Sistemi di supporto alle decisioni

M9Museum Project

Svolto da: Achilli Mattia, Rettaroli Andrea.

Obbiettivo

L'obbiettivo di questo progetto è quello di riuscire a fare delle previsioni sulle serie storiche dell'M9Museum di Venezia. Le serie storiche che compongono il dataset monitorano l'affluenza al museo nelle tre sale.
Utilizzando modelli statistici, di machine learning e infine modelli neurali si vuole prevedere l'afflusso di visitatori al museo.

Analisi

Analisi del problema

Il problema in se ci chiede di prevedere l'afflusso all'M9Museum partendo dai dati raccolti sull'afflusso al museo. Una volta analizzati i dati va determinata la serie storica che essi costituiscono al fine di applicare modelli statistici e neurali per fare forecasting. Da subito è chiaro che lavorando con dati reali si potrebbe incombere in problematiche legate ad essi come:

- Inconsistenza dei dati;
- Errori di misurazioni;
- Eventuali dati mancanti;
- Ingente mole di dati;

Analisi dei dati

Analizzando i dati risulta subito chiaro che la mole di dati in questione è molto vasta,
l'elaborazione degli stessi è costosa in termini di tempo, sarà determinante lavorare sull'ottimizzazione dei processi di lettura e in seguito di addestramento.

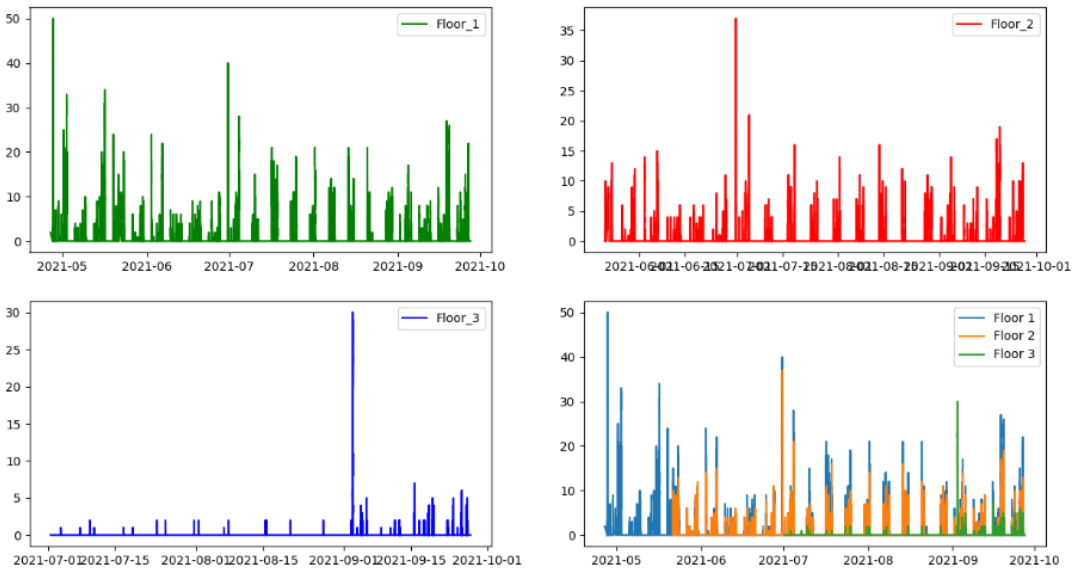
Al fine di riuscire ad applicare i modelli statistici e i modelli neurali, risulta di cruciale importanza analizzare e comprendere i dati del dominio. In seguito si riporta la struttura del file CSV.

timestamp,"area_code","totals","alarms","date"
1632669666838,"floor_3","0","0","2021-09-26"
1632669667149,"front_desk","0","0","2021-09-26"
1632669667226,"floor_1","17","0","2021-09-26"
1632669667600,"floor_2","9","0","2021-09-26"

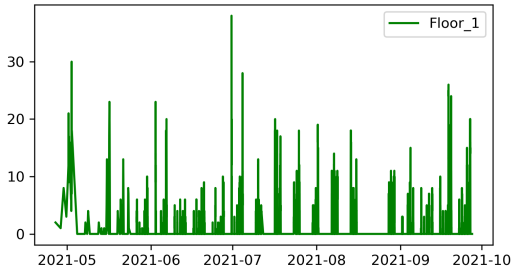
Dal CSV si individuano i dati timestamp, area_code e totals come i dati di rilievo su cui si costruisce la serie storica del numero di presenti nelle varie sale. Il timestamp indica il momento temporale in cui il dato è stato acquisito. il seguente esempio ci aiuta a capire il formato:

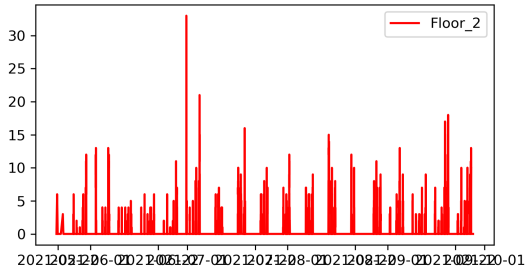
Epoch timestamp: 1640692068
Timestamp in milliseconds: 1640692068000
Date and time (GMT): Tuesday 28 December 2021 11:47:48
Date and time (your time zone): martedì 28 dicembre 2021 12:47:48 GMT+01:00

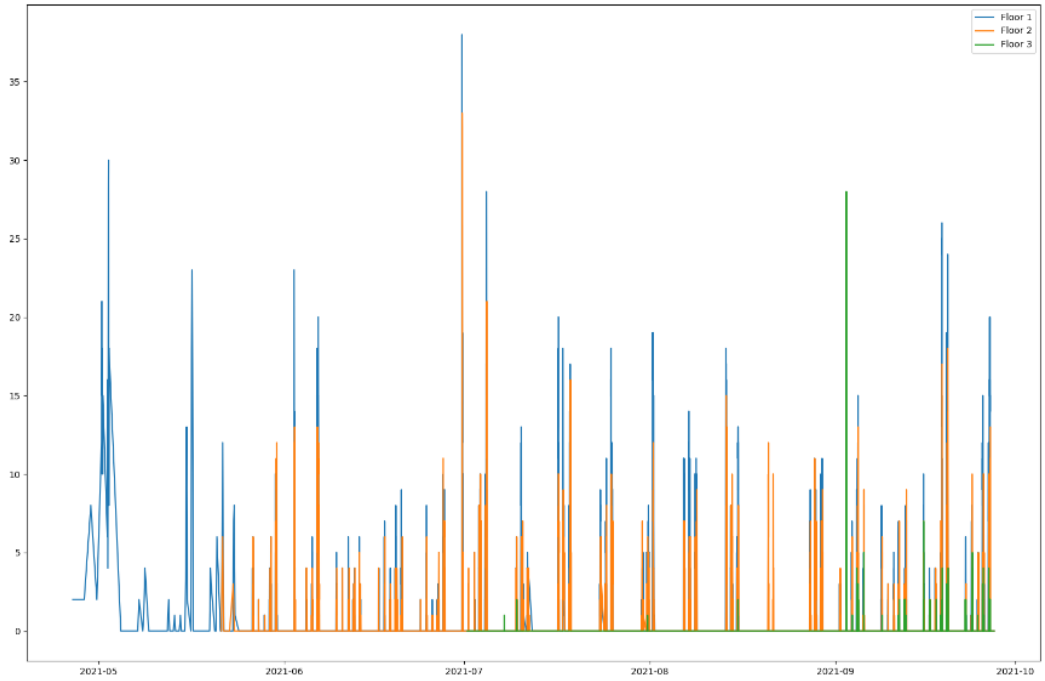
A seguito di questa conversione e di alcune operazioni per calcolare la media dei secondi tra un rilevamento e l'altro si nota che le misurazioni avvengono ogni 2 secondi per ognuno dei tre piani.
Si decide che per costruire la serie si prendono i riferimenti definendo un lasso temporale sensato. Il grafico seguente mostra la serie costruita dall'intero dataset.



Di seguito si vede il grafico visualizzando l'afflusso di persone in intervalli di 10 minuti.







Dal secondo grafico è facilmente rilevabile che all'interno della settimana vi sono dei giorni in cui l'afflusso è 0, ci siamo interfacciati con il Sig.re Luca Agatensi, coordinatore del progetto, per capire se questi dati fossero dovuti a delle chiusure causate dal Covid19 o a degli errori di misurazione o ad altro. In quei giorni il Museo non è aperto al pubblico, ciò prova la correttezza dei dati letti. La serie risulta continua, non si rileva la presenza di outlier.

Sviluppo

DVCS

Per lo sviluppo si decide di utilizzare Git, con Git flow per il versionamento del codice attraverso branch specifici.

Lettura dei dati

In questa fase si leggono i dati da file, essendo però 11.4M i record si pensa che sia meglio caricare i dati su un DB e ottenerli già filtrati per sala al fine di ottimizzare i tempi.

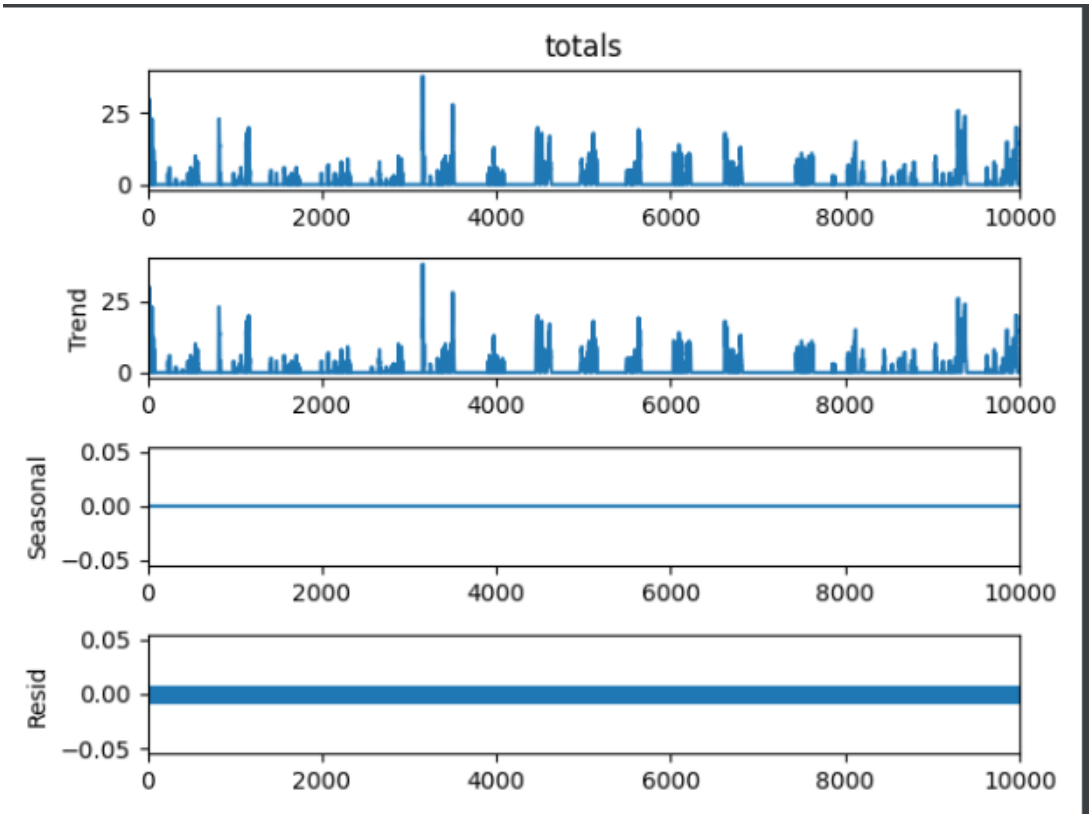
Si ragiona anche alla possibilità di eseguire i metodi neurali su Colab al fine di migliorare le prestazioni soprattutto in termini di tempi di addestramento; risultano però delle complicazioni legate alla memoria delle macchine gratuitamente disponibili e al caricamento dei dati sulle stesse.

Una volta letti e compresi i dati si decide di ottimizzare il processo di lettura al fine di migliorare le prestazioni; a tale scopo si decide di utilizzare un database, nello specifico MongoDB che permette l'import di dati da CSV in maniera molto semplice. Viene creato un file config.py che non viene pushato nel repository ma viene tenuto in locale con la stringa di connessione al DB. Si decide di ottimizzare anche le letture filtrando già i dati per piano in modo da lavorare su un singolo piano alla volta, così facendo i tempi necessari a questa fase si sono ridotti drasticamente. Successivamente si pensa di rimuovere anche le colonne non utili dal dataset per velocizzare le esecuzioni. Come detto recedentemente i dati vengono raccolti in intervalli di 10 minuti, in questo modo la quantità di dati da elaborare si riduce enormemente (si prende un dato ogni 300) pur mantenendo l'andamento reale dei dati.

Successivamente viene fatto preprocessing e analisi specifiche sui dati.

Preprocessing e analisi aggiuntive

Questa è una delle fasi principali e più importanti per l'elaborazione dati. In questa fase si determinano le caratteristiche della serie che permettono di individuare i metodi migliori per andare a fare forecasting su di esse. In particolare ci interessa individuare eventuali componenti di trend e di stagionalità nella serie. Si applica il seasonal decompose per verificare la presenza di queste componenti. La figura seguente mostra un esempio di seasonal decompose nella serie legata al floor 1.



E' evidente che non vi sono componenti di stagionalità né tanto meno di trend. E' più probabile infatti avere delle componenti di trend o stagionalità in un intervallo temporale minore come un mese o una sola stagione come l'estate.

Successivamente si è voluto verificare se fosse stazionaria, vengono calcolate media e varianza e in seguito si eseguono due test statistici: adf e kpss. Il test ADF Dickey-Fuller aumentato i cui risultati sono visibili nella figura seguente.

```
ADF Statistic: -12.94
Critial Values:
1%, -3.43
Critial Values:
5%, -2.86
Critial Values:
10%, -2.57

p-value: 0.00
Stationary
```

In seguito viene effettuato anche il test statistico kpss, che prende il nome dai sui creatori Kwiatkowski, Phillips, Schmidt e Shin, il cui risultato viene mostrato nell'immagine seguente.

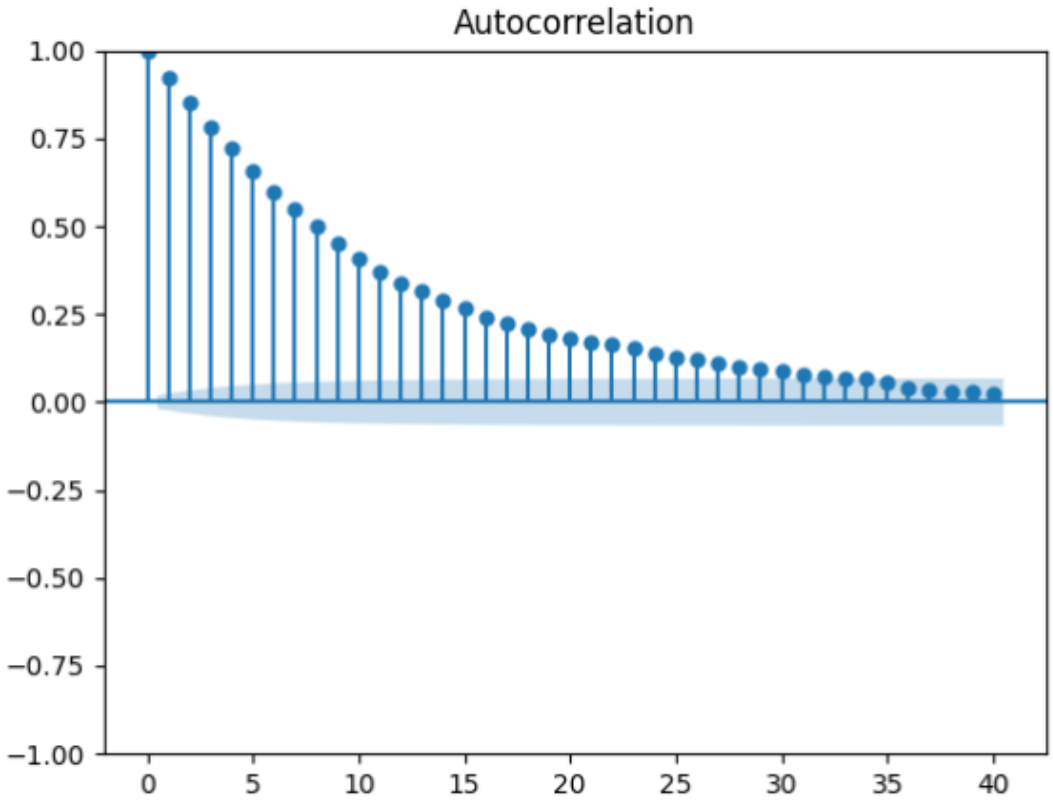
```
ADF Statistic: 0.19
Critial Values:
10%, 0.35
Critial Values:
5%, 0.46
Critial Values:
2.5%, 0.57
Critial Values:
1%, 0.74

p-value: 0.10
Stationary
```

A seguito dei due test statistici le tre serie temporali risultano stazionarie.

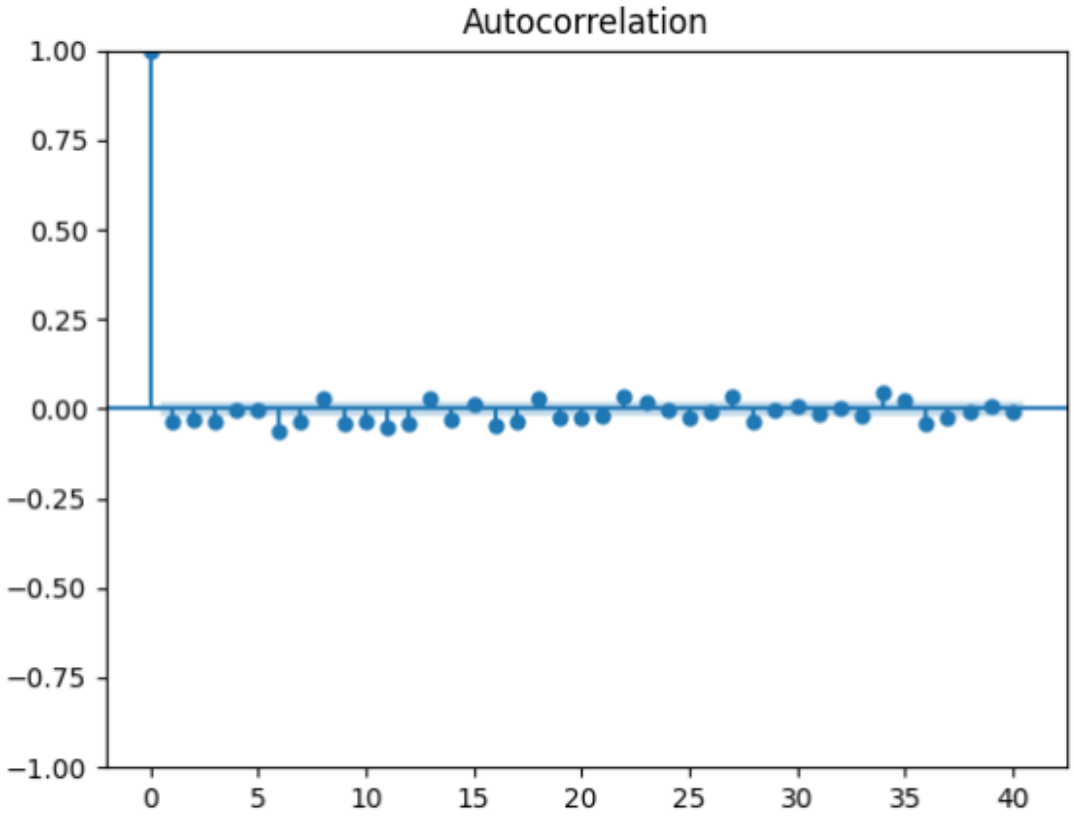
Come ulteriore verifica si sceglie di calcolare media e varianza per la prima e la seconda metà dei dati e verificare che queste rimangano pressochè uguali nel tempo. La figura mostra un esempio dei risultati ottenuti:

```
mean1=1.036378, mean2=1.301819
variance1=9.173406, variance2=9.860107
```

Viene anche determinata l'autocorrelazione, l'immagine seguente mostra l'autocorrelazione presente nella serie storica del primo piano.



L'immagine seguente mostra anche la differenza di primo ordine dei valori.



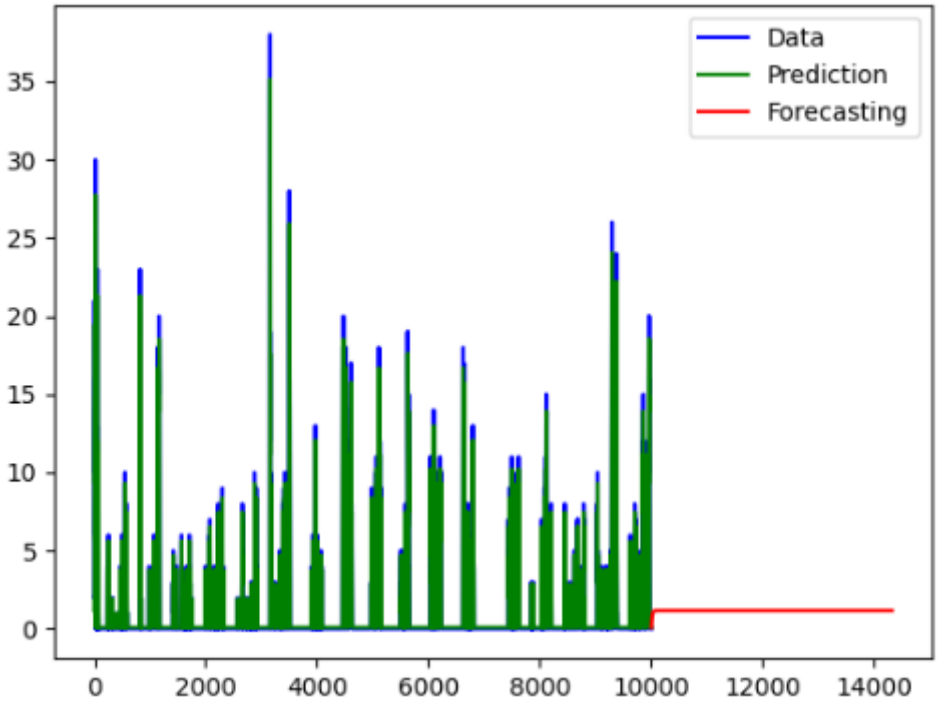
Determinare questi aspetti è essenziale al fine di capire se è necessario andare a fare delle operazioni alla serie prima di applicare i modelli. Una volta valutati i risultati si sceglie di non effettuare modifiche alla serie con le apposite tecniche.

Modelli per la previsione e il forecasting

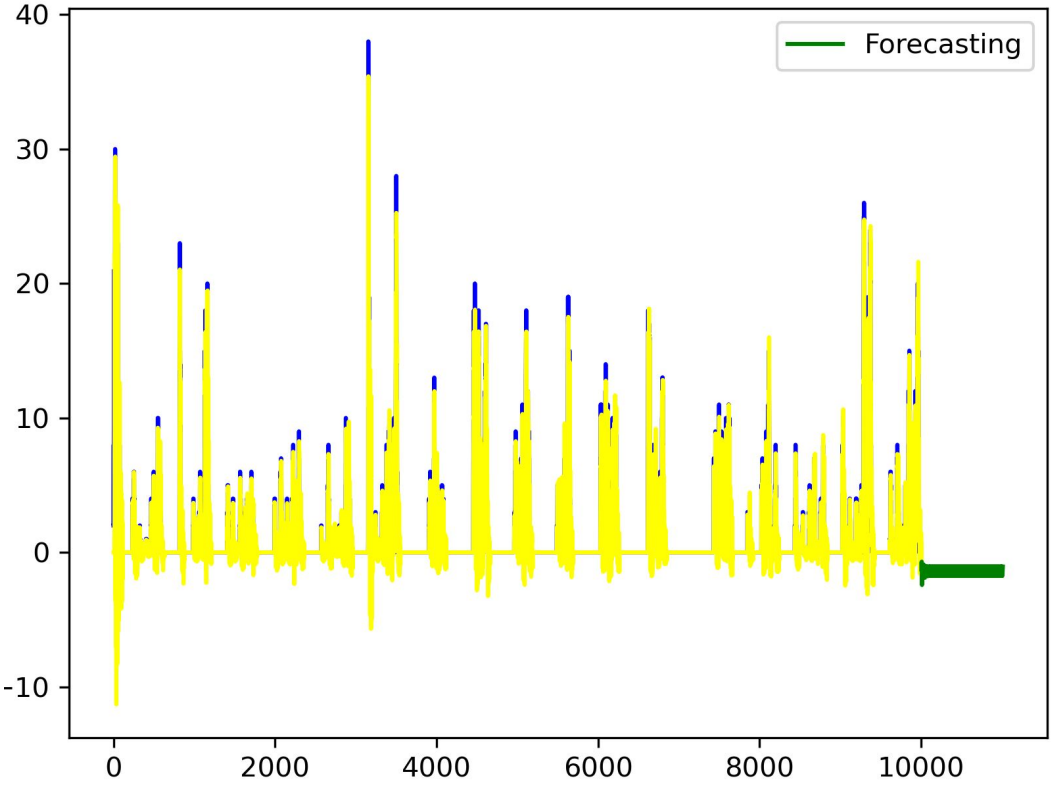
In questa fase vengono realizzati alcuni modelli di diverso tipo per la fase di previsione e di forecasting.

Modelli statistici

Il fatto che la serie non abbia stagionalità ci porta a scegliere ARIMA come metodo statistico da applicare. Come ci si aspettava però il modello ARIMA non è abbastanza potente per il forecasting mentre per la predizione dei dati in-sample il modello si comporta discretamente.



Su consiglio del Prof.re Maniezzo ci viene proposto di provare con SARIMAX, i risultati sono simili a quelli ottenuti con ARIMA.



Machine Learning

RandomForest

Come successivo modello si è provato un modello di machine learning come RandomForest anche in questo caso però ci sono stati dei problemi legati alla complessità dei dati intesa come dimensione che causava una elaborazione onerosa e non sostenibile da una macchina locale. Si è provato ad addestrare il modello RandomForest con 10 alberi(valore molto piccolo di prova) per circa 6 ore, stampando le singole previsioni, i risultati mostravano comunque un discreto errore, si è così deciso di lasciar perdere con questo modello perchè privi di macchine capaci di sostenere un'elaborazione così esosa.

Deep Learning

MLP

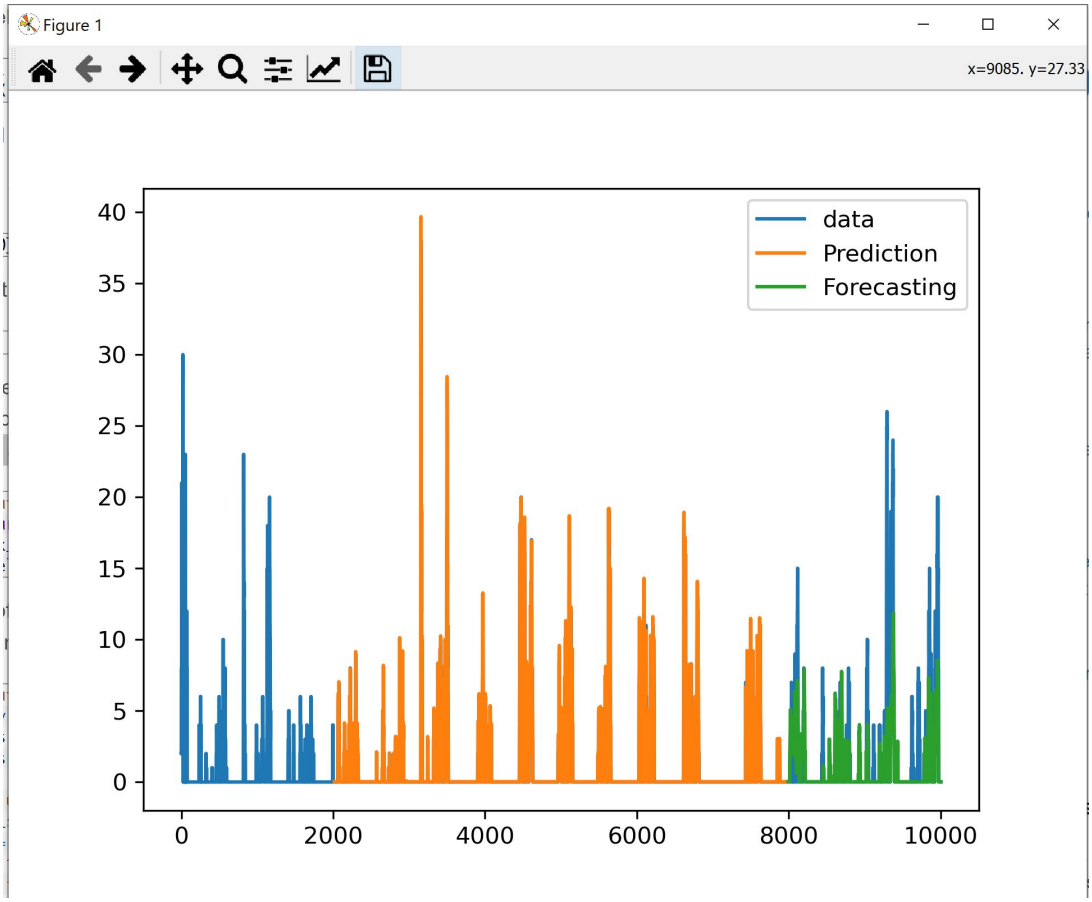
Passando ai modelli di deep learning il multi layer perceptron (MLP) è stato il modello che ha portato dei risultati migliori in termine di previsione e forecasting con un giusto tempo di addestramento. Il modello viene addestrato sui dati di train set che rappresentano l'insieme dei dati fino alla parte restante dedicata al test set configurabile con i parametri disponibili nel file constans.py.

La struttura della rete presenta un input di un numero di input dato dalla variabile **look_back** che indica quanti dati considerare per la finestra temporale. Successivamente gli hidden layer sono costituiti da layer che vanno dai 128 a 32 secondo la classica potenza di 2. La funzione di attivazione che viene utilizzata è relu in quanto è solitamente quella più efficace evitando il problema del vanishing gradient.

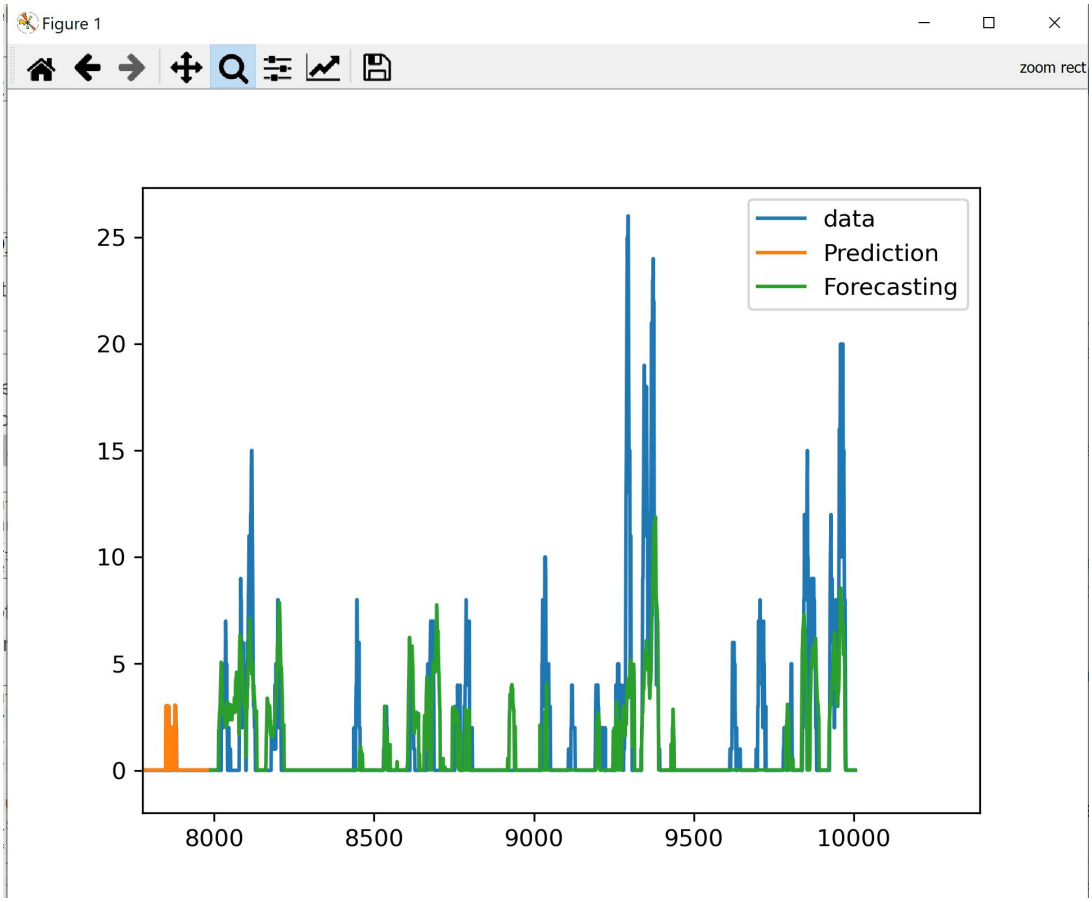
L'ultimo layer presenta solo un neurone di output.

Come funzione di loss viene utilizzato il mean squared error (MSE) e come ottimizzatore ADAM.
Il modello infine viene addestrato utilizzando 500 epoche con un batch size definito a 32 di default.

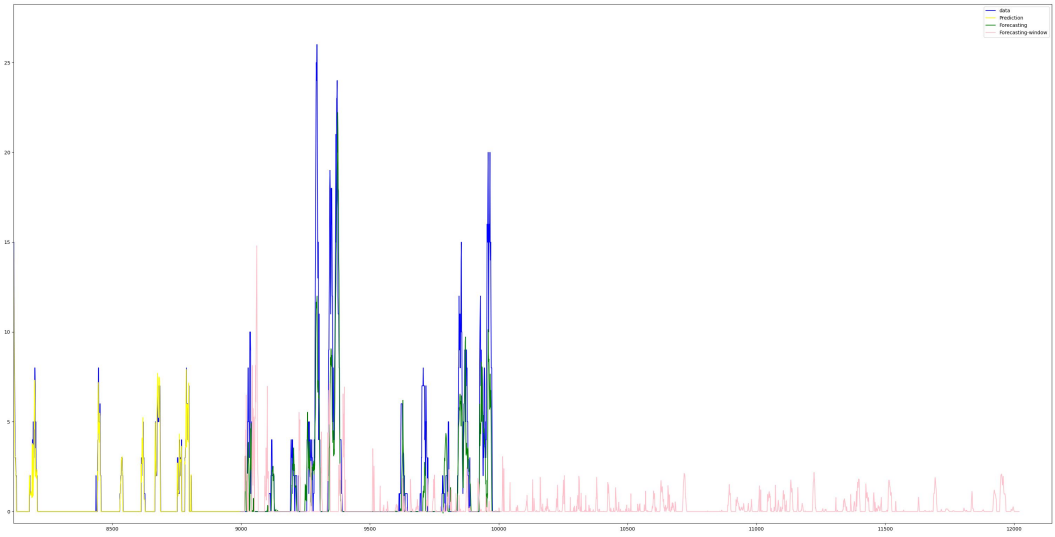
L'addestramento porta a una loss sul train decisamente buona ma questo è solito aspettarselo.
Mentre sulla parte di test l'errore è più rilevante ma comunque buono.



Ponendo il focus sulla predizione bi-settimanale si possono vedere meglio i risultati nella figura seguente.



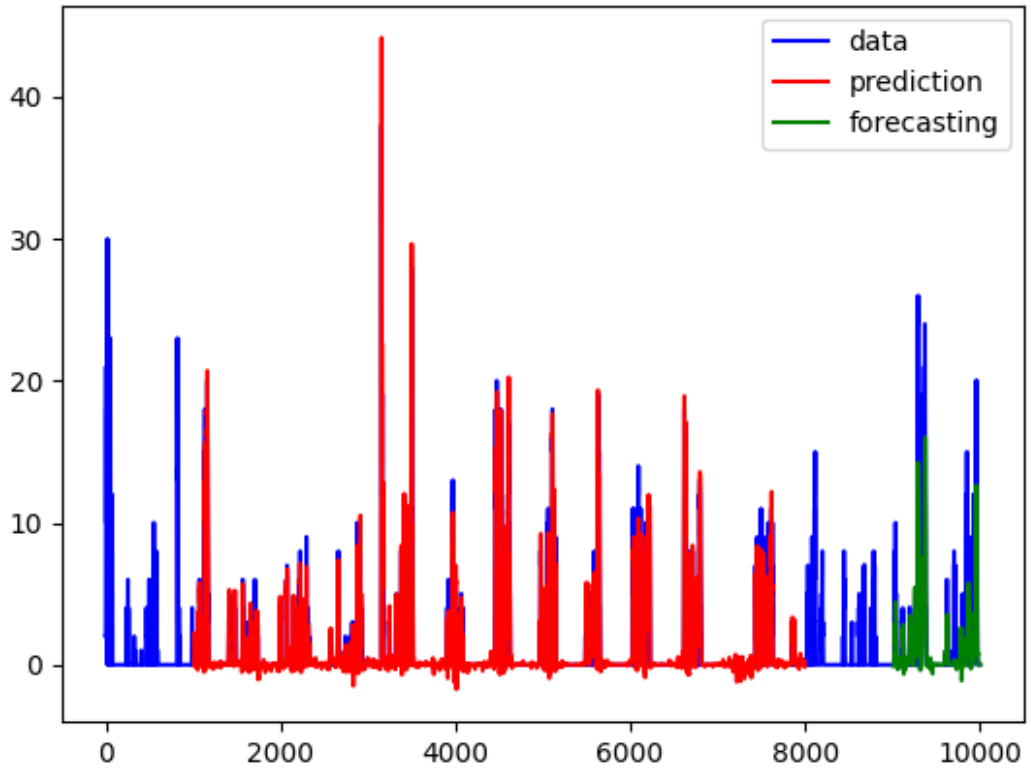
Inoltre come richiesto abbiamo provato a utilizzare una window anche al di fuori dei dati (out-of-sample) per cercare di predire l'andamento oltre ai dati forniti, l'andamento non è ottimale ma essendo che i dati a occhio e attraverso analisi hanno un andamento difficilmente prevedibile non è da considerare nemmeno pessimo in quanto rispecchia comunque l'andamento della funzione nei momenti di crescita e di decrescita.



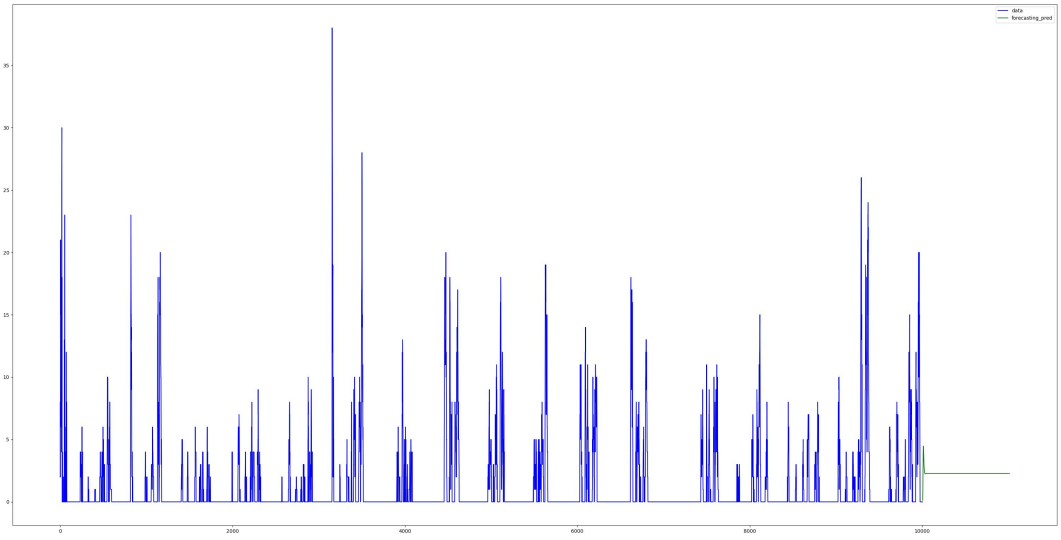
LSTM

L'ultimo modello neurale utilizzato è stato LSTM che ha riscontrato risultati inferiori ad MLP anche se sinceramente ci si aspettava un incremento delle prestazioni in quanto le RNN (Recurrent Neural Networks) sono utilizzate principalmente per problemi come le serie temporali in cui è importante la "memoria".

In questo caso la struttura della rete presenta 10 neuroni o celle LSTM con una loss uguale al modello MLP e otimizzatore ADAM. La rete viene addestrata solo utilizzando 10 epoche dal momento che le RNN come le reti convoluzionali richiedono più calcoli delle normali reti deep.



Anche qui si è provato a utilizzare una window oltre ai dati con risultati molto inferiori alla rete MLP.



Conclusioni

Abbiamo trovato questo progetto molto interessante perchè basato su dati reali; questo ha certamente complicato le cose rendendo lunghe e complesse le elaborazioni ma ci ha permesso di applicare i modelli statistici e neurali ad un caso reale. Negli esempi visti a lezione o nei dataset che si trovano in rete le elaborazioni sono molto più facili perchè presentano già delle caratteristiche marcate come stagionalità, ecc.. . I risultati raggiunti hanno un'accuratezza molto buona e ciò gratifica molto i nostri sforzi. Riteniamo di aver svolto un buon lavoro. Siamo sicuri che ci sia ancora molto margine di miglioramento sopprattutto la dove le nostre macchine mostravano segni di fatica a livello computazionale, certamente con macchine adatte all'addestramento moli di dati come queste non creerebbero complicazioni e sarebbe possibile produrre risultati efficienti in tempi ragionevoli. Confrontando i risulti ottentuti tramite forecasting con i nuovi dati registrati al museo risulta evidente una componente di errore, ma è possibile anche notare che il trend serie viene spesso indovinato correttamente. In conclusione ci riteniamo molto soddisfatti del lavoro svolto e dei risultati ottenuti.