Progetto Data Management

GIANNELLI ALESSIO (846586) - IMBONATI LORENZO (846577) - VALOTI DAVIDE (846737)

**Near real-time streaming di attività sportive**

Raccolta ed Arricchimento di attività caricate sull’applicazione Strava

*A partire dal primo semestre del 2020, il mondo del digital fitness ha subito una forte impennata dei consumi, dovuti alla situazione pandemica. Dal World Economic Forum 2020 si evince, in particolare, una crescita nell’utilizzo delle app di fitness, il cui tasso di utilizzo giornaliero è aumentato tra il primo e il secondo trimestre 2020 del 24%. Questo cambiamento delle abitudini degli sportivi viene definito come non transitorio, nonostante l’inevitabile rimbalzo negativo previsto con la riapertura delle palestre. Per osservare il fenomeno vengono analizzate le attività caricate dagli utenti su Strava, una delle applicazioni più utilizzate per il caricamento di attività sportive con tracciamento GPS, tanto da rendere popolare il detto “Se non è su Strava, non è successo”.*

1. **Introduzione e Obiettivo**

Il software Strava è un servizio che permette di tracciare mediante GPS le coordinate degli utenti che ne usufruiscono per attività quali ciclismo, corsa e sport acquatici. Oltre alle coordinate, gli utenti possono anche registrare le informazioni riguardanti le performance dell’attività fisica eseguita e caricare una moltitudine di dati diversi come velocità, altitudine e battito cardiaco. Si tratta di un’applicazione molto utilizzata sia dagli atleti con esperienza sia da tutte quelle persone che si approcciano da poco a questi sport. Uno dei punti di forza è quello di permettere il confronto tra diverse performance, sia dell’utente stesso sia con altri sportivi che hanno percorso lo stesso tracciato. Inizialmente, quando è nata nel 2009, si trattava di un servizio gratuito per tutti, mentre con il passare degli anni è diventato a pagamento per l’accesso ad alcune funzionalità pro.

L’obiettivo del lavoro è quello di riuscire a definire una struttura dati contenente le informazioni riguardanti i diversi atleti divisi per sport o topic, ed integrare queste informazioni, estratte da Strava con tecniche di scraping, a dati meteorologici per località e giorno ottenuti tramite chiamate API al portale web Visual Crossing. Le attività di interesse sono prodotte in near real-time e vengono immagazzinate e rese accessibili mediante un’architettura Kafka (pub/sub). L’utilizzo di questa infrastruttura, oltre ad una gestione del dato attraverso topics, permette di disaccoppiare il tempo di produzione del dato da quello di accesso allo stesso (modalità asincrona), limitando la possibilità di avere attività mancanti perché eliminate.

1. **Strumenti e tecnologie utilizzate**

Per raggiungere gli obiettivi definiti, si è deciso di concentrarsi sulle dimensioni di Velocità e di Varietà, date rispettivamente dall’acquisizione di dati in tempo reale e dalla struttura flessibile del dataset. L’ottenimento e la memorizzazione dei dati relativi alle attività registrate sul servizio Strava è stata attuata mediante l’utilizzo di algoritmi di web scraping e dell’architettura Apache Kafka, ossia una piattaforma per la gestione di dati in tempo reale.

Il vantaggio di effettuare l’operazione di scraping e di immagazzinamento dei dati singolarmente per ogni attività consiste nell’avere una visione in tempo reale delle informazioni disponibili, vincolata all’iscrizione al topic di interesse. L’idea è, infatti, quella di rendere istantanea e, soprattutto, aggiornata la lettura dei dati, focalizzandosi solo sugli sport di interesse; per questo motivo le attività sono state categorizzate nei seguenti topics: Ciclismo, Corsa, Sport acquatici ed Altro.

I dati ottenuti tramite Consumer Kafka sono stati in seguito integrati con informazioni riguardanti le condizioni meteorologiche tramite la Weather API resa disponibile dal sito *visualcrossing.com*. L’integrazione è basata sulla location e sul giorno di ogni singola attività fisica, permettendo così sia un’operazione di Data integration (per attributi come le raffiche di vento e le radiazioni solari) sia di Data Enrichment (essendo le informazioni meteo disponibili solo per alcuni utenti Strava).

Per alcuni record sono stati raccolti anche i testi dei commenti riportati dagli utenti della piattaforma. Al fine di aumentare l’accessibilità del dataset, i testi dei commenti sono stati tradotti in italiano mediante la libreria *deepl,* così come anche i titoli delle attività e la descrizione della situazione meteo.

Infine, il dataset ottenuto è stato caricato sul server MongoDB mediante la libreria *pymongo*, da cui è possibile accedere alla lettura e all’interrogazione. Per ogni topic Kafka è stata costruita una collezione documentale MongoDB relativa.

1. **Raccolta dei dati**

L’acquisizione e la memorizzazione dei dati è stata attuata utilizzando due sorgenti distinte: l’applicazione Strava, per definire gli utenti, l’attività fisica, il giorno e il luogo nel quale è avvenuta e l’API del sito *visualcrossing.com*, per trovare le indicazioni meteorologiche per le location e le giornate d’interesse.

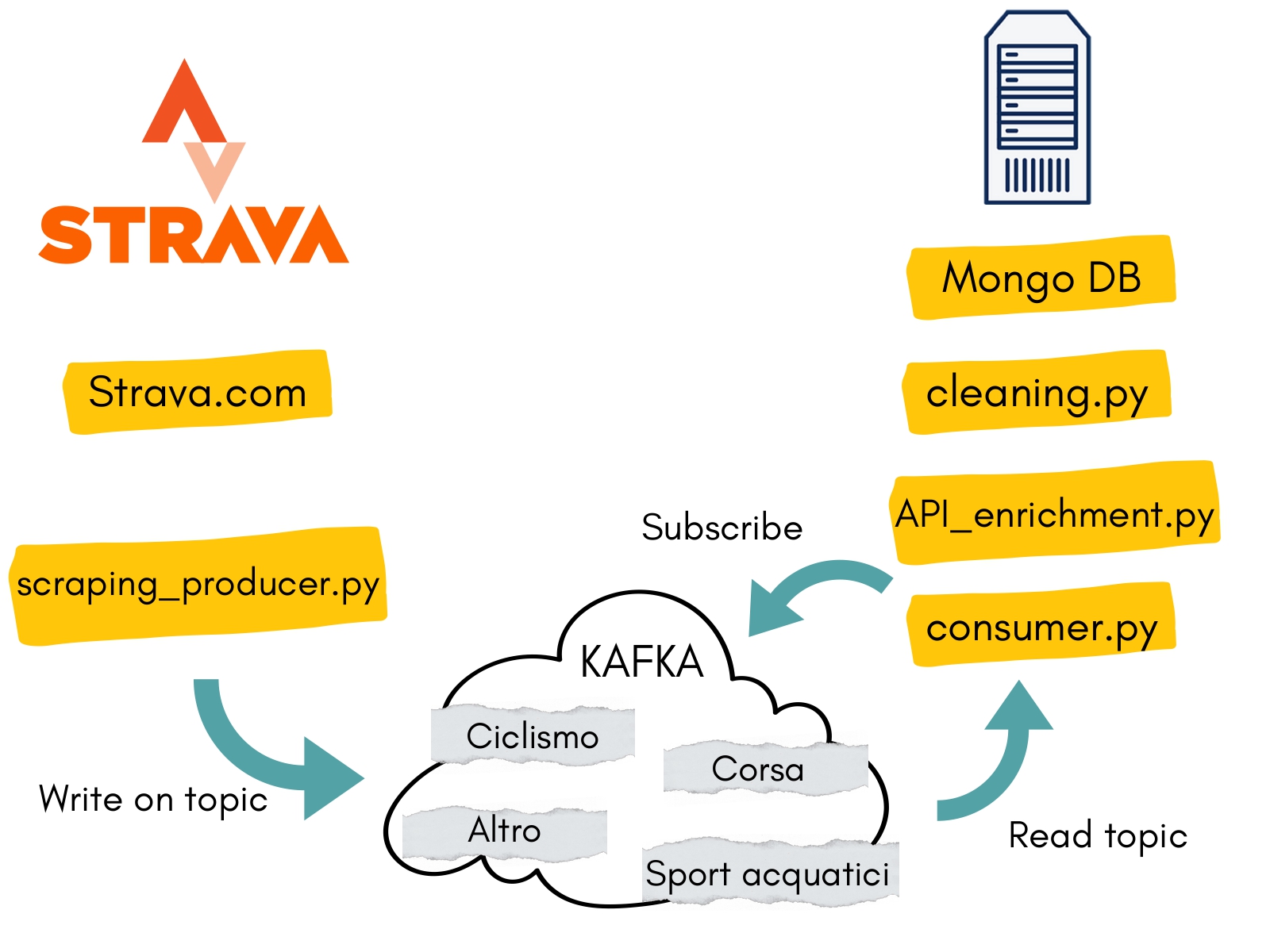


Figura 1 - Struttura del progetto

**3.1 Web Scraping per Strava**

Il primo problema riscontrato nello scaricamento dei dati da Strava è relativo alle impostazioni di privacy degli utenti. Essendo possibile creare un profilo privato, alcune attività risultano accessibili solamente dopo aver effettuato il login al sito. Per ovviare al problema, è stato creato un account fittizio, utile anche per effettuare diverse prove di creazione di attività fisiche per comprendere il funzionamento del sito. Questo ha permesso di notare come il sito associ ad ogni nuova attività un identificativo che segue un andamento progressivo ogni volta che un utente inserisce o trasmette da un dispositivo una nuova attività. Ad ognuna è associato un URL con la seguente struttura:

[*https://www.strava.com/activities/6621479194*](https://www.strava.com/activities/6621479194)

Una volta compreso il funzionamento del sito, si è potuto procedere alla creazione dell’algoritmo di scraping mediante l’utilizzo della libreria *selenium* di Python, la quale ci ha permesso di estrarre le principali informazioni che sono presenti sicuramente per ogni attività e altre che, invece, si possono considerare “facoltative” a seconda della tipologia di sport praticato. Per ciascuna attività viene creato un dizionario con la seguente struttura:

*6608453947: {'athlete': 'Miriam Bø',*

*'title': 'Lunch Walk',*

*'activity': 'Camminata',*

*'daytime': {'day\_week': 'Domenica', 'day': '30 gennaio 2022'},*

*'social': {'kudos': '0'},*

*'statistics': ['0,88km', '11:26', '12:57/km'],*

*'location': 'Sarpsborg, Norvegia',*

*'more\_statistics': ['27m', '56', '11:41']*

*}*

La chiave del dizionario rappresenta l’identificativo dell’attività, corrispondente al numero presente nel relativo URL. Gli elementi che si trovano sicuramente in tutte le attività sono il nome dell’atleta, il titolo dato all’attività eseguita e la tipologia di attività; successivamente si ha la variabile “daytime” che comprende il giorno della settimana, la data e l’orario, solo se disponibile; dopodichè si hanno il campo social, che contiene i “kudos”, ossia gli equivalenti dei “likes” sui social, e i commenti, se presenti; infine, il campo statistics comprende una lista di prestazioni sull’attività, variabili in base allo sport praticato.

Vengono raccolti, inoltre, alcuni attributi non sempre presenti, come i testi dei commenti e l’indicazione della location; spesso, ad esempio, un utente che si allena in casa non inserisce un’indicazione del luogo. Si sottolinea la decisione di mantenere nel dataset anche le attività che non presentano un’indicazione geografica, poiché considerate rilevanti al fine di valutare il numero di attività che vengono svolte in casa o in luoghi chiusi; naturalmente le attività così definite non potranno ricevere alcun tipo di integrazione con dati meteorologici.

Il processo di scraping è stato, sicuramente, l’elemento fondamentale per ottenere i dati d’interesse. Per definire la durata della raccolta e la currency, si è valutato quante fossero le attività caricate ogni minuto su Strava, in ottica di uno scaricamento, quasi in tempo reale, per definire il salto da eseguire tra un’attività e la successiva, in moda da cercare di raccogliere dati recenti. Si sono, quindi, effettuate diverse creazioni di attività, una progressiva all’altra cambiando il tempo di attesa, per valutare di quanto aumentasse il numero identificativo di ciascuna di esse e trovare quante ne venissero create ogni minuto. Per lo scraping si è deciso di fissare a 100 l’intervallo tra un identificativo e il successivo. Una volta identificato questo valore, si è proceduto con l’ascolto per 6 ore durante la giornata di giovedì 3 febbraio e sono stati raccolti oltre 5000 documenti.

**3.2 Architettura Kafka**

Una volta definito l’algoritmo di scraping, prima di eseguirlo, si è osservato che sarebbe stato possibile costruire un’architettura Kafka per acquisire e memorizzare i dati quasi in tempo reale. Infatti, poiché si tratta di un’applicazione utilizzata in tutto il mondo da milioni di persone, il numero di attività caricate è rapido e costante; inoltre, un’architettura di questo tipo avrebbe permesso la possibilità di visualizzare dati sempre aggiornati al momento in cui si accede al database.

Si è allora definito uno script all’interno dell’algoritmo di scraping per la creazione del Producer Kafka. Quest’ultimo invia al server Kafka le singole attività con il comando *producer.send(‘NomeTopic’)* una volta terminata la loro acquisizione. Si sottolinea che i messaggi vengono suddivisi per sport, tra cui corsa, ciclismo e sport acquatici e, quindi, inviati a topic Kafka distinti, di modo da creare una partizione di messaggi per ogni categoria di attività. Le varie partizioni vengono rese fruibili al Consumer Kafka, il quale può accedere al server tramite *socket* per fare operazioni enrichment e di quality condizionatamente ai topic. Per ogni topic si è creato un Consumer che fornisce in output un dizionario da usare come input per le fasi successive. Si fa presente che l’architettura è stata creata in locale, dopo aver verificato che per l’esecuzione di alcune operazioni non si avevano determinati permessi all’interno della Virtual Machine.

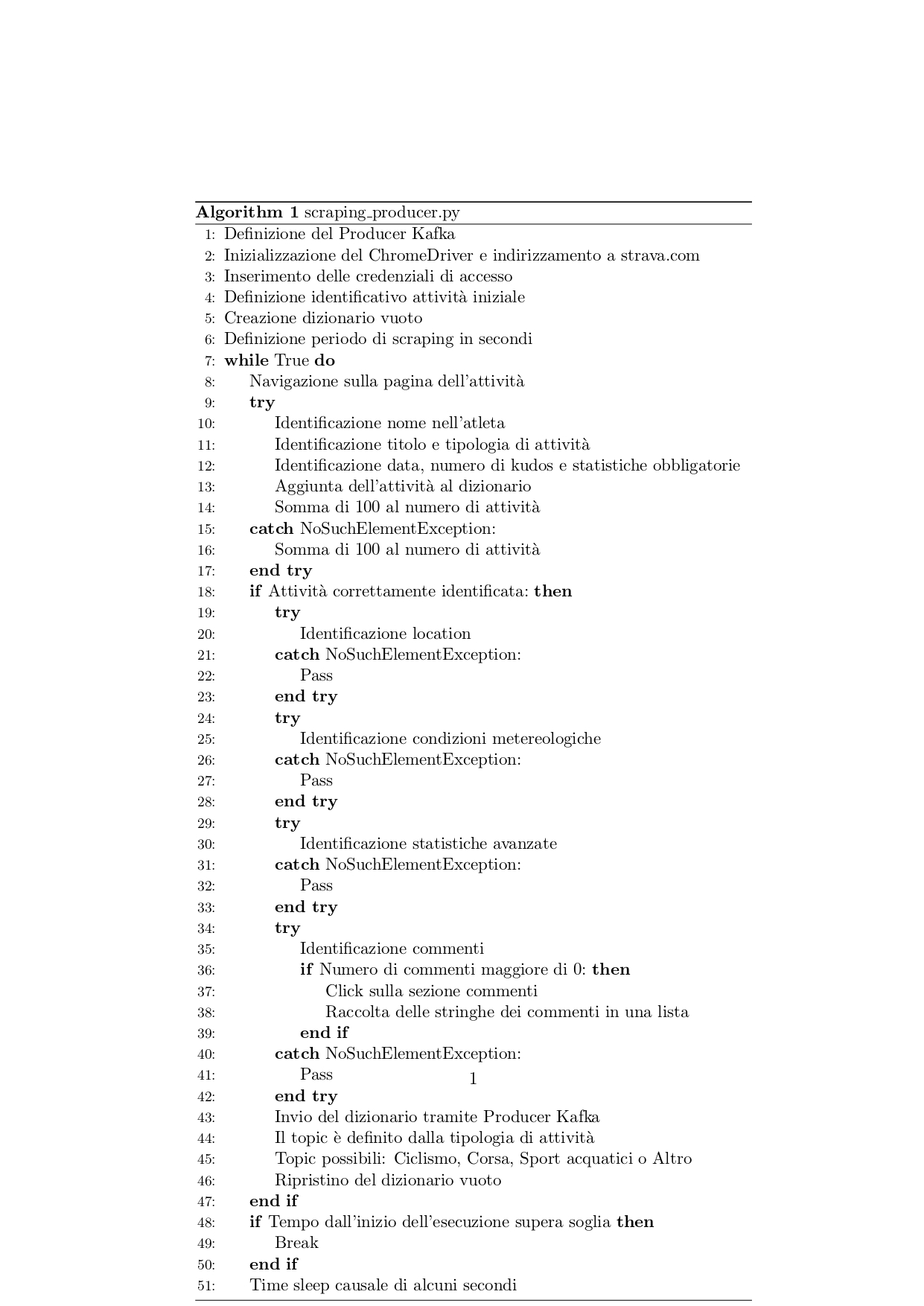


Figura 2 - Pseudocodice riguardante l’algoritmo di scraping e architettura Kafka

**3.3 Chiamate Weather API**

I dati riguardanti le previsioni meteorologiche per località e per giorno sono stati ricavati utilizzando una richiesta al servizio Weather API fornito dal sito *visualcrossing.com*, formalizzata come di seguito:

[**https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/weatherdata/history?&aggregateHours=24&startDateTime=2019-06-13T00:00:00&endDateTime=2019-06-20T00:00:00&unitGroup=uk&contentType=csv&dayStartTime=0:0:00&dayEndTime=0:0:00&location=Sterling,VA,US&key=YOURAPIKEY**](https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/weatherdata/history?&aggregateHours=24&startDateTime=2019-06-13T00:00:00&endDateTime=2019-06-20T00:00:00&unitGroup=uk&contentType=csv&dayStartTime=0:0:00&dayEndTime=0:0:00&location=Sterling,VA,US&key=YOURAPIKEY)

dove le variabili da inserire per la richiesta sono l’indicazione del giorno da ricercare, la località e la key che permette le chiamate. É presente, inoltre, la possibilità di aggregare le informazioni relative all’intera giornata in un unico dato in modo da garantire un’accuratezza maggiore per ogni singola attività, poiché per la maggior parte di esse non è presente l’orario durante la quale è stata eseguita.

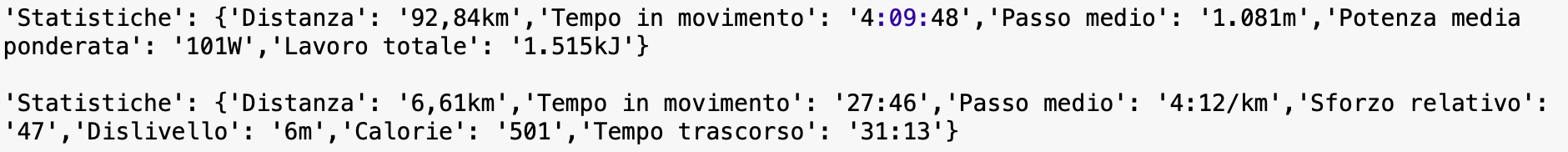
In generale, il sito Visual Crossing è una delle fonti più affidabili e facili da utilizzare poiché raccoglie e condivide una grandissima quantità di dati meteo storici e previsionali, e, appunto, la Weather API è progettata per integrarsi facilmente in qualsiasi editor o codice.

1. **Qualità dei dati**

Per quanto riguarda la Data Quality, una volta ottenuti i dati direttamente da Strava ed averli divisi nei diversi dataset in base al topic di appartenenza, è stato necessario effettuare alcune operazioni di pulizia e, soprattutto, di valutazione della qualità. Infatti, si è insistito molto al fine di ottenere un dataset il più possibile completo e che non presentasse dei valori non corretti per i diversi campi legati a ogni attività. Analizzando i dataset si sono, quindi, riscontrati problemi legati in particolare alle dimensioni di consistenza, completezza e accessibilità.

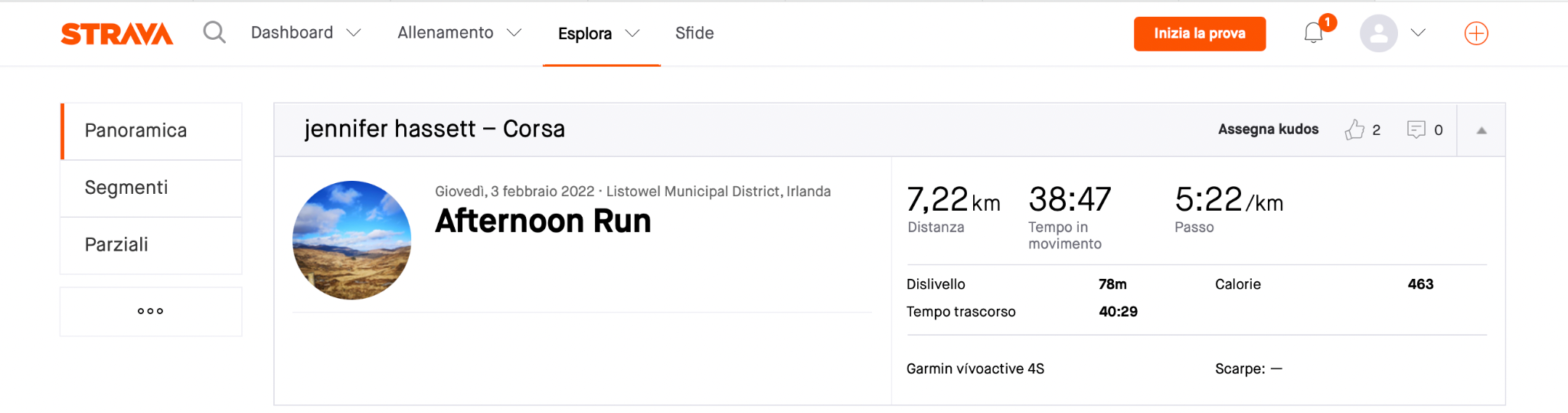
Prima di procedere con l’analisi, è doveroso sottolineare che alcune operazioni per migliorare la qualità del dato sono state eseguite sui dataset ottenuti tramite l’operazione di scraping su Strava, mentre altre sono state effettuate sui dataset arricchiti grazie ai dati ottenuti con chiamate API dal sito Visual Crossing.

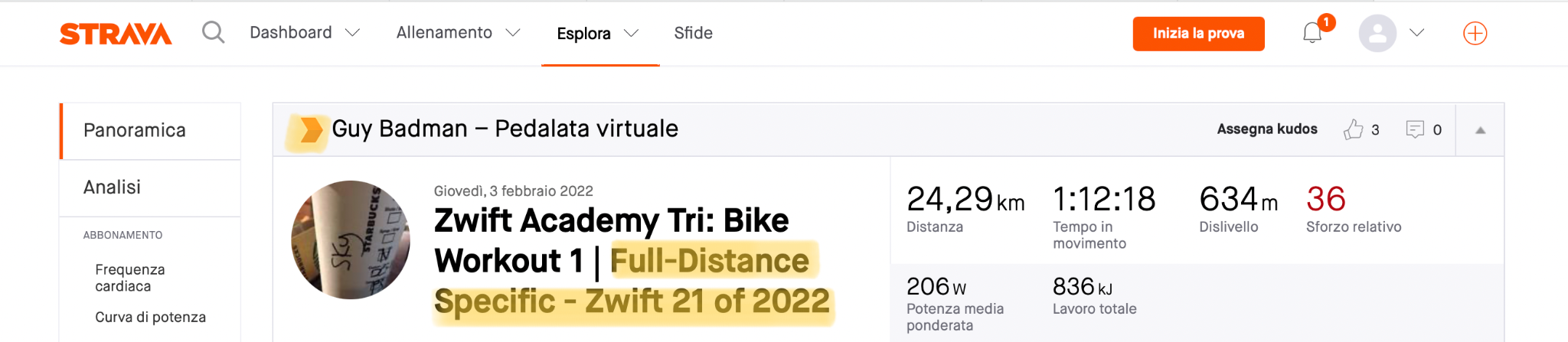
Una delle prime operazioni effettuate, da considerarsi come fase di pulizia del dato piuttosto che miglioramento della qualità, è stata l’importazione di tutte quelle statistiche etichettate dal campo *statistics\_advanced* o *more\_statistics* all’interno del dizionario che presenta come chiave il valore *statistics*. Infatti, essendo dei valori non presenti per tutte le attività, in fase di scraping essi sono stati inseriti all’interno di liste diverse per avere una struttura posizionale da cui ricavare il significato di ogni attributo. Dopo aver assegnato una chiave ad ogni valore, si è ritenuto opportuno uniformarli all’interno della chiave *statistics* (poi tradotto in *Statistiche*).



Per quanto riguarda i dati ottenuti mediante web scraping, sono emersi diversi problemi legati al concetto di consistenza, tra cui la presenza del valore dell’anno, della località o di parte del titolo relativo all'attività all’interno del field *day*; probabilmente questa imprecisione è stata causata da una differente struttura della pagina HTML per un numero limitato di attività. Una volta individuato l’errore e le chiavi delle attività interessate, si è corretto il problema modificando il valore relativo al giorno durante il quale è stata eseguita l’attività.

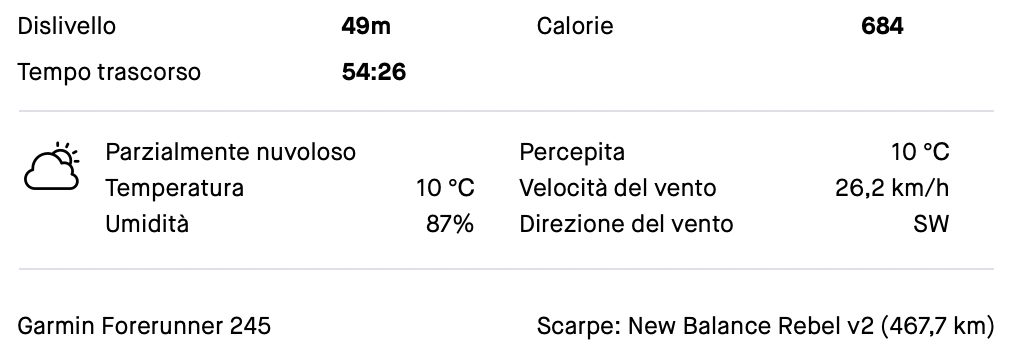
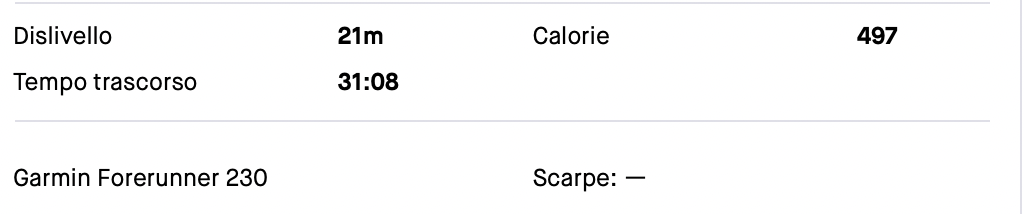
Successivamente si è riscontrata un’imprecisione nel nome di alcune attività, poiché è emerso che era presente una parte del titolo dell’allenamento. Anche in questo caso il problema era, molto probabilmente, causato dalla diversa impaginazione del formato HTML e si è, quindi, proceduto a correggere il field d’interesse. Si può affermare come il problema sia stato causato dalla struttura leggermente diversa dell’HTML perché si è eseguita una ricerca mirata per le attività che presentavano questo tipo di problema e si è notato come per alcuni atleti, in particolare quelli possessori di un account Premium, la pagina iniziale presentasse delle leggere modifiche.



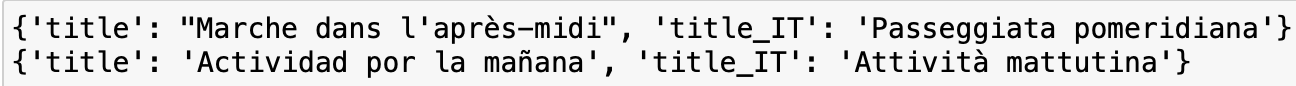


Infine, uno degli ultimi controlli effettuati sul dataset non ancora integrato è stato quello relativo alla validità delle statistiche, sempre per quanto riguarda la dimensione della consistenza. Il software Strava, infatti, per determinati allenamenti come attività in casa o in modo virtuale impone all’utente di inserire, ad esempio, la distanza del percorso sul quale ci si è allenati; ciò implica che alcune statistiche siano fittizie o comunque non veritiere (ad esempio, tempo in movimento pari a 0 secondi e distanza pari a 10 km) e si è deciso di escludere tutti quei valori non verosimili prossimi allo zero per mantenere una struttura flessibile o l’intera attività in caso presentasse sia distanza che tempo nulli.

Dopo aver migliorato la qualità del dato con riferimento alla dimensione della consistenza ed essersi assicurati che non fossero presenti altri errori simili, si è proceduto con l’integrazione e arricchimento dei dati tramite chiamate API. In questa situazione non si è riscontrato alcun tipo di problema riguardante la correttezza dei dati, anche perché le informazioni ottenute sono state ricavate da un archivio di dati storici molto affidabile. La qualità del dato è stata, quindi, perfezionata sull’aspetto della completezza; infatti, molte attività per le quali è presente una località non presentano valori relativi al meteo come temperatura o umidità a causa, probabilmente, di una mancata strumentazione che supporta questo tipo di informazioni. Mediante l’operazione di enrichment si è, quindi, anche migliorata la disponibilità di informazioni per quanto riguarda tutte quelle attività che presentano un’indicazione sulla località ma non presentano un’indicazione sulle condizioni meteorologiche.



Una volta completate tutte le attività che presentavano questa possibilità, si è voluto anche ottimizzare il dato per quanto riguarda la dimensione dell’accessibilità. Infatti, essendo Strava un’applicazione usata da utenti di tutto il mondo, i titoli delle attività e i commenti relativi all’aspetto social del software sono personalizzabili e, quindi, si presentano nella lingua madre di chi li scrive. Per migliorare questo aspetto e rendere tutti i dati accessibili agli utenti italiani si è utilizzata la libreria *deepl* di Python che permette la traduzione automatica da qualunque lingua a una lingua desiderata. Una volta tradotti i testi, i commenti e le descrizioni delle condizioni meteo essi sono stati importati all’interno del dataset sostituendo gli attributi nella lingua originale.

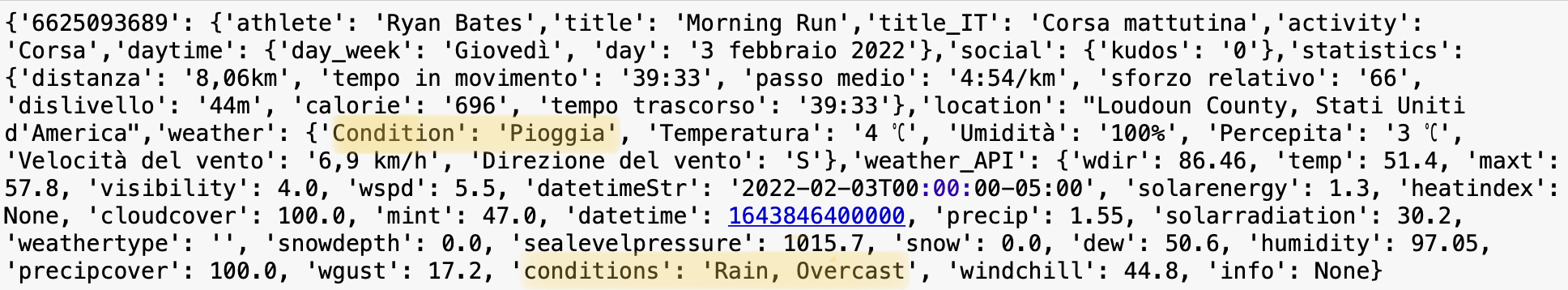


Un altro aspetto che si è voluto evidenziare nell’ambito dell’accessibilità è la conversione delle unità di misura ottenute tramite API in un formato più facilmente comprensibile da utenti italiani, come la conversione da gradi Fahrenheit a gradi Celsius, oppure da miglia al secondo a metri al secondo. Infine, dato che dallo scraping si erano ottenuti dei nominativi per i diversi campi in lingua inglese, si è deciso di tradurli in lingua italiana.

1. **Data Enrichment**

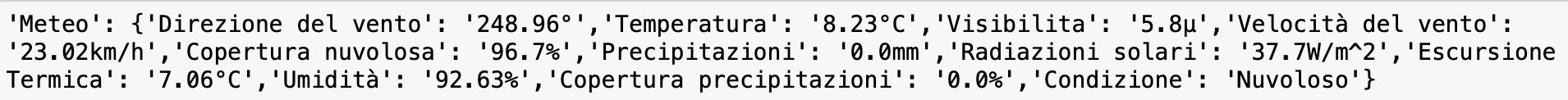
Una volta ottenuta la struttura dati contenente le informazioni relative alle diverse attività e aver applicato i possibili miglioramenti di qualità del dato citati precedentemente, è possibile procedere alla fase d’integrazione e arricchimento del dataset. Utilizzando il codice presente sul file *API\_enrichment.py* si effettua l’integrazione diretta sui dataset di partenza. Infatti, dato che si è utilizzato il metodo con le chiamate API, non si è ritenuto sensato creare un dataset con i valori del meteo per tutte le località in una certa data ma, piuttosto, anche a causa del numero giornaliero limitato di chiamate, si è preferito richiamare singolarmente per ogni attività che presentasse la località e inserire direttamente all’interno dei dataset di partenza tutti i dati relativi per la data in analisi. In questo modo non solo si risparmia tempo, ma si è anche certi riguardo alla corretta integrazione dei dati.

Questo tipo di operazione ha, quindi, permesso non solo di rendere complete quelle attività che non presentavano dati meteorologici, ma ha soprattutto permesso di arricchire il dataset di partenza con informazioni nuove e interessanti per gli utenti che vorranno avere accesso a questo tipo di attività.



Da questo esempio si evince come i valori per temperatura, umidità e velocità del vento siano molto simili; è opportuno ricordare che per ottenere i dati mediante API si sono aggregate tutte le 24 ore del giorno durante il quale è avvenuta l’attività, dato che non si dispone dell’ora precisa, e quindi è possibile non riscontrare lo stesso identico valore. Oltre alle corrispondenze tra i valori, si nota come le Weather API abbiano introdotto una diversa varietà di informazioni come visibilità, molto importante per chi pratica sport all’aperto, soprattutto in luoghi elevati o in città, oppure copertura nuvolosa combinato con la potenza delle radiazioni o energia solare permettono di capire quanto fosse solare e luminosa quella giornata indipendentemente dal valore di temperatura. É, inoltre, interessante il dato relativo alla quantità di precipitazioni e alla presenza di neve, di modo da sapere con maggiore precisione le condizioni “estreme” sotto le quali è stata svolta l’attività. Infine, un dato che potrebbe interessare a chi pratica sport acquatici marini quali surf o windsurf è quello relativo alle raffiche di vento (*wgust*).

Si sottolinea, infine, che non sono stati mantenuti tutti i valori ottenuti con le chiamate API, ma solamente quelli ritenuti di maggior interesse. Inoltre, per quelle attività che presentavano già dei valori per il dizionario *weather* ottenuto da Strava corrispondenti a valori del dizionario *weather\_API,* si è deciso di mantenere i dati di partenza poiché ritenuti maggiormente precisi dato che si riferiscono all’ora precisa della giornata, ad eccezione del dato relativo alla direzione del tempo. I campi relativi al meteo mantenuti sono mostrati in figura.



Una volta terminato il processo di integrazione, ed applicata il processo di Data Quality (consultabile nel file *Quality\_enrichment.py*) descritto nel capitolo precedente, il dataset finale viene caricato sul server MongoDB nel database chiamato Strava, suddiviso a sua volta in 4 diverse collection in base all’attività sportiva eseguita.

1. **Storage su MongoDB**

Una volta ottenuti dati puliti ed arricchiti relativi alla giornata del 3 febbraio 2022 riguardanti le informazioni legate alle attività fisiche eseguite da ogni utente, si è potuto immagazzinare i dati direttamente sul software MongoDB. É opportuno sottolineare che, prima della fase di caricamento dati, si è deciso di modificare la struttura del dizionario relativo al codice attività. Infatti, per facilitare l'interrogazione dei dataset tramite query su Mongo, si è creata una lista di dizionari e la chiave iniziale, che identificava l’attività, è stata inserita all’interno della lista come valore della chiave *id*.

A questo punto i dati sono stati caricati mediante il comando *collection.insert\_one()* all’interno di una collezione documentale; avendo creato 4 diversi topic su Apache Kafka, anche in MongoDB sono state create 4 collezioni distinte in modo da rendere indipendente l’accesso ai dati limitatamente allo sport di interesse. Si è deciso di adottare questo sistema di DBMS per garantire la memorizzazione di ingenti quantità di dati che si potrebbero accumulare in poco tempo data la grande mole di utenti e di attività praticate e, soprattutto, per mantenere la struttura flessibile del dato. Infatti, MongoDB, in quanto modello NoSQL, permette una struttura dati schema-free, molto importante per l’analisi in questione poiché non tutte le attività presentano gli stessi campi come *Location* o le diverse statistiche presenti all’interno del campo *Statistiche* che variano in base a sport effettuato e, soprattutto, in base alla strumentazione di cui si disponee, in questo modo, si evita di perdere informazioni o di introdurre campi con valori mancanti.

Viene riportato un esempio di documento presente nella collection Ciclismo:

*{*

*"ID": "6625110889",*

*"Atleta": "JOHN FREDY MORA",*

*"Attività": "Ciclismo",*

*"Tempo": {*

*"Giorno": "3 febbraio 2022",*

*"Giorno Settimana": "Giovedì"*

*},*

*"Social": {*

*"Kudos": "7",*

*"Commenti": [*

*"👍"*

*]*

*},*

*"Statistiche": {*

*"Distanza": "31,53km",*

*"Tempo in movimento": "1:15:32",*

*"Potenza media ponderata": "162W",*

*"Lavoro totale": "734kJ",*

*"Dislivello": "651m"*

*},*

*"Location": "Mosquera, Colombia",*

*"Meteo": {*

*"Direzione del vento": "248.96°",*

*"Temperatura": "8.23°C",*

*"Visibilità": "5.8μ",*

*"Velocità del vento": "23.02km/h",*

*"Copertura nuvolosa": "96.7%",*

*"Precipitazioni": "0.0mm",*

*"Radiazioni solari": "37.7W/m^2",*

*"Escursione Termica": "7.06°C",*

*"Umidità": "92.63%",*

*"Copertura precipitazioni": "0.0%",*

*"Condizione": "Nuvoloso"*

*}*

*}*

1. **Conclusioni e Sviluppi Futuri**

In generale si può affermare di aver ottenuto un dataset consistente, completo, accurato e accessibile da tutti gli utenti italiani per l’analisi di attività sportive raccolte in near real-time durante l’arco di una giornata. Si può considerare sicuramente soddisfacente il risultato raggiunto perché si sono creati diversi algoritmi avanzati per lo scaricamento di dati in modo automatizzato direttamente dal web e, contemporaneamente, si è progettata un’architettura Kafka che permette di effettuare scraping in tempo reale con il caricamento delle attività da parte degli utenti. Inoltre, i dati non solo sono stati migliorati con riferimento alle dimensioni della qualità, ma soprattutto integrati e arricchiti con informazioni provenienti da un archivio storico di dati meteo utilizzando il metodo delle chiamate API.

Si sono, inoltre, ipotizzati alcuni possibili sviluppi futuri per migliorare maggiormente la struttura dati creata. Innanzitutto, uno dei miglioramenti tecnicamente già implementati nel codice è quello di poter ampliare sempre di più il dataset, dato che non ci sono limiti né di spazio né di schema grazie al caricamento dei dati su MongoDB.

Un’altra idea potrebbe essere quella di migliorare il procedimento di integrazione e di enrichment, in particolare per alcune delle attività per cui non è stato possibile effettuare la chiamata API per via della scarsa chiarezza del campo *location* associato. Per ovviare a questo problema vi sono due possibili sviluppi futuri: il primo consiste nell’implementare un algoritmo con l’ausilio di RegEx che filtri meglio le stringhe relative alla location; il secondo consiste nell’applicare un processo di geocoding che google mette a disposizione tramite chiamate API col fine di ottenere le convertite il nome della località in coordinate geografiche e, successivamente, usare latitudine e longitudine per effettuare le “API call” al portale Visual Crossing. A causa di questa problematica legata alla sintassi delle località, alcuni dati relativi al meteo non possono essere integrati, nonostante il dato sia presente.

Un ulteriore sviluppo futuro potrebbe essere quello di trovare nuove possibilità d’integrazione, come l’identificazione di luoghi o tracciati molto frequentati dagli sportivi in base alla località, per permettere a chi accede alla lettura dei dati di poter trovare nuove località in cui allenarsi in base all’attività sportiva da svolgere.

Infine, l’ultimo sviluppo futuro è dettato da un processo di Sentiment Analysis, difatti, in fase di scraping abbiamo scelto di raccogliere i commenti relativi ad ogni singola attività e sarebbe interessante analizzarne le emozioni trasmesse.

**Appendice - Librerie Python utilizzate**

#kafka-python #platform

#selenium #random

#webdriver\_manager #pymongo

#cgi #json

#daytime #pickle

#time #requests

#deepl #numpy