Immagine che contiene emblema, cresta, Marchio, simbolo

Il contenuto generato dall'IA potrebbe non essere corretto.

**UNIVERSITÀ DEGLI STUDI DI BARI   
“ALDO MORO”**

Dipartimento di Informatica

Corso di Laurea in Informatica

*Tesi di Laurea in Algoritmi e Strutture Dati*

**Analisi automatizzata del mercato crypto attraverso tecniche di intelligenza artificiale.**

*Relatore*:  
Prof. Gianvito Pio

*Laureando*:  
Gioele Panico

Anno Accademico 2025/2026

**Abstract**

L’evoluzione delle tecnologie digitali e dell’intelligenza artificiale sta modificando profondamente i processi di analisi e previsione nel settore finanziario, con particolare attenzione al mercato delle criptovalute. Queste, a partire dal lancio di Bitcoin nel 2009, hanno visto una crescita esponenziale, caratterizzata da una forte volatilità e da una continua influenza del flusso informativo proveniente dal web e dai social media.

In questo contesto, la capacità di raccogliere, sintetizzare e classificare automaticamente le notizie rappresenta un elemento chiave per comprendere le dinamiche del mercato crypto e supportare decisioni di investimento più consapevoli.

Il lavoro di questa tesi propone un sistema automatizzato per l’analisi del mercato delle criptovalute basato su tecniche di intelligenza artificiale. In particolare, è stato sviluppato un flusso che prevede: la raccolta delle notizie tramite web scraping dalla piattaforma *CryptoPanic*; l’utilizzo di un modello linguistico di grandi dimensioni (LLaMA, versione 3.2) per la generazione di riassunti brevi e lunghi; la classificazione delle notizie tramite un approccio ibrido che combina un clustering non supervisionato per l’individuazione iniziale delle categorie e un classificatore supervisionato (Naïve Bayes) per l’assegnazione automatica delle etichette a nuove news; l’analisi del sentiment e la stima del peso informativo tramite un modello di regressione.

Infine, il sistema integra i risultati ottenuti all’interno di un report settimanale, arricchito dall’analisi del *Fear & Greed Index*, con l’obiettivo di fornire una panoramica sintetica ma completa sull’andamento del mercato.

L’approccio sviluppato dimostra come l’integrazione di tecniche di NLP, machine learning e metriche di sentiment analysis possa rappresentare un valido strumento di supporto per interpretare la complessità e la dinamicità del settore delle criptovalute, confermandone la rilevanza nell’ambito dell’analisi dei mercati finanziari digitali.

**Sommario**

1. **Introduzione1**

1.1 Contesto e motivazioni2

1.2 Obiettivi della tesi2

1.2 Struttura del documento 2

1. **Il mercato delle criptovalute1**

2.1 Cosa è una criptovaluta? 2

2.2 Evoluzione storica e tendenza attuali del mercato crypto2

2.3 Fonti informative del mondo crypto (Twitter, CryptoPanic, blog, ecc.)2

2.4 Limiti e problematiche dell’ analisi tradizionale del mercato2

1. **Tecnologie di riferimento1**

2.1 Intelliegnza Artificiale e Machine Learning: concetti di base 2

2.2 Natural Language Processing (NLP) e tecniche di analisi testuale2

2.3 Large Language Models (LLM): caratteristiche e modelli open-source2

2.4 Tecniche di classificazione supervisionata e non supervisonata 2

2.5 Modelli di regressione per sentiment e scoring 2

1. **Architettura del sistema sviluppato 1**

2.1 Panoramica della pipeline sviluppata 2

2.2 Workflow del sistema2

2.3 Scelte progettuali e criteri di implementazione 2

1. **Raccolta e gestione delle notizie (Data Collection)1**

2.1 Obiettivi della fase di scraping 2

2.2 Scelta delle fonti informative (CryptoPanic) 2

2.3 Struttura del database 2

2.2 Recupero nuovi articoli: titolo, data e url\_cryptopanic2

2.2 Recupero URL originale e contenuto dell’ articolo 2

2.2 Problemi comuni e soluzioni (duplicati, encodicng, contenuti mancanti) 2

1. **Sintesi automatica delle notizie 1**

2.1 Introduzione ai modelli LLM e scelta di LLaMA 3.2 2

2.2 Prompt engineering per la generazione dei riassunti 2

2.3 Generazione di riassunti brevi e lunghi 2

2.2 Integrazione della sintesi nel flusso di lavoro 2

1. **Classificazione delle notizie 1**

2.1 Classificazione non supervisionata (clustering) per identificare le categorie 2

2.2 Etichettatura manuale delle categorie emerse (cluster) 2

2.3 Addestramento del classificatore supervisionato (Naïve Bayes)2

2.2 Metriche di valutazione e risultati 2

2.3 Deployment della classificazione per nuove notizie 2

1. **Analisi quantitativa: sentiment e peso informativo 1**

2.1 Definizione dei concetti di “sentiment” e “peso”2

2.2 Linee guida per l’etichettatura manuale iniziale 2

2.3 Preprocessing dei testi per la regressione 2

2.2 Modelli di regressione testati (*Random Forest, XGBoost, regressore neurale*)2

2.3 Selezione del miglior modello e metriche di valutazione 2

2.2 Interpretazione dei risultati 2

1. **Generazione del report settimanale 1**

2.1 Obiettivi del report automatizzato 2

2.2 Struttura del PDF generato 2

2.3 Integrazione del Fear & Greed Index (media ponderata) 2

2.2 Visualizzazione del Fear & Greed Index (media ponderata) 2

2.3 Visualizzazione dinamica (grafici, gauge index) 2

2.2 Automazione del processo di reportistica 2

2.3 Esempio completo di report settimanale generato 2

1. **Implementazione e ambiente di sviluppo 1**

2.1 Linguaggi e strumenti utilizzzati (Python, librerie, GoogleColab) 2

2.2 Organizzazione del codice e struttura del repository GitHub2

2.3 Ambiente di sviluppo (IDE, versionamento, test)2

2.2 Risorse hardware e software utilizzate 2

1. **Discussione e sviluppi futuri 1**

2.1 Criticità riscontrate durante lo sviluppo 2

2.2 Limiti del sistema attuale 2

2.3 Possibili miglioramenti futuri2

* 1. Impatti nel mondo del trading e della finanza 2

**Bibliografia 1**

**Ringraziamenti 1**

1. **Introduzione**

**1.1 Contesto e motivazioni**

Negli ultimi anni il settore delle criptovalute ha registrato un’espansione senza precedenti, trasformandosi da fenomeno di nicchia a tema di rilevanza globale per mercati finanziari, media e opinione pubblica. La natura decentralizzata di questi strumenti, insieme alla loro elevata volatilità, li rende un campo di studio complesso ma estremamente interessante.

Accanto all’analisi dei grafici di prezzo e delle metriche economiche classiche, un ruolo sempre più determinante è ricoperto dalle informazioni diffuse attraverso articoli, social network e piattaforme dedicate. Notizie riguardanti regolamentazioni, attacchi informatici, partnership tecnologiche o semplici dichiarazioni di figure influenti (come Elon Musk o Vitalik Buterin) possono generare variazioni repentine e significative del mercato.

In questo contesto, l’utilizzo di tecniche di **intelligenza artificiale (IA)** per automatizzare la raccolta, l’elaborazione e l’interpretazione delle informazioni rappresenta un’opportunità strategica. L’IA consente di estrarre conoscenza da grandi quantità di dati testuali, riducendo il rumore informativo e fornendo indicatori sintetici capaci di supportare decisioni di investimento più consapevoli.

**1.2 Obiettivi della tesi**

Questa tesi si propone di sviluppare e analizzare un sistema automatizzato in grado di:

* **Raccogliere** notizie rilevanti dal mondo delle criptovalute tramite tecniche di web scraping, con focus sulla piattaforma *CryptoPanic*.
* **Sintetizzare** i contenuti delle notizie mediante l’uso di modelli linguistici di grandi dimensioni (LLM), generando riassunti sia brevi che estesi.
* **Classificare** automaticamente le news, sfruttando un approccio combinato: clustering non supervisionato per individuare categorie iniziali e classificazione supervisionata (Naïve Bayes) per etichettare nuove informazioni.
* **Valutare** il sentiment e il “peso” informativo di ciascuna notizia tramite l’addestramento di regressori.
* **Produrre** un report settimanale arricchito dal *Fear & Greed Index*, per offrire una panoramica sintetica e aggiornata sull’andamento del mercato crypto.

L’obiettivo finale è dimostrare come l’integrazione di tecniche di NLP, machine learning e metriche di sentiment analysis possa fornire strumenti pratici per interpretare la complessità del settore.

**1.3 Struttura del documento**

Il lavoro è articolato in più capitoli. Dopo l’introduzione e la presentazione del contesto (Cap. 1–2), il testo affronta le tecnologie utilizzate e l’architettura del sistema (Cap. 3–4). Nei capitoli successivi si analizzano la sintesi dei testi, la classificazione, la valutazione del sentiment e la costruzione del report (Cap. 5–8). Infine, si discutono criticità, prospettive di miglioramento e conclusioni (Cap. 9–Conclusioni).

1. **Il mercato delle criptovalute**

**2.1 Cosa è una criptovaluta?**

Una criptovaluta è una rappresentazione digitale di valore basata su tecniche crittografiche, che consente lo scambio sicuro di beni e servizi senza l’intermediazione di enti centrali come banche o governi. La prima e più nota criptovaluta è **Bitcoin**, introdotta nel 2009 da Satoshi Nakamoto.

Le criptovalute si basano generalmente su una **blockchain**, un registro distribuito che garantisce trasparenza e immutabilità delle transazioni. Oltre a Bitcoin, negli anni sono nate migliaia di criptovalute alternative (*altcoin*), come Ethereum, Ripple, Litecoin, Cardano e Solana, ognuna caratterizzata da specifiche funzioni (contratti intelligenti, privacy, velocità di transazione, interoperabilità).

**2.2 Evoluzione storica e tendenze attuali del mercato crypto**

Il mercato delle criptovalute è cresciuto rapidamente a partire dal 2013, con una capitalizzazione complessiva che ha raggiunto picchi superiori ai 2.000 miliardi di dollari nel 2021. La sua storia è costellata da fasi di crescita esplosiva seguite da crolli improvvisi (“bull market” e “bear market”), spesso legati a notizie di regolamentazione, scandali di exchange o innovazioni tecnologiche.

Oggi il settore si sta consolidando, con un interesse crescente da parte di investitori istituzionali, banche e governi. Parallelamente, continua a essere un mercato altamente volatile e speculativo, influenzato non solo da variabili economiche, ma anche da fattori emotivi e psicologici.

**2.3 Fonti informative del mondo crypto**

L’ecosistema crypto è alimentato da una mole enorme di informazioni prodotte quotidianamente. Le principali fonti comprendono:

* **Social media**: Twitter/X è uno degli strumenti più influenti, dove analisti, influencer e sviluppatori diffondono opinioni e breaking news.
* **Aggregatori di notizie**: piattaforme come *CryptoPanic* raccolgono in tempo reale articoli e post da centinaia di fonti, rendendole facilmente consultabili.
* **Blog e forum**: siti come *Medium* o forum come *Bitcointalk* offrono discussioni tecniche, spesso di nicchia, ma rilevanti.
* **Siti di news specializzati**: *CoinDesk, CoinTelegraph, Decrypt* forniscono articoli giornalistici e analisi di settore.

La quantità e l’eterogeneità delle informazioni rende necessaria l’adozione di strumenti di filtraggio e analisi automatizzata per distinguere contenuti rilevanti da rumore informativo.

**2.4 Limiti e problematiche dell’analisi tradizionale del mercato**

Le analisi tradizionali dei mercati finanziari si basano su modelli econometrici, serie storiche e indicatori tecnici. Tuttavia, nel settore crypto tali approcci presentano diversi limiti:

* **Volatilità estrema**: i prezzi possono variare drasticamente in pochi minuti, riducendo l’affidabilità delle previsioni basate su trend storici.
* **Influenza delle notizie**: eventi esterni e breaking news hanno un impatto molto più immediato rispetto ai mercati tradizionali.
* **Rumore informativo**: l’enorme quantità di contenuti non sempre strutturati (tweet, post, articoli) rende difficile individuare ciò che è realmente rilevante.
* **Assenza di regolamentazione uniforme**: differenze normative tra paesi generano incertezza e reazioni di mercato non prevedibili.

Queste criticità giustificano l’esigenza di approcci innovativi, basati su **intelligenza artificiale e analisi automatizzata delle notizie**, per migliorare la capacità predittiva e interpretativa del mercato delle criptovalute.

**3. Tecnologie di riferimento**

L'analisi automatizzata del mercato delle criptovalute rappresenta una sfida multidisciplinare, che coinvolge strumenti e metodologie derivanti dall’intelligenza artificiale, dal machine learning e dal natural language processing. Questo capitolo fornisce una panoramica delle principali tecnologie utilizzate nella presente tesi, con l’obiettivo di fornire un quadro teorico di riferimento necessario per comprendere le scelte progettuali e implementative nei capitoli successivi.

**3.1 Intelligenza Artificiale e Machine Learning: concetti di base**

L'**Intelligenza Artificiale (IA)** è un campo dell'informatica che mira a simulare l'intelligenza umana attraverso sistemi in grado di apprendere, ragionare e prendere decisioni. In particolare, il **Machine Learning (ML)** rappresenta una sottobranca dell’IA che consente ai sistemi di “imparare” da dati osservati, senza essere esplicitamente programmati.

Nel contesto di questa tesi, il machine learning è stato impiegato in diversi momenti della pipeline:

* per classificare le notizie in categorie,
* per stimare automaticamente il tono (sentiment) di un contenuto testuale,
* per assegnare un valore di importanza (peso) a ciascuna notizia.

I modelli supervisionati e non supervisionati utilizzati sono discussi più nel dettaglio nelle sezioni seguenti.

**3.2 Natural Language Processing (NLP) e tecniche di analisi testuale**

Il **Natural Language Processing (NLP)** è l’insieme di tecniche computazionali che permettono a una macchina di comprendere, analizzare, modificare o generare linguaggio naturale. Le attività di NLP si collocano a cavallo tra informatica, linguistica computazionale e statistica, e si articolano in diversi compiti, tra cui:

* **Tokenizzazione**, ovvero la suddivisione del testo in parole o frasi;
* **Rimozione delle stopword**, cioè parole comuni ma poco informative (es. “il”, “un”, “e”);
* **Lemmatizzazione**, che riconduce ogni parola alla sua forma base;
* **TF-IDF (Term Frequency-Inverse Document Frequency)**, una tecnica statistica per assegnare un peso alle parole in base alla loro rilevanza nei documenti.

Queste tecniche classiche sono state affiancate, in questa tesi, da approcci più avanzati basati su **trasformatori linguistici** (transformer), che hanno permesso una comprensione semantica più profonda dei testi.

**3.3 Large Language Models (LLM): caratteristiche e modelli open-source**

I **Large Language Models (LLM)** rappresentano la frontiera più avanzata dell'NLP. Questi modelli, basati su architetture **transformer**, sono addestrati su enormi quantità di dati testuali e sono capaci di comprendere e generare linguaggio naturale in modo coerente e contestuale.

Nel progetto è stato utilizzato **LLaMA 3.2**, un modello open-source di ultima generazione sviluppato da Meta. Questo LLM è stato impiegato per:

* **Generare automaticamente riassunti brevi e lunghi** delle notizie raccolte, migliorando la leggibilità e la fruibilità dell’informazione;
* Sperimentare approcci alla **regressione neurale** (tramite embeddings) per predire sentiment e peso.

L’utilizzo di LLM ha permesso di automatizzare processi che tradizionalmente richiederebbero l’intervento umano, mantenendo al contempo un’elevata qualità linguistica e coerenza semantica.

* 1. **Tecniche di classificazione supervisionata e non supervisionata**

Per categorizzare automaticamente le notizie crypto in macro-tematiche (es. “Sicurezza e hackeraggi”, “Adozione”, “Innovazione”, ecc.), sono state adottate due strategie complementari:

* **Classificazione non supervisionata** tramite **clustering** (es. K-Means), utilizzata per identificare gruppi semantici all’interno del corpus senza etichette predefinite. Questa fase ha permesso di derivare un primo insieme di categorie latenti.
* **Classificazione supervisionata**, in cui le notizie sono state manualmente etichettate sulla base delle categorie emerse, e successivamente è stato addestrato un modello di classificazione (Naïve Bayes) per automatizzare il processo.

Il passaggio da una fase non supervisionata a una supervisionata ha permesso di **combinare scoperta automatica e controllo umano**, garantendo risultati affidabili e adattabili a nuove notizie.

* 1. **Modelli di regressione per sentiment e scoring**

Per quantificare il **tono** e la **rilevanza** di ciascuna notizia, sono stati addestrati dei **modelli di regressione**, in grado di associare un valore numerico tra 0 e 1 a ogni articolo, secondo due assi principali:

* **Sentiment**, inteso come positività o negatività della notizia;
* **Peso**, ovvero l'importanza percepita della notizia rispetto al mercato crypto.

Sono stati testati diversi algoritmi di regressione:

* **Random Forest Regressor** e **XGBoost**, come modelli ad albero;
* **Ridge Regression**, sia con TF-IDF che con embeddings generati da LLM;
* **Regressori neurali**, sfruttando il modello **AIBERTo (BERT per l'italiano)** per estrarre rappresentazioni dense dei testi.

La metrica di valutazione principale utilizzata è il **coefficient of determination R²**, affiancata da **MAE**, **RMSE** e **MAPE**, per una comprensione più articolata delle performance dei modelli.

1. **Architettura del sistema sviluppato**

**4.1 Panoramica della pipeline sviluppata**

Il sistema sviluppato ha come obiettivo la raccolta, l’elaborazione e la sintesi di notizie relative al mercato delle criptovalute, al fine di generare report settimanali utili all’analisi delle dinamiche di settore.  
L’architettura è stata progettata come una **pipeline modulare**, in cui ogni fase si occupa di uno specifico compito e fornisce l’input alla fase successiva.

Le principali componenti della pipeline sono:

* **Data Collection** – Raccolta delle notizie tramite web scraping da *CryptoPanic* e salvataggio in un database SQLite.
* **Preprocessing** – Pulizia dei testi, gestione dei duplicati, normalizzazione dei caratteri ed encoding.
* **Sintesi con LLM** – Generazione di riassunti brevi ed estesi mediante il modello *LLaMA 3.2*.
* **Classificazione** – Identificazione delle categorie tramite clustering non supervisionato, seguita da classificazione supervisionata con Naïve Bayes.
* **Sentiment & Weight Analysis** – Valutazione della polarità e assegnazione di un peso informativo attraverso un regressore.
* **Report Generation** – Produzione del report settimanale in PDF, integrato con indicatori come il *Fear & Greed Index*.

Questa suddivisione consente al sistema di essere **scalabile e flessibile**: ogni modulo può essere aggiornato o sostituito senza compromettere l’intera architettura.

*(Qui è molto utile inserire un* ***diagramma a blocchi*** *che mostri i sei moduli collegati da frecce).*

**4.2 Workflow del sistema**

Il funzionamento del sistema può essere descritto attraverso il seguente workflow:

* **Input (News)**: il sistema scarica periodicamente notizie da *CryptoPanic* in formato JSON, contenenti titolo, descrizione, fonte e link.
* **Database Management**: le notizie vengono salvate in un database SQLite (*crypto\_news.db*), che rappresenta il repository centrale delle informazioni.
* **Text Cleaning & Normalization**: vengono rimossi caratteri speciali, link ridondanti e stopword, garantendo che i testi siano coerenti per l’elaborazione successiva.
* **Summarization**: i testi vengono processati dal modello *LLaMA 3.2*, che genera due tipi di output:
  + **Short summary** → sintetico (1–2 frasi).
  + **Long summary** → descrittivo (1–2 paragrafi).
* **Clustering e Categorizzazione**: inizialmente le notizie vengono suddivise in cluster tramite algoritmi non supervisionati (es. K-Means). Successivamente, le categorie vengono etichettate manualmente e utilizzate per addestrare un classificatore supervisionato (Naïve Bayes).
* **Sentiment & Weighting**: ogni notizia viene analizzata da un regressore per stimare sentiment (positivo/negativo/neutro) e assegnare un punteggio di rilevanza.
* **Output (Report PDF)**: settimanalmente il sistema genera un file PDF contenente:
  + elenco delle principali notizie classificate e sintetizzate,
  + grafici sull’andamento del sentiment,
  + un indicatore di *Fear & Greed Index* personalizzato.

Questo workflow permette di passare da una grande mole di dati grezzi a un documento finale strutturato e leggibile, riducendo i tempi di analisi per l’utente finale.

* 1. **Scelte progettuali e criteri di implementazione**

La progettazione del sistema è stata guidata da tre principi fondamentali:

* **Modularità**: ogni componente della pipeline è indipendente. Questo consente di sostituire, ad esempio, il modello di sintesi o il classificatore supervisionato senza modificare l’intero sistema.
* **Scalabilità**: l’utilizzo di un database leggero (SQLite) e di librerie standard Python (BeautifulSoup, scikit-learn, HuggingFace Transformers) garantisce compatibilità e possibilità di espansione.
* **Automazione**: l’intero processo è stato pensato per funzionare in maniera automatica, con script schedulabili che raccolgono e analizzano le notizie su base regolare, generando output settimanali senza intervento manuale.

Altri criteri rilevanti:

* **Efficienza computazionale**: per l’addestramento dei modelli sono state privilegiate soluzioni con un buon compromesso tra accuratezza e velocità (es. Naïve Bayes per la classificazione).
* **Riproducibilità**: il codice è stato organizzato in un repository GitHub con documentazione e istruzioni di esecuzione, in modo da permettere ad altri ricercatori o sviluppatori di replicare facilmente l’esperimento.
* **Estendibilità**: il sistema è pronto per l’integrazione di nuove fonti informative e per l’adozione di modelli di nuova generazione (es. Mistral, GPT-4 Turbo).

1. **Raccolta e gestione delle notizie (Data Collection)**

**5.1 Obiettivi della fase di scraping**

La fase di **raccolta delle notizie** costituisce il punto di ingresso della pipeline. L’obiettivo è acquisire automaticamente articoli aggiornati relativi al mercato delle criptovalute e salvarli in un database locale, in modo da renderli disponibili alle fasi successive di sintesi, classificazione e analisi.

**5.2 Scelta della fonte informativa (CryptoPanic)**

La piattaforma [CryptoPanic](https://cryptopanic.com) è stata selezionata come fonte per la raccolta di notizie, perché è un aggregatore di articoli del settore crypto e li fornisce in un feed strutturato facile da interpretare e recueprare tramite web scraping.

I motivi principali della scelta sono:

* **Completezza**: CryptoPanic integra notizie da più fonti eterogenee;
* **Struttura stabile**: la pagina presenta un layout HTML regolare, con blocchi <div> ben identificabili, adatto al parsing automatico;
* **Filtraggio per lingua**: è possibile restringere il feed alle sole notizie in italiano, caratteristica cruciale per la pipeline sviluppata;
* **Metadati immediatamente disponibili**: titolo, data e link sono già presenti nella pagina, riducendo la necessità di ulteriori elaborazioni preliminari.

**5.3 Struttura del database**

La gestione dei dati è affidata a un database **SQLite** denominato crypto\_news.db.  
La scelta di SQLite è motivata da:

* **Leggerezza** e semplicità di distribuzione;
* **Compatibilità con Python** (libreria sqlite3 integrata);
* **Adeguatezza al volume dei dati** atteso (migliaia di articoli, non milioni).

Immagine che contiene testo, linea, numero, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Il database è composto da due tabelle:

CREATE TABLE IF NOT EXISTS meta\_articoli (

id INTEGER PRIMARY KEY AUTOINCREMENT,

     url\_cryptopanic VARCHAR(512) UNIQUE,

     url\_articolo VARCHAR(512),

     data DATETIME,

);

CREATE TABLE IF NOT EXISTS articoli (

id INTEGER PRIMARY KEY,

     titolo VARCHAR(512),

     articolo\_completo\_html TEXT,

     riassunto\_breve TEXT,

     riassunto\_lungo TEXT,

     categoria VARCHAR(250),

     peso FLOAT,

     sentiment FLOAT,

     FOREIGN KEY (id) REFERENCES articoli\_meta(id) ON DELETE CASCADE

    );

Descrizione dei campi delle tabelle:

|  |  |
| --- | --- |
| **meta\_articoli** | |
| *id* | Identificativo univoco autoincrementale per ogni articolo. |
| *url\_cryptopanic* | URL dell’ articolo su CryptoPanic, usato come chiave naturlae (UNIQUE) perché non posso esistere articoli diversi con lo stesso URL. |
| *url\_articolo* | URL della fonte originale. |
| *data* | Data di pubblicazione dell’ articolo |

|  |  |
| --- | --- |
| **articoli** | |
| *id* | Identificativo legato 1:1 al record in *meta\_articoli* |
| *titolo* | Titolo della notizia |
| *articolo\_completo* | Contenuto testuale pulito dell’ articolo |
| *riassunto\_breve* | Riassunto sintetico generato con LLM |
| *riassunto\_lungo* | Riassunto esteso generato con LLM |
| *categoria* | Categoria assegnata tramite classificatore |
| *peso* | Rilevanza dell’ articolo generata da regressore |
| *sentiment* | Sentiment dell’ articolo calcolato da regresore |

Questa architettura separa chiaramente i **metadati sorgente** *(meta\_articoli)* dai **contenuti elaborati** *(articoli),* favorendo modularità e tracciabilità.

* 1. **Recupero nuovi articoli: titolo, data e url\_cryptopanic**

Il primo passo operativo della pipeline di raccolta consiste nell’estrazione, per ciascun articolo del feed di *CryptoPanic*, dei tre metadati fondamentali:

* URL interno a CryptoPanic (*url\_cryptopanic*), utilizzato per identificare in maniera univoca l’articolo all’interno della piattaforma e punto di accesso al dettaglio dell’ aritcolo;
* Titolo, che rappresenta il contenuto testuale principale della notizia;
* Data e ora di pubblicazione, necessarie per ordinare cronologicamente gli articoli e gestire in modo coerente gli aggiornamenti successivi.

Questi campi costituiscono la base dei record della tabella meta\_articoli e sono indispensabili per le successive fasi di arricchimento (recupero URL originale e contenuto).

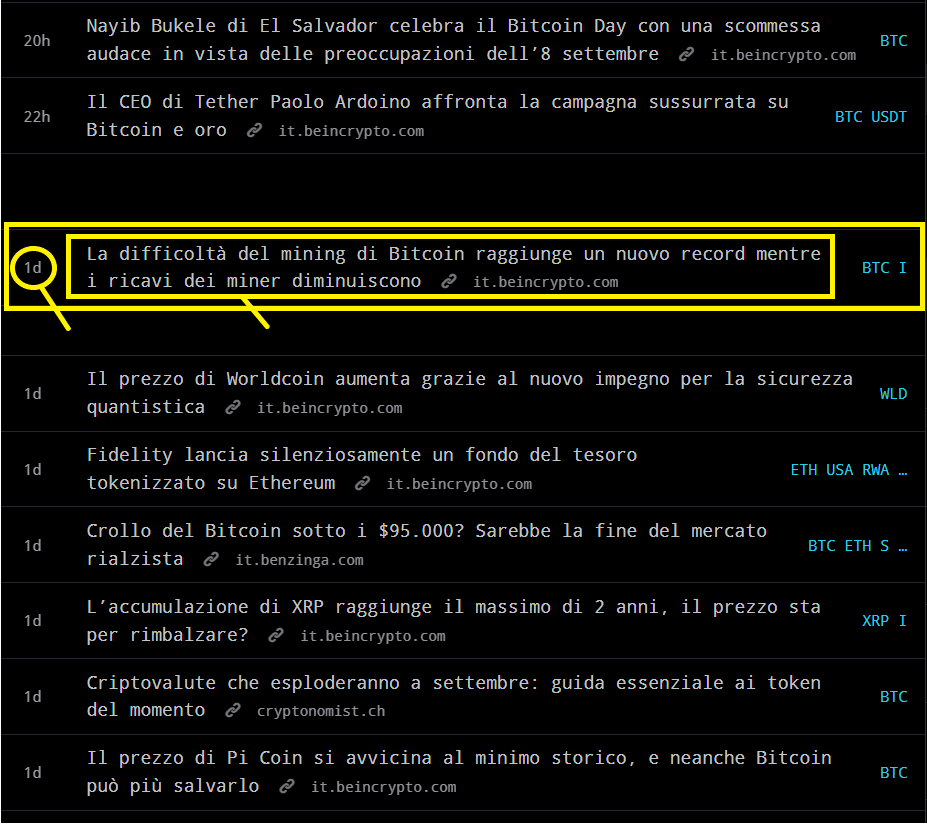
**Struttura della pagina di CryptoPanic**

La pagina delle notizie di *CryptoPanic* è organizzata in due sezioni principali:

* **Colonna sinistra**: elenco degli articoli ordinati cronologicamente. E’ una sezione scrollabile che permette di caricare progressivamente nuovi articoli.
* Immagine che contiene testo, schermata, software, Software multimediale

  Il contenuto generato dall'IA potrebbe non essere corretto.**Colonna destra**: dettaglio dell’articolo selezionato, che contiene titolo cliccabile (che rimanda alla fonte originale), anteprima del contenuto e metadati aggiuntivi.

Analizzando la struttura del sito abbiamo notato che la sezione di sinistra può essere considerata come un’ area iterabile (scroll verso il basso) contenente blocchi notizia, dove ogni articolo ha la seguente struttura.



*titolo*

*data*

Area cliccabile contenente *url\_cryptopanic*

Ovviamente la struttura visiva si riflette anche nel codice HTML,che verrà utilizzato per l’estrazione dei metadati, dove all’ interno di <div class = “news-container”>, che è l’intera area di sinistra, sono inclusi i blocchi notizia ognuno all’ interno del rispettivo blocco <div class = “news-row news-row-link”>

<div class="news-container">

    <div class="news-row news-row-link"> …

    </div>

    <div class="news-row news-row-link"> …

    </div>

    <div class="news-row news-row-link"> …

    </div>

</div

All’ interno poi di ogni blocco notizia, cioè ogni <div class="news-row news-row-link"> si trova il seguente codice che per ogni notizia contiene i metadati:

<div class="news-row news-row-link">

  <div class="news-cells">

    <a href="/news/25052339/La-difficolta-del-mining-di-Bitcoin

raggiunge-un-nuovo-record-mentre-i-ricavi-dei-miner-

diminuiscono"

       class="news-cell nc-date">

      <span>

        <time datetime="Sun Sep 07 2025 19:02:44 GMT+0200 (Ora

legale dell’Europa centrale)"

              title="Sun Sep 07 2025 19:02:44 GMT+0200 (Ora legale

dell’Europa centrale)">

        </time>

      </span>

    </a>

    <a href="/news/25052339/La-difficolta-del-mining-di-Bitcoin-

raggiunge-un-nuovo-record-mentre-i-ricavi-dei-miner-

diminuiscono"

       class="news-cell nc-title">

      <span class="title-text">

        <span>

          La difficoltà del mining di Bitcoin raggiunge un nuovo

record mentre i ricavi dei miner diminuiscono

        </span>

        <span class="si-source-name hidden-mobile">

          <span class="open-link-icon icon icon-link"></span>

          <span class="si-source-domain">it.beincrypto.com</span>

        </span>

      </span>

    </a>

  </div>

</div>

Per ognuna delle notizie però i tag più important che contengono i metadati che ci interessano sono i seguenti:

<div class="news-row news-row-link">

    <a href="="/news/25052339/La-difficolta-del-mining…"</a>

    <time datetime="Sun Sep 07 2025 19:02:44 GMT+0200">1d ago</time>

    <span class="title-text">

        <span> La difficoltà del mining di Bitcoin… </span>

    </span>

</div

Da questa struttura si ricavano i tre campi principali:

* **url\_cryptopanic**: contenuto nell’attributo href del tag <a>. Si tratta di un percorso relativo (es. /news/123456/) che viene concatenato a BASE\_URL = "https://cryptopanic.com" per ottenere l’URL assoluto (es. https://cryptopanic.com/news/123456/).
* **Data**: fornita dall’attributo datetime del tag <time>. La stringa viene normalizzata in formato SQL standard (YYYY-MM-DD HH:MM:SS) tramite la funzione utilities.convert\_to\_sql\_datetime().
* **Titolo**: contenuto nel secondo <span> annidato all’interno del blocco con classe title-text.

**Flusso di implementazione**

Quindi sfruttando la struttura HTML della pagina il codice pyhton implementato accede alla pagina web ed esegue i seguenti step:

1. **Scroll dinamico e caricamento completo del feed**

* Una funzione scorre il contenitore (<div class="news-container">)finché il numero di articoli (<div class="news-container">) non cresce più.

1. **Estrazione di titolo, data , url\_cryptopanic per ogni articolo.**

* Viene creato soup = BeautifulSoup(driver.page\_source, "html.parser"), che è un oggetto che contiene l’intero codice HTML della pagina ma con una struttura ad albero navigabile.
* Per ogni blocco notizia, la funzione estrai\_articoli\_da\_soup(soup) estrare le informazioni di interesse:

for div in soup.find\_all("div", class\_="news-row news-row-link"):

    a\_tag = div.find("a", href=True)        # url\_cryptopanic

    time\_tag = div.find("time", datetime=True)      # data

    title\_span = div.find("span", class\_="title-text")

    inner\_span = title\_span.find("span")

    title = inner\_span.get\_text(strip=True)         # titolo

    relative\_url = a\_tag["href"]

    full\_url = BASE\_URL + relative\_url

    published\_at\_raw = time\_tag["datetime"]

    published\_at = utilities.convert\_to\_sql\_datetime(published\_at\_raw)

articoli.append((full\_url, published\_at, title))

1. **Inserimento nel database (meta\_articoli)**

La lista degli articoli raccolti viene ordinata temporalmente.  
Infine, si inseriscono i record nella tabella meta\_articol e grazie al vincolo UNIQUE(url\_cryptopanic), eventuali duplicati vengono scartati.  
Per ogni nuovo record in meta\_articoli viene creata la corrispondente riga in articoli, con lo stesso id e il titolo.

**Risultato**

Al termine di questa fase, per ciascun articolo disponibile su *CryptoPanic* il database locale contiene:

* **url\_cryptopanic**: identificativo interno all’aggregatore;
* **data**: data e ora di pubblicazione, in formato SQL normalizzato;
* **titolo**: stringa testuale della notizia.

Questi tre campi costituiscono la **knowledge base iniziale** della pipeline e rappresentano il punto di partenza per la fase successiva (5.5), dedicata al recupero dell’URL originale e del contenuto completo degli articoli.

**5.5 Recupero URL originale e contenuto articolo**

Dopo aver recuperato i primi metadati (*titolo, data, url\_cryptopanic*), la pipeline procede recuperando, per ogni articolo, l’url originale (***url\_articolo***) e il contenuto testuale della pagina (***articolo\_completo\_html***).

Per ottenere queste informazioni, viene sfruttata la struttura del sito *CryptoPanic*.  
Caricando nel browser l’*url\_cryptopanic* precedentemente acquisito, la pagina di *CryptoPanic* mostra il **dettaglio dell’articolo** nella colonna di destra.

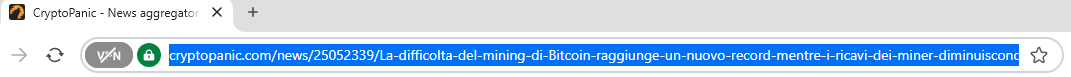
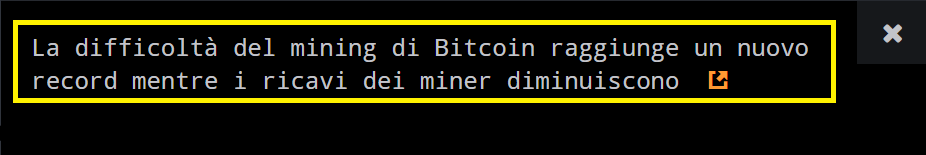


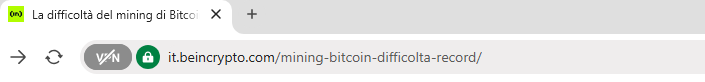
Immagine che contiene testo, schermata, software, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.



Area cliccabile contenente *url\_articolo* originale

In questa sezione il **titolo della notizia** non è un testo statico, ma un **link cliccabile** che punta direttamente alla pagina della fonte originale. Simulando il click su questo elemento, il browser apre una **nuova scheda** con il sito esterno che ha pubblicato l’articolo.





Da questa nuova pagina si ottengono:

* l’**url\_articolo**, leggendo la barra degli indirizzi del browser,
* il **contenuto HTML completo** della notizia, che viene acquisito e successivamente pulito per rimuovere elementi superflui (banner, menu di navigazione, ecc.).

**Iterazione del processo per tutti gli articoli**

Il procedimento appena descritto non viene eseguito manualmente per una singola notizia, ma è iterato automaticamente per tutti gli articoli presenti nel database che non dispongono ancora dell’URL originale e del contenuto.

L’automazione è divisa in tre step principali:

1. Interroga il database per ottenere la lista degli articoli che hanno solo i metadati iniziali (titolo, data, *url\_cryptopanic*);
2. per ciascun elemento della lista apre l’URL di CryptoPanic e ripete il flusso descritto in precedenza: attende il dettaglio nella colonna destra, clicca il titolo, passa alla nuova scheda e legge l’URL originale e l’HTML della fonte esterna;
3. aggiorna il database salvando *url\_articolo* in meta\_articoli e il contenuto HTML in *articolo\_completo\_html.*

Grazie a questa iterazione batch, l’intero dataset viene progressivamente arricchito, senza richiedere alcuna interazione manuale.

**5.5.1 Estrazione del solo testo rilevante dall’ HTML: logica e metrica di similarità**

Nella fase precedente è stato acquisito e salvato nel database l’HTML completo della pagina sorgente. Tuttavia, tale contenuto è spesso arricchito da numerosi elementi accessori (banner, cookie wall, sezioni promozionali, box di articoli correlati) che rendono il codice difficile da interpretare e, soprattutto, poco adatto a una successiva analisi testuale automatica.

*Esempio di HTML “sporco” di una pagina di articolo:*

<body>

  <!-- Banner cookie -->

  <div class="cookie-banner">

    <p>Questo sito utilizza cookie per migliorare l’esperienza. <a href="#">Accetta</a></p>

  </div>

  <!-- Titolo principale -->

  <h1>La difficoltà del mining di Bitcoin raggiunge un nuovo record</h1>

  <!-- Primo paragrafo (testo rilevante) -->

  <p><strong>La difficoltà della rete di Bitcoin è salita sopra i 136 trilioni, creando condizioni più difficili per i miner.</strong></p>

  <!-- Blocco pubblicitario -->

  <div class="ad-banner">

    <p>Sponsored: Scopri la nuova piattaforma di trading con bonus di benvenuto</p>

  </div>

  <!-- Secondo paragrafo (testo rilevante) -->

  <p>L’adeguamento ha segnato un aumento del 4% rispetto ai 129,6 trilioni,

  secondo i dati di Mempool.</p>

  <!-- Intestazione sezione -->

  <h2>I miner affrontano margini sempre più ristretti</h2>

  <!-- Paragrafo ulteriore -->

  <p>La combinazione di difficoltà record e ricavi in calo riduce la redditività dei miner, con possibili conseguenze sul mercato.</p>

  <!-- Box “potrebbe interessarti” (rumore) -->

  <div class="related">

    <p>Potrebbe interessarti: <a href="#">I 7 migliori hardware per il mining</a></p>

  </div>

</body>

Questo esempio evidenzia che non è sufficiente estrarre l’HTML così come appare: occorre un processo di **pulizia e selezione** che consenta di mantenere soltanto il testo coerente con l’articolo principale.

Per raggiungere questo obiettivo, l’algoritmo applica una combinazione di **parsing strutturale** e **filtraggio semantico**. I criteri adottati sono i seguenti:

**A) Logica di selezione (vista HTML)**

1. **Ambito di lavoro.** Si considera esclusivamente il contenuto racchiuso nel tag <body>, che corrisponde a ciè che conterrà i tag testuali dell’ articolo.
2. **Ancoraggio tematico.** Il primo elemento <h1> individuato viene assunto come titolo principale dell’articolo; tutto ciò che lo precede è scartato.
3. **Sequenza di elementi testuali.** Vengono presi in considerazione solo i tag più informativi: <h1>, <h2>, <p>.
4. **Gestione delle sottosezioni.** Ogni intestazione <h2> viene valutata insieme al primo blocco testuale che la segue. La coppia *(h2 + paragrafo)* è mantenuta solo se semanticamente coerente con il tema espresso dall’<h1>.
5. **Coerenza paragrafo–paragrafo.** Per i paragrafi ordinari (<p>), l’inclusione dipende dalla coerenza semantica con il paragrafo precedente già accettato: un paragrafo viene mantenuto se rafforza e prosegue il discorso, altrimenti viene scartato.
6. **Vincoli di qualità.** Infine, il testo aggregato viene validato in base a:
   * **lingua principale**: deve risultare italiano;
   * **ampiezza**: testi troppo brevi (rumore) o eccessivamente lunghi (outlier) vengono esclusi.

Questo processo “a imbuto”, basato sulla gerarchia editoriale tipica (H1 → sezioni H2 → paragrafi) e su criteri di coerenza semantica, riduce drasticamente l’ingresso di contenuti estranei. Il testo risultante, depurato dal rumore, costituisce così la base per le fasi successive di **riassunto automatico, classificazione e sentiment analysis**.

**B) Similarità semantica e soglie decisionali**

***Rappresentazione semantica.*** Ogni unità testuale (titolo <h1>, intestazioni <h2>, paragrafi <p>) viene trasformata in un **vettore denso** (*embedding*) tramite un modello **Sentence-BERT** multilingue. Tali vettori catturano il **significato** delle frasi oltre la semplice corrispondenza lessicale.

**Metrica.** La **similarità coseno** misura l’allineamento tra due vettori vA e vB:

Con vA⋅vB il **prodotto scalare** e ∥v∥ è la **norma euclidea**.

Il valore è compreso tra **–1** e **1**: quanto più è vicino a **1**, tanto più i due testi sono semanticamente affini.

**Soglie utilizzate.**

* **Soglia H2 (coerenza globale)**: la coppia *(H2 + paragrafo successivo)* viene accettata **solo** se la sua similarità con l’**H1** supera una soglia dedicata (es. **0,35**). In questo modo, sezioni fuori tema (FAQ, disclaimer, “correlati”) non entrano nel corpo.
* **Soglia tra paragrafi (coerenza locale)**: un paragrafo viene mantenuto **solo** se è sufficientemente simile al **paragrafo immediatamente precedente** già incluso (es. **0,35**). Si crea così una **catena di coerenza** che segue il filo del discorso ed evita “salti” tematici.

**Interpretazione pratica.**

* Valori **≥ 0,50–0,60** indicano forte continuità semantica (paragrafo da **tenere**).
* Valori **≤ 0,20** indicano discontinuità (paragrafo da **scartare**).
* L’intervallo **0,30–0,45** è una *zona di decisione* tipica: la scelta della soglia si calibra sul dominio e sul livello di rumore delle fonti (ad es. ottimizzando precision/recall su un piccolo set annotato).

***Perché la combinazione “globale + locale” è efficace?***

* Il confronto *(H2 + paragrafo) ↔ H1* garantisce che **ogni sezione** resti ancorata al **tema principale** dell’articolo.
* Il confronto **paragrafo ↔ paragrafo** assicura la **continuità narrativa** e impedisce l’ingresso di blocchi eterogenei (promo, box social, elementi UI) anche quando sono intercalati nel testo.

**C) Vincoli finali e limiti noti**

* **Lingua**: viene conservato solo testo la cui **lingua dominante è l’italiano**; in caso contrario l’articolo è scartato.
* **Lunghezza**: si impone una finestra dimensionale per evitare outlier (pagine troppo corte o troppo lunghe).
* **Limiti**: pagine prive di <h1>, paywall dinamici o contenuti fortemente visuali (grafici/tabelloni) possono ridurre la qualità dell’estrazione semantica.

**D) Risultato**

Il testo che supera i filtri **strutturali** (tag) e **semantici** (soglie coseno) viene aggregato e memorizzato come **corpo pulito dell’articolo**, pronto per le successive fasi di **riassunto con LLM**, **classificazione** e **analisi del sentiment**.

**Esempio di pulizia dell’ HTML con similarità del coseno**

Considerando “*Esempio di HTML “sporco” di una pagina di articolo”* supponiamo di avere tre frasi etsratte dall’ HTML:

* **F1 (titolo H1):**  
  *“La difficoltà del mining di Bitcoin raggiunge un nuovo record.”*
* **F2 (paragrafo coerente):**  
  *“L’aumento della difficoltà riduce la redditività dei miner di Bitcoin.”*
* **F3 (paragrafo fuori tema):**  
  *“Il prezzo di Dogecoin cresce del 10% nelle ultime 24 ore.”*

Ogni frase viene trasformata in un vettore denso (*embedding*)   
Per semplicità, ipotizziamo di ridurre i vettori a 3 dimensioni (nella realtà hanno 384 o più):

* vF1=(0.8,  0.5,  0.1) vF2=(0.7,  0.4,  0.2) vF3=(0.1,  0.9,  0.7)

**Successivamente si calcola la similarità del coseno per ognuno di essi:**

**Caso 1: F1 ↔ F2 (coerenti)**

➡️ Altissima similarità → il paragrafo viene **mantenuto**.

**Caso 2: F1 ↔ F3 (fuori tema)**

➡️ Similarità media-bassa → paragrafo **scartato** perché non tratta lo stesso tema.

Dopo l’applicazione delle regole e delle soglie di similarità (es. 0.35 come valore minimo):

La difficoltà del mining di Bitcoin raggiunge un nuovo record

La difficoltà della rete di Bitcoin è salita sopra i 136 trilioni,

creando condizioni più difficili per i miner.

L’adeguamento ha segnato un aumento del 4% rispetto ai 129,6 trilioni,

secondo i dati di Mempool.

I miner affrontano margini sempre più ristretti

La combinazione di difficoltà record e ricavi in calo riduce la redditività

dei miner, con possibili conseguenze sul mercato.

**6. Sintesi automatica delle notizie**

Dopo aver raccolto gli articoli e i relativi contenuti testuali, è stato eseguito il *fine-tuning* del modello *LLaMA 3.2 3B*, con l’obiettivo di adattarlo alla generazione automatica di sintesi, sia brevi che estese dell’ articolo.

LLaMA 3.2 3B è un modello basato sull’architettura *Transformer*, con circa 3 miliardi di parametri, progettato per comprendere e generare linguaggio naturale garantendo un equilibrio efficace tra accuratezza ed efficienza computazionale.

Il fine-tuning ha lo scopo di ottimizzare il modello rispetto al dominio di interesse (articoli sul mercato delle criptovalute), migliorandone la capacità di identificare e sintetizzare le informazioni chiave. In particolare, il sistema produce due tipologie di sintesi:

* *Sintesi brevi*: versioni compatte dei testi, adatte a fornire una rapida anteprima.
* *Sintesi estese*: versioni più dettagliate e articolate, utili per un’analisi approfondita.

**6.1 Introduzione ai modelli LLM e scelta di LLaMA 3.2**

Per produrre riassunti brevi ed estesi a partire dal corpo testuale pulito degli articoli raccolti, si ricorre ai Large Language Models (LLM), ovvero modelli di intelligenza artificiale addestrati su grandi quantità di testi per apprendere le regolarità statistiche della lingua naturale.

Un LLM si basa tipicamente sull’architettura *Transformer*, introdotta da Vaswani et al. (2017), che ha rivoluzionato il campo del Natural Language Processing (NLP). Il meccanismo di self-attention permette al modello di catturare relazioni di lungo raggio tra parole e frasi, superando i limiti dei modelli sequenziali precedenti (RNN, LSTM). Questo approccio ha reso i LLM strumenti estremamente potenti non solo per la generazione testuale, ma anche per il riassunto, la traduzione, la classificazione e la risposta a domande.

Negli ultimi anni sono stati sviluppati numerosi LLM open-source, come GPT-Neo, GPT-J, Falcon e Mistral. All’interno di questa tesi è stato scelto di utilizzare LLaMA (Large Language Model Meta AI) nella versione 3.2, rilasciata da Meta AI, per diversi motivi:

*Prestazioni elevate*: LLaMA 3.2 rappresenta un’evoluzione delle versioni precedenti, con miglioramenti in termini di coerenza, comprensione contestuale e capacità di generare testi in più lingue, tra cui l’italiano.

*Efficienza:* rispetto a modelli commerciali di dimensioni simili, LLaMA è ottimizzato per un’esecuzione più leggera, con consumi computazionali ridotti. Ciò lo rende adatto anche ad ambienti con risorse hardware limitate.

*Open-source e comunità attiva*: essendo rilasciato con licenza permissiva, permette la piena personalizzazione tramite *fine-tuning* e integrazione in pipeline specifiche. Inoltre, la comunità di ricerca fornisce aggiornamenti e ottimizzazioni costanti.

*Supporto multilingua*: l’addestramento su corpora multilingue rende LLaMA adatto a generare riassunti anche in italiano, caratteristica cruciale per questo progetto.

Nel contesto della pipeline sviluppata, LLaMA 3.2 è stato utilizzato in modalità *prompt-based*, fornendo in input il corpo pulito dell’articolo insieme a istruzioni testuali che definiscono il tipo di sintesi desiderata (breve o estesa). Questa scelta permette di sfruttare le capacità di comprensione contestuale del modello senza necessità di un addestramento da zero, riducendo i tempi di sviluppo e i costi computazionali.

L’impiego di LLaMA 3.2 consente dunque di trasformare testi lunghi e ridondanti in riassunti compatti e coerenti, migliorando l’accessibilità e l’utilità delle notizie per l’utente finale e costituendo la base per le fasi successive di classificazione e sentiment analysis.

**6.3 Strumenti utilizzati**

Il fine-tuning del modello è stato eseguito su *Google Colab*, una piattaforma che mette a disposizione risorse computazionali gratuite, tra cui GPU, facilitando l’addestramento di modelli di grandi dimensioni senza la necessità di infrastrutture dedicate.

Per l’implementazione del modello e la gestione del processo di addestramento sono state impiegate diverse librerie Python, ognuna con un ruolo specifico:

* + *unsloth*, caricamento del modello LLaMA 3.2 3B e supporto al processo di fine-tuning.
  + *torch (PyTorch)*, framework per il calcolo numerico e l’utilizzo della GPU nelle fasi di addestramento e ottimizzazione dei parametri.
  + *pandas* , manipolazione e gestione strutturata del dataset.
  + *datasets (Hugging Face)*, conversione e gestione del dataset in formati compatibili con i modelli Hugging Face.
  + *trl (Transformers Reinforcement Learning)*, addestramento supervisionato tramite la classe *SFTTrainer*, ottimizzata per scenari di fine-tuning.
  + *transformers (Hugging Face)*, definizione del modello, dei tokenizer e dei parametri di addestramento.
  + *scikit-learn (sklearn),* suddivisione del dataset in *training set* e *validation set*, utile per monitorare le prestazioni del modello.

**6.4 Creazione del dataset per il Fine-Tuning**

Il dataset utilizzato per il fine-tuning del modello è stato creato manualmente. Per ciascun articolo raccolto nella fase di data collection, sono stati generati due tipi di sintesi (breve ed estesa) tramite l’ausilio di assistenti AI, con l’obiettivo di fornire esempi supervisionati al modello LLaMA 3.2.

Il dataset comprende complessivamente 1.000 record, suddivisi come segue:

* *500 record con riassunti brevi*;
* *500 record con riassunti lunghi* .

Il dataset è organizzato in formato CSV, con due colonne principali:

* *User*: contenente il prompt e il testo completo dell’articolo;
* *Summary*: contenente il riassunto che ci si aspetta di ottenere in output.

Il file è disponibile nella cartella *Dataset* con il nome summaryDataset.csv.

* + **Suddivisione del dataset in Training e Validation Set**

Per garantire un addestramento corretto ed evitare fenomeni di overfitting, il dataset è stato suddiviso in:

* + *80% Training Set,* utilizzato per l’aggiornamento dei pesi del modello;
  + *20% Validation Set*, usato per valutare le prestazioni durante l’addestramento.

La suddivisione è stata effettuata tramite la funzione train\_test\_split della libreria **Scikit-Learn**:

from sklearn.model\_selection import train\_test\_split

train\_df,val\_df = train\_test\_split(df,test\_size=0.2,random\_state=42)

**Configurazione dei parametri di addestramento**

Il fine-tuning è stato condotto in modalità supervisionata, utilizzando la classe *SFTTrainer* della libreria *trl* (Hugging Face). I parametri principali sono stati configurati come segue:

args = TrainingArguments(

    per\_device\_train\_batch\_size=2,

    gradient\_accumulation\_steps=4,

    warmup\_steps=5,

    max\_steps=100,

    learning\_rate=0.0001,

    fp16=not is\_bfloat16\_supported(),

    bf16=is\_bfloat16\_supported(),

    optim="adamw\_8bit",

    eval\_strategy="steps",

    eval\_steps=10,

    save\_strategy="epoch",

    metric\_for\_best\_model="eval\_loss",

    greater\_is\_better=False,

    output\_dir="outputs",

    report\_to="none",

)

**Parametri principali e loro impatto**

**Learning Rate (LR)**

Il LR controlla la *velocità di apprendimento* del modello.

E’ un fattore moltiplicativo che regola di quanto vengono modificati i pesi del modello dopo ogni aggiornamento:

Con:

* , i pesi del modello
* indica la direzione e la grandezza della modifica necessaria per ridurre la loss.

Se il LR è troppo alto, il modello impara rapidamente ma potrebbe sfuggire a minimi locali a causa delle grandi oscillazioni.

Se è troppo basso, il modello potrebbe apprendere lentamente raggiungendo buone performance ma rischia di essere intrappolato in minimi locali.

**Batch Size**

Il *Batch Size* definisce quanti campioni vengono processati contemporaneamente dalla rete neurale prima di aggiornare i pesi.

Se è troppo piccolo aggiorna i pesi più frequentemente ma può causare oscillazioni grandi dei pesi poiché considera pochi esempi.

Se è troppo grande aggiorna i pesi meno frequentemente e considerando più esempi rendendo il training più stabile (meno oscillzazioni) ma rende l’overfitting più probabile imparando schemi troppo specifici ai dati di training. Inoltre non è applicabile richiede molta VRAM altrimenti può portare a errori di OOM (Out of memory)

**Gradient Accumulation**

Il modello accumula i gradienti per più batch prima di aggiornare i pesi.

Consente di simulare batch più grandi senza aumentare il consumo di memoria.

E’ una tecnica comunemente usata con batch size:

*Esempio:*

* *Senza Gradient Accumulation***:** batch\_size=8 → Aggiornamento dei pesi dopo ogni 8 campioni.
* *Con Gradient Accumulation*(steps=4, batch\_size=2):
  + Il modello elabora 4 batch da 2 campioni → accumula i gradienti.
  + Solo alla fine dei 4 batch aggiorna i pesi.

Il risultato è equivalente a un batch di 8, ma senza caricare 8 campioni in memoria contemporaneamente!

**Warmup Steps**

Controlla il numero di passi iniziali in cui il learning rate cresce gradualmente da zero fino al valore target. Utile perché quando il training inizia i pesi del modello sono casuali, quindi un learning rate alto può causare gradienti instabili.

Un warmup basso (1-5%) è ideale per il fine-tuning di modelli già pre-addestrati.

Un warmup alto (10-20%) è utile per modelli addestrati da zero.

**Max Steps**

Numero massimo di iterazioni di addestramento.

Troppi passi → rischio di overfitting.

Pochi passi → rischio di sotto-addestramento.

**Strategia di selezione dei parametri**

Per ottimizzare le prestazioni del modello è stata effettuata una ricerca di tipo *greedy search* testando diverse combinazioni di parametri chiave (Learning Rate, Batch Size, Gradient Accumulation), mantenendo fissi:

* + Max Steps = 100
  + Warmup Steps = 5 (pari al 5% dei passi)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Config ID** | **Learning Rate (LR)** | **Batch Size (BS)** | **Gradient Accumulation (GA)** | **Effettivo Batch Size (BS \* GA)** |
| A | 0.0002 | 8 | 8 | 64 |
| B | 0.0001 | 2 | 4 | 8 |
| C | 0.0005 | 4 | 4 | 16 |
| D | 0.0005 | 8 | 2 | 16 |
| E | 0.0005 | 2 | 2 | 4 |
| F | 0.0003 | 4 | 2 | 8 |
| G | 0.0001 | 2 | 8 | 16 |
| H | 0.00005 | 4 | 2 | 8 |
| I | 0.00005 | 8 | 2 | 16 |
| J | 0.0002 | 2 | 4 | 8 |
| K | 0.0004 | 4 | 4 | 16 |
| L | 0.003 | 6 | 3 | 18 |

**Addestramento con le configurazioni generate**

|  |  |  |
| --- | --- | --- |
| **Config ID = A** | **Config ID = B**  **Immagine che contiene testo, schermata, Carattere, numero  Il contenuto generato dall'IA potrebbe non essere corretto.** | **Config ID = C** |
| **Config ID = D**  **Immagine che contiene testo, schermata, Carattere, numero  Il contenuto generato dall'IA potrebbe non essere corretto.** | **Config ID = E** | **Config ID = F** |
| **Config ID = G**  **Immagine che contiene testo, schermata, Carattere, numero  Il contenuto generato dall'IA potrebbe non essere corretto.** | **Config ID = H** | **Config ID = I** |
| **Config ID = J** | **Config ID = K** | **Config ID = L** |

**Valutazione delle configurazioni dei parametri**

Per determinare la migliore configurazione, valutiamo ogni setup in base a:

* *Training Loss* – Indica quanto bene il modello apprende i dati di training.
* *Validation Loss* –Indica la capacità del modello di generalizzare su dati nuovi.
* *Differenza tra Training e Validation Loss* – Se la Training Loss è molto più bassa della Validation Loss, potrebbe esserci overfitting.

|  |  |  |  |
| --- | --- | --- | --- |
| **Config ID** | **Training Loss (TL)** | **Validation Loss (VL)** | **Valutazione** |
| A | 0.491000 | 0.903284 | **Overfitting**: TL troppo più bassa della VL. |
| B | 0.916300 | 0.858984 | **TL alta**: Il modello sta imparando lentamente. |
| C | 0.618100 | 0.833356 | **Bilanciata**: Buona generalizzazione e training loss accettabile. |
| D | 0.375000 | 0.993567 | **Overfitting**: VL troppo alta rispetto alla TL. |
| E | 0.607200 | 0.850270 | **Accettabile,** ma leggermente inferiore a C. |
| F | 0.698900 | 0.858345 | **Accettabile,** ma leggermente peggiore di C. |
| G | 0.793300 | 0.845042 | **TL alta**: Il modello sta imparando lentamente. |
| H | 0.721300 | 0.978577 | **Underfitting**: VL troppo pià alta della TL. |
| I | 0.385900 | 0.943293 | **Overfitting**: TL troppo più bassa della VL. |
| J | 0.679200 | 0.915365 | **Accettabile,** ma VL più alta di C, E ed F. |
| K | 0.254700 | 1.176784 | **Overfitting**: TL troppo più bassa della VL. |
| L | 0.530000 | 0.837541 | **Overfitting**: TL troppo più bassa della VL. |

**Selezione della configurazione migliore**

Sulla base delle valutazioni effettuate, selezionimao la configurazione C che risulta essere la migliore perché offre il miglior equilibrio tra training loss e validation loss, garantendo una buona capacità di apprendimento senza incorrere in overfitting. Con una training loss di 0.6183 e una validation loss di 0.8345, il modello dimostra di generalizzare bene ai dati non visti, evitando sia un apprendimento eccessivo dei dati di training sia un sotto-addestramento.

**NOTA:**

Generalmente, la selezione della configurazione ottimale dovrebbe avvenire in due fasi:

1. *Fase di test preliminare*: valutazione delle configurazioni su 50-100 steps.
2. *Fase di validazione definitiva:* test della migliore configurazione su 300, 500 o 1000 steps per consolidarne l'efficacia.

Tuttavia, in questo caso, il dataset dispone di circa 1000 esempi, e l'addestramento con un numero elevato di steps su tutte le configurazioni porterebbe a overfitting, poiché gli stessi esempi verrebbero considerati troppe volte.

Quindi, data la dimensione limitata del dataset, procederemo direttamente con la Configurazione C, senza ulteriori test con numeri differenti di steps, per evitare l’overfitting.

**Utilizzo della configurazione migliore**

La configurazione selezionata è stata utilizzata per creare il codice su Google Colab che espone l’API per utilizzare il modello addestrato e questa API verrà richiamata all’ interno del nostro script principale.

L’API è visibile al seguente [link](https://colab.research.google.com/drive/1BeXsoEdg8FoHh3cgpzLFJuGl8A4BTzII?usp=sharing).

**7 Classificazione delle notizie**

L'obiettivo di questa fase del progetto è classificare automaticamente le notizie del mondo crypto in diverse categorie (Regolamentazione, Mercato, Adozione, Innovazione, Sicurezza, ecc.).  
Per raggiungere questo scopo, viene adottato un approccio di *classificazione semi-supervisionata*, combinando due tecniche di classificazione:

* *Clustering rigido con K-Means:* utilizzato per raggruppare automaticamente gli articoli in cluster senza etichette iniziali.
* *Classificazione supervisionata con Naïve Bayes:* una volta ottenuti i cluster, questi vengono interpretati e assegnati a categorie significative. Successivamente, Naïve Bayes viene addestrato su questi dati per classificare nuovi articoli.

Questa metodologia permette di identificare pattern nascosti nei dati e ridurre la necessità di etichettatura manuale.

**Pre-elaborazione del dataset: conversione in vettori TF-IDF**

Poiché gli algoritmi di classificazionelavorano con dati numerici, il testo deve essere convertito in un formato utilizzabile.  
A questo scopo viene utilizzata la tecnica TF-IDF(Term Frequency - Inverse Document Frequency), che assegna un peso a ciascuna parola in base alla sua rilevanza nel dataset.

Passaggi chiave della trasformazione TF-IDF:

1. Eliminazione delle stopwords italiane *(es: "il", "la", "con", "per")* per migliorare la qualità dei dati.
2. Calcolo del TF (Term Frequency): Il TF indica quante volte una parola appare in un documento rispetto al numero totale di parole.
3. Calcolo del IDF (Inverse Document Frequency), riduce l'importanza delle parole troppo comuni e aumenta l'importanza delle parole distintive.

Dove:

* + N = numero totale di documenti
  + DF = numero di documenti in cui la parola appare

1. Calcolo di TF-IDF per ogni parola nel documento sarà calcolata TF-IDF che sarà alta se è una parola distintiva, mentre sarà bassa se è una parola comune.

**Esempio:**

Ipotizzando di aver applicato il metodo TF-IDF descritto precedentemente su degli articoli di esempio otterremo un risultato del genere:

|  |  |  |  |
| --- | --- | --- | --- |
| **Termini** | **Articolo 1** | **Articolo 2** | **Articolo 3** |
| Bitcoin | 0.45 | 0.50 | 0.00 |
| Ethereum | 0.00 | 0.30 | 0.00 |
| Hacker | 0.00 | 0.00 | 0.80 |
| Dollari | 0.35 | 0.10 | 0.00 |

Dall’ esempio precedente si capisce che ogni termine rappresenta una feature con un valore numerico e questo permette di rappresentare nello spazio il documento.

**Clustering rigido: K-Means per identificare cluster**

L'algoritmo *K-Means* è una tecnica di *clustering rigido,* che suddivide un insieme di dati in K gruppi distinti, minimizzando la distanza tra ogni punto e il centroide del suo cluster.

**Strumenti utilizzati**

Le principali librerie utilizzate per l’implementazione del modello e il processo di addestramento sono le seguenti:

* *pandas* - Per la gestione e manipolazione del dataset (gestione CSV)
* *matplotlib* - Per la visualizzazione della curva del gomito e l'identificazione del numero ottimale di cluster.
* *sklearn.feature\_extraction.text (TfidfVectorizer)* - Per convertire il testo degli articoli in vettori numerici (matrice TF-IDF) utilizzabili dall’algoritmo di clustering.
* *sklearn.cluster (KMeans)* - Per eseguire il clustering degli articoli basato sui vettori TF-IDF generati, e per determinare i centroidi di ogni cluster.
* *kneed (KneeLocator)* - Per identificare automaticamente il "gomito" nella curva dell'inertia e determinare il numero ottimale di cluster.
* *nltk (stopwords)* - Per rimuovere le parole comuni irrilevanti (stopwords) dalla matrice TF-IDF e migliorare l'efficacia del clustering.

**Dataset utilizzato**

Il dataset utilizzato è “datasetKMenas.csv” formattato nel seguente modo:

|  |  |  |
| --- | --- | --- |
| **id\_articolo** | **long\_summary** | **categoria** |
| 34 | Uniswap Labs ha lanciato la mainnet di Unichain, un Layer-2 di Ethereum con block time di un secondo… |  |
| 35 | Il prezzo di ENA, token di Ethena, è sceso del 10% dopo che una balena ha spostato 18 milioni di token su Binance…. |  |
| 36 | Ventiquattro stati USA stanno valutando proposte per creare riserve di Bitcoin…. |  |

Il campo “categoria” è rimasto vuoto perché sarà valorizzato dall’algoritmo Kmeans.

**Determinare il numero ottimale di cluster - Curva del Gomito**

Uno dei problemi principali di K-Means è la scelta del numero ottimale di cluster K. Per determinarlo,

viene utilizzato il metodo della curva del gomito, che segue questi passi:

1. *Eseguire K-Means con diversi valori di K (da 1 a 10).*
2. *Calcolare l'Inertia* *,* cioè la somma delle distanze quadrate tra ogni punto e il centroide del cluster.
3. *Plottare la curva del gomito*, osservando il punto in cui la riduzione dell'inertia diventa meno significativa.
4. *Selezionare il valore ottimale di K,* che corrisponde al "gomito" della curva.

Questa tecnica assicura che il numero di cluster scelto sia né troppo basso (raggruppamento impreciso), né troppo alto (cluster eccessivamente specifici).

L’algoritmo utilizzato è il seguente:

# Convertire il testo in numeri con TF-IDF

vectorizer = TfidfVectorizer(stop\_words=italian\_stopwords, max\_features=5000)

X\_tfidf = vectorizer.fit\_transform(df['titolo'])

# Determinare il numero ottimale di cluster usando la curva del gomito

inertia\_values = []

clusters\_range = range(1, 11)

for k in clusters\_range:

    kmeans = KMeans(n\_clusters=k, n\_init=5, init='random', random\_state=42)

    kmeans.fit(X\_tfidf)

    inertia\_values.append(kmeans.inertia\_)

# Identificare il "gomito" nella curva

knee\_locator = KneeLocator(clusters\_range, inertia\_values, curve="convex", direction="decreasing")

optimal\_clusters = knee\_locator.elbow

# Plottare la curva del gomito

plt.figure(figsize=(8, 5))

plt.plot(clusters\_range, inertia\_values, marker='o', linestyle='--')

plt.xlabel("Numero di cluster")

plt.ylabel("Inertia")

plt.title("Curva del Gomito per determinare il numero di cluster ottimale")

plt.axvline(optimal\_clusters, color='r', linestyle='--')

plt.show()

**Valutazione dei risultati – Curva del Gomito**Immagine che contiene testo, linea, schermata, Diagramma

Il contenuto generato dall'IA potrebbe non essere corretto.

Eseguendo l’algoritmo precedentemente verranno testate le classificazioni fatte con un numero diverso di cluster partendo da K=1, fino a K=10.

L'immagine mostra la Curva del Gomito utilizzata per determinare il numero ottimale di cluster. Si può vedere che il numero di cluster ottimale è K=6 perché dopo questa soglia la diminuzione dell’ inertia è meno significativa, quindi aggiungere altri cluster non porterebbe a una divisione milgiore.

**Esecuzione K-Means con K=6 e dataset aggiornato**

Eseguire K-Means con il numero ottimale di cluster

kmeans = KMeans(n\_clusters=optimal\_clusters, n\_init=5, init='random', random\_state=42)

df['categoria'] = kmeans.fit\_predict(X\_tfidf)

# Salvare il dataset aggiornato

output\_file = "dataset\_KMeans\_classificato\_long.csv"

df.to\_csv(output\_file, index=False)

Dopo aver analizzato il datest classificato è stato possibile individurare le seguenti categorie:

|  |  |  |
| --- | --- | --- |
| **Cluster** | **Numero di Articoli** | **Possibile Categoria** |
| 0 | 219 | Adozione e Mainstreaming |
| 1 | 291 | Analisi Tecniche e Sentimenti di Mercato |
| 2 | 20 | Non rilevanti |
| 3 | 244 | Andamento del Mercato e Prezzi |
| 4 | 342 | Innovazione e Nuovi Progetti |
| 5 | 74 | Sicurezza, hackeraggi e Truffe |

**Affinamento manuale della classificazione**

Il Kmeans è stato molto utile per una prima classificazione, ma analizzando i risultati e gli assegnamenti ci sono alcuni articoli e caetgorie molto simili, soprattutto le categorie: “Analisi Tecniche e Sentiment di Mercato” e la categoria “Andamento del Mercato e Prezzi” leggendo alcuni articoli è difficile stabilire effettivamente a quale categoria facciano parte.

Per questo motivo si è preferito accorpare queste due categorie nella categoria “News di Mercatom Analisi e Prezzi” ed aggiungere una nuova categoria più importante chiamata “Regolamentazione e Normative”.

Dopo questo affinamento manuale, il dataset è stato riclassificato con le seguenti 6 categorie con questa distribuzione:

|  |  |
| --- | --- |
| **Categoria** | **Numero di Articoli** |
| Adozione e Mainstreaming | 135 |
| Innovazione e Nuovi Progetti | 146 |
| News di Mercato, Analisi e Prezzi | 547 |
| Non rilevanti | 36 |
| Regolamentazione e Normative | 194 |
| Sicurezza, Hackeraggi e Truffe | 134 |

Ovviamente molte sono categorie di ambito di analisi di mercato e prezzi quindi è normale questa discrepanza di articoli , cioè è normale che esistono molti più articoli nella categoria “News di Marcato,Analisi e Prezzi” rispetto alle altre categorie.

**Apprendimento supervisionato: Naive Bayes**

Dopo aver utilizzato Kmeans e una classificazione manuale per la divisione degli articoli in categorie si vuole addestrare un modello in modo tale da utilizzarlo per la classificazione degli articoli successivi.

E’ stato scelto l’algoritmo Naive Bayes per la sua semplicità, efficienza e buona accuratezza nei problemi di classificazione di testo, in quanto assume che tutte le feature (parole nel testo) siano indipendenti tra loro, semplificando notevolmente il calcolo della probabilità.

**Strumenti utilizzati**

Le principali librerie utilizzate per l’implementazione del modello e il processo di addestramento sono le seguenti:

* **pandas** - Per la gestione e manipolazione dei dataset, inclusa la lettura e pulizia dei dati.
* **matplotlib** e **seaborn** - Per la creazione di grafici e visualizzazioni, in particolare la matrice di confusione normalizzata.
* **sklearn (scikit-learn)** - Libreria principale per:
  + **TfidfVectorizer** - Conversione del testo in rappresentazioni numeriche (matrici TF-IDF).
  + **train\_test\_split** - Suddivisione del dataset in training e test set.
  + **MultinomialNB (Naive Bayes)** - Algoritmo supervisionato per la classificazione testuale.
  + **classification\_report, confusion\_matrix, accuracy\_score** - Metriche per valutare il modello.
* **nltk (Natural Language Toolkit)** - Per l’uso delle stopwords italiane nel preprocessing del testo.
* **numpy** - Per operazioni matematiche, come la normalizzazione della matrice di confusione.
* **pickle** - Per salvare il modello addestrato e il vettorizzatore per futuri utilizzi.

**Suddivisione del dataset in Training e Validation Set**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_tfidf, y, test\_size=0.2, random\_state=42)

Il dataset è stato suddiviso in:

* 80% Training Set (utilizzato per l'addestramento del modello)
* 20% Validation Set (usato per la valutazione durante il training)

**Implementazione dell’ algoritmo**

Per implemenatre l’algoritmo è stato utilizzato il seguente codice:

# Inizializzare e addestrare il modello Naive Bayes

model = MultinomialNB()

model.fit(X\_train, y\_train)

# Prevedere le categorie sul set di test

y\_pred = model.predict(X\_test)

# Valutare il modello

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuratezza del modello: {accuracy:.2f}")

# Stampare il classification report

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

# Stampare la matrice di confusione

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

labels = sorted(df['category'].unique())  # Ordinare i nomi delle classi

# Normalizzare la matrice per mostrare le proporzioni

conf\_matrix\_norm = conf\_matrix.astype('float') / conf\_matrix.sum(axis=1)[:, np.newaxis]

# Plot della matrice di confusione

plt.figure(figsize=(14, 10))  # Aumenta la dimensione della figura

sns.heatmap(conf\_matrix\_norm, annot=True, cmap='Blues', fmt='.2%', xticklabels=labels, yticklabels=labels)

plt.xticks(rotation=45, ha='right', fontsize=10)  # Ruota le etichette e riduce il testo

plt.yticks(fontsize=10)  # Riduce il testo delle etichette Y

plt.title('Matrice di Confusione Normalizzata (Proporzioni %)')

plt.xlabel('Previsioni')

plt.ylabel('Vero Valore')

plt.tight\_layout()  # Per garantire che tutto rientri nel grafico

plt.show()

**Valutazione dei Risultati**

La valutazione del modello Naive Bayes addestrato è stata effettuata utilizzando diverse metriche standard per problemi di classificazione: Accuratezza, Precisione, Recall e F1-Score. I risultati delle metriche sono stati ottenuti confrontando le predizioni del modello con le etichette effettive del set di test.

**Metriche di Valutazione**

* Accuracy (Accuratezza): Percentuale di predizioni corrette sul totale degli esempi nel set di test.
* Precision: Percentuale di esempi correttamente classificati rispetto al totale degli esempi previsti per una determinata classe.
* Recall (Sensibilità): Percentuale di esempi correttamente classificati rispetto al totale degli esempi appartenenti a una determinata classe.
* F1-Score: Media armonica tra Precision e Recall, che bilancia l'importanza tra i due.

Il modello ha raggiunto un'accuratezza complessiva del 70%. La tabella seguente mostra il Classification Report suddiviso per ogni categoria:

Accuratezza del modello: 0.70

Classification Report:

                                   precision    recall  f1-score   support

         Adozione e Mainstreaming       0.86      0.30      0.44        20

     Innovazione e Nuovi Progetti       1.00      0.45      0.62        29

News di Mercato, Analisi e Prezzi       0.60      0.98      0.75       103

                    Non rilevanti       1.00      0.40      0.57         5

     Regolamentazione e Normative       0.90      0.59      0.71        46

   Sicurezza, Hackeraggi e Truffe       0.94      0.49      0.64        35

                         accuracy                           0.70       238

                        macro avg       0.88      0.53      0.62       238

                     weighted avg       0.79      0.70      0.68       238

**Matrice di Confusione**

È stata generata anche una Matrice di Confusione Normalizzata per mostrare graficamente le prestazioni del modello, indicando le proporzioni di predizioni corrette e sbagliate per ogni classe (Recall per ogni classe).

Ogni riga rappresenta le etichette effettive (vere categorie), mentre ogni colonna rappresenta le etichette previste (previsioni del modello). Le celle della matrice mostrano quante volte il modello ha assegnato correttamente o erroneamente un'etichetta.

Immagine che contiene testo, schermata, diagramma, Parallelo

Il contenuto generato dall'IA potrebbe non essere corretto.

Valutiamo i risultati per ogni categoria:

1. **Adozione e Mainstreaming**:

* Corretta classificazione: **30.00%**.
* Viene erroneamente classificato come **News di Mercato, Analisi e Prezzi (65.00%)**.
* Piccole percentuali di errore anche in altre Regolamentazione e Normatve (5%)

1. **Innovazione e Nuovi Progetti**:

* Corretta classificazione: **44.83%**.
* Confusione con **News di Mercato, Analisi e Prezzi (51.72%)**.
* Un leggero errore anche in **Sicurezza, Hackeraggi e Truffe (3.45%)**.

1. **News di Mercato, Analisi e Prezzi**:

* Corretta classificazione: **98.06%**. (Ottimo risultato)
* Errori trascurabili in altre categorie.

1. **Non rilevanti**:

* Corretta classificazione: **60.00%**.
* Confusione significativa con **Innovazione e Nuovi Progetti (40.00%)**.

1. **Regolamentazione e Normative**:

* Corretta classificazione: **58.70%**.
* Confusione rilevante con **News di Mercato, Analisi e Prezzi (39.13%)**.

1. **Sicurezza, Hackeraggi e Truffe**:

* Corretta classificazione: **48.57%**.
* Spesso confuso con **News di Mercato, Analisi e Prezzi (51.43%)**.

**Miglioramento con Laplace Smoothing**

Analizzando i risultati della matrice di confusione precedente, si può notare che alcune categorie presentano una percentuale di corretta classificazione inferiore al 60%. Questo comportamento è dovuto principalmente allo sbilanciamento dei dati, dove le notizie appartenenti alla categoria “News di Mercato, Analisi e Prezzi” sono predominanti rispetto alle altre categorie.

Per migliorare le prestazioni del modello, è stato utilizzato un metodo chiamato **Laplace Smoothing.**

La Laplace Smoothing è una tecnica utilizzata per gestire il problema dello zero-frequency problem. Questo problema si verifica quando una parola o caratteristica non è presente nei dati di addestramento per una determinata classe. Senza la Laplace Smoothing, la probabilità di quella parola sarebbe zero, e il modello considererebbe impossibile la classificazione corretta.

Il principio della Laplace Smoothing è quello di aggiungere un piccolo valore (detto alpha) a tutte le possibili combinazioni di parole e categorie, evitando così probabilità zero. Questo processo rende il modello più robusto e meno incline ad escludere categorie per mancanza di dati.

Il modello Naive Bayes viene configurato con il parametro **alpha** che controlla l'entità dello smoothing:

* **Alpha = 1.0** (default): Applicazione standard della Laplace Smoothing, aggiunge un conteggio di “1” a tutte le parole.
* **Alpha < 1.0** (es. 0.1): Riduce l'effetto dello smoothing, dando maggiore importanza ai termini rari.
* **Alpha > 1.0** (es. 2.0 o maggiore): Aumenta l'effetto dello smoothing, rendendo il modello più robusto ma meno sensibile ai dettagli.

Nel nostro caso, per dare maggiore peso ai termini rari, migliorando la capacità del modello di distinguere tra categorie con meno esempi di addestramento dobbiamo considerare le configurazioni con alpha < 1

model = MultinomialNB(alpha=0.1)  # Regolazione del parametro alpha

**Test con Laplace Smoothing**

Generiamo il parametro alpha da 0 a 0.9 con una distanza di 0.1 per ogni configurazione e selezioniamo la configurazione con percentuale di correttezza media migliore:

|  |  |
| --- | --- |
| **Test con alpha = 0** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 100,00% |
| Innovazione e Nuovi Progetti | 0% |
| News di Mercato, Analisi e Prezzi | 21,36% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 0% |
| Sicurezza, Hackeraggi e Truffe | 0% |
| **Media Percentuale Totale** | **26,89%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.1** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 80,00% |
| Innovazione e Nuovi Progetti | 86,21% |
| News di Mercato, Analisi e Prezzi | 82,52% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 86,96% |
| Sicurezza, Hackeraggi e Truffe | 85,71% |
| **Media Percentuale Totale** | **76,9%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.2** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 75,00% |
| Innovazione e Nuovi Progetti | 86,21% |
| News di Mercato, Analisi e Prezzi | 85,44% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 91,30% |
| Sicurezza, Hackeraggi e Truffe | 80,00% |
| **Media Percentuale Totale** | **76,325%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.3** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 55,00% |
| Innovazione e Nuovi Progetti | 86,21% |
| News di Mercato, Analisi e Prezzi | 90,29% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 84,78% |
| Sicurezza, Hackeraggi e Truffe | 77,14% |
| **Media Percentuale Totale** | **72,23%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.4** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 50,00% |
| Innovazione e Nuovi Progetti | 82,76% |
| News di Mercato, Analisi e Prezzi | 93,20% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 78,26% |
| Sicurezza, Hackeraggi e Truffe | 74,29% |
| **Media Percentuale Totale** | **69,75%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.5** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 55,00% |
| Innovazione e Nuovi Progetti | 65,52% |
| News di Mercato, Analisi e Prezzi | 94,17% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 76,09% |
| Sicurezza, Hackeraggi e Truffe | 74,29% |
| **Media Percentuale Totale** | **67,51%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.6** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 50,00% |
| Innovazione e Nuovi Progetti | 62,07% |
| News di Mercato, Analisi e Prezzi | 96,12% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 71,74% |
| Sicurezza, Hackeraggi e Truffe | 65,71% |
| **Media Percentuale Totale** | **64,26%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.7** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 40,00% |
| Innovazione e Nuovi Progetti | 62,07% |
| News di Mercato, Analisi e Prezzi | 96,12% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 65,22% |
| Sicurezza, Hackeraggi e Truffe | 65,71% |
| **Media Percentuale Totale** | **61,5%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.8** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 35,00% |
| Innovazione e Nuovi Progetti | 51,72% |
| News di Mercato, Analisi e Prezzi | 97,09% |
| Non rilevanti | 40% |
| Regolamentazione e Normative | 63,04% |
| Sicurezza, Hackeraggi e Truffe | 51,43% |
| **Media Percentuale Totale** | **56,38%** |

|  |  |
| --- | --- |
| **Test con alpha = 0.9** | |
| **Categorie** | **Percentuale Correttezza Valutazione** |
| Adozione e Mainstreaming | 35,00% |
| Innovazione e Nuovi Progetti | 51,72% |
| News di Mercato, Analisi e Prezzi | 98,06% |
| Non rilevanti | 40,00% |
| Regolamentazione e Normative | 58,70% |
| Sicurezza, Hackeraggi e Truffe | 48,57% |
| **Media Percentuale Totale** | **55,34%** |

Immagine che contiene schermata, testo, diagramma, Rettangolo

Il contenuto generato dall'IA potrebbe non essere corretto.**Selezione e valutazione della configurazione migliore (alpha = 0.1)**

Valutiamo i risultati per ogni categoria:

1. **Adozione e Mainstreaming**:

* **Accuratezza migliorata:** 80.00% (prima era circa 30.00%)
* Molta meno confusione con **News di Mercato, Analisi e Prezzi** (solo 15.00% adesso, contro il precedente 65.00%).

1. **Innovazione e Nuovi Progetti**:

* **Accuratezza migliorata:** 86.21% (prima era 44.83%).
* Ridotta la confusione con **News di Mercato, Analisi e Prezzi** (6.90% invece di 51.72%).
* C'è un piccolo errore anche su **Sicurezza, Hackeraggi e Truffe (3.45%)**.

1. **News di Mercato, Analisi e Prezzi**:

* **Accuratezza leggermente ridotta:** 82.52% (prima era 98.06%).
* Alcuni articoli sono ora assegnati erroneamente ad altre categorie, ma l'accuratezza resta comunque molto alta.

1. **Non rilevanti**:

* **Accuratezza invariata:** 40.00%.
* Gli articoli sono spesso confusi con **Regolamentazione e Normative (20.00%)** e **News di Mercato, Analisi e Prezzi (20.00%)**.

1. **Regolamentazione e Normative**:

* **Accuratezza migliorata:** 86.96% (prima era 58.70%).
* Ridotta la confusione con **News di Mercato, Analisi e Prezzi**.

1. **Sicurezza, Hackeraggi e Truffe**:

* **Accuratezza migliorata:** 85.71% (prima era 48.57%).
* Minore confusione con altre categorie, ma rimane una piccola sovrapposizione con **News di Mercato, Analisi e Prezzi (11.43%)**.

Il ridotto valore di alpha ha effettivamente migliorato l'accuratezza complessiva del modello. Le categorie deboli (con meno articoli) sono state migliorate significativamente, come mentre la categoria “News di Mercato, Analisi e Prezzi” ha perso un po' di accuratezza, ma resta comunque molto precisa.

Accuratezza del modello: 0.83

Classification Report:

                                   precision    recall  f1-score   support

         Adozione e Mainstreaming       0.59      0.80      0.68        20

     Innovazione e Nuovi Progetti       0.74      0.86      0.79        29

News di Mercato, Analisi e Prezzi       0.91      0.83      0.87       103

                    Non rilevanti       1.00      0.40      0.57         5

     Regolamentazione e Normative       0.87      0.87      0.87        46

   Sicurezza, Hackeraggi e Truffe       0.83      0.86      0.85        35

                         accuracy                           0.83       238

                        macro avg       0.82      0.77      0.77       238

                     weighted avg       0.85      0.83      0.83       238

L'accuratezza complessiva è: 0.83 (83%)  
Questo è un miglioramento significativo rispetto al valore precedente (70%) con alpha=1.0.

**Generazione del Modello utilizzato per le classificazioni successive**

Salviamo il modello con alpha=0.1 che verrà utilizzato per classificare nuovi articoli:

# Salvare il modello e il vettorizzatore

with open('modello\_naive\_bayes.pkl', 'wb') as model\_file:

    pickle.dump(model, model\_file)

with open('vectorizer\_tfidf.pkl', 'wb') as vectorizer\_file:

    pickle.dump(vectorizer, vectorizer\_file)