Dipartimento di Economia e Finanza

Indirizzo Banche ed Intermediari Finanziari

Machine Learning e Modelli a Stati Latenti per la Previsione dei Regimi di Mercato

Relatore:

Prof. Emilio Barone

Correlatore:

Prof.ssa Marta Catalano

Candidato:

Filippo Francesco Iuliano

783871

Anno Accademico 2024/2025

Indice

Introduzione

Negli ultimi decenni, l'evoluzione dei mercati finanziari e la crescente disponibilità di dati hanno reso sempre più rilevante l'applicazione di strumenti quantitativi avanzati per l'analisi e la previsione dei prezzi degli asset. In particolare, l'uso dell'intelligenza artificiale,e più specificamente del machine learning, ha aperto nuove possibilità nella modellazione delle dinamiche di mercato, consentendo di superare alcuni limiti intrinseci alle teorie classiche come l'Efficient Market Hypothesis (EMH).

Secondo l'EMH, i prezzi riflettono immediatamente e completamente tutte le informazioni disponibili, rendendo teoricamente impossibile battere il mercato in modo sistematico. Tuttavia, numerose evidenze empiriche e anomalie comportamentali hanno messo in discussione questo paradigma, suggerendo l'esistenza di inefficienze e pattern ricorrenti nei rendimenti. Tali osservazioni hanno stimolato l'interesse verso approcci alternativi, capaci di catturare regolarità latenti nei dati attraverso modelli flessibili e adattivi.

In questo contesto, il machine learning rappresenta un'opportunità concreta per identificare e prevedere regimi di mercato, come fasi rialziste (bull), ribassiste (bear) e laterali (static), attraverso l'elaborazione di ampi set di dati finanziari e macroeconomici. Modelli come le reti neurali ricorrenti (RNN), le Long Short-Term Memory (LSTM), XGBoost (XGB), Random Forest (RF) e i modelli a stati nascosti (Hidden Markov Models, HMM) si sono rivelati particolarmente efficaci per l'analisi di serie storiche comples-

se e non lineari, grazie alla loro capacità di apprendere relazioni temporali implicite e dinamiche non stazionarie.

Questa tesi si propone di valutare l'efficacia predittiva di due modelli (XGboost e LSTM). A partire da una rigorosa analisi sul funzionamento dei mercati finanziari e a seguire, delle proprietà statistiche delle serie storiche dei rendimenti e delle principali teorie sull'efficienza informativa, la tesi approfondisce l'utilizzo del machine learning in ambito finanziario, con particolare attenzione alla definizione dei regimi tramite modelli Hidden Markov, alla preparazione del dataset supervisionato, alla selezione delle feature rilevanti e alla validazione comparativa di diversi algoritmi predittivi. La metodologia culmina nell'implementazione della strategia, valutata infine attraverso indicatori sia statistici che finanziari.

L'approccio adottato sarà sia teorico che empirico: da un lato verranno esaminati i fondamenti dei modelli di machine learning utilizzati, dall'altro sarà implementato un framework di previsione che mira a testare in modo pratico l'affidabilità e la robustezza delle previsioni di regime, confrontandole con strategie passive come il buy and hold. L'analisi si concentrerà in particolare sull'indice Russell 3000, utilizzato come benchmark rappresentativo del mercato azionario statunitense.

Attraverso questo lavoro si intende offrire un contributo alla comprensione di come tecniche avanzate di apprendimento automatico possano essere integrate in modo efficace nei processi decisionali finanziari, non solo per migliorare le previsioni, ma anche per supportare una gestione del rischio più consapevole e dinamica.

Capitolo 1

Mercati finanziari: Struttura, funzionamento ed efficienza

1.1 Microstruttura dei mercati finanziari

La *microstruttura di mercato* studia la forma più essenziale dell'intermediazione finanziaria: lo scambio diretto di un'attività finanziaria, come azioni o obbligazioni. A differenza di altri tipi di intermediazione (come quella bancaria), qui gli asset non vengono trasformati, ma semplicemente trasferiti da un investitore a un altro.

Il servizio fornito dal mercato in questo contesto è chiamato *immediatezza*. Un investitore che desidera negoziare immediatamente inserisce un *ordine a mercato* per eseguire l'operazione al miglior prezzo disponibile (prezzo denaro per vendere, prezzo lettera per acquistare). I prezzi denaro e lettera sono stabiliti dagli operatori, che possono essere dealer professionisti oppure investitori che piazzano ordini limite.

Gli investitori operano in tre mercati distinti: il *mercato informativo*, il *mercato dei titoli* e il *mercato dei servizi di negoziazione*. La microstruttura si focalizza sul terzo,

con particolare attenzione ai costi di transazione come lo *spread denaro-lettera* (bid-ask spread) e le commissioni. Il mercato dei titoli si concentra invece sulla determinazione dei prezzi. In molte teorie di asset pricing si assume l'assenza di costi o frizioni.

Il *mercato informativo* riguarda invece la disponibilità e la qualità delle informazioni, nonché gli incentivi degli analisti. Pur essendo teoricamente separato, questo mercato è strettamente connesso al mercato delle transazioni, poiché la difficoltà e il costo degli scambi dipendono dalla quantità e qualità di informazione in possesso degli operatori.

Tra gli elementi fondamentali del mercato troviamo:

- *Investitori*, ovvero chi domanda o offre immediatezza. Possono essere soggetti privati o istituzionali, come fondi pensione e fondi comuni.
- *Broker e dealer*, che facilitano gli scambi. I broker possono essere:
  - *Upstairs brokers*: interagiscono direttamente con i clienti;
  - *Downstairs brokers*: eseguono gli ordini nei mercati.
- *Strutture di mercato*, ovvero il luogo in cui avvengono le negoziazioni.

I broker sono remunerati tramite commissioni, mentre i dealer traggono profitto dallo spread di negoziazione.

I *dealer* operano come controparti nei mercati finanziari, negoziando per conto proprio e ottenendo profitti dalla differenza tra il prezzo di acquisto e quello di vendita. Essi rappresentano una componente centrale nei mercati organizzati. Ad esempio, gli specialisti del NYSE (New York Stock Exchange) e i *market maker* del Nasdaq (National Association of Securities Dealers Automated Quotation) sono dealer che forniscono liquidità trattando con i broker che rappresentano il pubblico. Anche nei mercati obbligazionari e valutari, i dealer svolgono un ruolo cruciale nel quotare prezzi e garantire la liquidità.

La funzione essenziale di un mercato, ovvero far incontrare acquirenti e venditori (domanda e offerta) è rimasta sostanzialmente immutata nel tempo. Tuttavia, l'infrastruttura attraverso cui avvengono le negoziazioni ha subito importanti trasformazioni tecnologiche. Nel 1792, ad esempio, quando nacque il NYSE, la sede fisica del mercato era semplicemente rappresentata da un albero (il *buttonwood tree*) sotto cui si radunavano i primi 24 broker fondatori. Oggi, le borse come il NYSE e il Nasdaq sono costituite da una rete complessa di comunicazioni elettroniche ad alta velocità. La maggior parte degli scambi viene eseguita con poco o nessun intervento umano: gli investitori possono inviare ordini online, che vengono automaticamente indirizzati a un sistema di esecuzione e, successivamente, a un sistema di compensazione.

La tecnologia ha trasformato il rapporto tra investitori, broker, dealer e le piattaforme di negoziazione, modificando radicalmente la modalità con cui interagiscono.

I mercati tradizionali erano organizzati come associazioni di membri, tipicamente broker e dealer. I mercati più moderni sono invece costituiti da reti informatiche senza membri, gestiti come imprese private con fini di lucro e in grado di operare anche senza intermediari. Di conseguenza, la funzione di *fornire liquidità agli investitori* resterà centrale, anche se la struttura dei mercati in cui si svolgono le transazioni è destinata a cambiare profondamente nel tempo.

1.1.1 Tipologie di mercati

È utile distinguere tra le principali strutture di mercato, anche se nella realtà molti mercati presentano caratteristiche miste. Una distinzione fondamentale è quella tra mercati d'asta e mercati gestiti dai dealer.

Un mercato d'asta *puro* è un sistema in cui gli investitori (solitamente tramite broker) negoziano direttamente tra loro, senza l'intervento di dealer.

Un mercato d'asta *a chiamata* si svolge in momenti specifici in cui uno strumento viene reso disponibile alla negoziazione. Durante queste aste, gli investitori inoltrano ordini indicando prezzo e quantità, e gli scambi avvengono in base a regole predefinite, solitamente al prezzo che massimizza il volume degli scambi. Sebbene molti mercati, come il NYSE e quelli dell'Europa continentale, abbiano inizialmente operato con aste a chiamata, nel tempo si sono evoluti in *mercati ad asta continua* per far fronte all'aumento dei volumi.

In un mercato ad asta continua, gli ordini di acquisto sono generalmente eseguiti al prezzo denaro stabilito da ordini già presenti nel book, mentre gli ordini di vendita si realizzano al prezzo lettera, sempre determinato da ordini preesistenti. Il NYSE, ad esempio, è considerato un mercato ad asta continua con una *crowd*, mentre altri mercati elettronici operano come mercati ad asta continua privi di questa dinamica collettiva.

Un mercato *dealer puro*, invece, è un contesto in cui i dealer (ossia intermediari) pubblicano prezzi di acquisto e vendita a cui gli investitori possono negoziare. In tali mercati, l'investitore non può negoziare direttamente con altri investitori ma deve acquistare al prezzo di vendita stabilito dal dealer e vendere al suo prezzo di acquisto. I mercati obbligazionari e valutari ne sono esempi classici. Il Nasdaq, ad esempio, è nato come dealer market puro, ma nel tempo ha acquisito molte caratteristiche dei mercati d'asta, consentendo l'inserimento di ordini visibili agli altri operatori.

I market dealer sono fisicamente distribuiti: le negoziazioni avvengono tramite telefono o sistemi informatici. Al contrario, i mercati d'asta si sono storicamente concentrati in luoghi fisici come le borse. Tuttavia, grazie ai progressi nella comunicazione, tale distinzione si è andata attenuando. Oggi, la centralizzazione fisica del trading non è più necessaria.

1.1.2 Tipologie di ordini

Esistono principalmente due tipi di ordini utilizzati nei mercati finanziari: l'*ordine di mercato* e l'*ordine limite*.

Un *ordine di mercato* richiede l'esecuzione immediata al miglior prezzo disponibile in quel momento. È lo strumento tipico per chi vuole acquistare o vendere senza ritardi, accettando il prezzo corrente. Un *ordine limite*, invece, consente all'investitore di specificare il prezzo massimo che è disposto a pagare per acquistare, oppure il prezzo minimo al quale intende vendere. L'ordine viene eseguito solo se il mercato raggiunge quel prezzo. In un mercato ad asta continua centralizzato, i migliori ordini limite di acquisto e vendita (quelli in cima al *book*) definiscono il mercato. Le quantità disponibili a quei prezzi rappresentano la cosiddetta *profondità del mercato*. Le transazioni avvengono ogni volta che arrivano nuovi ordini di mercato, i quali sono eseguiti contro i migliori ordini limite presenti.

Nei mercati tradizionali, questo processo può essere mediato da operatori sul posto, come broker e dealer. Nei mercati elettronici moderni, invece, il meccanismo è completamente automatizzato. In un mercato puramente *dealer*, gli ordini limite non sono visibili pubblicamente, ma vengono conservati dal dealer a cui sono stati inviati. Gli ordini di mercato vengono eseguiti ai prezzi di acquisto o vendita del dealer, e non contro gli ordini limite.

Gli ordini si possono anche distinguere per dimensione. Gli ordini piccoli e medi solitamente seguono un flusso standard di esecuzione. Gli ordini di dimensione elevata, invece, richiedono spesso un trattamento speciale.

In questi casi, il broker può *lavorare* l'ordine, cioè suddividerlo e gestirlo durante l'arco della giornata, decidendo come e quando eseguire le varie porzioni. Le operazioni di dimensioni rilevanti possono essere trattate in blocchi, spesso pre-negoziati *upstairs* da

un broker che ha individuato entrambe le controparti della transazione. Lo scambio avviene poi sul mercato secondo regole precise e a prezzi già concordati. Le regole dell'exchange definiscono come questi ordini limite debbano essere eseguiti.

1.1.3 Bid-Ask spread

I mercati continui sono caratterizzati da prezzi di acquisto (denaro) e di vendita (lettera) ai quali le operazioni possono avvenire. Il *bid-ask spread* rappresenta la differenza tra il prezzo che gli acquirenti attivi devono pagare e quello che i venditori attivi ricevono. Esso fornisce un'indicazione sia del costo della transazione che del livello di illiquidità del mercato.

In alternativa, l'illiquidità può essere anche misurata attraverso il tempo necessario per vendere o acquistare in modo ottimale una certa quantità di un'attività finanziaria. I due approcci si avvicinano concettualmente, poiché lo spread può essere interpretato come il compenso richiesto dal dealer per assumersi una posizione indesiderata e smaltirla in modo efficiente.

Il nostro interesse si concentra sul *bid-ask spread*, che può variare sensibilmente tra diversi mercati. In quelli meno liquidi, come ad esempio il mercato immobiliare, lo spread tende ad essere più ampio. Ad esempio, una casa può essere offerta a $500.000 mentre la miglior offerta disponibile è pari a $450.000. Al contrario, nei mercati finanziari ad alta liquidità, come quello azionario, lo spread per azione può essere inferiore a 10 centesimi. Una questione fondamentale nella teoria della microstruttura è proprio la determinazione dello spread denaro-lettera e la sua variazione tra differenti strumenti finanziari.

Diversi fattori contribuiscono allo spread:

- *Costi operativi*: i dealer che garantiscono liquidità sostengono spese per la gestio-

ne degli ordini, il clearing e l'infrastruttura necessaria alla negoziazione. Questi costi sono inferiori nei mercati in cui gli ordini limite fissano i prezzi.

- *Prezzi non competitivi*: in assenza di concorrenza, i market maker possono stabilire spread elevati attraverso accordi o regole come i tick minimi.
- *Rischio di inventario*: chi compra al denaro o vende alla lettera si espone al rischio di detenere temporaneamente il titolo, assumendosi una possibile perdita in caso di variazione avversa del prezzo.
- *Effetto opzione*: pubblicare un prezzo equivale a concedere al mercato un'opzione. Se sopraggiungono nuove informazioni, il prezzo potrebbe non rifletterle tempestivamente.
- *Informazione asimmetrica*: chi fissa una quotazione può subire perdite se altri trader possiedono informazioni migliori.

Questi fattori non sono mutuamente esclusivi: spesso coesistono. Tra le fonti di incertezza che alimentano lo spread si identificano:

- *Effetto inventario*: dovuto alla possibilità che emergano informazioni pubbliche *dopo* la transazione.
- *Effetto opzione*: legato a informazioni pubbliche *prima* dell'esecuzione e all'impossibilità di aggiornare tempestivamente la quotazione.
- *Selezione avversa*: causata da informazioni *private* disponibili solo ad alcuni operatori prima dello scambio.

Le origini dello spread denaro-lettera possono essere analizzate sia dal punto di vista dei servizi forniti sia da quello delle risorse impiegate. Una prima interpretazione considera lo spread come una misura del costo dei servizi offerti dai fornitori di liquidità.

Tali operatori, infatti, elaborano ordini, assumono un rischio d'inventario e impiegano risorse reali per svolgere il proprio ruolo.

Una seconda interpretazione, di tipo informativo, sostiene invece che lo spread rappresenti una forma di compensazione per le perdite subite nei confronti di operatori più informati. In questo contesto, si presume che gli investitori informati ottengano un guadagno dagli operatori meno informati. Tuttavia, ciò non implica che vengano forniti servizi o impiegate risorse tangibili.

1.2 Teorie dell'efficienza del mercato

Dopo aver delineato le principali strutture del mercato e formazioni di prezzo, è naturale domandarsi quale sia il contenuto informativo realmente disponibile nei prezzi degli asset e se esistano margini per la previsione. Questo interrogativo conduce al concetto di *efficienza del mercato*, che rappresenta un principio cardine della teoria finanziaria e costituisce il collegamento diretto tra la modellazione stocastica dei rendimenti e la loro prevedibilità. L'origine della teoria del mercato efficiente può essere fatta risalire alle intuizioni di Louis Bachelier, che nel 1900, nella sua tesi di dottorato *"Théorie de la Spéculation"* [1] dove assumeva che i prezzi si muovessero in modo stocastico, anticipando di oltre mezzo secolo il concetto di random walk. Formalmente, Bachelier descrisse i prezzi come un *processo di Wiener*:

$$P_t = P_0 + \int_0^t \sigma dW_t$$

dove: P_t rappresenta il prezzo dell'asset al tempo t, σ è il parametro di volatilità, W_t è un processo di Wiener standard. Questa equazione implica che i movimenti di prezzo siano *indipendenti* e *normalmente distribuiti*, il che rende impossibile prevedere il mercato attraverso l'analisi dei prezzi passati.

1.2.1 La formalizzazione matematica: Samuelson e il processo Martingala

Partendo dalla critica al Random Walk, Paul Samuelson, premio Nobel per l'economia nel 1970. Nel suo articolo *"Proof That Properly Anticipated Prices Fluctuate Randomly"* (1965) [21] analizza il mercato dei futures come il luogo in cui vengono prezzate le anticipazioni sui prezzi futuri. Denotando con P_{t+T} la stima al tempo t del prezzo *spot* al tempo $t+T$ e con $X_{t,T}$ il prezzo del contratto *futures* sullo stesso asset valutato al tempo t con una scadenza T, egli si concentra sulla relazione tra la sequenza P_{t+T} e la sequenza $X_{t,T}$. In ogni istante, il prezzo futures è osservato e coincide con il prezzo spot di quel momento; inoltre, ad ogni istante gli investitori conoscono la distribuzione di probabilità che governa la sequenza P_{t+T} e la utilizzano per valutare $X_{t,T}$:

$$X_t = E[P_{t+T}|F_t]$$

dove $F_t = \{P_0, P_1, \ldots, P_t\}$ rappresenta le informazioni disponibili sui prezzi passati. Basandosi su queste assunzioni, Samuelson applica la *legge dell'eccezione iterativa* per concludere che la variabile casuale $X_{t,T}$ segue un processo Martingala, secondo il quale la miglior stima del prezzo di domani è il prezzo di oggi:

$$X_{t,T} = E[X_{t+1,T-1}|F_t]$$

dove $F_t = \{X_0, X_1, \ldots, X_t\}$. Questa rappresentazione riflette l'idea che il prezzo futuro non possa essere prevedibile. Per questo motivo, la Martingala è comunemente definita un *processo stocastico equo*: scommettere su un processo di Martingala non genera alcun guadagno atteso.

1.2.2 L'ipotesi di mercato efficiente di Eugene Fama

L'economista Eugene Fama fu il primo a strutturare l'Efficient Market Hypothesis (EMH) in una forma organica e testabile. Nel suo influente articolo *"Efficient Capital Markets: A Review of Theory and Empirical Work"* (1970) [8] in cui definì tre diversi livelli di efficienza di mercato:

- Efficienza debole : i prezzi degli asset riflettono completamente tutte le informazioni contenute nei prezzi passati. In questo scenario, nessun investitore può generare extra-profitti sfruttando modelli basati sull'analisi tecnica.
- Efficienza semi-Forte : i prezzi degli asset riflettono non solo le informazioni storiche, ma anche tutte le informazioni pubblicamente disponibili, come bilanci aziendali e annunci economici. In questo caso, l'analisi fondamentale non può generare rendimenti superiori alla media.
- Efficienza forte : i prezzi riflettono tutte le informazioni disponibili, incluse quelle privilegiate (*insider information*). Se il mercato fosse perfettamente efficiente in questa forma, nessun investitore, nemmeno con accesso a informazioni riservate, potrebbe ottenere rendimenti anomali.

L'EMH ha avuto un impatto significativo sulle teorie finanziarie, influenzando la gestione degli investimenti e le strategie di trading. Se i mercati sono realmente efficienti, allora:

- Non è possibile battere il mercato sistematicamente se non assumendosi rischi aggiuntivi.
- L'analisi tecnica e l'analisi fondamentale risultano inefficaci nel generare extra-profitti nel lungo periodo.

- Le strategie di investimento passive, come l'*index investing* o il *buy and hold*, risultano più vantaggiose rispetto alla gestione attiva.

Secondo la visione di Fama un mercato è efficiente quando i prezzi riflettono fedelmente i valori fondamentali degli asset. Se gli investitori agiscono con l'obiettivo di massimizzare i profitti e operano in un contesto di informazione simmetrica, allora i prezzi dovrebbero convergere verso i valori teoricamente corretti. Di conseguenza, ogni variazione del prezzo di un asset da un periodo al successivo sarà interamente dovuta all'assimilazione di nuove informazioni, che emergono tra due istanti temporali consecutivi. La casualità che caratterizza i movimenti di prezzo in un mercato efficiente può essere modellata attraverso un processo Random Walk. Inoltre, all'interno di un mercato razionale, i trader sono in grado di individuare immediatamente opportunità di *arbitraggio* non appena emergono nuove informazioni. Questo implica che quanto più efficiente è un mercato, tanto più casuali risultano le variazioni dei prezzi.

Per descrivere questa dinamica, definiamo le basi statistiche di un processo Random Walk. Supponiamo di considerare una sequenza di variabili casuali i.i.d. $Y_1, Y_2, \ldots$ tali che le variabili casuali Y_i assumono valori ± 1 con probabilità uguale. Supponendo che $X_0 = 0$, si ha:

$$X_t = \sum_{i=1}^{t} Y_i \quad \forall t \geq 1.$$

Questo definisce una distribuzione di probabilità sulle sequenze $X_0, X_1, \ldots$, caratterizzando così un processo stocastico *discreto nel tempo*, noto come Random Walk, che assume la seguente forma:

$$X_t = X_{t-1} + \alpha_t$$

dove X_0 è un valore iniziale arbitrario e $\{\alpha_t\}$ è un *white noise*, ovvero una serie temporale con media zero e senza autocorrelazione. Poiché la distribuzione di $\{\alpha_t\}$ è simmetrica attorno allo zero, la variabile aleatoria condizionata $X_t|X_{t-1}$ ha una probabilità del 50% di aumentare o diminuire, implicando che X_t segua una traiettoria casuale.

Dal teorema del limite centrale, sappiamo che per un orizzonte temporale sufficientemente ampio n, la distribuzione della variabile $\frac{1}{\sqrt{n}}X_n$

converge alla distribuzione normale con media 0 e varianza 1. Possiamo quindi enunciare le seguenti proprietà fondamentali del Processo Random Walk Semplice:

1. Media nulla (Zero-mean): $E[X_t] = 0 \quad \forall t \geq 1$, secondo il teorema del limite centrale.

2. Incrementi indipendenti: Per ogni sequenza $0 = t_0 \leq t_1 \leq \cdots \leq t_n$, le variabili casuali $(X_{t_{i+1}} - X_{t_i})$ per $0 \leq i \leq (n-1)$ sono mutuamente indipendenti.

3. Stazionarietà: Per ogni $t \geq 1$ e $k \geq 0$, la distribuzione della differenza $(X_{t+k} - X_k)$ rimane identica alla distribuzione di X_t.

1.2.3 Differenze e alternative all'EMH

Dopo aver descritto i due modelli, possiamo confrontarne il potere esplicativo. In primo luogo, possiamo osservare che il processo Martingala di Samuelson non assume l'indipendenza delle variazioni di prezzo, a differenza del Random Walk di Fama. Infatti, il Random Walk può essere visto come un caso particolare del modello di Martingala. Proprio per questa ragione, il Random Walk è caratterizzato da ipotesi più restrittive, rendendolo meno rappresentativo della realtà del mercato.

La seconda differenza rafforza questa osservazione: l'ipotesi di stazionarietà del Random Walk semplice implica che la volatilità delle variazioni di prezzo successive sia co-

stante nel tempo. Al contrario, il modello di Martingala non richiede questa assunzione, che spesso viene smentita da shock di mercato.

Il terzo aspetto da considerare riguarda la competizione di mercato. Samuelson assume che gli investitori in competizione abbiano la stessa conoscenza della distribuzione di probabilità del prezzo futuro. Fama, invece, non impone un'ipotesi così restrittiva e si limita ad affermare che gli operatori di mercato siano profit-maximizers, senza però specificare che abbiano una conoscenza esplicita delle distribuzioni probabilistiche. Infine, mentre Fama sostiene che il prezzo di oggi rappresenta la miglior stima del valore intrinseco degli asset, Samuelson adotta una visione meno rigida, limitandosi ad affermare che il prezzo attuale è la miglior previsione del prezzo futuro: Nonostante le differenze nei loro modelli, sia Fama che Samuelson condividono alcuni assunti di base che definiscono il concetto di Efficienza di Mercato:

- Concorrenza perfetta, garantita dalla presenza di un ampio numero di partecipanti al mercato;
- Comportamento razionale degli investitori, con l'obiettivo di massimizzare il profitto;
- Informazione comune, condivisa tra tutti gli operatori man mano che diventa disponibile.

Nonostante l'iniziale accettazione dell'EMH, diversi economisti e studiosi hanno individuato *anomalie di mercato* che mettono in discussione l'idea di efficienza assoluta. Le anomalie sono pattern sistematici nei prezzi degli asset che sembrano contraddire l'ipotesi che i mercati incorporino tutte le informazioni disponibili in modo istantaneo ed efficiente.

Uno dei primi studiosi a contestare l'EMH fu Benoît Mandelbrot (1963), il quale dimostrò che i rendimenti finanziari non seguono una distribuzione normale, come suggerito

dalla teoria classica, ma presentano *fat tails* e *clusterizzazione della volatilità*. Questi fenomeni indicano che i mercati finanziari sono caratterizzati da brusche fluttuazioni, con rendimenti estremamente elevati o bassi che si verificano con una frequenza maggiore rispetto a quanto previsto dalla distribuzione normale.

Un'altra critica significativa venne da Robert Shiller (1981), che evidenziò come i prezzi delle azioni mostrassero una eccessiva volatilità, non giustificata dai fondamentali economici. Attraverso un'analisi empirica, Shiller dimostrò che le fluttuazioni dei prezzi azionari sono molto più ampie rispetto alle variazioni nei dividendi futuri attesi. Questo risultato suggerisce che i mercati non sono completamente razionali, ma soggetti a periodi di speculazione e irrazionalità collettiva. Nel corso degli anni, numerosi studi hanno rivelato ulteriori anomalie nei mercati finanziari:

- Momentum Effect (*Jegadeesh e Titman, 1993* [12]): i titoli con buone performance passate tendono a mantenere il loro trend positivo, violando l'ipotesi di random walk.

- Overreaction e Underreaction (*De Bondt e Thaler, 1985* [4]): gli investitori tendono a reagire in modo eccessivo o insufficiente alle nuove informazioni, causando un'errata valutazione degli asset.

- Small Firm Effect (*Banz, 1981* [2]): le aziende a bassa capitalizzazione tendono a ottenere rendimenti superiori rispetto alle aziende di grandi dimensioni, contraddicendo l'idea che il rischio di mercato sia l'unico fattore determinante nei prezzi.

Queste anomalie suggeriscono che, in alcuni contesti, i prezzi degli asset possono essere prevedibili in una certa misura, aprendo la strada a strategie di investimento basate su fattori sistematici.

Negli anni ‘90, la *finanza comportamentale* ha offerto un’alternativa all’EMH, sostenendo che gli investitori non agiscono sempre in modo razionale e che le loro decisioni sono influenzate da bias cognitivi e psicologici. L’idea principale è che gli agenti di mercato siano soggetti a errori sistematici nel processo decisionale, portando a inefficienze persistenti nei prezzi degli asset. Uno dei principali studiosi di finanza comportamentale è Richard Thaler, il quale ha dimostrato che gli investitori soffrono di *overconfidence*, ovvero un’eccessiva sicurezza nelle proprie previsioni, e di *loss aversion*, una tendenza a dare maggiore peso alle perdite rispetto ai guadagni equivalenti. Questi fattori contribuiscono alla formazione di bolle speculative e al successivo scoppio delle stesse. Parallelamente, Daniel Kahneman e Amos Tversky hanno sviluppato la *Prospect Theory* (1979) [14], che descrive come gli individui valutano le decisioni sotto rischio. Contrariamente alla teoria dell’utilità attesa, la Prospect Theory sostiene che gli investitori tendono a sovrastimare le probabilità di eventi estremi e a prendere decisioni in base a punti di riferimento piuttosto che in termini assoluti di guadagno e perdita.

Un’altra teoria alternativa è l’*Adaptive Market Hypothesis (AMH)*, Andrew Lo (2004) [15]. Secondo Lo, l’efficienza del mercato non è una condizione statica, ma varia nel tempo in risposta alla competizione tra gli investitori e alla loro capacità di apprendere ed adattarsi. In periodi di maggiore stabilità e liquidità, i mercati possono essere più efficienti, mentre in fasi di turbolenza e crisi economica possono emergere inefficienze significative. Le ricerche nel campo della finanza comportamentale e le evidenze sulle anomalie di mercato dimostrano che l’EMH, pur essendo un punto di riferimento fondamentale nella teoria finanziaria, non è in grado di spiegare completamente la realtà dei mercati. Le moderne strategie di investimento e di gestione del rischio tengono oggi conto sia degli aspetti razionali dell’EMH sia delle intuizioni fornite dalla finanza comportamentale e dalla teoria dell’adattabilità dei mercati.

1.2.4 Oltre le teorie classiche: il contributo del machine learning

Le teorie classiche dell'efficienza dei mercati hanno avuto un ruolo centrale nello sviluppo della finanza moderna, offrendo un quadro elegante e coerente per spiegare il comportamento dei prezzi. Tuttavia, l'analisi delle anomalie empiriche e delle dinamiche osservate nei mercati reali evidenzia come questi modelli, per quanto utili, non riescano sempre a cogliere la complessità effettiva del sistema finanziario.

Proprio da questi limiti nasce l'interesse verso approcci alternativi, più flessibili e capaci di adattarsi a una realtà in continuo cambiamento. In questo contesto, il *machine learning* rappresenta una risorsa preziosa: anziché partire da ipotesi rigide, questi modelli apprendono direttamente dai dati, identificando schemi ricorrenti e relazioni non lineari che i metodi tradizionali spesso non riescono a catturare.

Tecniche come gli *Hidden Markov Models*, gli algoritmi *ensemble* (come *Random Forest* e *XGBoost*) o le *reti neurali ricorrenti* (RNN) permettono di affrontare in modo più realistico il problema della previsione dei regimi di mercato. In particolare, si rivelano efficaci quando si lavora con serie storiche caratterizzate da transizioni di stato non osservabili direttamente e da comportamenti non stazionari.

Nel prosieguo della tesi, questi strumenti verranno approfonditi sia dal punto di vista teorico sia applicativo, per valutare concretamente il loro contributo nella costruzione di strategie predittive più dinamiche e aderenti alla realtà dei mercati finanziari.

Capitolo 2

Teoria delle serie temporali finanziarie

2.1 Prezzi, rendimenti e capitalizzazione

Giunti a questo punto risulta utile chiarire ed esporre i concetti alla base delle variabili finanziarie. Tra tutte le serie temporali finanziarie rivestono un ruolo centrale, riferendosi comunemente all'andamento dei prezzi dei titoli scambiati sul mercato. Tuttavia, anziché analizzare i prezzi in forma assoluta, la maggior parte degli studi si concentra sui *rendimenti*, ritenuti più informativi dal punto di vista statistico ed economico. Praticamente ogni aspetto dell'economia finanziaria coinvolge i ritorni, e ci sono almeno due motivi principali per focalizzarsi su di essi piuttosto che sui prezzi. Anzitutto, per l'investitore medio, i mercati finanziari possono essere considerati prossimi alla perfetta concorrenza, tale per cui la dimensione dell'investimento non influisce sulle variazioni di prezzo. Dato che la "tecnologia" di investimento ha rendimenti di scala costanti, il rendimento rappresenta un riassunto completo e indipendente dalla scala delle opportunità di investimento. In secondo luogo, per motivazioni teoriche ed empiriche che diventeranno chiare successivamente, i rendimenti presentano proprietà statistiche più convenienti rispetto ai prezzi, come la stazionarietà e l'ergodicità.

Il comportamento delle serie temporali finanziarie è inevitabilmente influenzato dal contesto in cui si manifestano: i *mercati finanziari*, che sono regolati da fattori sistematici e non sistematici. I primi introducono rumore nella serie temporale sotto forma di volatilità, che varia nel tempo; i secondi invece determinano trend e schemi ciclici, che rimangono più costanti. Questi fattori, noti rispettivamente come trend, stagionalità e rumore, sono chiamati *componenti principali delle serie temporali finanziarie* .

2.1.1 Definizioni e convenzioni

Indichiamo con P_t il prezzo di un'attività alla data t e supponiamo, per ora, che questa attività non paghi dividendi. Il *rendimento netto semplice*, R_t, sull'attività tra le date $t-1$ e t è definito come

$$R_t = \frac{P_t}{P_{t-1}} - 1$$

Il *rendimento lordo semplice* sull'attività è semplicemente uno più il rendimento netto, cioè $1+R_t$.

Da questa definizione è evidente che il rendimento lordo dell'attività negli ultimi k periodi, dalla data $t-k$ alla data t, indicato con $1+R_t(k)$, è semplicemente uguale al prodotto dei k rendimenti a singolo periodo da $t-k+1$ a t, cioè

$$1+R_t(k) \equiv (1+R_t)\cdot(1+R_{t-1})\cdots(1+R_{t-k+1})$$

$$= \frac{P_t}{P_{t-1}} \cdot \frac{P_{t-1}}{P_{t-2}} \cdot \frac{P_{t-2}}{P_{t-3}} \cdots \frac{P_{t-k+1}}{P_{t-k}} = \frac{P_t}{P_{t-k}}$$

e il suo rendimento netto sugli ultimi k periodi, indicato con $R_t(k)$, è semplicemente uguale al rendimento lordo su k periodi meno uno. Questi rendimenti su più periodi sono detti *rendimenti composti*.

Sebbene i rendimenti siano privi di scala, è importante sottolineare che essi *non* sono privi di unità, ma sono sempre definiti rispetto a un certo intervallo temporale, ad esempio un "periodo." In effetti, R_t è più propriamente chiamato *tasso di rendimento*, termine più ingombrante ma più accurato per riferirsi a R_t come a un tasso o, nel gergo economico, una variabile di *flusso*. Pertanto, un rendimento del 20% non rappresenta una descrizione completa dell'opportunità di investimento senza una specificazione dell'orizzonte temporale. Tuttavia, tra i professionisti e nella letteratura finanziaria, un orizzonte di un anno è di solito assunto implicitamente; quindi, salvo indicazione contraria, un rendimento del 20% è generalmente interpretato come un rendimento *annuale* del 20%. Inoltre, i rendimenti su più periodi sono spesso *annualizzati* per rendere comparabili investimenti con orizzonti temporali diversi, come segue:

$$\text{Annualizzato}[R_t(k)] = \left[\prod_{j=0}^{k-1}(1+R_{t-j})\right]^{1/k} - 1$$

Poiché i rendimenti a singolo periodo sono generalmente piccoli in magnitudine, è spesso usata la seguente approssimazione, basata su uno sviluppo di Taylor al primo ordine, per annualizzare i rendimenti su più periodi:

$$\text{Annualizzato}[R_t(k)] \approx \frac{1}{k}\sum_{j=0}^{k-1} R_{t-j}$$

Se tale approssimazione sia adeguata dipende dalla particolare applicazione: può bastare per un confronto rapido e grossolano della performance tra molti asset, ma per calcoli più fini — in cui la volatilità dei rendimenti gioca un ruolo importante — è più facile calcolare una media aritmetica anziché una media geometrica.

2.1.2 Capitalizzazione continua

La difficoltà nel manipolare medie geometriche, motiva un altro approccio ai rendimenti composti, uno che non è approssimativo e ha anche importanti implicazioni nella modellazione dei rendimenti degli asset: si tratta della nozione di *capitalizzazione continua*. Il *rendimento composto in continuo*, o *log rendimento*, r_t, di un'attività è definito come il logaritmo naturale del suo rendimento lordo $(1+R_t)$:

$$r_t \equiv \log(1+R_t) = \log\left(\frac{P_t}{P_{t-1}}\right) = p_t - p_{t-1}$$

dove $p_t \equiv \log P_t$. Quando desideriamo sottolineare la distinzione tra R_t e r_t, ci riferiremo a R_t come *rendimento semplice*. La nostra notazione qui si discosta leggermente dalla convenzione secondo cui le lettere minuscole rappresentano i logaritmi delle lettere maiuscole, poiché qui abbiamo $r_t = \log(1+R_t)$ invece di $\log(R_t)$; facciamo ciò per mantenere la coerenza con le convenzioni standard.

I vantaggi della capitalizzazione continua diventano evidenti quando consideriamo rendimenti su più periodi, infatti:

$$r_t(k) = \log(1+R_t(k)) = \log((1+R_t)\cdot(1+R_{t-1})\cdots(1+R_{t-k+1}))$$

$$= \log(1+R_t) + \log(1+R_{t-1}) + \cdots + \log(1+R_{t-k+1})$$

$$= r_t + r_{t-1} + \cdots + r_{t-k+1}$$

e quindi il rendimento composto in continuo su più periodi è semplicemente la somma dei rendimenti composti in continuo a singolo periodo. La capitalizzazione, che è un'operazione moltiplicativa, viene convertita in un'operazione additiva tramite l'uso dei

logaritmi. Tuttavia, la semplificazione non risiede solo nella riduzione della moltiplicazione ad addizione (dato che, con calcolatrici moderne e computer, questo è banale), ma soprattutto nella modellazione del comportamento statistico dei rendimenti nel tempo: è molto più facile derivare le proprietà di serie storiche di processi additivi che di processi moltiplicativi. I rendimenti composti in continuo hanno però uno svantaggio. Il rendimento semplice di un portafoglio di attività è una media ponderata dei rendimenti semplici delle attività stesse, dove il peso su ciascuna attività è la quota del valore del portafoglio investito in quell'attività. Se il portafoglio p assegna il peso w_{ip} all'attività i, allora il rendimento del portafoglio al tempo t, R_{pt}, è legato ai rendimenti delle singole attività, R_{it}, per $i = 1 \dots N$, da

$$R_{pt} = \sum_{i=1}^{N} w_{ip} R_{it}.$$

Sfortunatamente, i rendimenti composti in continuo non condividono questa proprietà conveniente. Poiché il logaritmo di una somma non è uguale alla somma dei logaritmi, $r_{pt} \neq \sum_{i=1}^{N} w_{ip} r_{it}$

Nelle applicazioni empiriche questo problema è di solito trascurabile. Quando i rendimenti sono misurati su intervalli di tempo brevi, e quindi sono vicini a zero, il rendimento composto in continuo di un portafoglio è vicino alla media ponderata dei rendimenti composti in continuo dei singoli titoli:

$$r_{pt} \approx \sum_{i=1}^{N} w_{ip} r_{it}$$ [1]

[1] Nel limite in cui il tempo è continuo, il **Lemma di Itô**, può essere utilizzato per mettere in relazione i rendimenti semplici con quelli composti in continuo

2.1.3 Erogazione dei dividendi

Per gli asset che effettuano pagamenti periodici di dividendi, dobbiamo modificare le nostre definizioni di rendimento e capitalizzazione. Indichiamo con D_t il dividendo dell'attività alla data t e assumiamo, che tale dividendo venga pagato immediatamente prima che il prezzo alla data t, P_t, venga registrato; pertanto P_t è considerato il prezzo *ex-dividendo* alla data t. In alternativa, si può descrivere P_t come il prezzo dell'attività alla fine del periodo. Allora il rendimento netto semplice alla data t può essere definito come:

$$R_t = \frac{P_t + D_t}{P_{t-1}} - 1$$

I rendimenti su più periodi e quelli composti in continuo possono essere calcolati nello stesso modo del caso senza dividendi. Si noti che il rendimento composto in continuo su un'attività che paga dividendi,

$$r_t = \log(P_t + D_t) - \log(P_{t-1})$$

è una funzione non lineare dei logaritmi dei prezzi e dei dividendi. Tuttavia, quando il rapporto tra prezzi e dividendi non varia troppo, questa funzione può essere approssimata da una funzione lineare dei logaritmi di prezzi e dividendi.

2.1.4 Excess returns

È spesso conveniente lavorare con il *rendimento in eccesso* di un'attività, definito come la differenza tra il rendimento dell'attività e il rendimento di un'attività di riferimento. L'attività di riferimento è spesso assunta come priva di rischio e, nella pratica, è solita-

mente rappresentata dal rendimento di un buono del Tesoro a breve termine. Lavorando con i rendimenti semplici, il rendimento semplice in eccesso sull'attività i è:

$$Z_{it} = R_{it} - R_{0t}$$

dove R_{0t} è il rendimento di riferimento. In alternativa, si può definire il *log rendimento in eccesso* come:

$$z_{it} = r_{it} - r_{0t}$$

Il rendimento in eccesso può anche essere interpretato come il payoff di un *portafoglio di arbitraggio* che prende una posizione long sull'attività i e una posizione short sull'attività di riferimento, senza investimento netto alla data iniziale. Poiché l'investimento netto iniziale è nullo, il rendimento del portafoglio di arbitraggio è indefinito, ma il suo payoff in termini monetari è proporzionale al rendimento in eccesso come definito sopra.

2.1.5 Rendimento di portafoglio

Il rendimento netto semplice di un portafoglio composto da N attività è una media ponderata dei rendimenti netti semplici delle attività coinvolte, dove il peso di ciascuna attività rappresenta la percentuale del valore del portafoglio investito in essa. Sia p un portafoglio che assegna peso w_i all'attività i. Allora, il rendimento semplice del portafoglio p al tempo t è:

$$R_{p,t} = \sum_{i=1}^{N} w_i R_{it}$$

dove R_{it} è il rendimento semplice dell'attività i.

I rendimenti composti in continuo di un portafoglio, tuttavia, non godono di questa proprietà conveniente. Se i rendimenti semplici R_{it} sono tutti di piccola entità, allora abbiamo:

$$r_{p,t} \approx \sum_{i=1}^{N} w_i r_{it}$$

dove $r_{p,t}$ è il rendimento composto in continuo del portafoglio al tempo t. Questa approssimazione è spesso utilizzata per lo studio dei rendimenti di portafoglio.

2.1.6 Distribuzioni statistiche delle variabili finanziarie

Dopo aver definito attentamente i rendimenti delle attività, possiamo ora iniziare a studiarne il comportamento tra diversi asset e nel tempo. Forse la caratteristica più importante dei rendimenti è la loro *casualità*. Il rendimento di un'azione nel prossimo mese è sconosciuto oggi, e ciò che distingue l'economia finanziaria da altre scienze sociali è proprio la modellizzazione esplicita delle fonti e della natura di questa incertezza. Sebbene altri rami dell'economia e della sociologia possano disporre di modelli di fenomeni stocastici, in nessuno di essi l'incertezza gioca un ruolo centrale quanto nella determinazione dei prezzi degli asset finanziari, senza incertezza, gran parte della letteratura in economia finanziaria, sia teorica sia empirica, sarebbe superflua. Pertanto, dobbiamo chiarire fin dall'inizio i tipi di incertezza che i rendimenti degli asset possono presentare.

Distribuzione Congiunta

Consideriamo una collezione di N attività alla data t, ciascuna con rendimento R_{it} alla data t, dove $t = 1, \ldots, T$. Forse il modello più generale della collezione dei rendimenti $\{R_{it}\}$ è la sua *funzione di distribuzione congiunta*:

$$G(R_{11},\ldots,R_{N1};R_{12},\ldots,R_{N2};\ldots;R_{1T},\ldots,R_{NT};\mathbf{x}\mid\theta),$$

dove $\mathbf{x}$ è un vettore di *variabili di stato*, che riassumono l'ambiente economico in cui i rendimenti sono determinati, e θ è un vettore di parametri fissi che determinano univocamente G. Per comodità notazionale, sopprimeremo la dipendenza di G da θ a meno che non sia necessario. La legge di probabilità G governa il comportamento stocastico dei rendimenti e delle variabili $\mathbf{x}$, e rappresenta il totale dell'informazione conoscibile su di essi. Possiamo considerare l'econometria finanziaria come l'inferenza statistica su θ, dato G e le realizzazioni di $\{R_{it}\}$.

Distribuzione condizionata

Consideriamo la distribuzione congiunta $F(\{R_{i1},\ldots,R_{iT}\})$ per un dato asset i, e osserviamo che possiamo sempre riscrivere F come il seguente prodotto:

$$F(R_{i1},\ldots,R_{iT}) = F_{i1}(R_{i1})\cdot F_{i2}(R_{i2}\mid R_{i1})\cdot F_{i3}(R_{i3}\mid R_{i2},R_{i1})\cdots F_{iT}(R_{iT}\mid R_{i(T-1)},\ldots,R_{i1})$$

Dalla (1.4.11), le dipendenze temporali implicite in $\{R_{it}\}$ risultano evidenti. Le questioni di *predicibilità* dei rendimenti degli asset coinvolgono aspetti delle loro distribuzioni *condizionate* e, in particolare, come tali distribuzioni evolvono nel tempo.

Imponendo ulteriori restrizioni sulle distribuzioni condizionate $F_{it}(\cdot)$, possiamo stimare i parametri θ impliciti nella (1.4.11) ed esaminare esplicitamente la predicibilità dei rendimenti degli asset. Ad esempio, una versione dell'ipotesi del *random walk* si ottiene imponendo che la distribuzione condizionata del rendimento R_{it} sia uguale alla sua distribuzione marginale, ovvero:

$$F_{it}(R_{it}\mid\cdot) = F_{it}(R_{it})$$

Se ciò vale, i rendimenti sono indipendenti nel tempo e quindi non prevedibili sulla base dei rendimenti passati. Versioni più deboli del random walk si ottengono imponendo restrizioni meno forti su $F_{it}(R_{it} \mid \cdot)$.

Distribuzione non condizionata

Nei casi in cui la distribuzione condizionata del rendimento di un'attività differisca dalla sua distribuzione marginale o incondizionata, è chiaramente la distribuzione condizionata quella rilevante per le questioni di predicibilità. Tuttavia, le proprietà della distribuzione incondizionata dei rendimenti possono comunque essere di qualche interesse, soprattutto nei casi in cui si prevede che la predicibilità sia minima.

Uno dei modelli più comuni per i rendimenti degli asset è il modello normale IID (indipendenti e identicamente distribuiti) nel tempo, in cui si assume che i rendimenti siano indipendenti nel tempo , distribuiti in modo identico nel tempo e normalmente distribuiti.Sebbene il modello normale temporaneamente IID possa essere trattabile, presenta almeno due svantaggi importanti. Primo, la maggior parte degli asset finanziari presenta *responsabilità limitata*, per cui la perdita massima che un investitore può subire è pari al totale investito, e non oltre. Questo implica che il rendimento netto minimo raggiungibile è -1, ovvero -100%. Ma poiché il supporto della distribuzione normale copre l'intera retta reale, questo limite inferiore di -1 è chiaramente violato dalla normalità. Si può obiettare che scegliendo opportunamente media e varianza si possa rendere la probabilità di realizzazioni inferiori a -1 arbitrariamente piccola; tuttavia, essa non sarà mai zero, come richiede la responsabilità limitata. Secondo, se si assume che i rendimenti a singolo periodo siano normali, allora i rendimenti su più periodi non possono essere normali, poiché sono prodotti di rendimenti a singolo periodo. Ora, la *somma* di rendimenti normali a singolo periodo è effettivamente normale, ma la somma di rendimenti semplici a singolo periodo non ha un'interpretazione economica significativa.

2.1.7 Distribuzioni dei rendimenti

Il modello più generale per i rendimenti logaritmici $\{r_{it}; i = 1,\ldots,N; t = 1,\ldots,T\}$ è la sua funzione di distribuzione congiunta:

$$F_r(r_{11},\ldots,r_{N1}; r_{12},\ldots,r_{N2}; \ldots; r_{1T},\ldots,r_{NT}; \mathbf{Y}; \theta),$$

dove $\mathbf{Y} \in \mathbb{R}^k$ è un vettore di stato, e θ è un vettore di parametri. Il comportamento stocastico dei rendimenti r_{it} e di $\mathbf{Y}$ è governato da F_r. Spesso $\mathbf{Y}$ è trattato come dato, e l'interesse è nella distribuzione condizionata di $\{r_{it}\}$ dato $\mathbf{Y}$. L'analisi empirica dei rendimenti mira a stimare θ e fare inferenze statistiche sulla base di rendimenti log passati.

Questo è troppo generale per applicazioni pratiche, ma utile come quadro concettuale. Alcune teorie, come il CAPM di Sharpe (1964) [23], si focalizzano sulla distribuzione congiunta a un dato istante t. Altre pongono attenzione alla struttura dinamica dei rendimenti di un singolo asset i.

Nel nostro caso, l'interesse è sulla distribuzione congiunta di $\{r_{it}\}_{t=1}^{T}$, che può essere partizionata come:

$$F(r_{i1},\ldots,r_{iT};\theta) = F(r_{i1})F(r_{i2} \mid r_{i1})\cdots F(r_{iT} \mid r_{i1},\ldots,r_{i(T-1)}) = F(r_{i1})\prod_{t=2}^{T} F(r_{it} \mid r_{i,t-1},\ldots,r_{i1}),$$

dove θ è omesso per semplicità.

Questa forma mette in evidenza le dipendenze temporali. La questione principale è come la distribuzione condizionata $F(r_{it} \mid r_{i,t-1},\ldots)$ si evolva nel tempo.

In finanza, diverse specificazioni implicano teorie differenti. Ad esempio, l'ipotesi del random walk assume che $F(r_{it} \mid r_{i,t-1},\ldots,r_{i1}) = F(r_{it})$, ovvero che i rendimenti siano

indipendenti e non prevedibili nel tempo. È consuetudine trattare i rendimenti degli asset come variabili casuali continue, specialmente nel caso dei rendimenti di indici o azioni calcolati a bassa frequenza, utilizzando quindi le loro funzioni di densità di probabilità. In questo caso possiamo scrivere la partizione dell'Equazione come:

$$f(r_{i1},\ldots,r_{iT};\theta) = f(r_{i1};\theta)\prod_{t=2}^{T} f(r_{it} \mid r_{i,t-1},\ldots,r_{i1},\theta)$$

Per i rendimenti degli asset ad alta frequenza, la discrezione diventa un problema. Ad esempio, i prezzi delle azioni cambiano in multipli di una "tick size" sulla Borsa di New York (NYSE). La tick size era pari a un ottavo di dollaro prima di luglio 1997, ed è stata pari a un sedicesimo di dollaro da luglio 1997 a gennaio 2001. Pertanto, il rendimento tick-by-tick di un'azione quotata al NYSE non è continuo.

Distribuzione Normale

Un'ipotesi tradizionale negli studi finanziari è che i rendimenti semplici $\{R_{it} \mid t = 1,\ldots,T\}$ siano indipendenti e identicamente distribuiti secondo una distribuzione normale con media e varianza fisse. Questa ipotesi rende trattabili molte proprietà statistiche dei rendimenti degli asset.

Tuttavia, presenta diverse difficoltà. Primo, il limite inferiore di un rendimento semplice è -1. Ma la distribuzione normale può assumere qualsiasi valore sulla retta reale e quindi non ha un limite inferiore. Secondo, se R_{it} è normalmente distribuito, allora il rendimento semplice multiperiodale $R_{it}[k]$ non lo è, poiché è il prodotto di rendimenti a un periodo. Terzo, l'ipotesi di normalità non è supportata da molti dati empirici sui rendimenti, che tendono ad avere una curtosi in eccesso positiva.

Distribuzione Lognormale

Un'alternativa sensata è assumere che i rendimenti composti in continuo r_{it} a singolo periodo siano IID e distribuiti normalmente, il che implica che i rendimenti lordi semplici a singolo periodo siano distribuiti come variabili *lognormali IID*, poiché $r_{it} \equiv \log(1+R_{it})$. Possiamo quindi esprimere il modello lognormale come:

$$r_{it} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

Nel modello lognormale, se la media e la varianza di r_{it} sono rispettivamente μ_i e σ_i^2, allora la media e la varianza dei rendimenti semplici sono date da:

$$\mathbb{E}[R_{it}] = e^{\mu_i + \frac{\sigma_i^2}{2}} - 1$$

$$\text{Var}[R_{it}] = e^{2\mu_i + \sigma_i^2}\left(e^{\sigma_i^2} - 1\right)$$

In alternativa, se assumiamo che la media e la varianza dei rendimenti semplici R_{it} siano rispettivamente m_i e s_i^2, allora, nel modello lognormale, la media e la varianza di r_{it} sono date da:

$$\mathbb{E}[r_{it}] = \log\left(\frac{m_i + 1}{\sqrt{1 + \left(\frac{s_i}{m_i+1}\right)^2}}\right)$$

$$\text{Var}[r_{it}] = \log\left[1 + \left(\frac{s_i}{m_i + 1}\right)^2\right]$$

Il modello lognormale ha il vantaggio aggiuntivo di non violare il principio della responsabilità limitata, poiché questa impone un limite inferiore di zero su $(1+R_{it})$, che è soddisfatto da $(1+R_{it}) = e^{r_{it}}$ quando si assume che r_{it} sia normale.

Il modello lognormale ha una lunga e illustre storia, iniziando con la tesi del matematico francese Louis Bachelier (1900), che conteneva la matematica del moto browniano e della conduzione del calore, cinque anni prima del famoso articolo di Einstein (1905). Il modello lognormale è diventato il cavallo di battaglia della letteratura sul pricing degli asset finanziari.

Ma per quanto attraente sia, il modello lognormale non è coerente con tutte le proprietà dei rendimenti storici delle azioni. Su orizzonti brevi, i rendimenti storici mostrano debole evidenza di asimmetria (*skewness*) e forte evidenza di eccesso di curtosi (*kurtosis*). L'asimmetria, ovvero il terzo momento centrato normalizzato, di una variabile casuale ε con media μ e varianza σ^2 è definita da:

$$S[\varepsilon] \equiv \mathbb{E}\left[\frac{(\varepsilon-\mu)^3}{\sigma^3}\right]$$

La curtosi, o quarto momento centrato normalizzato, di ε è definita da:

$$K[\varepsilon] \equiv \mathbb{E}\left[\frac{(\varepsilon-\mu)^4}{\sigma^4}\right]$$

La distribuzione normale ha asimmetria pari a zero, come tutte le altre distribuzioni simmetriche. La distribuzione normale ha curtosi pari a 3, ma le distribuzioni *a code pesanti* (*fat-tailed*) con massa di probabilità aggiuntiva nelle code presentano curtosi più elevata o addirittura infinita. L'asimmetria e la curtosi possono essere stimate in un campione di dati costruendo le medie campionarie evidenti:

la *media campionaria* è

$$\hat{\mu} \equiv \frac{1}{T}\sum_{t=1}^{T}\varepsilon_t$$

La *varianza campionaria* è

$$\hat{\sigma}^2 \equiv \frac{1}{T}\sum_{t=1}^{T}(\varepsilon_t-\hat{\mu})^2,$$

l'*asimmetria campionaria* è

$$\hat{S} \equiv \frac{1}{T\hat{\sigma}^3} \sum_{t=1}^{T} (\varepsilon_t - \hat{\mu})^3,$$

e la *curtosi campionaria* è

$$\hat{K} \equiv \frac{1}{T\hat{\sigma}^4} \sum_{t=1}^{T} (\varepsilon_t - \hat{\mu})^4$$

In grandi campioni di dati normalmente distribuiti, gli stimatori $\hat{S}$ e $\hat{K}$ sono normalmente distribuiti con medie rispettivamente 0 e 3, e varianze $6/T$ e $24/T$ (vedi [25]). Poiché 3 è la curtosi della distribuzione normale, la *curtosi in eccesso* campionaria è definita come la curtosi campionaria meno 3.

Le stime campionarie dell'asimmetria dei rendimenti giornalieri degli indici azionari statunitensi tendono a essere negative, ma prossime allo zero o positive per le singole azioni. Le stime campionarie della curtosi in eccesso per i rendimenti giornalieri sono elevate sia per gli indici sia per le azioni individuali, il che indica che i rendimenti hanno una massa maggiore nelle code rispetto a quanto previsto da una distribuzione normale.

2.2 Caratteristiche principali delle serie storiche

Il primo passo nell'analisi delle serie temporali consiste nell'identificare le principali componenti che caratterizzano la sequenza di dati. Conoscere le caratteristiche della serie e distinguere quali elementi sono prevedibili e quali sono puramente casuali consente di ridurre i bias previsivi e migliorare l'accuratezza dei modelli statistici utilizzati.

2.2.1 Trend, stagionalità, residui

Di solito, le serie temporali vengono scomposte in tre componenti principali:

1. *Trend*: rappresenta il movimento graduale della serie temporale nel tempo, determinando un aumento o una diminuzione dei valori su lunghi periodi. Se la serie non è stazionaria, il trend può assumere due forme: *trend crescente* se i valori aumentano progressivamente; *trend decrescente* se i valori mostrano una pendenza negativa. Se invece non si osservano particolari tendenze, la serie può essere considerata stazionaria o con trend orizzontale.

2. *Stagionalità*: identifica pattern ricorrenti che si manifestano a intervalli regolari, ad esempio su base annuale, mensile o settimanale. La procedura per eliminare gli effetti stagionali da una serie è detta *destagionalizzazione* (*seasonal adjustment*).

3. *Residui*: noto anche come *fluttuazione irregolare*, rappresenta la parte casuale della serie temporale. Le metodologie previsive cercano di modellare questa componente attraverso distribuzioni di probabilità, in modo da comprendere e anticipare variazioni improvvise dovute a eventi inaspettati.

2.2.2 Stazionarietà

Uno dei concetti fondamentali nell'analisi delle serie temporali è la *stazionarietà*. In generale, una serie temporale stazionaria ha proprietà statistiche che rimangono costanti nel tempo. Rendere una serie stazionaria consente di lavorare con dati più prevedibili: è sufficiente stimare le proprietà statistiche della serie per ipotizzare che esse rimarranno invariate nel futuro, così come sono state nel passato. Possiamo identificare due tipologie diverse di stazionarietà:

- *Stazionarietà stretta* : una serie temporale $\{r_t\}$ è strettamente stazionaria se la distribuzione congiunta di $(r_1, r_2, ..., r_{t_k})$ è identica a quella di $(r_{1+\ell}, r_{2+\ell}, ..., r_{t_k+\ell})$ per ogni $t > 0$, quindi se la sua distribuzione è invariante rispetto a traslazioni nel tempo.

- *Stazionarietà debole o in covarianza* : una serie temporale $\{r_t\}$ è debolmente stazionaria se soddisfa le seguenti proprietà:

  1. $E[r_t] = \mu$
  2. $\text{Cov}(r_t, r_{t-\ell}) = \gamma_\ell$, dove ℓ è un intero arbitrario e γ_ℓ è una funzione di ℓ che non dipende dal tempo.

L'assunzione di stazionarietà debole implica che i primi due momenti della serie $\{r_t\}$ siano finiti e costanti. Se rappresentassimo la serie temporale su un grafico nel tempo, noteremmo che i valori della serie fluttuano attorno a un livello costante. Per questo motivo, le serie temporali debolmente stazionarie sono anche chiamate processi di *ritorno alla media* .

La covarianza γ_ℓ di questi processi è nota come *autocovarianza con lag* ℓ e possiede due proprietà fondamentali: (a) $\gamma_0 = \text{Var}(r_t)$ and (b) $\gamma_{-\ell} = \gamma_\ell$. [2]

Queste proprietà permettono di generalizzare l'autocovarianza con il coefficiente di autocorrelazione ρ_ℓ, che fornisce una misura più intuitiva della correlazione di r_t con i suoi valori ritardati $r_{t-\ell}$. La formula per il coefficiente di autocorrelazione è la seguente:

$$\rho_\ell = \frac{\text{Cov}(r_t, r_{t-\ell})}{\sqrt{\text{Var}(r_t)\text{Var}(r_{t-\ell})}} = \frac{\text{Cov}(r_t, r_{t-\ell})}{\text{Var}(r_t)} = \frac{\gamma_\ell}{\gamma_0}$$

dove $-1 \leq \rho_\ell \leq 1$.[3]

[2] Questa proprietà vale perché $\text{Cov}(r_t, r_{t-(-\ell)}) = \text{Cov}(r_{t-\ell}, r_t) = \text{Cov}(r_{t+\ell}, r_t)$.

[3] Qui si è utilizzata la proprietà $\text{Var}(r_t) = \text{Var}(r_{t-\ell})$.

La versione campionaria di questo coefficiente, data una serie di rendimenti $\{r_t\}_{t=1}^{T}$ con media campionaria $\bar{r} = \frac{\sum_{t=1}^{T} r_t}{T}$, è stimata tramite:

$$\hat{\rho}_\ell = \frac{\sum_{t=\ell+1}^{T}(r_t - \bar{r})(r_{t-\ell} - \hat{r})}{\sum_{t=1}^{T}(r_t - \hat{r})^2}, \quad 0 \leq \ell \leq (T-1).$$

Qui, $\hat{\rho}_\ell$ è uno stimatore distorto di ρ_ℓ, con un bias dell'ordine di $\frac{1}{T}$, il che significa che l'errore diventa trascurabile per campioni sufficientemente ampi.

L'ipotesi di stazionarietà stretta è molto restrittiva e difficile da verificare empiricamente. Per questa ragione, la letteratura finanziaria assume spesso che i log-rendimenti siano debolmente stazionari. Questa ipotesi può essere verificata empiricamente, purché siano disponibili dati storici a sufficienza. Per determinare se una serie temporale è debolmente stazionaria o deve essere trasformata per diventarlo, bisogna analizzare la sua *Funzione di Autocorrelazione(ACF)*, indicata con la sequenza $\hat{\rho}_1, \hat{\rho}_2, ..., \hat{\rho}_m$. Una serie temporale non è correlata serialmente se e solo se $\hat{\rho}_\ell = 0$ per ogni $\ell > 0$. Se tutti i coefficienti di autocorrelazione sono nulli, allora la serie temporale è un *white noise*, ovvero una sequenza di variabili indipendenti e identicamente distribuite, i coefficienti di autocorrelazione dovrebbero risultare prossimi allo zero per tutti i ritardi. Tuttavia, nel caso in cui almeno uno dei coefficienti sia significativamente diverso da zero per un particolare ritardo $\ell = d$, allora la serie mostra correlazione a quel ritardo specifico. Di conseguenza, sarà necessario applicare delle trasformazioni per renderla stazionaria.

Esistono numerosi test statistici basati sulla ACF che permettono di verificare se i coefficienti di autocorrelazione campionari $\{\hat{\rho}_\ell\}_{\ell=1}^{m}$ della serie siano statisticamente diversi da zero. Oltre ai test statistici, un primo approccio per verificare la stazionarietà può essere un'analisi visiva dell'ACF campionaria. Il grafico della funzione di autocorrelazione campionaria, noto anche come *correlogramma*, rappresenta i coefficienti $\{\hat{\rho}_\ell\}_{\ell=1}^{m}$ in funzione dei ritardi della serie temporale. Questo strumento è ampiamente utilizzato per valutare la casualità dei dati: se la serie è puramente casuale, le autocorrelazioni

dovrebbero essere vicine allo zero per tutti i ritardi. Se invece la serie non è casuale, uno o più coefficienti di autocorrelazione saranno significativamente diversi da zero.

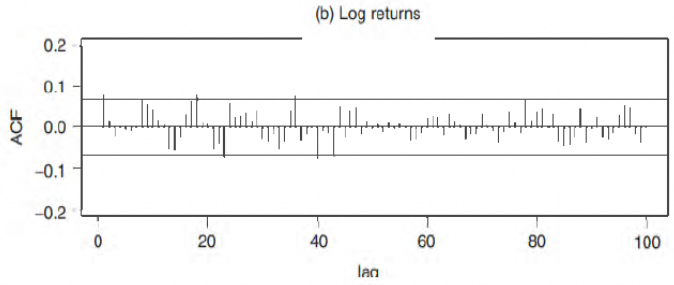


Fig. 2.1: ACF senza autocorrelazione

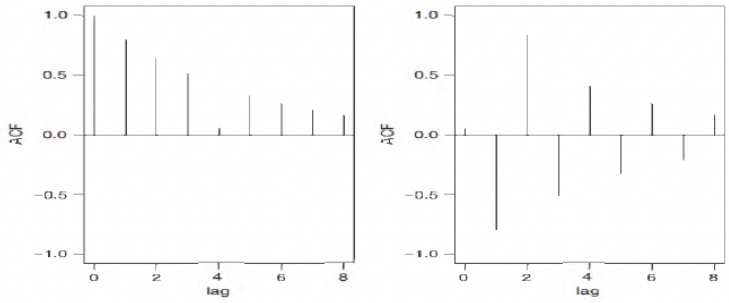


Fig. 2.2: ACF con autocorrelazione

Un altro criterio comune per valutare la significatività della correlazione è il *benchmark delle due deviazioni standard*: se i coefficienti di autocorrelazione campionaria rientrano entro due deviazioni standard, non si osserva alcuna correlazione lineare tra i valori della serie temporale al livello di significatività del 5%. Di conseguenza, la serie non necessita di trasformazioni per diventare stazionaria:

Se invece il grafico dell'ACF mostra picchi significativi a determinati ritardi, significa che la serie è correlata serialmente per quei ritardi e necessita di essere trasformata in una serie stazionaria. Nella pratica, è molto comune osservare una funzione di auto-

correlazione che decresce progressivamente con l'aumentare del ritardo, senza mostrare picchi significativi ai ritardi successivi. Questa situazione suggerisce che l'ordine del processo autoregressivo è $d = 1$, corrispondente al primo ritardo come mostrato in Fig.2.2

Che si tratti di test statistici o di un'analisi visiva del grafico dell'ACF, l'obiettivo è individuare il ritardo più significativo che rende la serie non stazionaria. Questo ritardo è noto come *ordine di differenziazione*, dove ciò si riferisce al processo di trasformazione di una serie temporale non stazionaria in una stazionaria, considerando le variazioni tra i valori successivi della serie.

Ad esempio, la *prima differenza* (con $d = 1$) di una serie temporale è semplicemente la sequenza delle variazioni tra periodi consecutivi:

$$c_t = r_t - r_{t-1}$$

Se la prima differenza di una serie temporale non stazionaria $\{r_t\}$ risulta stazionaria, allora non è necessario applicare ulteriori differenziazioni e si conclude che l'ordine corretto è $d = 1$. Se invece la prima differenza non è ancora stazionaria, possiamo applicare differenziazioni di ordine superiore, sebbene in genere non si superi mai $d = 2$. In base all'ordine di differenziazione utilizzato, possiamo trarre le seguenti conclusioni sul processo sottostante:

- Se $d = 0$ (nessuna differenziazione necessaria), la serie è già stazionaria ed è per natura un processo *mean-reverting*;
- Se $d = 1$, la serie originale è *trend-stationary*, ovvero presenta una tendenza media costante nel tempo;
- Se $d = 2$, la serie originale presenta un *trend variabile nel tempo*.

Se la serie temporale continua a non essere stazionaria anche dopo due ordini di differenziazione, potrebbe essere necessario applicare un'ulteriore trasformazione nota come *Differenziazione Stagionale* per rimuovere la componente stagionale. Quando una serie temporale mostra un pattern stagionale che si ripete regolarmente di anno in anno, può essere utile applicare aggiustamenti stagionali per stimare e proiettare correttamente tale schema. Le principali metodologie per la rimozione della stagionalità sono due:

1. *Aggiustamento Stagionale Moltiplicativo*: Questo metodo è più adatto quando l'effetto stagionale si manifesta in termini percentuali, ovvero quando l'ampiezza delle fluttuazioni stagionali aumenta con la crescita della serie nel tempo. Per eliminare il pattern stagionale moltiplicativo, ogni valore della serie deve essere diviso per un indice stagionale che rappresenta il livello percentuale normale osservato in quel periodo, sulla base dei dati storici.

2. *Aggiustamento Stagionale Additivo*: Questo metodo è più adatto quando le variazioni stagionali si mantengono relativamente costanti nel tempo. Nel caso dell'aggiustamento stagionale additivo, ogni valore della serie temporale viene corretto aggiungendo o sottraendo una quantità fissa, che rappresenta l'entità assoluta con cui la serie tende a trovarsi sopra o sotto il valore atteso in quella stagione, sulla base delle osservazioni storiche. Tuttavia, gli aggiustamenti stagionali additivi sono meno comuni in natura.

In alcuni casi, può risultare difficile determinare quale ordine di differenziazione sia necessario per rendere stazionaria la serie temporale. In queste situazioni, un test delle *radici unitarie* (*unit-root test*) può fornire una risposta più precisa.

Tra i vari test disponibili, il test di Dickey-Fuller (DF) si basa sull'ipotesi nulla che la serie temporale contenga una radice unitaria. Se questa ipotesi viene confermata, significa che la serie non è un processo stazionario, ma segue un modello autoregressivo con un

trend a ritardo 1. Una versione più avanzata di questo test è l'Augmented Dickey-Fuller (ADF), che è stato sviluppato per gestire dati più complessi, inclusi processi autoregressivi di ordine superiore. Nonostante ciò, il test ADF conserva le tre formulazioni del test originale di Dickey-Fuller, ciascuna corrispondente a un diverso modello del processo con radice unitaria:

- $\Delta Y_t = \gamma Y_{t-1} + \sum_{j=1}^{d} (\delta_j \cdot \Delta Y_{t-j}) + \varepsilon_t$
- $\Delta Y_t = \alpha + \gamma Y_{t-1} + \sum_{j=1}^{d} (\delta_j \cdot \Delta Y_{t-j}) + \varepsilon_t$
- $\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{j=1}^{d} (\delta_j \cdot \Delta Y_{t-j}) + \varepsilon_t$

dove:

Y_t è la serie osservata (nel nostro caso la serie dei log-prezzi), d è l'ordine di differenziazione considerato, ε_t è il termine di errore residuo i.i.d. α rappresenta il termine di drift costante, β è il coefficiente del trend temporale, γ è il coefficiente su cui si basa il test. L'attenzione si concentra sulla verifica dell'ipotesi nulla:$H_0 : \gamma = 0$ che indica la presenza di una radice unitaria, ovvero che il processo non è stazionario. L'ipotesi alternativa è invece: $H_1 : \gamma < 0$ che implica che la serie originale non contiene radici unitarie ed è stazionaria.

Le tre equazioni del test differiscono per la presenza del termine di drift α e del trend lineare βt. Sulla base di queste differenze, otteniamo le seguenti formulazioni dell'ipotesi nulla:

- $H_0 : \gamma = 0 \Rightarrow Y_t$ è un *random walk*, mentre $H_1 : \gamma < 0 \Rightarrow Y_t$ è un *processo stazionario*.
- $H_0 : \gamma = 0$ e $\alpha \neq 0 \Rightarrow Y_t$ è un *random walk con drift*, mentre $H_1 : \gamma < 0$ e $\alpha \neq 0 \Rightarrow Y_t$ è un *processo stazionario con livello costante*.

- $H_0 : \gamma = 0$ e $\beta \neq 0 \Rightarrow Y_t$ è un *random walk con trend*, mentre $H_1 : \gamma < 0$ e $\beta \neq 0 \Rightarrow Y_t$ è un *processo stazionario con trend*.

2.2.3 White noise e linearità

Una serie temporale r_t è detta *white noise* se $\{r_t\}$ è una sequenza di variabili casuali indipendenti e identicamente distribuite (i.i.d.) con media e varianza finite. In particolare, se r_t è normalmente distribuita con media zero e varianza σ^2, la serie è detta *white noise gaussiano*. Per una serie di rumore bianco, tutte le funzioni di autocorrelazione (ACF) sono nulle. Nella pratica, se tutte le ACF campionarie sono prossime a zero, allora si può considerare la serie come rumore bianco. Il comportamento delle autocorrelazioni campionarie dei rendimenti dell'indice ponderato per il valore indica che, per alcuni rendimenti, è necessario modellare la dipendenza seriale prima di poter effettuare ulteriori analisi. Nella sezione seguente, discuteremo alcuni semplici modelli di serie temporali che sono utili per modellare la struttura dinamica di una serie. I concetti qui presentati sono anche utili più avanti nella modellazione della volatilità dei rendimenti.

Una serie temporale r_t si dice *lineare* se può essere scritta come:

$$r_t = \mu + \sum_{i=0}^{\infty} \psi_i a_{t-i},$$

dove μ è la media di r_t, $\psi_0 = 1$, e $\{a_t\}$ è una sequenza di variabili casuali indipendenti e identicamente distribuite con media zero e distribuzione ben definita (cioè, $\{a_t\}$ è una serie di rumore bianco). Più avanti si vedrà che a_t rappresenta l'informazione nuova al tempo t nella serie temporale, ed è spesso chiamata *innovazione* o *shock* al tempo t.

In questo libro, ci concentriamo principalmente sul caso in cui a_t è una variabile casuale continua. Tuttavia, non tutte le serie finanziarie sono lineari. Studieremo modelli lineari e non lineari nel Capitolo 4.

Per una serie temporale lineare come in Eq. (2.4), la struttura dinamica di r_t è governata dai coefficienti ψ_i, chiamati ψ-*pesi* di r_t nella letteratura sulle serie temporali.

Se r_t è debolmente stazionaria, possiamo ricavarne media e varianza facilmente sfruttando l'indipendenza di $\{a_t\}$, come segue:

$$\mathbb{E}(r_t) = \mu, \qquad \text{Var}(r_t) = \sigma_a^2 \sum_{i=0}^{\infty} \psi_i^2,$$

dove σ_a^2 è la varianza di a_t. Poiché $\text{Var}(r_t) < \infty$, la serie $\{\psi_i^2\}$ deve essere convergente, cioè $\psi_i^2 \to 0$ per $i \to \infty$. Di conseguenza, per una serie stazionaria, l'impatto di uno shock lontano nel tempo a_{t-i} sul rendimento r_t svanisce con l'aumentare di i.

L'*autocovarianza al lag* ℓ di r_t è:

$$\gamma_\ell = \text{Cov}(r_t, r_{t-\ell}) = \mathbb{E}\left[\left(\sum_{i=0}^{\infty} \psi_i a_{t-i}\right)\left(\sum_{j=0}^{\infty} \psi_j a_{t-\ell-j}\right)\right]$$

$$\gamma_\ell = \mathbb{E}\left(\sum_{i,j=0}^{\infty} \psi_i \psi_j a_{t-i} a_{t-\ell-j}\right) = \sum_{j=0}^{\infty} \psi_{j+\ell} \psi_j \mathbb{E}(a_{t-\ell-j}^2) = \sigma_a^2 \sum_{j=0}^{\infty} \psi_j \psi_{j+\ell}$$

Di conseguenza, i ψ-pesi sono legati alle autocorrelazioni di r_t come segue:

$$\rho_\ell = \frac{\gamma_\ell}{\gamma_0} = \frac{\sum_{i=0}^{\infty} \psi_i \psi_{i+\ell}}{1 + \sum_{i=1}^{\infty} \psi_i^2}, \quad \ell \geq 0$$

dove $\psi_0 = 1$. I modelli lineari per serie temporali sono modelli econometrici e statistici utilizzati per descrivere l'andamento dei ψ-pesi di r_t. Per una serie temporale debolmente stazionaria, $\psi_i \to 0$ quando $i \to \infty$ e, di conseguenza, $\rho_\ell \to 0$ all'aumentare di ℓ.

Nel caso dei rendimenti degli asset, ciò implica che – come previsto – la dipendenza lineare del rendimento corrente r_t da uno shock passato remoto $r_{t-\ell}$ si riduce per valori grandi di ℓ.

Capitolo 3

Machine learning per la previsione dei regimi di mercato

3.1 Introduzione al Machine Learning

Il *machine learning* (ML) è una branca dell'intelligenza artificiale che si occupa dello sviluppo di algoritmi capaci di apprendere dai dati e migliorare le proprie prestazioni nel tempo, senza l'impiego di istruzioni programmate in modo esplicito. L'obiettivo principale del ML è individuare pattern nascosti all'interno dei dati e utilizzarli per effettuare previsioni o supportare decisioni. Grazie alla crescente disponibilità di informazioni e all'avanzamento delle tecnologie computazionali, il machine learning trova oggi applicazione in numerosi settori, tra cui finanza, sanità, marketing e logistica. È forse il settore dell'AI con il più alto potenziale trasformativo, sia per la ricerca scientifica sia per l'impatto diretto sulle dinamiche economiche e occupazionali. Con molta probabilità si tratta del processo che genera più entusiasmo e che potrebbe quasi completamente trasformare il mondo del lavoro. Ci si potrebbe chiedere, quali sono i vantaggi che la società potrebbe trarre da sostuire l'uomo con le macchine? Uno tra tutti è la velocità.

Le macchine possono elaborare dati e giungere a conclusioni in maniera molto più veloce degli essere umani. Le decisioni prodotte da un sistema automatizzato si distinguono spesso per coerenza, riproducibilità e velocità, qualità che superano in molti casi quelle umane.

Da moltissimi anni il settore finanziario ha iniziato a sperimentare soluzioni basate sul *machine learning*, ma è solo nell'ultimo ventennio che queste tecnologie si sono diffuse in modo capillare, diventando un pilastro fondamentale dell'intelligenza artificiale. L'idea centrale alla base del *machine learning* consiste nel dotare i sistemi informatici della capacità di apprendere dai dati e migliorare le proprie decisioni. Tale meccanismo è reso possibile da algoritmi, detti *learning algorithms*, progettati per riconoscere pattern e adattarsi sulla base dell'esperienza. Questi algoritmi sono spesso impiegati per compiti decisionali o previsivi, sfruttando l'analisi di dati storici. Il loro sviluppo è stato agevolato dalla disponibilità di grandi quantità di dati (*big data*) e dall'aumento della potenza di calcolo, che ne ha potenziato l'efficacia.

Il *machine learning* può essere classificato in quattro principali categorie, ciascuna caratterizzata da differenti approcci alla gestione e interpretazione dei dati:

- **Unsupervised learning**: modelli usati per identificare strutture nei dati senza etichette, spesso attraverso tecniche di clustering o riduzione della dimensionalità.

- **Supervised learning**: consiste nell'addestramento di un modello su un insieme di dati etichettati (input/output) al fine di apprendere una funzione predittiva.

- **Semi-supervised learning**: si basa su dataset parzialmente etichettati, combinando le logiche supervisionate e non supervisionate per migliorare la generalizzazione.

- **Reinforcement learning**: il modello interagisce con un ambiente dinamico e apprende attraverso un meccanismo di ricompensa, ottimizzando le azioni per massimizzare un obiettivo.

All'interno dell'apprendimento supervisionato, una sottofamiglia di tecniche particolarmente avanzate è rappresentata dal *deep learning*. Questo approccio si basa sull'uso di *reti neurali profonde*, ovvero strutture matematiche ispirate al funzionamento del cervello umano, composte da numerosi strati (layers) di elaborazione. Il termine "profondo" si riferisce proprio al numero elevato di livelli nella rete. Il *deep learning* ha trovato larga applicazione nel trattamento di dati non strutturati, come immagini, testo e audio, grazie alla sua capacità di estrarre rappresentazioni astratte e informative da dati complessi. Tuttavia, l'addestramento di queste reti richiede notevoli risorse computazionali e grandi quantità di dati, ma i benefici in termini di accuratezza ed efficienza predittiva sono spesso superiori rispetto ai modelli tradizionali. È per questo motivo che il *deep learning* rappresenta oggi una delle frontiere più promettenti dell'apprendimento automatico.

3.2 Modelli statistico-probabilistici

I modelli statistici classici e il *machine learning* condividono un obiettivo comune: comprendere e modellare i dati per formulare previsioni attendibili. Tuttavia, differiscono per approccio, flessibilità e ambito applicativo. I modelli statistici si basano su ipotesi esplicite circa la distribuzione dei dati e la relazione tra variabili, adottando una struttura formale che consente un'elevata interpretabilità. Al contrario, il machine learning privilegia la capacità predittiva e l'adattamento automatico ai dati, anche in assenza di assunzioni parametriche rigide.

Nonostante queste differenze, tra i due approcci non esiste una contrapposizione netta, bensì una progressiva convergenza. Molti algoritmi di machine learning traggono origine da modelli statistici, estendendoli per operare in contesti più complessi, ad alta dimensionalità e caratterizzati da forte non linearità. Alcuni modelli, come gli *Hidden Markov Models*, si trovano esattamente all'intersezione tra statistica e apprendimento automatico: pur avendo una struttura probabilistica ben definita, vengono spesso utilizzati in ambienti data-driven, integrandosi con tecniche di ML per classificazione, previsione e analisi dei regimi di mercato. Nel contesto di questa tesi, machine learning e i modelli statistico-probabilistici non vengono trattati come alternative, bensì come strumenti complementari. La scelta di utilizzare gli HMM come nucleo predittivo riflette proprio questa sintesi metodologica.

Modelli lineari classici: AR, MA, ARIMA

Il processo autoregressivo (Autoregressive, AR) è uno dei modelli più comuni e semplici per analizzare le serie temporali. In un processo AR di ordine p, denotato come AR(p), il valore attuale della serie è espresso come combinazione lineare dei suoi p valori passati, più un termine di errore casuale. La forma generale è:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

dove ϕ_0 è una costante, $\phi_1, \ldots, \phi_p$ sono i parametri autoregressivi, e ε_t è un rumore bianco con media zero e varianza costante.

L'idea alla base del modello AR è che il valore futuro di una variabile può essere previsto sulla base dei suoi valori passati. Questo approccio riflette spesso il comportamento di variabili economiche e finanziarie, come il PIL, i prezzi azionari o i tassi di cambio, che tendono a mostrare una certa dipendenza temporale.

Un AR(1), ad esempio, è dato da:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \varepsilon_t$$

e mostra che il valore di Y_t dipende linearmente da Y_{t-1} e da un errore casuale. La stazionarietà del processo AR richiede che i coefficienti ϕ_i rispettino determinate condizioni (ad esempio, per AR(1) si ha $|\phi_1| < 1$). Questa proprietà è essenziale per garantire che la serie non esploda nel tempo e presenti varianza finita.

Il processo AR è spesso utilizzato come componente nei modelli più complessi, come ARMA e ARIMA, ed è uno strumento base nelle previsioni a breve termine.

Il processo di media mobile (Moving Average, MA) è un modello fondamentale nell'analisi delle serie temporali. Un processo MA di ordine q, indicato come MA(q), rappresenta una serie in cui il valore corrente della variabile dipende linearmente da un numero finito di shock aleatori passati, noti come errori o *innovazioni*. La forma generale di un processo MA(q) è:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

dove μ è la media della serie, θ_i sono i coefficienti del modello, e ε_t è una sequenza di rumore bianco (*white noise*), cioè una variabile aleatoria a media zero e varianza costante.

A differenza dei processi autoregressivi (AR), dove i valori passati della variabile influenzano direttamente il presente, nei processi MA è l'effetto degli shock passati a determinare l'evoluzione della serie. Questo rende i processi MA particolarmente adatti a modellare fenomeni con effetti di breve durata o in cui gli shock si dissipano rapidamente.

Un esempio semplice è il MA(1), in cui solo l'errore precedente influenza il valore corrente. Questo tipo di processo è spesso utilizzato per modellare volatilità temporanee nei dati economici o finanziari. Tuttavia, poiché i processi MA non sono autoregressivi, l'identificazione e la stima dei loro parametri può essere più complessa e richiede metodi come la funzione di autocovarianza o strumenti di ottimizzazione numerica.

Il modello ARMA (Autoregressive Moving Average) rappresenta una combinazione dei modelli autoregressivo (AR) e media mobile (MA), ed è uno strumento fondamentale nell'analisi delle serie temporali stazionarie. Un processo ARMA(p,q) viene espresso come:

$$Y_t = \phi_0 + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

dove:

- ϕ_i sono i coefficienti autoregressivi (AR);
- θ_j sono i coefficienti della media mobile (MA);
- ε_t è un rumore bianco con media nulla e varianza costante;
- p è l'ordine della parte AR, q è l'ordine della parte MA.

Questo modello permette di cogliere sia la dipendenza diretta del valore attuale dai valori passati della variabile (componente AR), sia la dipendenza dagli shock passati (componente MA). La flessibilità del modello ARMA consente di rappresentare una vasta gamma di comportamenti dinamici nelle serie temporali, pur mantenendo una struttura relativamente semplice.

Il processo ARMA viene solitamente utilizzato quando la serie è stazionaria in senso stretto o può essere resa tale tramite trasformazioni (ad esempio differenziazione). Per serie non stazionarie si ricorre all'estensione ARIMA, che include un termine integrato.

L'identificazione di un modello ARMA adatto richiede una corretta analisi delle funzioni di autocorrelazione (ACF) e autocorrelazione parziale (PACF), che permettono di individuare gli ordini p e q più appropriati.

Nel campo della modellazione delle serie temporali, i modelli statistici classici come AR , MA e ARMA rappresentano da decenni un punto di riferimento teorico per l'analisi e la previsione di dati temporali. Questi modelli si basano su una struttura parametrica lineare e fanno uso esclusivo di informazioni interne alla serie storica, come i suoi valori passati e gli errori commessi nei periodi precedenti. Proprio grazie alla loro semplicità, trasparenza e interpretabilità, hanno trovato larga applicazione in ambito economico-finanziario, soprattutto per la stima a breve termine di variabili stazionarie come i rendimenti.

Tuttavia, questa stessa semplicità può costituire un limite significativo, soprattutto quando si tenta di applicare tali modelli a fenomeni complessi come i mercati finanziari. In contesti reali, infatti, le serie temporali mostrano spesso comportamenti non lineari, cambiamenti strutturali, volatilità variabile nel tempo e forti dipendenze da fattori esogeni. I modelli AR, MA e ARMA, per quanto teoricamente eleganti, non sono in grado di adattarsi autonomamente a queste condizioni. La loro efficacia dipende fortemente dal rispetto di alcune ipotesi fondamentali, prima fra tutte la stazionarietà e dalla corretta identificazione dell'ordine del modello. Inoltre, la loro capacità predittiva si limita a un orizzonte temporale ristretto, e la struttura lineare sottostante non consente di cogliere relazioni complesse o interazioni tra molteplici variabili.

Al contrario, i modelli di ML rappresentano un'evoluzione sostanziale nel campo della previsione. Essi non impongono una struttura funzionale rigida al processo generatore dei dati, ma apprendono direttamente dai dati stessi, individuando pattern anche molto complessi e non lineari. Questa flessibilità consente loro di adattarsi meglio a situazioni dinamiche e a dati rumorosi, tipici dei mercati finanziari. Inoltre, questa tipologia

di modelli possono sfruttare molteplici fonti di informazione contemporaneamente, includendo non solo i valori passati della serie target, ma anche indicatori tecnici, dati macroeconomici o variabili esterne.

Naturalmente, l'approccio *machine learning* non è privo di criticità. Uno degli svantaggi principali è la minore interpretabilità dei modelli, specialmente quelli più complessi come le reti neurali profonde o gli ensemble non lineari. Inoltre, l'efficacia predittiva dipende in larga parte dalla qualità dei dati e dalla fase di preprocessing, e l'assenza di una teoria economica sottostante rende più difficile giustificare i risultati dal punto di vista teorico.

Riassumendo, mentre i modelli AR, MA e ARMA forniscono una base solida per l'analisi descrittiva e la previsione a breve termine in contesti semplici e stazionari, i modelli di ML risultano spesso più efficaci in termini di accuratezza e adattabilità, soprattutto quando si ha a che fare con serie finanziarie complesse e altamente volatili. L'adozione di un approccio piuttosto che dell'altro dipende dunque dagli obiettivi dell'analisi, dalla disponibilità di dati e dalla necessità di bilanciare precisione, trasparenza e robustezza.

3.3 Hidden Markov Model (HMM)

3.3.1 Caratteristiche principali

I modelli HMM sono ampiamente utilizzati in numerosi ambiti applicativi, tra cui il riconoscimento vocale, l'identificazione di attività nei video, la localizzazione di geni e il tracciamento di gesti. In questa sezione verranno illustrati cosa sono gli HMM, come si utilizzano nell'apprendimento automatico, i loro vantaggi, le limitazioni. Risulta fondamentale l'applicazione degli HMM in ambito finanziario essendo utilizzati per classificare regimi di mercato, dunque fondamentale, all'interno della strategia proposta

in questo elaborato.

Un modello di Markov nascosto è uno strumento per rappresentare distribuzioni di probabilità su sequenze di osservazioni. In questo modello,come spiegato in [7], una variabile osservata X_t al tempo t è generata da un processo stocastico, ma lo stato Z_t che ha prodotto tale osservazione non è direttamente accessibile: è, appunto, *nascosto*.

Si assume che il processo nascosto soddisfi la *proprietà di Markov*: lo stato Z_t dipende unicamente dallo stato precedente Z_{t-1}, ovvero:

$$P(Z_t \mid Z_{t-1}, Z_{t-2}, \ldots, Z_1) = P(Z_t \mid Z_{t-1})$$

Questa assunzione definisce un *modello di Markov di primo ordine*. In modo più generale, un modello di ordine n dipenderebbe da n stati precedenti. La Fig. 3.5 rappresenta un HMM come rete bayesiana, con gli stati nascosti mostrati in grigio.

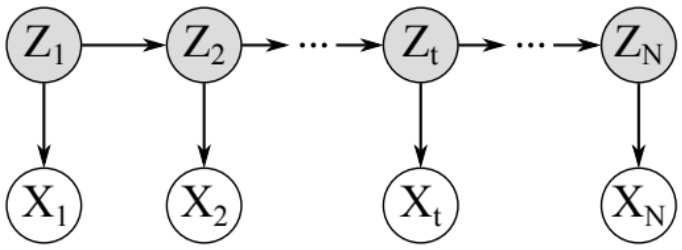


Fig. 3.1: rete bayesiana

È importante notare che il termine “tempo” in questo contesto, un HMM rappresenta una distribuzione probabilistica su sequenze temporali. non indica necessariamente un intervallo cronologico, ma piuttosto una sequenza ordinata di osservazioni. Può quindi riferirsi anche a posizioni all’interno di una sequenza simbolica o testuale. La distribuzione congiunta della sequenza di stati $Z_{1:N}$ e osservazioni $X_{1:N}$ in un HMM di primo ordine è data da:

$$P(Z_{1:N}, X_{1:N}) = P(Z_1)P(X_1 \mid Z_1)\prod_{t=2}^{N} P(Z_t \mid Z_{t-1})P(X_t \mid Z_t)$$

dove la notazione $Z_{1:N}$ rappresenta la sequenza di variabili $Z_1, Z_2, \ldots, Z_N$.

Questa espressione può anche essere scritta in forma alternativa come:

$$P(X_{1:N}, Z_{1:N}) = P(Z_1)\prod_{t=2}^{N} P(Z_t \mid Z_{t-1})\prod_{t=1}^{N} P(X_t \mid Z_t)$$

Le due formulazioni sono equivalenti, e sottolineano il fatto che le osservazioni sono condizionate dagli stati nascosti, che evolvono nel tempo secondo la dinamica markoviana. Ci sono cinque elementi che caratterizzano un modello di Markov nascosto:

1. *Numero di stati nel modello, K*: Questo è il numero di stati che il processo di Markov nascosto può assumere. Gli stati sono spesso collegati al fenomeno che si sta modellando. Ad esempio, se un HMM viene utilizzato per il riconoscimento gestuale, ogni stato può rappresentare un gesto diverso o una parte del gesto. Gli stati sono rappresentati come interi $1, \ldots, K$.Codificheremo lo stato Z_t al tempo t come un vettore binario di dimensione $K \times 1$, in cui l'unico elemento diverso da zero è quello nella k-esima posizione (cioè $Z_{tk} = 1$), che corrisponde allo stato $k \in K$ al tempo t. Anche se questa rappresentazione può sembrare artificiosa, ci sarà utile nei calcoli.

2. *Numero di osservazioni distinte, Ω*: Le osservazioni sono rappresentate come interi $1, \ldots, \Omega$. Codificheremo l'osservazione X_t al tempo t come un vettore binario di dimensione $\Omega \times 1$, in cui l'unico elemento diverso da zero è l'l-esimo (cioè $X_{tl} = 1$), che corrisponde allo stato $l \in \Omega$ al tempo t.

3. *Modello di transizione degli stati, A*: è una matrice $K \times K$ i cui elementi A_{ij} rappresentano la probabilità di transizione dallo stato $Z_{t-1,i}$ allo stato $Z_{t,j}$ in un

singolo passo temporale, dove $i, j \in \{1, \ldots, K\}$. Formalmente, si può scrivere:

$$A_{ij} = P(Z_t = j \mid Z_{t-1} = i)$$

Ogni riga della matrice A somma a 1, ovvero $\sum_j A_{ij} = 1$, e per questo motivo A è detta matrice stocastica. Se ogni stato può raggiungere qualsiasi altro stato in un solo passo (modello completamente connesso), allora $A_{ij} > 0$ per

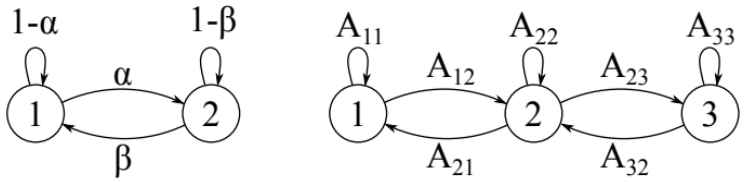


Fig. 3.2: Un diagramma di transizione di stato per una catena di Markov ergodica a 2 stati e a 3 stati.

tutti i, j, allora la matrice A è completamente connessa. Altrimenti, A conterrà alcuni elementi nulli.

La Figura 3.2 mostra due diagrammi di transizione degli stati: uno per un HMM con 2 stati e uno per un HMM con 3 stati, entrambi di primo ordine. Per questi diagrammi, i modelli di transizione degli stati sono dati da:

$$A_{(a)} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}, \quad A_{(b)} = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}$$

La probabilità condizionata può essere espressa come:

$$P(Z_t \mid Z_{t-1}) = \prod_{i=1}^{K} \prod_{j=1}^{K} A_{ij}^{Z_{t-1,i} Z_{t,j}}$$

Applicando il logaritmo, otteniamo:

$$\log P(Z_t \mid Z_{t-1}) = \sum_{i=1}^{K} \sum_{j=1}^{K} Z_{t-1,i} Z_{t,j} \log A_{ij}$$

che può essere scritta in forma matriciale compatta come:

$$\log P(Z_t \mid Z_{t-1}) = Z_t^\top \log(\mathbf{A}) Z_{t-1}$$

4. *Modello di osservazione, B*: chiamato anche *matrice di emissione*, B è una matrice di dimensione $\Omega \times K$ i cui elementi B_{kj} rappresentano la probabilità di osservare il simbolo k quando il processo si trova nello stato nascosto j al tempo t. Formalmente:

$$B_{kj} = P(X_t = k \mid Z_t = j)$$

La probabilità condizionata può essere espressa come:

$$P(X_t \mid Z_t) = \prod_{j=1}^{K} \prod_{k=1}^{\Omega} B_{kj}^{Z_{t,j} X_{t,k}}$$

Applicando il logaritmo, si ottiene:

$$\log P(X_t \mid Z_t) = \sum_{j=1}^{K} \sum_{k=1}^{\Omega} Z_{t,j} X_{t,k} \log B_{kj}$$

che può essere scritta in forma matriciale come:

$$\log P(X_t \mid Z_t) = X_t^\top \log(B) Z_t$$

5. *Distribuzione iniziale degli stati, π*: è un vettore di dimensione $K \times 1$ che rappresenta le probabilità iniziali associate a ciascuno stato. Ogni elemento π_i è definito come:

$$\pi_i = P(Z_{1i} = 1)$$

La probabilità condizionata di osservare Z_1 dato π può essere scritta come:

$$P(Z_1 \mid \pi) = \prod_{i=1}^{K} \pi_i^{Z_{1i}}$$

Date queste cinque componenti, un HMM è completamente specificato. Nella letteratura, il modello viene spesso indicato in forma compatta come:

$$\lambda = (A, B, \pi)$$

Secondo Rabiner [19], affinché un HMM sia effettivamente utile nelle applicazioni pratiche, è necessario risolvere i seguenti tre problemi:

1. *Problema*: Date una sequenza di osservazioni $X_1, \ldots, X_N$ e un modello $\lambda = (A, B, \pi)$, come si può calcolare in modo efficiente $P(X_{1:N} \mid \lambda)$, ovvero la probabilità della sequenza osservata dato il modello?

   Questo problema rientra nell'ambito dell'*inference esatta* ed è trattato nei materiali delle lezioni. Può essere risolto utilizzando la tecnica del *forward filtering*.

2. *Problema*: Date una sequenza di osservazioni $X_1, \ldots, X_N$ e un modello λ, come possiamo determinare la sequenza di stati nascosti $Z_1, \ldots, Z_N$ che "meglio spiega" le osservazioni? Questo equivale a trovare la sequenza più probabile di stati nascosti, come indicato nelle lezioni, e può essere risolto tramite l'*algoritmo di Viterbi*. Un problema correlato è quello di calcolare la probabilità che il sistema si trovi nello stato Z_{tk} dato l'insieme delle osservazioni, ovvero $P(Z_t = k \mid X_{1:N})$, il quale può essere risolto utilizzando l'*algoritmo forward-backward*.

3. *Problema*: Come possiamo aggiornare i parametri del modello $\lambda = (A, B, \pi)$ al fine di massimizzare $P(X_{1:N} \mid \lambda)$?

3.3.2 L'algoritmo Forward-Backward

L'algoritmo *forward-backward* è un algoritmo di programmazione dinamica che si basa sul passaggio di messaggi (*belief propagation*). Questo metodo consente di calcolare le distribuzioni marginali filtrate e smussate, che possono poi essere utilizzate per eseguire inferenza, stima MAP, classificazione di sequenze, rilevamento di anomalie e clustering basato su modello.

Qui verrà seguita la derivazione come viene presentata nel testo di Murphy [17].

1. *Algoritmo Forward*

   In questa fase, calcoliamo le marginali filtrate, $P(Z_t \mid X_{1:T})$, utilizzando il ciclo *predict-update*. La previsione a un passo in avanti, detta *one-step-ahead predictive density*, è data da:

$$P(Z_t = j \mid X_{1:t-1}) = \sum_{i=1}^{K} P(Z_t = j \mid Z_{t-1} = i) P(Z_{t-1} = i \mid X_{1:t-1})$$

   Questa distribuzione funge da nuova prior per il tempo t.

   Nel passo di aggiornamento, l'osservazione X_t al tempo t viene assimilata secondo la regola di Bayes:

$$\alpha_t(j) \triangleq P(Z_t = j \mid X_{1:t})$$

$$= P(Z_t = j \mid X_t, X_{1:t-1})$$

$$= \frac{P(X_t \mid Z_t = j, X_{1:t-1})\, P(Z_t = j \mid X_{1:t-1})}{\sum_j P(X_t \mid Z_t = j, X_{1:t-1})\, P(Z_t = j \mid X_{1:t-1})}$$

$$= \frac{1}{C_t} P(X_t \mid Z_t = j)\, P(Z_t = j \mid X_{1:t-1})$$

Poiché le osservazioni $X_{1:t-1}$ si annullano (sono d-separate da X_t), la costante C_t rappresenta il termine di normalizzazione. La costante è definita da:

$$C_t \triangleq P(X_t \mid X_{1:t-1}) = \sum_{j=1}^{K} P(X_t \mid Z_t = j)\, P(Z_t = j \mid X_{1:t-1})$$

Il vettore $\alpha_t = P(Z_t \mid X_{1:T})$ di dimensione $K \times 1$ è noto come *belief state filtrato* al tempo t.

In notazione matriciale, il passo ricorsivo può essere scritto come:

$$\alpha_t \propto \psi_t \odot \left(A^\top \alpha_{t-1} \right)$$

dove $\psi_t = [\psi_{t1}, \psi_{t2}, \ldots, \psi_{tK}] = \{P(X_t \mid Z_t = i)\}_{1 \le i \le K}$ rappresenta l'evidenza locale al tempo t.

Qui, A è la matrice di transizione e $\odot$ indica il prodotto di Hadamard, cioè la moltiplicazione elemento per elemento tra vettori.

Il logaritmo della probabilità della sequenza osservata (evidenza) è calcolato come:

$$\log P(X_{1:N} \mid \lambda) = \sum_{t=1}^{N} \log P(X_t \mid X_{1:t-1}) = \sum_{t=1}^{N} \log C_t$$

Questa è, di fatto, la soluzione del problema 1 enunciato da Rabiner [19]. Lavorare nel dominio logaritmico consente di evitare problemi numerici legati all'underflow durante i calcoli.

Algorithm 1 Forward algorithm

1: Input: $\boldsymbol{A}$, $\psi_{1:N}$, $\boldsymbol{\pi}$
2: $[\boldsymbol{\alpha}_1, C_1]$ = normalize$(\psi_1 \odot \boldsymbol{\pi})$;
3: **for** $t = 2 : N$ **do**
4: $\quad [\boldsymbol{\alpha}_t, C_t]$ = normalize$(\psi_t \odot (\boldsymbol{A}^\top \boldsymbol{\alpha}_{t-1}))$;
5: Return $\boldsymbol{\alpha}_{1:N}$ and $\log P(X_{1:N}) = \sum_t \log C_t$
6: Sub: $[\boldsymbol{\alpha}, C]$ = normalize$(\boldsymbol{u})$: $C = \sum_j u_j$; $\alpha_j = u_j/C$;

Fig. 3.3

2. *Algoritmo Forward-Backward*

Ora che abbiamo calcolato gli stati di credenza filtrati α tramite i messaggi forward, possiamo calcolare i messaggi backward per ottenere le marginali smussate.

Le marginali posteriori smussate sono proporzionali a:

$$P(Z_t = j \mid X_{1:N}) \propto P(Z_t = j, X_{t+1:N} \mid X_{1:t})$$

$$\propto P(Z_t = j \mid X_{1:t})\, P(X_{t+1:N} \mid Z_t = j, X_{1:t})$$

Questa quantità rappresenta la probabilità che lo stato nascosto al tempo t sia j.

Definiamo quindi la probabilità condizionata dell'evidenza futura come:

$$\beta_t(j) \triangleq P(X_{t+1:N} \mid Z_t = j)$$

Definiamo anche la marginale posteriore smussata desiderata:

$$\gamma_t(j) \triangleq P(Z_t = j \mid X_{1:N})$$

Possiamo quindi riscrivere l'equazione precedente come:

$$\gamma_t(j) \propto \alpha_t(j)\,\beta_t(j)$$

Possiamo ora calcolare i termini β ricorsivamente da destra verso sinistra:

$$\beta_{t-1}(i) = P(X_{t:N} \mid Z_{t-1} = i)$$

$$= \sum_j P(Z_t = j, X_t, X_{t+1:N} \mid Z_{t-1} = i)$$

$$= \sum_j P(X_{t+1:N} \mid Z_t = j, X_t, Z_{t-1} = i) \cdot P(Z_t = j, X_t \mid Z_{t-1} = i)$$

$$= \sum_j P(X_{t+1:N} \mid Z_t = j) \cdot P(X_t \mid Z_t = j) \cdot P(Z_t = j \mid Z_{t-1} = i)$$

$$= \sum_j \beta_t(j)\,\psi_t(j) A(i,j)$$

oppure in forma matriciale compatta:

$$\beta_{t-1} = A\,(\psi_t \odot \beta_t)$$

Il caso base della ricorsione è:

$$\beta_N(i) = P(X_{N+1:N} \mid Z_N = i) = P(\emptyset \mid Z_N = i) = 1$$

Infine, la marginale smussata è calcolata come:

$$\gamma_i = \frac{\alpha_i \odot \beta_i}{\sum_j \alpha_i(j)\,\beta_i(j)}$$

dove il denominatore garantisce che ogni colonna del vettore γ sia normalizzata, cioè sommi a 1, rendendola una distribuzione stocastica.

Algorithm 2 Backward algorithm

1: Input: $\boldsymbol{A}$, $\psi_{1:N}$, $\boldsymbol{\alpha}$
2: $\beta_N = 1$;
3: **for** $t = N-1:1$ **do**
4: $\quad \beta_t = \text{normalize}(\boldsymbol{A}(\psi_{t+1} \odot \beta_{t+1})$;
5: $\boldsymbol{\gamma} = \text{normalize}(\boldsymbol{\alpha} \odot \boldsymbol{\beta}, 1)$
6: Return $\boldsymbol{\gamma}_{1:N}$

Fig. 3.4

3.3.3 Algoritmo di Viterbi

Per calcolare la sequenza più probabile di stati nascosti (*Problema 2*), utilizziamo l'algoritmo di Viterbi. Questo algoritmo individua il cammino più probabile attraverso il diagramma a traliccio (*trellis*) dell'HMM. Il diagramma a traliccio rappresenta in che modo ciascuno stato del modello a un certo passo temporale si collega agli stati al passo successivo. Anche qui seguiremo la derivazione presentata in Murphy [17].

L'algoritmo di Viterbi prevede una fase forward e una fase backward. Nella fase forward, invece di usare l'algoritmo somma-prodotto, si adotta il metodo max-prodotto. Nella fase backward, si ricostruisce il percorso più probabile attraverso il trellis utilizzando una procedura di *traceback*, propagando indietro nel tempo lo stato più probabile al tempo t per risalire ricorsivamente alla sequenza più probabile nei tempi $1{:}t$.

Questo può essere espresso come:

$$\delta_t(j) \triangleq \max_{Z_1,\ldots,Z_{t-1}} P(Z_{1:t-1}, Z_t = j \mid X_{1:t})$$

Questa probabilità può essere espressa come una combinazione della transizione dallo

stato precedente i al tempo $t-1$ e percorso più probabile che conduce a i .

$$\delta_t(j) = \max_{1 \leq i \leq K} \delta_{t-1}(i) A_{ij} B_{X_t}(j)$$

Dove $B_{X_t}(j)$ rappresenta la probabilità di emissione dell'osservazione X_t dato lo stato j.

È inoltre necessario memorizzare lo stato precedente più probabile, attraverso:

$$a_t(j) = \arg\max_i \delta_{t-1}(i) A_{ij} B_{X_t}(j)$$

La probabilità iniziale si definisce come:

$$\delta_1(j) = \pi_j B_{X_1}(j)$$

Lo stato finale più probabile si determina con:

$$Z_N^* = \arg\max_i \delta_N(i)$$

La sequenza più probabile si può quindi ricostruire tramite backtracking con:

$$Z_t^* = a_{t+1}(Z_{t+1}^*)$$

Per evitare problemi di underflow numerico, si può lavorare nel dominio logaritmico. Questo è uno dei vantaggi principali dell'algoritmo di Viterbi, dato che:

$$\log\max = \max\log \quad \text{ma} \quad \log\sum \neq \sum\log$$

Pertanto, la versione logaritmica della ricorsione diventa:

$$\log\delta_t(j) = \max_i \left(\log\delta_{t-1}(i) + \log A_{ij} + \log B_{X_t}(j)\right)$$

Algorithm 3 Viterbi algorithm

Input: $X_{1:N}$, K, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\pi}$
Initialize: $\delta_1 = \boldsymbol{\pi} \odot B_{X_1}$, $a_1 = \mathbf{0}$;
for $t = 2 : N$ **do**
    for $j = 1 : K$ **do**
        $[a_t(j),\ \delta_t(j)] = \max_i(\log \delta_{t-1}(:) + \log \mathbf{A}_{ij} + \log B_{X_t}(j))$;
$Z_N^* = \arg\max(\delta_N)$;
for $t = N - 1 : 1$ **do**
    $Z_t^* = a_{t+1} Z_{t+1}^*$;
Return $Z_{1:N}^*$

Fig. 3.5

Applicazione degli HMM in ambito finanziario Come gli HMM sono utilizzati per classificare regimi di mercato e motivazione della loro scelta in questa tesi.

3.4 Modelli di Machine Learning

Qui di seguito vengono presentati i principali modelli di apprendimento automatico che verranno impiegati. Ogni modello è stato selezionato in funzione delle sue caratteristiche strutturali e delle sue prestazioni in contesti di previsione su dati temporali e finanziari.

L'obiettivo è quello di sfruttare la diversità tra le tecniche, dalle reti neurali ricorrenti ai modelli ensemble,per adattare il comportamento predittivo alle diverse condizioni di mercato, precedentemente individuate tramite il modello HMM.

Per ciascun modello verranno descritte le logiche di funzionamento, i punti di forza e le potenziali limitazioni, così da motivarne l'integrazione nella pipeline complessiva.

3.4.1 Supervised learning: Alberi decisionali

Gli alberi decisionali sono modelli di ML che rappresentano decisioni e relative conseguenze sotto forma di struttura ad albero. Sono ampiamente utilizzati per compiti di classificazione, regressione e previsione. Per una maggiore comprensione,supponiamo di avere una tabella di grandi dimensioni contenente *N* caratteristiche, e di voler ottenere informazioni sulle persone rappresentate da ciascuna riga. Immaginiamo che ad ogni individuo sia associata un'etichetta del tipo:

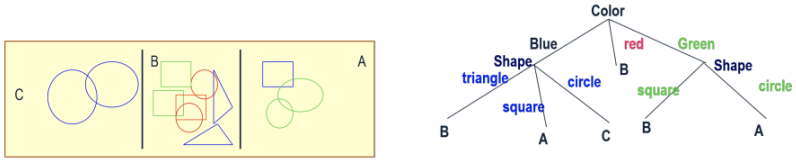
- Possiede un'auto costosa
- Non possiede un'auto costosa

Un modo semplice per sintetizzare queste informazioni consiste nell'utilizzare un istogramma basato sull'attributo "possiede". In un secondo momento, possiamo aggiungere un' altra caratteristica, ad esempio il genere, trasformando così l'istogramma in 4 voci distinte: (possiede & uomo), (possiede & donna), (non possiede & uomo), (non possiede & donna). Se ci sono in totale 16 qualità e si considerano solo istogrammi unidimensionali, si ottengono 16 voci nell'istogramma. In modo analogo, un istogramma bidimensionale conterà tutte le possibili combinazioni a coppie tra le caratteristiche. Un istogramma bidimensionale richiede 120 numeri, mentre uno tridimensionale ne richiede 560. Man mano che il numero di caratteristiche aumenta, il problema diventa rapidamente intrattabile. È quindi fondamentale trovare un modo per rappresentare i dati in maniera più significativa, in modo da poter individuare i punti più importanti.

Dunque, formalmente, un *albero decisionale* è una struttura dati gerarchica che organizza le informazioni seguendo una strategia di tipo. Qui a titolo di esempio ci focalizziamo sugli alberi decisionali con etichette categoriche, sebbene possano essere impiegati anche per classificazioni non parametriche e problemi di regressione. Nel problema

di classificazione, l'obiettivo è costruire un albero che rifletta la struttura dei dati di addestramento, consentendo così di predire le etichette di nuovi esempi.

Gli alberi decisionali operano su istanze descritte da vettori di attributi (es. colore=?, forma=?, etichetta=?). I nodi interni effettuano test sulle caratteristiche, mentre le foglie rappresentano l'etichetta assegnata. A ciascun nodo corrisponde un ramo per ogni possibile valore dell'attributo.



(a) Example Data (b) Decision Tree

Fig. 3.6: Esempio di albero decisionale

Come mostrato nell'esempio illustrato nella Figura 3.6 [20], possiamo utilizzare l'albero per determinare l'etichetta di una nuova forma sulla base delle sue caratteristiche.

Diviene naturale a questo punto chiedersi, quale sia una delle principali metodi utilizzati per decidere su quale attributo effettuare la divisione nei modelli ad albero è il cosiddetto *information gain*.

Per calcolare l'*information gain*, è necessario prima definire l'entropia: una misura di incertezza associata a una distribuzione di probabilità. L'entropia di un insieme di esempi S, rispetto a una classificazione binaria, è definita come:

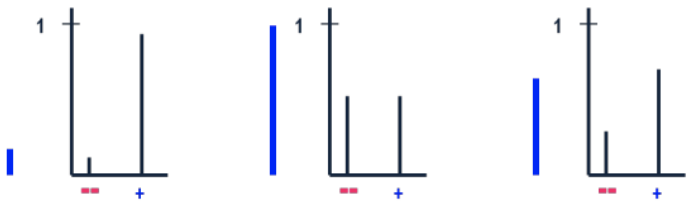
$$Entropy(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

dove p_+ è la proporzione di esempi positivi e p_- quella di esempi negativi all'interno di S.

Se tutti gli esempi appartengono alla stessa classe, l'entropia è nulla. Se invece le classi sono equamente distribuite, l'entropia raggiunge il valore massimo pari a 1. In generale, l'entropia misura quanto è incerta la classificazione.

Nel caso generale, con k classi e p_i proporzione degli esempi etichettati con la classe i, l'entropia si scrive:

$$Entropy(\{p_1, p_2, \ldots, p_k\}) = -\sum_{i=1}^{k} p_i \log(p_i)$$



(a) Entropy Intuition

Fig. 3.7: Intuizione grafica dell'entropia: le barre nere rappresentano le proporzioni delle classi; le barre blu l'entropia

Con questo concetto, possiamo definire il *guadagno informativo* (information gain) come la riduzione attesa di entropia che si ottiene suddividendo il dataset S in base a un attributo a:

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

dove S_v è il sottoinsieme di S in cui l'attributo a assume il valore v. L'entropia totale è data dalla somma pesata delle entropie dei sottoinsiemi.

Per quantificare l'information gain, un'alternativa è data dall'indice di Gini, definito come:

$$GiniIndex = 1 - \sum_{i=1}^{n} p_i^2$$

Questo indice viene utilizzato allo stesso modo dell'entropia. Calcolando il suo valore atteso è possibile calcolare anche l'information gain, essendo la riduzione del valore attesso dell'indice di Gini. Il più delle volte entropia e indice di Gini danno origine ad alberi decisionali simili.

Gli alberi decisionali rappresentano quindi, una delle tecniche di ML più semplici e intuitive. Questo rende il modello facilmente interpretabile, anche da chi non ha una formazione tecnica, e lo rende utile in contesti in cui la trasparenza del processo decisionale è importante. Un ulteriore vantaggio significativo è la capacità degli alberi decisionali di lavorare senza richiedere un'elevata preparazione dei dati: non è necessario normalizzare le variabili, né trasformare le *feature* categoriche. Inoltre, l'albero è in grado di individuare autonomamente le variabili più rilevanti per la previsione, riducendo il carico sul processo di *feature selection*. Tuttavia, gli alberi decisionali presentano anche alcuni limiti. Il principale è la tendenza all'*overfitting*: un albero molto profondo tende a modellare anche il rumore presente nei dati di addestramento, perdendo capacità di generalizzazione sui dati nuovi. Inoltre, sono modelli piuttosto instabili: piccole variazioni nei dati possono portare a strutture d'albero completamente diverse. Le performance di un singolo albero, infine, raramente competono con quelle di metodi più avanzati, come i modelli *ensemble* basati su foreste o boosting. Per questi motivi, sebbene un singolo albero decisionale possa rappresentare un buon punto di partenza per esplorare i dati e costruire un primo modello interpretabile, nella pratica è spesso preferibile integrarlo in architetture più robuste, come quelle basate su *bagging* (es. *Random Forest*) o *boosting* (es. *XGBoost*), che ne migliorano sensibilmente l'accuratezza e la stabilità.

3.4.2 XGBoost

In questa sezione si descrive il modello *XGBoost*, un sistema scalabile di machine learning per il boosting ad albero, proposto dallo straordinario lavoro *"XGBoost: A Scalable Tree Boosting System"* di Chen, Tianqi e Guestrin, Carlos (2016) [6]. Gli autori propongono un algoritmo di apprendimento supervisionato basato sul boosting di alberi decisionali. Si tratta di un'evoluzione del metodo del gradient boosting, che costruisce un modello come combinazione sequenziale di alberi di decisione, ognuno dei quali cerca di correggere gli errori commessi dal modello fino a quel punto.

L'impatto di questo sistema è stato ampiamente riconosciuto in numerose sfide di apprendimento automatico e data mining. Il fattore principale che ha reso XGBoost così efficace è la sua applicabilità in tutti gli scenari. Il sistema è oltre dieci volte più veloce delle soluzioni più popolari su una singola macchina, ed è in grado di scalare a miliardi di esempi sia in ambienti distribuiti sia in contesti con memoria limitata. Tale scalabilità è dovuta a numerosi accorgimenti sistemici e ottimizzazioni algoritmiche.

L'inclusione di questo modello all'interno della strategia non è casuale, ma motivata da una combinazione di ragioni teoriche, pratiche ed empiriche che ne rendono l'adozione particolarmente coerente con gli obiettivi della stessa. In primo luogo, XGBoost è riconosciuto per l'elevata efficacia predittiva in contesti caratterizzati da rumore, non linearità e interazioni complesse tra le variabili. Le serie temporali finanziarie, come noto, presentano proprio queste caratteristiche. Inoltre, grazie alla sua capacità di costruire alberi in sequenza ottimizzando la funzione di perdita a ogni iterazione, si adatta molto bene a questi contesti. L'algoritmo include meccanismi di regolarizzazione (L1 e L2) che aiutano a ridurre l'overfitting, un problema particolarmente delicato nella modellazione finanziaria. Dal punto di vista operativo, XGBoost si distingue anche per l'efficienza computazionale. È in grado di effettuare training e predizioni molto velocemente, il che lo rende adatto all'utilizzo. Nonostante l'efficienza, non sacrifica la

flessibilità: è in grado di apprendere pattern anche complessi senza la necessità di lunghe fasi di tuning o architetture sofisticate come nel deep learning. Un altro elemento rilevante è la sua buona interpretabilità. A differenza di modelli più "opachi" come le reti neurali, XGBoost permette l'analisi dell'importanza delle variabili, fornendo quindi informazioni preziose anche per un'analisi qualitativa dei segnali generati. Questo aspetto risulta particolarmente utile in ambito accademico, dove è spesso necessario spiegare il comportamento dei modelli e motivarne l'impiego.

Il modello predittivo di XGBoost è definito come somma di K alberi:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathscr{F}$$

dove ogni f_k è un albero di regressione e $\mathscr{F}$ rappresenta lo spazio delle possibili strutture ad albero. A differenza di altri modelli simili, XGBoost non si limita a minimizzare una funzione di perdita, ma include un termine di regolarizzazione per controllare la complessità degli alberi:

$$\mathscr{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \quad \text{dove} \quad \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

In questa espressione, T è il numero di foglie dell'albero, w rappresenta i pesi associati alle foglie, mentre λ e γ sono parametri di regolarizzazione. Questo approccio favorisce la costruzione di modelli semplici e ben generalizzabili.

L'addestramento avviene in modo incrementale: ad ogni iterazione si aggiunge un nuovo albero f_t che riduce ulteriormente l'errore del modello. Per rendere questa procedura efficiente, si utilizza un'approssimazione di secondo ordine della funzione obiettivo:

$$\mathscr{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

dove g_i e h_i sono rispettivamente il gradiente e l'hessiano della funzione di perdita rispetto alla previsione corrente $\hat{y}_i^{(t-1)}$. Ogni foglia dell'albero ha un peso ottimale calcolato

come:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

e il guadagno prodotto da uno split viene valutato con:

$$\text{Gain} = \frac{1}{2}\left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right) - \gamma$$

dove G_L, G_R, H_L, e H_R rappresentano le somme dei gradienti e degli hessiani per i sottoinsiemi a sinistra e a destra dello split.

Per migliorare ulteriormente la scalabilità, XGBoost introduce due strategie chiave: uno *sketch quantilico pesato*, che permette di determinare soglie di split approssimate senza dover considerare ogni valore possibile, e un algoritmo *sparsity-aware*, che consente di gestire efficacemente i valori mancanti e le rappresentazioni sparse (come le codifiche one-hot). In presenza di valori mancanti, ad esempio, XGBoost assegna una direzione predefinita di split che viene appresa automaticamente dai dati stessi. L'efficienza di XGBoost non deriva solo dalle scelte algoritmiche, ma anche da una progettazione attenta del sistema. I dati vengono memorizzati in blocchi ordinati per colonna (*column blocks*), permettendo una scansione lineare durante la ricerca degli split.

XGBoost si è rapidamente affermato come uno degli strumenti più efficaci e versatili nel machine learning supervisionato. I suoi principali pregi includono:

- *Alta accuratezza*: grazie all'uso di gradienti e hessiani, regolarizzazione e tecniche di subsampling, XGBoost riesce spesso a superare altri algoritmi in termini di performance.
- *Efficienza computazionale*: la combinazione di ottimizzazioni software e parallelizzazione lo rende estremamente rapido anche su grandi dataset.
- *Scalabilità*: supporta l'elaborazione distribuita e l'apprendimento da disco, permettendo di gestire dataset con miliardi di esempi.

- *Flessibilità*: è compatibile con una vasta gamma di compiti, come classificazione, regressione e ranking, e consente anche la definizione di funzioni di perdita personalizzate.
- *Robustezza*: è in grado di gestire dati con valori mancanti o altamente sparsi, adattandosi bene a problemi reali.

In definitiva, XGBoost rappresenta una sintesi ben riuscita tra potenza predittiva, efficienza e adattabilità, risultando una scelta particolarmente apprezzata sia in ambito accademico che industriale.

3.4.3 Reti neurali artificiali (ANN)

Le reti neurali artificiali rappresentano un algoritmo ispirato, in origine, al funzionamento del cervello umano, in cui innumerevoli neuroni trasmettono segnali tra loro. Vengono dette "neurali" proprio perché nate come astrazione semplificata del neurone biologico.

Nell'uso moderno, tuttavia, le ANN hanno perso il legame diretto con le neuroscienze, assumendo una natura puramente computazionale: una rete di piccole unità di calcolo, ciascuna delle quali riceve un vettore di input, elabora i valori e produce un output tramite funzioni matematiche. Le applicazioni attuali spaziano dall'ottimizzazione alla compressione delle immagini, dal riconoscimento dei caratteri fino alla previsione dei mercati finanziari.

In sostanza, una rete neurale può essere vista come un grafo diretto con pesi associati agli archi. Il modello più semplice è la rete *feed-forward*, in cui i neuroni sono organizzati a strati e i segnali fluiscono sempre in avanti, senza retroazioni. Ogni livello intermedio elabora i dati provenienti dal livello precedente fino ad arrivare a un output finale reale. Lo strato di ingresso e quello di uscita sono composti da un solo livel-

lo, mentre gli strati nascosti possono essere uno o più a seconda della complessità del problema.

Le unità computazionali applicano funzioni matematiche agli input pesati. Un nodo si attiva quando la combinazione lineare degli input supera una soglia, determinata dalla funzione di attivazione scelta. Una delle più comuni è la *sigmoide*, che mappa l'input in un valore compreso tra 0 e 1:

$$y = \sigma(z) = \frac{1}{1+e^{-z}}$$

Un'altra funzione di attivazione, oggi tra le più diffuse nelle reti profonde, è la *ReLU* (Rectified Linear Unit), definita come:

$$f(x) = x^{+} = \max(0, x)$$

Le ReLU hanno dimostrato di favorire l'addestramento di reti profonde, rendendo più efficiente la propagazione del gradiente.

Durante l'allenamento di una rete neurale artificiale (ANN), i parametri, rappresentati dai pesi associati agli archi e dai termini di bias, vengono ottimizzati in maniera iterativa per ciascun livello della rete, con l'obiettivo di produrre un output quanto più vicino possibile al valore reale. Il processo di ottimizzazione si basa su metodologie simili a quelle impiegate nella regressione logistica binaria, come la *discesa del gradiente* applicata a una funzione di perdita.

All'interno della rete, gli output di ciascun livello vengono calcolati tramite la *funzione di propagazione*, dove w rappresenta i pesi e x gli input. La formulazione matematica è la seguente:

Sia h_i la funzione di propagazione, x_i gli input e w_{ij} i pesi associati agli archi. Allora:

$$h_i = \sum_{j=0}^{n} x_i w_{ij}, \quad i = 1,2,3,\ldots,m$$

L'attivazione di ciascun nodo si ottiene applicando una funzione di attivazione f_h:

$$z_i = f_h(h_i)$$

Ad esempio:

$$f_h = \frac{1}{1+e^{-x}}, \quad \text{se sigmoid}$$

$$f_h = \max(0,x), \quad \text{se ReLU}$$

Infine, l'output complessivo della rete neurale artificiale al tempo t può essere espresso come:

$$y_t = \sum_{i=0}^{m} z_i w_{ij}$$

3.4.4 Long-Short Term Memory (LSTM)

Una rete neurale LSTM fa parte alla categoria delle *reti neurali ricorrenti* (RNN), una variante delle reti neurali artificiali che include almeno un ciclo all'interno delle proprie connessioni. Ciò significa che il valore di un'unità computazionale non dipende solo dagli input esterni, ma anche, direttamente o indirettamente, dal proprio output passato, che viene riutilizzato come input.

Le RNN si distinguono quindi dalle reti feed-forward descritte in precedenza in quanto sono in grado di sfruttare uno *stato interno*, o memoria, per elaborare sequenze di input, nelle quali l'ordine temporale e il contesto rivestono un ruolo fondamentale.

Questa caratteristica le rende particolarmente adatte all'analisi di serie temporali finanziarie, dove i movimenti precedenti di un asset possono fornire informazioni cruciali, ad esempio, riguardo alla volatilità futura.

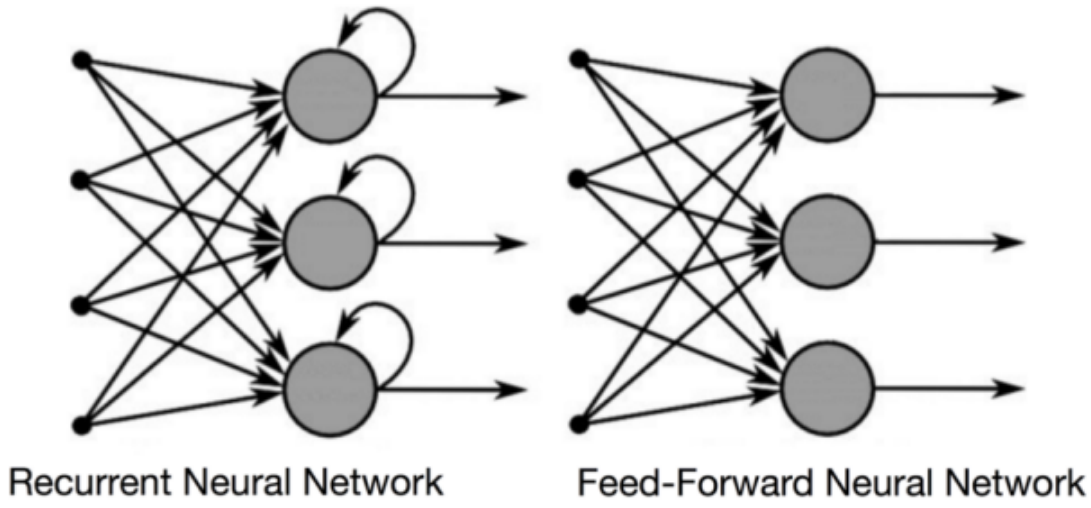


Fig. 3.8: ANN e RNN

La Figura 3.8 mostra in maniera semplificata la differenza tra una rete feed-forward e una rete ricorrente.

Il termine *Long Short-Term Memory* (LSTM) si riferisce a un'architettura di rete neurale ricorrente sviluppata per superare i problemi di *exploding* e *vanishing gradient*, che ostacolano l'addestramento delle RNN tradizionali.

In breve, questi problemi emergono a causa delle *dipendenze a lungo termine*: una cella deve "ricordare" informazioni per periodi estesi. Durante l'addestramento con metodi basati sul gradiente e backpropagation, il gradiente può diventare estremamente piccolo, impedendo agli aggiornamenti dei pesi di modificarsi, oppure crescere verso

l'infinito, destabilizzando la rete. Ciò si aggrava poiché i calcoli sono eseguiti con numeri a precisione finita.

Le LSTM affrontano parzialmente questo problema consentendo ai gradienti di fluire più agevolmente attraverso la rete, preservando l'informazione anche su orizzonti temporali lunghi.

Sebbene esistano diverse varianti dell'architettura standard, un'unità LSTM tipica è composta da *una cella*, una *input gate*, una *output gate* e una *forget gate*, come espongono Bergstrom e Hjelm nel loro lavoro *"Impact of Time Steps on Stock Market Prediction with LSTM"*[3].

La Figura 3.9 illustra una versione semplificata del modulo ricorrente in un'LSTM, dove ciascun modulo contiene quattro livelli interattivi, invece di un singolo livello (funzione di attivazione) come nelle RNN classiche.

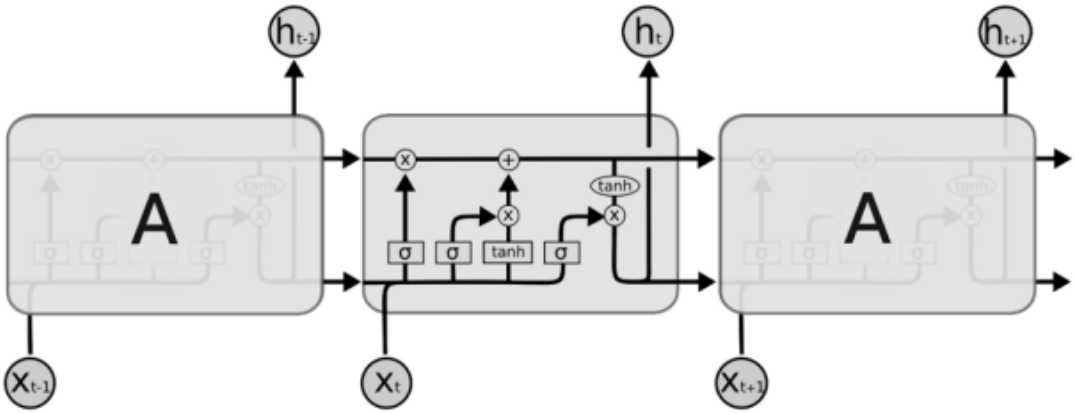


Fig. 3.9: Struttura di una LSTM

La linea orizzontale che attraversa la parte superiore della figura rappresenta lo *stato della cella*, che viene modificato dai diversi strati della cella LSTM.

Il primo strato, costituito da una funzione sigmoide, è comunemente chiamato *"forget gate layer"*. Questo livello analizza sia l'input esterno, sia l'output proveniente dal mo-

dulo precedente nella catena di moduli ricorrenti, producendo come risultato un valore compreso tra 0 e 1. Tale valore determina la quantità di informazione dello stato precedente che deve essere conservata o dimenticata. La funzione sigmoide della *forget gate* decide quali valori dello stato della cella devono essere mantenuti e quali eliminati, restituendo valori compresi tra 0 e 1 per ciascun elemento:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Il passo successivo consiste nel determinare quali nuove informazioni devono essere aggiunte allo stato della cella. La *input gate layer*, anch'essa basata su una sigmoide, stabilisce quali valori aggiornare:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

In parallelo, viene generato un vettore candidato tramite una funzione *tanh*, che contiene i nuovi valori potenzialmente inseribili nello stato della cella:

$$C_{new} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Combinando queste due componenti, lo stato della cella viene aggiornato come segue:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C_{new}$$

Infine, la *output gate* determina quali parti dello stato della cella devono essere restituite come output. Anche in questo caso viene utilizzata una sigmoide:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Lo stato aggiornato viene quindi trasformato tramite una funzione tanh (o ReLU, a seconda delle implementazioni) e moltiplicato per l'output della sigmoide, ottenendo lo stato nascosto h_t:

$$h_t = o_t \cdot \tanh(C_t)$$

Il valore h_t diventa l'output della cella LSTM, che viene poi passato al modulo successivo nella sequenza e al livello seguente dell'architettura.

In sintesi, l'architettura LSTM consente di superare le principali limitazioni delle reti neurali ricorrenti tradizionali, offrendo una struttura in grado di mantenere e aggiornare in modo selettivo le informazioni nel tempo. Questa capacità risulta particolarmente efficace nel contesto delle serie finanziarie, dove la presenza di pattern sequenziali e la necessità di catturare dipendenze temporali complesse rendono l'LSTM uno strumento potente per la classificazione dei regimi di mercato.

Capitolo 4

Modellazione e Strategia di Previsione dei Regimi

4.1 Introduzione

L'obiettivo principale di questo capitolo è descrivere in modo dettagliato l'implementazione pratica dei modelli sviluppati per il riconoscimento e la previsione dei regimi di mercato. Dopo aver illustrato le basi teoriche che motivano l'utilizzo di approcci come l'HMM e i modelli supervisionati, si passa ora alla fase in cui queste idee vengono tradotte in codice e testate empiricamente.

Tutto il lavoro è stato condotto in Python (si veda **Appendice A Codice Python**), utilizzando una combinazione di librerie ben consolidate come pandas, numpy, scikit-learn, xgboost e tensorflow. L'ambiente di sviluppo scelto è Jupyter Notebook, ideale per lavorare in modo iterativo, mantenere sotto controllo i dati e visualizzare rapidamente i risultati. Questo approccio ha consentito di costruire la pipeline passo dopo passo, testando ogni blocco e valutando i risultati a ogni stadio.

Il punto di partenza è stato quello di definire una serie di feature significative in grado di descrivere lo stato del mercato in ciascun istante temporale. Queste variabili, di natura tecnica e in parte anche macroeconomica, sono state selezionate tenendo conto della letteratura e della loro interpretabilità. Una volta calcolati questi indicatori, si è passati alla costruzione di un'etichetta che rappresentasse il regime di mercato, cioè la fase in cui si trovava l'indice in quel momento: bullish, bearish o neutrale.

Per costruire le etichette sono stati utilizzati due approcci distinti. Il primo, più sofisticato, si basa su un Hidden Markov Model (HMM) (vedi al 3.3) , applicato a variabili macro e tecniche per identificare in maniera automatica regimi latenti nel tempo. Le etichette così ottenute sono poi state "smussate" per aumentarne la stabilità temporale. Il secondo approccio, più semplice, è stato costruito manualmente applicando soglie ai ritorni logaritmici: un modo per creare un benchmark che potesse servire da confronto nei modelli successivi.

A questo punto si giunge all'addestramento dei modelli predittivi. Sono stati implementati due tipi di modelli: uno di tipo gradient boosting (XGBoost 3.4.2) e uno basato su reti neurali ricorrenti, nello specifico una LSTM (Long Short-Term Memory 3.4.4). Entrambi sono stati addestrati in due varianti: una utilizzando le etichette fornite dal modello HMM, e una seconda utilizzando le etichette naive.

Ogni modello è stato validato out-of-sample. Questo significa che l'intero dataset è stato suddiviso in una parte di training e una parte di test, e tutte le valutazioni sono state effettuate su dati che il modello non aveva mai visto in fase di addestramento. Inoltre, per evitare qualsiasi forma di lookahead bias, è stato utilizzato uno shift(1) che garantisce che le decisioni operative si basino solo su informazioni effettivamente disponibili al momento in cui vengono prese.

Una volta ottenute le predizioni, il passo successivo è stato quello di valutarne l'efficacia sia dal punto di vista della classificazione, sia da quello dell'impatto che avrebbero

prodotto se utilizzate in una strategia operativa. Per la parte classificativa sono state analizzate metriche standard come accuracy, precision, recall e F1-score, che misurano la capacità del modello di assegnare correttamente ciascun regime. Dal punto di vista operativo, è stata invece simulata una strategia di investimento basata sui regimi previsti, con posizioni long in fasi bullish, short in fasi bearish e flat in fasi neutre. I risultati ottenuti sono stati valutati attraverso metriche finanziarie classiche, come il Sharpe Ratio, il Maximum Drawdown, l'Ulcer Index, il Gain-to-Pain Ratio e il rendimento cumulato della strategia.

Il codice che segue racconta questo processo. Ogni blocco è stato progettato per essere chiaro, riproducibile e il più possibile aderente alla realtà operativa. Non si tratta solo di un esercizio accademico, ma di una simulazione credibile di come approcci quantitativi possano essere utilizzati per costruire modelli decisionali in ambito finanziario.

4.2 Preprocessing e Costruzione delle Features

Il primo passaggio fondamentale del lavoro consiste nell'acquisizione dei dati storici e nella costruzione di un set di variabili adatto alla modellizzazione dei regimi di mercato. In questo studio, si è scelto di utilizzare dati finanziari scaricati tramite il modulo yfinance di Python, che consente di accedere facilmente a serie storiche fornite da Yahoo Finance. In particolare, sono stati scaricati i prezzi giornalieri del benchmark di riferimento (Russell 3000), insieme ad alcune variabili macro e finanziarie rilevanti: volatilità implicita (VIX), oro (GC) e petrolio (CL).

Periodo di riferimento

I dati sono stati raccolti su un orizzonte temporale ampio, dal 1° gennaio 1995 fino al 1° gennaio 2025, per garantire che i modelli abbiano accesso a una quantità di osservazioni

sufficiente a identificare pattern ricorrenti e a testare la loro capacità di generalizzazione in diversi cicli di mercato.

Preprocessing dei dati grezzi

Dopo l'importazione, le serie temporali vengono allineate per data, e viene costruito un DataFrame principale contenente le variabili fondamentali: prezzi di apertura, massimo, minimo, chiusura e volumi. Tutte le feature vengono poi sincronizzate sull'indice temporale della serie principale (il benchmark), in modo da evitare disallineamenti e garantire una corrispondenza perfetta tra i diversi input temporali.

Costruzione delle feature

Una delle fasi più importanti dell'intero processo consiste nella costruzione delle variabili esplicative. Queste feature hanno lo scopo di catturare dinamiche di mercato che possano essere informative per la classificazione dei regimi di mercato. Sono state implementate più di 20 variabili, che si possono raggruppare nelle seguenti categorie:

Volatilità e rendimenti:

`LogRet`: rendimento logaritmico giornaliero.

`Vol20`: deviazione standard annualizzata sui rendimenti a 20 giorni.

`EWMA_vol_20`: volatilità esponenziale smussata a 20 giorni.

Indicatori di momentum e trend:

`Momentum10`: momentum a 10 giorni.

`MA10`, `EWMA10`: medie mobili semplici ed esponenziali.

`DonchianHigh20`, `DonchianLow20`: bande di breakout.

PctAbove50: percentuale di giorni in cui il prezzo ha chiuso sopra la media mobile a 50 giorni.

Indicatori tecnici classici:

RSI14: indice di forza relativa a 14 giorni.

CCI20: commodity channel index su 20 periodi.

ADX14: average directional index.

ATR14, ATR_normalized: true range e sua versione normalizzata.

BollingerWidth20: ampiezza delle bande di Bollinger.

Comportamento di mercato e sentiment:

VIX, VIX_change_pct: livello della volatilità implicita e sue variazioni.

Gold, Oil, GoldOil, Oil_change_pct: prezzi dell'oro e del petrolio, e loro dinamica.

Mass_Index: misura di espansione e compressione dei prezzi.

Altre trasformazioni:

ZLogRet20: z-score dei rendimenti logaritmici a 20 giorni.

Pct_above_50MA: % di giorni in cui il prezzo chiude sopra la media mobile a 50 giorni.

Return_1M, Return_3M: variazioni percentuali su base mensile e trimestrale.

Pulizia finale del dataset

Per evitare che la presenza di valori nulli (NaN) possa compromettere le analisi successive, tutte le righe contenenti dati mancanti vengono eliminate tramite il metodo dropna(). Inoltre, viene mostrata un'anteprima del dataset finale per verificarne la correttezza.

Analisi

Per offrire una panoramica preliminare sul contesto economico-finanziario di riferimento, sono stati analizzati alcuni indicatori di base relativi al mercato azionario (Russell 3000), alle materie prime (oro e petrolio), ai rendimenti e alla volatilità. I prezzi sono accompagnati dalle medie mobili a 50 e 200 giorni, utili a evidenziare trend di fondo e inversioni cicliche. La normalizzazione degli asset (base 100) consente un confronto diretto delle performance relative tra equity e commodities.

Inoltre, l'analisi dei rendimenti logaritmici standardizzati (*Z-LogRet*) 4.3 evidenzia la dinamica giornaliera dei movimenti di prezzo, mentre la volatilità rolling a 20 giorni 4.4 rappresenta un indicatore chiave per identificare periodi di stress di mercato. Queste variabili, oltre ad avere un significato finanziario autonomo, saranno successivamente utilizzate come input per i modelli di regime switching e classificazione, contribuendo alla costruzione di una base informativa robusta e interpretabile.

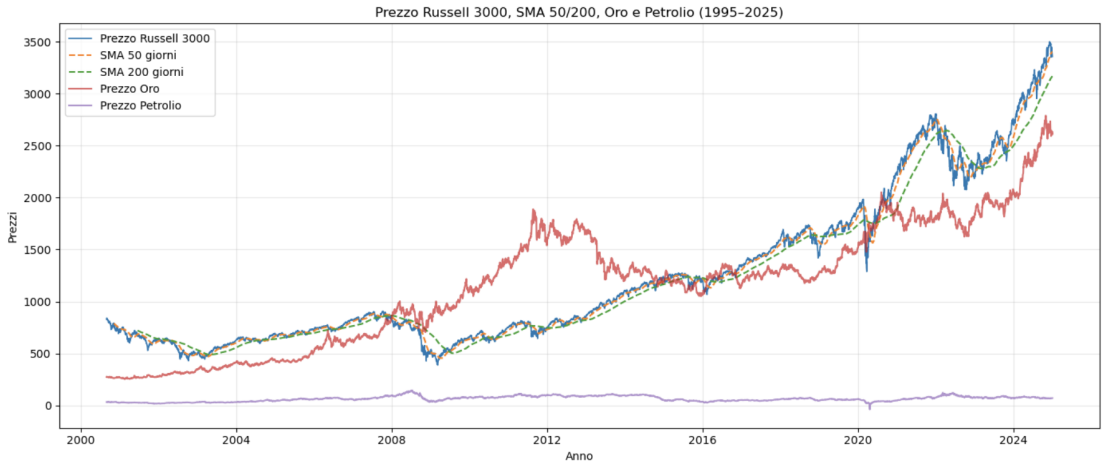


Fig. 4.1: Prezzo storico del Russell 3000 con medie mobili a 50 e 200 giorni, affiancato ai prezzi dell'oro e del petrolio.

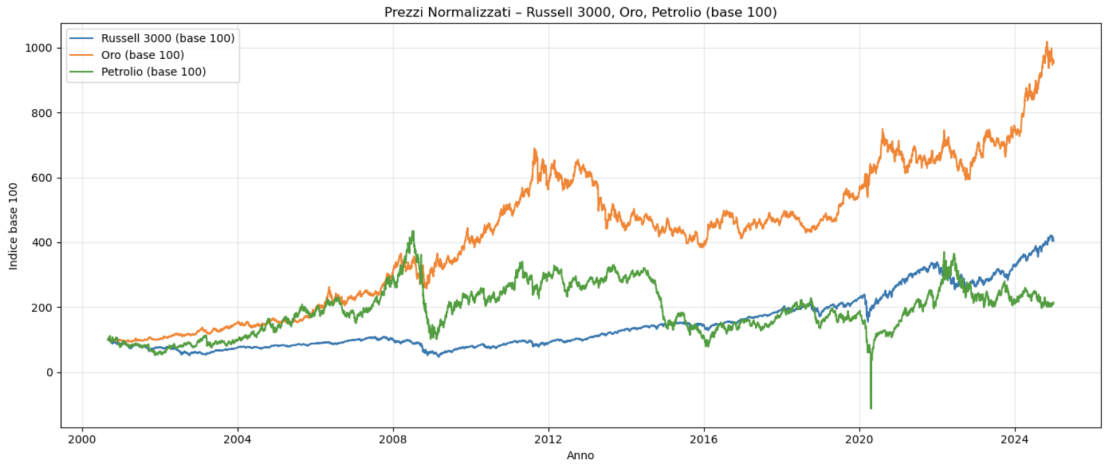


Fig. 4.2: Prezzi di Russell 3000, oro e petrolio su scala normalizzata (base 100) per confronto diretto tra asset.

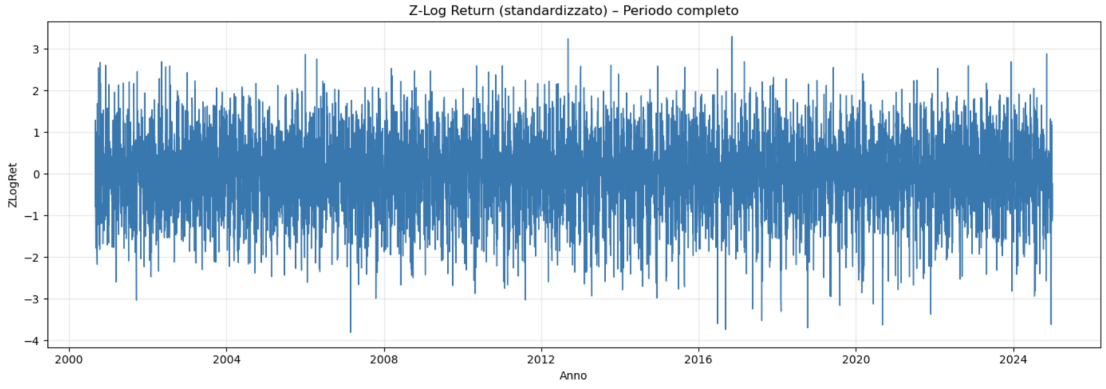


Fig. 4.3: Rendimenti logaritmici standardizzati (*Z-LogRet*) dell'indice Russell 3000.

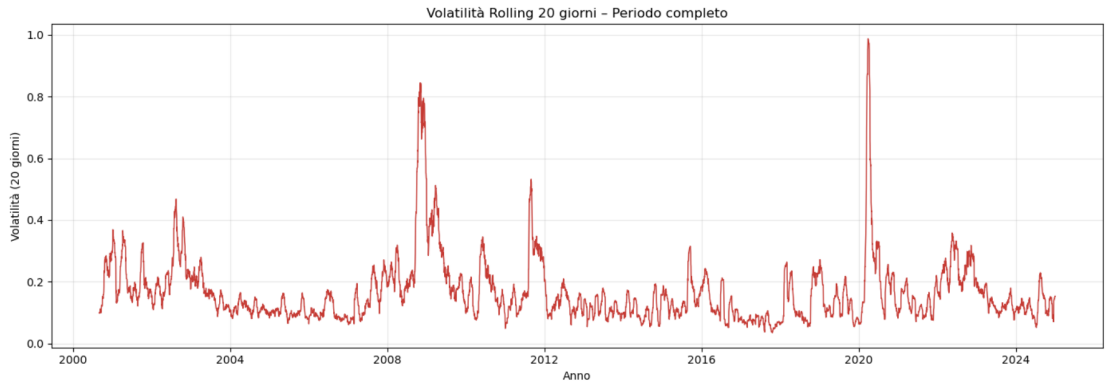


Fig. 4.4: Volatilità rolling a 20 giorni sull'indice Russell 3000, utile per l'analisi dei regimi di mercato.

4.3 Modello Hidden Markov a Finestra Mobile (Rolling HMM)

Una volta puliti e standardizzati i dati, è stato implementato un modello di Hidden Markov Model (HMM) per identificare regimi di mercato nascosti sulla base delle feature selezionate. Il modello è stato addestrato con un approccio rolling, ovvero su una finestra mobile di osservazioni, per riflettere meglio l'evoluzione temporale dei regimi e mantenere la natura out-of-sample della previsione. Questo approccio segue le indicazioni di Johns Hopkins University (2021) [29], che suggerisce l'uso di un HMM ricalibrato periodicamente per adattarsi a dinamiche di mercato non stazionarie.

Nello specifico, è stata definita una finestra mobile di 250 giorni, corrispondente a circa un anno di dati di mercato, con un refitting del modello ogni 20 giorni. A ogni passo, il modello è addestrato sui dati più recenti e produce una previsione per il giorno successivo. Questo setup consente di costruire una serie temporale di regimi stimati, mantenendo la separazione tra fase di training e fase di predizione.

Per la modellazione, si è optato per un Gaussian HMM, che presuppone che le osservazioni in ciascun stato siano distribuite secondo una distribuzione normale multivariata. Inoltre, è stata scelta una covarianza diagonale (`covariance_type='diag'`) per limitare la complessità del modello, come suggerito da Cornell University (2021)[28], il cui studio ha evidenziato come questa configurazione sia particolarmente efficace nel contesto finanziario con feature standardizzate.

Il numero di regimi è stato fissato a 3, un valore ampiamente adottato in letteratura per rappresentare i tre classici stati di mercato: rialzista (bull), neutrale e ribassista (bear). Una volta generata la sequenza di regimi previsti, è stato applicato un processo di smoothing, calcolando la moda (frequenza più alta) su una finestra centrata di 5 giorni. Questa operazione serve a ridurre la variabilità e la frequenza dei cambi di stato, mitigando i segnali erratici. Il procedimento è ispirato all'approccio di NYU Tandon (2020) [26], che propone una trasformazione post-modello per garantire maggiore robustezza operativa ai segnali generati.

La fase successiva ha previsto l'assegnazione ex post di significato economico ai regimi stimati, ordinandoli sulla base del rendimento medio e della volatilità. In questo modo, il regime con rendimento più elevato e volatilità contenuta è stato interpretato come "bullish", quello intermedio come "neutral" e quello con performance negativa e alta volatilità come "bearish". Tale approccio, ripreso da UC Berkeley Haas (2021)[27], consente di collegare i regimi latenti dell'HMM a condizioni di mercato osservabili e interpretabili, senza vincolare il modello in fase di training a etichette predefinite.

Infine, sono state calcolate alcune statistiche descrittive per ciascun regime, come il numero di giorni, il rendimento medio, la volatilità e l'indice di Sharpe, al fine di validare empiricamente la coerenza dei cluster. Queste statistiche permettono anche di valutare l'eventuale utilità dei regimi come segnale operativo per strategie di trading o per l'integrazione in modelli previsivi più complessi, come XGBoost o LSTM, trattati nei

capitoli successivi.

4.3.1 Analisi degli output del modello HMM

Una volta implementato il modello Hidden Markov con approccio rolling, è stata costruita una serie temporale dei regimi di mercato previsti. Le informazioni sui regimi sono memorizzate nelle colonne `Regime` e `Regime_Smooth` del DataFrame principale. La prima rappresenta la sequenza predetta dal modello giorno per giorno, mentre la seconda è una versione filtrata tramite smoothing, utile per addestrare successivamente modelli supervisionati.

Il modello HMM utilizzato è del tipo Gaussian HMM con 3 stati latenti, addestrato in modalità out-of-sample rolling con una finestra mobile di 250 giorni e aggiornamenti ogni 20 giorni. A ogni passo, il modello viene riaddestrato sulla finestra corrente e utilizzato per predire lo stato latente del giorno successivo, evitando problemi di lookahead. Questo schema consente di simulare fedelmente l'operatività in tempo reale. Poiché l'HMM è un modello *non supervisionato*, le etichette numeriche (0, 1, 2) associate agli stati latenti non hanno un significato economico diretto. Per attribuire una semantica ai regimi, si è adottata una *strategia ex post*, basata sulle statistiche descrittive calcolate per ciascun regime:

- Numero di osservazioni (*frequenza*)
- Rendimento medio
- Volatilità (deviazione standard dei log-return)
- Sharpe Ratio (rendimento medio / volatilità)

L'assegnazione è avvenuta ordinando i regimi in base alla performance e alla stabilità:

- Il regime con il rendimento medio più elevato e la volatilità più contenuta è stato etichettato come *bullish*.
- Il regime con rendimento negativo e volatilità elevata è stato classificato come *bearish*.
- Il regime intermedio, caratterizzato da performance moderate e bassa varianza, è stato associato a una fase *neutrale o laterale*.

Analisi degli output

Il codice implementa il rolling fit e produce automaticamente le statistiche riassuntive di ciascun regime. L'output finale mostra il numero di osservazioni per stato, il rendimento medio giornaliero, la deviazione standard e lo Sharpe ratio.

L'output è riportato in Tabella 4.1:

Tabella 4.1: Statistiche descrittive per ciascun regime HMM

Regime	**Num Giorni**	**Rendimento Medio**	**Volatilità**	**Sharpe Ratio**
0	1002	0.0004	0.0144	0.0261
1	2083	0.0001	0.0119	0.0065
2	2738	0.0004	0.0118	0.0335

Dall'analisi della tabella emerge che il regime 2 rappresenta la configurazione di mercato più favorevole in termini di rapporto rischio/rendimento: pur mantenendo un livello di volatilità contenuto (1.18%), riesce a offrire un rendimento medio relativamente elevato, con il miglior Sharpe Ratio tra i tre regimi. Questo tipo di comportamento è tipico di fasi economiche espansive o comunque stabili, in cui l'ottimismo degli investitori e la crescita economica favoriscono l'apprezzamento dei mercati azionari, con una volatilità sotto controllo. Nei dati considerati, che coprono un ampio arco temporale, tali fasi

possono essere associate a periodi post-crisi come il ciclo di crescita seguito al 2009, o la lunga fase rialzista tra il 2016 e il 2019.

Il regime 1, al contrario, si presenta come una fase di mercato piatta o incerta: rendimento molto contenuto (0.01%) e volatilità comunque presente (1.19%). Il basso Sharpe Ratio conferma l'inefficacia di strategie direzionali in questa fase, che potrebbe riflettere contesti macroeconomici stagnanti o di transizione, come quelli registrati nei periodi successivi a crisi o shock, in cui il mercato attende nuovi catalizzatori ma non mostra una direzionalità chiara. Questo tipo di regime tende a durare nel tempo, come confermato dalla sua durata elevata (2083 giorni), e costituisce una sfida per modelli predittivi o strategie trend-following.

Infine, il regime 0 presenta una dinamica peculiare: pur evidenziando un rendimento medio analogo a quello del regime 2, la volatilità è decisamente più alta (1.44%). Questo suggerisce un contesto di forte instabilità, in cui l'opportunità di profitto è accompagnata da un rischio elevato. Dal punto di vista macroeconomico, potrebbe riflettere periodi di elevata incertezza o eventi sistemici, come la crisi finanziaria del 2008, la pandemia del 2020 o shock geopolitici, in cui il mercato può alternare fasi di recupero e cadute improvvise. Questo regime ha una durata relativamente breve (1002 giorni), ma è probabilmente responsabile della maggior parte dei drawdown osservati nella serie storica.

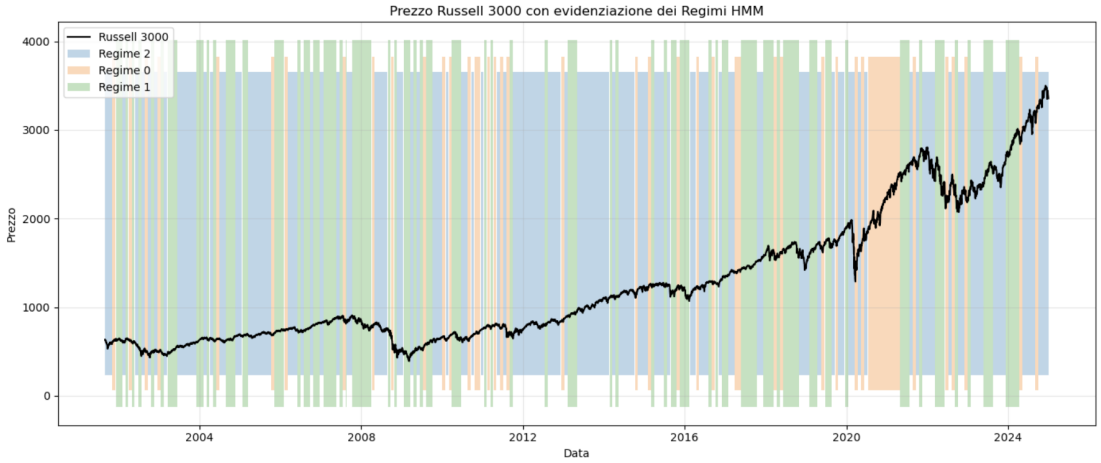


Fig. 4.5: Andamento del Russell 3000 con evidenziazione dei regimi identificati tramite modello Hidden Markov (HMM)

Nel complesso, la segmentazione ottenuta con l'HMM appare coerente con la dinamica dei mercati finanziari come mostrato in fig 4.5, ma fornisce anche un utile framework per comprendere le diverse condizioni di rischio e rendimento nel tempo. Questo tipo di clustering è particolarmente utile per alimentare modelli predittivi supervisionati, poiché consente di etichettare i dati storici con significato economico oltre che statistico.

4.4 Modelli Supervisionati: XGBoost e LSTM

4.4.1 Modello XGBoost: classificazione supervisionata con e senza HMM

Dopo aver generato i regimi di mercato tramite HMM, è stata implementata una fase supervisionata con l'obiettivo di predire tali regimi utilizzando un modello di classificazione. In particolare, è stato adottato l'algoritmo *XGBoost*, scelto per la sua robustezza,

capacità di gestione di feature eterogenee e ottima performance su dataset di dimensioni moderate.

Sono stati costruiti due modelli distinti:

1. *XGBoost con HMM*: il target supervisionato è la colonna `Regime_Smooth`, ovvero la sequenza di regimi generata dal modello HMM e filtrata tramite una media mobile su finestra centrata.
2. *XGBoost naïve*: in questo caso, il target è costituito dalla variabile `NaiveRegime`, costruita direttamente sui rendimenti `LogRet` attraverso soglie empiriche (ad esempio: rialzista se $r > 0.005$, ribassista se $r < -0.005$, altrimenti neutrale).

Entrambi i modelli sono stati addestrati su un set di feature comuni:

- `RSI14` : indicatore di forza relativa
- `CCI20` : Commodity Channel Index
- `Vol20` : volatilità a 20 giorni
- `PctAbove50` : percentuale di chiusure sopra la media mobile a 50 giorni
- `GoldOil` : rapporto oro/petrolio

Le etichette sono state codificate tramite `LabelEncoder` e i dati suddivisi in training e test set (`test_size = 0.2`) con stratificazione, così da mantenere la distribuzione delle classi.

Il classificatore `XGBClassifier` è stato configurato con i seguenti parametri:

- `objective='multi:softmax'`
- `max_depth=3`

- `learning_rate=0.1`
- `n_estimators=100`
- `eval_metric='mlogloss'`

Performance: XGBoost con HMM

L'accuratezza complessiva ottenuta è pari al 61%. Dall'analisi della confusion matrix e del classification report, emerge come il modello riesca a distinguere bene il regime rialzista (classe 2, recall = 0.80), mentre ha più difficoltà nel rilevare il regime ribassista (classe 0, recall = 0.23).

Dal punto di vista strategico, l'applicazione della previsione al trading regime-based mostra metriche interessanti:

- Sharpe Ratio: 0.0394
- Max Drawdown: -0.317
- Cumulative Strategy Return: +66%

Performance: XGBoost naïve

Il modello naïve ha ottenuto un'accuratezza leggermente inferiore, pari al 58%. Tuttavia, la distribuzione delle metriche è più equilibrata tra le classi. La classe ribassista (classe 0) mostra un recall più alto (0.55) rispetto alla versione HMM, ma le classi 1 e 2 (neutrale e rialzista) risultano meno precise nella classificazione.

A livello strategico, la performance risulta decisamente inferiore:

- Sharpe Ratio: negativo

- Cumulative Return: solo +42%

L'inclusione del regime HMM come label supervisionata ha dimostrato un miglioramento netto nella capacità del modello di apprendere la struttura dei regimi di mercato. In particolare, le previsioni si traducono in strategie operative più stabili e redditizie. Questo suggerisce che l'approccio ibrido, dove un modello non supervisionato viene usato come generatore di label per un modello supervisionato, può essere efficace nel contesto del regime switching.

4.4.2 Classificazione tramite LSTM

Per valutare la capacità predittiva di una rete neurale ricorrente nell'identificare i regimi di mercato [9], è stato implementato un modello *Long Short-Term Memory* (LSTM) in due varianti distinte:

1. LSTM *con HMM*, che utilizza come target la colonna `Regime_Smooth`, generata dal modello Hidden Markov;

2. LSTM *senza HMM* (naive), in cui i regimi sono derivati direttamente dai log-return storici attraverso soglie fisse.

Il modello adottato prevede una singola cella LSTM con 64 unità nascoste. Questa scelta nasce da un compromesso tra complessità computazionale e capacità espressiva, come discusso anche nel progetto NYU Tandon Team Alpha [18],che adotta configurazioni simili in contesti di regime switching: un numero eccessivo di neuroni può portare a overfitting, mentre un numero troppo ridotto potrebbe non catturare le dinamiche complesse dei mercati. Diversi studi applicati in ambito finanziario supportano l'efficacia di configurazioni simili per problemi di classificazione multiclasse basata su regimi di mercato.

Per contrastare il rischio di overfitting, è stato introdotto un meccanismo di `Dropout` con tasso pari a 30%. Questo valore, comunemente utilizzato in ambito time-series, riduce la dipendenza del modello da singoli neuroni, migliorando la sua capacità di generalizzazione.

Le feature in input (`RSI14`, `CCI20`, `Vol20`, `PctAbove50`, `GoldOil`) sono state normalizzate utilizzando un `MinMaxScaler`, per garantire che ogni variabile abbia lo stesso range e, di conseguenza, lo stesso impatto sull'ottimizzazione del modello.

Il parametro `lookback` è stato impostato a 20 giorni, per permettere al modello di apprendere le dinamiche recenti senza allungare troppo la sequenza — con il rischio di sovraccaricare la memoria della rete o disperdere l'informazione rilevante.

Il modello è stato addestrato per 50 epoche, con una dimensione del batch pari a 32 [9] e una suddivisione del 20% dei dati di training per la validazione. Questi parametri si sono rivelati sufficienti per garantire una buona stabilità del processo di apprendimento, mantenendo al contempo tempi di training contenuti.

Entrambi i modelli, LSTM con e senza HMM, sono stati valutati su un set di test non visto in fase di training. La classificazione è stata eseguita sulla base di tre classi di regime.

La valutazione si è basata anche in questo caso su:

- *Accuracy*, ossia la proporzione di etichette correttamente classificate sul test set;

- *Classification report* e *confusion matrix*, per esaminare la precisione, il recall e l'F1-score per ciascuna classe.

Tabella 4.2: Risultati delle performance del modello LSTM con e senza HMM

Modello	**Accuracy**	**Macro F1**	**Weighted F1**
LSTM con HMM	0.55	0.46	0.52
LSTM senza HMM (naive)	0.53	0.38	0.46

I risultati dimostrano che il supporto informativo fornito dall'HMM aiuta il modello LSTM a differenziare meglio i regimi estremi, in particolare quello rialzista. Al contrario, l'approccio naive, privo di un filtro probabilistico ex-ante, soffre di un forte sbilanciamento nella classificazione, penalizzando la precisione nelle fasi di mercato più significative dal punto di vista operativo.

In ottica strategica, ciò suggerisce che l'impiego del modello HMM per pre-elaborare i segnali di regime possa contribuire a migliorare l'efficacia dei modelli di deep learning su serie storiche di mercato.

4.5 Valutazione strategica delle performance

Dopo aver addestrato e confrontato i modelli su base predittiva, si è reso necessario valutare anche l'efficacia operativa delle previsioni in termini di risultati concreti su una strategia di investimento. Per questo, è stata implementata una procedura di simulazione delle performance basata sulle previsioni di regime fornite da ciascun modello, secondo una logica di investimento semplice ma coerente con la struttura del problema.

Le previsioni giornaliere dei regimi vengono trasformate in segnali operativi con la seguente mappatura:

- Regime 2 $\rightarrow$ posizione *long* (acquisto),
- Regime 1 $\rightarrow$ posizione *flat* (nessuna esposizione),

- Regime 0 $\rightarrow$ posizione *short* (vendita allo scoperto).

Il rendimento giornaliero della strategia viene quindi calcolato moltiplicando la posizione per il log-return giornaliero, applicando inoltre un eventuale *costo di transazione* per ogni cambio di regime.

La strategia viene valutata secondo una serie di indicatori ampiamente utilizzati nella letteratura finanziaria:

- Rendimento medio giornaliero (Return_Mean)
- Volatilità (deviazione standard dei rendimenti)
- Sharpe Ratio
- Maximum Drawdown
- Ulcer Index (profondità e durata delle fasi negative)
- Gain-to-Pain Ratio
- Cumulative Strategy (crescita cumulata della strategia)
- Cumulative Market (benchmark buy-and-hold sul mercato)

Di seguito sono riportati i risultati ottenuti dalle diverse strategie:

	Model	Return_Mean	Volatility	Sharpe	MaxDD	UlcerIndex	GainPain	CumulativeStrategy	CumulativeMarket
0	True_HMM	0.000107	0.010096	0.010631	-0.590929	0.227306	1.041439	1.868018	5.116785
1	XGB_HMM	0.000436	0.011067	0.039399	-0.317184	0.113267	1.163555	1.661924	1.001269
2	LSTM_HMM	-0.000016	0.009188	-0.001692	-0.322803	0.181262	0.994641	0.982111	1.156812
3	LSTM_Naive	0.000124	0.006852	0.018077	-0.186841	0.099496	1.081620	1.155232	1.165782
4	XGB_Naive	-0.000727	0.010350	-0.070277	-0.584310	0.397380	0.746735	0.428523	2.065416

Fig. 4.6: Confronto tra le strategie di trading basate sui modelli

I risultati mostrano chiaramente che il modello XGBoost con HMM ottiene il miglior equilibrio tra rendimento, rischio e stabilità: presenta un rendimento medio interessante, drawdown moderato, e un rapporto gain/pain maggiore di 1. Questo suggerisce una buona coerenza tra previsione e dinamica reale del mercato.

Il LSTM con HMM, al contrario, non mostra un vantaggio competitivo: ottiene rendimenti negativi e un *Sharpe Ratio* prossimo allo zero. In compenso, la sua variante naive ottiene performance migliori in termini di cumulative strategy.

Il modello XGB Naive, invece, sottoperforma chiaramente, con valori negativi di rendimento e Sharpe. Interessante notare anche il confronto con il True_HMM non predittivo, che fornisce una strategia positiva (ma non realisticamente implementabile) utile come benchmark.

Nel complesso, si conferma che l'integrazione dell'HMM come feature aggiuntiva migliora le performance del classificatore XGBoost, mentre risulta meno efficace se combinata con LSTM.

Conclusioni

Il confronto tra XGBoost e LSTM, all'interno di un framework supervisionato basato su regimi di mercato, ha rivelato dinamiche interessanti e insegnamenti preziosi. I risultati mostrano chiaramente che la forza predittiva di questi modelli non risiede solo nella loro architettura, ma nel modo in cui affrontano la complessità intrinseca dei dati finanziari e l'interpretabilità delle loro decisioni.

Da un lato, l'approccio tree-based si è confermato più robusto, efficiente e trasparente. XGBoost, grazie alla sua capacità di gestire dati tabulari ad alta dimensionalità e alla sua struttura regolarizzata, tende a fornire segnali operativi stabili, facilmente spiegabili anche in contesti "senza tempo". Questa interpretabilità è cruciale in finanza, dove ogni decisione deve poter essere giustificata a livello logico e quantitativo.

Dall'altro lato, l'LSTM, pur possedendo una naturale predisposizione a captare pattern temporali complessi, ha mostrato una certa vulnerabilità nei confronti del rumore e della scarsità di dati specifici: senza tecniche come early-stopping o tuning approfondito, tende a sovra-allenarsi, limitando la sua capacità di generalizzazione. Questo ne appiattisce l'efficacia operativa a favore di configurazioni più "ingranate" tipiche dei regimi, piuttosto che strategie lotte nel breve termine.

In sintesi, il vero vantaggio deriva dalla coesistenza di modelli diversi. XGBoost offre stabilità e interpretabilità; LSTM offre potenziale comprensione delle dinamiche temporali, ma richiede contesti favorevoli per esprimerlo. La sfida futura è unire queste

qualità, affiancando modelli che garantiscano robustezza decisionale con altri in grado di captare l'evoluzione nel tempo dei mercati. Una pista promettente in letteratura, confermata anche in ambito crypto e finanza aziendale, suggerisce che modelli ibridi, come LSTM+XGBoost, possano unire il meglio dei due mondi.

Il messaggio centrale di questa analisi è che, in finanza, l'efficacia predittiva procede a braccetto con la solidità operativa. Puntare su modelli interpretativi e ben calibrati significa costruire non solo predizioni, ma strategie finanziarie affidabili e adattive.

Appendice A Codice Python

A.1 Importazione dei dati e features engineering

```
# --- Importazione dati e features engineering ---
import yfinance as yf
import pandas as pd
import numpy as np
import ta
# Scarica i dati
start_date = "1995-01-01"
end_date = "2025-01-01"
df_price = yf.download("^RUA", start=start_date, end=end_date,
    auto_adjust=True)
vix = yf.download("^VIX", start=start_date, end=end_date,
    auto_adjust=True)['Close']
gold = yf.download("GC=F", start=start_date, end=end_date,
    auto_adjust=True)['Close']
oil = yf.download("CL=F", start=start_date, end=end_date,
    auto_adjust=True)['Close']

df = pd.DataFrame(index=df_price.index)
df[['Open', 'High', 'Low', 'Close', 'Volume']] =
    df_price[['Open', 'High', 'Low', 'Close', 'Volume']]
```

```
# Feature engineering
df['LogRet'] = np.log(df['Close'] / df['Close'].shift(1))
df['Vol20'] = df['LogRet'].rolling(20).std() * np.sqrt(252)
df['VIX'] = vix.reindex(df.index).to_numpy().ravel()
df['Gold'] = gold.reindex(df.index).to_numpy().ravel()
df['Oil'] = oil.reindex(df.index).to_numpy().ravel()
df['RSI14'] = ta.momentum.RSIIndicator(df['Close'],
    window=14).rsi()
df['CCI20'] = ta.trend.CCIIndicator(df['High'], df['Low'],
    df['Close'], window=20).cci()
df['ADX14'] = ta.trend.ADXIndicator(df['High'], df['Low'],
    df['Close'], window=14).adx()
df['ATR14'] = ta.volatility.AverageTrueRange(df['High'],
    df['Low'], df['Close'], window=14).average_true_range()
df['DonchianHigh20'] = df['High'].rolling(20).max()
df['DonchianLow20'] = df['Low'].rolling(20).min()
df['DonchianWidth'] = df['DonchianHigh20'] - df['DonchianLow20']
df['PctAbove50'] = (df['Close'] >
    df['Close'].rolling(50).mean()).astype(int)
df['MA10'] = df['Close'].rolling(10).mean()
df['MA50'] = df['Close'].rolling(50).mean()
df['MA200'] = df['Close'].rolling(200).mean()
df['EWMA10'] = df['Close'].ewm(span=10).mean()
df['Momentum10'] = df['Close'] / df['Close'].shift(10) - 1
df['ZLogRet20'] = (df['LogRet'] -
    df['LogRet'].rolling(20).mean()) /
    df['LogRet'].rolling(20).std()
df['GoldOil'] = df['Gold'] / df['Oil']
df['VIX_change_pct'] = df['VIX'].pct_change()
df['Oil_change_pct'] = df['Oil'].pct_change()
df['Pct_above_50MA'] = (
    (df['Close'] > df['Close'].rolling(50).mean())
    .rolling(50).mean() * 100
```

```
)
df['ATR_normalized'] = ta.volatility.AverageTrueRange(
    high=df['High'],
    low=df['Low'],
    close=df['Close'],
    window=14
).average_true_range() / df['Close']
bb = ta.volatility.BollingerBands(close=df['Close'], window=20,
   window_dev=2)
df['BollingerWidth_20'] = (bb.bollinger_hband() -
   bb.bollinger_lband()) / bb.bollinger_mavg()
# Commodity Channel Index (CCI) 20
tp = (df['High'] + df['Low'] + df['Close']) / 3
cci = (tp - tp.rolling(20).mean()) / (0.015 *
   tp.rolling(20).std())
df['CCI_20'] = cci
# Mass Index
high_low = df['High'] - df['Low']
ema1 = high_low.ewm(span=9, adjust=False).mean()
ema2 = ema1.ewm(span=9, adjust=False).mean()
mass = (ema1 / ema2).rolling(25).sum()
df['Mass_Index'] = mass
# RSI 14
delta = df['Close'].diff()
up = delta.clip(lower=0).ewm(alpha=1/14, adjust=False).mean()
down = -delta.clip(upper=0).ewm(alpha=1/14, adjust=False).mean()
rs = up / (down + 1e-12)
df['RSI_14'] = 100 - (100 / (1 + rs))
# Return 1M (21 giorni)
df['Return_1M'] = df['Close'].pct_change(21)
# Return 3M (63 giorni)
df['Return_3M'] = df['Close'].pct_change(63)
# EWMA Volatilità 20
```

```
df['EWMA_vol_20'] = df['Close'].pct_change().ewm(span=20).std()

df.dropna(inplace=True)
print(df.head())
```

A.2 Grafici descrittivi

```
# --- Grafici descrittivi features ---
import matplotlib.pyplot as plt

def plot_price_sma_gold_oil_fullperiod(df, savepath=None):
    df_plot = df.copy()
    df_plot['MA50'] = df_plot['Close'].rolling(50).mean()
    df_plot['MA200'] = df_plot['Close'].rolling(200).mean()

    fig, ax = plt.subplots(figsize=(14, 6))

    ax.plot(df_plot.index, df_plot['Close'], label='Prezzo
        Russell 3000', linewidth=1.2)
    ax.plot(df_plot.index, df_plot['MA50'], label='MA 50 giorni',
        linestyle='--')
    ax.plot(df_plot.index, df_plot['MA200'], label='MA 200
        giorni', linestyle='--')

    ax.plot(df_plot.index, df_plot['Gold'], label='Prezzo Oro',
        alpha=0.7)
    ax.plot(df_plot.index, df_plot['Oil'], label='Prezzo
        Petrolio', alpha=0.7)

    ax.set_title("Prezzo Russell 3000, MA 50/200, Oro e Petrolio
        (1995-2025)")
```

```
    ax.set_ylabel("Prezzi")
    ax.set_xlabel("Anno")
    ax.grid(True, alpha=0.3)
    ax.legend(loc='upper left')
    plt.tight_layout()
    if savepath:
        plt.savefig(savepath, dpi=300)
    plt.show()

def plot_price_vs_commodities_normalized(df, savepath=None):
    df_norm = df[['Close', 'Gold', 'Oil']].copy()
    df_norm = df_norm.div(df_norm.iloc[0]).mul(100)

    fig, ax = plt.subplots(figsize=(14, 6))
    ax.plot(df_norm.index, df_norm['Close'], label='Russell 3000
        (base 100)')
    ax.plot(df_norm.index, df_norm['Gold'], label='Oro (base
        100)')
    ax.plot(df_norm.index, df_norm['Oil'], label='Petrolio (base
        100)')

    ax.set_title("Prezzi Normalizzati     Russell 3000, Oro,
        Petrolio (base 100)")
    ax.set_ylabel("Indice base 100")
    ax.set_xlabel("Anno")
    ax.grid(True, alpha=0.3)
    ax.legend()
    plt.tight_layout()
    if savepath:
        plt.savefig(savepath, dpi=300)
    plt.show()

def plot_zlogret_full(df, savepath=None):
```

```
    fig, ax = plt.subplots(figsize=(14, 5))
    ax.plot(df.index, df['ZLogRet20'], color='tab:blue',
        linewidth=1)
    ax.set_title(" Z Log  Return (standardizzato) - Periodo
        completo")
    ax.set_ylabel("ZLogRet")
    ax.set_xlabel("Anno")
    ax.grid(True, alpha=0.3)
    if savepath:
        plt.savefig(savepath, dpi=300)
    plt.tight_layout()
    plt.show()

def plot_volatility_full(df, savepath=None):
    fig, ax = plt.subplots(figsize=(14, 5))
    ax.plot(df.index, df['Vol20'], color='tab:red', linewidth=1)
    ax.set_title("Volatilità Rolling 20 giorni - Periodo
        completo")
    ax.set_ylabel("Volatilità (20 giorni)")
    ax.set_xlabel("Anno")
    ax.grid(True, alpha=0.3)
    if savepath:
        plt.savefig(savepath, dpi=300)
    plt.tight_layout()
    plt.show()
# Genera tutti i grafici su tutto il periodo
plot_price_ma_gold_oil_fullperiod(df)
plot_price_vs_commodities_normalized(df)
plot_zlogret_full(df)
plot_volatility_full(df)
```

A.3 Preparazione dei dati

```
# Preparazione dati per il modello
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

# 1 Selezione delle colonne da usare nell'HMM

features_hmm = [
    'VIX',
    'Oil_change_pct',
    'Pct_above_50MA',
    'RSI_14' ]
# 2 Pulizia dati: rimuove infiniti e valori mancanti
df_clean = (
    df[features_hmm]
    .replace([np.inf, -np.inf], np.nan)
    .dropna()
    .copy()
)
assert np.isfinite(df_clean.values).all(), "Ci sono ancora valori
   non finiti nei dati!"

# 3 Rimozione collinearità: elimina variabili con correlazione >
   0.95
corr_matrix = df_clean.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
   k=1).astype(bool))
to_drop = [col for col in upper.columns if any(upper[col] > 0.95)]
if to_drop:
    print("Colonne eliminate per alta correlazione:", to_drop)
```

```
    df_clean = df_clean.drop(columns=to_drop)

# 4 Standardizzazione
scaler = StandardScaler()
df_clean[features_hmm] =
   scaler.fit_transform(df_clean[features_hmm])
print(df_clean.describe())
print(df_clean.isna().sum())
```

A.4 HMM

```
# --- HMM Rolling ---
from hmmlearn.hmm import GaussianHMM
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
X_full = df_clean.values
# Parametri Rolling
window_size = 250  # possibilità di aumentare per ottenere più
   stabilità
n_components = 3

# Rolling Out-of-Sample Fit & Predict
hidden_states_rolling = np.full(len(df_clean), np.nan)

model = None
last_fit_index = None
refit_freq = 20  # rifit ogni 20 giorni

for t in range(window_size, len(df_clean)):
    # Refit solo ogni 20 giorni o al primo passo
```

```
    if (last_fit_index is None) or ((t - last_fit_index) >=
        refit_freq):
        X_train = X_full[t - window_size:t]
        model = GaussianHMM(n_components=n_components,
            covariance_type="diag", n_iter=500,tol=1e-2,
            random_state=42)
        model.fit(X_train)
        last_fit_index = t

    # Predizione con il modello corrente
    X_test = X_full[t].reshape(1, -1)
    hidden_states_rolling[t] = model.predict(X_test)[0]

# Creazione DataFrame completo con tutte le feature
df_hmm = df.copy()
df_hmm['Regime'] = hidden_states_rolling

# Creazione della colonna "Regime_Smooth" anche la versione
   "smooth" per compatibilità con XGBoost/LSTM#

window_smooth = 15  # possibilità di aumentare per ottenere
   migliori risultati
df_hmm['Regime_Smooth'] = (
    df_hmm['Regime']
    .rolling(window_smooth, center=True)
    .apply(lambda x: x.mode().iloc[0] if not x.mode().empty else
        x.iloc[-1])
    .bfill()
    .ffill()
    .astype(int)
)

# Pulizia finale
```

```
df_hmm.dropna(inplace=True)

print("=== Statistiche per ciascun regime HMM ===")
stats = df_hmm.groupby('Regime').agg({
    'LogRet': ['count', 'mean', 'std'],
})
stats.columns = ['Num Giorni', 'Rendimento Medio', 'Volatilità']
stats['Sharpe'] = stats['Rendimento Medio'] / stats['Volatilità']
print(stats.round(4))


# Assumiamo df_hmm contenente indice datetime, colonne
    'Regime_Smooth' e 'Close'
fig, ax = plt.subplots(figsize=(14, 6))
ax.plot(df_hmm.index, df_hmm['Close'], label='Russell 3000',
    color='black')

# Evidenzia a intervalli come aree verticali colorate
for regime in df_hmm['Regime_Smooth'].unique():
    mask = df_hmm['Regime_Smooth'] == regime
    ax.fill_between(df_hmm.index, ax.get_ylim()[0],
        ax.get_ylim()[1],
                    where=mask, alpha=0.3,
                    label=f'Regime {regime}')

ax.set_title('Prezzo Russell 3000 con evidenziazione dei Regimi
    HMM')
ax.set_xlabel('Data')
ax.set_ylabel('Prezzo')
ax.legend()
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

A.5 XGboost con HMM

```
# --- XGBoost con HMM input: Regime_Smooth ---
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
    confusion_matrix
from sklearn.preprocessing import LabelEncoder
dates = df_hmm.index.to_series().reset_index(drop=True)
features = ['RSI14', 'CCI20', 'Vol20', 'PctAbove50', 'GoldOil']
X = df_hmm[features]
y = df_hmm['Regime_Smooth'].astype(int)
# Rimappatura classi [1,2]      [0,1]
encoder = LabelEncoder()
y = encoder.fit_transform(y)

num_classes = len(np.unique(y))

X_train, X_test, y_train, y_test, dates_train, dates_test =
    train_test_split(
     X, y, dates, stratify=y, test_size=0.2, random_state=42
)
model = XGBClassifier(
     objective='multi:softmax',
     num_class=num_classes,
     max_depth=3,
     learning_rate=0.1,
     n_estimators=100,
     use_label_encoder=False,
     eval_metric='mlogloss',
```

```
    random_state=42
)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Salva le predizioni XGBoost nel DataFrame
df_hmm_pred = df_hmm.copy()
df_hmm_pred.loc[X_test.index, 'PredictedRegime_XGB_HMM'] = y_pred

# Valutazione strategia regime-based per XGBoost
result_xgb = evaluate_strategy(df_hmm_pred,
   regime_col='PredictedRegime_XGB_HMM', name='XGB_HMM')
print(result_xgb)
print("Classi presenti (dopo LabelEncoder):", np.unique(y))
print("Classification Report:\n", classification_report(y_test,
   y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

A.6 XGboost senza HMM

```
#---XGboost senza HMM, input: NaiveRegime---
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
   confusion_matrix
from sklearn.preprocessing import LabelEncoder
# FEATURES
features = ['RSI14', 'CCI20', 'Vol20', 'PctAbove50', 'GoldOil']

X = df_lstm[features]
```

```
y = df_lstm['NaiveRegime'].astype(int)

# RIMAPPA le classi a 0-based con LabelEncoder
encoder_naive = LabelEncoder()
y = encoder_naive.fit_transform(y)  # es. [0, 1, 2] o [0, 1]

num_classes = len(np.unique(y))

# SPLIT
X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.2, random_state=42
)
# MODELLO XGBoost
model_naive = XGBClassifier(
    objective='multi:softmax',
    num_class=num_classes,
    max_depth=3,
    learning_rate=0.1,
    n_estimators=100,
    use_label_encoder=False,
    eval_metric='mlogloss',
    random_state=42
)
# FIT & PREDICT
model_naive.fit(X_train, y_train)
y_pred_naive = model_naive.predict(X_test)

# Aggiungi le predizioni naive al dataframe per la valutazione
    strategica
df_hmm_pred['PredictedRegime_XGB_Naive'] = np.nan
df_hmm_pred.loc[X_test.index, 'PredictedRegime_XGB_Naive'] =
    y_pred_naive
# VALUTAZIONE
```

```
print("Classi presenti (Naive):", np.unique(y))
print("Classification Report (XGBoost senza HMM):\n",
    classification_report(y_test, y_pred_naive))
print("Confusion Matrix:\n", confusion_matrix(y_test,
    y_pred_naive))
```

A.7 LSTM

```
# --- LSTM ---
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix,
    ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

# PARAMETRI
lookback = 20
features = ['RSI14', 'CCI20', 'Vol20', 'PctAbove50', 'GoldOil']

# DATASET
# df_hmm è già il risultato del tuo HMM (con colonna 'Regime')
df_lstm = df_hmm.copy()

# Normalizza le feature
scaler = MinMaxScaler()
df_lstm[features] = scaler.fit_transform(df_lstm[features])
```

```
# Etichette alternative (senza HMM) basate su log-returns
df_lstm['NaiveRegime'] = np.where(df_lstm['LogRet'] > 0.005, 2,
                                  np.where(df_lstm['LogRet'] < -
                                      0.005, 0, 1))

# FUNZIONE SEQUENZE
def create_sequences(df, features, target, lookback):
    X, y = [], []
    for i in range(lookback, len(df)):
        X.append(df[features].iloc[i - lookback:i].values)
        y.append(df[target].iloc[i])
    return np.array(X), to_categorical(np.array(y), num_classes=3)

# Crea sequenze per entrambe le versioni
X_hmm, y_hmm = create_sequences(df_lstm, features,
   'Regime_Smooth', lookback)
X_naive, y_naive = create_sequences(df_lstm, features,
   'NaiveRegime', lookback)

# SPLIT
Xh_train, Xh_test, yh_train, yh_test = train_test_split(X_hmm,
   y_hmm, test_size=0.2, random_state=42)
Xn_train, Xn_test, yn_train, yn_test = train_test_split(X_naive,
   y_naive, test_size=0.2, random_state=42)

# COSTRUISCI MODELLO
def build_lstm_model(input_shape, num_classes):
    model = Sequential()
    model.add(LSTM(64, input_shape=input_shape))
    model.add(Dropout(0.3))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
```

```
    model.compile(optimizer='adam',
        loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# TRAINING
model_hmm = build_lstm_model(Xh_train.shape[1:], 3)
model_naive = build_lstm_model(Xn_train.shape[1:], 3)

model_hmm.fit(Xh_train, yh_train, epochs=50, batch_size=32,
    validation_split=0.2)
model_naive.fit(Xn_train, yn_train, epochs=50, batch_size=32,
    validation_split=0.2)

# VALUTAZIONE
loss_hmm, acc_hmm = model_hmm.evaluate(Xh_test, yh_test)
loss_naive, acc_naive = model_naive.evaluate(Xn_test, yn_test)

print(f"\n LSTM con HMM:    Accuracy = {acc_hmm:.4f} | Loss =
    {loss_hmm:.4f}")
print(f" LSTM senza HMM: Accuracy = {acc_naive:.4f} | Loss =
    {loss_naive:.4f}")

# Previsioni
y_pred_hmm = np.argmax(model_hmm.predict(Xh_test), axis=1)
y_true_hmm = np.argmax(yh_test, axis=1)

y_pred_naive = np.argmax(model_naive.predict(Xn_test), axis=1)
y_true_naive = np.argmax(yn_test, axis=1)

# Confusion Matrix - LSTM con HMM
labels_hmm = np.unique(np.concatenate([y_true_hmm, y_pred_hmm]))
cm_hmm = confusion_matrix(y_true_hmm, y_pred_hmm,
    labels=labels_hmm)
```

```
disp_hmm = ConfusionMatrixDisplay(confusion_matrix=cm_hmm,
    display_labels=[f'Regime {i}' for i in labels_hmm])
disp_hmm.plot(cmap='Blues')
plt.title("Confusion Matrix     LSTM con HMM")
plt.grid(False)
plt.show()

# Confusion Matrix - LSTM senza HMM
labels_naive = np.unique(np.concatenate([y_true_naive,
    y_pred_naive]))
cm_naive = confusion_matrix(y_true_naive, y_pred_naive,
    labels=labels_naive)
disp_naive = ConfusionMatrixDisplay(confusion_matrix=cm_naive,
    display_labels=[f'Regime {i}' for i in labels_naive])
disp_naive.plot(cmap='Reds')
plt.title("Confusion Matrix - LSTM senza HMM")
plt.grid(False)
plt.show()
# classication report
print("\n Classification Report - LSTM con HMM")
print(classification_report(y_true_hmm, y_pred_hmm))
print("\n Classification Report - LSTM senza HMM")
print(classification_report(y_true_naive, y_pred_naive))
```

A.8 Analisi dei risultati

```
df_lstm_pred = df_hmm.copy()
df_lstm_pred.loc[:, 'PredictedRegime_LSTM_HMM'] = np.nan
df_lstm_pred.loc[:, 'PredictedRegime_LSTM_Naive'] = np.nan
```

```
df_lstm_pred.iloc[:len(y_pred_hmm),
    df_lstm_pred.columns.get_loc('PredictedRegime_LSTM_HMM')] =
    y_pred_hmm
df_lstm_pred.iloc[:len(y_pred_naive),
    df_lstm_pred.columns.get_loc('PredictedRegime_LSTM_Naive')] =
    y_pred_naive


def evaluate_strategy(df, regime_col, ret_col='LogRet',
    name='Model', tcost=0.0):
    df = df.copy()

    # Mappa stati -> posizione operativa
    # ATTENZIONE: cambia le chiavi 2,0,1 se i tuoi ID stati sono
        diversi
    df['Position'] = df[regime_col].map({2: 1, 0: -1, 1: 0})

    # Ritorno strategia con shift per evitare lookahead
    strat_ret = df['Position'].shift(1) * df[ret_col]

    # Costi di transazione per cambio posizione (per es. 10 bps =
        0.001)
    if tcost and tcost > 0:
        turns = df['Position'].diff().abs().fillna(0.0)    # 0,1,2
            (da long a short = 2)
        # costo proporzionale al numero di    giri    (1 per
            passare cash<->long/short, 2 per long<->short)
        strat_ret = strat_ret - tcost * turns

    # Drop prima riga (NaN da shift) ed eventuali altri NaN
    strat_ret = strat_ret.dropna()

    # Cumulativi
```

```
    # Se ret_col sono LOG returns:
    cum_strat = np.exp(strat_ret.cumsum())
    cum_mkt   = np.exp(df.loc[strat_ret.index, ret_col].cumsum())

    # Se invece usi RITORNI SEMPLICI, sostituisci le 2 righe
       sopra con:
    # cum_strat = (1 + strat_ret).cumprod()
    # cum_mkt   = (1 + df.loc[strat_ret.index, ret_col]).cumprod()

    # Drawdown
    roll_max  = cum_strat.cummax()
    drawdown  = cum_strat / roll_max - 1.0

    # Metriche
    ret_mean  = strat_ret.mean()
    vol       = strat_ret.std()
    sharpe    = ret_mean / vol if vol > 0 else np.nan
    maxdd     = drawdown.min()
    ulcer     = np.sqrt(np.mean((drawdown[drawdown < 0] ** 2)))
       if (drawdown < 0).any() else 0.0

    gains = strat_ret[strat_ret > 0].sum()
    pains = strat_ret[strat_ret < 0].sum()  # negativo
    gain_pain = gains / abs(pains) if pains != 0 else np.inf

    results = {
        'Model':        name,
        'Return_Mean':  ret_mean,
        'Volatility':   vol,
        'Sharpe':       sharpe,
        'MaxDD':        maxdd,
        'UlcerIndex':   ulcer,
        'GainPain':     gain_pain,
```

```
        'CumulativeStrategy': cum_strat.iloc[-1],
        'CumulativeMarket':   cum_mkt.iloc[-1],
    }
    return pd.Series(results)

results = []

# Aggiungi ogni strategia
results.append(evaluate_strategy(df_hmm, 'Regime_Smooth',
   name='True_HMM'))
results.append(evaluate_strategy(df_hmm_pred,
   'PredictedRegime_XGB_HMM', name='XGB_HMM'))
results.append(evaluate_strategy(df_lstm_pred,
   'PredictedRegime_LSTM_HMM', name='LSTM_HMM'))
results.append(evaluate_strategy(df_lstm_pred,
   'PredictedRegime_LSTM_Naive', name='LSTM_Naive'))
results.append(evaluate_strategy(df_hmm_pred,
   'PredictedRegime_XGB_Naive', name='XGB_Naive'))
# Converte in DataFrame e stampa
df_results = pd.DataFrame(results)
display(df_results)
```

Bibliografia

[1] Louis Bachelier, *Théorie de la spéculation*, Annales scientifiques de l'École Normale Supérieure **17** (1900).

[2] Rolf W. Banz, *The relationship between return and market value of common stocks*, Journal of Financial Economics **9** (1981).

[3] Carl Bergström, *Impact of time steps on stock market prediction with lstm*, Bachelor's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2019.

[4] Werner F. M. De Bondt and Richard H. Thaler, *Does the stock market overreact?*, The Journal of Finance **40** (1985).

[5] John Y. Campbell, Andrew W. Lo, and A. Craig MacKinlay, *The econometrics of financial markets*, Princeton University Press, Princeton, NJ, 1997.

[6] Tianqi Chen and Carlos Guestrin, *Xgboost: A scalable tree boosting system*, arXiv preprint arXiv:1603.02754 (2016).

[7] Arda Degirmenci, *Hidden markov models*, `https://scholar.harvard.edu/files/adegirmenci/files/hmm_adegirmenci_2014.pdf`, 2014.

[8] Eugene F. Fama, *Efficient capital markets: A review of theory and empirical work*, The Journal of Finance **25** (1970).

[9] UC Berkeley Haas, *Short squeezers - modeling market regimes with lstm*, `https://uc-berkeley-haas-short-squeezers.pdf`, 2021, Quantitative Open Tournament.

[10] John C. Hull, *Machine learning in business: Un'introduzione alla scienza dei dati*, Wiley, 2021, Edizione italiana a cura del Prof. E. Barone.

[11] IBM Cloud Education, *What is machine learning?*, 2021.

[12] Narasimhan Jegadeesh and Sheridan Titman, *Returns to buying winners and selling losers: Implications for stock market efficiency*, The Journal of Finance **48** (1993).

[13] John C. Hull, *Options, futures, and other derivatives*, 11th ed., Pearson Education, 2021.

[14] Daniel Kahneman and Amos Tversky, *Prospect theory: An analysis of decision under risk*, Econometrica **47** (1979).

[15] Andrew W. Lo, *The adaptive markets hypothesis: Market efficiency from an evolutionary perspective*, The Journal of Portfolio Management **30** (2004).

[16] MIT Sloan School of Management, *Machine learning, explained*, 2021.

[17] Kevin P. Murphy, *Machine learning: A probabilistic perspective*, MIT Press, Cambridge, MA, 2012.

[18] NYU Tandon School of Engineering, *Team alpha - regime detection using technical indicators and lstm*, `https://nyu-tandon-team-alpha.pdf`, 2021, Quantitative Open Tournament.

[19] L. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE **77** (1989).

[20] Dan Roth, *Decision trees*, https://www.cis.upenn.edu/~danroth/Teaching/CS446-17/LectureNotesNew/dtree/main.pdf, 2017, Lecture notes for CS446 - Machine Learning, University of Pennsylvania.

[21] Paul A. Samuelson, *Proof that properly anticipated prices fluctuate randomly*, Industrial Management Review **6** (1965), no. 2, 41–49.

[22] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, *Financial time series forecasting with deep learning: A systematic literature review*, Applied Soft Computing **100** (2020).

[23] William F. Sharpe, *Capital asset prices: A theory of market equilibrium under conditions of risk*, The Journal of Finance **19** (1964).

[24] Kevin Sheppard, *Financial econometrics: Notes*, 2021, Lecture notes, last updated February 2, 2021.

[25] Alan Stuart and J. Keith Ord, *Kendall's advanced theory of statistics, volume 1: Distribution theory*, 5 ed., Oxford University Press, 1987.

[26] NYU Tandon, *Five_1 guys*, https://rotman.rtslabs.io/files/Five_1_Guys.pdf, 2020, Approccio di smoothing dei regimi tramite finestra modale centrata per ridurre falsi segnali.

[27] UC Berkeley Haas, *Short squeezers*, https://rotman.rtslabs.io/files/Short_Squeezers.pdf, 2021, Modello di classificazione dei regimi basato su performance ex post dei cluster HMM.

[28] Cornell University, *The boy who cried bear: One price*, https://rotman.rtslabs.io/files/One_Price.pdf, 2021, Studio sui modelli di regime switching e strategie long-short, basato su HMM con rolling window.

[29] Johns Hopkins University, *Superficial intelligence*, https://rotman.rtslabs.io/files/Superficial_Intelligence.pdf, 2021, Tesi partecipante alla Rotman International Trading Competition, analisi di regime switching tramite HMM rolling.