Documentazione SIAE Academy

Week 1: Foundations & Intro GenAI

Day 1: Welcome & Setup

Mattina: Teoria, Basi di Git e Live Coding

- **Teoria:**
- `SIAE - Intro e Git.pdf`: Slides introduttive sull'accademia e sui concetti base di Git.
- `Git.png`: Immagine esplicativa sui flussi di lavoro di Git.
- `1 - Version Control - Basi.ipynb`: Notebook che introduce i concetti fondamentali del controllo di versione, con un focus su Git, repository, commit e branch.
- `3 - Git in contesto Agile e Scrum.ipynb`: Notebook che illustra come integrare Git nei processi di sviluppo Agile e Scrum, con esempi di strategie di branching.
- `git-cheat-sheet-education.pdf`: Una guida rapida con i comandi Git più comuni.
- **Live Coding:**
- `4 - github_actions_codebase_demo.ipynb`: Notebook con una demo pratica su come impostare una pipeline CI/CD di base con GitHub Actions per un progetto Python.
- `4 - github_actions_codebase_demo_ml.ipynb`: Notebook focalizzato su pipeline CI/CD per progetti di Machine Learning, mostrando l'automazione di training e valutazione.
- `train.py`: Script Python per l'addestramento di un modello di classificazione sul dataset `drug.csv`.
- `fix.sh`: Script di utility che formatta il codice Python utilizzando `black` e `isort`.
- `drug.csv`: Dataset contenente dati su pazienti (età, sesso, pressione sanguigna, etc.) e il farmaco loro assegnato.
- `drug_app.py`: Una semplice applicazione web, probabilmente sviluppata con Streamlit o Flask, che utilizza il modello addestrato per fare previsioni.

Pomeriggio: Esercitazioni

- **Esercizi su Git (`5 - Basi di Git`):**
- `01 - Branching.docx` e `1 - branching.ipynb` (soluzione): L'esercizio guida alla creazione e gestione di branch. Si impara a creare branch separati (`harry`, `snape`) da un punto comune, a fare commit indipendenti su ciascuno e a creare un branch (`lily`) da un altro branch esistente, simulando uno scenario di sviluppo parallelo con una storia a tema Harry Potter.
- `02 - Commit.docx` e `2 - committing.ipynb` (soluzione): Esercizio focalizzato sulla granularità dei commit. Si impara a modificare più file (`yard.txt`, `groceries.txt`), ma a committare le modifiche in modo selettivo (`git add `) per creare commit atomici e significativi, ognuno con un messaggio che descrive una modifica specifica.
- `03 - Git Diff.docx` e `3 - git diff.ipynb` (soluzione): Esercitazione sull'uso di `git diff`. Si esplora come confrontare le differenze tra branch (`git diff 1970s current`), tra commit specifici (`git diff HEAD HEAD~1`), e come ispezionare le modifiche nella working directory (`git diff`) rispetto a quelle nell'area di staging (`git diff --staged`).
- `04 - Github Basics.docx` e `4 - github basics.ipynb` (soluzione): Guida pratica all'interazione con GitHub. L'esercizio copre il ciclo completo: creare un repository locale e uno remoto, collegarli (`git remote add`), fare il push di branch locali (`foods`, `movies`) su GitHub, generare un conflitto di merge e risolverlo localmente prima di pushare il risultato finale.
- `05 - Merging in Git.docx` e `5 - merging.ipynb` (soluzione): Esercitazione dedicata a creare e comprendere i diversi tipi di merge. Si impara a generare intenzionalmente un **fast-forward merge** (nessuna divergenza), un **merge commit senza conflitti** (modifiche diverse su file diversi o righe diverse) e un **merge con conflitti** (modifiche sulla stessa riga), con i passaggi per risolverlo.

• `06 - Stashing.docx` e `6 - stashing.ipynb` (soluzione): Esercizio sull'uso di `git stash`. Simula uno scenario di "emergenza" in cui bisogna nascondere rapidamente delle modifiche non ancora pronte per il commit (`I HATE MY BOSS`) per passare a un altro branch, fare un commit urgente (`I love my boss`) e poi tornare al branch originale per recuperare (`git stash pop`) e finalizzare le modifiche.

• `07 - Undo Changes.docx` e `7 - undoing changes.ipynb` (soluzione): Esercitazione sull'annullamento di modifiche. Si impara a navigare la cronologia (`git checkout `), a scartare modifiche locali non salvate (`git checkout -- `) e a usare `git reset --soft HEAD~2` per rimuovere gli ultimi due commit dal branch corrente mantenendo però le modifiche nel working directory, pronte per essere committate su un nuovo branch.

• **Esercizi su GitHub Actions (`6 - Github actions - DE + Pipeline`):**

• `Esercizio 0 - Data Manipulation.pdf` e `Esercizio 1 - Github actions.pdf`: Questi documenti forniscono il contesto e gli obiettivi per l'esercitazione pratica.

• `6 - Github Actions - DE + Pipeline.ipynb` (soluzione): Questo è il cuore dell'esercitazione. È un notebook estremamente dettagliato che guida alla costruzione di una **pipeline di dati end-to-end** per video di YouTube. I passaggi includono:

• **Estrazione Dati**: Interazione con l'API di YouTube per ottenere ID e metadati dei video, e con `youtube_transcript_api` per scaricare le trascrizioni.

• **Validazione e Pulizia**: Ispezione dei dati per identificare problemi (valori null, caratteri speciali) e applicazione di trasformazioni per pulire e standardizzare i dati (es. `handleSpecialStrings`, `setDatatypes`).

• **Generazione di Embedding**: Utilizzo della libreria `sentence-transformers` per creare embedding semantici dai titoli e dalle trascrizioni, salvando l'output finale in un file Parquet (`video-index.parquet`).

• **Automazione**: La parte finale del notebook spiega in dettaglio come automatizzare l'intera pipeline utilizzando **GitHub Actions**, mostrando come creare un file di workflow `.yml` per eseguire lo script Python in modo schedulato, gestire le chiavi API tramite i secret di GitHub e fare il commit automatico dei dati aggiornati.

• **Dati di Esempio:** Le cartelle `data` e gli script (`data_pipeline.py`, `functions.py`) forniscono tutti i materiali di supporto per eseguire e testare la pipeline.

Day 2: Python Toolkit & Data Exploration

Mattina: Teoria su Python Toolkit e Live Coding

• **Teoria:**

• `SIAE - Python_EDA.pdf`: Slides che introducono le librerie fondamentali di Python per la manipolazione e l'analisi dei dati, come NumPy e Pandas.

• `Dispensa su EDA_Base.pdf` e `Dispensa su EDA_Advanced.pdf`: Materiali di approfondimento sull'Analisi Esplorativa dei Dati (EDA), dalla base alle tecniche più avanzate.

• **Live Coding:**

• `numpy (1).ipynb`: Un notebook che guida passo dopo passo alla scoperta di NumPy, mostrando come creare array, eseguire operazioni matematiche, indicizzazione, slicing e manipolazione della forma degli array.

• `pandas (1).ipynb`: Notebook introduttivo a Pandas, che spiega come creare e manipolare DataFrame, leggere dati da file, ispezionare, filtrare, modificare e aggregare dati.

• `2 - amazon_sales_eda.ipynb`: Un esempio pratico di EDA su un dataset di vendite di Amazon (`amazon.csv`), che applica le tecniche di Pandas per pulire, analizzare e visualizzare i dati.

• `3 - EDA_AIRBNB_Rome.ipynb`: Analisi esplorativa su dati di Airbnb a Roma, focalizzata sulla visualizzazione e sull'interpretazione di dati geospaziali e di mercato.

• `4 - invoices_data_eda_ml.ipynb`: Notebook che combina EDA e Machine Learning su un dataset di fatture (`newest_invoices_data.csv`), mostrando come preparare i dati per un modello predittivo.

Pomeriggio: Esercitazioni

- **Parte 1 (Esercizi Base di Manipolazione e Pulizia):**

• `esercizio_01.ipynb` e `esercizio_01_soluzione.ipynb`: Esercizio introduttivo che guida alla lettura di un file CSV (`happiness_survey_data.csv`) in un DataFrame Pandas, all'ispezione delle dimensioni (`.shape`) e delle statistiche di base (`.describe()`).

• `esercizio_02.ipynb` e `esercizio_02_soluzione.ipynb`: Un'esercitazione completa sulla pulizia dei dati (`Data Cleaning`). Lavorando su un dataset di sondaggi (`Alarm Survey Data.xlsx`), si impara a:

• Convertire colonne da testo a numerico (es. `'5 stars'` -> `5`).

• Identificare e gestire i dati mancanti (`.info()`, `.fillna()`).

• Standardizzare testo inconsistente (es. mappare `'light_activity'` e `'light'` allo stesso valore).

• Rimuovere righe duplicate (`.drop_duplicates()`).

• Identificare e rimuovere outlier estremi visualizzando i dati con istogrammi e boxplot.

• Creare nuove colonne da dati esistenti, sia tramite calcoli aritmetici, logica condizionale (`np.where`), calcoli su date (es. `Delivery Date` - `Purchase Date`) e manipolazione di stringhe (`.str.split()`, `.str.contains()`).

- **Parte 2 (Mini-Progetti di EDA):**

• `project1.ipynb` e `project1_soluzioni.ipynb`: Un progetto di Analisi Esplorativa (EDA) su un dataset di film di Rotten Tomatoes. L'analisi include:

• Filtraggio dei dati per anno (`>= 2010`) e per popolarità (numero di recensioni).

• Calcolo di statistiche aggregate (`.groupby()`) per trovare i film con i voti medi più alti e analizzare le performance per genere (`rating`).

• Creazione di nuove colonne basate su generi specifici (es. 'Animation', 'Comedy') per confrontare le medie dei voti tra diverse categorie di film.

• Visualizzazione delle relazioni tra le variabili con un pairplot di Seaborn.

• `project2.ipynb` e `project2_soluzioni.ipynb`: Un progetto più complesso che simula la preparazione di dati per un modello di churn prediction per un servizio musicale. Include:

• Unione di dati da più fonti (CSV e diversi fogli di un file Excel) tramite `.merge()`.

• Un ciclo completo di pulizia dati (gestione NaN, correzione di typo, standardizzazione di testo).

• Feature Engineering: creazione di colonne significative come `Cancelled` (basata sulla data di cancellazione), e calcolo di metriche aggregate per cliente (es. numero di sessioni di ascolto, percentuale di ascolti per genere).

• L'obiettivo finale è creare un DataFrame pulito e pronto per il modeling, con una riga per cliente e colonne numeriche che riassumono il suo comportamento.

- **Parte 3 (Pre-processing per Machine Learning):**

• `Feature Encoding`, `Feature Scaling`, `Outlier Treatment`: Questi tre esercizi, basati sullo stesso dataset (`UCI_Pediatric_Appendicitis.csv`), guidano attraverso le fasi finali della preparazione dei dati per un modello di classificazione.

• **Feature Encoding (**`feature_encoding_solution.ipynb`**):** Si impara a gestire le variabili categoriche. Viene mostrato come applicare `LabelEncoder` alla variabile target e come usare `OrdinalEncoder` (per variabili con un ordine intrinseco, come i livelli di chetoni nelle urine) e `OneHotEncoder` (tramite `pd.get_dummies`, per variabili senza un ordine specifico) alle feature.

• **Feature Scaling (**`feature_scaling_solution.ipynb`**):** Si affronta il problema delle diverse scale delle variabili numeriche. Dopo aver visualizzato la presenza di outlier con i boxplot, viene applicato il `RobustScaler`, una tecnica di scaling robusta agli outlier.

• **Outlier Treatment (**`outlier_treatment_solution.ipynb`**):** Un approccio più avanzato alla gestione degli outlier. Invece di rimuoverli, gli outlier vengono prima scalati con `RobustScaler`, poi identificati tramite il metodo IQR e convertiti in valori mancanti (NaN). Infine, i valori mancanti risultanti (sia quelli originali che quelli derivati dagli outlier) vengono imputati utilizzando `IterativeImputer`, una tecnica sofisticata che stima i valori mancanti basandosi sulle altre feature.

• **Dati per Esercizi:**

• La cartella `Data` contiene numerosi dataset in formati diversi (.csv, .xlsx, .db, .pkl, .txt) usati per le varie esercitazioni, coprendo scenari come vendite, sondaggi, dati di streaming musicale e altro.

Day 3: Intro LLM & Prompt Engineering

Mattina: Teoria su LLM, Prompt Engineering e Live Coding

• **Teoria:**

• `SIAE - LLM.pdf`: Slides introduttive ai Large Language Models (LLM), che ne spiegano i concetti base, le architetture principali (es. Transformer) e le loro applicazioni.

• **Live Coding:**

• `1 - gemma_ollama.ipynb`: Notebook che guida all'utilizzo di Gemma, un modello di Google, tramite Ollama. Mostra come installare l'ambiente, eseguire il modello per la generazione di testo base, gestire input multimodali (testo e immagini) e creare un semplice chatbot.

• `2 - gemma_concatenazione_di_prompt.ipynb`: Notebook che esplora la tecnica della concatenazione di prompt (prompt chaining) e della generazione iterativa per creare una storia complessa passo dopo passo, guidando il modello attraverso la creazione di un personaggio, una premessa e uno schema.

• `3 - gemma_tecniche_prompting_avanzate.ipynb`: Notebook che approfondisce tecniche di prompting avanzate, come il Few-shot prompting, il Chain of Thought (CoT) e la generazione di risposte in formati strutturati (es. JSON), per migliorare la qualità e l'affidabilità delle risposte del modello.

Pomeriggio: Esercitazioni

• **Esercizi:**

• `Esercizi su Gemma.pdf`: Un documento PDF con una serie di esercizi pratici per applicare le tecniche di prompt engineering apprese, utilizzando il modello Gemma per risolvere problemi specifici.

• **Soluzioni Esercizi:**

• `5 - gemma_inferenza_solo_testo.ipynb`: Questo notebook mostra come utilizzare Gemma per task basati su testo, come la generazione di testo, la risposta a domande e il riassunto, applicando le tecniche di prompt engineering viste al mattino.

• `6 - gemma_inferenza_immagini.ipynb`: La soluzione a questo esercizio guida all'uso delle capacità multimodali di Gemma per analizzare il contenuto di un'immagine. Si impara a passare al modello sia un'immagine che un prompt testuale per ottenere descrizioni o rispondere a domande specifiche sull'immagine.

• `7 - gemma_inferenza_video/`: Questa cartella contiene un esercizio più avanzato. Il notebook `7 - gemma_inferenza_video.ipynb` mostra una pipeline completa per l'analisi di un video:

• Per prima cosa, si utilizzano librerie come `cv2` per estrarre un numero definito di frame a intervalli regolari dal file video.

• Successivamente, questi frame (immagini) vengono passati a Gemma insieme a un prompt per generare un riassunto o descrivere cosa accade nel video, dimostrando come estendere le capacità di analisi di immagini a contenuti video.

• **Dati per Esercizi:**

• La cartella `Data` contiene file multimediali (`image1.jpg`, `video1.mp4`) utilizzati per testare le capacità multimodali dei modelli.

Day 4: RAG Kick-off

Mattina: Teoria su RAG e Live Coding

• **Teoria:**

• `Sistemi RAG_ Gestione di Fonti Errate, Contraddittorie e Mancanza di Risposta.pdf`: Un documento PDF che approfondisce le sfide dei sistemi RAG, come la gestione di informazioni errate o contraddittorie e come far sì che il modello ammetta di non conoscere la risposta.

• Slides e Notebook Teorici: `Slides/SIAE - RAG.pdf` e `Notebook/1 - rag_kick_off_teoria.ipynb` introducono i concetti fondamentali del Retrieval-Augmented Generation (RAG), spiegando l'architettura, i componenti (Retriever, Generator) e il flusso di lavoro.

• **Live Coding:**

• `2 - rag_kick_off_pratica.ipynb`: Un notebook pratico che guida passo dopo passo alla costruzione di una pipeline RAG completa. Utilizzando LangChain, si impara a:

• Caricare documenti di vari formati (PDF, TXT, DOCX, etc.).

• Suddividere i documenti in "chunk".

• Creare embeddings con i modelli di OpenAI.

• Memorizzare i chunk e i loro embeddings in un Vector Store (ChromaDB).

• Creare una catena di RAG (`RetrievalQA`) per interrogare i documenti, con la possibilità di personalizzare i prompt per guidare l'LLM.

• `documenti_per_rag/`: Una cartella contenente un set di documenti di esempio (report, contratti, email, etc.) su cui viene costruita e testata la pipeline RAG durante il live coding.

Pomeriggio: Esercitazioni

• **Esercizi - Demo:**

• `RAG - Live Coding Part 2, Projects Demo and Q&A.mov;`: Una registrazione video di una sessione pratica avanzata, che mostra demo di progetti RAG più complessi e una sessione di domande e risposte.

Week 2: RAG Document Intelligence

Day 1: Embeddings and Vector DB

Mattina: Teoria su Embeddings e Live Coding

- **Teoria:**

• `SIAE - Fondamenti Embeddings e Metriche.pdf`: Slides che introducono i concetti teorici fondamentali degli embeddings, spiegando cosa sono, come vengono generati (es. con modelli come Word2Vec o Transformer) e come si misura la loro qualità tramite diverse metriche.

- **Live Coding:**

• `ricerca_semantica_con_DistilBERT.ipynb`: Un notebook pratico che mostra come implementare un sistema di ricerca semantica. Utilizzando il modello DistilBERT da Hugging Face, si impara a:

• Generare embeddings per un corpus di documenti (in questo caso, un dataset sui personaggi Marvel).

• Creare un'interfaccia utente semplice (con Streamlit, come suggerito dal file `app.py`) per inserire una query.

• Calcolare la similarità del coseno tra l'embedding della query e quelli dei documenti per trovare i risultati più pertinenti.

• `app.py`: Script Python che crea un'applicazione web con Streamlit per mettere in pratica la ricerca semantica sviluppata nel notebook, fornendo un'interfaccia user-friendly.

• `Marvel Datastore.xlsx`: Il dataset di esempio utilizzato nel live coding, contenente informazioni sui personaggi dell'universo Marvel.

Pomeriggio: Esercitazioni

- **Esercizi:**

• `Esercizi su Embeddings.pdf`: Un documento con esercizi pratici per consolidare la comprensione degli embeddings, probabilmente richiedendo di generare embeddings per nuovi testi e calcolare la loro similarità.

- **Soluzioni Esercizi:**

• `0_genera_curricula.ipynb`, `1_classificazione_del_testo.ipynb`, `2_ricerca_semantica.ipynb`: Notebook con le soluzioni degli esercizi, che coprono argomenti come la generazione di dati sintetici (CV), la classificazione di testi basata su embeddings e un'ulteriore pratica sulla ricerca semantica, utilizzando dataset di CV (`resumes.csv`).

Day 2: Retrieval

Mattina: Teoria sul Retrieval e Live Coding

- **Teoria:**

• `SIAE - Retrieval.pdf` e `1 - retrieval_teoria.ipynb`: Materiali che approfondiscono la fase di Retrieval nei sistemi RAG. Vengono spiegate in dettaglio le strategie di indicizzazione (chunking, embedding, vector store) e le tecniche di recupero (ricerca di similarità, ricerca ibrida, re-ranking) per trovare le informazioni più pertinenti.

- **Live Coding:**

• `sistema_rag_langchain.ipynb`: Un notebook che illustra come costruire un sistema RAG completo utilizzando LangChain in un ambiente locale. Mostra come:

• Caricare e avviare un modello LLM (es. Gemma) e un modello di embedding (es. `nomic-embed-text`) tramite Ollama.

• Caricare documenti da fonti web (`WebBaseLoader`).

• Suddividere i documenti in chunk e creare un Vector Store con FAISS.

• Costruire una catena di RAG per interrogare i documenti, mostrando anche come personalizzare i prompt.

• `esempio_rag_llamaindex.ipynb`: Un notebook che offre un'alternativa a LangChain, utilizzando il framework LlamaIndex. In questo esempio si impara a:

• Impostare un modello di embedding da Hugging Face (`BAAI/bge-small-en-v1.5`).

• Caricare documenti locali (in questo caso, articoli in PDF) e indicizzarli in un Vector Store.

• Configurare un `RetrieverQueryEngine` per recuperare i documenti rilevanti.

• Integrare il contesto recuperato con un modello LLM ottimizzato (fine-tuned), come `Mistral-7B-Instruct-v0.2-GPTQ`, per generare risposte.

Pomeriggio: Esercitazioni

• **Esercizi su Retrieval (`Esercizi su Retrieval.pdf`):**

• Il documento PDF contiene gli esercizi teorici e pratici della giornata.

• **Soluzione Esercizi (`4_rag_completa_fino_a_retrieval.ipynb`):**

• Questo notebook è un'esercitazione completa che guida alla costruzione dei primi due stadi di una pipeline RAG (Retrieval-Augmented Generation), concentrandosi sull'**indicizzazione** e sul **retrieval**. Lavorando con un lungo documento PDF (un libro di testo sulla nutrizione), si impara a:

• **1. Ingestione e Chunking:** Caricare il testo dal PDF, formattarlo e suddividerlo in "chunk" (frammenti) di dimensioni gestibili. L'esercizio mostra come usare `spaCy` per dividere il testo in frasi e poi raggrupparle in chunk di dimensione fissa (es. 10 frasi per chunk).

• **2. Creazione degli Embeddings:** Utilizzare un modello di `sentence-transformers` (come `all-mpnet-base-v2`) per trasformare ogni chunk di testo in un embedding vettoriale. Il notebook confronta anche le performance di creazione degli embedding su CPU vs GPU e l'efficienza delle operazioni in batch.

• **3. Implementazione del Retrieval:** Costruire un sistema di ricerca semantica. Data una query dell'utente (es. "quali sono le funzioni dei macronutrienti?"), questa viene trasformata in un embedding e confrontata con gli embedding di tutti i chunk tramite **similarità coseno** (o prodotto scalare) per trovare i frammenti di testo più rilevanti. Il risultato finale è una lista ordinata dei chunk più pertinenti, pronti per essere passati alla fase di generazione.

Day 3: Advanced OCR for RAG

Mattina: Teoria su OCR Avanzato e Live Coding

• **Teoria:**

• `SIAE - OCR_RAG.pdf` e `1 - rag_ocr_avanzata_teoria.ipynb`: Materiali che esplorano i limiti dell'OCR tradizionale e introducono i concetti di Advanced OCR e Intelligent Document Processing (IDP). Vengono discusse tecniche come l'analisi del layout, l'estrazione di tabelle e l'uso di modelli multimodali per estrarre informazioni strutturate da documenti complessi (PDF scansionati, multi-colonna, ecc.).

• **Live Coding:**

• `2 - rag_ocr_avanzata_pratica.ipynb`: Un notebook che mette in pratica i concetti di IDP. Utilizzando la libreria `unstructured`, si impara a:

• Partizionare un PDF complesso in elementi logici (titoli, paragrafi, tabelle).

• Confrontare diverse strategie di estrazione (es. `fast`, `ocr_only`, `hi_res`).

• Creare chunk "intelligenti" basati sugli elementi strutturati estratti, invece che su una semplice suddivisione per numero di caratteri.

• Costruire una pipeline RAG utilizzando questi chunk di alta qualità per migliorare la pertinenza del retrieval e l'accuratezza delle risposte.

• `documenti_ocr_avanzato/`: Cartella contenente i PDF complessi utilizzati come esempio nel live coding.

Pomeriggio: Esercitazioni

- **Esercizi su OCR e RAG Multimodale (`Esercizi su OCR_RAG.pdf`):**
- Il PDF contiene le istruzioni per l'esercitazione pratica.
- **Soluzione Esercizi (`3 - rag_langchain_multimodal.ipynb`):**
- Questo notebook avanzato mostra come costruire una **pipeline RAG multimodale** in grado di comprendere e rispondere a domande su documenti complessi che contengono testo, tabelle e immagini (in questo caso, il paper scientifico "Attention Is All You Need"). I passaggi chiave sono:
- **1. Partizionamento del Documento:** Si utilizza la libreria `unstructured` per analizzare il layout del PDF ed estrarre in modo intelligente i diversi tipi di contenuto: paragrafi di testo, tabelle e immagini.
- **2. Generazione di Riassunti Multimodali:** Per ogni elemento estratto, viene creato un riassunto testuale. Per testo e tabelle si usa un LLM standard, mentre per le immagini si impiega un modello multimodale (`gpt-4o-mini`) che ne descrive il contenuto visivo.
- **3. Indicizzazione con Multi-Vector Retriever:** Questa è la parte cruciale. I riassunti (brevi e densi di informazione) vengono trasformati in embedding e indicizzati in un vector store. I documenti originali (i chunk di testo completi, le tabelle in formato HTML e le immagini in base64) vengono salvati in un document store separato. Il `MultiVectorRetriever` di LangChain fa da ponte, collegando gli embedding dei riassunti ai documenti originali.
- **4. Pipeline RAG Multimodale:** Quando l'utente pone una domanda, il sistema cerca prima tra gli embedding dei riassunti per trovare le informazioni più pertinenti. Successivamente, recupera i documenti originali associati (che possono essere testo, tabelle o immagini) e li inserisce nel contesto di un modello multimodale (`gpt-4o-mini`) per generare una risposta che sfrutta tutte le fonti di informazione disponibili.

Day 4: Conversation in RAG

Mattina: Teoria su RAG Conversazionale e Live Coding

- **Teoria:**
- Slides e Notebook Teorici: `Slides/SIAE - Conversation in RAG.pdf` e `Notebook/1 - rag_conversazionale_teoria_ed_esempi.ipynb` introducono la necessità della memoria nei sistemi RAG e presentano le architetture comuni per gestire la cronologia delle conversazioni e utilizzarla per contestualizzare le nuove domande.
- **Live Coding:**
- `2 - rag_completa.ipynb`: Un notebook che riassume e consolida la creazione di una pipeline RAG completa, dal caricamento dei documenti alla generazione della risposta, fungendo da base per gli esercizi successivi.
- `3 - rag_conversazionale_con_memoria.ipynb`: Introduce la memoria in una pipeline RAG utilizzando componenti sperimentali di Haystack. Mostra come usare `InMemoryChatMessageHistory` per salvare la conversazione e `ChatMessageRetriever` per recuperarla, permettendo al sistema di usare i messaggi precedenti come contesto.
- `4 - rag_avanzata_con_memoria.ipynb`: Un esempio più avanzato con LangChain che mostra come costruire un sistema RAG con memoria per chattare con documenti specifici. Introduce tecniche avanzate di chunking e l'integrazione di `ConversationBufferMemory` per mantenere il contesto della conversazione.

Pomeriggio: Esercitazioni

- **Esercizi su Pipeline RAG Conversazionali (`Esercizi - Pipelines RAG.pdf`):**
- Il PDF contiene le istruzioni per gli esercizi della giornata, che si concentrano sull'aggiungere memoria alle pipeline RAG.
- **Soluzioni Esercizi:**
- `5 - rag_completa_da_generazione_a_fine.ipynb`: Questo notebook funge da consolidamento, mostrando di nuovo l'intera pipeline RAG end-to-end, questa volta includendo esplicitamente la fase di **generazione**. Partendo dal PDF sulla nutrizione, si costruisce un sistema completo che riceve una domanda, recupera il contesto rilevante e utilizza un LLM

locale (`Gemma`) per generare una risposta testuale.

• `6 - conversational_rag/4 - conversational_rag.ipynb`: Questo è l'esercizio chiave della giornata. Si impara a trasformare una pipeline RAG stateless in un **chatbot conversazionale con memoria**. I passaggi principali sono:

• **Contestualizzazione della Domanda:** Si crea una catena specifica (`create_history_aware_retriever`) che prende la nuova domanda dell'utente e la cronologia della chat, e le fonde in una nuova query "autonoma" che ha senso anche senza aver letto la conversazione precedente.

• **Gestione della Memoria:** Si implementa un meccanismo per salvare e recuperare la cronologia della conversazione. L'esercizio mostra come utilizzare `ChatMessageHistory` per memorizzare i messaggi e `RunnableWithMessageHistory` di LangChain per integrare automaticamente la gestione della memoria nella pipeline, permettendo al chatbot di rispondere a domande di follow-up che si riferiscono a messaggi precedenti.

Week 3 - GenAI Advanced - Agents and Tools

Day 1: Agentic Systems

Mattina: Teoria su Sistemi Agentici e Live Coding

• **Teoria:**

• Slides e Notebook Teorici: `Slides/SIAE - Agentic Systems.pdf`, `Notebook/1 - agenti_ai_intro.ipynb` e `Notebook/2 - agenti_langchain_vs_openai_overview.ipynb` introducono il concetto di agenti AI, il ciclo Ragionamento-Azione (Reasoning-Action), i componenti chiave (tool, memoria, pianificazione) e confrontano diversi framework come LangChain e l'API Assistants di OpenAI.

• **Live Coding:**

• `3 - agenti_langchain_quickstart.ipynb`: Un'introduzione rapida a LangChain per costruire agenti. Mostra come definire tool, inizializzare un LLM, usare un prompt dal LangChain Hub e assemblare il tutto in un `AgentExecutor`.

• `4 - agenti_pratica.ipynb`: Un notebook più pratico che guida alla creazione di un agente specifico per la gestione di riparazioni. Dimostra come creare tool custom, integrare tool esistenti come Wikipedia, e gestire conversazioni multi-turno con l'agente.

Pomeriggio: Esercitazioni

• **Esercitazione su API e LangChain (`Esercitazione - API_LangChain.pdf`):**

• Il PDF contiene le istruzioni per gli esercizi pratici.

• **Soluzioni Esercizi:**

• `5 - agente_completo_from_scratch.ipynb`: Questo notebook è una guida completa all'uso dell'API di OpenAI per costruire agenti. L'esercitazione parte dalle basi, mostrando come effettuare semplici chiamate API, per poi passare a funzionalità più avanzate come:

• **Output Strutturato:** Come forzare il modello a restituire risposte in un formato JSON predefinito, utilizzando Pydantic per la validazione.

• **Tool Calling:** Si impara a definire e utilizzare "tool" (funzioni Python) che l'LLM può decidere di chiamare per ottenere informazioni esterne. L'esempio pratico mostra un tool per consultare un'API meteorologica.

• **Retrieval da Knowledge Base:** L'esercizio culmina nella creazione di un tool che funge da sistema di retrieval, permettendo all'agente di cercare risposte all'interno di una knowledge base locale (un file JSON), simulando di fatto un semplice agente RAG.

• `6 - react_langchain.ipynb`: Questo notebook introduce il framework **ReAct (Reasoning and Acting)**, uno dei pattern più importanti per gli agenti, utilizzando LangChain per astrar parte della complessità. L'esercizio guida passo dopo passo alla creazione di un agente ReAct, mostrando come:

• **Definire un Tool:** Si crea un tool custom (una semplice funzione per calcolare la lunghezza di una stringa).

• **Costruire il Prompt ReAct:** Si analizza e si costruisce il prompt template specifico che istruisce l'LLM a seguire il ciclo "Pensiero -> Azione -> Osservazione".

• **Utilizzare Callbacks:** Si implementa un `CallbackHandler` custom per monitorare e stampare ogni interazione tra l'agente e l'LLM, rendendo il processo di ragionamento trasparente e facile da debuggare.

• **Eseguire il Ciclo dell'Agente:** L'agente viene eseguito passo dopo passo per mostrare come riceve una domanda, "pensa" a quale strumento usare, esegue lo strumento e infine formula la risposta finale basandosi sull'osservazione.

Day 2: Prompting for agents

Mattina: Teoria su Prompting per Agenti e Live Coding

- **Teoria:**

• `Slides/SIAE - Prompting for Agents.pdf`: Le slides approfondiscono le tecniche di prompting specifiche per gli agenti, come la strutturazione del system prompt, l'importanza di descrizioni chiare per i tool e l'introduzione al framework ReAct.

- **Live Coding:**

• `1 - prompting_per_agenti_teoria_ed_esempi.ipynb`: Un notebook che esplora in dettaglio l'anatomia di un prompt per agenti ReAct. Vengono analizzati esempi pratici e si discute come la qualità della descrizione di un tool influenzi la capacità dell'agente di usarlo correttamente.

Pomeriggio: Esercitazioni

- **Esercitazione su Prompting Avanzato (`Esercitazione - Advanced Prompting.pdf`):**

• Il PDF contiene le istruzioni per gli esercizi pratici.

- **Soluzioni Esercizi:**

• `2 - agente_summarization (1).ipynb`: Questo esercizio introduce `LangGraph`, una potente estensione di LangChain per creare agenti stateful e ciclici. L'obiettivo è costruire un **agente riassuntore** che elabora una lista di documenti in sequenza. L'agente inizia riassumendo il primo documento, poi entra in un ciclo dove, per ogni documento successivo, "perfeziona" e aggiorna il riassunto esistente con le nuove informazioni, fino a produrre un riassunto complessivo finale.

• `3 - concatenazione_prompting_orchestratori.ipynb`: Questo notebook è una guida pratica a quattro pattern avanzati di orchestrazione di chiamate LLM, utilizzando l'API di OpenAI e Pydantic per l'output strutturato:

• **1. Prompt Chaining:** Si impara a scomporre un compito complesso (es. creare un evento di calendario) in una catena di chiamate LLM sequenziali, dove l'output di un passo diventa l'input del successivo, con "gate checks" per la logica condizionale.

• **2. Routing:** Si utilizza un LLM come "router" per analizzare l'input dell'utente e classificarne l'intento (es. creare un nuovo evento vs. modificare un evento esistente), indirizzando la richiesta all'handler corretto.

• **3. Parallelizzazione:** Si mostra come eseguire chiamate LLM multiple e indipendenti in parallelo utilizzando `asyncio`, una tecnica utile per migliorare la latenza (es. eseguendo contemporaneamente una validazione dei contenuti e un controllo di sicurezza).

• **4. Orchestrazione:** Si implementa un sistema multi-agente per un compito complesso come la scrittura di un blog post. Un LLM "Orchestratore" pianifica la struttura, diversi LLM "Worker" scrivono le singole sezioni in parallelo, e un LLM "Revisore" finale assembla e perfeziona il lavoro.

Day 3: Multi-agent systems

Mattina: Teoria su Sistemi Multi-Agente e Live Coding

- **Teoria:**

• Slides e Notebook Teorici: `Slides/SIAE - Multi-Agent Systems.pdf`, `Notebook/1 - intro_sistemi_multiagente.ipynb` e `Notebook/2 - sistemi_multi_agentici_teoria_ed_esempi.ipynb` introducono i sistemi multi-agente, le loro architetture (cooperative, gerarchiche), i meccanismi di comunicazione e coordinamento, e presentano framework emergenti come LangGraph, AutoGen e CrewAI.

- **Live Coding:**

• `3 - agente_langchain_multiple_tools.ipynb`: Un notebook che mostra come costruire un agente di ricerca avanzato con LangChain, capace di utilizzare più tool (DuckDuckGo, Wikipedia, salvataggio file) e di produrre output strutturati usando Pydantic.

• `4 - crewai_intro.ipynb`: Un'introduzione pratica a **CrewAI** per orchestrare agenti collaborativi. L'esempio mostra come definire una "Crew" di agenti specializzati (ricercatore, analista) e assegnare loro task per produrre un report personalizzato, integrando anche un

RAG per fornire conoscenza specifica.

Pomeriggio: Esercitazioni

• **Esercizi su Sistemi Multi-Agente (`Esercizi - Multi Agente.pdf`):**

• Il PDF contiene le istruzioni per gli esercizi pratici.

• **Soluzioni Esercizi:**

• `5 - code_interpreter.ipynb`: Questo esercizio mostra come costruire un **sistema di agenti gerarchico**, dove un "Grand Agent" funge da **router**. L'agente router riceve la richiesta dell'utente e, basandosi sulla sua natura, la inoltra al sotto-agente più appropriato. I sotto-agenti implementati sono:

• Un **Python Agent**: Utilizza il `PythonREPLTool` di LangChain per scrivere ed eseguire codice Python dinamicamente. L'esercizio mostra come può essere usato per compiti come la generazione di QR code.

• Un **CSV Agent**: Utilizza `create_pandas_dataframe_agent` per specializzarsi nell'analisi di dati tabellari, permettendo all'utente di porre domande in linguaggio naturale su un file CSV.

• `6 - basic_multi_agent_crewai.ipynb`: Questo notebook introduce **CrewAI**, un framework di alto livello per orchestrare agenti collaborativi. L'esercizio guida alla creazione di una "Crew" (squadra) che simula un team di content marketing. Si impara a:

• **Definire Agenti con Ruoli:** Creare diversi agenti, ognuno con un `role` (es. "Ricercatore Online"), un `goal` (obiettivo) e `tools` specifici.

• **Creare Task:** Definire una serie di `Task` (compiti) che devono essere eseguiti.

• **Orchestrare la Crew:** Assemblare agenti e task in una `Crew` e lanciarla. La `Crew` gestisce il flusso di lavoro, passando l'output di un agente come contesto al successivo, fino al completamento del task finale.

Day 4: Ethics and Governance

Mattina: Teoria su Etica e Governance in GenAI e Live Coding

• **Teoria:**

• Slides e Documenti: `Slides/SIAE - Ethics and Governance.pdf`, `Slides/SIAE - Data Governance e Compliance.pdf` e `Notebook/1 - Etica e Governance GenAI.docx` coprono i rischi etici (jailbreak, prompt injection, bias), le normative sulla privacy (GDPR) e l'importanza della governance dei dati.

• **Live Coding:**

• `2 - individuare_jailbreaks_e_prompt_injections.ipynb`: Un notebook che mostra come utilizzare la libreria `LangKit` per rilevare attacchi di tipo jailbreak e prompt injection, confrontando la similarità semantica tra prompt malevoli e l'input dell'utente.

• `3 - prompt_guard.ipynb`: Introduce **Prompt Guard**, un classificatore di Meta per rilevare prompt malevoli. L'esempio mostra come usarlo per identificare attacchi diretti e indiretti.

Pomeriggio: Esercitazioni

• **Soluzioni Esercizi (`4 - notebook_llm_judge.ipynb`):**

• L'esercitazione di questa giornata è un progetto affascinante basato su una competizione Kaggle, che esplora il concetto di **LLM as a Judge** e i bias dei modelli. L'obiettivo non è scrivere il saggio "migliore", ma quello che genera il **massimo disaccordo** tra diversi giudici LLM. Il notebook guida attraverso:

• **1. Strategie di Generazione:** Si definiscono diverse strategie di prompting (filosofico, metaforico, dialettico) per creare deliberatamente testi ambigui, contraddittori o stilisticamente polarizzanti.

• **2. Generazione dei Saggi:** Si utilizza un LLM (`Gemma`) per generare i saggi basati sui topic forniti, applicando casualmente le diverse strategie di prompting.

• **3. Valutazione Simulata:** Poiché l'uso di un panel di LLM reali come giudici è complesso, il processo viene **simulato**. Una funzione genera punteggi casuali per diverse dimensioni (qualità, coerenza, etc.) da parte di un numero fittizio di giudici.

• **4. Calcolo del Punteggio di Disaccordo:** Viene definita una metrica ad-hoc che premia i saggi che ottengono un'alta deviazione standard nei punteggi tra i giudici (alto disaccordo), dimostrando in modo pratico come si potrebbe quantificare questo fenomeno per analizzare i bias dei modelli.

Week 4: GenAI Applications and Rapid Prototyping

Day 1: Prototyping tools

Mattina: Teoria su Strumenti di Prototipazione

- **Teoria:**
- `SIAE - Vibe Coding Tools.pdf`: Slides che introducono strumenti e framework per la prototipazione rapida di applicazioni GenAI, come Streamlit, Gradio e altri tool "vibe/no-code" che accelerano lo sviluppo.
- `Guida Esercizio.pdf` e `Prompt completo per Lovable Dev - FAME (Food AI Meal Engine) - GPT 4.5.gdoc`: Documenti che forniscono una guida pratica e un esempio di prompt molto dettagliato per un esercizio di prototipazione, incentrato sulla creazione di un'applicazione (FAME) utilizzando uno degli strumenti presentati.

Pomeriggio: Esercitazioni

- **Esercitazione su Prototipazione Rapida (Progetto FAME):**
- La giornata è dedicata a un'esercitazione pratica di laboratorio per costruire **FAME (Food AI Meal Engine)**, una web-app full-stack che trasforma le linee guida nutrizionali di un dietista in un piano alimentare settimanale personalizzato.
- **Concept:** L'applicazione, utilizzando **Google Gemini**, genera un piano alimentare, una lista della spesa ottimizzata per stagionalità e numero di persone, e schede pasto dettagliate con ricette, valori nutrizionali e video-tutorial da YouTube.
- **Stack Tecnologico:** Il progetto è stato sviluppato con un backend **FastAPI (Python)** e un frontend **React (TypeScript)**, il tutto containerizzato e deployabile su piattaforme come Render o Replit.
- **Obiettivi Didattici:** Durante il laboratorio di circa 4 ore, i team imparano a progettare un flusso utente completo (autenticazione, onboarding, dashboard), integrare API esterne (Google Gemini, YouTube), sviluppare un'interfaccia mobile-first con strumenti moderni (React Big Calendar, Tailwind) e praticare il "vibe coding" (sviluppo assistito da AI) per accelerare la prototipazione. L'obiettivo finale è consegnare una web-app production-ready.

Day 2: MCP & Agentic RAG

Mattina: Teoria su MCP e RAG Agentico e Live Coding

- **Teoria:**
- `SIAE - MCP & Agentic RAG.pdf`: Slides che introducono due concetti avanzati: il **Model Context Protocol (MCP)**, un protocollo aperto per standardizzare la connessione tra LLM e risorse esterne (dati, tool), e il **RAG Agentico**, un approccio in cui agenti AI collaborano per migliorare il processo di recupero e generazione delle informazioni.
- **Live Coding:**
- `mcp_tutorial.ipynb`: Un notebook che spiega l'architettura di MCP e guida alla costruzione di un server MCP con strumenti personalizzati, mostrando come questo protocollo possa unificare l'accesso a diverse fonti di dati.
- `crewai_agentic_rag.ipynb`: Un esempio pratico molto potente che mostra come costruire un sistema **RAG Agentico** utilizzando **CrewAI**. Viene creata una "crew" di agenti specializzati:
- Un **Router Agent** che decide se cercare informazioni in un documento locale (un PDF sul self-attention) o sul web.
- Un **Retriever Agent** che esegue la ricerca.
- Vari **Grader Agents** che valutano la pertinenza e la presenza di allucinazioni nella risposta.
- Un **Answer Agent** che formula la risposta finale.

Pomeriggio: Esercitazioni

• **Esercitazioni su MCP e CrewAI:** Il pomeriggio è dedicato a due progetti distinti che esplorano due potenti framework per la creazione di applicazioni basate su agenti.

• **Soluzioni Esercitazioni/Progetti Completi:**

• **1. Automazione della Data Science con CrewAI (**`Automating Data Science with CrewAI Agents.ipynb`**):**

• **Obiettivo:** Insegnare agli studenti come automatizzare un intero flusso di lavoro di machine learning (regressione per predire le vendite di integratori) utilizzando un team di agenti AI.

• **Processo:** L'esercitazione inizia con un'analisi manuale del dataset. Successivamente, introduce `CrewAI` e guida lo studente nella creazione di un "crew" composto da tre agenti specializzati:

• Un **Planner Agent** che definisce la strategia e i passaggi del progetto.

• Un **Analyst Agent** che genera ed esegue codice Python per l'analisi esplorativa, la pulizia dei dati e il preprocessing.

• Un **Modeler Agent** che genera ed esegue codice per addestrare un modello di `RandomForestRegressor`, valutarne le performance e riportare le feature più importanti.

• **Componente Chiave:** Viene fornito e utilizzato un tool custom, `NotebookCodeExecutor`, che permette agli agenti di eseguire il codice Python da loro generato direttamente all'interno dell'ambiente del notebook, interagendo con variabili globali come il DataFrame dei dati.

• **2. Costruzione di un AI Tutor con MCP e OpenAI Agents SDK (Client-Server):**

• **Architettura:** Questo progetto insegna a costruire un'applicazione client-server completa utilizzando il **Model-Context-Protocol (MCP)**, uno standard per l'interoperabilità tra modelli AI e strumenti esterni.

• **Componente Server (**`MCP_Server.ipynb`**):**

• Gli studenti costruiscono un server web con **Gradio** che espone quattro strumenti di apprendimento come servizi MCP: un "spiega concetti", un "riassumi testo", un "genera flashcard" e un "interrogami".

• Una caratteristica fondamentale del server è l'uso dello **streaming di token** per fornire risposte in tempo reale e a bassa latenza, migliorando significativamente l'esperienza utente.

• **Componente Client (**`Build an AI Tutor Using MCP and OpenAI Agents SDK.ipynb`**):**

• Gli studenti imparano a costruire un client che interagisce con il server MCP.

• Il client prima **scopre** quali strumenti sono disponibili interrogando il manifest del server (l'endpoint `/mcp/schema`).

• Successivamente, utilizzando l'**SDK degli Agenti di OpenAI**, viene creato un "Assistente Intelligente" capace di comprendere le richieste in linguaggio naturale dell'utente, decidere quale dei quattro strumenti remoti è il più adatto e invocarlo per soddisfare la richiesta.

Day 3: Agents Hackathon (Agenti Cosmici)

Mattina & Pomeriggio: Hackathon

• **Materiali Hackathon:**

• La cartella `hackathon-agenti-cosmici/` contiene tutto il necessario per l'hackathon. L'obiettivo è sviluppare un sistema di agenti AI in grado di gestire missioni in un ambiente simulato a tema spaziale. I partecipanti devono implementare un layer di API Python (`galactic_apis.py`) per interagire con l'universo virtuale e completare task di navigazione, commercio e gestione risorse.

• **Dataset e Ambiente:** I file `galaxy_state.json` e `tasks.csv` (con versioni per i round successivi) definiscono lo stato dell'universo e gli obiettivi delle missioni.

• **Sistema di Valutazione:** Gli script `evaluate_json_missions.py` e `validate_json_format.py` permettono di valutare le soluzioni prodotte dagli agenti, basandosi su correttezza, efficienza (numero di chiamate API) e qualità del codice.

Day 4: RAG Hackathon (Hackapizza)

Mattina & Pomeriggio: Hackathon

- **Materiali Hackathon:**
- La cartella `hackapizza/` contiene i materiali per un hackathon focalizzato sulla creazione di un sistema RAG (Retrieval-Augmented Generation). L'obiettivo è costruire un chatbot in grado di rispondere a 50 domande (`domande.csv`) basate su un vasto corpus di documenti a tema "pizza spaziale".
- **Dataset:** La cartella `Hackapizza Dataset/` contiene un corpus di conoscenza molto variegato, con menu di pizzerie in formato PDF, blog post in HTML, documenti normativi in DOCX e PDF, e dati strutturati in CSV e JSON.
- **Sistema di Valutazione:** Lo script `evaluation.py` (non fornito ma descritto) valuta le performance del sistema RAG confrontando gli ID dei piatti restituiti per ogni domanda con la soluzione corretta, utilizzando la metrica della **Jaccard Similarity**.

Week 5 - MLOps for GenAI

Day 1: API, Payloads, FastAPI and Postman

Mattina: Teoria su API e Live Coding

- **Teoria:**
- `SIAE - MLOps - API, Payloads and more.pdf`: Slides che introducono i concetti fondamentali di MLOps applicati alla GenAI. Vengono trattati argomenti come la creazione di API per modelli AI, la struttura dei payload (richieste/risposte), l'uso di framework come FastAPI per costruire servizi web e di strumenti come Postman per testare gli endpoint.
- **Live Coding:**
- `llmops_fastapi_postman/`: Una cartella contenente un progetto completo che guida alla creazione e al deployment di un'API basata su **FastAPI** che interagisce con l'**API Assistant di OpenAI**. Il notebook `llmops_fastapi_postman.ipynb` mostra due approcci:

1. **Esecuzione Locale:** Utilizzando un ambiente virtuale Python e `uvicorn` per lo sviluppo e il test rapido.

2. **Deployment con Docker:** Creando un `Dockerfile` per containerizzare l'applicazione, rendendola portabile e pronta per la produzione.

- Il progetto include anche script per la configurazione dell'assistente OpenAI e per testare gli endpoint tramite `curl`.

Pomeriggio: Esercitazioni

- **Esercitazione su FastAPI e Postman (`Esercitazione - FastAPI.docx`, `Esercitazione - Postman.ipynb`):**
- I documenti guidano gli studenti in esercizi pratici per consolidare le competenze acquisite.
- **Soluzione Esercizi/Progetto Completo (`llmops_fastapi_postman.ipynb`):**
- Sebbene non sia presente un file di soluzione separato, il notebook del live coding (`llmops_fastapi_postman.ipynb`) funge da progetto completo e guida di riferimento. L'esercitazione pratica consiste nel seguire e replicare i passaggi del live coding, che includono:
- **1. Creazione dell'Assistant OpenAI:** Utilizzare l'SDK di OpenAI per creare un "Assistant", configurarlo con un file di conoscenza (`story.txt`) caricato in un "Vector Store" e abilitare lo strumento di `file_search`.
- **2. Sviluppo Locale con FastAPI:** Creare un'applicazione FastAPI (`main.py`) con un endpoint `/response` che:
- Riceve una domanda dall'utente.
- Crea un "thread" di conversazione con l'assistente.
- Esegue il `run` e attende il completamento.
- Estrae e restituisce la risposta generata dall'assistente.
- **3. Test Locale:** Avviare il server localmente con `uvicorn` e testare gli endpoint utilizzando strumenti come `curl` o Postman.
- **4. Deployment con Docker:** Creare un `Dockerfile` per containerizzare l'applicazione FastAPI, buildare l'immagine Docker ed eseguire il container, esponendo l'API su una porta specifica.

Day 2: Testing

Mattina: Teoria su Testing in MLOps e Live Coding

- **Teoria:**

• `SIAE - Testing in MLOps.pdf`: Slides che illustrano l'importanza del testing nel ciclo di vita di MLOps, coprendo diversi livelli di test: unit test, test di integrazione, test di validazione dei dati e dei modelli, e test end-to-end.

• **Live Coding:**

• `MLOps_Testing (1).ipynb`: Un notebook didattico completo che esplora le principali tipologie di test in MLOps. Attraverso la creazione di dataset sintetici con anomalie intenzionali, il notebook dimostra come implementare:

• **Data Testing:** Controlli su schema, qualità (valori nulli, duplicati), plausibilità statistica e simulazione di data drift.

• **Model Testing:** Test su performance (accuracy, precision, recall), robustezza (stress test con rumore e valori mancanti), stabilità (confronto tra run con seed diversi), fairness tra gruppi demografici e performance su sottopopolazioni specifiche (slicing).

• `mlops_pytest/mlops_pytest.ipynb`: Un notebook che simula un ciclo completo di debugging e testing per un progetto che integra un modello di Machine Learning. Partendo da una codebase con diversi test `pytest` falliti, l'esercitazione guida attraverso l'identificazione e la correzione di errori logici, problemi di configurazione (es. porte errate) e la gestione di dipendenze e moduli esterni.

• `Grafana.ipynb`: Una guida pratica al **testing continuo in produzione** tramite monitoraggio. Il notebook mostra come:

• Servire un modello di fraud detection tramite un'API **Flask**.

• Strumentare l'API con **Prometheus** per esporre metriche custom (es. conteggio predizioni, latenza).

• Configurare un ambiente Docker con **Grafana** per connettersi a Prometheus e creare una dashboard di monitoraggio in tempo reale per visualizzare il throughput, la latenza e la distribuzione delle predizioni, interpretando questi grafici come indicatori di data drift e salute del modello.

Pomeriggio: Esercitazioni

• **Esercitazioni su Testing (**`istruzioni_mlops_testing_exercise1.ipynb`**,** `istruzioni_mlops_testing_exercise2.docx`**):**

• I documenti guidano gli studenti in due esercitazioni pratiche complete sulla creazione e il testing di pipeline MLOps.

• **Soluzioni Esercitazioni/Progetti Completi:**

• **1. Testing di una Pipeline di Classificazione (Titanic) (**`mlops_testing_exercise1.ipynb`**):** Questo progetto guida passo dopo passo alla costruzione di una pipeline MLOps per il classico dataset del Titanic. L'enfasi è sulla scrittura di una suite di test robusta con `pytest`. I punti chiave includono:

• **Struttura del Progetto:** Creazione di una directory strutturata (`src`, `tests`, `data`).

• **Test di Validazione Dati:** Scrittura di test (`test_data_schema`) che verificano la presenza delle colonne necessarie nei dati di input.

• **Unit Test:** Test per le funzioni di preprocessing che assicurano la corretta gestione dei valori nulli e la trasformazione numerica delle feature.

• **Test di Validazione del Modello (Quality Gate):** Implementazione di un test cruciale (`test_model_performance_threshold`) che verifica che l'accuratezza del modello addestrato superi una soglia minima (es. 75%), agendo come un "cancello di qualità" per prevenire il deploy di modelli poco performanti.

• **2. Testing di un Progetto di Churn Prediction (**`mlops_testing_exercise2.ipynb`**):** Questo secondo progetto, più articolato, simula un ciclo di sviluppo MLOps per un modello di previsione del churn. L'esercitazione approfondisce ulteriormente i diversi livelli di testing:

• **Test di Integrità dei Dati:** Verifica che i dati grezzi rispettino vincoli fondamentali (es. il target è binario, le colonne chiave non sono vuote).

• **Unit Test per Feature Engineering:** Test isolati sulle funzioni di trasformazione dei dati.

• **Integration Test:** Un test che esegue l'intera pipeline di training (`run_training()`) per assicurarsi che tutti i componenti (caricamento dati, preprocessing, training, salvataggio modello) funzionino correttamente insieme.

• **Test Funzionali e di Qualità:** Test sullo script di predizione che verificano non solo che le performance del modello superino la soglia minima (`MINIMUM_ACCURACY`), ma anche che l'output della predizione abbia il formato corretto (es. un array di interi binari).

• **Simulazione di Bug:** La parte finale del notebook mostra il valore pratico dei test introducendo deliberatamente un errore nel codice e dimostrando come `pytest` lo catturi immediatamente, impedendo una "regressione" del software.

Day 3: Dockerizzazione

Mattina: Teoria su Dockerizzazione e Live Coding

• **Teoria:**

• `SIAE - MLOps - Dockerizzazione.pdf`: Slides che introducono Docker come strumento fondamentale per l'MLOps. Vengono spiegati i concetti di immagine e container, i vantaggi della containerizzazione (portabilità, riproducibilità, isolamento) e la sintassi di base di un `Dockerfile`.

• **Live Coding:**

• `mlops_docker/`: Una cartella che contiene un progetto completo sulla creazione di una **pipeline di training di Machine Learning riproducibile con Docker**. Il notebook `mlops_docker.ipynb` guida attraverso i seguenti passaggi:

1. **Strutturazione del Progetto:** Suddivisione del codice in moduli Python per la validazione dei dati (`data_validation.py`), il training del modello (`model_training.py`) e l'orchestrazione (`main.py`).

2. **Scrittura del `Dockerfile`:** Definizione delle istruzioni per costruire un'immagine Docker che includa il codice sorgente e le dipendenze Python.

3. **Build & Run:** Comandi per costruire l'immagine (`docker build`) e avviare un container (`docker run`) che esegue l'intera pipeline.

4. **Gestione degli Artefatti:** Utilizzo dei **volumi Docker** (`-v`) per salvare l'output della pipeline (il modello addestrato) dalla macchina host al container, dimostrando come recuperare gli artefatti generati.

Pomeriggio: Esercitazioni

• **Esercitazione su Docker (`Esercitazione - Docker.ipynb`, `Istruzioni - Dockerizzazione.ipynb`):**

• I notebook forniscono esercizi pratici sulla scrittura di Dockerfile e sulla gestione di container per applicazioni ML.

• **Soluzione Esercizi/Progetto Completo (`mlops_docker.ipynb`):**

• Il notebook del live coding è un progetto end-to-end che guida alla **containerizzazione di una pipeline di training di Machine Learning**. L'obiettivo è creare un ambiente riproducibile con Docker. L'esercitazione copre:

• **1. Strutturazione del Progetto:** Organizzare il codice in moduli Python separati (`data_validation.py`, `model_training.py`, `main.py`) per creare una pipeline pulita e testabile.

• **2. Scrittura del `Dockerfile`:** Creare un `Dockerfile` passo dopo passo, spiegando ogni direttiva (`FROM`, `WORKDIR`, `COPY`, `RUN`, `CMD`) e le best practice, come sfruttare la cache di Docker per le dipendenze.

• **3. Build dell'Immagine:** Utilizzare il comando `docker build` per creare un'immagine Docker portabile che contiene l'applicazione e tutte le sue dipendenze.

• **4. Esecuzione del Container con Volumi:** Eseguire la pipeline di training all'interno di un container con `docker run`. Viene spiegato in dettaglio l'uso dei **volumi** (`-v`) per mappare una cartella locale (`output`) a una interna al container, permettendo di salvare gli artefatti prodotti (come il modello addestrato) sulla macchina host.

Day 4: AWS, SageMaker, Lambdas

Mattina: Teoria su Cloud MLOps e Live Coding

- **Teoria:**
- `SIAE - MLOps - AWS, Sagemaker, Lambdas.pdf`: Slides che introducono i servizi cloud di AWS per l'MLOps, con un focus su Amazon SageMaker per il training e il deploy di modelli, e AWS Lambda per l'esecuzione di codice serverless.
- **Live Coding:**
- `llmops_sagemaker_test.ipynb`: Un notebook che mostra come fare il deploy di un modello da Hugging Face su un **endpoint SageMaker**. Include i passaggi per configurare il ruolo IAM, creare l'endpoint e inviare richieste per fare inferenza.
- `aws_sagemaker_lambda/`: Una cartella con un progetto più avanzato che illustra una pipeline MLOps completa:

1. **Training su SageMaker:** Un notebook (`training.ipynb`) che esegue il training di un modello su SageMaker, salvando l'artefatto su S3.

2. **Containerizzazione dell'Inferenza:** Un `Dockerfile` per creare un'immagine Docker che carica il modello e serve le predizioni.

3. **Deploy Serverless:** L'immagine viene caricata su ECR (Elastic Container Registry) e utilizzata da una **funzione AWS Lambda** per creare un endpoint di inferenza serverless, invocabile tramite API Gateway.

Pomeriggio: Esercitazioni

- **Esercitazione su AWS (`istruzioni_aws_sagemaker_lambda_exercise.ipynb`):**
- Il notebook contiene le istruzioni per l'esercitazione pratica.
- **Soluzioni Esercizi/Progetti Completi:**
- **1. Deployment Rapido con SageMaker (`llmops_sagemaker_test.ipynb`):** Questa esercitazione è un'introduzione rapida e diretta al deployment su Amazon SageMaker. Si impara a:
- Prendere un modello pre-addestrato direttamente da Hugging Face.
- Utilizzare la classe `HuggingFaceModel` di SageMaker per configurarlo.
- Eseguire il deploy con un solo comando (`.deploy()`) per creare un endpoint HTTPS gestito e pronto per l'inferenza.
- **2. Pipeline MLOps Completa con SageMaker e Lambda (`aws_sagemaker_lambda.ipynb`):** Questo è un progetto più completo che guida alla costruzione di una pipeline MLOps end-to-end su AWS. I passaggi includono:
- **Training su SageMaker:** Caricare un dataset su S3, scrivere uno script di training per un `RandomForestClassifier` e lanciare un job di training gestito su SageMaker.
- **Containerizzazione per Lambda:** Creare un'applicazione Python che può sia simulare le predizioni (versione "mock" per test) sia invocare l'endpoint SageMaker (versione "produzione").
- **Creazione del `Dockerfile`:** Scrivere un `Dockerfile` per pacchettizzare l'applicazione in un'immagine container.
- **Test Locale e Deployment Serverless:** Testare il container Docker localmente, fare il push dell'immagine su ECR (Elastic Container Registry) e infine deployare una funzione AWS Lambda basata su quell'immagine, creando un endpoint di inferenza serverless.

Week 6: Anomaly Detection and Optimisation

Day 1: Anomaly Detection: Basi - Structured data - Text data

Mattina: Teoria su Anomaly Detection e Live Coding

- **Teoria:**
- `Anomaly_Detection_Dispensa.pdf` e `Anomaly Detection: Parte 1.pdf`: Materiali teorici che introducono i concetti fondamentali dell'Anomaly Detection. Vengono classificati i metodi (supervisionati, non supervisionati, semi-supervisionati) e presentate le principali tecniche per dati strutturati e testuali.
- **Live Coding:**
- `AD_Strutturati_Semistrutturati.ipynb`: Un notebook completo che implementa e confronta un'ampia gamma di algoritmi di anomaly detection su un dataset reale di network intrusion (NSL-KDD). Vengono testati:
- **Metodi Statistici (Non Supervisionati):** Z-score, IQR, Elliptic Envelope.
- **Metodi di Clustering (Non Supervisionati):** K-Means, DBSCAN, Local Outlier Factor (LOF).
- **Metodi di Isolamento (Non/Semi-Supervisionati):** Isolation Forest, One-Class SVM.
- **Metodi Supervisionati:** Un confronto tra classificatori classici (Random Forest, SVM, etc.) e metodi ensemble (Bagging, Boosting, Stacking), mostrando la loro superiorità quando sono disponibili dati etichettati.
- Il notebook include anche una sezione su come estrarre feature da dati **semistrutturati** (JSON, Log files) per applicare gli stessi algoritmi.
- `AD_Text.ipynb` e `AD_Text_ISOT.ipynb`: Notebook focalizzati sull'anomaly detection in **dati testuali**. Viene affrontato il problema come un task di classificazione supervisionata (es. rilevamento di fake news dal dataset ISOT). La pipeline include:
- Preprocessing del testo con `nltk`.
- Feature engineering con **TF-IDF**.
- Gestione dello sbilanciamento delle classi.
- Training e valutazione di vari modelli (es. Logistic Regression), con analisi dell'interpretabilità (feature importance).
- Confronto con metodi non supervisionati per evidenziarne i limiti su questo tipo di dati.
- `AD_Documents_Images.ipynb`: Un notebook che estende l'anomaly detection a dati ancora più complessi come **documenti e immagini**, probabilmente introducendo tecniche basate su deep learning o embedding multimodali.

Pomeriggio: Esercitazioni

- **Esercizi:**
- `AD_Strutturati_Spotify.pdf` e `AD_Testuali_Magic.pdf`: Esercizi pratici per applicare le tecniche di anomaly detection su dataset reali a tema (dati di Spotify, testi di carte Magic).
- **Soluzioni Esercizi:**
- `spotify_anomaly_detection.ipynb` e `AD_Text_Magic.ipynb`: Notebook con le soluzioni degli esercizi.

Day 2: Anomaly Detection: Unstructured data

Mattina: Teoria su Anomaly Detection in Dati Non Strutturati e Live Coding

- **Teoria:**

• `Anomaly_Detection.pdf` e `AD_Parte 2.pdf`: Materiali teorici che estendono i concetti di anomaly detection a dati non strutturati, come serie temporali, dati geospaziali, documenti e immagini.

• **Live Coding:**

• `AD_Dati_finanziari.ipynb` e `AD_Fin.ipynb`: Notebook molto dettagliati che si focalizzano sull'**anomaly detection in serie temporali finanziarie** (dati azionari di AAPL scaricati con `yfinance`). Viene implementata una pipeline completa che include:

• **Feature Engineering Finanziario:** Calcolo di returns, volatilità, volume ratio, RSI, Bollinger Bands, etc.

• **Metodi Statistici e per Serie Temporali:** Applicazione di Z-score, **ARIMA** (con gestione avanzata dei warning e calibrazione del threshold) e **Prophet** per identificare anomalie basate su deviazioni dalle previsioni.

• **Metodi di Machine Learning (Non Supervisionati):** Uso di **Isolation Forest**, **LOF** e **One-Class SVM** sulle feature ingegnerizzate.

• **Analisi Consensus:** Viene creato un "consensus score" che combina i risultati di tutti i metodi per identificare le anomalie più robuste, con un'analisi dettagliata degli eventi di mercato reali corrispondenti.

• `AD_Geospatial.ipynb`, `AD_Geo_2.ipynb`, `AD_Geo_F1.ipynb`: Notebook che (dai nomi) affrontano l'anomaly detection su **dati geospaziali**, probabilmente analizzando traiettorie, densità di punti o eventi localizzati in mappe (es. dati di telemetria della F1).

• `AD_Documents_Images.ipynb`: Un notebook che, come nel giorno precedente, si occupa di dati complessi come documenti e immagini, probabilmente con tecniche più avanzate.

Pomeriggio: Esercitazioni

• **Esercizi su Dati Non Strutturati (`AD_DocImage_Pokemon.pdf`, `AD_Geospatial_F1.pdf`):**

• I PDF contengono le istruzioni per gli esercizi.

• **Soluzioni Esercizi:**

• `AD_DocImage_Pokemon.ipynb`: Questa esercitazione si concentra sull'identificazione di anomalie in un dataset di **immagini** (Pokémon). Le tecniche utilizzate probabilmente includono l'estrazione di feature dalle immagini (tramite metodi classici o reti neurali come gli Autoencoder) e l'applicazione di algoritmi di anomaly detection su queste feature per trovare immagini "strane" o atipiche (es. Pokémon con colorazioni anomale, forme strane, etc.).

• `F1_Geospatial.ipynb`: Un progetto completo e avanzato che analizza **dati geospaziali** di telemetria della Formula 1. I passaggi chiave sono:

• **Caricamento Dati Robusto:** Il notebook implementa un sistema di fallback che prova a caricare i dati da diverse fonti (API OpenF1, GitHub, Kaggle) e, in caso di fallimento, genera un dataset sintetico realistico.

• **EDA Geospaziale:** Creazione di visualizzazioni interattive per analizzare la distribuzione geografica dei circuiti e le performance dei piloti.

• **Metodi Multipli:** Vengono implementati e confrontati più di 10 algoritmi di anomaly detection, tra cui metodi statistici, di clustering, un **Autoencoder** con TensorFlow e, soprattutto, metodi specifici per dati **geospaziali** come l'analisi della densità spaziale e l'analisi delle traiettorie.

• **Ground Truth e Valutazione:** Viene creato un ground truth basato su regole di dominio (es. "velocità troppo bassa per un certo tratto di pista") per valutare e confrontare le performance di tutti i modelli.

• **Altri Esercizi (Unsupervised Data):** La cartella `Altri Esercizi (Unsupervised Data)/` contiene ulteriori notebook che esplorano l'anomaly detection non supervisionata su altri tipi di dati, come serie temporali del traffico di taxi a NYC e dati di Netflix.

Day 3 & 4: Anomaly Detection Hackathon

Mattina & Pomeriggio: Hackathon

• **Obiettivo:** Sviluppare algoritmi di Machine Learning per rilevare anomalie complesse e sottili nei dati SIAE, con l'obiettivo di massimizzare l'**F1-Score** su un test set nascosto.

• **Struttura:** L'hackathon è suddiviso in 4 track indipendenti, ciascuna con un dataset specifico e diverse tipologie di anomalie da individuare. I partecipanti devono generare i dataset tramite uno script (`generate_datasets.py`), sviluppare un modello per una o più track e sottomettere le predizioni in un formato JSON standardizzato per una valutazione automatica.

• **Track dell'Hackathon:**

• **Track 1: Live Events Anomaly Detection:**

• **Dati:** Eventi musicali live (concerti, festival).

• **Anomalie da Rilevare:** Combinazioni improbabili (es. grandi artisti con ricavi irrisori), comportamenti anomali di una venue, cluster multivariati sospetti, frodi sottili sui ricavi e pattern temporali/geografici impossibili per i tour degli artisti.

• **Track 2: Document Fraud Detection:**

• **Dati:** Metadati estratti da documenti ufficiali SIAE.

• **Anomalie da Rilevare:** Incoerenze tra età dichiarata e qualità digitale del documento, frodi basate su template condivisi, inconsistenza tra feature tecniche (es. alta nitidezza ma bassa confidenza OCR) e manipolazioni minime dei metadati.

• **Track 3: Music Anomaly Detection:**

• **Dati:** Tracce musicali con metadati e feature audio.

• **Anomalie da Rilevare:** Attività di streaming da bot con pattern geografici innaturali, "dirottamento" di account di artisti storicamente inattivi e frodi sulla qualità audio (file di bassa qualità "gonfiati" a bit rate superiori).

• **Track 4: Copyright Infringement Detection:**

• **Dati:** Opere creative e loro utilizzo.

• **Anomalie da Rilevare:** Evasione del Content ID tramite lievi modifiche audio (velocità, intonazione), "sleeper infringement" (opere tenute dormienti e poi promosse con picchi di attività improvvisi) e mascheramento di tracce protette con rumore di fondo.

Week 7: Capstone Project

Day 1: Scrum, Agile and Kanban

Mattina: Teoria su Metodologie Agili

- **Teoria:**
- `Metodologie Agile - Scrum e Kanban.pdf`: Slides e documenti che introducono le metodologie di gestione progetti fondamentali per il Capstone Project. Vengono trattati i principi del manifesto Agile, e i framework Scrum e Kanban, spiegando ruoli (Product Owner, Scrum Master), eventi (Sprint, Daily Stand-up) e artefatti (Product Backlog, Sprint Backlog). Viene inoltre presentato ufficialmente il Capstone Project, delineandone obiettivi, timeline e modalità di lavoro.

Pomeriggio: Esercitazioni

- La giornata è dedicata alla formazione dei team, alla comprensione approfondita del brief di progetto e all'impostazione degli strumenti di lavoro (es. board Kanban/Scrum su piattaforme come Trello, Jira o Notion) per iniziare a pianificare le prime attività del Capstone Project.

Day 2: Advanced Topics - Preference Datasets, Reward Models, DPO/RLHF

Mattina: Teoria su Tecniche di Allineamento Avanzate

- **Teoria:**
- `Preference Datasets, DPO e RLHF - Allineamento di LLM con Feedback Umano.pdf`: Slides che introducono tecniche avanzate per l'allineamento dei Large Language Models (LLM) con le preferenze umane. Vengono spiegati i concetti di:
- **Preference Datasets:** Come vengono creati i dataset di preferenze (es. coppie di risposte dove una è migliore dell'altra).
- **Reward Models:** Come si addestra un modello (il "Reward Model") per predire quale risposta un umano preferirebbe.
- **Reinforcement Learning from Human Feedback (RLHF):** Il processo completo che utilizza il Reward Model per "guidare" un LLM tramite reinforcement learning, ottimizzandolo per generare output più allineati.
- **Direct Preference Optimization (DPO):** Un'alternativa più recente e diretta a RLHF, che ottimizza il modello direttamente sui dati di preferenza senza la necessità di addestrare un Reward Model separato.
- **Live Coding:**
- `1 - dpo-from-scratch.ipynb`: Un notebook che implementa l'algoritmo **DPO (Direct Preference Optimization) da zero**. Partendo da un dataset di preferenze (con risposte "scelte" e "rifiutate"), il codice guida attraverso:
- La creazione di una classe `PreferenceDataset` e di una funzione di `collate` personalizzata per gestire le coppie di risposte.
- L'implementazione della **funzione di loss DPO**, spiegando matematicamente e in codice come questa ottimizzi direttamente il modello per massimizzare la probabilità delle risposte preferite rispetto a quelle rifiutate, senza bisogno di un reward model intermedio.
- Il training di un modello "policy" (da ottimizzare) contro un modello "reference" (fisso), mostrando come monitorare la loss e i "reward margins" per analizzare l'andamento dell'allineamento.
- `2_train_reward_model_rlhf.ipynb`: Un notebook che mostra la prima fase del classico processo **RLHF**: l'addestramento di un **Reward Model**. Utilizzando la libreria `trl` e la piattaforma `Argilla` per la raccolta di feedback, il notebook illustra come:

• Configurare un dataset in Argilla per raccogliere preferenze umane (es. "scegli la risposta migliore tra la A e la B").

• Estrarre i dati di preferenza etichettati (risposte scelte vs. rifiutate).

• Utilizzare il `RewardTrainer` di `trl` per addestrare un modello di classificazione (`distilroberta-base` in questo caso) che impara a predire quale risposta riceverebbe un "reward" più alto, simulando il giudizio umano.

Pomeriggio: Lavoro sul Capstone Project

• I team applicano le conoscenze acquisite per definire l'approccio tecnico del loro progetto, valutando se e come integrare tecniche di allineamento come DPO o RLHF, e continuando lo sviluppo del loro MVP.

Day 3: Testing & Hallucination

Mattina: Teoria su Testing Avanzato per LLM

• **Teoria:**

• `Hallucination & Toxicity tests.pdf`: Slides che si concentrano su due dei problemi più critici degli LLM: le **allucinazioni** (generazione di informazioni false o non fattuali) e la **tossicità** (generazione di contenuti offensivi o dannosi). Vengono presentate le strategie per testare e mitigare questi problemi.

• **Live Coding:**

• `deepeval_quantized_llm.ipynb`: Un notebook che mostra come utilizzare il framework **DeepEval** per effettuare test rigorosi su un LLM, anche in versione quantizzata (a 4-bit, per efficienza). L'esempio guida alla creazione di un test suite che valuta:

• **Hallucination:** Verifica che la risposta del modello sia fattualmente coerente con un contesto fornito.

• **Bias e Tossicità:** Controlla la presenza di bias e contenuti inappropriati nelle risposte.

• **LLM as a Judge:** Utilizza un LLM esterno (come GPT-4) come "giudice" per valutare la qualità complessiva delle risposte del modello testato.

• `llm_judge.ipynb`: Un notebook che approfondisce l'approccio **"LLM as a Judge"**. Mostra come impostare un sistema in cui un LLM valuta le risposte di un altro LLM basandosi su una serie di criteri predefiniti (una "rubric"), fornendo un punteggio e una giustificazione. Questo è utile per automatizzare la valutazione su larga scala.

Pomeriggio: Lavoro sul Capstone Project

• I team si dedicano all'implementazione dei test per il loro progetto. Utilizzando strumenti come DeepEval e l'approccio "LLM as a Judge", costruiscono pipeline di valutazione per misurare le performance, l'affidabilità e la sicurezza della loro soluzione GenAI.

Day 4: Polish and Present POCs

Mattina: Teoria e Preparazione

• **Teoria:**

• `Presentare POC e Progetti agli Stakeholder - Guide e Best Practice.pdf`: Un documento guida che fornisce consigli pratici e best practice su come presentare efficacemente un Proof of Concept (POC) e un progetto tecnico a un pubblico di stakeholder non tecnici. Copre argomenti come la strutturazione della presentazione, lo storytelling, la gestione delle domande e la focalizzazione sui benefici di business.

• I team utilizzano la mattinata per preparare le loro presentazioni, seguendo le linee guida fornite, e per fare le ultime rifiniture tecniche ai loro POC.

Pomeriggio: Presentazione dei POC e Consegna Finale

• I team presentano a turno i loro Capstone Project. Ogni presentazione include una demo del POC, una spiegazione dell'architettura e una discussione sui risultati e sui possibili sviluppi

futuri.

• La giornata si conclude con la consegna finale dei progetti, della documentazione e del codice sorgente.