UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Corso di Laurea Magistrale in Ingegneria
Informatica

Dipartimento di Ingegneria Elettrica e delle
Tecnologie dell'Informazione

Information Systems and Business Intelligence

Trend Analysis - Agritech Domain

Candidati:

Vincenzo D'Angelo M63001595
Giorgio Di Costanzo M63001579

Professoressa:

Flora Amato

Anno Accademico 2024/2025

Indice

1 Introduzione

Prevedere e comprendere le popolazioni di insetti è un compito di primaria importanza. In ambito agricolo la previsione di parassiti e specie impollinatrici può migliorare la resa, ridurre l'uso di pesticidi e promuovere pratiche sostenibili. L'obiettivo principale di questa iniziativa è sviluppare un modello predittivo in grado di stimare con precisione il numero di insetti, utilizzando dati ambientali giornalieri relativi alla temperatura e all'umidità.
I due task assegnati riguardano un problema di **regressione** e uno di **classificazione**.

L'obiettivo del problema di regressione è prevedere il numero totale di insetti catturati in un determinato periodo, basandosi su variabili indipendenti come temperatura, umidità, eventi e altre caratteristiche ambientali o temporali. Questo approccio mira a fornire una stima quantitativa precisa del numero di catture, utile per analisi approfondite o per la pianificazione di interventi.
Il problema di classificazione si concentra sull'identificazione della presenza di nuove catture in un determinato periodo, trattandolo come una variabile binaria o categoriale (es., 1 = nuove catture, 0 = nessuna nuova cattura). L'obiettivo è determinare se un evento specifico porterà o meno alla cattura di insetti, basandosi su caratteristiche ambientali e condizioni precedenti. Questo approccio è utile per monitorare dinamiche di cattura e individuare condizioni favorevoli o sfavorevoli.

Il dataset fornito è suddiviso in più file Excel, ciascuno dei quali contiene i dati raccolti da sensori installati nelle città di Imola e Cicalino. Ogni file riporta informazioni relative al conteggio giornaliero di insetti, alla temperatura media, all'umidità media e all'intervallo di temperatura, rendendo possibile l'analisi separata delle dinamiche locali delle popolazioni di insetti in ciascuna città.
I dati ricoprono complessivamente i mesi di luglio e agosto, ma non in maniera omogenea. Alcune città presentano periodi di raccolta più estesi o meglio distribuiti, mentre altre hanno dati frammentati o concentrati in specifici intervalli temporali. Questa disomogeneità rende difficile un'analisi congiunta senza perdere informazioni preziose e non offre una base per l'identificazione di trend stagionali.

Per l'implementazione e lo sviluppo del modello predittivo, è stato utilizzato Google Colab [1], una piattaforma basata su cloud che consente di eseguire codice Python in modo interattivo, sfruttando l'accesso gratuito a GPU e risorse computazionali avanzate.
Costruire un modello separato per ogni città è giustificato dai seguenti punti:

1. Dati non coincidenti temporalmente: Se i dati delle città non condividono lo stesso intervallo temporale o hanno gap temporali diversi, cercare di uniformarli eliminando informazioni non coincidenti comporterebbe una significativa perdita di dati. Per dataset già di piccole dimensioni, questa

eliminazione potrebbe ridurre la capacità del modello di catturare pattern rilevanti. In altre parole, mantenendo un modello specifico per ogni città, si utilizza il massimo delle informazioni disponibili per ciascun contesto, senza dover forzare una sincronizzazione temporale che ridurrebbe i dati utili.

2. Eterogeneità locale: Ogni città può avere condizioni climatiche uniche, come variazioni di temperatura e umidità, che influenzano in modo specifico le dinamiche delle popolazioni di insetti. Utilizzare un modello unico potrebbe non catturare adeguatamente queste variazioni locali, compromettendo l'accuratezza delle previsioni. Le popolazioni di insetti possono rispondere in modo diverso agli stessi fattori ambientali a seconda dell'area geografica. Fattori come l'adattamento locale, la presenza di predatori naturali o di altre specie concorrenti variano da città a città e rendono necessaria una modellazione personalizzata.

Modelli separati consentono di analizzare con maggiore precisione le variabili che influenzano le popolazioni di insetti in una città specifica, fornendo informazioni più utili per interventi mirati e strategie locali. Inoltre, sono più facili da aggiornare qualora i dati di una specifica città vengano ampliati o modificati, senza influenzare le altre città. Questo approccio migliora la scalabilità del progetto.

Si è deciso di affrontare il problema di regressione utilizzando il dataset di Cicalino, mentre il problema di classificazione è stato affrontato utilizzando il dataset di Imola.

2 Data loading and Inspection

I file erano inizialmente in formato HTML e sono stati successivamente convertiti in CSV utilizzando la libreria *BeautifulSoup*. Questo processo ha permesso di estrarre e strutturare i dati in un formato facilmente leggibile e utilizzabile per l'analisi. È stato sostituito l'indice predefinito del DataFrame con la colonna `DateTime`. Questo migliora la gestione temporale dei dati e consente di eseguire operazioni più efficaci su valori basati sul tempo, come ordinamenti e analisi di serie temporali. Le colonne chiave, come DateTime, sono state convertite in un formato adeguato, come il tipo datetime di pandas, per garantire che tutte le operazioni basate sul tempo (come filtri temporali, calcoli di intervalli, e resampling) possano essere eseguite correttamente.

2.1 Dati Meteo

Questo dataset contiene dati meteorologici organizzati su base oraria.

File	**Intervallo temporale**
Cicalino 1	Dal 5 luglio al 23 agosto
Cicalino 2	Dall'11 luglio al 23 agosto
Imola 1	Dal 30 luglio al 23 agosto
Imola 2	Dal 30 luglio al 23 agosto
Imola 3	Dal 30 luglio al 23 agosto

Tabella 1: File del dataset e intervalli temporali coperti

Colonna	**Descrizione**
`DateTime`	Timestamp che identifica giorno e ora di ciascuna rilevazione.
`Media Temperatura`	Temperatura media registrata in quell'ora.
`Temperatura Intervallo (low, high)`	Range di temperatura registrata (minima e massima).
`Media Umidità`	Percentuale media di umidità relativa durante l'ora.

Tabella 2: Colonne principali del dataset

I dati sono registrati a intervalli orari, consentendo un'analisi dettagliata delle variazioni meteorologiche durante la giornata. Non sono presenti valori mancanti. Di seguito si riportano i grafici utilizzati per analizzare le differenze tra i due dataset.

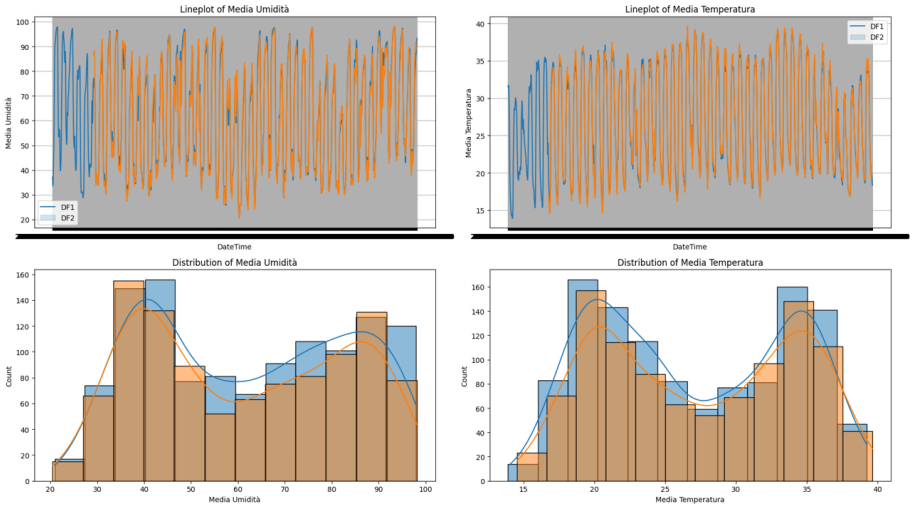

Figura 1: Dati Cicalino

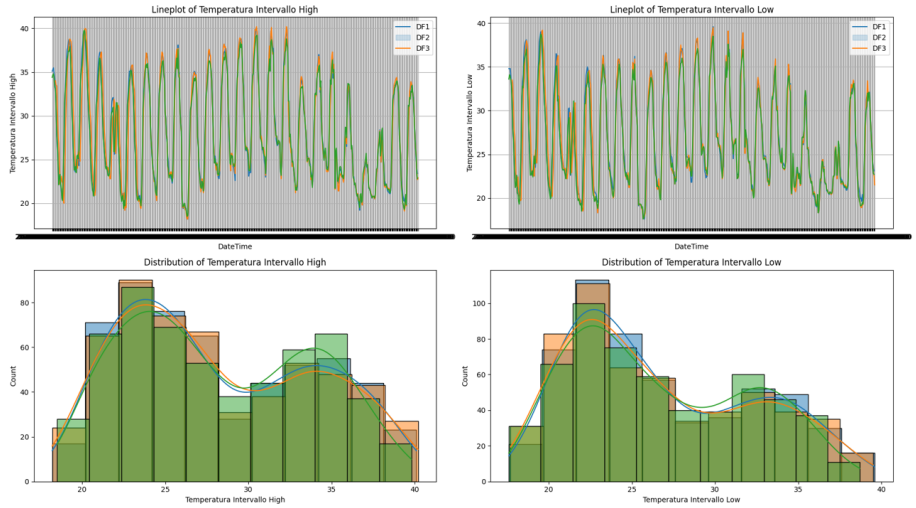

Figura 2: Dati Imola

Gli andamenti temporali (grafici superiori) mostrano poche discrepanze nei livelli medi generali di umidità e temperatura, mentre gli istogrammi (grafici inferiori) mostrano distribuzioni delle medie di temperatura e umidità simili, con picchi e varianza paragonabili.

Sulla base di queste considerazioni, si è deciso di accorpare le informazioni dei due dataset.

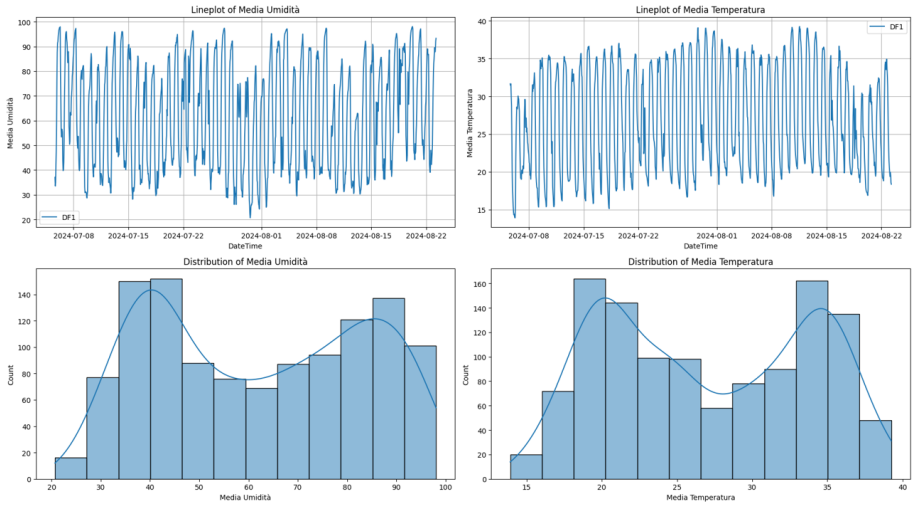

Figura 3: Dati Cicalino

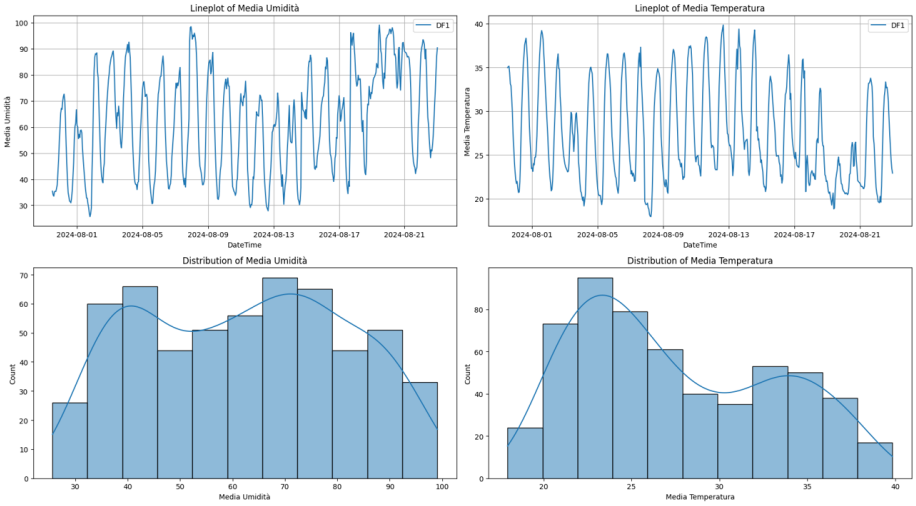

Figura 4: Dati Imola

I grafici seguenti mostrano la distribuzione media di temperatura e umidità per ogni ora del giorno.

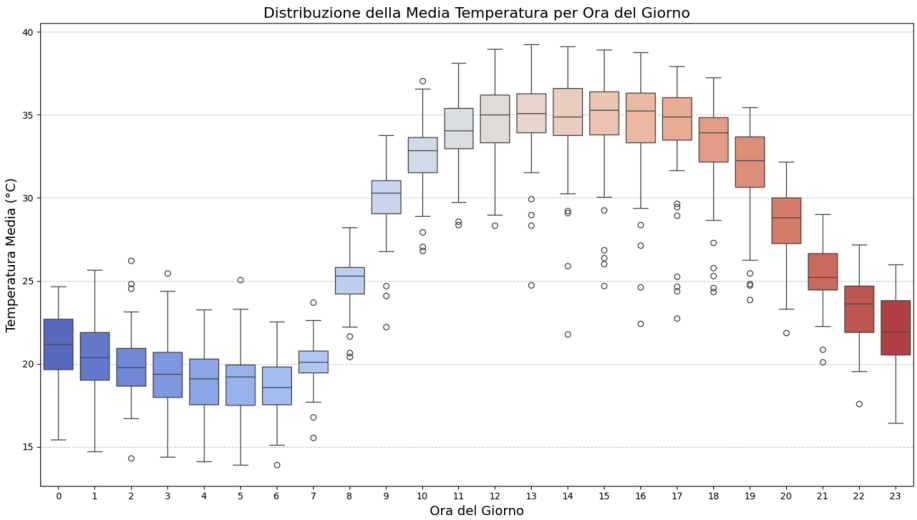

Figura 5: Distribuzione temperatura media per ora del giorno (Cicalino)

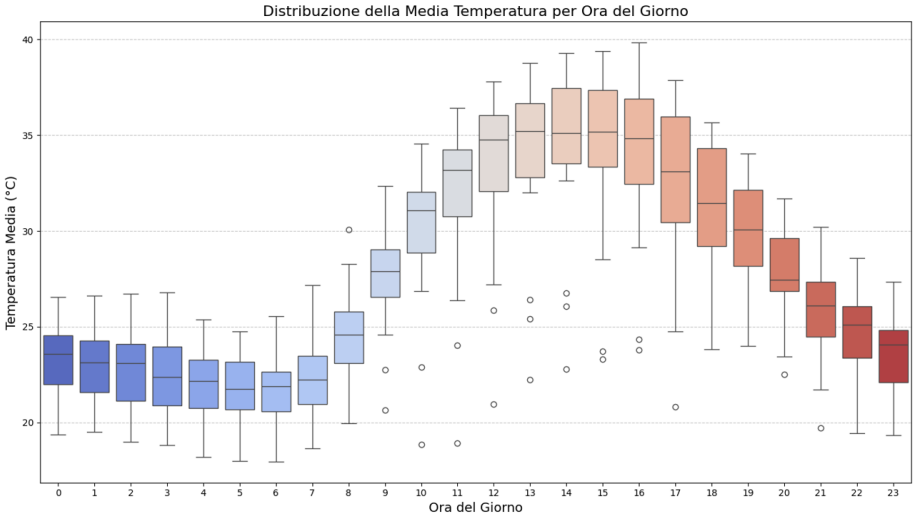

Figura 6: Distribuzione temperatura media per ora del giorno (Imola)

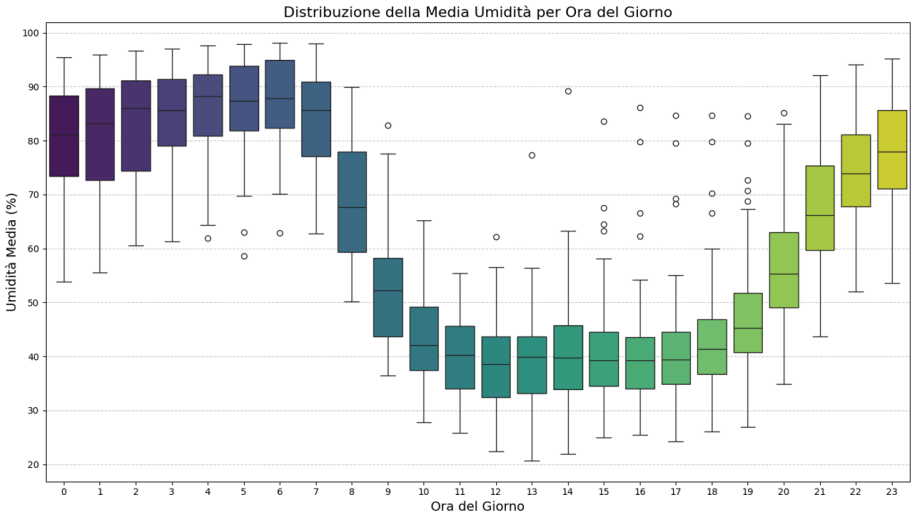

Figura 7: Distribuzione umidità media per ora del giorno (Cicalino)

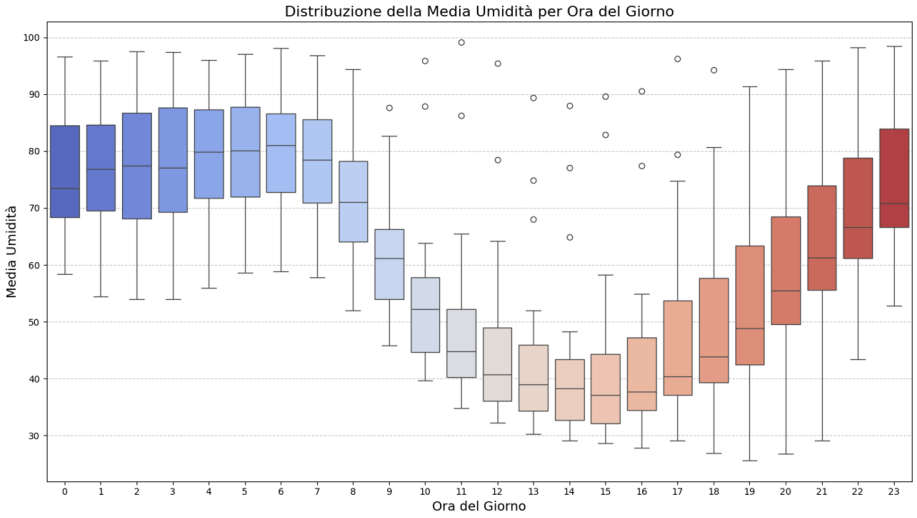

Figura 8: Distribuzione umidità media per ora del giorno (Imola)

Dal momento che i dati sulle catture sono forniti su base giornaliera, si sceglie di accorpare i dati meteorologici relativi allo stesso giorno. In questa aggregazione, si conservano le informazioni relative alla media e alla varianza di temperatura e umidità, in modo da mantenere sia una misura centrale sia una misura della variabilità per ciascun giorno. In più si sceglie di salvare la temperatura massima e minima di ogni giornata.
Questo approccio permette di ridurre la granularità temporale dei dati meteo senza perdere dettagli significativi sulla loro distribuzione giornaliera.

In questo passaggio, l'indice del dataset, che originariamente include informazioni temporali dettagliate come la data e l'orario, viene modificato per mantenere solo la data.

2.2 Dati Catture

Il dataset riportato rappresenta un registro delle catture di insetti in un intervallo temporale, con alcune informazioni aggiuntive per ogni osservazione.

File	**Intervallo temporale**
Cicalino 1	Dal 6 luglio al 23 agosto
Cicalino 2	Dal 5 luglio al 23 agosto
Imola 1	Dal 30 luglio al 23 agosto
Imola 2	Dal 31 luglio al 23 agosto
Imola 3	Dal 17 agosto al 23 agosto

Tabella 3: File del dataset e intervalli temporali coperti

Colonna	**Descrizione**
`DateTime`	Timestamp della rilevazione, con data e ora precise di ogni osservazione.
`Numero di insetti`	Numero totale di insetti catturati fino a quel momento (variabile cumulativa).
`Nuove catture (per evento)`	Numero di insetti catturati tra un'osservazione e la successiva (valore incrementale).
`Recensito`	Indica se l'osservazione è stata verificata o validata (valore "Si").
`Evento`	Annotazioni sugli eventi che possono aver influenzato i dati (es. "Cleaning" per pulizia trappole).

Tabella 4: Colonne principali del dataset

Le osservazioni sono prese a intervalli regolari, ma con leggere discrepanze nei minuti (es. 06:01:00, 06:04:00, ecc.).

L'evento `Cleaning` segnala l'azione di pulizia delle gabbie. Questo evento potrebbe influenzare il numero di catture successive (ad esempio, riducendo gli insetti per un periodo). Per semplificare l'analisi e rendere i dati più adatti a modelli numerici, la colonna è stata convertita in formato binario.

L'orario non costituisce un'informazione rilevante per l'analisi, poiché i dati sulle catture sono forniti con una cadenza giornaliera costante, sempre alla stessa ora, salvo leggere discrepanze nei minuti che non influenzano il risultato finale. Per questo motivo, si è deciso di eliminare l'orario dall'indice, mantenendo esclusivamente la componente della data, al fine di semplificare la gestione dei dati e la focalizzazione sull'aggregazione giornaliera.

La colonna `Recensito` presenta una varianza nulla, ovvero tutti i valori sono uguali. Questo implica che non fornisce informazioni utili per l'analisi o per i modelli predittivi, poiché non contribuisce a spiegare alcuna variabilità nei dati. Di conseguenza, si è deciso di eliminarla dal dataset.

I due dataset relativi alle catture sono stati inizialmente uniti tra loro, creando un unico dataset comprensivo di tutte le informazioni sulle catture. Successivamente, questo dataset è stato combinato con il dataset meteorologico, utilizzando come chiave comune la data. Questo processo ha permesso di ottenere il dataset finale, che integra sia i dati sulle catture sia le informazioni meteorologiche per ciascun giorno, garantendo una visione completa e coerente per l'analisi e la modellazione.

3 Problema di Regressione

Il problema di regressione è stato affrontato utilizzando i dataset relativi a *Cicalino*, con l'obiettivo di prevedere il **numero di insetti catturati** all'interno della trappola. La colonna target è una variabile intera, che rappresenta il conteggio degli insetti catturati giornalmente, e costituisce il valore da stimare attraverso i modelli predittivi.

3.1 Exploratory Data Analysis

Si procede quindi con un'analisi visivo-quantitativa, che integra tecniche statistiche con visualizzazioni grafiche, per esplorare in modo più approfondito i pattern, le relazioni e le caratteristiche dei dati. Questa fase comprende:

- **Analisi della variabilità**
  Box plot o violin plot, per confrontare la variabilità e le differenze tra gruppi o condizioni specifiche.
- **Relazioni tra variabili**
  Heatmap di correlazione, per calcolare e rappresentare graficamente le correlazioni tra variabili.
- **Valutazione multicollinearità**
  il Variance Inflation Factor (VIF) misura quanto la varianza stimata di un coefficiente di regressione è aumentata a causa della correlazione tra le variabili predittive. Un valore di VIF maggiore di 5 o 10 (a seconda della soglia scelta) indica una forte multicollinearità, suggerendo che alcune variabili potrebbero essere ridondanti. L'analisi del VIF consente di identificare e, se necessario, rimuovere o trasformare le variabili altamente correlate, migliorando l'affidabilità del modello.
- **Analisi delle distribuzioni**
  per valutare la distribuzione delle variabili e identificare eventuali asimmetrie o outlier.

Questa combinazione di metodi visivi e quantitativi consente di approfondire la comprensione dei dati, identificando relazioni e strutture complesse che potrebbero non essere immediatamente evidenti.

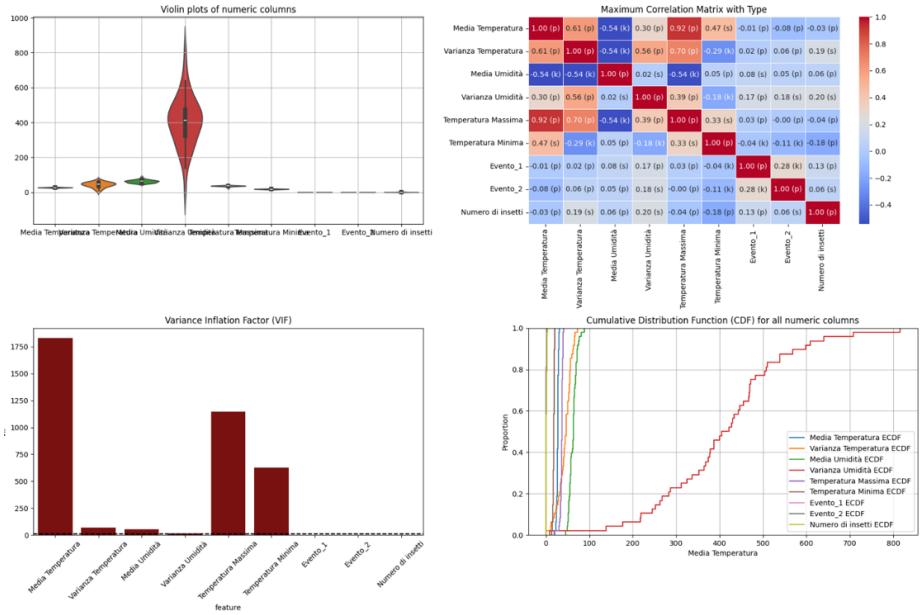

Figura 9: Analisi visivo-quntitativa

Analizzando i grafici e le correlazioni tra le variabili, si è deciso di eliminare le feature `Media Temperatura`, `Evento 2` e `Media Umidità` poiché risultano poco correlate alla variabile target (`numero di catture`). La loro bassa correlazione implica che non forniscono informazioni significative per il modello predittivo e potrebbero introdurre rumore.

Inoltre, si è scelto di applicare la Tecnica di Analisi delle Componenti Principali (PCA) sulle feature `Varianza Temperatura` e `Varianza Umidità`, che risultano essere più correlate alla variabile target. La PCA permette di ridurre la dimensionalità dei dati, combinando queste due variabili in una o più componenti principali, mantenendo la maggior parte dell'informazione e riducendo la multicollinearità. In questo modo, si ottimizza l'uso delle variabili più informative e si migliora l'efficienza del modello.

Si tiene inoltre traccia della somma delle varianze dei tre giorni precedenti. Questa scelta consente di includere nel modello un'informazione sintetica sulla variabilità recente della temperatura, che potrebbe avere un impatto significativo sul fenomeno osservato.

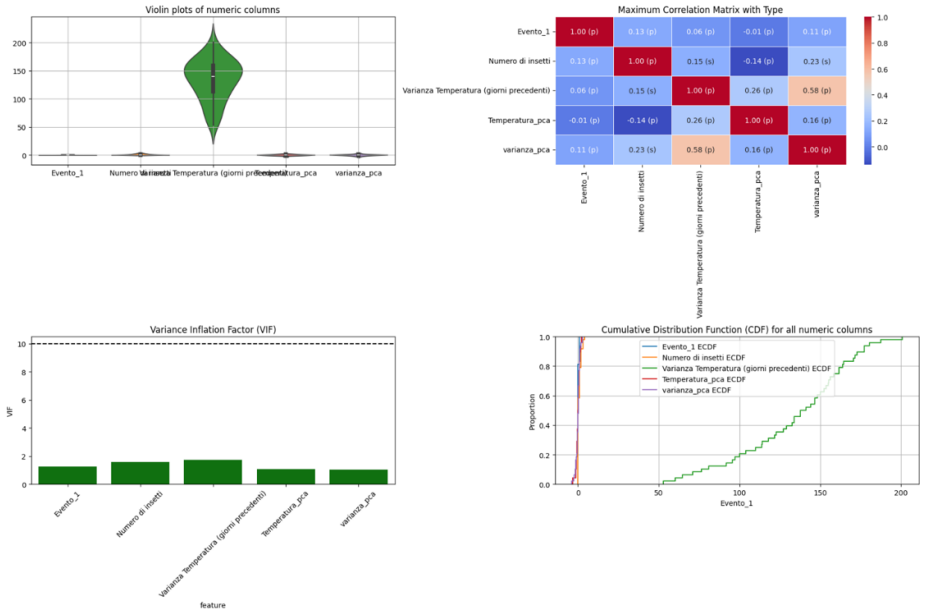

Figura 10: Analisi visivo-quntitativa

Una volta comprese le basi della serie temporale, il passo successivo è approfondire l'analisi tramite vari grafici e tecniche per esplorare le caratteristiche temporali dei dati. Tra gli strumenti più utilizzati, troviamo i grafici di autocorrelazione e partial autocorrelation, la scomposizione stagionale e i time series plot.

- *Autocorrelation Function (ACF)*: Mostra la correlazione tra una variabile e se stessa in vari lag temporali. Permette di capire se i valori passati influenzano i valori futuri, e di identificare la stagionalità o il trend nei dati.

- *Partial Autocorrelation Function (PACF)*: Aiuta a identificare l'effetto diretto di un lag sui dati, eliminando gli effetti indiretti. Questo è utile per determinare l'ordine del modello ARIMA (AutoRegressive Integrated Moving Average) nella modellazione delle serie temporali.

- *Seasonal Decomposition*: La scomposizione stagionale separa la serie temporale in tre componenti principali:

  1. Trend: L'andamento a lungo termine della variabile.
  2. Seasonality: La variazione periodica che si ripete a intervalli regolari (ad esempio, le fluttuazioni stagionali nelle catture).
  3. Residui: La parte dei dati che non è spiegata dal trend o dalla stagionalità.

La decomposizione aiuta a capire meglio come ciascuna di queste componenti contribuisce al comportamento complessivo della serie temporale e rende più semplice la modellazione.

- *Time Series Plot*: Il time series plot è un grafico lineare che mostra i dati nel tempo, con l'asse delle ascisse rappresentante il tempo (giorni, mesi, ecc.) e l'asse delle ordinate rappresentante la variabile osservata (ad esempio, numero di catture). Questo grafico aiuta a visualizzare chiaramente i trend, le stagionalità e le fluttuazioni temporali. È fondamentale per identificare eventuali picchi o anomalie nei dati, oltre che per comprendere la dinamica temporale complessiva.

Questi strumenti permettono di eseguire un'analisi visiva e statistica approfondita delle serie temporali, facilitando la comprensione delle dinamiche temporali e la preparazione per le successive fasi di modellazione e previsione. sul fenomeno osservato.

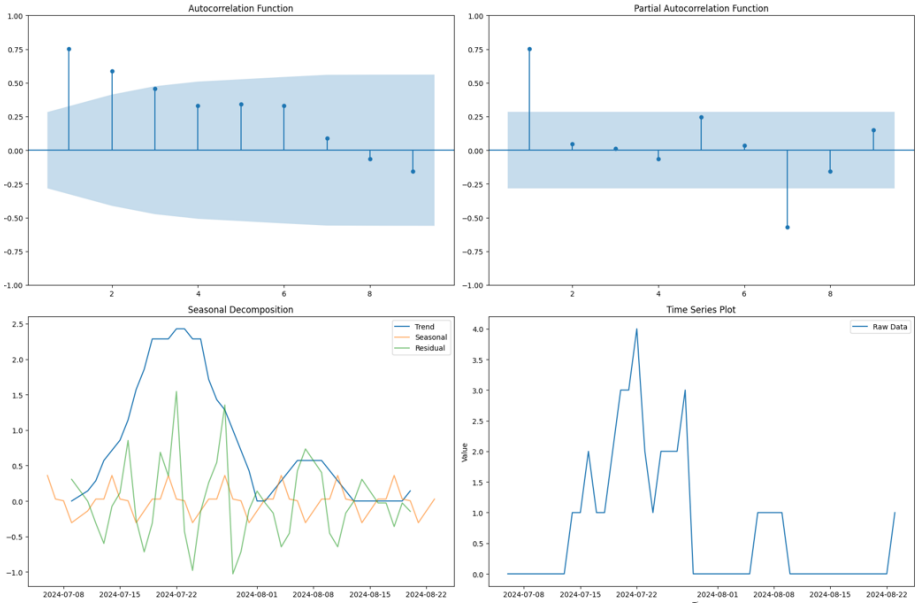

Figura 11: Analisi delle serie temporali

Come si può osservare, l'autocorrelazione è significativa per diversi lag iniziali perché i punti si trovano al di fuori della banda blu di confidenza. Il decadimento è graduale, suggerendo che la serie potrebbe non essere completamente stazionaria. Potrebbe essere necessario differenziare la serie una volta (d=1) per renderla stazionaria.

L'ACF fornisce indizi utili per determinare il parametro q (componente MA) del modello ARIMA. Il primo lag ha una correlazione significativa e positiva (ben

al di fuori della banda blu), con poche altre correlazioni significative nei lag successivi. Questo suggerisce un modello autoregressivo (AR) di basso ordine.

La componente trend (linea blu nel grafico in basso a sinistra) mostra un aumento iniziale, seguito da una stabilizzazione e un successivo calo. La componente stagionale (linea arancione) presenta un pattern oscillante e ben definito, indicando una stagionalità regolare. I residui (linea verde) sembrano mostrare variabilità casuale senza un chiaro pattern strutturale. La serie ha una componente stagionale evidente.

La serie temporale originale mostra un comportamento variabile, con un aumento marcato seguito da un calo brusco e periodi di valori stabili. Questo andamento non è stazionario, come suggerito dall'ACF e dalla decomposizione. Differenziare la serie potrebbe eliminare il trend, rendendola adatta per il modello ARIMA.

3.2 Addestramento

Per l'addestramento del modello, si è deciso di utilizzare e confrontare tre approcci distinti, ognuno con i propri vantaggi e caratteristiche:

1. **Modello Statistico: ARIMAX**

   - Componenti autoregressive (AR): che catturano la dipendenza tra i dati passati e i dati futuri.
   - Componenti di media mobile (MA): che catturano l'effetto di shock temporanei.
   - Differenziazione (I): per rendere la serie stazionaria.
   - Variabili esogene (X): come variabili meteorologiche o altre informazioni esterne che possono influenzare la variabile target.

   Questo approccio è utile per modellare serie temporali con una forte dipendenza dai dati passati e variabili esterne.

2. **Tecnica di Machine Learning: Ensemble Learning**

   L'Ensemble Learning combina i risultati di più modelli per migliorare le performance predittive. In particolare, si può utilizzare una combinazione di modelli come:

   - Random Forest: un algoritmo che costruisce molteplici alberi decisionali e combina i loro risultati per migliorare la previsione.
   - Gradient Boosting: un metodo che costruisce un modello forte combinando iterativamente modelli più deboli, migliorando progressivamente la previsione.

L'idea alla base dell'Ensemble Learning è quella di ridurre l'overfitting, migliorare la robustezza e ottenere predizioni più accurate combinando più modelli.

3. **Tecnica di Deep Learning: MLP con Cross Validation**

   Il Multilayer Perceptron (MLP) è una rete neurale artificiale costituita da più strati di neuroni (input, nascosti e output). Viene addestrato per apprendere le relazioni non lineari tra le variabili di input e la variabile target. L'uso di Cross Validation aiuta a valutare la performance del modello su diversi sottoinsiemi di dati, riducendo la possibilità di overfitting e migliorando la generalizzazione.
   Questo approccio è particolarmente utile per modelli complessi, dove le relazioni tra variabili sono difficili da catturare con modelli tradizionali. Il MLP può essere efficace per predire risultati non lineari, come nel caso delle catture influenzate da variabili meteorologiche.

Questi tre approcci permettono di esplorare diverse tecniche e di scegliere il modello più adatto per la previsione del numero di catture, in base alle caratteristiche specifiche dei dati e agli obiettivi del progetto.

3.2.1 Statistical Models

Si è optato per una suddivisione dell'80% per il training e del 20% per il test.

- Numero di lag autoregressivi: 4
- Differenziazione per rendere stazionario il segnale: 1
- Numero di termini della media mobile: 5

I risultati ottenuti sono i seguenti:

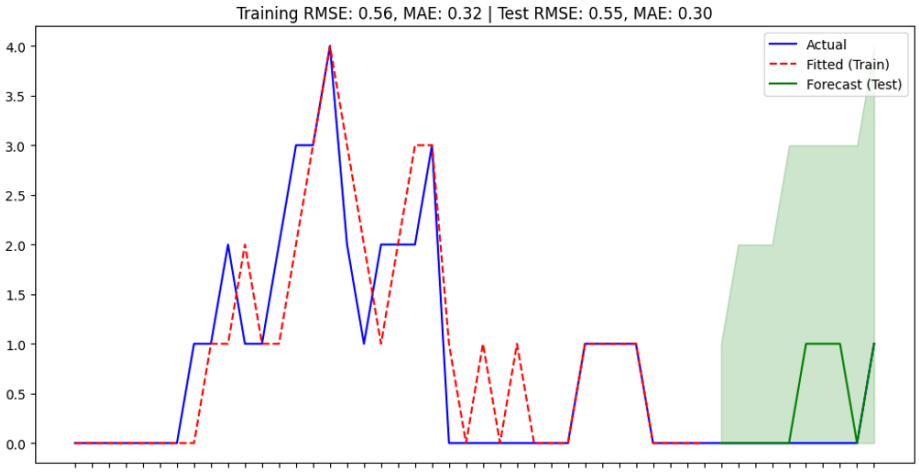

Figura 12: Modello ARIMAX - training e forecasting

3.2.2 Machine Learning Models

Per i modelli Gradient Boosting e Random Forest è stato effettuato il seguente spilt dei dati: 80% train e 20% test.

- Numero di lags: 2
- Numero di estimatori: 100
- Massima profondità (Gradient Boosting): 2
- Massima profondità (Random Forest): 3
- Learning rate (Gradient Boosting): 0,001

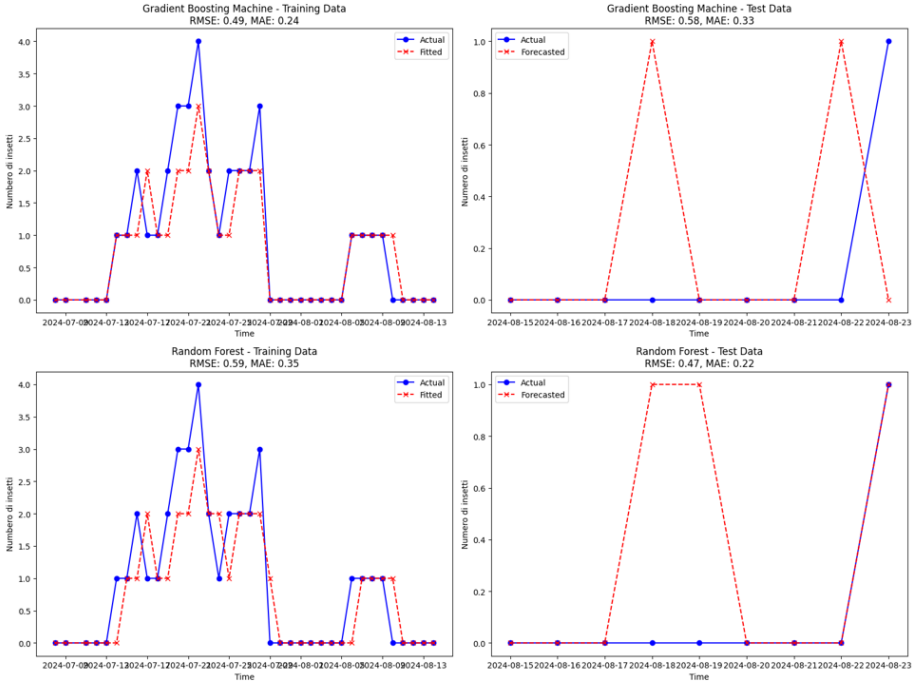

Figura 13: Gradient Boosting e Random Forest - training e forecasting

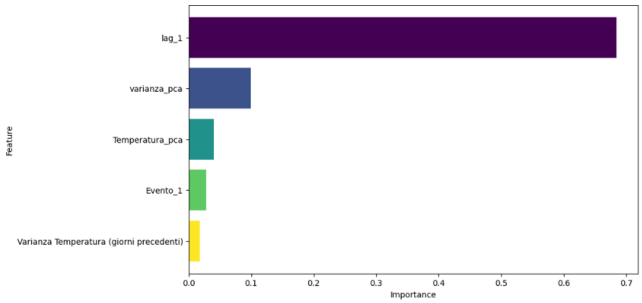

Figura 14: Feature importances (Gradient Boosting)

3.2.3 Deep Learning Models

Per il modello di Deep Learning è stato utilizzato un Multi-Layer Perceptron (MLP). La rete è stata strutturata nel seguente modo:

```
model_mlp = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(
        X_train_scaled.shape[1],),
                          kernel_regularizer=tf.keras.
                              regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.2),  # Dropout layer for
        regularization
    tf.keras.layers.Dense(64, activation='relu',
        kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.BatchNormalization(),  # Batch
        Normalization layer for normalizing activations
    tf.keras.layers.Dense(16, activation='relu',
        kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.2),  # Another Dropout layer
    tf.keras.layers.Dense(1, activation='linear')
])
```

Codice 1: Definizione modello MLP

Ogni layer Dense è seguito da una regolarizzazione L2. Essa aggiunge un termine al costo (loss) che penalizza i pesi troppo grandi, aiutando a prevenire l'overfitting e migliorando la generalizzazione del modello. Per gestire l'intensità della penalizzazione è stato scelto un valore moderato.
Sono inclusi due layer di Dropout nel modello con una probabilità di disattivazione di 0.2. Il dropout è una tecnica di regolarizzazione che "disattiva" casualmente una percentuale dei neuroni durante l'allenamento. Questo impedisce che la rete si adatti troppo ai dati di addestramento e aiuta a prevenire l'overfitting, migliorando la generalizzazione. La probabilità di 0.2 significa che il 20% dei neuroni verrà disattivato casualmente durante ogni passo di allenamento.
Si è fatto uso di Batch Normalization per normalizzare i valori di attivazione all'interno di ciascun batch, in modo da stabilizzare e accelerare l'apprendimento.
Il learning rate schedule usato definisce un Exponential Decay, che riduce il learning rate in modo esponenziale man mano che l'allenamento avanza. Ciò è utile per evitare che il modello faccia aggiornamenti troppo grandi dopo che ha raggiunto una buona parte della convergenza

Come funzione di loss, è stata utilizzata una MSE (Mean Squared Error). Si è tentato anche l'utilizzo di una sua versione pesata in modo da penalizzare meno gli errori in corrispondenza di assenza di catture, permettendo al modello di concentrarsi maggiormente sui dati più significativi. Come metrica è stato scelto il Mean Absolute Error (MAE) che misura la differenza assoluta media tra le predizioni e i valori reali.

I risultati ottenuti sono i seguenti:

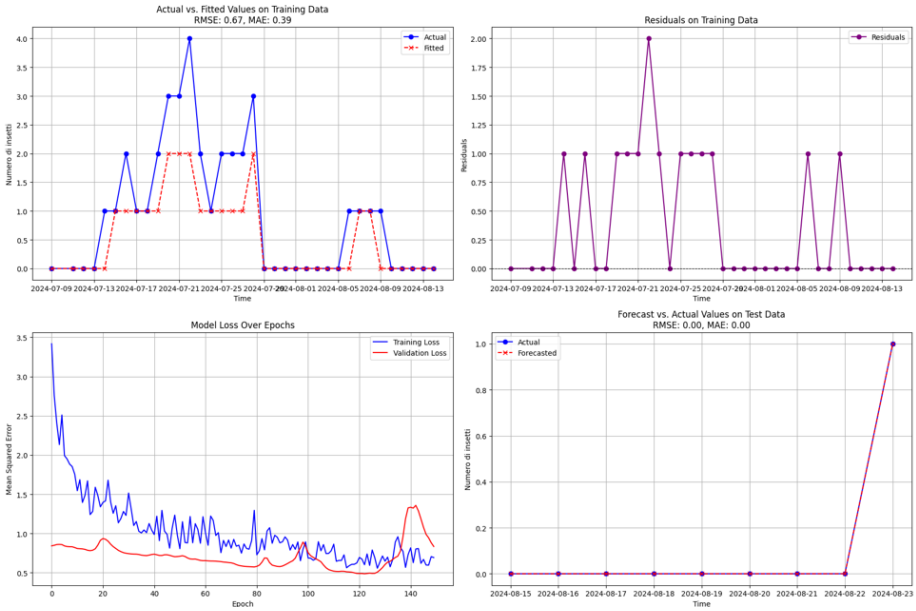

Figura 15: Modello MLP - training e forecasting

Risultati con la Cross Validation con 5 fold:

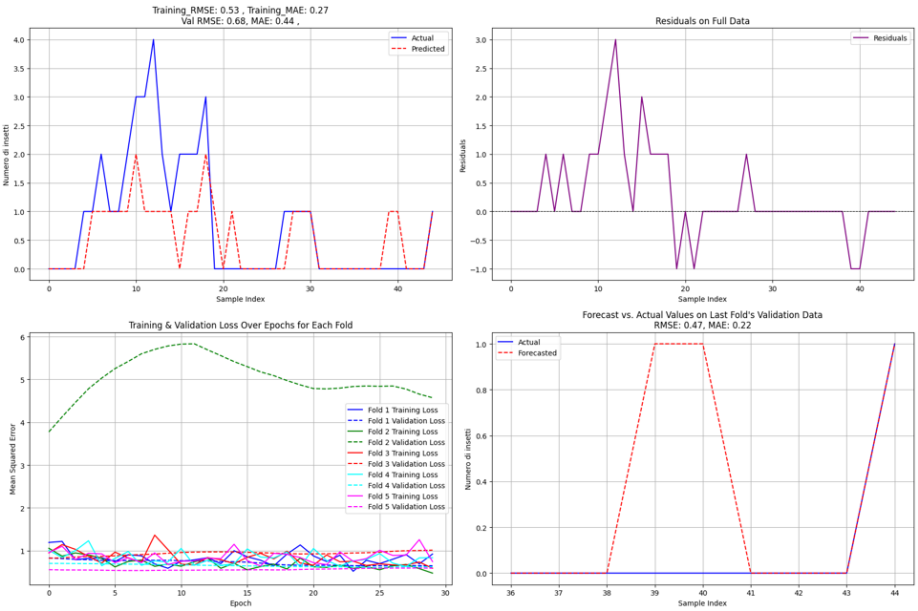

Figura 16: Modello MLP, cross validation - training e forecasting

4 Problema di Classificazione

Il problema di classificazione è stato affrontato utilizzando i dataset relativi a *Imola*. A differenza di quelli di Cicalino, i dataset di Imola sono tre: due di questi sono stati uniti per creare un dataset principale, mentre il terzo, contenendo pochi dati di catture, è stato utilizzato come test set. Questo approccio ha permesso di sfruttare il dataset più ampio per l'addestramento del modello, garantendo al contempo un test set separato per la validazione delle prestazioni del modello.
L'ultima istanza dei dataset è stata rimossa, poiché nei dati meteorologici relativi al giorno 23/08 era presente solo una singola fascia oraria. Si è osservato che mantenere tale istanza e aggiungere i valori mancanti comportava un cambiamento significativo nella distribuzione dei dati all'interno del dataset, motivo per cui si è scelto di escluderla.
Per quanto riguarda la colonna target, è stata trasformata da numerica intera a binaria secondo il seguente criterio: se c'è stata almeno una cattura, il valore è stato impostato a 1; altrimenti, a 0.

4.1 Exploratory Data Analysis

Si procede con un'analisi visiva e quantitativa, durante la quale sono stati generati i seguenti elementi: il Violin plot, la matrice di correlazione tra le variabili, il Variance Inflation Factor (VIF) e le Funzioni di Distribuzione Cumulativa (CDF) degli attributi.

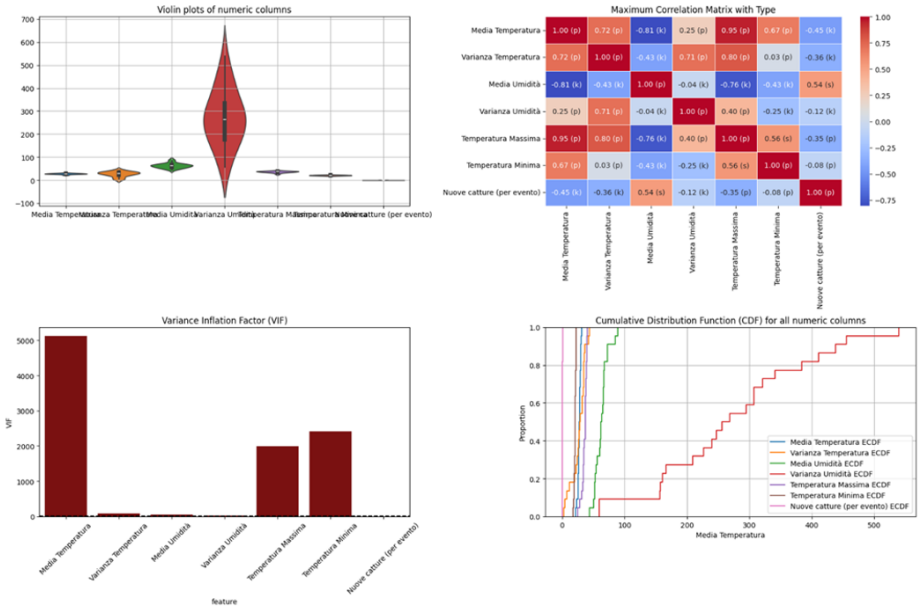

Figura 17: Analisi visivo-quntitativa

Dall'analisi dei grafici, in particolare della correlazione tra gli attributi e la variabile target, si è deciso di eliminare la feature `Temperatura minima`.

Successivamente, sono state create e accorpate nuove feature come segue:

- `Temperatura_pca`: è stata generata applicando la PCA sulle colonne `Media Temperatura` e `Temperatura Massima`.
- `Varianza_pca`: è stata generata applicando la PCA sulle colonne `Varianza Temperatura` e `Varianza Umidità`.

La colonna `Media Umidità` è stata mantenuta invariata, nonostante mostri una correlazione con altre colonne. Questo perché si è osservato durante l'addestramento (in particolare con l'algoritmo Random Forest) che mantenerla porta a risultati migliori rispetto alle altre scelte possibili.

Di seguito i risultati ottenuti:

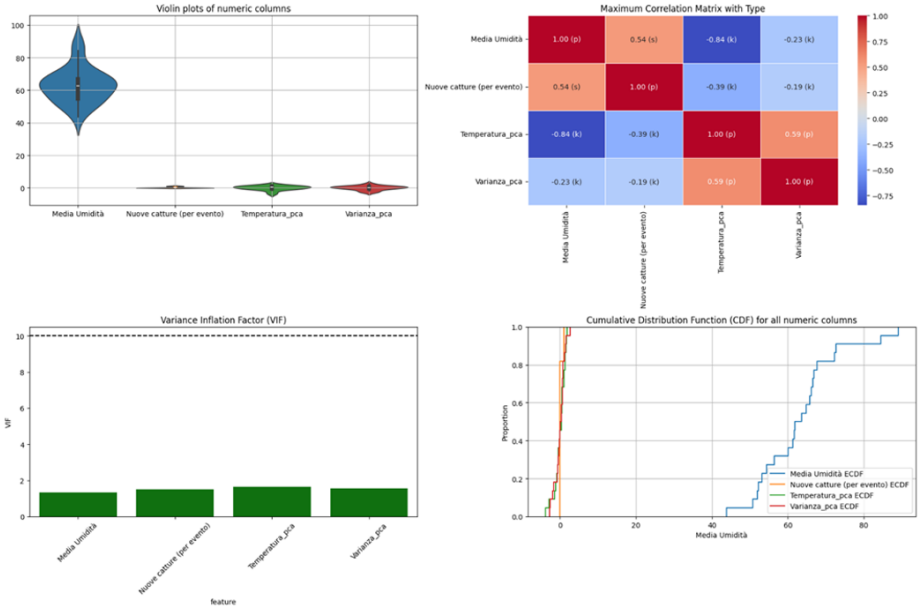

Figura 18: Analisi visivo-quntitativa

Sono stati nuovamente utilizzati i grafici di autocorrelazione, autocorrelazione parziale, scomposizione stagionale e time series per analizzare le caratteristiche temporali dei dati.

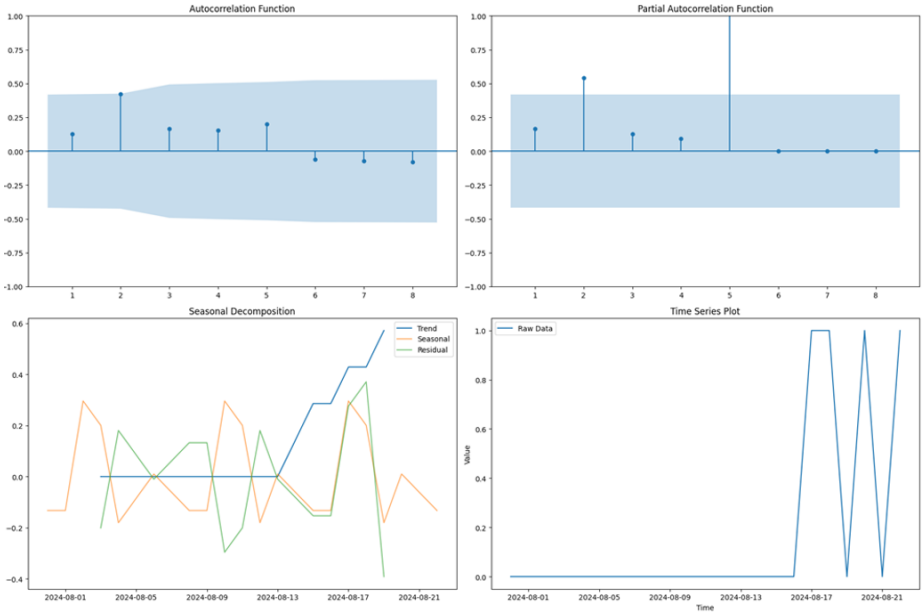

Figura 19: Analisi delle serie temporali

Come si può osservare, i dati mostrano una certa stagionalità. Tuttavia, le variabili target presentano una bassa dipendenza temporale, che potrebbe manifestarsi su una scala mensile piuttosto che giornaliera.

4.2 Addestramento

Anche in questo caso, sono state adottate diverse tecniche di addestramento:

- ARIMA come modello statistico.
- Ensemble Learning: utilizzando Random Forest e Gradient Boosting.
- Deep Learning: con un Multi-Layer Perceptron (MLP) combinato con la Cross Validation per ottimizzare l'addestramento e valutare le prestazioni.

4.2.1 Statistical Models

Per questo modello si è deciso di utilizzare esclusivamente il dataset risultante dall'unione dei primi due, mentre il terzo è stato scartato. Dato il limitato numero di dati disponibili, si è optato per una suddivisione dell'85% per il training e del 15% per il test.

- Numero di lag autoregressivi: 10
- Differenziazione per rendere stazionario il segnale: 0
- Numero di termini della media mobile: 11

I risultati ottenuti sono i seguenti:

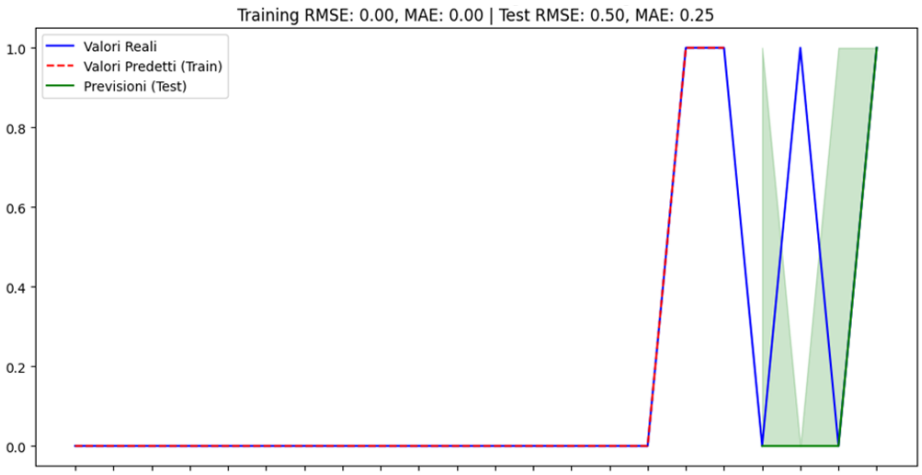

Figura 20: Modello ARIMAX - training e forecasting

4.2.2 Machine Learning Models

Per i modelli Gradient Boosting e Random Forest, sono stati utilizzati tutti e tre i dataset. Il dataset ottenuto dall'unione dei primi due è stato impiegato per il training, mentre il terzo dataset è stato utilizzato per il test.

- Numero di lags: 2
- Numero di estimatori: 200
- Massima profondità: 2
- Learning rate (Gradient Boosting): 0,008

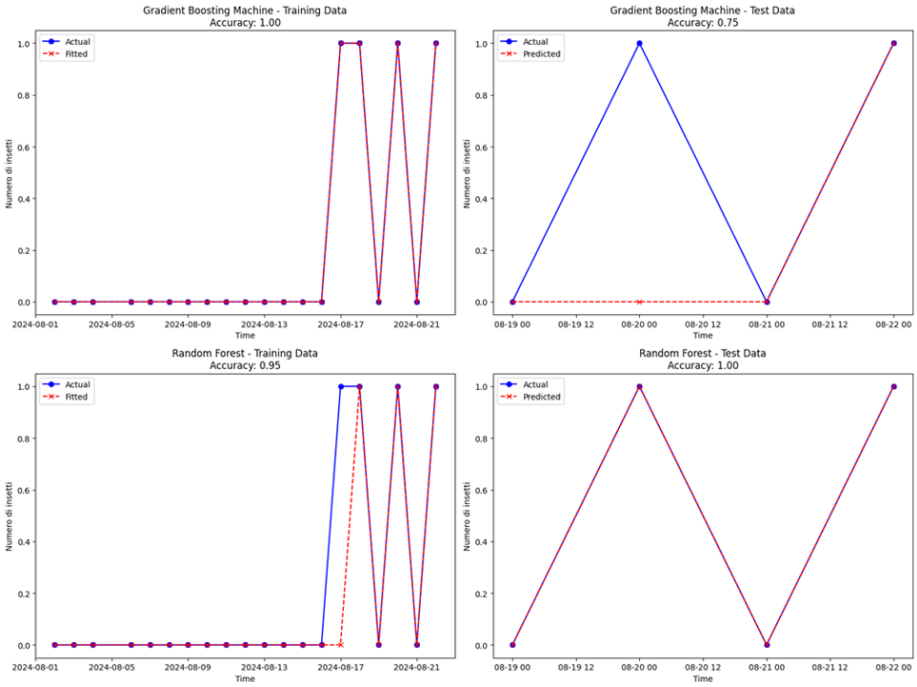

Figura 21: Gradient Boosting e Random Forest - training e forecasting

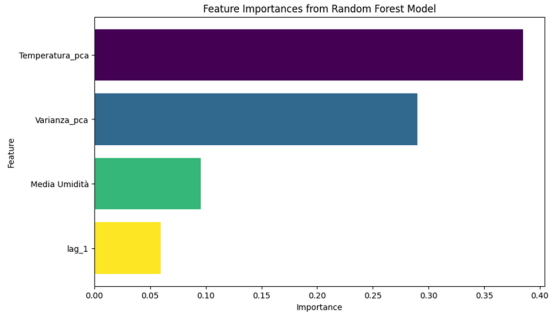

Figura 22: Feature importances (Gradient Boosting)

4.2.3 Deep Learning Models

Per il modello di Deep Learning, è stato utilizzato un Multi-Layer Perceptron (MLP). Come anticipato in precedenza, per l'addestramento della rete è stato usato il dataset ottenuto dall'unione dei dati relativi ai file *Imola 1* e *Imola 2*, mentre per il test è stato usato il dataset di *Imola 3*. La rete è stata strutturata nel seguente modo:

```
inputs = tf.keras.Input(shape=(X_train_scaled.shape[1],))
x = tf.keras.layers.Dense(5, activation='relu')(inputs)
x = tf.keras.layers.Dense(20, activation='relu')(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)  #
    Attivazione sigmoide per classificazione binaria
model_binary = tf.keras.Model(inputs=inputs, outputs=outputs)
])
```

Codice 2: Definizione modello MLP

Gli iperparametri scelti sono stati:

- Epoche: 50
- Batch size: 16
- Pesi per la loss: 0 : 1, 1: 2
- Numero di lags: 1

Come funzione di loss, è stata utilizzata la cross entropy, mentre come metrica è stata scelta l'accuracy. I risultati ottenuti sono i seguenti:

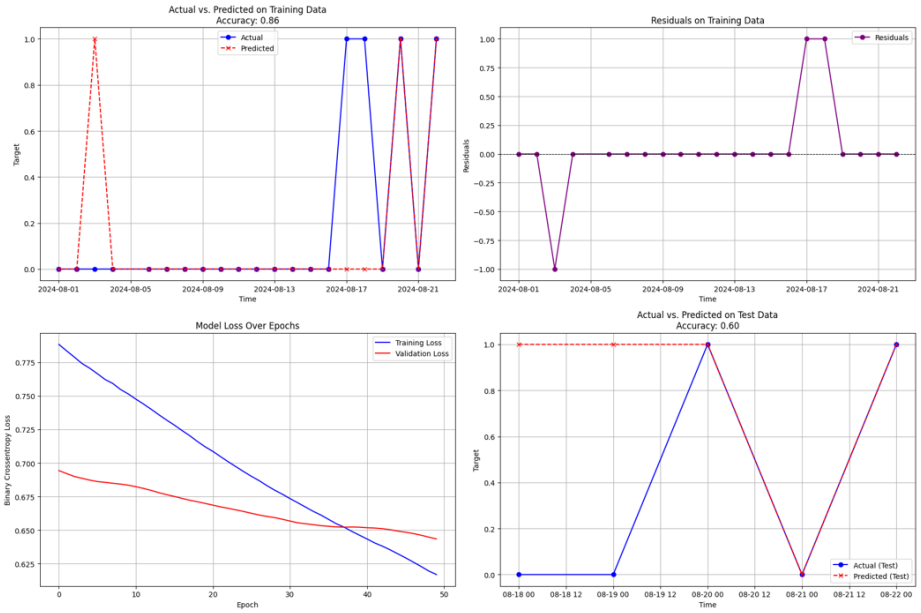

Figura 23: Modello MLP - training e forecasting

Per migliorare il modello, è stata utilizzata la Cross Validation al fine di superare il problema delle poche istanze disponibili e massimizzare l'uso dei dati. I risultati sul test set sono i seguenti:

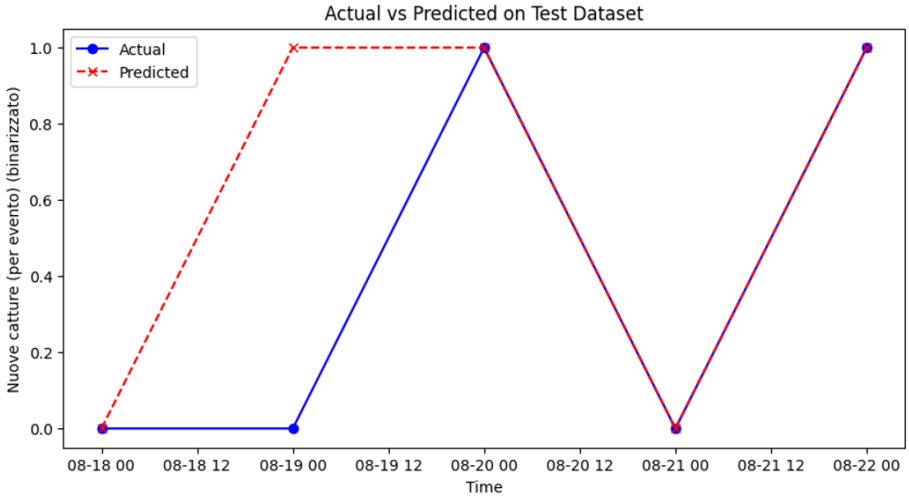

Figura 24: Modello MLP, cross validation - training e forecasting

5 Dashboard

In ultimo, è stata creata una Dashboard Streamlit con *Prophet*. Essa rappresenta una soluzione interattiva per la visualizzazione e la previsione di serie temporali. Streamlit [2] è una libreria Python che consente di creare facilmente applicazioni web interattive, mentre Prophet [3] è un modello di previsione sviluppato da Facebook, progettato per la previsione di serie temporali.
Negli esempi riportati i dataset caricati sono quelli pre-elaborati negli script python.

I settaggi e il caricamento del dataset sono stati spostati nella sidebar per migliorare l'organizzazione e lasciare lo spazio principale per visualizzare i contenuti interattivi come grafici e tabelle:

- La *sidebar* contiene i controlli per caricare il dataset, selezionare le colonne da usare per la serie temporale e configurare le impostazioni del modello Prophet. Ciò permette all'utente di interagire facilmente con le impostazioni senza ingombrare la zona centrale.
- La *sezione principale* è riservata alla visualizzazione dei dati originali, della data analysis, dei grafici delle serie temporali, delle previsioni e delle componenti del modello Prophet. In questo modo, l'utente ha una vista chiara e concisa dei risultati e delle analisi senza distrazioni.

Questa struttura migliora l'usabilità e consente di concentrarsi sui risultati visivi, mentre le impostazioni possono essere regolate rapidamente nella sidebar.

Il caching memorizza i risultati di una funzione per evitare calcoli quando si ripete lo stesso input. Questo è particolarmente utile in Streamlit, dove il codice può essere eseguito frequentemente (ad esempio, quando un utente interagisce con l'interfaccia).
Ogni volta che l'utente interagisce con un widget in Streamlit tutto il sorgente viene reinterpretato. Non ha senso replicare calcoli intensivi finché il metodo che restituisce tali computazioni non presenta modifiche del sorgente o delle variabili in ingresso.

Per migliorare le performance e ridurre i tempi di ricalcolo, sono stati aggiunti i decoratori `@st.cache_data` alle funzioni di caricamento dei dati e addestramento del modello:

- Caching dei dati: Una volta caricato il file CSV, i dati vengano memorizzati nella cache e non vengano ricaricati ogni volta che l'utente interagisce con l'app. Questo riduce i tempi di caricamento quando si lavora con lo stesso dataset.
- Caching del modello: Il modello viene settato "on the fly" in base ai dati e ai parametri configurati dall'utente. Questo significa che ogni volta che un utente fornisce un nuovo dataset o modifica i parametri, il modello viene

addestrato dinamicamente. Il caching del modello permette di memorizzare il modello addestrato per evitare che venga ricalcolato ogni volta che l'utente interagisce con l'app, migliorando così le performance. Tuttavia, dato che si vuole un modello diverso per dati diversi, il caching assicura che ogni utente possa avere un proprio modello senza interferenze. Se più utenti accedono simultaneamente all'app, ognuno si aspetta di lavorare con il proprio modello personalizzato, e il caching garantisce che i modelli siano distinti e non vengano sovrascritti o modificati da altre sessioni.

In questo modo, l'app ottimizza l'uso delle risorse e rende l'esperienza utente più fluida e reattiva, mantenendo al contempo l'efficacia del modello e la facilità di interazione. Garantisce, inoltre, la consistenza e l'integrità dei dati e del modello attraverso diverse sessioni.

Più nello specifico, le funzionalità della Dashboard sono le seguenti:

- Caricamento del file CSV da parte dell'utente;
- Visualizzazione Data Analysis: l'utente può scegliere tramite un checkbox se visualizzare o meno l'analisi sui dati, che presenta: anteprima dei dati, statistiche descrittive, valori mancanti, distribuzione dei dati, analisi visiva, grafico 3D, heatmap della correlazione;
- Selezione colonne Data e Target;
- Visualizzazione serie temporale
- Configurazione Modello Prophet: valore massimo possibile (Cap), valore minimo possibile (Floor), tipo di crescita (logistica o lineare), sensibilità ai cambiamenti: da 0 a 3, numero di periodi da prevedere;
- Visualizzazione risultato previsioni;
- Visualizzazione grafico delle previsioni;
- Visualizzazione componenti delle previsioni.

Figura 25: Dashboard

Riferimenti

[1] Google Research. *Google Colaboratory*. Ambiente notebook nel cloud con GPU/TPU. 2017. URL: https://colab.research.google.com (visitato il giorno 16/08/2025).

[2] Streamlit Inc. *Streamlit*. Framework per applicazioni web di data science e ML. 2019. URL: https://streamlit.io (visitato il giorno 16/08/2025).

[3] Taylor, Sean J. e Letham, Benjamin. *Prophet: Forecasting at Scale*. Libreria open-source per il forecasting sviluppata da Facebook (ora Meta). 2017. URL: https://facebook.github.io/prophet/ (visitato il giorno 16/08/2025).