moduli

Una pipeline unificata per affrontare i tre livelli di complessità del task:

Subtask A · Binary Detection

Human vs Machine.

Classificazione binaria su 500K sample. Include scenari *Unseen Languages* (es. Go, PHP) e *Unseen Domains*.

Struttura del Progetto: I 3 Moduli

Una pipeline unificata per affrontare i tre livelli di complessità del task:

Subtask A · Binary Detection

Human vs Machine.

Classificazione binaria su 500K sample. Include scenari *Unseen Languages* (es. Go, PHP) e *Unseen Domains*.

Subtask B · Attribution

Multi-Class (11 Families).

Identificazione fine-grained dell'autore (Human, GPT, Llama, Mistral, ecc.). Sfida: Forti sbilanciamenti di classe.

Struttura del Progetto: I 3 Moduli

Una pipeline unificata per affrontare i tre livelli di complessità del task:

Subtask A · Binary Detection

Human vs Machine.

Classificazione binaria su 500K sample. Include scenari *Unseen Languages* (es. Go, PHP) e *Unseen Domains*.

Subtask B · Attribution

Multi-Class (11 Families).

Identificazione fine-grained dell'autore (Human, GPT, Llama, Mistral, ecc.). Sfida: Forti sbilanciamenti di classe.

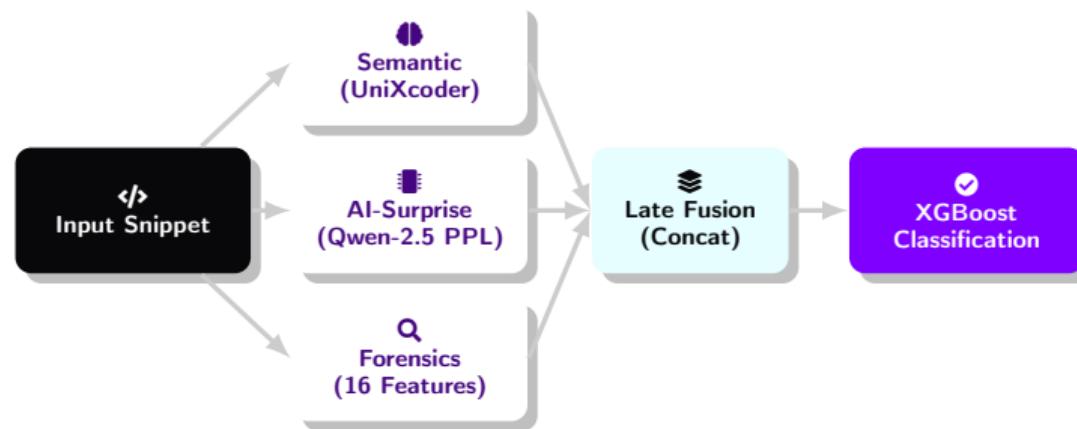
Subtask C · Hybrid Analysis

Mixed & Adversarial (4 Classi).

Classificazione complessa: Human, Machine, Hybrid (Mixed) e Adversarial (Codice offuscato). 900K sample.

Subtask A: Hybrid Forensics Pipeline

L'architettura proposta combina la potenza degli LLM con tecniche di analisi forense classica tramite una strategia di **Late Fusion**.



Rationale: Gli LLM lasciano tracce statistiche ("Surprise") e stilistiche (Forensics) diverse, che i modelli semantici puri (UniXcoder) potrebbero ignorare.

1. AI-Surprise & Stylometric Forensics



Perplexity Engine

Utilizziamo **Qwen2.5-Coder-1.5B** per calcolare quanto il modello è "sorpreso" dal codice.

- ▶ **Ipotesi:** Il codice generato da AI ha una Perplexity (PPL) inferiore rispetto a quello umano.
- ▶ **Implementazione:**
 - ▶ Sliding Window Processing.
 - ▶ OOM Auto-Recovery (Dynamic Batching).
 - ▶ Feature: `cross_entropy_loss`.



Forensic Features (16 dim)

Estrazione di pattern stilistici tramite Regex:

- ▶ **Entropy & TTR:** Complessità lessicale.
- ▶ **Naming Inconsistency:** Mix di `snake_case` e `camelCase` (tipico umano).
- ▶ **Layout:** Spazi bianchi e indentazione.
- ▶ **Identifiers:** Entropia nomi variabili.

2. Semantic Backbone & Late Fusion



Semantic Encoding (Microsoft UniXcoder)

Per catturare la logica del codice, utilizziamo **UniXcoder-base**.

- ▶ **Processo:** Input (512 token) → Forward Pass → Mean Pooling.
- ▶ **Output:** Vettore denso a **768 dimensioni**.



Late Fusion & XGBoost

Le feature vengono concatenate in un unico vettore [784 dim]:

$$V_{final} = \text{Concat}(\mathbf{E}_{sem}, \mathbf{F}_{style}, \mathbf{P}_{ppl})$$



Perché XGBoost?

- ▶ Gestisce bene dati eterogenei (Dense + Sparse).
- ▶ **Feature Importance:** Conferma che *Perplexity* e *Entropy* sono discriminatori chiave.
- ▶ Training con **Early Stopping**.

Subtask B: The Attribution Challenge

A differenza del task binario, qui l'obiettivo è identificare l'esatta famiglia del modello (Fine-Grained Classification) in uno scenario fortemente sbilanciato.

11 Target Classes

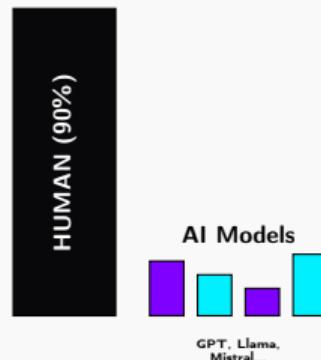
Il modello deve distinguere tra:

- ▶ **1 Classe Umana** (Dominante).
- ▶ **10 Famiglie AI**: GPT-4, Llama-3, StarCoder, Mistral, Qwen, DeepSeek, ecc.

Key Challenges:

- ▶ **Extreme Imbalance**: Human »> AI (442k vs 2k-10k).
- ▶ **OOD Detection**: Gestire modelli "Unseen" nel test set.

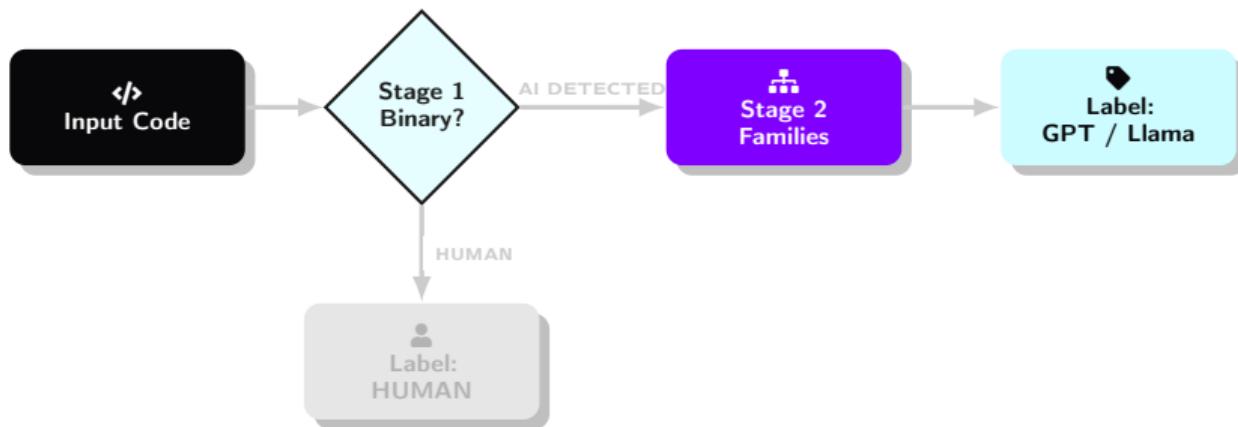
Data Distribution



Approccio diretto → Bias verso "Human".
Soluzione: **Cascade Inference**.

Strategy: Cascade Inference Pipeline

Per mitigare i falsi positivi sulla classe maggioritaria (Human), abbiamo diviso il problema in due stadi logici sequenziali.



🔒 Stage 1 (Binary Filter)
Protegge dai falsi positivi umani.

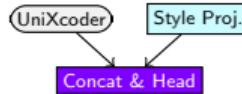
🔍 Stage 2 (Fine-Grained)
Specializzato solo sulle differenze tra LLM.

Architecture: Style-Injected Neural Network

⚙️ Custom Model Architecture

Estensione di **UniXcoder-base** con iniezione di feature stilistiche nello spazio latente.

- ▶ **Attention Pooling:** Somma pesata dei token (meglio del [CLS] standard).
- ▶ **Style Projector:** 8 feature manuali (es. density, snake_case) proiettate in uno spazio denso.
- ▶ **Feature Fusion:** Concatenazione (Semantic + Projected Style) pre-classifica.



☒ Advanced Training Strategy

- ▶ **U Supervised Contrastive Loss (SupCon):** Avvicina nello spazio vettoriale gli snippet della stessa famiglia (es. Llama-2 e Llama-3) e allontana famiglie diverse.
- ▶ **⌚ Focal Loss:** Penalizza gli errori sulle classi difficili/rare, mitigando l'impatto dello sbilanciamento delle classi.

Subtask C: Hybrid & Adversarial Code

Il livello più avanzato della competizione: rilevare non solo chi ha scritto il codice, ma **come** è stato modificato o offuscato.

☒ 4 Target Classes

Il modello deve classificare lo snippet in:

- ▶ 0: Human-Written
- ▶ 1: Machine-Generated
- ▶ 2: Hybrid (Mixed): Codice umano completato o refactorizzato da AI.
- ▶ 3: Adversarial: Codice AI generato con prompt avversari per evadere la detection.

🔒 Robustness Goal

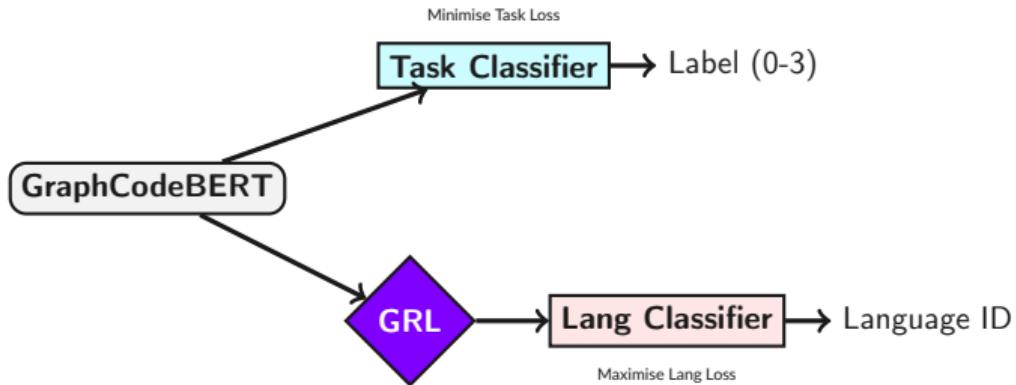
Problem: I modelli standard imparano correlazioni spurie (es. "C++ è sempre umano").

Solution: Dobbiamo costringere il modello a ignorare il linguaggio e guardare la struttura.

Key Tech: Domain Adversarial Training (DANN)

Architecture: Language-Invariant Learning

Per generalizzare su linguaggi non visti (es. Go, PHP), utilizziamo un **Gradient Reversal Layer (GRL)**.



Logic: Il GRL inverte il gradiente durante la backpropagation. Il modello impara feature che sono *utili per la classificazione* ma *inutili per indovinare il linguaggio*, diventando così "Language Agnostic".

Training Strategy & Ensemble Submission

Multi-Objective Loss Function

$$\mathcal{L}_{total} = \mathcal{L}_{Task} + \lambda_1 \mathcal{L}_{SupCon} + \lambda_2 \mathcal{L}_{DANN}$$

- ▶ **Focal Loss (\mathcal{L}_{Task})**: Gestisce lo sbilanciamento delle 4 classi.
- ▶ **SupCon (\mathcal{L}_{SupCon})**: Clusterizza feature simili (es. Hybrid con Hybrid).
- ▶ **Adversarial (\mathcal{L}_{DANN})**: Rimuove il bias del linguaggio.

5-Fold Ensemble Inference

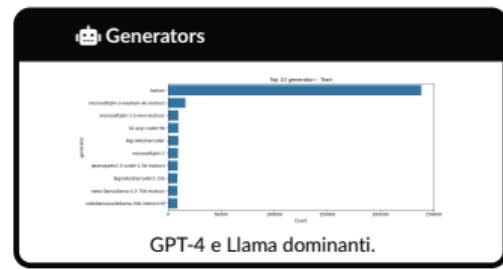
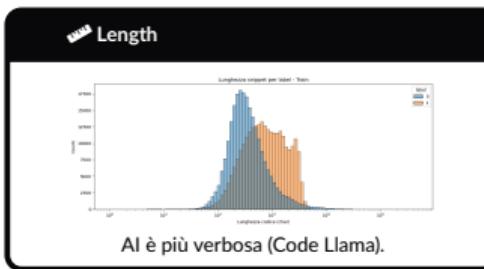
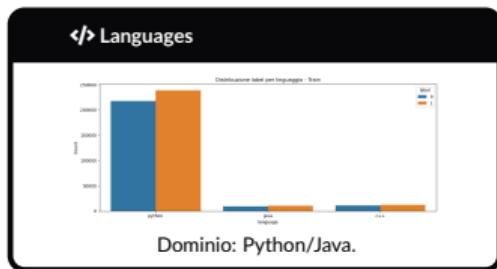


Per la sottomissione finale, non usiamo un singolo modello ma un **Ensemble di 5 modelli addestrati su fold diversi (Cross-Validation)**.

- ▶ **Aggregazione**: Soft Voting (Media delle probabilità).
- ▶ **Beneficio**: Riduce la varianza e aumenta la robustezza su dati OOD.

Task A [Train]: Learning Source Bias

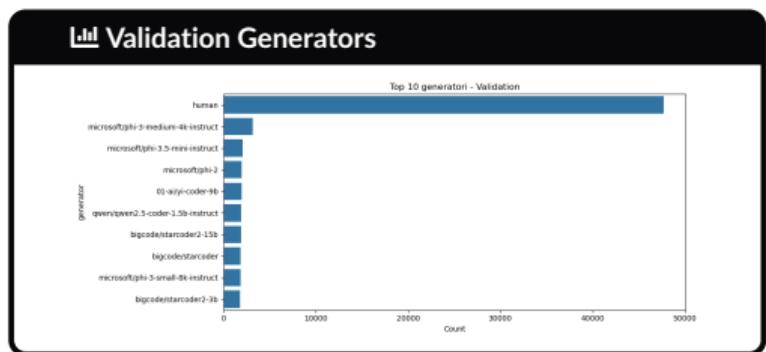
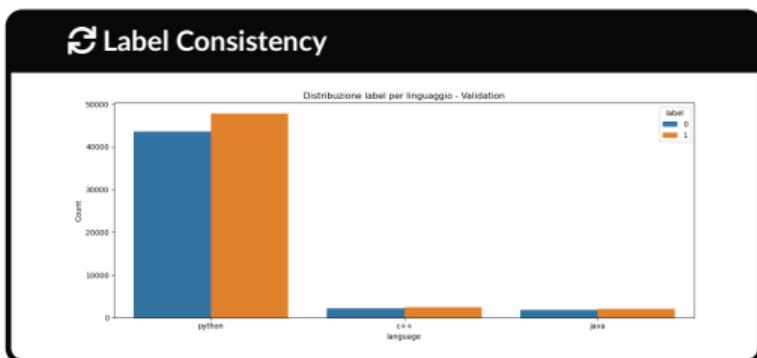
Analisi del Training Set (500k sample). Notiamo un forte bias verso linguaggi popolari e generatori specifici.



Insight: Il modello rischia di sovra-adattarsi allo stile di GPT-4 se non regolarizzato.

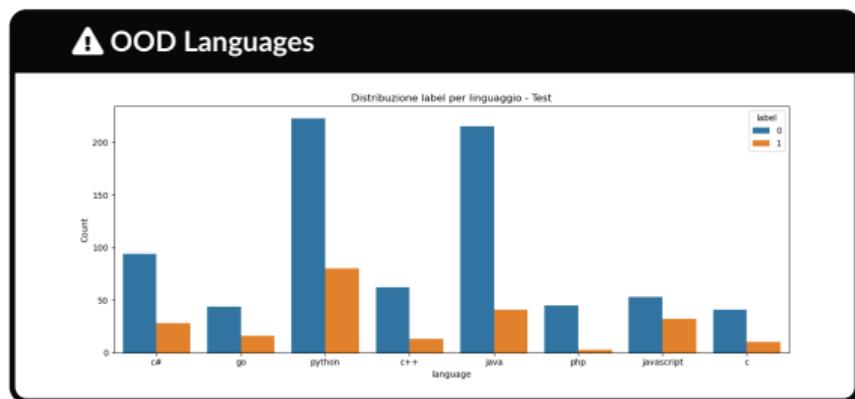
Task A [Validation]: Consistency Check

Verifica che il Validation Set rispecchi la distribuzione del Train per evitare il "Data Leakage" o shift distribuzionali.



Task A [Test]: The "Unseen" Challenge

Il Test Set introduce la vera difficoltà: linguaggi e generatori **MAI VISTI** durante il training.



Impact Analysis

Il grafico mostra la comparsa di:

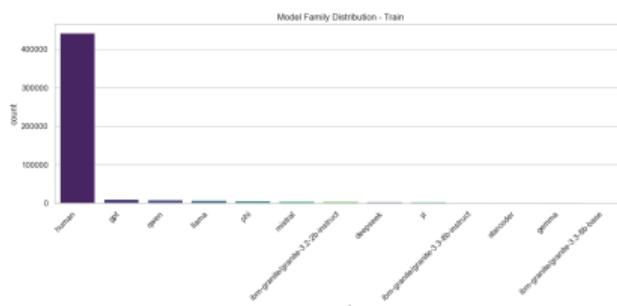
- ▶ Go, PHP, C#: Assenti nel train.
- ▶ **Obiettivo:** Valutare se il modello ha imparato la "struttura dell'AI" o solo la "sintassi di Python".

Ecco perché usiamo feature agnostiche come la **Perplexity**.

Task B [Train]: Fingerprinting Analysis

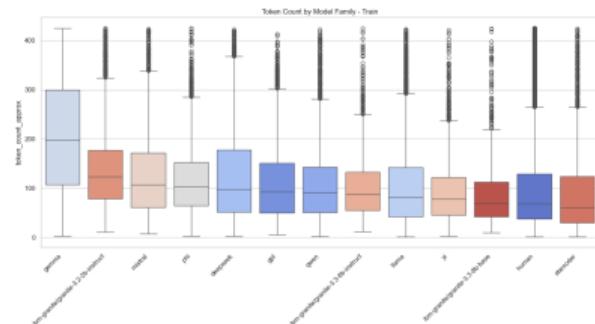
Analisi delle caratteristiche uniche dei modelli AI per l'attribuzione di paternità.

Extreme Imbalance



Human (90%) oscura le 10 classi AI.

Verbosity (Tokens)

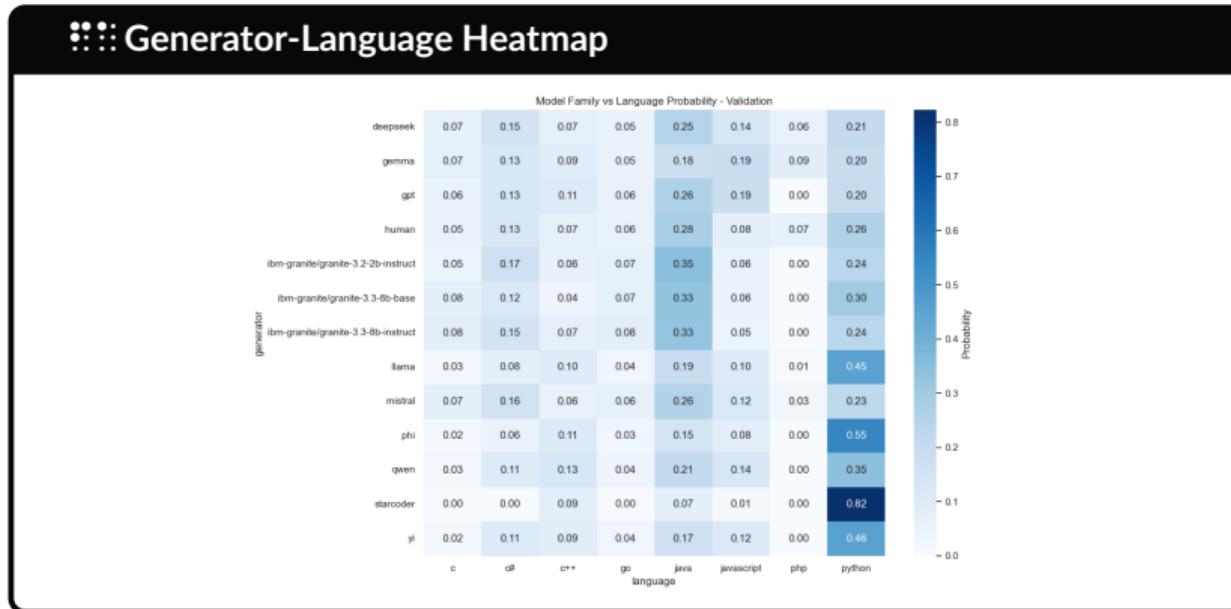


GPT-4 (verboso) vs Llama (conciso).

Strategy: Utilizzo di Focal Loss per le classi rare e feature di lunghezza per distinguere i modelli.

Task B [Validation]: Correlation Map

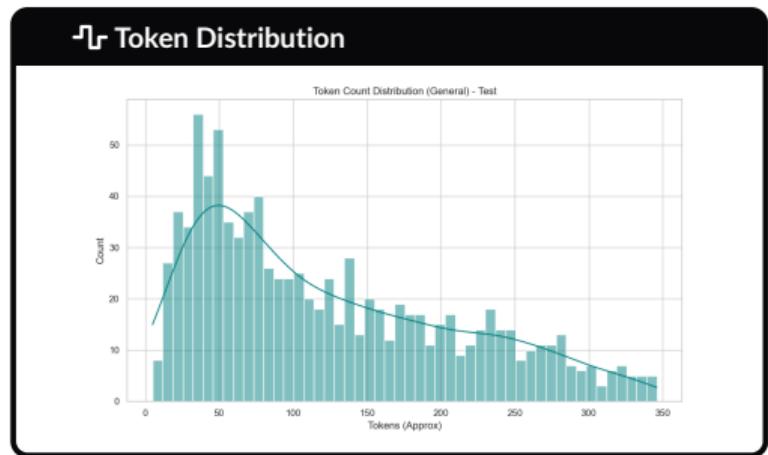
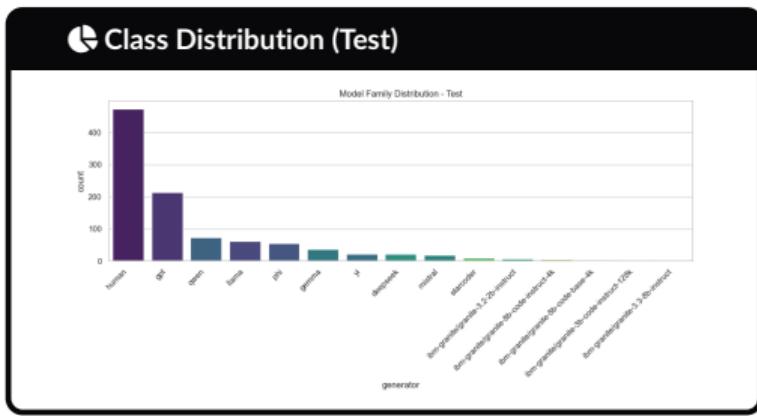
Analisi delle correlazioni condizionate: "Quale modello scrive in quale linguaggio?"



Alcuni modelli sono specializzati (es. StarCoder su Java), creando pattern riconoscibili.

Task B [Test]: Robustness Verification

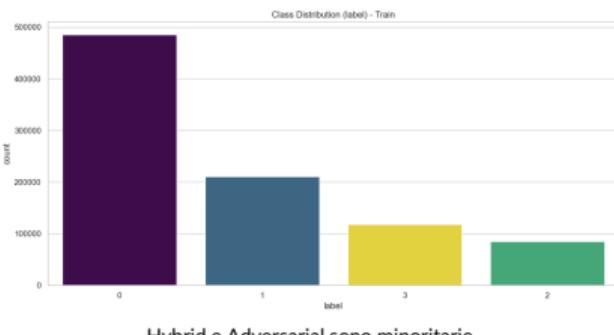
Verifica finale sulla distribuzione dei token nel set di Test per confermare l'assenza di drift anomali.



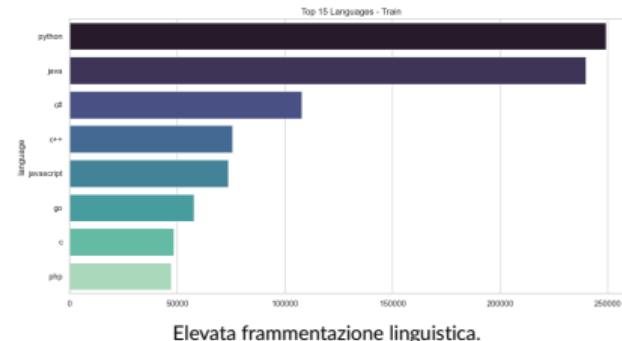
Task C [Train]: The 4-Class Problem

Analisi delle classi complesse: **Hybrid** (Uomo+AI) e **Adversarial** (Offuscato).

⌚ Target Distribution



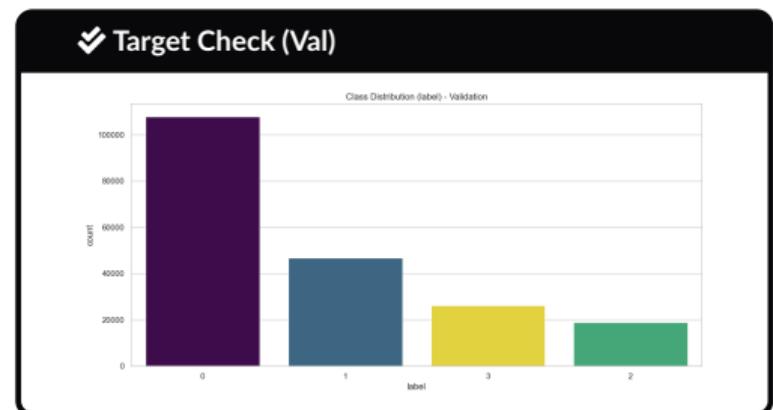
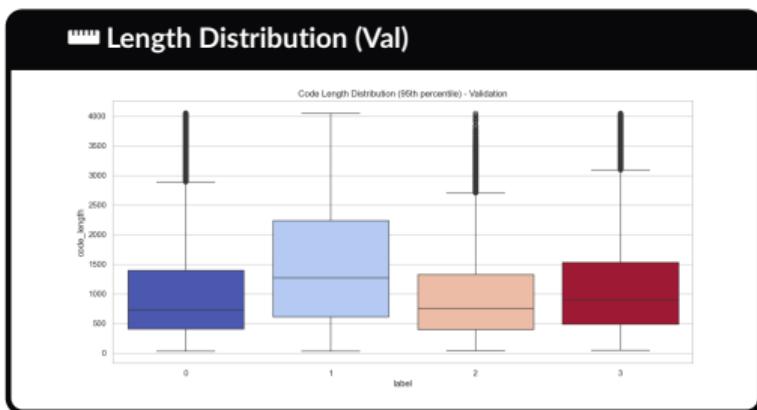
🔤 Language Variety



Solution: *Structural Noise Injection* per aumentare i campioni rari.

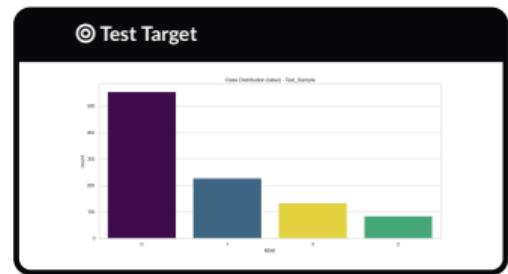
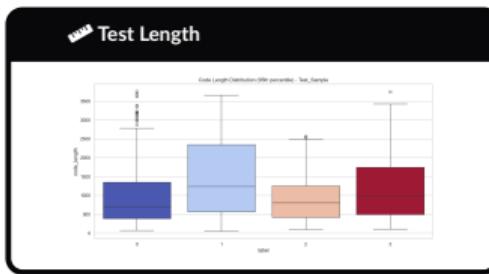
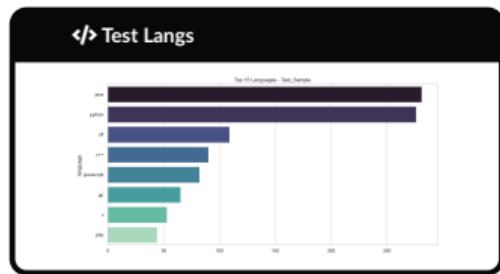
Task C [Validation]: Length & Features

Controllo della distribuzione delle lunghezze per settare correttamente la *max_length* del Transformer (GraphCodeBERT).



Task C [Test]: Preparing for Submission

Analisi finale sul Sample Test Set per garantire che la pipeline di inferenza gestisca correttamente i dati non visti.



Grazie per l'attenzione

 github.com/giovanniIacuzzo
 giovanni.iacuzzo@unikore.it