Prova Tecnica: Generazione Sintetica e Classificazione Multiclasse delle Email

Francesco Sannicola

29 settembre 2024

1 Introduzione

Il presente documento ha l'obiettivo di descrivere l'intero ciclo di vita del progetto, partendo da un'analisi iniziale dei requisiti. Successivamente, verranno trattate le fasi di progettazione, sviluppo e testing. Sarà effettuata un'analisi sui dati generati e, infine, verrà illustrata la strategia adottata per il deployment.

2 Analisi dei Requisiti

L'obiettivo principale della prova consiste nella creazione e classificazione di un dataset sintetico, fornendo così una valutazione delle competenze del candidato nell'implemetazione di un classificarore multi-classe a partire da dati generati artificialmente. Di seguito sono riportati i requisiti chiave del progetto.

2.1 Requisiti Funzionali

1. Creazione del Dataset Sintetico:
    - Il candidato deve generare un dataset con almeno 1000 esempi di testo.
    - Il dataset deve contenere almeno 3 classi/label/categorie distinte.
    - Le classi devono essere rappresentative di un dominio specifico a scelta del candidato, come ad esempio:
        - Articoli di giornale
        - Post su forum
        - Post sui social media
        - Recensioni
        - Email
    - È consentito l'uso di modelli pre-addestrati e librerie di supporto.
2. Tecnica di Classificazione:
    - Sviluppare una tecnica automatizzata per classificare gli esempi di testo.
    - La classificazione deve avvenire in base alla classe di appartenenza, implicando che si tratti di un problema di classificazione multi-classe.
    - È accettabile utilizzare tecniche di apprendimento supervisionato, con l'assunzione che il numero di classi sia noto a priori.

3. (Opzionale) Progettazione End-to-End:
    - Fornire un approccio completo che copra l'intero ciclo di vita del progetto, dall'analisi dei requisiti alla gestione del servizio.
    - Indicare le risorse hardware necessarie per lo sviluppo e il deploy.
    - Consegnare il progetto in un ambiente dockerizzato.

2.2 Requisiti Non Funzionali

1. Qualità del Dataset:
    - I dati sintetici devono essere rappresentativi delle classi scelte, garantendo variabilità e realisticità.
2. Efficacia e Comprensibilità del Codice:
    - Il codice deve essere efficiente e scalabile.
    - Deve essere leggibile e di facile comprensione, con commenti che chiariscano il funzionamento delle parti principali.

2.3 Materiale da Consegnare

Il candidato deve presentare il seguente materiale:

- Una descrizione dettagliata del dataset, comprendente:
    - Formato del testo
    - Lunghezza del testo
    - Distribuzione delle classi
- Il codice utilizzato per generare il dataset.
- Il codice per l'implementazione del modello scelto, con focus sulle inferenze.
- Una relazione breve sulle scelte implementative e sui risultati ottenuti.

3 Progettazione

Il progetto prevede la creazione di un dataset sintetico di email utilizzando un Large Language Model tramite la libreria `Langchain`. Le email generate vengono successivamente sottoposte a un processo di preprocessing e utilizzate per la sperimentazione di diversi algoritmi di machine learning, come la regressione logistica, SVM, random forest e XGBoost. L'obiettivo finale è identificare il modello di classificazione più performante, basandosi sull'accuratezza media ottenuta durante la fase di validazione incrociata, e procedere con il salvataggio del modello ottimale per l'uso futuro.

3.1 Architettura del Sistema

3.1.1 Generazione del Dataset Sintetico

- **Input**: Un insieme di email di esempio suddivise in quattro categorie principali: Spam, Social, Work e Promotion.

- **Strumento**: Generazione di testi tramite un LLM, utilizzando il framework `Langchain`.
- **Obiettivo**: Creare un dataset sintetico composto da 1000 email etichettate secondo le quattro categorie definite.
- **Output**: Un file CSV contenente le email generate in lingua inglese, complete di etichette.

3.1.2 Preprocessing del Testo

- **Obiettivo**: Pulire e trasformare le email in un formato adatto per il training di modelli di machine learning.
- **Fasi**:
  1. **Rimozione della punteggiatura**: Le email vengono ripulite dalla punteggiatura per evitare che influisca negativamente durante il training.
  2. **Rimozione delle stop words**: Vengono eliminate le parole comuni che non aggiungono significato semantico rilevante (es. "the", "with", "for").
  3. **Tokenizzazione**: Le email vengono trasformate in sequenze di token (parole) tramite il *word tokenizer*.
  4. **Lemmatizzazione**: Riduzione delle parole alla loro forma base o lemma.

3.1.3 Sperimentazione degli Algoritmi di Machine Learning

Questa fase ha l'obiettivo di testare vari algoritmi di classificazione per determinare quale sia il più performante.

Pipeline di sperimentazione:

1. Utilizzare la tecnica di **k-fold cross-validation** per valutare le prestazioni di ciascun modello. Quindi dividere i dati in k sottoinsiemi (folds). Dopodiché, ad ogni iterazione, uno dei sottoinsiemi viene utilizzato come test set, mentre gli altri k-1 vengono usati per il training.
2. Calcolare l'*accuratezza* media.
3. Tracciare i tempi di esecuzione del training e del testing.
4. Salvare le metriche per ogni algoritmo.

3.1.4 Selezione del Miglior Modello

- **Obiettivo**: Identificare l'algoritmo con la migliore performance media sugli n-fold di cross-validation.
- **Criterio**: Selezione del modello in base all'accuratezza media.
- **Output**: Il nome del miglior modello.

3.1.5 Salvataggio del Modello

- **Strumento**: Pickle.
- **Obiettivo**: Salvare il miglior modello addestrato per l'utilizzo in un contesto di produzione o per future valutazioni.
- **Output**: File contenente il modello salvato, pronto per essere caricato e utilizzato per classificare nuove email.

3.2 Flusso di Lavoro del Sistema

1. **Generazione Dati Sintetici**: Utilizzo di Langchain con GPT-4o per generare un dataset di email sintetiche classificate. Le email vengono salvate in un file CSV.
2. **Preprocessing**: Il dataset di email viene pulito e trasformato: rimozione della punteggiatura, stop words, tokenizzazione e vettorizzazione. Viene generato un file preprocessato pronto per l'addestramento del modello.
3. **Addestramento e Valutazione Modelli**: I dati preprocessati vengono utilizzati per addestrare diversi modelli di classificazione (Logistic Regression, SVM, Random Forest, XGBoost). I modelli vengono valutati utilizzando la tecnica di k-fold cross-validation. Viene calcolata l'accuratezza media per ciascun modello.
4. **Selezione e Salvataggio del Modello**: Il modello con la migliore accuratezza media sugli n fold viene selezionato, salvato e reso disponibile per utilizzi futuri.

3.3 Possibili evolutive future

- **Ottimizzazione**: Ottimizzare l'addestramento dei modelli utilizzando tecniche di ricerca iperparametrica come `GridSearchCV` o `RandomSearchCV`.
- **Scalabilità**: In caso di aumento del volume dei dati, distribuire il preprocessing e l'addestramento su sistemi distribuiti utilizzando framework come `Dask` o `Ray`.
- **Espandere il dataset sintetico**: Aumentare il numero di email generate e includere ulteriori categorie di classificazione.
- **Migliorare la generazione di dati**: Sperimentare prompt più complessi per migliorare la qualità del testo generato.
- **Integrazione in un sistema di produzione**: Integrare il modello in un sistema di classificazione delle email reale con un'interfaccia utente o un'API.

4 Sviluppo

Il progetto *Synthetic Email Generation and Classification* è stato sviluppato per classificare automaticamente le email in categorie quali *Spam*, *Work*, *Social*, e *Promotion*. Il sistema è stato progettato per gestire dataset sintetici generati con l'ausilio di modelli GPT, attraverso il framework *Langchain*.

4.1 Generazione del Dataset

In assenza di dati reali, è stato utilizzato un generatore di email basato su GPT-4o per creare un dataset sintetico composto da 1000 email, suddivise in varie categorie. I dati generati sono stati salvati in file CSV, pronti per essere processati dal sistema.

4.2 Preprocessing dei Dati

Prima dell'addestramento dei modelli, le email sono state sottoposte a una fase di preprocessamento che ha incluso:

- Rimozione della punteggiatura.
- Eliminazione delle *stop word*.
- Tokenizzazione delle email.

Questo processo ha trasformato il testo grezzo in un formato utilizzabile per l'addestramento dei modelli di machine learning.

4.3 Modelli di Machine Learning

Successivamente, sono stati testati diversi modelli di machine learning per valutare le performance nella classificazione delle email. I modelli considerati sono stati:

- Regressione Logistica.
- Support Vector Machine (SVM).
- Random Forest.
- XGBoost.

Ciascun modello è stato sottoposto a una validazione incrociata *k-fold* per calcolare l'accuratezza media. Il modello con la migliore performance è stato selezionato e salvato per l'uso successivo.

4.4 Considerazioni Tecniche

- Frameworks e Librerie utilizzate:
  - `Langchain`: Per gestire le chiamate a GPT-4o.
  - `scikit-learn`: Per la vettorizzazione, l'addestramento e la valutazione dei modelli di machine learning.
  - `pandas`: Per la gestione e la manipolazione dei dati (lettura e scrittura CSV).
  - `NLTK`: Per la rimozione delle stop words, della punteggiatura e la tokenizzazione del testo.
  - `pickle`: Per il salvataggio dei modelli addestrati.

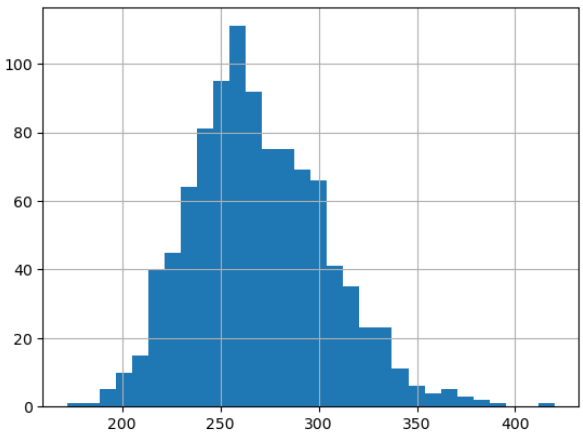

Figura 1: Istogramma delle lunghezze delle mail generate

5 Testing

5.1 Descrizione del dataset

Il dataset utilizzato per i test è stato generato artificialmente tramite il modello GPT-4o, suddiviso in quattro categorie principali: Social, Work, Spam e Promotion. Sono stati creati quattro file contenenti rispettivamente 300, 300, 300 e 100 email, per un totale di 1000 email, tenendo conto dei costi associati a ciascuna chiamata API a OpenAI, che ha comportato una spesa di circa 7.50€. I file hanno una struttura uniforme con due colonne: una contenente il testo dell'email e l'altra con la categoria corrispondente, entrambe in formato plain-text.

La lunghezza del testo delle email varia tra 172 e 420 caratteri, con una media di 272 caratteri e una deviazione standard di circa 35. La Figura 1 presenta un istogramma che descrive la distribuzione delle lunghezze delle email: la maggior parte delle email ha una lunghezza compresa tra 250 e 300 caratteri, con una distribuzione asimmetrica che mostra un numero ridotto di email significativamente più brevi o più lunghe.

5.2 Valutazione degli Algoritmi di Machine Learning

È stata condotta un'analisi comparativa di diversi algoritmi di classificazione (regressione logistica, SVM, gradient boosting, random forest) applicati al dataset di email. I modelli sono stati valutati tramite validazione incrociata, considerando sia metriche di performance, come l'accuratezza, sia il tempo di addestramento e predizione. I risultati ottenuti sono riportati di seguito.

5.2.1 Risultati

- **Regressione Logistica:**
  - Accuratezza Media: 0.847
  - Punteggi di Cross-Validation: [0.845, 0.875, 0.845, 0.84, 0.83]
  - Tempo di Cross-Validation: 0.27 secondi
- **SVM:**
  - Accuratezza Media: 0.813
  - Punteggi di Cross-Validation: [0.79, 0.835, 0.825, 0.81, 0.805]

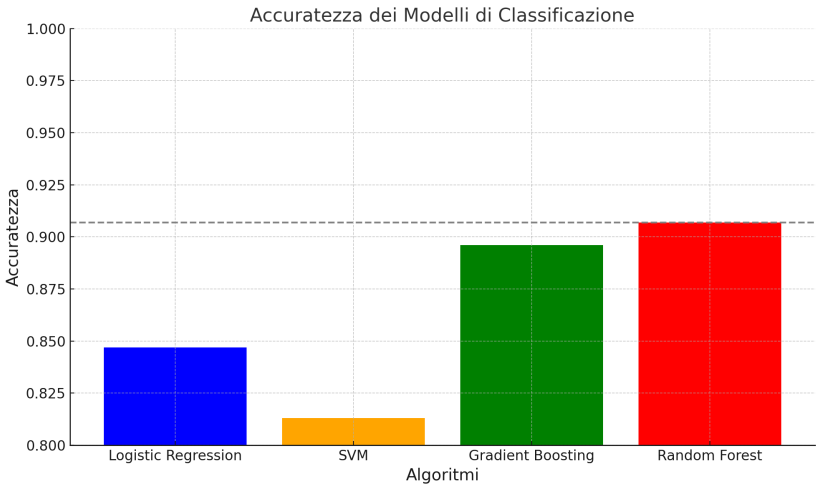

Figura 2: Confronto score di accuratezza dei quattro modelli

  - Tempo di Cross-Validation: 0.99 secondi

- **Gradient Boosting:**
  - Accuratezza Media: 0.896
  - Punteggi di Cross-Validation: [0.89, 0.91, 0.89, 0.88, 0.91]
  - Tempo di Cross-Validation: 13.89 secondi
- **Random Forest:**
  - Accuratezza Media: 0.907
  - Punteggi di Cross-Validation: [0.915, 0.915, 0.915, 0.895, 0.895]
  - Tempo di Cross-Validation: 1.76 secondi

I tempi di esecuzione mostrano differenze significative tra i vari algoritmi, con la regressione logistica che emerge per la sua rapidità. Sia Random Forest che SVM offrono un buon compromesso tra velocità e prestazioni, mentre il gradient boosting, sebbene richieda tempi di elaborazione maggiori, si distingue per le sue avanzate capacità predittive.

Nella Figura 2, è evidente che il modello Random Forest ha raggiunto l'accuratezza più elevata, seguito dal Gradient Boosting. Al contrario, la regressione logistica e l'SVM hanno mostrato prestazioni inferiori. La linea tratteggiata nella figura indica l'accuratezza massima ottenuta dai modelli testati.

5.2.2 Modello Migliore: Random Forest

I risultati dei test hanno dimostrato che il modello **Random Forest** ha raggiunto la miglior accuratezza nella classificazione, combinata con tempi di elaborazione ottimi. Di seguito sono presentati i risultati in dettaglio:

- Accuratezza (singolo train): 0.928

- **Classification Report:**

Categoria	**Precisione**	**Recall**	**F1-Score**
Spam	0.96	0.96	0.96
Promotion	0.90	0.90	0.90
Work	0.92	0.93	0.92
Personal	0.90	0.88	0.89
Media Ponderata	0.93	0.93	0.93

Tabella 1: Risultati delle metriche di classificazione per ciascuna categoria

- **Confusion Matrix:**

$$\begin{bmatrix} 92 & 3 & 1 & 0 \\ 3 & 38 & 0 & 1 \\ 1 & 1 & 67 & 3 \\ 0 & 0 & 5 & 35 \end{bmatrix}$$

6 Deploy

Il deploy del sistema di classificazione delle email è stato realizzato utilizzando Docker, consentendo così una distribuzione e un'esecuzione coerente in ambienti diversi. Questa sezione descrive il processo di configurazione del container Docker e l'implementazione del servizio utilizzando Docker Compose.

6.1 Dockerfile

Il Dockerfile definisce l'ambiente necessario per eseguire l'applicazione. Di seguito è riportato il contenuto del Dockerfile utilizzato:

```
FROM python:3.10-slim

LABEL authors="SANNICOLAF"

WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

CMD ["python", "main.py"]
```

Questo Dockerfile utilizza l'immagine base `python:3.10-slim` per garantire che il runtime Python 3.10 sia disponibile e in uno stato ridotto per ottimizzare le prestazioni e le dimensioni del container. Il file `requirements.txt` viene copiato nella cartella di lavoro `/app` e le dipendenze vengono installate utilizzando `pip`. Infine, tutto il codice dell'applicazione viene copiato nella stessa cartella e l'app viene avviata eseguendo `main.py`.

6.2 Docker Compose

Per semplificare la gestione del container, è stato utilizzato Docker Compose, che permette di definire e gestire i servizi in modo più efficiente. Ecco il file `docker-compose.yml` utilizzato:

```
version: "3.8"

services:
  email_classifier:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: email_classifier_app
    volumes:
      - .:/app
    command: ["python", "main.py"]
```

Nel file di configurazione Docker Compose, è definito un servizio chiamato `email_classifier`. Questo servizio costruisce l'immagine utilizzando il Dockerfile fornito nel contesto corrente. Il nome del container è impostato su `email_classifier_app`. Inoltre, viene montata la directory corrente nel container sotto `/app`, il che è utile durante lo sviluppo per poter vedere le modifiche apportate al codice senza dover ricostruire il container. Infine, il comando per avviare l'applicazione rimane lo stesso, eseguendo `main.py`.

6.3 Esecuzione del Deploy

Per avviare l'applicativo, è sufficiente eseguire il seguente comando nella directory contenente il file `docker-compose.yml`:

```
docker-compose up --build
```

Questo comando costruirà l'immagine Docker se non è già presente e avvierà il container. Una volta in esecuzione, il servizio sarà disponibile per la classificazione delle email, garantendo la consistenza e la portabilità dell'applicazione in vari ambienti di esecuzione.

6.4 Specifiche della Macchina di Test

La macchina utilizzata per eseguire i test del sistema ha le seguenti caratteristiche hardware:

- **Processore:** 12th Gen Intel(R) Core(TM) i7-1255U a 1.70 GHz
- **RAM:** 16 GB
- **Sistema operativo:** Windows 11 Pro

Queste specifiche sono più che sufficienti per garantire un'esecuzione fluida e ottimale dell'applicazione. I test eseguiti sulla macchina hanno dimostrato prestazioni stabili e tempi di risposta rapidi.