Università degli Studi di Padova

Dipartimento di Scienze Statistiche

Corso di Laurea Triennale in
Statistica per l'Economia e l'Impresa

I Foundation Models per le serie storiche: un confronto empirico tra modelli locali e Chronos-Bolt

Relatore Prof. Daniele Girolimetto
Dipartimento di Scienze Statistiche

Laurendo/a Luca Trevisan
Matricola N 2087857

Anno Accademico 2025–2026

Indice

Elenco dei codici

Elenco delle tabelle

Introduzione

Nel mondo economico e aziendale prevedere cosa accadrà domani (le vendite di un prodotto, il prezzo di un'azione o la domanda di un certo prodotto nel mercato) è fondamentale e, storicamente, si è sempre ricorsi a modelli matematici costruiti *ad-hoc* per svolgere un dato compito. Questo approccio dunque richiedeva che per ogni set di dati si costruisse, testasse e si aggiornasse uno specifico modello atto a svolgere un determinato compito.

Nel Marzo del 2024 i ricercatori di Amazon Web Services (AWS) gettarono le basi per quella che potrebbe rivelarsi una vera e propria rivoluzione nell'ambito delle previsioni con il rilascio di un articolo intitolato: "Chronos, Learning the Language of Time Series". Chronos viene presentato come un cambio di paradigma, lanciato come un Foundation Model open-source, la più grande novità è quella di applicare l'architettura dei Transformer alle serie temporali, dimostrando quindi la capacità del modello di fornire previsioni affidabili su serie storiche inedite senza la necessità di un addestramento specifico sui dati in esame.

Il nuovo modello rompe gli schemi tradizionali attraverso un processo di quantizzazione che trasforma i dati storici in *token* che un'architettura *transformer* può utilizzare per individuare dei pattern globali e fornire previsioni immediate su dati mai visti durante l'addestramento. Il modello si dimostra subito estremamente efficace attraverso il suo approccio *language model*, tuttavia la sua architettura risulta ancora troppo pesante per diverse applicazioni aziendali in tempo reale, portando al rilascio di una versione che permette al sistema di essere fino a 250 volte più rapido, chiamata Chronos-Bolt.

La seguente tesi ha l'obiettivo di valutare le prestazioni del modello Chronos-Bolt attraverso l'analisi di dataset reali e pubblici. Il lavoro mette

a confronto l'efficacia di questo innovativo approccio con quella dei metodi tradizionali quali il modello ARIMA o modelli di *machine learning* addestrati *ad-hoc*, al fine di determinare se i *foundation models* rappresentino oggi una valida e superiore alternativa nelle applicazioni aziendali moderne.

Nel capitolo 1 viene introdotta l'evoluzione moderna nell'ambito del *forecasting*, si illustrano le principali caratteristiche dei modelli globali e dei *foundation models*, approfondendo dunque il passaggio nel tempo da una metodologia all'altra. Nel Capitolo 2 viene presentata una descrizione tecnica di Chronos-bolt, riservando particolare attenzione al funzionamento della libreria di *machine learning* AutoGluon-TimeSeries. Dopo aver trattato gli aspetti teorici, nel Capitolo 3 viene effettuato un confronto empirico basato su dati reali, mettendo a paragone le prestazioni dei modelli statistici tradizionali con quelle dell'innovativo Chronos-Bolt. Infine, la tesi presenta i risultati ottenuti mettendo a confronto le capacità predittive dei diversi modelli utilizzati, focalizzandosi principalmente sulla valutazione comparativa dell'accuratezza statistica e dell'efficienza computazionale.

Capitolo 1

Modelli globali e Foundation Models

Con l'avvento dell'intelligenza artificiale, il mondo del *forecasting* ha subito una profonda trasformazione, spostando il focus dalla semplice analisi statistica all'apprendimento profondo dei dati. Il motore di questo cambiamento è stata l'introduzione dell'architettura *transformer*, un sistema originariamete concepito per l'elaborazione del linguaggio che ha rivoluzionato la capacità delle macchine di interpretare le sequenze. Il presupposto di questa transizione risiede infatti nel trattare le serie temporali non come semplici serie numeriche, ma come un vero e proprio linguaggio dei dati. A differenza dei modelli locali, il *transformer* non analizza i singoli dati uno alla volta, ma riesce ad osservare l'intera serie storica contemporaneamente, identificando quali eventi del passato siano davvero rilevanti per prevedere il futuro attraverso il meccanismo di *Self-Attention*, il quale permette al modello di selezionare le informazioni più importanti all'interno di una massa enorme di dati, creando una gerarchia di rilevanza che rende la previsione molto più precisa. Queste innovazioni tecniche hanno permesso di superare i limiti dei modelli locali, dando vita ai modelli globali: sistemi capaci di apprendere pattern comuni da migliaia di serie temporali diverse.

1.1 I modelli Globali

I modelli globali rappresentano una delle più grandi evoluzioni nell'ambito del *forecasting*. Si tratta di approcci avanzati di analisi delle serie temporali

che adottano una prospettiva d'insieme attraverso la creazione di un singolo modello che consideri tutte le serie temporali contemporaneamente. L'obiettivo principale è dunque quello di individuare e catturare i *pattern* comuni tra diverse serie, permettendo al modello di distinguere i pattern strutturali dal possibile rumore stocastico che ogni singola serie potrebbe introdurre. I modelli globali risultano quindi particolarmente efficaci in presenza di serie temporali con pochi dati storici (*cold start*), in quanto sfruttano le informazioni provenienti dalle serie più ricche di dati per fare previsioni, utilizzando tecniche di *machine learning*, come le *reti neurali*, o approcci statistici avanzati per catturare relazioni non lineari complesse, ottenendo quindi maggiore accuratezza e una migliore gestione di stagionalità e trend rispetto ai modelli locali.

1.1.1 Modelli Locali e Globali a confronto

Storicamente, il campo del *forecasting* è stato dominato dall'utilizzo di modelli locali quali ARIMA, ARCH, GARCH e tanti altri. Questo approccio più tradizionale considera ogni serie storica come un'entità singola, sulla quale stimare dei parametri ad hoc e adattare una specifica funzione statistica. Negli ultimi anni si è invece assistito ad un cambio di paradigma verso i modelli globali, che, come detto in precedenza, mirano a definire un'unica funzione universale capace di operare su una vasta gamma di dataset.

Per capire come i modelli globali stiano progressivamente sostituendo l'approccio locale è fondamentale analizzarne le differenze strutturali. In primo luogo, contrariamente a quanto si potrebbe ipotizzare, l'approccio globale non risulta più 'rigido' o vincolato rispetto a quello locale, infatti, pur applicando una singola funzione a dataset eterogenei, riesce a garantirne un'elevata precisione senza dover presupporre necessariamente una somiglianza tra le diverse serie temporali. Un secondo aspetto cruciale riguarda l'efficienza, mentre i modelli locali diventano più 'pesanti' e difficili da gestire all'aumentare dei dati, quelli Globali mantengono una complessità costante. Questa caratteristica permette loro di gestire enormi quantità di informazioni in modo sempre efficiente e di beneficiare di una 'memoria' molto più estesa, ricordando così nel tempo le informazioni più rilevanti per formulare pre-

visioni più accurate. Infine, ampie evidenze empiriche hanno dimostrato che anche modelli lineari che sfruttano la struttura globale sono in grado di superare in termini di prestazioni i modelli locali più complessi (Montero-Manso e Hyndman 2021).

1.1.2 Machine Learning: il motore dei Modelli Globali

L'implementazione dei Modelli Globali trova ampio utilizzo nell'ambito del *machine learning*. Sebbene la distinzione tra approccio locale e globale sia di tipo teorico e statistico, è grazie alle capacità computazionali e algoritimiche del *machine learning* che tale operazione diventa operativamente vantaggiosa. Il *machine learning* rappresenta un ambito specialistico statistico capace di svolgere attività di analisi sia in modalità non supervisionata, estraendo informazioni senza un target specifico, sia in modalità supervisionata, dove l'algoritmo viene addestrato su relazioni note tra predittori e variabili risposta per massimizzare l'accuratezza predittiva. Tale tecnologia si distingue per l'abilità di elaborare grandi volumi di dati, caratteristica principale dei modelli globali, in modo da identificare pattern comuni, caratteristiche e legami tra i dati analizzati che fino ad allora erano rimasti sconosciuti. L'idea alla base del concetto di *machine learning* riguarda l'esistenza di una relazione matematica tra dati di input e di output. Tuttavia, questa relazione non è nota anticipatamente al modello, esso può elaborarla solo attraverso un adeguato lavoro di addestramento, nel quale gli vengono forniti esempi di dati di input e output, in modo da elaborare come suo fondamento una funzione matematica modificabile. Il *machine learning* prevede quindi quattro fasi ben distinte:

- **Pre-elaborazione dei dati:** nella quale i dati grezzi vengono puliti e trasformati per addestrare al meglio un modello di Machine Learning;
- **Addestramento del modello:** i dati precedentemente trasformati vengo utilizzati per addestrare il modello, è proprio in questa fase che il modello ricerca le ralazioni tra i dati e i pattern comuni;
- **Valutazione del modello:** questa fase rappresenta un passaggio di estrema importanza, l'obiettivo è quello di verificare se il modello, pre-

cedentemente addestrato, riesce a svolgere bene il suo lavoro, tutto ciò attraverso un set di dati simili che svolge la funzione di *Benchmark*;

- **Ottimizzazione:** quest'ultima fase implica il perfezionamento del modello al fine di migliorarne le prestazioni e le capacità;

Risulta quindi evidente come il patrimonio informativo e le relative procedure di acquisizione rappresentino un asset di vitale importanza. Il Machine Learning favorisce dunque l'automatizzazione e l'ottimizzazione del processo di raccolta e gestione dei dati, garantendo quindi una solida base per la creazione di modelli globali (AWS Team 2021b).

Deep Learning e Reti Neurali

All'interno del vasto panorama del *machine learning*, il ramo che riveste una particolare importanza nella formazione di modelli globali è il *deep learning*. Il *deep learning* rappresenta una branca intelligenza artificiale, progettata per consentire ai sistemi informatici di elaborare e analizzare i dati in modo simile al cervello umano. Grazie a questo approccio esso è in grado di riconoscere suoni, testi, immagini e ad automatizzare processi che solitamente necessiterebbero dell'intelligenza umana. Questa tecnologia è estremamente importante in quanto fornisce un supporto a moltissime applicazioni di intelligenza artificiale utilizzate quotidianamente quali chatbot, assitenti digitali, telecomandi ad attivazione vocale, rilevamento frodi e riconoscimento facciale automatico.

I modelli di *deep learning* si basano sulle *reti neurali artificiali*, modelli computazionali che simulano il principio di funzionamento del sistema nervoso biologico. Nel cervello umano l'apprendimento deriva dalla sinergia di milioni di neuroni che, attraverso la loro interconnessione, ci permettono di elaborare e analizzare informazioni. In ambito informatico tale processo è replicato da unità logiche definite 'nodi'. Questi neuroni artificiali utilizzano algoritmi e funzioni matematiche per l'elaborazione dei dati e sono organizzati in molteplici strati gerarchici che, comunicando l'uno con l'altro, conferiscono alle *reti neurali* la capacità di risolvere problemi complessi.

L'architettura di una *rete neurale profonda* si articola in diverse componenti fondamentali, ognuna delle quali ricopre un ruolo specifico nel processo

di elaborazione. Seguendo lo schema teorico proposto da AWS Team 2021a i livelli principali sono descritti di seguito:

- **Livello di input:** rappresenta l'inizio della rete Neurale, composto da una serie di nodi il cui compito è ricevere e trasmettere i dati grezzi all'interno del sistema;
- **Livello nascosto:** analizza le informazioni a diversi livelli adattando il suo comportamento man mano che il carico di informazioni aumenta;
- **Livello di output:** é costituito dai nodi che generano le 'riposte' del modello. Un modello più semplice che genera come risposte solo 'si' o 'no' presenta solamente due nodi, i modelli più complessi, che permettono una gamma di risposte più ampia e accurata, ne presentano migliaia;

1.2 I foundation Models

L'evoluzione del *deep learning* ha portato all'emergere di un paradigma innovativo nella creazione di sistemi di Intelligenza artificiale, basato su una classe di modelli noti come *foundation models*. I *foundation models* sono modelli addestrati su una vasta e diversificata quantità di dati. Essi poggiano sulle architetture delle reti neurali profonde e sfruttano il principio del *transfer learning*, ovvero la capacità di trasferire le abilità acquisite durante l'addestramento generale verso contesti applicativi più specifici. L'utilizzo di questi modelli introduce quindi il concetto di *omogeneizzazione*, l'utilizzo di un unico modello di base come fondazione per molteplici applicazioni, tuttavia il loro utilizzo richiede estrema cautela in quanto eventuali difetti o bias del modello potrebbero quindi propagarsi in modo sistemico (Bommasani et al. 2021).

1.2.1 Come funzionano i Foundation Models

Un *foundation model* opera identificando schemi e correlazioni all'interno dei dati per prevedere l'elemento successivo di una sequenza; ciò può tradursi, ad esempio, nella ricostruzione o perfezionamento di un'immagine oppure,

nell'ambito del linguaggio naturale, nell'analisi del contesto delle parole precedenti per determinare il termine più coerente per continuare la frase. Una caratteristica distintiva dei *foundation models* è il *self-supervised learning*, metodologia che permette al sistema di generare autonomamente le proprie etichette a partire dai dati grezzi in input. Questo approccio elimina la necessità di un intervento umano per la classificazione manuale dei dataset di addestramento, rappresentando un netto distacco rispetto ai paradigmi classici del *machine learning*.

Nonostante la fase di pre-addestramento, i *foundation models* mantengono la capacità di apprendere e adattarsi contestualmente attraverso i dati o i prompt forniti durante la fase di inferenza. Tale proprietà permette di ottenere risultati estremamente articolati e specifici semplicemente attraverso l'elaborazione di istruzioni accuratamente strutturate. Le attività che i *foundation models* possono svolgere includono l'elaborazione del linguaggio, con la possibilità di dare risposte o scrivere brevi script, la comprensione visiva, con l'identificazione di immagini o oggetti fisici, la generazione di codice in diversi linguaggi di programmazione, la possibilità di intraprendere un dialogo attivo con l'utente, cercando di imparare nuove informazioni dai suoi prompt, e, infine, può sostituire il contenuto testuale in formati audio o sottotitoli mediante tecniche di sintesi vocale (AWS Team 2023a).

1.2.2 Il concetto di Transfer Learning

Il *transfer learning* è un concetto precedentemente introdotto e rappresenta una caratteristica fondamentale dei *foundation models*. Si tratta di una tecnica di *machine learning* nella quale un modello, precedentemente addestrato per svolgere uno specifico compito, viene riadattato per operare su una nuova attività correlata. Questo permette di superare il processo di addestramento tradizionale, che richiede quantità massive di dati e grande potenza di calcolo, semplicemente 'trasferendo' le conoscenze acquisite dal modello su un dato compito ad uno simile (AWS Team 2023b).

Il *transfer learning* presenta diversi vantaggi quali:

- **Costi computazionali:** riduce drasticamente le risorse e i costi di calcolo che solitamente vengono richiesti per la creazione di nuovi modelli,

il tutto proponendo modelli già esistenti e già precedentemente addestrati. Questo permette di contrarre i tempi di sviluppo e la potenza di calcolo necessaria per raggiungere gli obiettivi prefissati;

- **Dimensioni dei set di dati:** Il *TL* aiuta a risolvere le problematiche legate all'acquisizione di grandi set di dati. Ad esempio, i modelli linguistici necessitano di quantità enormi di dati per essere addestrati, tuttavia i set di qualità disponibili al pubblico sono estremamente limitati;

- **Capacità di generalizzazione:** questo approccio, oltre a tagliare costi e tempi, permette anche di aumentare la capacità di generalizzazione. Il modello 'riqualificato', infatti, sarà dotato di conoscenza apprese da più set di dati;

Per quanto riguarda gli svantaggi, il *transfer learning* non presenta criticità intrinseche, le problematiche di questo approccio sono infatti legate solamente a un suo utilizzo errato. Il successo di questa tecnica dipende dunque dalla somiglianza tra le attività di apprendimento, dall'affinità delle distribuzioni dei dati tra il set di origine e quello di destinazione, e dall'applicabilità di un modello analogo a entrambi i contesti. Qualora queste condizioni non venissero soddisfatte, si rischia di incorrere nel cosiddetto *negative transfer*, il quale compromette le prestazioni del modello sul nuovo compito. Proprio per ovviare a questa problematica, le recenti ricerche suggeriscono l'utilizzo di test preliminari al fine di verificare il grado di compatibilità tra i dataset e prevenire un potenziale peggioramento dei risultati (AWS Team 2023b).

Capitolo 2

AutoGluon-TimeSeries e Chronos-Bolt

Dopo aver analizzato l'evoluzione dai modelli di *machine learning* ai moderni *foundation models*, questo capitolo si focalizza maggiormente sugli strumenti applicativi AutoGluon-TimeSeries e Chronos-Bolt. Il primo rappresenta una delle più recenti innovazioni nella libreria AutoGluon, un *framework* di *AutoML (Automated Machine Learning) open-source* sviluppato da Amazon Web Service. Tale strumento permette di automatizzare l'intero ciclo di vita dei modelli di *deep learning*, gestendo autonomamente le fasi di pre-elaborazione, addestramento e ottimizzazione (Shchur et al. 2023).

Chronos-Bolt si colloca nell'ambito dei *foundation models* come un'evoluzione significativa nel campo del *forecasting*. Esso rappresenta l'ottimizzazione della versione originale del modello rilasciata da AWS, la quale presentava come criticità principale una ridotta velocità di esecuzione. Pre-addestrato su una vasta serie di dati eterogenei, Chronos-Bolt rappresenta un'evoluzione grazie alla sua peculiarità principale, ovvero il *zero-shot forecasting*, fornendo stime accurate anche su dataset mai incontrati durante la fase di addestramento (Ansari et al. 2024).

2.1 AutoGluon-TimeSeries

Sviluppato come l'estensione più recente dell'ecosistema AutoGluon, il modulo TimeSeries eredita la solidità di uno dei principali *framework open-source* per l'*AutoML*. Il suo principale obiettivo è la massima semplificazione

della costruzione di sistemi predittivi complessi, il tutto attraverso un'interfaccia che richiede solamente poche righe di codice e che rispetta il *time budget* impostato dall'utente. L'operatività di AutoGluon-TimeSeries si basa su tre punti fondamentali: in primo luogo, la capacità di fornire un *forecasting quantilico*, superando così i limiti della previsione puntuale offrendo una stima più rigorosa dell'incertezza; in secondo luogo, l'integrazione di *covariate eterogenee*, fondamentali per catturare dinamiche esterne al dataset principale; infine, l'impiego di strategie *multi-model ensembling*, che permettono di combinare modelli statistici e di *deep learning* per massimizzare l'accuratezza e la robustezza della previsione finale.

2.1.1 L'Automated Machine Learning

AutoGluon si colloca nel panorama dei software *open-source* come uno dei *framework* di *AutoML* più efficaci e versatili. L'*AutoML* rappresenta l'insieme di metodologie volte ad automatizzare l'intero ciclo di vita dei modelli di *machine learning*, dallo sviluppo iniziale fino al deployement finale. L'obiettivo primario è quello di permettere anche ai meno esperti di creare e implementare modelli di intelligenza artificiale, ma anche a ottimizzare i flussi di lavoro di esperti e data scientist. Gli strumenti di *AutoML* facilitano dunque le fasi di addestramento, validazione e implementazione di modelli avanzati, come i sistemi di *deep learning* e di IA generativa. Un vantaggio cruciale risiede nella capacità di garantire risultati riproducibili e spiegabili, fondamentale per l'adozione dell'IA in settori ad alta regolamentazione. Senza gli stumenti di *AutoML* ogni passaggio della pipeline di *machine learning* richiederebbe dunque un intervento manuale lungo e soggetto a errori (Belcic e Stryker 2026).

Per questo motivo una pipeline di *AutoML* è stata studiata e progettata per automatizzare tutti i processi di creazione e sviluppo di un modello di *machine learning*, dalla preparazione dei dati alla distribuzione del modello. In seguito vengono analizzate le fasi che compongono la pipeline:

- **Preparazione dei dati (*Data Preprocessing*):** parte dalla raccolta dei dati da diverse fonti, provvede dunque alla loro pulizia, correzione di errori o rimozione di duplicati. In questa fase può subentrare la

tecnica particolare di aumento dei dati, ovvero la creazione di nuovi dati generando varianti di quelli già esistenti.

- **Ingegneria delle funzioni (*Feature Engineering*):** attraverso l'analisi di varianza, correlazioni e algoritmi di selezione automatica si individuano e selezionano la caratteristiche più importanti dei dati.
- **Selezione del modello (*Model Selection*):** Si definiscono i modelli di *Machine Learning* da considerare, scegliendo tra modelli tradizionali e architetture neurali.
- **Ottimizzazione degli iperparametri (*Hyperparameter Optimization, HPO*):** si effettua una ricerca automatizzata della configurazione ottimale dei parametri per ogni modello selezionato.
- **Combinazione dei modelli (*Ensembling*):** aggregazione dei modelli più performanti per migliorare la robustezza e l'accuratezza della previsione finale.

l'integrazione di tutte queste fasi in un'unica pipeline permette dunque di ridurre drasticamente il tempo necessario per la sperimentazione di nuovi modelli per il *forecasting*. L'*AutoML* non rappresenta solo la base su cui poggia il funzionamento di Chronos-Bolt, ma un vero e proprio cambio di paradigma per il mondo delle previsioni (Salehin et al. 2024).

2.1.2 Previsione probabilistica di serie temporali

La formalizzazione del problema predittivo all'interno di AutoGluon - TimeSeries si basa sulla gestione di un dataset $\mathcal{D} = \{\mathbf{y}_{i,1:T_i}\}_{i=1}^{N}$ inteso come l'insieme di N serie temporali univariate, dove ogni valore del dataset rappresenta quindi il valore della i-esima serie al tempo t. L'obiettivo principale del *framework* è la stima dei valori futuri per un determinato orizzonte di previsione, basandosi sui dati storici e sull'integrazione di covariate. Le variabili esogene possono essere di natura statica, come l'ID di un prodotto o la posizione geografica di uno store, o variabili nel tempo, distinguendosi tra variabili note nel futuro e variabili note solo nel passato. Dal punto di vista matematico, il sistema mira a modellare la distribuzione condizionata dei

valori futuri, $\mathbf{y}_{i,T+1:T+H}$, dati i valori passati, $\mathbf{y}_{i,1:T}$, e le covariate, $\mathbf{X}_{i,1:T+H}$, traducendo tale complessità in previsioni probabilistiche basate sui quantili:

$$\hat{\mathbf{y}}_{i,T+1:T+H} = \mathbb{E}_p \left[\mathbf{y}_{i,T+1:T+H} \mid \mathbf{y}_{i,1:T}, \mathbf{X}_{i,1:T+H} \right]$$

Questo approccio permette di superare il limite della previsione puntuale, fornendo una stima dell'incertezza che risulta fondamentale per una valutazione robusta della variabilità dei dati nell'orizzonte temporale considerato (Shchur et al. 2023).

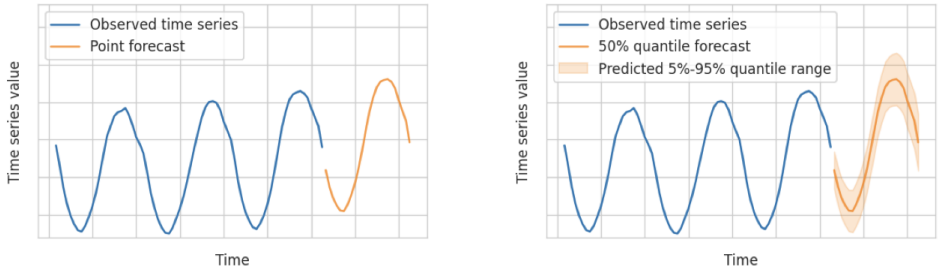

Figura 2.1: Previsioni puntuali (sinistra) e previsioni quantiliche (destra). Fonte: Shchur et al. 2023.

2.1.3 L'Ensembling

L'ensemble learning è una tecnica di *machine learning* che si basa sull'aggregazione di molteplici modelli predittivi, meglio noti come *weak learners*. L'idea alla base di tale approccio è che la cooperazione tra diversi modelli permetta di ottenere prestazioni superiori, accompagnate a una capacità di generalizzazione e accuratezza più elevata rispetto al singolo modello che agisce in solitaria (Ultralytics Team 2024). Come dimostrato da numerosi studi, l'approccio *ensemble* permette dunque di sintetizzare i punti di forza dei diversi modelli presi in considerazione, eliminando quindi problemi di regressione quali l'*overfitting* e rappresentando uno strumento capace di controllare il compromesso tra distorsione e varianza, due elementi che rappresentano inversamente l'accuratezza del modello (Murel e Kavlakoglu 2024).

Esistono diverse metodologie di applicazione dell'*ensemble learning*. Di seguito vengono analizzate le tecniche principali:

- **Bagging:** consiste nel generare molteplici versioni di un dataset attraverso il *bootstrapping*. Su ogni versione viene addestrato un modello indipendente e le previsioni di questi modelli vengono poi aggregate tramite media o votazione. Questa strategia permette dunque di ridurre drasticamente la varianza del sistema, evitando che il modello si adatti troppo ai singoli dettagli del dataset originale.

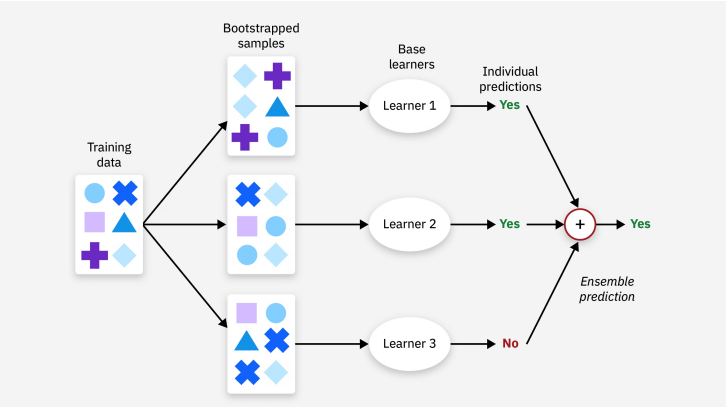

Figura 2.2: Tecnica del Bagging. Fonte: Murel e Kavlakoglu 2024.

- **Stacking:** questa tecnica combina modelli di natura diversa, come alberi decisionali o reti neurali, addestrati sullo stesso dataset. Il processo prevede quindi la generazione di previsioni iniziali da parte dei modelli di base, le quali verranno poi utilizzate come input per un modello di livello superiore, definito *Meta-Learner*. Per evitare problemi di overfitting è inoltre necessario che il nuovo modello non venga addestrato su dati già visti in precedenza.

- **Boosting:** si tratta di una metodologia di apprendimento che mira a trasformare i *weak learners* in un modello robusto. Il processo avviene per iterazioni successive: ogni nuovo modello viene addestrato con l'obiettivo specifico di correggere gli errori commessi dai predecessori. Si basa sulla ridistribuzione dei pesi o sull'analisi dei residui, focalizzando l'addestramento sulle istanze più difficili da classificare.

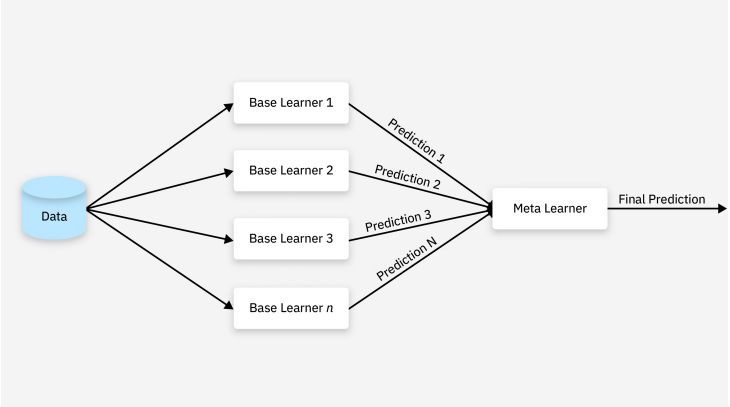

Figura 2.3: Tecnica dello Stacking. Fonte: Murel e Kavlakoglu 2024.

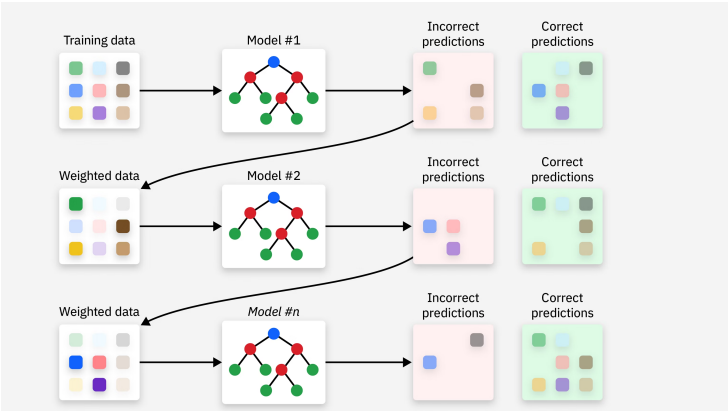

Figura 2.4: Tecnica del Boosting. Fonte: Murel e Kavlakoglu 2024.

Quelle appena analizzate rappresentano le metodologie di *ensemble* più consolidate e diffuse, tuttavia, il mondo del *machine learning* offre numerosi altri approcci in continua evoluzione (Murel e Kavlakoglu 2024).

2.2 Chronos-Bolt

Chronos-Bolt rappresenta una delle ultime invenzioni nell'ambito dei *foundation models*. Integrato all'interno del *framework* AutoGluon-TimeSeries, questo modello eredita la logica dei grandi modelli linguistici per trasformare i dati numerici in sequenze di informazioni comprensibili da un'architettura basata su *transformer*. Rispetto alla versione originale di Chronos, *Bolt* è stato progettato con l'obiettivo primario dell'efficienza operativa. Grazie a una profonda ottimizzazione strutturale, il modello è dunque in grado di generare previsioni con una velocità fino a 250 volte superiore rispetto al suo predecessore. Il suo funzionamento si basa su alcune caratteristiche fondamentali che ne definiscono l'efficacia: il concetto di *zero-shot forecasting*, la capacità di fornire previsioni accurate anche su dataset mai visti durante l'addestramento, la tecnica del *patching*, che permette di segmentare il set di dati al fine di elaborare sequenze più lunghe in minor tempo, e infine il *fine-tuning*, ovvero una tecnica che permette di ri-addestrare il proprio modello su set di dati specifici.

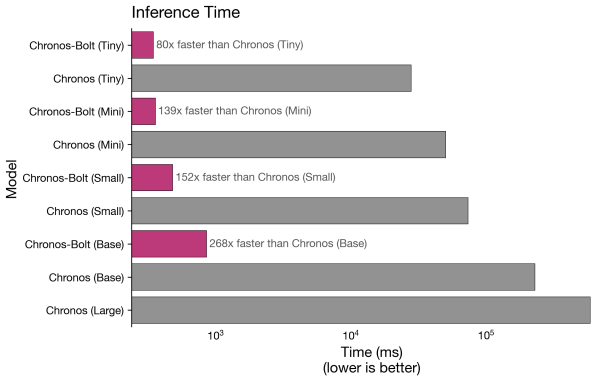

Figura 2.5: Confronto del tempo di inferenza tra Chronos e Chronos Bolt. Fonte: Ansari et al. 2024.

2.2.1 Lo Zero-Shot forecasting

Chronos-Bolt fornisce previsioni *zero-shot*, una capacità che lo distingue nettamente dai modelli di *forecasting* tradizionali. Mentre l'approccio classico richiede un addestramento specifico sul dataset in esame, i modelli che effettuano questa tipologia di previsioni sono in grado di operare applicando pattern statistici appresi durante una vasta fase di pre-addestramento a dataset mai visti in precedenza (Ansari et al. 2024). L'efficacia di Chronos-Bolt nel fornire tali previsioni è potenziata dall'integrazione delle *covariate*, come analizzato nella Sezione 2.1.2.

A differenza degli approcci tradizionali, i modelli *zero-shot* devono cercare di apprendere la relazione tra covariate e serie target direttamente durante la fase di inferenza. Tale processo è noto come *in-context learning*: il modello osserva il comportamento delle covariate nel contesto dei dati attuali per dedurne l'impatto sul futuro. Nonostante il concetto di *in-context learning* possa apparire limitato rispetto all'applicazione di modelli più tradizionali addestrati a livello globale, le ricerche di Brown et al. 2020 dimostrano come i *transformer* siano capaci di un sofisticato apprendimento *in-context*. Il modello dunque non si limita solo ai dati storici della serie target, ma integra variabili ausiliarie per bilanciare la capacità di generalizzazione tipica dei *foundation models* con le specificità del contesto locale, garantendo previsioni probabilistiche robuste e immediate (Auer et al. 2025).

2.2.2 La tecnica del Patching

Sebbene la capacità di operare in modalità *zero-shot* garantisca flessibilità operativa, l'efficienza computazionale di Chronos-Bolt è determinata da una specifica innovazione: la tecnica del *patching*. Nelle architetture *transformer* tradizionali applicate alle serie temporali ogni punto della serie viene osservato singolarmente, risultando estremamente oneroso per serie storiche lunghe. La necessità del *patching* deriva dunque da un limite intrinseco delle serie temporali: a differenza di una parola in una frase, un singolo valore numerico ad un istante t non possiede un significato semantico completo. Per comprendere le connessioni profonde tra i dati è quindi necessario estrarre informazioni semantiche locali. Mentre i modelli classici quindi si limitano

ad osservare i dati uno ad uno, Chronos-Bolt aggrega i singoli step temporali in *patch*, considerandole come delle sottoserie. Questo approccio permette di catturare una struttura informativa più ricca e completa, migliorando le capacità del modello di interpretare le dinamiche locali della serie (Nie et al. 2022).

2.2.3 Il fine-tuning

Sebbene la capacità di effettuare previsioni *zero-shot* di Chronos-Bolt offra prestazioni elevate su dataset mai visti prima, l'architettura dei *foundation models* permette di ottimizzarle ulteriormente attraverso il processo di *fine-tuning*. Il *fine-tuning* rappresenta una specifica startegia di *transfer learning* (si veda la sezione 1.2.2) in cui la conoscenza estratta da un modello durante una vasta fase di pre-addestramento viene trasferita e adattata a un compito specifico. Il processo utilizza i parametri già utilizzati nella fase di addestramento globale, un approccio che permette al modello di partire da una configurazione che già comprende le strutture fondamentali delle serie temporali, come i trend o le componenti stagionali. Durante il *fine-tuning* i parametri vengono aggiornati tramite l'algoritmo di *backpropagation* su un dataset specifico, il tutto con un tasso di apprendimento più contenuto, in modo da evitare la perdita di capacità di generalizzazione del modello.

L'integrazione di questa tecnica nel contesto di Chronos-Bolt risulta vantaggiosa per due ragioni: in primo luogo, riduce drasticamente il volume di dati necessari per ottenere previsioni su dati di 'nicchia'; in secondo luogo, permette di rifinire dei segmenti temporali creati attraverso la tecnica del *patching*. In sintesi, il *fine-tuning* agisce come una fase di specializzazione che permette al modello di operare in maniera più esperta, ponendo insieme la robustezza dei *foundation models* con la sensibilità alle dinamiche particolari del caso di studio (Bengio, Goodfellow, Courville et al. 2017).

Capitolo 3

Confronto empirico tra modelli locali e Chronos-Bolt

Il presente capitolo è dedicato alla fase sperimentale dell'elaborato, finalizzata al confronto empirico tra le capacità predittive di specifici modelli locali selezionati e il *foundation model* pre-addestrato Chronos-Bolt. L'analisi è stata condotta su tre differenti dataset, caratterizzati da diverse frequenze e lunghezze temporali, integrando le potenzialità di due differenti ambienti di programmazione: Python, impiegato per l'implementazione di modelli basati sull'architettura *transformer*, e R, utilizzato per l'applicazione della modellistica statistica locale e i test di validazione inferenziale. In ambiente Python, attraverso l'utilizzo di *TimeSeriesDataFrame*, una sottoclasse di *PandasDataFrame* specificamente progettata per garantire l'integrazione con la libreria AutoGluon, e *TimeSeriesPredictor*, utilizzato per la generazione delle stime, il modello Chronos-Bolt è stato applicato nelle sue due principali configurazioni, ovvero *zero-shot* e *fine-tuned*.

Al fine di valutare la bontà dei risultati ottenuti dal *foundation model*, sono stati stimati i seguenti modelli di confronto locali: il modello *ARIMA*, integrato, a seconda delle caratteristiche della serie, con componenti stagionali o covariate, i modelli *Naïve* e *Seasonal Naïve*, e infine il modello *ETS*, il quale, formalizzato da Hyndman et al. 2002, rappresenta l'evoluzione statistica dello smorzamento esponenziale classico, permettendo di scomporre la serie nelle sue componenti chiave e di calcolare anche i relativi margini di

errore probabilistici.

Dal punto di vista operativo, la valutazione dell'accuratezza predittiva è stata condotta attraverso un esperimento di previsione a finestra espandibile. Per ciascuna serie storica, la porzione iniziale pari al 65-75% delle osservazioni totali è stata isolata come *training set* iniziale. Il processo si articola in modo ciclico: alla prima iterazione, fissata un'origine temporale t, il modello viene addestrato sulle informazioni disponibili fino a quel momento per generare una previsione multi-step in avanti fino al tempo $t+h$. Alla successiva iterazione l'origine viene traslata in avanti di una singola unità al tempo $t+1$; quest'ultima osservazione viene dunque inserita nel nuovo dataset di addestramento e il modello proietta una nuova stima a h passi avanti, estendendo così l'orizzonte predittivo fino al tempo $t+1+h$. Questo ciclo si ripete per tutta la porzione rimanente del dataset, permettendo di simulare il comportamento predittivo dei modelli in un contesto di continuo monitoraggio e aggiornamento.

Per il confronto tra le capacità predittive dei modelli implementati sono state utilizzate due metriche principali: il *Mean Absolute Error (MAE)* e il *Mean Squared Error (MSE)*. Il primo viene calcolato determinando il valore assoluto della differenza tra ciascun valore reale y_i e la corrispondente previsione $\hat{y}_i$, offrendo così una misura diretta dell'errore:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.1}$$

Il secondo, invece, valuta lo scarto quadratico tra il valore reale y_i e la stima $\hat{y}_i$, elevando tale differenza al quadrato al fine di penalizzare in modo più severo gli errori di maggiore entità:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.2}$$

3.1 Analisi della temperatura media europea

Il primo dataset oggetto di studio comprende le rilevazioni mensili della temperatura media europea nel periodo compreso tra gennaio 1960 e ottobre 2024, i dati provengono dal *framework open-source* E-OBS, sviluppati

nell'ambito del progetto *European Climate Assessment and dataset* e distribuiti ufficialmente attraverso la piattaforma *Copernicus Climate Data Store* (Cornes et al. 2018). La scelta di questa specifica serie storica è dettata dalla volontà di testare la capacità di Chronos-Bolt di identificare e modellare una stagionalità forte e regolare, tipica dei fenomeni climatici.

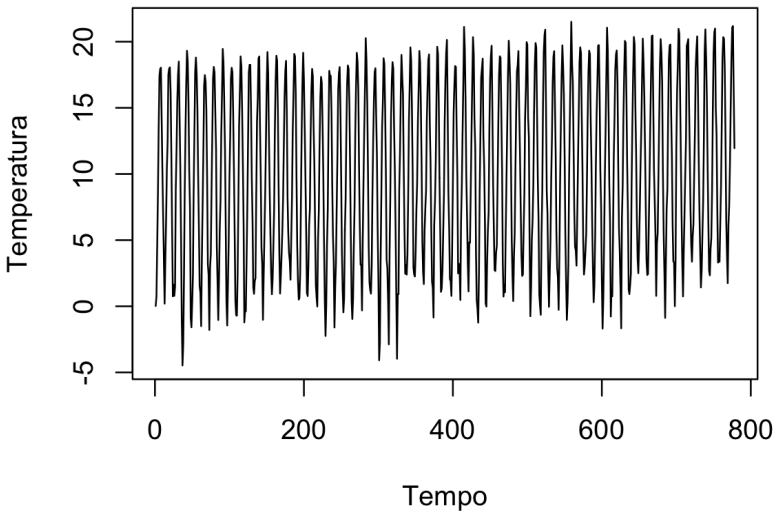

Figura 3.1: Temperatura media europea (1960-2024).

Da una prima analisi visiva della serie si nota la presenza di una forte componente stagionale, elemento centrale per la definizione del modello locale di riferimento. Per individuare il modello che meglio descrive i dati sono state usate due metriche di riferimento: l'*Akaike Information Criterion (AIC)*, fondamentale nella selezione dei modelli in quanto ne valuta sia la precisione che l'efficienza (penalizzando l'eccessiva complessità), e il test di *Ljung-Box*, il quale si assicura che i residui del modello non contengano più alcuna struttura o informazione sistemica. Seguendo questa metodologia, è stato selezionato un modello SARIMA$(1,0,0)(0,1,1)_{12}$, con un AIC pari a 2474.771. La bontà della scelta è stata confermata dal test di Ljung-Box che, calcolato su 20 ritardi, restituisce un p-value di 0.053, indicando che i

residui sono effettivamente privi di informazione residua e che il modello è statisticamente solido per essere utilizzato come benchmark.

Tabella 3.1: MSE delle previsioni sulla temperatura media europea.

Orizzonte	Chronos FT	Chronos ZS	ETS	Naïve	SARIMA	S. Naïve
$h = 1$	1.3959	11.1253	**1.2951**	13.3865	1.3789	2.5844
$h = 2$	1.4501	11.0071	**1.3199**	44.8558	1.5049	2.5941
$h = 3$	1.4602	10.9934	**1.3325**	86.7857	1.5361	2.6020
$h = 4$	1.4472	11.0844	**1.3518**	129.2575	1.5535	2.6078
$h = 5$	1.4899	11.1875	**1.3561**	160.6525	1.5583	2.6182
$h = 6$	1.4865	11.4186	**1.3641**	172.5777	1.5619	2.6296
$h = 7$	1.5056	11.3382	**1.3710**	161.8724	1.5691	2.6412
$h = 8$	1.5349	11.6749	**1.3801**	131.0365	1.5764	2.6482
$h = 9$	1.5429	11.6476	**1.3703**	88.4785	1.5666	2.6158
$h = 10$	1.5325	11.4714	**1.3665**	45.2776	1.5644	2.6260
$h = 11$	1.4922	11.1608	**1.3560**	13.6948	1.5550	2.6359
$h = 12$	1.4915	11.1677	**1.3608**	2.6477	1.5615	2.6477
$h = 1, \ldots, 12$	1.4858	11.2731	**1.3520**	87.5436	1.5406	2.6209

Nel presente dataset l'esperimento di previsione ha origine in corrispondenza dell'osservazione t = 545 (relativa a maggio 2005), e si estende fino al tempo t = 778 (ottobre 2024). Esaminando i risultati empirici mostrati nelle tabelle 3.1 e 3.2 emergono alcune informazioni interessanti. In primo luogo, si osserva come il modello *ETS* fornisca le migliori performance predittive per entrambi gli indicatori, dimostrando una spiccata capacità di adattamento a dataset caratterizzati dalla presenza di una forte componente stagionale. Nello specifico, l'accuratezza massima del modello si registra in corrispondenza dell'orizzonte temporale $h = 1$, dove si evidenzia un valore del *MSE* pari a 1.2951 e del *MAE* pari a 0.8654.

Un elemento di rilievo per l'analisi è rappresentato dalle prestazioni del modello *fine-tuned*, il quale, assieme al modello *SARIMA*, rappresenta il secondo miglior modello dell'esperimento. Questi ultimi presentano errori solo leggermente superiori rispetto al modello *ETS*, raggiungendo il loro apice predittivo anch'essi in corrispondenza dell'orizzonte $h = 1$ e presentando valori di *MSE* aggregati pari a 1.4858, per il modello *fine-tuned*, e 1.5406 per il modello *SARIMA*, e di *MAE* aggregati pari a 0.9579, per il modello *fine-tuned*,

Tabella 3.2: MAE delle previsioni sulla temperatura media europea.

Orizzonte	Chronos FT	Chronos ZS	ETS	Naïve	SARIMA	S. Naïve
$h = 1$	0.9198	2.9726	**0.8654**	3.2205	0.9031	1.2290
$h = 2$	0.9368	2.9533	**0.8747**	6.0115	0.9433	1.2318
$h = 3$	0.9425	2.9532	**0.8808**	8.3903	0.9611	1.2334
$h = 4$	0.9374	2.9667	**0.8824**	10.1359	0.9637	1.2339
$h = 5$	0.9551	2.9735	**0.8841**	11.3323	0.9647	1.2371
$h = 6$	0.9563	2.9973	**0.8870**	11.6404	0.9648	1.2425
$h = 7$	0.9605	2.9859	**0.8890**	11.3977	0.9692	1.2476
$h = 8$	0.9772	3.0368	**0.8913**	10.2354	0.9734	1.2486
$h = 9$	0.9884	3.0451	**0.8900**	8.4908	0.9692	1.2401
$h = 10$	0.9851	3.0230	**0.8882**	6.0379	0.9672	1.2431
$h = 11$	0.9713	2.9747	**0.8857**	3.2285	0.9630	1.2457
$h = 12$	0.9645	2.9730	**0.8880**	1.2512	0.9651	1.2512
$h = 1, \ldots, 12$	0.9579	2.9879	**0.8839**	7.6144	0.9590	1.2403

e 0.9590, per il modello *SARIMA*, valori solo lievemente differenti rispetto ai valori 1.3520 e 0.8839 forniti dal modello *ETS* . Di contro, il modello *zero-shot* presenta invece risultati nettamente peggiori con valori di *MSE* e *MAE* aggregati pari a 11.2731 e 2.9879, ben più alti rispetto a quelli dei modelli appena citati.

Il modello decisamente meno efficiente si rivela essere il *Naïve*, con valori dei due indicatori nettamente maggiori rispetto a tutti i modelli confrontati (*MSE* aggregato pari a 87.5436 e *MAE* aggregato pari a 7.6144). Risulta però importante notare come il modello presenti una dinamica d'errore a parabola, con picchi massimi nelle fasi centrali delle previsioni (in corrispondenza di h compreso tra 3 e 10) evidenziando la sua incapacità intrinseca di cogliere le fluttuazioni stagionali del dataset, presentando un valore di *MSE* in corrispondenza dell'orizzonte previsivo $h = 6$ addirittura pari a 172.5777.

L'analisi empirica di questo primo dataset evidenzia dunque un risultato di particolare rilievo: di fronte a una serie storica caratterizzata da una forte componente stagionale, tipica di dati metereologici, il modello *ETS* dimostra una capacità di scomposizione e previsione della stagionalità estremamente precisa, superando le preformance dei modelli di *deep learning* basati su un'architettura *transformer* estremamente complessa. Questo risultato pro-

pone quindi uno spunto di riflessione interessante, dimostrando come determinati modelli locali, in presenza di dataset con componenti strutturali ben definite e semplici da prevedere, riescono a superare in termini di capacità predittive anche i modelli più complessi.

3.2 Analisi del consumo energetico italiano

Il secondo dataset analizzato nell'elaborato comprende le osservazioni mensili riguardanti il consumo energetico in Italia (in migliaia di kwh) nel periodo compreso tra gennaio 1980 e dicembre 1995, rappresentando una serie storica molto più ristretta rispetto a quella precedentemente analizzata. I dati provengono dalle serie storiche ufficiali sul fabbisogno elettrico nazionale raccolte e conservate da *Terna S.p.A.*, gestore della rete di trasmissione nazionale (Terna S.p.A. 2026). L'obiettivo di questa scelta è quello di valutare la robustezza e le capacità predittive di Chronos-Bolt in condizioni di scarsità di dati storici nel set di training. Come evidenziato dallo studio di Montero-Manso e Hyndman 2021, infatti, in contesti caratterizzati da campioni ridotti (*low-data regimes*), i modelli tradizionali tendono spesso a superare i modelli globali in termini di accuratezza predittiva. Questi ultimi, infatti, data la loro complessa architettura, necessitano solitamente di una mole di dati storici molto più ampia per sintonizzarsi correttamente e adattarsi al dataset oggetto di studio.

Dall'analisi visiva della serie storica si nota chiaramente la presenza di una forte componente stagionale, caratterizzata da ricorrenti picchi di consumo energetico in determinati periodi dell'anno. Inoltre, nonostante l'orizzonte temporale sia di soli 15 anni, si osserva come nel tempo il consumo medio sia cresciuto notevolmente, evidenziando un chiaro trend positivo. Per modellare questa serie è stato selezionato il modello SARIMA$(0,1,1)(0,1,1)_{12}$. Questa specifica è stata individuata come la migliore sia in termini di parsimonia che adeguatezza, presentando un valore dell'indice AIC pari a 125.917 e un p-value relativo al test di Ljung-Box pari a 0.838.

Nel dataset corrente l'origine dell'esperimento di previsione è stata fissata in corrispondenza dell'osservazione $t = 125$, corrispondente a maggio 1990, consentendo l'esecuzione di 67 cicli di previsione. L'analisi comparativa dei

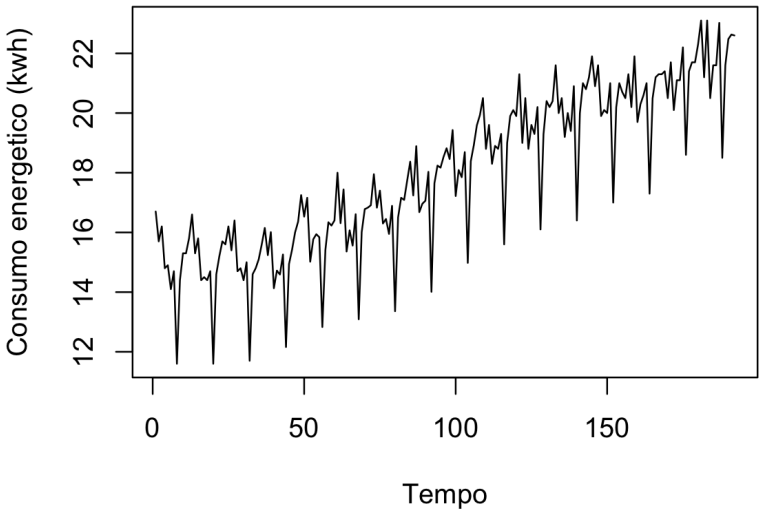

Figura 3.2: Consumo energetico italiano (1980-1995).

Tabella 3.3: MSE delle previsioni sul consumo energetico mensile in Italia.

Orizzonte	**Chronos FT**	**Chronos ZS**	**ETS**	**Naïve**	**SARIMA**	**S. Naïve**
$h = 1$	0.2948	0.2316	0.2233	3.2441	**0.1594**	0.4040
$h = 2$	0.4776	0.2504	0.3561	2.8644	**0.1918**	0.4063
$h = 3$	0.5947	0.3164	0.4953	3.2854	**0.2215**	0.4001
$h = 4$	0.6889	0.3471	0.6153	2.7547	**0.2620**	0.4024
$h = 5$	0.8505	0.3470	0.6839	3.9959	**0.2913**	0.4074
$h = 6$	1.2105	0.3958	0.6701	2.7301	**0.3113**	0.4099
$h = 7$	1.3433	0.4019	0.6267	4.1536	**0.3441**	0.4165
$h = 8$	1.4386	0.4350	0.5418	2.7217	**0.3797**	0.4193
$h = 9$	1.6606	0.4396	0.4421	3.0641	**0.3931**	0.4248
$h = 10$	1.7252	0.4384	0.3513	2.9096	**0.4410**	0.4149
$h = 11$	1.7178	0.4448	**0.2859**	3.3723	0.4537	0.4222
$h = 12$	1.9719	0.4683	**0.3036**	0.4269	0.4808	0.4269
$h = 1, \ldots, 12$	1.1645	0.3763	0.4663	2.9602	**0.3275**	0.4129

Tabella 3.4: MAE delle previsioni sul consumo energetico mensile in Italia.

Orizzonte	Chronos FT	Chronos ZS	ETS	Naïve	SARIMA	S. Naïve
$h = 1$	0.4390	0.4075	0.3944	1.3060	**0.3303**	0.5273
$h = 2$	0.5891	0.4149	0.4821	1.2268	**0.3596**	0.5277
$h = 3$	0.6796	0.4422	0.5752	1.3245	**0.3738**	0.5220
$h = 4$	0.7284	0.4837	0.6165	1.1977	**0.4020**	0.5223
$h = 5$	0.8381	0.4650	0.6363	1.3738	**0.4140**	0.5259
$h = 6$	1.0170	0.5046	0.6424	1.3111	**0.4282**	0.5263
$h = 7$	1.0675	0.5168	0.6286	1.3667	**0.4720**	0.5333
$h = 8$	1.0988	**0.5078**	0.6114	1.2478	0.5147	0.5338
$h = 9$	1.1913	0.5202	0.5453	1.2961	**0.5160**	0.5378
$h = 10$	1.2165	0.5232	**0.4812**	1.2598	0.5519	0.5298
$h = 11$	1.2105	0.5088	**0.4024**	1.2812	0.5565	0.5391
$h = 12$	1.3097	0.5181	**0.4267**	0.5416	0.5857	0.5416
$h = 1, \ldots, 12$	0.9488	0.4844	0.5369	1.2278	**0.4587**	0.5306

modelli, i cui risultati in termini di *MSE* e *MAE* sono riportati nelle Tabelle 3.3 e 3.4, evidenzia dinamiche differenti rispetto a quanto osservato nel primo dataset. In questo specifico contesto, caratterizzato da un campione più ristretto, il modello locale *SARIMA* presenta i risultati migliori. La massima accuratezza predittiva si registra in corrispondenza dell'orizzonte temporale $h = 1$, dove il modello ottiene i valori minimi sia per l'*MSE*, 0.1594, che per l'*MAE*, 0.3303. Tuttavia, la superiorità del modello *SARIMA* non si estende uniformemente all'intero orizzonte di previsione. Si possono osservare, infatti, dei deterioramenti delle sue performance predittive all'aumentare dell'orizzonte h. Negli step predittivi più distanti, corrispondenti a $h = 11$ e $h = 12$, il modello *ETS* dimostra una maggiore robustezza, superando il *SARIMA* in entrambe le metriche d'errore.

Un aspetto estremamente interessante riguarda il comportamento di Chronos. La versione *zero-shot* dimostra un netto miglioramento delle capacità predittive nella presente serie storica, posizionandosi su livelli di errore molto vicini a quelli dei modelli locali tradizionali, presentando un *MAE* aggregato pari a 0.4844 contro lo 0.4587 del modello *SARIMA*. Al contrario, la variante *fine-tuned* dimostra valori di errore significativamente superiori, registrando performance raddoppiate in termini di *MAE* aggregato, pari a 0.9488.

Tale evidenza suggerisce la presenza di un problema frequente nei modelli globali, noto come *overfitting*: l'elevata complessità parametrica dell'architettura *transformer*, se combinata con un dataset di training numericamente limitato, conduce alla memorizzazione del rumore stocastico di fondo, compromettendo le capacità di generalizzazione del modello nello spazio fuori campione (Montero-Manso e Hyndman 2021).

Infine, le performance peggiori sono ancora una volta associate al modello *Naïve*, il quale presenta valori aggregati di *MSE* e *MAE* pari a 2.9602 e 1.2278, dimostrando capacità predittive peggiori rispetto a tutti i modelli considerati, pur avvicinandosi, in termini di *MAE*, al modello *fine-tuned*. Un elemento di forte interesse scientifico emerge, per l'appunto, dal confronto tra le performance predittive del modello *Naïve* e quelle del modello *fine-tuned*. Sebbene le due metodologie poggino su paradigmi teorici e computazionali opposti, i loro output previsivi mostrano un comportamento per certi versi analogo, presentando *MAE* aggregati nettamente superiori rispetto agli altri modelli confrontati, dimostrando come l'effetto dell'*overfitting* subito da Chronos *fine-tuned* annulli quasi completamente il vantaggio tecnologico del modello globale. In questo scenario, dunque, la conoscenza generalista del modello *zero-shot* si rivela più solida di un addestramento globale insufficiente, avvicinandosi, e superandolo in termini di *MAE* all'orizzonte temporale $h = 8$, alle performance del modello *SARIMA*.

3.3 Previsioni sui volumi di vendita Walmart

Il terzo e ultimo dataset oggetto di studio comprende le rilevazioni giornaliere dei volumi di vendita registrati da dieci store Walmart dislocati in tre stati americani (California, Texas e Winsconsin), nell'arco temporale compreso tra il 29 gennaio 2011 e il 22 maggio 2016. I dati provengono dal dataset *M5 forecasting*, reso disponibile da Walmart per la quinta edizione della competizione di previsione organizzata da *Makridakis Open Forecasting Center*. L'analisi condotta in questa sezione delinea un cambio di paradigma metodologico all'interno dell'elaborato. A differenza delle serie storiche precedentemente esaminate, infatti, il dataset viene qui esteso mediante l'in-

tegrazione di covariate specifiche al fine di valutare l'efficacia predittiva e la robustezza dei *foundation models* anche in presenza di variabili esogene.

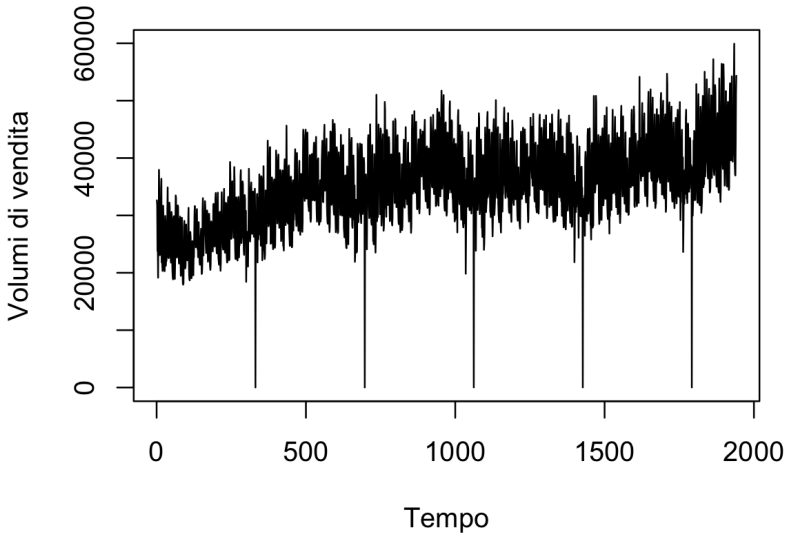

Figura 3.3: Volumi di vendita Walmart (2011-2016).

Dall'analisi visiva della serie storica emergono alcune caratteristiche strutturali di fondamentale interesse. In primo luogo, si evidenzia la presenza di cinque picchi negativi in corrispondenza dei quali i volumi di vendita si azzerano completamente. Tali anomalie coincidono sistematicamente con il giorno di natale degli anni compresi nel campione (2011-2015), l'unica ricorrenza annuale in cui i punti vendita Walmart rimangono chiusi al pubblico, e rappresentano indubbiamente dei valori *outliers*, i quali sono stati trattati attraverso un processo di *interpolazione*, permettendo così a tutti gli algoritmi di mantenere l'allineamento perfetto con le covariate esterne e la corretta struttura dei dati. Al di la di tali discontinuità, la serie presenta un evidente trend positivo e una componente stagionale ad alta frequenza particolarmente definita.

Per modellare al meglio la serie storica considerata è stato selezionato, sempre osservando i due indicatori considerati nelle analisi precedenti, il mo-

dello SARIMAX$(1,0,0)(0,1,1)_7$. Come detto in precedenza, il dataset analizzato e i modelli considerati per l'analisi sono stati integrati con le seguenti variabili esogene: "wday", la quale indica il giorno della settimana (da 1 a 7), introdotta per modellare la stagionalità di breve periodo tipica del settore retail, "month", indicatore mensile volto a isolare i macro-trend stagionali su base annua, "is_event", variabile binaria atta a segnalare la concomitanza di festività o eventi speciali, e infine "snap_CA", "snap_TX" e "snap_WI", tre variabili binarie corrispondenti ai tre stati considerati indicanti i giorni di erogazione dei sussidi per le famiglie con basso reddito.

L'origine dell'esperimento di previsione per il presente dataset è individuata dall'osservazione al tempo $t = 1359$, corrispondente al 18 ottobre 2014. Da questo punto di partenza si sviluppano 582 cicli di previsione con un orizzonte temporale mobile pari a $h = 30$, fino a raggiungere il termine della serie storica al tempo $t = 1941$, ovvero il 22 maggio 2016.

Nelle Tabelle 3.5 e 3.6 sono presentati i valori degli indicatori *MSE* e *MAE* ottenuti attraverso l'analisi empirica. Osservando i risultati si nota subito la predominanza del modello Chronos-Bolt nella versione *zero-shot*, la quale, in questo specifico dataset con covariate, non si limita a generare una previsione alla cieca sulla base della sola serie storica passata, ma combina la potenza predittiva dello *zero-shot forecasting* con uno studio analitico delle covariate fornite in fase di addestramento. Questo modello registra i suoi valori migliori in corrispondenza del primo orizzonte previsivo ($h = 1$), con un valore di *MSE* pari a 6773967 e di *MAE* pari a 1912.521. La superiorità della configurazione *zero-shot* si mantiene inalterata per tutti i trenta passi previsivi considerati, conducendo a un MSE aggregato pari a 8889647 e un MAE aggregato pari a 2233.709. Si rileva tuttavia un peggioramento delle performance predittive al dilatarsi dell'orizzonte temporale, presentando però valori dei due errori sempre molto simili tra loro e mantenendo comunque valori più bassi rispetto ai modelli utilizzati per il confronto. Di cruciale interesse scientifico risulta inoltre il confronto con la seconda versione di Chronos, ovvero la versione *fine-tuned*. Quest'ultima fa registrare risultati solo lievemente peggiori rispetto al modello *zero-shot*, presentando valori di *MSE* e *MAE* aggregati pari a, rispettivamente, 11660794 e 2560.676, e presentando, in corrispondenza degli ultimi orizzonti predittivi, valori di *MAE* vicini o addirittura più alti del

modello *ETS* .

Il confronto tra risultati dei due modelli globali e dei modelli locali considerati nello studio rappresenta un ulteriore risultato di estrema rilevanza. La costante superiorità dei modelli globali dimostra come i meccanismi legati alla loro architettura *transformer* siano in grado di cogliere le fluttuazioni dei dati e gli shock legati all'introduzione delle variabili esogene, superando l'efficacia e la semplicità dei modelli tradizionali locali. Quasi tutti i modelli locali presentano, infatti, dei risultati, in termini di capacità predittive, peggiori rispetto ai due modelli globali considerati.

I valori dei due indicatori confermano ancora una volta la scarsa qualità previsiva del modello *Naïve*, il quale, anche in questa circostanza, presenta capacità predittive ben peggiori rispetto a tutti i modelli considerati nello studio, con valori aggregati di *MSE* pari a 72845270 e di *MAE* pari a 6760.198. Presenta, inaspettatamente, dei valori simili il modello *SARIMAX*, il quale non dimostra miglioramenti in termini di capacità predittive rispetto ai dataset precedentemente considerati, un insuccesso causato probabilmente dall'elevata complessità del dataset analizzato.

In conclusione, lo studio empirico di questo terzo dataset consolida l'assunto teorico secondo cui fenomeni caratterizzati da un'elevata complessità strutturale ed informativa trovano una migliore rappresentazione in architetture altrettanto complesse. Al contrario, scenari caratterizzati da dinamiche più lineari possono essere modellati più efficacemente da approcci locali.

Tabella 3.5: MSE delle previsioni sui volumi di vendita Walmart.

Orizzonte	Chronos FT	Chronos ZS	ETS	Naïve	SARIMAX	S. Naïve
$h = 1$	8208519	**6773967**	9447643	36828622	21476676	18631311
$h = 2$	9022883	**7682940**	12237581	83947558	37112240	18644919
$h = 3$	9237029	**7735635**	13742722	108742409	46253566	18674144
$h = 4$	10013074	**7885065**	14394735	110151383	46541034	18690025
$h = 5$	10083058	**7783387**	14555355	88284118	43557409	18654287
$h = 6$	10316443	**7787770**	14453102	44215398	42385577	18683172
$h = 7$	10815517	**7816370**	15077621	18715008	43166126	18715008
$h = 8$	10926039	**8082532**	16169843	44514783	47402556	24271925
$h = 9$	11153965	**8190896**	17502418	89021087	50638144	24239628
$h = 10$	10766747	**8127779**	18492113	112888818	52603413	24272375
$h = 11$	11238006	**8298819**	19273720	113972263	53143402	24285308
$h = 12$	10866515	**8296123**	19882264	92730768	53063112	24250736
$h = 13$	10686240	**8441516**	20171440	50209103	53143023	24293274
$h = 14$	10738888	**8386837**	20255048	24335526	53380186	24335526
$h = 15$	10912385	**8659355**	20859682	51075128	54124321	21933127
$h = 16$	11100218	**9056250**	20376849	93191935	54624147	21971168
$h = 17$	11229899	**9192590**	19973555	115058729	55009250	21965674
$h = 18$	12231743	**9548487**	19657907	115307311	55163423	22004247
$h = 19$	12858025	**9578428**	18996220	92438750	55164924	22023300
$h = 20$	13091569	**9794959**	17993425	48929902	55260289	22043298
$h = 21$	13487783	**9630871**	16990911	22079488	55391415	22079488
$h = 22$	13313305	**9850254**	16542037	46624240	55662168	15150716
$h = 23$	13377802	**9869437**	16119197	89332980	55836715	15177039
$h = 24$	12783418	**9854241**	15316391	110492252	56022196	15202858
$h = 25$	13280484	**10097553**	14931989	111283699	56125580	15230168
$h = 26$	13253537	**9949228**	13979888	88933420	56165570	15232903
$h = 27$	13401102	**10080281**	12334254	44191217	56251924	15257690
$h = 28$	13997412	**9960001**	10727296	15282182	56358273	15282182
$h = 29$	13895872	**10143935**	10681149	40297311	56590709	20446125
$h = 30$	13536342	**10133889**	10541566	82282720	56722990	20464427
$h = 1, \ldots, 30$	11660794	**8889647**	16055944	72845270	51144679	20070202

Tabella 3.6: MAE delle previsioni sui volumi di vendita Walmart.

Orizzonte	Chronos FT	Chronos ZS	ETS	Naïve	SARIMAX	S. Naïve
$h = 1$	2112.628	**1912.521**	2186.069	4609.410	3707.822	3310.319
$h = 2$	2213.994	**2051.692**	2505.329	7678.390	5081.113	3310.373
$h = 3$	2246.732	**2069.947**	2653.849	8739.137	5491.615	3313.810
$h = 4$	2356.621	**2104.858**	2795.774	8764.820	5197.378	3314.208
$h = 5$	2368.839	**2094.451**	2845.833	7804.380	4930.294	3309.082
$h = 6$	2393.901	**2102.650**	2891.160	5299.903	4973.324	3312.350
$h = 7$	2458.345	**2110.033**	3043.546	3317.005	5126.020	3317.005
$h = 8$	2458.053	**2139.460**	3222.008	5190.506	5361.903	4155.182
$h = 9$	2496.332	**2145.286**	3272.047	7805.850	5527.943	4151.016
$h = 10$	2452.610	**2138.422**	3522.255	8874.732	5609.528	4154.163
$h = 11$	2510.317	**2152.382**	3614.585	8924.326	5635.297	4154.239
$h = 12$	2463.735	**2149.881**	3695.157	7986.830	5652.195	4149.888
$h = 13$	2433.206	**2172.532**	3758.983	5667.518	5680.518	4156.667
$h = 14$	2451.345	**2164.776**	3836.301	4162.975	5706.819	4162.975
$h = 15$	2472.420	**2196.924**	3851.138	5739.569	5745.294	3725.977
$h = 16$	2505.806	**2248.637**	3757.872	8027.181	5765.501	3731.398
$h = 17$	2526.320	**2289.335**	3718.722	8979.096	5786.832	3729.137
$h = 18$	2644.986	**2338.995**	3611.490	8940.101	5795.544	3734.874
$h = 19$	2725.332	**2346.538**	3491.585	7927.941	5796.159	3735.539
$h = 20$	2753.433	**2374.445**	3380.535	5492.285	5807.151	3736.339
$h = 21$	2820.400	**2352.084**	3248.757	3740.630	5819.455	3740.630
$h = 22$	2776.050	**2377.445**	3129.741	5376.200	5832.837	2875.243
$h = 23$	2774.959	**2373.956**	3042.465	7809.658	5839.336	2879.177
$h = 24$	2704.527	**2366.710**	2959.198	8838.212	5850.318	2882.745
$h = 25$	2764.601	**2401.121**	2908.479	8854.655	5856.637	2887.683
$h = 26$	2746.134	**2371.813**	2815.808	7788.314	5860.336	2886.214
$h = 27$	2752.430	**2382.521**	2597.742	5128.599	5866.176	2889.213
$h = 28$	2831.468	**2346.271**	2416.906	2892.071	5874.397	2892.071
$h = 29$	2822.110	**2368.826**	2443.736	4901.156	5885.251	3351.091
$h = 30$	2782.822	**2367.071**	2423.622	7544.501	5888.877	3351.330
$h = 1, \ldots, 30$	2560.676	**2233.709**	3124.690	6760.198	5565.062	3509.998

Capitolo 4

Conclusioni

Nel presente elaborato è stato condotto un rigoroso confronto empirico tra capacità predittive del modello globale *Chronos-Bolt* nelle sue due versioni principali, *zero-shot* e *fine-tuned*, e specifici modelli locali quali *SARIMA*, *ETS*, *Naïve* e *Seasonal Naïve*. L'esperimento è stato strutturato attraverso un'infrastruttura algoritmica iterativa basata su una strategia a finestra mobile, valutando e confrontando l'accuratezza dei modelli mediante l'analisi delle metriche d'errore *MAE* e *MSE*.

Le evidenze emerse dall'analisi smentiscono l'assunto secondo cui una maggiore complessità computazionale apporti sistematicamente un miglioramento in termini di accuratezza predittiva. L'esperimento dimostra infatti come, in svariate situazioni, gli approcci locali riescano a ottenere dei valori delle due metriche migliori, anche se con scarti contenuti, rispetto ai modelli globali. Osservando i risultati dei primi due casi di studio, riguardanti la temperatura media europea e il consumo energetico italiano, si nota come i modelli locali *ETS* e *SARIMA* mostrino capacità predittive superiori a *Chronos*, nonostante quest'ultimo, con la versione *fine-tuned* nel primo, e con la versione *zero-shot* nel secondo, si avvicini alle loro performance predittive. Il secondo dataset rappresenta la prova più chiara della veridicità di questo concetto, in presenza di un numero ridotto di osservazioni, infatti, la variante *fine-tuned* di *Chronos* mostra un severo deterioramento delle metriche, raddoppiando l'errore medio aggregato e rendendo il modello comparabile ad un semplice modello *Naïve*, confutando la teoria secondo cui maggiore

complessità equivale sempre a maggiori risultati.

Se nei primi due casi i modelli classici hanno avuto la meglio, nel terzo dataset la situazione si ribalta completamente. Nell'analisi delle vendite di Walmart, grazie a una quantità di dati più elevata e all'inserimento di variabili esogene, i modelli globali dominano nettamente. La versione *zero-Shot* di *Chronos* supera tutte le altre metodologie per l'intero orizzonte di previsione. Anche la versione *fine-tuned* mostra ottimi risultati, staccandosi di poco dalla versione *zero-shot*, mentre quasi tutti i modelli locali registrano performance decisamente peggiori.

In conclusione, i dati proposti evidenziano dei risultati di cruciale importanza per l'evoluzione del mondo del *forecasting*, delineando un quadro in cui ancora non esiste un modello universale superiore. Nei primi due dataset le performance predittive di *Chronos* riescono ad avvicinarsi di molto a quelle dei modelli locali, tuttavia senza mai superarli. Si tratta di due dataset relativamente semplici, che presentano dinamiche strutturali evidenti e semplici da prevedere, questo dimostra come serie storiche poco complesse possano ancora essere ben descritte con grande capacità dai modelli locali tradizionali, senza bisogno di architetture estremamente elaborate. Il terzo dataset considerato invece rappresenta il vero e proprio cambio di paradigma. Si tratta di un dataset molto vasto al quale sono integrate una serie di variabili esogene, una complessità che le due versioni di *Chronos* hanno compreso ed elaborato al meglio, mostrando capacità nettamente superiori ai modelli di confronto e confermando come i modelli globali in dataset di complessità più elevata dimostrino un vero e proprio miglioramento nel campo del *forecasting*.

Bibliografia

Ansari, Abdul Fatir et al. (2024). «Chronos: Learning the language of time series». In: *arXiv preprint arXiv:2403.07815*.

Auer, Andreas et al. (2025). «Zero-shot time series forecasting with covariates via in-context learning». In: *arXiv preprint arXiv:2506.03128*.

AWS Team (2021a). *What is Deep Learning?* URL: https://aws.amazon.com/what-is/deep-learning/?trk=faq_card.

— (2021b). *What is Machine Learning?* URL: https://aws.amazon.com/what-is/machine-learning/?trk=faq_card.

— (2023a). *What are Foundation Models?* URL: https://aws.amazon.com/what-is/foundation-models/?trk=faq_card.

— (2023b). *What is Transfer Learning?* URL: https://aws.amazon.com/what-is/transfer-learning/?trk=faq_card.

Belcic, Ivan e Cole Stryker (2026). *Che cos'è AutoML?* URL: https://www.ibm.com/it-it/think/topics/automl.

Bengio, Yoshua, Ian Goodfellow, Aaron Courville et al. (2017). *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA.

Bommasani, Rishi et al. (2021). «On the opportunities and risks of foundation models». In: *arXiv preprint arXiv:2108.07258*.

Brown, Tom et al. (2020). «Language models are few-shot learners». In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Cornes, Richard C et al. (2018). «An ensemble version of the E-OBS temperature and precipitation data sets». In: *Journal of Geophysical Research: Atmospheres* 123.17, pp. 9391–9409.

Hyndman, Rob J et al. (2002). «A state space framework for automatic forecasting using exponential smoothing methods». In: *International Journal of forecasting* 18.3, pp. 439–454.

Montero-Manso, Pablo e Rob J Hyndman (2021). «Principles and algorithms for forecasting groups of time series: Locality and globality». In: *International Journal of Forecasting* 37.4, pp. 1632–1653.

Murel, Jacob e Eda Kavlakoglu (2024). *Cos'è l'apprendimento d'insieme?* URL: https://www.ibm.com/it-it/think/topics/ensemble-learning.

Nie, Yuqi et al. (2022). «A time series is worth 64 words: Long-term forecasting with transformers». In: *arXiv preprint arXiv:2211.14730*.

Salehin, Imrus et al. (2024). «AutoML: A systematic review on automated machine learning with neural architecture search». In: *Journal of Information and Intelligence* 2.1, pp. 52–81.

Shchur, Oleksandr et al. (2023). «AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting». In: *International Conference on Automated Machine Learning*. PMLR, pp. 9–1.

Terna S.p.A. (2026). *Dati Statistici sull'Energia Elettrica in Italia - Serie Storiche del Fabbisogno Nazionale*. Disponibile online al sito ufficiale di Terna. Accesso ai dati effettuato tramite materiale didattico accademico.

Ultralytics Team (2024). *What is Ensemble Learning? Improve AI Accuracy.* URL: https://www.ultralytics.com/it/glossary/ensemble.

Appendice A

Codice R e Python utilizzato

La seguente Appendice presenta il codice R e Python utilizzato per lo svolgimento del confronto empirico tra i modelli globali e i modelli locali. Le procedure computazionali riportate nelle sezioni A.1 e A.2 fanno riferimento principalmente ai dataset relativi alla temperatura media europea e ai volumi di vendita di Walmart. Per quanto riguarda il terzo dataset analizzato, riguardante il consumo energetico medio italiano, il codice risulta sostanzialmente analogo a quello utilizzato per il primo dataset considerato (Temperatura media europea). Nella sezione A.3 viene invece presentato il codice utilizzato per il calcolo degli indici *MSE* e *MAE* e la creazione delle tabelle per il confronto tra i modelli locali e i modelli globali. I comandi riportati fanno riferimento esclusivamente al dataset riguardante la temperatura media europea, per gli altri dataset considerati sono stati applicati comandi sostanzialmente analoghi.

A.1 Modellistica e analisi in ambiente Python

A.1.1 Temperatura media europea (1960-2024)

Listing A.1: Caricamento dataset di training

```
from autogluon.timeseries import TimeSeriesDataFrame, TimeSeriesPredictor
```

```
train_data = TimeSeriesDataFrame.from_path(
    "temperature_545.csv",
    id_column="item_id",
    timestamp_column="timestamp",
)
```

Listing A.2: Inizializzazione del modello Zero-Shot

```
predictor = TimeSeriesPredictor(prediction_length=12, target="temperature"
    ).fit(train_data, presets="bolt_base")
)
```

Listing A.3: Caricamento dataset completo

```
test_data = TimeSeriesDataFrame.from_path(
    "temperature_completo.csv",
    id_column="item_id",
    timestamp_column="timestamp",
)
```

Listing A.4: implementazione dell'algoritmo iterativo per il calcolo delle previsioni Zero-Shot

```
import pandas as pd
current_data = train_data.copy()
all_rolling_forecasts = []
n_iterations = 778 - 545
for i in range(n_iterations):
    forecast = predictor.predict(current_data, model="Chronos[bolt_base]")
    forecast_row = forecast['mean'].values.tolist()
    all_rolling_forecasts.append(forecast_row)
    indice_prossimo_punto = 545 + i
    next_real_point = test_data.iloc[[indice_prossimo_punto]]
    current_data = pd.concat([current_data, next_real_point])
```

Listing A.5: Salvataggio previsioni in un file csv

```
df_temp = pd.DataFrame(all_rolling_forecasts)
df_risultati = df_temp.iloc[:, :12].copy()
```

```
df_risultati.columns = [f"h_{h}" for h in range(1, 13)]
df_risultati.insert(0, "origine", range(545, 545 + len(df_risultati)))
df_risultati.to_csv("previsioni_chronos_ZS.csv", index=False)
```

Listing A.6: Implementazione del processo di Fine-Tuning

```
predictor = TimeSeriesPredictor(prediction_length=12, target="temperature"
    , eval_metric="MASE").fit(
    train_data,
    hyperparameters={
        "Chronos": [
            {"model_path": "bolt_small", "fine_tune": True, "ag_args": {"
                name_suffix": "FineTuned"}},
        ]
    },
    enable_ensemble=False,
    time_limit=600,
)
```

Listing A.7: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni Fine-tuned

```
import pandas as pd
current_data = train_data.copy()
all_rolling_forecasts = []
n_iterations = 778 - 545
for i in range(n_iterations):
    forecast = predictor.predict(current_data, model="ChronosFineTuned[
        bolt_small]")
    forecast_row = forecast['mean'].values.tolist()
    all_rolling_forecasts.append(forecast_row)
    indice_prossimo_punto = 545 + i
    next_real_point = test_data.iloc[[indice_prossimo_punto]]
    current_data = pd.concat([current_data, next_real_point])
```

Listing A.8: Salvataggio previsioni in un file csv

```
df_temp = pd.DataFrame(all_rolling_forecasts)
```

```
df_risultati = df_temp.iloc[:, :12].copy()
df_risultati.columns = [f"h_{h}" for h in range(1, 13)]
df_risultati.insert(0, "origine", range(545, 545 + len(df_risultati)))
df_risultati.to_csv("previsioni_chronos_FT.csv", index=False)
```

A.1.2 Volumi di vendita Walmart (2011-2016)

Listing A.9: Caricamento dataset di training

```
from autogluon.timeseries import TimeSeriesDataFrame, TimeSeriesPredictor

train_data = TimeSeriesDataFrame.from_path(
    "M5_dataset1359.csv",
    id_column="item_id",
    timestamp_column="date",
)
```

Listing A.10: Inizializzazione del modello Zero-Shot con covariate

```
predictor = TimeSeriesPredictor(
    prediction_length=30,
    eval_metric="MASE",
    target="sales",
    freq="D",
    known_covariates_names=["wday", "month", "is_event","snap_CA","snap_TX
        ","snap_WI"],
).fit(
    train_data,
    hyperparameters={
        "Chronos": [
            {"model_path": "bolt_small", "ag_args": {"name_suffix": "
                ZeroShot"}},
            {
                "model_path": "bolt_small",
                "covariate_regressor": "CAT",
                "target_scaler": "standard",
                "ag_args": {"name_suffix": "WithRegressor"},
```

```
            },
        ],
    },
    time_limit=600,
    enable_ensemble=False,
)
```

Listing A.11: Caricamento dataset completo

```
test_data = TimeSeriesDataFrame.from_path(
    "dataset_m5_esteso.csv",
    id_column="item_id",
    timestamp_column="date",
)
known_covariates = test_data.drop(columns=["sales"])
```

Listing A.12: implementazione dell'algoritmo iterativo per il calcolo delle previsioni Zero-Shot

```
import pandas as pd
from autogluon.timeseries import TimeSeriesDataFrame
current_data = train_data.copy()
all_rolling_forecasts = []
n_iterations = 1941 - 1359

for i in range(n_iterations):
    current_known_covariates = known_covariates.iloc[i:]
    forecast = predictor.predict(
        current_data,
        known_covariates=current_known_covariates,
        model="ChronosWithRegressor[bolt_small]"
    )
    forecast_row = forecast['mean'].values.tolist()
    all_rolling_forecasts.append(forecast_row)
    posizione len(train_data) + i
    next_real_row = test_data.iloc[[len(train_data) + i]].copy()
    current_data = pd.concat([current_data, next_real_row])
```

Listing A.13: Salvataggio previsioni in un file csv

```
df_previsioni = pd.DataFrame(all_rolling_forecasts, columns=[f'h_{j}' for
    j in range(1, 31)])
df_previsioni.insert(0, 'origine', range(1359, 1359 + n_iterations))
df_previsioni.to_csv("previsioni_M5_chronos_bolt_ZS.csv", index=False)
```

Listing A.14: Implementazione del processo di Fine-Tuning

```
predictor = TimeSeriesPredictor(
    prediction_length=30,
    eval_metric="MASE",
    target="sales",
    freq="D",
    known_covariates_names=["wday", "month", "is_event","snap_CA","snap_TX
        ","snap_WI"],
).fit(
    train_data,
    hyperparameters={
        "Chronos": [
            {
                "model_path": "bolt_small",
                "fine_tune": True,
                "covariate_regressor": "CAT",
                "target_scaler": "standard",
                "ag_args": {"name_suffix": "WithRegressor_FT"},
            },
        ],
    },
    time_limit=600,
    enable_ensemble=False,
)
```

Listing A.15: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni Fine-tuned

```
import pandas as pd
from autogluon.timeseries import TimeSeriesDataFrame
current_data = train_data.copy()
```

```
all_rolling_forecasts = []
n_iterations = 1941 - 1359

for i in range(n_iterations):
    current_known_covariates = known_covariates.iloc[i:]
    forecast = predictor.predict(
        current_data,
        known_covariates=current_known_covariates,
        model="ChronosWithRegressor_FT[bolt_small]"
    )
    forecast_row = forecast['mean'].values.tolist()
    all_rolling_forecasts.append(forecast_row)
    posizione len(train_data) + i
    next_real_row = test_data.iloc[[len(train_data) + i]].copy()
    current_data = pd.concat([current_data, next_real_row])
```

Listing A.16: Salvataggio previsioni in un file csv

```
df_previsioni = pd.DataFrame(all_rolling_forecasts, columns=[f'h_{j}' for
    j in range(1, 31)])
df_previsioni.insert(0, 'origine', range(1359, 1359 + n_iterations))
df_previsioni.to_csv("previsioni_M5_chronos_bolt_FT.csv", index=False)
```

A.2 Modellistica e analisi in ambiente R

Listing A.17: Librerie utilizzate nelle analisi

```
library(forecast)
library(sarima)
```

A.2.1 Temperatura media europea (1960-2024)

Listing A.18: Caricamento dati e generazione grafico della serie

```
dati <- read.csv("temperature_545.csv", header = T)
complete_dataset <- read.csv("temperature_completo.csv", header = T)
```

```
y <- as.matrix(dati$temperature)
plot(data$temperature, type = "l", xlab = "Tempo", ylab = "Temperatura")
```

Listing A.19: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello SARIMA

```
Algoritmo_previsioni_arima <- function(data, n_iterazioni, ordine,
    stagionale, complete_dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 12)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 12)
    current_fit <- Arima(data, order = ordine, seasonal = stagionale)
    previsione <- forecast(current_fit, h = 12)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[545+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:12)
  df_modello$origine <- 545:(545 + nrow(df_modello) - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:12))]
  write.csv(df_modello, "previsioni_sarima.csv", row.names = FALSE)
}

Algoritmo_previsioni_arima(y, 233, c(1, 0, 0), list(order = c(0, 1, 1),
    period = 12), complete_dataset$temperature )
```

Listing A.20: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello ETS

```
y_serie <- ts(dati$temperature, frequency = 12)
fit_ets <- ets(y_serie)
Algoritmo_previsioni_ets <- function(data, n_iterazioni, modello, complete
    _dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 12)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 12)
    current_fit <- ets(temp_ts, model = modello)
```

```
    previsione <- forecast(current_fit, h = 12)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[545+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:12)
  df_modello$origine <- 545:(545 + nrow(df_modello) - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:12))]
  write.csv(df_modello, "previsioni_ets.csv", row.names = FALSE)
}

Algoritmo_previsioni_ets(y, 233, fit_ets, complete_dataset$temperature)
```

Listing A.21: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello Seasonal-Naive

```
Algoritmo_previsioni_snaive <- function(data, n_iterazioni, complete_
    dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 12)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 12)
    previsione <- snaive(temp_ts, h = 12)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[545+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:12)
  df_modello$origine <- 545:(545 + nrow(df_modello) - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:12))]
  write.csv(df_modello, "previsioni_snaive.csv", row.names = FALSE)
}

Algoritmo_previsioni_snaive(y, 233, complete_dataset$temperature)
```

Listing A.22: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello Naive

```
Algoritmo_previsioni_naive <- function(data, n_iterazioni, complete_
    dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 12)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 12)
    previsione <- naive(temp_ts, h = 12)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[545+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:12)
  df_modello$origine <- 545:(545 + nrow(df_modello) - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:12))]
  write.csv(df_modello, "previsioni_naive.csv", row.names = FALSE)
}

Algoritmo_previsioni_naive(y, 233, complete_dataset$temperature)
```

A.2.2 Volumi di vendita Walmart (2011-2016)

Listing A.23: Pulizia dei dati dal valore corrispondente al 25 dicembre e interpolazione

```
dati <- read.csv("dataset_m5_totale.csv", header = T)
View(dati)

#eliminazione dei valori corrispondenti al 25-12
library(dplyr)
library(lubridate)
# Assicuriamoci che la colonna date sia riconosciuta come data reale
dati <- dati %>%
  mutate(date = as.Date(date))
# Sostituzione del 25 dicembre con NA nella colonna sales
```

```
dati_puliti <- dati %>%
  mutate(sales = if_else(format(date, "%m-%d") == "12-25", NA_real_, sales
      ))

dati_puliti <- dati_puliti %>%
  mutate(
    valore_prima = lag(sales, n = 1),
    valore_dopo  = lead(sales, n = 1)
  ) %>%
  mutate(
    sales = if_else(
      grepl("-12-25", date),                          # Cerca il testo
          -12-25 (Natale)
      (valore_prima + valore_dopo) / 2,
      sales
    )
  ) %>%
  select(-valore_prima, -valore_dopo)

View(dati_puliti)

write.csv(dati_puliti, "dataset_m5_esteso.csv", row.names = F)

#dati completi puliti
data_completo <- dati_puliti[1:1941,]
write.csv(data_completo, "M5_dataset_completo.csv", row.names = F)

#dati ridotti puliti
data_1359 <- data_completo[1:1359,]
write.csv(data_1359, "M5_dataset1359.csv", row.names = F)
```

Listing A.24: Caricamento dati e generazione del grafico

```
dati <- read.csv("M5_dataset1359.csv", header = T)
data_esteso <- read.csv("dataset_m5_esteso.csv", header = T)
complete_data <- read.csv("M5_dataset_completo.csv", header = T)
data_totale <- read.csv("M5_dataset_originale.csv", header = T)
```

```
plot(data_totale$sales, type = "l", xlab = "Tempo", ylab = "Volumi di
    vendita")
```

Listing A.25: Inserimento delle covariate in una tabella

```
known_covariates <- as.matrix(complete_data[, c("wday", "month", "is_event
    ", "snap_CA","snap_TX","snap_WI")])
all_covariates <- as.matrix(data_esteso[, c("wday", "month", "is_event", "
    snap_CA","snap_TX","snap_WI")])
```

Listing A.26: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello SARIMAX

```
Algoritmo_previsioni_arima <- function(target_data, n_iterazioni, ordine,
    stag, all_covariates, complete_dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 30)
  for (i in 1:n_iterazioni) {
    current_origin_idx <- 1359 + i - 1
    cov_train <- all_covariates[1:current_origin_idx, ]
    cov_future <- all_covariates[(current_origin_idx + 1):(current_origin_
        idx + 30), ]
    current_fit <- Arima(target_data, order = ordine, seasonal = stag,
        xreg = cov_train)
    previsione <- predict(current_fit, n.ahead = 30, newxreg = cov_future)
    matrice_previsioni[i, ] <- as.numeric(previsione$pred)
    nuovo_punto_reale <- complete_dataset$sales[current_origin_idx + 1]
    target_data <- c(target_data, nuovo_punto_reale)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:30)
  df_modello$origine <- 1359:(1359 + n_iterazioni - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:30))]
  write.csv(df_modello, "previsioni_sarimax.csv", row.names = FALSE)
}
ordine <- c(1,0,0)
stagionalitÃă <- c(0,1,1)
Algoritmo_previsioni_arima(dati$sales, 582, ordine, stagionalitÃă, all_
    covariates, data_esteso )
```

Listing A.27: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello ETS

```
y <- ts(dati$sales, frequency = 7)
fit2 <- ets(y)
summary(fit2)

Algoritmo_previsioni_ets <- function(data, n_iterazioni, modello, complete
    _dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 30)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 7)
    current_fit <- ets(temp_ts, model = modello)
    previsione <- forecast(current_fit, h = 30)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[1359+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:30)
  df_modello$origine <- 1359:(1359 + n_iterazioni - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:30))]
  write.csv(df_modello, "previsioni_M5_ets.csv", row.names = FALSE)
}

Algoritmo_previsioni_ets(y, 582, fit2, complete_data$sales)
```

Listing A.28: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello Seasonal-Naive

```
Algoritmo_previsioni_snaive <- function(data, n_iterazioni, complete_
    dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 30)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 7)
    previsione <- snaive(temp_ts, h = 30)
```

```
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[1359+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:30)
  df_modello$origine <- 1359:(1359 + n_iterazioni - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:30))]
  write.csv(df_modello, "previsioni_snaive.csv", row.names = FALSE)
}

Algoritmo_previsioni_snaive(y, 582, complete_data$sales)
```

Listing A.29: Implementazione dell'algoritmo iterativo per il calcolo delle previsioni con modello Naive

```
Algoritmo_previsioni_naive <- function(data, n_iterazioni, complete_
    dataset){
  matrice_previsioni <- matrix(NA, nrow = n_iterazioni, ncol = 30)
  for (i in 1:n_iterazioni) {
    temp_ts <- ts(data, frequency = 7)
    previsione <- naive(temp_ts, h = 30)
    matrice_previsioni[i, ] <- as.numeric(previsione$mean)
    nuovo_punto <- complete_dataset[1359+i]
    data <- c(data, nuovo_punto)
  }
  df_modello <- as.data.frame(matrice_previsioni)
  colnames(df_modello) <- paste0("h_", 1:30)
  df_modello$origine <- 1359:(1359 + n_iterazioni - 1)
  df_modello <- df_modello[, c("origine", paste0("h_", 1:30))]
  write.csv(df_modello, "previsioni_naive.csv", row.names = FALSE)
}

Algoritmo_previsioni_naive(y, 582, complete_data$sales )
```

Listing A.30: Algoritmo per la rimozione dei valori superflui

```
pulisci_previsioni <- function(nome_file, limite = 1941) {
```

```
  df <- read.csv(nome_file)
  for (h in 1:30) {
    colonna <- paste0("h_", h)
    df[df$origine + h > limite, colonna] <- NA
  }
  nuovo_nome <- paste0("pulito_", nome_file)
  write.csv(df, nuovo_nome, row.names = FALSE)
  cat("File", nuovo_nome, "creato con successo!\n")
}

pulisci_previsioni("previsioni_chronos_bolt_ZS.csv")
pulisci_previsioni("previsioni_chronos_bolt_FT.csv")
pulisci_previsioni("previsioni_ets.csv")
pulisci_previsioni("previsioni_snaive.csv")
pulisci_previsioni("previsioni_naive.csv")
pulisci_previsioni("previsioni_sarimax.csv")
```

A.3 Algoritmi iterativi per il calcolo di MSE e MAE

Listing A.31: Caricamento dei dati e creazione della lista dei file csv con le previsioni

```
data <- read.csv("temperature_completo.csv", header = T)
lista_file <- c("chronos_FT.csv", "chronos_ZS.csv", "ets.csv", "naive.csv"
    , "sarima.csv", "snaive.csv")
```

Listing A.32: Implementazione dell'algoritmo per la creazione della tabella di MSE

```
calcolo_mse <- function(n_rows, real_data, file_list, start_point){
  etichette_h <- c(paste0("h=", 1:n_rows), "h=1,...,12")
  tabella_finale <- data.frame(h = etichette_h)
  for (file in file_list) {
    data_modello <- read.csv(file, header = T, row.names = 1)
    mse_vettore <- numeric(n_rows)
    for (i in 1:n_rows) {
      real <- real_data[(start_point + i):length(real_data)]
```

```
      previsioni <- data_modello[,i]
      mse_vettore[i] <- mean((real - previsioni)^2, na.rm = T)
    }
    nome_modello <- gsub(".csv", "", file)
    colonna_modello <- c(mse_vettore, mean(mse_vettore))
    tabella_finale[[nome_modello]] <- colonna_modello
  }
  return(tabella_finale)
}

x <- calcolo_mse(12, data$temperature, lista_file, 545)
x
```

Listing A.33: Implementazione dell'algoritmo per la creazione della tabella di MAE

```
calcolo_mae <- function(n_rows, real_data, file_list, start_point){
  etichette_h <- c(paste0("h=", 1:n_rows), "h=1,...,12")
  tabella_finale <- data.frame(h = etichette_h)
  for (file in file_list) {
    data_modello <- read.csv(file, header = T, row.names = 1)
    mae_vettore <- numeric(n_rows)
    for (i in 1:n_rows) {
      real <- real_data[(start_point + i):length(real_data)]
      previsioni <- data_modello[,i]
      mae_vettore[i] <- mean(abs(real - previsioni), na.rm = T)
    }
    nome_modello <- gsub(".csv", "", file)
    colonna_modello <- c(mae_vettore, mean(mae_vettore))
    tabella_finale[[nome_modello]] <- colonna_modello
  }
  return(tabella_finale)
}

z <- calcolo_mae(12, data$temperature, lista_file, 545)
z
```