

Web Data Analytics

Crypto Sentiment Analyzer


BUFFA Salvatore

DI FINA Domenico

GRISTINA Salvatore Antonino

?? giugno 2021

Codice sorgente disponibile su [GitHub](#)



| | |
|-----------------------------------|-----------|
| 1. Introduzione | 2 |
| 2. Dataset | 3 |
| 2.1 Sentiment dataset | 3 |
| 2.2 Emotions dataset | 5 |
| 2.2.1 Emotions dataset for NLP | 5 |
| 2.2.2 Emotion Detection from Text | 6 |
| 2.2.3 Unione Dataset | 7 |
| 3. Preprocessing | 8 |
| 4. Modelli | 9 |
| 4.1 Sentiment Model | 9 |
| 4.2 Emotion Model(bozza) | 9 |
| 4.3 Topic Recognition | 10 |
| 5. Interfaccia Grafica | 12 |



1. Introduzione

Il progetto descritto successivamente nasce con l'idea di applicare quanto appreso durante il corso di web data analytics del professore Pilato Giovanni. Nello specifico all'interno di questo progetto si sono applicate tecniche di Information Retrieval, Text Preparation, Sentiment Analysis ed Emotion Detection che verranno trattate nel dettaglio successivamente. Tutto questo è stato fatto applicando tali tecniche al progetto in questione, **Crypto Sentiment Analyzer**. Questo applicativo, costruito a scopo didattico, fornisce all'utente finale una panoramica dell'andamento di una particolare criptovaluta tra quelle messe a disposizione sfruttando i tweet presenti in rete entro un certo arco di tempo scelto dall'utente e mostrandogli tre diversi grafici in merito a: sentimenti, emozioni e combinazione dei due.

2. Dataset

La prima fase del progetto, consiste nel prelevare quanti più dati possibili per cercare di effettuare un corretto addestramento dei due modelli previsti, ovvero il modello per i sentimenti ed il modello per le emozioni

2.1 Sentiment dataset


Per l'addestramento del modello per i sentimenti, si è deciso di prendere il dataset fornito gratuitamente dalla piattaforma kaggle presente a questo link:

<https://www.kaggle.com/kazanova/sentiment140>

Tale dataset contiene 1.600.000 di tweets estratti utilizzando le API messe a disposizione da Twitter e possiede la seguente struttura:

- **Target:** ovvero la polarità del tweet (0 negativo, 2 neutrale, 4 positivo)
- **Ids:** ID del tweet
- **Date:** la data del tweet
- **Flag:** la query utilizzata per trovare quel dato tweet, se non è stata utilizzata alcuna query il valore di default è NO_QUERY
- **User:** username di chi ha scritto quel particolare tweet
- **Text:** il testo del tweet

Una volta caricato tutti il dataset, sono state eliminate tutte le informazioni non necessarie quali: user, flag, date, ids. Inoltre prima di effettuare la fase di pre-processing è stata fatta una accurata analisi del dataset per capirne la forma ed i dati messi a disposizione. Di seguito vengono riportate tutte le informazioni ricavate dal dataset utilizzando la libreria **pandas** di Python.



| | |
|--------------------------------------|-----------|
| Dimensione dataset | 160000000 |
| Sentimenti unici riscontrati | 0,4 |
| Numero di elementi nulli nel dataset | 0 |
| Percentuale di tweets positivi | 50% |
| Percentuale di tweets negativi | 50% |

Questa analisi preliminare fornisce molte informazioni sul dataset fornito. Nello specifico oltre alla dimensione già definita dalla piattaforma è possibile notare come all'interno del dataset non siano presenti sentimenti neutrali ma solo positivi e negativi riducendo di fatto il modello dei sentimenti al riconoscimento di sentimenti positivi e negativi. E' possibile inoltre notare come non siano presenti elementi nulli all'interno del dataset, questo facilita enormemente la fase di pre-processing poiché non è necessario rimuovere delle righe o introdurre del rumore all'interno del dataset per eliminare eventuali valori mancanti. In merito alla distribuzione dei dati invece è possibile notare come vi sia una perfetta distribuzione tra i tweets negativi ed i tweets positivi presenti nel dataset.

Infine si vuole precisare che vista la grande quantità di dati messi a disposizione con questo dataset, non sono stati considerati ulteriori dataset per non gravare eccessivamente sulla parte computazionale richiesta per l'addestramento del modello.



2.2 Emotions dataset

Un altro obiettivo del progetto è quello di riconoscere le **emozioni** espresse dagli utenti tramite dei tweet in un lasso di tempo per comprendere come sta procedendo il mercato delle crypto.

Per acquisire questa funzione nel modo più efficiente possibile, è stato deciso di addestrare una rete neurale con i seguenti dataset: [Emotions dataset for NLP](#) e [Emotion Detection from Text](#).

2.2.1 Emotions dataset for NLP

Questo primo dataset è formato da tre file **test.txt**, **train.txt**, **val.txt**, con un totale di circa 20.000 elementi.

Ognuno di questi file ha una struttura a doppia colonna, dove la prima rappresenta le frasi contenute nel dataset e la seconda contiene l'emozione predominante di quest'ultima.

Le **frasi** non sono dei tweet.

Le **emozioni** sono: **sadness, joy, fear, anger, love, surprise**.

Per semplicità è stato deciso di unificare i tre file mantenendo l'unicità dei dati e posticipare le divisioni in set di test, set di addestramento e set di validazione in seguito.

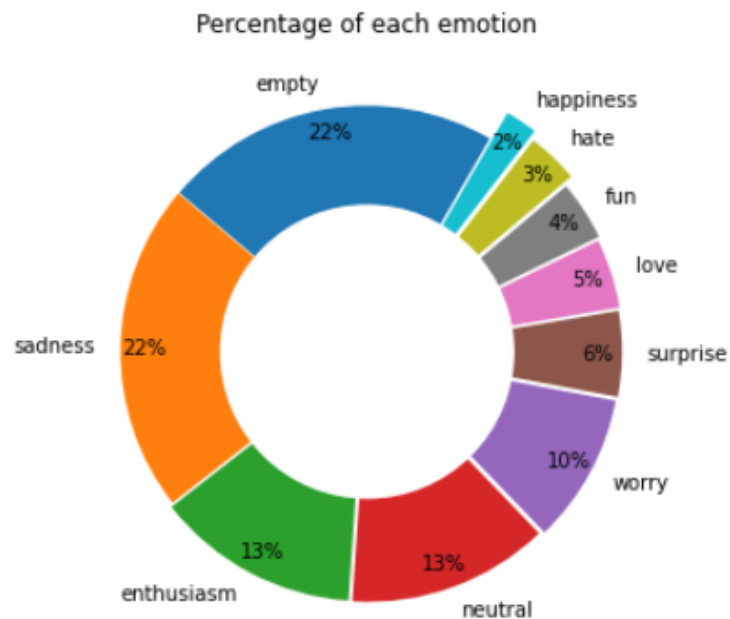
2.2.2 Emotion Detection from Text

Questo secondo dataset è formato da un unico file **tweet_emotions.csv**, con un totale di circa 40.000 elementi.

Il file ha una struttura a tre colonne, dove la prima rappresenta gli id delle frasi, la seconda contiene le emozioni e infine la terza colonna contiene le frasi.

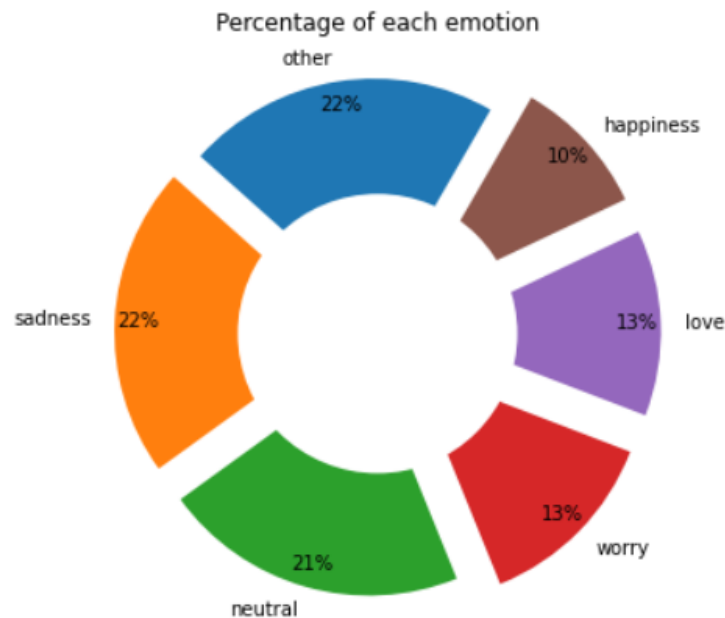
Le **frasi** sono dei tweet

Le **emozioni** sono rappresentate tramite il seguente diagramma circolare



Come si può notare questo dataset risulta molto sbilanciato quindi per ottimizzare l'apprendimento della rete neurale è stato deciso di ridurre il numero delle emozioni.

Per fare ciò sono state mantenute solamente le seguenti emozioni



Durante lo sviluppo delle reti neurali è stato ritenuto opportuno rimuovere anche l'emozione **other** poiché troppo predominante per rendere l'addestramento dei modelli più specifico.

2.2.3 Unione Dataset

Come si è potuto capire, sono stati scelti due dataset per colmare la mancanza di elementi da dare in pasto alla rete neurale. I due dataset sono abbastanza diversi tra loro quindi è stato deciso di lavorare sul primo modificando la struttura del dataset e le emozioni al suo interno, per potere unire i due file ottenendo un unico dataset più grande di circa 60.000 elementi

3. Preprocessing

Una volta recuperati i dati utili per il progetto, un passo importante nel processo di data mining è il **preprocessing**, che ha lo scopo di trasformare i dati testuali in dati comprensibili e utilizzabili da un sistema di text mining.

Questo obiettivo è stato raggiunto attraverso i seguenti passi:

- la prima operazione effettuata nei dati è stata la trasformazione di ogni testo in **minuscolo**, così da poter confrontare le singole parole in maniera corretta, sostituendo inoltre anche i caratteri di *new line* con un carattere di spaziatura;
- considerato che il nostro insieme di dati è costituito prevalentemente da *tweet* (con un limite dei caratteri preimpostato) si è presentata la necessità di trasformare tutti gli **acronimi** che nella lingua inglese vengono utilizzati al posto delle frasi complete (es. *asap* = *as soon as possible*). Inoltre, i verbi con la contrazione vengono portati alla forma base (es. *there's* = *there is*);
- nei testi, soprattutto quelli ricavati direttamente dal web, vi possono essere presenti dei **link**, oppure dei riferimenti ad username. Per questo motivo si è deciso di eliminare tutti questi elementi dal testo, poiché non sono ritenuti utili;
- infine, si sono eliminati tutti i **caratteri simbolici** (es. #) e di punteggiatura.

Dopo aver eseguito i passi di pulitura, ogni singolo dato testuale viene tokenizzato attraverso la libreria *nltk*, utilizzando la funzione **word_tokenize**, che restituisce per ogni frase la lista di termini in essa. Quindi in tali liste vengono eliminate le cosiddette **stop_words**, ovvero le parole che non portano particolare informazione al testo e che quindi possono essere ignorate nei processi successivi.

Nota. Non si è utilizzato alcun processo di **stemming**, poiché ritenuto non molto efficace per quanto riguarda la lingua inglese, come studiato a lezione.

4. Modelli

Una delle parti cruciali di questo progetto riguarda il corretto addestramento dei tre modelli utilizzati per: classificare correttamente i tweet in **positivi** e **negativi** per il modello riguardante i sentimenti; classificare correttamente i tweet in **neutral**, **happiness**, **love**, **worry** e **sad** per il modello riguardante le emozioni; riconoscere correttamente il topic di uno specifico tweet mediante apposite funzioni studiate durante il corso. Dopo una lunga serie di prove utilizzando diversi tipi di reti (KNN, LSTM, etc.) si è giunti alla conclusione che per i modelli di classificazione si è utilizzata una rete LSTM che come si evince empiricamente in letteratura risulta il modello più efficace nel caso in cui è necessario trattare frasi e testi mantenendo una finestra temporale. Naturalmente trattandosi di reti neurali è necessario prima effettuare una tokenizzazione dei tweets in input con una successiva fase di padding per uniformare tutti i vettori delle frasi, rispettivamente sono state utilizzate due funzioni messe a disposizione dal framework **keras** denominate `Tokenizer` e `pad_sequences`. Mentre per il terzo modello riguardante l'individuazione del topic, si è scelto di utilizzare la funzione `TD-IDF` descritta successivamente.

4.1 Sentiment Model

Come descritto precedentemente tra le varie reti di machine learning provate, la più efficiente risulta essere quella **LSTM**, che come si vede anche empiricamente è la più utilizzata nel campo della speech recognition e di tutte quelle tematiche che implicano l'analisi di frasi e testi. Nello specifico viene riportato di seguito il modello creato tramite il framework **keras**:

```
Model: "sequential"
```

| Layer (type) | Output Shape | Param # |
|-----------------------|-----------------|---------|
| embedding (Embedding) | (None, 50, 300) | 9000000 |
| lstm (LSTM) | (None, 196) | 389648 |
| dense (Dense) | (None, 2) | 394 |

```
Total params: 9,390,042  
Trainable params: 9,390,042  
Non-trainable params: 0  
None
```

composto da un primo layer di embedding che permette di trasformare un vettore di interi positivi, ottenuti nello specifico tramite la fase di tokenizzazione e successivo padding, in un vettore denso di dimensione fissata. Il secondo layer invece è un layer LSTM formato da 196 unità con dropout impostato a 0.2. Infine il terzo layer ovvero quello di output è un layer Dense che restituisce in uscita un vettore composto da due elementi che indica quanto più una predizione è vicina ad essere negativa o positiva.

Di seguito vengono riportati i migliori risultati ottenuti dal modello dopo una lunga fase di addestramento del modello, ovvero si è ottenuta una accuray durante la fase di train con 10 epoche pari a 84% mentre una accuracy durante la fase di validation pari a 78%

```

Epoch 1/10
1250/1250 [=====] - 1559s 1s/step - loss: 0.4970 - accuracy: 0.7549
Epoch 2/10
1250/1250 [=====] - 1551s 1s/step - loss: 0.4399 - accuracy: 0.7930
Epoch 3/10
1250/1250 [=====] - 1550s 1s/step - loss: 0.4241 - accuracy: 0.8032
Epoch 4/10
1250/1250 [=====] - 1546s 1s/step - loss: 0.4129 - accuracy: 0.8098
Epoch 5/10
1250/1250 [=====] - 1548s 1s/step - loss: 0.4012 - accuracy: 0.8168
Epoch 6/10
1250/1250 [=====] - 1547s 1s/step - loss: 0.3915 - accuracy: 0.8231
Epoch 7/10
1250/1250 [=====] - 1548s 1s/step - loss: 0.3804 - accuracy: 0.8293
Epoch 8/10
1250/1250 [=====] - 1549s 1s/step - loss: 0.3696 - accuracy: 0.8354
Epoch 9/10
1250/1250 [=====] - 1551s 1s/step - loss: 0.3570 - accuracy: 0.8424
Epoch 10/10
1250/1250 [=====] - 1550s 1s/step - loss: 0.3444 - accuracy: 0.8482
***** EVALUATION *****
10000/10000 [=====] - 277s 28ms/step - loss: 0.4800 - accuracy: 0.7826

```

Inoltre per giustificare tale scelta si è deciso anche di provare con una rete SVM ma raggiungendo scarsi risultati, infatti abbiamo ottenuto una accuracy sul validation set pari a 50%.

4.2 Emotion Model

La prima implementazione di questo modello è stata effettuata tramite un classificatore, nello specifico un classificatore **kNN** che è un classificatore lazy poiché non costruisce un modello di classificazione a priori, ma la classificazione avviene quando un nuovo documento arriva per essere classificato. Questa implementazione non è risultata opportuna per lo studio delle emozioni che è richiesto in questo progetto, ecco i risultati del kNN:

```

k = 1, accuracy: 0.30616636053529256
k = 2, accuracy: 0.132983468905799
k = 3, accuracy: 0.21052217265809497
k = 4, accuracy: 0.1182104434531619
k = 5, accuracy: 0.1597480976121753
k = 6, accuracy: 0.10244030438205196
k = 7, accuracy: 0.12920493308842823
k = 8, accuracy: 0.09202309105221726
k = 9, accuracy: 0.10881658357386513
k = 10, accuracy: 0.08171083705064287

```

come si può notare il valore massimo di accuratezza raggiunto è del 30%.

Quindi si è deciso di implementare una rete neurale **LSTM**, ecco una rappresentazione della rete tramite il framework keras

```
Model: "sequential"
```

| Layer (type) | Output Shape | Param # |
|-----------------------|-----------------|---------|
| embedding (Embedding) | (None, 40, 300) | 1500000 |
| lstm (LSTM) | (None, 128) | 219648 |
| dense (Dense) | (None, 5) | 645 |

```
Total params: 1,720,293  
Trainable params: 1,720,293  
Non-trainable params: 0
```

composto da un primo layer di embedding che permette di trasformare un vettore di interi positivi, ottenuti nello specifico tramite la fase di tokenizzazione e successivo padding, in un vettore denso di dimensione fissata. Il secondo layer invece è un layer LSTM formato da 128 unità con dropout impostato a 0.2. Infine il terzo layer ovvero quello di output è un layer Dense che restituisce in uscita un vettore composto da cinque elementi che indica una predizione dell'emozione predominante in una frase.

Infine vengono mostrati i risultati ottenuti dalla rete in una fase di training formata da 35 epoche

```
1787/1787 [=====] - 452s 253ms/step - loss: 0.1309 - accuracy: 0.9536
Epoch 26/35
1787/1787 [=====] - 452s 253ms/step - loss: 0.1237 - accuracy: 0.9552
Epoch 27/35
1787/1787 [=====] - 470s 263ms/step - loss: 0.1204 - accuracy: 0.9588
Epoch 28/35
1787/1787 [=====] - 469s 263ms/step - loss: 0.1138 - accuracy: 0.9598
Epoch 29/35
1787/1787 [=====] - 501s 281ms/step - loss: 0.1072 - accuracy: 0.9617
Epoch 30/35
1787/1787 [=====] - 528s 295ms/step - loss: 0.1056 - accuracy: 0.9630
Epoch 31/35
1787/1787 [=====] - 479s 268ms/step - loss: 0.1017 - accuracy: 0.9637
Epoch 32/35
1787/1787 [=====] - 467s 261ms/step - loss: 0.0984 - accuracy: 0.9657
Epoch 33/35
1787/1787 [=====] - 494s 276ms/step - loss: 0.0941 - accuracy: 0.9670
Epoch 34/35
1787/1787 [=====] - 460s 257ms/step - loss: 0.0902 - accuracy: 0.9683
Epoch 35/35
1787/1787 [=====] - 521s 291ms/step - loss: 0.0893 - accuracy: 0.9689
```

La rete LSTM raggiunge un'accuratezza quasi del **97%** sul set di training e un'accuratezza dell'**82%** sul set di test

```
1191/1191 [=====] - 34s 27ms/step - loss: 0.8589 - accuracy: 0.8251
```

4.3 Topic Recognition

Uno degli obiettivi del *text mining* è quello di identificare in un testo il **topic centrale**.

In particolare, per quanto detto in precedenza, il progetto riguarda il mondo crypto e le sue valute. Quindi, il nostro scopo è quello di trovare la lista di criptovalute di cui si discute nei *tweet* in esame.

Per fare ciò, si è deciso di utilizzare **TF-IDF**, in particolare la funzione *TfidfVectorizer* di *Scikit-learn* che permette la conversione di documenti in un vettore di pesi TF-IDF.

La funzione di peso tf-idf (term frequency–inverse document frequency) è una funzione utilizzata in information retrieval per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti.

Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti.


La formula utilizzata per il calcolo dei pesi $w_{i,j}$ è la seguente:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

dove $tf_{i,j}$ è la frequenza del termine k_i nel documento d_j , N è il numero di documenti nella collezione e df_i è il numero di documenti che contengono il termine k_i .

Quindi, una volta creati tutti i vettori corrispondenti ai tweet da analizzare, si applica la stessa procedura di vettorializzazione alla lista contenente le 20 criptovalute più famose con i rispettivi simboli :

[Bitcoin BTC, Ethereum ETH, Ripple XRP, Binance Coin BNB, Tether USDT, Cardano ADA, Dogecoin DOGE, Polkadot DOT, Internet Computer ICP, XRP, Uniswap UNI, Polygon MATIC, Stellar XLM, Litecoin LTC, VeChain VET, Solana SOL, SHIBA INU SHIB]



Attraverso la rappresentazione vettoriale di ogni singola criptovaluta, possiamo misurare la similarità tra un tweet e ogni singolo elemento nella lista sopra citata, così da ricavare le criptovalute che sono citate nel tweet.

Come misura di similarità si è deciso di utilizzare il **coseno**:

$$\cos(\theta) = \frac{A \bullet B}{||A|| \times ||B||}$$

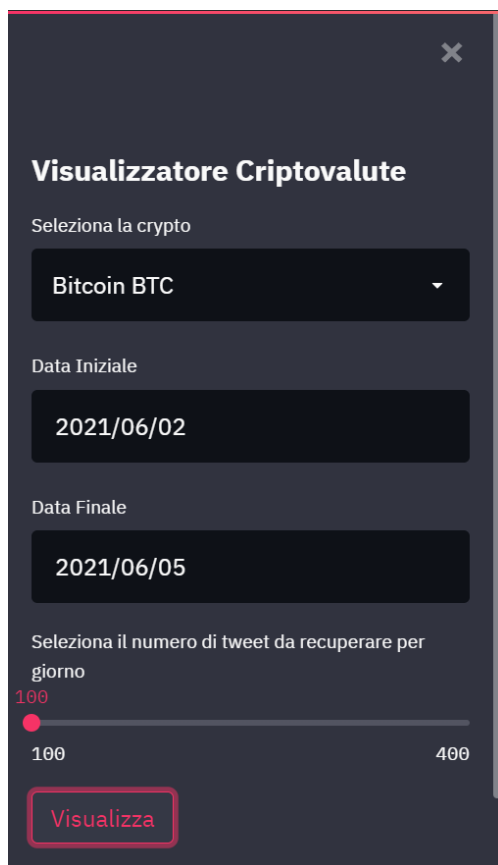
5. Interfaccia Grafica

Con l'obiettivo di rendere il più utilizzabile possibile il progetto, si è deciso di creare un'interfaccia grafica per l'utente.

In particolare, si è creato un sito web

(<https://share.streamlit.io/domenicodifina/crypto-sentiment-project/main/gui.py>)

utilizzando una libreria in Python chiamata **Streamlit**, che permette di creare applicativi web in maniera rapida e semplice, sfruttando direttamente la repository Github del progetto.



La figura a sinistra mostra la barra laterale dell'interfaccia. In questa barra l'utente può scegliere di quale criptovaluta vuole conoscere l'andamento del sentiment/emotion, può scegliere la data iniziale e la data finale della ricerca e può selezionare il numero di tweet per giorno da recuperare.

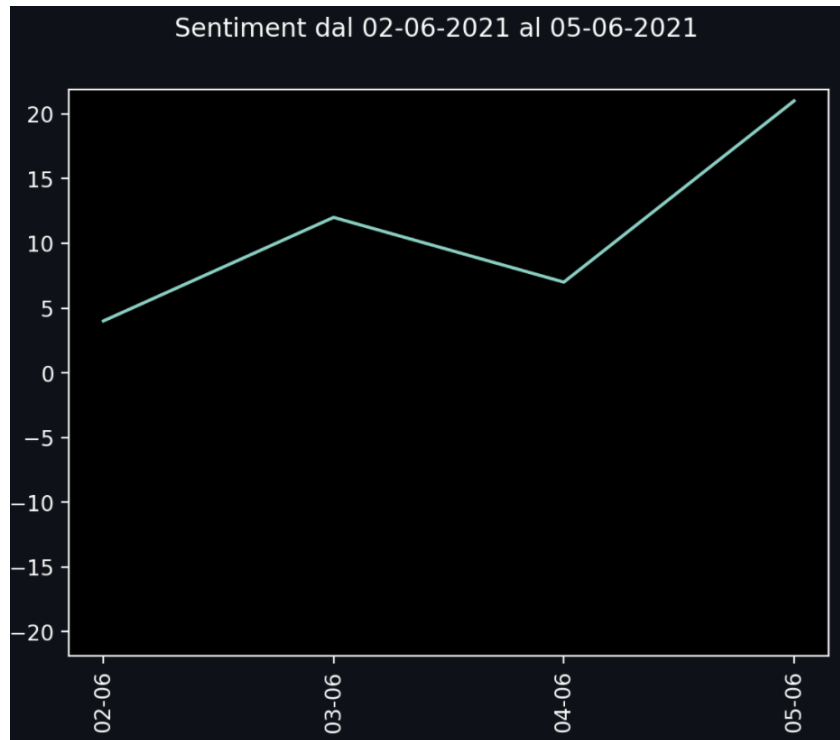
I tweet vengono recuperati in tempo reale tramite la libreria **Twint**, che permette di effettuare scraping dei tweet direttamente da Twitter, senza avere il limite di dati (imposto dall'API ufficiale) che è possibile ricevere.

Tale libreria fornisce molte informazioni e permette di specificare molti parametri utili, tra cui la possibilità (utilizzata nel progetto) di ricavare tweet da account verificati, considerando quindi un grado di prestigio.

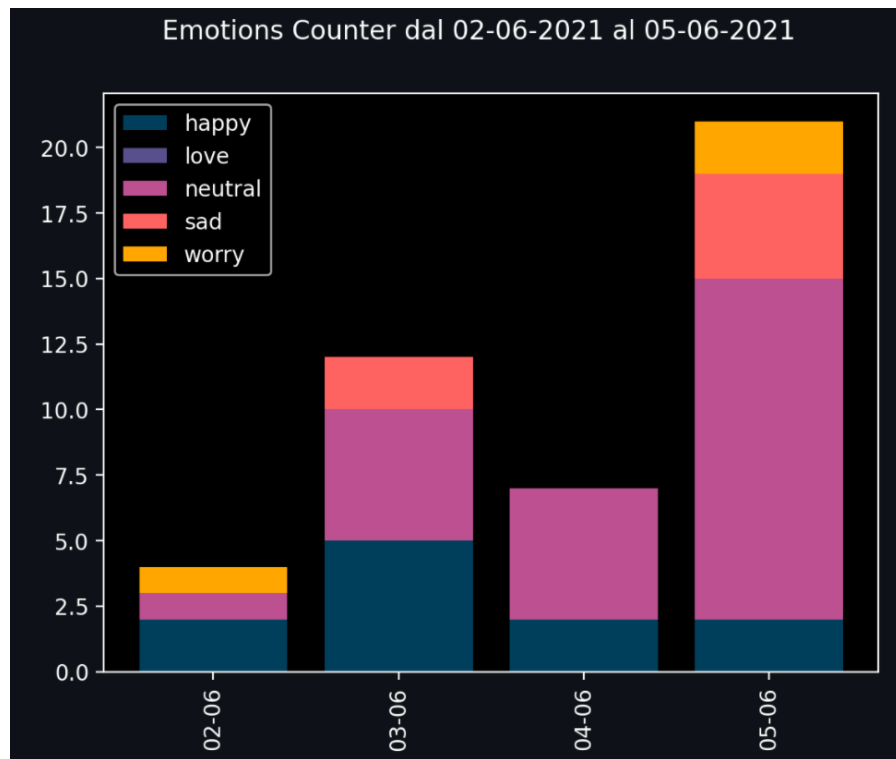
Una volta che l'utente specifica gli input, premendo il tasto *Visualizza*, verranno mostrati i seguenti 3 grafici:

1. **Sentiment Analysis:** grafico riguardante l'andamento del **sentiment**, in particolare si assegna un punteggio per ogni tweet: 1 se il sentiment è positivo, -1 altrimenti,

quindi vengono sommati tutti i punteggi raggruppati per giorno, così da ottenere il seguente grafico, dove l'ascissa rappresenta i giorni, mentre l'ordinata il punteggio:



2. **Emotion Analysis:** grafico riguardante il numero di **emozioni** presenti nei tweet raggruppati per giorno:



3. Sentiment/Emotion Analysis: grafico che combina le due informazioni riassunte nei due grafici precedenti.

In particolare, il punteggio definito precedentemente per il sentiment viene moltiplicato per un punteggio assegnato all'emozione. I punteggi in questione sono i seguenti:

- neutral:** 1
- happiness** e **worry:** 2
- love** e **sad:** 3

Per esempio, se il sentiment è negativo e l'emozione è *worry*, allora il punteggio da sommare sarà $-1 \times 2 = -2$.

