Report task di tesi

Leonardo Poggiani

October 17, 2020

Abstract

In questo task veniva richiesto di confrontare visivamente una giornata anomala e una giornata predetta attraverso le giornate normali che la precedono. Inoltre viene anche individuata la *cosine distance* che servirà a predire una giornata anomala senza avvalersi del confronto visivo.

1 Motivazioni

Il task si articolava in tre fasi distinte:

- Preparazione del dato
- Salvataggio del dato
- Recupero del dato e predizione mediante classe Predittore

La prima fase è resa necessaria che il dataset su cui predire le giornate era in forma eterogenea. Infatti il formato su cui effettuare la predizione era del tipo [serie di giorni normali, giorno anomalo].
Quindi era necessario un unico giorno anomalo per ogni *slice* di dati e questo deve essere preceduto da tutti i giorni normali consecutivi fino al giorno anomalo precedente.

La riorganizzazione del dataset permetterà una predizione più efficiente in quanto basterà passare alla classe che si occupa della predizione il dataset target della predizione, che conterrà tutti e solamente i dati necessari alla predizione.
Il salvataggio del dato è un altro punto problematico, perchè viene reso necessario dall'esigenza di poterlo esportare e recuperare con facilità in seguito.
Altrettanto importante darà la riorganizzazione del codice volta alla creazione di un'unica classe **Predittore** che si occuperà di generare i risultati voluti dopo aver ricevuto come input i dati nel nuovo formato.

2 Problema

Il problema in questione riguarda la struttura del dataset iniziale. Infatti la predizione che vogliamo andare a fare richiede un formato specifico che si può

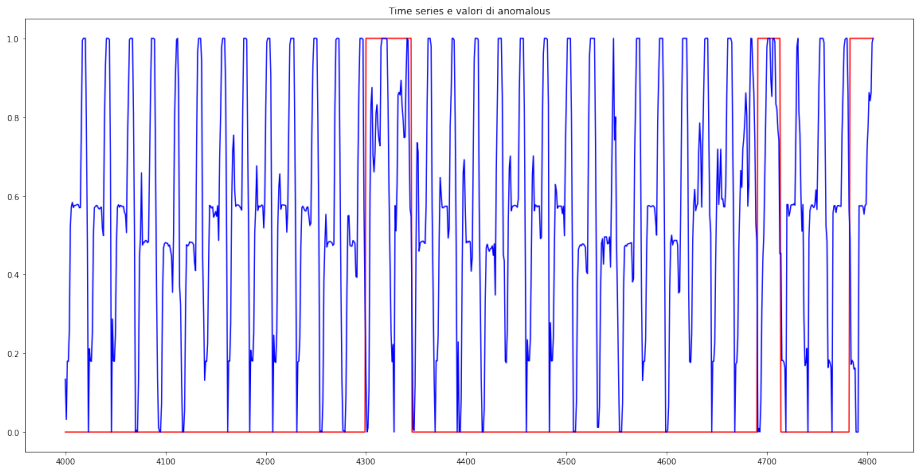

Figure 1: Come veniva raffigurato il dataset in precedenza.

ottenere raggruppando un giorno anomalo e tutti i giorni normali precedenti a dato giorno anomalo.
Un altro problema che si pone è quello di cercare di salvare i dati in un unico file, in quanto più facilmente recuperabile rispetto a salvare ogni dataframe *target* in un file separato.
Tutto questo viene fatto nell'ottica di scrivere un'unica classe dotata di due metodi principali che si occuperanno di predire l'ultima giornata normale all'interno del dataset di riferimento per confrontarla con la time-series reale della stessa giornata, questo al fine di mostrare la bontà delle predizioni effettuate dalla rete, e di predire l'andamento dell'ultima giornata del dataset (quella anomala) a partire dalle precedenti giornate normali, per mostrare le differenze tra la time-series di una giornata normale (quella predetta) e una anomala (quella reale).

3 Risultati

Per risolvere i problemi sopra riportati, per prima cosa si è data importanza alla riorganizzazione del dato.
Per prima cosa si è aggiunta una colonna al dataframe originale denominata "*Finestra*" che ha il compito di indicare il numero di righe appartenenti a giorni normali che precedono un giorno anomalo. Incidentalmente questa informazione fornisce anche l'indicazione dei giorni normali (consecutivi) che precedono dato giorno anomalo in maniera molto semplice:

$$giorninormaliconsecutiviprecedenti = \frac{numerodirighe}{23}$$

	Index	Affluenza	Anomalous	Finestra
0	1	1.0	1	0
1	1	0.0	1	0
2	1	0.29989084300651303	1	0
3	1	0.195044014019326	1	0
4	1	0.198812234343099	1	0
5	1	0.194608629371612	1	0
6	1	0.195873221700534	1	0
7	1	0.0	1	0
8	1	0.5836151837018779	1	0
9	1	0.653947165851926	1	0

Figure 2: Colonna finestra.

	Index	Affluenza	Anomalous	Finestra
206	10	0.0	1	184
1862	82	0.37736370310840794	1	1265
3242	142	0.343491640559844	1	1357

Figure 3: Valori tipici della colonna finestra.

Questa colonna è quindi giustapposta al dataframe originale e si rivelerà molto utile per evitare di dover scorrere ogni volta il dataset, potendo agire direttamente sugli indici per filtrare le entrate che ci interessano.
Dopo questo viene fatta un'iterazione sul dataset scartando eventuali giorni anomali consecutivi, per i quali si terrà in considerazione solo il primo giorno dei due. Durante l'iterazione vengono aggiunti ad una lista tutti i dataframe ottenuti considerando i giorni normali consecutivi precedenti forniti dal campo "Finestra" e il giorno anomalo di riferimento.

L'ultimo passaggio necessario è quello di salvare i dati ricavati, per adesso mantenuti sotto forma di lista di dataframe di dimensioni diverse.
A questo scopo si è deciso di usare un modulo fornito da Python chiamato *Pickle*. *Pickle* implementa un protocollo binario per la serializzazione e la de-serializzazione degli oggetti, convertendo l'intero oggetto in una stringa di byte e salvandolo in un unico file *.pkl*.
Questo modulo si è rivelato essere molto comodo ed utile in quanto permette di mantenere la struttura dei dati iniziale al momento dell'*unpickling*, ovvero al momento del passaggio inverso.
Infatti se attraverso il comando *dump* vengono serializzati gli oggetti nel file, attraverso il comando *load* questi vengono recuperati esattamente nel formato in cui erano stati salvati. Tutto questo però, ha come controindicazione la perdita della leggibilità dei dati salvati sul file. Infatti essendo salvati come stringhe di byte e quindi in formato binario, non è possibile andare a leggere i file per verificare la consistenza dei dati, ma bisogna prima de-serializzarli.

Questo "difetto" di Pickle è sembrato comunque accettabile a fronte dell'estrema comodità del salvataggio e recupero delle strutture dati.

```
with open('dati_riorganizzati.pkl', 'wb') as output:
  pickle.dump(df_list, output, pickle.HIGHEST_PROTOCOL)
```

Listing 1: Pickling dei dati.

```
with open('dati_riorganizzati.pkl', 'rb') as input:
  df_list = pickle.load(input)
```

Listing 2: Unpickling dei dati.

Dopo esserci occupati del salvataggio dei dati nel nuovo formato, si può passare al refactoring del codice andando a costruire una classe **Predittore** che si occuperà di fornire i risultati voluti.
La classe **Predittore** è fornita di due metodi:

- predizione ultima giornata non anomala
- predizione giornata anomala

predizione ultima giornata non anomala: Il primo metodo prende in input un elemento della lista di dataframe salvata su file e toglie l'ultima giornata normale. Dopo questo effettua la predizione usando come input tutte le giornate normali tranne l'ultima.
Fornisce in output l'immagine che confronta la predizione con la serie temporale reale dell'ultima giornata normale e la loro *cosine distance*.

predizione giornata anomala: Il secondo shifta di uno le giornate normali, non considerando la prima e considerando invece l'ultima nel dataframe usato per la predizione.
In output fornisce un'immagine che confronta la giornata predetta (che assomiglierà molto alle giornate normali) e la giornata anomala, per visualizzare le differenza e la loro *cosine distance*.

A questo punto una semplice esecuzione prevede l'unpickling dei dati e in seguito con un ciclo iteriamo la lista di dataframe.
Per ogni elemento a questo punto si fa una considerazione: il numero medio di giorni normali consecutivi che compongono la cosiddetta *"finestra"* è 22 e si può facilmente calcolare.
Questo valore verrà usato come soglia. Infatti se non si hanno un numero sufficiente di giorni per effettuare una predizione la predizione non sarà molto precisa.
Quindi alla fine risulteranno 3 dataframe interessanti nel dataset fornito.

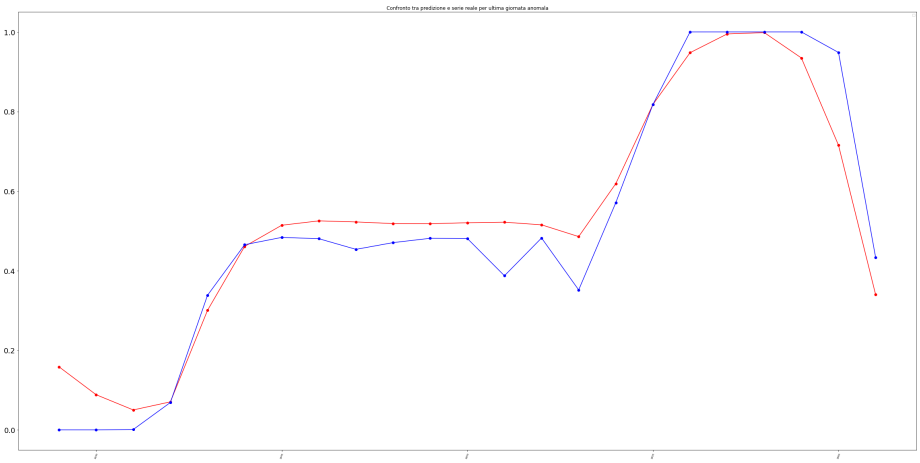

Figure 4: Primo dataset: confronto tra giornata predetta e giornata normale

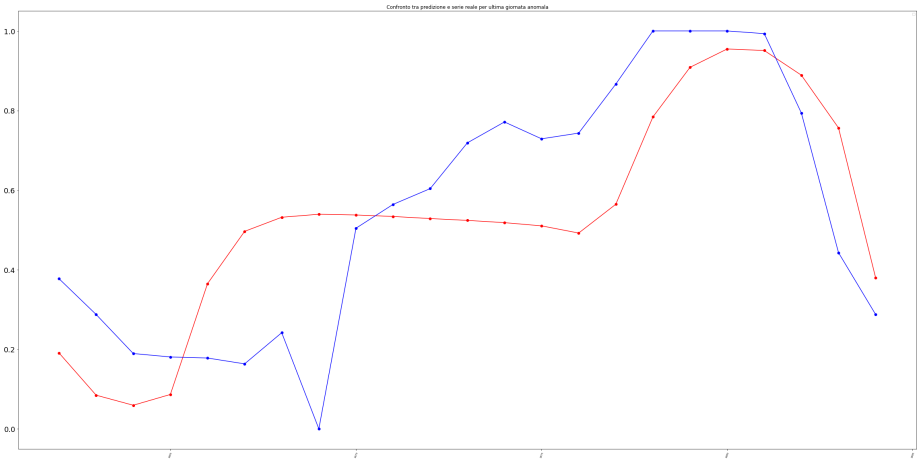

Figure 5: Primo dataset: confronto tra giornata predetta e giornata anomala

4 Conclusioni

Osservando le immagini prodotte si nota l'evidente differenza tra la predizione e la giornata normale e la giornata anomala.
E' possibile anche notare la quasi identicità delle figure rappresentanti le predizioni con le time-series reali, tenuto conto che più la predizione può contare su un maggior numero di giorni normali da usare e più questa sarà precisa.
Per quanto riguardo la *cosine distance* possiamo osservare che nel caso del secondo dataframe la distanza è la minore rilevata.
Questo è dovuto al fatto che il secondo dataframe è quello contenente le entrate

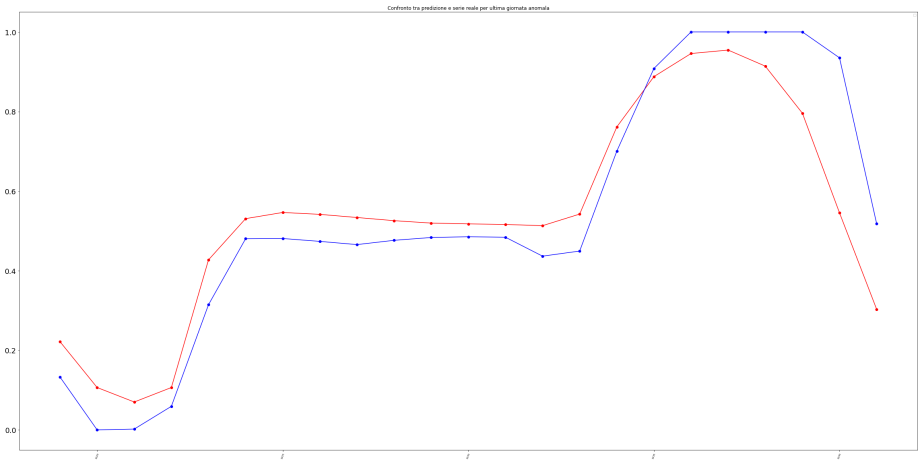

Figure 6: Secondo dataset: confronto tra giornata predetta e giornata normale

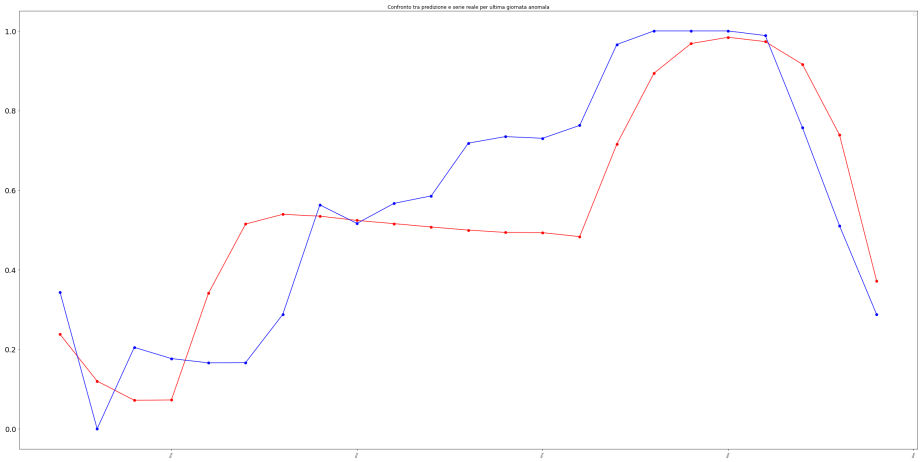

Figure 7: Secondo dataset: confronto tra giornata predetta e giornata anomala

relative alla massima finestra disponibile, ovvero composta dal massimo numero di giorni normali consecutivi.

Possiamo inoltre notare che tutti i valori di *cosine distance* dei giorni anomali superano i valori relativi ai giorni normali.

References

[1] http://docs.python.it/html/lib/module-pickle.html, Documentazione ufficiale di Python per Pickle.

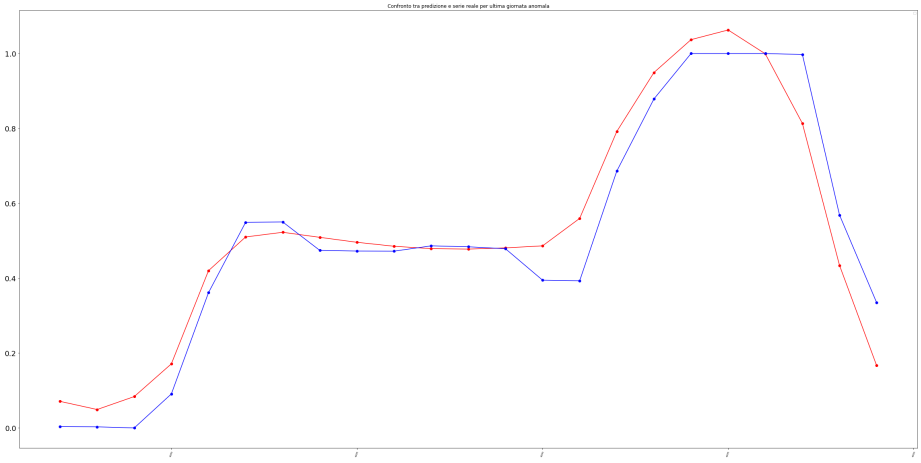

Figure 8: Terzo dataset: confronto tra giornata predetta e giornata normale

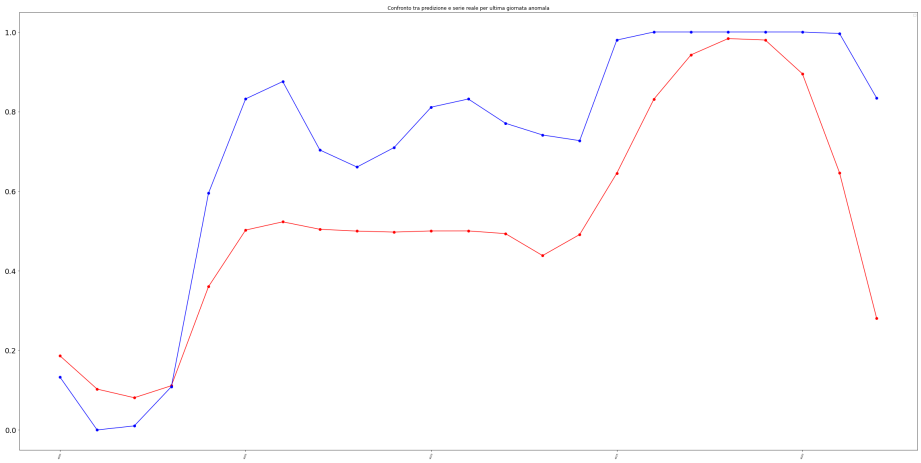

Figure 9: Terzo dataset: confronto tra giornata predetta e giornata anomala

```
No handles with labels found to put in legend.
Cosine distance: 0.0111138907662238 22
```

Figure 10: Primo dataset: Valore di cosine distance tra giorni normali

```
No handles with labels found to put in legend.
Cosine distance: 0.08301010623249161
```

Figure 11: Primo dataset: Valore di cosine distance tra giorni anomali

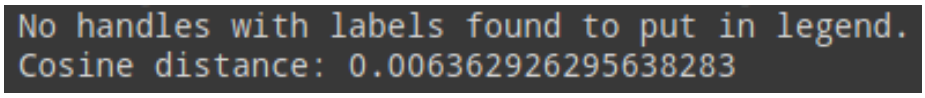

Figure 12: Secondo dataset: Valore di cosine distance tra giorni normali

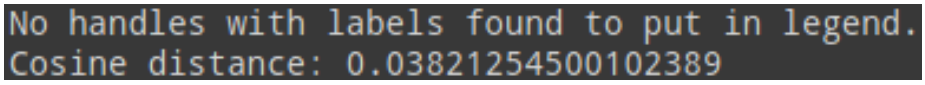

Figure 13: Secondo dataset: Valore di cosine distance tra giorni anomali

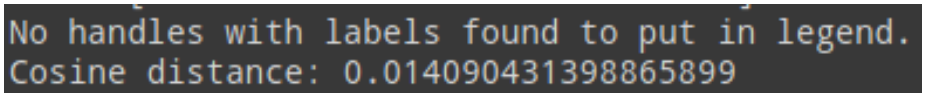

Figure 14: Terzo dataset: Valore di cosine distance tra giorni normali

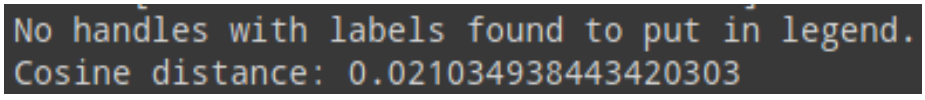

Figure 15: Terzo dataset: Valore di cosine distance tra giorni anomali