Profice

AI ACADEMY

Applicare l’Intelligenza Artificiale nello sviluppo software

AI ACADEMY

Model evaluation avanzata e osservabilità
04/07/2025

INTRODUZIONE DELL'ISTRUTTORE

Tamas Szakacs

Formazione

- Laureato come programmatore matematico
- MBA in management

Principali esperienze di lavoro

- Amministratore di sistemi UNIX
- Oracle DBA
- Sviluppatore di Java, Python e di Oracle PL/SQL
- Architetto (solution, enterprise, security, data)
- Ricercatore tecnologico e interdisciplinare di IA

Dedicato alla formazione continua

- Teorie, modelli, framework IA
- Ricerche IA
- Strategie aziendali
- Trasformazione digitale
- Formazione professionale

email: tamas.szakacs@proficegroup.it

MOTIVI E RIASSUNTO DEL CORSO

L'**Intelligenza Artificiale (AI)** è oggi il motore dell'innovazione in ogni settore, grazie alla sua capacità di analizzare dati, automatizzare processi e generare nuove soluzioni. Questo corso offre una panoramica completa e pratica sullo sviluppo di applicazioni AI moderne, guidando i partecipanti dall'ideazione al rilascio in produzione.

Attraverso una **combinazione di teoria chiara ed esercitazioni pratiche**, saranno affrontate le tecniche e gli strumenti più attuali: **machine learning, deep learning, reti neurali, Large Language Models (LLM), Transformers, Retrieval Augmented Generation (RAG)** e progettazione di agenti AI.
Le competenze acquisite saranno applicate in progetti concreti, dallo sviluppo di chatbot all'integrazione di modelli generativi, fino al deploy di soluzioni AI in ambienti reali e collaborativi.

Il percorso è pensato per chi vuole imparare a progettare, valutare e integrare sistemi AI di nuova generazione, con particolare attenzione alle best practice di programmazione, collaborazione in team, sicurezza, valutazione delle performance ed etica dell'AI.

DURATA: 17 GIORNI

OBIETTIVI

Il percorso formativo è progettato per **giovani consulenti junior**, con una conoscenza base di programmazione, che stanno iniziando un percorso professionale nel settore AI.

L’obiettivo centrale è fornire una panoramica pratica, completa e operativa sull’intelligenza artificiale moderna, guidando ogni partecipante attraverso tutte le fasi fondamentali.

OBIETTIVI

- Allineare conoscenze AI, ML, DL di tutti i partecipanti
- Saper usare e orchestrare modelli LLM (closed e open-weight)
- Costruire pipeline RAG complete (retrieval-augmented generation)
- Progettare agenti AI semplici con strumenti moderni (LangChain, tool calling)
- Capire principi di valutazione, robustezza e sicurezza dei sistemi GenA
- Migliorare la produttività come sviluppatori usando tool GenAI-driven
- Padroneggiare best practice di sviluppo, versioning e deploy AI
- Introdurre i fondamenti di Graph Data Science e Knowledge Graph
- Ottenere capacità di valutazione dei modelli e metriche
- Comprensione dell'etica e dei bias nei modelli di intelligenza artificiale
- Approfondire le normative di riferimento: AI Act, compliance e governance AI

Il corso è **estremamente pratico** (circa il 40% del tempo in esercitazioni hands-on, notebook, challenge e hackathon), con l'utilizzo di Google Colab, GitHub, e tutti gli strumenti necessari per lavorare su progetti reali e simulati.

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
1	Git & Python clean-code	Collaborazione su progetti reali, versionamento, codice pulito e testato
2	Machine Learning Supervised	Modelli supervisionati per predizione e classificazione
3	Machine Learning Unsupervised	Clustering, riduzione dimensionale, scoperta di pattern
4	Prompt Engineering avanzato	Scrivere e valutare prompt efficaci per modelli generativi
5	LLM via API (multi-vendor)	Uso pratico di modelli LLM via API, autenticazione, deployment
6	Come costruire un RAG	Pipeline end-to-end per Retrieval-Augmented Generation
7	Tool-calling & Agent design	Progettare agenti AI che usano strumenti esterni
8	Hackathon: Agentic RAG	Challenge pratica: chatbot agentico RAG in team

STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).

La progettazione sintetica delle giornate:

Giorno	Tema	Breve descrizione
9	Hackathon: Rapid Prototyping	Da prototipo a web-app con Streamlit e GitHub
10	AI Productivity Tools	Workflow con IDE AI-powered, automazione e refactoring assistito
11	Docker & HF Spaces Deploy	Deployment di app GenAI containerizzate o su HuggingFace Spaces
12	AI Act & ISO 42001 Compliance	Fondamenti di compliance e governance AI
13	Knowledge Base & Graph Data Science	Introduzione a Knowledge Graph e query con Neo4j
14	Model evaluation & osservabilità	Metriche avanzate, explainability, strumenti di valutazione
15	AI bias, fairness ed etica applicata	Analisi dei rischi, metriche e mitigazione dei bias
16-17	Project Work & Challenge finale	Lavoro a gruppi, POC/POD, presentazione e votazione progetti

METODOLOGIA DEL CORSO

1. Approccio introduttivo ma avanzato

Il corso è introduttivo nei concetti base dell'AI applicata allo sviluppo, ma affronta anche tecnologie, modelli e soluzioni avanzate per garantire un apprendimento completo.

2. Linguaggio adattato

Il linguaggio utilizzato è chiaro e adattato agli studenti, con spiegazioni dettagliate dei termini tecnici per favorirne la comprensione e l'apprendimento graduale.

3. Esercizi pratici

Gli esercizi pratici sono interamente svolti online tramite piattaforme come Google Colab o notebook Python, eliminando la necessità di installare software sul proprio computer.

4. Supporto interattivo

È possibile porre domande in qualsiasi momento durante le lezioni o successivamente via email per garantire una piena comprensione del materiale trattato.

NOTA

Il corso segue un **approccio laboratoriale**: ogni giornata combina sessioni teoriche chiare e concrete con molte attività pratiche supervisionate, per sviluppare *competenze reali* immediatamente applicabili.

I partecipanti lavoreranno spesso in gruppo, useranno notebook in Colab e versioneranno codice su GitHub, vivendo una vera simulazione del lavoro in azienda AI.

Nessun prerequisito avanzato richiesto: si partirà dagli strumenti e flussi fondamentali, con una crescita graduale verso le tecniche più attuali e richieste dal mercato.

ORARIO TIPICO DELLE GIORNATE

Orario	Attività	Dettaglio
09:00 – 09:30	Teoria introduttiva	Concetti chiave, schema della giornata
09:30 – 10:30	Live coding + esercizio guidato	Esempio pratico, notebook Colab
10:30 – 10:45	*Pausa breve*	
10:45 – 11:30	Approfondimento teorico	Tecniche, best practice
11:30 – 12:30	Esercizio hands-on individuale	Sviluppo o completamento di codice
12:30 – 13:00	Discussione soluzioni + Q&A	Condivisione e correzione
13:00 – 14:00	*Pausa pranzo*	
13:30 – 14:15	Teoria avanzata / nuovi tools	Nuovi strumenti, pattern, demo
14:15 – 15:30	Esercizio a gruppi / challenge	Lavoro di squadra su task reale
15:30 – 15:45	*Pausa breve*	
15:45 – 16:30	Sommario teorico e pratico	
16:30 – 17:00	Discussioni, feedback	Riepilogo, best practice, domande aperte

DOMANDE?

Cominciamo!

OBIETTIVI DELLA GIORNATA

Obiettivi della giornata

- Comprendere le metriche avanzate di valutazione NLP/generative.
- Saper valutare modelli con criteri robusti (oltre accuracy/F1).
- Analizzare modelli generativi e detection di hallucination.
- Applicare explainability di AI.
- Progettare e realizzare la valutazione dei modelli.

TIPI DI VALUTAZIONE NEI MODELLI AI

Supervised Evaluation

- Modelli: classificazione, regressione, sequence labeling.
- Si valuta confrontando le predizioni con le etichette vere.
- Metriche tipiche: accuracy, precision, recall, F1, AUC, MAE, confusion matrix.
- Esempio: classificazione di messaggi SMS come “positivo/negativo”.

Generative Evaluation

- Modelli: LLM, generatori di testo, riassunto, traduzione, Q&A aperto.
- Si valuta la qualità, coerenza e aderenza all’input delle risposte generate.
- Metriche automatiche: BLEU, ROUGE, METEOR, BERTScore.
- Spesso serve anche valutazione manuale (“human eval”) per verificare senso, factual correctness, stile.
- Esempio: generare una risposta a una domanda aperta o scrivere un riassunto.

In sintesi:

La valutazione supervised misura l’esattezza rispetto a target noti.

La valutazione generative richiede metriche diverse e spesso anche un giudizio umano.

METRICHE CLASSICHE DI VALUTAZIONE SUPERVISIONATA

Accuracy

- Percentuale di predizioni corrette sul totale dei casi.
- *Formula:* (True Positive + True Negative) / Totale campioni

Precision

- Quante delle predizioni positive sono davvero corrette.
- Formula: True Positive / (True Positive + False Positive)
- Esempio: Su 100 “spam” previsti, 80 erano davvero spam → Precision = 80%

Recall (Sensibilità)

- Quanti dei veri positivi sono stati riconosciuti dal modello.
- Formula: True Positive / (True Positive + False Negative)
- Esempio: Su 100 spam reali, 75 individuati → Recall = 75%

F1 Score

- Media armonica tra precision e recall, bilancia i due valori.
- Formula: 2 × (Precision × Recall) / (Precision + Recall)
- Esempio: Precision = 80%, Recall = 75% → F1 ≈ 77%

Utilizzo:
Le metriche classiche sono fondamentali per valutare modelli di classificazione come NER, sentiment analysis, spam detection.

Attenzione: Su dataset sbilanciati, l’accuracy può essere fuorviante!

LIMITI DELLE METRICHE CLASSICHE NEL NLP

Accuracy, precision, recall, f1-score hanno i loro limiti

- **Non valutano la qualità semantica**
  Le metriche classiche misurano corrispondenza esatta tra output e target, non la “bontà” del testo generato.
- **Non adatte per risposte multiple valide**
  Una domanda può avere risposte diverse, tutte corrette (es. Q&A, traduzione, riassunto).
- **Non valutano la fluidità o la coerenza**
  Un output grammaticalmente corretto, ma semanticamente errato, può avere punteggi elevati.
- **Insensibili a piccoli errori**
  Parole fuori ordine o sinonimi possono penalizzare eccessivamente l’output.
- **Problemi su generazione di testo lunga**
  Accuracy, precision, recall funzionano bene su classificazione, ma poco su generazione di frasi o documenti.

Conclusione:
Per valutare sistemi generativi (es. chatbot, summarization, traduzione automatica) servono metriche più avanzate e specifiche, capaci di cogliere la qualità linguistica e la vicinanza semantica tra output e riferimento.

ESERCIZI: METRICHE DI CLASSIFICAZIONE

Dataset SMS: metriche di classificazione

Obiettivo: Valutare modelli di classificazione (spam, sentiment, intent).

Metriche: accuracy, precision, recall, F1.

Come fare:

- Confronta i risultati con le etichette vere.
- Confronta le metriche del modello con i dati originali e dopo il fine-tuning.
- Usa metriche (`sklearn.metrics`) per la classificazione dei messaggi SMS etichettati (spam/ham o sentiment).
- Confronta le predizioni di diversi modelli (ad es. un classificatore RandomForest) con le etichette vere.

ESERCIZI: METRICHE DI CLASSIFICAZIONE

Dataset SMS: metriche di classificazione

Obiettivo: Valutare modelli di classificazione (spam, sentiment, intent).

Metriche: accuracy, precision, recall, F1.

Le diverse metriche:

```
from sklearn.metrics import ...

# y_true = etichette vere
# y_pred = etichette previste dal modello
classification_report(y_true, y_pred) # Varie metriche

accuracy_score(y_true, y_pred)

precision_score(y_true, y_pred, average='macro'/'weighted'/...)

recall_score(y_true, y_pred, ...)

f1_score(y_true, y_pred, ...)
```

METRICHE AVANZATE NLP: INTRODUZIONE

Perché servono nuove metriche?
Le metriche classiche non bastano per valutare la qualità semantica, la naturalezza e la varietà delle risposte generate dai modelli AI.

Obiettivi delle metriche avanzate:

- Confrontare testo generato e testo di riferimento su **livello semantico** e non solo sintattico.
- Premiare sinonimi, riformulazioni corrette e flessibilità linguistica.
- Misurare la similarità e la comprensibilità tra output e target.

Esempi di metriche avanzate:

- **BLEU**
- **ROUGE**
- **METEOR**
- **BERTScore**
- …e altre ancora, specifiche per NLP generativo.

Queste metriche sono essenziali per valutare modelli di:

- Generazione di testo (chatbot, summary)
- Q&A avanzati
- Modelli generativi multi-turno
- Traduzione automatica

Prossime slide: Approfondimento sulle singole metriche.

BLEU (BILINGUAL EVALUATION UNDERSTUDY)

Come funziona:

- **BLEU** misura la qualità del testo generato confrontandolo con uno o più riferimenti “gold”.
- Analizza le **n-gram** (sequenze di parole): più le sequenze del testo generato coincidono con quelle dei riferimenti, più alto è il punteggio.
- Calcola una **precisione media** su diverse lunghezze di n-gram (di solito da 1 a 4).
- Applica un “brevity penalty” per penalizzare risposte troppo corte.

Vantaggi:

- **Standard de facto** per la valutazione automatica della traduzione e sintesi testuale.
- **Semplice, veloce e interpretabile.**
- Non richiede giudizio umano.

Limiti:

- Non valuta la correttezza semantica: **premia solo la coincidenza letterale delle sequenze**.
- Penalizza sinonimi, riformulazioni e risposte corrette ma diverse dai riferimenti.
- Sensibile alla lunghezza delle frasi e all’ordine delle parole.
- **Non adatto** per dialoghi complessi o generazione creativa.

In sintesi:

BLEU è utile per confrontare versioni di modelli su compiti ben definiti e strutturati (es: traduzione), ma **non è sufficiente da solo** per valutare la qualità di testo generato in contesti più aperti.

ROUGE: COSA MISURA, VARIANTI

Cos'è ROUGE?

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) è una famiglia di metriche sviluppate per valutare la qualità di testi generati automaticamente (come riassunti, traduzioni, risposte di chatbot), confrontandoli con uno o più testi di riferimento umani (gist: essenziale).

Cosa misura?

- Sovrapposizione tra parole, frasi o sequenze**:** misura quanto il testo generato contiene elementi (n-grammi, frasi, sequenze) presenti nel testo di riferimento.
- È focalizzato su recall (copertura del riferimento), quindi premia il contenuto "copiato" correttamente dal riferimento.
- Usato soprattutto per valutare riassunti e risposte dove la copertura informativa è prioritaria.

Principali varianti:

- **ROUGE-N:** Conta gli n-grammi in comune tra generato e riferimento (es. ROUGE-1: parole singole, ROUGE-2: coppie di parole).
- **ROUGE-L:** Basato sulla Longest Common Subsequence (sequenza più lunga comune tra due testi).
- **ROUGE-S/SU:** Basato sugli skip-bigram, coppie di parole che possono non essere adiacenti.

Sintesi:

ROUGE è una metrica versatile per la valutazione automatica della qualità e completezza dei testi generati, con varianti adattabili ai diversi compiti di NLP.

METEOR: SIGNIFICATO E DIFFERENZE

Cos'è METEOR?

METEOR (Metric for Evaluation of Translation with Explicit ORdering) è una metrica automatica sviluppata per valutare la qualità delle traduzioni automatiche e di altri testi generati (riassunti, chatbot). È stata progettata per superare alcune limitazioni delle metriche come BLEU e ROUGE.

Caratteristiche principali:

- **Allineamento intelligente:** METEOR trova la miglior corrispondenza tra parole di testo generato e riferimento, usando non solo corrispondenza esatta, ma anche lemmi, sinonimi e stem (radici).
- **Calcolo di precisione e recall:** Combina entrambe (a differenza di BLEU che si concentra su precisione).
- **Penalità per l'ordine errato:** Penalizza frasi con parole giuste ma ordine sbagliato, migliorando la valutazione della fluidità.

Differenze rispetto a BLEU e ROUGE:

- BLEU misura solo la precisione degli n-grammi, ROUGE principalmente il recall: METEOR combina entrambe, risultando spesso più "equilibrato".
- METEOR tiene conto di varianti morfologiche e sinonimi, non solo corrispondenze esatte.
- Più correlato con il giudizio umano nella valutazione della qualità dei testi generati.

In sintesi:

METEOR è una metrica avanzata, sensibile alle sfumature linguistiche, che supera alcuni limiti delle metriche classiche, offrendo una valutazione più realistica della qualità del testo prodotto da modelli di NLP.

BERTSCORE: COS'È E PERCHÉ USARLO

Cos'è BERTScore?

BERTScore è una metrica recente per la valutazione automatica di testi generati (es. traduzioni, riassunti, risposte AI). Invece di confrontare solo le parole in comune, confronta la similarità semantica tra il testo generato e quello di riferimento usando modelli di linguaggio basati su BERT (o simili, come RoBERTa, DeBERTa).

Come funziona?

- **Embedding semantico:** Ogni parola o token nei due testi viene trasformato in un vettore tramite un modello BERT pre-addestrato.
- **Matching ottimale:** Si calcola la similarità coseno tra i vettori di ogni parola/token del testo generato e quelli del riferimento.
- **Score aggregato:** Il punteggio BERTScore riflette quanto il significato dei due testi è simile, indipendentemente dalla corrispondenza esatta delle parole.

BERTSCORE: COS'È E PERCHÉ USARLO

Perché usarlo?

- **Cattura il significato:** Valuta la semanticità e non solo l'overlap superficiale di parole.
- **Meno sensibile a sinonimi/parafrasi:** Premia le risposte che sono corrette ma espresse con parole diverse.
- **Correla meglio con il giudizio umano** rispetto a BLEU e ROUGE, specie in compiti di generazione complessa (QA, dialogo, riassunto, traduzione).

In sintesi:

BERTScore è ideale per valutare la qualità di testi generati dall'AI quando è importante il senso generale e non solo la corrispondenza letterale delle parole.

METRICHE CUSTOM: TASK-SPECIFIC EVALUATION

Cos'è una metrica custom?

Sono metriche definite ad hoc per valutare la performance di un modello AI su uno specifico compito (task-specific). Si costruiscono in base agli obiettivi reali dell'applicazione, ai vincoli di business, o alle esigenze dell'utente finale.

Perché usarle?

- Le metriche generiche (accuracy, BLEU, ecc.) a volte non riflettono bene l'utilità reale del modello nel contesto d'uso.
- Una metrica custom può considerare aspetti pratici, errori critici o casi d'uso particolari.

Esempi pratici:

- **NER per privacy:** Misurare la percentuale di dati sensibili non anonimizzati in un testo.
- **QA su knowledge base:** Penalizzare risposte senza fonte attendibile, o troppo vaghe.
- **Sentiment analysis in finanza:** Dare più peso agli errori che identificano erroneamente clienti insoddisfatti.
- **Document retrieval:** Valutare la precisione sui documenti più rilevanti (es. solo contratti sopra una certa soglia di importanza).

METRICHE CUSTOM: TASK-SPECIFIC EVALUATION

Come si progettano?

1. **Definisci cosa conta:** Quali errori sono davvero importanti?
2. **Progetta la formula:** Calcola lo score in modo che rifletta il valore per l'utente/business.
3. **Testa la correlazione:** La metrica custom deve corrispondere al giudizio umano o alle esigenze operative.

In sintesi:

Le metriche custom servono a misurare il successo "reale" del modello su compiti aziendali o applicativi specifici, andando oltre le metriche standard.

VALUTAZIONE MODELLI GENERATIVI: PECULIARITÀ

Cosa cambia rispetto ai modelli classici?

Nei modelli generativi (LLM, traduzione, sintesi testo, chatbot) **non esiste una "risposta unica e corretta"**.
La valutazione è spesso **qualitativa e soggettiva**: serve giudicare creatività, coerenza, utilità, stile.

Sfide principali

- **Multipli output validi:** Diversi testi possono essere "giusti" per la stessa domanda.
- **Metriche classiche poco efficaci:** Accuracy o F1 hanno poco senso, bisogna valutare similarità, qualità linguistica, informatività.
- **Allucinazioni:** I modelli possono inventare fatti plausibili ma errati, difficile da rilevare solo con metriche automatiche.
- **Bias e sicurezza:** Serve attenzione a bias nei dati e nei risultati generati.

Metodi di valutazione

- **Automatici:** BLEU, ROUGE, METEOR, BERTScore, MAUVE.
- **Umani:** Giudizio diretto degli esperti, ranking, preferenze utente, pairwise comparison.

VALUTAZIONE MODELLI GENERATIVI: PECULIARITÀ

Cosa si valuta nei modelli generativi?

- **Flessibilità e creatività:** L'output deve essere vario, interessante, non ripetitivo.
- **Aderenza alle istruzioni:** Il testo prodotto rispetta il prompt?
- **Fatti e fonti:** L'output è corretto e affidabile?
- **Fluenza linguistica:** L'output è naturale e leggibile.
- **Utilità nel contesto d'uso:** Soddisfa la richiesta dell'utente?

In sintesi:

Valutare modelli generativi richiede **metriche più sofisticate** e spesso **un intervento umano**, perché la varietà degli output e le allucinazioni pongono sfide uniche rispetto ai modelli classici.

ESERCIZI: METRICHE DI MODELLI NLP

NER + GPT: metriche di estrazione e generazione

Obiettivo: Pianifica di valutare l'accuratezza di estrazione entità e qualità delle risposte/generazione.

Metriche:

- **NER:** precision, recall, F1 per ogni tipo di entità (usando le etichette vere e le predette dal modello).
- **GPT (risposta/testo):** BLEU, ROUGE, BERTScore tra la risposta generata e una risposta di riferimento ("gold").

Come fare:

- Per il modello NER e GPT scegli quale valutazione usare.
- Considera come confrontare gli input documenti aziendali (o email/contratti finti).
- Prepara NER e/o chiedi a GPT di rispondere a una domanda.
- Pianifica come valutare le risposte o le entità con quelle corrette, calcolando le metriche.

MAUVE: METRICA PER GENERAZIONE

Cos’è MAUVE?

MAUVE è una metrica avanzata per valutare la qualità dei testi generati dai modelli linguistici.
Confronta la **distribuzione dei testi generati** dal modello con quella dei testi reali (gold reference).
Utilizza metodi di informazione teorica per misurare **quanto i testi generati “assomigliano” a quelli umani**.

Come funziona?

- Calcola una distanza tra la distribuzione delle sequenze generate dal modello e quella dei dati reali, usando tecniche di embedding e curve ROC.
- Produce un valore tra 0 e 1:
  - **1** = i testi generati sono indistinguibili da quelli reali
  - **0** = i testi sono completamente diversi

MAUVE: METRICA PER GENERAZIONE

Vantaggi

- Tiene conto sia della **diversità** sia della **qualità linguistica** dei testi.
- È robusto rispetto a semplici matching parola-per-parola (più generale di BLEU/ROUGE).
- Può essere usato per valutare qualsiasi tipo di generazione di testo, anche non allineata frase per frase.

Limiti

- È una metrica “globale”: valuta distribuzioni, non singoli esempi.
- Non fornisce feedback dettagliato sul perché un testo sia buono o meno.
- Richiede una quantità significativa di testi generati e reali per funzionare bene.

Quando usarla?

- Per confrontare modelli generativi (es. diverse versioni di LLM).
- Per capire se un modello “copia” troppo o se produce testo troppo casuale.

In sintesi:

MAUVE aiuta a capire **quanto un modello generativo sia vicino al comportamento umano**, valutando insieme creatività e coerenza rispetto ai dati reali.

GPT-EVAL & LLM-AS-A-JUDGE

Cos'è GPT-Eval?

- **GPT-Eval** è un approccio di valutazione automatica che sfrutta un modello LLM (come GPT-4) per valutare la qualità di un output generato da un altro modello o dallo stesso LLM.
- Può essere usato per **comparare risposte** (es. output A vs output B) o per **dare un punteggio** a una singola risposta rispetto a un prompt o una reference.

LLM-as-a-judge: cosa significa?

- Significa **usare un modello di linguaggio** (LLM) come "giudice automatico" per valutare la pertinenza, correttezza o stile di un output generato.
- L'LLM riceve un prompt che gli chiede di confrontare, valutare, assegnare punteggi o motivare una preferenza tra più risposte.

Come si usa?

- Si prepara un **prompt di valutazione** (evaluation prompt) che spiega al modello i criteri da seguire (accuratezza, completezza, chiarezza, ecc.).
- L'LLM "legge" la risposta da valutare e produce un giudizio sintetico, una motivazione o un punteggio.

GPT-EVAL & LLM-AS-A-JUDGE

Vantaggi

- **Scalabilità**: è possibile valutare migliaia di esempi senza bisogno di annotatori umani.
- **Flessibilità**: si possono cambiare facilmente i criteri di valutazione.
- **Consistenza**: elimina parte della soggettività umana (ma introduce quella del modello!).

Limiti

- **Bias**: l'LLM eredita i bias e le "idee" del modello di partenza.
- **Trasparenza**: talvolta le motivazioni sono generiche o poco interpretabili.
- **Necessità di prompt engineering**: la qualità della valutazione dipende molto da come si scrive il prompt.

Quando usarlo?

- Per confrontare due modelli in modo rapido.
- Per pre-filtrare risultati da sottoporre poi a validazione umana.
- In progetti di **auto-eval** in produzione o test A/B automatici.

In sintesi:

GPT-Eval (o "LLM-as-a-judge") è un modo potente e rapido per valutare la qualità dei modelli generativi, ma **va usato insieme a controlli umani e ad altre metriche**.

EVAL HARNESS – FRAMEWORK OPEN-SOURCE PER LLM

Cos'è lm-evaluation-harness?

Un framework open-source creato da EleutherAI per **benchmarking LLM** su oltre 60 task standard (come MMLU, GSM8K, ecc.

Usato nel popolare **Open LLM Leaderboard di Hugging Face** e da organizzazioni come NVIDIA, Cohere, Stability AI.

Caratteristiche principali

- Supporta modelli **locali** (HuggingFace, GPT-NeoX, vLLM, etc.) e **API commerciali** (OpenAI).
- Configurazione modulare per task tramite **YAML** o classi Python.
- Include tool per **prompt design (Promptsource)**, post-processing e supporto few-shot/jinja2.
- Compatibile con ambiente HPC/GPU, MPS, inference efficiente grazie a vLLM

Perché usarlo?

- **Riproducibilità** – mette ordine tra implementazioni diverse, migliorando la trasparenza .
- **Versatilità** – copre tasks diversi con metriche automatiche (accuracy, log-prob, generative scores).
- **Estendibile** – si possono aggiungere nuovi task o metriche personalizzate con facilità.

EVAL HARNESS – FRAMEWORK OPEN-SOURCE PER LLM

Quando è utile?

- Confronti tra modelli in modo rigoroso e su larga scala.
- Valutazioni standardizzate e comparabili.
- Ricerca, benchmarking e auditing.

Uso base

```
git clone https://github.com/EleutherAI/lm-evaluation-harness
cd lm-evaluation-harness
pip install -e .
lm_eval --model hf \
        --model_args pretrained=EleutherAI/gpt-j-6B \
        --tasks hellaswag,lambada_openai \
        --device cuda:0 \
        --batch_size auto
```

Conclusione:

lm-evaluation-harness è il **tool di riferimento open-source** per valutare in modo ripetibile, organizzato ed estensibile la qualità dei grandi modelli di linguaggio.

DOMANDE?

PAUSA

HALLUCINATION: COS'È E PERCHÉ È UN PROBLEMA

Cos'è una Hallucination nei modelli AI

In ambito AI e NLP, per "hallucination" si intende quando un modello genera risposte che sembrano plausibili ma sono **inesatte, inventate o completamente false**.

Tipicamente riguarda informazioni che **non sono presenti nei dati di input** o non hanno fondamento nei dati reali.

Perché è un problema

- **Affidabilità:** Le hallucination riducono la fiducia nell'output dell'AI, soprattutto in contesti sensibili (medicina, legale, business).
- **Impatto su decisioni:** Se l'utente prende decisioni basandosi su risposte allucinate, può incorrere in errori anche gravi.
- **Valutazione e auditing:** Rende più difficile il controllo della qualità dei modelli, perché gli errori non sono sempre prevedibili.
- **Compliance:** In alcuni casi (EU AI Act, regolamenti industriali) le hallucination possono costituire un rischio legale e di conformità.

HALLUCINATION: COS'È E PERCHÉ È UN PROBLEMA

Esempio pratico

- Un chatbot che, alla domanda "Qual è la capitale della Svizzera?", risponde "Stoccolma" anziché "Berna": l'informazione sembra plausibile ma è errata.
- Un sistema che inventa riferimenti bibliografici mai esistiti.

Conclusione

Rilevare e minimizzare le hallucination è una priorità nella valutazione di modelli generativi avanzati, sia per la **qualità** sia per la **sicurezza** delle applicazioni AI.

DETECTION DI HALLUCINATION: AUTOMATICO E MANUALE

Approcci automatici

- **Fact-checking automatico:** Si usano modelli o API che confrontano la risposta AI con fonti affidabili (Wikipedia, database pubblici).
- **Score di confidenza:** Analisi della probabilità associata alle risposte: valori troppo bassi possono segnalare possibili hallucination.
- **Similarity search:** Si confrontano le risposte con il materiale di input o con knowledge base; risposte poco correlate possono essere segnali di allucinazione.
- **Prompting con LLM:** Si può chiedere a un altro modello AI di valutare se la risposta è supportata dai dati (LLM-as-a-judge).
- **Metriche dedicate:** Alcuni strumenti, come MAUVE, sono progettati per rilevare incoerenze tra output generati e dati reali.

DETECTION DI HALLUCINATION: AUTOMATICO E MANUALE

Approcci manuali

Revisione umana: Gli output generati vengono letti e verificati da esperti o annotatori umani, che identificano risposte inventate o non supportate dai dati.
Checklist di validazione: Gli umani seguono linee guida specifiche per marcare le allucinazioni (es. controllo di fatti, verifica di nomi e dati).
Crowdsourcing: Si può distribuire il controllo a più revisori per aumentare la copertura e l'oggettività.

Conclusione
Nella pratica, si combinano metodi **manuali** e **automatici** per una rilevazione più efficace delle hallucination, specie nei contesti produttivi o critici.

ESERCIZIO: VALUTARE LE HALLUCINATION DI GPT+NER

Obiettivo:

Verificare se GPT, supportato da NER, inventa dati o entità inesistenti rispondendo su documenti aziendali.

Compito

- Scegli 5 estratti di email, fatture, contratti (anche inventati).
- Fai domande specifiche su nomi, date, importi, riferimenti contenuti nei testi.
- Usa NER+GPT per rispondere alle domande.

Controlla la risposta:

Se l'output menziona entità/dati che NON sono nel testo → segna come *hallucination*.
Compila una tabella: domanda | risposta | hallucination? (Sì/No).

Esempi di hallucination tipiche:

- GPT inventa un nome di cliente che non c'è nel documento.
- Fornisce una data sbagliata o mai menzionata.
- Riporta dettagli su un pagamento non presenti.

Risultato:

Tabella di risposte con evidenza delle hallucination, più 2 righe di commento su come evitarle (es. prompt migliore, dati di supporto, validazione manuale).

PROMPT EVALUATION: AUTOMATED VS HUMAN EVAL

Valutazione automatica

Descrizione: Utilizzo di metriche e modelli AI per misurare la qualità delle risposte a un prompt senza intervento umano.

Strumenti: BLEU, ROUGE, BERTScore, MAUVE, GPT-Eval, script di scoring.

Vantaggi:

- Veloce e scalabile su grandi dataset.
- Coerente (stesse regole applicate ovunque).

Limiti:

- Può non cogliere la qualità semantica profonda o le sfumature contestuali.
- Possibili errori nella valutazione di creatività, correttezza fattuale, o allineamento all'intento.

PROMPT EVALUATION: AUTOMATED VS HUMAN EVAL

Valutazione umana

Descrizione: Esaminatori umani leggono e giudicano la qualità delle risposte ai prompt.

Metodi:

- Valutazione diretta (rating su scala).
- Pairwise (confronto tra output diversi).
- Analisi qualitativa (note, commenti).

Vantaggi:

- Cattura sottigliezze, creatività, contesto e aderenza all'intento.
- Utile per tasks complessi o ad alto rischio.

Limiti:

- Costosa e lenta.
- Soggetta a bias individuali e variazioni soggettive.

In pratica:

Si usano entrambe: automatica per **screening veloce**, umana per **validazione finale** e casi critici.

PROMPT EVALUATION: BEST PRACTICE DI PROMPT TESTING

- **Definisci criteri chiari di successo**
  Prima di testare, stabilisci cosa significa una “buona risposta” (accuratezza, creatività, completezza, tono, ecc.).
- **Crea una suite di prompt rappresentativi**
  Usa esempi vari e realistici che riflettano i veri casi d’uso e edge case.
- **Automatizza dove possibile**
  Implementa test automatici con metriche (BLEU, ROUGE, ecc.) per valutare rapidamente molte risposte.
- **Valuta la robustezza**
  Modifica leggermente i prompt (parafrasi, errori di battitura, varianti) per testare la resilienza e la coerenza del modello.
- **Usa test umani per i casi complessi**
  Chiedi a persone reali di valutare risposte su task critici o soggettivi.

PROMPT EVALUATION: BEST PRACTICE DI PROMPT TESTING

- **Documenta i risultati**
  Tieni traccia dei test, delle metriche e delle osservazioni umane per confronti futuri e auditing.
- **Itera e migliora**
  Analizza gli errori, affina i prompt e ripeti il processo per migliorare le performance.

In sintesi:
Un buon prompt testing combina automazione, esempi realistici, valutazione umana e documentazione dei risultati per garantire affidabilità e qualità delle risposte.

EXPLAINABILITY: INTRODUZIONE AL CONCETTO

Ll'explainability nell'AI

L'explainability (spiegabilità) indica la capacità di un modello di AI di rendere comprensibili le sue decisioni e predizioni agli esseri umani.

Perché è importante

Favorisce fiducia, trasparenza e responsabilità nell'uso dei sistemi AI, specialmente in ambiti regolati (finanza, sanità, legale).

Obiettivi dell'explainability

- Comprendere **perché** il modello ha preso una certa decisione
- Individuare errori, bias e potenziali rischi
- Consentire auditing e conformità normativa

Sfide principali

I modelli avanzati (es. deep learning) sono spesso "scatole nere" difficili da interpretare; serve bilanciare performance e comprensibilità.

In sintesi:

L'explainability è la chiave per usare l'AI in modo responsabile, trasparente e conforme alle normative, rendendo chiari i ragionamenti dietro ogni predizione o output.

SHAP: VISUALIZZARE L'IMPORTANZA DELLE FEATURE

Cos'è SHAP

SHAP (SHapley Additive exPlanations) è una tecnica di explainability che assegna un valore a ciascuna feature per indicare quanto ha contribuito alla decisione di un modello.

Principio matematico

Si basa sui valori di Shapley della teoria dei giochi, che quantificano il contributo di ogni "giocatore" (feature) al risultato finale.

Come funziona

Per ogni predizione, SHAP calcola e visualizza il peso positivo o negativo di ogni feature rispetto al risultato.

Output tipico

Diagrammi e grafici (es. barplot o beeswarm plot) che mostrano le feature più rilevanti per una specifica predizione o per l'intero modello.

SHAP: VISUALIZZARE L'IMPORTANZA DELLE FEATURE

Esempio pratico

In un modello che valuta rischi di credito, SHAP può mostrare che "reddito annuo" aumenta la probabilità di approvazione, mentre "precedenti insoluti" la riduce.

Vantaggi

- Offre spiegazioni coerenti, locali (per ogni esempio) e globali (per il modello).
- Compatibile con molti modelli di machine learning.

Conclusione:

SHAP è uno strumento efficace per rendere trasparente l'operato dei modelli AI, facilitando interpretazione e auditing anche in contesti complessi.

LIME: INTERPRETAZIONE LOCALE DI MODELLI BLACK-BOX

Cos’è LIME

LIME (Local Interpretable Model-agnostic Explanations) è un metodo per spiegare le predizioni di qualsiasi modello AI (“black-box”), generando spiegazioni locali semplici.

Come funziona

- Dato un singolo input, LIME crea piccole variazioni dello stesso input (perturbazioni) e osserva come il modello cambia le sue predizioni.
- Poi costruisce un modello interpretabile (es. regressione lineare) che approssima il comportamento del modello originale solo attorno a quell’input.

Risultato

LIME mostra quali feature sono state più influenti per la decisione del modello in quel caso specifico.

LIME: INTERPRETAZIONE LOCALE DI MODELLI BLACK-BOX

Esempio pratico:
Su un testo classificato come “spam”, LIME può evidenziare che la presenza di parole come “gratis” e “vincere” ha influenzato la decisione.

Vantaggi:
- Spiegazioni comprensibili e focalizzate su singole predizioni.
- Utile anche con modelli complessi (black-box) e in vari domini (testi, immagini, tabellari).

Conclusione:
LIME aiuta sviluppatori e utenti a comprendere il “perché” di una predizione, migliorando fiducia e trasparenza nei sistemi di intelligenza artificiale.

CAPTUM: EXPLAINABILITY IN PYTORCH

Cos'è Captum

Captum è una libreria open source sviluppata da Meta/Facebook per aggiungere strumenti di explainability ai modelli PyTorch.
Permette di interpretare come i modelli deep learning prendono le decisioni, analizzando l'importanza delle feature nei dati di input.

Come si usa

1. Si applica Captum al modello PyTorch già addestrato.
2. Si seleziona il metodo di explainability desiderato.
3. Si ottengono score di importanza e, spesso, visualizzazioni per capire "perché" il modello ha scelto un certo output.

Funzioni principali

- Implementa vari metodi di interpretabilità: Integrated Gradients, Saliency, DeepLIFT, GradientSHAP, ecc.
- Supporta classificazione, regressione, NLP e visione artificiale.
- Visualizza graficamente l'impatto di ogni feature sull'output del modello.

CAPTUM: EXPLAINABILITY IN PYTORCH

Vantaggi

- Integrato nativamente con PyTorch.
- Supporta casi d’uso avanzati, inclusi modelli complessi.
- Aiuta a diagnosticare bias, errori e problemi nei dati.

Conclusione:

Captum rende trasparente il comportamento di modelli deep learning in PyTorch, facilitando l’analisi, il debugging e la conformità a standard di explainability richiesti anche dalla normativa europea.

ESEMPI VISUALI DELLE FEATURE CON SHAP/LIME

SHAP
Produce grafici (barplot, beeswarm, waterfall) che mostrano quanto ogni feature contribuisce a una specifica previsione.

Esempio: In una classificazione di SMS, una feature “parole sospette” con alto valore SHAP spingerà la previsione verso “spam”.
Si vedono le feature colorate per capire quali aumentano o diminuiscono la probabilità dell’output.

LIME
Crea spiegazioni locali: per ogni input, evidenzia le feature più rilevanti.

Esempio: Mostra che la presenza di “vincita” e “clicca qui” nel messaggio ha avuto un impatto determinante per classificare lo spam.
Visualizza con grafici a barre l’influenza positiva/negativa di ogni parola nel singolo esempio.

ESEMPI VISUALI DELLE FEATURE CON SHAP/LIME

Utilità pratica

- Questi grafici aiutano a spiegare il comportamento del modello agli utenti non tecnici.
- Consentono di individuare bias, errori, o feature inutili.
- Sono spesso richiesti per audit, compliance e report.

Nota:

Le visualizzazioni possono essere generate automaticamente con poche righe di codice usando le API di SHAP e LIME, integrandole nei notebook di sviluppo o nei report di progetto.

ESERCIZIO: EXPLAINABILITY CON SHAP/LIME

Obiettivo:

Capire quali feature influenzano le decisioni del modello di classificazione (es: classificazione email, fatture, contratti).

Compito

1. Prendi un modello di classificazione già addestrato (ad es. logistic regression o random forest, anche su SMS, NER o dati aziendali).
2. Seleziona 5 record di test (es. 5 email o fatture con etichetta).
3. Applica SHAP o LIME per spiegare la predizione su ogni record:
   - Visualizza quali parole/caratteristiche hanno pesato di più per la decisione.
4. Crea una tabella/plot con i risultati delle spiegazioni per ciascun record.

Record	Predicted	Top Features (SHAP/LIME contribution)
Email 1	Spam	“win” (+), “free” (+), “dear” (-)
Email 2	Not spam	“appointment” (+), “tomorrow” (+), “money” (-)

Conclusione:

Scrivi 1-2 frasi su cosa hai scoperto:

- Le feature più importanti sono coerenti con la logica umana?
- Il modello si basa su segnali corretti o su “bias”?

Legenda:
(+) / (-) indica se la feature contribuisce o allontana il risultato
Nota: Usa parole o anche altre feature, come il numero di cifre, la presenza di link, ecc.

METRICHE DI ROBUSTEZZA E STRESS-TEST

Cos’è la robustezza

La capacità del modello di mantenere prestazioni stabili anche in presenza di rumore, input imprevisti o attacchi adversariali.
Un modello robusto non “collassa” quando incontra dati fuori distribuzione o manipolati.

Metriche tipiche

- **Accuracy su dati disturbati:** Testare il modello su dati modificati (errori di battitura, sinonimi, shuffle delle frasi) e confrontare con la baseline.
- **Adversarial accuracy:** Valutare la performance su input creati per ingannare il modello (adversarial examples).
- **Out-of-Distribution (OOD) detection:** Percentuale di casi in cui il modello riconosce di non sapere rispondere.
- **Stress-test:** Serie di test “al limite” (input molto lunghi, caratteri speciali, ripetizioni) per verificare se il modello regge senza crash o risposte errate.

METRICHE DI ROBUSTEZZA E STRESS-TEST

Esempio pratico

- Test su messaggi SMS con errori ortografici intenzionali o parole insolite.
- Valutazione delle risposte LLM a prompt manipolati per indurre errore o allucinazione.

Utilità

- Obbligatorio per modelli destinati a uso critico (compliance, audit, sicurezza).
- Aiuta a selezionare modelli più affidabili e a progettare sistemi di fallback o validazione automatica.

CHECKLIST PER UNA VALUTAZIONE COMPLETA

1. **Definizione degli obiettivi**
   - Identificare il compito (classificazione, generazione, estrazione, ecc.)
   - Definire le metriche chiave (accuracy, F1, BLEU, ecc.)
2. **Preparazione del dataset**
   - Dati bilanciati e rappresentativi
   - Suddivisione train/val/test
   - Verifica di dati duplicati o leakage
3. **Valutazione quantitativa**
   - Calcolo delle metriche classiche e avanzate
   - Analisi di robustezza (input avversari, dati OOD)
   - Stress-test e edge cases
4. **Valutazione qualitativa**
   - Ispezione manuale dei casi critici
   - Analisi di hallucination e risposte non attese
   - Verifica explainability (SHAP, LIME, Captum)

CHECKLIST PER UNA VALUTAZIONE COMPLETA

5. **Confronto con baseline**
   - Confrontare il modello con soluzioni esistenti o versioni precedenti
6. **Logging & tracciamento**
   - Registrare tutte le metriche e i parametri sperimentali (Weights & Biases o simili)
   - Annotare errori e anomalie rilevate
7. **Documentazione finale**
   - Riassumere risultati e limiti
   - Redigere raccomandazioni per miglioramenti o rilascio
   - Allegare esempi significativi e script di valutazione

Nota:
Questa checklist garantisce una valutazione realistica, trasparente e ripetibile, conforme alle best practice AI e ai requisiti normativi.

CASI D'USO: VALUTAZIONE AVANZATA IN PRODUZIONE

Obiettivo:
Garantire performance, affidabilità e trasparenza dei sistemi AI anche dopo il rilascio, rispondendo sia a esigenze operative che regolatorie.

Monitoraggio continuo delle prestazioni

- Ricalcolo periodico delle metriche su nuovi dati reali.
- Segnalazione automatica di drift e calo di accuratezza.

Alert su output critici

- Notifica immediata in caso di risposte anomale, bias o hallucination rilevate.
- Applicazione di soglie dinamiche per metriche di robustezza.

A/B Testing di modelli

- Valutazione comparativa tra versioni diverse in produzione.
- Analisi dell'impatto reale su utenti e business.

CASI D’USO: VALUTAZIONE AVANZATA IN PRODUZIONE

A/B Testing di modelli

- Valutazione comparativa tra versioni diverse in produzione.
- Analisi dell’impatto reale su utenti e business.

Explainability per compliance

- Visualizzazione delle feature più rilevanti con SHAP/LIME per auditing o revisione.
- Log delle spiegazioni per ogni predizione critica.

Human-in-the-loop

- Routing automatico di casi dubbi o anomali a revisori umani.
- Raccolta di feedback per migliorare costantemente il modello.

Stress-test periodici

- Simulazione di scenari avversi e input “difficili” direttamente in produzione.
- Valutazione della resilienza e della sicurezza operativa.

DOMANDE?

Esercizi

ESERCIZIO: VALUTAZIONE METRICA E FEEDBACK UMANO

Obiettivo:
Rendere capace l'agente RAG (NER + GPT su dati aziendali) che i suoi risultati vengano valutati automaticamente tramite una metrica (es. F1-score, ROUGE, ecc.)
(Opzionale) I risultati possono essere valutati opzionalmente tramite feedback esplicito dell'utente (voto, commento). L'interazione successiva deve considerare questa valutazione per dare la risposta..

Step operativi:

1. **Generazione risposta:**
   - L'agente riceve una domanda (query) e produce una risposta sfruttando i documenti e la knowledge base.
2. **Valutazione automatica:**
   - Per ogni risposta, calcolare una metrica di qualità rispetto alla risposta attesa, ad esempio:
     - F1-score, se è classificazione.
     - ROUGE o BERTScore, se è risposta testuale.
   - Visualizzare e registrare il valore calcolato nella chat.

ESERCIZIO: VALUTAZIONE METRICA E FEEDBACK UMANO

Step operativi:

3. **Feedback utente (opzionale):**
   - L'interfaccia deve permettere all'utente di valutare la risposta (ad es. punteggio 1-5, o "utile/non utile", oppure un commento testuale).
   - Salvare questo feedback insieme ai risultati automatici.
4. **Analisi e miglioramento (opzionale):**
   - Dopo almeno 10 risposte, mostrare statistiche aggregate (es. media dei punteggi automatici e dei feedback umani).
   - Permettere all'utente di rivedere alcune risposte con bassa valutazione, e correggere (ad es. riformulare prompt o migliorare l'agente).

GRAZIE PER L'ATTENZIONE