

POLITECNICO DI TORINO

Corso di Laurea
in Matematica per l'Ingegneria

Tesi di Laurea

Stima delle Copule in Finanza: Analisi sul DAX



Relatore
Franco Pellerey

Candidato
Andrea Rostagno

Anno Accademico 2024-2025

Sommario

Questa tesi si concentra sull'uso delle copule nella modellizzazione della dipendenza tra asset finanziari, con un'applicazione specifica al mercato azionario tedesco (DAX). L'obiettivo principale è analizzare l'efficacia delle copule nel descrivere le relazioni di dipendenza non lineare tra rendimenti finanziari e confrontarle con modelli più tradizionali.

1. Introduzione

Viene presentata la motivazione della ricerca, evidenziando l'importanza della modellizzazione della dipendenza nei mercati finanziari e il ruolo delle copule nel superare le limitazioni dei modelli lineari.

2. Teoria delle Copule

Si introduce il concetto matematico di copula, descrivendone le proprietà fondamentali e le principali famiglie (Gaussiane, t-Student, Archimedea). Si analizza il legame tra copule e funzioni di distribuzione congiunta.

3. Modelli di Dipendenza Finanziaria

Si discutono le misure tradizionali di dipendenza (correlazione lineare, correlazione di Spearman e di Kendall) e si confrontano con l'approccio delle copule.

4. Applicazione al DAX

Si descrive il dataset utilizzato, la metodologia di stima dei parametri delle copule e la scelta del modello più adatto ai dati finanziari del DAX.

5. Analisi dei Risultati

Si confrontano le prestazioni delle copule rispetto ai modelli tradizionali, evidenziando vantaggi e svantaggi dell'approccio basato sulle copule.

6. Esempio di Portafoglio

Si riassumono i risultati ottenuti e si suggerisce un esempio pratico di applicazione ad un portafoglio long.

Crediti e Supporto Tecnologico

Questa tesi è stata sviluppata con il supporto di **OOPS TECH SRL**, una PMI innovativa nel settore Fintech situata a Torino. L'azienda ha messo a disposizione le avanzate tecnologie e le risorse del proprio laboratorio. Questo contributo ha permesso di arricchire il contenuto e la qualità del lavoro svolto, offrendo accesso a strumenti e metodi all'avanguardia fondamentali per il completamento della ricerca.

Ringraziamenti

Desidero esprimere la mia sincera gratitudine al mio relatore, il cui supporto costante è stato fondamentale per il completamento di questa tesi. La sua disponibilità nel rispondere rapidamente alle mie domande e nel fornirmi preziosi consigli e correzioni ha reso questo percorso più chiaro e stimolante.

Un ringraziamento speciale va a **Davide**, che ha rappresentato una guida essenziale all'interno di **OOPS TECH SRL**. Grazie alla sua competenza e pazienza, ho potuto approfondire la programmazione in Python e ricevere risposte esaustive a numerose domande, oltre a preziose linee guida che hanno orientato il mio lavoro.

Infine, un pensiero di riconoscenza va a tutte le persone che, con la loro passione e il loro entusiasmo, mi stanno facendo scoprire e apprezzare sempre di più il mondo della statistica e della finanza. Il loro contributo è stato fonte di ispirazione e motivazione nel mio percorso di crescita accademica e professionale.

Indice

I Fondamenti delle Copule	9
1 Introduzione	11
1.1 Definizione e motivazione dello studio	11
2 Fondamenti Matematici delle Copule	13
2.1 Definizione di copula e Teorema di Sklar	13
2.1.1 Definizione di Copula	13
2.1.2 Teorema di Sklar	14
2.2 Famiglie principali di copule (Gaussiane, t-Student, Archimedee)	14
2.2.1 Famiglie di copule	14
2.2.2 Proprietà delle copule	15
II Applicazioni Finanziarie delle Copule	17
3 Copule e Gestione del Rischio Finanziario	19
3.1 Superamento della correlazione lineare nelle distribuzioni non normali	19
3.1.1 Coefficiente di Pearson	19
3.1.2 Vantaggi delle Copule rispetto alla Correlazione Lineare	20
3.2 Dipendenza di coda e impatti su Value-at-Risk (VaR) e Expected Shortfall	21
3.2.1 Tipi di dipendenza di coda	21
3.2.2 Importanza nel VaR e nell'Expected Shortfall	22
3.3 Pricing di derivati multivariati con copule e gestione del rischio	22
3.3.1 Tariffare le opzioni	22
3.3.2 Gestione dei rischi	23
3.4 Modelli di copula	23
3.4.1 Tipi di modelli di copula	23
III Preparazione dei Dati e Assunzioni	25
4 Preparazione dei Dati per la Modellazione con Copule	27
4.1 Struttura e caratteristiche dei dati	27
4.2 Pulizia e preprocessing	28
4.3 Trasformazione dei dati	29
4.4 Stima delle Distribuzioni Marginali e Trasformazione Uniforme	31
4.4.1 Stima delle Distribuzioni Marginali	31
4.4.2 Confronto tra Approcci Parametrico e Non Parametrico	33
4.5 Distribuzione dei rendimenti	35
4.6 Assunzioni generali	36
4.6.1 Conclusione sulle assunzioni	37

IV Stima dei Parametri delle Copule e Implementazione Pratica 39

5 Stime dei parametri	41
5.1 Introduzione alla Stima dei Parametri delle Copule	41
5.2 Metodi di stima dei parametri	42
5.2.1 Stima di Massima Verosimiglianza (MLE)	42
5.2.2 Implementazione Python per una copula Student t:	45
5.2.3 Metodo dei Momenti	47
5.2.4 Il Tau di Kendall	49
5.2.5 Relazioni tra parametri delle copule e misure di dipendenza	51
5.2.6 Codice Python metodo dei momenti	51
5.2.7 Stima Bayesiana	54
5.2.8 MCMC	56
5.2.9 Metodo MCMC in Python	57
5.3 Implementazione Pratica: Applicazione ai Dati del DAX	61
5.3.1 Preparazione dei dati	61
5.3.2 Stima dei Parametri tramite MLE	67
5.3.3 Stima dei parametri tramite metodo dei momenti	75
5.3.4 Stima dei parametri tramite MCMC	77
5.3.5 Scelta della copula migliore	85

V Simulazione e Ottimizzazione di un Portafoglio con Copule 91

6 Esempio simulazione portafoglio	93
6.1 Simulazione di un Portafoglio Long con Copule	93
6.1.1 Impostazione dei Parametri di Simulazione	93
6.1.2 Ottimizzazione del Portafoglio con i Nuovi Parametri	96
6.1.3 Simulazione Monte Carlo con la Copula Stimata	101
6.1.4 Visualizzazione dei risultati	105

Parte I

Fondamenti delle Copule

Capitolo 1

Introduzione

1.1 Definizione e motivazione dello studio

Le copule sono strumenti matematici che permettono di modellare e stimare la dipendenza tra diverse variabili casuali. Sono particolarmente utili in finanza, dove la dipendenza tra i rendimenti degli asset, i tassi di interesse e i tempi di default sono fattori cruciali per la valutazione del rischio e la determinazione del prezzo di strumenti finanziari complessi. L'importanza delle copule risiede nella loro capacità di separare la modellazione delle distribuzioni marginali delle singole variabili dalla modellazione della loro struttura di dipendenza.

In altre parole, invece di dover specificare una funzione di distribuzione congiunta per tutte le variabili, è possibile utilizzare una copula per combinare le distribuzioni marginali di ciascuna variabile in una distribuzione congiunta che riflette la dipendenza desiderata. Questo approccio offre grande flessibilità nella modellazione, poiché consente di scegliere le distribuzioni marginali e la copula in modo indipendente, a seconda delle caratteristiche specifiche dei dati e del problema in esame.

Ad esempio, si potrebbe utilizzare una distribuzione t di Student per modellare i rendimenti degli indici azionari, che spesso presentano code più spesse rispetto alla distribuzione normale, e quindi utilizzare una copula di Gumbel per rappresentare la dipendenza asimmetrica tra i mercati, con una maggiore probabilità di movimenti congiunti al rialzo rispetto a quelli al ribasso.

La teoria delle copule si basa sul teorema di Sklar, che afferma che ogni funzione di distribuzione congiunta può essere espressa in termini di una copula e delle distribuzioni marginali delle variabili. Il teorema di Sklar garantisce l'esistenza e l'unicità della copula nel caso di variabili casuali continue.

Esistono diverse famiglie di copule, ciascuna con proprietà specifiche in termini di dipendenza di coda, simmetria e altre caratteristiche. Alcune delle famiglie di copule più utilizzate in finanza includono la copula gaussiana, la copula t di Student, le copule Archimedee (come la copula di Gumbel, la copula di Clayton e la copula di Frank) e la copula di Marshall-Olkin. La scelta della copula più adatta dipende dalla natura del problema e dalle caratteristiche della dipendenza che si desidera modellare.

Ad esempio, la copula t di Student è spesso preferita alla copula gaussiana quando si vogliono modellare dipendenze di coda più elevate, mentre le copule Archimedee consentono di modellare diversi tipi di dipendenza asimmetrica.

Le copule trovano applicazione in diversi ambiti della finanza, tra cui:

- **Pricing di opzioni multivariate e altri derivati:** le copule possono essere utilizzate per modellare la dipendenza tra i sottostanti di un'opzione basket, un'opzione rainbow o altri derivati multi-asset, consentendo una valutazione più accurata del prezzo di questi strumenti.
- **Gestione del rischio:** le copule sono ampiamente utilizzate nella modellazione del rischio di credito, dove consentono di stimare la probabilità di default congiunta di diverse attività o controparti. Le copule sono anche utilizzate nella stima del Value at Risk (VaR) di portafogli contenenti attività con distribuzioni non normali e dipendenze complesse.
- **Calibrazione e simulazione:** la flessibilità delle copule consente di calibrare i modelli ai dati di mercato in modo efficiente e di simulare scenari di mercato realistici che tengano conto della dipendenza tra le variabili.

In sintesi, le copule rappresentano uno strumento matematico versatile e potente per la modellazione della dipendenza in finanza, con un ampio spettro di applicazioni pratiche nella valutazione del rischio, nel pricing di derivati e nella gestione del portafogli.

Capitolo 2

Fondamenti Matematici delle Copule

2.1 Definizione di copula e Teorema di Sklar

Le copule sono strumenti matematici che permettono di modellare e rappresentare la dipendenza tra variabili casuali. A differenza di misure di dipendenza tradizionali come la correlazione lineare, le copule catturano la dipendenza in modo più completo, includendo la dipendenza nelle code delle distribuzioni e non limitandosi a relazioni lineari.

Ecco una spiegazione delle formule e delle proprietà chiave:

2.1.1 Definizione di Copula

Una d -copula è una funzione $C : [0,1]^d \rightarrow [0,1]$, dove $d \geq 2$ (numero di variabili; nelle proprietà seguenti consideriamo le copule bivariate), che soddisfa le seguenti proprietà:

1. **Groundedness:**

$$C(u,0) = C(0,v) = 0, \quad \forall u, v \in [0,1]^2$$

Ciò significa che la copula è zero se una delle variabili è zero.

2. **Marginalità:**

$$C(u,1) = u, \quad C(1,v) = v, \quad \forall u, v \in [0,1]^2$$

Questa proprietà assicura che la copula sia coerente con le distribuzioni marginali, ovvero che quando una delle variabili assume il suo valore massimo, la copula coincide con la funzione di ripartizione dell'altra variabile.

3. **2-crescita (o 2-increasing):**

$$C(u_2, v_2) - C(u_2, v_1) - C(u_1, v_2) + C(u_1, v_1) \geq 0, \quad \forall u_1 \leq u_2, v_1 \leq v_2 \in [0,1]^2$$

Questa proprietà assicura che la copula sia non decrescente in entrambe le variabili, il che è necessario affinché la copula rappresenti una dipendenza positiva o negativa tra le variabili.

2.1.2 Teorema di Sklar

Questo teorema, centrale nella teoria delle copule, stabilisce un legame tra le copule e le funzioni di distribuzione congiunta.

In breve, il teorema afferma che: Data una funzione di distribuzione congiunta $F(x, y)$ con marginali $F_1(x)$ e $F_2(y)$, esiste una copula C tale che:

$$F(x, y) = C(F_1(x), F_2(y))$$

Inoltre, se $F_1(x)$ e $F_2(y)$ sono continue, allora la copula C è unica.

Conseguenze del Teorema di Sklar

- **Costruzione di modelli di dipendenza:** Permette di costruire una funzione di distribuzione congiunta a partire da distribuzioni marginali arbitrarie e da una copula che ne modella la dipendenza. Questa proprietà è particolarmente utile per modellare dati reali, dove spesso si conoscono le distribuzioni marginali ma non la struttura di dipendenza.
- **Separazione tra marginali e dipendenza:** Mette in luce come la struttura di dipendenza tra le variabili sia completamente catturata dalla copula, indipendentemente dalle distribuzioni marginali.

2.2 Famiglie principali di copule (Gaussiane, t-Student, Archimedee)

2.2.1 Famiglie di copule

Esistono diverse famiglie di copule, classificate in base alla loro struttura o ai metodi utilizzati per la loro costruzione. Di seguito, vengono elencate alcune delle principali famiglie:

- **Fréchet-Hoeffding:** Questa famiglia include le copule che rappresentano i limiti inferiore (W) e superiore (M) della dipendenza tra due variabili casuali. La copula W rappresenta la perfetta dipendenza negativa, mentre la copula M rappresenta la perfetta dipendenza positiva.
- **Cuadras-Augé:** Questa famiglia di copule è costruita come una media geometrica ponderata delle copule M e P , dove P rappresenta l'indipendenza tra le variabili.
- **Marshall-Olkin:** Questa famiglia di copule è spesso utilizzata per modellare la dipendenza tra variabili casuali che rappresentano tempi di vita.
- **Archimedee:** Queste copule sono generate da una funzione detta "generatore". Le copule Archimedee sono popolari per la loro flessibilità e la relativa facilità di utilizzo.

2.2.2 Proprietà delle copule

Le copule possiedono diverse proprietà che le rendono utili per la modellazione della dipendenza. Alcune di queste proprietà sono:

- **Invarianza rispetto a trasformazioni monotone crescenti:** Le copule sono invarianti rispetto a trasformazioni strettamente crescenti delle variabili marginali.
- **Misure di concordanza:** Diverse misure di concordanza come la rho di Spearman e la tau di Kendall possono essere espresse in termini di copule.
- **Dipendenza di coda:** Le copule possono catturare la dipendenza tra le code delle distribuzioni marginali, ovvero la tendenza delle variabili ad assumere valori estremi congiuntamente.

Possiamo quindi affermare che le copule offrono un approccio potente e flessibile per la modellazione della dipendenza tra variabili casuali. La loro capacità di separare la struttura di dipendenza dalle distribuzioni marginali, la loro invarianza rispetto a trasformazioni monotone crescenti e la loro capacità di catturare la dipendenza di coda le rendono strumenti preziosi in molte applicazioni pratiche.

Parte II

Applicazioni Finanziarie delle Copule

Capitolo 3

Copule e Gestione del Rischio Finanziario

3.1 Superamento della correlazione lineare nelle distribuzioni non normali

L'assunzione di normalità dei rendimenti, tipico di modelli come quello di Black-Scholes, è spesso disatteso nei mercati finanziari. Le serie storiche di strumenti come azioni ed obbligazioni dimostrano la presenza di code più pesanti rispetto a quanto previsto dalla distribuzione normale e la diffusione di prodotti derivati con payoff non lineari amplifica ulteriormente questo fenomeno.

In questo contesto, la **correlazione lineare**, misurata ad esempio con il coefficiente di Pearson, si dimostra uno strumento limitato. Essa cattura solo le dipendenze lineari tra le variabili, mentre le relazioni tra gli asset finanziari possono assumere forme ben più complesse. La correlazione lineare è efficace solo quando le variabili sono legate da relazioni lineari. Tuttavia, in presenza di legami non lineari, la correlazione lineare potrebbe essere fuorviante. Ad esempio, una variabile con distribuzione chi-quadrato è perfettamente correlata al suo quadrato, che ha distribuzione normale, ma la correlazione lineare non sarebbe in grado di rappresentare correttamente questa relazione.

3.1.1 Coefficiente di Pearson

Il coefficiente di correlazione lineare, noto anche come correlazione di Pearson, è una misura della dipendenza lineare tra due variabili casuali che assumono valori reali e che hanno varianza finita. È definito come la covarianza delle due variabili divisa per il prodotto delle loro deviazioni standard. Formalmente, per due variabili casuali non degeneri X e Y appartenenti a L^2 , il coefficiente di correlazione lineare ρ_{XY} è:

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}}$$

Il coefficiente di correlazione di Pearson assume valori compresi tra -1 e +1, dove:

- +1 indica una perfetta correlazione lineare positiva: all'aumentare di una variabile, l'altra aumenta in modo perfettamente lineare.
- -1 indica una perfetta correlazione lineare negativa: all'aumentare di una variabile, l'altra diminuisce in modo perfettamente lineare.
- 0 indica l'assenza di correlazione lineare: non c'è una relazione lineare tra le due variabili.

È importante sottolineare che il coefficiente di correlazione di Pearson misura solo la dipendenza lineare. Due variabili possono essere fortemente dipendenti in modo non lineare e avere comunque un coefficiente di correlazione di Pearson pari a zero.

3.1.2 Vantaggi delle Copule rispetto alla Correlazione Lineare

Le **copule**, invece, offrono un approccio più flessibile per modellare la dipendenza tra variabili casuali, anche in presenza di distribuzioni non normali. Il vantaggio principale delle copule risiede nella loro capacità di separare la struttura di dipendenza dalle distribuzioni marginali. Questo permette di combinare diverse distribuzioni marginali, capaci di cogliere la non-normalità dei rendimenti (ad esempio la distribuzione t di Student o distribuzioni asimmetriche), con una vasta gamma di copule che descrivono la struttura di dipendenza.

Il **Teorema di Sklar**, fondamento della teoria delle copule, afferma che ogni funzione di distribuzione congiunta può essere espressa in termini di distribuzioni marginali e di una copula. Questa proprietà permette di costruire modelli di dipendenza altamente flessibili, adatti a rappresentare le complesse relazioni tra gli asset finanziari. Ad esempio, è possibile utilizzare una copula gaussiana per modellare la struttura di dipendenza, pur mantenendo distribuzioni marginali non gaussiane per i singoli asset. In questo modo, si ottiene un modello in grado di catturare sia la non-normalità dei rendimenti sia le strutture di dipendenza tra gli stessi.

In definitiva, le copule rappresentano uno strumento più completo e affidabile rispetto alla correlazione lineare per modellare le dipendenze tra variabili casuali, soprattutto in presenza di distribuzioni non normali. La loro flessibilità e capacità di rappresentare accuratamente le complesse relazioni tra gli asset le rendono essenziali per una corretta valutazione del rischio, un pricing accurato e una migliore comprensione delle dinamiche dei mercati finanziari.

3.2 Dipendenza di coda e impatti su Value-at-Risk (VaR) e Expected Shortfall

La dipendenza di coda si riferisce alla tendenza di due o più variabili casuali a muoversi insieme in modo più estremo nelle code delle loro distribuzioni, rispetto a quanto previsto da una distribuzione normale con la stessa correlazione lineare. In altre parole, la dipendenza di coda misura la probabilità che si verifichino eventi estremi congiuntamente.

Sappiamo che la non normalità a livello univariato è associata al cosiddetto problema della *fat-tail*. In un contesto multivariato, il problema della *fat-tail* può essere riferito sia alle distribuzioni marginali univariate che alle distribuzioni congiunte di probabilità di grandi movimenti di mercato. Questo concetto è chiamato **tail dependence**. L'uso di funzioni copula ci permette di modellare separatamente queste due caratteristiche. Per rappresentare la dipendenza dalla coda consideriamo la probabilità che un evento con probabilità inferiore a v si verifichi nella prima variabile, dato che un evento con probabilità inferiore a v si verifica nella seconda. In concreto, ci chiediamo quale sia la probabilità di osservare, ad esempio, un crollo con probabilità inferiore di $v = 1\%$ nell'indice Nikkei 225, dato che nell'indice S&P 500 si è verificato un crollo con probabilità inferiore all'1%. Si ha:

$$\begin{aligned}\lambda(v) &\equiv \Pr(Q_{NKY} \leq v \mid Q_{SP} \leq v) \\ &= \frac{\Pr(Q_{NKY} \leq v, Q_{SP} \leq v)}{\Pr(Q_{SP} \leq v)} \\ &= \frac{C(v, v)}{v}\end{aligned}$$

3.2.1 Tipi di dipendenza di coda

Dopo che abbiamo calcolato il nostro $\lambda(v)$, possiamo dividere la dipendenza di coda in due tipi principali:

- **Dipendenza di coda inferiore:** misura la probabilità che entrambe le variabili assumano valori estremamente bassi contemporaneamente.

$$\lambda_L \equiv \lim_{v \rightarrow 0^+} \frac{C(v, v)}{v}$$

- **Dipendenza di coda superiore:** misura la probabilità che entrambe le variabili assumano valori estremamente alti contemporaneamente.

$$\begin{aligned}\lambda_U &= \lim_{v \rightarrow 1^-} \lambda_v \equiv \lim_{v \rightarrow 1^-} \frac{\Pr(Q_{NKY} > v, Q_{SP} > v)}{\Pr(Q_{SP} > v)} \\ &= \lim_{v \rightarrow 1^-} \frac{1 - 2v + C(v, v)}{1 - v}\end{aligned}$$

3.2.2 Importanza nel VaR e nell'Expected Shortfall

La dipendenza di coda ha un impatto significativo sul calcolo del **VaR** e dell'**Expected Shortfall**, due misure di rischio ampiamente utilizzate nella gestione del rischio finanziario.

- **VaR (Value at Risk)**: rappresenta la perdita massima stimata che un portafoglio potrebbe subire in un determinato orizzonte temporale e con un dato livello di confidenza.
- **Expected Shortfall**: rappresenta la perdita media attesa in caso di superamento del VaR.

In presenza di dipendenza di coda, il VaR e l'Expected Shortfall calcolati assumendo una distribuzione normale tendono a sottostimare il rischio effettivo del portafoglio. Questo perché la distribuzione normale non riesce a catturare adeguatamente la probabilità di eventi estremi congiunti. Mentre utilizzando le copule per modellare la dipendenza tra gli asset di un portafoglio, è possibile ottenere una stima più accurata del VaR e dell'Expected Shortfall, tenendo conto della probabilità di eventi estremi congiunti. Quindi grazie alle copule è possibile una misura più accurata del rischio di portafoglio e si possono prendere decisioni più consapevoli.

3.3 Pricing di derivati multivariati con copule e gestione del rischio

3.3.1 Tariffare le opzioni

La valutazione di opzioni multivariate, come le opzioni basket o rainbow, che dipendono da più attività sottostanti, rappresenta una sfida significativa in finanza. Le copule forniscono un potente strumento per affrontare questo problema.

- In sostanza, una copula viene utilizzata per costruire la distribuzione congiunta dei prezzi delle attività sottostanti alla scadenza.
- Questa distribuzione viene quindi utilizzata per calcolare il valore atteso del payoff dell'opzione in base a tutti i possibili risultati dei prezzi delle attività sottostanti.
- Attualizzando questo valore atteso al tasso privo di rischio, si ottiene il prezzo dell'opzione.

Nei mercati incompleti, dove non esiste una misura di probabilità unica priva di arbitraggio, le copule sono fondamentali per derivare strategie di super-replicazione. Queste strategie mirano a creare un portafoglio di attività negoziabili che replichi il payoff dell'opzione in tutte le possibili situazioni future, garantendo così l'assenza di opportunità di arbitraggio. Le copule consentono di determinare i limiti superiori e inferiori del prezzo dell'opzione in base alle diverse ipotesi sulla struttura di dipendenza tra le attività sottostanti. Mostriamo ora degli esempi:

- **Opzioni arcobaleno**: queste opzioni, che dipendono dal minimo o dal massimo di un paniere di attività, possono essere valutate utilizzando le copule per catturare la dipendenza tra i rendimenti delle attività. Le fonti dimostrano come le copule possano essere utilizzate per derivare strategie di super-replicazione per le opzioni arcobaleno, fornendo limiti superiori e inferiori al prezzo.
- **Opzioni barriera**: per le opzioni in cui l'esercizio è condizionato al fatto che il prezzo dell'attività sottostante raggiunga o meno una determinata barriera, le copule possono essere utilizzate per modellare la dipendenza tra il processo del prezzo dell'attività e l'evento di attivazione della barriera.

3.3.2 Gestione dei rischi

Le copule possono essere utilizzate per modellare la dipendenza tra diversi tipi di rischio, come rischio di mercato, rischio di credito e rischio operativo. Ciò è particolarmente utile per le istituzioni finanziarie che sono esposte a più tipi di rischio, in quanto consente loro di valutare il rischio complessivo a cui sono esposte. Ad esempio, una banca può utilizzare le copule per modellare la dipendenza tra le insolvenze sui prestiti e i movimenti dei tassi di interesse, consentendo loro di valutare il rischio del proprio portafoglio prestiti in diversi scenari economici.

In particolare, nella gestione del rischio di credito, le copule vengono utilizzate nella valutazione di strumenti di debito strutturati come le obbligazioni garantite da crediti (CDO). Le CDO sono obbligazioni garantite da un pool di attività sottostanti, come mutui o prestiti alle imprese. Il rischio di credito di una CDO dipende dalla dipendenza tra le insolvenze delle attività sottostanti. Le copule forniscono un modo flessibile per modellare questa dipendenza, consentendo agli investitori di valutare il rischio e il rendimento delle CDO in modo più accurato.

3.4 Modelli di copula

3.4.1 Tipi di modelli di copula

- **Copula gaussiana:** descrive la dipendenza tra variabili casuali utilizzando la distribuzione normale multivariata. Non è in grado di catturare la dipendenza dalla coda. È definita come la funzione di distribuzione congiunta di un vettore normale multivariato standard, dove ogni variabile marginale è stata trasformata nella sua forma univariata standard utilizzando la funzione di distribuzione normale standard inversa.

$$C_R^{Ga}(u, v) = \Phi_R(\Phi^{-1}(u), \Phi^{-1}(v))$$

$$= \int_{-\infty}^{\Phi^{-1}(u)} \int_{-\infty}^{\Phi^{-1}(v)} \frac{1}{2\pi\sqrt{1-\rho_{XY}^2}} \exp\left(\frac{2\rho_{XY}st - s^2 - t^2}{2(1-\rho_{XY}^2)}\right) ds dt$$

- **Copula t di Student:** Questa copula può catturare la dipendenza dalla coda e viene spesso utilizzata per modellare i rendimenti degli asset che mostrano code pesanti. La copula t di Student bivariata è data dalla seguente formula:

$$C_t(u, v) = \int_{-\infty}^{t_{\nu}^{-1}(u)} \int_{-\infty}^{t_{\nu}^{-1}(v)} \frac{\Gamma(\frac{\nu+2}{2})}{\Gamma(\frac{\nu}{2}) \pi \nu \sqrt{1-\rho^2}} \left(1 + \frac{x^2 - 2\rho xy + y^2}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}} dx dy$$

dove t_{ν}^{-1} è l'inversa della funzione di distribuzione t di Student univariata con ν gradi di libertà e ρ è il coefficiente di correlazione.

- **Copula di Clayton:** questa copula mostra una forte dipendenza dalla coda inferiore, il che significa che le variabili hanno maggiori probabilità di assumere insieme valori estremi bassi. È esaustiva e fornisce la copula del prodotto se $\alpha = 0$, il limite inferiore di Fréchet $\max(v + z - 1, 0)$ quando $\alpha = -1$ e quello superiore per $\alpha \rightarrow +\infty$. È definita dalla seguente formula:

$$C(v, z) = \max[(v^{-\alpha} + z^{-\alpha} - 1)^{-\frac{1}{\alpha}}, 0]$$

- **Copula di Gumbel:** questa copula mostra una forte dipendenza dalla coda superiore, indicando che le variabili hanno maggiori probabilità di assumere insieme valori estremi elevati. Fornisce la copula del prodotto se $\alpha = 1$ e il limite superiore di Fréchet $\min(v, z)$ per $\alpha \rightarrow +\infty$. È definita dalla seguente formula:

$$C(v, z) = \exp \left(-[(-\ln v)^\alpha + (-\ln z)^\alpha]^{\frac{1}{\alpha}} \right)$$

- **Copula di Frank:** questa copula è una copula simmetrica che può catturare sia la dipendenza dalla coda superiore che quella dalla coda inferiore. Si riduce alla copula del prodotto se $\alpha = 0$ e raggiunge i limiti inferiori e superiori di Fréchet rispettivamente per $\alpha \rightarrow -\infty$ e $\alpha \rightarrow +\infty$. È definita dalla seguente formula:

$$C(v, z) = -\frac{1}{\alpha} \ln \left(1 + \frac{(e^{-\alpha v} - 1)(e^{-\alpha z} - 1)}{e^{-\alpha} - 1} \right)$$

Parte III

Preparazione dei Dati e Assunzioni

Capitolo 4

Preparazione dei Dati per la Modellazione con Copule

4.1 Struttura e caratteristiche dei dati

Il seguente dataset contiene dati storici sul DAX, che rappresentano prezzi o indici di mercato rilevanti. La raccolta dati fornisce i valori di Open, High, Low, Close partendo dal giorno 02/01/2020 ore 01:15:00 e terminando con il giorno 03/03/2022 ore 09:14:00. La raccolta e la gestione adeguata di tali dati sono fondamentali per analizzare le dipendenze tra strumenti finanziari tramite modelli di copula. Tramite questi dati calcoleremo i rendimenti giornalieri, un passaggio necessario per la modellazione delle dipendenze.

DateTime	Open	High	Low	Close
02/01/2020 01:15:00 +01:00	13174	13194.5	13171.5	13177.5
02/01/2020 01:16:00 +01:00	13177	13185	13177	13180.5
02/01/2020 01:17:00 +01:00	13180.5	13183	13179	13181.5
02/01/2020 01:18:00 +01:00	13181.5	13182	13180.5	13182
02/01/2020 01:19:00 +01:00	13182	13183	13180.5	13181.5
...
03/03/2022 09:10:00 +01:00	14019	14031	14010	14013
03/03/2022 09:11:00 +01:00	14013	14019	14010	14000
03/03/2022 09:12:00 +01:00	13999	14013	13996	14006
03/03/2022 09:13:00 +01:00	14007	14015	13995	14009
03/03/2022 09:14:00 +01:00	14009	14020	14009	14020

Tabella 4.1. 616397 osservazioni raccolte di open, high, low and close

4.2 Pulizia e preprocessing

I dati finanziari spesso includono anomalie come valori mancanti o outlier che devono essere gestiti prima dell'analisi. Saranno implementate le seguenti tecniche di pulizia dei dati:

- **Gestione dei valori mancanti:** rimuovere eventuali righe con valori mancanti per evitare distorsioni.
- **Gestione degli outlier:** utilizzare tecniche di filtraggio per identificare ed eliminare gli outlier, assicurando che l'analisi si concentri sui valori centrali più rappresentativi.

Outlier: nel contesto dei dati azionari, un outlier (o valore anomalo) è un'osservazione che si discosta significativamente dalla norma o dalla tendenza generale del dataset. In altre parole, si tratta di un dato che è molto diverso rispetto agli altri valori presenti nel campione.

Il motivo per cui vanno eliminati è che possono distorcere le analisi statistiche, poiché influenzano la media, la deviazione standard e altre misure di dispersione dei dati.

Ecco alcune righe di codice utilizzate per “pulire” i dati:

```
import pandas as pd

# Load the data
data = pd.read_csv('DAX_3Y-1M.csv', index_col='DateTime',
parse_dates=True)

# Drop rows with missing values
data.dropna()

# Verifica e gestione degli outlier tramite interquartile
range (IQR)
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
filtered_data = data[~((data < (Q1 - 1.5 * IQR)) |
(data > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Circa 6000 righe di dati sono state eliminate da questo processo.

4.3 Trasformazione dei dati

Dopo la pulizia, è necessario trasformare i dati per ottenere una scala uniforme. Poiché i modelli di copula richiedono margini uniformi, trasformeremo i dati in rendimenti logaritmici per ottenere stazionarietà e calcoleremo i punteggi standardizzati:

Stazionarietà dei rendimenti logaritmici

I rendimenti logaritmici sono generalmente considerati stazionari per diverse ragioni teoriche ed empiriche:

1. Rimozione del trend

I prezzi finanziari tendono a seguire un trend crescente nel lungo periodo, il che li rende non stazionari. La differenziazione logaritmica rimuove questo trend, trasformando i prezzi in una serie di rendimenti logaritmici che risultano più stabili.

2. Proprietà statistiche

- La **media** dei rendimenti logaritmici tende a essere costante nel tempo.
- La **varianza** è più stabile rispetto ai rendimenti lineari, anche se può mostrare clustering di volatilità (ad esempio nei modelli ARCH/GARCH).
- L'**autocorrelazione** decade rapidamente verso zero.

3. Evidenza empirica

Vari test statistici, come il **test di Dickey-Fuller (ADF)** e il **test KPSS**, di solito confermano la stazionarietà dei rendimenti logaritmici.

4. Modelli stocastici e teoria finanziaria

- Il **moto browniano geometrico**, alla base della modellizzazione dei prezzi degli asset, implica che i rendimenti logaritmici siano stazionari.

5. Additività

I rendimenti logaritmici sono **additivi nel tempo**, a differenza dei rendimenti percentuali, facilitando analisi multi-periodo e semplificando la modellizzazione.

Ecco alcune righe di codice utilizzate per “trasformare” i dati:

```
import numpy as np

# Calcolo dei rendimenti logaritmici
log_returns = np.log(filtered_data / filtered_data.shift(1)).dropna()

# Standardizzazione dei dati
# (sottraggo la media e divido per la deviazione standard)
standardized_data = (log_returns - log_returns.mean()) / log_returns.std()
```

DateTime	Open	High	Low	Close
02/01/2020 01:16:00 +01:00	0.513283	-1.507703	0.853297	0.498567
02/01/2020 01:17:00 +01:00	0.598720	-0.555875	0.310070	0.166015
02/01/2020 01:18:00 +01:00	0.170871	0.079182	0.154914	0.028292
02/01/2020 01:19:00 +01:00	0.085317	0.158563	0.077349	-0.083337
02/01/2020 01:20:00 +01:00	-0.085772	0.237926	0.154896	0.498415
...
03/03/2022 09:10:00 +01:00	-1.447601	-0.000208	-0.437935	-1.250808
03/03/2022 09:11:00 +01:00	-0.965660	-1.791008	-1.459979	-2.033947
03/03/2022 09:12:00 +01:00	-2.254511	-0.149511	-0.146421	0.938266
03/03/2022 09:13:00 +01:00	1.288211	-0.448180	-0.584449	0.469065
03/03/2022 09:14:00 +01:00	0.321767	0.746358	2.043913	1.719639

Tabella 4.2. Rendimenti logaritmici standardizzati

4.4 Stima delle Distribuzioni Marginali e Trasformazione Uniforme

Nella modellizzazione delle dipendenze tra variabili finanziarie attraverso copule, è fondamentale trasformare i dati in un dominio uniforme $[0,1]$, poiché le copule descrivono esclusivamente la struttura di dipendenza, indipendentemente dalle distribuzioni marginali. A tal fine, si possono adottare due approcci principali:

- **Metodo empirico**, che utilizza la funzione di ripartizione empirica (*ECDF*).
- **Metodo parametrico**, che assume una forma funzionale per le distribuzioni marginali e stima i parametri corrispondenti.

In questo studio, è stato adottato un approccio parametrico per stimare le distribuzioni marginali e successivamente applicare la trasformazione inversa a valori uniformi tramite la funzione di ripartizione cumulativa (*CDF*).

4.4.1 Stima delle Distribuzioni Marginali

Definizione dei Modelli

Per stimare le distribuzioni marginali dei rendimenti delle open e delle close, sono stati considerati diversi modelli statistici candidati:

- **Distribuzione Normale** $\mathcal{N}(\mu, \sigma)$, caratterizzata dalla media μ e dalla deviazione standard σ .
- **Distribuzione t-Student** $t_\nu(\mu, \sigma)$, che introduce un ulteriore parametro ν per modellare code più pesanti.

Stima Parametrica

Per ogni serie di dati, i parametri delle distribuzioni candidate sono stati stimati mediante il metodo della massima verosimiglianza (*MLE - Maximum Likelihood Estimation*). Ad esempio, per la distribuzione normale:

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu, \sigma} \sum_{i=1}^n \log f_{\mathcal{N}}(x_i | \mu, \sigma) \quad (4.1)$$

mentre per la distribuzione t-Student:

$$\hat{\nu}, \hat{\mu}, \hat{\sigma} = \arg \max_{\nu, \mu, \sigma} \sum_{i=1}^n \log f_t(x_i | \nu, \mu, \sigma) \quad (4.2)$$

dove $f_{\mathcal{N}}(x)$ e $f_t(x)$ sono le densità delle distribuzioni normale e t-Student, rispettivamente.

Selezione del Miglior Modello

Per determinare quale distribuzione meglio descrive i dati, è stato utilizzato il criterio di **Akaike Information Criterion (AIC)**, definito come:

$$AIC = -2 \log L + 2k \quad (4.3)$$

dove:

- L è la verosimiglianza massima,
- k è il numero di parametri stimati.

Il modello con il valore di *AIC* più basso è stato selezionato per ogni serie.

Implementazione Python

```

# Stima parametrica delle distribuzioni marginali
print(".1 Stima delle distribuzioni marginali")
print("-" * 50)

# Definizione dei modelli candidati
margin_models = ['norm', 't', 'skewt', 'garch']
margin_results = {}

for col, data in [('vol1', vol_returns['vol1']),
                  ('vol2', vol_returns['vol2'])]:
    print(f"\nStima marginali per {col}:")
    model_fits = {}

    # Normale
    loc, scale = stats.norm.fit(data)
    loglik_norm = np.sum(stats.norm.logpdf(data, loc, scale))
    aic_norm = -2 * loglik_norm + 2 * 2 # 2 parametri
    model_fits['norm'] = {'params': (loc, scale), 'loglik': loglik_norm, 'aic': aic_norm}
    print(f"Normale: $\mu$={loc:.4f}, $\theta$={scale:.4f}, AIC={aic_norm:.2f}")

    # t-Student
    params = stats.t.fit(data)
    loglik_t = np.sum(stats.t.logpdf(data, *params))
    aic_t = -2 * loglik_t + 2 * 3 # 3 parametri
    model_fits['t'] = {'params': params, 'loglik': loglik_t, 'aic': aic_t}
    print(f"t-Student: df={params[0]:.4f}, $\mu$={params[1]:.4f}, $\sigma$={params[2]:.4f}, AIC={aic_t:.2f}")

    # Seleziona il modello migliore
    best_model = min(model_fits.items(), key=lambda x: x[1]['aic'])[0]
    margin_results[col] = {'best_model': best_model, 'fits': model_fits}
    print(f"Miglior modello per {col}: {best_model}")

```

Tabella 4.3. Stima delle distribuzioni marginali

Modello	Parametri	AIC	Miglior Modello
Open			
Normale	$\mu = 0.0056, \sigma = 0.0050$	-366834.42	
t-Student	$df = 1.8257, \mu = 0.0042, \sigma = 0.0018$	-410122.46	t-Student
Close			
Normale	$\mu = 0.0058, \sigma = 0.0052$	-364325.59	
t-Student	$df = 1.8538, \mu = 0.0043, \sigma = 0.0018$	-406841.51	t-Student

Trasformazione Uniforme con la Funzione di Ripartizione Cumulativa

Una volta stimati i parametri, i rendimenti originali sono stati trasformati in variabili uniformi [0,1] utilizzando la funzione di ripartizione cumulativa (*CDF*) della distribuzione scelta. ecco il codice:

```
# Trasformazione usando le marginali parametriche stimate
u1_param = []
u2_param = []

# Trasforma open usando il miglior modello
if margin_results['vol1']['best_model'] == 'norm':
    params = margin_results['vol1']['fits']['norm']['params']
    u1_param = stats.norm.cdf(vol_returns['vol1'], *params)
elif margin_results['vol1']['best_model'] == 't':
    params = margin_results['vol1']['fits']['t']['params']
    u1_param = stats.t.cdf(vol_returns['vol1'], *params)

# Trasforma close usando il miglior modello
if margin_results['vol2']['best_model'] == 'norm':
    params = margin_results['vol2']['fits']['norm']['params']
    u2_param = stats.norm.cdf(vol_returns['vol2'], *params)
elif margin_results['vol2']['best_model'] == 't':
    params = margin_results['vol2']['fits']['t']['params']
    u2_param = stats.t.cdf(vol_returns['vol2'], *params)
```

4.4.2 Confronto tra Approcci Parametrico e Non Parametrico

Per verificare l'efficacia della trasformazione, sono stati confrontati i risultati ottenuti con l'approccio empirico (basato su *rank transformation*) e l'approccio parametrico. Il confronto è stato realizzato graficamente mediante **scatter plot** delle coppie (u_1, u_2):

1. **Approccio empirico (ECDF):** utilizza la trasformazione basata sui ranghi:

$$u_i = \frac{\text{rank}(x_i)}{n + 1}$$

```
u1 = stats.rankdata(vol_returns['vol1']) / (len(vol_returns) + 1)
u2 = stats.rankdata(vol_returns['vol2']) / (len(vol_returns) + 1)
```

2. **Approccio parametrico:** utilizza la funzione di ripartizione cumulativa del modello selezionato.

Questa comparazione ha permesso di valutare visivamente se l'approccio parametrico mantiene la struttura di dipendenza tra le variabili in modo coerente rispetto all'approccio non parametrico.

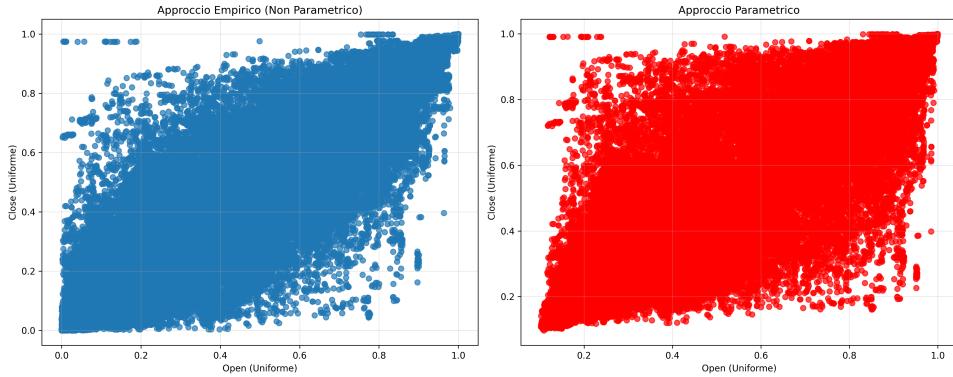


Figura 4.1. Empirico sinistra, Parametrico destra

Scelta Finale: Normalizzazione Empirica

Dopo aver confrontato le due metodologie, si è scelto di adottare la **funzione di ripartizione empirica (ECDF)** per la trasformazione uniforme. Questa scelta è motivata da:

- **Maggiore flessibilità:** non richiede assunzioni sulla distribuzione dei dati.
- **Adattabilità ai dati reali:** evita problemi dovuti alla cattiva specificazione dei modelli marginali.
- **Miglior gestione delle code:** mantiene la struttura di dipendenza nei dati senza introdurre distorsioni dovute a scelte parametriche.

L'approccio empirico garantisce una normalizzazione più robusta e affidabile per l'applicazione delle copule, permettendo di modellare correttamente la struttura di dipendenza tra le variabili.

4.5 Distribuzione dei rendimenti

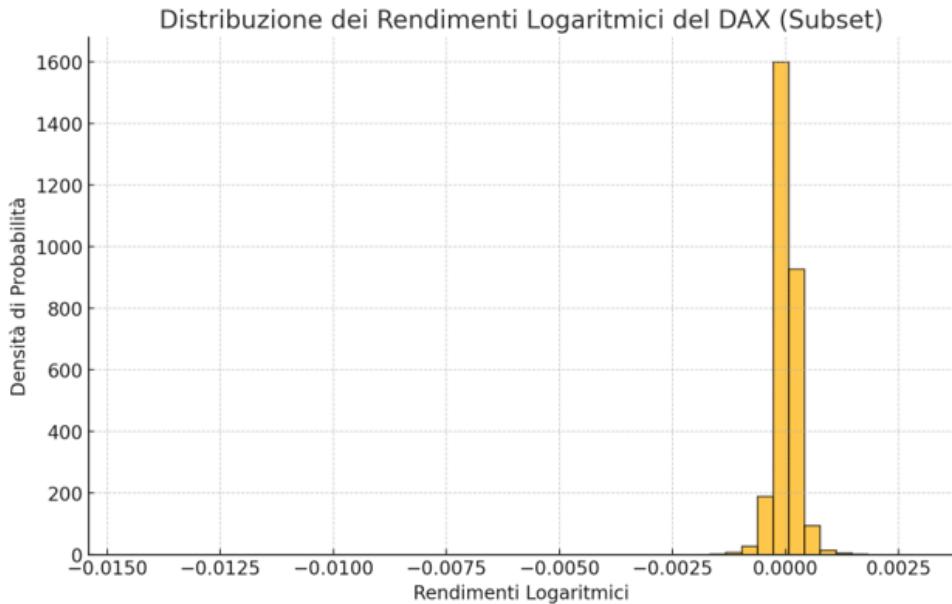


Figura 4.2. Distribuzione dei Rendimenti Logaritmici del DAX (Subset)

Ecco la distribuzione dei rendimenti logaritmici del DAX, calcolata su un subset dei dati disponibili. La distribuzione mostra la tipica forma a campana, con alcune code più pesanti, suggerendo la presenza di eventi estremi più frequenti rispetto a una normale distribuzione Gaussiana. Questa caratteristica supporta l’idea di utilizzare copule come la Student-t, che meglio cattura queste dipendenze nelle code.

Di seguito la matrice di correlazione tra i rendimenti logaritmici delle diverse colonne di prezzo (Open, High, Low, Close) sempre del subset.

	Open	High	Low	Close
Open	1.0000	0.4716	0.4890	0.3141
High	0.4716	1.0000	0.3353	0.5610
Low	0.4890	0.3353	1.0000	0.5142
Close	0.3141	0.5610	0.5142	1.0000

Tabella 4.4. Matrice di correlazione tra Open, High, Low e Close

4.6 Assunzioni generali

Considerando i dati analizzati e i risultati ottenuti dalle elaborazioni, possiamo formulare le seguenti assunzioni per l'applicazione dei modelli di copula:

1. Uniformità marginale

Una delle assunzioni principali per applicare i modelli di copula è che le variabili marginali siano uniformi. Nel nostro caso, i rendimenti logaritmici calcolati per le diverse variabili (Open, High, Low, Close) sono stati trasformati in modo tale da poter essere considerati approssimativamente stazionari, ma non sono ancora stati resi uniformi. Per l'applicazione delle copule, sarà necessario trasformare i rendimenti normalizzati in una distribuzione uniforme sull'intervallo [0,1], ad esempio usando la funzione di distribuzione cumulativa empirica. Questo passaggio garantisce che le dipendenze tra variabili siano modellate correttamente senza distorsioni derivanti dalle distribuzioni marginali.

2. Stazionarietà dei dati

Per applicare correttamente i modelli di copula, è necessario che le serie temporali siano stazionarie, ovvero che le proprietà statistiche come media e varianza siano costanti nel tempo. Abbiamo trasformato i prezzi in rendimenti logaritmici per ottenere una serie più stazionaria rispetto ai dati originali di prezzo. Tuttavia, è possibile che ci siano ancora componenti non stazionarie, come trend residui o stagionalità, che potrebbero influire sui risultati. La verifica e il trattamento di eventuali residui non stazionari sono fondamentali per garantire la validità dei modelli di copula.

3. Struttura di dipendenza

L'analisi dei rendimenti logaritmici ha mostrato una correlazione positiva tra le variabili, sebbene con valori differenti per ciascuna coppia (ad esempio, correlazione relativamente più alta tra High e Close, e più bassa tra Open e Close). Questa osservazione implica che esiste una struttura di dipendenza tra le variabili di prezzo, che va oltre la correlazione lineare. Le copule ci permetteranno di catturare meglio questa dipendenza, soprattutto nei casi in cui le correlazioni sono condizionate da situazioni estreme (code pesanti).

4. Dipendenze di coda

Osservando la distribuzione dei rendimenti logaritmici, è evidente che la distribuzione presenta code più pesanti rispetto a una normale distribuzione Gaussiana. Questo suggerisce una maggiore probabilità di eventi estremi (sia positivi che negativi), specialmente durante periodi di volatilità del mercato. Pertanto, è ragionevole assumere che le variabili presentino dipendenza di coda, rendendo modelli come la copula di Student-t o la copula di Clayton adatti per catturare le correlazioni nelle code inferiori, particolarmente durante i ribassi di mercato.

5. Non-normalità delle distribuzioni marginali

I rendimenti logaritmici non seguono una distribuzione normale; piuttosto, mostrano asimmetria e code più pesanti. Sebbene la copula Gaussiana possa essere utilizzata per una prima analisi, è preferibile utilizzare copule come la Student-t per gestire deviazioni dalla normalità, particolarmente utili per modellare le code e le correlazioni durante gli eventi estremi.

6. Condizioni di diversificazione

La copula di Frank potrebbe essere adatta per modellare le dipendenze tra le variabili che non mostrano comportamenti particolarmente forti nelle code (ovvero, dipendenze moderate e stabili). Tuttavia, i dati indicano la presenza di code pesanti, quindi questa copula potrebbe essere utilizzata solo come confronto con modelli che catturano meglio le dipendenze estreme.

4.6.1 Conclusione sulle assunzioni

Uniformità e stazionarietà sono requisiti fondamentali per l'applicazione dei modelli di copula. I dati sono stati trasformati per soddisfare parzialmente queste assunzioni.

Dipendenze di coda e non-normalità suggeriscono l'uso di copule robuste come la Student-t o modelli asimmetrici come la Clayton o la Gumbel per catturare meglio le relazioni tra variabili durante condizioni di stress di mercato.

L'asimmetria delle correlazioni evidenziata dalla matrice di correlazione indica la necessità di copule che possano gestire differenti tipi di dipendenze nelle code.

Queste assunzioni ci permettono di scegliere il modello di copula più appropriato per analizzare le dipendenze tra i rendimenti del DAX e comprendere meglio il comportamento del mercato in diverse condizioni economiche.

Parte IV

Stima dei Parametri delle Copule e Implementazione Pratica

Capitolo 5

Stime dei parametri

5.1 Introduzione alla Stima dei Parametri delle Copule

La stima dei parametri delle copule è cruciale per descrivere con precisione il tipo e il grado di dipendenza tra variabili. In particolare, in finanza:

- **Gestione del rischio:** La corretta modellazione della dipendenza è essenziale per calcolare indicatori come il Value-at-Risk (VaR) e il Conditional VaR (CVaR), nonché per valutare il rischio di portafoglio in condizioni estreme.
- **Ottimizzazione del portafoglio:** La conoscenza della dipendenza consente di costruire portafogli ottimizzati, tenendo conto delle correlazioni non lineari tra asset.
- **Eventi estremi:** La modellazione della *tail dependence* (dipendenza nelle code) permette di catturare correttamente la probabilità di eventi congiunti estremi, come crisi finanziarie o fallimenti simultanei di istituzioni.

L'importanza della stima risiede nella capacità di rappresentare accuratamente la struttura di dipendenza osservata nei dati, migliorando l'affidabilità e la precisione dei modelli finanziari.

Per stimare i parametri delle copule, esistono diversi metodi, ciascuno con vantaggi e limitazioni:

1. **Metodo della Massima Verosimiglianza (MLE - Maximum Likelihood Estimation):** Massimizzando la funzione di verosimiglianza costruita sui dati, si ottengono degli stimatori per i vari parametri.
2. **Metodo dei Momenti:** Confronta i momenti osservati con quelli teorici per stimare i parametri della copula.
3. **Metodi Bayesiani:** Utilizzano distribuzioni a priori per stimare i parametri delle copule e aggiornano le stime con i dati osservati.

La stima dei parametri delle copule rappresenta un passaggio fondamentale nella modellazione della dipendenza tra variabili finanziarie. La scelta del metodo più appropriato dipende dalla complessità del modello, dalla disponibilità di dati e dai vincoli computazionali. L'utilizzo corretto delle copule migliora significativamente l'accuratezza dei modelli finanziari, con importanti applicazioni nella gestione del rischio, nella costruzione di portafogli e nella previsione di eventi estremi.

Nelle prossime pagine mostreremo un'analisi dettagliata dei vari metodi di stima (anche un implementazione Python) e un'applicazione pratica utilizzando i dati del DAX.

5.2 Metodi di stima dei parametri

5.2.1 Stima di Massima Verosimiglianza (MLE)

Il Metodo della Massima Verosimiglianza (MLE) è una delle tecniche più utilizzate per la stima dei parametri delle copule grazie alla sua efficienza e alla solidità teorica. Questo metodo si basa sul principio di determinare i parametri che massimizzano la probabilità (o verosimiglianza) dei dati osservati sotto il modello scelto.

Procedura:

1. Supponiamo di avere n osservazioni $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$, con $i = 1, \dots, n$, da una distribuzione congiunta $F(x; \theta)$, dove θ è il vettore dei parametri della copula.
2. La funzione di verosimiglianza è costruita come:

$$L(\theta; x_1, \dots, x_n) = \prod_{i=1}^n f(x_i; \theta)$$

dove $f(x_i; \theta)$ è la densità congiunta.

3. Usando il Teorema di Sklar, la densità congiunta può essere scritta come:

$$f(x_i; \theta) = c(F_1(x_{i1}), \dots, F_d(x_{id}); \theta) \prod_{j=1}^d f_j(x_{ij})$$

dove $f_j(x_{ij})$ sono le densità marginali e $c(\cdot; \theta)$ è la densità della copula.

4. Se le distribuzioni marginali sono note, la funzione di verosimiglianza dipende solo dalla copula:

$$L(\theta) = \prod_{i=1}^n c(F_1(x_{i1}), \dots, F_d(x_{id}); \theta)$$

5. Se le marginali non sono note, si utilizza la Pseudo MLE, stimando prima le distribuzioni marginali $F_j(x)$ e poi massimizzando la verosimiglianza della copula sulle pseudo-osservazioni:

$$L^*(\theta) = \prod_{i=1}^n c(U_{i1}, \dots, U_{id}; \theta), \quad \text{con } U_{ij} = F_j(x_{ij}).$$

6. La stima $\hat{\theta}$ dei parametri si ottiene massimizzando il logaritmo della funzione di verosimiglianza:

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^n \log c(F_1(x_{i1}), \dots, F_d(x_{id}); \theta)$$

Dopo aver spiegato come funziona il metodo matematicamente, forniamo dei motivi per cui l'MLE è comunemente utilizzato per le copule.

Vantaggi del MLE:**• Efficienza statistica:**

L'MLE produce stime consistenti, non distorte e asintoticamente efficienti sotto ipotesi regolari, garantendo la massima accuratezza possibile per grandi campioni.

• Flessibilità:

L'MLE è adatta sia a copule con marginali note che a copule con marginali sconosciute (in quest'ultimo caso, si utilizza la variante del *Pseudo-MLE*).

• Compatibilità con il teorema di Sklar:

Grazie alla separazione tra marginali e dipendenza, l'MLE consente di stimare parametri che riflettono esclusivamente la struttura di dipendenza, riducendo l'influenza delle marginali.

• Applicazioni pratiche:

In ambito finanziario, l'MLE è particolarmente efficace per stimare copule complesse (es. copule *t* o Archimedee) che modellano fenomeni come la *tail dependence* (dipendenza nelle code).

• Supporto computazionale:

L'MLE è ben supportata da software statistici e librerie numeriche (ad esempio, Python, R, MATLAB), rendendola una scelta praticabile anche per problemi reali con dataset di grandi dimensioni.

Il metodo della Massima Verosimiglianza è una tecnica robusta e versatile per la stima dei parametri delle copule, garantendo precisione ed efficienza nella modellazione della dipendenza. La sua applicazione, unita alla separazione tra marginali e struttura di dipendenza, ne fa uno strumento indispensabile nella modellazione finanziaria e in molte altre discipline quantitative.

Vediamo anche qualche limitazione di questo metodo.

Limitazioni del MLE:

- **Complessità Computazionale:**

- La funzione di verosimiglianza può diventare complessa da calcolare, specialmente per modelli con molte variabili o copule con parametri complessi.
- Richiede spesso ottimizzazione numerica iterativa (es. Newton-Raphson, metodi stocastici), che può essere lenta o soggetta a problemi di convergenza.

- **Sensibilità ai Dati:**

- L'MLE assume che il modello scelto sia corretto. In ambito finanziario, però, i dati reali possono violare le ipotesi standard (es. non normalità delle marginali, dati mancanti, errori di misurazione).
- Le stime possono risultare inefficaci se il modello non rappresenta bene i dati.

- **Dipendenza dalle Condizioni Asintotiche:**

- L'efficienza dell'MLE è garantita solo per campioni sufficientemente grandi. Nei dataset finanziari con pochi dati (es. eventi estremi rari), l'MLE può essere instabile.

- **Problemi di Robustezza:**

- L'MLE è sensibile agli *outlier*, che sono comuni nei dati finanziari (ad esempio, picchi improvvisi di volatilità).
- Per mitigare questo problema, possono essere necessari metodi alternativi.

- **Vincoli Numerici e Positività:**

- Nel caso di copule multivariate, come la copula Gaussiana o t-Student, la matrice di correlazione deve essere positiva definita. Questo perché se la matrice non fosse positiva definita, implicherebbe che una combinazione lineare delle variabili potrebbe avere varianza negativa o nulla, una condizione non ammissibile in statistica.
- L'imposizione di vincoli aggiunge complessità computazionale.

Considerazioni finali

L'MLE è uno strumento potente e teoricamente solido per la stima dei parametri in contesti finanziari. Tuttavia, la sua applicazione pratica deve tenere conto delle caratteristiche specifiche dei dati finanziari e della complessità computazionale del modello scelto. In alternativa, quando i limiti dell'MLE diventano problematici, possono essere considerati altri approcci, come il metodo dei momenti, il metodo bayesiano o il metodo pseudo-MLE.

5.2.2 Implementazione Python per una copula Student t:

Mostriamo ora un'implementazione in Python di questo metodo.

Iniziamo mostrando le librerie necessarie per l'esecuzione dello script.

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import scipy.optimize as optimize
import matplotlib.pyplot as plt
from scipy.special import gamma
import os
```

Successivamente sarà necessario un dataset già normalizzato, ovvero dove ogni valore è stato distribuito in [0,1] tramite metodi parametrici o approcci empirici. Di seguito viene utilizzato *uniform_data* i cui valori sono stati ottenuti tramite norm.cdf(standardized_data). In questo esempio utilizzeremo una copula Student t bivariata, le cui variabili saranno i valori *Open* e i valori *Close* che andremo ad indicare rispettivamente con u e v .

```
# Selezione di due colonne per la copula
u, v = uniform_data['Open'].values, uniform_data['Close'].values
```

Implementiamo direttamente la Copula Student-t ricordando che la sua densità è:

$$c(u, v; \rho, \nu) = \frac{\Gamma\left(\frac{\nu+2}{2}\right) \Gamma\left(\frac{\nu}{2}\right)^{-1} \left(1 + \frac{x^2+y^2-2\rho xy}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}}}{\sqrt{1-\rho^2} \cdot \Gamma\left(\frac{\nu+1}{2}\right)^2 \Gamma\left(\frac{\nu}{2}\right)^{-2} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} \left(1 + \frac{y^2}{\nu}\right)^{-\frac{\nu+1}{2}}}$$

dove $x = t_\nu^{-1}(u)$ e $y = t_\nu^{-1}(v)$.

```
def t_copula_pdf(u, v, rho, df):
    x = stats.t.ppf(u, df)
    y = stats.t.ppf(v, df)

    # Protezione contro overflow/underflow
    try:

        numerator = gamma((df + 2) / 2) * gamma(df / 2) * (
            1 + (x ** 2 + y ** 2 - 2 * rho * x * y) /
            (df * (1 - rho ** 2))) ** (-df + 2) / 2

        denominator = gamma((df + 1) / 2) ** 2 * df * np.pi *
            np.sqrt(1 - rho ** 2) * (1 + x ** 2 / df) ** (
                -(df + 1) / 2) * (1 + y ** 2 / df) ** (-df + 1) / 2

        result = numerator / denominator

        # Sostituire infiniti o NaN con un valore molto piccolo ma positivo
        result = np.where(np.isfinite(result) & (result > 0), result, 1e-10)
        return result
    except:
        # In caso di errore, ritorna un valore di default
        return np.ones_like(u) * 1e-10
```

Implementiamo la funzione likelihood

```
def t_copula_likelihood(params, u, v):
    rho = np.tanh(params[0])
    df = np.exp(params[1]) + 2
    likelihoods = t_copula_pdf(u, v, rho, df)

    # Gestione di valori non validi
    valid_idx = ~np.isnan(likelihoods) & (likelihoods > 0)
    if not np.any(valid_idx):
        return 1e10
    return -np.sum(np.log(likelihoods[valid_idx]))
```

Implementiamo la stima MLE

```
def estimate_t_copula_MLE(u, v):
    initial_guess = [0.5, np.log(8)]
    bounds = [(-10, 10), (-10, 10)] # Limiti per rho e log(df)
    result = optimize.minimize(t_copula_likelihood, initial_guess, args=(u, v),
                               method='L-BFGS-B', bounds=bounds)
    rho_estimated = np.tanh(result.x[0])
    df_estimated = np.exp(result.x[1]) + 2
    return rho_estimated, df_estimated
```

Infine possiamo osservare i risultati ottenuti in questo modo

```
rho_t_mle, df_t_mle = estimate_t_copula_MLE(u, v)
print(f" Parametro stimato (rho) per la Copula t-Student: {rho_t_mle:.4f}")
print(f" Gradi di libertà stimati per la Copula t-Student: {df_t_mle:.2f}")
```

I risultati numerici e i valori di stima del nostro dataset saranno calcolati e mostrati più avanti

5.2.3 Metodo dei Momenti

Il metodo dei momenti è una tecnica di stima che si basa sull'equiparazione dei momenti teorici di una distribuzione con i momenti campionari calcolati dai dati osservati. Nel contesto delle copule, il metodo può essere utilizzato per stimare i parametri che descrivono la dipendenza tra le variabili aleatorie, garantendo coerenza tra la struttura di dipendenza modellata e quella osservata.

Applicazione alle copule

Le copule sono funzioni che legano le distribuzioni marginali di variabili aleatorie alla loro distribuzione congiunta, separando la dipendenza dalla struttura marginale. Per stimare i parametri di una copula con il metodo dei momenti:

1. **Scelta dei momenti di interesse:** Si identificano statistiche che catturano la dipendenza (ad esempio, Kendall's τ , Spearman's ρ o altre misure di concordanza).
2. **Calcolo dei momenti campionari:** Si calcolano i momenti empirici dai dati osservati, ad esempio, calcolando τ o ρ sui campioni.
3. **Imposizione di uguaglianza:** I momenti teorici della copula, funzione dei parametri da stimare, vengono egualati ai momenti campionari.
4. **Risoluzione del sistema:** Si risolvono le equazioni risultanti per determinare i parametri della copula.

Ambiti di Applicazione

Il metodo dei momenti applicato alle copule trova largo impiego in settori dove è cruciale modellare la dipendenza tra variabili, come:

- **Finanza:** Per analizzare la dipendenza tra rendimenti di asset.
- **Assicurazioni e gestione del rischio:** Per modellare eventi estremi correlati, come sinistri catastrofici.

Vantaggi e Svantaggi

Elichiamo alcuni pro e contro di utilizzare questo metodo:

Vantaggi

- **Semplicità computazionale:** Rispetto ad altri metodi (ad esempio, la massima verosimiglianza), il metodo dei momenti è spesso più semplice da implementare e richiede meno assunzioni sul modello.
- **Intuitività:** L'approccio è facilmente interpretabile grazie al legame diretto con misure di dipendenza osservabili come τ e ρ .
- **Robustezza:** È meno sensibile a errori nelle specifiche delle distribuzioni marginali.

Svantaggi

- **Perdita di efficienza:** Gli stimatori ottenuti non sono sempre efficienti, il che significa che potrebbero avere una varianza maggiore rispetto a quelli della massima verosimiglianza.
- **Dipendenza dalla scelta dei momenti:** La qualità della stima dipende fortemente dalla scelta dei momenti, che potrebbero non catturare adeguatamente la complessità della dipendenza.
- **Limitazioni con dati scarsi:** Con campioni piccoli, i momenti campionari potrebbero essere poco rappresentativi.

Limiti in contesti multivariati

Un altro fattore da tenere in considerazione è il numero di variabili a cui è applicato, in quanto potrebbe avere delle limitazioni come:

1. Difficoltà di Generalizzazione in Dimensioni Elevate:

- Le misure di concordanza (ad esempio, Kendall's τ) diventano difficili da calcolare per coppie di variabili.
- Il numero di parametri della copula aumenta esponenzialmente con la dimensione, rendendo il sistema di equazioni derivato dai momenti difficile da risolvere.

2. Sensibilità ai Dati Empirici:

I momenti campionari utilizzati per la stima sono sensibili alla presenza di outlier o dati scarsi, introducendo bias significativi nella stima dei parametri.

3. Rappresentazione Limitata della Dipendenza:

- Il metodo dei momenti si basa su misure sintetiche della dipendenza (come τ o ρ), che potrebbero non cogliere adeguatamente relazioni complesse, come:
 - Dipendenze non lineari.
 - Code pesanti o asimmetrie estreme.

4. Convergenza Non Ottimale:

Gli stimatori ottenuti con il metodo dei momenti non sono necessariamente efficienti in termini di varianza, soprattutto quando i dati presentano una struttura di dipendenza complessa.

5.2.4 Il Tau di Kendall

Abbiamo parlato più volte della misura τ di Kendall, spieghiamo meglio in cosa consiste.

Il tau di Kendall è una misura di concordanza per due variabili casuali continue. È definito come la differenza tra la probabilità di concordanza e la probabilità di discordanza. Può essere interpretato come la probabilità che due coppie di osservazioni scelte a caso dal campione siano concordanti meno la probabilità che siano discordanti.

Principio di base

Dato un insieme di n coppie di dati $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, il Kendall's Tau si basa sul confronto tra tutte le coppie di osservazioni:

1. **Coppia concordante:** Una coppia (x_i, y_i) e (x_j, y_j) è *concordante* se i segni delle differenze $(x_j - x_i)$ e $(y_j - y_i)$ sono uguali (entrambi positivi o entrambi negativi). In parole semplici:

$$(x_j - x_i)(y_j - y_i) > 0$$

2. **Coppia discordante:** Una coppia (x_i, y_i) e (x_j, y_j) è *discordante* se i segni delle differenze $(x_j - x_i)$ e $(y_j - y_i)$ sono opposti (uno positivo e l'altro negativo). In parole semplici:

$$(x_j - x_i)(y_j - y_i) < 0$$

3. **Coppia legata:** Una coppia è *legata* se $x_j = x_i$ o $y_j = y_i$.

Formula del Kendall's Tau

Il Kendall's Tau è calcolato come:

$$\tau = \frac{C - D}{\binom{n}{2}}$$

Dove:

- C è il numero di coppie concordanti.
- D è il numero di coppie discordanti.
- $\binom{n}{2} = \frac{n!}{2(n-2)!} = \frac{n(n-1)}{2}$ è il numero totale di coppie.

Valori di τ :

- $\tau = 1$: perfetta concordanza.
- $\tau = -1$: perfetta discordanza.
- $\tau = 0$: assenza di relazione (casuale).

Rho di Spearman e Tau di Kendall

Il **rho di Spearman** ρ_C è una misura di dipendenza non parametrica che valuta il grado di monotonicità della relazione tra due variabili casuali. Esso è definito in termini di copula come:

$$\rho_C = 12 \int \int_{I^2} C(u, v) dudv - 3$$

o, in forma equivalente,

$$\rho_C = 12 \int \int_{I^2} (C(u, v) - uv) dudv.$$

Questa espressione mostra che il rho di Spearman misura la distanza media tra la distribuzione congiunta, rappresentata dalla copula C , e il caso di indipendenza $P(u, v) = uv$. Valori elevati di ρ_C indicano una forte dipendenza monotona tra le variabili, mentre un valore vicino a zero suggerisce indipendenza.

Analogamente, il **tau di Kendall** τ_C è un'altra misura non parametrica di concordanza tra coppie di osservazioni, definita come la differenza tra la probabilità di concordanza e discordanza. Per le copule Archimedee, il tau di Kendall può essere calcolato direttamente dal generatore della copula, rendendolo un parametro utile per la caratterizzazione della dipendenza.

Entrambe le misure forniscono un'alternativa alla correlazione lineare, particolarmente utile in contesti in cui le variabili non seguono una distribuzione normale o presentano relazioni non lineari. Il rho di Spearman, in particolare, enfatizza la distanza tra la struttura congiunta dei dati e l'indipendenza, offrendo un'indicazione robusta della dipendenza tra le variabili.

5.2.5 Relazioni tra parametri delle copule e misure di dipendenza

Le formule che legano il tau di Kendall ai parametri delle diverse copule:

1. Copula Gaussiana

$$\tau = \frac{2}{\pi} \arcsin(\rho)$$

Inversione:

$$\rho = \sin\left(\frac{\pi\tau}{2}\right)$$

2. Copula t-Student

$$\tau = \frac{2}{\pi} \arcsin(\rho)$$

Stessa relazione della Gaussiana, indipendente dai gradi di libertà ν .

3. Copula di Clayton

$$\tau = \frac{\theta}{\theta + 2}$$

Inversione:

$$\theta = \frac{2\tau}{1 - \tau}$$

4. Copula di Gumbel

$$\tau = 1 - \frac{1}{\theta}$$

Inversione:

$$\theta = \frac{1}{1 - \tau}$$

5. Copula di Frank

$$\tau = 1 - \frac{4}{\theta} (1 - D_1(\theta))$$

dove $D_1(\theta)$ è la funzione di Debye:

$$D_1(\theta) = \frac{1}{\theta} \int_0^\theta \frac{t}{e^t - 1} dt$$

Non esiste una formula di inversione semplice.

5.2.6 Codice Python metodo dei momenti

Applicazione alla copula Gumbel

Per prima cosa definiamo la relazione tra il tau di Kendall e il parametro theta della Gumbel.

```
def gumbel_theta_kendall(tau):
    """
    Calcola il parametro theta della copula di Gumbel a partire da tau di Kendall.
    Formula: theta = 1 / (1 - tau)
    """
    return 1 / (1 - tau)
```

Successivamente impostiamo una funzione che calcoli il tau di Kendall in maniera ottimizzata (non possiamo calcolarlo come nell'esercizio numerico in seguito all'elevato costo computazionale).

```
from scipy.stats import kendalltau

def kendall_tau_multivariate_optimized(dataset):
    """
    Calcola il Kendall tau medio per un dataset multivariato usando scipy.
    """
    n_vars = dataset.shape[1]
    taus = []

    # Calcola Kendall tau per ogni coppia di variabili
    for i in range(n_vars):
        for j in range(i + 1, n_vars):
            tau, _ = kendalltau(dataset[:, i], dataset[:, j])
            taus.append(tau)

    # Restituisce il tau medio e il numero di coppie concordanti/discordanti
    tau_mean = np.mean(taus)
    return tau_mean
```

Per concludere implementiamo il metodo dei momenti ottimizzato.

```
def method_of_moments_gumbel_optimized(dataset):
    """
    Applica il metodo dei momenti per stimare
    il parametro theta della copula di Gumbel.
    """
    # Calcola il tau di Kendall medio
    tau_mean = kendall_tau_multivariate_optimized(dataset)

    # Calcola il parametro theta
    theta = gumbel_theta_kendall(tau_mean)
    return theta, tau_mean
```

Ora non ci resta che applicare questo metodo al nostro dataset *uniform_data* e ottenere il risultato.

```
# Stima il parametro theta
theta_gumbel, tau_mean = method_of_moments_gumbel_optimized(uniform_data)
print(f"Parametro theta stimato della Gumbel: {theta_gumbel:.4f}")
```

Applicazione alla copula Clayton

Definiamo la relazione tra il tau di Kendall e il parametro theta della Clayton.

```
def clayton_theta_kendall(tau):
    """
    Calcola il parametro theta della copula di Clayton a partire da tau di Kendall.
    Formula: theta = 2*tau / (1 - tau)
    """
    return 2*tau / (1 - tau)
```

La funzione per il calcolo del tau di Kendall è esattamente uguale alla precedente.

Implementiamo il metodo dei momenti ottimizzato.

```
def method_of_moments_clayton_optimized(dataset):
    """
    Applica il metodo dei momenti per stimare
    il parametro theta della copula di Clayton.
    """
    # Calcola il tau di Kendall medio
    tau_mean = kendall_tau_multivariate_optimized(dataset)

    # Calcola il parametro theta
    theta = clayton_theta_kendall(tau_mean)
    return theta, tau_mean
```

Applichiamo questo metodo al nostro dataset *uniform_data* e otteniamo il risultato.

```
# Stima il parametro theta
theta_clayton, tau_mean = method_of_moments_clayton_optimized(uniform_data)
print(f"Parametro theta stimato della Clayton: {theta_clayton:.4f}")
```

5.2.7 Stima Bayesiana

La stima bayesiana è un approccio alla statistica basato sul teorema di Bayes, che combina informazioni a priori con dati osservati per aggiornare la nostra conoscenza su un fenomeno incerto. È ampiamente utilizzata in applicazioni come l'apprendimento automatico, l'inferenza statistica e il processo decisionale.

Teorema di Bayes

La statistica bayesiana si fonda sul teorema di Bayes, che dice che, data due variabili aleatorie (anche vettoriali) X e Y , allora:

$$f(x | y) = \frac{f(y, x)}{f(y)} = \frac{f(y | x)f(x)}{f(y)}$$

dove $f(y) = \int f(y, x)dx$.

Il teorema di Bayes permette di passare dalla condizionata di y rispetto a x a quella di x rispetto a y .

Un altro modo di vedere il teorema di Bayes è di tipo “iterativo”:

- Ho una distribuzione a priori su x , $f(x)$.
- Osservo una nuova variabile y , che dipende da x , tramite $f(y | x)$.
- Allora l'informazione che ho su x , dopo aver osservato y , cambia in $f(x | y)$.

Consideriamo ora $x = \theta$ come parametri di una densità di probabilità (ad esempio μ, σ^2 se parliamo di una normale), allora avremo:

- $f(\theta | y)$ è chiamata distribuzione a posteriori di θ .
- $f(y | \theta)$ è la congiunta delle osservazioni, che è possibile vedere anche come la verosimiglianza.
- $f(\theta)$ è la distribuzione a priori. Questa distribuzione riflette ciò che sappiamo dei parametri prima di osservare il campione y .
- $f(y)$ è la costante di normalizzazione, in genere meno rilevante poiché non dipende da θ .

La stima bayesiana, specialmente in combinazione con le copule, offre numerosi vantaggi in contesti caratterizzati da dati limitati o mercati volatili. Questi vantaggi emergono dal fatto che il paradigma bayesiano consente di integrare informazioni a priori, aggiornare le credenze in modo dinamico e modellare con precisione dipendenze complesse.

Vantaggi Generali della Stima Bayesiana

1. Integrazione di conoscenze a priori

La stima bayesiana consente di utilizzare informazioni preesistenti, come storie storiche o conoscenze di esperti, per costruire una distribuzione a priori. Questo è particolarmente utile in scenari con dati limitati.

2. Aggiornamento dinamico delle credenze

Il paradigma bayesiano permette di aggiornare i parametri di interesse man mano che nuovi dati diventano disponibili, utilizzando il teorema di Bayes.

3. Distribuzione a posteriori invece di stime puntuali

La stima bayesiana produce distribuzioni a posteriori che rappresentano l'incertezza sui parametri, offrendo un quadro probabilistico più robusto rispetto all'uso di stime puntuali.

4. Gestione di modelli complessi

Grazie alla capacità di combinare la struttura di dipendenza (copule) con modelli marginali, la stima bayesiana è particolarmente adatta per problemi multivariati o non lineari.

Vantaggi Specifici nei Contesti con Copule

Le **copule** sono utili per modellare le dipendenze tra variabili aleatorie, anche quando queste non seguono distribuzioni gaussiane o hanno comportamenti estremi (ad esempio, correlazioni nelle code). Quando abbinate alla stima bayesiana, i vantaggi si amplificano nei seguenti modi:

1. Modellazione della dipendenza in contesti di dati limitati

Nei mercati finanziari o assicurativi con dati scarsi, l'approccio bayesiano consente di sfruttare distribuzioni a priori sui parametri della copula, riducendo il rischio di sottostimare o sovrastimare la dipendenza.

2. Robustezza in mercati volatili

Le copule consentono di modellare cambiamenti nella dipendenza tra variabili in mercati turbolenti. L'approccio bayesiano aggiorna dinamicamente questi parametri con nuovi dati, migliorando la risposta ai cambiamenti rapidi.

3. Modellazione delle code

Le copule come la Gumbel o la t-Student sono in grado di catturare la dipendenza nelle code della distribuzione. In combinazione con l'approccio bayesiano, è possibile stimare parametri di coda con maggiore affidabilità anche in presenza di pochi dati osservati.

4. Riduzione del rischio di overfitting

La stima bayesiana utilizza distribuzioni a priori regolarizzanti, che prevengono l'overfitting tipico nei modelli di dipendenza con molti parametri. Questo è particolarmente importante in contesti con dati limitati.

Applicazioni Specifiche

1. Finanza (Portafogli e Rischio)

- La stima bayesiana con copule permette di modellare la dipendenza tra rendimenti degli asset in portafoglio, considerando eventi estremi (correlazioni nelle code).
- È utile per stimare la *Value at Risk* (VaR) e il *Conditional VaR* (CVaR) in mercati volatili.

2. Assicurazioni (Rischi Multivariati)

- Modellare la dipendenza tra sinistri assicurativi (ad esempio, danni da eventi naturali correlati).
- L'approccio bayesiano consente di incorporare conoscenze a priori su rischi rari ma severi.

3. Analisi Multivariata e Dati Mancanti

- Quando alcune variabili o dati sono mancanti, le copule e la stima bayesiana consentono di stimare la dipendenza tra variabili osservate e non osservate.

5.2.8 MCMC

Il metodo Markov Chain Monte Carlo (MCMC) è una tecnica di simulazione numerica utilizzata per generare campioni da distribuzioni di probabilità complesse, quando il campionamento diretto è impossibile o inefficiente. Il suo obiettivo principale è quello di costruire una catena di Markov che, nel tempo, converga verso la distribuzione target desiderata.

Utilità di MCMC

In molti problemi statistici e probabilistici è difficile ottenere la distribuzione a posteriori in forma chiusa a causa della **costante di normalizzazione sconosciuta**. MCMC risolve questo problema generando campioni che, nel tempo, approssimano la distribuzione target senza doverla calcolare esplicitamente.

MCMC è utile in contesti come:

- **Inferenza Bayesiana:** per stimare distribuzioni a posteriori quando l'integrale di normalizzazione è intrattabile.
- **Modelli gerarchici e dati mancanti:** gestisce con facilità modelli con dipendenze complesse.
- **Finanza e Copule:** per stimare parametri di modelli di dipendenza tra variabili finanziarie.
- **Machine Learning:** per l'ottimizzazione di modelli probabilistici complessi.

Funzionamento di MCMC

L'idea di base è costruire una **catena di Markov** che abbia la distribuzione target $\pi(\theta)$ come distribuzione stazionaria. Indipendentemente dal punto di partenza, dopo un numero sufficiente di iterazioni, la catena converge alla distribuzione desiderata.

Il metodo MCMC funziona in due fasi principali:

1. **Generazione di una proposta:** si propone un nuovo valore θ^* basandosi sul valore corrente della catena.
2. **Accettazione o rifiuto:** il nuovo valore viene accettato con una probabilità dipendente dalla densità target.

Principali Algoritmi MCMC

Gibbs Sampling

- Particolarmente utile quando possiamo decomporre la distribuzione target in distribuzioni condizionate più semplici.
- Si campiona ogni variabile condizionatamente alle altre.
- È un caso particolare del **Metropolis-Hastings** con accettazione sempre pari a 1.

Metropolis-Hastings

- Algoritmo più generale, che funziona anche quando le distribuzioni condizionate non hanno forma chiusa.
- Si propone un valore θ^* da una distribuzione proposta $q(\theta^* | \theta)$ e lo si accetta con probabilità:

$$\alpha = \min \left(1, \frac{\pi(\theta^*) q(\theta | \theta^*)}{\pi(\theta) q(\theta^* | \theta)} \right)$$

- Se la distribuzione proposta è simmetrica ($q(\theta^* | \theta) = q(\theta | \theta^*)$), la formula si semplifica.

Problemi e Diagnosi della Convergenza

- **Burn-in:** le prime iterazioni sono scartate perché non appartengono ancora alla distribuzione target.
- **Thin:** per ridurre la dipendenza tra campioni consecutivi, si prendono solo alcuni campioni a intervalli regolari.
- **Autocorrelazione:** viene monitorata con la funzione di autocorrelazione (ACF) per assicurarsi che i campioni siano abbastanza indipendenti.

5.2.9 Metodo MCMC in Python

Di seguito viene presentata un'implementazione di Metropolis-Hastings con proposal $\mathbb{U}(-\delta, \delta)$ in Python, utilizzando la copula gaussiana per modellare la dipendenza tra variabili finanziarie.

Librerie utilizzate

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

Utilizziamo sempre le due colonne *Open* e *Close* del dataset *uniform_data* come campioni chiamndole corrispettivamente *u_unif* e *v_unif*.

Definizione della funzione di verosimiglianza per la Copula Gaussiana

```
def gaussian_copula_log_likelihood(rho, u, v):
    """Calcola la log-verosimiglianza della copula gaussiana."""
    if abs(rho) >= 0.999: # Evita problemi numerici ai bordi
        return -np.inf

    x = stats.norm.ppf(u)
    y = stats.norm.ppf(v)

    # Log-verosimiglianza della copula gaussiana
    term1 = -0.5 * np.log(1 - rho ** 2)
    term2 = -(rho ** 2 * (x ** 2 + y ** 2) - 2 * rho * x * y) / (2 * (1 - rho ** 2))

    return np.sum(term1 + term2)
```

Ora definiamo l'algoritmo MCMC migliorato con multiple chain e diagnostica robusta

```
def improved_metropolis_hastings(log_likelihood, u, v, n_chains=4,
n_iter=20000, burn_in=5000,
target_acceptance=0.234):
    """
    Implementazione migliorata di Metropolis-Hastings con catene multiple
    e adattamento del passo basato su Gelman et al. (1996).
    """
    n_params = 1 # Dimensione del parametro (solo rho)

    # Inizializzazione di catene multiple con punti di partenza diversi
    chains = np.zeros((n_chains, n_iter, n_params))

    # Punti di partenza distribuiti uniformemente tra -0.9 e 0.9
    starting_points = np.linspace(-0.8, 0.8, n_chains)

    # Parametri per l'adattamento
    gamma1 = 0.75 # Parametro che controlla l'adattamento iniziale
    adaptation_steps = int(n_iter * 0.6) # Numero di passi di adattamento

    # Memorizza i tassi di accettazione e le larghezze delle proposte
    acceptance_rates = np.zeros(n_chains)
    proposal_widths = np.ones(n_chains) * 0.1 # Valori iniziali

    print("\n===== AVVIO MCMC CON MULTIPLE CATENE =====")

    for c in range(n_chains):
        chain = chains[c]
        chain[0, 0] = starting_points[c] # Punto di partenza
        accepted = 0

        print(f"- Catena {c + 1}: punto di partenza = {starting_points[c]:.4f}")

        for i in range(1, n_iter):
            current_rho = chain[i - 1, 0]

            # Passo adattivo che diminuisce con le iterazioni
```

```
if i <= adaptation_steps:  
    adapt_factor = (i / adaptation_steps) ** gamma1  
    current_width = proposal_widths[c] *  
        (1 - adapt_factor) + 0.1 * adapt_factor  
else:  
    current_width = proposal_widths[c]  
  
# Proposta: U(-$\delta$, $\delta$) centrata sull'attuale valore  
delta = current_width  
proposal = current_rho + np.random.uniform(-delta, delta)  
  
# Rifletti se fuori dai limiti (-0.999, 0.999)  
if proposal <= -0.999:  
    proposal = -0.999 + ((-0.999) - proposal)  
elif proposal >= 0.999:  
    proposal = 0.999 - (proposal - 0.999)  
  
# Calcola il rapporto di accettazione  
log_p_current = log_likelihood(current_rho, u, v)  
log_p_proposal = log_likelihood(proposal, u, v)  
log_accept_ratio = log_p_proposal - log_p_current  
  
# Accetta o rifiuta  
if np.log(np.random.random()) < log_accept_ratio:  
    chain[i, 0] = proposal  
    accepted += 1  
else:  
    chain[i, 0] = current_rho  
  
# Adatta la larghezza della proposta  
if i % 500 == 0 and i <= adaptation_steps:  
    batch_acceptance = accepted / i  
  
# Aggiusta la larghezza per avvicinarsi al tasso di accettazione target  
if batch_acceptance > target_acceptance:  
    proposal_widths[c] *= 1.1 # Aumenta  
else:  
    proposal_widths[c] *= 0.9 # Diminuisci  
  
# Limita la larghezza per sicurezza  
proposal_widths[c] = max(0.01, min(1.0, proposal_widths[c]))  
  
if i % 5000 == 0:  
    print(f" Iterazione {i}, catena {c + 1}: accettazione = {batch_acceptance:.4f}, "  
        f"larghezza = {proposal_widths[c]:.4f}")  
  
# Statistiche finali per questa catena  
acceptance_rates[c] = accepted / n_iter  
print(f"- Catena {c + 1} completata: accettazione = {acceptance_rates[c]:.4f}, "  
    f"larghezza finale = {proposal_widths[c]:.4f}")
```

```
# Rimuovi burn-in e unisci le catene per l'analisi
combined_samples = chains[:, burn_in:, 0].flatten()

return chains, acceptance_rates, combined_samples
```

Ora eseguiamo MCMC migliorato

```
n_chains = 4
n_iter = 20000
burn_in = 5000

chains, acceptance_rates, combined_samples = improved_metropolis_hastings(
gaussian_copula_log_likelihood,
u_unif, v_unif, n_chains=n_chains, n_iter=n_iter,
burn_in=burn_in
)
```

Verifichiamo la convergenza del metodo tramite la diagnostica di Gelman-Rubin.

La diagnostica di Gelman-Rubin, nota anche come \hat{R} , è un metodo per valutare la convergenza delle catene MCMC. Confronta la varianza tra le catene con la varianza interna delle singole catene, fornendo un'indicazione di stabilità del campionamento. Se $\hat{R} \approx 1$, le catene sono considerate convergenti; valori superiori a 1.1 indicano potenziali problemi di convergenza e la necessità di iterazioni aggiuntive.

```
def gelman_rubin(chains, burn_in=0):
    """Calcola la diagnostica R-hat di Gelman-Rubin."""
    n_chains, n_iter, n_params = chains.shape

    if burn_in > 0:
        chains = chains[:, burn_in:, :]

    n_iter = chains.shape[1]

    # Medie delle catene
    chain_means = np.mean(chains, axis=1)  # shape: (n_chains, n_params)

    # Varianza tra le catene
    B = n_iter * np.var(chain_means, axis=0, ddof=1)  # shape: (n_params,)

    # Varianza entro le catene
    W = np.mean(np.var(chains, axis=1, ddof=1), axis=0)  # shape: (n_params,)

    # Stima complessiva della varianza
    var_hat = ((n_iter - 1) / n_iter) * W + (1 / n_iter) * B

    # Calcolo R-hat
    R_hat = np.sqrt(var_hat / W)

    return R_hat
```

Infine visualizziamo i risultati.

```
rho_mean = np.mean(combined_samples)
rho_std = np.std(combined_samples)
rho_median = np.median(combined_samples)
rho_ci = np.percentile(combined_samples, [2.5, 97.5])

print("\n===== RISULTATI FINALI =====")
print(f" Stima Bayesiana di $\rho$ con MCMC: {rho_mean:.4f} ± {rho_std:.4f}")
print(f" Mediana: {rho_median:.4f}")
print(f" Intervallo di credibilità al 95%: [{rho_ci[0]:.4f}, {rho_ci[1]:.4f}]")
print(f" Dimensione del campione efficace: {len(combined_samples)}")
```

5.3 Implementazione Pratica: Applicazione ai Dati del DAX

5.3.1 Preparazione dei dati

La descrizione dettagliata del dataset utilizzato è già fornita nella terza parte della tesi. In questa sezione, ci concentreremo invece sull'implementazione di un processo strutturato per la preparazione e la verifica dei dati, con l'obiettivo di garantirne la qualità e l'affidabilità per le analisi successive.

Il codice sviluppato esegue una serie di operazioni di diagnostica e pulizia sui dati di mercato, consentendo di ottenere un dataset coerente, privo di errori e pronto per le operazioni analitiche.

Librerie utilizzate

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import csv
```

Parte 1: Diagnostica del file CSV

La prima fase del processo di preparazione dei dati consiste nella **diagnostica del file CSV**. L'obiettivo principale di questa sezione è garantire che il file di dati esista, sia leggibile e abbia una struttura coerente. Per farlo, vengono eseguite le seguenti operazioni:

1. Verifica dell'esistenza del file

- Il codice controlla se il file specificato (`DAX_3Y-1M.csv`) è presente nella directory corrente.
- Se il file non esiste, viene stampato un messaggio di errore con l'indicazione della directory corrente e dei file disponibili, facilitando il debugging.

2. Ispezione delle prime righe del file

- Se il file è stato trovato, vengono lette e stampate le prime dieci righe del file per ottenere una visione preliminare della sua struttura.
- Questo aiuta a identificare eventuali problemi evidenti, come caratteri strani, righe mancanti o errori di formattazione.

3. Identificazione del delimitatore

- Poiché i file CSV possono essere separati da virgole (,), punti e virgola (;), tabulazioni (\t) o altri caratteri, viene utilizzata la funzione `csv.Sniffer()` per rilevare automaticamente il delimitatore corretto.
- Questo passaggio è essenziale per evitare errori di lettura quando si carica il file con `pandas`.

4. Verifica dell'intestazione

- Il codice verifica se il file contiene un'intestazione (ovvero nomi di colonne) e la conferma all'utente.
- Se l'intestazione non è presente, sarà necessario gestire i nomi delle colonne in modo esplicito nelle fasi successive.

Questa prima parte assicura che il file sia disponibile e che la sua struttura sia chiara prima di procedere con la lettura e la pulizia dei dati. Se il file presenta problemi, gli errori vengono segnalati tempestivamente, evitando che il codice fallisca nelle fasi successive. Nel terminale visualizzo:

Diagnostica del File CSV

File "DAX_3Y-1M.csv" trovato.

Ispezione delle Prime Righe del File

DateTime	Open	High	Low	Close
02/01/2020 01:15:00 +01:00	13174	13194.5	13171.5	13177.5
02/01/2020 01:16:00 +01:00	13177	13185.0	13177.0	13180.5
02/01/2020 01:17:00 +01:00	13180.5	13181.5	13179.0	13181.5
02/01/2020 01:18:00 +01:00	13181.5	13182.0	13180.5	13182.0
02/01/2020 01:19:00 +01:00	13182.0	13183.0	13180.5	13181.5
02/01/2020 01:20:00 +01:00	13181.5	13184.5	13181.5	13184.5
02/01/2020 01:21:00 +01:00	13183.5	13186.0	13184.0	13184.0
02/01/2020 01:22:00 +01:00	13185.5	13185.5	13181.5	13181.5
02/01/2020 01:23:00 +01:00	13183.5	13183.5	13181.5	13182.0

Tabella 5.1. Estratto delle prime righe del file CSV.

Delimitatore e Intestazione

Delimitatore rilevato: ','
Intestazione rilevata: False

Lettura Riuscita con Delimitatore ','

Tentativo di Diversi Metodi di Lettura

DtypeWarning: Columns (1,2,3,4) have mixed types.
Specify dtype option on import or set low_memory=False.

DateTime	Open	High	Low	Close	Unnamed: 5
02/01/2020 01:15:00 +01:00	13174	13194.5	13171.5	13177.5	NaN
02/01/2020 01:16:00 +01:00	13177	13185.0	13177.0	13180.5	NaN
02/01/2020 01:17:00 +01:00	13180.5	13181.5	13179.0	13181.5	NaN
02/01/2020 01:18:00 +01:00	13181.5	13182.0	13180.0	13182.0	NaN
02/01/2020 01:19:00 +01:00	13182.0	13183.0	13180.5	13181.5	NaN

Tabella 5.2. Estratto delle prime righe del file CSV.

Lettura Base Riuscita

DateTime	Open	High	Low	Close	Unnamed: 5
02/01/2020 01:15:00 +01:00	13174	13194.5	13171.5	13177.5	NaN
02/01/2020 01:16:00 +01:00	13177	13185.0	13177.0	13180.5	NaN
02/01/2020 01:17:00 +01:00	13180.5	13181.5	13179.0	13181.5	NaN
02/01/2020 01:18:00 +01:00	13181.5	13182.0	13180.0	13182.0	NaN
02/01/2020 01:19:00 +01:00	13182.0	13183.0	13180.5	13181.5	NaN

Tabella 5.3. Estratto delle prime righe del file CSV.

Parte 2: Lettura e Correzione dei Dati

Dopo aver verificato l'esistenza e la struttura del file CSV nella fase di diagnostica, questa seconda parte si concentra sulla **lettura e correzione dei dati**, assicurandosi che il file venga caricato correttamente e che le colonne siano interpretate nel formato corretto.

Le operazioni principali svolte in questa fase sono:

1. Tentativi di lettura del file con diversi delimitatori

- I file CSV possono essere separati da diversi caratteri (virgole ,, punti e virgola ;, tabulazioni \t, barre verticali |).
- Il codice prova a leggere il file con ciascun delimitatore e controlla quale funziona senza errori.
- Se la lettura con un certo delimitatore ha successo, il codice interrompe il ciclo e memorizza il delimitatore corretto per le fasi successive.

2. Gestione di possibili errori di encoding

- I file CSV possono essere salvati con differenti codifiche (UTF-8, ISO-8859-1, latin1, cp1252, ecc.), e una codifica errata può causare errori nella lettura.
- Viene tentata la lettura con UTF-8, ma se fallisce, vengono provate altre codifiche.
- Se tutte le codifiche testate falliscono, viene segnalato un errore.

3. Verifica della struttura del DataFrame

- Dopo aver letto il file, vengono stampate le prime righe per confermare che i dati siano stati caricati correttamente.
- Viene controllato se il dataset contiene colonne indesiderate, come `Unnamed: X`, che potrebbero derivare da errori nel file originale.
- Se esistono colonne numeriche con dati misti (numeri e stringhe), viene segnalato un avviso.

4. Conversione dei dati al formato corretto

- Le colonne numeriche (`Open`, `High`, `Low`, `Close`) vengono convertite in formato numerico, forzando la conversione (`errors='coerce'`), in modo da trasformare eventuali valori errati in `NaN` (valori mancanti).
- Se una colonna essenziale come `DateTime` esiste, viene convertita in un formato di data/ora (`datetime64`).
- Se alcune righe contengono errori di conversione (es. date errate o valori non numerici), vengono rimosse.

Questa fase assicura che il file venga letto in modo robusto e senza errori, adattando la configurazione al formato effettivo del CSV. Il risultato finale è un dataset pulito e ben strutturato, pronto per ulteriori operazioni di analisi e modellazione.

Nel terminale visualizzeremo:

Correzione e Preprocessing dei Dati

Inizio verifica della corretta scrittura dei dati.

Impossibile convertire 'DateTime': il valore "13/01/2020 01:15:00 +01:00" non corrisponde al formato "%m/%d/%Y %H:%M:%S %z" alla posizione 8672. Possibili soluzioni:

- Usare `format` se le stringhe hanno un formato coerente.
- Usare `format="ISO8601"` se le stringhe sono tutte in ISO8601 ma non esattamente nello stesso formato.
- Usare `format="mixed"` per inferire automaticamente il formato per ogni elemento individualmente.

Tentativo di correzione del formato della data: Esempi di date convertite:

'02/01/2020 01:15:00 +01:00', '02/01/2020 01:16:00 +01:00',
'02/01/2020 01:17:00 +01:00', '02/01/2020 01:18:00 +01:00'

Conversione colonne di prezzo

Colonne convertite: Open, High, Low, Close Tipi di dati iniziali:

```
Open      object  
High     object  
Low      object  
Close    object  
dtype: object
```

Le seguenti colonne sono state convertite correttamente:

- Colonna `Open`: convertita da formato con virgole
- Colonna `High`: convertita da formato con virgole
- Colonna `Low`: convertita da formato con virgole
- Colonna `Close`: convertita da formato con virgole

Rimosse 0 righe con valori mancanti

Statistiche sui dati numerici

	Open	High	Low	Close
count	616397	616397	616397	616397
mean	13884.68	13887.52	13881.79	13884.66
std	1747.75	1763.92	1733.78	1747.75
min	7970.00	7970.00	7970.00	7970.00
25%	12878.50	12878.50	12878.50	12878.50
50%	13914.00	13914.00	13914.00	13914.00
75%	15487.00	15487.00	15487.00	15487.00
max	16294.00	16294.00	16294.00	16294.00

Tabella 5.4. Statistiche descrittive delle colonne numeriche

Identificazione degli outlier

- Colonna Open: 6270 potenziali outlier (1.02%)
- Colonna High: 6232 potenziali outlier (1.01%)
- Colonna Low: 6331 potenziali outlier (1.03%)
- Colonna Close: 6287 potenziali outlier (1.02%)

Impostazione indice e salvataggio

La colonna `DateTime` è stata impostata come indice. I dati puliti sono stati salvati nel file `DAX_cleaned.csv`.

Visualizzazione dei dati puliti

DateTime	Open	High	Low	Close	Unnamed: 5
02/01/2020 01:15:00 +01:00	13174.0	13194.5	13171.5	13177.5	NaN
02/01/2020 01:15:00 +01:00	13174.0	13194.5	13171.5	13177.5	NaN
02/01/2020 01:16:00 +01:00	13177.0	13185.0	13171.0	13180.5	NaN
02/01/2020 01:16:00 +01:00	13177.0	13185.0	13171.0	13180.5	NaN
02/01/2020 01:17:00 +01:00	13180.5	13181.5	13181.5	13182.5	NaN
02/01/2020 01:16:00 +01:00	13177.0	13185.0	13171.0	13180.5	NaN
02/01/2020 01:16:00 +01:00	13177.0	13185.0	13171.0	13180.5	NaN
02/01/2020 01:16:00 +01:00	13177.0	13185.0	13171.0	13180.5	NaN
02/01/2020 01:15:00 +01:00	13174.0	13194.5	13171.5	13177.5	NaN

Tabella 5.5. Dati puliti dopo il preprocessing

Visualizzazione grafica dei dati puliti

Per comodità mostriamo solo il prezzo delle Open.

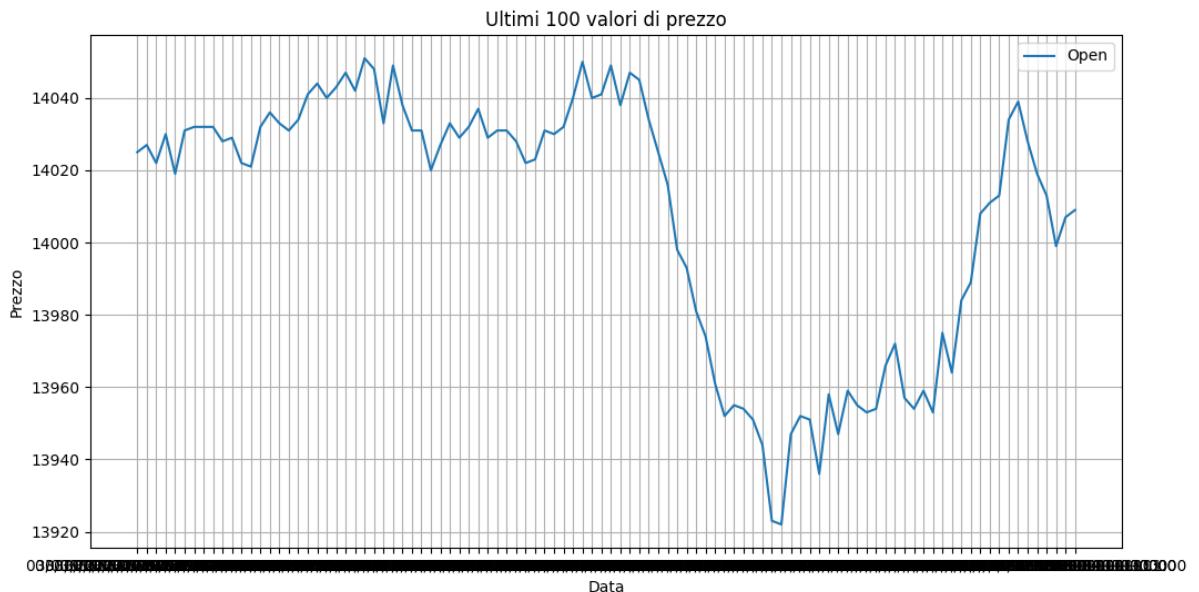


Figura 5.1. Ultimi 100 valori di prezzo

5.3.2 Stima dei Parametri tramite MLE

In questa sezione, stimiamo i parametri di cinque diverse copule: Gaussiana, t-Student, Clayton, Gumbel e Frank utilizzando la MLE. Per ciascuna di esse, calcoliamo la log-verosimiglianza massima e il valore stimato dei parametri. Visto la simmetria di queste copule, le applicheremo a coppie di variabili (Open e Close)

Il dataset utilizzato sarà quello ottenuto precedentemente, *DAX_cleaned.csv(df)*.

Librerie utilizzate

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.optimize import minimize
import warnings
```

Parte 1: Preparazione dei dati per l'analisi delle copule

La prima parte del codice si occupa della preparazione dei dati per l'analisi delle copule. Dopo il caricamento e la pulizia del dataset, vengono selezionate due colonne rappresentative per l'analisi. Nel nostro caso abbiamo preso *Open* e *Close*.

Successivamente vengono calcolati i rendimenti e standardizzati.

```
price_cols = [col for col in df.columns if col in
              ['Open', 'Close']]
col1, col2 = price_cols[:2]
print(f"Utilizzo delle colonne: '{col1}' e '{col2}'")

# Conversione in valori numerici
df[col1] = pd.to_numeric(df[col1], errors='coerce')
df[col2] = pd.to_numeric(df[col2], errors='coerce')

# Rimozione di valori mancanti
df = df.dropna(subset=[col1, col2])

# Calcolo dei rendimenti logaritmici
returns = pd.DataFrame({
    'r1': np.log(df[col1] / df[col1].shift(1)),
    'r2': np.log(df[col2] / df[col2].shift(1))
}).dropna()
```

Parte 2: Calcolo della volatilità rolling

Il nostro obiettivo è modellare la struttura di dipendenza tra le volatilità di due asset finanziari, non solo tra i loro rendimenti.

Questo permetterà alla procedura MCMC di stimare come la volatilità di un asset dipende dalla volatilità di un altro, il che è particolarmente utile per la modellazione del rischio e gli stress test. Per analizzare la volatilità degli asset finanziari, è necessario calcolare la deviazione standard su una finestra temporale mobile (21 giorni). Questo metodo, noto come "volatilità rolling", consente di valutare come il rischio di un asset varia nel tempo. Inoltre sono stati applicati degli accorgimenti per diminuirne il costo computazionale.

```
# 3. Calcolo della volatilità rolling
print("\nCalcolo della volatilità rolling...")
window = 21 # Finestra di 21 giorni (circa un mese di trading)

# Per dataset molto grandi, campionamento per migliorare le prestazioni
if len(returns) > 50000:
    print(f"Dataset molto grande, campionamento 1 ogni
          {len(returns) // 50000 + 1} righe...")
    sample_step = len(returns) // 50000 + 1
    returns_sample = returns.iloc[::sample_step].copy()
    print(f"Dataset ridotto da {len(returns)} a {len(returns_sample)} righe")
else:
    returns_sample = returns.copy()
```

```
# Calcolo della volatilità
returns_sample['vol1'] = returns_sample['r1'].rolling(window=window).std() *
np.sqrt(252) # Annualizzata
returns_sample['vol2'] = returns_sample['r2'].rolling(window=window).std() *
np.sqrt(252) # Annualizzata

# Rimozione dei NaN
vol_returns = returns_sample.dropna()
print(f"Volatilità calcolata su {len(vol_returns)} punti")
print(f"Calcolata volatilità rolling su finestra di {window} giorni")
print(f"Volatilità media di {col1}: {vol_returns['vol1'].mean():.2%}")
print(f"Volatilità media di {col2}: {vol_returns['vol2'].mean():.2%}")
```

Ottenendo questi valori

```
Calcolata volatilità rolling su finestra di 21 giorni
Volatilità media di Open: 0.56%
Volatilità media di Close: 0.58%
```

Il calcolo della volatilità rolling permette di monitorare le variazioni della volatilità nel tempo, offrendo una visione dinamica del rischio di mercato. Questo approccio è essenziale per comprendere la stabilità degli asset finanziari e valutare strategie di gestione del rischio.

Parte 3: Trasformazione in distribuzioni marginali uniformi

Le distribuzioni originali della volatilità potrebbero avere forme non gaussiane. La trasformazione con ranking empirico le rende uniformi, permettendo un'analisi di dipendenza più robusta.

```
# Utilizziamo il ranking empirico per mappare i dati in [0,1]
u1 = stats.rankdata(vol_returns['vol1']) / (len(vol_returns) + 1)
u2 = stats.rankdata(vol_returns['vol2']) / (len(vol_returns) + 1)
```

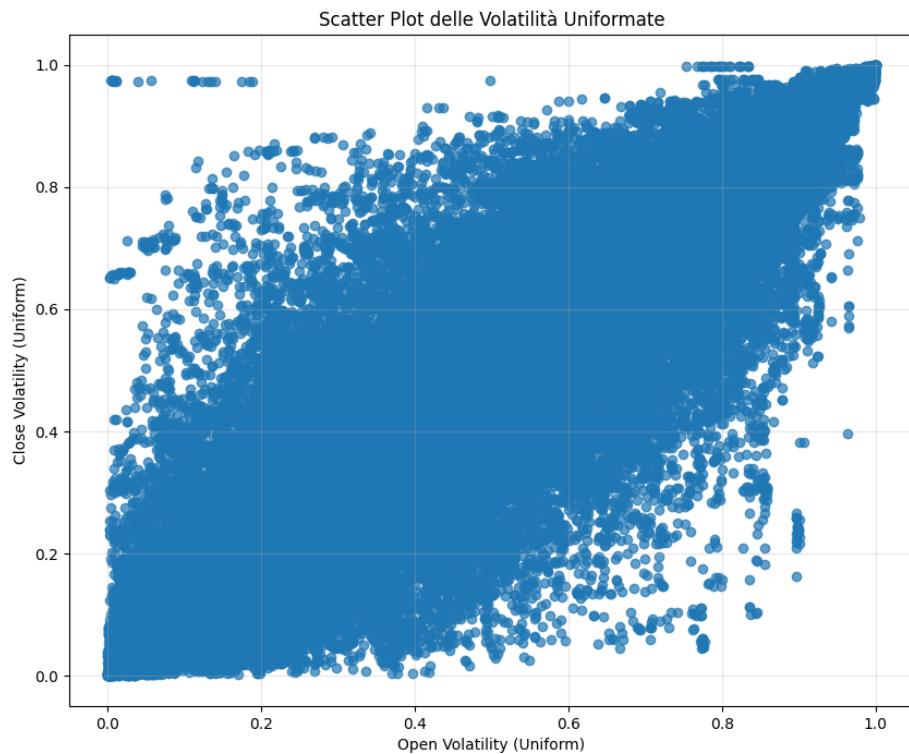


Figura 5.2. Scatter Plot delle Volatilità Uniformate

Parte 4: Implementazione delle funzioni di log-verosimiglianza per diverse copule

Implementiamo le funzioni di log-verosimiglianza delle copule Gaussiana, t-Student, Clayton, Gumbel e Frank.

```
# 5.1 Copula Gaussiana
def gaussian_copula_loglik(rho, u, v):
    if abs(rho) >= 1:
        return -np.inf

    norm_u = stats.norm.ppf(u)
    norm_v = stats.norm.ppf(v)
```

```
term1 = -0.5 * np.log(1 - rho ** 2)
term2 = -0.5 * (rho ** 2 * (norm_u ** 2 + norm_v ** 2) - 2 * rho * norm_u *
norm_v) / (1 - rho ** 2)

return np.sum(term1 + term2)

# 5.2 Copula t-Student
def t_copula_loglik(params, u, v):
    rho, nu = params

    if abs(rho) >= 1 or nu <= 2:
        return -np.inf

    t_u = stats.t.ppf(u, nu)
    t_v = stats.t.ppf(v, nu)

    w = (t_u ** 2 + t_v ** 2 - 2 * rho * t_u * t_v) / (1 - rho ** 2)
    term1 = -0.5 * np.log(1 - rho ** 2)
    term2 = -(nu + 2) / 2 * np.log(1 + w / nu)
    term3 = -(stats.t.logpdf(t_u, nu) + stats.t.logpdf(t_v, nu))

    return np.sum(term1 + term2 + term3)

# 5.3 Copula di Clayton
def clayton_copula_loglik(theta, u, v):
    if theta <= 0:
        return -np.inf

    term1 = np.log(1 + theta)
    term2 = -(1 + theta) * np.log(u * v)
    term3 = -(2 + 1 / theta) * np.log(u ** (-theta) + v ** (-theta) - 1)

    return np.sum(term1 + term2 + term3)

# 5.4 Copula di Gumbel
def gumbel_copula_loglik(theta, u, v):
    if theta < 1:
        return -np.inf

    log_u = -np.log(u)
    log_v = -np.log(v)
    w = (log_u ** theta + log_v ** theta) ** (1 / theta)

    term1 = -w
    term2 = np.log(w * (log_u * log_v) ** (theta - 1))
    term3 = np.log(theta - 1 + w) - np.log(u * v)

    return np.sum(term1 + term2 + term3)
```

```
# 5.5 Copula di Frank
def frank_copula_loglik(theta, u, v):
    if abs(theta) < 1e-10:
        return -np.inf

    term1 = np.log(abs(theta)) - np.log(1 - np.exp(-abs(theta)))
    term2 = -theta * (u + v)
    term3 = -2 * np.log(1 + np.exp(-theta)) * (np.exp(-theta * u) - 1) *
           (np.exp(-theta * v) - 1)

    return np.sum(term1 + term2 + term3)
```

Parte 5: Stima dei parametri per ciascuna copula

Iniziamo a calcolare delle misure di dipendenza che ci serviranno per assumere dei valori di partenza durante il processo di stima.

```
# Calcolo delle misure di dipendenza
print("\nMisure di dipendenza tra le volatilità:")
pearson_corr = np.corrcoef(vol_returns['vol1'], vol_returns['vol2'])[0, 1]
kendall_tau, _ = stats.kendalltau(vol_returns['vol1'], vol_returns['vol2'])
spearman_rho, _ = stats.spearmanr(vol_returns['vol1'], vol_returns['vol2'])
```

Misura	Valore
Correlazione di Pearson	0.8873
Tau di Kendall	0.6598
Rho di Spearman	0.8424

Tabella 5.6. Misure di dipendenza tra le volatilità

Definiamo ancora le funzioni negative per la minimizzazione.

```
# Funzioni negative per la minimizzazione
def neg_gaussian_loglik(rho, u, v):
    return -gaussian_copula_loglik(rho[0], u, v)

def neg_t_loglik(params, u, v):
    return -t_copula_loglik(params, u, v)

def neg_clayton_loglik(theta, u, v):
    return -clayton_copula_loglik(theta[0], u, v)

def neg_gumbel_loglik(theta, u, v):
    return -gumbel_copula_loglik(theta[0], u, v)

def neg_frank_loglik(theta, u, v):
    return -frank_copula_loglik(theta[0], u, v)
```

Adesso possiamo stimare i parametri di ciascuna copula.

```
# 6.1 Stima per la copula Gaussiana
init_rho = pearson_corr
bounds_gaussian = [(-0.999, 0.999)]
result_gaussian = minimize(neg_gaussian_loglik, [init_rho], args=(u1, u2),
                            bounds=bounds_gaussian, method='L-BFGS-B')
rho_mle = result_gaussian.x[0]
print(f"Copula Gaussiana - $\rho$: {rho_mle:.4f},
      log-verosimiglianza: {-result_gaussian.fun:.4f}")

# 6.2 Stima per la copula t-Student
init_params_t = [pearson_corr, 5]
bounds_t = [(-0.999, 0.999), (2.001, 30)]
result_t = minimize(neg_t_loglik, init_params_t, args=(u1, u2),
                     bounds=bounds_t, method='L-BFGS-B')
rho_t_mle, nu_t_mle = result_t.x
print(f"Copula t-Student - $\rho$: {rho_t_mle:.4f}, $\nu$: {nu_t_mle:.4f},
      log-verosimiglianza: {-result_t.fun:.4f}")

# 6.3 Stima per la copula di Clayton
result_clayton = None
if kendall_tau > 0:
    init_theta_clayton = (2 * kendall_tau) / (1 - kendall_tau)
    bounds_clayton = [(0.001, 20)]
    result_clayton = minimize(neg_clayton_loglik, [init_theta_clayton],
                               args=(u1, u2),
                               bounds=bounds_clayton, method='L-BFGS-B')
    theta_clayton_mle = result_clayton.x[0]
    print(f"Copula di Clayton - $\theta$: {theta_clayton_mle:.4f},
```

```

log-verosimiglianza: {-result_clayton.fun:.4f}")
else:
print("Copula di Clayton non applicabile ($\tau$ $\leq$ 0)")

# 6.4 Stima per la copula di Gumbel
result_gumbel = None
if kendall_tau > 0:
    init_theta_gumbel = 1 / (1 - kendall_tau)
    bounds_gumbel = [(1.001, 20)]
    result_gumbel = minimize(neg_gumbel_loglik, [init_theta_gumbel],
    args=(u1, u2), bounds=bounds_gumbel, method='L-BFGS-B')
    theta_gumbel_mle = result_gumbel.x[0]
    print(f"Copula di Gumbel - $\theta$: {theta_gumbel_mle:.4f},
    log-verosimiglianza: {-result_gumbel.fun:.4f}")
else:
print("Copula di Gumbel non applicabile ($\tau$ $\leq$ 0)")

# 6.5 Stima per la copula di Frank
init_theta_frank = 0.5
bounds_frank = [(-20, 20)]
result_frank = minimize(neg_frank_loglik, [init_theta_frank],
args=(u1, u2), bounds=bounds_frank, method='L-BFGS-B')
theta_frank_mle = result_frank.x[0]
print(f"Copula di Frank - $\theta$: {theta_frank_mle:.4f},
log-verosimiglianza: {-result_frank.fun:.4f}")

```

Copula	Parametro	Log-verosimiglianza
Gaussian	$\rho = 0.8569$	31393.9252
t-Student	$\rho = 0.8599, \nu = 6.3485$	119436.7486
Clayton	$\theta = 2.0808$	21220.4682
Gumbel	$\theta = 1.1457$	18049.9447
Frank	$\theta = -0.4603$	19543.0218

Tabella 5.7. Parametri stimati per le diverse copule e log-verosimiglianza

5.3.3 Stima dei parametri tramite metodo dei momenti

Per poter stimare i parametri delle copule con il metodo dei momenti è necessario conoscere la loro relazione con il tau di Kendall, le quali abbiamo già elencato nella sezione 5.2.5. Come valore di tau , utilizzeremo quello calcolato in precedenza ovvero 0.8708 .

Definiamo inizialmente le varie relazioni

```
# Copula Gumbel
def gumbel_theta_kendall(tau):
"""
Calcola il parametro theta della copula di Gumbel a partire da tau di Kendall.
Formula: theta = 1 / (1 - tau)
"""
return 1 / (1 - tau)

# Copula Clayton
def clayton_theta_kendall(tau):
"""
Calcola il parametro theta della copula di Clayton a partire da tau di Kendall.
Formula: theta = 2*tau / (1 - tau)
"""
return 2*tau / (1 - tau)

# Copula gaussiana
def gaussian_rho_kendall(tau):
"""
Calcola il parametro theta della copula Gaussiana a partire da tau di Kendall.
Formula: rho = sin(tau * pi / 2)
"""
return np.sin(tau * np.pi / 2)

#Copula student-t
def student_rho_kendall(tau):
"""
Calcola il parametro theta della copula Student t a partire da tau di Kendall.
Formula: rho = sin(tau * pi / 2)
"""
return np.sin(tau * np.pi / 2)

#Copula Frank
def debye_function(theta):
"""Calcola la funzione di Debye D1(theta)."""
integral, _ = spi.quad(lambda t: t / (np.exp(t) - 1), 0, theta)
return integral / theta

def equation_to_solve(theta, tau):
"""Equazione per trovare theta dato tau di Kendall."""

```

```
D1 = debye_function(theta)
return 1 - (4/theta) * (1 - D1) - tau

def frank_theta_kendall(tau_kendall, theta_guess=1):
    """Stima il valore di theta risolvendo l'equazione per tau di Kendall."""
    result = spo.root_scalar(equation_to_solve, args=(tau_kendall,), bracket=[-50, 50], method='brentq')
    return result.root if result.converged else None
```

Ottenendo questi risultati

Copula	Parametro Stimato
Gumbel	2.9394
Clayton	3.8789
Gaussian	0.8606
Student t	0.8606
Frank	9.7812

Tabella 5.8. Stima dei parametri per diverse copule

5.3.4 Stima dei parametri tramite MCMC

Le copule sono strumenti fondamentali per modellare la dipendenza tra variabili casuali, consentendo di separare la struttura di dipendenza dalle distribuzioni marginali. In questa sezione, definiamo le principali copule implementate in **PyMC3**, un potente framework per la modellazione statistica bayesiana.

PyMC3 permette di definire distribuzioni personalizzate attraverso `DensityDist`. Qui definiamo alcune delle copule più utilizzate, sfruttando funzioni di densità logaritmica per garantire una corretta implementazione all'interno del framework di inferenza bayesiana.

Librerie

Librerie impiegate nello sviluppo del codice

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import pymc3 as pm
import arviz as az
import theano.tensor as tt
import warnings
```

Copula Gaussiana

La copula Gaussiana è definita a partire dalla distribuzione normale di densità pari a:

$$c(u, v; \rho) = \frac{1}{\sqrt{1 - \rho^2}} \exp \left(-\frac{\rho^2(x^2 + y^2) - 2\rho xy}{2(1 - \rho^2)} \right)$$

dove $x = \Phi^{-1}(u)$ e $y = \Phi^{-1}(v)$.

Nella funzione sottostante ne calcoleremo il logaritmo.

```
def gaussian_copula_logp(u, v, rho):
    norm_u = stats.norm.ppf(u)
    norm_v = stats.norm.ppf(v)

    # Evitiamo valori estremi
    norm_u = tt.clip(norm_u, -8, 8)
    norm_v = tt.clip(norm_v, -8, 8)

    term1 = -0.5 * tt.log(1 - rho ** 2)
    term2 = -1 / (2 * (1 - rho ** 2)) * (norm_u ** 2 + norm_v ** 2 - 2 *
    rho * norm_u * norm_v)
    term3 = 0.5 * (norm_u ** 2 + norm_v ** 2)

    return term1 + term2 + term3
```

Copula Student-t

La copula Student-t è definita a partire dalla distribuzione normale di densità pari a:

$$c(u, v; \rho, \nu) = \frac{\Gamma\left(\frac{\nu+2}{2}\right) \Gamma\left(\frac{\nu}{2}\right)^{-1} \left(1 + \frac{x^2+y^2-2\rho xy}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}}}{\sqrt{1-\rho^2} \cdot \Gamma\left(\frac{\nu+1}{2}\right)^2 \Gamma\left(\frac{\nu}{2}\right)^{-2} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} \left(1 + \frac{y^2}{\nu}\right)^{-\frac{\nu+1}{2}}}$$

dove $x = t_\nu^{-1}(u)$ e $y = t_\nu^{-1}(v)$.

Nella funzione sottostante ne calcoleremo il logaritmo.

```
class StudentTCopula(pm.Continuous):
    """Distribuzione della copula t-Student"""

    def __init__(self, rho, nu, u=None, v=None, *args, **kwargs):
        self.rho = rho
        self.nu = nu
        self.u = u
        self.v = v
        super().__init__(*args, **kwargs)

    def logp(self, x):
        """Log densità della copula t-Student"""
        u = self.u
        v = self.v
        rho = self.rho
        nu = self.nu

        t_u = stats.t.ppf(u, nu)
        t_v = stats.t.ppf(v, nu)

        # Evita valori estremi
        t_u = tt.clip(t_u, -8, 8)
        t_v = tt.clip(t_v, -8, 8)

        w = (t_u ** 2 + t_v ** 2 - 2 * rho * t_u * t_v) / (1 - rho ** 2)

        term1 = tt.log(tt.gamma((nu + 2) / 2)) + tt.log(tt.gamma(nu / 2)) -
        2 * tt.log(tt.gamma((nu + 1) / 2))
        term2 = -0.5 * tt.log(1 - rho ** 2)
        term3 = -(nu + 2) / 2 * tt.log(1 + w / nu)
        term4 = -(nu + 1) / 2 * tt.log(1 + t_u ** 2 / nu) -
        (nu + 1) / 2 * tt.log(1 + t_v ** 2 / nu)

        return term1 + term2 + term3 - term4
```

Copula Clayton

La copula Clayton è definita a partire dalla distribuzione normale di densità pari a:

$$c(u, v; \theta) = (1 + \theta)(u \cdot v)^{-(1+\theta)}(u^{-\theta} + v^{-\theta} - 1)^{-2-\frac{1}{\theta}}$$

Nella funzione sottostante ne calcoleremo il logaritmo.

```
def clayton_copula_logp(u, v, theta):
    """Log densità della copula di Clayton"""
    # Evita problemi numerici per valori molto piccoli
    u = tt.clip(u, 0.001, 0.999)
    v = tt.clip(v, 0.001, 0.999)

    term1 = tt.log(1 + theta)
    term2 = -(1 + theta) * (tt.log(u) + tt.log(v))
    term3 = -(2 + 1 / theta) * tt.log(u ** (-theta) + v ** (-theta) - 1)

    return term1 + term2 + term3
```

Copula Gumbel

La copula Gumbel è definita a partire dalla distribuzione normale di densità pari a:

$$c(u, v; \theta) = \frac{C(u, v; \theta) \cdot [(-\ln u)^\theta + (-\ln v)^\theta]^{2/\theta-2} \cdot (\ln u \cdot \ln v)^{\theta-1}}{u \cdot v}$$

$$\times \left(\theta - 1 + [(-\ln u)^\theta + (-\ln v)^\theta]^{1/\theta} \right)$$

Nella funzione sottostante ne calcoleremo il logaritmo.

```
def gumbel_copula_logp(u, v, theta):
    """Log densità della copula di Gumbel"""
    # Evita problemi numerici per valori molto piccoli
    u = tt.clip(u, 0.001, 0.999)
    v = tt.clip(v, 0.001, 0.999)

    log_u = -tt.log(u)
    log_v = -tt.log(v)

    w = (log_u ** theta + log_v ** theta) ** (1 / theta)

    term1 = -w
    term2 = tt.log(w * (log_u * log_v) ** (theta - 1))
    term3 = tt.log(theta - 1 + w) - tt.log(u * v)

    return term1 + term2 + term3
```

Copula Frank

La copula Frank è definita a partire dalla distribuzione normale di densità pari a:

$$c(u, v; \theta) = \frac{\theta(1 - e^{-\theta})e^{-\theta(u+v)}}{(1 - e^{-\theta} - (1 - e^{-\theta u})(1 - e^{-\theta v}))^2}$$

Nella funzione sottostante ne calcoleremo il logaritmo.

```
def frank_copula_logp(u, v, theta):
    """Log densità della copula di Frank"""
    # Evita problemi numerici per valori molto piccoli
```

```
u = tt.clip(u, 0.001, 0.999)
v = tt.clip(v, 0.001, 0.999)

# Gestione dei casi limite per theta
eps = 1e-10
theta = tt.switch(tt.abs_(theta) < eps, eps, theta)

term1 = tt.log(tt.abs_(theta)) - tt.log(1 - tt.exp(-tt.abs_(theta)))
term2 = -theta * (u + v)
term3 = -2 * tt.log(1 + tt.exp(-theta)) * (tt.exp(-theta * u) - 1) *
(tt.exp(-theta * v) - 1)

return term1 + term2 + term3
```

Funzione per la stima MCMC delle copule

La funzione `fit_copula_mcmc` esegue la stima dei parametri di una copula utilizzando il metodo MCMC. Essa definisce un modello probabilistico in PyMC3, specificando i parametri da stimare con le loro distribuzioni a priori e la verosimiglianza del modello. Successivamente, esegue il campionamento MCMC e restituisce un riassunto statistico dei parametri stimati.

```
def fit_copula_mcmc(u, v, copula_type, samples=2000,
                     tune=1000, chains=2, cores=1):
    """
    Stima i parametri di una copula usando MCMC

    Parametri:
    u, v (array): Dati uniformi
    copula_type (str): Tipo di copula ('gaussian', 't', 'clayton',
                       'gumbel', 'frank')
    samples (int): Numero di campioni MCMC
    tune (int): Numero di step di tuning
    chains (int): Numero di catene
    cores (int): Numero di core da utilizzare

    Returns:
    dict: Risultati della stima con trace, summary e parametri
    """
    print(f"\n Iniziando stima MCMC per copula {copula_type}...")

    with pm.Model() as model:
        # Priors
        if copula_type == 'gaussian':
            # Parametro di correlazione tra -1 e 1
            rho = pm.Uniform('rho', -0.99, 0.99)

        # Likelihood
        pm.DensityDist('likelihood', lambda x: gaussian_copula_logp(u, v, rho),
                      observed=0)

        elif copula_type == 't':
```

```
# Parametri: correlazione e gradi di libertà
rho = pm.Uniform('rho', -0.99, 0.99)
nu = pm.Gamma('nu', alpha=2, beta=0.1) # Prior per i gradi di libertà

# Likelihood
StudentTCopula('likelihood', rho=rho, nu=nu, u=u, v=v, observed=0)

elif copula_type == 'clayton':
# Parametro theta > 0
theta = pm.Gamma('theta', alpha=1, beta=1)

# Likelihood
pm.DensityDist('likelihood', lambda x: clayton_copula_logp(u, v, theta),
observed=0)

elif copula_type == 'gumbel':
# Parametro theta >= 1
# Usiamo un offset per garantire theta >= 1
theta_offset = pm.Exponential('theta_offset', 1)
theta = pm.Deterministic('theta', 1 + theta_offset)

# Likelihood
pm.DensityDist('likelihood', lambda x: gumbel_copula_logp(u, v, theta),
observed=0)

elif copula_type == 'frank':
# Parametro theta può essere positivo o negativo
theta = pm.Normal('theta', mu=0, sd=5)

# Likelihood
pm.DensityDist('likelihood', lambda x: frank_copula_logp(u, v, theta),
observed=0)
else:
raise ValueError(f"Tipo di copula non supportato: {copula_type}")

# Sampling
print(f" Esecuzione sampling con {chains} catene, {samples} campioni,
{tune} passi di tuning...")
trace = pm.sample(samples, tune=tune, chains=chains, cores=cores,
return_inferencedata=True)

# Calcola la media posteriore dei parametri
print(" Calcolo statistiche posteriori...")
summary = az.summary(trace)

result = {
'trace': trace,
'summary': summary,
'parameters': {param: summary.loc[param, 'mean'] for param in
summary.index}
}
```

```
print(f" Stima completata per copula {copula_type}")
print(f"    Parametri stimati: {result['parameters']}")  
return result
```

Definizione del Modello Probabilistico

All'interno del blocco `with pm.Model() as model:`, viene creato un modello bayesiano in PyMC3. I parametri della copula vengono definiti con distribuzioni a priori adeguate alla loro interpretazione statistica:

- **Copula Gaussiana:** il parametro di correlazione ρ è modellato con una distribuzione uniforme tra -0.99 e 0.99. La verosimiglianza è definita tramite la funzione di densità `gaussian_copula_logp`.
- **Copula t-Student:** oltre alla correlazione ρ , si stima anche il numero di gradi di libertà ν , modellato con una distribuzione $\text{Gamma}(2, 0.1)$. Questo permette di catturare strutture di dipendenza più complesse, specialmente nelle code.
- **Copula Clayton:** il parametro θ , che controlla la dipendenza nelle code inferiori, è modellato con una distribuzione $\text{Gamma}(1, 1)$.
- **Copula di Gumbel:** il parametro θ deve essere maggiore di 1, quindi viene modellato con una distribuzione esponenziale(1) e traslato di 1.
- **Copula di Frank:** il parametro θ può assumere valori sia positivi che negativi, quindi è modellato con una distribuzione normale(0,5).

Per ciascun tipo di copula, viene utilizzata una funzione di densità appropriata per modellare la verosimiglianza, utilizzando `pm.DensityDist` o una classe personalizzata.

Dopo aver definito il modello, la funzione esegue il campionamento MCMC con:

```
trace = pm.sample(samples, tune=tune, chains=chains, cores=cores,
    return_inferencedata=True)
```

Infine, una volta completato il campionamento, viene calcolato il riassunto delle distribuzioni a posteriori dei parametri con `arviz`:

```
summary = az.summary(trace)
```

e creata una struttura con i risultati.

Confronto fra copule

Definiamo ora una funzione che stimi tutti parametri di tutte le copule.

Successivamente vengono calcolati dei criteri per ciascuna copula come ad esempio il **WAIC** (Widely Applicable Information Criterion), un criterio di selezione del modello basato sulla teoria bayesiana dell'informazione.

Infine viene indicata la copula migliore.

```
def compare_copulas_mcmc(u, v, samples=2000, tune=1000, chains=2, cores=1):
    """
    Stima e confronta diverse copule usando MCMC

    Parametri:
    u, v (array): Dati uniformi
    samples, tune, chains, cores: Parametri per MCMC

    Returns:
    
```

```
dict: Risultati per ciascuna copula
DataFrame: Confronto dei modelli
"""

print("\n2 Avvio stima MCMC per tutte le copule")
print("-" * 50)
print(" Questo processo potrebbe richiedere tempo...")

results = {}
copula_types = ['gaussian', 't', 'clayton', 'gumbel', 'frank']

for copula in copula_types:
    try:
        results[copula] = fit_copula_mcmc(u, v, copula_type=copula,
                                            samples=samples, tune=tune,
                                            chains=chains, cores=cores)
    except Exception as e:
        print(f" Errore nella stima della copula {copula}: {e}")

# Calcolo delle metriche di confronto dei modelli
print("\ Confronto dei modelli")
print("-" * 50)

model_comparison = []

for copula, result in results.items():
    try:
        waic = az.waic(result['trace'])
        loo = az.loo(result['trace'])

        model_info = {
            'Copula': copula,
            'WAIC': waic.waic,
            'WAIC_SE': waic.waic_se,
            'LOO': loo.loo,
            'LOO_SE': loo.loo_se,
            'Parametri': str(result['parameters'])
        }
        model_comparison.append(model_info)
    except Exception as e:
        print(f" Errore nel calcolo delle metriche per copula {copula}: {e}")

model_comparison_df = pd.DataFrame(model_comparison).sort_values('WAIC')
print("\nRisultati del confronto dei modelli (ordinati per WAIC crescente):")
print(model_comparison_df)
```

Copula	WAIC	WAIC_SE	LOO	LOO_SE	Parametri
2 gumbel	1840.241063	182.236454	1840.238152	182.236452	$\theta = 1.138$
3 frank	2066.282870	5.007498	2066.282686	5.007500	$\theta = -0.46$
1 clayton	2270.514815	56.262035	2270.511483	56.262161	$\theta = 2.154$
0 gaussian	3359.988932	70.134976	3359.986659	70.135136	$\rho = 0.863$

Tabella 5.9. Risultati del confronto dei modelli (ordinati per WAIC crescente)

Per la Student-t non è riuscito ad ottenere una stima, mentre la copula **migliore** in stima bayesiana è una Gumbel.

5.3.5 Scelta della copula migliore

Per la scelta della copula migliore utilizziamo i modelli basati su **Akaike Information Criterion (AIC)** e **Bayesian Information Criterion (BIC)**, ovvero funzioni che catturano la dipendenza tra variabili casuali mantenendo le loro distribuzioni marginali.

AIC e BIC

Nel contesto delle copule, AIC e BIC servono a confrontare diversi modelli di copula e scegliere quello più adatto ai dati. L'idea è penalizzare la complessità del modello per evitare overfitting.

- **AIC** è definito come:

$$AIC = -2 \log L + 2k \quad (5.1)$$

dove:

- L è la massima verosimiglianza del modello stimato,
- k è il numero di parametri del modello.

- **BIC** è definito come:

$$BIC = -2 \log L + k \log n \quad (5.2)$$

dove:

- n è il numero di osservazioni.

Entrambi i criteri cercano un equilibrio tra bontà di adattamento (log-verosimiglianza) e complessità del modello (numero di parametri), ma il **BIC penalizza maggiormente i modelli complessi** rispetto all'AIC.

Mostriamo nella pagina successiva i dati raccolti

Copula	Parametro
Gaussiano	$\rho = 0.8569$
t-Student	$\rho = 0.8599, \nu = 6.3485$
Clayton	$\theta = 2.0808$
Gumbel	$\theta = 1.1457$
Frank	$\theta = -0.4603$

Tabella 5.10. Parametri stimati con MLE

Copula	Parametro
Gumbel	$\theta = 2.9394$
Clayton	$\theta = 3.8789$
Gaussiano	$\rho = 0.8606$
Student t	$\rho = 0.8606, \nu = \text{null}$
Frank	$\theta = 9.7812$

Tabella 5.11. Parametri stimati con il metodo dei momenti

Copula	Parametro
Gaussiano	$\rho = 0.863$
t-Student	$\rho = \text{null}, \nu = \text{null}$
Clayton	$\theta = 2.154$
Gumbel	$\theta = 1.138$
Frank	$\theta = -0.460$

Tabella 5.12. Parametri stimati con MCMC

Implementazione Python

Per prima cosa facciamo una media tra i parametri ottenuti con le tre metodologie di stima in quanto saranno i valori che andremo ad utilizzare.

Copula	Parametro
Gaussian	$\rho = 0.8602$
t-Student	$\rho = 0.8603, \nu = 6.3485$
Clayton	$\theta = 2.7046$
Gumbel	$\theta = 1.7410$
Frank	$\theta = 2.9536$

Tabella 5.13. Parametri stimati per le diverse copule conclusivi

Successivamente implementiamo il calcolo di AIC e BIC su python ed otteniamo la copula migliore.

```
# Numero di parametri per ciascun modello
n_params_gaussian = 1
n_params_t = 2
n_params_clayton = 1
n_params_gumbel = 1
n_params_frank = 1

# Dimensione del campione
n_samples = len(u1)

# Valore di ogni parametro
rho_gaussian=0.8602
rho_student=0.8603
nu_t=6.3485
theta_clayton=2.7046
theta_gumbel=1.7410
theta_frank=2.9536

# Calcolo dell'AIC: -2*log-likelihood + 2*k
aic_gaussian = -2 * gaussian_copula_loglik(rho_gaussian,u1, u2) +
2 * n_params_gaussian
aic_t = -2 * t_copula_loglik([rho_student, nu_t],u1, u2) +
2 * n_params_t
aic_clayton = -2 * clayton_copula_loglik(theta_clayton,u1, u2) +
2 * n_params_clayton
aic_gumbel = -2 * gumbel_copula_loglik(theta_gumbel,u1, u2) +
2 * n_params_gumbel
aic_frank = -2 * frank_copula_loglik(theta_frank,u1, u2) +
2 * n_params_frank

# Calcolo del BIC: -2*log-likelihood + k*log(n)
bic_gaussian = -2 * gaussian_copula_loglik(rho_gaussian,u1, u2) +
n_params_gaussian * np.log(n_samples)
bic_t = -2 * t_copula_loglik([rho_student, nu_t],u1, u2) +
n_params_t * np.log(n_samples)
```

```

bic_clayton = -2 * clayton_copula_loglik(theta_clayton,u1, u2) +
n_params_clayton * np.log(n_samples)
bic_gumbel = -2 * gumbel_copula_loglik(theta_gumbel,u1, u2) +
n_params_gumbel * np.log(n_samples)
bic_frank = -2 * frank_copula_loglik(theta_frank,u1, u2) +
n_params_frank * np.log(n_samples)

models = ['Gaussian', 't-Student', 'Clayton', 'Gumbel', 'Frank']
log_likelihoods = [gaussian_copula_loglik(rho_gaussian,u1, u2),
t_copula_loglik([rho_student, nu_t],u1, u2),
clayton_copula_loglik(theta_clayton,u1, u2),
gumbel_copula_loglik(theta_gumbel,u1, u2),
frank_copula_loglik(theta_frank,u1, u2)]
aics = [aic_gaussian, aic_t, aic_clayton, aic_gumbel, aic_frank]
bics = [bic_gaussian, bic_t, bic_clayton, bic_gumbel, bic_frank]
params = [f"rho={rho_gaussian:.4f}", f"rho={rho_student:.4f}",
nu={nu_t:.4f}", f"theta={theta_clayton:.4f}",
f"theta={theta_gumbel:.4f}", f"theta={theta_frank:.4f}"]

# Creazione di un DataFrame per il confronto
results_df = pd.DataFrame({
    'Modello': models,
    'Parametri': params,
    'Log-Verosimiglianza': log_likelihoods,
    'AIC': aics,
    'BIC': bics
})

# Ordinamento per AIC crescente (il più basso è il migliore)
results_df = results_df.sort_values('AIC')
print(results_df)

# Identifica il miglior modello
best_model = results_df.iloc[0]
print(f"\nIl modello migliore è la copula {best_model['Modello']} con
{best_model['Parametri']}")
print(f"AIC: {best_model['AIC']:.4f}, BIC: {best_model['BIC']:.4f}")

```

Riassunto dati finali

Modello	Parametri	Log-Verosimiglianza
t-Student	$\rho = 0.8603, \nu = 6.3485$	119436.683443
Gaussian	$\rho = 0.8602$	31387.260783
Clayton	$\theta = 2.7046$	20215.123381
Gumbel	$\theta = 1.7410$	7954.291589
Frank	$\theta = 2.9536$	-88617.940953

Tabella 5.14. Confronto dei modelli: Parametri e Log-Verosimiglianza

Modello	AIC	BIC
t-Student	-238869.366887	-238851.834301
Gaussiana	-62772.521566	-62763.755273
Clayton	-40428.246761	-40419.480468
Gumbel	-15906.583179	-15897.816886
Frank	177237.881906	177246.648199

Tabella 5.15. Confronto dei modelli: AIC e BIC

Il modello migliore è la copula **t-Student** con:

$$\rho = 0.8603$$

$$\nu = 6.3485$$

I criteri di selezione del modello sono:

$$\text{AIC} = -238869.3669$$

$$\text{BIC} = -238851.8343$$

Parte V

Simulazione e Ottimizzazione di un Portafoglio con Copule

Capitolo 6

Esempio simulazione portafoglio

6.1 Simulazione di un Portafoglio Long con Copule

La simulazione del portafoglio long viene effettuata utilizzando dati relativi al DAX e sfruttando la struttura di dipendenza stimata tramite copule. In particolare, l'obiettivo è valutare l'evoluzione di un portafoglio ottimale su un orizzonte temporale annuale, considerando rendimenti attesi positivi e una correlazione tra i titoli modellata con una copula gaussiana.

6.1.1 Impostazione dei Parametri di Simulazione

La prima fase del codice riguarda la definizione dei parametri fondamentali per la simulazione Monte Carlo. Essi includono:

Rendimenti attesi annualizzati

- Il rendimento atteso per il prezzo di apertura (*Open*) è fissato al 7% annuo.
- Il rendimento atteso per il prezzo di chiusura (*Close*) è leggermente superiore, pari al 7.5% annuo.
- Queste stime si basano su dati storici del DAX, considerando una crescita media sostenibile nel lungo periodo.

Volatilità annualizzata

- La volatilità dei rendimenti del prezzo di apertura è impostata a 0.79% annuo.
- La volatilità dei rendimenti del prezzo di chiusura è leggermente superiore, pari a 0.81% annuo.
- Questi valori derivano dal calcolo della volatilità rolling calcolata su dati storici.

Correlazione tra i rendimenti

- Si assume una correlazione di **0.95**, ottenuta dalla copula gaussiana stimata sui dati storici.
- Questa alta correlazione indica una forte dipendenza tra i due asset, tipica di titoli appartenenti allo stesso indice di mercato.

Tasso risk-free

- Il tasso di interesse privo di rischio viene fissato al **2% annuo**, un valore comunemente usato nelle simulazioni finanziarie per attualizzare i rendimenti. Di solito per stimare il risk-free rate si utilizzano gli interessi dei bond tedeschi.

Parametri di Simulazione Monte Carlo

- Si generano **10.000 simulazioni** per ottenere una distribuzione statistica robusta degli scenari di rendimento del portafoglio.
- L'orizzonte temporale è di **252 giorni di trading**, corrispondente a un anno lavorativo nei mercati finanziari.

Questa fase iniziale è fondamentale poiché definisce le ipotesi di base per la simulazione. L'uso delle copule permette di modellare la dipendenza tra i rendimenti degli asset in modo più flessibile rispetto a modelli tradizionali basati sulla correlazione lineare.

Implementazione python

Librerie necessarie per l'esecuzione del codice.

```
print("\n SIMULAZIONE PORTAFOGLIO LONG CON COPULE")
print("=" * 70)

# 1. Impostazione dei parametri di simulazione
print(" Impostazione parametri di simulazione")
print("-" * 50)

# Rendimenti attesi annualizzati positivi, basati su stime a lungo termine
# Per il DAX, un rendimento medio annuo del 6-8% è storicamente plausibile
expected_return_open = 0.07 # 7% annuo
expected_return_close = 0.075 # 7.5% annuo

# Volatilità dai dati originali (dall'output precedente)
volatility_open = 0.0079 # 0.79% annualizzato
volatility_close = 0.0081 # 0.81% annualizzato

# Correlazione, possiamo usare quella stimata dalla copula
# Supponiamo che la copula Gaussiana abbia dato rho = 0.95
correlation = 0.95

# Tasso risk-free
risk_free_rate = 0.02 # 2% annuo

# Parametri per la simulazione Monte Carlo
n_simulations = 10000
time_horizon = 252 # Un anno di trading

print(f"Rendimento atteso Open: {expected_return_open:.2%}")
print(f"Rendimento atteso Close: {expected_return_close:.2%}")
print(f"Volatilità Open: {volatility_open:.4f}")
print(f"Volatilità Close: {volatility_close:.4f}")
```

```
print(f"Correlazione: {correlation:.4f}")
print(f"Tasso risk-free: {risk_free_rate:.2%}")
print(f"Numero simulazioni: {n_simulations}")
print(f"Orizzonte temporale: {time_horizon} giorni")
```

Parametro	Valore
Rendimento atteso Open	7.00%
Rendimento atteso Close	7.50%
Volatilità Open	0.0079
Volatilità Close	0.0081
Correlazione	0.9500
Tasso risk-free	2.00%
Numero simulazioni	10.000
Orizzonte temporale	252 giorni

Tabella 6.1. Impostazione parametri di simulazione

6.1.2 Ottimizzazione del Portafoglio con i Nuovi Parametri

Dopo aver impostato i parametri di simulazione, si procede con l'ottimizzazione del portafoglio utilizzando tecniche di minimizzazione del rischio e massimizzazione del rendimento. L'analisi prevede la costruzione di due portafogli distinti:

1. **Portafoglio con massimo Sharpe Ratio** – ottimizzato per massimizzare il rapporto di Sharpe. Portafoglio utilizzato se l'obiettivo è ottenere il miglior rendimento possibile per ogni unità di rischio.
2. **Portafoglio a varianza minima** – ottimizzato per ridurre al minimo la volatilità complessiva. Portafoglio utilizzato se l'obiettivo principale è minimizzare il rischio, indipendentemente dal rendimento atteso. Solitamente una varianza elevata indica una maggiore dispersione dei rendimenti rispetto al valore medio, il che si traduce in una maggiore incertezza e, quindi, in un rischio più alto per l'investitore.

Definizione della Matrice di Covarianza e dei Rendimenti Attesi

Per modellare la relazione tra gli asset, si costruisce la matrice di covarianza:

$$\Sigma = \begin{bmatrix} \sigma_{open}^2 & \rho \cdot \sigma_{open} \cdot \sigma_{close} \\ \rho \cdot \sigma_{open} \cdot \sigma_{close} & \sigma_{close}^2 \end{bmatrix}$$

dove:

- σ_{open} e σ_{close} sono le volatilità annualizzate dei due asset,
- ρ è la correlazione stimata tra gli asset.

Il vettore dei rendimenti attesi è definito come:

$$\mu = \begin{bmatrix} \mu_{open} \\ \mu_{close} \end{bmatrix}$$

dove μ_{open} e μ_{close} sono i rendimenti attesi annualizzati.

Funzioni di Ottimizzazione

Si definiscono tre funzioni chiave per l'ottimizzazione del portafoglio:

- **Volatilità del portafoglio:** il rischio del portafoglio è dato da:

$$\sigma_p = \sqrt{w^T \Sigma w}$$

dove w è il vettore dei pesi del portafoglio.

- **Rendimento del portafoglio:** il rendimento atteso del portafoglio è:

$$R_p = \sum w_i \cdot \mu_i$$

- **Sharpe Ratio:** l'obiettivo è massimizzare il rapporto di Sharpe, definito come:

$$SR = \frac{R_p - R_f}{\sigma_p}$$

dove R_f è il risk-free rate.

L'ottimizzazione dello Sharpe Ratio viene eseguita minimizzando il valore negativo della funzione sopra descritta.

Vincoli e Limiti

L’ottimizzazione del portafoglio è soggetta ai seguenti vincoli:

- **Somma dei pesi pari a 1:**

$$\sum w_i = 1$$

- **Limiti sui pesi:** ogni asset deve avere un’allocazione compresa tra 0 e 1 (non è permessa la vendita allo scoperto).

Ottimizzazione del Portafoglio con Massimo Sharpe Ratio

Si utilizza un metodo di ottimizzazione numerica per trovare il portafoglio ottimale in termini di Sharpe Ratio. Il risultato fornisce:

- Allocazione ottimale tra i due asset (% in *Open* e *Close*).
- Rendimento atteso annualizzato.
- Rischio annualizzato.
- Sharpe Ratio.

Ottimizzazione del Portafoglio a Varianza Minima

Analogamente, si effettua un’ottimizzazione volta a minimizzare la volatilità complessiva del portafoglio. Il risultato fornisce:

- Allocazione ottimale con minor rischio.
- Rendimento atteso annualizzato.
- Rischio annualizzato.
- Sharpe Ratio associato.

Questa analisi permette di confrontare le due strategie e scegliere l’allocazione ottimale in base alla propensione al rischio dell’investitore.

Implementazione e risultati

```

print(" Ottimizzazione del portafoglio")
print("-" * 50)

# Creiamo la matrice di covarianza
cov_matrix = np.array([
[volatility_open**2, correlation * volatility_open * volatility_close],
[correlation * volatility_open * volatility_close, volatility_close**2]
])

# Vettore dei rendimenti attesi
expected_returns = np.array([expected_return_open, expected_return_close])

# Funzione per calcolare il rischio del portafoglio
def portfolio_volatility(weights, cov_matrix):

```

```

return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

# Funzione per calcolare il rendimento del portafoglio
def portfolio_return(weights, expected_returns):
    return np.sum(weights * expected_returns)

# Funzione per calcolare lo Sharpe Ratio negativo (da minimizzare)
def neg_sharpe_ratio(weights, expected_returns, cov_matrix, risk_free_rate):
    p_ret = portfolio_return(weights, expected_returns)
    p_vol = portfolio_volatility(weights, cov_matrix)
    return -(p_ret - risk_free_rate) / p_vol

# Vincoli: la somma dei pesi deve essere 1
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

# Limiti: ogni peso deve essere tra 0 e 1
bounds = tuple((0, 1) for _ in range(2))

# Ottimizzazione per massimizzare lo Sharpe Ratio
initial_weights = [0.5, 0.5]
optimal_sharpe = minimize(neg_sharpe_ratio, initial_weights,
                           args=(expected_returns, cov_matrix, risk_free_rate),
                           method='SLSQP', bounds=bounds, constraints=constraints)

optimal_weights_sharpe = optimal_sharpe['x']
optimal_return_sharpe = portfolio_return(optimal_weights_sharpe,
                                           expected_returns)
optimal_volatility_sharpe = portfolio_volatility(optimal_weights_sharpe,
                                                   cov_matrix)
optimal_sharpe_ratio = (optimal_return_sharpe - risk_free_rate) /
    optimal_volatility_sharpe

print(f"Portafoglio con massimo Sharpe Ratio:")
print(f"Allocazione in Open: {optimal_weights_sharpe[0]*100:.2f}%")
print(f"Allocazione in Close: {optimal_weights_sharpe[1]*100:.2f}%")
print(f"Rendimento atteso annualizzato: {optimal_return_sharpe*100:.2f}%")
print(f"Rischio annualizzato: {optimal_volatility_sharpe*100:.2f}%")
print(f"Sharpe Ratio: {optimal_sharpe_ratio:.4f}")

# Ottimizzazione per il portafoglio a minima varianza
def min_variance(weights, cov_matrix):
    return portfolio_volatility(weights, cov_matrix)

optimal_variance = minimize(min_variance, initial_weights,
                            args=(cov_matrix,), method='SLSQP', bounds=bounds, constraints=constraints)

optimal_weights_var = optimal_variance['x']
optimal_return_var = portfolio_return(optimal_weights_var,
                                       expected_returns)
optimal_volatility_var = portfolio_volatility(optimal_weights_var,
                                               cov_matrix)

```

```
cov_matrix)
optimal_sharpe_ratio_var = (optimal_return_var - risk_free_rate) /
optimal_volatility_var

print(f"\nPortafoglio a varianza minima:")
print(f"Allocazione in Open: {optimal_weights_var[0]*100:.2f}%")
print(f"Allocazione in Close: {optimal_weights_var[1]*100:.2f}%")
print(f"Rendimento atteso annualizzato: {optimal_return_var*100:.2f}%")
print(f"Rischio annualizzato: {optimal_volatility_var*100:.2f}%")
print(f"Sharpe Ratio: {optimal_sharpe_ratio_var:.4f}")
```

Parametro	Portafoglio Max Sharpe	Portafoglio Min Varianza
Allocazione in Open	0.00%	50.00%
Allocazione in Close	100.00%	50.00%
Rendimento atteso annualizzato	7.50%	7.25%
Rischio annualizzato	0.81%	0.79%
Sharpe Ratio	6.7901	6.6461

Tabella 6.2. Confronto tra il portafoglio a massimo Sharpe Ratio e il portafoglio a minima varianza

Da notare che entrambi i portafogli hanno uno sharpe ratio molto elevato, inoltre i dati ottenuti sono coerenti con le premesse iniziali. Il portafoglio massimo sharpe ha un rendimento atteso superiore al portafoglio di minima varianza che in compenso presenta un rischio minore.

6.1.3 Simulazione Monte Carlo con la Copula Stimata

Dopo aver definito l’allocazione ottimale del portafoglio, si procede con una **simulazione Monte Carlo** utilizzando la **copula Gaussiana stimata** sui dati storici. Questa tecnica permette di modellare la dipendenza tra i due asset e di analizzare il comportamento del portafoglio in scenari futuri.

Modello della Copula Gaussiana

Per fare un esempio di facile comprensione, simuliamo i rendimenti futuri degli asset con una copula Gaussiana che conserva la correlazione storica ρ stimata tra i due titoli. La copula Gaussiana è generata con una matrice di covarianza normalizzata:

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

I passi principali della simulazione sono:

1. Generazione di variabili casuali normali multivariate con media 0 e covarianza Σ .
2. Trasformazione delle variabili normali in uniformi usando la funzione di ripartizione cumulativa (CDF).
3. Conversione delle variabili uniformi in rendimenti assumendo una distribuzione normale per semplicità.

Questa tecnica permette di preservare la struttura di dipendenza stimata dai dati reali.

Simulazione delle Traiettorie dei Prezzi

Per ciascun asset (*Open* e *Close*), si simula una traiettoria di prezzo su un orizzonte temporale di **252 giorni**.

Si parte da un prezzo iniziale di 100 e si aggiornano i prezzi giornalieri utilizzando i rendimenti simulati.

La formula per aggiornare il prezzo P_t a ogni passo temporale t è:

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

dove r_t è il rendimento giornaliero simulato.

Per ogni iterazione della simulazione Monte Carlo:

- Si genera un nuovo set di rendimenti usando la copula.
- Si aggiorna il prezzo per ogni giorno del periodo simulato.

Simulazione del Portafoglio Ottimale

Dopo aver simulato le traiettorie dei singoli asset, si calcola l’evoluzione del valore del portafoglio ottimale con pesi ottimali derivati dalla massimizzazione dello Sharpe Ratio:

$$P_t^{(portafoglio)} = w_{open} \cdot P_t^{(open)} + w_{close} \cdot P_t^{(close)}$$

dove w_{open} e w_{close} sono i pesi del portafoglio a massimo Sharpe.

Analisi delle Statistiche di Rischio

Al termine della simulazione, vengono calcolate le seguenti metriche di rischio:

- **Valore Finale Atteso:** il valore medio finale del portafoglio simulato, che permette di stimare il rendimento atteso.
- **Value at Risk (VaR) 5%:** la soglia di perdita massima che il portafoglio potrebbe subire con una probabilità del 5%.

$$VaR_{5\%} = \text{Percentile}_5(P_T^{(portafoglio)})$$

- **Conditional Value at Risk (CVaR) 5%:** la perdita media nei peggiori scenari (ovvero nei casi in cui il portafoglio subisce una perdita superiore al VaR).

$$CVaR_{5\%} = \mathbb{E}[P_T^{(portafoglio)} | P_T^{(portafoglio)} \leq VaR_{5\%}]$$

Il CVaR è una misura più conservativa rispetto al VaR perché tiene conto dell'entità delle perdite oltre la soglia del 5%.

Interpretazione dei Risultati

I risultati della simulazione Monte Carlo permettono di valutare il comportamento del portafoglio ottimale in condizioni di mercato variabili.

- Un **elevato valore finale atteso** indica che il portafoglio è in grado di generare rendimenti positivi nel lungo termine.
- Un **VaR contenuto** segnala che la probabilità di perdite eccessive è ridotta.
- Un **CVaR basso** indica che, anche negli scenari peggiori, le perdite rimangono gestibili.

Questa analisi aiuta gli investitori a prendere decisioni informate sulla gestione del rischio e sulla robustezza della strategia di investimento.

Implementazione Python

```
print(" Simulazione Monte Carlo con copula")
print("-" * 50)

# Useremo la copula Gaussiana con il parametro rho stimato
rho = correlation

# Genera simulazioni dalla copula Gaussiana
mean = [0, 0]
cov = [[1, rho], [rho, 1]]
z = np.random.multivariate_normal(mean, cov, n_simulations)
simulated_uniform = stats.norm.cdf(z)

# Trasforma le simulazioni uniformi in rendimenti
# Assumiamo una distribuzione normale per semplicità
sim_returns_open = stats.norm.ppf(simulated_uniform[:, 0],
expected_return_open/time_horizon,
volatility_open/np.sqrt(time_horizon))
sim_returns_close = stats.norm.ppf(simulated_uniform[:, 1],
expected_return_close/time_horizon,
volatility_close/np.sqrt(time_horizon))

# Crea traiettorie di prezzi per entrambi gli asset
initial_price = 100
price_paths_open = np.zeros((n_simulations, time_horizon))
price_paths_close = np.zeros((n_simulations, time_horizon))

for i in range(n_simulations):
# Inizializza con il prezzo iniziale
price_paths_open[i, 0] = initial_price
price_paths_close[i, 0] = initial_price

# Simula la traiettoria dei prezzi
for t in range(1, time_horizon):
# Genera nuovi rendimenti per ogni passo temporale
new_z = np.random.multivariate_normal(mean, cov, 1)
new_uniform = stats.norm.cdf(new_z)

new_return_open = stats.norm.ppf(new_uniform[0, 0],
expected_return_open/time_horizon,
volatility_open/np.sqrt(time_horizon))
new_return_close = stats.norm.ppf(new_uniform[0, 1],
expected_return_close/time_horizon,
volatility_close/np.sqrt(time_horizon))

# Aggiorna i prezzi
price_paths_open[i, t] = price_paths_open[i, t-1] * (1 + new_return_open)
price_paths_close[i, t] = price_paths_close[i, t-1] * (1 + new_return_close)

# Simula il portafoglio ottimale
```

```

portfolio_paths = optimal_weights_sharpe[0] * price_paths_open +
    optimal_weights_sharpe[1] * price_paths_close

# Calcola statistiche delle simulazioni
final_portfolio_values = portfolio_paths[:, -1]
expected_final_value = np.mean(final_portfolio_values)
portfolio_var = np.percentile(final_portfolio_values, 5) # 5% VaR
portfolio_cvar = np.mean(final_portfolio_values[final_portfolio_values <=
    portfolio_var])

print(f"Statistiche del portafoglio simulato (Massimo Sharpe):")
print(f"Valore iniziale: {initial_price:.2f}")
print(f"Valore finale atteso: {expected_final_value:.2f}")
(rendimento: {(expected_final_value/initial_price - 1)*100:.2f}%)")
print(f"Value at Risk (5%): {portfolio_var:.2f}")
(perdita massima: {(portfolio_var/initial_price - 1)*100:.2f}%)")
print(f"Conditional VaR (5%): {portfolio_cvar:.2f}")

```

Statistiche del Portafoglio Simulato	Valore
Valore iniziale	100.00
Valore finale atteso	107.76
Rendimento atteso	7.76%
Value at Risk (5%)	106.32
Perdita massima (VaR 5%)	6.32%
Conditional VaR (5%)	105.96

Tabella 6.3. Statistiche della simulazione Monte Carlo per il portafoglio a massimo Sharpe

6.1.4 Visualizzazione dei risultati

Dopo aver ottimizzato il portafoglio e simulato le sue traiettorie future utilizzando una copula Gaussiana, questa sezione è dedicata alla visualizzazione grafica dei risultati. La rappresentazione visiva aiuta a comprendere meglio la relazione tra rischio e rendimento, le traiettorie simulate e la distribuzione dei valori finali del portafoglio.

1. Frontiera efficiente del Portafoglio

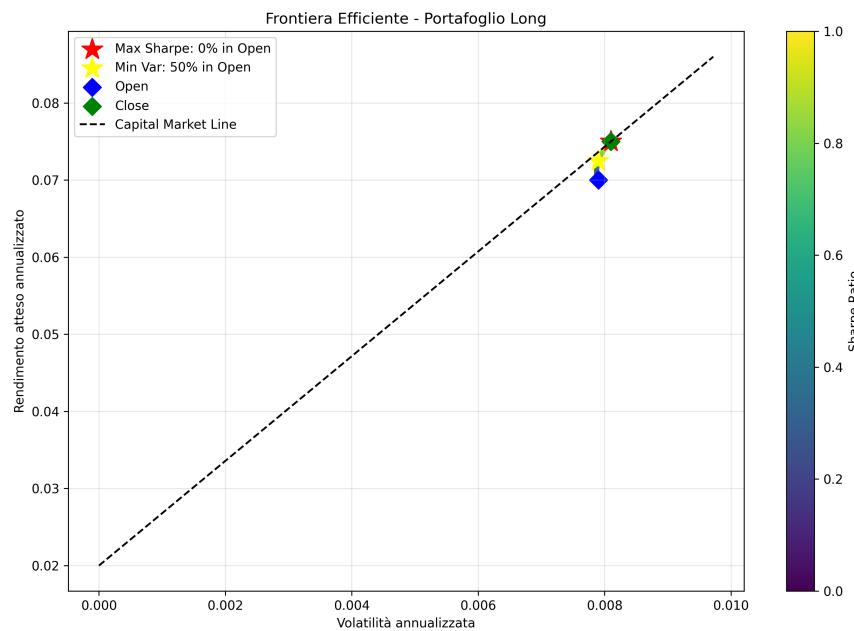


Figura 6.1. Frontiera efficiente - portafoglio long

Nel grafico risultante:

- I punti rappresentano i diversi portafogli, colorati in base al loro Sharpe Ratio.
- Il portafoglio a massimo Sharpe Ratio** è evidenziato con un **asterisco rosso**.
- Il portafoglio a minima varianza** è evidenziato con un **asterisco giallo**.
- Sono inclusi anche i **singoli asset** (Open e Close) come punti di riferimento.
- La **Capital Market Line (CML)** è tracciata per mostrare la relazione tra rischio e rendimento quando si include l'attività priva di rischio.

Questa rappresentazione aiuta a identificare **quale portafoglio offre il miglior rendimento per ogni unità di rischio**.

2. Simulazione Monte Carlo delle Traiettorie del Portafoglio

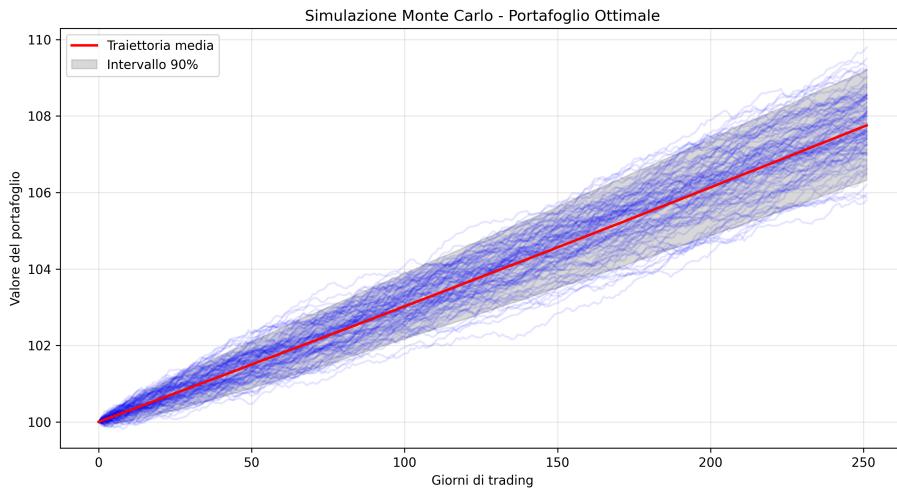


Figura 6.2. Simulazione monte carlo - portafoglio ottimale

Per visualizzare l'evoluzione del portafoglio nel tempo, si generano **100 traiettorie casuali** tra le 10.000 simulate.

- Ogni traiettoria mostra l'andamento nel tempo del valore del portafoglio ottimale.
- La **traiettoria media** è evidenziata in rosso.
- Sono inclusi gli **intervalli di confidenza al 90%**, evidenziando le fasce in cui è probabile che si trovi il valore del portafoglio nel tempo.

Questa rappresentazione consente di valutare la **volatilità del portafoglio nel tempo** e **l'incertezza associata** alle simulazioni.

3. Distribuzione dei Valori Finali del Portafoglio

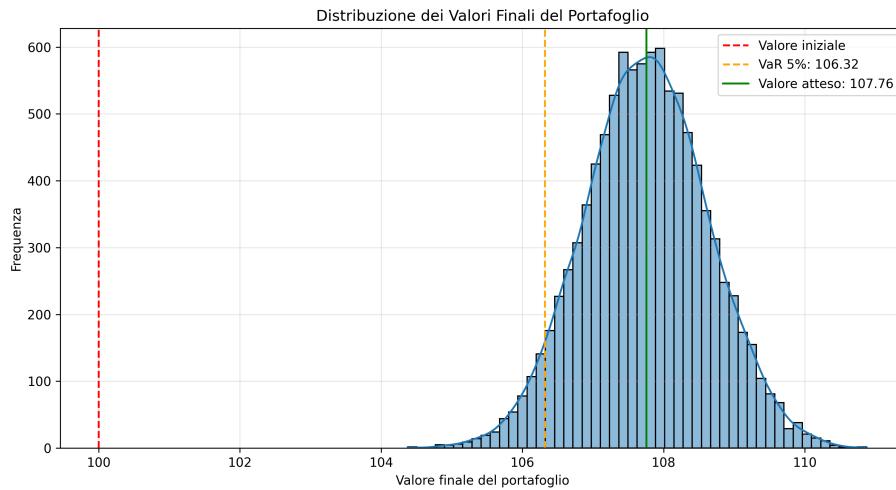


Figura 6.3. distribuzione valori finali portafoglio

Questo grafico mostra la **distribuzione dei valori finali del portafoglio simulato**.

- L'**istogramma** rappresenta la frequenza dei valori finali ottenuti nelle simulazioni Monte Carlo.
- Sono inclusi i seguenti riferimenti:
 - **Linea rossa tratteggiata:** valore iniziale del portafoglio.
 - **Linea arancione tratteggiata:** *Value at Risk (VaR 5%)*, ossia il valore che il portafoglio raggiunge o supera nel 95% dei casi.
 - **Linea verde continua:** valore finale atteso del portafoglio.

Questa rappresentazione è utile per quantificare la **probabilità di ottenere un rendimento positivo** e valutare i **rischi di perdita estrema**.

Bibliografia

Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula Methods in Finance*. Wiley Finance, 2004.

David G. Clayton. A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika*, 65:141–151, 1978.

Simone Demarta and Alexander J. McNeil. The t copula and related copulas. *International Statistical Review*, 73(1):111–129, 2005.

M. J. Frank. On the simultaneous associativity of $f(x, y)$ and $x + y - f(x, y)$. *Aequationes Mathematicae*, 19:194–226, 1979.

Emil J. Gumbel. Bivariate exponential distributions. *Journal of the American Statistical Association*, 55:698–707, 1960.

Harry Joe. *Dependence Modeling with Copulas*. Chapman and Hall/CRC, 2014.

Alexander J. McNeil, Rüdiger Frey, and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, 2015.

Roger B. Nelsen. *An Introduction to Copulas*. Springer, New York, 2nd edition, 2006.

Andrea Rostagno. Dax copula analysis thesis. <https://github.com/Andrea-Rostagno/DAX-Copula-Analysis-Thesis.git>, 2024.