

A thick dark green vertical bar runs along the left edge of the page. In the lower-left quadrant, several thin, curved lines in dark green and light grey sweep upwards and to the right, creating an abstract, organic shape.

Modelli e Tecniche per i Big
Data

PROGETTO: Analisi Dati Meteorologici

Apache Spark

Francesco Arcuri (235125) – Fulvio D'Atri (235344)
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Sommario

1.	Introduzione.....	2
1.1.	Analisi del contesto	2
1.1.1.	Rilevanza dell'analisi dei dati meteorologici.....	2
1.1.2.	Tecniche utilizzate nell'analisi dei dati meteorologici	2
1.2.	Dataset del progetto	3
1.3.	Tecnologie utilizzate.....	3
1.3.1.	Front-end:	3
1.3.2.	Back-end:.....	4
2.	Front-end	5
2.1.	Select Stations	6
2.2.	Climate Summary	7
3.	Preprocessing.....	10
3.1.	Station dataset	10
3.2.	Hourly dataset	11
4.	Spark jobs.....	13
4.1.	Caricamento dataset job.....	13
4.2.	Clustering job	14
5.	Conclusioni e sviluppi futuri.....	16

1. Introduzione

1.1. Analisi del contesto

L'analisi di dati meteorologici è di grande importanza e rilevanza in diversi settori, tra cui la meteorologia stessa, la climatologia, l'agricoltura, l'energia, la gestione delle risorse idriche, il turismo e molte altre discipline. L'analisi di dati meteorologici può fornire un contesto interessante e ricco di sfide.

L'analisi dei dati meteorologici è il processo di acquisizione, archiviazione, elaborazione e interpretazione di dati raccolti da stazioni meteorologiche, satelliti, radar e altri dispositivi di monitoraggio. Questi dati comprendono informazioni sulle condizioni atmosferiche, quali temperatura, umidità, pressione atmosferica, vento, precipitazioni, nebulosità e molti altri parametri. L'obiettivo dell'analisi è di comprendere meglio il comportamento dell'atmosfera, prevedere il tempo, studiare i cambiamenti climatici e supportare decisioni in vari settori.

1.1.1. Rilevanza dell'analisi dei dati meteorologici

- **Previsioni meteorologiche:** l'analisi dei dati meteorologici è essenziale per le previsioni del tempo. L'uso di modelli matematici e algoritmi di apprendimento automatico su grandi dataset consente di fornire previsioni più accurate, che sono fondamentali per la pianificazione di attività quotidiane, per agricoltura, trasporti e pubblica sicurezza;
- **Monitoraggio dei cambiamenti climatici:** gli scienziati utilizzano l'analisi dei dati meteorologici per studiare i cambiamenti climatici a lungo termine. Questi dati contribuiscono a identificare tendenze, anomalie e impatti dei cambiamenti climatici globali, come l'aumento delle temperature medie e i fenomeni meteorologici estremi;
- **Gestione delle risorse naturali:** l'agricoltura, la gestione delle risorse idriche e la produzione di energia dipendono dalle condizioni meteorologiche. L'analisi dei dati aiuta a ottimizzare l'uso delle risorse naturali, ad esempio, per decidere quando irrigare i campi agricoli o dove pianificare la produzione di energia solare o eolica;
- **Sicurezza e prevenzione dei disastri:** l'analisi dei dati meteorologici è cruciale per prevedere e gestire situazioni di emergenza, come tempeste, alluvioni e incendi boschivi. Fornisce informazioni vitali per evacuazioni e prevenzione di situazioni di emergenza.

1.1.2. Tecniche utilizzate nell'analisi dei dati meteorologici

L'analisi dei dati meteorologici richiede l'uso di tecnologie avanzate. Alcune delle tecniche e degli strumenti comuni includono:

- **Elaborazione dei dati in tempo reale:** l'analisi dei dati meteorologici in tempo reale è fondamentale per le previsioni meteorologiche. I dati grezzi vengono catturati, filtrati e analizzati in tempo reale utilizzando sistemi ad alta velocità;
- **Modelli meteorologici numerici:** questi modelli utilizzano equazioni fisiche per simulare il comportamento dell'atmosfera e generare previsioni. I big data sono utilizzati per inizializzare e validare questi modelli;
- **Apprendimento automatico e intelligenza artificiale:** l'IA viene utilizzata per migliorare le previsioni meteorologiche, rilevare modelli climatici complessi e analizzare grandi dataset storici;

- **Visualizzazione dei dati:** gli strumenti di visualizzazione dei dati aiutano a comunicare in modo efficace le informazioni meteorologiche complesse ai decisori e al pubblico;
- **Archiviazione e gestione dei dati:** a causa della grande quantità di dati meteorologici generati quotidianamente, è essenziale avere sistemi di archiviazione e gestione dati robusti e scalabili.

1.2. Dataset del progetto

La banca dati di partenza è un archivio storico dei dati meteorologici misurati da stazioni terrestri sul territorio USA dell'anno 2013. Questi dati sono raggruppati per mese e a loro volta si suddividono in tre categorie:

- **Hourly:** misurazioni puntuali dei dati, effettuate appunto ogni ora;
- **Daily:** misurazioni aggregate per giorno, riportando dati medi, valori massimi e minimi di giornata (dove questi abbiano significato e/o interesse);
- **Monthly:** aggregazione per mese.

I Dati così organizzati sono presenti per ogni stazione, che viene identificata attraverso il codice WBAN. Infatti, il dataset comprende anche il dataset **Station** che descrive tutte le caratteristiche delle singole stazioni meteorologiche, quali WBAN, coordinate, nome, stato, ecc.

Per la realizzazione del progetto si è deciso di utilizzare la versione meno aggregata dei dati (il dataset **Hourly**) per avere maggiore grado di libertà nelle aggregazioni da effettuare, oltre al fatto di poter sfruttare maggiormente le prestazioni di Apache Spark, avendo un dataset con un maggior numero di righe. Oltre a questo dataset, si è fatto uso anche del dataset **Station**, così da poter sfruttare anche analisi geolocalizzate.

La scelta di questi dataset è stata fatta non solo per avere la massima personalizzazione nell'elaborazione dei dati e operare un utilizzo più massiccio di Spark, ma anche per realizzare un applicativo che fosse in grado di lavorare in diverse condizioni. Infatti, servendosi di questa soluzione, non vi è necessità di avere dati storici già registrati, ma si può utilizzare l'applicativo anche con stream di dati.

1.3. Tecnologie utilizzate

1.3.1. Front-end:

Per la realizzazione del Front-end si è fatto uso della libreria *dash* di python. In particolare, la realizzazione del FE non è stata effettuata da zero, bensì, attraverso il compimento di diverse ricerche nel mondo open source, si è trovata un'applicazione realizzata in *dash* con *python* specifica sull'analisi di dati meteorologici. Per tale ragione, si è deciso di modificare quest'applicazione per lo scopo del progetto in questione. Così si è dato maggiore focus all'implementazione del back-end con l'utilizzo di Spark (obiettivo principale del progetto) e meno all'implementazione della parte estetica dell'applicazione.

L'applicazione open source modificata è la seguente: <https://cbe-berkeley.gitbook.io/clima/>.

Il codice GitHub: <https://github.com/CenterForTheBuiltEnvironment/clima>.

1.3.2. Back-end:

Per la realizzazione del back-end si è fatto uso di PySpark, che è la libreria Python per interagire con il framework Spark, come richiesto dalla traccia del progetto.

Nel particolare PySpark è stato impiegato in due fasi chiave dell'applicazione:

- **Caricamento dei Dati:** PySpark è stato utilizzato per effettuare il caricamento dei dati meteorologici in un DataFrame distribuito. Questa procedura ha consentito di gestire grandi quantità di dati in modo efficiente, eseguendo contemporaneamente operazioni di trasformazione e pulizia dei dati;
- **Esecuzione dell'algoritmo di Clustering K-Means:** la potenza computazionale di Spark, attraverso l'utilizzo di PySpark, è stata sfruttata per l'esecuzione dell'algoritmo di clustering K-Means sui dati meteorologici. Ciò ha reso possibile il raggruppamento delle stazioni meteorologiche in base alle condizioni climatiche in maniera rapida e scalabile.

2. Front-end

L'applicazione realizzata consta di un'interfaccia grafica web-based, costituita da un insieme di tab view. Ogni tab rappresenta una pagina diversa dell'applicazione, che sono le seguenti:

- **Select Stations:** è la pagina principale, da cui parte tutta l'analisi dei dati. È necessaria per selezionare la stazione o il cluster di stazioni di cui si vuole andare a visualizzare le misurazioni nelle altre pagine;
- **Climate Summary:** è una pagina che riporta il sommario della selezione appena effettuata nella pagina *Select Stations*. Offre una panoramica completa delle informazioni rilevanti sulle stazioni meteorologiche selezionate, facilitando l'individuazione della loro posizione, dei requisiti di riscaldamento e raffreddamento mensili e delle variazioni nelle misure meteorologiche durante il giorno e la notte;
- **Temperature and Humidity:** in questa pagina sono presenti grafici per la visualizzazione di misure riferite a temperatura e umidità della/e stazione/i selezionata/e;
- **Sun Path:** in questa pagina è presente un diagramma solare, nel quale vengono visualizzati i valori di misure meteorologiche lungo la traiettoria apparente del sole nel cielo durante l'arco di una giornata e durante l'anno, nella posizione geografica relativa alla stazione selezionata;
- **Wind:** questa pagina è incentrata sull'analisi del vento, in particolare in ordine a direzione e velocità;
- **Psychrometric Chart:** consente all'utente di visualizzare tutte le condizioni meteorologiche annuali su un diagramma psicrometrico. È uno strumento utilizzato in meteorologia e nell'ingegneria dell'aria condizionata per rappresentare le proprietà termodinamiche e umide dell'aria. Questo grafico fornisce una rappresentazione visuale delle variazioni di temperatura, umidità relativa, umidità specifica, pressione atmosferica e altre grandezze relative all'aria in un ambiente specifico;
- **Natural Ventilation:** permette di stimare le ore in cui gli ambienti interni possono essere ventilati, con aria esterna, in funzione della temperatura esterna;
- **Outdoor Comfort:** la pagina mostra una panoramica delle condizioni ambientali percepite in base al modello UTCI. L'Indice Universale di Clima Termico (UTCI), introdotto nel 1994, mira a misurare la reazione fisiologica dell'uomo all'ambiente atmosferico;
- **Data Explorer:** l'ultima pagina, consente un approfondimento di tutti i dati presenti nel dataframe climatico. La scheda è suddivisa in tre sezioni:
 - Analisi di una singola variabile;
 - Analisi a singola variabile + filtro;
 - Analisi a tre variabili + filtro.

Per un maggiore approfondimento delle pagine, si consiglia di visitare il seguente sito web: <https://cbe-berkeley.gitbook.io/clima/>. Nelle prossime sezioni, ci si concentrerà esclusivamente sulle prime due pagine, in quanto sono state oggetto delle modifiche più significative, principalmente rispetto all'aggiunta della funzionalità di calcolo dei cluster di stazioni non presente nell'applicazione originale.

2.1. Select Stations

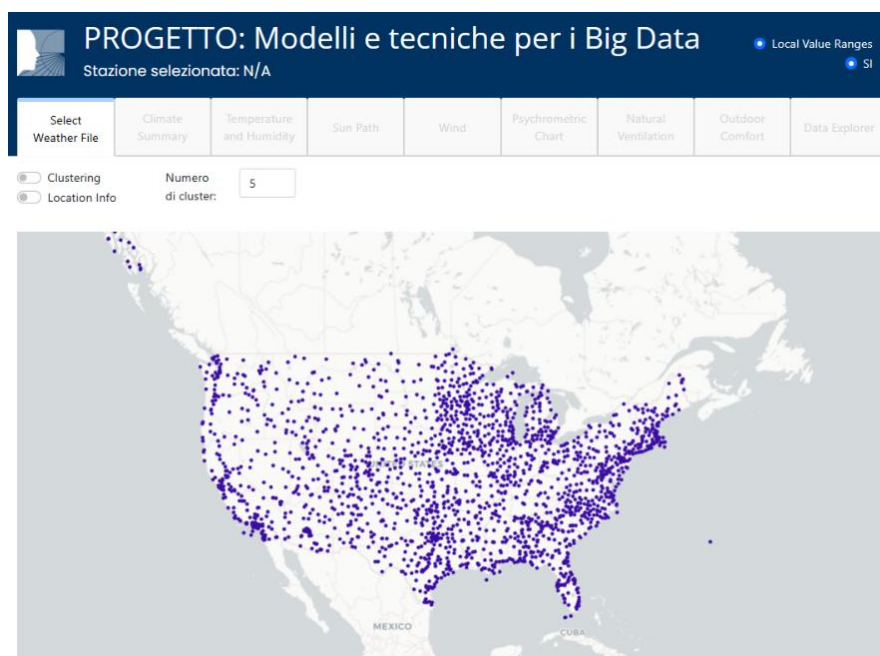
Come precedentemente menzionato, *Select Stations* è la pagina principale dell'applicazione che svolge un ruolo centrale nell'analisi dei dati. Essa funge da punto di partenza per condurre l'analisi, consentendo agli utenti di selezionare la stazione meteorologica specifica o il cluster di stazioni di loro interesse e visualizzare le relative misurazioni nelle altre schede. La pagina include una mappa che mostra le posizioni delle stazioni meteorologiche disponibili per la selezione.

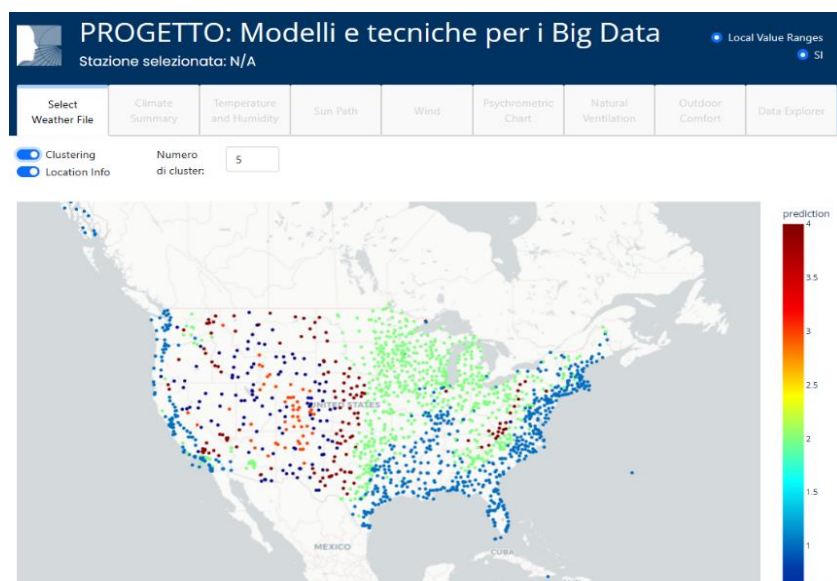
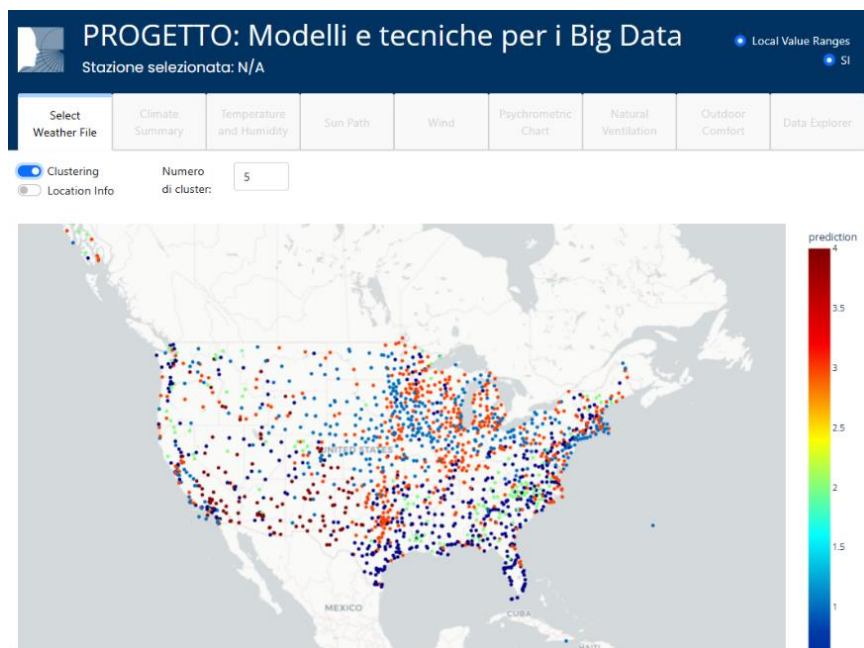
Sopra la mappa, sono presenti due pulsanti: *Clustering* e *Location Info*. Questi pulsanti consentono agli utenti di selezionare non solo singole stazioni, ma anche cluster di stazioni. Attivando il pulsante *Clustering*, si avvia un processo Spark che applica l'algoritmo di clustering Kmeans alle misurazioni delle stazioni, come verrà descritto di seguito. Il pulsante *Location Info* permette di decidere se considerare o meno la posizione geografica (latitudine, longitudine e altitudine) delle stazioni nel calcolo dei cluster. Inoltre, è disponibile un campo di testo in cui specificare il numero di cluster k da utilizzare nell'algoritmo.

In modalità standard, gli utenti possono selezionare immediatamente la stazione che desiderano esaminare. In modalità clustering, è possibile selezionare il cluster di interesse solo dopo l'esecuzione dell'algoritmo Kmeans. Per selezionare un cluster, è sufficiente scegliere una qualsiasi stazione appartenente a tale cluster.

Una volta selezionata la stazione o il cluster di stazioni, saranno disponibili le altre schede per visualizzare le analisi delle misurazioni relative a tali stazioni.

Nelle figure sottostanti sono mostrati alcuni screen della pagina nelle diverse modalità (singola stazione, clustering e clustering con location):





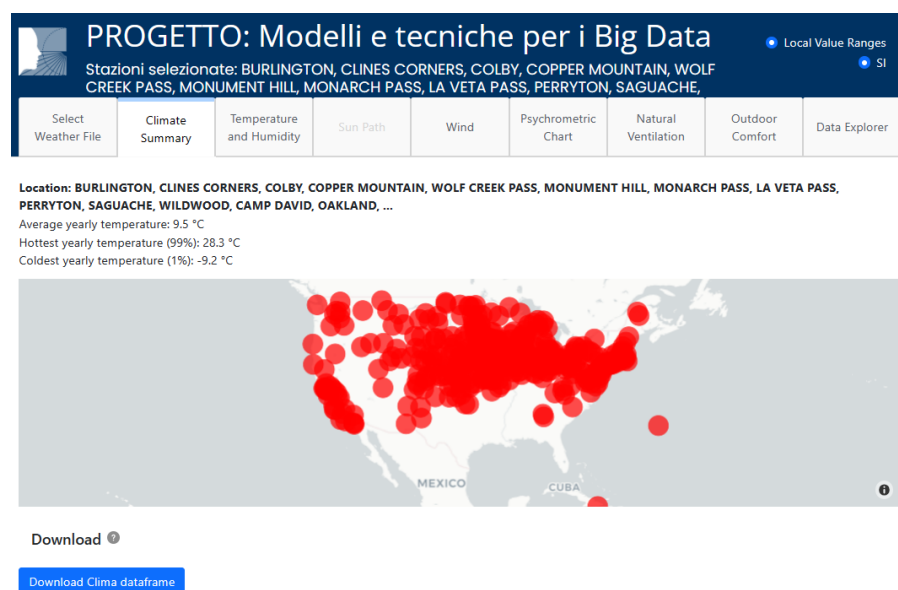
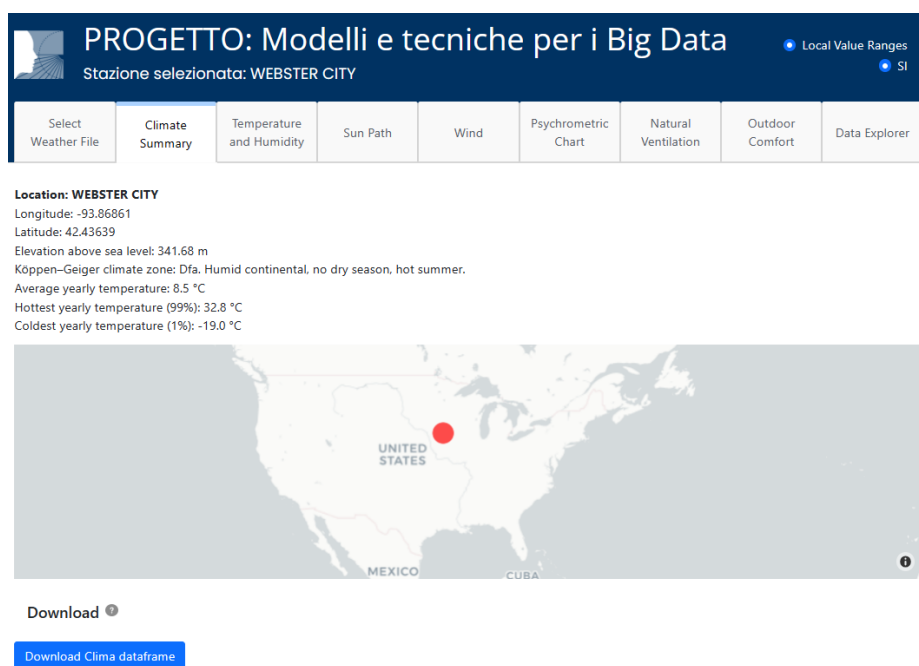
2.2. Climate Summary


La pagina *Climate Summary* ha l'obiettivo di fornire un sintetico e puntuale riepilogo delle stazioni o della stazione selezionata. Questa pagina è suddivisa in quattro sezioni:

1. **Informazioni testuali:** nella prima sezione vengono presentate in formato testuale alcune caratteristiche rilevanti della stazione o del cluster di stazioni selezionate;
2. **Mappa:** la seconda sezione presenta una mappa con la quale poter visualizzare la posizione geografica della stazione o delle stazioni selezionate. Questo è particolarmente utile in modalità clustering perché consente di comprendere facilmente la disposizione delle stazioni all'interno di quel cluster, senza confondere con stazioni appartenenti ad altri cluster;

3. **Grafico a barre:** nella terza sezione è presente un grafico a barre che riporta i valori dei gradi di riscaldamento e di raffreddamento per ogni mese. Questi valori sono relativi ai parametri base che l'utente può selezionare inizialmente. Suddetto grafico fornisce una visione chiara delle fluttuazioni stagionali nei requisiti di riscaldamento e raffreddamento;
4. **Grafico a violino:** nella quarta ed ultima sezione è presente un grafico a violino in riferimento a tre misure meteorologiche: temperatura dell'aria secca (*Dry bulb temperature*), umidità relativa (*Relative humidity*) e velocità del vento (*Wind speed*). Ciascun grafico è suddiviso tra il giorno e la notte, in tal modo consentendo di esplorare le variazioni di queste misure durante diverse fasce orarie.

Nelle figure sottostanti vengono mostrati alcuni screen delle prime due parti di questa pagina nelle tre diverse modalità (singola stazione, clustering e clustering con location):





PROGETTO: Modelli e tecniche per i Big Data

Stazioni selezionate: ONTARIO, GILA BEND, RIVERSIDE, CHINO, LA VERNE, PHOENIX, MESA, GOODYEAR, SCOTTSDALE, ASHEBORO, WINCHESTER, WESTMINISTER, PETERSBURG, ...

Local Value Ranges

SI

Select Weather File

Climate Summary

Temperature and Humidity

Sun Path

Wind

Psychrometric Chart

Natural Ventilation

Outdoor Comfort

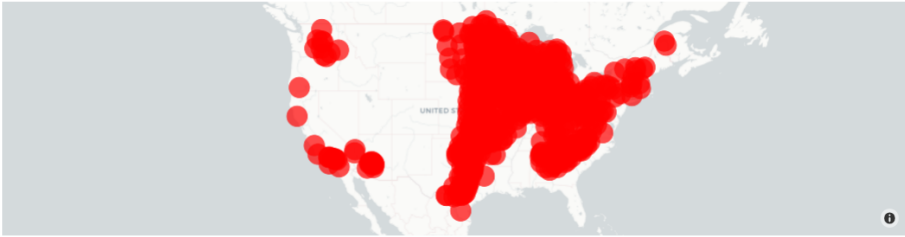
Data Explorer

Location: ONTARIO, GILA BEND, RIVERSIDE, CHINO, LA VERNE, PHOENIX, MESA, GOODYEAR, SCOTTSDALE, ASHEBORO, WINCHESTER, WESTMINISTER, PETERSBURG, ...

Average yearly temperature: 10.4 °C

Hottest yearly temperature (99%): 29.5 °C

Coldest yearly temperature (1%): -9.9 °C



Download ⓘ

Download Clima dataframe

3. Preprocessing

Prima di procedere alla realizzazione dell'intera app di analisi dei dati meteorologici, si è inteso spendere un po' di tempo nel capire la natura dei dati a disposizione ed effettuare processi di data cleaning su quest'ultimi.

Questa fase è stata svolta inizialmente su ambiente Jupyter, in modo da avere a disposizione un notebook che permettesse di scrivere e testare velocemente i passi di pulizia dei dati che si stavano implementando. Di seguito a ciò, avendo ormai consolidato le operazioni di pulizia, si è deciso di implementarle in uno script python, così da rendere il processo più automatizzato e generale possibile.

Dato che tale fase è stata realizzata all'inizio, la pulizia è stata applicata su tutti i dataset a disposizione, sebbene successivamente, nella fase di sviluppo dell'applicazione, si sia deciso di utilizzare solamente il dataset delle misure orarie.

Per tali ragioni, di seguito, verranno descritte solamente le operazioni di pulizia applicate ai due dataset di interesse all'interno dell'applicazione: *Station* (dati caratteristici della singola stazione) e *Hourly* (dati riferiti alle misure orarie delle stazioni nell'anno 2013).

3.1. Station dataset

Le operazioni di pulizia del dataset *Station* sono descritte nelle figure sottostanti, ove viene mostrato il codice del metodo.

```
1 def preprocessing_stations(file):
2     df_stations = pd.read_csv(file, sep="|")
3     columns = ['WBAN', 'Name', 'State', 'Location',
4               'Latitude', 'Longitude', 'StationHeight',
5               'Barometer', 'TimeZone']
6     df_stations = df_stations[columns]
7     # hg to hPa
8     df_stations['Barometer'] = np.where(df_stations['Barometer'].notnull(),
9                                         (df_stations['Barometer'].astype(float) * 33.8639).round(2),
10                                         df_stations['Barometer'].astype(float))
11     # foot to meter
12     df_stations['StationHeight'] = np.where(df_stations['StationHeight'].notnull(),
13                                             (df_stations['StationHeight'].astype(float) * 0.3048).round(2),
14                                             df_stations['StationHeight'].astype(float))
15
16     directory = os.path.dirname(file)
17     nuovo_nome_file = os.path.splitext(os.path.basename(file))[0] + '.M.csv'
18     new_file = os.path.join(directory, nuovo_nome_file)
19     df_stations.to_csv(new_file, index=False)
20
21 # PREPROCESSING STATIONS-----
22 path = "../../../dataset_completo/"
23 folders = ["QCLCD201301", "QCLCD201302", "QCLCD201303", "QCLCD201304",
24           "QCLCD201305", "QCLCD201306", "QCLCD201307", "QCLCD201308",
25           "QCLCD201309", "QCLCD201310", "QCLCD201311", "QCLCD201312"]
26 for folder in folders:
27     folder_name = os.path.join(path, folder)
28     file = os.path.join(folder_name, folder[5:]+".station.txt")
29     preprocessing_stations(file)
30
31 dataframes = [pd.read_csv(os.path.join(os.path.join(path, folder), folder[5:]+".stationM.csv"))
32               for folder in folders]
33
34 #Si prende l'intersezione:
35 result_df = dataframes[0].merge(dataframes[1], how='inner')
36 for df in dataframes[2:]:
37     result_df = result_df.merge(df, how='inner')
38 print(result_df.shape)
39
40 file = "2013stationM.csv"
41 result_df.to_csv(os.path.join(path, file), index=False)
42 # -----
```

La prima operazione effettuata è stata una selezione delle sole colonne di interesse per l'analisi.

Successivamente sono state effettuate operazioni di cambio di unità di misura, in particolare da hg in hPa per l'attributo *Barometer* e da piedi in metri per l'attributo *StationHeight*.

Tali operazioni sono state racchiuse nel metodo *preprocessing_stations(file)*, e sono state richiamate per ogni mese dell'anno 2013. In conclusione, per selezionare le stazioni presenti nei dataset di tutti i mesi dell'anno si è scelto di selezionare l'intersezione delle stazioni nei diversi mesi del 2013, in modo da avere come output il file finale: *2013stationM.csv*, contenente tutte le informazioni sulle stazioni meteorologiche nell'anno 2013.

3.2. Hourly dataset

Le operazioni di pulizia del dataset *Hourly* sono descritte nelle figure sottostanti, in cui viene mostrato il codice del metodo.

```
1 def preprocessing_hourly(file):
2     df_hourly = pd.read_csv(file, sep=";", )
3     columns = ['WBAN', 'Date', 'Time', 'SkyCondition', 'Visibility', 'WeatherType',
4               'DryBulbCelsius', 'WetBulbCelsius', 'DewPointCelsius', 'RelativeHumidity',
5               'WindSpeed', 'WindDirection', 'StationPressure', 'SeaLevelPressure',
6               'HourlyPrecip', 'Altimeter']
7     columns_flag = ['SkyConditionFlag', 'VisibilityFlag', 'WeatherTypeFlag',
8                    'DryBulbCelsiusFlag', 'WetBulbCelsiusFlag', 'DewPointCelsiusFlag',
9                    'RelativeHumidityFlag', 'WindSpeedFlag', 'WindDirectionFlag',
10                   'ValueForWindCharacterFlag', 'StationPressureFlag', 'PressureTendencyFlag',
11                   'PressureChangeFlag', 'SeaLevelPressureFlag', 'RecordTypeFlag',
12                   'HourlyPrecipFlag', 'AltimeterFlag']
13     df_hourly[columns_flag] = df_hourly[columns_flag].replace(['s', 'S'], np.nan)
14
15     df_hourly.dropna(subset=columns_flag, inplace=True)
16     df_hourly = df_hourly[columns]
17
18     df_hourly.replace('M', np.nan, inplace=True)
19     df_hourly.replace(r'^\s*$', np.nan, regex=True, inplace=True)
20     # df_hourly.replace(' ', np.nan, inplace=True)
21     df_hourly.replace('VR ', np.nan, inplace=True)
22     # df_hourly.replace(' ', np.nan, inplace=True)
23     df_hourly.replace(' T', 'T', inplace=True)
24
25     # Celsius in float
26     df_hourly['DryBulbCelsius'] = df_hourly['DryBulbCelsius'].astype(float)
27     df_hourly['WetBulbCelsius'] = df_hourly['WetBulbCelsius'].astype(float)
28     df_hourly['DewPointCelsius'] = df_hourly['DewPointCelsius'].astype(float)
29     df_hourly['RelativeHumidity'] = df_hourly['RelativeHumidity'].astype(float)
30     df_hourly['WindDirection'] = df_hourly['WindDirection'].astype(float)
31     # Pollici to centimetri
32     condition = (df_hourly['HourlyPrecip'].str.strip() != 'T') & (df_hourly['HourlyPrecip'].notnull())
33     df_hourly['HourlyPrecip'] = np.where(condition,
34                                         (pd.to_numeric(df_hourly['HourlyPrecip'], errors='coerce') * 2.54).round(2),
35                                         df_hourly['HourlyPrecip'])
36     # hg to hPa
37     df_hourly['StationPressure'] = np.where(df_hourly['StationPressure'].notnull(),
38                                             (df_hourly['StationPressure'].astype(float) * 33.8639).round(2),
39                                             df_hourly['StationPressure'].astype(float))
40     df_hourly['SeaLevelPressure'] = np.where(df_hourly['SeaLevelPressure'].notnull(),
41                                              (df_hourly['SeaLevelPressure'].astype(float) * 33.8639).round(2),
42                                              df_hourly['SeaLevelPressure'].astype(float))
43     df_hourly['Altimeter'] = np.where(df_hourly['Altimeter'].notnull(),
44                                      (df_hourly['Altimeter'].astype(float) * 33.8639).round(2),
45                                      df_hourly['Altimeter'].astype(float))
```

```

46     # mph to kmph
47     df_hourly['WindSpeed'] = df_hourly['WindSpeed'].apply(convert_mph_to_kmph)
48     df_hourly['Visibility'] = df_hourly['Visibility'].apply(convert_visibility_to_km)
49     # int to date
50     df_hourly['Date'] = df_hourly['Date'].apply(convert_integer_to_date)
51     # int to time
52     df_hourly['Time'] = df_hourly['Time'].apply(convert_time_int)
53     df_hourly['Visibility'] = df_hourly['Visibility'].astype(float)
54
55     df_hourly = aggrega_minuti(df_hourly)
56
57     directory = os.path.dirname(file)
58     nuovo_nome_file = os.path.splitext(os.path.basename(file))[0] + '.M.csv'
59     new_file = os.path.join(directory, nuovo_nome_file)
60     df_hourly.to_csv(new_file, index=False)

1  # PREPROCESSING HOURLY-----
2  # path = "../../../dataset_completo/"
3  path = "../../../dataset_completo/"
4  folders = ["QCLCD201301", "QCLCD201302", "QCLCD201303", "QCLCD201304",
5            "QCLCD201305", "QCLCD201306", "QCLCD201307", "QCLCD201308",
6            "QCLCD201309", "QCLCD201310", "QCLCD201311", "QCLCD201312"]
7  for folder in tqdm(folders, desc="Processing files", unit="folder"):
8      folder_name = os.path.join(path, folder)
9      file = os.path.join(folder_name, folder[5:]+".hourly.txt")
10     preprocessing_hourly(file)
11  # -----

```

Il processo di pulizia del dataset *Hourly* comprende, dapprima, una rimozione di tutte le misure con flag attivo, ovvero le misure dichiarate errate, seguite da una selezione delle sole colonne di interesse per l'analisi. Infine, come nel caso precedente, è presente il cambio delle unità di misura.

4. Spark jobs

Il cuore del progetto è proprio l'utilizzo di Spark per l'analisi dei dati meteorologici. Come già anticipato, sono stati creati dei job spark solamente in due parti critiche dell'applicazione, ovvero quelle parti in cui l'elaborazione era particolarmente complessa, vista la grande quantità di dati da elaborare. Queste due parti sono state: il caricamento del dataset in memoria delle misure della stazione o stazioni selezionate e l'algoritmo Kmeans per il calcolo dei cluster di stazioni.

4.1. Caricamento dataset job

Non appena l'utente seleziona la stazione o il cluster di stazioni da ispezionare, viene lanciato lo Spark job che carica l'intero dataset delle misure orarie dell'anno riferite alla stazione o stazioni selezionate.

In particolare, lo spark job è stato implementato all'interno del metodo `create_df_hourly(stations)`. Come si può notare, anche nel codice sottostante il primo passo eseguito è quello di ottenere lo Spark Context, in modo da poter effettuare con esso tutte le possibili transformations e actions sul dataset.

Il passaggio successivo è stato quello di caricare il dataset a disposizione in un dataframe Spark. Questo è stato possibile grazie al metodo `load_dataset(spark, location=False)`.

I passi che esegue questo metodo sono:

1. Creazione di un array di dataframe Spark, uno per ciascun dataset orario pulito nella fase di preprocessing, per un totale di 12 dataframe (uno per ciascun mese);
2. Creazione di `merged_df` mediante l'unione dei 12 dataframe Spark attraverso la funzione di trasformazione `union`;
3. Se il parametro `location` è impostato su `False`, significa che non si desiderano informazioni sulla posizione geografica della stazione o delle stazioni. In tal caso, è possibile restituire direttamente `merged_df`. Altrimenti, se `location` è `True`, è necessario aggiungere le informazioni geospaziali delle stazioni a `merged_df` mediante una semplice operazione di join tra `merged_df` e il dataframe Spark delle stazioni meteorologiche.

```
1 def load_dataset(spark, location=False):
2     # Carica i dataset mensili
3     path = "../../dataset_completo/"
4     folders = ["QCLCD201301", "QCLCD201302", "QCLCD201303", "QCLCD201304",
5               "QCLCD201305", "QCLCD201306", "QCLCD201307", "QCLCD201308",
6               "QCLCD201309", "QCLCD201310", "QCLCD201311", "QCLCD201312"]
7
8     # Carica ciascun dataset in un DataFrame separato
9     dataframes = [spark.read.csv(os.path.join(path, folder, folder[5:] + "hourlyM.csv"), header=True, inferSchema=True)
10                  for folder in folders]
11
12     # Unisci i DataFrame in uno
13     merged_df = dataframes[0]
14     for df in dataframes[1:]:
15         merged_df = merged_df.union(df)
16
17     if location:
18         df_stations = spark.read.csv("../assets/data/2013stationM.csv", header=True, inferSchema=True)
19         df_stations = df_stations.withColumnRenamed('WBAN', 'wban')
20         df_stations = df_stations.select("wban", "Latitude", "Longitude", "StationHeight")
21         merged_df = merged_df.join(df_stations, "wban", "inner")
22
23     return merged_df
```

Dopo il caricamento del dataframe Spark con il metodo `load_dataset`, è possibile procedere alla selezione delle sole tuple relative alla stazione o alle stazioni selezionate. Il parametro del metodo `create_df_hourly(stations)` è una lista di identificatori WBAN delle stazioni. In particolare, se la modalità di clustering è disattivata, `stations` è una lista con un solo elemento, rappresentante la singola stazione da esaminare. Se, invece, la modalità di clustering è attiva, `stations` sarà una lista contenente tutti gli identificatori WBAN delle stazioni appartenenti al cluster selezionato. In entrambi i casi, il metodo rimane invariato, e i passaggi successivi dopo la lettura del dataset includono:

1. Selezione delle sole righe del dataframe `merged_df`, che contengono uno degli identificatori WBAN presenti nella lista `stations`;
2. Creazione di nuove colonne relative al tempo, tra cui anno, mese e giorno;
3. Raggruppamento per ore. In questo modo, se `stations` dovesse contenere una singola stazione, nulla cambierebbe, poiché i dati sono già aggregati per ora. Al contrario, se ci fossero misurazioni diverse per diverse stazioni in `stations`, sarebbe necessario affrontare il problema aggregando i dati per ora, utilizzando una media;
4. Ordinamento per anno, mese, giorno e ora. Questo passaggio è cruciale soprattutto per la visualizzazione dei grafici nelle successive schede dell'applicazione, che rappresentano le misurazioni in funzione del tempo;
5. Infine, viene eseguita l'azione `toPandas` per trasferire il dataframe Spark alla macchina locale e convertirlo in un dataframe Pandas. Questa scelta è motivata dal fatto che, a questo punto, il dataframe è stato filtrato e contiene al massimo un numero di righe pari al numero di ore in un anno. Inoltre, questa trasformazione permette di sfruttare le analisi presenti nelle schede dell'applicazione per una consultazione in diretta, senza tempi di attesa e caricamento.

```

1  def create_df_hourly(stations):
2      spark = init_spark()
3      merged_df = load_dataset(spark)
4      merged_df = merged_df.filter(merged_df["WBAN"].isin(stations))
5
6      # Estrai anno, mese, giorno e ora dalle colonne Date e Time
7      merged_df = merged_df.withColumn("Datetime", to_date("Date", "dd/MM/yyyy"))
8      merged_df = merged_df.withColumn("year", year("Datetime"))
9      merged_df = merged_df.withColumn("month", month("Datetime"))
10     merged_df = merged_df.withColumn("day", dayofmonth("Datetime"))
11
12     features = ["Altimeter", "DewPointCelsius", "DryBulbCelsius", "RelativeHumidity",
13                "SeaLevelPressure", "StationPressure", "Visibility", "WetBulbCelsius",
14                "WindDirection", "WindSpeed"]
15     agg_exprs = [avg(col(colonna)).alias(colonna) for colonna in features]
16     result = merged_df.groupBy("Datetime", "Date", "year", "month", "day", "Time").agg(*agg_exprs)
17     df_ordinato = result.orderBy(col("year"), col("month"), col("day"), col("Time"))
18     df_pandas = df_ordinato.toPandas()
19
20     spark.stop()
21     return df_pandas

```

4.2. Clustering job

L'altro job Spark implementato riguarda l'esecuzione dell'algoritmo di clustering Kmeans. Il metodo che gestisce questo job è `clustering(k, location=True)`, il quale accetta due parametri: il numero di

cluster *k* e *location*, se si desidera considerare anche le posizioni geografiche delle stazioni durante il calcolo dei cluster.

L'esecuzione di questo job ha inizio quando l'utente attiva il pulsante *Clustering* nella scheda *Select Stations*. In quel momento, il metodo *clustering* inizia a eseguire le seguenti operazioni:

1. Acquisizione dello Spark Context per eseguire le trasformazioni e azioni sul dataframe Spark;
2. Caricamento del dataframe Spark utilizzando il metodo descritto in precedenza *load_dataset*;
3. Creazione di una lista chiamata *features*, che contiene i nomi di tutte le colonne del dataframe utilizzate per il calcolo dei cluster. In particolare, gli attributi selezionati includono: *DewPointCelsius*, *DryBulbCelsius*, *RelativeHumidity*, *Visibility*, *WetBulbCelsius*, *WindDirection* e *WindSpeed*. Quindi, ogni cluster conterrà stazioni simili in termini di temperatura, umidità, visibilità e condizioni del vento;
4. Aggiunta di attributi relativi alla posizione geografica nella lista *features* se il parametro *location* è impostato a *True*. Questi attributi aggiuntivi includono: *Latitude*, *Longitude* e *StationHeight*. In tal caso le stazioni di un cluster condivideranno anche la posizione geografica, consentendo, così, di visualizzare le stazioni dello stesso cluster sulla mappa, evidenziando la loro vicinanza geografica e non solo climatica;
5. Rimozione di tutte le righe che presentano almeno un valore nullo in una delle colonne presenti nella lista *features*;
6. Aggregazione delle righe nel dataframe in base all'identificativo WBAN della stazione. Questo è necessario perché, per il calcolo dei cluster, deve essere presente una sola riga per stazione nel dataframe. Le misure in *features* vengono aggregate con la loro media;
7. Creazione di un *VectorAssembler* per convertire le colonne di *features* in un array di colonne. Questa operazione è spesso richiesta prima di eseguire un algoritmo di machine learning in Spark;
8. Esecuzione dell'algoritmo Kmeans, grazie l'ausilio della libreria MLlib di Spark;
9. Creazione di un dataframe che include, oltre alle colonne precedenti, anche la colonna *prediction*, che rappresenta l'ID del cluster a cui appartiene ogni riga (cioè, ogni stazione);
10. Conversione del dataframe Spark in un dataframe Pandas, permettendo così di utilizzarlo per la visualizzazione dei punti dei cluster sulla mappa nella scheda *Select Stations*.

```
1 def clustering(k, location=True):
2     spark = init_spark()
3     merged_df = load_dataset(spark, location)
4     features = ["DewPointCelsius", "DryBulbCelsius", "RelativeHumidity",
5               "Visibility", "WetBulbCelsius", "WindDirection", "WindSpeed"]
6     if location:
7         features += ["Latitude", "Longitude", "StationHeight"]
8     df_senza_null = merged_df.na.drop(subset=features)
9     agg_exprs = [avg(col(colonna)).alias(colonna) for colonna in features]
10    result = df_senza_null.groupBy("wban").agg(*agg_exprs)
11
12    assembler = VectorAssembler(inputCols=features, outputCol="features")
13    df = assembler.transform(result)
14
15    kmeans = KMeans().setK(k)
16    model = kmeans.fit(df)
17
18    clustered_data = model.transform(df)
19    clustered_data_pandas = clustered_data.toPandas()
20    clustered_data_pandas.rename(columns={'wban': 'WBAN'}, inplace=True)
21
22    spark.stop()
23    return clustered_data_pandas
```


5. Conclusioni e sviluppi futuri

Mentre il progetto ha raggiunto i suoi obiettivi principali, ci sono molte opportunità per sviluppi futuri che potrebbero migliorare ulteriormente l'analisi dei dati meteorologici:

- **Implementazione di Algoritmi Avanzati:** esplorare e implementare algoritmi di apprendimento automatico più complessi potrebbe consentire, infatti, una segmentazione delle stazioni meteorologiche ancora più accurata. Ad esempio, l'uso di algoritmi di clustering gerarchici o basati sulla densità potrebbero portare risultati più dettagliati.
- **Integrazione con Dati Esterni:** considerare l'opportunità di integrare dati esterni, come dati demografici o geografici, potrebbe fornire una prospettiva più completa sulle condizioni climatiche e le loro implicazioni su diverse comunità;
- **Analisi geografiche avanzate:** non limitarsi ad analizzare le singole stazioni meteorologiche o il cluster di stazioni ma compiere ulteriori analisi, avendo la possibilità di selezionare un insieme di stazioni meteorologiche a piacere oppure interessando un'intera area geografica, come ad esempio una regione;
- **Approfondimenti Climatici:** estendere l'analisi per identificare tendenze climatiche a lungo termine potrebbe essere di grande interesse per scopi di previsione e mitigazione dei rischi legati al clima.

In conclusione, il progetto ha gettato le basi per un'analisi avanzata delle condizioni meteorologiche. Gli sviluppi futuri possono contribuire a migliorare ulteriormente la precisione e l'applicabilità dei risultati, offrendo opportunità significative per l'ottimizzazione delle decisioni e delle risorse legate al clima.