Documentazione applicazione

Alessio Bonizzato

1 Introduzione

L'obiettivo dell'applicazione è di analizzare dati ambientali e sviluppare modelli di regressione predittiva basati su dati storici. Attraverso questo processo, l'applicazione mira a creare modelli che possano stimare una variabile di interesse in base alle variabili ambientali disponibili. L'obiettivo finale è fornire agli utenti uno strumento che consenta di fare previsioni basate su dati raccolti, contribuendo così a guidare decisioni informate e ottimizzate in vari contesti. I modelli generati dall'applicazione possono essere utilizzati per anticipare le tendenze, ottimizzare le risorse e prendere decisioni strategiche basate su analisi dati approfondite.

2 Installazione e Configurazione

Per installare l'applicazione, è necessario seguire attentamente i passaggi riportati di seguito:

Installazione

1. Scaricare l'intera repository dal repository GitHub ufficiale dell'applicazione. È possibile farlo tramite il comando 'git clone' oppure scaricando l'archivio ZIP direttamente dal sito.

2. Prima di procedere con l'installazione, è fondamentale verificare di avere 'pip' installato nel sistema. 'pip' è il gestore dei pacchetti di Python e viene utilizzato per installare le dipendenze dell'applicazione.

3. 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`

4. 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. Le dipendenze sono specificate nel file `requirements.txt`

Configurazione

Per eseguire il server, è possibile seguire due metodi:

- Utilizzando il comando `python3 run.py`.
- Impostando una variabile d'ambiente seguendo quanto segue:
    1. Imposti la variabile d'ambiente eseguendo il comando: `export FLASK_APP=flaskr`
    2. Eseguire il seguente comando da terminale per avviare il server `python3 -m flask run`.

Il secondo metodo consente di evitare di dover specificare esplicitamente il nome del file dell'applicazione Flask ogni volta che si avvia il server.

3 Guida all'Utilizzo

1. **Accedi al Server:** Utilizza il tuo browser per accedere alle funzionalità di creazione dei modelli, collegandoti utilizzando l'indirizzo IP del server Flask.

2. **Fornitura dei Parametri**: Per creare un modello, è necessario fornire i seguenti parametri tramite la pagina web del server:

   - Seleziona o inserisci un nuovo sensore al quale verranno associati i modelli che si andranno a creare
   - Carica il file di input contenente i dati dei sensori.
   - Carica uno o più file di output rappresentanti la variabile di risposta che vuoi prevedere.
   - Imposta il valore 'window', un numero intero che determina quanti istanti temporali precedenti verranno considerati nella creazione del modello.
   - Assegna un nome al modello. Se stai creando più file di output, il nome verrà affiancato da un numero intero che incrementerà in modo progressivo.
   - Decidi se eseguire o meno il test del modello. In caso positivo, i dati di input verranno divisi in un rapporto 80% addestramento (ovvero i dati utilizzati la creazione del modello) e 20% test (ovvero i dati che verranno utilizzati per verificare che il modello sia oppure no preciso).

3. **Visualizzazione dei Risultati:** Una volta completata la creazione del modello, l'applicazione mostrerà alcuni indicatori derivanti dal modello stesso. Questi valori daranno un'idea della qualità delle previsioni e dell'adeguatezza del modello rispetto ai dati.

4. **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 dei sensori o scartarlo completamente.

5. **Salvataggio con Dati Broker (opzionale):** Nel caso si decida di salvare il modello, sarà possibile creare un file di configigurazione 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

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/
│   │   ├── index.js
│   │   └── models_creator.js
│   ├── templates/
│   │   └── index.html
│   ├── __init__.py
│   ├── views.py
│   └── uploads/
├── ca-root-cert.crt
├── const.py
├── install.sh
├── models_creator.py
├── README.md
├── reg_class.py
├── reg.py
├── requirements.txt
├── run.py
├── subscriber.py
└── utils.py
```

4.1 Gestione modelli e utenti

La direcotry 'dir_of_models' rappresenta la gestione degli utenti con i loro vari modelli. All'interno della directory, troviamo una directory per ogni utente al quale viene associata una directory nascosta '.tmp_models_dir' in cui andranno messi i modelli creati non ancora salvati o scartati (quindi saranno temporanei). Ogni utente avrà anche 0 o più directory che rappresentano i sensori, dentro le quali vi saranno i vari modelli precedentemente salvati.

4.2 Struttura Flask

Dentro la directory 'flaskr' troviamo tutta la struttura dell'applicazione Flask.

- **static**: directory che contiene eventuali file 'css' e gli script 'JavaScript' che interagiscono dinamicamente con le pagine web.
- **templates**: directory che contiene i file HTML che vengono renderizzati sul browser.
- **uploads**: directory che conterrà in modo temporaneo i file CSV che gli utenti caricano quando calcolano i modelli.

- **__init__.py**: file di inizializzazione che rende una cartella un 'package' Python valido. Contiene codice di configurazione, inizializzazione delle estensioni Flask e altre impostazioni globali dell'app.
- **views.py** file che definisce il comportamento dell'app quando un utente iteragisce con la pagina web.

4.3 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.
- **utils.py**: file che contiene funzioni di utilità come la trasformazione da timestamp alla matrice MxDxH o l'allineamento di 2 file csv etc.
- **models_creator.py**: script progettato per agevolare la creazione di molteplici modelli mediante una singola esecuzione.
- **const.py**: file che contiene delle costanti utili per l'applicazione

5 Dettagli implementativi

5.1 Formato dei File CSV

I dati in formato CSV devono essere strutturati seguendo un formato preciso e conforme alle specifiche richieste. La prima riga dei file deve rappresentare l'intestazione, delineando i contenuti delle colonne, mentre la prima colonna è riservata ai timestamps, che devono essere opportunamente disposti in conformità con le indicazioni fornite.

1. **Formato data e ora separati da spazio:**
   - Esempio: 2023-08-10 15:30:00
   - Formato: YYYY-MM-DD HH:MM:SS
2. **Formato data e ore in stile europeo:**
   - Esempio: 10/08/2023 15:30:00
   - Formato: DD/MM/YYYY HH:MM:SS

5.2 Matrice MxDxH

La matrice in questione è una matrice creata ad hoc per poter rappresentare tutti i giorni dell'anno e tutte le ore con valolri di 0 e 1. Questa è composta da 12 colonne dedicati ai mesi dell'anno, 31 dedicati ai giorni in un mese, e 24 dedicate alle ore. Ogni timestamp verrà convertito e rappresentato con 0 e 1 all'interno din una riga della matrice create. Vedi esempio:

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

Tabella 1: MxDxH di 2023-08-10 15:30:00

5.3 Parametro window

Il parametro window dato un valore intero positivo 'i' modificherà le righe dei vari timestamps. Ad ogni riga verranno concatenati i valori degli 'i' timestamp precedenti. Se un timestamp non ha abbastanza valori precedenti viene eliminato dai dati finali. Vedi esempio

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

Tabella 2: 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

Tabella 3: 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

Tabella 4: window impostato a 2

5.4 Manipolazione dati

Per poter lavorare con dei file CSV, verrà utilizzata la libreria `pandas`, infatti li convertiremo in DataFrame mediante la funzione `csv_read(path: string)`. Oltre alla conversione si prenderanno degli accorgimenti utili, come l'eliminazione di colonne che non sono di tipo numerico o adirittura non definite e l'eliminazione di eventuali righe ripetute. Per evitare imprevisti o che il modello non risulti ben calibrato verranno eliminate inoltre, tutte le righe che avranno almeno un valore impostato 'null' così che i dati che andremo ad utilizzare siano completi.

Essendo che per calcolare il modello finale dobbiamo usare 2 DataFrame, faremo si, con l'utilizzo di `alligned()`, che questi possiedano solo timestamps comuni.

5.5 Missing values

Considerando che la matrice MxDxH è statica, potrebbe verificarsi il caso in cui 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.

5.6 Maschera dei valori

Al termine del processo di calcolo di un modello, il Dataframe risultante potrebbe avere una dimensione inferiore rispetto a quello iniziale, poiché alcune colonne vengono selezionate o deselezionate durante le fasi di step-forward e step-backward. Questo potrebbe complicare l'adattamento di tutti 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 conterrà i valori dei coefficienti associati a ogni colonna del Dataframe originale. Pertanto, le colonne non presenti nel Dataframe finale avranno un valore di coefficiente pari a zero, mentre le colonne selezionate conserveranno il loro valore di coefficiente precedentemente calcolato. Questo approccio garantisce la coerenza dei dati futuri con il modello, anche quando le dimensioni del Dataframe variano.

6 Come funziona il tutto

6.1 Iterazione Server e Creazione dei modelli

Il server Flask utilizza lo script `models_creator.py` per creare i modelli di regressione lineare. L'interfaccia web di Flask consente di impostare vari parametri per il corretto funzionamento di quest'ultimo. I file caricati tramite l'interfaccia verranno rinominati come 'input.csv' e 'st1_output.csv' (il valore aumenterà progressivamente in base al numero dei file di output) e verranno temporaneamente salvati nella directory `uploads`. Lo script interagirà con questa directory, dalla quale estrarrà i file di input necessari per la creazione del modello. Alla fine dell'esecuzione, lo script salva i modelli creati in una sottodirectory dell'utente chiamata '.tmp_models_dir'.

6.2 Calcolo del modello

verrà calcolato il modello secondo la funzione `make_regression` del file `reg.py` passando i 2 DataFrame ottenuti seguendo le precedenti operazioni. In questa funzione è possibile notare un parametro 'test' che, se impostato a True, permette di dividere i dati in 80% train e 20% test. Il calcolo dei modelli avviene secondo un accurata scelta delle colonne seguendo i metodi della step-wise regression. Successivamente reintroduciamo tutte le colonne della matrice MxDxH non selezionate dalla regressione, e creiamo il modello. Fatto ciò è possibile tramite il parametro 'test' verificare che le predizioni siano accettabili. Infine ritorniamo un oggetto RegressionModel definito dalla classe RegressionModel nel file `reg_class.py`.

6.2.1 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.

6.2.2 Salvataggio dei Modelli

Al termine dell'esecuzione verrano mostrati all'utente degli indicatori che daranno un'idea della qualità delle previsioni del modello creato. A questo punto l'utente potra 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.