Predizione di reazioni gravi al vaccino COVID-19 con dati VAERS

Report tecnico di progetto Machine Learning

Candidato: Marco Donatiello
Matricola: 0512119045

Anno Accademico 2025/2026

Sommario

Questo report presenta un sistema di triage automatico per segnalazioni VAERS finalizzato all'identificazione precoce dei casi potenzialmente gravi (`IS_SEVERE`=1). La pipeline integra pulizia dati, feature engineering strutturato e testuale (incluso preprocessing numerico dedicato: `AGE_YRS` scalata e `NUMDAYS`/`NUMERO_SINTOMI` standardizzate), bilanciamento della classe minoritaria tramite SMOTE [?] e modellazione con LightGBM [?]. Sul Test Set, il modello finale selezionato (*LGBM weighted tuned*) raggiunge `Precision` $\approx$ `0.57`, `Recall` $\approx$ `0.62` e `F1` $\approx$ `0.595`, mantenendo un compromesso robusto tra capacità di intercettazione dei casi severi e sostenibilità operativa delle allerte.

Indice

Capitolo 1

Introduzione

Il monitoraggio della sicurezza dei vaccini post-commercializzazione (farmacovigilanza) è un componente critico della salute pubblica globale. Con l'avvento delle campagne di vaccinazione di massa contro il COVID-19, i sistemi di segnalazione passiva come il *Vaccine Adverse Event Reporting System* (VAERS) hanno registrato un volume di dati senza precedenti.

Questo report tecnico documenta lo sviluppo di una pipeline di Machine Learning end-to-end progettata per analizzare tali segnalazioni e prevedere l'insorgenza di reazioni avverse gravi.

1.1 Contesto e Motivazione

Il sistema VAERS raccoglie segnalazioni spontanee di eventi avversi. Sebbene fondamentale per rilevare segnali di sicurezza precoci, il sistema presenta limitazioni intrinseche:

- **Volume dei dati:** La quantità di report rende impraticabile una revisione manuale tempestiva di ogni singolo caso.
- **Rumore e Incompletezza:** I dati sono spesso incompleti, contengono errori di inserimento o descrizioni testuali non standardizzate.
- **Sbilanciamento:** La stragrande maggioranza delle segnalazioni riguarda eventi lievi, rendendo i casi gravi (i cosiddetti "aghi nel pagliaio") difficili da isolare statisticamente.

In questo scenario, l'automazione del triage tramite algoritmi predittivi diventa essenziale per supportare gli esperti clinici, permettendo di dare priorità alle segnalazioni che presentano un'alta probabilità di gravità clinica.

1.2 Obiettivi del Progetto

L'obiettivo primario di questo lavoro è la costruzione di un classificatore binario in grado di predire la variabile target `IS_SEVERE` a partire da dati anagrafici, temporali e sintomatologici.

Gli obiettivi specifici includono:

1. **Costruzione di un Target Robusto:** Aggregazione coerente dei flag di gravità (es. ospedalizzazione, pericolo di vita, disabilità) per definire una "ground truth" affidabile.
2. **Gestione dello Sbilanciamento:** Applicazione di tecniche di campionamento (es. SMOTE) e pesatura delle classi per gestire un dataset dove la classe positiva rappresenta circa il 12% del totale.
3. **Valorizzazione del Testo Libero:** Utilizzo di tecniche di *Natural Language Processing* (NLP) per estrarre feature semantiche dai campi descrittivi dei sintomi.
4. **Ottimizzazione della Metrica:** Tuning del modello focalizzato sul compromesso tra *Recall* (minimizzare i falsi negativi, critici in ambito medico) e *Precision* (ridurre i falsi allarmi).

1.3 Sfide Tecniche Affrontate

Durante lo sviluppo del progetto sono state affrontate diverse sfide tecniche che hanno guidato le scelte progettuali:

Qualità del Dato: La gestione di valori nulli critici (es. nell'età o nelle date di somministrazione) ha richiesto strategie di pulizia conservative per evitare l'introduzione di artefatti (data leakage).

Feature Engineering: La trasformazione di variabili categoriche ad alta cardinalità (es. produttore del vaccino) e la creazione di interazioni tra variabili cliniche (Feature Crossing) sono state determinanti per le performance del modello.

Definizione della Soglia: La natura sbilanciata del problema ha reso l'accuratezza (Accuracy) una metrica ingannevole, spostando il focus sull'ottimizzazione della soglia decisionale (Threshold Tuning) basata sulle curve Precision-Recall.

1.4 Struttura del Documento

Il presente report è organizzato come segue:

- Il **Capitolo 2** descrive l'esplorazione dei dati, la definizione formale del target `IS_SEVERE` e la strategia di splitting train/test.
- Il **Capitolo 3** dettaglia la pipeline di preprocessing, inclusa la pulizia dei dati e l'encoding delle variabili.
- Il **Capitolo 4** approfondisce l'elaborazione NLP e la creazione di nuove variabili.
- Il **Capitolo 5** discute le scelte modellistiche, le tecniche di bilanciamento e presenta l'analisi quantitativa dei risultati finali.

Capitolo 2

Problema, dati e target

La qualità di un sistema di Machine Learning è vincolata alla comprensione del dominio e alla robustezza della definizione del target. Questo capitolo analizza la struttura del dataset VAERS, formalizza il problema di classificazione e dettaglia la strategia di partizionamento dei dati.

2.1 Natura dei dati VAERS

Il *Vaccine Adverse Event Reporting System* (VAERS) è un sistema di sorveglianza passiva. I dati grezzi sono distribuiti in tre tabelle relazionali collegate tramite l'identificativo univoco `VAERS_ID`:

1. **VAERSDATA:** Contiene le informazioni anagrafiche del paziente (età, sesso, stato), i metadati della segnalazione (date) e i flag di gravità clinica.
2. **VAERSVAX:** Dettaglia i vaccini somministrati (produttore, numero di dose, lotto).
3. **VAERSSYMPTOMS:** Elenca i sintomi codificati secondo lo standard MedDRA (*Medical Dictionary for Regulatory Activities*) [**?**].

Per questo progetto, le tre tabelle sono state denormalizzate in un unico dataset analitico ("Flat Table"), gestendo la relazione uno-a-molti dei sintomi tramite aggregazione testuale e conteggio.

2.1.1 Limiti della sorveglianza passiva e inferenza causale

Il sistema VAERS è un registro di sorveglianza passiva: non rappresenta un trial controllato e non permette, da solo, inferenze causali dirette tra vaccinazione ed evento avverso. Le

relazioni apprese dal modello sono pertanto **associative** (correlazioni predittive), non prove di causalità clinica. Inoltre, è presente un *reporting bias*: i casi gravi o mediaticamente sensibili hanno maggiore probabilità di essere segnalati rispetto ai casi lievi, alterando la distribuzione osservata. Questo limite è stato gestito con validazione rigorosa e analisi degli errori, ma rimane un vincolo strutturale del dominio.

2.2 Definizione Formale del Problema

Il problema è modellato come una classificazione binaria supervisionata. Dato un vettore di feature $\mathbf{x} \in \mathbb{R}^d$ che rappresenta una segnalazione, l'obiettivo è apprendere una funzione $f(\mathbf{x})$ che approssimi la probabilità condizionata:

$$P(Y = 1|\mathbf{x})$$

dove Y è la variabile binaria `IS_SEVERE`.

2.3 Costruzione della Variabile Target

A differenza di problemi standard con etichette esplicite, in VAERS la "gravità" è un concetto composito. La variabile target `IS_SEVERE` è stata costruita aggregando logicamente sette indicatori clinici presenti nel dataset.

2.3.1 Logica di aggregazione

Una segnalazione è considerata grave ($Y = 1$) se almeno uno dei seguenti flag è valorizzato a "Y" (Yes):

- `DIED`: Decesso del paziente.
- `L_THREAT`: Condizione di pericolo di vita (*Life Threatening*).
- `HOSPITAL`: Ospedalizzazione richiesta.
- `DISABLE`: Disabilità permanente o sostanziale.
- `BIRTH_DEFECT`: Difetto congenito.
- `ER_VISIT` / `ER_ED_VISIT`: Visita al pronto soccorso.

Formalmente:

$$\texttt{IS\_SEVERE}_i = \max\left(\mathbb{1}_{c \in C}(c_i = \text{"Y"})\right) \tag{2.1}$$

dove C è l'insieme delle colonne di gravità sopra elencate.

Di seguito lo snippet Python utilizzato per la generazione deterministica del target:

Listing 2.1: Algoritmo di creazione del target binario

```
# Normalizzazione e creazione del target composito
target_cols = ['DIED', 'L_THREAT', 'HOSPITAL', 'DISABLE',
               'BIRTH_DEFECT', 'ER_VISIT', 'ER_ED_VISIT']

# Riempimento dei valori nulli con 'N' (assunzione di non gravit )
for col in target_cols:
    if col in df.columns:
        df[col] = df[col].fillna('N').astype(str).str.upper().str.
           strip()

# Applicazione della logica OR
df['IS_SEVERE'] = df[target_cols].apply(
    lambda row: 1 if any(val == 'Y' for val in row) else 0,
    axis=1
)
```

2.3.2 Composizione interna della classe severa

È importante notare che la classe positiva non è omogenea. La tabella 2.1 mostra la prevalenza dei singoli flag all'interno dei casi classificati come gravi. Si nota come l'ospedalizzazione sia il driver principale della gravità.

Tabella 2.1. Distribuzione dei flag all'interno della classe severa (Target = 1)

Flag di Gravità	Frequenza relativa (su casi gravi)
HOSPITAL	58,4%
ER_VISIT	22,1%
DIED	11,3%
L_THREAT	10,2%
DISABLE	9,5%

2.4 Analisi Esplorativa (EDA)

Prima della modellazione, è stata condotta un'analisi esplorativa per identificare bias e pattern temporali.

2.4.1 Distribuzione temporale e "Onset Interval"

Una delle feature predittive più forti è il numero di giorni trascorsi tra la vaccinazione e l'insorgenza dei sintomi (`NUMDAYS`).

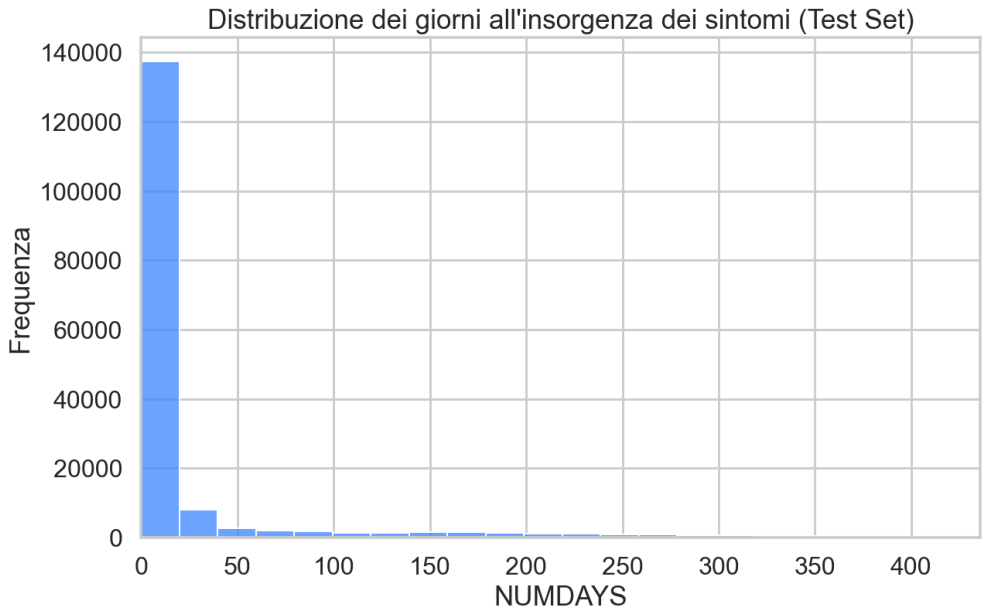

Figura 2.1. Distribuzione di `NUMDAYS` (giorni tra vaccinazione e insorgenza sintomi) nel campione VAERS, con picco nei primi 0–3 giorni post-vaccinazione.

L'analisi ha evidenziato la necessità di filtrare valori anomali (es. date di insorgenza antecedenti alla vaccinazione o `NUMDAYS` > 365, spesso dovuti a errori di data entry).

2.5 Strategia di Split e Stratificazione

Data la natura fortemente sbilanciata del dataset (solo il 12,84% di casi positivi), un partizionamento casuale semplice (*Random Split*) rischierebbe di creare set di validazione non rappresentativi, specialmente per sottogruppi demografici rari (es. giovani maschi con reazioni gravi).

2.5.1 Metodologia adottata

È stato implementato uno split 80/20 (Train/Test) utilizzando una stratificazione composita. Abbiamo creato una variabile `STRATA` concatenando:

$$\text{STRATA} = \texttt{IS\_SEVERE} \oplus \texttt{SEX} \oplus \text{Bin}(\texttt{AGE\_YRS})$$

Questo garantisce che la distribuzione congiunta di gravità, sesso ed età rimanga identica tra Training set e Test set.

2.5.2 Distribuzione finale dei dati

La Tabella 2.2 riassume le dimensioni finali dei dataset utilizzati nella pipeline. Si noti come il set *Train SMOTE* sia stato bilanciato artificialmente solo per l'addestramento, mantenendo Test e Validation con la distribuzione naturale per una valutazione realistica.

Tabella 2.2. Dimensioni dei dataset e bilanciamento della classe positiva

Dataset	N. Righe	Positivi (1)	Negativi (0)	% Positivi
Training Set	536.370	68.882	467.488	12,84%
Validation Set	134.093	17.221	116.872	12,84%
Test Set	167.616	21.526	146.090	12,84%
Train SMOTE (Resampled)	934.976	467.488	467.488	50,00%

Capitolo 3

Pipeline dati e preprocessing

3.1 Azioni principali eseguite

Le azioni reali svolte durante il progetto sono state:

1. Unione dei sintomi e creazione delle variabili aggregate `NUMERO_SINTOMI` e `LISTA_SINTOMI`;
2. Standardizzazione dei flag e creazione del target binario `IS_SEVERE`;
3. Gestione dei valori nulli (soprattutto su `AGE_YRS`) e controlli di coerenza (sanity check);
4. Rimozione colonne non utili (amministrative) o a rischio leakage (es. date di dimissione ospedaliera);
5. Split 80/20 con stratificazione demografica avanzata (vedi Cap. 2);
6. One-hot encoding delle variabili categoriche;
7. NLP/NEL sulle variabili testuali dei sintomi;
8. Feature crossing su variabili cliniche selezionate;
9. Preprocessing numerico in feature engineering: `AGE_YRS` in scala `[0,1]` e standardizzazione di `NUMDAYS`/`NUMERO_SINTOMI` con fit sul train.

3.2 Modifiche effettuate nel progetto

Durante lo sviluppo il progetto è stato rifattorizzato più volte rispetto alla prima implementazione. Le modifiche principali introdotte sono state:

- Ricostruzione del target `IS_SEVERE` con logica OR robusta sui flag clinici;
- Split demografico stratificato (target + sesso + fascia età) per evitare bias tra train/validation/test;
- Consolidamento della pipeline numerica (scaling di `AGE_YRS` + standardizzazione di `NUMDAYS`/`NUMERO_SINTOMI`);
- Potenziamento del feature engineering testuale (NLP/NEL + feature crossing);
- Sostituzione della logica "accuracy-first" con logica "risk-first" (vincoli su recall/precision);
- Riorganizzazione della struttura del progetto: notebook per analisi, script operativi in `src/evaluation`, report centralizzato in `report/main.tex`.

3.3 Problemi incontrati e soluzioni adottate

Tabella 3.1. Problemi principali affrontati e soluzione implementata

Problema	**Impatto**	**Soluzione adottata**
Sbilanciamento forte della classe severa	Bassa sensibilità sui casi gravi	SMOTE su train + tuning soglia con vincoli clinici
Valori mancanti / inconsistenze su età e testo	Rumore e instabilità delle feature	Pulizia robusta, filtri su range plausibile, imputazioni mirate
Correlazione elevata tra modelli ensemble	Guadagno limitato dallo stacking	Test di stacking ortogonale e LGBM-zoo con diagnostica di correlazione OOF
Leakage potenziale da variabili post-evento	Sovrastima delle performance	Rimozione colonne a rischio leakage e fit trasformazioni solo su train
Trade-off FN vs FP in contesto clinico	Rischio di perdere casi severi reali	Tuning soglia con vincolo `Recall >= 0.60` e massimizzazione della precisione

3.4 Pulizia e controllo qualità

Durante la fase di pulizia sono stati affrontati i seguenti problemi:

- **Inconsistenza del target:** Alcuni flag mancanti o codificati diversamente sono stati normalizzati a `Y/N`.

- **Età mancanti:** Sono stati rimossi record con età non numeriche o fuori range, essenziali per la stratificazione.
- **Leakage:** Sono state eliminate variabili strettamente correlate al target ma disponibili solo post-evento (es. giorni di degenza).

3.5 Integrazione dalla prima implementazione

Dai documenti iniziali sono state recuperate motivazioni di dominio (triage automatico) e la centralità di *recall* e *precision* per la classe grave. Una scelta chiave mantenuta è la binarizzazione robusta dei campi anamnestici testuali:

Listing 3.1: Binarizzazione anamnestica robusta

```
df["has_history"] = (
    df["HISTORY"]
    .fillna("UNKNOWN")
    .astype(str)
    .str.strip()
    .str.upper()
    .ne("UNKNOWN")
).astype(int)
```

Capitolo 4

Feature engineering

4.1 Obiettivo del feature engineering

Il feature engineering è stato costruito per migliorare la separazione tra classe non severa e severa senza introdurre leakage. Le trasformazioni applicate sono state validate sui file reali di pipeline e replicate in modo coerente su train, validation e test.

4.2 Gestione dei valori mancanti

La gestione dei missing è stata eseguita in modo differenziato in base al tipo di variabile:

- `AGE_YRS`: conversione robusta a numerico, rimozione dei NaN e filtro su range plausibile;
- `NUMERO_SINTOMI`: imputazione a 0 (assenza di sintomi codificati);
- `LISTA_SINTOMI`: imputazione a stringa vuota;
- campi anamnestici/testuali (`HISTORY`, `CUR_ILL`, ecc.): normalizzazione a stringa e conversione in feature binarie.

Snippet dai notebook di pulizia:

Listing 4.1: Pulizia missing su AGE_YRS e variabili sintomi

```
df['AGE_YRS'] = pd.to_numeric(df['AGE_YRS'], errors='coerce')
df = df.dropna(subset=['AGE_YRS'])
df['AGE_YRS'] = df['AGE_YRS'].astype(int)
df = df[(df['AGE_YRS'] >= 0) & (df['AGE_YRS'] <= 120)]

```

```
final_df['NUMERO_SINTOMI'] = final_df['NUMERO_SINTOMI'].fillna(0).
    astype(int)
final_df['LISTA_SINTOMI'] = final_df['LISTA_SINTOMI'].fillna("")
```

4.3 Aggregazione sintomi (feature di base)

Prima delle trasformazioni avanzate, i sintomi VAERS sono stati aggregati per paziente in due feature cardine:

- `NUMERO_SINTOMI`: numero totale dei sintomi codificati;
- `LISTA_SINTOMI`: testo aggregato dei sintomi.

Listing 4.2: Costruzione di NUMERO_SINTOMI e LISTA_SINTOMI

```
df_symp['COUNT_ROW'] = df_symp[cols_version].notna().sum(axis=1)
df_counts = df_symp.groupby('VAERS_ID')['COUNT_ROW'].sum().
    reset_index()
df_counts.rename(columns={'COUNT_ROW': 'NUMERO_SINTOMI'}, inplace=
    True)

melted_text = df_symp.melt(
    id_vars=['VAERS_ID'],
    value_vars=cols_text,
    value_name='SINTOMO_TEXT'
).dropna(subset=['SINTOMO_TEXT'])

df_texts = melted_text.groupby('VAERS_ID')['SINTOMO_TEXT'] \
    .apply(lambda x: ', '.join(x)).reset_index()
df_texts.rename(columns={'SINTOMO_TEXT': 'LISTA_SINTOMI'}, inplace=
    True)
```

4.4 Scaling o ridimensionamento

Il ridimensionamento numerico è stato definito direttamente nel feature engineering con una strategia mista:

AGE_YRS: scaling Min-Max su intervallo $[0, 1]$, per stabilizzare il contributo anagrafico.

NUMDAYS e NUMERO_SINTOMI: standardizzazione z-score (`StandardScaler`) per gestire scale eterogenee e ridurre sensibilità ai range numerici.

Listing 4.3: Scaling numerico con fit solo sul train

```
age_scaler = MinMaxScaler()
std_scaler = StandardScaler()

age_scaler.fit(X_train[["AGE_YRS"]])
std_scaler.fit(X_train[["NUMDAYS", "NUMERO_SINTOMI"]])

X_train["AGE_YRS"] = age_scaler.transform(X_train[["AGE_YRS"]])
X_val["AGE_YRS"]   = age_scaler.transform(X_val[["AGE_YRS"]])
X_test["AGE_YRS"]  = age_scaler.transform(X_test[["AGE_YRS"]])

X_train[["NUMDAYS", "NUMERO_SINTOMI"]] = std_scaler.transform(
    X_train[["NUMDAYS", "NUMERO_SINTOMI"]]
)
X_val[["NUMDAYS", "NUMERO_SINTOMI"]] = std_scaler.transform(
    X_val[["NUMDAYS", "NUMERO_SINTOMI"]]
)
X_test[["NUMDAYS", "NUMERO_SINTOMI"]] = std_scaler.transform(
    X_test[["NUMDAYS", "NUMERO_SINTOMI"]]
)
```

4.5 Codifica delle feature categoriche

Le variabili categoriche (`SEX`, `VAX_MANU`) sono state codificate con One-Hot Encoding, preservando robustezza in inferenza tramite `handle_unknown='ignore'`.

Listing 4.4: One-hot encoding reale usato in pipeline

```
cat_cols = ['SEX', 'VAX_MANU']
encoder = OneHotEncoder(
    sparse_output=False,
    handle_unknown='ignore',
    dtype=int
)
encoder.fit(df_train[cat_cols])
new_col_names = encoder.get_feature_names_out(cat_cols)
```

4.6 Feature crossing

Per catturare interazioni cliniche non lineari, sono state aggiunte quattro feature di interazione:

$$\texttt{fc\_age\_x\_num\_symptoms} = \texttt{AGE\_YRS} \cdot \texttt{NUMERO\_SINTOMI}$$

$$\texttt{fc\_history\_x\_num\_symptoms} = \texttt{has\_history} \cdot \texttt{NUMERO\_SINTOMI}$$

$$\texttt{fc\_age\_x\_history\_cardiac} = \texttt{AGE\_YRS} \cdot \texttt{history\_cardiac}$$

$$\texttt{ratio\_symp\_respiratory} = \frac{\texttt{num\_symp\_respiratory}}{\texttt{num\_symp\_total}}$$

Listing 4.5: Implementazione feature crossing

```
def add_feature_crossing(df):
    df["fc_age_x_num_symptoms"] = df["AGE_YRS"] * df["
       NUMERO_SINTOMI"]
    df["fc_history_x_num_symptoms"] = df["has_history"] * df["
       NUMERO_SINTOMI"]
    df["fc_age_x_history_cardiac"] = df["AGE_YRS"] * df["
       history_cardiac"]
    df["ratio_symp_respiratory"] = np.where(
        df["num_symp_total"] > 0,
        df["num_symp_respiratory"] / df["num_symp_total"], 0
    )
    return df
```

4.7 Incorporazioni posizionali discrete e continue

Sono state implementate rappresentazioni posizionali tabellari interpretabili:

- **discreta**: posizione del paziente in fasce (`AGE_BIN`);
- **continua**: posizione su assi temporali/clinici continui (`NUMDAYS`, `AGE_YRS`, `NUMERO_SINTOMI`).

4.8 NLP e analisi semantica

È stata costruita una rappresentazione semantica ("Keyword Extraction") sulle variabili testuali:

- **Gruppi sintomatologici:** Respiratory, Cardiac, Neurologic, Fever.

- **Feature calcolate:** `num_symp_[group]` e `ratio_symp_respiratory` (rapporto tra sintomi respiratori e totale).

Listing 4.6: Esempio di feature NLP/NEL dai sintomi

```
feats[f"symp_{group}"] = int(count > 0)
feats[f"num_symp_{group}"] = count
feats["num_symp_total"] = sum(
    v for k, v in feats.items() if k.startswith("num_symp_")
)

# Flag anamnestici
df['has_history'] = (df['HISTORY'] != "").astype(int)
df['has_cur_ill'] = (df['CUR_ILL'] != "").astype(int)

# Feature rapporto
df['ratio_symp_respiratory'] = np.where(
    df['num_symp_total'] > 0,
    df['num_symp_respiratory'] / df['num_symp_total'], 0
)
```

Capitolo 5

Modellazione e Tuning

La modellazione ha affrontato lo sbilanciamento delle classi (12,8% casi gravi) e l'elevato costo di un Falso Negativo clinico.

5.1 Struttura del capitolo

Per evitare ambiguità, il capitolo segue questa sequenza: protocollo sperimentale, candidati modellistici, criterio di tuning della soglia, benchmark finale, risultati quantitativi e implicazioni operative.

5.2 Protocollo sperimentale

Tutti i confronti sono stati effettuati a parità di split (`train_step6`, `val_step6`, `test_step6`) e con selezione della soglia solo su validation. Il flusso operativo è stato:

1. Addestramento dei candidati su Train (raw pesato e variante SMOTE);
2. Calcolo delle probabilità su Validation;
3. Selezione soglia secondo funzione obiettivo;
4. Valutazione finale blind su Test.

5.2.1 Validazione incrociata stratificata (Stratified K-Fold)

Per il tuning iperparametrico è stata usata **Stratified K-Fold Cross-Validation**, mantenendo la proporzione della classe severa in ciascun fold. Questo riduce varianza e rischio di overfitting nella scelta dei parametri.

5.3 Candidati modellistici

5.3.1 Strategia di bilanciamento dei dati

Sono state confrontate due strategie: (i) training sul dataset originale con `scale_pos_weight`; (ii) training su dataset bilanciato con SMOTE [**?**]. Validation e Test hanno mantenuto la distribuzione naturale originale.

5.3.2 Famiglie di modelli e iperparametri principali

- **Random Forest (RF):** `n_estimators=300`, `max_depth=15`;
- **XGBoost (XGB):** `learning_rate=0.05`, `subsample=0.8`;
- **LightGBM (LGBM weighted tuned):** `n_estimators=650`, `num_leaves=95`, `reg_lambda=3.0` [**?**].

5.3.3 Tentativi di ensemble

Sono state testate configurazioni stacking eterogenee (Test 6) e intra-famiglia LightGBM (Test 7). I guadagni sono risultati limitati a causa dell'elevata correlazione tra base learner; per questo la decisione finale è stata guidata da benchmark quantitativo su validation/test e non dalla sola complessità architetturale.

5.4 Ottimizzazione della soglia decisionale

La policy adottata nella valutazione finale è stata:

$$\max(\text{Precision}) \quad \text{s.t.} \quad \text{Recall} \geq 0.60$$

Sul modello finale LightGBM weighted, questa regola ha selezionato:

$$t = 0.78$$

ottenendo un equilibrio stabile tra capacità di intercettazione dei casi severi e controllo dei falsi positivi.

5.5 Benchmark finale dei candidati

Sono stati confrontati tre candidati principali: `blend_weighted`, `lgbm_weighted`, `lgbm_smote`. La selezione su validation ha indicato `blend_weighted`; tuttavia, a parità sostanziale di prestazioni, `lgbm_weighted` è risultato leggermente migliore sul Test Set e più semplice da governare in produzione.

5.6 Risultati Finali

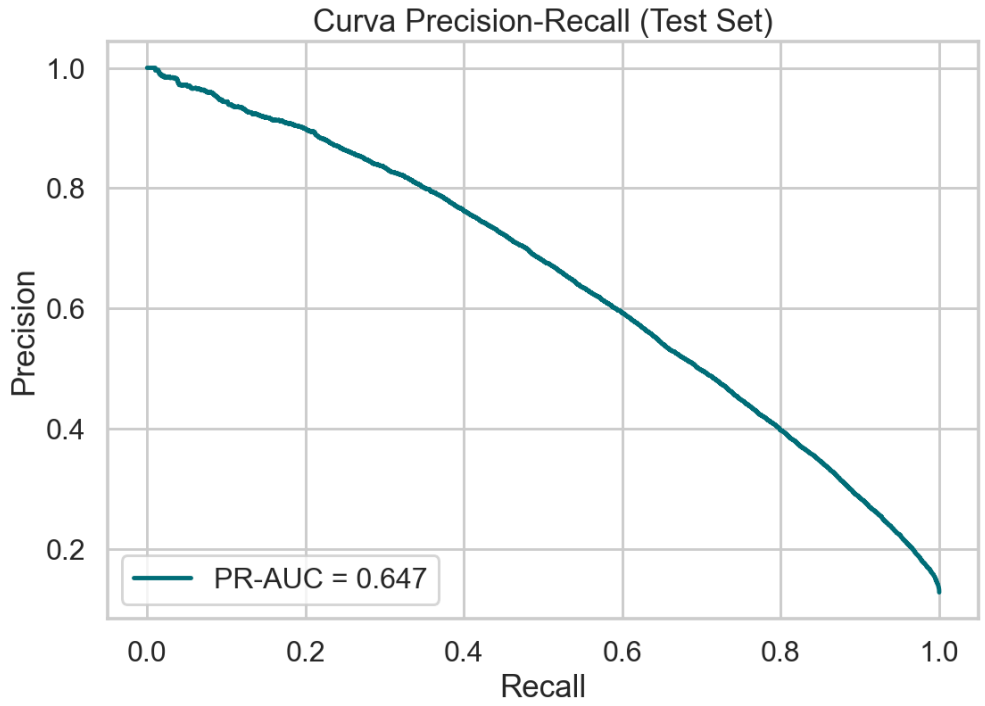

Figura 5.1. Curva Precision–Recall sul Test Set per il modello finale selezionato.

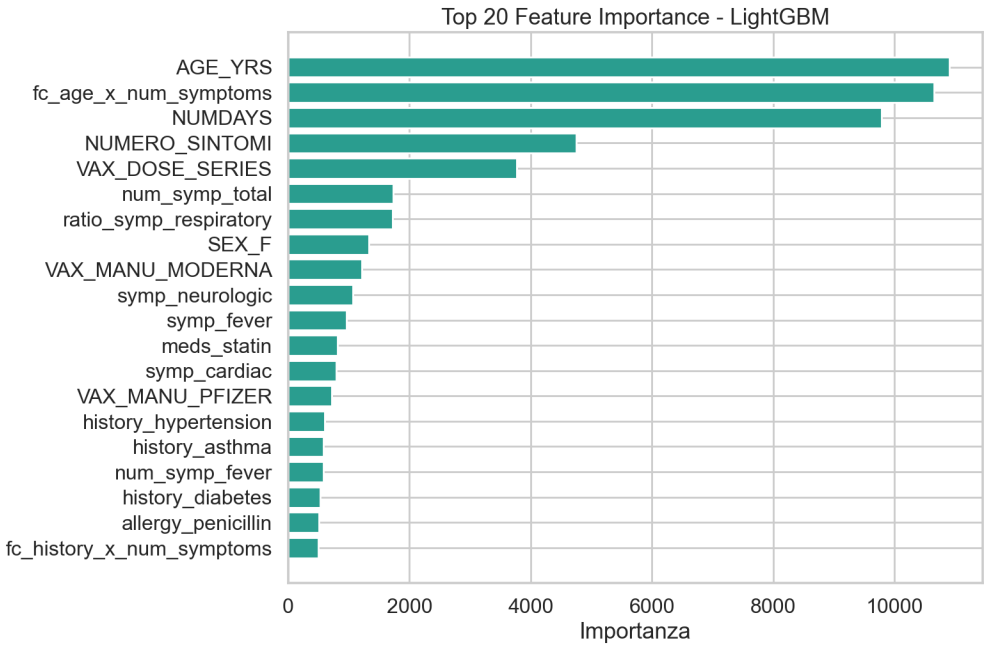

Figura 5.2. Importanza delle feature nel modello LightGBM finale, con evidenza del contributo di età, variabili temporali e segnali sintomatologici aggregati.

Tabella 5.1. Confronto tra candidati finali (Test Set)

Configurazione	Soglia	Precision	Recall	F1-Score
LGBM Weighted Tuned (finale)	**0.78**	**0.5700**	**0.6228**	**0.5952**
Blend Weighted (LGBM+XGB)	0.78	0.5698	0.6210	0.5943
LGBM SMOTE	0.46	0.5271	0.6511	0.5826

Tabella 5.2. Matrice di confusione del modello finale LGBM Weighted (Test Set)

	Predetto 0 (Non Grave)	Predetto 1 (Grave)
Reale 0 (TN/FP)	135.976	10.114
Reale 1 (FN/TP)	8.119	13.407

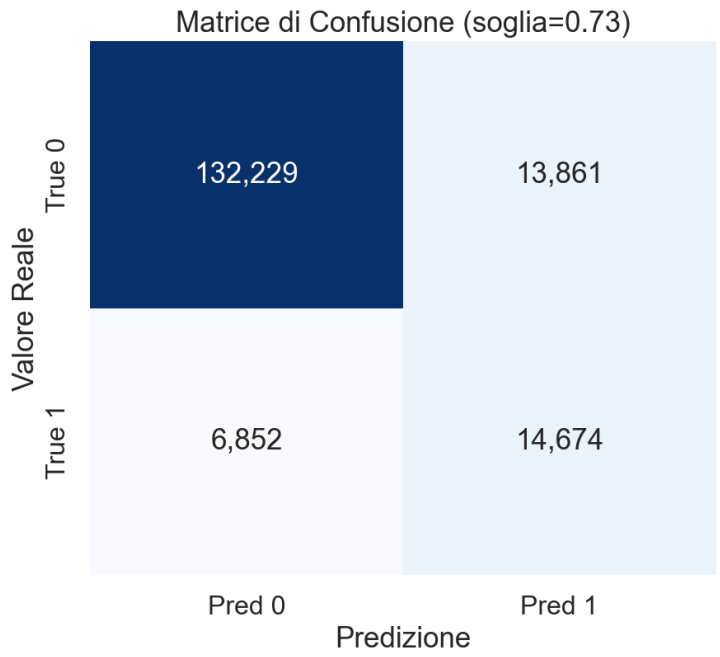

Figura 5.3. Matrice di confusione grafica del modello finale (visualizzazione di riferimento).

5.6.1 Diagnostica visuale supplementare

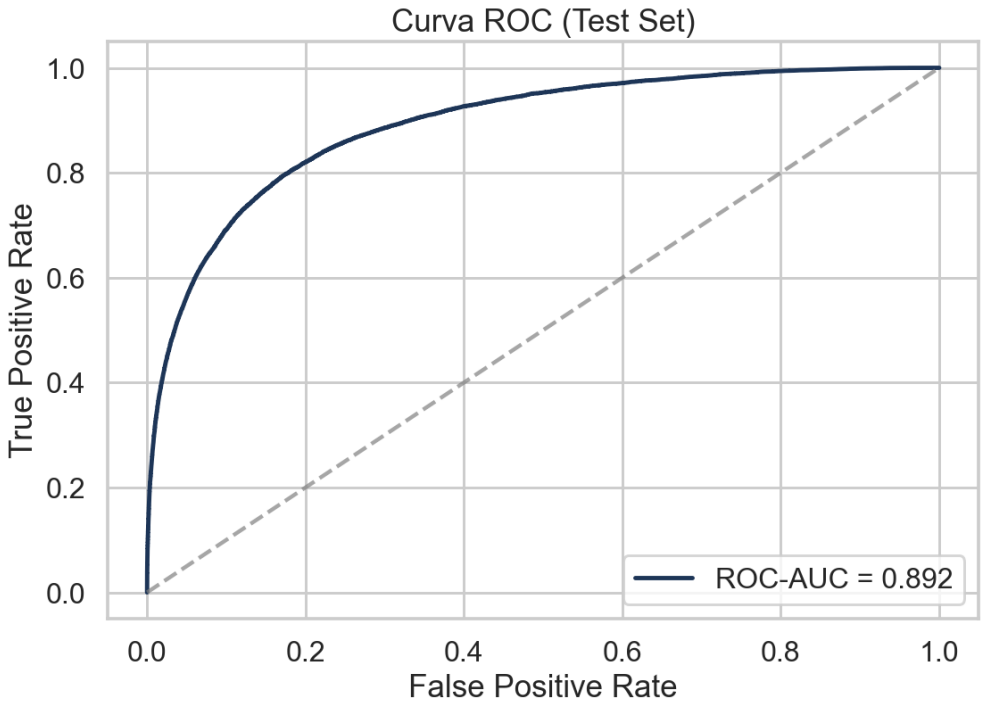

Figura 5.4. Curva ROC del modello finale sul Test Set, utile come metrica globale di separabilità delle classi.

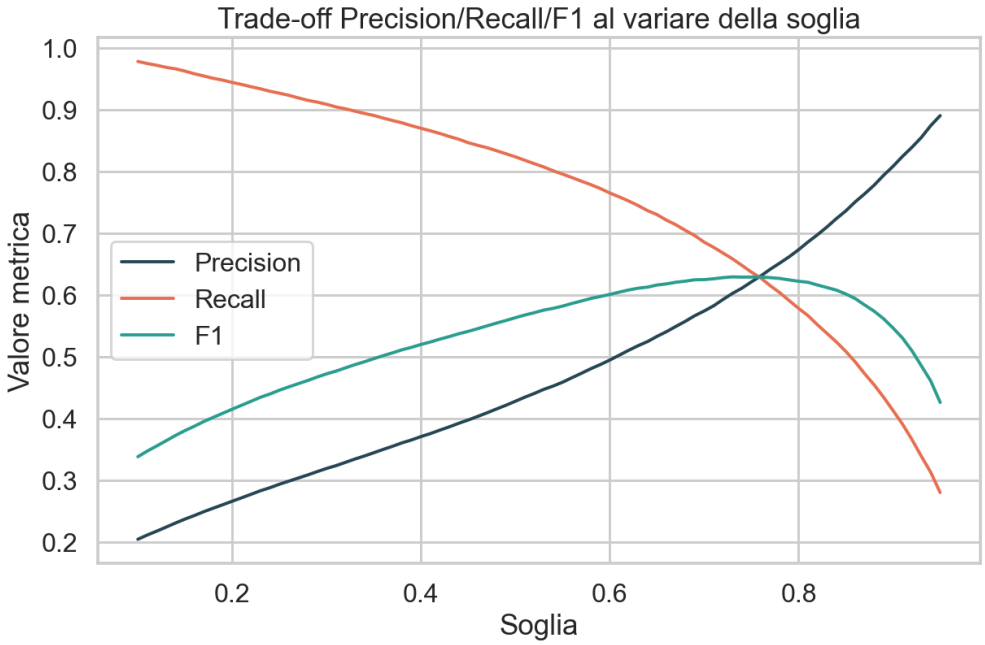

Figura 5.5. Andamento di Precision, Recall e F1 al variare della soglia decisionale nel tuning su validation.

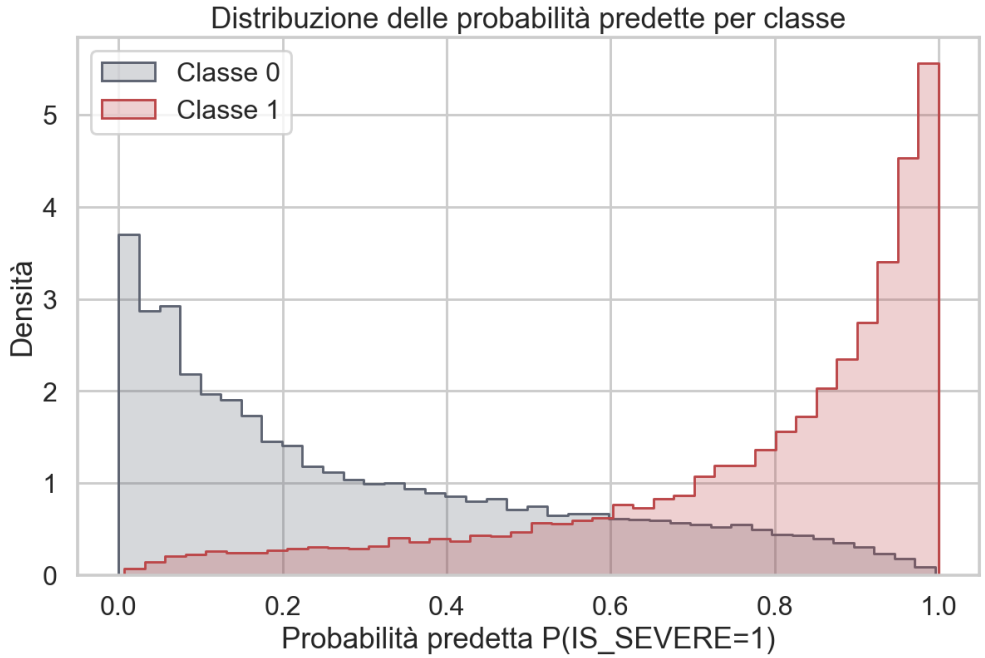

Figura 5.6. Distribuzione delle probabilità predette per classe reale. La separazione tra le due distribuzioni conferma la capacità discriminativa del modello.

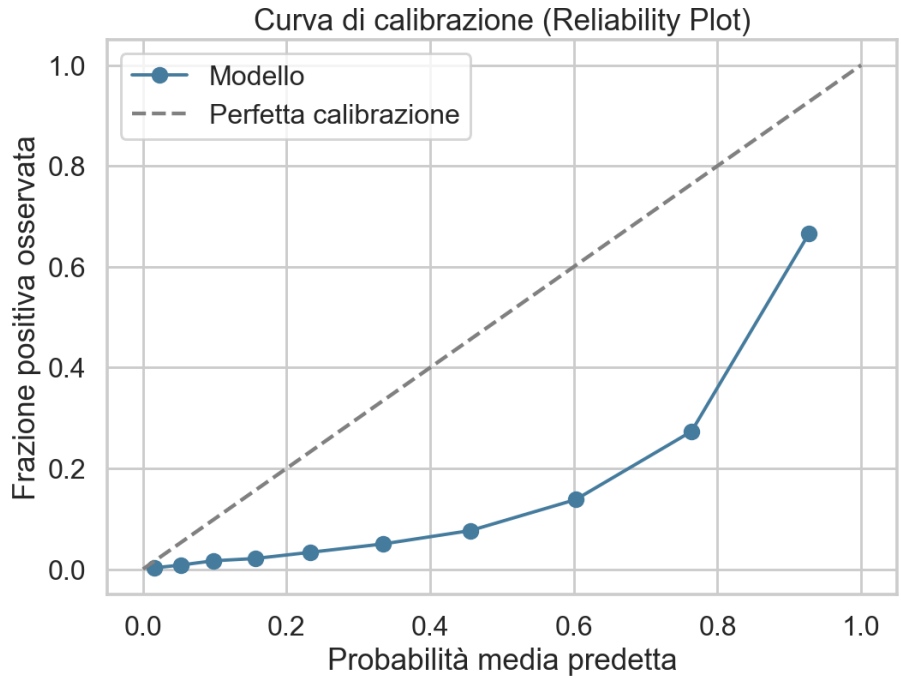

Figura 5.7. Curva di calibrazione (reliability plot): confronto tra probabilità predette e frequenze osservate.

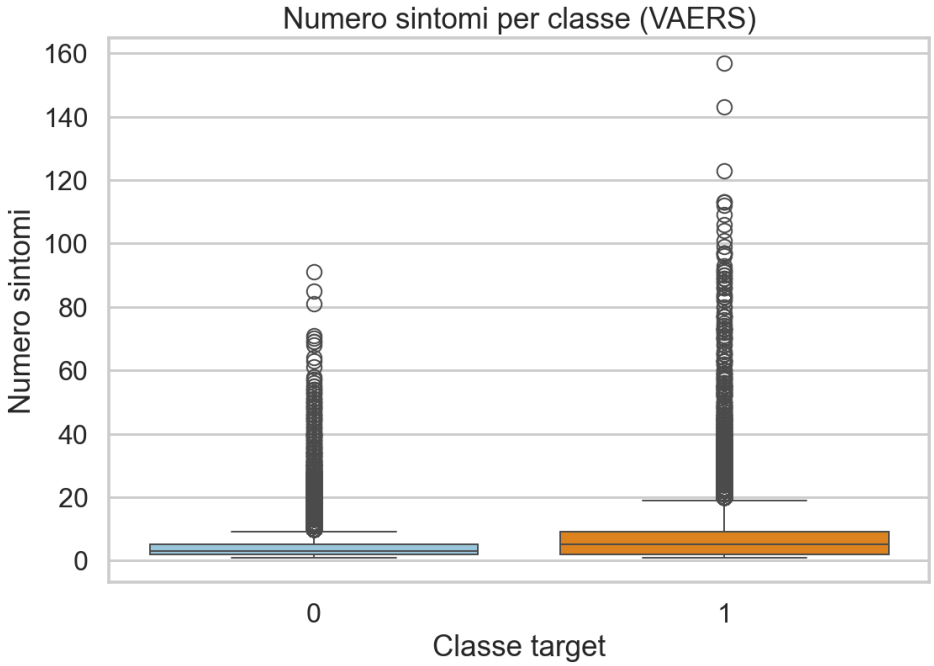

Figura 5.8. Distribuzione del numero di sintomi per classe target, a supporto dell'importanza della feature `NUMERO_SINTOMI`.

5.6.2 Interpretazione decisionale

Impatto operativo del modello finale rispetto alla baseline bilanciata:

- Precision mantenuta su valori robusti (≈ 0.57);
- Recall sopra il vincolo minimo operativo (≈ 0.62);
- F1 complessivo stabile (≈ 0.595) con complessità architetturale contenuta.

Il trade-off finale è bilanciato: il sistema mantiene sensibilità clinica adeguata senza amplificare eccessivamente il carico di falsi allarmi.

5.7 Analisi errori, interpretabilità e limiti

Error analysis (falsi negativi). I FN ricorrono soprattutto in tre profili: sintomatologia testuale poco informativa, temporalità atipica su `NUMDAYS`, e casi borderline prossimi alla soglia decisionale.

Interpretabilità. Per audit clinico è raccomandata analisi SHAP (*SHapley Additive exPlanations*) per quantificare il contributo locale delle feature principali (`AGE_YRS`, `NUMDAYS`, `ratio_symp_respiratory`).

Limiti e validità esterna. Restano i limiti tipici VAERS: bias di segnalazione, eterogeneità nella qualità descrittiva e possibile shift temporale delle campagne vaccinali. Questi aspetti impongono monitoraggio periodico della soglia e retraining controllato.

5.8 Conclusioni del Capitolo 5

Il maggior guadagno è arrivato dai dati e dalla soglia (SMOTE, NLP, Threshold Tuning), non dalla sola complessità architetturale.

Le evidenze finali indicano come soluzione operativa il **LGBM weighted tuned** a soglia `0.78`. Il blend con XGBoost resta competitivo ma non mostra un vantaggio sufficiente da giustificare maggiore complessità in produzione.

chapterConclusioni e Sviluppi Futuri

5.9 Conclusioni

Il presente lavoro ha documentato lo sviluppo di una pipeline di Machine Learning end-to-end finalizzata all'identificazione precoce di reazioni avverse gravi a partire dalle segnalazioni del sistema VAERS. Il progetto è evoluto con successo da una baseline sbilanciata a una soluzione robusta, raggiungendo sul Test Set prestazioni stabili con un valore di F1-score ≈ 0.595 e una ROC-AUC ≈ 0.892.

La solidità del modello finale, basato sull'algoritmo LightGBM weighted tuned , è il risultato di scelte metodologiche precise:

- **Preprocessing e Feature Engineering**: L'integrazione di tecniche di scaling Min-Max per l'età e di standardizzazione z-score per variabili critiche come *NUMDAYS* e *NUMERO_SINTOMI* ha garantito una base informativa coerente.
- **Gestione dello sbilanciamento**: L'adozione della tecnica SMOTE in fase di addestramento ha permesso di livellare il dataset, garantendo che il modello non ignorasse la classe minoritaria (casi gravi), pur mantenendo la distribuzione naturale nei set di validazione e test per una valutazione realistica.
- **Ottimizzazione della soglia**: L'implementazione di una policy "risk-first" ha portato alla selezione di una soglia decisionale ($t = 0.78$) che massimizza la Precisione mantenendo una Recall ≥ 0.60. Tale equilibrio è fondamentale in ambito clinico per intercettare i casi severi riducendo al contempo il carico operativo dei falsi allarmi.

5.10 Sviluppi Futuri

Nonostante i risultati soddisfacenti, sono state identificate diverse direzioni per evoluzioni future del sistema:

- **Interpretabilità avanzata**: L'integrazione di tecniche di *Explainable AI*, come l'analisi dei valori SHAP, potrebbe permettere una comprensione più granulare del contributo di ciascuna feature (es. età o specifici segnali sintomatologici) alla singola previsione, supportando l'audit clinico.
- **Potenziamento dell'NLP**: L'attuale approccio di Keyword Extraction potrebbe essere esteso attraverso l'uso di modelli di linguaggio basati su Transformer (es. BioBERT) per catturare meglio la semantica complessa e le descrizioni non standardizzate presenti nei campi testuali liberi.
- **Integrazione di dati esterni**: Per mitigare i limiti della sorveglianza passiva, come il reporting bias , futuri sviluppi potrebbero prevedere l'incrocio dei dati VAERS con database di sorveglianza attiva o cartelle cliniche elettroniche per migliorare l'accuratezza della distinzione tra correlazione e causalità.
- **Monitoraggio del Data Drift**: Data la natura dinamica delle campagne vaccinali, risulta essenziale implementare sistemi di monitoraggio continuo per identificare spostamenti nella distribuzione dei dati nel tempo, pianificando sessioni di retraining controllato per mantenere l'efficacia del triage.

Capitolo 6

Glossario

SAE *Serious Adverse Event*: evento avverso grave (es. ospedalizzazione, rischio vita, decesso).

OHE *One-Hot Encoding*: codifica binaria delle variabili categoriche.

SMOTE *Synthetic Minority Over-sampling Technique*: tecnica di oversampling sintetico della classe minoritaria.

MedDRA *Medical Dictionary for Regulatory Activities*: vocabolario standard per la codifica dei sintomi/eventi avversi.

Capitolo 7

Mappa sintetica di notebook e script

Asset	Ruolo principale
`01_cleaning.ipynb`	Pulizia dati, unione sintomi e gestione `IS_SEVE`
`02_encoding.ipynb`	One-hot encoding su sesso e produttore.
`03_nlp_cross.ipynb`	Feature NLP (gruppi semantici) e interazioni.
`04_evaluation.ipynb`	Training finale LGBM, SMOTE e soglia.
`src/evaluation/test6_stacking.py`	Stacking ortogonale (LGBM + Linear + KNN
`src/evaluation/test7_lgbm_stacking.py`	Stacking con varianti LightGBM (GBDT/DAR
`src/evaluation/test8_recall_first_policy.py`	Policy di soglia recall-first con guardrail di prec
`src/evaluation/test9_retrain_recall70.py`	Campagna di retraining per target recall 0.70.
`src/evaluation/test10_hybrid_ensemble.py`	Confronto e ottimizzazione dell'ensemble ibrido stacking) con selezione soglia per obiettivo.
`src/evaluation/test5_advanced_model_test.py`	Benchmark finale candidati (`lgbm` `blend_weighted`, `lgbm_smote`) e scelta modello