PROGETTO RETI GEOGRAFICHE 2021/2022

Twitter Sentiment Analysis during italian TV show: “Amici 2021”

Docente:

Delfina Malandrino
Rocco Zaccagnino

Componenti gruppo:

Antonio Zizzari
Vincenzo Marrazzo
Simone Giglio

Sommario

1. Abstract

Ogni giorno all'interno di Twitter avvengono moltissimi tweets, ed essi giocano un ruolo fondamentale all'interno degli show televisivi, poiché il voto del pubblico ha un'importanza molto rilevante.

Essendo che le opinioni dei telespettatori su internet sono in qualche modo un indice di gradimento essenziale, valutare se un'opinione è positiva o negativa su un concorrente di "Amici 2021" è l'obiettivo del nostro progetto.

Per una panoramica generale sull'indice di gradimento dei concorrenti se si andasse a identificare manualmente i tweets positivi o negativi risulterebbe un lavoro laborioso se non impossibile. È per questo motivo che attraverso tecniche per l'analisi della linguistica, frutto anche di un lavoro di psicologia, di lemmatizzazione e di machine-learning, si va a sviluppare un classificatore che determini se un tweet su un concorrente è positivo, tendente positivo, neutro, tendente negativo o negativo, cercando di ottenere un'accuratezza quanto prossima al 100%.

2. Introduzione

Con la crescita della popolarità del Web, i post e le opinioni degli utenti hanno un impatto considerevole sull'immagine dello show televisivo, e ovviamente sono frutto di un potenziale guadagno monetario.

Le opinioni derivanti dai tweets tramite la scelta accurata di hashtags mirati al tipo di analisi che si vuole effettuare, sono fondamentali per l'individuazione dei gusti degli utenti nei confronti dei concorrenti del programma. Una scelta sbagliata degli hashtags da ricercare potrebbe portare ad un'analisi non coerente oppure con risultati scarni.

Lo scopo del progetto è comprendere attraverso dei classificatori se un commento è positivo, tendente positivo, neutro, tendente negativo o negativo, utilizzando le conoscenze acquisite al corso di "Reti Geografiche". Il motivo per il quale è stato scelto questo argomento è dovuto alla scarsa quantità di elaborati relativi a tale ambito.

3. Twitter vs Instagram

#1 **Twitter** ha una migliore copertura dell'utilizzo in più categorie di siti web. Tra cui Arte e intrattenimento, Elettronica e tecnologia per computer, Scienza e istruzione, Giochi e altre 20 categorie.

#2 **L'API di Instagram** non ha un vantaggio su Twitter in nessuna categoria di siti Web.

È stato scelto, quindi, Twitter come base per la cattura dei commenti che riguardano il mondo di Amici2021 perché secondo i dati illustrati precedentemente Twitter viene utilizzato maggiormente in diversi categorie di siti.

4. Modalità di Analisi

In questa fase lo studio relativo a quali tecnologie, a quale social media, a come raccogliere i dati e l'analisi statistica di questi ultimi, è significativo ai fini di un'analisi corretta che possa dare dei risultati ottimali per il nostro scopo.

Il social media preso in considerazione è stato **Twitter**; la scelta di tale social è dipesa dal fatto che Twitter mette a disposizione delle API gratuite e semplici da usare, a differenza degli altri social media.

Il linguaggio di programmazione scelto per la cattura, analisi e valutazione è stato Python, in quanto risulta vantaggioso per la semplice implementazione e supporto alla moltitudine di librerie e modelli per il machine-learning, in particolare le librerie Tweepy, Vader e Spacy (progetti open source) ci hanno permesso di elaborare in modo semplificato ed efficace i dati catturati.

Per quanto riguarda la gestione dei dati raccolti, si è optato per il formato ".csv", che ci ha permesso di strutturare ed immagazzinare dati, per poi importare tali file di questo formato sull'ambiente di sviluppo RStudio.

Tramite RStudio, si fanno analisi e statistiche dei dati immagazzinati per evidenziare eventuali situazioni non facilmente intuibili con la semplice visualizzazione dei dati, ma attraverso strumenti grafici fatti ad hoc per il tipo di problema analizzato.

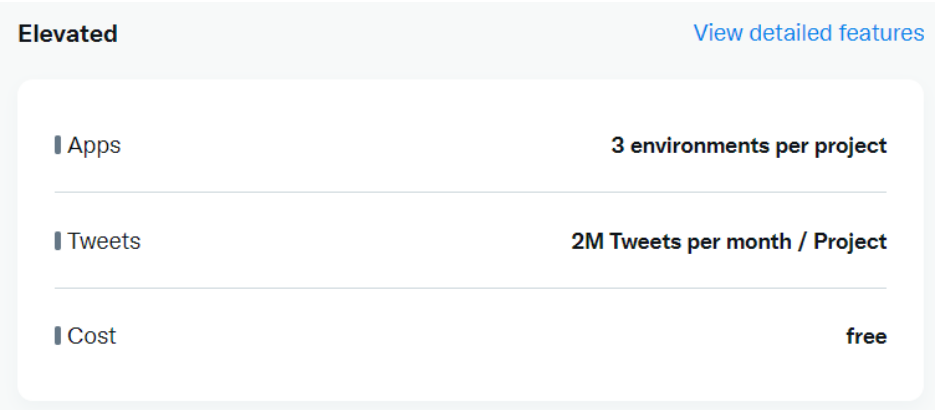
5. Progettazione e Implementazione

In questa fase analizzeremo come e con quali strumenti è stato implementato il nostro problema.

Twitter Capture

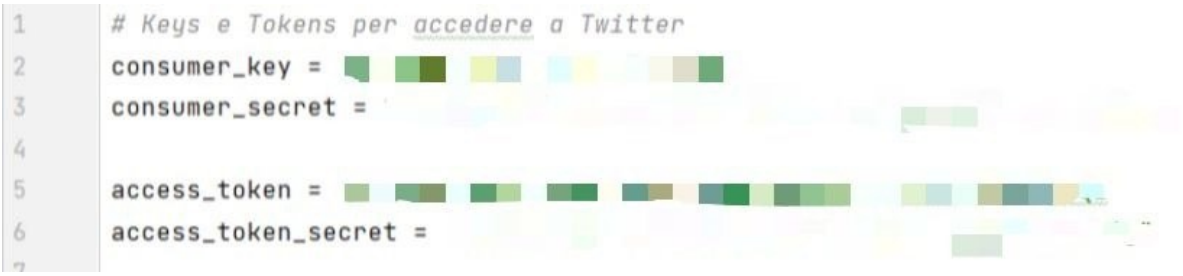
Il primo problema che ci siamo posti è stato: "come catturare i tweets?". Tra i vari tool analizzati, l'uso delle API di Twitter è stato di particolare importanza, infatti l'utilizzo della libreria *tweepy* ci ha permesso di catturare tutti i post su Twitter attraverso delle query specifiche.

Queste API hanno dei limiti imposti da Twitter che sono i seguenti:



Infatti il limite risulta essere di 2mln di catture al mese, e le catture sono limitate anche temporalmente, infatti il limite massimo è di 1 settimana antecedente al giorno corrente.

Come primo approccio abbiamo creato le variabili alle quali sono state associate le credenziali delle API:



In un secondo momento, abbiamo scritto la query che ci ha permesso di catturare i tweets. I criteri di ricerca hanno seguito delle regole ben precise, che sono:

- **Hashtags**: Abbiamo creato uno snippet di codice che prende in input i concorrenti aggiornati in modo dinamico e li mette all'interno della query di *tweepy*, garantendo una maggiore precisione e accuratezza sulla cattura dei tweets.

```
# Inserimento Hashtag da cercare
inputTerm = "#Amici21 AND ("
for i in range(len(concorrenti_amici)):
    if (i < (len(concorrenti_amici)-1)):
        inputTerm += "#" + concorrenti_amici[i] + " OR "
    elif (i == (len(concorrenti_amici)-1)):
        inputTerm += "#" + concorrenti_amici[i] + ")"
```

- **Retweets**: Abbiamo impostato il parametro `-filter:retweets` per filtrare dalla ricerca i retweet, per evitare di avere tweets ridondanti;
- **Giorno limite cattura tweet**: Abbiamo usato il parametro `until` che ci ha permesso di settare il limite temporale per la cattura di un tweet. Nella fase di cattura è importante specificare che la query viene eseguita sui tweet antecedenti alla data in input fino ad una settimana, limiti imposti dalle API;
- **Numero max tweets**: Uno dei limiti che ci impone l'API di Twitter è il numero massimo di tuple che possiamo ottenere da una query. Il limite è fissato a 100 per cattura, anche se si andasse ad aumentare tale limite, la cattura massima rimarrebbe invariata.

Dopo queste analisi preliminari si procede con l'effettiva cattura dei tweets mediante il seguente codice Python:

```
for tweet in tweepy.Cursor(api.search_tweets,q=searchTerm, lang="it",
                           until = date_tweepy,
                           result_type="mixed",
                           tweet_mode="extended").items(100):
    rt_count = tweet.retweet_count
    p2 = ["", tweet.full_text]
    hast = ""
    a = tweet.full_text.count("#")
    if (a >= 0):
        for c in range(a):
            p1 = p2[1].split("#", 1)
            try:
                if (p1[1].find(" ") < 0):
                    hast += "#" + p1[1]
                    hast = hast.replace("\n", "")
                    break
                else:
                    p2 = p1[1].split(" ", 1)
                    hast += "#" + p2[0]
                    hast = hast.replace("\n", "")
            except IndexError:
                hast+="#"+p1[0]
        tweet.full_text = cleanTweet(tweet.full_text)
        hast = cleanHashtags(hast)
        tt = {
            "Id": tweet.id,
            "Screen_name": tweet.user.name,
            "Created_at": tweet.created_at,
            "Retweet": str(rt_count),
            "Text": tweet.full_text,
            "Hashtags": hast,
            "Sentiment": 0,
            "Compound": 0
        }
        tt = Tweet(tweet.id, tweet.user.name, tweet.created_at, str(rt_count), tweet.full_text, hast, 0, 0)
        if (str(tt.getCreated_at()).find(date_to_search) != -1):
            tweets.append(tt)
            tweetsTranslated.append(translatorTweet(tt))
```

In questo snippet di codice è stato possibile ricavare la tupla del tweet, con i criteri sopra citati. Il testo del tweet prima di esser salvato viene elaborato tramite un processo di filtraggio. In questo processo vengono rimosse dal tweet, attraverso i metodi `cleanHashtags()` e `cleanTweet()`, le componenti non adatte come i simboli, i caratteri speciali, stringhe http, ecc…, in modo da ottenere un testo pulito.

Infine, si crea un dizionario che raccoglie tutti gli elementi principali dei tweets e successivamente li si aggiunge in una lista.

Dato che la libreria per i sentiment funziona solamente con il dizionario inglese, per prevenire tale situazione, si traduce in anticipo il tweet, con la funzione `translatorTweet()`.

Una volta ottenuta la nostra lista di tweets tradotti, procediamo con il Sentiment Analysis.

Sentiment Analysis

Una volta eseguita la traduzione del messaggio del tweet, nello stesso snippet, viene avviata la procedura di sentiment, tramite la funzione `analyzer.polarity_scores()`, si fa uso della libreria Vader per ottenere un parametro in output che corrisponde al sentimento del testo passato; successivamente il valore ottenuto in output viene passato al metodo `sentiment()` e tale funzione permette di fornire come output i punteggi relativi a 5 classi di sentiment:

1. Negativo;
2. Tendente Negativo;
3. Neutro;
4. Tendente Positivo;
5. Positivo.

Queste classi vengono restituite e salvate all'interno di un parametro appartenente al dizionario che si stava in antecedenza strutturando.

I codici in Python che permettono tale analisi, sono:

```python
def calculate_and_set_compound_score_to_tweet(tweet):
    analyzer = SentimentIntensityAnalyzer()
    sentiment = analyzer.polarity_scores(tweet.getText())
    compound = sentiment["compound"]
    tweet.setCompound(compound)

def calculate_and_set_sentiment_score_to_tweet(tweet):
    sentimentAnalysis = Sentiment(tweet.getCompound())
    tweet.setSentiment(sentimentAnalysis[0])

def print_emotion_score(tweet):
    sentimentAnalysis = Sentiment(tweet.getCompound())
    print("\033[1m" + "Risultato: " + "\033[0m" + sentimentAnalysis[1])

def Sentiment(compound):
    emotion = ""
    if(compound <= -0.5501 and compound >= -1):
        sentimentAnalysis = 1
        emotion = "Negativo"
    if(compound <= -0.2001 and compound >= -0.5500):
        sentimentAnalysis = 2
        emotion = "Tendente negativo"
    if(compound >= -0.2000 and compound <= 0.2000):
        sentimentAnalysis = 3
        emotion = "Neutro"
    if(compound >= 0.2001 and compound <= 0.5500):
        sentimentAnalysis = 4
        emotion = "Tentendente positivo"
    if(compound >= 0.5501 and compound <= 1):
        sentimentAnalysis = 5
        emotion = "Positivo"

    return sentimentAnalysis, emotion
```

Lemmatizzatore

Adesso si passa alla 2ª parte del progetto, che include l'analisi grammaticale del tweet passato. Attraverso la libreria Spacy, è possibile avere un'analisi grammaticale del testo passato abbastanza accurata dovuto dal modello di machine-learning che ha una accuratezza seguente:

NAMED ENTITY RECOGNITION SYSTEM	ONTONOTES	CONLL '03
spaCy RoBERTa (2020)	89.8	91.6
Stanza (StanfordNLP)[1]	88.8	92.1
Flair[2]	89.7	93.1

Poiché da essa è possibile ricavare il soggetto o i soggetti del testo in questione, è stato possibile associare uno score ai concorrenti del programma televisivo.

Il modello, durante l'analisi, è stato allenato, per renderlo ad hoc riguardo il problema analizzato (sfruttando la tecnologia CUDA di NVIDIA).

Lo snippet di codice che ha permesso tale implementazione è il seguente:

```
# SECONDA PARTE DEL CODICE
print("Analisi più dettagliata dei Tweet")
nlp = spacy.load("it_core_news_sm")

print(concorrenti_amici)
for tweet in tweets:
    amici_objects_clone = copy.deepcopy(concorrenti_amici_objects)
    doc = nlp(tweet.getText())
    print("TWEET: ")
    print("Testo tweet: " + tweet.getText())
    print("Sentiment: " + str(tweet.getSentiment()))

    ents = []
    for entity in doc.ents:
        for concorrente in concorrenti_amici_objects:
            # non mettiamo il break, a fine 'if', poichè all'interno di un soggetto potrebbero esserci più
             concorrenti e quindi vogliamo
            # prima estrapolare tutti i concorrenti dal soggetto e poi andare avanti col prossimo soggetto
            if ((concorrente.getName().upper() in entity.text.upper()) and (concorrente in amici_objects_clone)):
                ents.append(concorrente.getName())
                print("text: " + entity.text, ", concorrente_name: " + concorrente.getName())
                concorrente.addScore(tweet.getSentiment())
                amici_objects_clone.pop(amici_objects_clone.index(concorrente))
                #print("FINE " + "text: " + entity.text, "type/label: " + entity.label_)

    tweet.setEnts(ents)
    print("Ents: " + str(tweet.getEnts()))
    print()
```

Gestione dei dati

I dati dei concorrenti e dei tweets sono stati immagazzinati all'interno di un file formato ".csv". Esso ci servirà più avanti, per effettuare l'elaborazione dei dati attraverso il linguaggio R.

Lo snippet di codice è il seguente:

```
# Nomi dei file
name_file_tweets = "./Dataset_tweets/Amici(15-12-2021).csv"
name_file_concorrenti_score = "./Dataset_tweets/Amici_Score_Concorrenti(15-12-2021).csv"

# 1 parte
print(f"\nWriting in file '{name_file_tweets}' ...")
# Creazione nostro file
csv_file = open(name_file_tweets, 'w', encoding='utf-8', newline='')
writer = csv.writer(csv_file)
head1 = ["Id", "Screen_name", "Created_at", "Retweet", "Text", "Hashtags", "Sentiment", "Compound", "Ents"]
writer.writerow(head1)
# Aggiunta dei tweets nel file CSV
for t in tweets:
    lista_ents = converto_in_list(t.getEnts())
    writer.writerow([str(t.getId()), str(t.getScreen_name()), str(t.getCreated_at()), str(t.getRetweet()), str(t.getText()),
                     str(t.getHashtags()), str(t.getSentiment()), str(t.getCompound()), lista_ents])
csv_file.close()
```

I file che creiamo sono:

- ❖ i dati relativi agli score dei singoli concorrenti;
- ❖ i tweet con le relative informazioni, suddivisi per giorni del mese

6. Analisi in R dei dati Raccolti

Prima di procedere con l'analisi vera e propria i dati sono stati elaborati ulteriormente.

L'elaborazione dei dati ha incluso vari step quali:

- Si è raccolto tutti i file .csv delle catture giornaliere e sono stati uniti in un unico file contenente tutti i tweets (1 mese di cattura) denominato "File_finale.csv" attraverso il seguente snippet di codice:

```python
import csv
import os

array_stampa=[]

# Cattura delle righe dei csv dei tweets 2021
folder_path_tweets_2021 = os.getcwd() + "\\tweets (2021)"
for filename in os.listdir(folder_path_tweets_2021):
    if (filename=="desktop.ini"):
        continue
    print(f"File name: {filename}")
    with open(folder_path_tweets_2021 + "\\" + filename, 'r', encoding='utf-8', newline='') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=",", quotechar='"')
        next(csv_reader)
        for row in csv_reader:
            array_stampa.append(row)
    csv_file.close()

# Cattura delle righe dei csv dei tweets 2022
folder_path_tweets_2021 = os.getcwd() + "\\tweets (2022)"
for filename in os.listdir(folder_path_tweets_2021):
    if (filename=="desktop.ini"):
        continue
    print(f"File name: {filename}")
    with open(folder_path_tweets_2021 + "\\" + filename, 'r', encoding='utf-8', newline='') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=",", quotechar='"')
        next(csv_reader)
        for row in csv_reader:
            array_stampa.append(row)
    csv_file.close()

# Scrittura di tutti i tweets nel file 'File_finale.csv'
csv_file = open("File_finale.csv", 'w', encoding='utf-8', newline='')
writer = csv.writer(csv_file)
head = ["Id", "Screen_name", "Created_at", "Retweet", "Text", "Hashtags", "Sentiment", "Compound", "Ents"]
writer.writerow(head)
writer.writerows(array_stampa)
csv_file.close()
```

Questa unione ci ha permesso di ottenere gli scores complessivi di 1 mese di cattura di tutti i concorrenti lanciando nuovamente l'algoritmo di sentiment analysis e di lemmatizzazione su di esso.

- Creazione di un metodo di ranking per la valutazione degli sfidanti di Amici volto a classificarli in base a come ha reagito il pubblico di Twitter nei loro confronti.

```
56      # Metodi d'appoggio
57      def calculate_rank(self):
58          rank = 0
59          rank += (self.getScoreNegativo() * -1)
60          rank += (self.getScoreTendentePositivo() * -0.5)
61          rank += (self.getScoreNeutro() * 0)
62          rank += (self.getScoreTendentePositivo() * +0.5)
63          rank += (self.getScorePositivo() * +1)
64
65          return rank
66
```

Questo valore è ponderato, facciamo un esempio sul concorrente Alex con i seguenti valori (ottenuti tramite l'applicazione di VaderSentiment sui tweets):

	Negativo	Tend. Negativo	Neutro	Tend. Positivo	Positivo
Alex	10	2	35	23	20

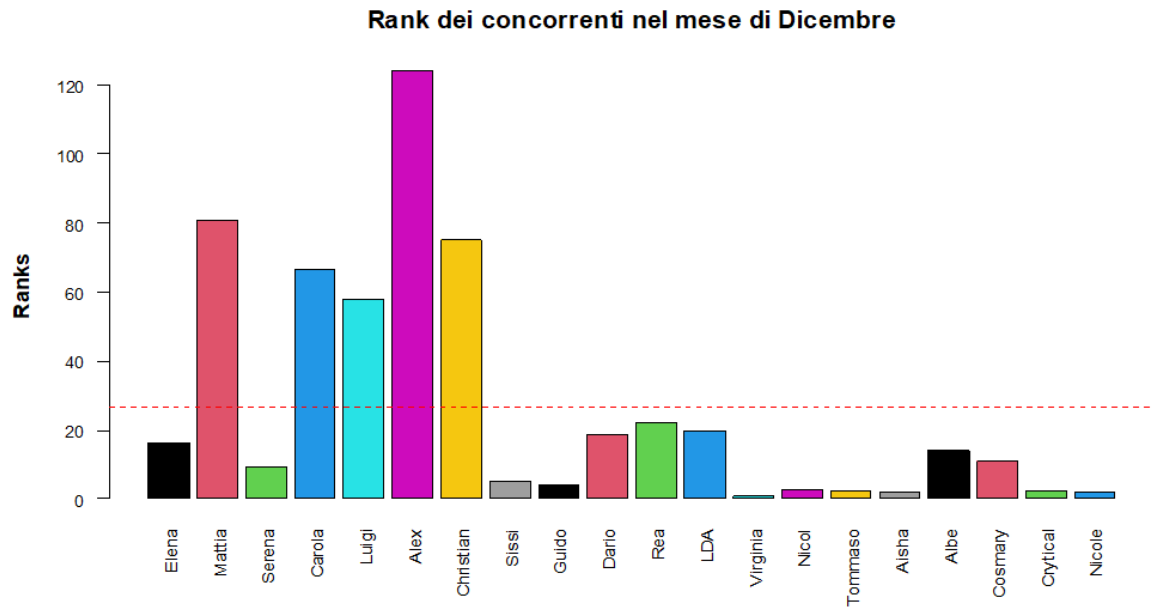
Il calcolo che viene eseguito è il seguente:

$$\boldsymbol{RANK} = \big(\boldsymbol{Negativo} * (-1)\big) + \big(\boldsymbol{Tend.Negativo} * (-0{,}5)\big) + \big(\boldsymbol{Neutro} * (0)\big) + \big(\boldsymbol{Tend.Positivo} * (+0{,}5)\big) + \big(\boldsymbol{Positivo} * (+1)\big) = \mathbf{20{,}5}$$

- E la creazione di un file .csv contenente tutti i punteggi di ranking che verrà analizzata in dettaglio con R usando la retta di regressione con gli scarti complementari.

Indici di Sintesi e Grafici dei Dati

Prima di mostrare uno studio approfondito, è meglio avere una panoramica preliminare dei dati inerenti i punteggi derivati dall'algoritmo di ranking per concorrente che andremo ad elaborare:



Il codice in R che ci ha permesso di ottenere il seguente grafico è:

```r
createBarplot<-function(ranks, main){

  x<-barplot(ranks, ylab="Ranks", main=main,
             col=1:40,
             las=2,
             names.arg=`amici_rank_concorrenti(TOTALE)`[,1:1],
             cex.axis=0.80,
             cex.names=0.80,
             font.lab=2)
  abline(h=mean(ranks),col="red",lty=2,xpd=FALSE)
}

createBarplot(`amici_rank_concorrenti(TOTALE)`[,2:2], "Rank dei concorrenti
              nel mese di Dicembre")
```

Come si può notare vi è una linea rossa tratteggiata che taglia il grafico, essa rappresenta la media, e ci permette di capire fin da subito i partecipanti allo show televisivo che hanno un punteggio ben superiore alla media. Infatti notiamo fin da subito che i concorrenti: Alex, Mattia, Christian, Carola e Luigi sono quelli con i punteggi(Rank) più alti. Successivamente, andremo a studiare i concorrenti che hanno ottenuto un punteggio superiore alla media.

Gli indici di sintesi dei nostri dati sono i seguenti:

Indice	Valore
Media	26.875
Mediana	12.5
Varianza	1201.181
Deviazione standard	34.65806

Tramite essi descriviamo:

- La **media** rappresentante la media aritmetica dei dati;
- La **mediana** che ci aiuta nella statistica descrittiva a capire se i dati sono simmetrici(quando media e mediana coincidono) o meno
- La **varianza** per valutare se le osservazioni analizzate sono dipendenti da uno o più fattori al fine di decidere quali sono i dati più rilevanti
- La **deviazione standard** è l'indice statistico che misura la dispersione delle singole osservazioni di un fenomeno intorno alla media.

Regressione lineare

In statistica si parla di regressione lineare indicando un approccio che modella le relazioni tra una variabile detta dipendente e una o più variabili dette indipendenti.

Il caso in cui la variabile indipendente sia una sola è detto regressione lineare semplice, mentre quando ci sono più variabili indipendenti abbiamo a che fare con la regressione lineare multipla.

Regressione lineare semplice

Nella regressione lineare le relazioni sono modellate usando funzioni di predizione lineare i cui parametri del modello sono stimati dai dati. Questi modelli sono detti modelli lineari.

Il modello di regressione lineare semplice è esprimibile attraverso l'equazione di una retta che riesce meglio di qualunque altra ad interpolare la nuvola di punti.

Data dunque l'equazione

$$Y = \alpha + \beta X$$

Dove

1. **α** è l'intercetta, cioè indica l'ordinata con la quale la retta di regressione si interseca con l'asse delle ordinate
2. **β** è il coefficiente angolare, cioè indica la "pendenza" della retta.
    i. Se positivo allora la retta di regressione è crescente
    ii. Se negativo allora la retta di regressione è discendente
    iii. Se nullo allora la retta di regressione è orizzontale

Questa retta viene ottenuta con il metodo dei minimi quadrati. Per calcolare i coefficienti di regressione è necessario considerare la **somma Q dei quadrati degli errori**

$$Q = \sum_{i=1}^{n} [y_i - (\alpha + \beta x_i)]^2$$

e minimizzarla. Quindi la variabile dipendente y viene "spiegata" attraverso una **relazione lineare della variabile indipendente x** (cioè: α + βx).
Il problema della regressione si traduce nella determinazione di α e β in modo da esprimere al meglio la relazione funzionale tra y e x.
Derivando Q rispetto α e β e ponendo le derivate parziali ottenute a 0, il metodo dei minimi quadrati conduce a:

$$\beta = \frac{S_y}{S_x} r_{xy}, \qquad a = \overline{y} - \beta \overline{x}$$

Notiamo come il coefficiente di correlazione influenza fortemente β e se quest'ultimo è 0, allora la retta è **orizzontale**.

Procediamo dunque con l'analisi delle rette di regressione e residui dei concorrenti di Amici che hanno riscosso un maggiore successo secondo le nostre analisi.

Si prendono in considerazione 2 vettori inerenti ai punteggi(rank) del mese di Dicembre: Il primo contiene i primi

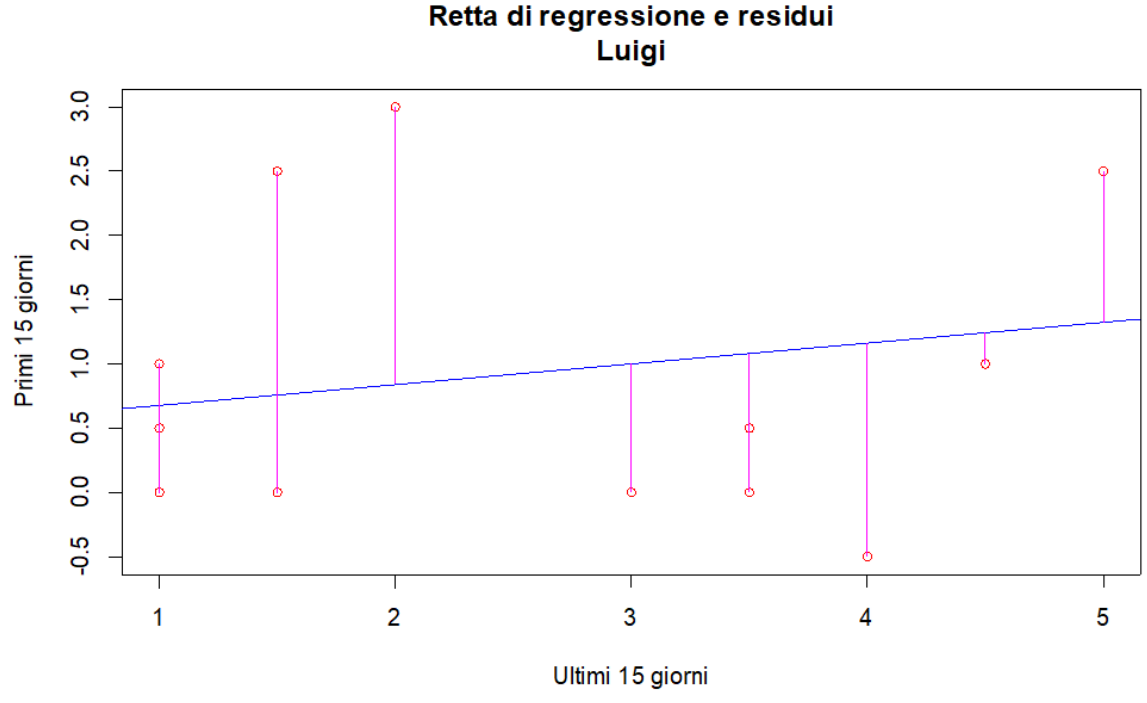
15 giorni del mese e il secondo gli ultimi 15. Ed in base a ciò andremo ad analizzare 2 casi:

1. Se i dati risultano molto dispersi attorno alla retta di regressione, analizzare i fenomeni su tale concorrente risulta essere poco affidabile, poiché frutto di casualità.
2. Mostrare come all'aumentare del tempo, il giudizio del pubblico sul concorrente aumenti positivamente o negativamente.

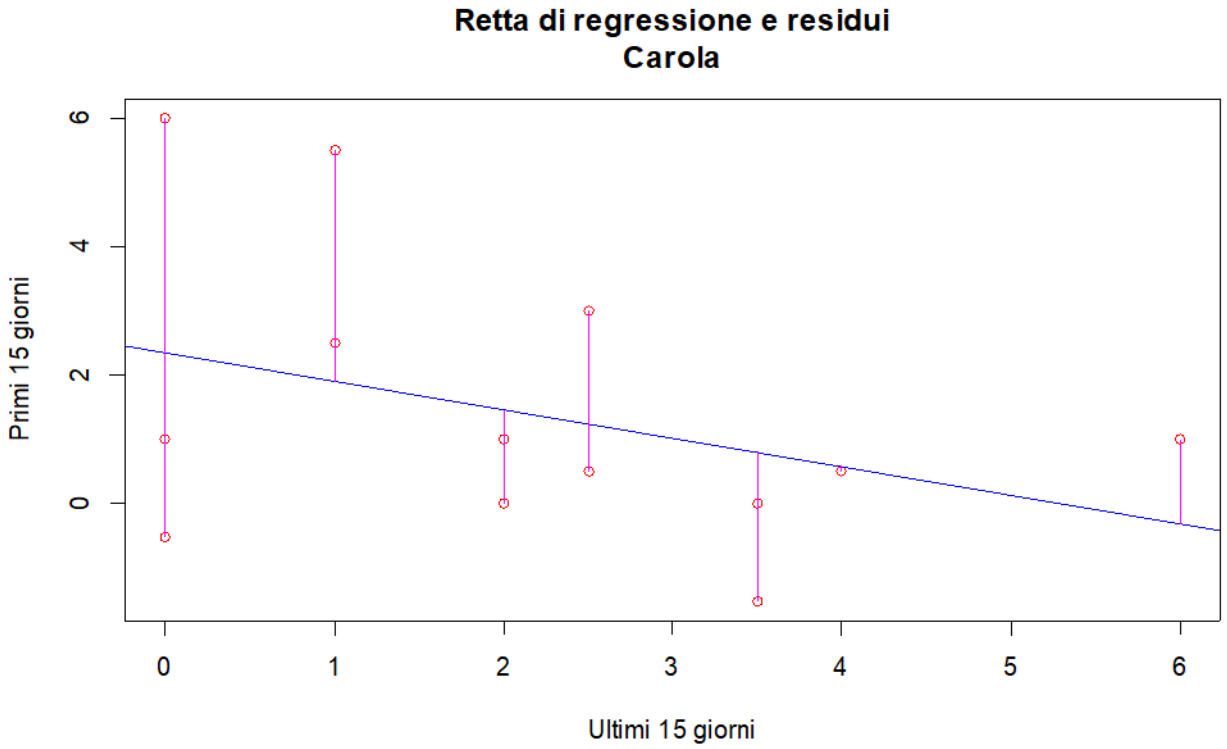
Dato il seguente codice:

```r
function_retta_di_regressione <- function(indice_concorrente, main) {
  matrix <- as.matrix(score_days)
  concorrente<-indice_concorrente
  start_column<-3
  end_column <- 32
  first_15_rank <- c()
  last_15_rank <- c()
  for (start_column in start_column:end_column) {
    if (start_column<=17) {
      first_15_rank = c(
        first_15_rank,
        as.double(matrix[[concorrente:concorrente,start_column:start_column]])
        )
    } else {
      last_15_rank = c(
        last_15_rank,
        as.double(matrix[[concorrente:concorrente,start_column:start_column]])
      )
    }
    start_column = start_column + 1
  }

  x1 <- first_15_rank
  x2 <- last_15_rank

  plot(x1,x2,main=main,
       xlab="Ultimi 15 giorni",ylab="Primi 15 giorni",
       col="red")
  abline(lm(x2~x1),col="blue")
  stime<-fitted(lm(x2~x1))
  segments(x1,stime,x1,x2,col="magenta")
}

function_retta_di_regressione(5, "Retta di regressione e residui\nLuigi")
function_retta_di_regressione(4, "Retta di regressione e residui\nCarola")
function_retta_di_regressione(7, "Retta di regressione e residui\nChristian")
function_retta_di_regressione(2, "Retta di regressione e residui\nMattia")
function_retta_di_regressione(6, "Retta di regressione e residui\nAlex")
```

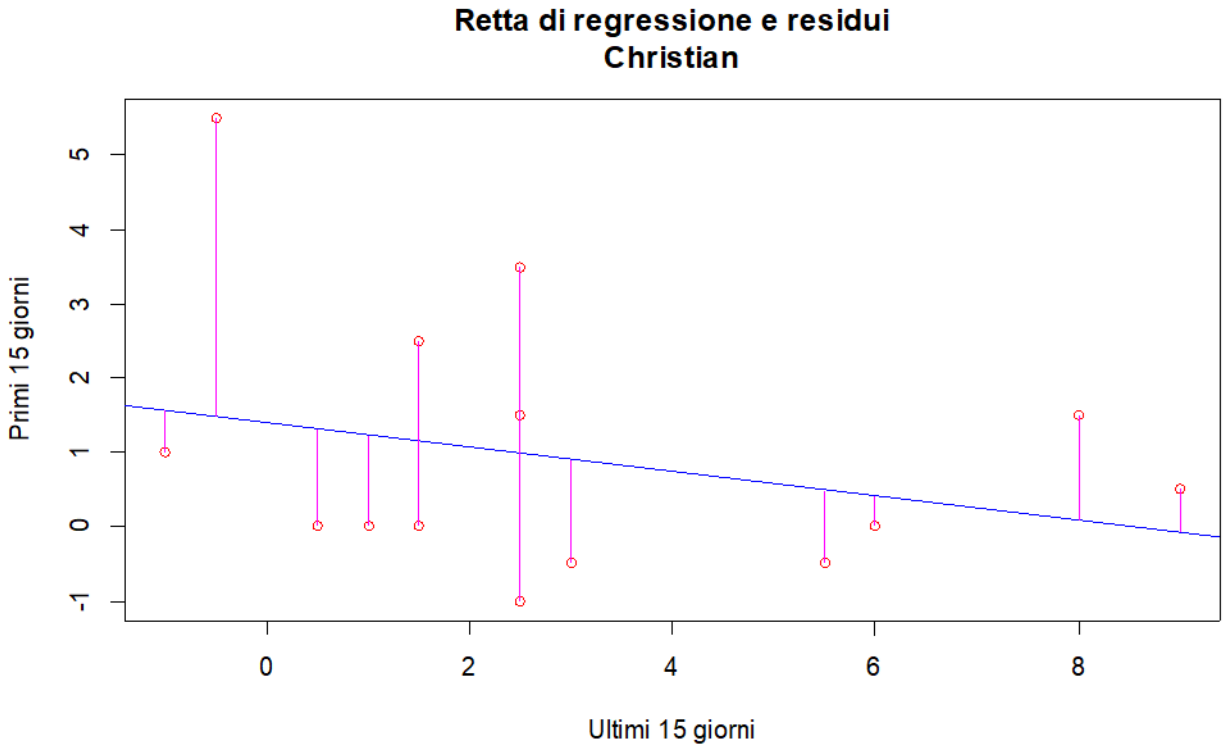
Otteniamo le seguenti stampe:



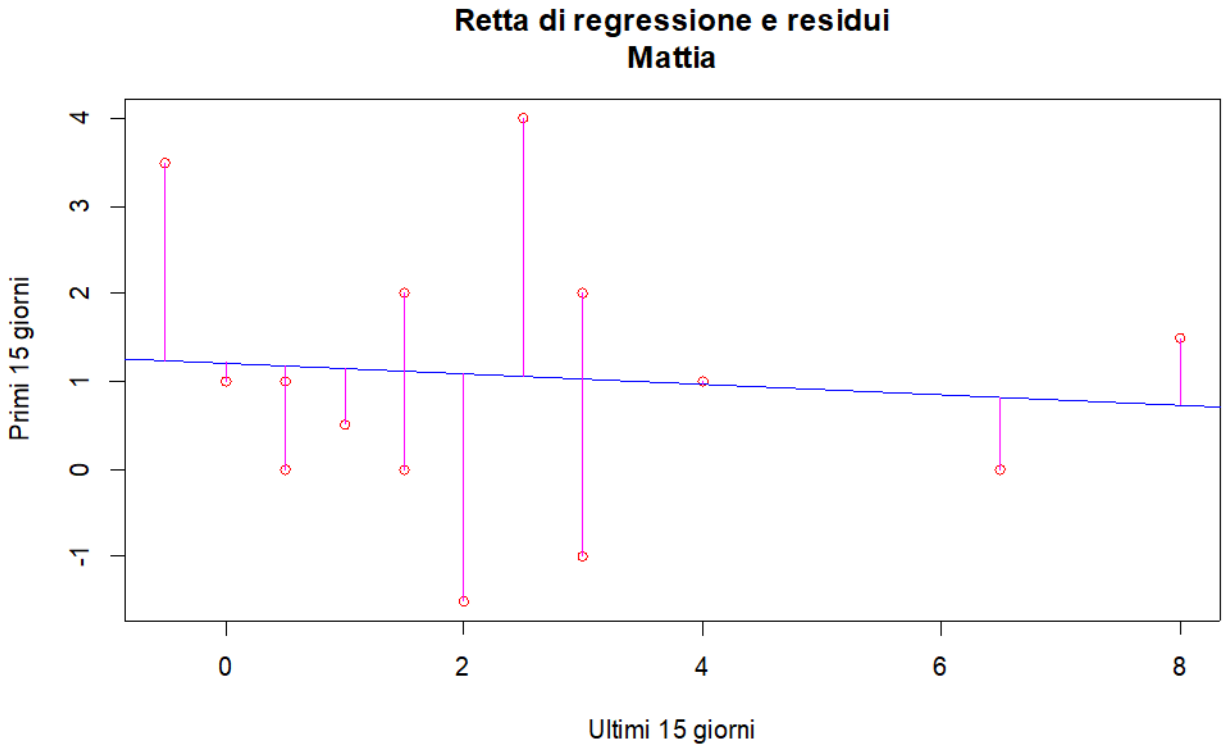
Possiamo affermare che su Twitter Luigi negli ultimi giorni del mese è stato valutato più positivamente.



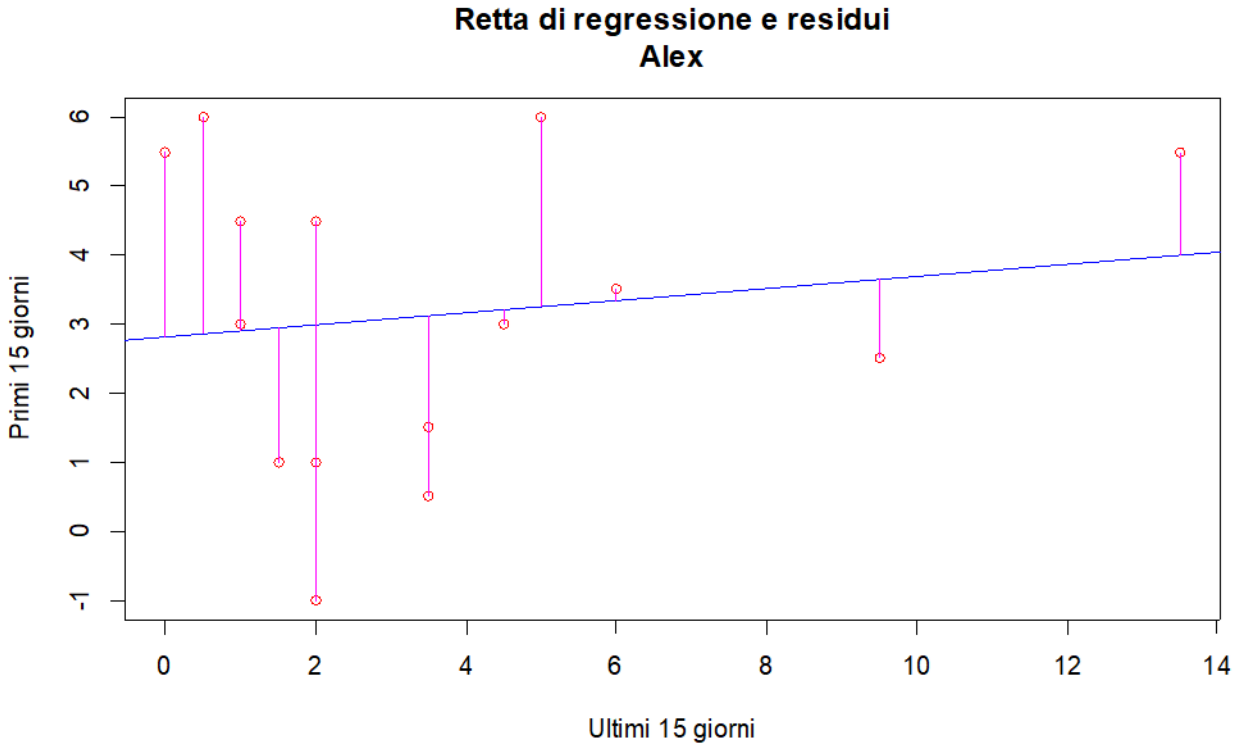
Possiamo affermare che su Twitter Carola negli ultimi giorni del mese è stata valutata negativamente.



Possiamo affermare che su Twitter Christian negli ultimi giorni del mese è stato valutato più negativamente.



Possiamo affermare che su Twitter Mattia negli ultimi giorni del mese è stato valutato leggermente più negativo.



Possiamo affermare che su Twitter Alex negli ultimi giorni del mese è stato valutato più positivamente.

7. Conclusioni

Questo progetto ci ha permesso di estendere le nostre conoscenze sul: Machine Learning, Python, Linguaggio R ed il sentiment analysis. Senza questo progetto non avremmo potuto approfondire le tematiche sopra citate.

Nonostante qualche difficoltà, è stato comunque possibile raccogliere, salvare ed analizzare dati ricavando osservazioni molto interessanti che danno una visione più ampia sui dati raccolti.

Analizzando tutti gli output, forniti dal nostro studio, ne evince come il concorrente di Amici Alex abbia un tasso di popolarità molto più elevato rispetto agli altri partecipanti del talent show, e quindi con quasi certezza finirà per entrare al serale di Amici.

Altri concorrenti che risultano avere punteggi interessanti risultano essere Mattia e Luigi.

8. Riferimenti

- Api Tweeter: https://developer.twitter.com/en/docs
- Vader: https://pypi.org/project/vaderSentiment/
- Spacy: https://spacy.io/
- Cuda: https://developer.nvidia.com/cuda-downloads
- RStudio: https://docs.rstudio.com/