

UNIVERSITÀ DEGLI STUDI DI VERONA

DIPARTIMENTO DI INFORMATICA

LAUREA TRIENNALE - INFORMATICA

---

# Sviluppo di una libreria Python per la creazione e gestione di sensori virtuali

---

*Autore:*

Alessio BONIZZATO

*Matricola:*

VR446437

*Relatore:*

Davide QUAGLIA

*Co-Relatore:*

Elia BRENTAROLLI

Anno accademico 2022-2023

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
<b>2</b>	<b>Installazione ed esecuzione</b>	<b>2</b>
<b>3</b>	<b>Guida all'Utilizzo</b>	<b>3</b>
3.1	Creazione dei Modelli . . . . .	3
<b>4</b>	<b>Struttura del codice</b>	<b>6</b>
4.1	Gestione modelli . . . . .	7
4.2	Struttura Flask . . . . .	7
4.3	Gestione Utenti . . . . .	8
4.4	Strumenti di Creazione dei Modelli . . . . .	8
4.5	Iterazione col Broker . . . . .	8
<b>5</b>	<b>Dettagli implementativi</b>	<b>9</b>
5.1	Formato dei File CSV . . . . .	9
5.2	Manipolazione dati . . . . .	9
5.3	Parametro window . . . . .	10
5.4	Matrice MxDxH . . . . .	11
5.5	Missing values . . . . .	12
5.6	Flask . . . . .	12
5.7	Maschera dei valori . . . . .	12
5.8	Formato dei modelli . . . . .	13
5.9	MQTT . . . . .	15
<b>6</b>	<b>Funzionamento</b>	<b>16</b>
6.1	Registrazione ed autenticazione . . . . .	16
6.2	Iterazione Server e Creazione dei modelli . . . . .	16
6.3	Calcolo del modello . . . . .	16
6.3.1	Stepwise-regression . . . . .	17
6.4	Salvataggio dei Modelli . . . . .	17
6.5	Configurazione Broker . . . . .	17

# Elenco delle figure

3.1	Pagina di Registrazione . . . . .	3
3.2	Pagina di Login . . . . .	3
3.3	Pagina di creazione dei modelli . . . . .	4
3.4	Risultati calcolo del modello . . . . .	4
3.5	Pagina di configurazione del broker . . . . .	5
5.1	Gestione richieste HTTP di più utenti . . . . .	12

# Elenco delle tabelle

5.1	File csv di input . . . . .	10
5.2	File csv di output . . . . .	10
5.3	File di input dopo la funzione alligned . . . . .	10
5.4	File csv di output dopo la funzione alligned . . . . .	10
5.5	window impostato a 0 . . . . .	11
5.6	window impostato ad 1 . . . . .	11
5.7	window impostato a 2 . . . . .	11
5.8	MxDxH di 2023-08-10 15:30:00 . . . . .	11

# Capitolo 1

## Introduzione

Questa libreria è stata sviluppata con l'obiettivo di agevolare il processo decisionale basato sull'analisi dei dati attraverso l'utilizzo di modelli di regressione lineare. Il suo scopo principale consiste nel fornire agli utenti uno strumento efficiente ed intuitivo, consentendo loro di generare modelli di regressione lineare tramite l'interazione con una interfaccia web dedicata. Gli utenti possono fornire specifici parametri di input, dai quali la libreria trae origine per la creazione dei modelli. La libreria, offre uno strumento per condurre analisi predittive di alta precisione. Grazie a essa, è possibile migliorare notevolmente la capacità di prendere decisioni informate in diversi contesti professionali.

Questo documento è suddiviso come segue: nel capitolo 2 è esposto il procedimento di importazione della libreria e l'installazione dei prerequisiti necessari per il corretto funzionamento della stessa. Nel capitolo 3, si fornisce una guida sull'uso della libreria, includendo una breve descrizione di ogni campo di input visualizzato nella pagina di interazione con l'utente. Nel capitolo 4, viene esaminata l'organizzazione strutturale della libreria, con una breve illustrazione del loro contenuto di ogni file. Nel capitolo 5, vengono esplicate le scelte progettuali effettuate e il significato dei parametri utilizzati. Infine, nel capitolo 6, si offre un'analisi approfondita delle interazioni tra le diverse componenti del codice.

# Capitolo 2

## Installazione ed esecuzione

Per installare la libreria, è necessario seguire attentamente i passaggi riportati di seguito:

### Installazione

1. Scaricare l'intera repository dal repository GitHub ufficiale della libreria. È possibile scaricare l'archivio ZIP direttamente dal sito oppure su Linux tramite il comando `git clone`.
2. Spostarsi da terminale all'interno della directory scaricata. Su Linux è possibile farlo col comando `cd models_creator`
3. Prima di procedere con l'installazione, è fondamentale verificare di avere installato sul proprio sistema il gestore dei pacchetti `pip`, che verrà utilizzato per installare le dipendenze della libreria
4. Nel caso in cui si stia utilizzando un sistema operativo Linux, è necessario rendere il file `install.sh` eseguibile. Questo può essere fatto tramite il seguente comando da terminale: `chmod +x install.sh`
5. Eseguire il seguente comando da terminale: `./install.sh`. L'uso di questo script semplifica il processo di installazione, in quanto gestirà automaticamente l'installazione di tutte le dipendenze necessarie appositamente indicate nel file `requirements.txt`

### Esecuzione

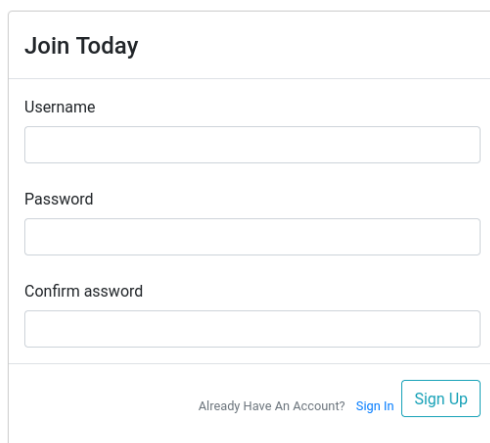
Per eseguire il server è necessario aprire da terminale la directory della libreria ed eseguire il seguente comando: `python3 run.py`.

# Capitolo 3

## Guida all'Utilizzo

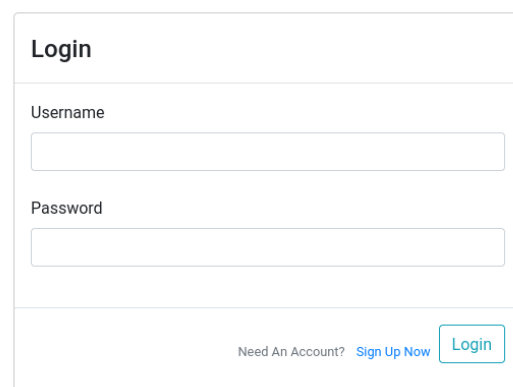
### 3.1 Creazione dei Modelli

1. **Accedi al Server:** Utilizza il tuo browser per accedere alla pagina di creazione dei modelli della libreria, collegandoti utilizzando l'indirizzo IP del server.
2. **Login:** Per poter utilizzare la libreria è necessario effettuare l'autenticazione, fornendo username e password. In caso non si sia già in possesso di un account sarà necessario crearne uno attraverso l'apposita pagina di registrazione.



The registration form, titled "Join Today", contains three input fields: "Username", "Password", and "Confirm password". At the bottom right, there is a "Sign Up" button. At the bottom left, there is a link "Already Have An Account? Sign In".

Figura 3.1: Pagina di Registrazione



The login form, titled "Login", contains two input fields: "Username" and "Password". At the bottom right, there is a "Login" button. At the bottom left, there is a link "Need An Account? Sign Up Now".

Figura 3.2: Pagina di Login

3. **Fornitura dei Parametri:** Per creare un modello, è necessario fornire i seguenti parametri tramite la pagina web del server:
  - Scegliere tra le opzioni disponibili o introdurre un nuovo nominativo per un sensore, al quale verranno connessi i modelli oggetto della creazione imminente.
  - Carica il file di input contenente i dati storici relativi al sensore.
  - Carica uno o più file di output che rappresentino la variabile di risposta da prevedere.

- Imposta il valore 'window', espresso come numero intero non negativo, che stabilirà quanti istanti temporali precedenti verranno considerati durante il processo di creazione del modello.
- Assegna un nome al modello. Se stai caricando più file di output, il nome verrà affiancato da un numero intero progressivo.
- Scegli se eseguire il test del modello. In caso affermativo, i dati verranno divisi in addestramento (80%) e test (20%) per la verifica della precisione.

## Models Calculator

The screenshot shows the 'Models Calculator' web interface. At the top is a 'Broker Config' button. Below it is a 'Model Directory' label followed by a text input field containing 'Enter or select an existing f'. Underneath is an 'Input:' label followed by a file selection area with a 'Choose file' button and a 'Browse' button. Below that is an 'Output:' label followed by a file selection area with a 'Choose files' button and a 'Browse' button. Further down are two input fields: 'Window:' with 'Enter window value' and 'Model Name:' with 'Enter the name of model'. Below these is a 'Test:' label followed by an unchecked checkbox. At the bottom are three buttons: a blue 'Run Script' button, a green 'Save' button, and a red 'Delete' button. Below the buttons is a blue 'Logout' link.

Figura 3.3: Pagina di creazione dei modelli

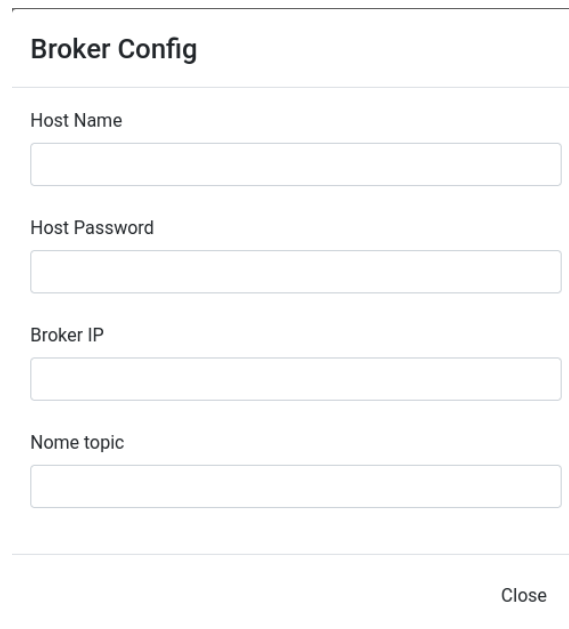
4. **Visualizzazione dei Risultati:** Una volta creato il modello, la libreria fornirà indicatori che valutano l'adeguatezza del modello rispetto ai dati forniti. Nel caso in cui l'utente abbia scelto di effettuare il test, verranno impiegati dei dati per valutare l'effettiva bontà delle previsioni, generando ulteriori indicatori.

The screenshot shows the results of a model calculation. At the top is a blue 'Run Script' button. Below it is the text 'Missing values in:' followed by '[2m', '3m]'. Underneath are three lines of text: 'AdjR2 del modello: 0.9641', 'BMSEadj del modello: 0.0685', and 'RRSE del modello: 0.1889'. At the bottom are three buttons: a green 'Save' button, a red 'Delete' button, and a blue 'Logout' link.

Figura 3.4: Risultati calcolo del modello



5. **Decisione di Salvare o Scartare:** A questo punto, si ha la possibilità di prendere una decisione. È possibile scegliere di salvare il modello nella directory del sensore precedentemente indicata o scartarlo completamente.
6. **Salvataggio con Dati Broker (opzionale):** Nel caso si decida di salvare il modello, sarà possibile creare un file di configurazione per il broker che verrà automaticamente salvato nella directory del sensore. I parametri di configurazione del broker sono i seguenti:
  - **Name:** Nome utente da utilizzare per connettersi al broker
  - **Password:** Password associata al nome utente
  - **IP del broker:** indirizzo IP che identifica il broker al quale connettersi
  - **Topic:** Topic sul quale pubblicare le predizioni utilizzando i modelli del sensore in cui è contenuto il file di configurazione



The image shows a web form titled "Broker Config". It contains four input fields: "Host Name", "Host Password", "Broker IP", and "Nome topic". Each field is a simple text box with a light gray border. At the bottom right of the form, there is a "Close" button. The form is enclosed in a thin gray border.

Figura 3.5: Pagina di configurazione del broker

7. **Logout** Sarà possibile effettuare il logout il quale consentirà all'utente di disconnettersi dal proprio account attuale. L'utente potrà quindi accedere con un altro account o eseguire nuovamente il login se lo desidera.

# Capitolo 4

## Struttura del codice

La struttura della repository si presenta nel seguente modo:

```
/ (models_creator)
├── dir_of_models/
│   ├── Utente/
│   │   ├── .tmp_models_dir/
│   │   ├── topic/
│   │   └── model.json
├── flaskr/
│   ├── static/
│   │   ├── main.css
│   │   ├── index.js
│   │   └── models_creator.js
│   ├── templates/
│   │   ├── index.html
│   │   ├── login.html
│   │   └── register.html
│   ├── __init__.py
│   ├── views.py
│   ├── login.py
│   ├── models.py
│   └── uploads/
├── instance
│   └── site.db
├── ca-root-cert.crt
├── const.py
├── install.sh
├── models_creator.py
├── reg_class.py
├── reg.py
├── requirements.txt
├── run.py
├── subscriber.py
└── utils.py
```

## 4.1 Gestione modelli

Il percorso di `dir_of_models` ospita una gerarchia di directory che riflette l'organizzazione dei modelli generati da ciascun utente. Ogni utente è rappresentato da una propria sottodirectory all'interno della quale si trovano i modelli da essi creati. È rilevabile che all'interno di queste sottodirectory utente, vi sono ulteriori suddivisioni in sottodirectory. È presente per ogni utente una directory denominata `.tmp_models_dir`, la quale contiene modelli che non sono ancora stati ne salvati ne scartati, illustrando chiaramente la loro connotazione temporanea. Per ciascun utente potrebbero esistere ulteriori sottodirectory il cui numero può variare. Queste ulteriori directory sono finalizzate a contenere i modelli creati per ogni sensore. Tale suddivisione consente una gestione ordinata dei vari modelli che l'utente crea nel tempo.

## 4.2 Struttura Flask

All'interno della directory `flaskr` troviamo l'intera struttura dell'applicazione Flask.

- **static:** directory che contiene i file CSS e gli script JavaScript che vengono utilizzati per la parte front-end dell'applicazione e interagiscono dinamicamente con le pagine web. Questi file sono utilizzati per personalizzare l'aspetto e il comportamento dell'app.
- **templates:** directory che contiene i file HTML che vengono renderizzati sul browser dell'utente. Questi file HTML rappresentano le diverse pagine web dell'applicazione e contengono il markup e i template per la visualizzazione dei dati.
- **uploads:** directory destinata a contenere temporaneamente i file CSV che gli utenti caricano quando eseguono il calcolo dei modelli.
- **\_\_init\_\_.py:** file di inizializzazione che conferisce validità di 'package' Python alla directory. Inoltre, questo file incorpora il codice di configurazione, l'inizializzazione delle estensioni Flask e altre impostazioni globali dell'applicazione.
- **views.py:** file che definisce il comportamento dell'app in risposta alle interazioni degli utenti con la pagina web. Contiene le definizioni delle route URL e delle funzioni di visualizzazione che gestiscono le richieste dell'utente e restituiscono le risposte appropriate.
- **login.py:** file che definisce due classi di form per l'applicazione Flask: una per la registrazione dell'utente e l'altra per l'accesso.
- **models.py:** file che definisce come vengono rappresentati gli utenti nel database dell'applicazione. Questo file contiene le definizioni delle classi dei modelli di dati che rappresentano gli utenti e le relative informazioni nel database.

## 4.3 Gestione Utenti

All'interno della directory `instance` troviamo il database nel quale vengono memorizzati i vari utenti. Gli utenti sono caratterizzati dai loro nomi utente unici, dalle password per l'autenticazione e da un ID unico nel database assegnato automaticamente durante la fase di registrazione.

## 4.4 Strumenti di Creazione dei Modelli

- **reg\_class.py**: definizione della classe python 'RegressionModel' che funge da contenitore per tutte le informazioni associate ad un modello, che può essere creato o importato nell'applicazione.
- **reg.py**: codice che si occupa della parte di calcolo e di creazione dei modelli. Questo modulo è responsabile per l'implementazione delle operazioni di creazione dei modelli basati sui dati forniti dall'utente.
- **utils.py**: file che contiene funzioni di utilità che aiutano a formattare i file di input in modo specifico per l'applicazione. Queste funzioni sono utilizzate per elaborare e preparare i dati in modo che possano essere utilizzati nel processo di creazione del modello.
- **models\_creator.py**: script progettato per semplificare la creazione di più modelli di regressione mediante un'unica esecuzione. Utilizza la libreria `argparse` [1] per accettare input da riga di comando. Facilita l'automatizzazione della creazione di modelli, consentendo agli utenti di ottimizzare il processo senza dover ripetere manualmente le stesse operazioni.
- **const.py**: file utilizzato per definire e dichiarare costanti che vengono utilizzate in diverse parti della libreria

## 4.5 Iterazione col Broker

Il file `subscriber.py` si occuperà di stabilire una connessione con un intermediario (broker) attraverso le credenziali specificate in un file posizionato all'interno di una directory dedicata al sensore dell'utente e ad un certificato `q`. Dopo aver stabilito con successo questa connessione, il programma riceverà dati dal broker a cui è connesso. Questi dati verranno utilizzati per effettuare previsioni di valori utilizzando modelli preesistenti. Le previsioni risultanti saranno quindi inviate al topic specificato nel file di configurazione. Questo codice fornisce un'applicazione basata sul protocollo MQTT [2], che utilizza la libreria `Paho` [3] per connettersi a un broker MQTT, pubblicare messaggi e sottoscrivere argomenti per ricevere messaggi pubblicati.

# Capitolo 5

## Dettagli implementativi

### 5.1 Formato dei File CSV

I dati in formato CSV devono essere strutturati seguendo un formato preciso e conforme. La prima riga dei file deve rappresentare l'intestazione, delineando i contenuti delle colonne, mentre la prima colonna è riservata ai timestamps, i quali devono essere accuratamente organizzati in linea con le indicazioni fornite.

#### 1. Formato ISO:

- Esempio: 2023-08-10 15:30:00
- Formato: YYYY-MM-DD HH:MM:SS

#### 2. Formato europeo:

- Esempio: 10/08/2023 15:30:00
- Formato: DD/MM/YYYY HH:MM:SS

(Nota: è possibile tralasciare i secondi, se necessario)

### 5.2 Manipolazione dati

Nel processo di manipolazione di dati espressi attraverso file CSV, sarà impiegata la libreria `pandas` [4]. L'operazione preliminare consisterà nella conversione di questi file in strutture dati di tipo `DataFrame` mediante l'utilizzo della funzione `csv_read`. Tuttavia, è di fondamentale importanza porre attenzione a un insieme di strategie utili atte a garantire la qualità e l'integrità dei dati, nonché a fornire una base solida per le analisi successive.

Tra queste strategie figura l'adozione di accorgimenti che mirano a eliminare le colonne prive di attributi numerici o prive di definizione, con l'obiettivo di assicurare che solo le informazioni rilevanti siano conservate all'interno dei `DataFrame`. Parallelamente, si compirà l'azione di rimozione delle righe duplicate, allo scopo di evitare duplicazioni superflue e garantire la coerenza dei dati. Un ulteriore passo sarà rappresentato dall'eliminazione delle righe contenenti valori non definiti, in modo da garantire l'integrità e la completezza dei dati destinati all'analisi.

Risulta, inoltre, essenziale considerare che l'analisi richiede l'utilizzo di due DataFrame distinti per il calcolo del modello finale. Al fine di garantire la congruità tra questi due elementi, sarà impiegata la funzione `aligned`. Questa misura mira a sincronizzare gli istanti temporali all'interno dei due DataFrame, assicurando che solo istanti comuni siano inclusi nei dati sottoposti all'analisi. Ciò contribuirà a evitare conflitti temporali e a ottimizzare l'omogeneità nell'analisi stessa.

timestamps	Temp	Umi
2023-08-10 14:30	25,1	8
2023-08-10 15:30	22,1	12
2023-08-10 16:30	21,1	11
2023-08-10 17:30	20,3	13

Tabella 5.1: File csv di input

timestamps	Var
2023-08-10 13:30	25,3
2023-08-10 14:30	25,1
2023-08-10 15:30	22,1
2023-08-10 16:30	21,1

Tabella 5.2: File csv di output

timestamps	Temp	Umi
2023-08-10 14:30	25,1	8
2023-08-10 15:30	22,1	12
2023-08-10 16:30	21,1	11

Tabella 5.3: File di input dopo la funzione `aligned`

timestamps	Var
2023-08-10 14:30	25,1
2023-08-10 15:30	22,1
2023-08-10 16:30	21,1

Tabella 5.4: File csv di output dopo la funzione `aligned`

### 5.3 Parametro window

Il parametro 'window', denotato da un valore intero positivo 'i', assume una funzione di rilievo nell'ambito della temporizzazione dei dati, contribuendo a determinare l'articolazione e l'interconnessione degli istanti temporali all'interno di un contesto sequenziale.

Quando il parametro 'window' è impostato a 0, ogni istante temporale viene trattato come un'entità isolata, distinta dalle altre in termini di rappresentazione e relazioni temporali. In questa configurazione, le informazioni associate a ciascun istante sono espresse senza coinvolgimento di istanti successivi o precedenti. Contrariamente, quando il parametro 'window' è definito con un valore positivo 'i', esso condiziona una dinamica di aggregazione temporale. Ogni istante temporale, oltre a presentare i propri dati distintivi, incorpora anche i dati relativi agli 'i' istanti temporali precedenti. Questa configurazione crea una catena sequenziale di istanti collegati, consentendo l'analisi delle tendenze temporali su un intervallo di 'i' istanti. È importante notare che, in situazioni in cui il numero di istanti temporali precedenti sia inferiore a 'i', l'istante corrente potrebbe non essere incluso nell'analisi aggregata.

timestamps	Temp	Umi
2023-08-10 14:30	25,1	8
2023-08-10 15:30	22,1	12
2023-08-10 16:30	21,1	11
2023-08-10 17:30	20,3	13

Tabella 5.5: window impostato a 0

timestamps	Temp	Umi	Temp-1	Umi-1
2023-08-10 15:30	22,1	12	25,1	8
2023-08-10 16:30	21,1	11	22,1	12
2023-08-10 17:30	20,3	13	21,1	11

Tabella 5.6: window impostato ad 1

timestamps	Temp	Umi	Temp-1	Umi-1	Temp-2	Umi-2
2023-08-10 16:30	21,1	11	22,1	12	25,1	8
2023-08-10 17:30	20,3	13	21,1	11	22,1	12

Tabella 5.7: window impostato a 2

## 5.4 Matrice MxDxH

La matrice MxDxH è stata progettata con l'obiettivo di rappresentare l'intero arco temporale annuale, inclusi i mesi, i giorni e le fasce orarie, mediante l'utilizzo di valori discreti espressi in forma binaria. Questa rappresentazione binaria dei timestamp ne facilita l'inclusione nel calcolo del modello, agevolando analisi e previsioni basate su tendenze temporali e relazioni tra diversi istanti. La matrice è strutturata in modo tale da avere 12 colonne corrispondenti ai mesi dell'anno (indicate con 'm'), 31 colonne per i giorni rilevanti in ciascun mese (indicate con 'd') e ulteriori 24 colonne per le diverse fasce orarie (indicate con 'h'). Per convertire un timestamp all'interno di questa matrice, vengono seguite queste fasi:

1. **Identificazione del Mese:** La colonna 'm' corrispondente al mese del timestamp contiene il valore 1, mentre tutte le altre colonne 'm' sono a 0.
2. **Identificazione del Giorno:** La colonna 'd' corrispondente al giorno del timestamp contiene il valore 1, mentre le altre colonne 'd' sono a 0.
3. **Identificazione dell'Ora:** La colonna 'h' corrispondente all'ora del timestamp contiene il valore 1, mentre le altre colonne 'h' sono a 0.

1m	...	8m	...	12m	1d	...	10d	...	31d	00h	...	15h	...	23h
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0

Tabella 5.8: MxDxH di 2023-08-10 15:30:00

## 5.5 Missing values

Considerando che la matrice  $M \times D \times H$  è statica, è possibile che l'utente non fornisca dati sufficienti per garantire che ogni colonna di questa matrice sia diversa da zero. Questa situazione potrebbe generare problematiche durante il processo di elaborazione. Per prevenire tali problematiche, è stata adottata una strategia di default: ogni colonna della matrice priva di informazioni viene rilevata e segnalata all'utente. Successivamente, questa colonna viene esclusa dai dati utilizzati nell'analisi, al fine di assicurare una corretta elaborazione dei modelli. Infine queste colonne verranno segnalate all'utente insieme agli indicatori di bontà del modello calcolato.

## 5.6 Flask

Flask [5] è un framework web leggero e flessibile scritto in Python, progettato per semplificare la creazione di applicazioni web. L'integrazione di Flask con la libreria consente la creazione di un'interfaccia web intuitiva per configurare i parametri del modello. L'utilizzo di Flask semplifica non solo la gestione delle richieste HTTP e delle route, ma fornisce anche un sistema di gestione degli utenti, rendendo la libreria accessibile e utilizzabile in modo sicuro e intuitivo da parte di più utenti contemporaneamente.

```
192.168.1.24 - - [27/Sep/2023 13:48:49] "GET / HTTP/1.1" 302 -
192.168.1.24 - - [27/Sep/2023 13:48:49] "GET /login HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:48:49] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.147 - - [27/Sep/2023 13:48:55] "GET / HTTP/1.1" 302 -
192.168.1.147 - - [27/Sep/2023 13:48:55] "GET /login HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:49:07] "POST /login HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:49:07] "GET /static/main.css HTTP/1.1" 304 -
192.168.1.147 - - [27/Sep/2023 13:49:07] "GET /static/index.js HTTP/1.1" 304 -
192.168.1.147 - - [27/Sep/2023 13:49:07] "GET /static/models_creator.js HTTP/1.1" 304 -
192.168.1.24 - - [27/Sep/2023 13:49:33] "POST /login HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:35] "GET /register HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:44] "POST /register HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:49] "POST /register HTTP/1.1" 302 -
192.168.1.24 - - [27/Sep/2023 13:49:50] "GET /login HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:57] "POST /login HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:58] "GET /static/main.css HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:58] "GET /static/index.js HTTP/1.1" 200 -
192.168.1.24 - - [27/Sep/2023 13:49:58] "GET /static/models_creator.js HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:50:00] "POST /models_creator HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:50:05] "POST /saving HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:50:07] "POST /login HTTP/1.1" 200 -
192.168.1.147 - - [27/Sep/2023 13:50:07] "GET /static/main.css HTTP/1.1" 304 -
192.168.1.147 - - [27/Sep/2023 13:50:07] "GET /static/index.js HTTP/1.1" 304 -
192.168.1.147 - - [27/Sep/2023 13:50:07] "GET /static/models_creator.js HTTP/1.1" 304 -
192.168.1.24 - - [27/Sep/2023 13:50:10] "GET /logout HTTP/1.1" 302 -
192.168.1.24 - - [27/Sep/2023 13:50:10] "GET /login HTTP/1.1" 200 -
```

Figura 5.1: Gestione richieste HTTP di più utenti

## 5.7 Maschera dei valori

Al completamento del procedimento di calcolo di un modello, è possibile che il Data-Frame risultante si presenti con una dimensione inferiore rispetto a quella originale. Questa riduzione dimensionale deriva dal fatto che durante le fasi di selezione step-forward e step-backward, alcune colonne vengono scelte o escluse dal modello stesso. Questa dinamica introduce la complessità di adattare i dati futuri al modello stesso.

Per affrontare questa situazione, è stata introdotta una soluzione: l'implementazione di una maschera per i coefficienti. Questa maschera costituirà una mappatura



dei valori dei coefficienti associati a ciascuna colonna del DataFrame originale. Pertanto, le colonne che non risultano presenti nel DataFrame finale verranno assegnate un valore di coefficiente pari a zero. Al contrario, le colonne selezionate, che preservano il loro valore di coefficiente precedentemente calcolato, sono riconosciute come significative nel contesto del modello. Questa metodologia di mascheramento dei coefficienti si traduce in un vantaggio determinante per l'adattamento dei dati futuri al modello, persino in presenza di variazioni dimensionali nel DataFrame.

## 5.8 Formato dei modelli

I modelli di previsione sono memorizzati nel formato JSON attraverso l'utilizzo della classe definita nel file `reg_class.py`. Questa classe è stata appositamente progettata per raccogliere i dettagli fondamentali dei modelli di regressione lineare tenendo traccia dei coefficienti associati alle colonne del file di input, delle medie e delle deviazioni standard di queste colonne, oltre alla media e alla deviazione standard dell'unica colonna del file di output e alle colonne non selezionate durante la fase di step-forward e backward. Infine viene indicato il valore di `winodw` usato per calcolare il modello. (Nel modello non sarà presente la colonna `timestamps` in quanto sostituita dalla matrice `MxDxH`). Alcuni coefficienti saranno impostati a 0, questo in quanto viene rappresentata la maschera dei valori.

Listing 5.1: Esempio modello json creato.

```

1 {
2   "B": {
3     "const": 0.0005468847767569886,
4     "1m": 2929050859.3160734,
5     "2m": 0.0,
6     "3m": 0.0,
7     "4m": 3039033727.51237,
8     "5m": 9931225092.367886,
9     "...":
10    "12m": 4736472045.944808,
11    "1d": 36154595302.28574,
12    "...":
13    "31d": 23077162313.65703,
14    "00h": 2486308315.6457458,
15    "...":
16    "23h": 0.1997236318436963,
17    "Barometer_HPa": 0.0,
18    "Temp_°C": 0.0,
19    "HighTemp_°C": 0.43701578124138385,
20    "HeatingDegreeDays": -0.11052685874796615,
21    "WindSpeed_Km_h": 0.27501578124138385
22  },
23  "calib_input_media": {
24    "1m": 0.014922898358481181,
25    "2m": 0.0,
26    "3m": 0.0,

```

```

27     "4m": 0.016083568230807494,
28     "5m": 0.2153871663074117,
29     ...
30     "12m": 0.040043110595257836,
31     "1d": 0.047587464765378874,
32     ...
33     "31d": 0.01881943292986238,
34     "00h": 0.04211573536726911,
35     ...
36     "23h": 0.041618305421986405,
37     "Barometer_HPa": 1014.2843143757254,
38     "Temp__C": 19.519134471895207,
39     "HighTemp__C": 19.706682142264963,
40     "HeatingDegreeDays": 0.025987730061349697,
41     "WindSpeed_Km_h": 17.756127731061358169
42 },
43 "calib_input_stddev": {
44     "1m": 0.1212494300378763,
45     "2m": 1.0,
46     "3m": 1.0,
47     "4m": 0.12580222229621932,
48     "5m": 0.4111077068511333,
49     ...
50     "12m": 0.19606847526564067,
51     "1d": 0.21290057719171604,
52     ...
53     "31d": 0.13589257839688937,
54     "00h": 0.2008615070943291,
55     ...
56     "23h": 0.1997236318436963,
57     "Barometer_HPa": 5.316383422851651,
58     "Temp__C": 7.47267783333523,
59     "HighTemp__C": 7.514185801621021,
60     "HeatingDegreeDays": 0.04416205072783581,
61     "WindSpeed_Km_h": 9.620528358061351418
62 },
63 "calib_output_media": 21.24883128834356,
64 "calib_output_stddev": 8.274022377012576,
65 "unselected_columns": [
66     "2m",
67     "3m",
68     "Barometer_HPa",
69     "Temp__C",
70 ],
71 "window": 0
72 }

```

## 5.9 MQTT

MQTT è un protocollo di messaggistica leggero e flessibile progettato per dispositivi con limitate risorse di rete e di calcolo. Nel contesto delle interazioni con sensori, MQTT svolge un ruolo cruciale per diverse ragioni:

- Comunicazione in tempo reale: consente ai sensori di inviare dati in tempo reale alla libreria.
- Publish-Subscribe Model: MQTT utilizza un modello di publish-subscribe, dove i sensori agiscono come publisher che inviano dati a specifici argomenti (topics) e l'applicazione agisce come subscriber, ovvero che si iscrivono a tali argomenti per ricevere i dati pertinenti. Questo modello consente una gestione efficiente delle informazioni da parte dei dispositivi e delle applicazioni.
- Scalabilità: MQTT è altamente scalabile e può gestire grandi quantità di dispositivi con facilità.

Con il modello di pubblicazione-sottoscrizione di MQTT, il programma può iscriversi a specifici argomenti (topics) per ricevere solo i dati di interesse. Questo meccanismo semplifica la gestione delle informazioni provenienti da diversi sensori.

# Capitolo 6

## Funzionamento

### 6.1 Registrazione ed autenticazione

Per poter interagire con la libreria, è necessario essere in possesso di un account. Qualora l'utente non disponga di un account, sarà possibile procedere con la registrazione, la quale richiederà un 'username' e una 'password' (con la necessità di confermare quest'ultima). Una volta inseriti l'username e la password, il sistema eseguirà una ricerca all'interno del database per verificare la corrispondenza delle credenziali. Se le credenziali fornite risultano essere corrette, sarà possibile instaurare interazioni con la libreria. Tuttavia, nel caso in cui non emerga alcuna corrispondenza nel database in relazione alle informazioni fornite, l'utente verrà invitato a reinserire le proprie credenziali.

### 6.2 Iterazione Server e Creazione dei modelli

Il server Flask implementa il processo di creazione dei modelli di regressione lineare mediante l'utilizzo dello script `models_creator.py`. L'interfaccia web di Flask agevola la configurazione dei parametri essenziali volti a garantire il corretto funzionamento di tale script. I file caricati attraverso questa interfaccia, vengono rinominati come 'input.csv' e 'st1\_output.csv' (il valore aumenta progressivamente in base al numero dei file di output) e vengono temporaneamente salvati all'interno della directory `uploads`. Lo script interagirà con questa directory, dalla quale estrarrà i file di input necessari per la creazione del modello. Al termine dell'esecuzione, lo script salva i modelli creati nella sottodirectory dell'utente chiamata `.tmp_models_dir`.

### 6.3 Calcolo del modello

Il processo di calcolo del modello si attua mediante l'invocazione della funzione `make_regression` contenuta nel file `reg.py`. Questa procedura riceve in ingresso i due DataFrame derivanti dalle fasi di trattamento dati precedentemente delineate. In tale funzione, è osservabile la presenza di un parametro denominato 'test', che se configurato a 'True', consente la suddivisione dei dati in un set di addestramento (train) costituente l'80% e un set di test del 20%. Il calcolo dei modelli avviene

secondo un accurata scelta delle colonne seguendo i metodi della libreria `stepwise-regression` [6]. Successivamente reintroduciamo tutte le colonne della matrice  $M \times D \times H$  non selezionate dalla regressione, e creiamo il modello mediante l'utilizzo della libreria `statsmodels` [7]. Dopo la creazione, è possibile tramite il parametro 'test' verificare che le predizioni siano accettabili. In conclusione, viene restituito un oggetto `RegressionModel` definito dalla classe `RegressionModel` nel file `reg_class.py` che rappresenterà il modello creato.

### 6.3.1 Stepwise-regression

L'approccio `stepwise` alla regressione lineare procede in due fasi principali: `forward` e `backward`. Inizialmente, inizia con un modello vuoto e aggiunge iterativamente le variabili predittive una alla volta, valutando l'effetto di ciascuna aggiunta sulla qualità del modello. Successivamente, potrebbe rimuovere variabili che non contribuiscono in modo significativo al modello. Il processo di aggiunta e rimozione continua fino a quando il modello raggiunge un miglioramento minimo del coefficiente di determinazione.

## 6.4 Salvataggio dei Modelli

Al termine dell'esecuzione, verranno mostrati all'utente degli indicatori che daranno un'idea della qualità delle previsioni del modello creato. A questo punto l'utente potrà decidere se scartare o se salvare il modello. Nel primo caso verrà eliminato il modello dalla directory che lo contiene, mentre nel secondo caso verrà spostato nella directory del sensore precedentemente scelta.

## 6.5 Configurazione Broker

L'utente autenticato ha la possibilità di selezionare la directory del sensore e configurare il broker in qualsiasi momento tramite l'apposito modulo. Una volta premuto il pulsante 'Salva', il file di configurazione viene generato automaticamente all'interno della directory del sensore. Se esistesse già un file di configurazione precedente, questo verrebbe sovrascritto con il nuovo.

# Bibliografia

- [1] <https://docs.python.org/3/library/argparse.html>.
- [2] <https://mqtt.org/>.
- [3] <https://pypi.org/project/paho-mqtt/>.
- [4] <https://pandas.pydata.org/>.
- [5] <https://flask.palletsprojects.com/en/2.1.x/>.
- [6] <https://github.com/AakkashVijayakumar/stepwise-regression>.
- [7] <https://www.statsmodels.org/stable/index.html>.