

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



**OGGETTO: Report – Financial Big Data Pipeline**

<b>INSEGNAMENTO</b>	Strutture e Algoritmi Big Data
<b>DOCENTE</b>	Giovanni Farina
<b>STUDENTE</b>	Alfredo La Rosa
<b>MATRICOLA</b>	IN32000135
<b>TIPO DI DOCUMENTO</b>	<i>Project Report</i>
<b>DATA DI APPELLO</b>	Febbraio 2026

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## Sommario

1. Introduzione .....	4
1.1 Scopo del progetto.....	4
1.2 Obiettivi tecnici.....	4
1.3 Ambito del sistema .....	4
1.4 Struttura del documento .....	4
2. Data Sources e Caratteristiche dei Dati .....	5
2.1 Panoramica delle fonti dati.....	5
2.2 Dati finanziari intraday.....	5
2.3 Dataset delle news finanziarie.....	5
2.4 Caratteristiche Big Data dei dataset .....	6
2.5 Considerazioni progettuali.....	6
3. Progettazione della Data Pipeline.....	7
3.1 Overview Architetturale .....	7
3.2 Data Ingestion Layer .....	8
3.3 Distributed Storage – Data Lake su HDFS .....	8
3.4 Data Processing Layer – Apache Spark .....	8
3.5 Polyglot Persistence – Storage Operativo .....	9
3.6 Considerazioni Progettuali.....	9
4. Analytics e Analisi dei Dati.....	10
4.1 Obiettivo del Modulo Analitico.....	10
4.2 Dataset Utilizzati.....	10
4.3 Analisi del Trend di Mercato .....	10
4.4 Integrazione News–Prezzo.....	10
4.5 Logica di Validazione.....	10
4.6 Output del Processo Analitico .....	11
4.7 Ruolo del Layer Analytics nella Pipeline .....	11
5. Implementazione del Sistema .....	12
5.1 Ambiente di sviluppo.....	12
5.2 Containerizzazione dell'infrastruttura .....	12
5.3 Inizializzazione dei servizi .....	12
5.4 Implementazione dei Job Spark.....	13
6. Testing e Validazione del Sistema .....	15
6.1 Strategia di Validazione .....	15
6.2 Avvio Infrastruttura Docker .....	15
6.3 Inizializzazione dell'Infrastruttura Dati .....	15

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



6.3.1 Inizializzazione HDFS.....	15
6.3.2 Inizializzazione Cassandra .....	17
6.3.3 Inizializzazione MongoDB .....	17
6.4 Fase della Data Ingestion .....	18
6.4.1 Ingestion dei dati intraday .....	18
6.4.2 Ingestion delle news finanziarie .....	19
6.5 Automazione della pipeline Spark .....	20
6.5.1 Validazione della conversione Spark (Bronze → Silver Layer) .....	21
6.5.2 Validazione del popolamento Cassandra.....	21
6.5.3 Validazione del popolamento MongoDB .....	22
6.6 Validazione del Modulo Analytics.....	22
7 Conclusioni Tecniche e Scalabilità del Framework .....	24
Appendice.....	25

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 1. Introduzione

### 1.1 Scopo del progetto

Il presente documento descrive la progettazione, l'implementazione e il testing di una **data pipeline** **Big Data** per la raccolta, gestione e analisi di dati finanziari utilizzati in ambito trading.

Il sistema sviluppato consente l'acquisizione automatica di dati provenienti da fonti esterne, la loro memorizzazione in un'infrastruttura distribuita e l'esecuzione di analisi sui dati mediante tecnologie Big Data.

In particolare, il progetto prevede:

- acquisizione automatica di dati finanziari intraday e news di mercato;
- memorizzazione dei dati grezzi in ambiente distribuito HDFS;
- processamento dei dati tramite Apache Spark;
- utilizzo di un'architettura **polyglot persistence** mediante database NoSQL differenti;
- esecuzione di analisi finalizzate all'identificazione di possibili segnali operativi basati su indicatori tecnici e pubblicazione di notizie.

Il sistema è stato progettato per essere completamente riproducibile tramite container Docker.

### 1.2 Obiettivi tecnici

Gli obiettivi principali del progetto sono:

- progettare una pipeline di gestione dati scalabile;
- gestire dataset eterogenei caratterizzati da diversa struttura e frequenza di aggiornamento;
- separare le componenti di ingestion, processing, storage e analytics;
- ottimizzare la memorizzazione dei dati in funzione del tipo di accesso previsto;
- realizzare un ambiente facilmente distribuibile e replicabile.

### 1.3 Ambito del sistema

Il sistema gestisce due tipologie di dati finanziari:

- **dati di mercato intraday** con frequenza di campionamento pari a 2 minuti;
- **dati giornalieri relativi a news finanziarie** associate ai titoli analizzati.

I dati vengono acquisiti dalla piattaforma Yahoo Finance mediante script automatici sviluppati in Python.

La pipeline implementata comprende le seguenti fasi:

1. acquisizione dei dati;
2. caricamento dei dati grezzi in HDFS;
3. trasformazione e ottimizzazione dei formati tramite Apache Spark;
4. caricamento nei database operativi;
5. analisi distribuita dei dati.

### 1.4 Struttura del documento

Il documento è organizzato come segue:

- Capitolo 2: descrizione delle fonti dati e delle caratteristiche dei dataset;
- Capitolo 3: progettazione della data pipeline e dell'architettura del sistema;
- Capitolo 4: analytics e query sviluppate;
- Capitolo 5: dettagli implementativi e ambiente di esecuzione;
- Capitolo 6: attività di testing e validazione del sistema;
- Capitolo 7: valutazioni finali e possibili evoluzioni.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 2. Data Sources e Caratteristiche dei Dati

### 2.1 Panoramica delle fonti dati

Il sistema utilizza dati finanziari provenienti dalla piattaforma **Yahoo Finance**, selezionata in quanto fornisce accesso programmatico a dati di mercato storici e informazioni testuali relative alle notizie finanziarie.

Le fonti dati considerate sono suddivise in due categorie principali:

- dati di mercato intraday;
- dati informativi relativi alle news finanziarie.

Le due tipologie presentano caratteristiche strutturali e frequenze di aggiornamento differenti, rendendo necessaria l'adozione di strategie di storage e processamento dedicate.

### 2.2 Dati finanziari intraday

I dati intraday rappresentano serie temporali dei prezzi di mercato relative a un insieme di titoli azionari selezionati.

#### Caratteristiche principali

Proprietà	Descrizione
Fonte	Yahoo Finance
Frequenza	2 minuti
Tipo dato	Strutturato
Formato origine	CSV
Aggiornamento	Automatico
Natura	Serie temporale

Ogni record contiene le seguenti informazioni:

- timestamp;
- prezzo di apertura (*Open*);
- prezzo massimo (*High*);
- prezzo minimo (*Low*);
- prezzo di chiusura (*Close*);
- volume di scambio.

Questa tipologia di dati genera un elevato numero di record nel tempo, risultando particolarmente adatta a sistemi distribuiti ottimizzati per scritture frequenti e query su intervalli temporali.

### 2.3 Dataset delle news finanziarie

Il secondo dataset è costituito da informazioni testuali relative alle news finanziarie associate ai titoli analizzati.

#### Caratteristiche principali

Proprietà	Descrizione
Fonte	Yahoo Finance
Frequenza	Giornaliera
Tipo dato	Semi-strutturato
Formato origine	CSV / JSON

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



Contenuto	Testuale
Natura	Documento

Ogni elemento informativo include:

- identificativo del titolo;
- data di pubblicazione;
- titolo della notizia;
- descrizione o contenuto sintetico;
- sorgente della notizia.

A differenza dei dati intraday, le news presentano struttura flessibile e contenuti testuali variabili, richiedendo un sistema di memorizzazione orientato ai documenti.

## 2.4 Caratteristiche Big Data dei dataset

I dataset utilizzati presentano proprietà riconducibili al paradigma Big Data.

### Volume

L'acquisizione continua dei dati intraday produce un'elevata quantità di record nel tempo, con crescita proporzionale al numero di titoli monitorati e alla durata della raccolta dati.

### Velocity

I dati di mercato vengono aggiornati con frequenza elevata (2 minuti), richiedendo meccanismi di ingestione automatica e scrittura efficiente.

### Variety

Il sistema gestisce contemporaneamente:

- dati strutturati numerici (prezzi di mercato);
- dati semi-strutturati testuali (news finanziarie).

Questa eterogeneità ha motivato l'adozione di un'architettura di storage poliglotta descritta nei capitoli successivi.

## 2.5 Considerazioni progettuali

L'analisi delle caratteristiche delle fonti dati ha guidato le principali scelte architettoniche del sistema:

- utilizzo di **HDFS** per la memorizzazione dei dati grezzi;
- impiego di **Apache Spark** per l'elaborazione distribuita;
- utilizzo di database differenti in funzione del modello dati:
  - Cassandra per serie temporali ad alta frequenza;
  - MongoDB per dati documentali relativi alle news.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 3. Progettazione della Data Pipeline

#### 3.1 Overview Architetturale

Il sistema è progettato come una **pipeline Big Data end-to-end** finalizzata alla gestione di dati finanziari eterogenei provenienti da sorgenti esterne.

L'architettura adotta un approccio modulare basato sulla separazione delle responsabilità tra le diverse fasi di gestione del dato:

- acquisizione;
- storage distribuito;
- processamento;
- persistenza operativa;
- analytics.

La pipeline implementa una struttura multilivello ispirata al modello **Data Lake (Bronze–Silver–Gold)**.

Layer	Funzione
Bronze	Dati grezzi acquisiti
Silver	Dati trasformati e ottimizzati
Gold	Database operativi per analytics

L'infrastruttura è stata progettata per gestire flussi di dati eterogenei in un ambiente completamente containerizzato tramite **Docker**. Il flusso segue tre stadi logici (Bronze, Silver, Gold). Di seguito è rappresentato lo schema dell'architettura:

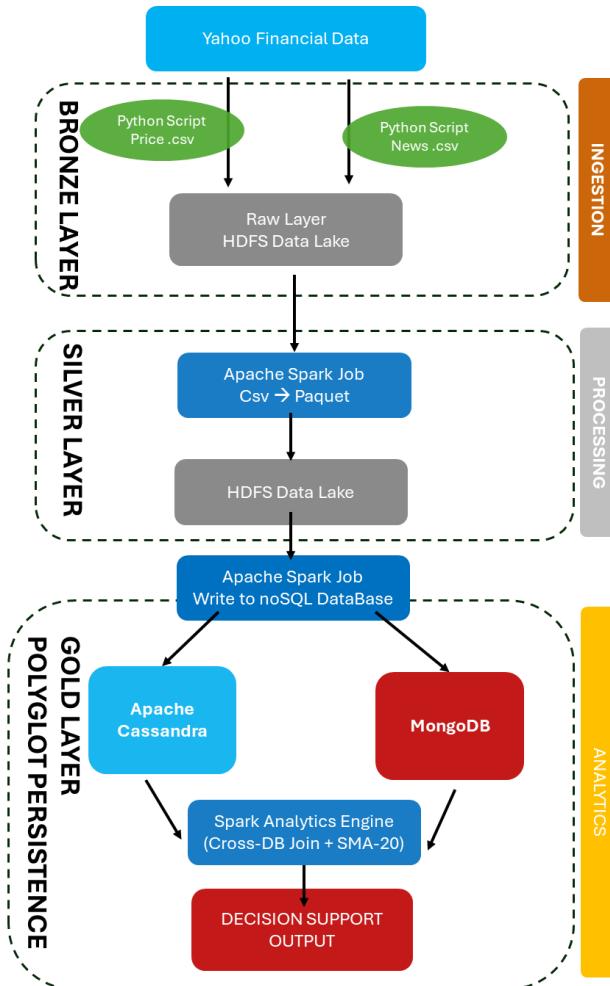


Figura 1: Architettura

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 3.2 Data Ingestion Layer

La fase di ingestion è responsabile della raccolta automatica dei dati da Yahoo Finance.

Sono state implementate due pipeline indipendenti sviluppate in Python:

#### Flusso prezzi (Structured Stream)

Lo script di acquisizione:

- interroga periodicamente i ticker selezionati;
- estrae dati OHLCV con granularità temporale di 2 minuti;
- genera dataset CSV;
- trasferisce i dati direttamente su HDFS tramite protocollo WebHDFS.

I dati vengono trasmessi mediante buffer in memoria (*StringIO*) evitando scritture locali intermedie.

#### Flusso news (Semi-structured Stream)

Le news finanziarie vengono acquisite sotto forma di feed JSON contenenti informazioni testuali associate ai titoli monitorati.

Durante l'ingestion è stata implementata una logica di **Safe Ingestion** per gestire record incompleti:

- in presenza di timestamp mancanti;
- viene assegnato automaticamente il tempo di ricezione del dato.

Questa soluzione garantisce continuità operativa della pipeline.

### 3.3 Distributed Storage – Data Lake su HDFS

HDFS viene utilizzato come **Data Lake centrale** per la memorizzazione persistente dei dati.

La struttura adottata è la seguente:

```
/data/raw/prices/  
/data/raw/news/  
/data/processed/prices/
```

Il Data Lake consente:

- disaccoppiamento tra ingestion e processing;
- fault tolerance tramite replica;
- accesso parallelo ai job Spark;
- conservazione del dato originale.

Il layer /raw rappresenta il **Bronze Layer** della pipeline.

### 3.4 Data Processing Layer – Apache Spark

Apache Spark costituisce il motore di elaborazione distribuita del sistema.

Le attività di processing includono:

- lettura dei dataset raw da HDFS;
- pulizia e normalizzazione dei dati;
- gestione dei record incompleti;
- conversione dei dati in formato colonnare Parquet.

#### **Conversione CSV → Parquet**

I dati intraday vengono convertiti in formato Parquet con:

- compressione Snappy;
- accesso colonnare;
- supporto al partition pruning.

Il dataset viene partizionato fisicamente per ticker:

```
/data/processed/prices/prices_parquet/
```

Questo livello rappresenta il **Silver Layer** del Data Lake.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 3.5 Polyglot Persistence – Storage Operativo

Per ottimizzare le prestazioni di accesso ai dati è stata adottata un'architettura **Polyglot Persistence**.

#### Cassandra – Time Series Storage

Apache Cassandra è utilizzata per i dati intraday.

Motivazioni progettuali:

- elevato throughput in scrittura;
- distribuzione orizzontale;
- ottimizzazione per query temporali;
- gestione efficiente di serie storiche.

I dati vengono caricati tramite Spark Connector.

#### MongoDB – Document Storage

MongoDB gestisce le news finanziarie.

Il database consente:

- schema flessibile;
- gestione nativa JSON;
- indicizzazione su attributi testuali;
- eliminazione di duplicati tramite indice unico composto.

### 3.6 Considerazioni Progettuali

Le principali decisioni architettonurali derivano da:

- eterogeneità dei dataset;
- elevata frequenza dei dati di mercato;
- necessità di scalabilità orizzontale;
- separazione tra storage analitico e operativo.

L'approccio multilivello consente estensione futura verso pipeline streaming o modelli predittivi.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 4. Analytics e Analisi dei Dati

### 4.1 Obiettivo del Modulo Analitico

Il modulo di analytics ha lo scopo di estrarre informazioni operative dai dati finanziari gestiti dalla pipeline mediante l'integrazione tra eventi informativi di mercato e indicatori tecnici derivati dalle serie storiche dei prezzi.

L'analisi implementata realizza un processo di **validazione tecnica delle news finanziarie**, nel quale la pubblicazione di una notizia rappresenta un evento trigger successivamente valutato rispetto allo stato del mercato.

L'elaborazione viene eseguita tramite job distribuiti Apache Spark che operano direttamente sui database del layer operativo.

### 4.2 Dataset Utilizzati

L'analisi combina dati provenienti da due sistemi di storage differenti:

Dataset	Database	Tipologia
News finanziarie	MongoDB	Documentale
Prezzi intraday	Cassandra	Serie temporale

Le news rappresentano eventi discreti associati a specifici ticker, mentre i dati intraday descrivono l'evoluzione temporale del prezzo di mercato.

Prima dell'analisi viene effettuata una normalizzazione dei campi identificativi e temporali per garantire coerenza tra i dataset.

### 4.3 Analisi del Trend di Mercato

Per valutare lo stato tecnico di ciascun titolo viene calcolato un indicatore di trend basato su **media mobile semplice a 20 periodi (SMA-20)**.

L'indicatore consente di stimare la direzione prevalente del prezzo nel breve periodo:

- valori di prezzo superiori alla media indicano trend rialzista;
- valori inferiori indicano trend ribassista.

Il calcolo viene effettuato sulle serie temporali ordinate cronologicamente per ciascun ticker.

### 4.4 Integrazione News–Prezzo

Successivamente al calcolo dell'indicatore tecnico, ogni news viene associata allo stato di mercato del titolo nel giorno di pubblicazione.

L'integrazione avviene mediante correlazione temporale tra:

- ticker finanziario;
- data della news;
- dati di prezzo corrispondenti.

Questa operazione permette di contestualizzare ogni evento informativo rispetto alle condizioni tecniche del mercato nel medesimo intervallo temporale.

### 4.5 Logica di Validazione

Il sistema applica una regola decisionale finalizzata a stabilire se una notizia risulti coerente con il trend tecnico osservato.

La news viene considerata **validata** quando il prezzo del titolo risulta superiore alla media mobile calcolata nello stesso periodo temporale, indicando la presenza di un trend rialzista.

In caso contrario, la notizia viene classificata come non supportata dal contesto tecnico.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



Il risultato dell'analisi consiste quindi in una classificazione automatica degli eventi informativi secondo due categorie:

- **Trend positivo (Bullish)**
- **Trend negativo (Bearish)**

#### 4.6 Output del Processo Analitico

Il modulo analytics produce un dataset contenente, per ciascun evento informativo:

- ticker del titolo;
- data della notizia;
- informazioni testuali associate;
- valore del prezzo di mercato;
- indicatore tecnico calcolato;
- esito della validazione.

I risultati vengono ordinati temporalmente e resi disponibili per successive attività di analisi o backtesting.

#### 4.7 Ruolo del Layer Analytics nella Pipeline

Il modulo analitico rappresenta il livello finale della pipeline Big Data e dimostra la capacità del sistema di:

- integrare dati eterogenei provenienti da database differenti;
- eseguire analisi distribuite su grandi volumi di dati;
- trasformare dati grezzi in informazioni utilizzabili.

La separazione tra pipeline dati e logica analitica consente l'estensione futura verso modelli predittivi o sistemi di supporto alle decisioni.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 5. Implementazione del Sistema

### 5.1 Ambiente di sviluppo

L'intero sistema è stato sviluppato e testato in un ambiente locale containerizzato al fine di garantire riproducibilità e isolamento delle componenti.

L'ambiente di sviluppo utilizzato è composto da:

Componente	Tecnologia
IDE	Visual Studio Code
Containerizzazione	Docker Desktop
Orchestrazione	Docker Compose
Sistema Operativo Host	Windows
Interfaccia CLI	PowerShell

Docker Compose è stato utilizzato per orchestrare i diversi servizi Big Data simulando un'infrastruttura distribuita mediante rete virtuale dedicata (bdnet).

### 5.2 Containerizzazione dell'infrastruttura

Ogni componente della pipeline è eseguito all'interno di un container indipendente.

I principali servizi deployati sono:

- **HDFS NameNode e DataNode**
- **Apache Spark**
- **Apache Cassandra**
- **MongoDB**

La containerizzazione consente:

- isolamento delle dipendenze;
- gestione controllata delle versioni;
- avvio automatico dell'intero stack;
- portabilità del sistema su ambienti differenti.

L'infrastruttura viene inizializzata tramite comando:

- docker-compose up -d

### 5.3 Inizializzazione dei servizi

Per automatizzare il deployment sono stati sviluppati script di inizializzazione dedicati.

#### Inizializzazione HDFS

Uno script PowerShell (init\_hdfs.ps1) crea automaticamente la struttura del Data Lake:

/data/raw/prices

/data/raw/news

/data/processed/prices

Lo script viene eseguito dopo l'avvio del cluster HDFS per garantire la disponibilità delle directory richieste dalla pipeline.

#### Inizializzazione Cassandra

L'inizializzazione del database Cassandra avviene tramite script CQL eseguiti automaticamente nel container.

Le operazioni includono:

- creazione del keyspace;
- definizione delle tabelle per serie temporali;

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



- configurazione delle chiavi primarie ottimizzate per query temporali.

Esempio logico:

```
CREATE TABLE market.prices_by_ticker (
    ticker text,
    ts timestamp,
    open double,
    high double,
    low double,
    close double,
    volume double,
    PRIMARY KEY (ticker, ts)
);
```

### **Inizializzazione MongoDB**

MongoDB viene configurato tramite script JavaScript eseguiti all'avvio del container.

Le operazioni comprendono:

- creazione del database;
- creazione della collection news;
- definizione di indice unico composto per prevenire duplicazioni.

Esempio:

```
db.news.createIndex(
  { ticker: 1, title: 1 },
  { unique: true }
)
```

## 5.4 Implementazione dei Job Spark

La pipeline Spark è suddivisa in job indipendenti per garantire modularità.

### **Job 1 — Conversione dati in Parquet**

Responsabile della trasformazione dei dati raw presenti su HDFS.

Operazioni principali:

- lettura CSV intraday;
- pulizia dei campi;
- normalizzazione timestamp;
- partizionamento per ticker;
- scrittura in formato Parquet.

Output:

/data/processed/prices/prices\_parquet/

### **Job 2 — Caricamento Cassandra**

Questo job utilizza il **Spark Cassandra Connector** per trasferire i dati processati nel database operativo.

Durante l'implementazione è stata effettuata una revisione dello schema per garantire coerenza tra DataFrame Spark e struttura Cassandra.

### **Job 3 — Caricamento MongoDB**

Il job Spark dedicato alle news esegue:

- selezione campi rilevanti;

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



- rimozione record incompleti;
- inserimento documenti JSON in MongoDB.

Filtro applicato:

```
news_final = df_news.filter(col("title").isNotNull())
```

#### Job 4 — Analisi Dati

Il modulo di analytics estrae le informazioni operative dai dati finanziari gestiti dalla pipeline mediante l'integrazione tra eventi informativi di mercato e indicatori tecnici derivati dalle serie storiche dei prezzi e ne generano un indicatore di compravendita.

#### 5.5 Automazione della pipeline

L'esecuzione della pipeline segue il seguente ordine operativo:

##### 1. Bronze Layer (Dati Grezzi)

In questa fase avviene l'acquisizione dei dati dalle API di Yahoo Finance tramite script Python dedicati:

- Prezzi Intraday:** Estratti con frequenza di 2 minuti e salvati in formato **CSV**.
- News Finanziarie:** Acquisite sotto forma di feed **JSON** multi-linea.
- Storage:** Entrambi i dataset vengono caricati nel Data Lake su **HDFS** nelle directory /data/raw/.

##### 2. Silver Layer (Dati Ottimizzati)

Questa fase utilizza **Apache Spark** per trasformare i dati grezzi e migliorarne le performance di accesso:

- Trasformazione:** I file CSV dei prezzi vengono convertiti in formato **Parquet** con compressione Snappy.
- Partizionamento:** I dati vengono partizionati fisicamente per ticker (es. ticker=AAPL), permettendo il *partition pruning* durante le query.
- Storage:** Il risultato viene salvato nuovamente su **HDFS** nel percorso /data/processed/.

##### 3. Gold Layer (Polyglot Persistence)

Spark carica i dati raffinati nei database NoSQL specializzati per il tipo di dato:

- Apache Cassandra:** Utilizzato per i prezzi in formato Parquet, essendo ottimizzato per serie temporali e alti volumi di scrittura.
- MongoDB:** Utilizzato per le news in formato JSON, sfruttando lo schema flessibile per gestire documenti di testo semi-strutturati.

##### 4. Analytic Engine (Analisi Finale)

Il modulo finale (indicator\_spark.py) rappresenta il livello di **Business Intelligence**:

- Integrazione (Join):** Spark esegue una join cross-database tra MongoDB e Cassandra usando il ticker e la data come chiavi.
- Calcolo Indicatori:** Viene calcolata una **Media Mobile Semplice (SMA-20)** sui dati storici di Cassandra tramite Window Functions.
- Output:** Il sistema genera una tabella decisionale che classifica ogni notizia con un consiglio operativo (**APPROVE** in trend rialzista, **REJECT** in trend ribassista).

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 6. Testing e Validazione del Sistema

### 6.1 Strategia di Validazione

La validazione del sistema è stata effettuata seguendo l'intero flusso operativo della pipeline, simulando l'avvio dell'infrastruttura da ambiente inizialmente non operativo fino alla generazione dei risultati analitici finali.

Ogni fase è stata verificata tramite evidenze sperimentali ottenute durante l'esecuzione del sistema containerizzato.

### 6.2 Avvio Infrastruttura Docker

La prima fase di test ha previsto l'avvio dell'intero ecosistema Big Data tramite Docker Compose.

Comando eseguito:

- ***docker-compose up -d***

L'operazione avvia automaticamente:

- cluster HDFS;
- Apache Spark;
- Cassandra;
- MongoDB;
- servizi di rete dedicati.

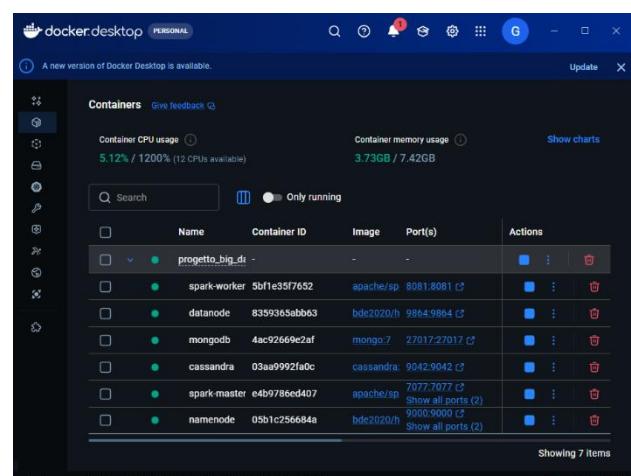
```

Run Service
25      ▶ Run Service
26      datanode:
27          image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1
              container_name: datanode

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
PS C:\Progetto_Big_Data>

PS C:\Progetto_Big_Data> ...
● PS C:\Progetto_Big_Data> docker-compose up -d
[+] up 6/6
  ✓ Container spark-master Running
  ✓ Container mongodb Running
  ✓ Container cassandra Running
  ✓ Container namenode Running
  ✓ Container spark-worker Running
  ✓ Container datanode Running
○ PS C:\Progetto_Big_Data>
  
```

Screenshot 1 - Avvio Servizi Docker



Screenshot 2 – Interfaccia Docker Desktop

Obiettivo della verifica:

- corretto deployment infrastrutturale
- disponibilità dei container

### 6.3 Inizializzazione dell'Infrastruttura Dati

Dopo l'avvio dell'ambiente containerizzato, è stata eseguita la fase di inizializzazione dell'infrastruttura dati necessaria al corretto funzionamento della pipeline.

Questa fase ha lo scopo di predisporre automaticamente:

- il Data Lake HDFS;
- il database Cassandra;
- il database MongoDB.

L'inizializzazione viene effettuata tramite script dedicati inclusi nella code-base del progetto.

#### 6.3.1 Inizializzazione HDFS

Lo script di inizializzazione HDFS crea la struttura logica del Data Lake utilizzata dalla pipeline.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



Struttura generata:

- /data/raw/prices
- /data/raw/news
- /data/processed/prices
- /data/features

Comando eseguito: **init\_hdfs.ps1**

```

automation > > init_hdfs.ps1
 1 Write-Host "Inizio creazione della struttura gerarchica del Data Lake su HDFS..." -ForegroundColor Cyan
 2
 3 # --- DEFINIZIONE STRUTTURA ---
 4 # BRONZE LAYER: Dati grezzi (Prezzi e News)
 5 # SILVER LAYER: Dati trasformati (Parquet)
 6 # GOLD LAYER: Risultati finali (Features)
 7
 8 docker exec namenode /opt/hadoop-3.2.1/bin/hdfs dfs -mkdir -p ` 
 9   /data/raw/prices ` 
10   /data/raw/news ` 
11   /data/processed/prices ` 
12   /data/features
13
14 Write-Host "`nStruttura HDFS completata. Il Data Lake è ora pronto per l'ingestion." -ForegroundColor Green

```

Screenshot 3: Inizializzazione data lake HDFS

Verifica effettuata tramite interfaccia:

- <http://localhost:9870>

Name	Size	Last Modified	Block Size
features	0 B	Feb 17 17:14	0 B
processed	0 B	Feb 17 17:14	0 B
raw	0 B	Feb 17 17:14	0 B

Screenshot 4 - Struttura HDFS tramite WEB Interfase

L'esecuzione dello script garantisce la disponibilità preventiva dei percorsi richiesti dai processi di ingestion e processing.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 6.3.2 Inizializzazione Cassandra

Il database Cassandra viene configurato tramite uno script di bootstrap che definisce automaticamente lo schema dati necessario alla gestione delle serie temporali finanziarie.

Comando: ***init\_cassandra.ps1***

Le operazioni effettuate includono:

- creazione del keyspace applicativo;
- definizione delle tabelle per dati intraday;
- configurazione della chiave primaria ottimizzata per interrogazioni temporali.

Lo script CQL viene eseguito automaticamente tramite script PowerShell di inizializzazione.

```
● PS C:\Progetto_Big_Data> .\scripts\init_cassandra.ps1
>>
    Initializing Cassandra schema...
    Cassandra schema ready.
○ PS C:\Progetto_Big_Data> █
```

Screenshot 5 - Inizializzazione Cassandra

La verifica è stata effettuata da terminale connettendomi direttamente a Cassandra e visualizzando la tabella creata, come mostrato nella figura seguente.

```
○ PS C:\Progetto_Big_Data> docker exec -it cassandra cqlsh
>>
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.10 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> USE market;
cqlsh:market> DESCRIBE TABLE prices_by_ticker;

CREATE TABLE market.prices_by_ticker (
    ticker text,
    ts timestamp,
    close double,
    high double,
    low double,
    open double,
    volume bigint,
    PRIMARY KEY (ticker, ts)
) WITH CLUSTERING ORDER BY (ts DESC)
    AND additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_compaction_threshold': '16'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compr
    AND memtable = 'default'
    AND read_repair_chance = 0.01
    AND speculative_retry = '99p'@'local'
```

Screenshot 6 - Verifica Cassandra

### 6.3.3 Inizializzazione MongoDB

La configurazione MongoDB prepara il database documentale destinato alla gestione delle news finanziarie.

Lo script di inizializzazione crea:

- database applicativo market;
- collection news;
- indice unico composto per prevenire duplicazioni;
- indice temporale per ottimizzare le query analitiche.

L'indice unico garantisce l'idempotenza della pipeline evitando l'inserimento multiplo della stessa notizia durante successive esecuzioni Spark

Comando: ***init\_mongo.ps1***

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



```
PS C:\Progetto_Big_Data> .\scripts\init_mongo.ps1
Initializing MongoDB (market.news)...
test>
test> market
market>
market>
market> { ok: 1 }
market>
market>
market>
market>
market> ... ... ... ticker_1_title_1
market>
market>
market>
market> ts_-1
market>
market> --- Configurazione MongoDB completata: ho preparato il database 'market' ---
```

Screenshot 7 - Inizializzazione MongoDB

La verifica è stata effettuata da terminale connettendomi direttamente a Mongodbd e visualizzando la tabella creata, come mostrato nella figura seguente.

```
PS C:\Progetto_Big_Data> docker exec -it mongodb mongosh
Current Mongosh Log ID: 699737647bda30a0a28ce5af
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.6.0
Using MongoDB: 7.0.30
Using Mongosh: 2.6.0
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
-----
The server generated these startup warnings when booting
2026-02-19T14:38:01.174+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2026-02-19T14:38:01.868+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
test> show dbs
admin 40.00 KiB
config 84.00 KiB
local 72.00 KiB
market 16.00 KiB
test> use market
switched to db market
market> show collections
news
```

Screenshot 8 – Verifica MongoDB

Pertanto al termine della fase di bootstrap risultano verificate:

- accessibilità dei database;
- presenza degli schemi definiti;
- corretta inizializzazione delle strutture dati.

Questa fase garantisce che l'ambiente sia pronto per l'esecuzione della pipeline di ingestion.

#### 6.4 Fase della Data Ingestion

La fase di Data Ingestion è stata validata verificando il corretto trasferimento dei dati finanziari dalle sorgenti esterne al Data Lake HDFS.

L'acquisizione dei dati è realizzata tramite due pipeline indipendenti sviluppate in Python:

- ingestion dei dati di mercato intraday;
- ingestion delle news finanziarie.

Entrambi gli script comunicano direttamente con il cluster HDFS mediante protocollo WebHDFS, consentendo la scrittura dei dati nel Data Lake senza utilizzo di file temporanei locali.

##### 6.4.1 Ingestion dei dati intraday

Lo script di acquisizione prezzi effettua automaticamente:

- download dei dati di mercato da Yahoo Finance con granularità temporale di 2 minuti;
- normalizzazione dei campi e dei nomi delle colonne;
- aggregazione dei dati provenienti da più ticker;
- trasferimento diretto del dataset nel Bronze Layer HDFS.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



Comando: ***python .\scripts\ingestion\download\_yahoo\_news.py***

Durante la validazione sono stati verificati i seguenti aspetti:

- corretta connessione al NameNode HDFS;
- acquisizione dei dati per tutti i ticker configurati;
- generazione del dataset unificato;
- scrittura del file CSV nel percorso previsto.

L'output del terminale mostra il download dei dati per ciascun ticker e la conferma della scrittura su HDFS.

```
PS C:\Progetto_Big_Data> python .\scripts\ingestion\download_yahoo_prices_2min.py
--- START INGESTION (DOCKER SETUP) ---
Scaricamento AAPL... OK (6425 righe)
Scaricamento NVDA... OK (6421 righe)
Scaricamento TSLA... OK (6425 righe)
Scaricamento ENI.MI... OK (8825 righe)
Scaricamento ISP.MI... OK (8836 righe)

Tentativo di scrittura su HDFS (Docker)... SUCCESSO!
File salvato in: /data/raw/prices/prices_intraday_2m.csv
```

Screenshot 9 - Esecuzione ingestion price

#### 6.4.2 Ingestion delle news finanziarie

La seconda pipeline acquisisce feed informativi associati ai titoli monitorati e genera documenti JSON destinati al Data Lake.

Durante questa fase è stata implementata una procedura di Safe Ingestion per garantire robustezza del sistema in presenza di dati incompleti provenienti dalla sorgente esterna.

In particolare:

- i timestamp mancanti vengono sostituiti con l'istante di acquisizione;
- eventuali errori su singoli ticker non interrompono la pipeline;
- ogni documento viene arricchito con informazioni di tracciabilità dell'ingestion.

Le verifiche effettuate hanno confermato:

- recupero corretto delle news per i ticker configurati;
- generazione del dataset JSON;
- scrittura persistente su HDFS.

```
PS C:\Progetto_Big_Data> python .\scripts\ingestion\download_yahoo_news.py
Recupero news per MSFT... OK (10 news)
Recupero news per GOOGL... OK (10 news)
Recupero news per AMZN... OK (10 news)
Recupero news per NVDA... OK (10 news)
Recupero news per META... OK (10 news)
Recupero news per TSLA... OK (10 news)
Recupero news per BRK-B... OK (10 news)
Recupero news per V... OK (10 news)
Recupero news per JNJ... OK (10 news)
Recupero news per ENI.MI... OK (10 news)
Recupero news per ISP.MI... OK (10 news)
Recupero news per UCG.MI... OK (10 news)
Recupero news per ENEL.MI... OK (10 news)
Recupero news per STLAM.MI... OK (10 news)
Recupero news per RACE.MI... OK (10 news)
Recupero news per G.MI... OK (10 news)
Recupero news per AZA.MI... OK (4 news)
Recupero news per AMD... OK (10 news)
Recupero news per INTC... OK (10 news)
Recupero news per NFLX... OK (10 news)
Recupero news per PYPL... OK (10 news)
Recupero news per BABA... OK (10 news)
Recupero news per DIS... OK (10 news)
Recupero news per BA... OK (10 news)
Recupero news per CSCO... OK (10 news)
Recupero news per PEP... OK (10 news)
Recupero news per KO... OK (10 news)

Scrittura file JSON su HDFS (/data/raw/news/market_news.json)... SUCCESSO!
```

Screenshot 10 - Esecuzione ingestion news

#### Verifica Persistenza su HDFS

A conclusione della fase di ingestion è stata verificata la presenza dei dati nel Data Lake tramite interfaccia web HDFS.

Percorsi validati:

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



/data/raw/prices/

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

### Browse Directory

/data/raw/prices								
Show 25 entries		Search:						
□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	root	supergroup	3.66 MB	Feb 17 22:46	1	128 MB	prices_intraday_2m.csv

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2019.

Screenshot 11 – Interfaccia HDFS

/data/raw/news/

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

### Browse Directory

/data/raw/news								
Show 25 entries		Search:						
□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	root	supergroup	64.83 KB	Feb 17 23:15	1	128 MB	market_news.json

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2019.

Screenshot 12 – Interfaccia HDFS

La verifica conferma:

- corretto popolamento del Bronze Layer;
- integrità dei file generati;
- funzionamento end-to-end della fase di acquisizione.

## 6.5 Automazione della pipeline Spark

In questo step è stato automatizzato il popolamento dei database operativi attraverso l'esecuzione coordinata di tre processi Apache Spark gestiti direttamente tramite uno script di orchestrazione dedicato. Lo script consente l'esecuzione sequenziale della pipeline di processamento e caricamento dati senza interventi manuali, garantendo la corretta transizione dei dataset tra i diversi layer della piattaforma.

In particolare, l'automazione prevede:

- trasformazione dei dati raw presenti su HDFS in formato Parquet (Silver Layer);
- caricamento dei dati intraday nel database Cassandra;
- caricamento e normalizzazione delle news finanziarie nel database MongoDB.

Una volta avviato lo script di esecuzione della pipeline, è stato verificato il corretto completamento dei job Spark mediante analisi dei log di esecuzione e assenza di errori bloccanti.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 6.5.1 Validazione della conversione Spark (Bronze → Silver Layer)

La prima fase riguarda la trasformazione dei dati grezzi presenti nel Bronze Layer HDFS in dataset ottimizzati in formato Parquet.

Il job Spark esegue:

- lettura dei file CSV intraday dal Data Lake;
- normalizzazione dei campi temporali;
- partizionamento fisico dei dati per ticker;
- scrittura nel layer /data/processed.

Dopo l'esecuzione dello script è stata verificata la corretta generazione delle directory Parquet tramite interfaccia HDFS.

```
PS C:\Progetto_Big_Data> docker exec -it namenode hdfs dfs -ls /data/processed/prices/prices_parquet
Found 6 items
-rw-r--r--  3 spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/_SUCCESS
drwxr-xr-x - spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/ticker=AAP
L
drwxr-xr-x - spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/ticker=ENI
.MI
drwxr-xr-x - spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/ticker=ISP
.MI
drwxr-xr-x - spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/ticker=NVD
A
drwxr-xr-x - spark supergroup          0 2026-02-20 09:51 /data/processed/prices/prices_parquet/ticker=TSL
A
```

Screenshot 13 – Dataset Parquet generati su HDFS

La presenza delle partizioni conferma il corretto completamento della fase di processamento distribuito.

### 6.5.2 Validazione del popolamento Cassandra

La seconda fase della pipeline prevede il caricamento dei dati intraday nel database Cassandra, destinato alla gestione delle serie temporali finanziarie.

Il job Spark legge i dataset Parquet precedentemente generati e trasferisce i record nella tabella operativa del keyspace applicativo.

Il buon esito dell'operazione è stato verificato mediante interrogazione diretta del database Cassandra.

```
cqlsh>
cqlsh> USE market;
cqlsh:market> DESCRIBE TABLES;

prices_by_ticker

cqlsh:market> SELECT *
... FROM prices_by_ticker
... LIMIT 10;

ticker | ts                                | close | high  | low   | open  |
-----+-----+-----+-----+-----+-----+
ISP.MI | 2026-02-17 16:28:00.000000+0000 | 5.721 | 5.726 | 5.721 | 5.721 |
ISP.MI | 2026-02-17 16:26:00.000000+0000 | 5.72 | 5.722 | 5.72 | 5.72
ISP.MI | 2026-02-17 16:24:00.000000+0000 | 5.72 | 5.724 | 5.72 | 5.723 |
ISP.MI | 2026-02-17 16:22:00.000000+0000 | 5.722 | 5.725 | 5.7155 | 5.722 |
ISP.MI | 2026-02-17 16:20:00.000000+0000 | 5.722 | 5.725 | 5.721 | 5.722 |
ISP.MI | 2026-02-17 16:18:00.000000+0000 | 5.724 | 5.727 | 5.718 | 5.718 |
ISP.MI | 2026-02-17 16:16:00.000000+0000 | 5.719 | 5.72 | 5.716 | 5.719 |
ISP.MI | 2026-02-17 16:14:00.000000+0000 | 5.72 | 5.722 | 5.717 | 5.722 |
ISP.MI | 2026-02-17 16:12:00.000000+0000 | 5.722 | 5.724 | 5.719 | 5.719 |
ISP.MI | 2026-02-17 16:10:00.000000+0000 | 5.719 | 5.721 | 5.714 | 5.714 |

(10 rows)
cqlsh:market>
```

Screenshot 13 – Query di verifica Cassandra

La presenza dei record OHLCV ordinati temporalmente conferma il corretto popolamento del database time-series.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



### 6.5.3 Validazione del popolamento MongoDB

L'ultima fase riguarda il caricamento delle news finanziarie nel database documentale MongoDB.

Durante questa fase Spark esegue:

- lettura dei dataset JSON dal Bronze Layer;
- selezione dei campi rilevanti;
- filtraggio dei record incompleti;
- inserimento dei documenti nella collection news.

La validazione è stata effettuata interrogando direttamente il database MongoDB.

```
PS C:\Progetto_Big_Data> docker exec -it mongo mongo
Current Mongosh Log ID: 699885a1e103f582d8ce5af
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appname=mongosh+2.6.0
Using MongoDB:     7.0.30

test> use market
switched to db market
market> db.news.countDocuments()
274
market>   { ticker:1, title:1, publish_time:1 }
...
...
market> // Mostra una news reale per dimostrare che titolo e link non sono più NULL
... db.news.find({ title: { $ne: null } }).limit(1).pretty()
[
  {
    _id: ObjectId('699885a18ba24d44a899dc25'),
    ticker: 'AAPL',
    ts: ISODate('2026-02-20T15:07:54.000Z'),
    title: 'OpenAI developing AI devices including smart speaker, The Information reports',
    source: null
  }
]
```

Screenshot 14 – Query di verifica MongoDB

La presenza dei documenti conferma il corretto trasferimento dei dati semi-strutturati nel layer operativo.

### 6.6 Validazione del Modulo Analytics

La fase finale del progetto implementa un motore di Decision Support System (DSS). Il modulo indicator\_spark.py esegue una join complessa tra i due layer Gold (NoSQL):

- Dataset News (da MongoDB): Fornisce il trigger informativo.
- Dataset Market (da Cassandra): Fornisce la validazione quantitativa tramite il calcolo della Media Mobile Semplice (SMA) a 20 periodi utilizzando le *Window Functions* di Spark.

Il risultato finale (come mostrato in tabella) permette di filtrare i segnali speculativi delle notizie, approvando l'operatività solo quando il trend tecnico del titolo è coerente con il sentiment, realizzando così una pipeline di Algorithmic Trading Validation."

```
PS C:\Progetto_Big_Data> .\scripts\indicator.ps1
iver, e4b9786ed407, 34993, None)
26/02/20 16:15:55 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, e4b9786ed407, 34993, None)
26/02/20 16:15:55 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, e4b9786ed407, 34993, None)
26/02/20 16:15:55 INFO StandaloneAppClient$ClientEndpoint: Executor updated: app-20260220161555-0007/0 is now RUNNING
26/02/20 16:15:55 INFO StandaloneSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.0

=====
SISTEMA DI VALIDAZIONE NEWS-DRIVEN (MONGODB -> CASSANDRA)
=====

+-----+-----+-----+-----+-----+
|ticker| date | title | close | sma_20 | entry_advice |
+-----+-----+-----+-----+-----+
|ENI.MI|2026-02-17|Eni Makes Major Gas Discovery Offshore Côte d'I...|18.145999908447266|18.17923809233166|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni Makes Major Gas Discovery Offshore Côte d'I...|18.148000717163086|18.170285815284366|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni's African Offshore Discoveries And What The...|18.145999908447266|18.17923809233166|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|BP and Eni Expand Angola Footprint With Algaita...|18.158000946044922|18.175523848766697|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni Makes Major Gas Discovery Offshore Côte d'I...|18.148000717163086|18.173809596470424|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni's African Offshore Discoveries And What The...|18.148000717163086|18.173809596470424|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni makes Calao South discovery offshore Côte d...|18.148000717163086|18.173809596470424|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|BP and Eni Expand Angola Footprint With Algaita...|18.148000717163086|18.173809596470424|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni Makes Major Gas Discovery Offshore Côte d'I...|18.13599967956543|18.171714328584216|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni's African Offshore Discoveries And What The...|18.13599967956543|18.171714328584216|REJECT - Bearish Trend|
|ENI.MI|2026-02-17|Eni makes Calao South discovery offshore Côte d...|18.13599967956543|18.171714328584216|REJECT - Bearish Trend|
+-----+-----+-----+-----+-----+
only showing top 20 rows
```

Screenshot 15 – Tabella di analisi

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## Analisi del Risultato (Business Intelligence)

Guardando i dati di **ENI.MI**:

- **Incrocio Dati:** Spark ha unito correttamente la news "Eni Makes Major Gas Discovery..." con i dati di borsa del 17 Febbraio 2026.
- **Indicatore Tecnico:** È stata calcolata una Media Mobile (**sma\_20**) di circa **18.17**.
- **Logica Decisionale:** Poiché il prezzo di chiusura (**18.14**) era inferiore alla media mobile, il sistema ha generato un segnale di **REJECT - Bearish Trend**.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## 7 Conclusioni Tecniche e Scalabilità del Framework

L'infrastruttura sviluppata non costituisce unicamente un sistema di data processing statico, ma rappresenta il **core funzionale** per un framework evoluto di **Algorithmic Trading**. L'architettura è stata progettata per supportare l'integrazione di moduli analitici complessi attraverso le seguenti direttive tecniche:

### 1. Engine di Backtesting Quantitativo

- **Storage Optimization:** L'utilizzo di **Apache Cassandra** come Wide-Column Store permette l'archiviazione di serie storiche ad alta densità, fondamentale per l'esecuzione di test retrospettivi su strategie multi-asset.
- **Feature Engineering:** Il Silver Layer basato su **Parquet** consente l'estrazione efficiente di feature tecniche (es. medie mobili, volatilità) necessarie per la validazione statistica delle strategie su ampi orizzonti temporali.

### 2. Transizione verso il Real-Time Processing

- **Stream Processing:** La logica implementata in `indicator_spark.py` è nativamente compatibile con le API di **Spark Structured Streaming**.
- **Latenza di Ingestion:** Sostituendo il driver di ingestion batch con un broker di messaggistica (es. Apache Kafka), il sistema può evolvere in una **Lambda Architecture**, riducendo il tempo intercorso tra l'evento informativo e la generazione del segnale decisionale.

### 3. Validazione Cross-Database (Polyglot Persistence)

- **Data Fusion:** L'architettura valida la coerenza tra flussi non strutturati (News su **MongoDB**) e flussi strutturati (Prezzi su **Cassandra**), fornendo una base solida per l'addestramento di modelli di Machine Learning finalizzati alla previsione dei trend di mercato.

**Sintesi Finale:** Il framework garantisce la separazione dei livelli di calcolo e persistenza, permettendo l'estensione verso sistemi di trading automatico dove l'analisi tecnica e fondamentale convergono in un unico output decisionale validato.

<b>Tipo di documento:</b>	Report
<b>Titolo:</b>	<i>Financial Big Data Pipeline</i>
<b>Insegnamento:</b>	Algoritmi e Strutture dati per i Big Data
<b>Docente:</b>	Giovanni Farina



## Appendice

Il processo di sviluppo ha seguito un approccio iterativo di tipo **Agile Debugging**. Le criticità emerse, tipiche della gestione di flussi Big Data (Data Drift, Schema Mismatch, Environment Configuration), sono state risolte attraverso l'integrazione di strumenti di monitoraggio distribuito (Web UI di Hadoop e Spark) e test interattivi tramite PySpark Shell. Questo ha permesso di validare la pipeline non solo dal punto di vista funzionale, ma anche sotto il profilo della **Data Resilience**.

Fase Pipeline	Problema Riscontrato	Causa Identificata	Risoluzione Applicata
<b>Ingestion (Yahoo)</b>	Errore <code>NoneType object has no attribute 'get'</code>	Cambio formato API: i dati erano annidati nel nuovo campo <code>content</code> .	Implementazione di <b>Safe Navigation</b> con controlli <code>isinstance</code> e valori di fallback.
<b>Raw Storage (HDFS)</b>	Errore <code>Spark PATH_NOT_FOUND</code>	Discrepanza tra il path cercato dallo script e quello reale su HDFS.	Ispezione via <b>Hadoop Web UI</b> (porta 9870) e correzione del puntamento del file.
<b>Gold Layer (Mongo)</b>	Conteggio record in MongoDB pari a 0	Il parser JSON di Spark ignorava i dati a causa della struttura array del file.	Abilitazione dell'opzione <code>multiLine=true</code> nel DataFrameReader di Spark.
<b>Gold Layer (Mongo)</b>	Campi <code>title</code> e <code>link</code> risultanti <code>NULL</code>	Schema inference fallito o filtri <code>IsNotNull</code> troppo restrittivi.	Aggiornamento del parser per navigare la gerarchia <code>content.title</code> nel JSON sorgente.
<b>Analytic Engine</b>	Errore <code>JAVA_HOME</code> not set su Windows	Tentativo di esecuzione dello script Spark direttamente sull'host Windows.	Esecuzione containerizzata tramite <code>docker exec</code> e <code>spark-submit</code> nel cluster master.
<b>Analytic Engine</b>	Errore <code>ModuleNotFoundError: No module named 'pyspark'</code>	Python di sistema nel container senza variabili d'ambiente Spark caricate.	Utilizzo del path assoluto per l'eseguibile: <code>/opt/spark/bin/pyspark</code> .