Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**

**Corso di Laurea in Informatica per il Management**

# REALIZZAZIONE E VALIDAZIONE SPERIMENTALE DI UN DATASET OPEN PER L'INTERNET OF THINGS

**Relatore:**
**Chiar.mo Prof.**
**MARCO DI FELICE**

**Presentata da:**
**MATTIA MANIEZZO**

**Correlatore:**
**Dott. FEDERICO MONTORI**

**Sessione I**
**Anno Accademico 2018-2019**

*If you can meet with Triumph and Disaster*
*And treat those two impostors just the same*

---

*Rudyard Kipling, If*

*Watch, learn and don't eat my cookie*

---

*Phoebe Buffay, Friends*

# Abstract

L'incremento e il continuo sviluppo dei dispositivi inerenti all'Internet of Things (IoT) ha causato un aumento esponenziale dei dati prodotti da ognuno di noi. Oltre all'incremento dei dati generati, è stata resa più semplice la condivisione libera di essi attraverso la rete internet. Una parte dei dati IoT generati dai dispositivi eterogenei possono essere liberamente accessibili e vengono chiamati Open Data, i quali possono essere reperibili in repository pubbliche fornite da enti, organizzazioni e studi/esperimenti, o realizzate da utenti grazie al crowdsourcing. Gli Open Data sono in costante aumento, ma presentano dei problemi come la scarsità di informazioni fornite, come metadati assenti o incompleti, che, a volte, li rendono poco comprensibili. Questo comporta un problema dal punto di vista dell'utilizzo dei dati vista la possibile scarsa riconoscibilità e comprensibilità. Per risolvere questo problema bisogna effettuare operazioni di annotazione e classificazione automatica. Le operazioni necessitano di essere automatiche visto che gli Open Data sono caratterizzati da grandi quantità di dati, quindi farle manualmente è impossibile.

Nello studio sperimentale si è realizzato un dataset contente dati provenienti dalla piattaforma online Thingspeak, che è un repository pubblico che sfrutta il crowdsourcing. Il dataset realizzato è sottoposto ad un'attività di validazione e di sperimentazione. La validazione sperimentale serve per verificare se il dataset realizzato in questo studio si comporta in maniera simile a dataset ottenuti da repository pubbliche fornite da enti e studi/esperimenti esterni. La sperimentazione serve a verificare quale algoritmo di classificazione è più efficiente per i dataset realizzato e quelli considerati, e a confermare che la classificazione basata sulla successione dei dati non funzioni per il tipo di dataset considerato.

# Indice

# Elenco delle figure

# Listings

# Elenco delle tabelle

# Introduzione

Nell'ultima decade il numero di dispositivi elettronici ha superato il numero di persone di tutto il mondo. Una grande fetta di questi dispositivi fa riferimento all'Internet of Things (IoT). Questo sviluppo e incremento dei dispositivi IoT ha portato un aumento esponenziale della quantità di dati prodotti da ognuno di noi, facendoci diventare sia consumatori che produttori di dati. Con la nascita di dispositivi sempre più semplici da utilizzare è stata facilitata la generazione di dati IoT e la possibilità di condividerli liberamente attraverso la rete internet.

I dati IoT prodotti da dispositivi eterogenei vengono chiamati Open Data e possono essere accessibili in repository pubbliche create da enti e studi, o tramite utenti con meccanismi di crowdsourcing. Un repository che sfrutta il crowdsourcing è Thingspeak: una piattaforma online che consente agli utenti registrati di caricare liberamente i propri dati IoT, provenienti generalmente da alcuni sensori. Negli ultimi anni sono nati diversi studi nell'ambito degli Open Data visto il loro costante aumento di disponibilità in tutto il mondo e la loro facilità di reperimento. Gli Open Data, però, non sono esenti da problemi nonostante i loro vantaggi. Essi forniscono spesso scarse informazioni, risultando così poco comprensibili. Questo genera problemi come il dover effettuare un'annotazione e una classificazione dei dati.

All'interno di questo elaborato si è provato a realizzare un dataset contenente datastream di dati di tipo ambientale provenienti dalla piattaforma online Thingspeak. Dopo la creazione del dataset appena citato, si è provato a svolgere alcuni esperimenti utilizzando tecniche di machine learning supervisionato su di esso. Gli obiettivi posti al principio di questo elaborato sono: la realizzazione di un dataset che si comporti in maniera simile

ai dataset pubblici forniti da enti o studi; l'individuazione dell'algoritmo di classificazione maggiormente performante sui dataset utilizzati; la verifica che la classificazione basata sulla successione dei dati non funzioni per il tipo di dataset considerati.

Per la realizzazione del dataset sono stati scaricati tutti i dati pubblici disponibili su Thingspeak e, in seguito, è stato applicato un algoritmo di clustering spaziale (machine learning non supervisionato) per trovare i cluster con più datastream in base alle coordinate in cui sono stati prodotti. Dopo aver trovato i cluster, ad essi è stato applicato un algoritmo di clustering temporale, creato ad hoc per questo elaborato, per individuare il giorno in cui c'è il numero maggiore di datastream con almeno un certo numero di misurazioni. In seguito alle due fasi di clustering si è proceduto ad applicare l'interpolazione sui datastream per averli tutti con lo stesso numero di misurazioni. Infine, si è passati all'annotazione di tutti i datastream, decidendo così le classi con cui etichettare i dati.

Alla conclusione della realizzazione del dataset, per effettuare gli esperimenti sono state individuate le feature da utilizzare. Queste sono state poi ordinate dalla più alla meno significativa tramite lo svolgimento di un'analisi della varianza. Gli esperimenti sono proseguiti applicando al dataset diversi algoritmi di classificazione per trovare quello con le prestazioni migliori. Infine, l'algoritmo scelto è stato applicato ai datastream con diversi partizionamenti delle loro misurazioni. Gli esperimenti sono stati svolti sia per il dataset creato con dati da Thingspeak che per due dataset pubblici forniti da studi esterni.

La tesi è strutturata in 7 capitoli.

Il Capitolo 1 (Stato dell'arte) presenta il tema dell'Internet of Things (IoT) e degli Open Data in ambito generale. Vengono descritti i problemi relativi agli Open Data, sui quali si fonda il lavoro svolto in questo studio sperimentale.

Nel Capitolo 2 (Machine Learning) viene presentata un'overview generale sul machine learning e sulle tecniche, supervisionate e non, che hanno interessato gli esperimenti realizzati.

All'interno del Capitolo 3 (Progettazione) vengono descritte in breve le

varie fasi di lavorazione e gli strumenti che hanno interessato la realizzazione dello studio sperimentale.

Nei Capitoli 4 (Realizzazione del dataset) e 5 (Sperimentazione) vengono descritte con maggior dettaglio, quindi anche con frammenti di codice, le scelte implementative effettuate per le fasi di realizzazione del dataset e di sperimentazione su di esso.

Il Capitolo 6 (Risultati) presenta l'analisi dei risultati ottenuti dalla fase di sperimentazione svolta sul dataset realizzato in questo studio.

Nel Capitolo 7 (Conclusioni e possibili sviluppi), infine, vengono tratte le conclusioni finali relative allo studio sperimentale realizzato e vengono indicati alcuni possibili sviluppi futuri.

# Capitolo 1

# Stato dell'arte

L'Intenet of Things (IoT) oggi fa riferimento ad un vasto numero di "cose" che sono connesse ad internet e possono condividere dati con altre cose. Tra queste "cose/things" rientrano le applicazioni IoT, i connected device, le macchine industriali e molto altro. I connected device utilizzano sensori built-in per raccogliere dati e, in certi casi, agire su essi. Il nostro mondo è ormai pieno zeppo di smart device e delle loro connessioni. Tutti i dispositivi e le macchine IoT hanno l'obiettivo di cercare, se possibile, di migliorare il modo in cui viviamo e lavoriamo. Per questo, negli ultimi anni, la richiesta di maggiori smart thing, device ed environment è aumentata esponenzialmente, portando molte aziende ad investire tantissimo denaro in questo business [11].

## 1.1 Internet of Things - IoT

Il termine "Internet of Things" è stato coniato per la prima volta nel 1999 da Kevin Ashton [2], uno dei fondatori dell'Auto-ID Center al MIT. Egli era membro di un team che scoprì come collegare oggetti ad internet tramite un tag RFID (Radio Frequency IDentifier). Ashton è stato il primo ad usare il termine Internt of Things, ma il concetto di connected device, in particolare le connected machine, esisteva da molto più tempo. Basti pensare ai primi telegrafi elettronici del 1800, le trasmissioni radio, le tecnologie wireless (Wi-Fi) e molo altro ancora.

Oggi viviamo in un mondo dove sono presenti più IoT connected device che persone [9]. Questi dispositivi IoT sono sempre in continua evoluzione e la loro quantità è in continuo aumento. Gli IoT connected device sono dispositivi che comunicano attraverso la rete o piattaforme basate sul cloud (Figura 1.1 [1]). Tutto l'IoT oggi ha un forte impatto sul nostro mondo visto che la maggior parte dei settori industriali ne fanno uso [19]. Alcuni esempi di ambiti che utilizzano l'IoT sono il settore delle vendite (retail), quello manifatturiero [3], quello logistico e dei trasporti [24], quello della sicurezza [7], quello sanitario [10] e quello ambientale [3].

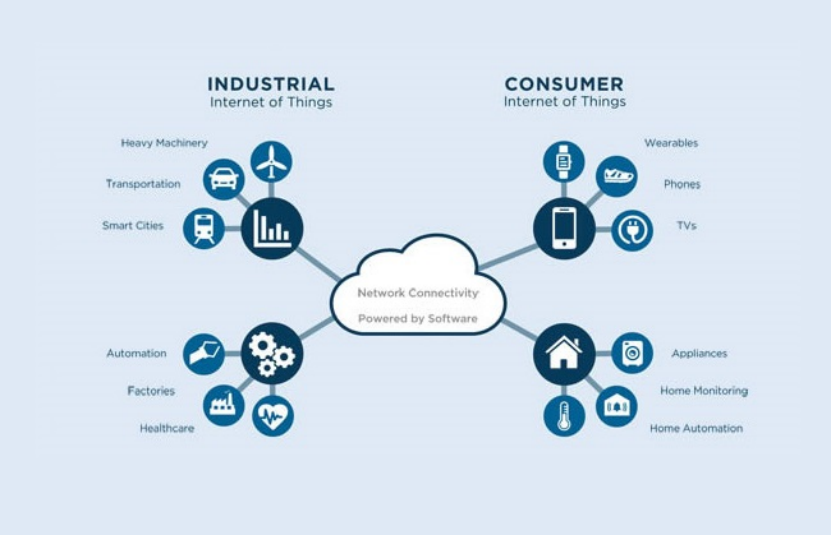



Figura 1.1: Il mondo dell'Internet of Things

Dal punto di vista tecnologico e digitale, l'IoT svolge un ruolo significativo visto la sua costante evoluzione negli anni. Grazie alla sua vastità e al suo continuo sviluppo, l'IoT possiede potenziali ancora inespressi; per questo negli ultimi anni stanno nascendo molti studi a riguardo [8]. Gli ambiti di ricerca più importanti riguardanti l'Internet of Things sono i seguenti:

- Data management: l'IoT permette di ottenere una grande quantità di dati provenienti da sensori e per questo vengono richieste operazioni di

[1] https://www.isipc.it

data management e analisi dei dati importanti. A volte queste operazioni di gestione e analisi vengono richieste in real-time per rendere i dati IoT ottenuti più validi e utili.

- Big Data analytics: l'IoT è un grande contributore al mondo dei big data visto che permette di raccogliere, ogni giorno, una vastissima quantità di dati strutturati e non strutturati. Oggi si vogliono sfruttare i big data in IoT per ottenere valori e informazioni utili, e per fare ciò sono necessarie tecniche di bid data analytics. Esempi di queste tecniche sono: predictive analytics, text mining, cloud computing, data mining, ecc.

- Artificial Intelligence (AI): l'AI può far aumentare il valore dell'IoT utilizzando i dati provenienti dagli smart connected device per promuovere l'apprendimento e l'intelligenza collettiva. Alcune tecniche utilizzate dall'AI per i suoi obiettivi sono: deep learning, natural language processing e, soprattutto, il machine learning [13].

### 1.1.1 Il paradigma IoT

L'IoT, sotto un certo punto di vista, può essere definito come un paradigma in cui ogni oggetto o concetto del mondo reale (le cosiddette "things") può essere mappato attraverso dei dati nel mondo digitale con una corrispondenza 1-a-1 (Figura 1.2) [15]. Questo viene permesso attraverso un "perception layer" formato da sensori, i quali possono raccogliere informazioni dal mondo reale e inviarli ad un hub di raccolta tramite l'uso di tecnologie di comunicazione. I dati IoT sono generalmente non interpretabili da noi esseri umani. Per trasformare questi dati in conoscenza valida e utile, vengono utilizzati diversi algoritmi per il data processing come, ad esempio, metodi statistici, il data mining e il machine learning. Oltre ai sensori nel perception layer si possono aggiungere anche gli "actuator", i quali sono particolari sensori che si preoccupano di agire nel mondo reale sulla base di dati e comandi provenienti dal mondo digitale. Inoltre, si può aggiungere anche il "context" (contesto) che è un insieme di meta-informazioni ottenuto dagli stessi dati dei sensori e spesso viene fornito insieme alle informazioni raccolte dagli stessi.

Infine, esistono meccanismi che, in base ad una certa quantità di conoscenze acquisite, permettono di attivare certi comportamenti.

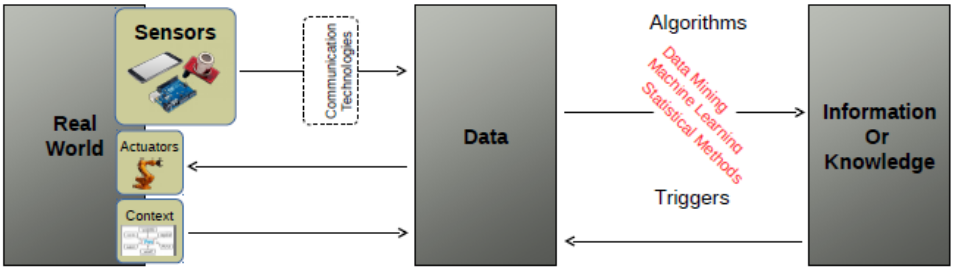



Figura 1.2: Il paradigma IoT [15]

Come descritto in precedenza, l'Internet of Things nell'ultima decade è diventato il focus di molte ricerche scientifiche. L'interesse nei suoi confronti è sempre più ampio e i campi d'applicazione e ricerca sono in aumento, come ad esempio il monitoraggio ambientale, le Smart City, l'home automation e molto altro [21]. Nell'ambito dello studio sperimentale descritto all'interno di questa tesi ci si concentrerà sul ruolo e l'influenza degli Open Data (derivanti dai big data) nelle applicazioni e in ambienti IoT.

## 1.2 Open Data

Al giorno d'oggi l'Internet of Things è in costante crescita e vede la nascita di sempre più dispositivi IoT eterogenei. Questo fenomeno non fa altro che incrementare la generazione di dati IoT da qualsiasi utente intorno al mondo e liberamente accessibili attraverso la rete Internet. I dati IoT prodotti da dispositivi diversi fra loro prendono il nome di "Open Data". Essi, come indicato dal nome, sono dati scritti in un formato comprensibile alle macchine e liberamente accessibili in repository pubbliche [20]. Le Open Data repository vengono suddivise in due tipologie [17]:

- Reliable repositories: sono repository create e mantenute da organizzazioni, governi o studi/esperimenti. Esse sono la tipologia preferibile

tra le due. Questo perché i dati da loro forniti seguono certe regole di annotazione, quindi si sa di che cosa si tratta, vengono aggiornati regolarmente e la qualità è garantita vista che vengono prodotti da enti o studi specialistici.

- Unreliable repositories: sono repository create attraverso il crowdsourcing, grazie al quale degli utenti contribuiscono gratuitamente a caricare i propri datastream. Esse sono caratterizzate dal fornire dati senza alcuna garanzia di veridicità e, spesso, senza indicare nemmeno cosa viene misurato. I dati, infatti, sono spesso scarsamente annotati e senza informazioni.

Un esempio di crowdsourced unreliable repository è Thingspeak: una piattaforma online che consente a qualsiasi utente registrato di poter caricare in essa i propri dati IoT. Questa piattaforma permette agli utenti di caricare dati generati dai loro dispositivi personali (la maggior parte sono riguardanti l'ambiente) in "data channel" e renderli pubblici o meno. Nel proprio channel l'utente può registrare fino ad otto "datastream", i quali sono una serie ordinata temporalmente di data point numerici, ognuno dei quali corrisponde ad una certa misurazione. Oltre ai data point, ogni datastream possiede dei metadati (sempre prodotti dal sensore integrato nel dispositivo IoT), i quali contengono informazioni quali il nome del datastream, il nome del channel, la descrizione, le coordinate geografiche, il numero di misurazioni e altro ancora [14].

Negli ultimi anni l'interesse nei confronti degli Open Data è aumentato. In particolar modo, nello studio sperimentale effettuato in questi tesi, ci si è concentrati sugli Open Data forniti da repository di tipo unreliable. Ciò porta alla domanda: perché questi repository dovrebbero essere considerati e sfruttati? Alcune motivazioni per decidere di prendere in considerazione gli unreliable repository possono essere le seguenti [15]:

1. I reliable repository forniscono dati relativi a zone geografiche ristrette, spesso per singole città. Questo potrebbe portare ad avere delle piccole imprecisioni in alcune tipologie di dati. Gli unreliable repository, es-

sendo crowdsourced permettono di accedere a dati da diversi parti del mondo, permettendo così di coprire più aree geografiche.

2. Le piattaforme online di unreliable repository, come Thingspeak, sono in continuo aumento dal punto di vista di utenti registrati e, quindi, di datasteam caricati. [16] Questo fenomeno può essere motivato dalla riduzione dei costi dei componenti necessari e dalla facilità d'uso di nuovi linguaggi e tool di sviluppo.

### 1.2.1 Problematiche relative all'utilizzo di Open Data

Come definito in precedenza, insiemi di dati eterogenei, come i datastream, ottenuti da fonti Open Data (repository), spesso forniscono scarse informazioni e, di conseguenza, risultano poco comprensibili. Oltre ai dati numerici, spesso anche i metadati dei datastream non forniscono informazioni utili a comprendere cosa viene effettivamente misurato [22]. Questo genera la necessità di effettuare un'annotazione e una classificazione per poter comprendere al meglio che tipologie di dati i datastream stanno fornendo. Questo problema viene generalmente chiamato "Open IoT datastream classification and annotation problem" [18], il quale non è ancora stato approfondito abbastanza da parte di altri studi. In letteratura, infatti, esistono pochi casi che trattano il problema appena enunciato. In [6], gli autori utilizzano un approccio PAA che tratta il problema della classificazione dei dati prodotti da sensori come un problema basato su dizionari, e utilizza gli interval slope come feature. In [4], gli autori estraggono da una piattaforma pubblica i dati di sensori annotati da parte degli utenti e deducono l'affidabilità e l'attendibilità dei valori ottenuti in relazione ad un valore di riferimento, il quale è preso da una fonte certificata. In [17], infine, gli autori classificano i datastream in base ai metadati forniti da essi.

Nell'ambito dello studio sperimentale svolto e spiegato all'interno di questa tesi, il focus viene posto sul cercare di annotare e classificare dei datastream prodotti da dispositivi e ambienti IoT eterogenei. Si cerca, quindi, di affrontare le problematiche e le sfide poste dall'utilizzo di Open Data forniti da unreliable repository come Thingspeak. Nel caso preso in esame,

l'obiettivo sarà quello di ottenere un dataset contenente un buon numero di datastream provenienti da Thingspeak, i quali saranno poi annotati e classificati come se fossero presi da dataset forniti da reliable repository. Il risultato che si cerca di ottenere è quello di avere un dataset che si comporterà in maniera "simile" a quelli forniti da reliable repository nell'applicazione di determinati algoritmi di classificazione.

# Capitolo 2

# Machine Learning

Questa è l'epoca in cui tutti siamo produttori di dati. Una volta solo le imprese li possedevano e li producevano; oggi, grazie al miglioramento delle tecnologie e, soprattutto, alle comunicazioni wireless, siamo tutti in grado di produrre e condividere dei dati. Ognuno di noi non è solo un produttore di dati, ma anche un consumatore. Basta pensare al fatto che ogni individuo vuole avere prodotti e servizi specializzati per i propri bisogni. Vogliamo anche che le nostre necessità vengano capite e che i nostri interessi vengano predetti senza doverli specificare [1]. Un esempio può essere un sito web di abbigliamento, il quale vorrebbe cercare di capire il comportamento dei consumatori per poter, ad esempio, consigliare loro cosa comprare in abbinamento alla merce presente nel carrello. Per cercare di predire i comportamenti dei consumatori bisogna andare a ricercare dei pattern all'interno dei dati prodotti dal sito web ogni giorno: che capi sono stati acquistati, quali sono stati visionati e molti altri. Si vanno a ricercare dei pattern perché la gente non acquista cose a caso, ma segue sempre certe logiche. Con questo si cerca di risolvere il problema della poca conoscenza andando ad analizzare i dati, si cerca di "apprendere" da essi. Da queste operazioni non si cerca di identificare completamente un processo d'acquisto, ma si cerca di ottenerne una buona approssimazione. Essa potrà non essere applicabile per tutti i dati, ma sarà applicabile per una parte di essi per cercare pattern di comportamento e utili per effettuare delle predizioni. Il voler cercare approssimazioni, pattern e regolarità attraverso i dati è uno degli obiettivi principali del machine

learning.

## 2.1 Che cos'è il machine learning?

Machine learning è la programmazione di computer per ottimizzare uno o più criteri di performance utilizzando dati o esperienze passate. Avendo dei modelli definiti con certi parametri, la parte di apprendimento (learning) avviene con l'esecuzione di un programma/algoritmo (computer program) che ha il compito di ottimizzare i parametri utilizzando i training data (dati con cui "addestrare" l'algoritmo) o le esperienze passate. I modelli possono essere "predictive" se si vogliono fare delle previsioni sul futuro, "descriptive" se si vuole acquisire conoscenza dai dati, o di entrambe le tipologie [1]. Il machine learning utilizza teorie statistiche e modelli matematici per svolgere le sue attività, questo perché quella principale è di effettuare inferenze da un campione di dati.

Il machine learning, in generale, ha un grande varietà di possibili applicazioni: il learning association, la classificazione, la regressione e tante altre. La classificazione è uno dei maggiori problemi a cui è dedicata una parte degli studi del machine learning. Principalmente, i metodi di machine learning possono essere di due tipologie: supervised (supervisionati) o unsupervised (non supervisionati).

L'applicazione a grandi database delle tecniche e dei metodi di machine learning viene chiamata "Data Mining". Con il data mining si cerca di processare un grande volume di dati per cercare di costruire un modello semplice, valido e comprensibile. Il modello ricavato dovrà essere utile per raggiungere diversi obiettivi, come cercare di fornire previsioni il più accurate possibili.

## 2.2 Supervised Learning & Unsupervised Learning

Come affermato in precedenza, esistono due tipologie principali di metodi di machine learning: supervised learning e unsupervised learning.

I metodi supervised hanno l'obiettivo di individuare la relazione esistente fra attributi in input e in output dei dati. Successivamente, questi metodi si fanno carico di costruire il modello con i risultati ottenuti. Infatti, con i metodi supervised: si hanno classi decise, conosciute e numerate, si utilizzano training set per classificare le osservazioni future. I più importanti metodi supervisionati sono la classificazione e la regressione.

I metodi unsupervised raggruppano i campioni di dati senza avere una conoscenza preliminare o uno schema già pronto da seguire. Quindi, con questo tipo di metodi le classi e la loro numerazione non sono conosciute in partenza. I metodi non supervisionati vengono utilizzati nell'esplorazione dei dati. I più importanti metodi unsupervised sono l'associazione, la generalizzazione ("dimension reduction") e il clustering.

Durante l'implementazione pratica dello studio sperimentale presentato in questa tesi sono stati utilizzati sia metodi supervised che metodi unsupervised. Tra quelli unsupervised è stato utilizzato il clustering, mentre tra quelli supervised è stata utilizzata la classificazione tramite diversi algoritmi. L'utilizzo di questi metodi di machine learning è spiegato con maggiori dettagli nei capitoli successivi.

## 2.3 Algoritmi di classificazione

La classificazione, come accennato in precedenza, fa parte dei metodi di supervised learning, quindi necessita di dati già annotati con determinate classi. Infatti, le tecniche di classificazione necessitano di un training set contenente dati etichettati con classi note e precise. Utilizzando il training set, gli algoritmi di classificazione costruiscono un modello che verrà utilizzato per classificare nuovi dati contenuti all'interno di test set.

Esistono tantissimi algoritmi di classificazione [12] e durante la realizzazione di questo studio ne sono stati utilizzati cinque, i quali sono descritti qui di seguito:

- Random Forest: [5] è un algoritmo di classificazione che, a partire dai dati passati in input, consiste nel creare un elevato numero di deci-

sion tree[1], i quali opereranno come un insieme unico (una foresta). Ogni decision tree generato produrrà una previsione sulla classe risultante e quella con più voti (appare più volte nelle previsioni dei decision tree) sarà presa come la previsione del modello. Questo algoritmo si basa sul seguente concetto: se tanti modelli relativamente non correlati (tree) operano come un unico insieme, questo sarà in grado di superare le prestazioni di qualsiasi modello che lo costituisce preso individualmente.

- k-Nearest Neighbor (k-NN): è un algoritmo di classificazione che, per ogni data point del test set fornito, va a "guardare" tra i K data point più vicini del training set e prende la classe che ha più occorrenze tra di essi. La classe ottenuta viene assegnata al dato del test set. Il numero K rappresenta il numero di data point più vicini al dato preso in considerazione dal test set, di cui si vuole predire la classe.

- C4.5: è un algoritmo di classificazione che sfrutta i decision tree come Random Forest. Esso genera un decision tree a partire da un training set passato in input insieme alle relative classi dei suoi data point. Questo algoritmo è uno dei successori dell'algoritmo ID3 e utilizza il concetto di "information entropy"[2].

- Naive Bayes: è un algoritmo di classificazione che si avvale del teorema di Bayes[3]. Esso si basa anche sul fatto che tutte le caratteristiche non siano influenzate l'una dall'altra. Il Naive Bayes calcola innanzitutto le probabilità delle classi fornite, poi calcola la probabilità condizionata delle caratteristiche del problema e, infine, calcola la probabilità per

[1]Il decision tree è un classificatore con struttura ad albero, i cui nodi sono nodi interni (il test effettuato su un attributo) o foglie (valore della classe assegnato ad un'istanza). Il suo obiettivo è quello di individuare gli attributi utili per classificare le istanze di un training set.

[2]Information entropy è la media con cui le informazioni vengono prodotte da una certa sorgente di dati.

[3]Il teorema di Bayes, in statistica e teoria della probabilità, descrive la probabilità di un evento in base alla conoscenza delle condizioni che potrebbero essere legate all'evento. Questo metodo è utile per capire la probabilità condizionata.

prevedere la classe di appartenenza della nuova istanza. La decisione finale è la classe che ottiene il valore di probabilità maggiore.

- Support Vector Machine (SVM): è un algoritmo di classificazione che si basa su iper-piani[4] e vettori di supporto[5]. In questo algoritmo, ogni data point viene posizionato graficamente su un piano di N dimensioni, dove N è il numero delle feature che descrivono i dati (il valore di ogni feature rappresenta il valore di una specifica coordinata). Successivamente, l'algoritmo esegue la classificazione trovando l'iper-piano che divide al meglio il set di dati in due classi.

L'applicazione pratica di questi algoritmi sarà descritta all'interno del Capitolo 5. I risultati riguardanti i test effettuati con essi saranno mostrati e analizzati all'interno del Capitolo 6.

[4]Per un'attività di classificazione con due dimensioni spaziali, X e Y, un iper-piano è una linea che separa e classifica un set di dati.

[5]I vettori di supporto (support vector) sono i punti più vicini all'iper-piano. Essi vengono considerati come gli elementi critici di un set di dati.

# Capitolo 3

# Progettazione

In questo capitolo verranno descritte ad alto livello le varie fasi di lavoro che hanno interessato lo sviluppo dello studio sperimentale preso in esame per questa tesi.

## 3.1 Obiettivi e task

Gli obiettivi principali dello studio sperimentale svolto sono stati: (1) realizzare un dataset che si comporti in maniera simile a dataset esterni forniti da esperimenti, studi o organizzazioni; (2) individuare l'algoritmo maggiormente performante sui dataset considerati; (3) verificare che la classificazione basata sulla successione dei dati non funziona sul tipo di dataset considerati. I dataset cosiddetti esterni, utilizzati per il confronto in questo studio, sono due e vengono cosí chiamati: Swissex[1] (Swiss Experiment) e Urban Observatory[2]. Entrambi sono stati creati da esperimenti di ricerca riguardanti il Sensor Metadata Management e la sua applicazione in ambito di ricerche ambientali collaborative. Il dataset creato durante questo studio, invece, è stato realizzato utilizzando i dati provenienti da Thingspeak[3]: una piattaforma open source per l'IoT che permette di raccogliere e archiviare dati inviati

[1] https://www.swiss-experiment.ch/
[2] http://newcastle.urbanobservatory.ac.uk/
[3] https://thingspeak.com/

da vari dispositivi su internet o su una rete locale. Maggiori informazioni su Thingspeak sono state fornite all'interno del Capitolo 1.

Per raggiungere gli obiettivi prefissati alla base di questo studio si è suddiviso il lavoro in due macro-fasi: (1) Realizzazione del dataset e (2) Sperimentazione su di esso. Nella prima macro-fase è stato creato il dataset con i dati provenienti da Thingspeak, mentre nella seconda macro-fase sono state svolte le attività di validazione e gli esperimenti sul dataset realizzato.

La macro-fase di realizzazione del dataset è stata suddivisa, a sua volta, in diverse fase di lavorazione, le quali sono le seguenti:

- Preparazione dei dati
- Clustering spaziale
- Clustering temporale
- Interpolazione
- Annotazione

La macro-fase riguardante le sperimentazioni sul dataset, invece, è stata suddivisa nelle seguenti fasi di lavorazione:

- Analisi della varianza (validazione del dataset)
- Test con algoritmi di classificazione (esperimenti per individuare l'algoritmo con le performance migliori e per verificare che la classificazione basata sulla successione dei dati non funziona sul tipo di dataset considerati)

Tutte le fasi di lavorazione delle due macro-fasi sono descritte brevemente nelle sezioni successive di questo capitolo. Le fasi relative alla realizzazione del dataset saranno approfondite all'interno del Capitolo 4, quelle relative alla fase di sperimentazione, invece, saranno approfondite e analizzate nei Capitoli 5 e 6.

## 3.2 Preparazione dei dati

Come accennato nella sezione precedente, la prima fase di lavoro è stata quella di preparazione dei dati, durante la quale si sono dovuti ottenere ed elaborare i dati dalla piattaforma online Thingspeak. Questa fase di lavoro è stata suddivisa in due task: (a) Download dei channel pubblici da Thingspeak; (b) Estrazione dei metadati.

### 3.2.1 Download channel

Durante questo primo task è stato effettuato il download di tutti i channel di Thingspeak. Nel download sono stati considerati solamente i channel pubblici disponibili. Ogni canale è stato salvato in un file in formato json.

### 3.2.2 Estrazione dei metadati

Dopo aver completato il download dei channel pubblici da Thingspeak l'attenzione è stata riposta alla realizzazione di uno script che si occupasse dell'estrazione dei metadati dai file JSON generati. L'obiettivo, infatti, è stato quello di estrarre i metadati nel seguente ordine: id autoincrementale (generato automaticamente nello script), nome del datastream, id Thingspeak del datastream, nome del canale, descrizione (se presente), data di creazione, latitudine, longitudine, numero di letture. I metadati estratti sono stati successivamente scritti in file con estensione csv.

## 3.3 Clustering spaziale

Dopo la prima fase, il focus si è spostato verso il dover effettuare un clustering spaziale sui dati ottenuti precedentemente. Quando si parla di clustering ci si riferisce ad un insieme di tecniche di analisi dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati. Per il clustering spaziale, ovviamente, ci si è basati sulle coordinate, latitudine e longitudine, dei datastream presenti nel file csv creato precedentemente. Tutto ciò è stato finalizzato alla ricerca di grandi zone (cluster) all'interno

delle quali sono concentrati un numero elevato di datastream. Queste grandi zone possono essere, visto che sono state prese in considerazione latitudine e longitudine, continenti, Stati, regioni o altro. L'obiettivo finale, in poche parole, è stato quello di trovare i migliori cluster possibili per continuare con le fasi di lavorazione successive.

## 3.4 Clustering temporale

Completato il clustering spaziale si è passati al clustering temporale. In questa tipologia di clustering sono state prese in considerazione le misurazioni fornite per ogni datastream nei rispettivi channel pubblici su Thingspeak. Questo perché l'obiettivo principale del clustering temporale è trovare il periodo di tempo in cui è presente il maggior numero di datastream con almeno un determinato numero K di misurazioni. Nel caso trattato in questa tesi, come periodo di tempo è stato preso in considerazione un giorno intero, mentre come K sono stati provati diversi valori. Al termine di questa fase si avrà un cluster di dimensioni leggermente ridotte rispetto a quello di partenza, il quale sarà memorizzato all'interno di un file csv.

## 3.5 Interpolazione

La fase successiva a quelle di clustering è l'interpolazione. Per interpolazione si intende un metodo per "indovinare" i valori mancanti a partire da un insieme di dati conosciuti. In questo caso di studio si è in possesso di un cluster di datasteam aventi un certo numero di misurazioni durante un giorno intero. Per effettuare l'interpolazione l'idea è stata quella di dividere un giorno in quarti d'ora ottenendo 96 slot di tempo. Ogni slot conterrà la media delle misurazioni effettuate al relativo quarto d'ora. Un problema può nascere dal fatto che un datastream possa non avere misurazioni in un certo quarto d'ora. La soluzione a ciò è l'utilizzo dell'interpolazione che produrrà il valore mancante per quello slot di tempo. Così facendo, ogni datastream avrà 96 misurazioni oltre ai suoi metadati.

## 3.6 Annotazione

Nella fase di annotazione l'attenzione è stata posta sul dover assegnare una classe ad ogni datastream del nostro cluster memorizzato su file. Per classe si intende un'etichetta che definisce l'appartenenza ad una certa tipologia di dati o ambito. I dati utilizzati in questa tesi sono di tipo ambientale, quindi si troveranno classi come temperatura, umidità, pressione atmosferica, ecc. Il prodotto finale di questa fase di lavorazione sarà il dataset sul quale andremo a testare i vari algoritmi di classificazione.

## 3.7 Analisi della varianza

Prima di andare ad effettuare i test con gli algoritmi di classificazione è stato necessario individuare le feature da dare in input agli stessi. Per feature si intende un data point che viene scelto per definire l'input. Dopo aver individuato le feature si è proseguito nell'effettuare l'analisi della varianza ANOVA per ciascuna di esse. Questa analisi ha permesso di individuare le feature più e meno significative. Al termine di questa fase le feature sono state ordinate da quella maggiormente significativa a quella minormente significa.

## 3.8 Test con algoritmi di classificazione

Dopo aver individuato ed ordinato le feature, l'ultima fase di lavorazione rimanente è quella di effettuare dei test con alcuni algoritmi di classificazione sul dataset Thingspeak creato precedentemente e sui dataset Swissex e Urban Observatory. Questa fase di test è stata suddivisa in due task.

### 3.8.1 Selezione dell'algoritmo maggiormente performante

Il primo task consiste nel provare cinque algoritmi di classificazione diversi e calcolarne l'accuracy [23], la quale è una statistica per la valutazione dei modelli di classificazione. Ogni algoritmo viene testato aumentando progressivamente il numero di feature prese in considerazione (da 1 fino ad N). Dai

risultati di queste prove verrà scelto l'algoritmo maggiormente performante (Sezione 5.2).

### 3.8.2 Test dell'algoritmo Random Forest su partizionamenti dei datastream

In questo secondo task si è deciso di effettuare le stesse attività svolte nel task precedente con l'algoritmo di classificazione scelto, però applicandolo ad ogni dataset con datastream partizionati. Per ogni prova con l'algoritmo di classificazione, in questo caso, sono state calcolate sia l'accuracy che la f-measure. F-measure [23], o F1-score, è una statistica per la valutazione simile ad accuracy. Il processo di partizionamento verrà spiegato con maggior dettaglio nella sezione (Sezione 5.3).

## 3.9 Strumenti utilizzati

Per realizzare gli script che hanno permesso di effettuare gli studi relativi a questa tesi si è utilizzato il linguaggio Python[4]. Esso è un linguaggio di programmazione ad alto livello, interattivo e multi-paradigma. Viene utilizzato per la computazione numerica, lo sviluppo di applicazioni distribuite ed è uno dei linguaggi di programmazione utilizzati maggiormente per il machine learning.

Un altro strumento fondamentale per la realizzazione di questo studio sperimentale è Scikit-learn[5], una libreria open source di Python che supporta il machine learning. Essa fornisce molti strumenti utili quali algoritmi di classificazione, clustering e regressione.

Per la traduzione di alcuni nomi e descrizione è stata utilizzata la libreria Translate[6] fornita da Google. Per la realizzazione dei grafici 2D e 3D, sono stati utilizzati diversi metodi forniti dalla libreria Matplotlib[7]. Per i grafici con le cartine geografiche, invece, è stata utilizzata la libreria Basemap[8].

[4]https://www.python.org/
[5]https://scikit-learn.org/stable/
[6]https://cloud.google.com/translate/
[7]https://matplotlib.org/
[8]https://matplotlib.org/basemap/

Per tutte le operazioni matematiche come l'interpolazione e il calcolo delle feature sono state utilizzate le librerie Numpy[9] e Scipy[10].

[9]https://www.numpy.org/
[10]https://www.scipy.org/

# Capitolo 4

# Realizzazione del dataset

In questo capitolo si andranno a descrivere con maggior dettaglio le fasi di lavoro riguardanti l'attività di creazione del dataset con dati Thingspeak, descritte brevemente nel capitolo precedente riguardante la progettazione. Verranno discussi alcuni pezzi di codice degli script Python, grafici intermedi, scelte implementative e molto altro.

## 4.1 Preparazione dei dati

Come descritto nella sezione 3.1 questa è la fase in cui si è dovuto ottenere e poi elaborare i dati da Thingspeak.

### 4.1.1 Thingspeak public channels download

In questa attività si è deciso di effettuare il download dei dati dai vari channel presenti sulla piattaforma Thingspeak. I channel sono delle locazioni in cui un utente registrato può archiviare i propri dati. Ognuno di questi channel può contenere un massimo di 8 datastream (serie di dati). Per ottenere dati più recenti si è deciso di effettuare il download solo dei channel pubblici che fornissero dati a partire dalle ore 00:00 del 01/01/2019. Inoltre, i dati scaricati dovevano essere formattati in un file JSON.

Per il download si è dovuto creare un breve script che effettuasse una serie di richieste GET HTTP verso Thingspeak. Il percorso utilizzato per le richieste è il seguente:

https://thingspeak.com/channels/idChannel/feed.json?
results=8000&start=2019-01-01 00:00:00

dove idChannel è il numero identificativo del channel all'interno di Thingspeak, results indica il numero massimo di letture da ottenere e start indica la data e l'orario da cui partire a prendere le misurazioni.

L'idChannel massimo raggiunto è stato 756322, mentre il numero totale di channel pubblici scaricati sotto forma di file json è stato 192320. Questo fa capire la grandezza del bacino di dati contenuti nella piattaforma Thingspeak.

Al termine delle attività di download, i file json ottenuti si presentano con la struttura visibile nella figura 4.1.


```
b'{"channel":{"id":82797,"name":"My room & outside","description":"Temperature inside and outside
of my cosy room. Powered by two DS17B20 temperature sensors controlled by Raspberry Pi.",
"latitude":"52.4166667","longitude":"16.9666667","field1":"Room temperature","field2":"Outside
temperature","created_at":"2016-01-30T13:06:48Z","updated_at":"2018-12-02T13:16:38Z",
"last_entry_id":187978},"feeds":[{"created_at":"2019-04-04T09:45:25Z","entry_id":179979,
"field1":"20.3","field2":"18.9"},{"created_at":"2019-04-04T09:46:25Z","entry_id":179980,
"field1":"20.4","field2":"19.1"},{"created_at":"2019-04-04T09:47:25Z","entry_id":179981,
"field1":"20.5","field2":"19.3"},{"created_at":"2019-04-04T09:48:25Z","entry_id":179982,
```


Figura 4.1: Un esempio di come si presenta il file json di un channel

Come mostrato in figura, il file json ha due campi principali: channel e feeds. Il campo channel è quello che contiene, a sua volta, altri campi, i quali forniscono informazioni riguardanti il channel stesso: i metadati. Il campo feeds, invece, contiene un array con tutte le misurazioni. Importanti sono i campi denominati field all'interno di channel, visto che rappresentano i datastream.

### 4.1.2 Estrazione dei metadati

Completata la fase di download dei dati si è deciso di sviluppare uno script che permettesse l'estrazione dei metadati dai file json, così da ottenere un unico file, con estensione csv, contenente tutte le informazioni su datastream e channel. Al termine di questa attività, ogni riga del file csv sarà formattata come mostrato in Figura 4.2.

```
1826$TEMPERATURE$173889$METEOROLOGICAL STATION$null$2016-10-22T12:36:40Z$41.0$2.0$8000
1827$HUMIDITY$173889$METEOROLOGICAL STATION$null$2016-10-22T12:36:40Z$41.0$2.0$8000
1828$DEW POINT$173889$METEOROLOGICAL STATION$null$2016-10-22T12:36:40Z$41.0$2.0$8000
1829$ATMOSPHERIC PRESSURE$173889$METEOROLOGICAL STATION$null$2016-10-22T12:36:40Z$41.0$2.0$8000
1830$TEMPERATURE BMP180$173889$METEOROLOGICAL STATION$null$2016-10-22T12:36:40Z$41.0$2.0$8000
```

Figura 4.2: Esempio di formattazione dei metadati

La formattazione seguita in ogni riga comprende i seguenti metadati: id autoincrementale (generato automaticamente), nome del datastream, id Thingspeak del datastream, nome del canale, descrizione (se presente), data di creazione, latitudine, longitudine, numero di letture. Una decisione importante è stata quella di estrarre i metadati soltanto dai channel che rispettassero le seguenti condizioni:

- Avere valori di latitudine e longitudine validi, cioè non devono essere uguali a zero o assumere valori non validi tipo “null” o “None”.
- Avere un numero di misurazioni maggiore di zero.

Un problema sorto durante lo sviluppo è stato quello di trovare molti metadati scritti in lingue diverse da quella inglese. I metadati interessati da questo problema sono stati: il nome del datastream, il nome del channel e la descrizione. La soluzione proposta è stata quella di effettuare una traduzione dei metadati citati precedentemente prima della loro scrittura all’interno del file csv. Per effettuare la traduzione è stata utilizzata la libreria Translate fornita da Google.

Il file csv risultante conteneva 20105 righe, quindi 20105 datastream.

## 4.2 Clustering spaziale

La fase successiva a quella di preparazione dei dati è il clustering spaziale. Come descritto brevemente nella sezione 3.3, l’obiettivo di questa fase è quello di trovare i migliori cluster geografici, visto che ci si basa su latitudine e longitudine dei datastream, per poter proseguire lo studio sperimentale di questa tesi. I dati relativi alla latitudine e longitudine dei datastream sono stati reperiti dal file csv ottenuto delle attività precedenti.

Per realizzare il clustering spaziale si è utilizzato l'algoritmo DBSCAN, la cui applicazione si può vedere in Listing 4.1. DBSCAN è fornito dalla libreria Scikit-learn ed è un metodo di clustering basato sulla densità e necessita di due parametri:

- eps: la massima distanza che deve esserci tra due punti per essere considerati nello stesso vicinato.
- min_samples: il numero minimo di punti, in un vicinato, richiesto per formare un cluster.

```
#pos = array con latitudine e longitudine
tmp = np.array([x for x in pos])
X = tmp[0:len(tmp)]

#fit dei dati in un formato comprensibile per DBSCAN
ss = StandardScaler()
X_scaled = ss.fit_transform(X)

#applicazione DBSCAN
db = DBSCAN(eps=0.01, min_samples=300).fit(X_scaled)
y_pred = db.fit_predict(X_scaled)
```

Listing 4.1: Applicazione DBSCAN

Il risultato dello script (Figura 4.3) con l'utilizzo di DBSCAN è stato visualizzato su una cartina geografica dove è possibile vedere visivamente i vari cluster trovati. Dalla cartina in Figura 4.3 si può notare come siano stati trovati cinque maggiori cluster: Europa (rosso), Asia(verde), Australia (giallo), Nord e Centro America (blu), America Latina (marrone). Tutti i punti di color grigio rappresentano i punti noise, ovvero quei datastream che DBSCAN non è riuscito ad accorpare in uno dei cluster da esso trovati. Per realizzare la cartina geografica è stata utilizzata la libreria Basemap, mentre per i punti su di essa è stato utilizzato uno scatter plot della libreria Matplotlib.

Analizzando i dati mostrati graficamente in Figura 4.3 si è deciso di mantenere solamente i cluster contenenti un numero maggiore di mille punti,

rimanendo cosí con tre macro-cluster principali: Nord-Centro America, Europa e Asia.

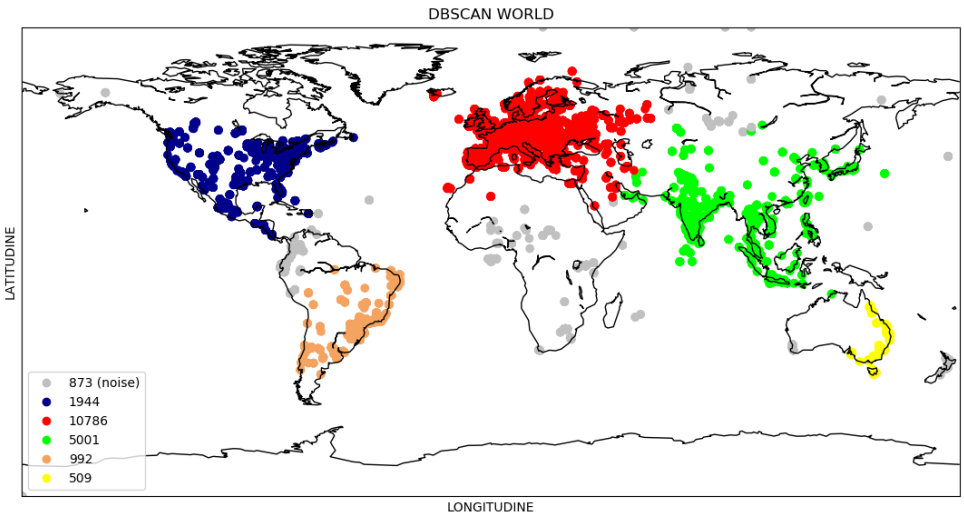



Figura 4.3: Cluster individuati da DBSCAN

I tre macro-cluster, successivamente, sono stati trascritti in un file con estensione csv ciascuno. Questa azione si è resa necessaria per poter effettuare un secondo giro di clustering (sub-clustering) su ognuno di essi per cercare di ottenere un maggior dettaglio e coerenza dei dati per le successive fasi di lavorazione. Per implementare il sub-clustering, si è realizzato un nuovo script quasi completamente identico a quello utilizzato per il clustering. I risultati (Figura 4.4, Figura 4.5 e Figura 4.6) di questa seconda implementazione sono stati visualizzati, ancora una volta, tramite delle cartine geografiche.

Osservando i risultati del sub-clustering del Nord-Centro America è stato notato come i datastream (punti) fossero molto distanti fra loro e l'unico cluster individuato avesse dimensioni modeste (cluster rosa in Figura 4.6). Per questo, il cluster della regione americana è stato scartato e non comparirà più nelle fasi successive. I risultati del sub-clustering europeo, invece, hanno mostrato l'individuazione di un cluster con un buon numero di da-

tastream (cluster rosso in Figura 4.5), quindi è stato deciso di tenerlo per le fasi successive memorizzandolo in un file csv. Il sub-clustering della zona asiatica è stato quello maggiormente difficile da analizzare visto che sono stati individuati ben 4 cluster. Alla fine si è preferito optare per il cluster comprendente le zone della Cina, Taiwan, Corea e Giappone (cluster blu in Figura 4.4). Il cluster selezionato è stato anch'esso memorizzato in un file csv.

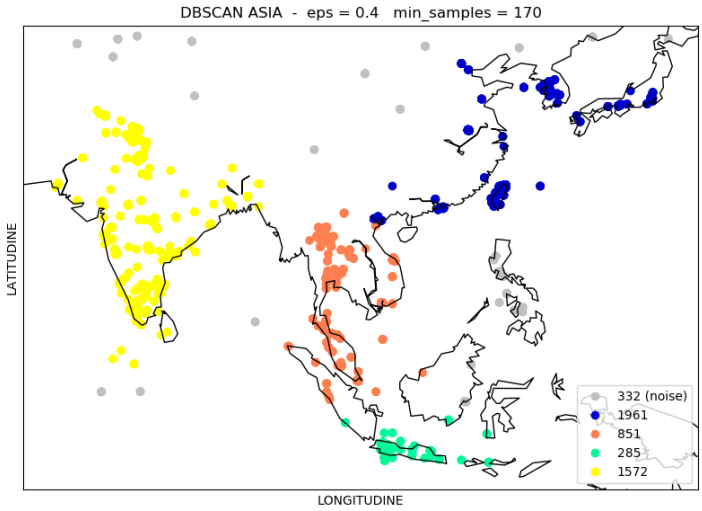



Figura 4.4: Sub-clustering asiatico

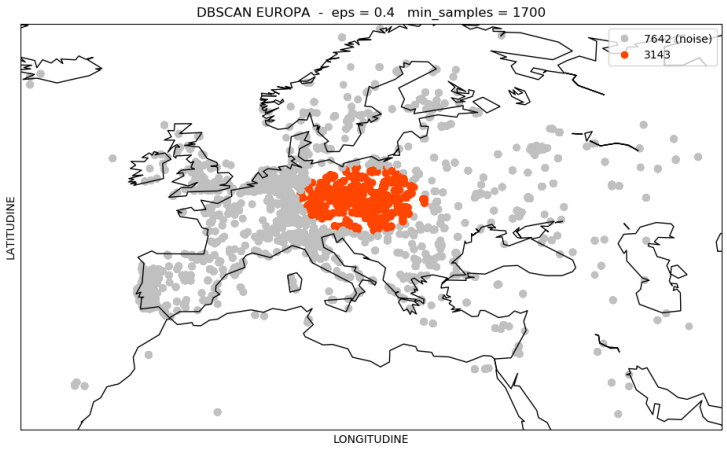



Figura 4.5: Sub-clustering europeo

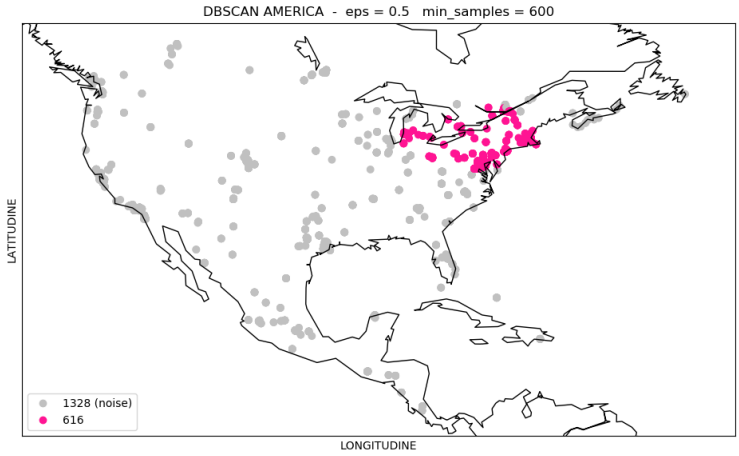



Figura 4.6: Sub-clustering americano

## 4.3 Clustering temporale

Dopo aver terminato il clustering spaziale e generato i rispettivi file csv per ogni cluster selezionato, l'attenzione si è spostata verso la realizzazione della fase di clustering temporale. Ricordando che ogni cluster è composto da un certo numero di datastream, il clustering temporale ha l'obiettivo di individuare il periodo di tempo nel quale è presente il maggior numero di datastream che abbiano almeno K misurazioni. Il periodo di tempo che è stato fissato è di un giorno intero (24 ore complete).

Innanzitutto, è stato necessario recuperare tutte le misurazioni per ogni datastream visto che nella fase descritta nella Sezione 3.1.2 sono stati estratti solo i metadati (informazioni). Per fare questo, si è realizzato un veloce script che, utilizzando l'id del channel Thingspeak e il nome del datastream, accedesse ai file json ottenuti al termine del download dei dati (Sezione 4.1.1). Le misurazioni sono così state memorizzate in un file csv, dove ogni riga rappresenta un datastream (Figura 4.7). All'inizio di ogni riga è stato posto l'id autoincrementale assegnato al datastream al momento dell'estrazione dei metadati. Le misurazioni sono state scritte all'interno del file secondo la seguente formattazione:

timestamp_misurazione$valore_misurazione

dove timestamp_misurazione si presenta con il formato DD-MM-YY hh-mm-ss (giorno-mese-anno ore-minuti-secondi) e rappresenta il giorno e l'orario in cui è avvenuta la misurazione; mentre valore_misurazione rappresenta il valore numerico misurato in quell'istante. Questa fase di estrazione delle misurazioni è stata svolta sia per il cluster europeo che per quello asiatico (Sezione 4.2).

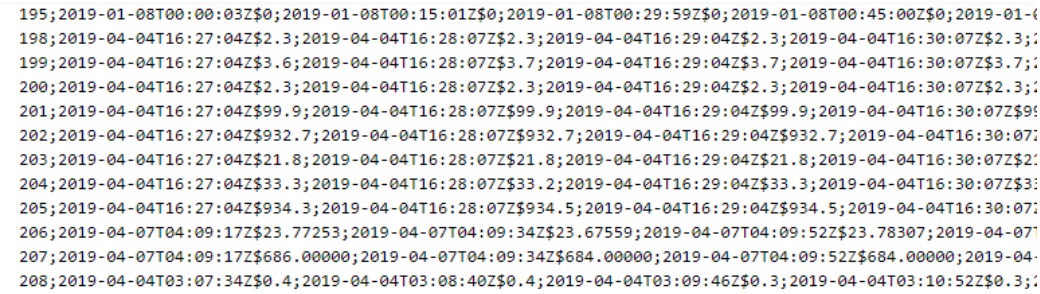



Figura 4.7: Struttura del file contenente le misurazioni dei datastream

Completata questa operazione intermedia si è passati alla realizzazione vera e propria del clustering temporale. La soluzione pensata è stata suddivisa in tre parti:

1. Creare un array per ogni datastream in cui ogni suo elemento contenesse il numero totale di misurazioni avvenute nell'arco di un'ora a partire dal 01/01/2019 alle ore 00:00 al giorno in cui è stato fatto partire il download dei channel pubblici di Thingspeak, ovvero il 12/04/2019 alle ore 12:00. Questo array viene definito d'ora in avanti come "intermedio". Ogni array avrà una lunghezza di 2436 posizioni.

2. Realizzare un nuovo array per ogni datastream della stessa lunghezza di quello intermedio creato al punto 1. Ogni elemento di questo nuovo array conterrà un valore binario che indicherà se, nelle 24 ore a partire dall'ora indicata da quella posizione, è presente almeno un certo numero K di misurazioni, avendo, inoltre, misurazioni sia nella prima che nell'ultima ora del periodo di tempo descritto. Se i requisiti appena descritti sono soddisfatti, l'elemento in posizione P avrà come valore 1, altrimenti 0. Questo array viene chiamato "binario". Un esempio di

funzionamento è il seguente: l'elemento in prima posizione di binario rappresenta le 24 ore a partire dall'orario rappresentato nella stessa posizione in intermedio, ovvero la mezzanotte dell'1 gennaio 2019. Per trovare il valore dell'elemento in prima posizione di binario si dovrà fare la somma di 24 elementi (rappresentano le 24 ore) a partire dalla stessa posizione (la prima) in intermedio. Se la somma sarà maggiore di un determinato valore K e si avranno valori maggiori di zero nella prima e nell'ultima ora, allora l'elemento in prima posizione di binario sarà 1 e non 0.

3. Effettuare la somma delle colonne degli array binari. Cosí facendo si ottiene un array di somme in cui si dovrà individuare il valore maggiore (il massimo). La posizione del valore massimo all'interno del vettore delle somme indicherà anche il giorno e l'orario in cui è presente il numero maggiore di datastream.

Vista la complessita della soluzione pensata, sono stati realizzati due script:

- Uno che creasse e andasse a memorizzare in un file csv gli array intermedi (un pezzo significativo di script si può vedere in Listing 4.2)
- Uno che realizzasse gli array binari e ne effettuasse la somma per colonne trovando cosí il giorno e l'orario con il maggior numero di datastream (un pezzo significativo di script si può vedere in Listing 4.3)

```
#letture = array contenente tutte le misurazioni di un datastram
for read in letture:
    #controllo che il valore di una misurazione sia numerico
    if read[1] != "None" and read[1] != "null" and read[1] != "
   nan" and read[1] != "love":
        posizione = None
        m = read[0].month
        d = read[0].day
        h = read[0].hour
        mesi_precedenti = 0

```

```
11          #trovo la posizione all'interno dell'array intermedio in
        cui aggiungere la misurazione
12          if m == GENNAIO:
13              mesi_precedenti = 0
14              posizione =  mesi_precedenti + (((d-1)*24) + h)
15          elif m > GENNAIO:
16              if m == FEBBRAIO: mesi_precedenti = 31*24
17              elif m == MARZO: mesi_precedenti = 31*24 + 28*24
18              elif m == APRILE: mesi_precedenti = (31*24)*2 + 28*24
19              posizione = mesi_precedenti + (((d-1)*24) + h)
20          contatori[posizione] += 1
21
22 #salvo tutti gli array intermedi creati per ogni datastream
23 with open(percorso_res + 'intermedio.csv', 'a', encoding='utf8')
        as ff:
24      vett = str(riga[0])
25      for cnt in contatori:
26          tmp = ';' + str(cnt)
27          vett += tmp
28      ff.write(vett + "\n")
```

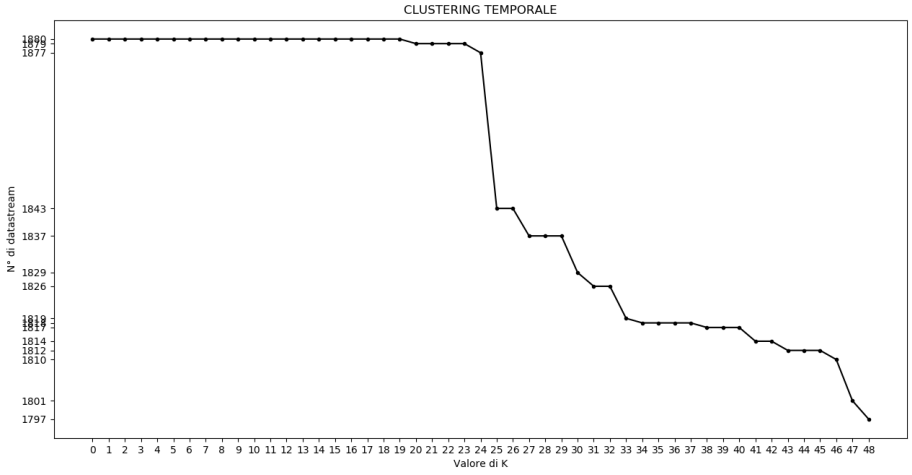
Listing 4.2: Realizzazione array intermedi

```
1 with open(path + 'intermedio.csv', 'r', encoding='utf8') as
        intermedio:
2      #prendo un array intermedio alla volta
3      for row in intermedio:
4          riga = row.strip().split(';')
5          ids.append(riga[0])
6          contatori = [int(value) for value in riga[1:len(riga)]]
7          binario = []
8          #indici per le 24 ore nell'array intermedio
9          i = 0
10         j = 23
11
12         while i<=NUM_TIME_SLOTS_min and j<NUM_TIME_SLOTS_max:
13             #controllo delle condizioni per avere 0 o 1 nell'
        elemento dell'array binario
14             if contatori[i] > 0 and contatori[j] > 0:
15                 somma = sum(x for x in contatori[i:j+1])
16                 if somma >= k:
```

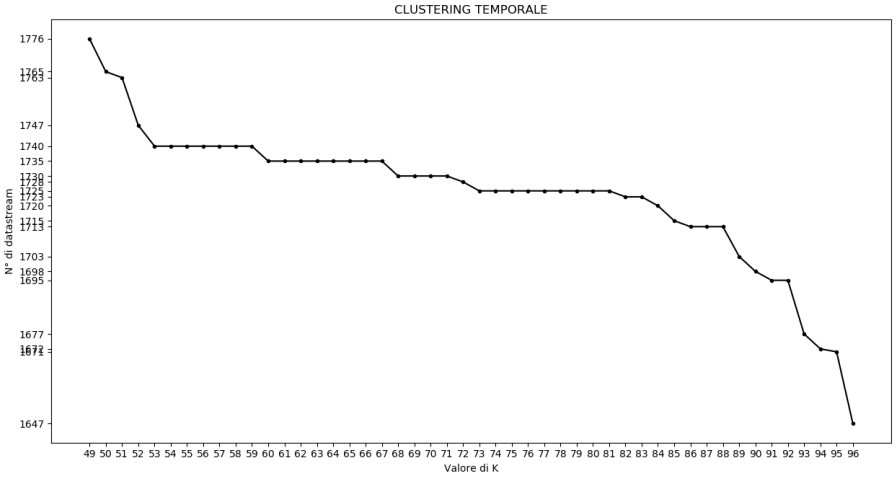
```
                    binario.append(1)
                else:
                    binario.append(0)
            else:
                binario.append(0)
            i += 1
            j += 1
        binari.append(binario)

#somma colonne, individuare valore max e posizione
somme = [sum(x) for x in zip(*binari)]
max_value = max(somme)
pos = [index for index,value in enumerate(somme) if value == max(
    somme)]
```

Listing 4.3: Realizzazione array binari e somma delle colonne

I due script realizzati sono stati applicati ai cluster trovati al termine del clustering spaziale, cioè quello europeo e quello asiatico. Il risultato con il cluster asiatico (area Cina, Taiwan, Giappone e Corea), partendo con un numero di datastream pari a 1961, è stato di 790 datastream il giorno 07/04/2019 a partire dalle ore 23. Considerando che il K utilizzato in questo caso è stato 1, quindi un valore bassissimo per la somma, si è deciso di scartare definitivamente il cluster asiatico vista la modesta dimensione. I risultati con il cluster europeo si sono dimostrati migliori e sono stati provati tutti i valori di K da 0 fino a 96 (grafici in Figura 4.8). Da ricordare è il fatto che la dimensione del cluster europeo sia di 3143 datastream. Il valore massimo di K è da intendere come il numero di quarti d'ora che formano un giorno intero. Maggiori dettagli su questa scelta verranno forniti nella fase di Interpolazione (Sezione 4.4).




(a) 0 - 48




(b) 49 - 96

Figura 4.8: Risultati clustering temporale per diversi valori di K

Analizzando i risultati nei grafici in Figura 4.8 si è notato come questi non fossero molto convincenti visto che dai 3143 datastream di partenza si è passati a 1700/1800. Vedendo questi risultati si è deciso di voler ottenere un cluster contenente almeno più di 2000 datastream al termine delle attività di clustering temporale. Per avere un maggior numero di datastream al termine del clustering temporale si necessita di un numero maggiore di essi in entrata, cioè al termine del clustering spaziale. La soluzione, quindi, è stata quella di

ripetere il clustering spaziale utilizzando nuovamente l'algoritmo DBSCAN modificando i valori di eps e min_samples.

### 4.3.1 Scelta del miglior cluster

Come accennato precedentemente è stato necessario ripetere il clustering spaziale e quello temporale. Per evitare di andare per tentativi si è deciso di effettuare più volte il clustering spaziale andando a modificare il solo parametro eps di DBSCAN. In questo caso si sono presi ben venti valori compresi tra 0.4 a 0.6 (0.4 era il valore utilizzato per il primo clustering spaziale). Individuate e memorizzate in file csv le varie versioni dei cluster, si è passati a ripetere il clustering temporale su ognuna di esse utilizzando tutti i K (il numero minimo di misurazioni in un giorno intero per un datastream) compresi tra 24 e 96. Al termine di tutte queste attività, i risultati del clustering temporale sono stati tutti uniti in un unico file con estensione csv utilizzando la seguente formattazione:

valore_K$numero_datastream$valore_eps

dove valore_k indica il valore di K utilizzato nel clustering temporale, numero_datastream indica il numero totale di datastream risultante dal clustering temporale e valore_eps, infine, indica di quale cluster sia il risultato presente in quella stessa riga.

Per confrontare meglio i risultati ottenuti è stato realizzato un grafico 3D (Figura 4.9). Esso presenta venti linee di punti di colore diverso (è stato utilizzato uno scatter plot), dove ognuna di esse rappresenta i risultati con i vari valori di K per un cluster ottenuto con un certo valore di eps. Sull'asse delle X si trovano i valori di K, sull'asse delle Y i valori di eps e su quello delle Z il numero di datastream ottenuto con il clustering temporale. Il grafico è stato realizzato tramite l'utilizzo di Axes3D fornito dalla libreria Matplotlib.

Osservando i risultati grafici e quelli scritti su file si è deciso di optare per il cluster generato da DBSCAN con eps uguale a 0.45 con una dimensione di 5497 datastream (la sesta linea di colore blu a partire dal basso nel grafico in Figura 4.9). Per quanto riguarda il clustering temporale, il risultato scelto per il cluster selezionato è quello con K uguale a 45. Il numero totale massimo

di datastream, alla fine, è stato di 2974, risultante nel giorno 08/04/2019 a partire dalle ore 10. Successivamente questi datastream sono stati trascritti in un file con estensione csv utilizzando un breve e semplice script.

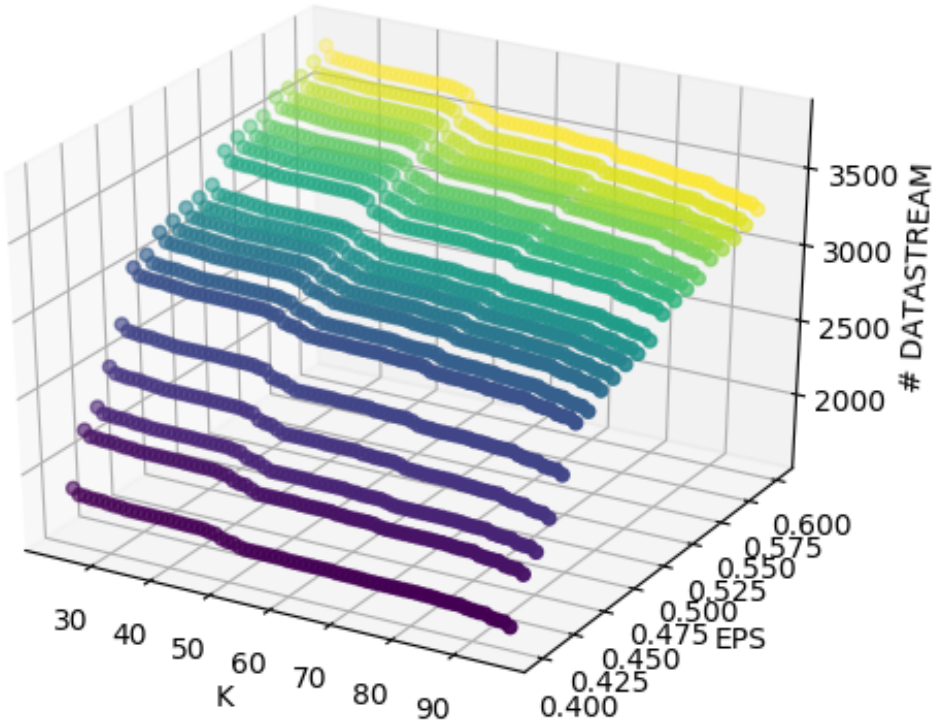



Figura 4.9: Scatter plot 3D dei risultati del clustering temporale su più cluster

## 4.4 Interpolazione

Dopo le operazioni di clustering l'obiettivo prefissato è stato quello di unire i due file csv principali realizzati fino a quel momento: il file con i metadati dei datastream ottenuto alla fine del clustering temporale e il file contente le varie misurazioni dei datastream. Da questo obiettivo è sorto immediatamente un problema abbastanza delicato: i datastream non possiedono lo stesso numero di misurazioni durante il giorno individuato al termine

del clustering temporale. La soluzione adottata per ovviare questo problema è stata quella di dividere una giornata in quarti d'ora, quindi in 96 time slot. Ogni time slot rappresenta un quarto d'ora e contiene la media delle misurazioni del datastream avvenute in quel periodo di tempo. Così facendo, ogni datastream avrà 96 valori numerici (misurazioni) oltre ai suoi metadati.

La soluzione dei 96 time slot, però, fa sorgere un nuovo problema: può capitare che un datastream non abbia misurazioni in certo quarto d'ora. Per ovviare a questa situazione di valore mancante è stato deciso di effettuare un'interpolazione, un metodo che cerca di "indovinare" i valori mancanti a partire da un insieme di dati conosciuti.

Per realizzare l'interpolazione sono stati realizzati 2 script:

1. Il primo script si è preoccupato di prendere il file delle misurazioni e creare un array, per ogni datasteam, con tanti elementi quanti il numero di quarti d'ora in un giorno, ovvero 96. Il giorno in cui ricercare le misurazioni per la creazione dell'array è quello che parte va dalle ore 10 del 08/04/2019 alle 10 del 09/04/2019. Nel caso in cui risultino quarti d'ora senza misurazioni, il relativo elemento nell'array avrà come valore zero (0.0 nel caso vengano utilizzati valori float). Al termine di queste operazioni gli array saranno scritti in un file csv insieme agli id autoincrementali (assegnati durante la fase di estrazione dei metadati) dei rispettivi datastream.

2. Il secondo script si è preoccupato di recuperare gli array memorizzati nel file csv con lo script precedente e di effettuare l'interpolazione (Listing 4.4). Essa è stata effettuata utilizzando il metodo interp1d fornito dal modulo interpolate della libreria Scipy. Al termine dell'interpolazione gli array completi risultanti sono stati scritti in un file con estensione csv.

```
1 #applico il metodo di interpolazione cubic spline e ottengo la
    funzione per l'interpolazione
2 f = interp1d(x, mean, kind='cubic', fill_value='extrapolate',
    bounds_error=False)
```

```
3
4 #applico la funzione per cercare di "indovinare" i valori
5 xn = np.linspace(0,95, num=96)
6 yn = f(xn)
```

Listing 4.4: Applicazione del metodo di interpolazione

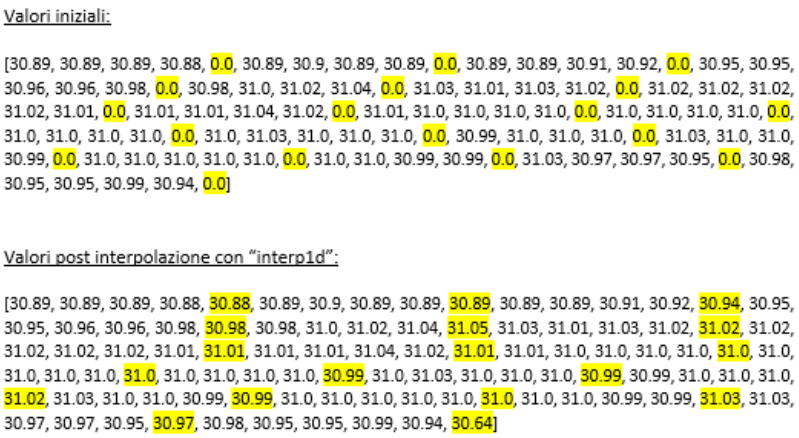
Il codice mostrato in Listing 4.4 mostra l'applicazione del metodo interp1d all'interno del secondo script per l'interpolazione. I parametri maggiormente importanti passati a questo metodo sono:

- x: un array monodimensionale. In questo caso è rappresentato da un array denominato x contente i valori relativi al numero delle posizioni nell'array contenente le 96 medie dei quarti d'ora prodotto nel primo script. I valori di questo array saranno: 0, 1, 2, 3, ..., 95.
- y: un array che può avere più di una dimensione e serve per addestrare la funzione di interpolazione. In questo caso è rappresentato dall'array mean, il quale contiene le medie vere e proprie dei 96 quarti d'ora.
- kind: definisce il tipo di interpolazione. In questo caso è stato scelto il metodo 'cubic' (cubic spline), uno dei più semplici.

Un esempio di interpolazione con l'uso di interp1d su uno dei datastream è mostrato in Figura 4.10. Nell'esempio in figura i valori evidenziati in giallo rappresentano prima i valori mancanti, poi i valori predetti grazie all'interpolazione.

Al termine della fase di interpolazione si è ottenuto un file csv contenente i valori interpolati dei 96 time slot per ogni datastream del cluster europeo. Ogni riga del file, quindi, conterrà l'id progressivo del datastream e i 96 valori numerici. Osservando i risultati scritti su file è stato notato come ci fosse un buon numero di datastream con 96 valori time slot uguali a zero. La spiegazione più ovvia è stata quella in cui le misurazioni ottenute da quel datastream fossero state annotate come zero piuttosto che non inserirle all'interno del channel Thingspeak. La soluzione adottata è stata quella di rimuovere manualmente le righe contenenti questi valori inopportuni. Questa operazione ha portato alla riduzione della dimensione del cluster europeo

che è passato dai 2974 datastream post clustering temporale ai 2828 post operazioni di interpolazione.


Valori iniziali:

[30.89, 30.89, 30.89, 30.88, 0.0, 30.89, 30.9, 30.89, 30.89, 0.0, 30.89, 30.89, 30.91, 30.92, 0.0, 30.95, 30.95, 30.96, 30.96, 30.98, 0.0, 30.98, 31.0, 31.02, 31.04, 0.0, 31.03, 31.01, 31.03, 31.02, 0.0, 31.02, 31.02, 31.02, 31.02, 31.01, 0.0, 31.01, 31.01, 31.04, 31.02, 0.0, 31.01, 31.0, 31.0, 31.0, 31.0, 0.0, 31.0, 31.0, 31.0, 31.0, 0.0, 31.0, 31.0, 31.0, 31.0, 0.0, 31.0, 31.0, 31.0, 31.0, 0.0, 31.0, 31.03, 31.0, 31.0, 31.0, 0.0, 30.99, 31.0, 31.0, 31.0, 0.0, 31.03, 31.0, 31.0, 30.99, 0.0, 31.0, 31.0, 31.0, 31.0, 31.0, 0.0, 31.0, 31.0, 30.99, 30.99, 0.0, 31.03, 30.97, 30.97, 30.95, 0.0, 30.98, 30.95, 30.95, 30.99, 30.94, 0.0]

Valori post interpolazione con "interp1d":

[30.89, 30.89, 30.89, 30.88, 30.88, 30.89, 30.9, 30.89, 30.89, 30.89, 30.89, 30.89, 30.91, 30.92, 30.94, 30.95, 30.95, 30.96, 30.96, 30.98, 30.98, 30.98, 31.0, 31.02, 31.04, 31.05, 31.03, 31.01, 31.03, 31.02, 31.02, 31.02, 31.02, 31.02, 31.02, 31.01, 31.01, 31.01, 31.01, 31.04, 31.02, 31.01, 31.01, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 30.99, 31.0, 31.03, 31.0, 31.0, 31.0, 30.99, 30.99, 31.0, 31.0, 31.0, 31.02, 31.03, 31.0, 31.0, 30.99, 30.99, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 31.0, 30.99, 30.99, 31.03, 31.03, 30.97, 30.97, 30.95, 30.97, 30.98, 30.95, 30.95, 30.99, 30.94, 30.64]

Figura 4.10: Esempio di interpolazione con interp1d

## 4.5 Annotazione

Prima di passare alla successiva fase di lavorazione è stato necessario unire il file contenente i dati interpolati a quello contente i metadati dei datastream ottenuto al termine del clustering temporale. Il file csv risultate, quindi, in ogni riga conterrà prima i metadati del datastream e poi i 96 valori time slot interpolati.

La fase successiva all'interpolazione è stata la fase di annotazione, durante la quale l'obiettivo finale era quello di assegnare manualmente una classe ad ogni datastream del cluster europeo. Assegnare le classi significa andare ad etichettare i datastream come appartenenti a determinate tipologie di dati, categorie o ambiti. I dati che sono stati utilizzati in questo studio sono di tipo ambientale, quindi le classi, ad esempio, saranno: temperatura, umidità, pressione atmosferica, qualità dell'aria, ecc. In questo caso non sono state decise classi a priori in partenza, ma sono state decise direttamente durante lo svolgimento dell'annotazione.

Per implementare l'annotazione manuale delle classi è stato realizzato uno script interattivo (Listing 4.5). In questo script venivano visualizzati, uno alla volta, i datastream all'interno della console (Figura 4.11). I dati di ciascun datastream mostrati in console erano: l'id autoincrementale, il suo nome, il nome del channel di provenienza, la descrizione del channel e i 96 valori interpolati. Al termine della visualizzazione di questi dati nella console si doveva scrivere il numero associato alla classe a cui si voleva assegnare il datastream. Non tutti i datastream sono stati riconducibili a determinate classi, per questo a loro è stato assegnato il valore -1 durante l'esecuzione dello script. Dopo aver inserito il numero della classe e premuto invio, lo script si occupava direttamente di trascrivere tutti i dati del datastream in un nuovo file csv, solamente se la classe assegnata aveva un valore diverso da -1. La classe assegnata è andata a sostituire il numero delle letture in ogni datastream.

```
#vado ad estrarre tutti i dati dal file contenente i datastream
with open(path + 'COMPLETO.csv', 'r', encoding='utf8') as dati:
    for row in dati:
        riga = row.strip().split(';')
        id_ds = int(riga[0])
        metadati = riga[1:9]
        misurazioni = riga[9:len(riga)]
        stream = [id_ds, metadati, misurazioni]
        datastreams.append(stream)

#annotazione di un datastream alla volta
for ds in datastreams:
    #visualizzo in console i dati e chiedo la classe
    da_annotare = '\n' + str(ds[0]) + '\n' + "Nome datastream: "
    + ds[1][0] + '\n' + "Nome channel: " + ds[1][2] + '\n' + "
    Descrizione: " + ds[1][3] + '\n' + "Misurazioni: " + str(ds
    [2]) + '\n' + "Classe: "

    #prendo la classe assegnata in input
    classe = int(input(da_annotare))
    classi_annotate.append(classe)

    #scrittura del datastream annotato nel nuovo file
```

```
21      if classe >= 0:
22          ds[1][7] = str(classe)
23          stampa = str(ds[0]) + ';' + ';'.join(ds[1]) + ';' + ';'.
    join(ds[2])
24          with open(path + 'datastream_annotati.tkse', 'a',
    encoding='utf8') as ff:
25                  ff.write(stampa + '\n')
```

Listing 4.5: Annotazione

```
10421
Nome datastream: Air temperature
Nome channel: Hotpot Adelboden
Descrizione: Adelboden hotpot temperature (CH) .Update every 3 minutes.
Misurazioni: ['18.01', '19.43', '19.61', '20.2', '19.55', '16.98', '16.9', '15.92', '15.32', '16.07'
, '16.47', '16.52', '14.97', '14.75', '15.32', '16.45', '16.41', '14.82', '16.31', '13.39', '12.15',
 '11.33', '10.79', '10.27', '9.84', '9.36', '9.0', '8.61', '8.17', '8.34', '8.39', '8.32', '8.19', '
8.62', '8.85', '8.75', '8.59', '8.43', '8.35', '7.98', '7.83', '7.89', '7.77', '7.6', '7.56', '7.46'
, '7.18', '6.93', '6.73', '6.67', '6.59', '6.42', '6.31', '6.3', '6.22', '6.06', '5.9', '5.81', '5.7
1', '5.49', '5.38', '5.34', '5.24', '4.83', '4.8', '4.5', '4.4', '4.3', '4.3', '4.25', '4.16', '4.1'
, '4.07', '3.87', '3.93', '3.82', '3.83', '3.9', '3.99', '4.1', '4.49', '15.75', '25.8', '29.96', '3
1.42', '31.4', '30.43', '28.92', '27.84', '26.8', '25.91', '25.34', '22.8', '21.13', '20.62', '18.54
']
Classe: 0
```

Figura 4.11: Script annotazione in funzione

Al termine di tutte le operazioni di annotazione sono state individuate 21 classi (Tabella 4.1). Esse sono le seguenti: temperature, humidity, pressure, indoor temperature, outdoor temperature, wind speed, light, air quality, wind direction, wind temperature, Volt, Ampere, wireless/RSSI, heat index, dew point, rain, UV, PM 1, PM 2.5, PM 10 e CO2. Ad ogni classe è stato assegnato un identificatore numerico.

I datastream scartati al termine dell'annotazione sono stati 707, portando quindi il numero totale di datastream rimanenti nel cluster a 2121. Il file csv prodotto dallo script di annotazione è il dataset Thingspeak che questo studio si era prefissato di realizzare.

Dopo aver terminato l'attività di creazione del dataset le fasi di lavorazione successive saranno incentrate sulla sperimentazione su di esso. Le attività volte alla sperimentazione sul dataset Thingspeak verranno descritte all'interno del Capitolo 5.

| Classi Thingspeak | | |
|---|---|---|
| ID classe | Nome classe | # Datastream associati |
| 0 | Temperature | 267 |
| 1 | Humidity | 433 |
| 2 | Pressure | 267 |
| 3 | Indoor temperature | 302 |
| 4 | Outdoor temperature | 337 |
| 5 | Wind Speed | 45 |
| 6 | Light | 44 |
| 7 | Air quality | 12 |
| 8 | Wind direction | 16 |
| 9 | Wind temperature | 4 |
| 10 | Volt | 98 |
| 11 | Ampere | 10 |
| 12 | Wireless/RSSI | 32 |
| 13 | Heat index | 16 |
| 14 | Dew point | 30 |
| 15 | Rain | 12 |
| 16 | UV | 5 |
| 17 | PM 1 | 23 |
| 18 | PM 2.5 | 76 |
| 19 | PM 10 | 74 |
| 20 | CO2 | 18 |
| -1 | Scartato | 707 |

Tabella 4.1: Classi individuate durante l'annotazione

## 4.6 Dataset analoghi: Swissex & Urban Observatory

Come accennato brevemente nella sezione 3.1, il dataset Thingspeak creato in questo studio sperimentale è stato messo a confronto con dataset esterni: Swissex e Urban Observatory. Questi sono dataset creati da due esperimenti

di ricerca nell'ambito del Sensor Metadata Management applicato a ricerche ambientali collaborative. Quindi, questi due dataset esterni trattano dati ambientali come quello creato nelle fasi di lavorazione descritte precedentemente. Una delle differenze principali tra i tre datastream presi in considerazione è la loro dimensione, ovvero il numero di datastream da essi contenuti. Il dataset Thingspeak possiede 2121 datastream (si veda Sezione 4.5), mentre Swissex ne possiede 346 e Urban Observatory 1065.

Il vantaggio di Swissex e Urban Observatory è stato quello di possedere datastream già annotati e, quindi, appartenenti a classi ben definite. Questo viene catalogato come vantaggio perché permette di sapere con precisione che tipo di misurazioni si stanno utilizzando. Nel caso del dataset costruito con i dati provenienti da Thingspeak, invece, si è dovuto lavorare duramente per cercare di capire cosa misurassero certi datastream con informazioni spesso insufficienti o mancanti (nomi di poco aiuto, descrizioni mancanti, ecc.).

Il dataset Swissex possiede undici classi: (0) CO 2, (1) humidity, (2) lysimeter, (3) moisture, (4) pressure, (5) radiation, (6) snow height, (7) temperature, (8) voltage, (9) wind speed, (10) wind direction.

Il dataset Urban Observatory, invece, possiede sedici classi: (0) NO 2, (1) wind direction, (2) humidity, (3) wind speed, (4) temperature, (5) pressure, (6) wind gust, (7) rainfall, (8) soil moisture, (9) avg speed, (10) congestion, (11) traffic flow, (12) journey time, (13) sound, (14) CO, (15) NO.

# Capitolo 5

# Sperimentazione

Dopo la realizzazione del dataset con i dati provenienti da Thingspeak, per raggiungere gli obiettivi principali di questo studio sperimentale è stata necessaria una serie di attività volte ad effettuare diversi esperimenti sul dataset stesso. Gli esperimenti svolti sono necessari per verificare: se il dataset creato nelle fasi di lavorazione precedenti si comporta in maniera simile ai dataset forniti, qual è l'algoritmo di classificazioni maggiormente performante, se la classificazione basata sulla successione dei dati non funziona sul tipo di dataset considerati. Nel seguente capitolo viene descritta l'implementazione delle fasi relative a questa attività di sperimentazione, le quali sono state descritte in breve all'interno del Capitolo 3.

## 5.1 Analisi della varianza

Con la conclusione della fase di annotazione si è ottenuto finalmente il dataset che si voleva realizzare, il quale, d'ora in avanti, verrà chiamato come "dataset Thingspeak" per il resto della trattazione. L'obiettivo fissato post annotazione è il seguente: verificare se il dataset costruito, con l'applicazione di diversi algoritmi di classificazione, si comporta in maniera simile a dataset forniti esternamente. Per applicare i diversi algoritmi di classificazione, però, si necessita di un qualcosa che permetta di descrivere e definire i dati che vengono dati in input a essi. Questo qualcosa sono le feature, le quali sono proprietà misurabili e individuali di un fenomeno osservato. I valori assunti

dalle feature, in questo studio, devono essere espressi in forma numerica. La scelta delle feature è fondamentale per arrivare ad ottenere efficienti algoritmi di classificazione.

### 5.1.1 Feature individuate

Le feature vengono calcolate sui 96 valori interpolati presenti in ogni datastream del dataset Thingspeak. Nei due dataset esterni, Swissex e Urban Observatory, le feature vengono calcolate su un numero di valori maggiore di 96. Le feature individuate e utilizzate in questo studio sono state undici ed esse rappresentano tutte delle statistiche numeriche. Queste unidici feature sono le seguenti:

1. Media aritmetica: assegnati N dati $x_1$, ..., $x_N$ si chiama media aritmetica il valore che si ottiene sommando tutti i dati $x_1$, ..., $x_N$ e poi dividendo per N.

2. Mediana: data una serie di N dati, si definisce mediana come il valore assunto dai dati che si trovano nel mezzo della serie. Si dovrà ordinare la serie di dati. Se N è dispari, la mediana sarà il valore centrale, se N è paro essa sarà la media fra i due valori centrali.

3. Massimo: è il valore più grande all'interno di una serie di N dati.

4. Minimo: è il valore più basso all'interno di una serie di N dati.

5. Deviazione standard: essa indica quanto ogni valore si allontana dalla media aritmetica dei valori.

6. RMS: è la radice quadrata dell'errore quadratico medio (MSE). Essa indica la discrepanza media fra i valori dei dati osservati ed i valori dei dati stimati.

7. Quantile: sono utilizzati in statistica per frazionare in N parti uguali un insieme di dati numerici disposti in ordine progressivo crescente o decrescente.

8. Range interquantile (IQR): è la differenza tra il terzo e il primo quartile, cioè l'ampiezza della area di valori che contiene la metà centrale dei valori osservati.

9. Simmetria: in questo caso è un indice il cui valore cerca di dare una misura alla mancanza di simmetria di una distribuzione numerica.

10. Curtosi: è l'altezza massima raggiunta nella curva di frequenze (la forma del "picco" del grafico) di una distribuzione statistica. La sua rappresentazione grafica permette di comprendere le caratteristiche e la natura del fenomeno statistico e dell'intera distribuzione.

11. Range: è la differenza tra il valore massimo e il valore minimo di una serie di dati.

### 5.1.2 Analisi della varianza per la ricerca delle feature più significative

Dopo aver individuato le feature per i tre dataset da utilizzare negli algoritmi di classificazione, si è cercato di capire quali fossero quelle maggiormente e minormente significative. Per fare ciò si è pensato di effettuare un'analisi della varianza ANOVA per tutte le feature in tutte le classi dei dataset. In questo studio la varianza ANOVA è stata calcolata facendo il seguente rapporto:

$$\frac{varianza\ feature\ classe}{varianza\ feature\ dataset}$$

dove "varianza feature classe" indica la varianza di una feature calcolata su tutti i datastream appartenenti ad una determinata classe; mentre "varianza feature dataset" indica la varianza di una feature calcolata su tutti i datastream del dataset.

Lo studio della varianza è stato implementato tramite uno script (Listing 5.1) che ha operato nel seguente modo:

- Recupero dei dati dai file csv del dataset (Thingspeak, Swissex o Urban Observatory) e calcolo delle unidici feature per ogni datastream. Viene creato un array denominato "features" di lunghezza uguale al numero di datastream del dataset. Ogni elemento dell'array conterrà i valori delle undici feature. Viene creato un altro array denominato "streams" che, oltre ai valori delle feature, conterrà anche la classe a cui appartiene ogni datastream.

- All'array "features" creato al punto precedente viene applicato il metodo per calcolare la varianza lungo le sue colonne. Con questa operazione si ottengono le varianze per ogni feature su tutti i datastream del dataset (denominatore del rapporto evidenziato precedentemente). Questi valori ottenuti sono stati memorizzati all'interno di un array.

- Viene effettuato un ciclo for sul numero di classi totali del dataset e viene calcolata la varianza per ogni feature su tutti i datastream appartenenti ad ogni classe (numeratore del rapporto evidenziato precedentemente). I valori ottenuti saranno memorizzati all'interno di una matrice dove le colonne indicheranno le feature e le righe indicheranno le classi.

- Ogni riga della matrice viene divisa per l'array contenente le varianze per ogni feature su tutto il dataset. Il risultato di questa operazione è una matrice dove ogni elemento è una varianza ANOVA.

- Ad ogni colonna della matrice, quindi per ogni feature, viene calcolata la media di tutte le varianze. Viene così prodotto un array contente le medie delle varianze ANOVA per ogni feature. Questo array verrà poi ordinato in ordine crescente.

```
1 #features = array con i soli valori delle features
2 #streams = array con valori della features e della classe
3 #calcolo della varianza di ogni features per tutto il dataset
4 feat_values = np.array(features)
5 varianza_feat_dataset = np.apply_along_axis(variance, 0,
    feat_values)
```

```
#calcolo la matrice contenente tutte le varianze per ogni
    features su ogni classe
varianza_feat_classi = []
for cl in range(0, NUMERO_CLASSI_TOT):
    ds = []
    for el in streams:
        if el[0] == cl:
            ds.append(el[1])
    var = np.apply_along_axis(variance, 0, np.array(ds))
    varianza_feat_classi.append(var)

#calcolo la matrice contenente tutte le varianze anova
matrice_varianze = []
for a in varianza_feat_classi:
    risultato = a / varianza_feat_dataset
    matrice_varianze.append(risultato)

#calcolo la media per ogni feauture delle relative varianze anova
medie_varianze = np.apply_along_axis(np.mean, 0, matrice_varianze
    )
```

Listing 5.1: Analisi della varianza

L'array ottenuto al termine dello script contiene tutti i risultati dell'analisi della varianza delle undici feature individuate per questo studio. Importante è sottolineare il fatto che più il valore della varianza di una feature è vicino a zero, più questa è significativa. Essendo l'array ordinato in maniera crescente, esso permette di avere le feature disposte dalla più significativa a quella meno significativa. L'ordine risultante delle feature è il seguente: range, valore massimo, deviazione standard, RMS, media, valore minimo, quantile, mediana, curtosi, simmetria, IQR.

L'analisi della varianza è stata realizzata su tutti e tre i dataset, in modo tale da poter avere già un primo confronto tra essi. I risultati di questa fase di lavorazione sono analizzati nel capitolo successivo all'interno Sezione 6.1.

## 5.2 Test con algoritmi di classificazione: selezione dell'algoritmo maggiormente performante

L'analisi della varianza ha permesso di individuare e ordinare le feature con cui descrivere i datastream dei tre dataset. Il passo successivo è stato quello di utilizzare le feature individuate per definire i dati in input degli algoritmi di classificazione. In questa fase vengono, quindi, effettuati test con determinati algoritmi di classificazione e viene selezionato l'algoritmo maggiormente performante/efficiente. I test sono stati effettuati sia sul dataset Thingspeak realizzato precedentemente che sui dataset Swissex e Urban Observatory.

Questa fase di lavorazione ha l'obiettivo di effettuare dei test con cinque diversi algoritmi di classificazione e calcolarne l'accuracy. Essa è una statistica che indica il numero di previsioni corrette dell'algoritmo di classificazione diviso il numero totale di previsioni effettuate. Gli algoritmi di classificazione utilizzati per questa fase di test sono: Random Forest, C4.5 (algoritmo Decision Tree Classifier in Scikit-learn), Support Vector Machine, Naive Bayes e k-Nearest Neighbors (k-NN). Ognuno di questi algoritmi è stato testato aumentando progressivamente il numero di feature prese in considerazione. Per implementare i vari test è stato realizzato uno script che ha svolto le seguenti operazioni:

- Estrazione dei dati del dataset preso in considerazione e calcolo delle undici feature per ogni suo datastream. Tutti i valori ottenuti sono stati memorizzati all'interno di un array. Inoltre, si memorizzano in un array denominato Y tutte le classi cui appartengono i datastream.

- Effettuare un ciclo for sul numero di features in modo tale da aumentare progressivamente ad ogni giro il numero di feature prese in considerazione (Listing 5.2). Così facendo, ad ogni giro si prende dall'array contenente le feature una parte con sempre più valori su ogni riga. Questo partizionamento creerà un nuovo array X, il quale verrà dato, insieme all'array Y, in input al metodo StratifiedShuffleSplit, fornito dalla li-

breria Scikit-learn. Il metodo permette di creare i training e i test set, i quali sono fondamentali per utilizzare gli algoritmi di classificazione. Il training set serve per addestrare l'algoritmo, mentre il test set per fare le previsioni. Dopo aver creato i training e i test set verranno applicati gli algoritmi di classificazione con relativo calcolo dell'accuracy. I risultati verranno memorizzati all'interno di un array.

- Visualizzazione grafica dei valori delle accuracy ottenuti come descritto nel punto precedente. I risultati verranno visualizzati tramite un grafico a barre orizzontali, dove sull'asse X saranno presenti i valori delle accuracy e sull'asse Y il numero di feature considerate (da 1 a 11). Per ogni valore sull'asse Y saranno presenti cinque barre di colore differente, rappresentanti gli algoritmi di classificazione testati.

I risultati grafici dello script appena descritto per i tre dataset considerati sono analizzati nel capitolo successivo all'interno della sezione 6.2. L'algoritmo che ha mostrato maggior efficienza è Random Forest.

```
# CALCOLO ACCURACY CON AUMENTO PROGRESSIVO DELLE FEATURES CONSIDERATE
#ciclo for sul numero di feature da prendere
for i in range(0, NUM_FEATURES_TOT):

    #valori_features = array con le features di tutti i datastream
    X = np.array([f[0:i+1] for f in valori_features])
    Y = np.array([c for c in classi])

    #genero training test e test set
    shuffle_split = StratifiedShuffleSplit(n_splits=1, test_size=0.3, train_size=0.7)
    for train_index, test_index in shuffle_split.split(X, Y):
        X_train, X_test = X[train_index], X[test_index]
        Y_train, Y_test = Y[train_index], Y[test_index]

    #preparazione valori
    sc = StandardScaler()
```

```
17      X_train = sc.fit_transform(X_train)
18      X_test = sc.transform(X_test)
19      #applico gli algoritmi di classificazione
20      accuracyRF = accuracyRandomForest(X_train, X_test, Y_train,
        Y_test)
21      accuracyDT = accuracyDecisionTree(X_train, X_test, Y_train,
        Y_test)
22      accuracyKN = accuracyKNN(X_train, X_test, Y_train, Y_test)
23      accuracySVM = accuracySVC(X_train, X_test, Y_train, Y_test)
24      accuracyNB = accuracyNaiveBayes(X_train, X_test, Y_train,
        Y_test)
25
26      acc_values = [accuracyRF, accuracyDT, accuracyKN, accuracySVM
        , accuracyNB]
27      accuracy_finali.append(acc_values)
```

Listing 5.2: Test di cinque algoritmi di classificazione

## 5.3 Test dell'algoritmo Random Forest su partizionamenti dei datastream

La fase di lavorazione successiva consiste nel testare nuovamente l'algoritmo scelto precedentemente, ovvero Random Forest, sui tre dataset disponibili. L'unica differenza con la fase precedente è che, in questo caso, i valori numerici associati ai datastream sono stati partizionati progressivamente.

Per ogni test con Random Forest sono state calcolate due statistiche: accuracy, come nel task precedente, e F-measure (o F1-score). F-measure è un'altra misura dell'accuratezza di un test di un algoritmo di classificazione. Essa è la media armonica delle misure precision e recall. Precision, per una classe, è il rapporto tra il numero di veri positivi (elementi etichettati correttamente alla classe) e il numero totale di elementi etichettati alla classe. Questo numero totale di elementi etichettati è la somma dei veri positivi e i falsi positivi, i quali sono gli elementi etichettati per errore alla classe. Recall, invece, è il rapporto tra il numero di veri positivi e il numero totale di elementi che appartengono attualmente alla classe. Il numero totale di elementi, in questo caso, è la somma dei vari positivi e i falsi negativi, i quali

sono gli elementi che dovrebbero esser stati etichettati alla classe, ma non lo sono stati.

L'obiettivo di questa attività è stato: osservare se le misurazioni di accuracy e F-measure miglioravano o peggioravano con un partizionamento crescente dei datastream.

Il partizionamento delle misurazioni di ogni datastream è stato effettuato fino a otto volte, le quali sono state indicate tramite un valore K (K = 1, ..., 8). Su ogni partizione, poi, sono state misurate le undici feature, aumentando così progressivamente il numero di indicatori per l'input dell'algoritmo di classificazione Random Forest. Le varie fasi di partizionamento sono riassunte qua di seguito:

- K = 1: vengono prese tutte le misurazioni di un datastream, non ci sono divisioni. Sulle misurazioni vengono calcolate le feature, che saranno 11 in totale.

- K = 2: le misurazioni di un datastream vengono divise in due parti distinte. Su ciascuna partizione vengono misurate le undici feature, che saranno 22 in totale.

- K = 3: le misurazioni di un datastream vengono divise in tre parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 33 in totale.

- K = 4: le misurazioni di un datastream vengono divise in quattro parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 44 in totale.

- K = 5: le misurazioni di un datastream vengono divise in cinque parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 55 in totale.

- K = 6: le misurazioni di un datastream vengono divise in sei parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 66 in totale.

- K = 7: le misurazioni di un datastream vengono divise in sette parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 77 in totale.
- K = 8: le misurazioni di un datastream vengono divise in otto parti distinte. Su ogni partizione vengono misurate le undici feature, che saranno 88 in totale.

Per effettuare i test di Random Forest con datastream partizionati sono stati realizzati due script, uno che calcoli l'accuracy e uno F-measure, simili a quello mostrato in Listing 5.2, ma con 3 differenze principali:

1. Nelle operazioni di recupero dei dati dal dataset viene effettuato il partizionamento discusso precedentemente (mostrato in Listing 5.3).
2. Le misure dell'accuracy/F-measure vengono effettuate 10 volte e il valore ottenuto è la media di quelle misurazioni.
3. Il risultato grafico finale viene visualizzato tramite un grafico 3D, realizzato grazie al metodo surface_plot fornito da Axes3D della libreria Matplotlib.

```
#ottengo classe e misurazioni dai datastream del file csv
riga = row.strip().split(';')
classe = int(riga[8])
classi.append(classe)
valori = np.array(riga[9:]).astype(np.float)

#for sui valori di K - da 1 a 8
for k in range(K_MIN,K_MAX+1):
    values = partition(valori, k) #partizionamento
    for val in values:
        # per ogni partizione calcolo le varie features e
    inserisco ognuna di esse nei relativi array
        media = np.mean(val)
        medie.append(media)
        mediana = np.median(val)
        mediane.append(mediana)
        maxim = np.max(val)
```

```
17          massimi.append(maxim)
18          minim = np.min(val)
19          minimi.append(minim)
20          std_dev = np.std(val)
21          deviazioni.append(std_dev)
22          rms = np.sqrt(np.mean(np.square(val)))
23          rms_multipli.append(rms)
24          quantile = np.quantile(val, 0.4)
25          quantili.append(quantile)
26          i_q_r = iqr(val)
27          range_interquantili.append(i_q_r)
28          simmetria = skew(val)
29          simmetrie.append(simmetria)
30          curtosi = kurtosis(val)
31          curtosi_multiple.append(curtosi)
32          rang = maxim - minim
33          range_multipli.append(rang)
34
35  # creo gli array finali necessari per ogni K
36  array_k = []
37  for x in range(K_MIN,K_MAX+1):
38      arr = np.array([])
39      lun = sommaPrecedenti(x) #per capire quanti valori di una
        feature devo prendere per il valore K dato
40      arr = range_multipli[0:lun] + massimi[0:lun] + deviazioni[0:
        lun] + rms_multipli[0:lun] + medie[0:lun] + minimi[0:lun] +
        quantili[0:lun] + mediane[0:lun] + curtosi_multiple[0:lun] +
        simmetrie[0:lun] + range_interquantili[0:lun]
41      array_k.append(arr)
```

Listing 5.3: Partizionamento dei datastream

I due script realizzati per calcolare accuracy e F-measure sono stati utilizzati per tutti e tre i dataset presi in considerazione. Questo ha prodotto per ognuno di essi due grafici tridimensionali: uno con i valori dell'accuracy e uno con quelli delle F-measure. I sei grafici ottenuti al termine di questa fase sono mostrati all'interno del capitolo dedicato ai risultati di questo studio sperimentale (Capitolo 6).

# Capitolo 6

# Risultati

In questo capitolo verranno osservati e analizzati i risultati delle fasi finali dello studio sperimentale descritto in questa tesi. Le fasi considerate sono in ordine: analisi della varianza (Sezione 5.1), test dei cinque algoritmi di classificazione (Sezione 5.2) e test dell'algoritmo Random Forest con il partizionamento dei datastream (Sezione 5.3). Dopo aver analizzato i risultati di queste fasi, verranno tratti i risultati e le conclusioni finali riguardanti l'intero studio sperimentale e i suoi obiettivi principali.

## 6.1 Risultati dell'analisi della varianza

La prima fase di cui sono stati analizzati i risultati è quella che ha interessato l'analisi della varianza. Questa fase è stata necessaria per individuare l'importanza delle undici feature scelte nei tre dataset utilizzati. Lo script realizzato ha permesso di calcolare tutte le varianze delle varie feature e di poterle ordinare dalla più alla meno significativa. Importante è sottolineare il fatto che più il valore della varianza di una feature è vicino a zero, più questa è significativa.

L'analisi della varianza è stata svolta su tutti e tre i dataset, in modo tale da poter avere già un primo confronto tra essi. Solo con il dataset Thingspeak le feature sono state ordinate dalla più alla meno significativa. Le feature di Swissex e Urban Observatory, invece, saranno ordinate nello stesso ordine di quelle di Thingspeak, questo per facilitarne il confronto. L'ordine

risultante delle feature è quello descritto al termine della sezione 5.1.2, ovvero: range, valore massimo, deviazione standard, RMS, media, valore minimo, quantile, mediana, curtosi, simmetria, IQR. I risultati dello script sono visibili in Tabella 6.1 e in Tabella 6.2.

| Analisi della varianza | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Range | Max | Dev.std. | RMS | Media | Min |
| Thingspeak | 0.233256413 | 0.233256414 | 0.23325643 | 0.2332565 | 0.2332568 | 0.58764 |
| Swissex | 0.73299 | 0.46830 | 0.69599 | 0.41504 | 0.43417 | 0.49688 |
| Urban Obs. | 0.26057 | 0.20234 | 0.28390 | 0.05614 | 0.04928 | 0.08174 |

Tabella 6.1: Risultati analisi della varianza pt.1

| Analisi della varianza | | | | | |
|---|---|---|---|---|---|
| Dataset | Quantile | Mediana | Curtosi | Simmetria | IQR |
| Thingspeak | 0.60252 | 0.68351 | 0.69973 | 0.82878 | 2.06261 |
| Swissex | 0.42309 | 0.42843 | 1.03406 | 1.12800 | 0.80336 |
| Urban Obs. | 0.04840 | 0.05192 | 1.65371 | 1.06806 | 0.44493 |

Tabella 6.2: Risultati analisi della varianza pt.2

Come si può vedere dalle due tabelle le feature maggiormente problematiche, cioè con valori superiori a uno, sono: IQR per il dataset Thingspeak, simmetria e curtosi per Swissex e Urban Observatory. Si è notato, inoltre, come la maggior parte delle feature del dataset Thingspeak assumano valori vicini allo zero come nei dataset esterni. Questi risultati hanno dimostrato come il dataset realizzato sia, per certi versi, simile ai due forniti da fonti esterne, visto che le varianze delle feature possiedono valori in linea tra di loro. Tutto questo ha permesso di validare il dataset Thingspeak. Per osservare meglio i dati presentati in Tabella 6.1 e Tabella 6.2, è stato realizzato un istogramma con i valori contenuti in esse. L'osservazione attenta dell'istogramma mostrato in Figura 6.1 conferma il risultato finale di questa fase di analisi della varianza.

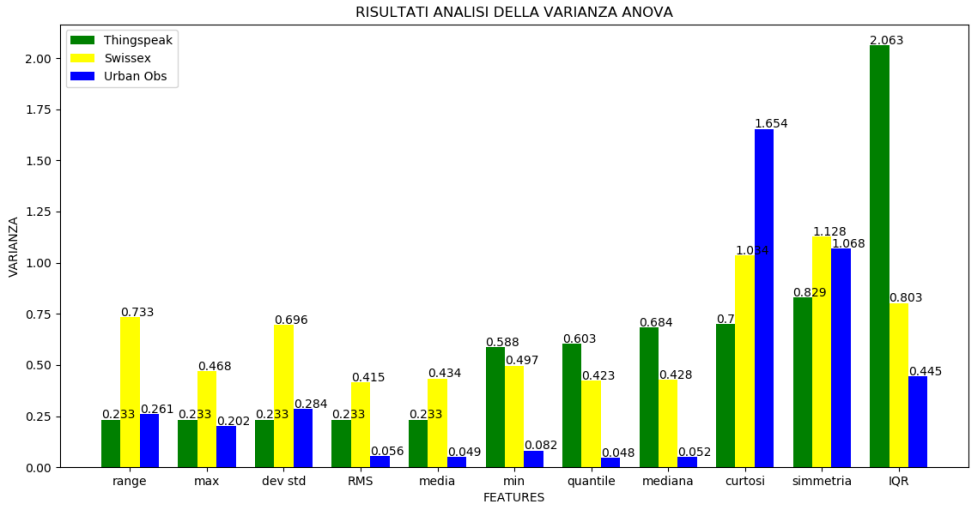
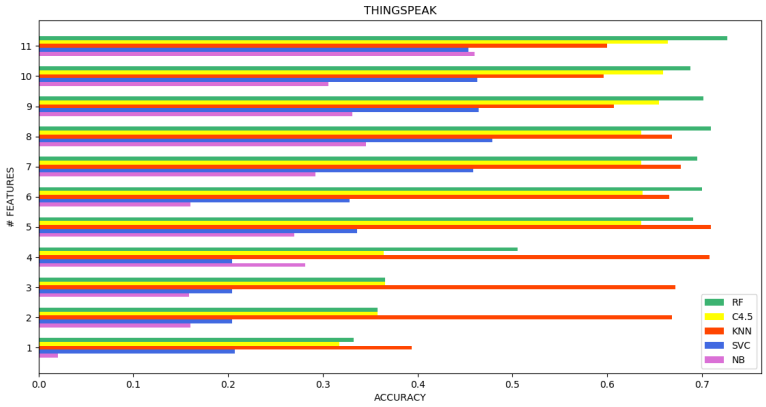



Figura 6.1: Istogramma delle varianze

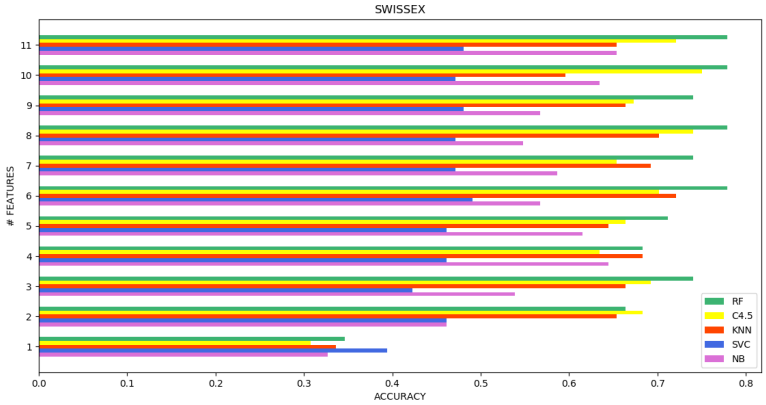
## 6.2 Risultati dei test degli algoritmi di classificazione

La seconda fase di lavoro di cui sono stati analizzati i risultati è quella dove sono stati testati cinque diversi algoritmi di classificazione su ciascun dataset considerato. Ognuno dei cinque algoritmi è stato testato aumentando progressivamente il numero di feature prese in considerazione e calcolandone l'accuracy. Gli algoritmi di classificazione utilizzati in questa fase sono: Random Forest, Naive Bayes, Support Vector Machine, C4.5 (algoritmo DecisionTreeClassifier in Scikit-learn) e k-Nearest Neighbors.

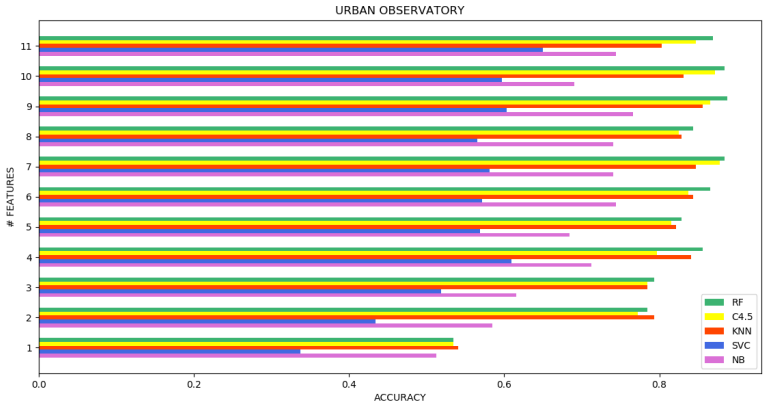
I risultati prodotti dagli algoritmi di classificazione sono mostrati nei tre grafici in Figura 6.2. Ogni grafico a barre orizzontali rappresenta i risultati per un determinato dataset. Come si può osservare in tutti e tre i grafici, l'algoritmo maggiormente performante tra i cinque è Random Forest (barra color verde scuro nei grafici). Quindi, il risultato tratto da questa fase di lavoro è che l'algoritmo Random Forest è quello che funziona nel modo migliore con i dataset Thingspeak, Swissex e Urban Observatory. Per questo esso è stato utilizzato nell'ultima fase di lavorazione e, quindi, nella prossima sezione di analisi dei risultati.




(a) Dataset Thingspeak




(b) Dataset Swissex




(c) Dataset Urban Observatory

Figura 6.2: Accuracy dei cinque algoritmi di classificazione

## 6.3 Risultati di Random Forest con partizionamento dei datastream

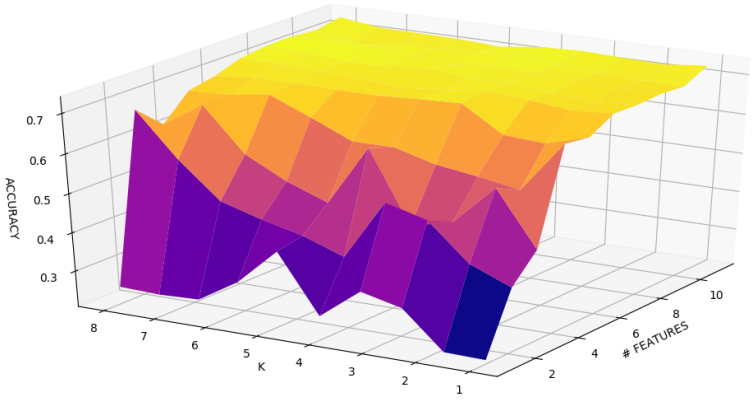
La terza fase di lavoro di cui sono stati analizzati i risultati è quella dove sono stati effettuati test dell'algoritmo Random Forest (selezionato nella fase precedente) su partizionamenti crescenti dei valori numerici associati ai datastream dei tre dataset considerati. Dopo ogni test effettuato con l'algoritmo di classificazione sono state misurate due statistiche: accuracy e F-measure. L'obiettivo di questa fase è stato quello di verificare se le misurazioni di accuracy e F-measure miglioravano o peggioravano all'aumentare del partizionamento dei datasteam. I due script realizzati per effettuare i test con Random Forest (Sezione 5.3) hanno prodotto due grafici tridimensionali per ognuno dei tre dataset presi in considerazione. Nel grafico 3D riguardante l'accuracy sono rappresentate le seguenti misure:

- K: rappresenta il valore che indica il numero di partizioni in cui sono stati suddivisi i dati dei datastream. È rappresentato sull'asse X.
- Numero Feature (# Features): rappresenta il numero delle feature considerate per i dati in input all'algoritmo di classificazione. È rappresentato sull'asse Y.
- Accuracy: rappresenta i valori di accuracy ottenuti dai test dell'algoritmo di classificazione. È rappresentato sull'asse Z.

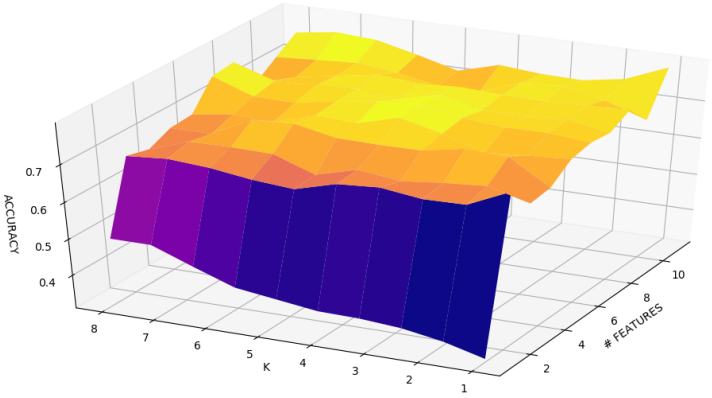
Nel grafico 3D riguardante la F-measure, invece, sono rappresentate le seguenti misure:

- K: come nel grafico dell'accuracy. È rappresentato sull'asse X.
- Numero Feature (# Features): come nel grafico dell'accuracy. È rappresentato sull'asse Y.
- F-measure: rappresenta i valori di F-measure ottenuti dai test dell'algoritmo di classificazione. È rappresentato sull'asse Z.

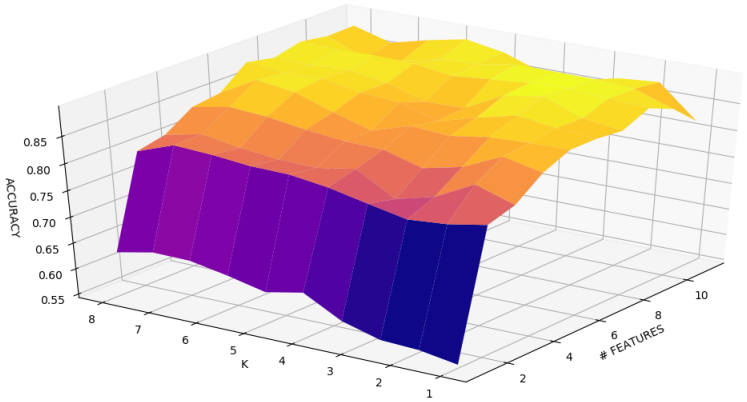
I grafici tridimensionali in totale sono sei: tre per accuracy e tre per F-measure. Essi sono rappresentati in Figura 6.3 e Figura 6.4.




(a) Dataset Thingspeak

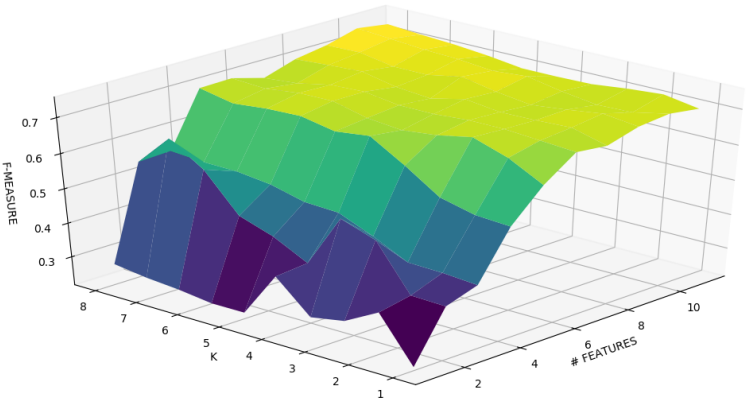



(b) Dataset Swissex

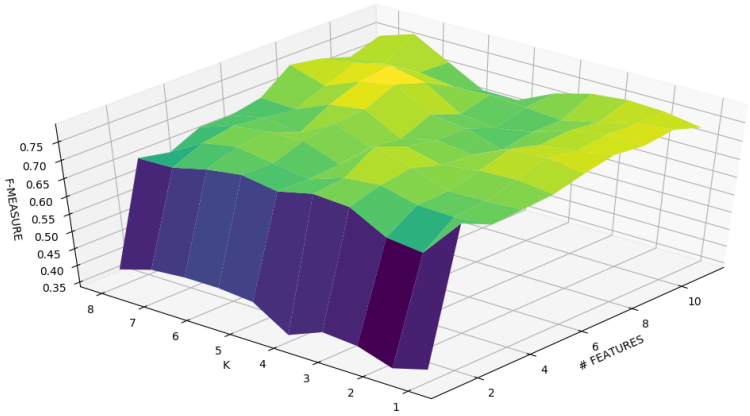



(c) Dataset Urban Observatory

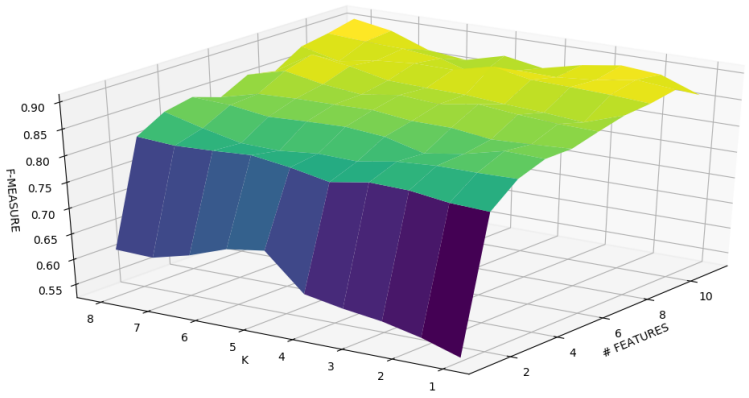
Figura 6.3: Grafici 3D - Accuracy




(a) Dataset Thingspeak




(b) Dataset Swissex




(c) Dataset Urban Observatory

Figura 6.4: Grafici 3D - F-measure

Analizzando i grafici inerenti al dataset Thingspeak creato in questo studio si è notato come sia l'accuracy che la F-measure assumano valori altalenanti all'aumentare del numero delle feature considerate. I valori delle due statistiche, comunque, si allineano dopo che vengono considerate almeno 4/5 feature. Osservando i grafici dal punto di vista del valore K si nota come nelle prime fasi con poche feature i risultati siano molto altalenanti come indicato precedentemente. Anche in questo caso si nota come le variazioni di K non migliorino di molto le due statistiche quando vengono superate le 4/5 feature considerate.

Analizzando i grafici inerenti ai dataset Swissex e Urban Observatory si è notato immediatamente come le variazioni di K non producano miglioramenti significativa per quanto riguarda l'accuracy e la F-measure. Dal punto di vista delle feature, invece, si è notato come dopo un paio di feature considerate i valori delle due statistiche si stabilizzino con poche variazioni minime. Infine, si è osservato che Urban Obervatory sia il dataset esterno migliore tra quelli considerati sia a livello di accuracy che di F-measure.

Il risultato finale che può essere tratto da questi grafici tridimensionali è il seguente: per i dataset considerati la successione dei dati conta fino ad un certo punto. Ciò è verificato dal fatto che con Random Forest:

- Se si aumenta il valore K, cioè il numero delle partizioni, i risultati dell'algoritmo non migliorano di molto o non migliorano proprio.
- Se si aumenta, invece, il numero delle feature, si ottengono dei miglioramenti nei risultati dell'algoritmo.

# Capitolo 7

# Conclusioni e possibili sviluppi

## 7.1 Risultati finali

Prima di trarre le conclusioni finali è necessario ricapitolare i risultati maggiormente rilevanti ottenuti durante le varie fase di lavoro di questo studio sperimentale:

1. Il dataset realizzato (Thingspeak) è, per certi versi, simile ai due forniti da fonti esterne, visto che le varianze delle feature possiedono valori in linea tra loro.

2. Per i dataset considerati l'algoritmo di classificazione Random Forest è quello che funziona meglio tra tutti quelli testati. Esso risulta essere l'algoritmo con la maggior efficienza visti i suoi valori di accuracy per i vari test effettuati con le feature considerate progressivamente.

3. Per i dataset considerati la successione dei dati conta fino ad un certo punto visto che all'aumentare di K i risultati non migliorano di molto o non migliorano proprio; mentre all'aumentare del numero di feature si ottengono miglioramenti. Questo conferma l'ipotesi in [18], dove è stato affermato che la classificazione basata sulla successione dei dati (time series) non funzioni sul tipo di dataset considerati.

## 7.2 Conclusioni finali

Con i risultati ottenuti e descritti nel capitolo precedente si è dimostrato come sia possibile realizzare un dataset, con relative classi annotate, anche con dati provenienti da una unreliable repository pubblica come Thingspeak. Si è dimostrata, inoltre, l'esistenza della possibilità di creare un dataset che, con l'applicazione di certi algoritmi di classificazione, si comporti in maniera simile a dataset provenienti da reliable repository, quindi forniti da enti, organizzazioni o studi/esperimenti. Questo implica che l'obiettivo da raggiungere al termine di questo studio sperimentale è stato raggiunto. Tutti questi risultati fanno intendere come il potenziale degli Open Data è ancora tutto da scoprire e pieno di possibilità.

Per arrivare ad un comportamento quasi identico la strada è ancora lunga e tutta in salita visto che, in letteratura, non ci sono ancora molti casi di studio. La stessa cosa vale anche per gli studi relativi al problema della classificazione dei datastream con IoT Open Data.

## 7.3 Possibili sviluppi e migliorie future

Per migliorare i risultati ottenuti al termine di questa tesi si potrebbero migliorare, con studi più approfonditi, gli esperimenti svolti dopo la realizzazione del dataset con i dati provenienti da Thingspeak. Un'analisi della varianza delle feature maggiormente approfondita insieme a nuove feature potrebbe portare benefici nei successivi test degli algoritmi di classificazione. Si potrebbero considerare anche nuovi algoritmi, sia forniti da librerie o realizzati personalmente. Inoltre, si potrebbe cercare di individuare un numero maggiore o minore di classi o, addirittura, di essere molto più intransigenti nell'assegnazione di esse ai datastream che non forniscono informazioni sufficienti. Infine, si potrebbe considerare anche l'opzione di utilizzare altri unreliable repository pubblici diversi da Thingspeak.

Per quanto riguarda gli sviluppi futuri relativi a questo studio realizzato, sono state pensate diverse possibilità: (1) realizzazione di un articolo scientifico in seguito all'approfondimento della fase di sperimentazione sul dataset

costruito; (2) realizzazione di una piattaforma web contenente il dataset realizzato, alla quale è possibile fare richieste per ottenere informazioni elaborate per confrontarle con altri studi sperimentali futuri.

# Bibliografia


[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2009.

[2] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[4] João Borges Neto, Thiago Silva, Renato Assunção, Raquel Mini, and Antonio Loureiro. Sensing in the collaborative internet of things. *Sensors*, 15(3):6607–6632, 2015.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Jean-Paul Calbimonte, Oscar Corcho, Zhixian Yan, H Jeung, and Karl Aberer. Deriving semantic sensor metadata from raw measurements. 2012.

[7] Mauro Conti, Ali Dehghantanha, Katrin Franke, and Steve Watson. Internet of things security and forensics: Challenges and opportunities, 2018.

[8] Nilanjan Dey, Aboul Ella Hassanien, Chintan Bhatt, Amira Ashour, and Suresh Chandra Satapathy. *Internet of things and big data analytics toward next-generation intelligence*. Springer, 2018.

[9] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[10] Farshad Firouzi, Amir M Rahmani, Kunal Mankodiya, Mustafa Badaroglu, Geoff V Merrett, P Wong, and Bahar Farahani. Internet-of-things and big data for smarter healthcare: from device to architecture, applications and analytics, 2018.

[11] Cisco Visual Networking Index. Forecast and methodology, 2016–2021. *White Paper, June*, 2017.

[12] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[13] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, and Amit P Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175, 2018.

[14] Marcello A Gómez Maureira, Daan Oldenhof, and Livia Teernstra. Thingspeak–an api and web service for the internet of things. *World Wide Web*, 2011.

[15] Federico Montori. *Delivering IoT Services in Smart Cities and Environmental Monitoring through Collective Awareness, Mobile Crowdsensing and Open Data.* PhD thesis, alma, Aprile 2019.

[16] Federico Montori, Luca Bedogni, and Luciano Bononi. On the integration of heterogeneous data sources for the collaborative internet of things. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE, 2016.

[17] Federico Montori, Luca Bedogni, and Luciano Bononi. A collaborative internet of things architecture for smart cities and environmental monitoring. *IEEE Internet of Things Journal*, 5(2):592–605, 2017.

[18] Federico Montori, Kewen Liao, Prem Prakash Jayaraman, Luciano Bononi, Timos Sellis, and Dimitrios Georgakopoulos. Classification and annotation of open internet of things datastreams. In *International*

*Conference on Web Information Systems Engineering*, pages 209–224. Springer, 2018.

[19] Charith Perera, Chi Harold Liu, Srimal Jayawardena, and Min Chen. A survey on internet of things from industrial market perspective. *IEEE Access*, 2:1660–1679, 2014.

[20] Partha Pratim Ray. A survey of iot cloud platforms. *Future Computing and Informatics Journal*, 1(1-2):35–46, 2016.

[21] Partha Pratim Ray. A survey on internet of things architectures. *Journal of King Saud University-Computer and Information Sciences*, 30(3):291–319, 2018.

[22] Eugene Siow, Thanassis Tiropanis, Xin Wang, and Wendy Hall. Tritandb: time-series rapid internet of things analytics. *arXiv preprint arXiv:1801.07947*, 2018.

[23] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[24] Krzysztof Witkowski. Internet of things, big data, industry 4.0–innovative solutions in logistics and supply chains management. *Procedia Engineering*, 182:763–769, 2017.

# Ringraziamenti

Dopo quattro mesi infiniti, finalmente il grande giorno è arrivato: scrivere queste poche frasi di ringraziamento è il mio tocco finale e personale a questa tesi. È stato un periodo di profondo apprendimento a livello scientifico, ma anche personale. Lavorare a questa tesi ha avuto un forte impatto sulla mia personalità: non riesco più a procrastinare e, questa, era la cosa che sapevo far meglio (poche certezze ma buone). Adesso vorrei spendere qualche parola di ringraziamento nei confronti di tutte le persone che mi hanno aiutato e sostenuto durante questi mesi.

Desidero innanzitutto ringraziare particolarmente il Dott. Federico Montori, correlatore di questa tesi, per il tempo dedicatomi e la pazienza avuta nei miei confronti anche quando ho fatto disastri con il codice capendo in maniera errata le sue direttive.
Un grande ringraziamento va al Prof. Marco Di Felice, relatore di questa tesi, per la cortesia e la pazienza che ha avuto nei miei confronti nella scelta dell'argomento da sviluppare all'interno di questa tesi.

Un doveroso ringraziamento va ai miei compagni di corso, in particolare, Miriana, Alice e Gabriel, con cui questi anni di studio sono stati un'avventura. Ne sono successe di tutti i colori, soprattutto durante i tanto amati progetti.

Un importantissimo ringraziamento va a tutti i miei amici che hanno sopportato (o forse no) ogni mia lamentela durante questi ultimi mesi/anni. Non volendo dimenticarmi di qualcuno ho deciso di ringraziarli in un modo molto politically correct, assegnando un personaggio di una serie tv qualsiasi ad ognuno di loro, così non potranno sapere a chi mi riferisco. Detto questo,

ringrazio: Joey Tribbiani, Chandler Bing, Ross Geller, Monica Geller, Phoebe Buffay, Rachel Green, Barney Stinson, Ted Mosby, Lily Aldrin, Marshall Eriksen, Robin Scherbatsky, Tracy McConnell, Sheldon Cooper, Jessica Day, Nick Miller, Winston Bishop, Cersei Lannister, Arya Stark, Ned Stark, Jon Snow, Tyrion Lannister, Annalise Keating, Thomas Shelby, Polly Gray, Jemma Simmons, Leopold Fitz e tanti altri.

Il ringraziamento più importante va ai miei genitori e ai miei nonni, Renzo e MariaGrazia, che sono stati e saranno sempre i miei punti di riferimento e il mio sostegno. Non servono tante parole, grazie.

*Bologna, 16/07/2019*

*Mattia Maniezzo*