Rilevamento delle Fasi nel Modello di Ising tramite Machine Learning

Progetto ML

February 17, 2026

1 Introduzione

Il Machine Learning (ML) è un campo di studio che consente ai computer di apprendere dai dati senza essere esplicitamente programmati. In questo progetto, applichiamo tecniche di ML a un problema classico della fisica statistica: il **Modello di Ising**.

Il Modello di Ising descrive il ferromagnetismo in materiali statistici. Il sistema è composto da variabili discrete di spin $s_i \in \{-1, +1\}$ disposte su un reticolo. L'obiettivo è classificare lo stato del sistema in una delle due fasi termodinamiche:

- **Fase Ordinata (Ferromagnetica)**: $T < T_c$, dove gli spin tendono ad allinearsi.
- **Fase Disordinata (Paramagnetica)**: $T > T_c$, dove l'agitazione termica domina e gli spin sono casuali.

Per un reticolo 2D quadrato, la temperatura critica teorica è $T_c \approx 2.269$.

2 Data Engineering

Come discusso nel capitolo sul Data Engineering, la qualità dei dati è fondamentale per il successo di un modello di ML. Abbiamo implementato una pipeline robusta per generare, corrompere e pulire i dati.

2.1 Generazione dei Dati (Ising Simulation)

Utilizziamo l'algoritmo **Metropolis-Hastings** per campionare configurazioni di equilibrio dalla distribuzione di Boltzmann. Per affrontare il problema del *Critical Slowing Down* (la tendenza del sistema a evolvere lentamente vicino alla temperatura critica), abbiamo implementato una **Termalizzazione Adattiva**:

```
if 2.0 < T < 2.5:
    steps = 5000  # Piu' passi vicino alla T critica per garantire
   equilibrio
else:
    steps = 1000  # Meno passi lontano dalla criticita'
```

Listing 1: Termalizzazione Adattiva in data_generator.py

Il dataset è bilanciato e consiste in:

- 1000 matrici a bassa temperatura ($T < 2.0$, Classe 0).
- 1000 matrici ad alta temperatura ($T > 2.5$, Classe 1).
- 500 matrici vicino alla temperatura critica ($T \approx 2.27$) per il test.

2.2 Simulazione Sensori e Corruzione Dati

Simuliamo un ambiente reale in cui i sensori possono fallire. Introduciamo due tipi di difetti:

1. **Missing Values (MCAR)**: Il 5% dei pixel viene perso (impostato a 'NaN'). Assumiamo che i dati manchino in modo completamente casuale (*Missing Completely At Random*).
2. **Outliers**: Il 2% dei pixel registra valori errati (es. 50), fisicamente impossibili dato che lo spin deve essere ± 1.

2.3 Data Cleaning

Nel modulo `data_cleaning.py`, implementiamo strategie per ripristinare i dati:

Gestione Outlier: Rileviamo valori anomali utilizzando una soglia fisica. Poiché $|s_i| = 1$, qualsiasi valore con $|x| > 1.5$ viene considerato un errore del sensore e marcato come mancante ('NaN') per essere successivamente imputato.

Imputazione (Imputation): Per i valori mancanti, utilizziamo una strategia di imputazione basata sulla moda (*Most Frequent*).

```
# Da data_cleaning.py
imputer = SimpleImputer(strategy='most_frequent')
flat_imputed = imputer.fit_transform(flat_data)
```

Questa scelta è giustificata dalla natura discreta degli spin: se la maggioranza dei campioni ha valore $+1$, è probabile che anche il dato mancante sia $+1$.

3 Feature Engineering

Esploriamo due approcci distinti per rappresentare i reticoli di Ising, come descritto nelle pratiche di Feature Engineering.

3.1 Approccio 1: Physics-Based (Domain Knowledge)

Sfruttiamo la conoscenza del dominio fisico per estrarre feature globali informative. Calcoliamo due osservabili termodinamiche fondamentali:

1. **Magnetizzazione Media (M)**: Parametro d'ordine del sistema.

$$M = \frac{1}{N} \left| \sum_i s_i \right|$$

2. **Energia Media (E)**: Hamiltoniana del sistema (interazione tra primi vicini).

$$E = -\frac{1}{2N} \sum_{\langle i,j \rangle} s_i s_j$$

Queste feature riducono la dimensionalità da $16 \times 16 = 256$ a sole 2 dimensioni, rendendo il modello estremamente interpretabile.

3.2 Approccio 2: Raw Data (Data-Driven)

In questo approccio, non facciamo assunzioni fisiche. Appiattiamo l'intera matrice del reticolo in un vettore di feature:

$$\text{Shape} : (N, 16, 16) \to (N, 256)$$

Questo permette al modello di apprendere autonomamente pattern spaziali complessi, sebbene a costo di una maggiore complessità computazionale e minore interpretabilità ("Black Box").

4 Model Development

Abbiamo selezionato i modelli in base alla natura delle feature, seguendo i principi di selezione del modello (Occam's Razor, Trade-off Bias-Variance).

4.1 Modelli Selezionati

- **Logistic Regression** (per Feature Fisiche): Dato che la transizione di fase è ben definita nello spazio (Magnetizzazione, Energia), un modello lineare semplice è sufficiente ed efficace.
- **Random Forest** (per Raw Data): Per gestire la complessità dei 256 pixel grezzi e le loro interazioni non lineari, utilizziamo un modello basato su alberi (Random Forest) con 100 estimatori.

5 Risultati Sperimentali e Valutazione

5.1 Accuratezza e Critical Slowing Down

Valutiamo i modelli non solo con l'accuratezza globale, ma analizzando le prestazioni in funzione della temperatura. Come atteso, osserviamo una **Critical U-Shape**:

- Lontano da T_c, l'accuratezza è vicina al 100%.
- Vicino a $T_c \approx 2.27$, l'accuratezza cala drasticamente. Questo riflette la fisica del sistema: al punto critico, le fluttuazioni sono massime e le due fasi sono difficilmente distinguibili.

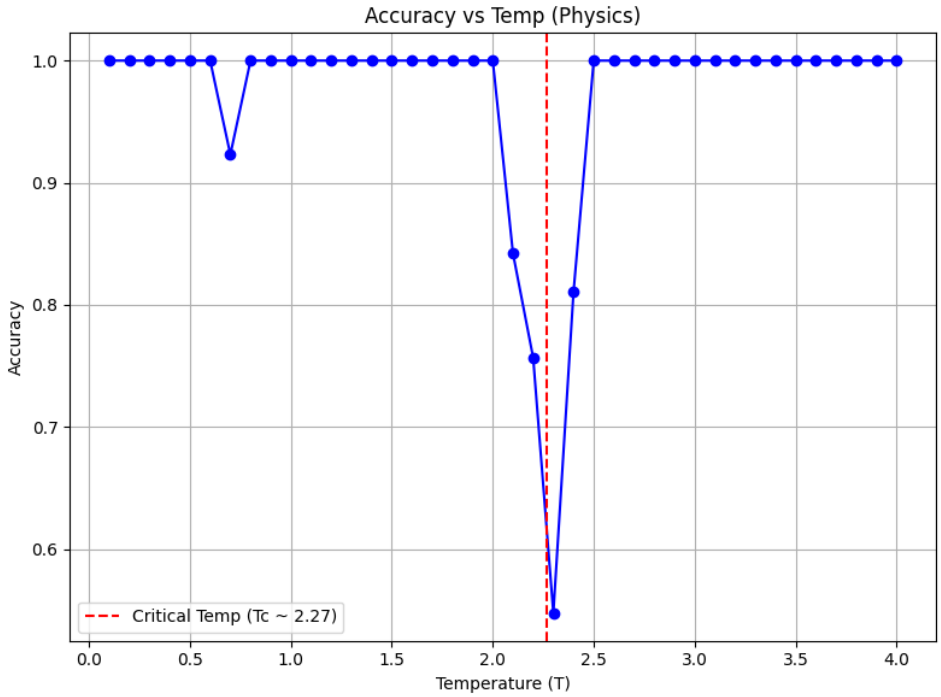

Figure 1: Accuratezza vs Temperatura (Modello Physics-Based).

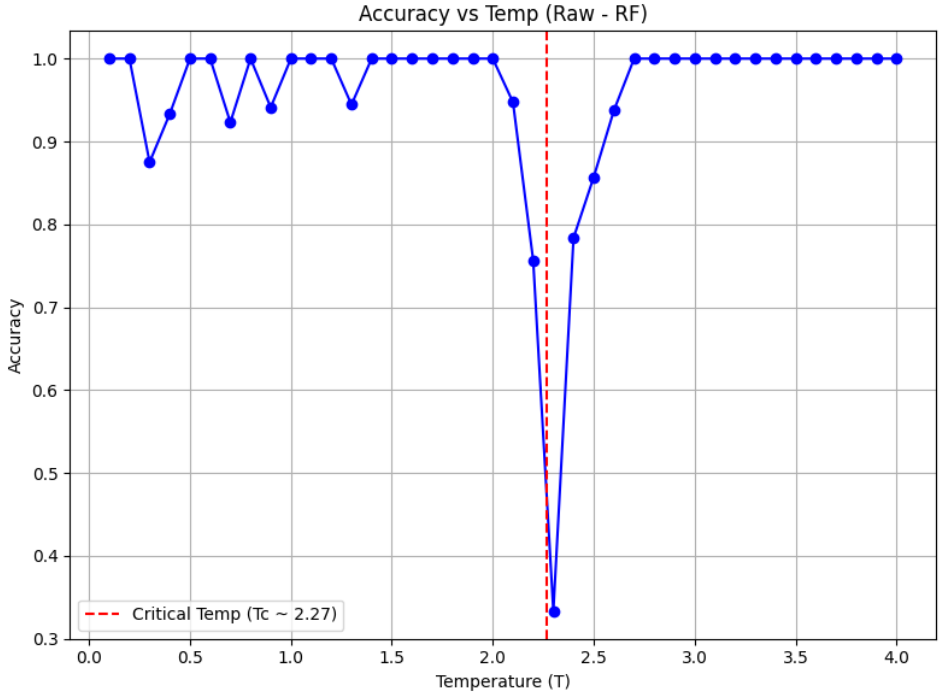

Figure 2: Accuratezza vs Temperatura (Modello Raw Data).

5.2 Matrice di Confusione

La matrice di confusione ci aiuta a visualizzare i Falsi Positivi/Negativi, fondamentali per capire se il modello è sbilanciato verso una delle due fasi.

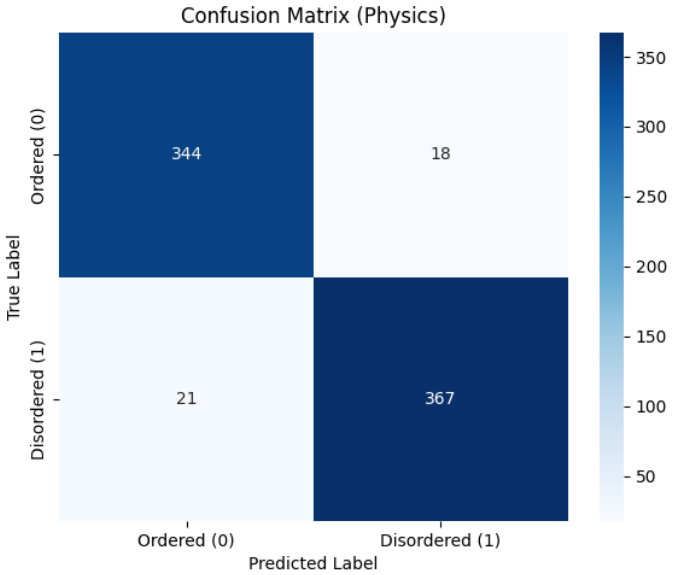

Figure 3: Matrice di Confusione (Modello Physics-Based).

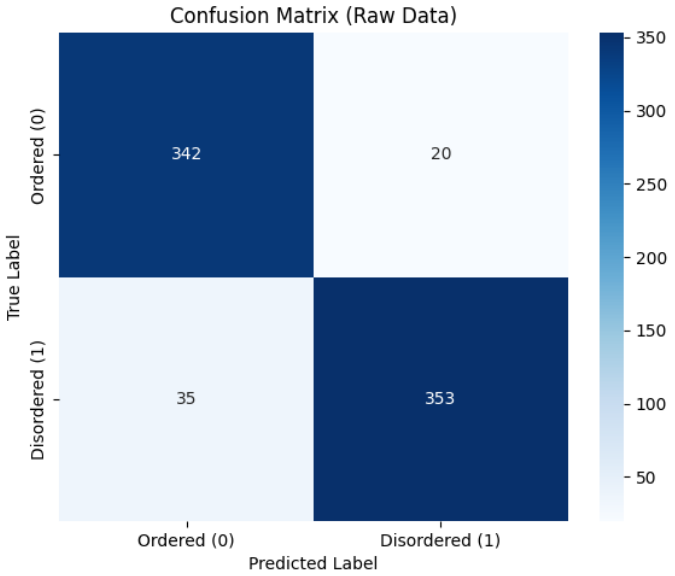

Figure 4: Matrice di Confusione (Modello Raw Data).

5.3 Curva ROC

La curva ROC (Receiver Operating Characteristic) mostra la capacità del modello di discriminare tra le due classi al variare della soglia di decisione. L'Area Sotto la Curva (AUC) è una metrica di performance aggregata: un valore di 1.0 indica un classificatore perfetto.

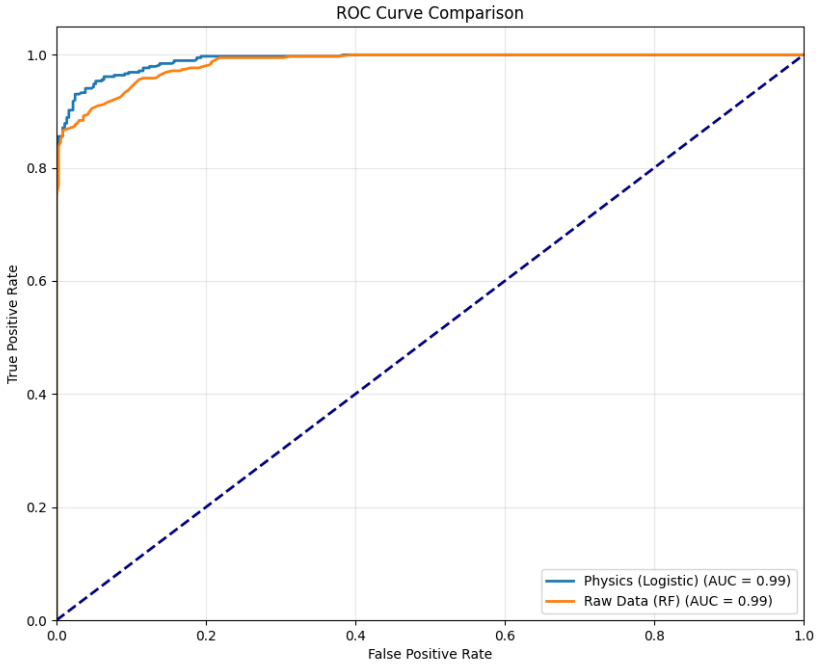

Figure 5: Confronto Curve ROC: Physics-Based vs Raw Data.

6 Applicazione Interattiva

Il progetto include una Web App sviluppata con **Streamlit** (`app.py`) per l'inferenza real-time. Funzionalità principali:

- Simulazione live di reticoli di Ising a temperatura variabile.
- Calcolo dinamico di Magnetizzazione ed Energia.
- Visualizzazione delle predizioni del modello ML vs Temperatura teorica.