Documentazione applicazione

Alessio Bonizzato

Indice

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 ed esecuzione

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. 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 `pip` installato nel sistema. `pip` è il gestore dei pacchetti di Python e viene utilizzato per installare le dipendenze dell'applicazione.

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

Esecuzione

Per eseguire il server è necessario aprire da su terminale la directory dell'applicazione ed il seguente comando: `python3 -m flask run`.

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. **Login:** Per poter accedere all'applicazione è necessario effetturare l'autenticazione, fornendo username e passsword. In caso non si sia già in possesso di un account sarà necessario crearne uno attraverso l'apposita pagina di registrazione. Questo permette di creare una gerarchia e mantenere i modelli in modo più organizzato senza che si creino conflitti tra i vari modelli di diversi utenti.

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 relativi ai sensori.
    - Carica uno o più file di output che rappresentino la variabile di risposta da prevedere.
    - Imposta il valore 'window', espresso come numero intero, che stabilirà quanti istanti temporali precedenti verranno considerati durante della creazione del modello.
    - Assegna un nome al modello. Se stai caricando 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 dell'80% per l'addestramento (ovvero i dati utilizzati la creazione del modello) e del 20% per il test (ovvero i dati che verranno utilizzati per verificare che il modello sia oppure no preciso).

4. **Visualizzazione dei Risultati:** Una volta completata la fase di creazione del modello, l'applicazione fornirà alcuni indicatori derivanti dal modello stesso. Tali indicatori offriranno un'idea riguardo alla qualità delle previsioni e all'adeguatezza del modello in relazione ai dati forniti.

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 dei sensori 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 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
│   │   ├── login.html
│   │   └── register.html
│   ├── __init__.py
│   ├── views.py
│   ├── login.py
│   ├── models.py
│   └── uploads/
├── instance
│   └── site.db .2 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

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 sotto-cartelle. È presente per ogni utente una directory denominata `.tmp_models_dir`, la quale contiene modelli che ancora 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 (la directory rappresenta il 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 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 destinata a contenere temporaneamente i file CSV che gli utenti caricano al momento del calcolo dei modelli.
- **__init__.py**: file di inizializzazione che conferisce validità di 'package' Python alla directory. Esso 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.
- **login.py** file che definisce due classi di form per un'applicazione web 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

4.3 Gestione Utenti

All'interno della directory `instance` troviamo il database nel quale vengono memorizzati i vari utenti. Ongi utente è caratterrizzato da una coppia di parametri ovvero 'username' e 'password'.

4.4 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

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. Dopo aver stabilito con successo questa connessione, il programma riceverà dati provenienti dal broker al quale è stato collegato. Tali dati saranno sfruttati per effettuare previsioni di valori tramite l'utilizzo di modelli già preesistenti. Le previsioni risultanti saranno poi inviate al topic indicato nel file di configurazione.

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, i quali devono essere accuratamente organizzati in linea 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

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

5.2 Matrice MxDxH

La matrice in esame è stata concepita ad hoc al fine di offrire una rappresentazione esaustiva dell'intero arco temporale annuale congiuntamente a tutte le suddivisioni orarie, mediante l'utilizzo di valori discreti espressi in forma binaria. La struttura matriciale è costituita da un totale di 12 colonne corrispondenti ai dodici mesi dell'anno, 31 colonne assegnate ai giorni rilevanti in ciascun mese e ulteriori 24 colonne riservate alla denotazione delle differenti fasce orarie. Ciascun istante temporale sarà sottoposto a un processo di trasposizione nella forma di una sequenza binaria, specificamente '0' e '1', la quale troverà collocazione all'interno di una singola riga all'interno della struttura matriciale così concepita. 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', 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.

Nel caso in cui il parametro 'window' sia settato a 0, ciascun istante temporale è trattato come un'entità isolata, distinta dalle altre in termini di rappresentazione e di relazioni temporali. In questa prospettiva, le informazioni ad esso associate 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 da rilevare 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 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

Nel processo di manipolazione di dati espressi attraverso file CSV, sarà impiegata la libreria `pandas`. 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, potenzialmente, 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 `alligned()`. 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.

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 completamento del procedimento di calcolo di un modello, è possibile che il DataFrame 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.

6 Funzionamento

6.1 Registrazione ed autenticazione

Per poter interagire con l'applicazione, 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 forniti username e password, si procederà con l'esecuzione di una ricerca all'interno del database. Se le credenziali fornite risultano essere corrette, sarà possibile instaurare interazioni con l'applicativo. 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, 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 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 in una sottodirectory dell'utente chiamata `.tmp_models_dir`.

6.3 Calcolo del modello

Il processo di computazione del modello si configura 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 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`.

6.3.1 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.

6.3.2 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.