# Unige 23/09/25 First lesson  IP

Enrico puppo- teoria

Machine learning

Spiegazione “ai” (chat gpt,gemini,llama,clanche). Nominated Large language models LLM.

Generative pre-trained transformer (gpt).”Le macchine non sono intelligenti ma fanno operazioni simili a quelli umani”.L impatto del informatico.
Large- hanno cercato tutto internet e generano una risposta “plausibile-coerente di per se stessa e coerente con la domanda che abbiamo fatto”. Plausibili e diverse da corrette.
Noi facciamo domande al ai ma ci piacerebbe ottenere dei risultati corrette .la differenza
Anche pericolosi LLM,Ai  2 motivi,1 si occupano loro, il problema che ci riguarda da vicino e che queste cose sapevano e sanno già fare molti dei task che fino a ieri venivano svolti da umani.(informatici).tendenza affidarsi prendere risposte plausibili come se fosse corretta.
Uso corretto degli LLM

- Svolgere compiti “banali”,nel senso che io sono in grado di valutare se è corretto o no.
- Farsi suggerire modi di affrontare problemi complessi.Solo suggerimenti perché se no ci troviamo dove non si capisce se la risposta è plausibile o corretta.Anche qui dobbiamo essere in grado di capire se il suggerimento e utile o no.

Uso scoretto.

- Chiedere e fidarsi della risposta

  * carlo m cipolla -allegro non troppo
  Grafico cartesiano

Asse y  vantaggio per gli altri
Asse x vantaggi per se

## 4 categories-nei lati del disegno del grafico

1. Intelligenti
2. Sprovveduti
3. Banditi
4. stupidi

—----------

## Appunti Lezione del 23/09/25 FATTE DA CHAT GPT* IN ORDINE

**Corso:** IP (Informatica e Processi) **Docente:** Enrico Puppo

---

## 1. Introduzione all'Intelligenza Artificiale (AI)

- **Machine Learning:** Concetto di base.
- **Spiegazione dell'AI:** L'AI non è intelligente nel senso umano, ma esegue operazioni che simulano il comportamento umano. L'impatto dell'informatica è significativo.
- **Large Language Models (LLM):** Modelli come ChatGPT, Gemini, Llama e Claude.
- **Caratteristiche degli LLM:**
    - **"Large":** Si basano sull'analisi di una vasta quantità di dati (es. tutto Internet) per generare risposte.
    - **"Generative":** Producono una risposta che è **plausibile** e **coerente** sia con sé stessa che con la domanda posta.
    - **Attenzione:** La plausibilità è diversa dalla **correttezza**. Il rischio è che le persone si fidino di risposte solo perché sembrano coerenti, senza verificarne l'esattezza.

---

## 2. Usi Corretti e Scorretti degli LLM

- **Uso corretto:**
    - Svolgere compiti "banali" di cui si è in grado di verificare la correttezza.
    - Chiedere suggerimenti per affrontare problemi complessi, ma senza fidarsi ciecamente e mantenendo la propria capacità di giudizio.
- **Uso scorretto:**
    - Chiedere risposte e fidarsi della loro correttezza senza alcuna verifica.

---

## 3. La Teoria di Carlo M. Cipolla

Questa parte si riferisce al libro "Allegro ma non troppo" di Carlo M. Cipolla e descrive le categorie umane basate sui loro comportamenti.

- **Concetto:** La teoria si basa su un grafico cartesiano per classificare gli individui.
- **Assi del grafico:**
    1. **Asse Y:** Vantaggio per gli altri.
    2. **Asse X:** Vantaggio per sé stessi.
- **Le 4 Categorie:**
    1. **Intelligenti:** Chi trae vantaggio per sé e allo stesso tempo ne crea anche per gli altri (quadrante in alto a destra).
    2. **Sprovveduti:** Chi crea vantaggio per gli altri, ma a proprio svantaggio (quadrante in alto a sinistra).

3. **Banditi:** Chi crea vantaggio per sé stesso, ma a svantaggio degli altri (quadrante in basso a destra).
4. **Stupidi:** Chi causa svantaggio per gli altri senza trarne alcun vantaggio per sé stesso, causando anche un danno a sé (quadrante in basso a sinistra).

---

**Sito del corso:** 2025.aulaweb.unige.it

Teoria
2025.aulaweb.unige.it

**Come funziona passo passo:**

1. **Caso base 1**
   **Se la stringa ha lunghezza `0` o `1`, restituisce `true` (perché ogni stringa vuota o con un solo carattere è palindroma).**

2. **Caso base 2**
   **Se il primo carattere è diverso dall'ultimo, restituisce `false` (perché non può essere un palindromo).**

3. **Passo ricorsivo**
   **Se invece coincidono, la funzione:**

   - **elimina il primo e l'ultimo carattere (`substr(1, s.size()-2)`),**
   - **stampa la nuova stringa (per debug),**
   - **e richiama sé stessa (`is_palindrome`) su quella sottostringa.**

**—----------**

# 1) Calcolare l'età media degli studenti di IP

**Dati necessari:** numero di studenti `N`, e un vettore `ETA[1..N]` contenente l'età di ciascuno.

**Algoritmo (idea):** sommare tutte le età e dividere per il numero di studenti. Gestire il caso `N = 0`.

**Pseudocodice**

```
Algoritmo EtàMedia
  Input: N, ETA[1..N]
  Se N = 0 allora
    Stampa "Nessuno studente: impossibile calcolare la media"
    Ritorna
  FineSe

  somma ← 0
  Per i da 1 a N
    somma ← somma + ETA[i]
  FinePer

  media ← somma / N
  Stampa "Età media = ", media
  Ritorna media
FineAlgoritmo
```

---

# 2) Perimetro e area di un rettangolo

**Dati necessari:** lunghezza dei due lati (`latoA`, `latoB`) — spesso chiamati base e altezza.

**Formule:**

- Perimetro = `2 * (latoA + latoB)`
- Area = `latoA * latoB`

**Pseudocodice**

```
Algoritmo Rettangolo
  Input: latoA, latoB   // devono essere > 0
  Se latoA <= 0 oppure latoB <= 0 allora
    Stampa "Dimensioni non valide"
    Ritorna
  FineSe
```

```
  perimetro ← 2 * (latoA + latoB)
  area ← latoA * latoB

  Stampa "Perimetro = ", perimetro
  Stampa "Area = ", area
  Ritorna perimetro, area
FineAlgoritmo
```

---

# 3) Perimetro e area di un triangolo

**Dati necessari (opzioni):**

- Se conosci **i tre lati** `a`, `b`, `c` → puoi calcolare il perimetro e, con Heron, l'area.
- Se conosci **base** `b` e **altezza relativa** `h` → area = `b * h / 2` (più semplice).

**Controlli:** se usi i lati, verificare la disuguaglianza triangolare: `a + b > c` e `a + c > b` e `b + c > a`.

**Pseudocodice (Heron)**

```
Algoritmo Triangolo_Heron
  Input: a, b, c   // lunghezze dei lati
  Se a <= 0 oppure b <= 0 oppure c <= 0 allora
    Stampa "Lati non validi"
    Ritorna
  FineSe

  Se non (a + b > c e a + c > b e b + c > a) allora
    Stampa "Non è un triangolo valido"
    Ritorna
  FineSe

  perimetro ← a + b + c
  s ← perimetro / 2        // semiperimetro
  area ← radice_quadrata( s * (s - a) * (s - b) * (s - c) )
  Stampa "Perimetro = ", perimetro
  Stampa "Area (Heron) = ", area
```

```
  Ritorna perimetro, area
FineAlgoritmo
```

**Pseudocodice (base/altezza)**

```
Algoritmo Triangolo_BaseAltezza
  Input: base, altezza
  Se base <= 0 oppure altezza <= 0 allora
    Stampa "Dimensioni non valide"
    Ritorna
  FineSe

  area ← base * altezza / 2
  Stampa "Area = ", area
  Ritorna area
FineAlgoritmo
```

---

# 4) Giocare a forbice-carta-sasso (rock-paper-scissors)

**Dati necessari:** scelta del giocatore 1 (`scelta1`) e del giocatore 2 (`scelta2`). Dominio delle scelte: {`"sasso"`, `"forbice"`, `"carta"`}.

**Regole:**

- sasso batte forbice
- forbice batte carta
- carta batte sasso

**Pseudocodice (una singola partita)**

```
Algoritmo ForbiceCartaSasso
  Input: scelta1, scelta2    // stringhe: "sasso", "forbice",
"carta"

  Se scelta1 = scelta2 allora
    Stampa "Pareggio"
```

```
    Ritorna "pareggio"
  FineSe

  Se (scelta1 = "sasso" e scelta2 = "forbice") oppure
     (scelta1 = "forbice" e scelta2 = "carta") oppure
     (scelta1 = "carta" e scelta2 = "sasso") allora
    Stampa "Giocatore 1 vince"
    Ritorna 1
  Altrimenti
    Stampa "Giocatore 2 vince"
    Ritorna 2
  FineSe
FineAlgoritmo
```

**Variante: giocatore contro computer (scelta random del computer)**

```
Algoritmo GiocaVsComputer
  Input: sceltaGiocatore
  sceltaComputer ← Random_Choice(["sasso","forbice","carta"])
  Vincitore ← ForbiceCartaSasso(sceltaGiocatore, sceltaComputer)
  Stampa "Computer ha scelto: ", sceltaComputer
  Stampa risultato basato su Vincitore
  Ritorna Vincitore
FineAlgoritmo
```

**Opzionale:** loop "best of N" accumulando punti finché un giocatore raggiunge `ceil(N/2)`.

---

# 5) Calcolare i numeri primi tra 1 e 120

Due approcci comuni: controllo per divisori (trial division) o il Crivello di Eratostene (più efficiente).

**Pseudocodice: Crivello di Eratostene**

```
Algoritmo Crivello(n)
  Input: n
  crea array booleano isPrime[0..n], inizializza tutti a True
  isPrime[0] ← False
  isPrime[1] ← False
```

```
  Per p da 2 a floor(sqrt(n))
    Se isPrime[p] allora
      Per multiplo da p*p a n con passo p
        isPrime[multiplo] ← False
      FinePer
    FineSe
  FinePer

  // raccolta risultati
  ListaPrimi ← vuota
  Per i da 2 a n
    Se isPrime[i] allora
      aggiungi i a ListaPrimi
    FineSe
  FinePer
  Ritorna ListaPrimi
FineAlgoritmo
```

**Primi tra 1 e 120 (risultato):**
2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113

(totale: 30 numeri primi)

---

# 6) Esercizi aggiuntivi — altezza media, età più giovane, età più vecchia

Possiamo risolvere i tre problemi con **un unico passaggio** sui dati, mantenendo somma, minimo e massimo.

**Dati necessari:** numero studenti `N`, vettore `ETA[1..N]` (per le età) o `ALTEZZA[1..N]` (per le altezze). Qui mostro l'algoritmo che calcola media altezza, età minima e massima in un unico ciclo se si hanno entrambe le informazioni; altrimenti adatta.

**Pseudocodice (media altezza)**

```
Algoritmo AltezzaMedia
  Input: N, ALTEZZA[1..N]
  Se N = 0 allora
```

```
    Stampa "Nessuno studente"
    Ritorna
  FineSe

  somma ← 0
  Per i da 1 a N
    somma ← somma + ALTEZZA[i]
  FinePer

  media ← somma / N
  Stampa "Altezza media = ", media
  Ritorna media
FineAlgoritmo
```

**Pseudocodice (età più giovane e più vecchio)**

```
Algoritmo EtàMinMax
  Input: N, ETA[1..N]
  Se N = 0 allora
    Stampa "Nessuno studente"
    Ritorna
  FineSe

  minEta ← ETA[1]
  maxEta ← ETA[1]
  Per i da 2 a N
    Se ETA[i] < minEta allora minEta ← ETA[i]
    Se ETA[i] > maxEta allora maxEta ← ETA[i]
  FinePer

  Stampa "Età più giovane = ", minEta
  Stampa "Età più vecchia = ", maxEta
  Ritorna minEta, maxEta
FineAlgoritmo
```

**Pseudocodice combinato (media altezza, età min, età max in un passaggio)**

```
Algoritmo StatisticheStudenti
  Input: N, ETA[1..N], ALTEZZA[1..N]   // se non fornite alcune
info, adattare
  Se N = 0 allora
```

```
    Stampa "Nessuno studente"
    Ritorna
  FineSe

  sommaAltezza ← 0
  minEta ← ETA[1]
  maxEta ← ETA[1]

  Per i da 1 a N
    sommaAltezza ← sommaAltezza + ALTEZZA[i]
    Se ETA[i] < minEta allora minEta ← ETA[i]
    Se ETA[i] > maxEta allora maxEta ← ETA[i]
  FinePer

  mediaAltezza ← sommaAltezza / N

  Stampa "Altezza media = ", mediaAltezza
  Stampa "Età più giovane = ", minEta
  Stampa "Età più vecchia = ", maxEta
  Ritorna mediaAltezza, minEta, maxEta
FineAlgoritmo
```