Sentiment Analysis: creazione di un database per criptovalute

Ermellino Andrea, Kolyszko Matteo, Serino Antonio

Dipartimento di Informatica Sistemistica e Comunicazione, DISCO , Università degli Studi di Milano Bicocca
Piazza dell'Ateneo Nuovo, 1, 20126 Milano MI

Abstract: La *Sentiment Analysis* ricopre un ruolo fondamentale nel comprendere, ed eventualmente provare a prevedere, le dinamiche del mercato delle criptovalute. Data la natura speculativa delle criptovalute, il sentiment della community, del numero sempre crescente di investitori in questo campo, può influenzare enormemente il trend di mercato. In particolare, una delle piattaforme principali su cui la community di investori esprime le proprie opinioni è Twitter che permette di esprimere brevi pensieri e impressioni sulle repentine variazioni di prezzo di questi asset.
In questo documento verrà descritto il processo di creazione di un database contenente dati provenienti da diverse fonti, con l'obiettivo finale di costruire un dataset su cui successivamente effettuare un'analisi del sentiment.

Key words: Data Management, MongoDB, Crypto

Indice

1 Introduzione

Nella prima parte del documento verrà descritto il processo di acquisizione dei dati, verranno descritti i vari metodi con cui questi sono stati ottenuti e le diverse fonti da cui sono stati estrapolati. Una volta ottenuti e puliti, sui dati è stata effettuata un'analisi esplorativa così da scoprire eventuali correlazioni e capire come i valori dei diversi attributi sono distribuiti. In seguito alla memorizzazione nel DBMS, sono state valutate diverse dimensioni di qualità per ottenere una panoramica generale dell'integrazione effettuata.

2 Data Acquisition

I dati per raggiungere l'obiettivo prefissato, sono stati reperiti da tre fonti diverse. Di seguito vengono elencate le fonti, le modalità e gli attributi presi per comporre il nostro dataset.

2.1 Web Scraping

La prima fonte da cui sono stati presi i dati è `https://coinmarketcap.com`. Attraverso un tool di web scraping chiamato WebHarvy, sono stati presi i seguenti attributi:

- *img*: Indica l'immagine della criptovaluta;

- *name*: Indica il nome della criptovaluta;
- *abbr*: Indica l'abbreviazione della criptovaluta;
- *price*: Indica il prezzo della criptovaluta;
- *7d%*: Indica la variazione percentuale settimanale del valore della criptovaluta;
- *sign*: Indica se la variazione è positiva o negativa;
- *market_cap*: Indica la capitalizzazione di mercato della criptovaluta;
- *volume(7d)*: Indica il volume di scambio settimanale della criptovaluta;
- *total_supply*: Indica la quantità totale disponibile della criptovaluta;
- *dominance*: Indica la percentuale con cui la criptovaluta domina il mercato.

2.2 API

Per la seconda e terza fonte sono state utilizzate le API. Di seguito vengono descritte le librerie, le API e il contenuto preso per comporre il dataset.

- La seconda fonte dati è Twitter con la sua API. Attraverso la libreria Python chiamata Tweepy, sono stati presi tutti i tweet che all'interno del loro testo contenessero il nome o l'abbreviazione della criptovaluta (per un totale di circa 1 milione di tweet, 1061494 per la precisione);
- Per la terza ed ultima fonte dati è stata utilizzata l'API di CoinGecko. Attraverso la libreria Python chiamata pycoingecko, sono stati presi gli OHLC Data degli ultimi 7 giorni, aggiornati ogni 4 ore.

3 Exploratory Data Analysis

Una volta conclusa la fase di data acquisition e dopo la fase di cleaning dei dati (descritta con maggior dettaglio nella sezione 5.1 del documento), è possibile passare alla Exploratory Data Analysis.

Inoltre la colonna 'sign' era stata presa in quanto nel sito web il segno della variazione percentuale veniva rappresentato con il colore, rosso se negativo, verde se positivo. Al fine di ottenere una colonna che rappresentasse il dominio corretto è stato preso il testo presente nel tag html della colonna ***sign***. Nel caso in cui il testo contenesse la parola "up" al rispettivo record della colonna ***7d(%)*** verrà inserito il segno "+". Nel caso in cui il testo contenesse la parola "down" il segno sarà "-". Infine le colonne ***price***, ***7d(%)***, ***market_cap($)***, ***volume_7d($)***, ***total_supply*** sono state convertite in numeric.
Di seguito viene presentato il dataset ottenuto:

	img	name	ticker	price	7d(%)	market_cap($)	volume_7d($)	total_supply	dominance(%)
0	https://s2.coinmarketcap.com/static/img/coins/...	Bitcoin	BTC	38113.66	4.05	722090979732	169996819809	18945725	41.0662
1	https://s2.coinmarketcap.com/static/img/coins/...	Ethereum	ETH	2727.98	11.61	325718294832	89815589895	119399127.94	18.524
2	https://s2.coinmarketcap.com/static/img/coins/...	Tether	USDT	1.00	0.01	78014264209	293486741428	80074893058.66	4.4464
3	https://s2.coinmarketcap.com/static/img/coins/...	BNB	BNB	379.83	1.44	62715950446	10537360532	165116760.89	3.5745
4	https://s2.coinmarketcap.com/static/img/coins/...	USD Coin	USDC	0.9998	-0.07	49990699582	22245547728	49998886896.8	2.843

Figura 1: Head del dataset ottenuto

Una volta conclusa la fase di prepocessing è possibile partire con l'exploratory data analysis. Per conoscere da quante colonne e da quanti record è composto il dataset applichiamo la seguente funzione:

```
crypto.shape
```

Per conoscere quanti possibili duplicati contiene il dataset utilizziamo la seguente funzione:

```
df.nunique(axis = 0)
```

Infine vogliamo avere un'idea generale dei valori che compongono il dataset. Attraverso la seguente funzione possiamo conoscere per ogni colonna di tipo *numeric*:

- Media;
- Deviazione Standard;
- Il valore minimo;
- Il valore massimo;
- I tre quartili.

```
df.describe().apply(lambda s: s.apply(lambda x:
    format (x, 'f')))
```

Il prossimo passo è quello di scoprire possibili relazioni tra le variabili. La seguente funzione genererà il seguente grafico:

```
corr = df.corr()
sns.heatmap(corr, xticklabels = corr.columns,
    yticklabels = corr.columns, annot = True,
    cmap = sns.diverging_palette(220, 20,
    as_cmap = True))
```

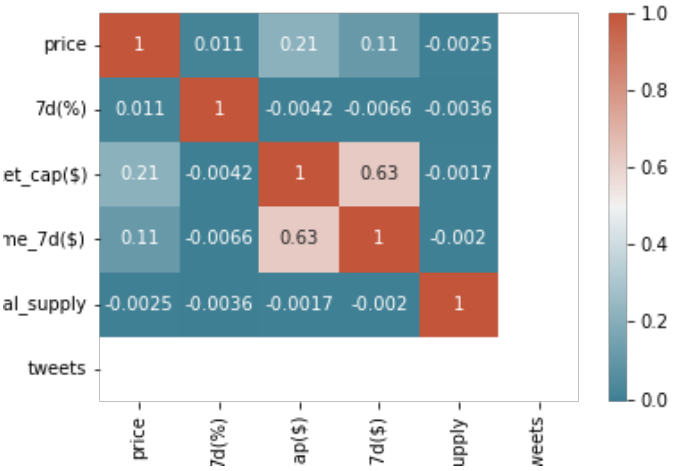

Figura 2: Matrice di correlazione

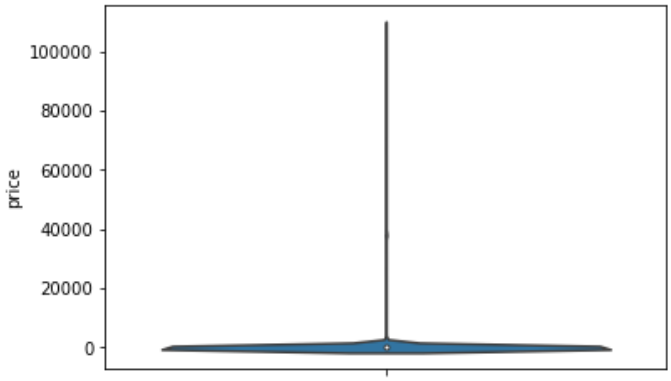

Figura 3: Distribuzione dei prezzi

Dalla visualizzazione è possibile notare come quasi la totalità delle criptovalute si concentri sul un valore di poco superiore allo zero, con molti outliers che toccano anche i 100mila$.

Dalla matrice è possibile vedere una leggera correlazione tra la colonna ***market_cap($)*** e la colonna ***volume_7d($)***.

Si vuole ora vedere la distribuzione dei valori di determinate colonne. Attraverso la seguenti funzioni, verranno generati i violinplot che ci aiuteranno a capire meglio come è distribuito il prezzo, la capitalizzazione di marcato, la quantità di moneta disponibile, la variazione percentuale settimanale e infine il volume di moneta scambiata in una settimana delle varie criptovalute:

```
sns.violinplot( y ="price",
            data = df)
sns.violinplot( y = 'market_cap($)',
            data = df)
sns.violinplot( y = 'total_supply',
            data = df)
sns.violinplot( y = '7d(%)',
            data = df)
sns.violinplot( y = 'volume_7d($)',
            data = df)
sns.violinplot( y = 'volume_7d($)',
            data = df)
```

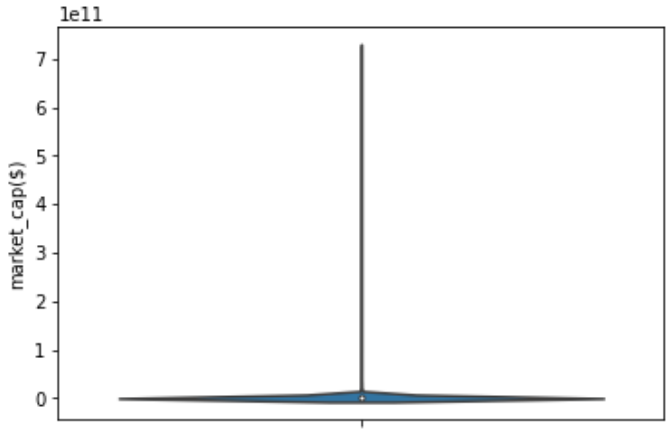

Figura 4: Distribuzione della capitalizzazione di mercato

Dalla visualizzazione è possibile notare come quasi la totalità delle criptovalute si concentri nel range che va da 0 a 1 in una scala che però deve essere moltiplicata per 10^{11}, con la presenza, anche qui, di molti outliers.

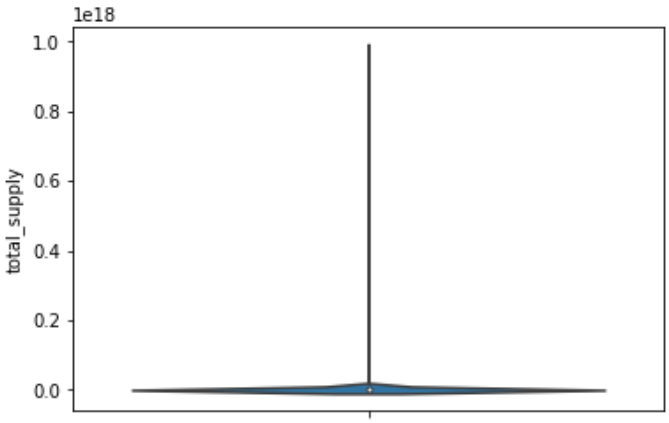

Figura 5: Distribuzione della moneta disponibile

Dalla visualizzazione è possibile notare come quasi la totalità delle criptovalute si concentri nel range che va da 0.0 a 0.1 in una scala che però deve essere moltiplicata per 10^{18}, con la presenza, anche qui, di molti outliers.

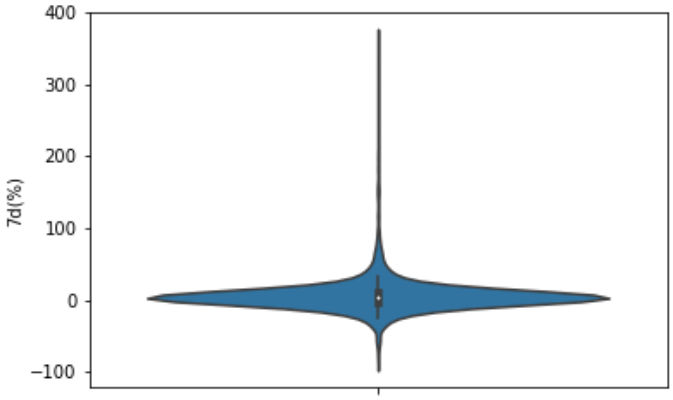

Figura 6: Distribuzione della variazione del valore della criptovaluta

Dalla visualizzazione è possibile notare come quasi la totalità delle criptovalute tenda a mantenere il proprio valore costante durante il corso di una settimana. Inoltre la maggior parte degli outliers si concentra su una variazione positiva del proprio valore, con cifre che toccano anche il 400%, questo può essere dovuto alla natura speculativa della moneta.

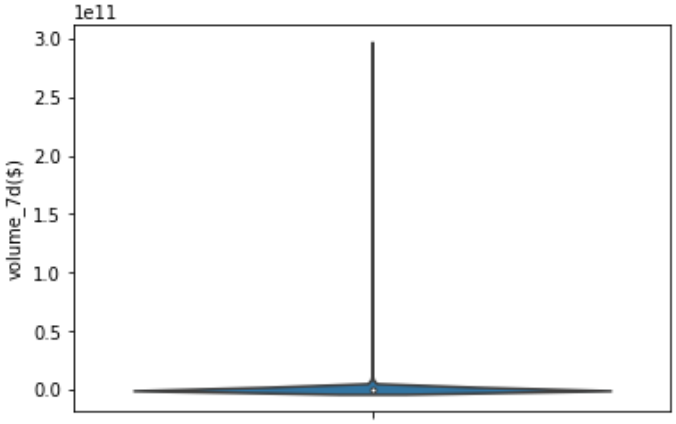

Figura 7: Distribuzione della quantità di criptovaluta scambiata settimanalmente

Dalla visualizzazione è possibile notare come quasi la totalità delle criptovalute si concentri nel range che va da 0.0 a 0.2 in una scala che però deve essere moltiplicata per 10^{11}, con la presenza, anche qui, di molti outliers.

4 Data Modelling

Il DBMS selezionato per memorizzare i dati è MongoDB. La scelta del modello documentale è dettata dalla flessibilità che dona la caratteristica di assenza di schema che, per questo particolare task, è fondamentale. Inoltre, siccome si ipotizza che il carico applicativo sia incentrato soprattuto su operazioni di lettura, MongoDB permette di ottenere delle performance estremamente efficienti nell'esecuzione delle query. Infine, siccome le relazioni criptovaluta-tweet e criptovaluta-ohlc sono relazioni di tipo uno-a-molti, la rappresentazione mediante *embedding* permette la gestione dei dati estremamente agile. È stata creata un'unica collezioni di documenti denominata "cryptofinal": all'interno sono stati memorizzati i dati, ottenuti dal lavoro di web scraping, delle prime 2000 criptovalute più influenti sul mercato. Ai dati ottenuti tramite scraping sono stati aggiunti poi i dati relativi ai tweets e i dati OHLC ottenuti per le singole criptovalute. Nel DBMS sono dunque rappresentati i seguenti dati:

- *_id*: di tipo "ObjectId", funge da identificatore univoco del documento della criptovaluta;
- *img*: di tipo "String", rappresenta il link del logo della criptovaluta;

- *name*: di tipo “String”, indica il nome della criptovaluta;
- *ticker*: di tipo “String”, rappresenta l’abbreviazione della criptovaluta;
- *price*: di tipo “Double”, corrisponde al prezzo della criptovaluta;
- *7d(%)*: di tipo “Double”, indica la variazione percentuale settimanale del valore della criptovaluta;
- *market_cap($)*: di tipo “Int64”, rappresenta la capitalizzazione di mercato della criptovaluta;
- *volume_7d($)*: di tipo “Int64”, corrisponde al volume di scambio settimanale della criptovaluta;
- *total_supply*: di tipo “Int32”, indica la quantita totale disponibile della criptovaluta;
- *dominance(%)*: di tipo “string”, rappresenta la percentuale con cui la criptovaluta domina il mercato.
- *tweets*: di tipo “Array”, è un array di oggetti contenente al proprio interno per ogni tweet ritrovato:
  - *id*: di tipo “Int64”, indica l’identificatore univoco del tweet;
  - *text*: di tipo “String”, rappresenta il testo del tweet;
- *ohlc*: di tipo “Array”, rappresenta per ogni criptovaluta con una frequenza di 4 ore i valori, degli ultimi 7 giorni rispetto alla rilevazione, di:
  - *Data*: di tipo “String”, indica la data e l’ora a cui i dati i riferiscono;
  - *Open*: di tipo “Double”, rappresenta il prezzo d’apertura della criptovaluta, ovvero il prezzo all’ora indicata nel campo *Data*;
  - *High*: di tipo “Double”, rappresenta il prezzo più alto raggiunto dalla criptovaluta nell’arco delle 4 ore successive all’ora indicata nel campo *Data*;
  - *Low*: di tipo “Doule”, rappresenta il prezzo più basso raggiunta dalla criptovaluta nell’arco delle 4 ore successive all’ora indicata nel campo *Data*;
  - *Close*: di tipo “Double” rappresenta il prezzo di chiusura della criptovaluta.

È da segnalare che, tra le 2000 criptovalute di cui sono stati raccolti i dati tramite web scraping, per 87 non è stato possibile ottenere i tweet poiché, effettivamente, non sono stati pubblicati tweet negli ultimi 7 giorni con citazioni al nome o al ticker di tali criptovalute, mentre per 317 non è stato possibile ottenere i dati OHLC poiché la piattaforma CoinGecko non fornisce i dati di tali criptovalute.

5 Data Preparation

5.1 Data Cleaning

Una volta scaricati i dati delle 2000 criptovalute tramite web scraping è necessaria una fase di preprocessing. Innanzitutto, si è notato che tutti i dati scaricati sono di tipo ”String”, inoltre è possibile vedere che molte colonne contengono al loro interno caratteri che possono influire negativamente sull’analisi esplorativa, pertanto è necessario effettuare degli accorgimenti.
Come prima cosa sono state rinominate le colonne in modo tale da rendere chiaro il dominio della singola colonna. Attraverso il seguente ciclo for vengono rimossi i caratteri indesiderati (il simbolo $ e %, le virgole che nella rappresentazione americana rappresentano il separatore per le migliaia, le spaziature a inizio e fine stringa) all’interno delle singole celle:

```
for index, row in crypto.iterrows():
    row['price'] = row['price'].replace('$', '')
    row['price'] = row['price'].replace(',', '')
    row['7d(%)'] = row['7d(%)'].replace('%', '')
    row['market_cap($)'] =
        row['market_cap($)'].replace('$', '')
    row['market_cap($)'] =
        row['market_cap($)'].replace(',', '')
    row['market_cap($)'] =
        row['market_cap($)'].replace(' ', '')
    row['volume_7d($)'] =
        row['volume_7d($)'].replace('$', '')
    row['volume_7d($)'] =
        row['volume_7d($)'].replace(',', '')
    row['volume_7d($)'] =
        row['volume_7d($)'].replace(' ', '')
```

```
row['total_supply'] =
    row['total_supply'].replace(''+row['ticker'],
    '')
row['total_supply'] =
    row['total_supply'].replace(',', '')
row['dominance(%)'] =
    row['dominance(%)'].replace('%', '')
```

Inoltre la colonna 'sign' era stata presa in quanto, sul sito web, il segno della variazione percentuale del prezzo (cioè "7d(%)") veniva rappresentato con il colore, rosso se negativo, verde se positivo. Al fine di ottenere una colonna che rappresentasse il dominio corretto è stato preso il testo presente nel tag html della colonna ***sign***. Nel caso in cui il testo contenesse la parola "up" al rispettivo record della colonna ***7d(%)*** verrà inserito il segno "+". Nel caso in cui il testo contenesse la parola "down" il segno sarà "-". Infine le colonne ***price***, ***7d(%)***, ***market_cap($)***, ***volume_7d($)***, ***total_supply*** sono state convertite in numeric.
Infine, siccome alcune criptovalute avevano un prezzo per unità davvero basso (anche dieci 0 dopo la virgola), il processo di scraping ha preso quei dati rappresentando le cifre dopo la virgola con la dicitura "..."; il prezzo di tali criptovalute è stato inserito manualmente per correggere la rappresentazione errata.
Di seguito vengono presentato le prime righe del dataset ottenuto:

	img	name	ticker	price	7d(%)	market_cap($)	volume_7d($)	total_supply	dominance(%)
0	https://s2.coinmarketcap.com/static/img/coins/...	Bitcoin	BTC	38113.66	4.05	722090979732	169996819809	18945725	41.0662
1	https://s2.coinmarketcap.com/static/img/coins/...	Ethereum	ETH	2727.98	11.61	325718294832	89815589895	119399127.94	18.524
2	https://s2.coinmarketcap.com/static/img/coins/...	Tether	USDT	1.00	0.01	78014264209	293486741428	80074893058.66	4.4464
3	https://s2.coinmarketcap.com/static/img/coins/...	BNB	BNB	379.83	1.44	62715950446	10537360532	165116760.89	3.5745
4	https://s2.coinmarketcap.com/static/img/coins/...	USD Coin	USDC	0.9998	-0.07	49990699582	22245547728	49998886896.8	2.843

Figura 8: Head del dataset ottenuto

Una volta conclusa la fase di prepocessing è possibile partire con l'exploratory data analysis (descritta con maggior dettaglio nella sezione 3 del documento).

5.2 Data Quality

5.2.1 Accuratezza Semantica

L'accuratezza semantica dei dati scaricati, in particolare i tweet, è stata svolta utilizzando delle librerie di R. In particolare, sono state utilizzate le librerie R specifiche per il text mining *tm* e *udpipe*. Per valutare se effettivamente i tweet scaricati fossero significativi e semanticamente accurati rispetto al nostro task, sono state effettuate delle operazioni di base di text mining sul corpo dei tweet per poi effettuare un'analisi finale. A causa del costo computazionale estremamente elevato di tali operazioni, l'analisi è stata effettuata su 250mila tweet estratti in modo casuale dal milione di tweet scaricati in totale.
Dopo aver estratto il campione e dopo averlo salvato in un file testuale, questo è stato letto mediante la funzione di base in R readLines, ottenendo dunque un oggetto contenente 250mila elementi distinti. Successivamente, i 250mila elementi distinti sono stati uniti in un unico testo che è stato vettorializzato e reso un Corpus attraverso la funzione VCorpus della libreria tm. Successivamente, sono stati rimossi tutti i numeri, i segni di punteggiatura e le stop words (in lingua inglese, italiana, spagnola, portoghese, francese, tedesca, russa e infine le stop words basate sullo SMART Information Retrieval System), oltre chiaramente ai whitespace. Il testo ottenuto è stato poi trasformato in una matrice di termini, sono stati sommati i valori per ogni riga in modo tale da trovare la frequenza di ripetizione dei termini. Analizzando il risultato ottenuto, le parole più frequentemente utilizzate sono le seguenti:

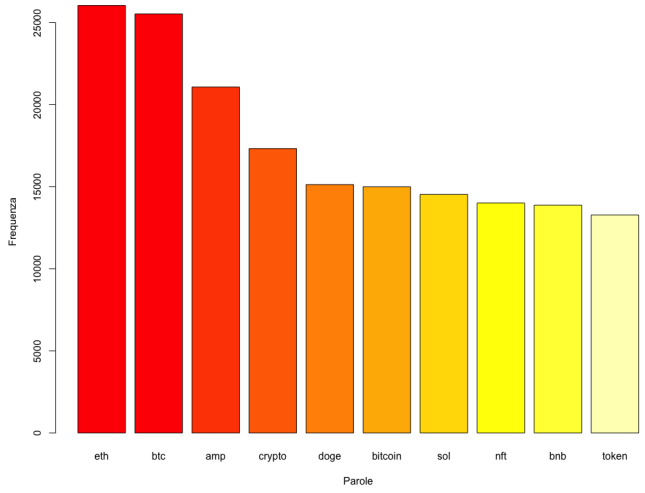

Figura 9: Bar chart delle parole più utilizzate

Estendendo l'analisi ad un insieme più ampio di termini, utilizzando la libreria R *wordcloud* è possibile generare una wordcloud:

Figura 10: Wordcloud dei tweet

Dalle due visualizzazioni è possibile notare come effettivamente tutti i termini appartengano al vocabolario di dominio, dunque è possibile concludere che (seppur su un campione dei tweet), i dati scaricati sono semanticamente accurati.

5.2.2 Completezza

La completezza di un insieme di dati rappresenta la copertura con la quale il fenomeno osservato è rappresentato nell'insieme di dati.
Misurando la completezza di attributo, ovvero il numero di valori mancanti per un attributo del dataset, si ottiene quanto segue: in un primo momento, dopo il web scraping, avevamo i dati di 2000 criptovalute nelle quali è stato possibile riscontrare un'incompletezza nella colonna "volume_7d($)" (95 valori mancanti / 2000 valori totali); per quanto riguarda l'attributo "tweets" (87 criptovalute senza tweets / 2000 valori totali) e "ohlc" (317 criptovalute senza OHLC Data / 2000 valori totali).
Ai fini del nostro task, le criptovalute per cui non sono presenti né tweets né dati OHLC sono inutili, per questo motivo è stato deciso di rimuovere i dati delle 25 criptovalute per cui non sono stati trovati né tweets né OHLC Data. Dopo la rimozione quindi sono 12 le criptovalute senza "tweet" (su 1975), 292 quelle senza OHLC Data (su 1975) e 90 quelle senza volume_7d($).
Si ricorda comunque che, data l'assenza di schema, i valori mancanti non sono NULL ma semplicemente non viene rappresentato l'attributo nei documenti delle criptovalute.

5.3 Data Enrichment

È stata effettuata effettuata la fase di arricchimento dei dati al fine di poter aggiungere attributi al dataset iniziale ottenuto mediante web scraping. Le sorgenti utilizzate per effettuare la fase di arricchimento sono l'API di Twitter, tramite la libreria python "tweepy", e l'API di CoinGecko tramite la libreria python "pycoingecko".
Attraverso l'API di Twitter, per recuperare i tweet si è pensato di effettuare due chiamate API per ogni criptovaluta, andando a passare nella prima chiamata il nome della criptovaluta e nella seconda il suo ticker: in questo modo sono stati ottenuti i tweet più recenti (degli ultimi 7 giorni a partire dalla richiesta) nel cui corpo viene citato o il nome intero della criptovaluta, o il suo ticker, o entrambi. Vengono così restituiti due dataframe contenenti id e testo dei tweets, che sono stati uniti tra loro per poi procedere con l'eliminazione dei duplicati sulla base dell'id del tweet.
Per quanto riguarda invece gli OHLC Data, prima di tutto è stata effettuata una chiamata API per ottenere la lista degli id di tutte le criptovalute disponibili sulla piattaforma CoinGecko. Una volta ottenuti tutti gli id, sono stati estratti tutti gli identificativi univoci delle criptovalute di nostro interesse per poi andare ad effettuare una seconda chiamata per ognuna di esse, ottenendo dunque i dato storici degli ultimi 7 giorni aggiornati ogni 4 ore.

6 Memorizzazione nel DBMS

6.1 Tweet

La memorizzazione all'interno del DBMS si divide in 2 parti. La prima parte è stata effettuata mediante MongoDB Compass: è stato manualmente caricato il dataset ottenuto dall'operazione di web scraping all'interno della collezione "cryptofinal".
La seconda parte invece è stata effettuata mediante python: grazie alla libreria pymongo e in particolare il modulo MongoClient è stato possibile realizzare la connessione al DBMS:

```
clientcrypto = MongoClient('localhost', 27017)
db = clientcrypto.test
dbcrypto = clientcrypto.dataman
collection = dbcrypto["crypto"]
```

Grazie a "collection" è possibile accedere alla collection "cryptofinal" posta all'interno del database "dataman". Una volta connesso alla collection viene ef-

fettuata una query per poter accedere all'intera lista delle criptovalute, dalla quale verranno estrapolati nome, abbreviazione e id della criptovaluta che verranno inseriti all'interno di tre distinte liste.

```
response = collection.find({})
n = 0
list_nome = []
list_abbr = []
list_id = []

for document in response:
    list_nome.append(document['name'])
    list_abbr.append(document['ticker'])
    list_id.append(document['_id'])
```

Siccome il processo di download dei tweet e dei dati OHLC richiede molto tempo, memorizzare i dati relativi a id del documento, nome e ticker delle criptovalute nelle tre distinte liste ci permette di evitare problemi relativi al timeout del cursore di MongoDB (timeout che si attiva automaticamente dopo 10 minuti di inattività del cursore o al massimo, in seguito alla modifica di alcuni parametri, dopo 30 minuti di inattività).
Le liste contenenti il nome e il ticker delle criptovalute vengono utilizzate per effettuare le chiamate API per il download dei tweet. In particolare, il download viene avviato dalla funzione downloadTweets definita come segue:

```
def downloadTweets(query):
    tweets = client.search_recent_tweets(query,
        max_results=100)
    currentCRYPTOdataset =
        tweetsDataFrame(tweets)
    print("Presi i tweet di "+query)
    print(tweets.meta)
    for _ in range(3):
        if 'next_token' in tweets.meta:
            tweets =
                client.search_recent_tweets(query,
                max_results=100,
                next_token=tweets.meta['next_token'])
            df = tweetsDataFrame(tweets)
            currentCRYPTOdataset =
                currentCRYPTOdataset.append(df,
                ignore_index=True)
        else:
            break

        print("Presi i tweet di "+query)
    return currentCRYPTOdataset
```

La funzione downloadTweets si occupa innanzitutto di recuperare tutti i tweets della prima pagina per passarli alla funzione tweetsDataFrame che creerà un dataframe contenente l'id e il testo di ogni tweet trovato. In secondo luogo viene controllata la presenza di pagine successive alla prima fino ad un massimo di 4 pagine: per ogni pagina successiva alla prima verranno raccolti tutti i tweets e verrà richiamata tweetsDataFrame che restituirà il dataframe aggiornato.
La funzione tweetsDataFrame è definita come segue:

```
def tweetsDataFrame(tweets):
    testo = list()
    id = list()

    for tweet in tweets.data:
        testo.append(tweet.text)
        id.append(tweet.id)

    data_vuoto = []
    df = pd.DataFrame(data_vuoto)

    df['id'] = id
    df['text'] = testo


    return df
```

Una volta ottenuti i due dataframe contenenti gli id e il testo dei tweet per nomi e abbreviazioni, questi vengono uniti e successivamente puliti mediante il metodo drop_duplicates che va ad eliminare i duplicati basandosi sull'id univoco di ogni tweet.
Dopo l'eliminazione dei duplicati, il dataframe viene trasformato in un dizionario che successivamente viene caricato all'interno di MongoDB mediante il metodo update_many andando ad aggiornare il documento della criptovaluta che si sta processando in base al suo "_id".

```
df_unione.drop_duplicates(subset=['id'],
    keep='first', inplace=True,
    ignore_index=True)
                print('la lunghezza del dataframe
                    di' , nome, 'dopo il drop :',
                    len(df_unione))
                df_load =
                    df_unione.to_dict("records")
                collection.update_many({"_id":ide},
                    {"$set":{"tweets":df_load}})
```

```
            print("ho inserio i documenti
                di", nome)
```

Per gestire le eccezioni, il tutto è inserito all'interno di un try catch generico per evitare problematiche relative alla gestione del numero di richieste all'API di Twitter: qualora infatti il codice d'errore risultasse essere corrispondente al 429, viene messa in timeout l'esecuzione del codice grazie al metodo sleep (parte della libreria time), che terrà il codice in sleep mode per 15 minuti, ovvero il tempo utile affinché sia possibile effettuare nuove richieste.

```
except Exception as e:
        print(e)
        if str(e) == "429 Too Many Requests":
                print("Pausa 15 minuti")
                time.sleep(15*60)
```

6.2 OHLC Data

Per quanto riguarda gli OHLC Data, la memorizzazione nel DBMS avviene in seguito al termine del download e della memorizzazione dei tweets per tutte le criptovalute.
Come prima operazione è stato inizializzato il client per l'API CoinGecko ed è stata richiesta la lista di tutte le criptovalute, attraverso la funzione get_coins_list, gestite dalla piattaforma per poter recuperare il loro id, fondamentale per le richieste successive.

```
id_list = cg.get_coins_list()
cg_list = pd.DataFrame(id_list)
cg_list.head()
```

	id	symbol	name
0	01coin	zoc	01coin
1	0-5x-long-algorand-token	algohalf	0.5X Long Algorand Token
2	0-5x-long-altcoin-index-token	althalf	0.5X Long Altcoin Index Token
3	0-5x-long-ascendex-token-token	asdhalf	0.5X Long AscendEx Token Token
4	0-5x-long-bitcoin-cash-token	bchhalf	0.5X Long Bitcoin Cash Token

Figura 11: Richiesta della lista e stampa delle prime righe

Una volta ottenuta la lista completa, sono estate estratte le righe relative alle sole criptovalute di nostro interesse, ovvero quelle presenti nel dataset precedentemente ottenuto.
Dopo aver estratto gli id delle criptovalute di nostro interesse, è stata effettuata una chiamata API attraverso la funzione *get_coin_ohlc_by_id* per ognuna di esse, ottenendo i dati OHLC degli ultimi 7 giorni aggiornati ogni 4 ore. La funzione sopracitata restituisce un dataframe con 42 righe (una per ogni range di 4 ore) e 5 colonne (Timestamp, Open, High, Low, Close). Ogni riga è identificata univocamente da un timestamp in millisecondi che, per motivi di leggibilità, è stato convertito in formato di data tradizionale. Dopo la conversione, il dataframe ottenuto viene convertito in un dizionario e caricato in MongoDB attraverso la funzione update_many, andando ad aggiornare il documento della criptovaluta appena processata.

```
for i in range(0, len(coins_presenti)):
    idc = ObjectId(coins_presenti['id_x'][i])
    try:
        ohlc = cg.get_coin_ohlc_by_id
        (coins_presenti['id_y'][i],
            vs_currency='usd', days=7)
        print('Presi i dati di ',
            coins_presenti['abbr'][i])
        for j in range(0, len(ohlc)):
            ohlc[j][0] = datetime.
            fromtimestamp(ohlc[j][0]/1000).
            strftime(format = '%Y-%m-%d %X%z')
        ohlc_df = pd.DataFrame(ohlc)
        ohlc_df.columns = ['Data', 'Open',
            'High', 'Low', 'Close']
        ohlc_dict = ohlc_df.to_dict("records")
        collection.update_many({"_id": idc},
            {"$set": {"ohlc": ohlc_dict}})
```

Per gestire le eccezioni, il tutto è inserito in un try-catch generico per evitare ancora una volta problematiche legate al numero di richieste all'API di CoinGecko: nel caso in cui venisse sollevata un'eccezione con codice d'errore 429, l'esecuzione del codice viene messa in timeout grazie al metodo sleep che la terrà in sleep mode per 3 minuti, per poi effettuare nuovamente la richiesta e memorizzare i dati ottenuti in MongoDB.

```
except Exception as e:
        print(e)
        if "429" in str(e):
            print("Pausa")
            time.sleep(3*60)
            ohlc = cg.get_coin_ohlc_by_id
            (coins_presenti['id_y'][i],
                vs_currency='usd', days=7)
            for x in range(0, len(ohlc)):
                ohlc[x][0] =
```

```
            datetime.fromtimestamp
        (ohlc[x][0]/1000).strftime
        (format = '%Y-%m-%d %X%z')
    ohlc_df = pd.DataFrame(ohlc)
    ohlc_df.columns = ['Data', 'Open',
        'High', 'Low', 'Close']
    ohlc_dict =
        ohlc_df.to_dict("records")
    collection.update_many({"_id": idc},
        {"$set": {"ohlc": ohlc_dict}})
    print('Caricati i dati di',
        coins_presenti['abbr'][i])
```

7 Conclusioni e Sviluppi Futuri

Una prima idea di sviluppo futuro potrebbe essere quella di ampliare il codice sorgente relativo all'acquisizione dei tweet in modo tale da creare una pipeline di text mining, così da realizzare un modello per il ritrovamento dei tweet rilevanti rispetto alla query formulata. Questa pipeline si occuperebbe di:

- Suddividere ogni parola in token;
- Rimuovere le *stop words*;
- Normalizzare i termini andando ad attribuire a token sintatticamente diversi ma semanticamente uguali lo stesso significato;
- Gestione sinonimi e omonimi.

Un secondo sviluppo potrebbe essere la realizzazione di una sentiment analysis relativa alla variazione dell'andamento delle criptovalute, così da poter realizzare raccomandazioni riguardanti gli investimenti più convenienti da effettuare.
Per portare a compimento eventuali task futuri, il piano *Academic Research* dell'API di Twitter potrebbe essere utile in modo tale da svincolarsi dal limite di 2milioni di tweet al mese da poter scaricare.

8 Librerie Utilizzate

8.0.1 Python

- os;
- time;
- tweepy;
- pycoingecko;
- pymongo;
- pandas;
- numpy;
- datetime;
- bson;
- json;
- matplotlib;
- seaborn.

8.1 R

- tm;
- udpipe;
- wordcloud;
- RColorBrewer.

9 Bibliografia

Di seguito vengono elencate le librerie utilizzate per lo svolgimento del progetto:

- `https://docs.python.org/3/library/os.html`;
- `https://docs.python.org/3/library/time.html`;
- `https://docs.tweepy.org/en/stable/`;
- `https://seaborn.pydata.org`;
- `https://pandas.pydata.org/docs/`;
- `https://matplotlib.org`;
- `https://numpy.org/doc/`;
- `https://docs.python.org/3/library/json.html`;
- `https://pymongo.readthedocs.io/en/stable/`;
- `https://pycoingecko.readthedocs.io/en/latest/`;
- `https://docs.python.org/3/library/datetime.html`;

- https://pymongo.readthedocs.io/en/stable/api/bson/index.html;
- https://www.rdocumentation.org;
- https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-annotation.html;
- https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf;
- https://cran.r-project.org/web/packages/RColorBrewer/index.html;