UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

# Relazione Progetto Big Data

## Analisi di dati Meteorologici con Apache Storm

Docenti

Trunfio Paolo,

Marozzo Fabrizio

Studenti

Carreri Domenico, 235184,

Cavallo Giuseppe, 235328,

Mandarino Francesco, 235402

Anno Accademico 2022/2023

# Indice

# 1 Introduzione

Il seguente progetto vuole fornire uno strumento di **analisi di dati meteorologici** rilevati sul territorio degli Stati Uniti nell'anno **2013**, offrendo anche la possibilità di visualizzare i risultati ottenuti attraverso un'**interfaccia grafica interattiva**. Per la realizzazione del progetto sono state impiegate le seguenti **tecnologie**:

- **APACHE STORM:** sistema di calcolo distribuito in tempo reale per l'elaborazione di grandi volumi di dati ad alta velocità.
- **DASH:** un framework Python open source, usato per creare applicazioni web di analisi dati.
- **PLOTLY:** una libreria di grafici dichiarativa di alto livello. Viene usata per importare e analizzare dati, creando grafici interattivi browser-based.

Lo sviluppo del progetto si è costituito di varie fasi collegate tra loro, a partire dallo studio del Dataset fornito dai docenti, a cui è seguita la scelta di quali dati prioritizzare e quali invece scartare. Si è stabilito in seguito quali analisi realizzare e si è scelto come strutturare la **Topologia Storm**. La realizzazione di quest'ultima ha dovuto tenere conto di come rappresentare i risultati intermedi da fornire a **Dash**, il framework utilizzato per la visualizzazione delle analisi. La creazione dell'applicazione web ha quindi a sua volta creato l'esigenza di rivedere quanto fatto in precedenza per adattarlo alle esigenze del Frontend.

Trattandosi di un progetto a cui hanno lavorato tre persone, è stato inoltre aggiunto lo studio e la descrizione di un algoritmo di regressione.

Ciascuno degli aspetti coinvolti nella realizzazione del progetto di cui si è appena accennato verrà ora descritto più in dettaglio, al fine di offrire una visione più completa e dettagliata dell'elaborato.

# 2 Analisi e filtraggio del Dataset

Il dataset fornito riguarda l'analisi meteorologica degli Stati Uniti d'America relativi all'anno 2013. In particolare, tra i vari file messi a disposizione ci si è concentrati sulla elaborazione dei dati giornalieri e dei dati mensili per ogni mese.

Data l'enorme dimensione del dataset, inoltre, è stata fatta una pulizia dei dati considerando solo quelli relativi alle temperature e quelli relativi alle piogge. L'analisi è stata effettuata sulle cinquanta capitali degli stati americani, a tal proposito è stata ritenuta necessaria la creazione di un nuovo file usato in input al cui interno sono state mappate le correlazioni tra uno stato e la rispettiva capitale riportando anche per ogni capitale i valori di latitudine e longitudine, utili durante la rappresentazione grafica nel frontend. In ogni file del dataset i parametri relativi alle città vengono rappresentati in forma numerica esprimendo quello che è il codice WBAN di una città, per cui è stato utile per la creazione del file trovare la corrispondenza WBAN-Capitale nel file allstation. La creazione di questo file ha portato a nuove idee di analisi che racchiudono lo studio delle temperature e delle piogge nelle città prese in considerazione.

L'operazione fondamentale del filtraggio dei dati è stata ideata con l'obiettivo di alleggerire il carico di lavoro dei **Bolt** utilizzati andando a mettere dei controlli all'interno delle classi **Spout**, in particolare nel metodo **nextTuple()** descritto dalla figura seguente:

```java
@Override
public void nextTuple() {
    if (!isCompleted) {
        for (int i = 0; i < Utilities.MESI.length; i++) {
            String path = Utilities.get_dataSet_path(new DecimalFormat( pattern: "00").format( number: i + 1), file_name);
            try (CSVReader leggi_file = new CSVReader(new FileReader(path))) {
                List<String[]> contenuto = leggi_file.readAll();
                for (String[] riga : contenuto)
                    if (Utilities.capitali.containsKey(riga[0]))
                        spoutOutputCollector.emit(new Values(riga[0], riga[1].substring( beginIndex: 4), riga[3]));
            } catch (Exception e) {
                throw new RuntimeException(e);
            }//catch
            isCompleted = true;
        }//for
    } else this.close();
}//nextTuple
```

Come prima operazione è stato letto il file delle capitali andando a salvare la coppia WBAN-<NOME_CITTÀ> in una mappa inserita nella Classe di Utilità Utilities. Tale mappa, come si può vedere dalla figura è servita per effettuare un controllo inviando al Bolt solo i dati relativi alle capitali, in particolare, i dati emessi sono stati filtrati in base al tipo di analisi:

- **Spout Piogge:** dallo Spout delle piogge sono stati filtrati ed emessi i seguenti valori: il wban della città, la coppia mese-giorno, il valore di pioggia in quel giorno del mese;
- **Spout Temperature:** dallo Spout delle temperature sono stati filtrati ed emessi i seguenti valori: il wban della città, il mese, il valore di Temperatura Massima per ogni giorno del mese, il valore di Temperatura Minima per ogni giorno del Mese, il valore di Temperatura Media per ogni giorno del mese.

Lato Spout non si vanno a pulire i dati emessi da eventuali valori mancanti. Questa cosa viene fatta però nei vari Bolt. Nello specifico il Bolt delle **piogge** vede arrivarsi come valore di pioggia tre tipologie di dati:

1. **Dato mancante:** che nel dataset è espresso da un valore vuoto o dalla lettera M. Nel Bolt il dato mancante viene posto ad un valore pari a 0 e non viene considerato nell'analisi;

2. **Valore di T:** indica che c'è stata una pioggia di lieve entità il cui valore può essere approssimato a 0.005mm;
3. **Valore numerico:** valore di temperatura corretto e che viene mantenuto così come arriva.

La figura seguente mostra come viene effettuata la pulizia appena descritta sulle tuple ricevute:

```
double qta_pioggia;
if (pioggia.contains("T"))
    qta_pioggia = 0.005;
else if (pioggia.contains("."))
    qta_pioggia = Double.parseDouble(pioggia);
else qta_pioggia = 0.0;
```

I valori di pioggia che arrivano inoltre, sono espressi tramite l'unità di misura del "centesimo di pollice" ma all'utilizzo i dati vengono convertiti in mm (millimetri) di pioggia come mostrato nel frammento di codice seguente nella variabile pioggia_totale:

```
for (String chiave : Utilities.capitali.keySet()) {
    double pioggia_totale = mappa.get(String.format("%02d_%s", i + 1, chiave)) * 25.4;
    Utilities.aggiornaMappa(mese_pioggia, Utilities.MESI[i], pioggia_totale);
    media_pioggia_citta_mese.put(chiave, pioggia_totale / giorni_mese);
}//for
```

Per quanto riguarda le temperature invece per le tuple relative ai valori di temperatura massima, temperatura media e temperatura minima possono arrivare due tipologie di valori:

1. **Valore M:** indica un valore di temperatura mancante. Tale valore viene ignorato nell'analisi;

2. **Valore numerico:** esprime il valore corretto di temperatura. Tale valore arriva dal dataset con l'unità di misure del "grado Fahrenheit" e vengono convertiti nell'unità di misura del "grado Celsius".

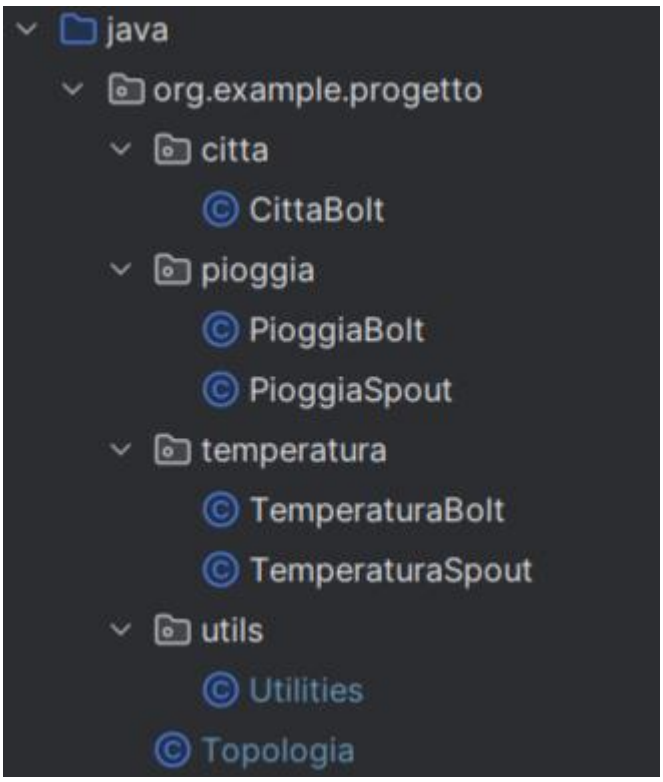
Il metodo successivo mostra come vengono pulite le tuple sulla temperatura:

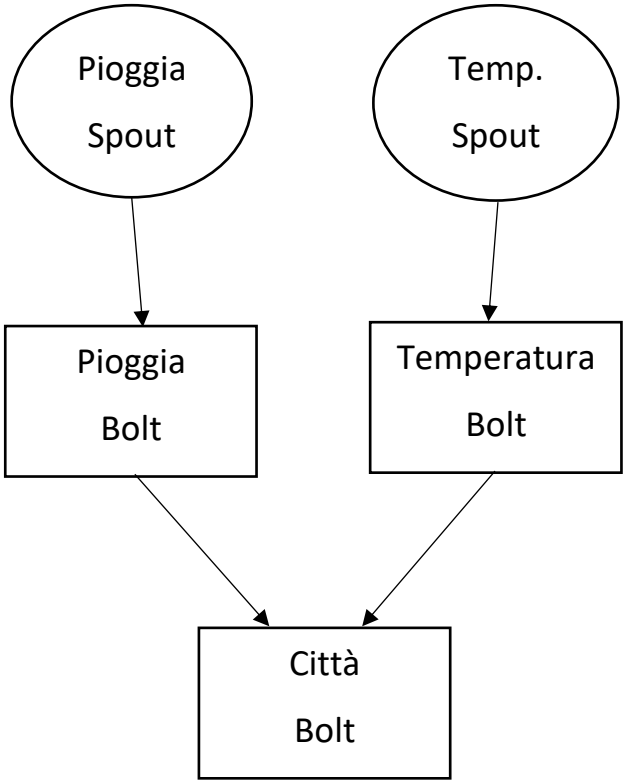
```
public static void aggiornaMappa(Map<String, List<Double>> mappa, String chiave, String temperatura) {
    if (!temperatura.contains("M"))
        if (mappa.containsKey(chiave))
            mappa.get(chiave).add((Double.parseDouble(temperatura) - 32) / 1.8);
        else {
            List<Double> valori_temperatura = new LinkedList<>();
            valori_temperatura.add((Double.parseDouble(temperatura) - 32) / 1.8);
            mappa.put(chiave, valori_temperatura);
        }//else
}//aggiornaMappa
```

# 3 Backend: Storm

Come accennato nella sezione precedente, si è reso necessario effettuare delle operazioni di pulizia sui dati prima di poterci lavorare. Dopo aver quindi filtrato le informazioni di interesse, il passo successivo è stato organizzare e implementare la logica dell'applicazione, che è stata strutturata seguendo un pattern lineare per facilitarne lo sviluppo.

Si è deciso dunque di creare tre **package**: **Pioggia, Temperatura e Città**, ciascuno di essi pone il focus sulla tematica suggerita dal nome stesso. All'interno dei primi due sono presenti le relative classi **Spout** e **Bolt**, mentre nel terzo solamente il Bolt. Questo avviene poiché il **CittaBolt** lavora utilizzando i dati emessi dalle classi **PioggiaBolt** e **TemperaturaBolt**, ma effettua chiaramente analisi differenti. In figura l'organizzazione dei package e della **Topologia**.



Il package utils contiene la classe **Utilities,** che fornisce metodi di supporto per semplificare operazioni ripetitive.

La classe **Topologia** invece definisce la topologia **Storm**, creando i vari Spout e Bolt che descrivono il comportamento della stessa, impostando il grado di parallelismo e il timeout che fa terminare l'esecuzione nel caso in cui nessuno Spout stia più emettendo tuple. Non avendo a disposizione un cluster su cui eseguirla, la topologia si compone di un solo nodo che è contemporaneamente **Nimbus, Zookeeper** e **Supervisor**.

In merito alle classi **PioggiaBolt, TemperaturaBolt** e **CittaBolt**, esse lavorano sui dati precedentemente filtrati e inseriti all'interno di **HashMap.** Quest'ultime sono organizzate in modo da avere come **chiave** una combinazione tra **wban** e **mese** in cui si è registrato il dato (anche il **giorno** nel caso del PioggiaBolt), e come **valore** una lista di misurazioni, un totale o una media, in base allo scopo della mappa presa in esame.

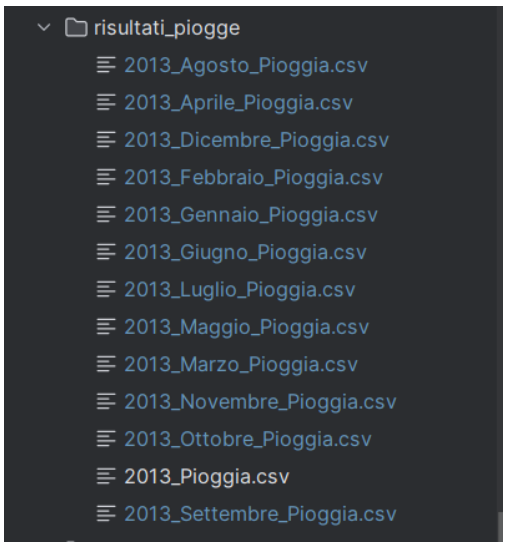
Le mappe in questione sono state utilizzate principalmente per effettuare operazioni di calcolo di **media, minimi e massimi** di valori di **pioggia/temperatura** in intervalli temporali variabili e prestabiliti.

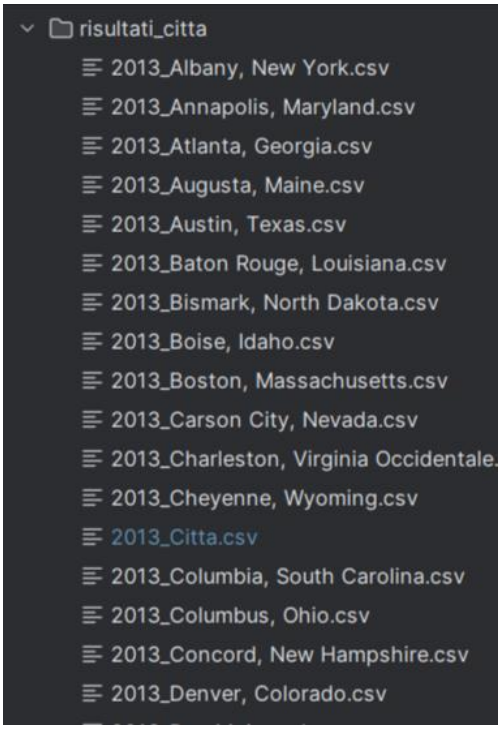
Alcuni esempi delle analisi effettuate sono:

- **Calcolo dei mm di pioggia medi giornalieri nei mesi dell'anno 2013**
- **Calcolo della città mediamente più piovosa e mediamente meno piovosa per ciascun mese**
- **Calcolo dell'elenco ordinato delle città per qtà di pioggia media**
- **Calcolo di medie, massimi e minimi assoluti di temperature medie, massime e minime per ciascun mese e per l'intero anno 2013**
- **Calcolo della città mediamente più calda e mediamente più fredda per ciascun mese**
- **Etc.**

I risultati prodotti dalle analisi dei **Bolt** vengono inizialmente conservati dentro strutture dati di supporto, per essere poi scritti, mediante l'uso di oggetti della classe **FileWriter** all'interno di file **.csv** organizzati sostanzialmente in due categorie, **Annuali** e **Mensili**.

Fa eccezione la classe **CittaBolt**, in quanto genera sia un file dedicato a **ciascuna** delle 49 capitali, che un file denominato "2013_Citta.csv" contenente statistiche che mettono a **confronto tutte le varie città** tra di loro. Nelle figure seguenti è possibile visualizzare l'organizzazione dei risultati prodotti:





Si è usato il formato .csv per rappresentare i risultati delle analisi poiché il Frontend si è deciso di realizzarlo utilizzando **Python**, in particolare la libreria **Pandas,** che è l'ideale per lavorare con i **DataFrames**. Per popolare questi ultimi è stato sufficiente leggere i valori direttamente dai file contenenti i risultati prodotti in precedenza. Ulteriori informazioni in merito saranno fornite nel prossimo capitolo.

# 4 Frontend: Dash e Plotly

L'elaborazione del backend sul dataset fornito in ingresso ha prodotto dei risultati che si è ritenuti opportuni rappresentare in forma tabellare oppure sottoforma di grafici di diverso tipo per cui si è andati alla ricerca di un framework che fosse adatto a questo tipo di rappresentazione. Alla fine, si è scelto l'utilizzo della libreria **Dash** di **Python** molto versatile alla realizzazione delle pagine web che grazie alla cooperazione con la libreria **Plotly** hanno permesso di rappresentare in maniera appropriata i risultati.

Ogni file del frontend per apparire pulito in fase di lettura e chiaro durante lo sviluppo è stato suddiviso in tre sezioni principali:

1. **La sezione delle funzioni:** in questa sezione si trovano le funzioni che descrivono la logica applicativa del file di riferimento;
2. **La sezione delle componenti:** descrive la parte del file in cui sono state dichiarate tutte le componenti web del linguaggio html utili alla visualizzazione e alla generazione di eventi input o output del lavoro svolto dalle funzioni;
3. **La sezione delle variabili:** contiene le variabili di carattere generale usate nel file, in particolare contiene i dataframe che hanno permesso di leggere ed utilizzare in modo molto semplice i risultati prodotti dal backend.

Come detto nel capitolo precedente i risultati sono stati scritti in file di formato csv per questo motivo è stato ritenuto opportuno usare la libreria **Pandas** che offre il metodo read_csv specializzato nella lettura di questo tipo di files. Tale metodo produce in output un dataframe

Il file iniziale che prende il nome di Index.py si occupa di caricare la pagina principale dell'applicazione. Il framework Dash ha al suo interno un server in

grado di ospitare le applicazioni web, per cui per prima cosa va creato e lanciato il server Dash.

```
''' ---------------------------------------------- Inizializzazione app ----------------------------------------------'''

app = dash.Dash(__name__, suppress_callback_exceptions=True,
                meta_tags=[{'name': 'viewport',
                            'content': 'width=device-width, initial-scale=1.0'}],
                external_stylesheets=[dbc.themes.VAPOR]
                )
```

La figura precedente mostra come viene inizializzato il server Dash. Da prestare particolare attenzione al parametro external_stylesheets che permette all'applicazione di impostare un tema principale per la sua visualizzazione. Questa cosa viene resa possibile grazie all'integrazione in Dash della libreria dash-bootstrap-component (oggetto dbc in figura) che introduce l'utilizzo degli stili del noto framework Bootstrap che offre una vasta gamma di componenti e temi personalizzabili che rendono migliore la navigazione nelle pagine web.

```
app.layout = html.Div(children=[
        html.Link(
            rel='stylesheet',
            href='/static/style.css',
        ),
        dcc.Location(id='url', refresh=False),
        navbar,
        html.Div(id='page-content', children = [])
    ])
```

La figura soprastante descrive uno dei concetti chiave di Dash, ovvero il componente **layout** che definisce la struttura e il layout dell'applicazione. Il layout descrive dunque l'oggetto che rappresenta l'aspetto visivo dell'applicazione e definisce come i componenti dovrebbero essere organizzati sulla pagina includendo la collaborazione tra i componenti HTML, gestiti dall'apposito modulo **html**, e i componenti di Dash veri e propri.

La successiva immagine invece descrive come viene avviato un server Dash:

```
if __name__ == '__main__':
    app.run_server()
```

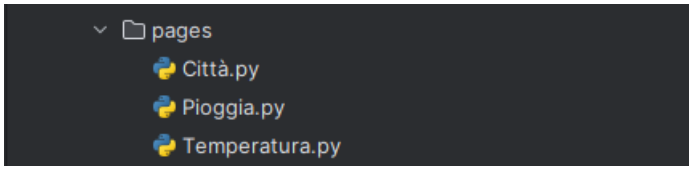
Tale server solitamente viene messo in ascolto all'indirizzo 127.0.0.1 sulla porta 8050 che può essere modificata nel caso fosse occupata.

La pagina principale si presenta semplice e non mostra nulla di particolare se non una navbar con alcuni pulsanti che andranno a costituire le Route dell'applicazione, il titolo che descrive il fine per cui stata creata e infine un footer che mostra i candidati.

```
@app.callback( *_args: Output( component_id: 'page-content', component_property: 'children'),
              [Input( component_id: 'url', component_property: 'pathname')])
def display_page(pathname):
    if pathname == '/pages/pioggia':
        return Pioggia.layout()
    if pathname == '/pages/temperatura':
        return Temperatura.layout()
    if pathname == '/pages/citta':
        return Città.layout()
    else:
        return home
```

La precedente immagine illustra come vengono gestite le Routes, in particolare si utilizza il decoratore di callback che computa in base all'evento generato dal click di uno dei bottoni sulla barra di navigazione quale pagina mostrare.

I layout delle pagine che visualizzano i risultati della nostra sono stati organizzati nelle tre componenti del package **Pages**:




Queste componenti mostrano:

- **pioggia.py:** rappresenta graficamente i dati generali relativi alle piogge nell'anno 2013;
- **temperatura.py:** rappresenta graficamente i dati generali relativi alle temperature nell'anno 2013;

- **città.py:** rappresenta graficamente i dati relativi sia alle piogge che alle temperature sulle specifiche capitali degli stati degli USA.

Il componente pioggia quando viene raggiunto dal metodo **display_page** del componente **index.py** descritto in precedente mostra come pagina principale una schermata che visualizza un grafico a barre che per ogni mese rappresenta la quantità media di pioggia giornaliera espressa in mm durante tutto l'anno in analisi nelle capitali degli Stati Uniti. Al di sotto del grafico troviamo delle checklists ognuna etichettata con il nome di un mese. Inizialmente queste checklists sono tutte abilitate, tuttavia viene permesso all'utente di disabilitare uno o più mesi spuntando le corrispondenti checklists per visualizzare l'andamento su un sottogruppo di mesi (ad esempio lasciare solo attive le checklists di giugno, luglio e agosto per vedere l'andamento medio delle piogge nella stagione estiva…). In alto si trovano due radio buttons. Di default quello spuntato è etichettato con "Resoconto Annuale", se cliccato ha l'obiettivo di mostrare la pagina appena descritta. Il secondo radio button etichettato con "Analisi Mensile" ci porta su un'altra pagina in cui viene mostrata una mappa degli Stati Uniti al cui interno sono state inseriti dei pallini colorati sulle capitali degli USA. Il mese a cui fa riferimento questo grafico viene riportato in un DropDownMenù che di default è posto sul mese di gennaio ma se cliccato apre una lista di DropDownItem in cui si permette all'utente di visualizzare uno specifico mese.

La caratteristica dei pallini rappresentati sulla mappa è che la loro grandezza è direttamente proporzionale alla quantità media di pioggia giornaliera nella capitale rappresentata: più in quella città il valore di pioggia medio giornaliero è alto e più il pallino risulta essere grande. Sotto il grafico, infine, sono riportate due tabelle che mostrano rispettivamente il giorno più piovoso e il giorno meno piovoso in quel mese e la città in cui tali dati sono stati registrati.

Anche la pagina di default del componente temperatura è abilitata da un radio button che, come quello delle piogge, prende l'etichetta di "Resoconto Annuale".

Questa pagina mostra diverse tabelle che descrivono l'andamento della temperatura durante l'interno anno.

La prima tabella mostra per ogni mese dell'anno la media delle temperature massime, la media delle temperature minime e la media delle temperature medie. Ogni cella della tabella è stata colorata con un colore scelto da una tavola di otto colori. Il criterio di scelta del colore da attribuire alla cella è dipendente dal valore di temperatura scritto nella cella: più è alto il valore di temperatura e più la cella assume un colore caldo (tendente verso il rosso), più il valore di temperatura è basso e più la cella assume un colore di tonalità fredda (tendente verso il blu).

Sotto questa tabella principale sono riportate altre tre tabelle:

Nella prima colonna troviamo due tabelle che mostrano la temperatura massima e la temperatura minima in assoluto registrate nell'anno 2013 riportando il mese e il luogo in cui si sono registrate;

Nella seconda colonna si può visualizzare un'altra tabella che descrive l'andamento dell'escursione termica esaminata per ogni categoria di temperatura come la differenza tra la temperatura massima e la temperatura minima e il mese a cui tale valore fa riferimento:

Ad esempio, per la categoria di temperatura massima l'escursione termica è stata calcolata dalla seguente formula:

*escursione_massima = massima_temperatura_massima – minima_temperatura_massima.*

Se da questa finestra viene cliccato il radio button etichettato da "Analisi Mensile" si viene reindirizzati su una pagina che mostra le analisi di temperatura per ogni mese dell'anno. In particolare, la struttura della pagina è la seguente:

In alto a destra si ha un DropDownMenù che, come per le piogge, riporta il nome di un mese. Come valore di default è impostato il mese di gennaio ma viene permesso all'utente di cambiarlo spostando la visualizzazione sui dati del mese

scelto. La parte principale della pagina è costituita da un istogramma dove le ascisse descrivono ogni giorno del mese mentre le ordinate descrivono, per ogni giorno, la media di temperatura massima espressa da una barra rossa, la media di temperatura minima espressa in azzurro e la temperatura media espressa da una barra arancione.

Subito sotto al grafico vengono mostrate quattro tabelle tre di queste descrivono rispettivamente il valore massimo e il valore minimo di temperatura media, di temperatura minima e di temperatura massima registrati nel mese di riferimento riportando anche le città in cui tali valori sono stati registrati. Nella quarta tabella vengono riportati i dati relativi all'escursione termica mensile.

Se sulla barra di navigazione viene cliccata la voce "Capitali" viene mostrata la pagina di default del componete delle capitali che in base al valore di una checklist posizionata in alto a destra mostra i valori di temperatura o di pioggia di tutte le città capitali americane.

Nello specifico viene mostrato un grafico che esprime la mappa degli Stati Uniti dove ogni stato viene colorato in base al valore di temperatura (o di pioggia rispettivamente, a seconda del grafico che si sta osservando) della sua capitale. In particolare, per il grafico delle temperature è stata scelta una scala di colori che spazia tra colori più freddi per i valori di temperatura bassi fino a colori più caldi per temperature più alte. Per le piogge invece si è presa una tavola di colori basata su una scala di azzurro e si assume un colore più chiaro per un valore di pioggia basso, un colore più scuro invece viene usato per valori di pioggia più intensi.

Di fianco alla mappa degli USA si posizionano due tabelle che mostrano, sulla base di un'analisi media dei valori durante l'intero anno, le dieci città più calde (o più piovose rispettivamente) e le dieci città più fredde (o meno piovose rispettivamente).

In alto come al solito si hanno due radio buttons: "Tutte le Città" mostra la pagina appena descritta mentre "Singola Città" porta su un'altra pagina in cui vengono visualizzati i dati relativi alle singole capitali.

Di default in questa pagina vengono mostrati i dati della città di *Albany* capitale dello stato di *New York*, tuttavia, l'utente può cambiare la città di riferimento tramite una Select posizionata in alto a destra. Il contenuto di questa pagina sono due grafici, il primo mostra la mappa degli USA evidenziando lo stato della capitale che si sta visualizzando, il secondo grafico riporta l'andamento delle temperature, espresse da una retta spezzata rossa e quello delle piogge, espresso da una retta spezzata blu.

In basso abbiamo quattro tabelle:

Le prime due mostrano il giorno più piovoso e il giorno meno piovoso durante l'anno, le successive due mostrano la giornata più calda e quella più fredda.

# 5 Algoritmo di Regressione

La regressione lineare è una tecnica statistica che si utilizza per studiare la relazione tra due o più variabili.Da un punto di vista delle formule, la regressione lineare è una funzione matematica basata dall'equazione della retta.Nello specifico, un modello di regressione lineare è composto da:

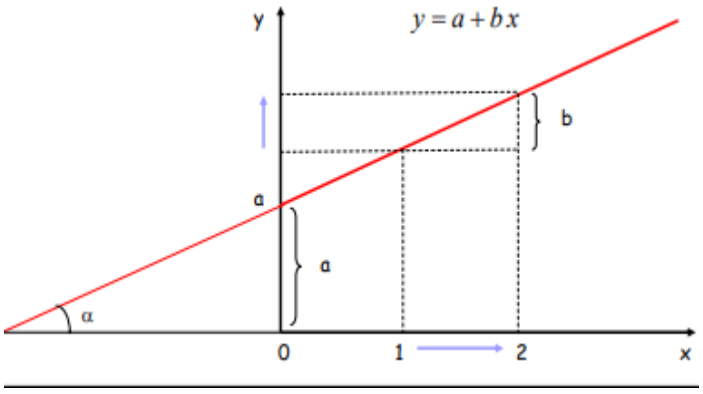
- Una sola variabile risposta quantitativa (detta anche dipendente o Y)
- Una o più variabili esplicative (dette anche X o regressori)
- Un coefficiente di regressione per ogni variabile esplicativa più un coefficiente per l'intercetta (β)
- Un termine di errore (ε). Questo perché la relazione tra due variabili non è quasi mai perfettamente riassumibile tramite un'equazione matematica. Questo accadrebbe solo nel caso in cui tutte le unità statistiche si comportassero esattamente nello stesso modo (un po' come degli scalatori in cordata).

Nello specifico, la variabile risposta (la Y) è determinata dai valori dell'intercetta (β0) a cui vengono sommati i valori delle variabili esplicative (le X) moltiplicate per i loro coefficienti (β), più un termine d'errore (ε). L'equazione quindi è:

- Se c'è un solo regressore:$Y=\beta0+\beta1*X1+\varepsilon$
- Con due regressori:$Y=\beta0+\beta1*X1+\beta2*X2+\varepsilon$
- Se ci sono n regressori:$Y=\beta0+\beta1*X1+\beta2*X2+\ldots.+\beta n*Xn+\varepsilon$.

Questa equazione sarà tanto più lunga quanti più sono i regressori del modello. Esistono vari tipi di regressione lineare,il modello più semplice di relazione tra due variabili è quello lineare di primo grado, rappresentato da una retta, la cui equazione è : **y=a+bx.**Una volta determinata la retta, il modello permetterà di stimare il valore della variabile Y sulla base del valore assunto dalla X.Per ottenere un buon modello, e quindi delle buone previsioni, occorre determinare la retta che meglio descrive i punti osservati. Concentriamoci sul significato del termine a e del termine b che sono presenti nella formula precedente.

- il parametro a è l'intercetta della retta con l'asse delle ordinate
- il parametro b è il coefficiente angolare: misura l'inclinazione della retta (è la tangente dell'angolo α formato dalla retta con l'asse delle ascisse)




Per determinare la retta di regressione che meglio descrive i dati osservati è allora necessario stabilire un criterio statistico di valutazione della bontà del modello:

.

**A cosa serve il modello di regressione lineare?**

Il modello di regressione si utilizza per stimare con la maggior precisione possibile il valore delle variabile risposta, partendo dai valori delle variabili esplicative.La

regressione lineare è un'estensione dell'analisi della correlazione lineare. Come l'analisi di correlazione, la regressione lineare permette infatti di analizzare la relazione tra variabili. Ci permette infatti di studiarne sia la direzione che la significatività. Inoltre, la regressione ci permette di quantificare di quanto in media aumenterà o diminuirà la y all'aumentare del punteggio di una variabile esplicativa

Per mettere in atto e sfruttare un algoritmo di regressione si è pensato a due tipologie di analisi: previsione dei dati medi mensili e previsione della temperatura media annuale per l'anno 2014. Come tipologia di algoritmo di regressione si è usata una regressione lineare assumendo che tra le temperature e l'anno ci fosse una relazione lineare.

Una volta calcolati i valori sono stati confrontati con i valori reali in modo da poter confrontare la veridicità e la correttezza del risultato calcolato.

Per rendere realizzabile il calcolo delle previsioni mensili sono stati trovati dei dataset relativi agli anni 2010, 2011 e 2012 che rispecchiano la struttura di quello fornito e unendo i valori presi da questi dataset con quelli del 2013 già in nostro possesso è stato applicato l'algoritmo.

Nella realizzazione dell'analisi della temperatura annuale invece è stato utilizzato un documento in cui sono riportati i valori di temperatura media per ogni anno a partire dal 1875.

Per quanto riguarda il backend sono state create due classi **RegressioneSpout** e **RegressioneBolt:** lo spout si occupa di leggere dai vari dataset i valori di interesse e come operazione di filtraggio va ad emettere al bolt solo i valori inerenti alle capitali passando come parametri l'anno, il mese e il rispettivo valore di temperatura. Il bolt invece quando riceve i dati va a pulire i valori di temperatura convertendoli da gradi Fahrenheit in gradi Celsius per poi usarli nell'algoritmo di regressione. Il bolt inoltre non riceve i valori annuali dallo spout ma implementa lui stesso la funzione di lettura dal rispettivo file.

Per implementare l'algoritmo di regressione è stata usata la classe **Simple Regression** della libreria **Apache Commons Math** di Java.