**1. Introduzione**

L’evoluzione delle tecnologie digitali e, in particolare, dell’intelligenza artificiale, sta modificando in maniera profonda i processi di analisi e previsione nel settore finanziario. Tra le aree più dinamiche spicca il mercato delle criptovalute, che dalla nascita di Bitcoin (2009) ha conosciuto una crescita esponenziale, caratterizzata da forte volatilità e da un flusso informativo continuo e spesso contraddittorio proveniente dal web e dai social media.

In un simile contesto, la capacità di raccogliere, sintetizzare e classificare automaticamente le notizie diventa cruciale per interpretare le dinamiche di mercato e per supportare decisioni di investimento più consapevoli.

**1.1 Contesto e motivazioni**

Negli ultimi anni le criptovalute sono passate da fenomeno di nicchia a tema di rilevanza globale, attirando l’attenzione di investitori, media, istituzioni e opinione pubblica. La natura *decentralizzata* di questi strumenti, unita alla loro *elevata volatilità*, li rende un campo di studio complesso ma estremamente interessante.

Non sono soltanto le metriche economiche tradizionali (prezzi, volumi, capitalizzazione) a influenzare il settore: spesso l’andamento del mercato è determinato dalle informazioni che circolano sotto forma di notizie, post social, comunicati ufficiali o dichiarazioni di figure influenti (es. Elon Musk, Vitalik Buterin). Una singola news può innescare variazioni di prezzo improvvise e significative.

Di fronte a questa mole di contenuti testuali, le tecniche di *Natural Language Processing (NLP****)***e di *Machine Learning* offrono strumenti per estrarre segnali rilevanti, ridurre il rumore informativo e trasformare dati grezzi in indicatori sintetici. L’automazione di tali processi può quindi rappresentare un vantaggio strategico nell’analisi dei mercati digitali.

**1.2 Obiettivi della tesi**

Questa tesi si propone di sviluppare e analizzare un sistema automatizzato per l’analisi delle notizie relative al mercato delle criptovalute, con i seguenti obiettivi:

* **Raccolta delle notizie**: utilizzo di tecniche di *web scraping* per acquisire articoli dalla piattaforma *CryptoPanic*, aggregatore specializzato nel settore crypto.
* **Sintesi automatica**: generazione di riassunti brevi ed estesi tramite un modello linguistico di grandi dimensioni *(LLaMA 3.2)*,così da rendere i contenuti più accessibili.
* **Classificazione**: identificazione delle categorie informative attraverso un approccio ibrido, che combina clustering non supervisionato *(K-Means)* e classificazione supervisionata *(Naïve Bayes).*
* **Analisi di sentiment e peso informativo**: addestramento di regressori *(Ridge Regression)* per stimare la polarità emotiva e l’impatto potenziale delle notizie sul mercato.
* **Reportistica**: produzione di un report settimanale che integra i risultati precedenti e include il *Fear & Greed Index*, al fine di fornire una panoramica sintetica e aggiornata dell’andamento del mercato.

L’obiettivo finale è dimostrare come l’integrazione di tecniche di NLP, Machine Learning e sentiment analysis possa offrire un supporto concreto all’interpretazione del mercato crypto, aumentando la capacità di comprendere la sua complessità e dinamicità.

**1.3 Struttura del documento**

Il lavoro di tesi è organizzato in più capitoli, ciascuno dedicato a un aspetto specifico del sistema sviluppato:

* **Capitoli 1–2**: introduzione generale, contesto teorico e motivazioni, con particolare attenzione all’evoluzione del mercato delle criptovalute e alle problematiche legate all’analisi tradizionale delle notizie.
* **Capitoli 3–4**: presentazione dell’architettura complessiva del sistema, con una panoramica della pipeline progettata e descrizione delle tecnologie adottate.
* **Capitoli 5–8**: trattazione dettagliata delle varie fasi operative: raccolta degli articoli (data collection), generazione di riassunti tramite LLM, classificazione automatica delle notizie, analisi di sentiment e peso informativo, fino alla produzione del report settimanale.
* **Capitolo 9 e Conclusioni**: discussione critica dei risultati ottenuti, delle principali difficoltà incontrate e delle prospettive di miglioramento, seguita dalle considerazioni conclusive.

Questa struttura consente di passare progressivamente dal quadro generale del problema all’analisi tecnica della soluzione proposta, fino alla valutazione dei risultati e alla riflessione finale sugli sviluppi futuri.

**2. Il mercato delle criptovalute**

**2.1 Cosa è una criptovaluta?**

Una **criptovaluta** può essere definita come un *asset digitale decentralizzato*, la cui emissione e gestione avviene tramite algoritmi crittografici e meccanismi di consenso distribuiti, senza il controllo di un’autorità centrale. A differenza delle valute tradizionali (*fiat money*), non esiste un ente emittente (banca centrale o governo), ma la validità delle transazioni è garantita collettivamente dalla rete dei nodi.

L’infrastruttura tecnologica alla base delle criptovalute è la **blockchain**, un registro distribuito e immutabile che memorizza in sequenza cronologica tutte le transazioni. Ogni blocco è collegato crittograficamente al precedente, assicurando integrità e trasparenza dei dati.

Il primo esempio di criptovaluta è **Bitcoin**, introdotto nel 2009 da Satoshi Nakamoto, che ha rappresentato un punto di svolta nella storia della finanza digitale. Successivamente si sono sviluppate migliaia di altre valute digitali (*altcoin*), tra cui:

* **Ethereum**, che ha introdotto i contratti intelligenti (*smart contracts*);
* **Ripple (XRP)**, orientata ai pagamenti transfrontalieri;
* **Monero e Zcash**, focalizzate sulla privacy;
* **Cardano e Solana**, con soluzioni per scalabilità e interoperabilità.

Le criptovalute non sono quindi un fenomeno monolitico, ma un insieme eterogeneo di strumenti digitali che condividono l’uso della crittografia e della blockchain, differenziandosi per finalità, meccanismi di consenso (Proof-of-Work, Proof-of-Stake, ecc.) e livello di adozione nei mercati.

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

Il mercato delle criptovalute ha iniziato a crescere in maniera significativa a partire dal 2013, raggiungendo nel 2021 una capitalizzazione complessiva che ha superato i 2.000 miliardi di dollari. La sua evoluzione è stata caratterizzata da fasi di crescita esplosiva, spesso definite *bull market*, seguite da repentine contrazioni (*bear market*). Queste dinamiche sono frequentemente collegate a fattori esterni, quali nuove regolamentazioni, scandali legati a piattaforme di scambio (exchange) o innovazioni tecnologiche di rilievo.

Negli ultimi anni il settore sta mostrando segnali di consolidamento, con un interesse crescente da parte di investitori istituzionali, banche e governi. Tuttavia, rimane un mercato altamente volatile e speculativo, il cui andamento è influenzato non solo da variabili economiche oggettive, ma anche da fattori emotivi e psicologici, come la fiducia degli utenti o il sentiment espresso sui social media.

**2.3 Fonti informative del mondo crypto**

L’ecosistema delle criptovalute è alimentato quotidianamente da una mole considerevole di informazioni eterogenee. Le principali fonti comprendono:

* **Social media**: *X* è uno degli strumenti più influenti, utilizzato da analisti, influencer e sviluppatori per diffondere opinioni e breaking news.
* **Aggregatori di notizie**: piattaforme come *CryptoPanic* raccolgono in tempo reale articoli e post da centinaia di fonti, rendendoli facilmente consultabili in un unico feed.
* **Blog e forum**: siti come *Medium* e forum storici come *Bitcointalk* ospitano discussioni tecniche, spesso di nicchia ma di grande rilevanza per la comunità.
* **Siti di news specializzati**: portali come *CoinDesk, CoinTelegraph, Decrypt* forniscono articoli giornalistici, approfondimenti e analisi di settore.

Tra queste, *CryptoPanic* è stato selezionato come fonte per il sistema sviluppato. In qualità di aggregatore, infatti, permette di raccogliere in un unico flusso centralizzato notizie provenienti da un ampio ventaglio di testate.

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

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

* **Volatilità estrema**: i prezzi possono variare drasticamente in pochi minuti, compromettendo l’affidabilità delle previsioni basate su trend storici.
* **Influenza delle notizie**: eventi esterni e breaking news incidono in modo immediato e marcato, molto più che nei mercati tradizionali.
* **Rumore informativo**: l’enorme quantità di contenuti non sempre strutturati (tweet, post, articoli) rende complessa l’individuazione delle informazioni realmente utili.
* **Assenza di regolamentazione uniforme**: la mancanza di un quadro normativo coerente a livello globale genera incertezza e reazioni di mercato difficili da prevedere.

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

**3. Architettura del sistema sviluppato**

**3.1 Panoramica della pipeline**

Il sistema progettato ha come obiettivo la raccolta, l’elaborazione e la sintesi di notizie relative al mercato delle criptovalute, così da produrre report settimanali in grado di supportare l’analisi delle dinamiche di settore.

L’architettura è stata concepita come una *pipeline modulare*, nella quale ogni fase esegue un compito specifico e fornisce in output i dati necessari al modulo successivo. In questo modo il sistema risulta al tempo stesso scalabile e flessibile, con la possibilità di sostituire o aggiornare singoli componenti senza compromettere l’intero flusso.

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.Le macro-fasi che compongono la pipeline sono le seguenti:

1. **Data Collection** – acquisizione delle notizie tramite *web scraping* dalla piattaforma *CryptoPanic*.
2. **Preprocessing** – pulizia dei testi delle notizie raccolte.
3. **Sintesi automatica** – generazione di riassunti brevi ed estesi tramite il modello linguistico LLaMA 3.2.
4. **Classificazione** – identificazione delle categorie informative attraverso una fase iniziale di clustering non supervisionato e un successivo addestramento supervisionato basato su Naïve Bayes.
5. **Sentiment & Weight Analysis** – valutazione della polarità emotiva e attribuzione di un peso informativo ad ogni articolo tramite regressione.
6. **Report Generation** – produzione di un documento PDF settimanale che integra le notizie elaborate con indicatori come il *Fear & Greed Index*.

**3.2 Workflow operativo**

Il funzionamento complessivo del sistema può essere descritto come una sequenza di trasformazioni dei dati, in cui ciascun modulo della pipeline prende in input l’output del precedente:

* **Input (News):** le notizie vengono acquisite periodicamente dalla piattaforma *CryptoPanic* tramite web scraping. Ogni record contiene i metadati principali: titolo, data di pubblicazione, URL interno alla piattaforma e link alla fonte, quest’ultimo utilizzato per accedere al sito originale e recuperarne dall’ HTML il contenuto dell’ articolo.
* **Pulizia e normalizzazione:** il contenuto recuperato viene preprocessato per eliminare contenuti estranei (banner pubblicitari, avvisi sui cookie, ecc.). La funzione di pulizia combina parsing strutturale e metriche di similarità semantica, così da isolare soltanto i paragrafi pertinenti al corpo dell’articolo.
* **Sintesi automatica:** il testo pulito viene fornito al modello *LLaMA 3.2*, e per ogni articolo vengono generate due tipologie di sintesi:
  + *short summary* (1–2 frasi), utile per un’anteprima rapida;
  + *long summary* (1–2 paragrafi), pensato per una lettura più approfondita.
* **Classificazione:** ogni articolo viene assegnato a una categoria tematica (es. *News di Mercato - Analisi e Prezzi,* *Regolamentazione e Normative*, *Innovazione e Nuovi Progetti*, ecc.) tramite un classificatore supervisionato. Tale modello è stato addestrato a partire da categorie identificate in una fase preliminare di clustering non supervisionato.
* **Sentiment & Weight Analysis:** due regressori indipendenti stimano rispettivamente:
  + il **sentiment**, rappresentato da un valore compreso tra 0 e 1 (valori bassi → notizia negativa; valori alti → notizia positiva);
  + il **peso informativo**, anch’esso compreso tra 0 e 1, che riflette la rilevanza della notizia. Ad esempio, articoli su Bitcoin o su nuove normative tendono ad avere un peso elevato, mentre notizie marginali hanno un peso basso.
* **Output (Report PDF):** il sistema produce un documento PDF che integra:
  + Il riassunto lungo delle *tre notizie più rilevanti in assoluto*, selezionate in base al valore massimo di peso;
  + Per ciascuna *categoria*, il riassunto breve delle notizie più importanti;
  + il *Fear & Greed Index,* calcolato come media ponderata dei sentiment associati agli articoli e rappresentato graficamente.

Grazie a questo workflow, un’enorme mole di dati testuali non strutturati viene trasformata in un documento compatto e leggibile, riducendo drasticamente i tempi di analisi manuale e migliorando l’accessibilità delle informazioni.

**4. Tecnologie di riferimento**

L’analisi automatizzata del mercato delle criptovalute costituisce una sfida multidisciplinare che richiede l’integrazione di concetti e strumenti provenienti dall’intelligenza artificiale, dal machine learning e dal natural language processing.  
Questo capitolo offre una panoramica delle principali tecnologie utilizzate nello sviluppo del sistema descritto in questa tesi, fornendo il quadro teorico di riferimento necessario a comprendere le scelte progettuali e implementative illustrate nei capitoli successivi.

**4.1 Web Scarping**

Il **Web Scraping** è una tecnica che consente di automatizzare la navigazione e l’estrazione di contenuti dal web, simulando le azioni che un utente compirebbe manualmente ma attraverso comandi programmati.

Nel progetto presentato in questa tesi, il web scraping viene impiegato in due fasi principali:

* **Recupero dei metadati da CryptoPanic:** dalla piattaforma di aggregazione vengono acquisiti titolo, data di pubblicazione, URL interno e link alla fonte, per ogni articolo.
* **Recupero del contenuto dagli articoli originali:** a partire dal link alla fonte, il sistema accede al sito dell’articolo e ne estrae il corpo testuale, contenuto principalmente nel tag <body>.

In entrambi i casi, dunque, l’operazione di scraping sfrutta la struttura delle pagine web per trasformare dati non strutturati in input utilizzabili dalla pipeline di analisi.

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

Il ***Natural Language Processing (NLP)***comprende l’insieme di tecniche che permettono a un calcolatore di analizzare e manipolare il linguaggio naturale. Le applicazioni vanno dal pre-processing linguistico alla comprensione semantica profonda.

Le tecniche tradizionali utilizzate includono:

* **Tokenizzazione**: suddivisione del testo in unità elementari (parole o frasi).
* **Rimozione di stopword**: eliminazione di parole funzionali a bassa informatività (es. “il”, “di”, “con”).
* **Lemmatizzazione**: ricondurre le parole alla loro forma base.
* **TF-IDF (Term Frequency – Inverse Document Frequency)**: pesatura statistica per stimare l’importanza relativa di ciascun termine.

Accanto a queste metodologie classiche, negli ultimi anni si sono affermati approcci più avanzati basati su **architetture Transformer** e sulla generazione di **embedding**, cioè rappresentazioni vettoriali dense che catturano il significato semantico delle frasi. Queste tecniche consentono di cogliere relazioni complesse tra le parole e migliorano sensibilmente l’accuratezza nei compiti di **sintesi, classificazione e regressione**.

Nel sistema sviluppato, l’NLP rappresenta quindi un passaggio cruciale: è lo strato intermedio che trasforma il testo grezzo, recuperato tramite web scraping, in una forma coerente e strutturata, adatta per l’elaborazione da parte dei modelli di sintesi (LLM) e delle analisi quantitative successive.

**4.3 Large Language Models (LLM) per la sintesi automatica**

I Large Language Models (LLM) rappresentano lo stato dell’arte nel campo dell’elaborazione del linguaggio naturale. Basati sull’architettura **Transformer** (Vaswani et al., 2017), questi modelli sono addestrati su enormi corpora testuali e sono in grado di generare testi coerenti, contestuali e stilisticamente naturali. Il loro punto di forza risiede nel meccanismo di **self-attention**, che consente di catturare relazioni a lungo raggio tra le parole e di gestire in maniera efficace sequenze di testo anche molto lunghe.

Nel progetto descritto in questa tesi, i LLM svolgono un ruolo centrale nella fase di **sintesi automatica delle notizie**. Dopo che i testi grezzi sono stati recuperati tramite web scraping e sottoposti a preprocessing, il contenuto pulito viene fornito al modello per la generazione di due tipologie di riassunti:

* **short summary**: una sintesi di 1–2 frasi, pensata per fornire una panoramica immediata della notizia;
* **long summary**: una sintesi più articolata (1–2 paragrafi), utile per una comprensione più dettagliata.

Per questa pipeline è stato scelto **LLaMA 3.2 (3B)**, un modello open-source sviluppato da Meta, selezionato per diversi motivi:

* **Qualità linguistica**: capacità di produrre testi fluidi e coerenti, adatti anche a un contesto formale.
* **Efficienza computazionale**: grazie alle sue dimensioni contenute (3 miliardi di parametri), può essere eseguito anche in ambienti con risorse limitate, come *Google Colab*.
* **Supporto multilingua**: addestramento su corpora multilingue, con buone prestazioni anche in italiano.
* **Licenza open-source**: consente piena personalizzazione e fine-tuning, rendendo il modello adattabile a esigenze specifiche.

Il modello è stato impiegato in **modalità prompt-based**, ovvero fornendo al modello sia il corpo testuale dell’articolo sia istruzioni esplicite sul tipo di sintesi desiderata (breve o estesa). Questa modalità ha permesso di sfruttare le capacità generative del modello senza dover addestrare un sistema da zero, riducendo tempi e costi computazionali.

**4.4 Tecniche di classificazione supervisionata e non supervisionata**

La **classificazione non supervisionata** è un approccio di *machine learning* che lavora su dati non etichettati. L’algoritmo analizza le caratteristiche intrinseche dei dati e cerca di individuare autonomamente raggruppamenti naturali (cluster), senza che vi sia una conoscenza preliminare delle categorie. In questo modo è possibile scoprire strutture latenti e relazioni nascoste all’interno del dataset.

La **classificazione supervisionata**, al contrario, si basa su un dataset già etichettato, in cui a ogni esempio di input è associata la categoria corretta (target). Il modello viene addestrato a riconoscere i pattern che caratterizzano ciascuna classe, così da poter assegnare l’etichetta più probabile a nuove osservazioni non viste in fase di training.

In questa tesi è stato adottato un approccio ibrido a due fasi che combina entrambi i metodi:

1. *Classificazione non supervisionata (clustering)*
   * È stato utilizzato l’algoritmo K-Means per analizzare il corpus di articoli e individuare automaticamente insiemi di notizie semanticamente affini.
   * Questa fase esplorativa ha permesso di derivare un primo insieme di cluster, poi etichettati manualmente in categorie, come ad esempio *“Adozione e regolamentazioni”*, *“Sicurezza e hackeraggi”*, *“Mercati e investimenti”*.
2. *Classificazione supervisionata*
   * Il dataset etichettato è stato quindi utilizzato per addestrate un classificatore supervisionato (Naïve Bayes).
   * Il modello supervisionato addestrato viene ora utilizzato per classificare automaticamente nuove notizie in arrivo, sfruttando la conoscenza acquisita nella fase di training.

Questo approccio ibrido ha permesso di combinare la scoperta automatica delle categorie (fase esplorativa) con l’accuratezza di un classificatore supervisionato, migliorando la robustezza del sistema

**4.5 Modelli di regressione per sentiment e peso**

La regressione è una tecnica di *machine learning supervisionato* che, a differenza della classificazione, non assegna un’etichetta discreta a un input, maproduce un valore continuo. Questo la rende particolarmente adatta quando si vogliono stimare grandezze numeriche, come un punteggio o una misura di intensità.

Nel sistema sviluppato, i modelli di regressione sono stati utilizzati per associare a ciascun articolo due valori quantitativi fondamentali:

* **Sentiment**: esprime la polarità emotiva della notizia, con valori vicini a 0 che indicano contenuti negativi e valori prossimi a 1 che indicano contenuti positivi.
* **Peso**: riflette la rilevanza della notizia per il mercato crypto. Notizie riguardanti Bitcoin o nuove regolamentazioni tendono a ricevere punteggi elevati, mentre contenuti marginali assumono valori bassi.

Il processo adottato prevede due passaggi principali:

1. **Estrazione di embeddings**: ogni articolo viene trasformato in una rappresentazione vettoriale densa tramite un modello linguistico pre-addestrato (*UmBERTo*, variante italiana di BERT). Questo step consente di catturare il contenuto semantico dei testi in forma numerica.
2. **Addestramento dei regressori:** sugli embeddings ottenuti vengono addestrati due modelli di regressione distinti (uno per il sentiment e uno per il peso), basati su Ridge Regression con validazione incrociata (RidgeCV). Questa scelta è stata motivata dalla robustezza del modello rispetto all’overfitting e dalla sua capacità di gestire in maniera efficace feature ad alta dimensionalità.

In questo modo, da un testo non strutturato si ottengono due misure numeriche interpretabili, che arricchiscono la pipeline con informazioni quantitative esono alla base della generazione del report settimanale.

**4.5 Reti neurali**

Le **reti neurali artificiali (Artificial Neural Networks, ANN)** sono modelli computazionali ispirati al funzionamento del cervello umano, in cui l’informazione viene elaborata da un insieme di unità elementari chiamate **neuroni artificiali**. Ogni neurone riceve in ingresso un vettore di valori numerici, li combina linearmente tramite **pesi**, aggiunge un **bias** e applica una **funzione di attivazione** non lineare.

L’insieme dei neuroni è organizzato in **strati**:

* **Strato di input**: riceve i dati grezzi o pre-processati.
* **Strati nascosti**: composti da molteplici neuroni che trasformano progressivamente l’informazione.
* **Strato di output**: produce la previsione finale (classe, valore numerico, probabilità, ecc.).

Matematicamente, il funzionamento di un singolo neurone può essere espresso come:

dove:

* sono gli input
* ​ i pesi che collegano l’input al neurone
* il termine di bias,
* la funzione di attivazione,
* l’uscita del neurone.

**Funzioni di attivazione**

Le funzioni di attivazione introducono la **non linearità**, necessaria per permettere alla rete di approssimare funzioni complesse. Le più utilizzate sono:

* **ReLU (Rectified Linear Unit)**:

Semplice, efficiente e molto usata nei modelli moderni.

* **Sigmoide**:

Trasforma i valori in un intervallo (0,1), utile per output probabilistici.

* **Tangente iperbolica (tanh)**:

Mappa in (-1,1) ed è centrata rispetto allo zero.

* **Softmax**: usata nello strato di output per classificazione multiclasse, normalizza i valori in probabilità che sommano a 1.

**Algoritmo di apprendimento: Backpropagation**

Il processo di apprendimento in una rete neurale avviene modificando progressivamente i pesi ​ e i bias ​per ridurre l’errore di previsione.  
Il meccanismo principale è l’**algoritmo di retropropagazione (backpropagation)**, che combina:

1. **Forward pass**: i dati attraversano la rete e si calcola l’output.
2. **Loss function**: viene calcolato l’errore confrontando l’output con il target (es. MSE per regressione, cross-entropy per classificazione).
3. **Backward pass**: tramite la **regola della catena** del calcolo differenziale, si propaga all’indietro il gradiente della loss rispetto a ciascun parametro.
4. **Aggiornamento dei pesi**: i pesi vengono corretti con una variante della **discesa del gradiente**:

dove è il **learning rate** e è il gradiente della funzione di perdita rispetto al peso.

Questo processo si ripete per molte **epoche**, fino a che la rete converge a una soluzione che approssima la funzione desiderata.

**Tipologie di reti**

A seconda del compito, le reti neurali assumono diverse architetture:

* **Feed-forward (MLP, Multi-Layer Perceptron)**: struttura più semplice, utilizzata per compiti di classificazione e regressione.
* **Reti convoluzionali (CNN)**: specializzate nell’elaborazione di immagini e segnali, sfruttano la convoluzione per estrarre pattern locali.
* **Reti ricorrenti (RNN, LSTM, GRU)**: adatte a sequenze (testo, audio, serie temporali), grazie alla memoria interna.
* **Transformer**: architetture basate su self-attention, oggi standard per il NLP (GPT, BERT, LLaMA).

**Reti neurali nel progetto**

Nel progetto descritto in questa tesi, le reti neurali non sono state implementate “from scratch”, ma sfruttate tramite modelli pre-addestrati come:

* **UmBERTo** (variante italiana di BERT), utilizzato per estrarre **embeddings** testuali da articoli e riassunti.
* **LLaMA 3.2**, modello transformer di grandi dimensioni, impiegato per la **sintesi automatica** delle notizie.

In entrambi i casi, i principi descritti (feed-forward multilayer, funzioni di attivazione non lineari, backpropagation) costituiscono la base matematica su cui poggiano i modelli moderni.

**5. Raccolta delle notizie (Data Collection)**

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.

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

La piattaforma CryptoPanic 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;

**5.2 Struttura del database**

La gestione dei dati è affidata a un database SQLite.

La scelta di SQLite è motivata da:

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

Il database è composto da due tabelle:Immagine che contiene testo, linea, numero, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

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).*

**5.3 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 pricnipale della notizia;
* Data e 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).

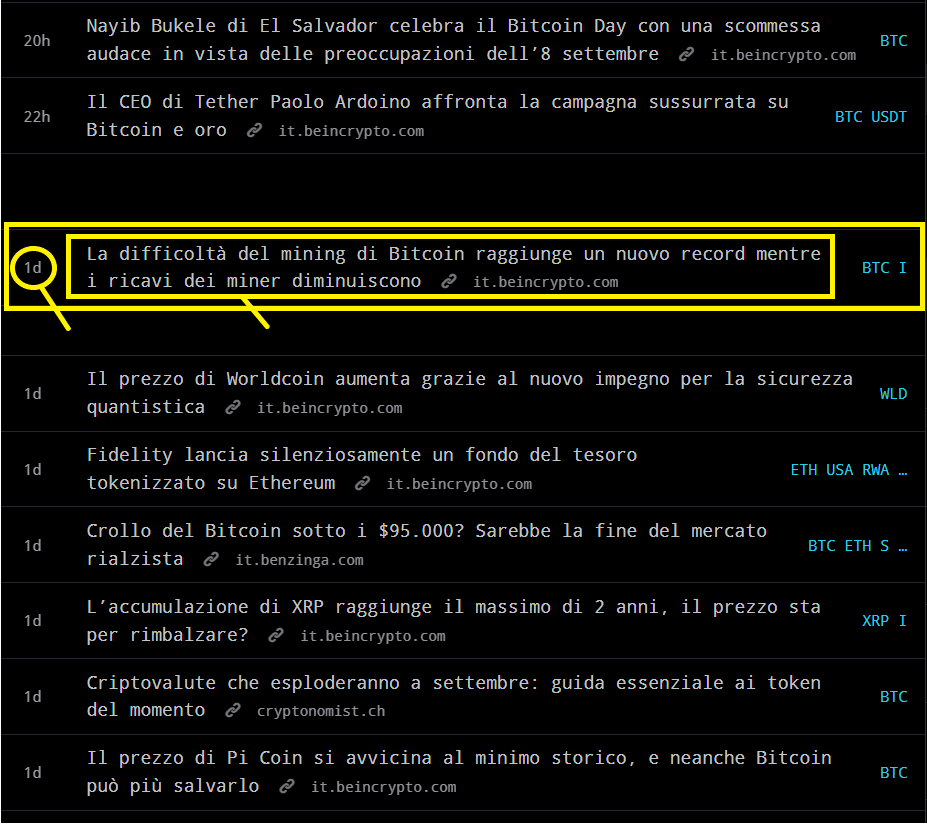
**Struttura della pagina di CryptoPanic**

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 si nota 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.



Area cliccabile contenente *url\_cryptopanic*

Data

Titolo

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…"

       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…"

       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).
* **Titolo**: contenuto nel secondo <span> annidato all’interno del blocco con classe title-text.

**Recupero metadati e salvataggio nel database**

Utilizzando web scraping e sfruttando la struttura di CryptoPanic appena vista, l’algoritmo sviluppato recupera i metadati che vengono infine salvati nelle tabelle database SQLite, costituendo la base di partenza per tutte le fasi successive della pipeline.

Immagine che contiene testo, schermata, numero, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.url\_cryptopanic, url\_articolo e data salvati nella tabella meta\_articolo, mentre titolo è salavato in articoli.

**5.4 Recupero contenuto articolo e preprocessing**

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***).

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.

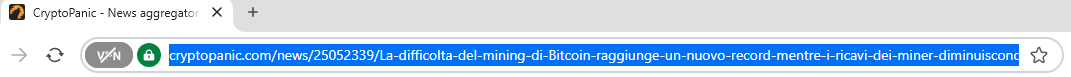


Immagine che contiene testo, schermata, software, Carattere

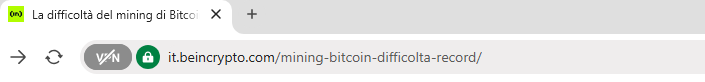
Il contenuto generato dall'IA potrebbe non essere corretto.

Immagine che contiene testo, schermata, 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 ogni nuova pagina articolo si ottengono:



* **url\_articolo**: leggendo la barra degli indirizzi del browser.
* Il **contentuo HTML completo della notizia**, che viene acquisito e successivamente pulito per rimuovere elementi superflui (banner, menu di navigazion ecc..)

**Preporcessing: Estrazione del solo testo rilevante dall’ HTML**

Nella fase precedente è stato acquisito il contenuto del <body> della pagina dell’ articolo. Tuttavia, tale contenuto presenta spesso del rumore informativo (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 (rumore)-->

  <div class="cookie-banner">

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

  </div>

  <!-- Titolo principale (rilevante) -->

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

  <!-- Primo paragrafo (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 (rumore) -->

  <div class="ad-banner">

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

  </div>

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

  <p>L’adeguamento ha segnato un aumento del 4% rispetto ai 129,6 trilioni,secondo i dati di Mempool.</p>

  <!-- Intestazione sezione (rilevante) -->

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

  <!-- Paragrafo ulteriore (rilevannte) -->

  <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** 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 ciò che conterrà i tag testuali dell’ articolo.
2. **Sequenza di elementi testuali.** Vengono presi in considerazione solo i tag più informativi: <h1>, <h2>, <p>.
3. **Ancoraggio tematico.** Il primo elemento <h1> è il titolo dell’ articolo e viene utilzzato come ancora semantica da confrontare con le altre sezioni di testo dell’ articolo per stabilire se le sezioni sono inerenti all’ articolo o no.
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.

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, pulito dal rumore, costituisce così la base per le fasi successive di riassunto automatico, classificazione e sentiment analysis.

Considerimao le seguenti frasi per gli step successivi:

“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.”

“Sponsored: Scopri la nuova piattaforma di trading con bonus di benvenuto”

**B) Calcolo degli embeddings**

L’obiettivo è trasformare ogni unità testuale (titolo, intestazione, paragrafo) in un **vettore denso** di dimensione fissa che ne catturi il **significato**. Nel sistema viene usato il modello **Sentence-BERT** multilingue (MiniLM), che produce un embedding per frase/paragrafo.

***Che cos’è un embedding (e perché è “denso”)?***

Un *embedding* è un vettore di numeri reali (es: d=384) che rappresenta il contenuto semantico di un testo.

*“Denso”* significa che tutte le d componenti sono (in genere) non nulle; è l’opposto dei vettori *sparsi* tipici di TF-IDF (dimensione = vocabolario, quasi tutti zeri). Con questo approccio frasi con significato simile finiscono vicine nello spazio vettoriale (alta similarità del coseno) anche se usano parole diverse (sinonimi, parafrasi).

**1. Tokenizzazione**

La tokenizzazione è il processo che scompone il testo in parole (token).

*Esempio:*

["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","."]

Si ha quindi in output una sequenza di di token:

2. **Lookup table degli embedding** (token, posizione, segmento)

La **Lookup table** è una matrice che mappa ogni token a un vettore:

È una “tabella” di parametri del modello.

Esempio (numeri fittizi, in realtà sono 384 dimensioni):

* “Bitcoin” → [0.12, −0.45, 0.33, ...]
* “mining” → [0.88, 0.05, −0.41, ...]
* “difficoltà” → [0.34, 0.71, −0.22, ...]

Quindi la frase F1 diventa una lista di vettori:

**Positional embeddings.** Poiché i Transformer non hanno ricorrenza, si aggiunge per ogni posizione i un vettore per codificare l’ordine:

Senza posizione, il modello non distinguerebbe “A vende B” da “B vende A”

Esempio (sempre fittizio in 3 dimensioni per semplicità):

* “Bitcoin” e=[0.12,−0.45,0.33], posizione 6 → p=[0.01,0.07,−0.02] ⇒ x=[0.13,−0.38,0.31]
* “mining” e=[0.88,0.05,−0.41], posizione 4 → p=[−0.02,0.10,0.05] ⇒ x=[0.86,0.15,−0.36]

Da questa fase si ha in output la sequenza

 F1 →

 F2 →

 F3 →

Questi X sono la “rappresentazione di partenza” che entrerà nel **Trasformatore** per la fase di **self-attention**

**3. Contestualizzazione con il Transformer (self-attention)**

Una volta ottenuti i vettori iniziali dei token (embedding + posizione), il modello li elabora con una serie di blocchi Transformer encoder, il cui meccanismo fondamentale è la self-attention.

L’idea è che ogni parola possa “guardare” tutte le altre nella frase e pesare la loro importanza relativa per costruire la propria rappresentazione. Questo permette al modello di distinguere, ad esempio, il significato di *bank* in *river bank* rispetto a *central bank*.

La formula principale della self-attention è:

dove:

* (query) è un vettore che rappresenta la domanda “quali altri token sono importanti per il mio significato?”
* (key) è la “descrizione” di ciascun token, che permette agli altri di capire quanto sia rilevante
* (value) è la matrice dell’ embedding del token che contiene le informazioni.
* il prodotto calcola quanto un token è “rilevante” rispetto a un altro,
* la softmax normalizza questi pesi di attenzione,
* moltiplicando per si ottiene una nuova rappresentazione di ciascun token come media pesata delle altre.

NOTA: Q, K e V non sono copie identiche dell’embedding originale, ma derivano da tre diverse matrici di pesi (W) apprese dal modello durante l’addestramento:

Questa proiezione in tre spazi distinti permette al modello di specializzare i vettori: uno per fare la domanda (Q), uno per fornire la chiave di confronto (K), e uno per condividere contenuto (V).

Alla fine, per ogni token si ottiene un vettore contestualizzato , che non rappresenta più solo la parola isolata ma la parola nel suo contesto. È proprio questa contestualizzazione che consente agli embedding finali di esprimere la semantica della frase.

**Esempio semplificato:**

**1. Calcoliamo Q, V, K per i token “Bitcoin” e “raggiunge”**

Supponimao che gli embedding iniziali abbiano dimensione d = 3 e che scegliamo = 2.

Quindi le matrici di pesi saranno (pesi ipotizzati):

E il token “Bitcoin” ha embedding iniziale (dopo lookup e posizioni):

Allora:

Mentre il token “raggiunge” ha embedding iniziale

Allora:

**2. Calcoliamo la compatibilità**

Prendiamo la Query di “raggiunge”:

La confrontiamo con le key di “Bitcoin” e “raggiunge”:

* compatibilità con “Bitcoin”:
* compatibilità con “raggiunge”:

**3. Normalizziamo con softmax**

:

* Bitcoin: 1.2741 / 1.41 = 0.903
* Raggiunge: 2.1234 / 1.41 = 1.507

Applichiamo softmax:

**4. Costruiamo la nuova rappresentazione**

La nuova rappresentazione di “raggiunge” sarà una combinazione pesata di:

Il nuovo vettore di **“raggiunge”** non è più solo un verbo generico, ma porta dentro parte del significato di “Bitcoin” (peso 0.354) e parte di sé stesso (peso 0.646). Se avessimo incluso anche “record”, molto probabilmente avrebbe preso **ancora più peso**, perché “raggiungere un record” è un legame semantico forte.

In parallelo, anche “Bitcoin” e gli altri token aggiornano la propria rappresentazione. Alla fine otteniamo una sequenza [t1​,…,tm​] di vettori contestualizzati, in cui ciascun token incorpora il contesto degli altri. Questo è il passo che conferisce agli embedding finali la loro natura semantica.

**C) Pooling di frase (mean pooling)**

Una frase o un paragrafo, dopo essere passati attraverso il Transformer, è rappresentata non da un singolo vettore ma da una sequenza di vettori di token   
Ogni descrive un token (parola o sottoparola) nel suo contesto, ma per confrontare due frasi serve un’unica rappresentazione globale.

Per ottenere questo, Sentence-BERT applica un’operazione di **pooling**, cioè una funzione che condensa più vettori in uno solo.

Il modello utilizza il **mean pooling**: in pratica si fa la media dei vettori di tutti i token validi (escludendo quelli di padding usati solo per uniformare la lunghezza).

La formula è:

dove:

* è una maschera che vale 1 se il token è reale, 0 se è padding,
* ​ è il vettore contestualizzato del token i,
* è il vettore finale della frase.

In altre parole: si calcola la **media aritmetica** dei vettori dei token effettivi.

Il risultato è un singolo embedding con d=384 nel modello MiniLM utilizzato nel progetto.

**Esempio:**

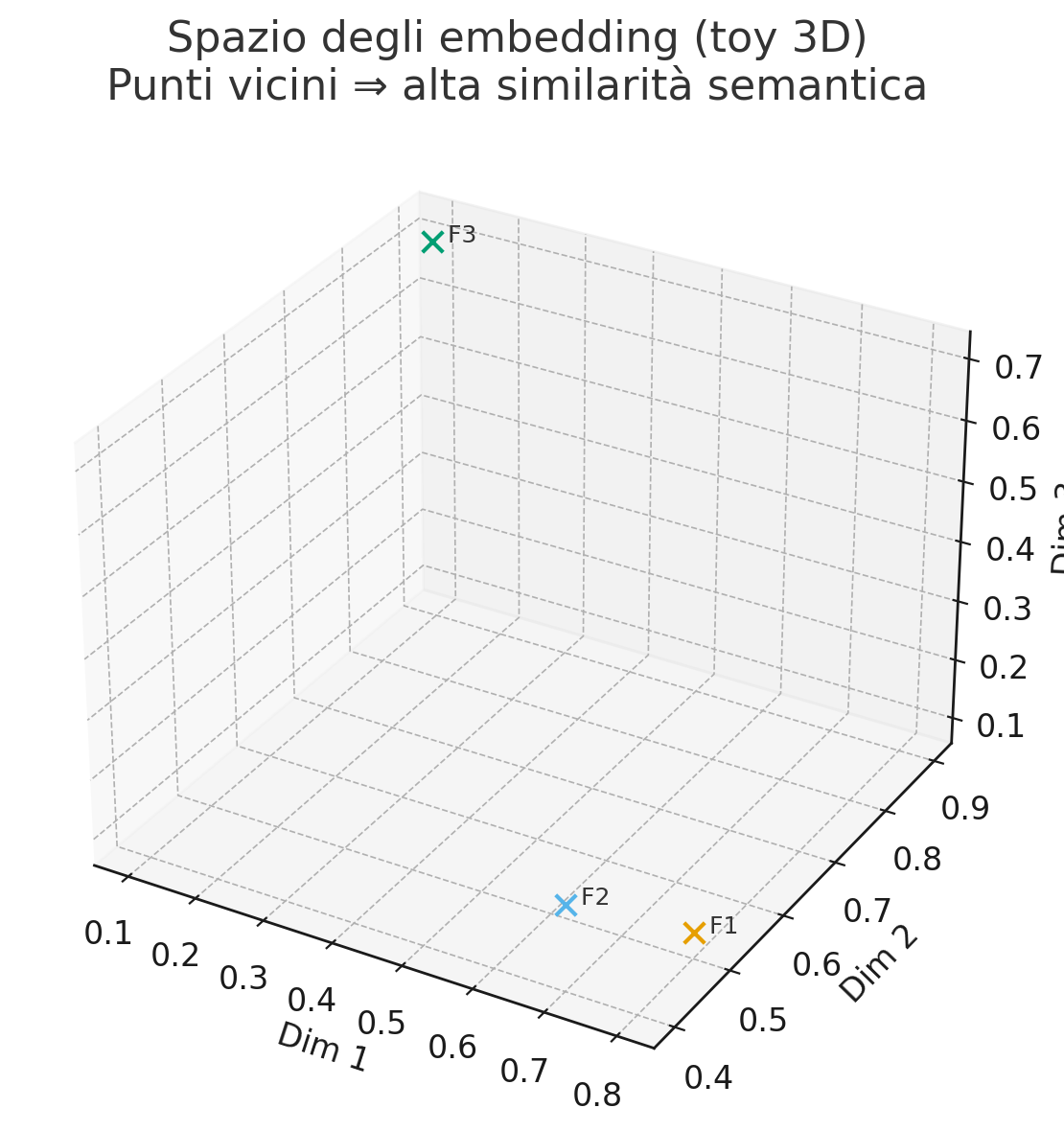
Applicando il mean pooling ai vettori dei due token ottenuti precedentemente tb e tr abbiamo:

Questo vettore s è l’embedding della frase *“Bitcoin raggiunge”* nello spazio R2 ( in reltà sarebbe R384 ma lo abbiamo ridotto per semplicità).

Questo è l’**embedding di frase**, cioè il vettore che rappresenta l’intero paragrafo o titolo e che può essere confrontato con altri tramite la similarità coseno.

Quindi dopo l’embedding è possibile rappresentare le frasi in uno spazio vettoriale conforntabile, cioè è possibile capire se le frasi sono simili semnatcimanete vedendo la distanza fra i punti nello spazio.

Esempio delle tre frasi rappresentate nello spazio 3D dopo embeddings:



**D) Similarità semantica e soglie decisionali**

Dopo lo step preceedente ogni unità testuale (<h1>, <h2>, <p>) viene è stata trasformata in un vettore denso, cioè vettori che catturano il significato delle frasi oltre la semplice corrispondenza lessicale. Per stabilire quali frasi mantenere e quali scartare si confronta h1 con le altre tramite la similarità del coseno.

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.

**Uso nella pipeline.**

* Accetto (h2 + primo paragrafo) se la similarità col titolo <h1> ≥ soglia.
* Accetto un nuovo <p> se la similarità col **paragrafo precedente accettato** ≥ soglia.

**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.

**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.

**Recupero articolo e salvataggio nel database**

Il procedimento appena descritto di recupero del contenuto HTML e pulizia del testo tramite Embeddings e Similarità del coseno viene iterato 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.*

Al termine dell’ iterazione il database si presenta nel seguente modo:

Tabella “meta\_articoli” con url\_articolo valorizzato:

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

Tabella “articoli” con “contenuto\_articolo” valorizzato.

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

**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**

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 recuperato e pulito nelle fasi precedenti.
* *Summary*: contenente il riassunto che ci si aspetta di ottenere in output (target).

Esempio del dataset:

|  |  |
| --- | --- |
| User | Summary |
| "Genera un riassunto conciso di 30-40 parole per il seguente articolo, offrendo una panoramica chiara e immediata della notizia.  Se l'articolo ha scarso contenuto informativo, restituisci invece la frase: ""L'articolo non contiene contenuti rilevanti."".  Articolo:  Il disegno di legge sulla riserva di Bitcoin della Carolina del Sud prevede di investire il 10% dei fondi statali… | La Carolina del Sud ha introdotto un disegno di legge per creare una Riserva di Bitcoin, consentendo allo stato di investire il 10% dei fondi statali in Bitcoin. Il disegno è parte di un trend crescente a livello nazionale. Tuttavia, potrebbe incontrare resistenza da parte dei conservatori fiscali. |
| "Genera un riassunto dettagliato di 150-200 parole del seguente articolo. Il riassunto deve includere i punti chiave, il contesto e le implicazioni della notizia.  Se l'articolo ha scarso contenuto informativo, restituisci invece la frase: ""L'articolo non contiene contenuti rilevanti."".  Articolo:  Il disegno di legge sulla riserva di Bitcoin della Carolina del Sud prevede di investire il 10% dei fondi statali… | Il disegno di legge della Carolina del Sud prevede di investire il 10% dei fondi statali in Bitcoin, un'idea che ha suscitato dibattiti tra i conservatori fiscali. Attualmente, quasi la metà degli stati USA ha un disegno di legge simile. Il rappresentante Jordan Pace ha introdotto la proposta per proteggere i dollari degli investitori dall’inflazione. Tuttavia, proposte simili in stati come Montana e Wyoming hanno fallito a causa di preoccupazioni sulle riserve pubbliche. Alcuni stati, come il Texas, hanno adottato approcci più flessibili, consentendo agli amministratori finanziari di investire in Bitcoin senza obbligo. La Carolina del Sud potrebbe seguire un modello simile, offrendo maggiore flessibilità nel gestione delle riserve statali. Se approvato, il disegno di legge potrebbe aprire la strada a un maggior numero di stati a seguire un approccio simile. |

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

**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).

[Mettere esempio di riassunto generato prima dell’ addestramento e riassunto generato dopo l’addestramento]

**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.

**7.1 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()

Immagine che contiene testo, linea, schermata, Diagramma

Il contenuto generato dall'IA potrebbe non essere corretto.**Valutazione dei risultati – Curva del Gomito**

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)