UNIVERSITA’ DI PISA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dipartimento di Informatica

Tesi conclusiva del master in

Big Data Analytics & Social Mining

Anno 2017

ANALISI DEL DATASET: STRAGI NAZIFASCISTE SUL TERRITORIO ITALIANO

Presentato da: Tutor:

STEFANO GHELARDINI Prof. SALVO RINZIVILLO

Indice

[IL DATASET 4](#_Toc535664746)

[Descrizione del dataset 4](#_Toc535664747)

[Utilizzare Python per connettersi al dataset 4](#_Toc535664748)

[Informazioni che possiamo ottenere dalla tabella dba\_\_stragi 6](#_Toc535664749)

[Numero di vittime al giorno 7](#_Toc535664750)

[vittime di ogni giorno 7](#_Toc535664751)

[Vittime complessive in un certo giorno 8](#_Toc535664752)

[Numero di eventi (stragi) giornalieri 10](#_Toc535664753)

[Confronto grafico tra numero di vittime e numero di eventi 11](#_Toc535664754)

[Correlazione eventi-vittime 12](#_Toc535664755)

[Visualizzazione della posizione delle stragi 14](#_Toc535664756)

[Sequenza delle stragi 15](#_Toc535664757)

[Intervallo: 1 luglio 1943, 8 settembre 1943 16](#_Toc535664758)

[Intervallo: 8 settembre 1943, 31 dicembre 1943 17](#_Toc535664759)

[Intervallo: 1 gennaio 1944, 30 marzo 1944 18](#_Toc535664760)

[Intervallo: 1 aprile 1944, 30 giugno 1944 19](#_Toc535664761)

[Intervallo: 1 luglio 1944, 30 settembre 1944 20](#_Toc535664762)

[Intervallo 1 ottobre 1944, 31 dicembre 1944 21](#_Toc535664763)

[Intervallo: 1 gennaio 1945, 30 marzo 1945 21](#_Toc535664764)

[Intervallo: 1 aprile 1945, 4 maggio 1945 23](#_Toc535664765)

[Intervalli temporali in cui si verifica il numero massimo di eventi 24](#_Toc535664766)

[Come funziona l’algoritmo 24](#_Toc535664767)

[Poiché a noi interessa identificare il numero massimo di eventi, le due soglie saranno: soglia di inizio, di valore alto; soglia di fine, di valore basso. 24](#_Toc535664768)

[Il dataframe 24](#_Toc535664769)

[Applicazione dell’algoritmo 25](#_Toc535664770)

[Posizione delle stragi del primo intervallo avente numero massimo di eventi (1943-09-09, 1943-11-13) 26](#_Toc535664771)

[Posizione delle stragi del secondo intervallo avente numero massimo di eventi (1944-06-06, 1944-10-16) 28](#_Toc535664772)

[Posizione delle stragi del terzo intervallo avente numero massimo di eventi (1945-04-24, 1945-05-04) 29](#_Toc535664773)

[Stragi compiute dalla XMAS 29](#_Toc535664774)

[MACHINE LEARNING: Regole associative 33](#_Toc535664775)

[Cosa sono le regole associative 33](#_Toc535664776)

[supporto della regola A=>B 33](#_Toc535664777)

[Confidenza della regola A=>B 33](#_Toc535664778)

[Come vengono calcolate le regole associative 34](#_Toc535664779)

[Troviamo gli insiemi di oggetti più frequenti (grandi) 34](#_Toc535664780)

[Algoritmo Apriori 34](#_Toc535664781)

[Troviamo le regole associative 35](#_Toc535664782)

[Come applichiamo le regole associative al nostro database 35](#_Toc535664783)

[MACHINE LEARNING: Cluster 38](#_Toc535664784)

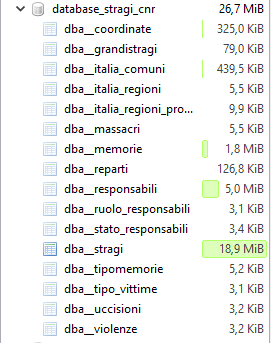
[Algoritmo K-means 38](#_Toc535664785)

[Come applichiamo tale algoritmo al nostro dataset 38](#_Toc535664786)

[Cluster posizione geografica, data, vittime complessive 44](#_Toc535664787)

# IL DATASET

## Descrizione del dataset



Il dataset “database\_stragi\_cnr” contiene le informazioni relative a tutte le stragi avvenute in italia, nel periodo che va dall’armistizio dell’8 settembre 1943 fino alla liberazione, avvenuta il 25 aprile 1945, provocate dalle truppe naziste e fasciste .

In realtà nel database, le stragi iniziano il 1 luglio 1943 (il 10 luglio 1943 gli alleati sbarcano in Sicilia) e terminano il 4 maggio 1945.

Questo database, realizzato molto bene, è in forma normalizzata.

Per quanto riguarda questa tesi, la tabella che prenderemo come riferimento è “dba\_\_stragi” che è formata da 81 colonne.

In particolare, di questa tabella, analizzeremo i seguenti dati: numero di vittime data in cui si verificano le stragi, posizione geografica in cui si verificano le stragi.

Le altre tabelle servono a completare l’informazione fornita da “dba\_\_stragi”.

Vediamo ora come sia possibile trasformare la tabella SQL “dba\_\_stragi” in un data frame della libreria Pandas in modo da poter studiare le caratteristiche di tale tabella.

## Utilizzare Python per connettersi al dataset

Il codice con cui python accede al database è il seguente:

*import MySQLdb*

*import datetime*

*import pandas as pd*

*# Open database connection*

*db = MySQLdb.connect("localhost","root","paperino","database\_stragi" )*

*# prepare a cursor object using cursor() method*

*cursor = db.cursor()*

*sql = '''*

*SELECT ST.datainizio, CO.lat, CO.lng*

*FROM dba\_\_stragi AS ST JOIN dba\_\_coordinate AS CO ON ST.id\_strage=CO.id\_strage*

*ORDER BY ST.datainizio*

*'''*

*cursor.execute(sql)*

*results = cursor.fetchall()*

*lista=list(results)*

il metodo **connect()** genera una connessione con il software che gestisce il database.

Per poter effettuare questa connessione è necessario fornire a tale metodo: l’indirizzo IP della macchina su cui si trova il database (nel nostro caso localhost sta a significare che la macchina su cui si trova il database è la stessa macchina su cui gira il nostro engine di Python), user e password del database, il nome del database.

L’oggetto ritornato dal metodo connect() viene utilizzato per ottenere un cursore (tramite il metodo cursor()).

Il cursore permette sia di eseguire una query (per mezzo del metodo execute()), sia di leggerne i risultati tramite il metodo fetchall().

L’istruzione

*results = cursor.fetchall()*

crea una t-upla di t-uple.

Poiché a noi interessa avere una lista per poi creare un dataframe (un data frame è un oggetto della libreria Pandas e serve per memorizzare tabelle), grazie al costruttore list() creiamo una lista di t-uple a partire dalla t-upla di t-uple..

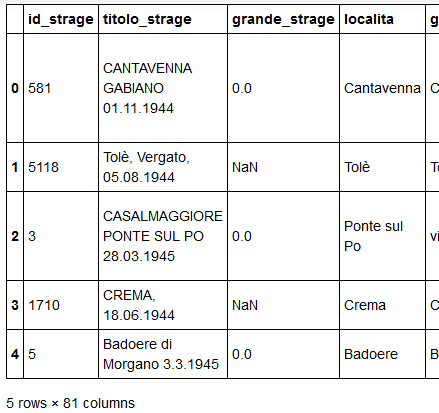
Il data frame viene creato con le seguenti istruzioni:

*import pandas as pd*

*database = pd.DataFrame(lista1,columns=[i[0] for i in cursor.description])*

in particolare la lista (intelligente) *[i[0] for i in cursor.description]* permette di accedere all’oggetto name dell’attributo *description* del cursore.

L’oggetto name contiene i nomi delle colonne che otteniamo grazie all’operazione select di SQL.



## Informazioni che possiamo ottenere dalla tabella dba\_\_stragi

con la seguente istruzione otteniamo il numero di righe della tabella dba\_\_stragi

*numero\_di\_righe = database.shape[0]*

che risulta essere formata da 5764 righe e da 81 colonne (vedi immagine precedente).

Utilizzando invece l’istruzione

*database['vittimetot'].sum()*

valutiamo il numero complessivo dei morti causati dalle stragi che è: 25346.

Un metodo interessante per valutare le caratteristiche di un data frame è describe().

Ad esempio con la seguente istruzione:

*database['vittimetot'].describe()*

otteniamo una serie di valori relativi alla colonna vittimetot che sono:

count 5764.000000

mean 4.397294

std 15.903264

min 0.000000

25% 1.000000

50% 1.000000

75% 4.000000

max 770.000000

**count** ci dice il numero di righe diverse da zero.

**mean** e **std** ci dicono il valore medio e la deviazione standard dei valori della colonna; notare come ad un valore medio basso, corrisponda una deviazione standard molto elevata che sta a indicare una notevole dispersione dei valori e questo si evince considerando che i valori Min e Max variano tra 1 e 770.

**50%** rappresenta la mediana, **25%** e **75%** i percentili.

Utilizziamo adesso nuovamente il metodo sum() per ottenere i valori complessivi delle varie colonne.

*database.sum()*

vittimetot 25346.0

vittimeutot 20302.0

vittimeubambini 561.0

vittimeuragazzi 594.0

vittimeuadulti 15212.0

vittimeuanziani 2282.0

vittimeusenzainfo 1696.0

vittimedtot 3378.0

vittimedbambine 487.0

vittimedragazze 224.0

vittimedadulte 1894.0

vittimedanziane 629.0

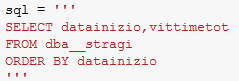
vittimedsenzainfo 151.0

vittimeignote 1482.0

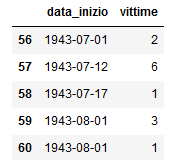
# Numero di vittime al giorno

## vittime di ogni giorno

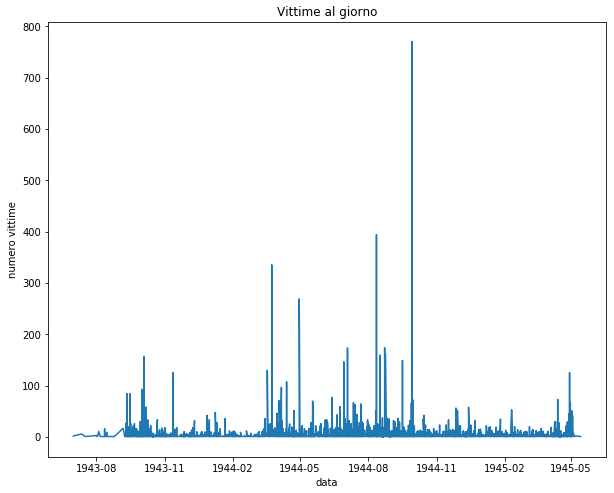
Partiamo dal mondo dei database relazionali e usiamo il linguaggio SQL per estrarre dalla tabella “dba\_\_stragi” i dati relativi alla data di inizio delle stragi e al numero totale delle vittime della strage.



Fatto questo passiamo al mondo di Python ed in particolare alla libreria Pandas, ed otteniamo, dopo una opportuna pulizia dei dati, il seguente Dataframe.



e il seguente grafico:



Questo grafico ci dice, per ogni strage, il numero delle vittime causate da tale strage.

I picchi rappresentano le stragi con il maggior numero di morti che sono state perpetrate in Italia.

## Vittime complessive in un certo giorno

Poiché ogni giorno succedevano in genere più stragi , in diversi parti dell’Italia, vediamo

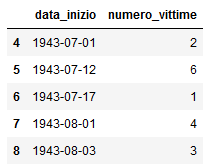
adesso le vittime totali che si verificarono sul suolo italiano, ogni giorno.

Osserviamo che in questo caso, nel linguaggio SQL, abbiamo un groupBy che raggruppa tutti gli eventi che avvengono in una certa data.

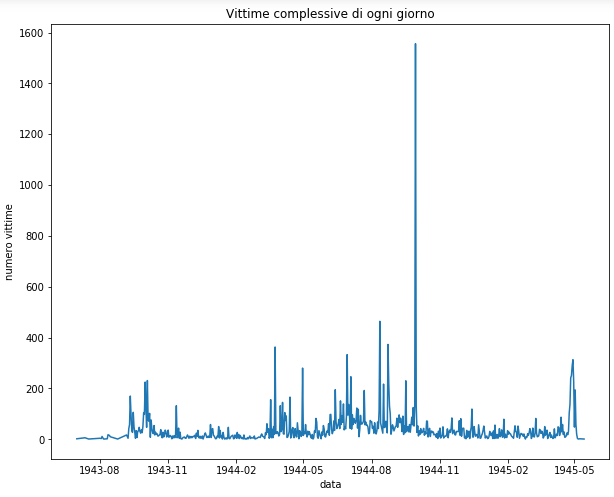
La funzione sum() permette di calcolare il numero complessivo delle vittime che avvengono in Italia.



Otteniamo il seguente dataframe:



ed il seguente grafico:



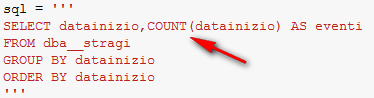
# Numero di eventi (stragi) giornalieri

Adesso vogliamo valutare l’attività complessiva che si svolge in Italia durante la guerra.

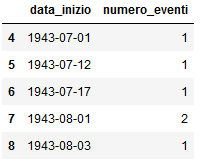
Per fare questo valutiamo il numero di eventi che si svolgono giorno per giorno.

Più eventi implica maggiore attività sul territorio nazionale, meno eventi, meno attività.

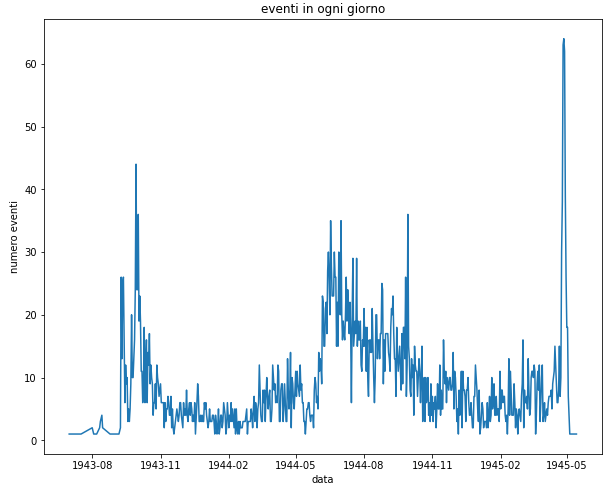
Per ottenere questo, in SQL, operiamo in modo simile al caso precedente, solo che al posto della somma delle vittime conteggiamo il numero di date contenute nei gruppi che otteniamo con il groupby.



Otteniamo il seguente dataframe:



ed il seguente grafico:



Dal grafico si nota come esistano periodi di attività più intensa (cioè con un numero maggiore di eventi) alternati ad altri di intensità minore.

Si nota inoltre una grande variabilità delle attività tra un giorno ed un altro.

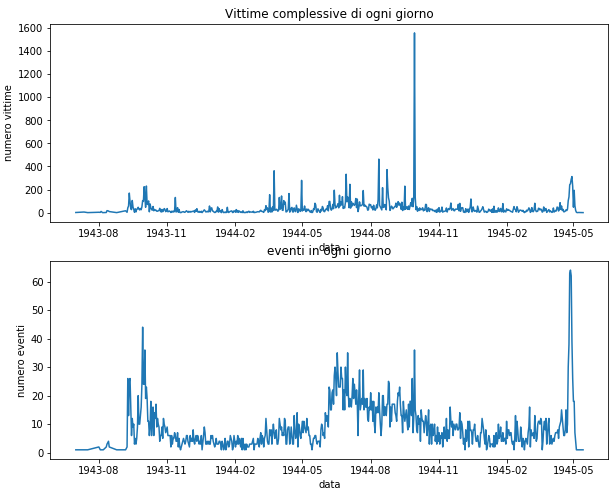
Per valutare questa variabilità, calcoliamo il valor medio e la deviazione standard:



Il valore della deviazione standard ( 7.94 ) è molto vicino al valore della media ( 9.28 ) questo sta a significare grande variabilità dei dati.

# Confronto grafico tra numero di vittime e numero di eventi

Mettiamo insieme il grafico degli eventi e delle perdite:



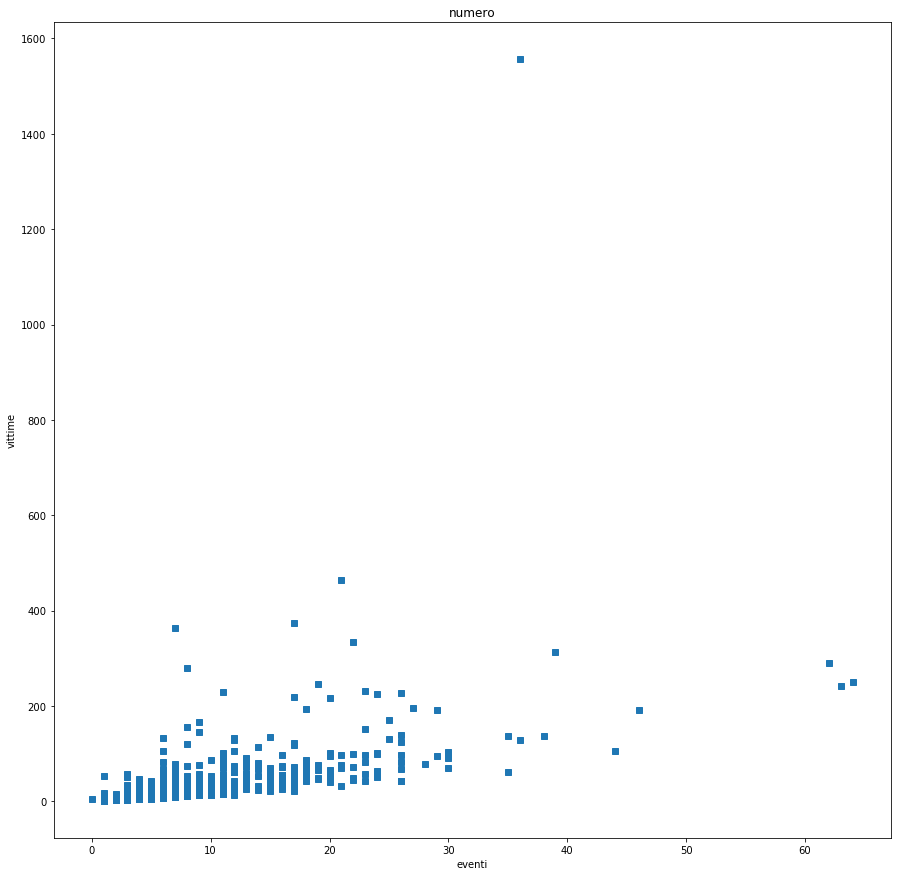
# Correlazione eventi-vittime

Studiamo adesso se esiste una correlazione tra il numero di eventi che si verificano in una giornata ed il numero delle vittime.

In particolare noi vogliamo stabilire se all’aumentare del numero degli eventi aumenta il numero delle vittime (oppure viceversa).

Per vedere se esiste questa correlazione tra eventi e vittime creiamo un grafico con in ascisse il numero degli eventi ed in ordinate il numero delle vittime giornaliere

Il grafico che otteniamo è il seguente:



dal grafico possiamo osservare che esiste una correlazione tra il numero degli eventi e il numero delle vittime cioè al crescere del numero degli eventi cresce il numero delle vittime.

Possiamo notare che in massima parte i punti ottenuti sul grafico sono contenuti in una zona compresa tra 30 eventi giornalieri e 150 vittime.

Esiste un unico punto in cui abbiamo circa 1600 morti con un numero di eventi relativamente basso.

Questo punto rappresenta la strage di Marzabotto.

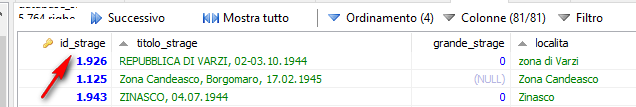
Notiamo come, oltre a questa strage esistono altri punti che rappresentano stragi di grosse dimensioni avvenute sul territorio italiano e queste stragi si riconoscano sul grafico poiché sono rappresentate da punti in cui il numero di morti è superiore a 200 mentre il numero di eventi rimane relativamente basso (parte sinistra e verso l’alto del grafico).

# Visualizzazione della posizione delle stragi

Vediamo adesso di visualizzare la posizione geografica di tutti gli eventi che si verificano nel periodo considerato.

Per poter visualizzare la posizione geografica degli eventi, utilizziamo 2 tabelle del database:

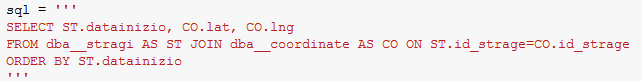
la tabella principale “dba\_\_stragi”



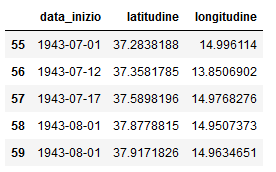
e la tabella “dba\_\_coordinate”



unite in join tramite l’attributo id\_strage.



Il dataframe che otteniamo, visualizza la data dell’evento e la posizione geografica di dove è avvenuto.





# Sequenza delle stragi

Nel grafico precedente abbiamo visto l’insieme delle stragi sul territorio nazionale nel periodo che va dall’8 settembre 1943 fino al 25 aprile 1945.

Adesso vediamo la disposizione geografica delle stragi al passare del tempo, confrontata con le stragi complessive.

In particolare suddividiamo il periodo temporale in cui avvengono le stragi, che è di circa 2 anni,

in intervalli di tre mesi l’uno.

Scegliamo tre mesi perché rappresentano le stagioni di un anno e le guerre in genere seguono le stagioni (maggiore attività durante la primavera e l’estate, e minore attività durante l’inverno).

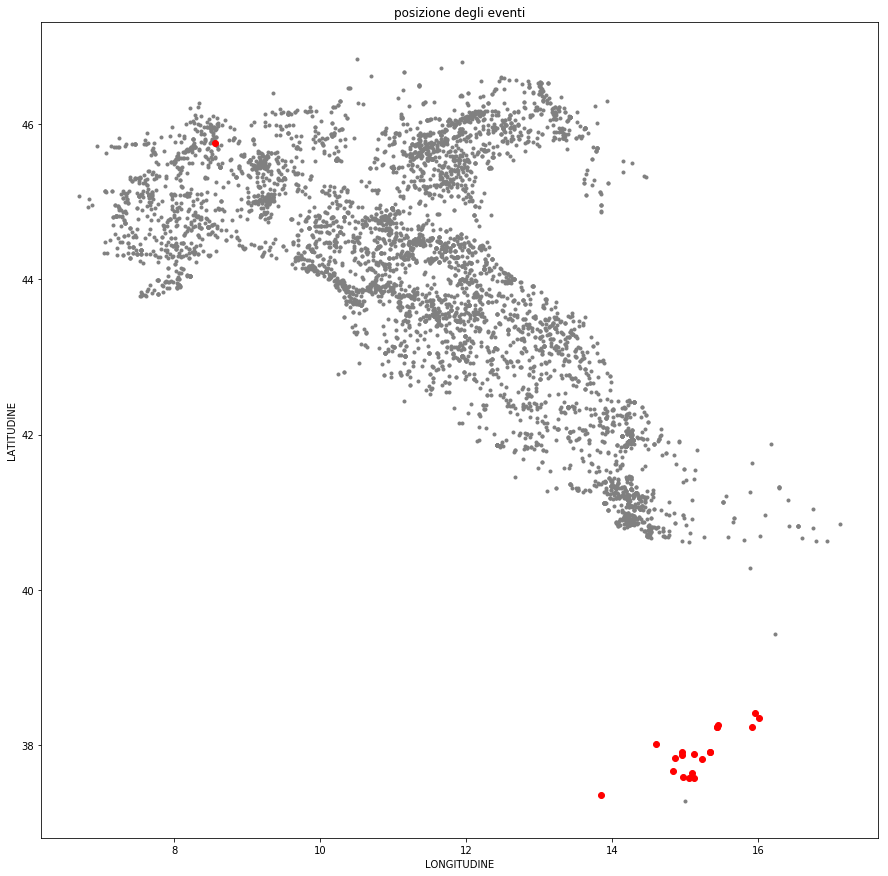
Per semplicità non consideriamo l’esatto inizio della stagione (giorni 21, 23 eccetera) ma consideriamo intervalli di tre mesi che partono dal primo giorno del mese e finiscono al trentesimo giorno del terzo mese.

Ad esempio per quanto riguarda l’inverno del 44, non consideriamo l’intervallo di date comprese tra il 23 dicembre 1943 e il 21 marzo 1944 , ma consideriamo l’intervallo che va tra il 1 gennaio del 1944 e il 30 marzo del 1944.

L’unica eccezione la facciamo per il primo periodo in quanto consideriamo come data limite dei primi 2 intervalli l’8 settembre 1943 che rappresenta la data dell’armistizio dell’Italia con gli alleati.

Quindi nei primi 2 intervalli temporali consideriamo le stragi che avvengono tra il 1 luglio del 1943 e l’8 settembre del 1943 e poi come 2º intervallo le stragi che avvengano tra l’8 settembre 1943 e il 31 dicembre del 1943.

### Intervallo: 1 luglio 1943, 8 settembre 1943

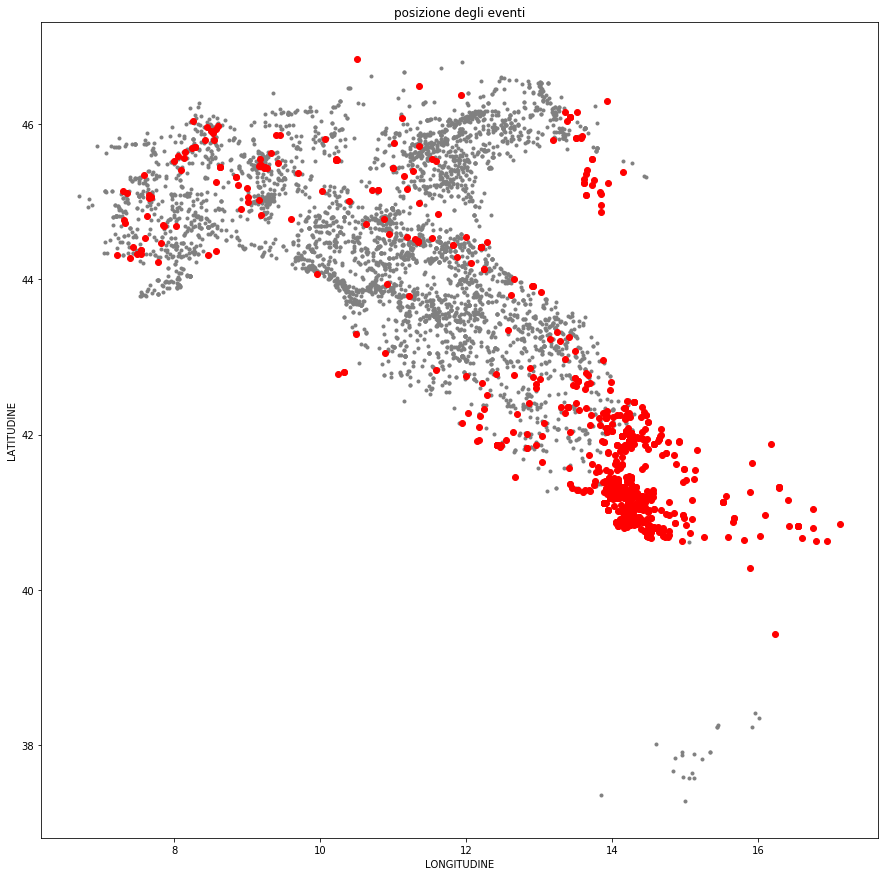


La prima data presente nel database è relativa ad una strage la cui data di inizio risale al 1 luglio del 1943.

In realtà lo sbarco degli alleati avviene il 10 luglio del 1943 e questa prima strage, datata 1 luglio del 1943, in realtà avviene successivamente, durante la battaglia della piana di Catania combattuta tra gli eserciti dell’asse e quello inglese.

Ad ogni modo faccio partire l’intervallo da questo evento anche se, tale evento, è avvenuto dopo.

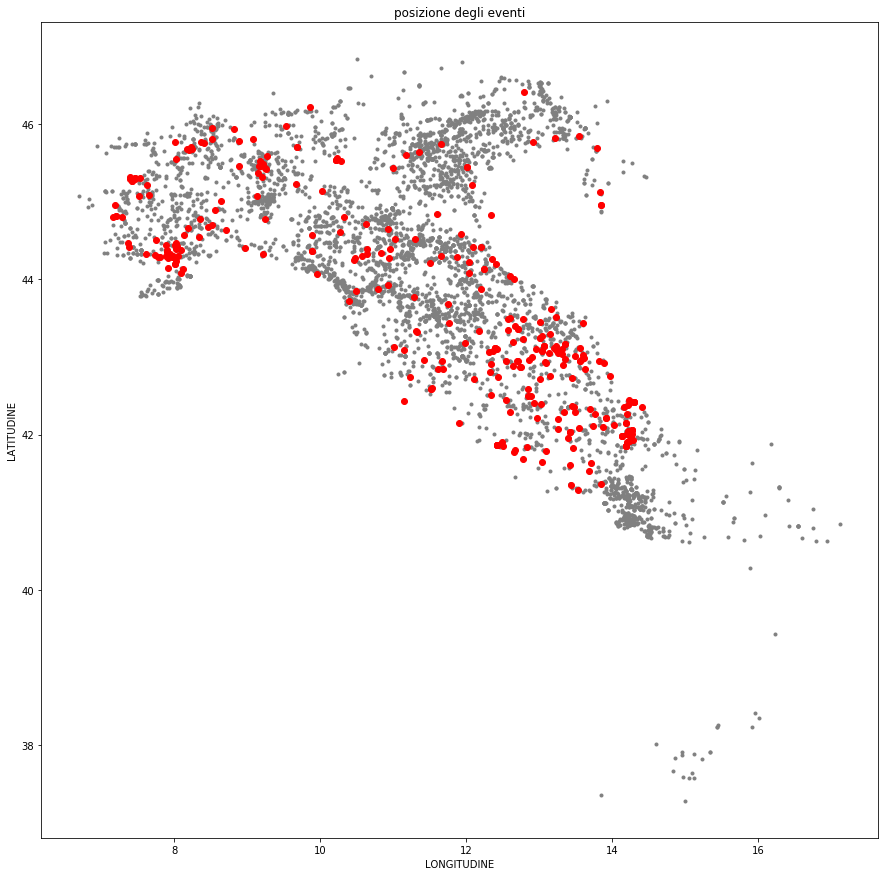
### Intervallo: 8 settembre 1943, 31 dicembre 1943



in questo intervallo di tempo, si nota una forte attività nella regione Campania.

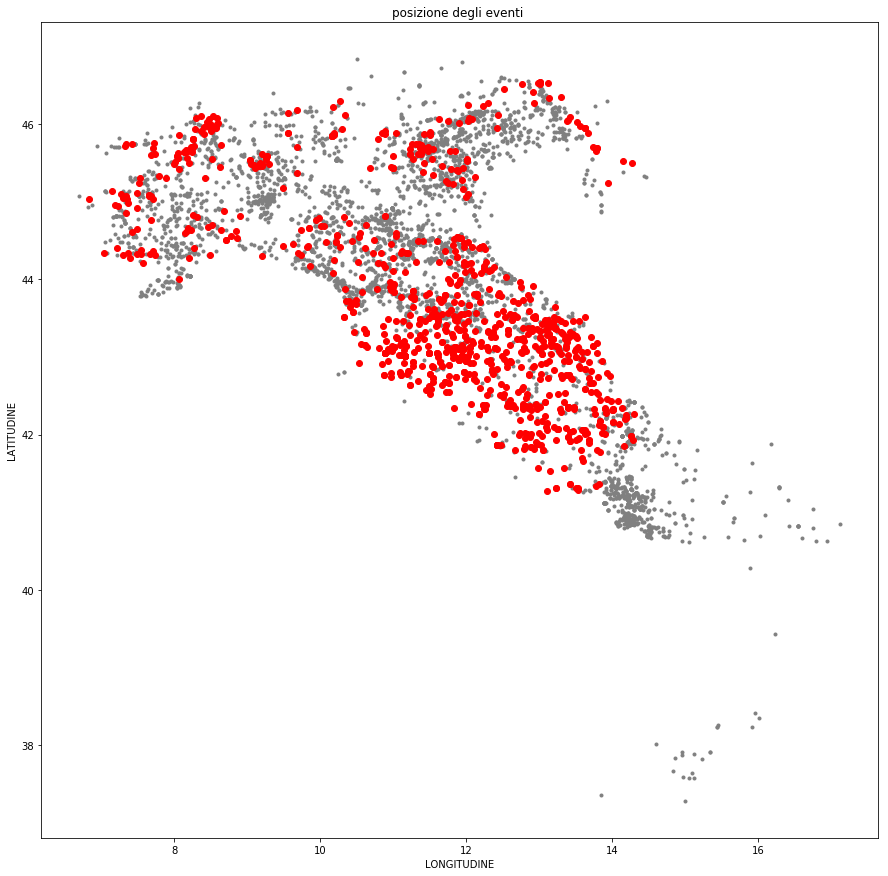
Questa attività probabilmente è legata alle quattro giornate di Napoli che permisero alla città di liberarsi dall’occupazione nazifascista.

### Intervallo: 1 gennaio 1944, 30 marzo 1944



notiamo che in questo periodo il numero delle stragi è ripartito su tutto il territorio ancora occupato ma tale numero è piuttosto basso probabilmente ciò è dovuto al fatto che siamo nel periodo invernale e quindi le attività militari subiscono un rallentamento.

### Intervallo: 1 aprile 1944, 30 giugno 1944

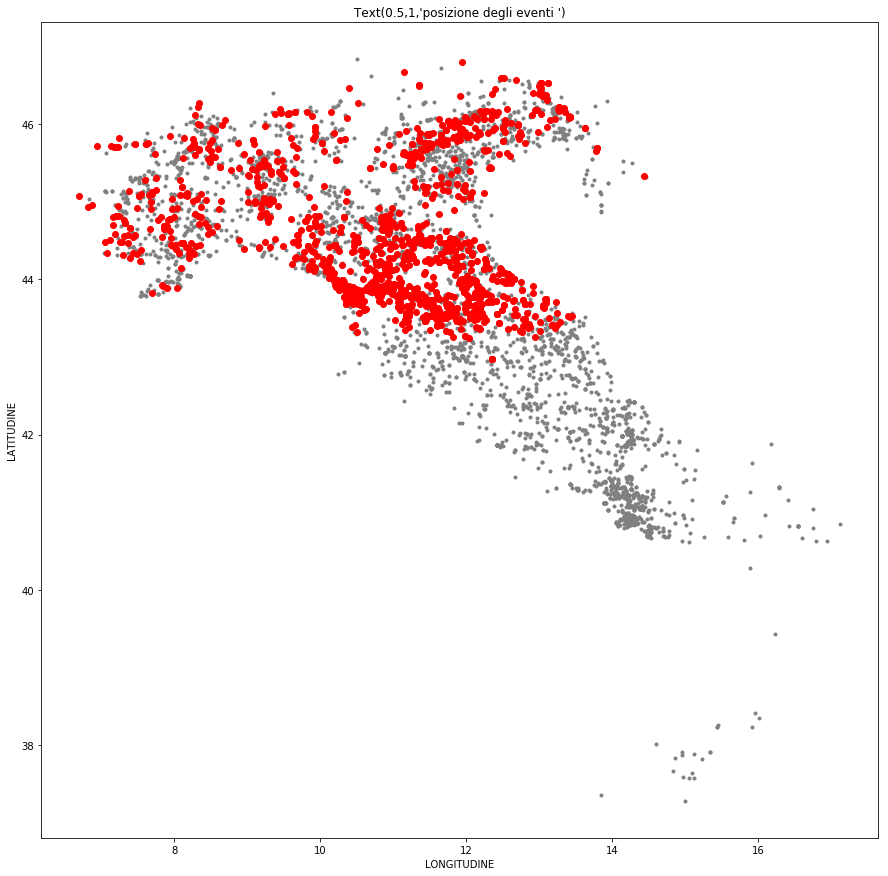


Osserviamo che in questo intervallo di tempo, grazie al fatto di essere in estate, l’attività è molto intensa specie sull’Appennino nell’Italia del nord est.

In questo periodo gli eserciti alleati sono bloccati dalle varie linee difensive che fanno riferimento a Montecassino.

A causa di questo i tedeschi e i fascisti hanno avuto modo di riorganizzarsi.

### Intervallo: 1 luglio 1944, 30 settembre 1944

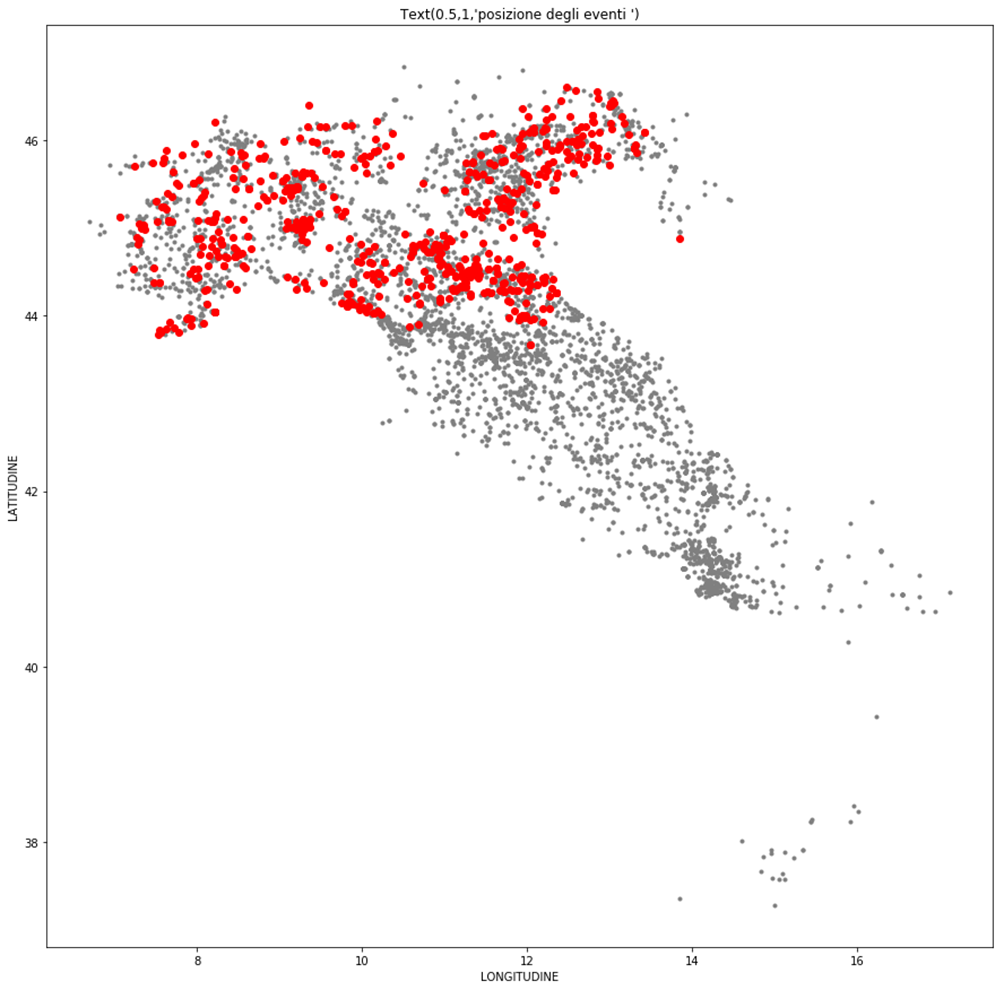


notiamo che in questo intervallo di tempo, nonostante l’inverno, l’attività è intensa.

Gli alleati sono riusciti a sfondare le linee di Montecassino e a liberare Roma in conseguenza di ciò la superficie dell’Italia occupata dalle truppe nazifascista, si è ridotto fortemente, inoltre le forze partigiane hanno avuto modo di organizzarsi soprattutto sugli Appennini, sulle Alpi, oppure nelle grosse città.

Vediamo quindi che le stragi sono presenti in quasi tutte le parti dell’Italia centro-settentrionale.

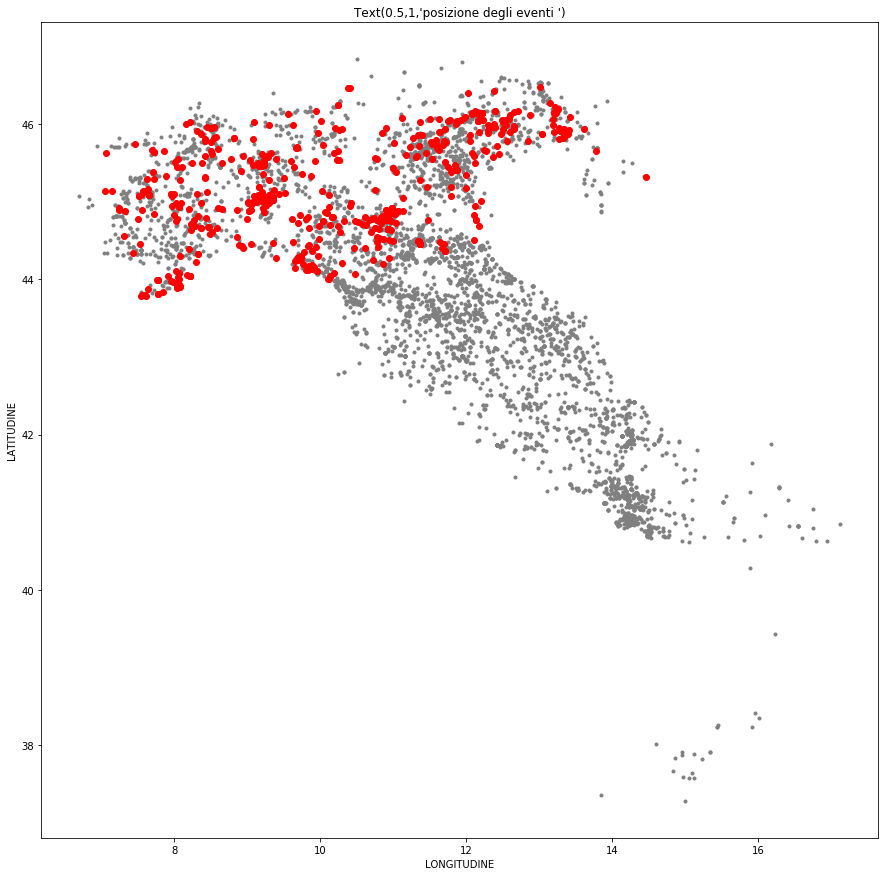
### Intervallo 1 ottobre 1944, 31 dicembre 1944



### Intervallo: 1 gennaio 1945, 30 marzo 1945

la linea del fronte si attesta sulla linea gotica che passa intorno a Bologna.

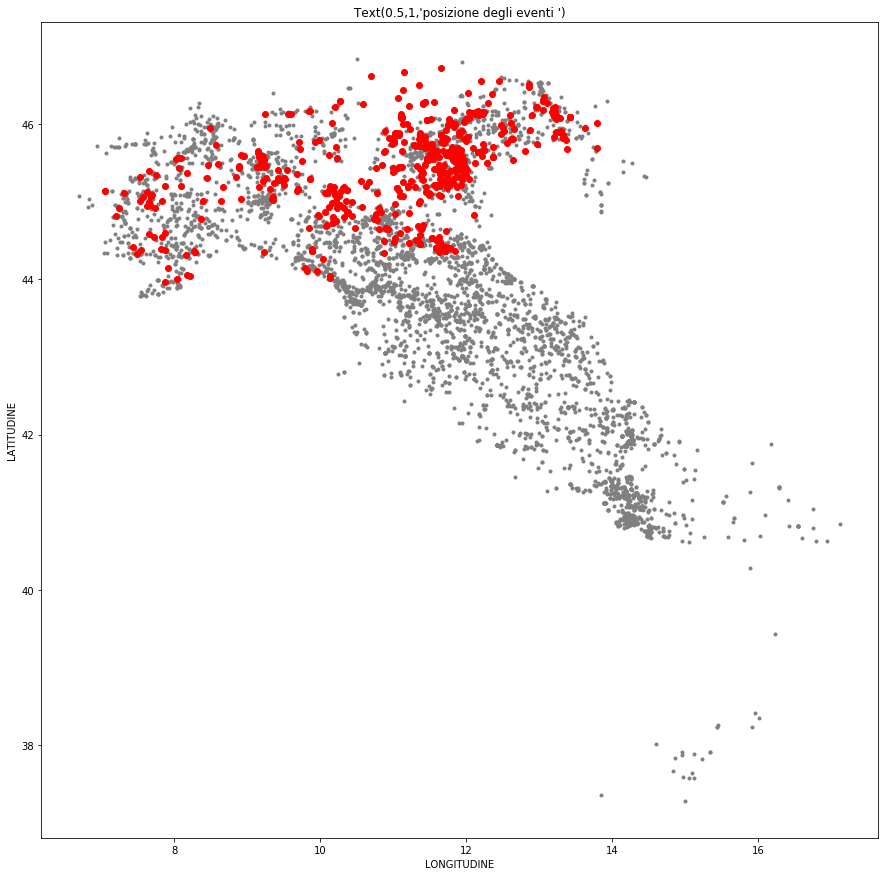
Nonostante l’inverno, le stragi continuano su tutta la parte nord dell’Italia.



La linea gotica cede.

Nonostante questo l’attività delle stragi continua in tutto il Nord.

### Intervallo: 1 aprile 1945, 4 maggio 1945



la guerra in Europa si avvia alla conclusione ma in Italia, le stragi continuano.

Notiamo infatti un numero rilevante di stragi che avvengono nella zona nord-est (prossima a Salò sede di alcuni ministeri della Repubblica Sociale Italiana dopo l’armistizio) e sull’Appennino tosco emiliano.

Osserviamo che alcune stragi avvengono anche dopo la data del 25 aprile 1945 riconosciuta come la data della liberazione ovvero della fine dell’occupazione nazifascista.

In realtà il 25 aprile 1945 non rappresenta la fine dei combattimenti in Italia ma la liberazione di città importanti come Milano.

Di fatto i combattimenti continueranno oltre questa data a causa di reparti tedeschi che si stanno ritirando verso la Germania.

# Intervalli temporali in cui si verifica il numero massimo di eventi

Vogliamo trovare gli intervalli di date (tempo) in cui si verifica l’attività più intensa.

Per fare questo, ci affidiamo al computer creando un algoritmo apposito.

## Come funziona l’algoritmo

L’algoritmo si basa sul principio dell’isteresi, cioè utilizza 2 soglie diverse.

Una soglia è utilizzata per determinare l’inizio dell’intervallo e l’altra è utilizzata per trovare la fine dell’intervallo.

### Poiché a noi interessa identificare il numero massimo di eventi, le due soglie saranno: soglia di inizio, di valore alto; soglia di fine, di valore basso.

### 

Abbiamo visto però che i dati presentano una notevole variabilità.

Di questa cosa bisogna tenerne conto quando si crea l’algoritmo.

Consideriamo quindi un generico evento che si verifica in una generica data.

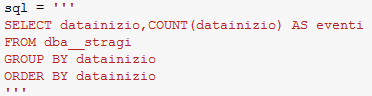
Per determinare se tale evento genere l’innesco (o la chiusura) di un intervallo, oltre alla soglia è necessario valutare il valore medio di una serie di eventi successivi a quello da noi considerato.

Solo se tale media è superiore ad un certo valore (scelto da noi) l’algoritmo innesca (o chiude) l’intervallo.

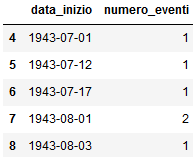
La media serve ad evitare che, a causa della grande variabilità dei dati, gli intervalli ottenuti siano molto piccoli, quindi del tutto inutilizzabili per la nostra analisi.

### Il dataframe

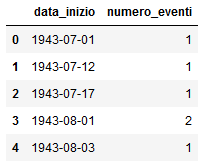
Per quanto riguarda il data frame su cui applicheremo il nostro algoritmo, è quello visto in precedenza per il calcolo del numero degli eventi quindi avremo il seguente codice SQL:



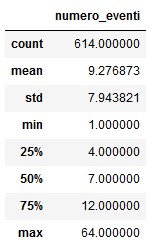
Che dopo l’opportuna pulizia, darà origine al seguente dataframe:



Poiché voglio che l’indice del data frame parta da zero, effettuo su questo data frame una operazione di reindicizzazione, ottenendo il seguente data frame



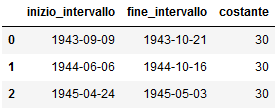
A questo punto utilizzo la funzione describe() per vedere tutte le grandezze statistiche che caratterizzano questo data frame.



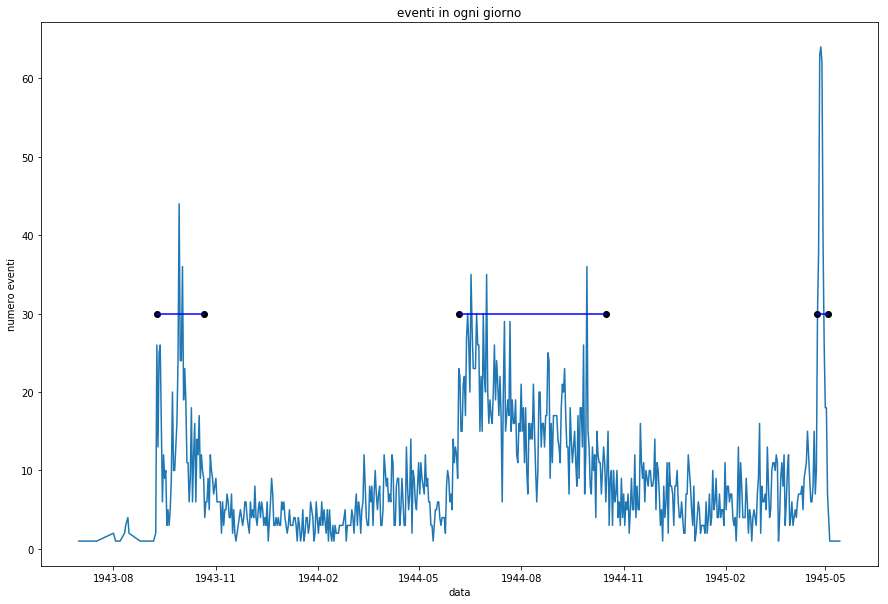
Questa funzione ci mostra in modo ancora più chiaro la notevole variabilità dei dati con cui abbiamo a che fare

### Applicazione dell’algoritmo

quando applichiamo l’algoritmo al data frame sopra, otteniamo i seguenti intervalli:



graficamente possiamo vedere la cosa nel seguente modo:

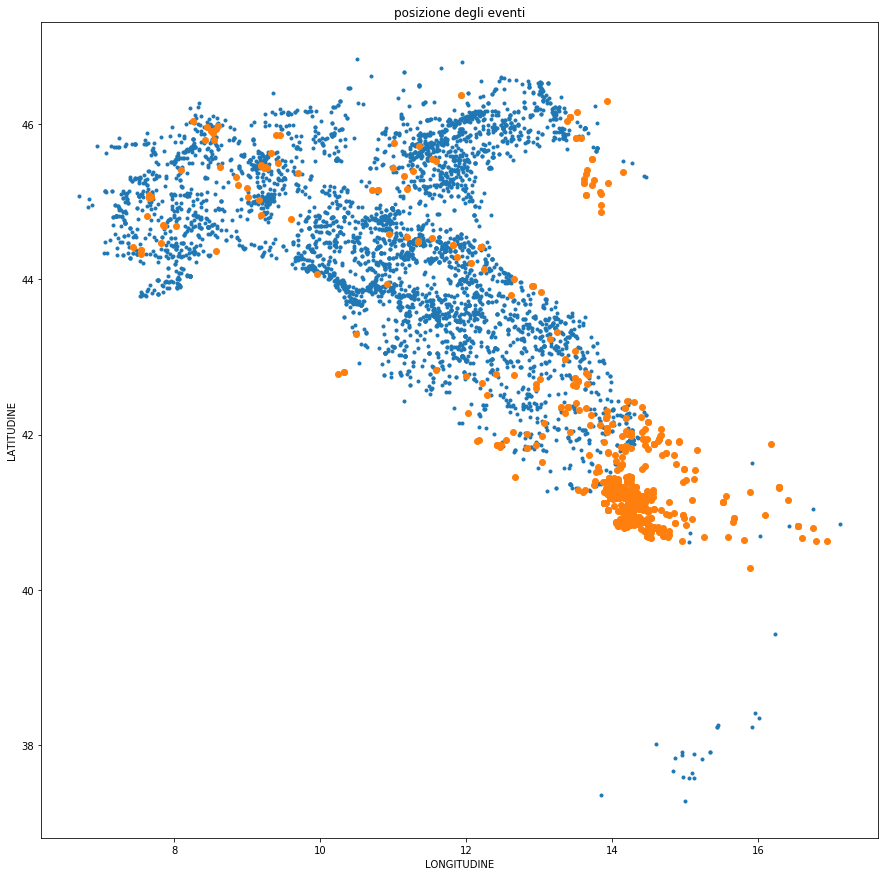


### Posizione delle stragi del primo intervallo avente numero massimo di eventi (1943-09-09, 1943-11-13)

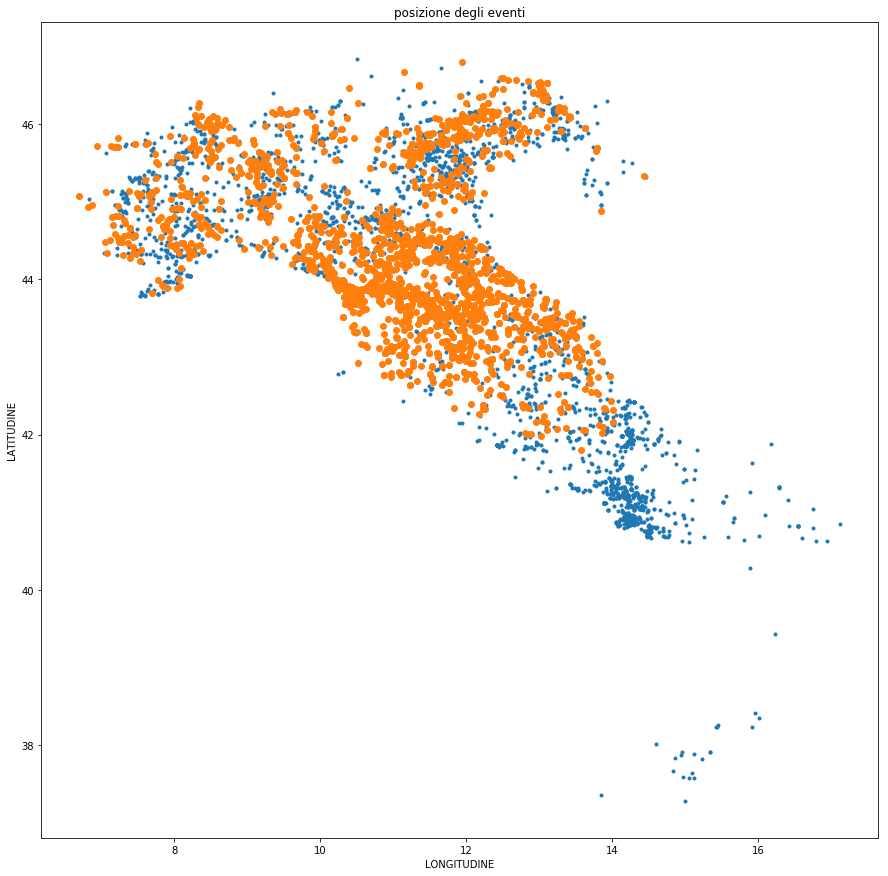
Il primo intervallo temporale va dal 9 settembre 1943 al 21 ottobre del 1943.

Noi adesso visualizziamo le stragi che avvengano in questo periodo insieme alle stragi che avvengono complessivamente.

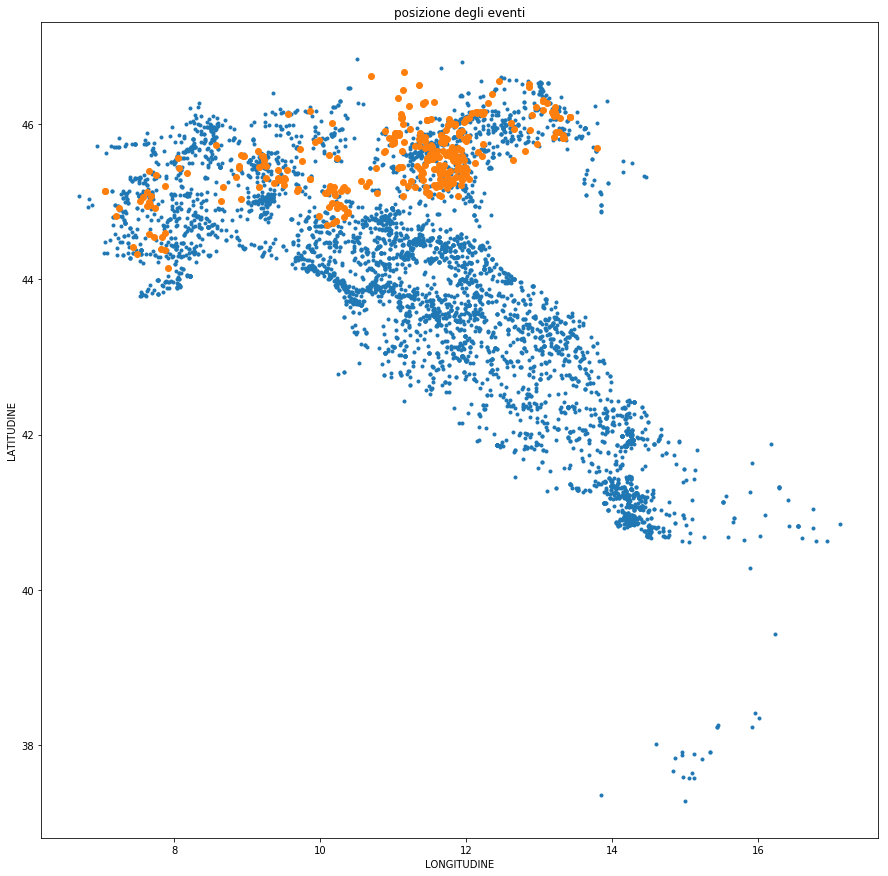
Utilizzando un colore diverso per le stragi che avvengano nell’intervallo considerato, possiamo effettuare un confronto visivo tra esse.



### Posizione delle stragi del secondo intervallo avente numero massimo di eventi (1944-06-06, 1944-10-16)



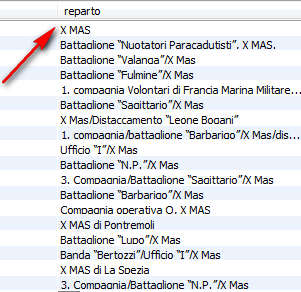
### Posizione delle stragi del terzo intervallo avente numero massimo di eventi (1945-04-24, 1945-05-04)



# Stragi compiute dalla XMAS

Analizziamo adesso in particolare le stragi effettuate da un corpo specifico: la X MAS di La Spezia.

Dal database si vede che esistono più reparti militari nel cui nome compare X-MAS.



Noi però prenderemo in considerazione solo le stragi effettuate dal reparto di nome X MAS.

In particolare analizzeremo tutte le righe della tabella dba\_\_stragi che contengono il nome X MAS

in una delle 7 colonne che rappresentano i vari reparti responsabili delle singole stragi.

Per ottenere questo risultato effettuiamo una query sul database in base alla quale recuperiamo: la data di inizio della strage, la latitudine e la longitudine della strage, ed i reparti che sono coinvolti in tale strage.

Il codice parziale SQL di questa query è il seguente:

*SELECT ST.datainizio, ST.id\_strage,CO.lat, CO.lng, REP1.reparto AS reparto1 , REP2.reparto AS reparto2, REP3.reparto AS reparto3, REP4.reparto AS reparto4, REP5.reparto AS reparto5, REP6.reparto AS reparto6, REP7.reparto AS reparto7*

*FROM dba\_\_stragi AS ST JOIN dba\_\_coordinate AS CO*

*ON ST.id\_strage=CO.id\_strage*

*LEFT JOIN dba\_\_reparti AS REP1*

*ON ST.repartoresponsabile1=REP1.id\_reparto*

*LEFT JOIN dba\_\_reparti AS REP2*

*ON ST.repartoresponsabile2=REP2.id\_reparto*

*…*

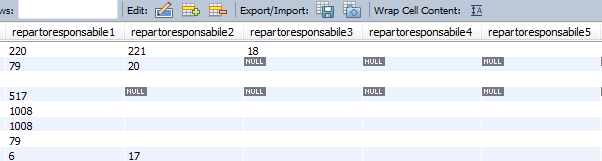
Notiamo che in questa query vengono utilizzati dei LEFT JOIN.

Infatti se utilizzassimo dei JOIN normali, (INNER JOIN) la query selezionerebbe solo le righe che rappresentano le stragi in cui sono presenti sette reparti diversi.

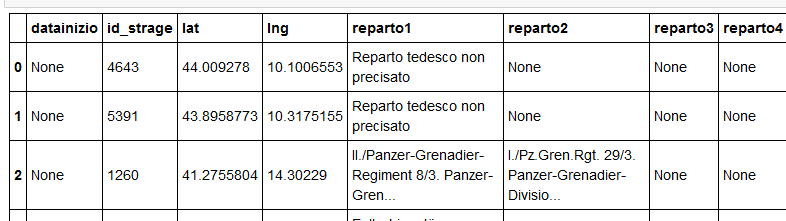
Poiché, come si vede nell’immagine sotto, tante colonne relative ai reparti coinvolti presentano valori nulli o vuoti, questo ridurrebbe di molto la dimensione della query vanificando la sua utilità.

Viceversa gli outer join (di cui il LEFT JOIN è una parte) a differenza degli inner join, selezionano i risultati anche in assenza di una corrispondenza su entrambe le tabelle.

Più precisamente il LEFT JOIN estrae tutti i valori della tabella a sinistra , che nel nostro caso è dba\_stragi , anche se non hanno corrispondenza nella tabella a destra che nel nostro caso è rappresentata dalla tabella dba\_reparti.



Effettuato la query, trasformiamo i risultati ottenuti da essa in un data frame.



Quindi con il seguente codice analizziamo le varie colonne che rappresentano i reparti coinvolti nelle stragi, e selezioniamo solo quelle righe in cui compare il nome X MAS nei vari reparti.

*import pandas as pd*

*lista=[]*

*for indice, row in df6.iterrows():*

*if row['reparto1']=='X MAS'or row['reparto2']=='X MAS'or row['reparto3']=='X MAS'or*

*row['reparto4']=='X MAS' or row['reparto5']=='X MAS'or row['reparto6']=='X MAS'or*

*row['reparto7']=='X MAS':*

*lista.append(df6.iloc[indice])*

*df7=pd.DataFrame(lista)*

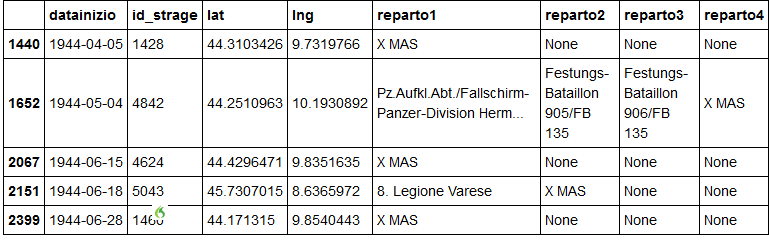
*df7.head()*

Il codice opera nel seguente modo: creiamo una lista vuota.

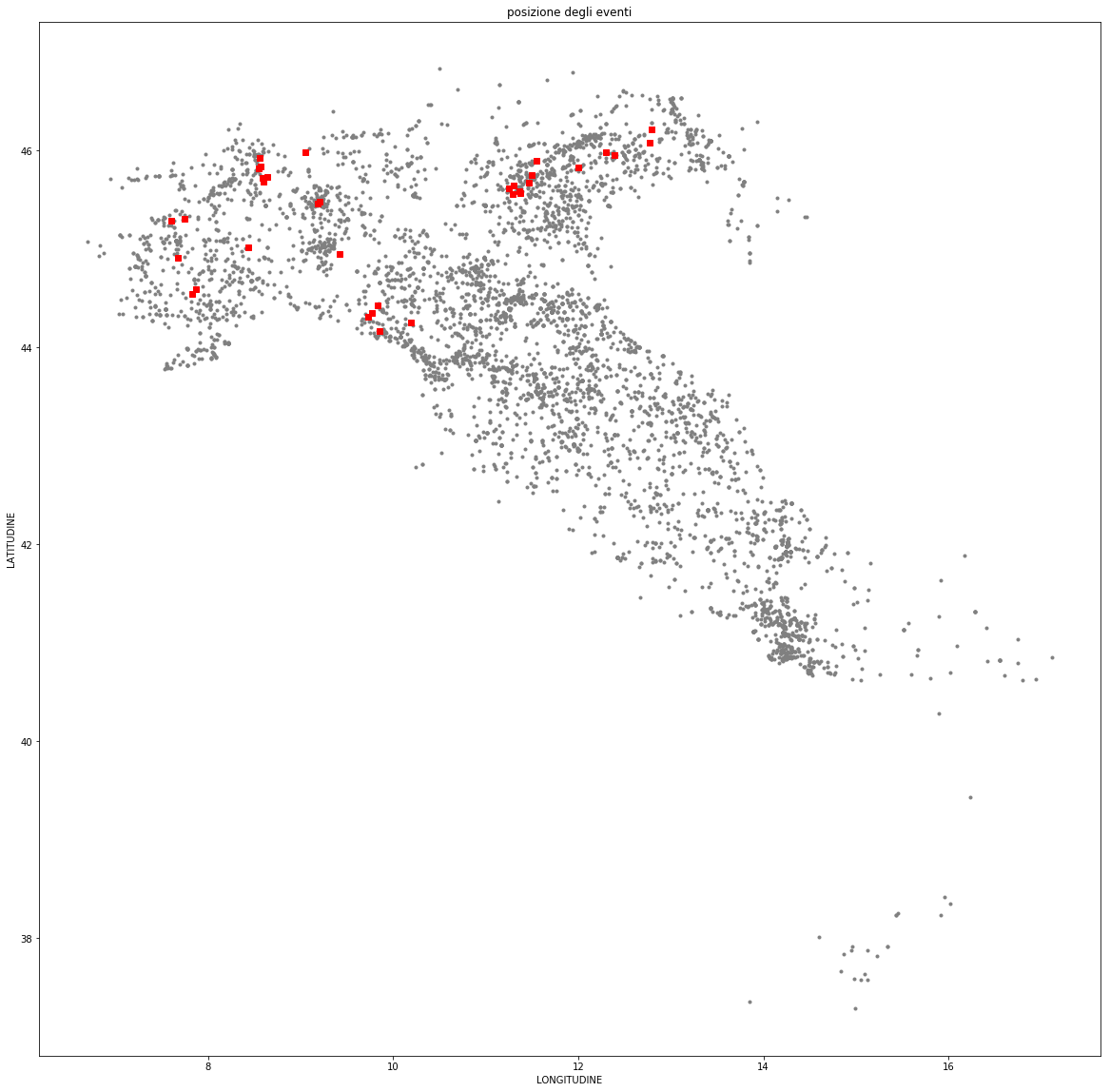
Tramite il metodo iterrows() facciamo una scansione del data frame riga per riga e, per ogni riga, valutiamo le colonne che rappresentano i reparti.

Quando troviamo una colonna in cui compare il nome X MAS, appendiamo tale riga alla lista.

Completata la scansione e quindi completata la lista, otteniamo un nuovo data frame a partire da tale lista, formato da tutte le righe del data frame di partenza in cui una colonna delle varie colonne reparto contiene la parola X MAS.



Usiamo quindi le colonne latitudine e longitudine di tale data frame per realizzare il seguente grafico.



# MACHINE LEARNING: Regole associative

## Cosa sono le regole associative

<http://www.unife.it/ing/lm.infoauto/data-mining-analytics/materiale-didattico-1/16-regole%20associative.pdf>

Abbiamo visto che nel database delle stragi, ad una singola strage possono prendere parte fino a 7 reparti diversi.

Quello che noi vogliamo fare adesso è utilizzare le tecniche di Machine learning per vedere se esistono reparti che prendono parte a una serie di stragi, in associazione con altri reparti.

Per fare questo utilizziamo le regole di associazione.

Una regola di associazione A=>B associa un oggetto A (antecedente) con un oggetto B (conseguente).

Le regole di associazione sono molto usate per valutare le associazioni tra articoli presenti nella spesa dei clienti dei grandi supermercati.

Un esempio di regola associativa presente in una spesa può essere:

Chi compra biscotti compra anche latte

cioè biscotti => latte.

Le regole di associazione sono uno strumento molto flessibile che viene utilizzato anche al di fuori dei grandi supermercati.

È quello che faremo noi: utilizzeremo le regole di associazione per valutare i reparti che con più frequenza operavano assieme nelle varie stragi.

### supporto della regola A=>B

numero di righe della tabella contenenti sia l’oggetto A che l’oggetto B

diviso

numero di righe della tabella.

Supporto della regola A=>B al 10% significa che di tutte le righe della tabella, il 10% di queste

contiene sia oggetti A che B.

**minimo supporto**: valore minimo del supporto che devono avere le regole di associazione per essere accettate; è un parametro che inseriamo noi nell’algoritmo in modo da cambiare le prestazioni dell’algoritmo stesso.

Supporto minimo:

* alto vuol dire avere poche regole valide che sono verificate molto spesso;
* basso vuol dire avere molte regole che sono verificate raramente

### Confidenza della regola A=>B

numero di righe della tabella contenenti sia l’oggetto A che l’oggetto B

diviso

numero di righe della tabella contenenti l’oggetto A

Confidenza della regola A=>B al 90% significa che il 90% delle righe della tabella che contengono l’oggetto A contengono anche l’oggetto B.

### Come vengono calcolate le regole associative

Dato un database D, il compito di scoprire le regole associative può essere riformulato come segue – scoprire tutte le regole associative con almeno un minimo supporto (chimato minsup ) e una minima confidenza (chiamata minconf ), dove minsup e minconf sono valori specificati dall’utente

Il compito di scoprire regole associative può essere decomposto in due sottoproblemi :

* Trovare tutti gli insiemi di oggetti che hanno supporto sopra il minimo .

Tali insiemi di oggetti sono chiamati insiemi di oggetti più frequenti (grandi) .

Questo sottoproblema è risolto dall’algoritmo APRIORI

* A partire degli insiemi di oggetti più frequenti (grandi) considerare come regole associative quegli oggetti più frequenti (grandi) che hanno almeno la minima confidenza .

### Troviamo gli insiemi di oggetti più frequenti (grandi)

Un database formato da N oggetti diversi può generare insiemi, ogni insieme formato da oggetti diversi.

Ad esempio consideriamo quattro oggetti A, B, C, D.

Vediamo quali sono tutti i possibili insiemi che possono essere generati a partire da questi 4 oggetti diversi.

Poiché abbiamo 4 oggetti, la formula superiore ci dice che sono possibili 15 insiemi formati da oggetti diversi;

essi sono:

A B C D

A,B A,C A,D B,C B,D C,D

A,B,C A,B,D A,C,D B,C,D

A,B,C,D

Le regole di associazione ci permettono di selezionare quali di questi insiemi si presenta più frequentemente all’interno del database.

Per fare questo utilizziamo un algoritmo che prende il nome di algoritmo Apriori.

### Algoritmo Apriori

Vediamo adesso come funziona l’algoritmo Apriori.

Consideriamo la seguente affermazione:

se l’insieme di oggetti A,B compare di frequente nelle transazioni, allora anche l’oggetto A sottoinsieme di A, B compare di frequente nelle transazioni.

L’algoritmo a priori sfrutta una conseguenza di tale frase:

**se l’oggetto A non è frequente, non è necessario generare e valutare gli insiemi che lo includono.**

Vediamo ora come funziona l’algoritmo Apriori.

* Passo 1: viene calcolato il supporto di ogni oggetto in questo modo si ottengono gli oggetti (singoli) che compaiono più frequentemente nelle varie righe.
* Passo 2: gli oggetti singoli più frequenti vengono estesi ottenendo nuovi insiemi formati da due oggetti.
* Passo 3: viene calcolato il supporto dei nuovi insiemi formati da due oggetti e vengono mantenuti solo quelli che soddisfano al criterio di **minimo supporto ;** tale valore lo inseriamo noi dell’algoritmo.

Di volta in volta il numero dei candidati diminuisce, la procedura finisce quando non ci sono più candidati che soddisfano la condizione di minimo supporto.

### Troviamo le regole associative

Trovati gli insiemi di oggetti più frequenti, applichiamo la confidenza per ottenere le regole associative

## Come applichiamo le regole associative al nostro database

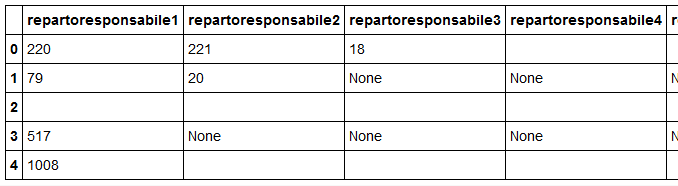
Iniziamo con il ricavare dalla tabella dba\_stragi l’ID di ogni reparto coinvolto nelle varie stragi.

Per fare questo usiamo la seguente semplice query:

*SELECT ST.repartoresponsabile1, ST.repartoresponsabile2, ST.repartoresponsabile3, ST.repartoresponsabile4, ST.repartoresponsabile5, ST.repartoresponsabile6, ST.repartoresponsabile7*

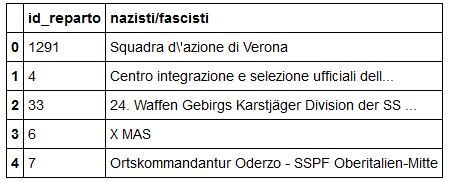
*FROM dba\_\_stragi AS ST*

quindi inseriamo i risultati della query in un data frame.



Ora a partire dalla tabella dba\_reparti, creiamo un data frame formato dall’ID di ogni reparto e il suo nome.

Il numero di righe di questo data frame ci permetterà di valutare il numero di reparti diversi coinvolti nelle varie stragi.



Dal data frame estraggo una Series formata dall’ID dei reparti.

*id\_reparti\_series=df1['id\_reparto']*

Questa series ha una lunghezza pari al numero dei reparti diversi che partecipano alle stragi.

Creiamo adesso il data frame df\_per\_regole\_associative su cui applicheremo l’algoritmo Apriori.

Questo data frame è formato da soli zero ed ha un numero di colonne, pari al numero dei reparti diversi che partecipano alle stragi, ed un numero di righe pari al numero delle stragi.

Le colonne le ottengo grazie alla series del passo precedente.

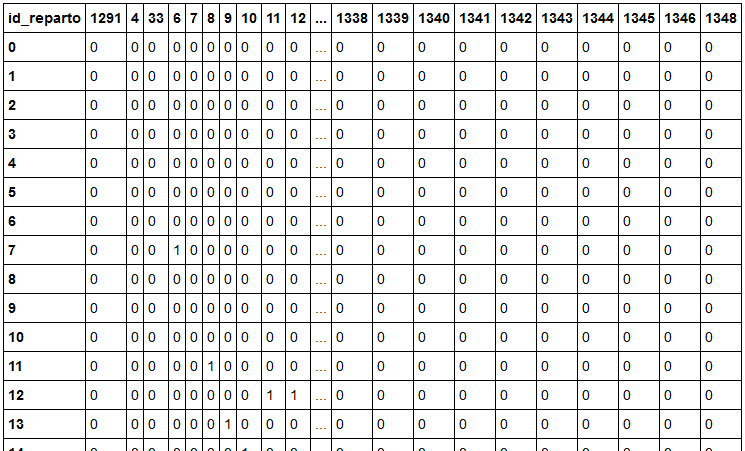
L’istruzione che permette la generazione di questo data frame vuoto è la seguente:

*df\_per\_regole\_associative = pd.DataFrame(0, index=np.arange(df\_reparti.shape[0]), columns=id\_reparti\_series)*

adesso dobbiamo riempire il data frame vuoto.

Per fare questo, facciamo una scansione del data frame reparti che è il primo data frame che abbiamo creato e che contiene gli ID di tutti i reparti coinvolti nelle varie stragi.

Grazie alla scansione, ogni volta che incontriamo l'ID di un reparto, aggiungiamo un 1 alla colonna del dataframe df\_per\_regole\_associative (che è il data frame che stiamo preparando per apriori) nella riga relativa alla strage considerata.



Adesso applichiamo l’algoritmo a priori.

Questo algoritmo non è presente nella libreria scikit learn di Piton quindi ho installato la libreria mlxtend (http://rasbt.github.io/mlxtend/).

Come prima cosa calcoliamo gli oggetti più frequenti con le seguenti istruzioni:

*import pandas as pd*

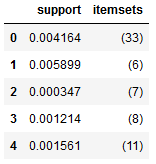
*from mlxtend.frequent\_patterns import apriori*

*from mlxtend.frequent\_patterns import association\_rules*

*frequent\_itemsets = apriori(df\_per\_regole\_associative, min\_support=0.0003,use\_colnames=True)*

*frequent\_itemsets.head()*

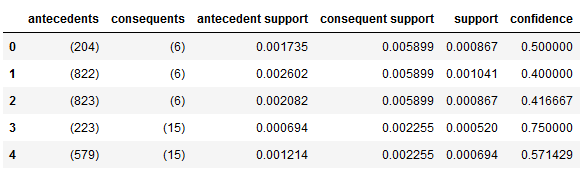
Otteniamo quindi:



adesso cerchiamo le regole di associazione più comuni con le seguenti istruzioni:

*rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.3)*

*rules*

**

Nel grafico sopra vediamo le regole di associazione che abbiamo ottenuto.

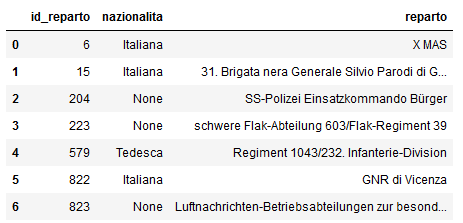
Per passare dai numeri ai nomi effettivi, utilizziamo una query sulla tabella dba\_reparti

*SELECT REP1.\**

*FROM dba\_\_reparti AS REP1*

*WHERE REP1.id\_reparto IN (204,822,823,6,223,15,579)*

Trasformiamo la query in data frame, selezioniamo le colonne che ci interessano e otteniamo le seguenti informazioni



Visualizziamo il nome dei reparti che partecipano alle regole di associazione

(204)=>(6) SS-Polizei Einsatzkommando Bürger => X MAS

(822)=>(6) GNR di Vicenza => X MAS

(823)=>(6) Luftnachrichten-Betriebsabteilungen zur besonderen

Verwendung 11 => X MAS

(223)=>(15) schwere Flak-Abteilung 603/Flak-Regiment 39

=> 31. Brigata nera Generale Silvio Parodi di Genova

(579)=>(15) Regiment 1043/232. Infanterie-Division

=> 31. Brigata nera Generale Silvio Parodi di Genova

# MACHINE LEARNING: Cluster

## Algoritmo K-means

l’algoritmo K-means è un modello di Machine learning senza supervisione, che tenta di raggruppare i punti dei dati in cluster basati su centroidi.

K-means è un metodo iterativo che suddivide un insieme di dati in K cluster.

Prevede 4 passi:

1) scegliere K centroidi iniziali (notare che K è un input),

2) per ogni punto, assegnare tale punto al centroide più vicino,

3) ricalcolare la posizione dei centroidi in base alle assegnazioni dei punti fatta al passo 2),

4) ripetere i passi 2) e 3) fino a raggiungere il criterio di arresto

## Come applichiamo tale algoritmo al nostro dataset

ricaviamo dalla tabella dba\_stragi le coordinate di ogni strage e ordiniamo le stragi in base alla loro data di inizio.

*SELECT ST.datainizio, CO.lat, CO.lng*

*FROM dba\_\_stragi AS ST JOIN dba\_\_coordinate AS CO ON ST.id\_strage=CO.id\_strage*

*ORDER BY ST.datainizio*

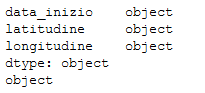
Trasformiamo la query in un data frame.

Faccio la pulizia dei dati del data frame: tolgo le stragi di cui non conosciamo la data e quelle in cui la data è sbagliata.

Inoltre togliamo una riga del data frame in cui una coordinata è scritta in modo scorretto.

Consideriamo adesso i tipi contenuti nel data frame

*print(df['data\_inizio'].dtypes)*



Per poter lavorare con l’algoritmo K-means, dobbiamo trasformare i tipi object in tipi numerici.

Iniziamo quindi a lavorare sulle date e trasformiamo tali date (che sono delle stringhe ) in un valore numerico che può essere gestito dall’algoritmo k-means.

Per fare questo prendiamo una data di riferimento e valutiamo la differenza in giorni che passa tra essa e le date in cui avvengono le varie stragi.

Come data di riferimento considero la data in cui avviene la prima strage.

Tale data è il 1 luglio 1943 (1943-7-1).

trasformo la stringa "1943-7-1" in un oggetto date

*anno,mese,giorno = "1943-7-1".split("-")*

*d = date(int(anno), int(mese), int(giorno))*

creo la colonna 'data\_modif' che contiene il timedelta tra una generica data e la data d="1943-7-1"

*df['data\_modif']=df['data\_inizio']-d*

Adesso dobbiamo trasformare le colonne latitudine e longitudine da object a numeri.

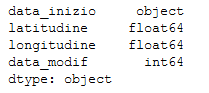
Questa cosa la facciamo con le seguenti istruzioni:

*df.longitudine=df.longitudine.astype(float)*

*df.latitudine=df.latitudine.astype(float)*

adesso abbiamo i seguenti tipi:

*print(df.dtypes)*



Abbiamo trasformato tutte le colonne che ci servono per generare il modello del cluster in numeri però abbiamo un altro problema.

data, latitud e longit hanno scale diverse:

il valore dei giorni varia tra 0 e 683 mentre latitudine e longitudine variano in una finestra ("valore\_massimo" , "valore\_minimo") di valori.

Per evitare risultati inesatti, devo normalizzare i dati su cui calcolo i cluster in modo da avere dati comparabili tra loro.

Noi utilizzeremo la normalizzazione 0,1 che è una normalizzazione mediante la quale ogni valore contenuto nelle varie colonne, viene mappato su un intervallo che varia tra zero e uno.

Per fare questo si considerano tutti i valori numerici contenuti nella singola colonna, si prende il valore massimo ed il valore minimo quindi si sottrae da ogni valore contenuto nella colonna, il valore minimo, e si divide il risultato ottenuto per la differenza tra valore massimo e valore minimo.

Queste operazioni le facciamo con le seguenti istruzioni:

*# normalizzo i giorni rispetto il loro valore massimo*

*df['data\_modif']=df['data\_modif']/df['data\_modif'].max()*

*# normalizzo la longitudine rispetto la sua finestra*

*longitudine\_Max=df.longitudine.max()*

*longitudine\_min=df.longitudine.min()*

*df['longitudine\_norm']=(df['longitudine']-longitudine\_min)/(longitudine\_Max-longitudine\_min)*

*# normalizzo la latitudine rispetto la sua finestra*

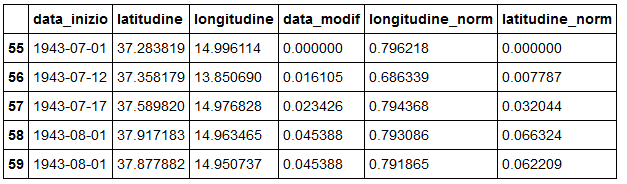
*latitudine\_Max=df.latitudine.max()*

*latitudine\_min=df.latitudine.min()*

*df['latitudine\_norm']=(df['latitudine']-latitudine\_min)/(latitudine\_Max-latitudine\_min)*

*df.head()*

Otteniamo il seguente data frame:



selezioniamo quindi le colonne che utilizziamo per calcolare i cluster:

*colonne\_da\_selezionare=['data\_modif','longitudine\_norm','latitudine\_norm']*

*X=df[colonne\_da\_selezionare]*

e finalmente, dopo aver preparato i dati, arriviamo ad applicare l’algoritmo k-means che mi genererà i cluster.

*# K-means con 6 cluster*

*from sklearn.cluster import KMeans*

*km = KMeans(n\_clusters=6, random\_state=1)*

km rappresenta l’algoritmo.

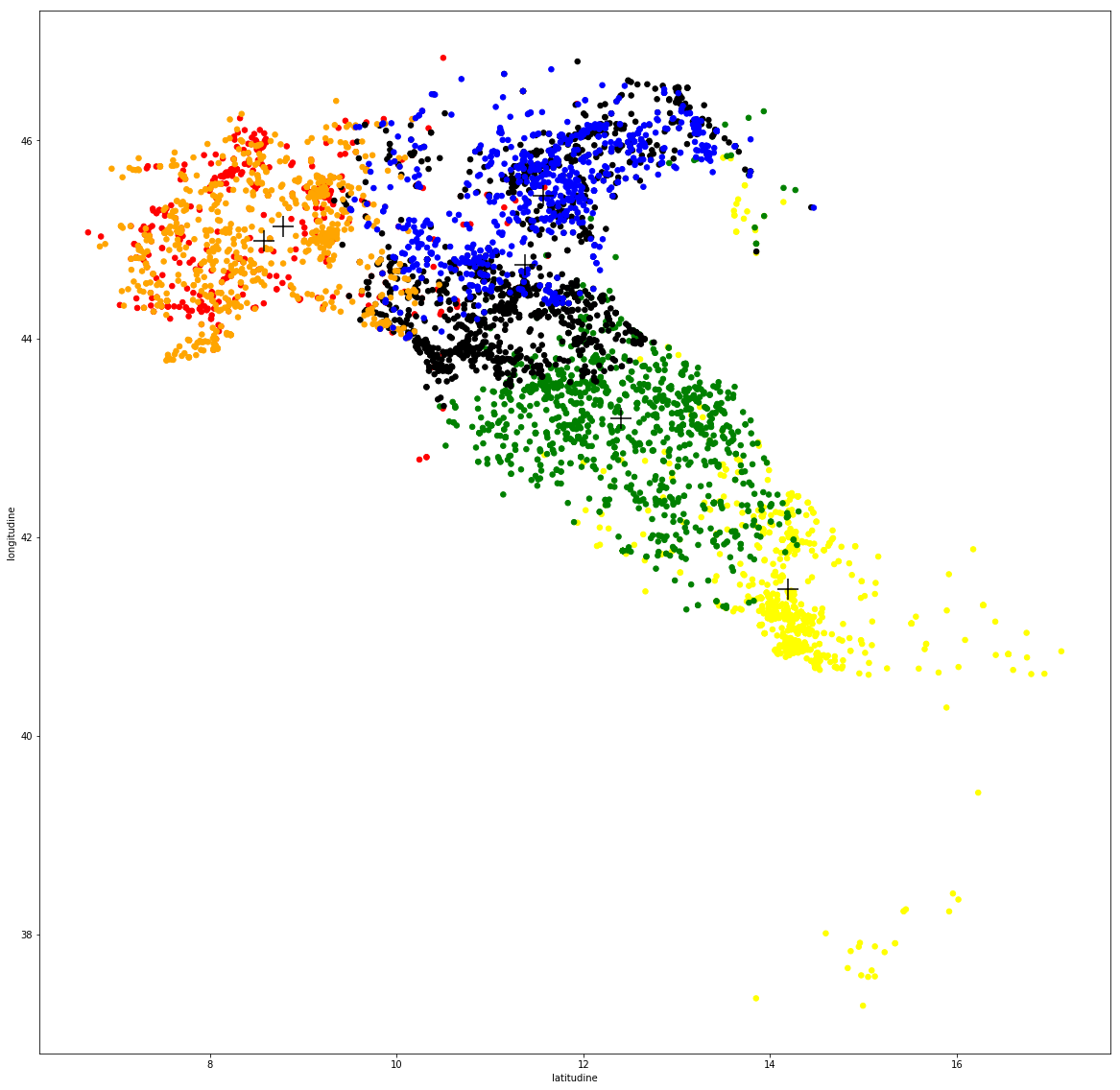
Questo algoritmo deve essere addestrato con i dati X che abbiamo ottenuto in precedenza.

Per fare questo utilizziamo la seguente istruzione:

*km.fit(X)*

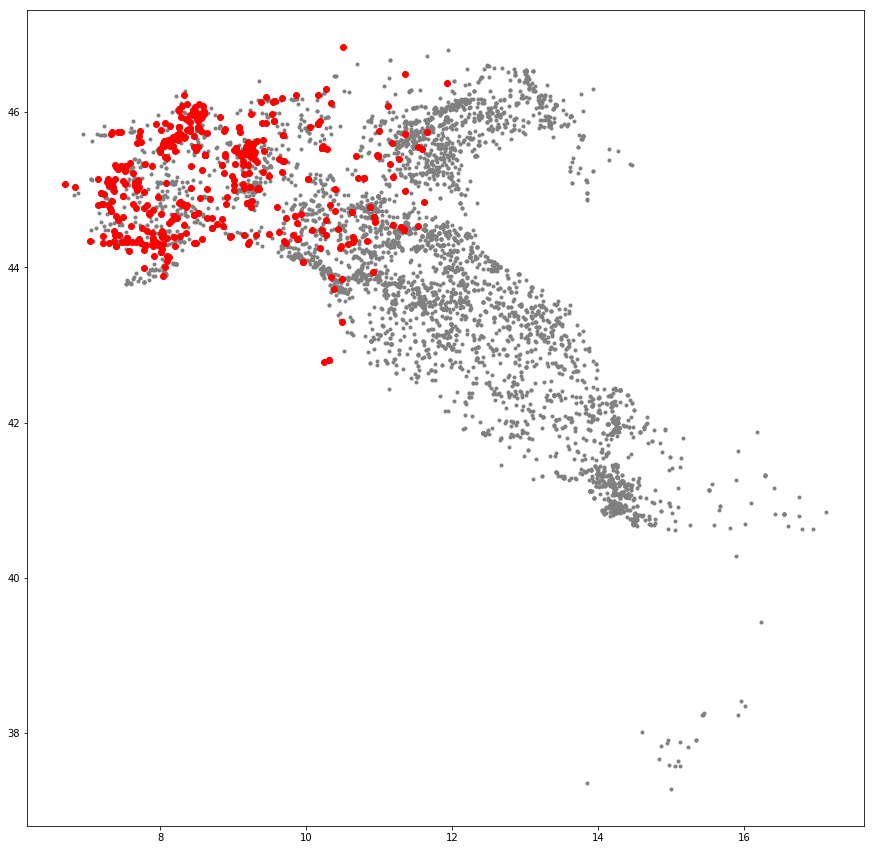
adesso il nostro algoritmo è addestrato quindi abbiamo un modello che ottiene i cluster a partire dai nostri dati.

Visualizziamo i risultati ottenuti:

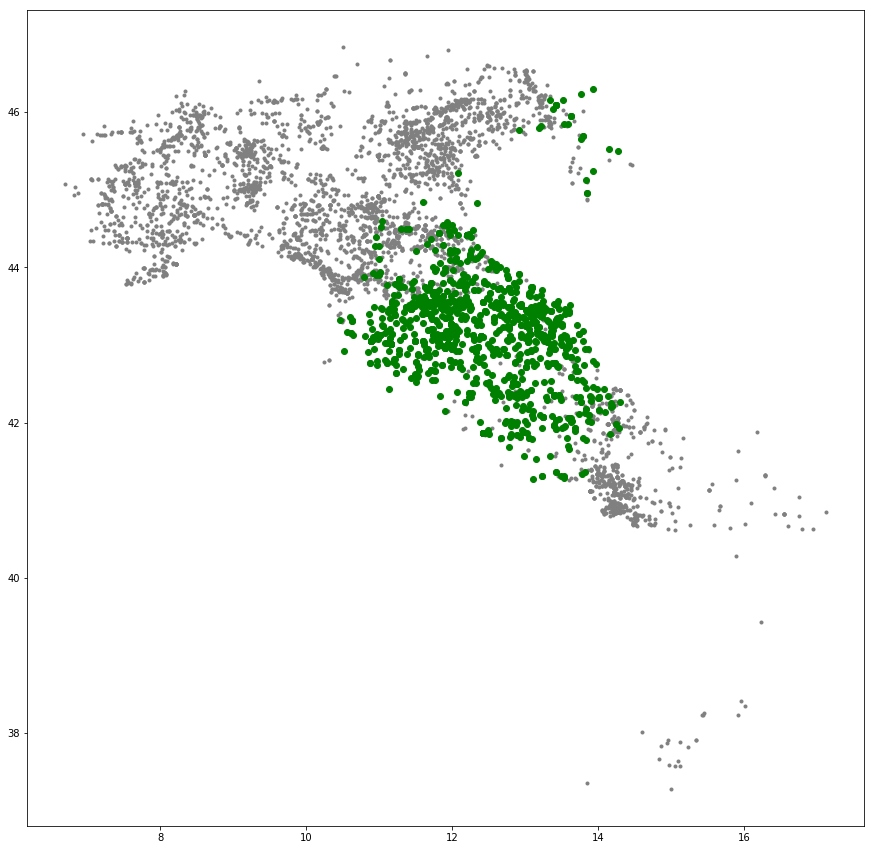


vediamo più in dettaglio i singoli cluster:

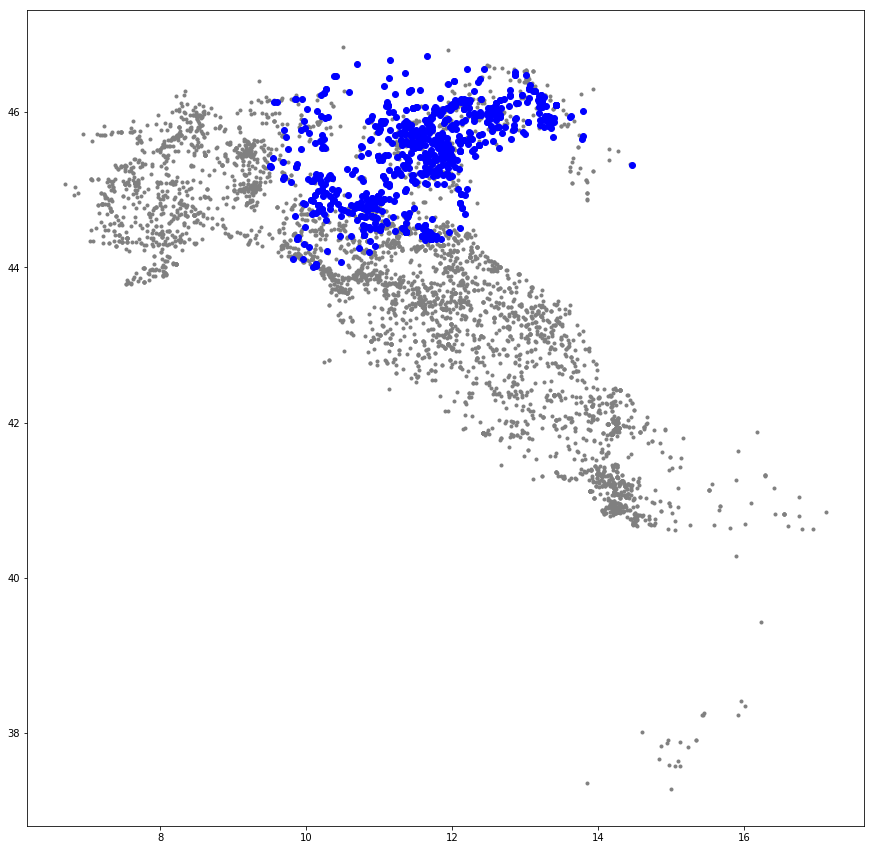
cluster 0



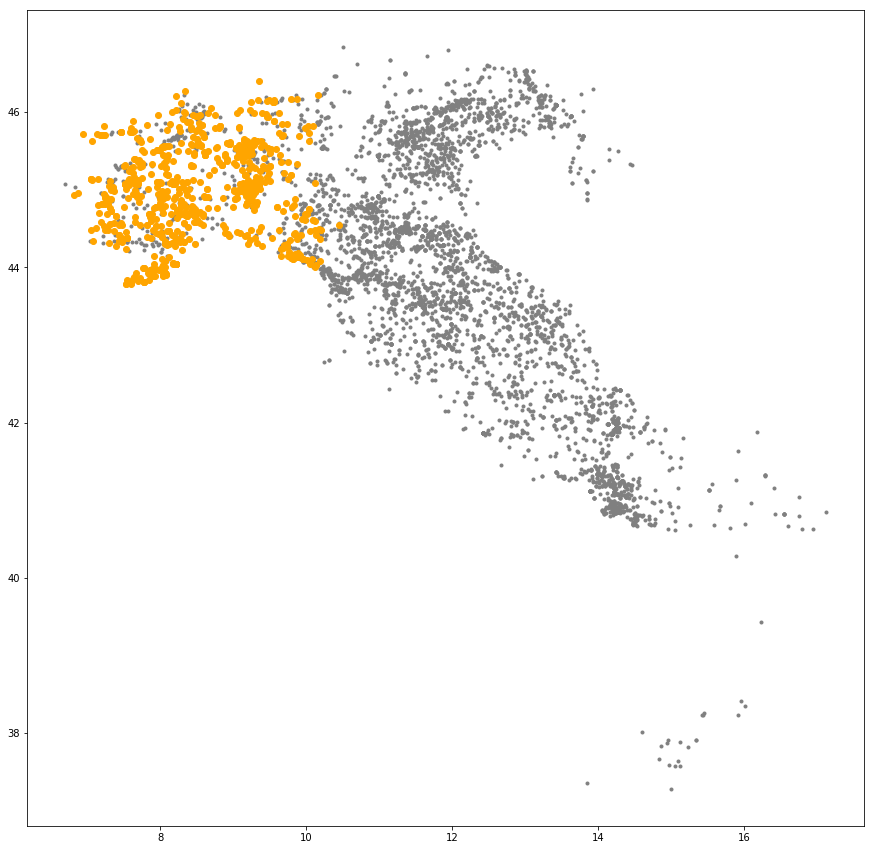
cluster 1



cluster 2



e per finire esaminiamo il cluster 5



nel nostro esempio noi abbiamo utilizzato sei cluster.

Osserviamo come i singoli cluster assomiglino molto ai grafici che avevamo trovato in precedenza soprattutto quando tali grafici analizzavano le varie stragi avvenute in intervalli di tempo ben definiti.

Questo è dovuto al fatto che i nostri cluster, che abbiamo realizzato a partire dalle coordinate geografiche e dalla data in cui avviene la strage, tendono a riunire stragi che avvengono in località prossime una all’altra e che hanno date vicine.

## Cluster posizione geografica, data, vittime complessive

Vediamo adesso di ottenere i cluster aggiungendo una ulteriore colonna che è quella delle vittime complessive.

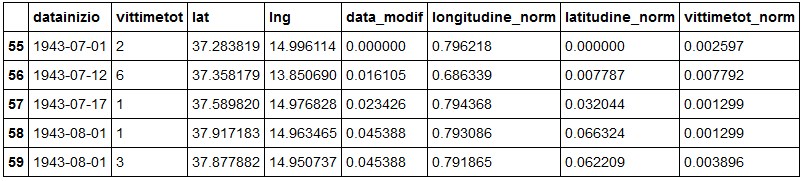
Ripetiamo tutte le operazioni visto in precedenza ma alla query di partenza aggiungiamo la colonna vittime totali:

*SELECT ST.datainizio,ST.vittimetot, CO.lat, CO.lng*

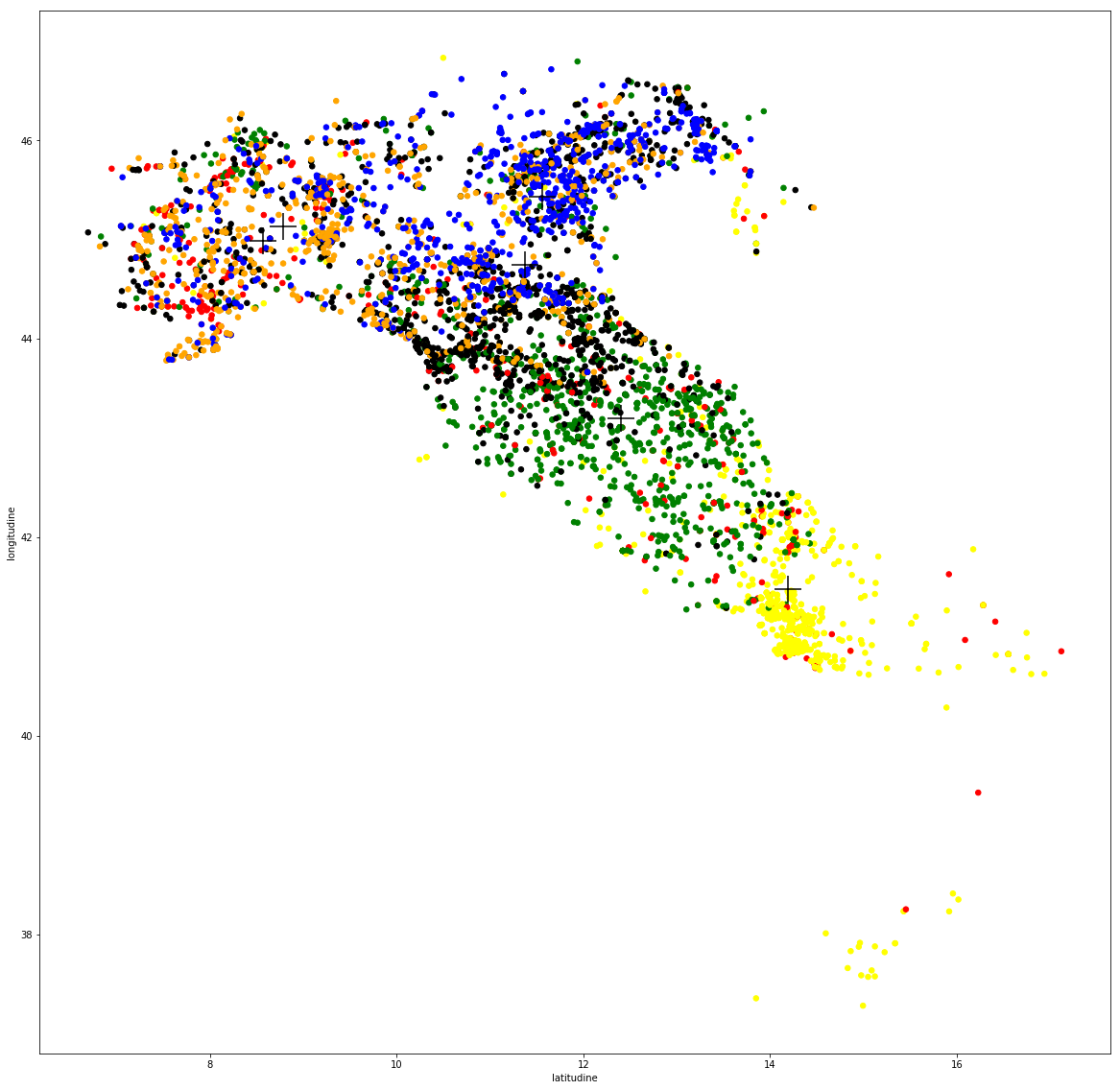
*FROM dba\_\_stragi AS ST JOIN dba\_\_coordinate AS CO ON ST.id\_strage=CO.id\_strage*

*ORDER BY ST.datainizio*

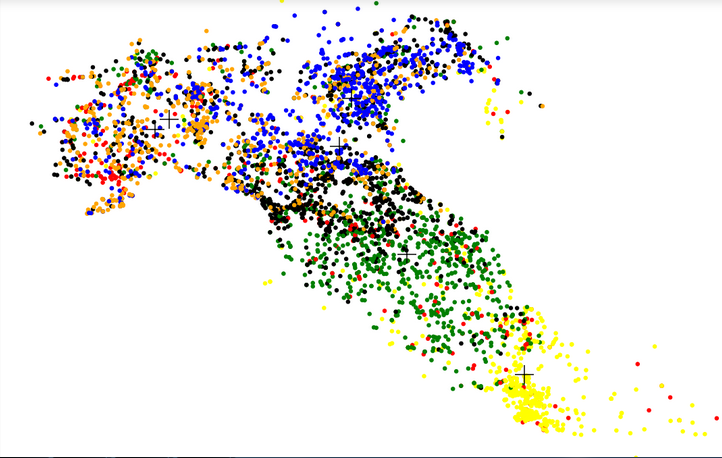
dopo aver effettuato le operazioni di pulizia e normalizzazione viste nel caso precedente, otteniamo il seguente data frame:



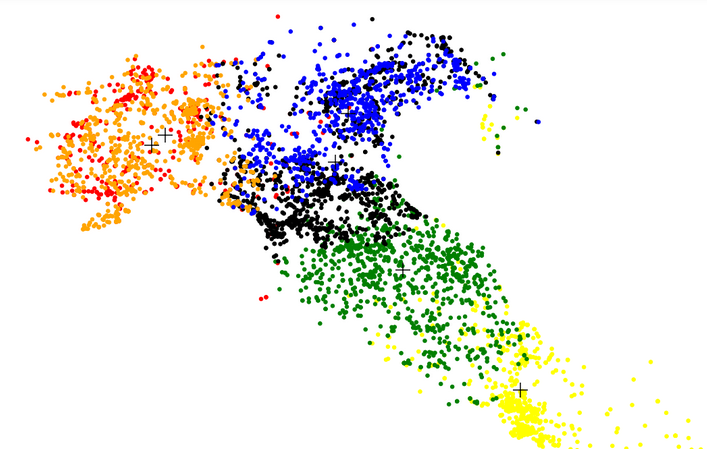
applichiamo quindi nel nostro algoritmo che ci permette di calcolare i cluster e visualizziamo il risultato:



confrontiamo adesso i due grafici:



**Cluster ottenuti considerando anche le vittime totali**



**Cluster ottenuti con coordinate geografiche e data**

Dal confronto dei due grafici si evince come l’aggiunta della colonna vittime totali, nel calcolo dei cluster, genera una certa dispersione dei cluster stessi.