Progetto Metodi Quantitativi per l’informatica

Indice

1. Descrizione del progetto

Ogni anno molti voli vengono cancellati o subiscono ritardi per le più svariate ragioni.
Questi ritardi non sono soltanto un danno per i viaggiatori ma anche per le compagnie aeree.
Bisogna considerare che alcune delle motivazioni di cancellazioni/ritardi dei voli sono puramente casuali mentre altre possono essere predette tramite lo studio dei dati delle compagnie aeree.
Quello che proponiamo è uno studio dei dati delle compagnie aeree atto a costruire un modello che ci permetta di predire il ritardo di un volo in base a determinate caratteristiche del volo stesso.

2. Analisi del problema

L'andamento dei ritardi aerei è senz'altro legato al particolare periodo dell'anno preso in considerazione, motivo per cui sarà intuitivamente più probabile subire un ritardo in un periodo di vacanza, come Natale o agosto, ma esso è anche legato a tutti i ritardi subiti nell'intervallo di tempo precedente al momento del volo. Basti pensare ad esempio ad una particolare condizione di maltempo che causa disagi in un aeroporto per delle ore consecutive, se non addirittura per dei giorni.
Proprio per questa caratteristica di dipendenza temporale, abbiamo pensato che avremmo potuto utilizzare una rete neurale ricorsiva, in particolare una rete LSTM, che non solo può imparare le ricorrenze annuali particolari (come i giorni di festa), ma che catturi anche le particolari dipendenze in archi di tempo successivi tra loro.
La nostra idea iniziale era quella di poter consigliare la compagnia aerea che si prevede faccia meno ritardi in un determinato giorno, su una determinata tratta. Il problema di questa soluzione consisteva nella complessità di impostazione del problema, visto e considerato che, in letteratura, abbiamo visto che le LSTM vengono utilizzate per predire un valore di un determinato dato e non siamo riusciti a trovare casi analoghi in cui venivano analizzate delle probabilità con cui confrontarci. Per questa problematica e per la difficoltà nel manipolare la grande mole di dati iniziali in relazione alle nostre capacità computazionali (circa 12 GB di memoria in cui erano memorizzati i voli statunitensi di 4 anni), abbiamo deciso di restringerci a un sottoinsieme di voli (nel nostro caso abbiamo analizzato il traffico aereo dallo stato di New York alla California) e, anziché cercare di predire quale compagnia performa meglio, abbiamo deciso di predire il ritardo di un determinato volo (senza eliminare in ogni caso l'informazione della compagnia aerea, in quanto alcune potrebbero avere spesso più ritardi di altre) sulla base dei voli nelle 24 ore precedenti.

3. Dataset

Il dataset è stato preso dal "Bureau of Transportation Statistics" che contiene le informazioni riguardo tutti i voli commerciali dal 1987 ad oggi ma per avere dati gestibili in quanto a dimensioni e recenti si è scelto di ridurre l'intervallo dal 2014 al 2017.
Il dataset è scaricabile dal sito mese per mese ed ha 109 features.

3.1 Cleaning e features selection

La prima operazione che abbiamo fatto è stata scegliere le features più importanti/adatte/pregnanti per il nostro problema ed eliminare le superflue in modo da avere dati più compatti e gestibili dai nostri calcolatori.
Le features sono state ridotte da 109 iniziali a 21 finali con una riduzione considerevole della pesantezza del dataset.

3.2 Features

Year,Month,DayOfMonth, DayOfWeek	Informazioni sulla data del volo
AirlineID	ID della compagnia aerea
OriginAirportID, OriginState	Dati su aeroporto di partenza
DestAirportID, DestState	Dati su aeroporto di arrivo
CRSDepTime, DepTime,DepDelayMinutes	Ritardo alla partenza
CRSArrTime, ArrTime, ArrDelayMinutes	Ritardo all'arrivo
Distance Group	Variabile che indica l'appartenenza a intervalli di distanze
CarrierDelay	Ritardo causato dal velivolo
WeatherDelay	Ritardo causato dal tempo
NASDelay	Ritardo causato
SecurityDelay	Ritardo causato da rallentamenti ai controlli di sicurezza
LateAircraftDelay	Ritardo delle coincidenze

3.3 Unione dei dati

I dati sono stati uniti utilizzando le funzioni messe a disposizione dalla libreria Pandas in modo da avere csv per anno e non per mese.
Alla fine abbiamo ottenuto un csv unico contenente tutti i voli dal 2014 al 2017, in cui abbiamo considerato unicamente i voli che partivano dallo stato di New York e che arrivavano nello stato della California. Abbiamo anche fatto dei test sui dati di un singolo anno, considerando tutte le tratte, ma alla fine abbiamo deciso di restringerci al caso di studio su un'unica tratta tra i due stati citati nell'arco temporale dei quattro anni.

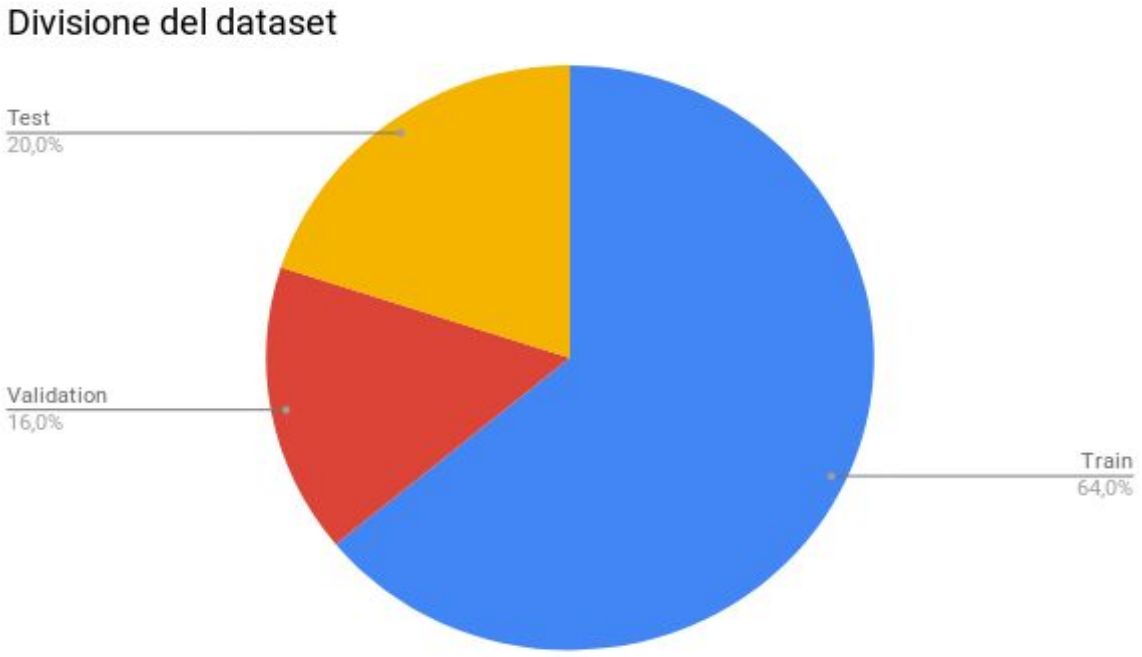
3.4 Divisione in training set e test set

Il dataset è stato splittato con due approcci diversi:

1)Il dataset viene prima shufflato e poi diviso in 80% train e 20 % test. Il 20% del training set compone il validation set. Dopo aver suddiviso l'intero dataset, i singoli train set e test set vengono riordinati seguendo l'ordine temporale dei voli.

2)Il dataset non shufflato viene diviso in train e test. Il 20% del training set compone il validation set.

Non shufflare il dataset è una scelta fatta analizzando numerosi esempi di prediction su sequenze temporali trovate in letteratura. L'obiettivo è quello di mantenere il trend presente nei dati confidando nella casualità degli stessi.



4. Rete

4.1 Scelta della rete

Analizzando casi di studio simili al nostro presenti in letteratura, abbiamo preso in considerazione l'utilizzo delle Recurrent Neural Network, poiché queste sono tra le più usate in problemi di prediction in cui le sequenze temporali dei casi precedenti hanno una grande incisione sul risultato. Difatti, ipotizzando ragionevolmente che i ritardi dei voli precedenti a un dato volo possano incidere sul ritardo dello stesso, abbiamo voluto creare una rete che catturi tali correlazioni e che individui al tempo stesso i pattern all'interno del dataset, come potrebbero essere ad esempio dei particolari disagi creati dal fatto che in alcuni momenti dell'anno il numero di passeggeri e voli può essere notevolmente più alto (si pensi ai giorni di partenza per le vacanze, in cui generalmente si hanno dei picchi di flusso aereo). Il tipo di rete RNN, pertanto, ci è sembrato adatto in quanto riesce sia a catturare le relazioni che ci potrebbero essere tra i voli precedenti e il volo preso in considerazione in una certa finestra temporale, sia ad individuare i pattern di cui abbiamo parlato in precedenza, come ad esempio i giorni di picco per le partenze.

4.2 RNN

Le RNN generalmente sono penalizzate dal problema del vanishing gradient. Idealmente si vorrebbe che la rete possa connettere le relazioni tra i dati a distanze di tempo significative, tuttavia più sono i time step utilizzati, più sono le possibilità che il gradiente della back-propagation si accumuli fino a esplodere, o che si annulli del tutto.
Data la seguente rappresentazione di una RNN:

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{V}\mathbf{h}_{t-1})$$

Dove U e V sono le matrici dei pesi che collegano rispettivamente gli input e gli output ricorsivi, se si va indietro di tre time step si avrà

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{V}(\sigma(\mathbf{U}\mathbf{x}_{t-1} + \mathbf{V}(\sigma(\mathbf{U}\mathbf{x}_{t-2})))$$

Ovvero nel lavorare a ritroso nel tempo si vanno ad aggiungere strati sempre più profondi alla rete. Considerando il gradiente dell'errore rispetto la matrice dei pesi U durante la backpropagation nel tempo

$$\frac{\partial E_3}{\partial U} = \frac{\partial E_3}{\partial out_3}\frac{\partial out_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial U}$$

Ognuno di questi gradienti consiste nel calcolare il gradiente della sigmoide, il che è un problema quando i valori in input sono tali che l'output è vicino ai valori 0 o 1, per cui il gradiente raggiunge valori molto bassi (molto spesso al di sotto dello 0.25). Questo implica che quando si vanno a moltiplicare più gradienti della sigmoide uno dopo l'altro si arriva a un gradiente $\partial E/\partial U$ che svanisce. Trattando molti time step precedenti si arriva a un gradiente praticamente nullo, quindi i pesi non si aggiornano correttamente e di conseguenza la rete non apprende nulla delle relazioni separate da un periodo di tempo significativo, il che rende le RNN non molto utili.

4.3 LSTM Layer

Per tenere conto della sequenzialità temporale dei campioni ma allo stesso tempo evitare il vanishing (/exploding) gradient abbiamo usato una rete LSTM (long-short term memory). Questa infatti, grazie alla particolare struttura interna delle sue celle, è caratterizzata da una formula diversa per il calcolo del gradiente.

4.4 Dense Layer

Uno strato totalmente connesso, che implementa l'operazione:

output = activation(dot(input, kernel) + bias)

dove activation è la funzione di attivazione element-wise passata, kernel è una matrice dei pesi creata dallo strato, e bias è un vettore di bias creato dallo strato (se abilitato).

4.5 Input e Output

I dati sono stati elaborati e dimensionati opportunamente per essere usati come input della rete. La struttura dell'input è la seguente:

	F1(t-LB)	...	F21(t-LB)	F1(t-LB+1)	...	F21(t-1)	RE(t)
1							
2							
3							
...							
...							
...							
N°samples							

Dove :

1. Fi è la feature i-esima;
2. LB è il lookback;
3. t è lo step temporale corrente;
4. RE(t) è il ritardo effettivo al tempo t;

Abbiamo anche provato a usare come input della rete un dataset con una sola feature (il ritardo dei voli precedenti) ispirandoci a numerosi casi analoghi. L'output della rete risultava peggiore rispetto al caso con più features.

L'output è un vettore di lunghezza N°samples in cui ogni cella *i* contiene il ritardo predetto dalla rete per il volo *i*.

5. Evoluzione del progetto

5.1.1 DataSet ridotti (subset)

In un primo momento abbiamo deciso di provare la rete su un dataset ridotto in modo da avere una quantità di dati gestibile più facilmente che non richiedesse un eccessivo sforzo computazionale e che consentisse un allenamento rapido.

5.1.1 DataSet unica tratta

Il dataset comprensivo dei voli americani dal 2014 al 2017 è molto corposo dunque abbiamo provato a lavorare sui dati riguardanti i voli dallo stato di New York alla California. Questa scelta si è rivelata valida poiché i dati sono risultati sufficienti per l'allenamento e questo ha consentito di svolgere il training con l'hardware a nostra disposizione.

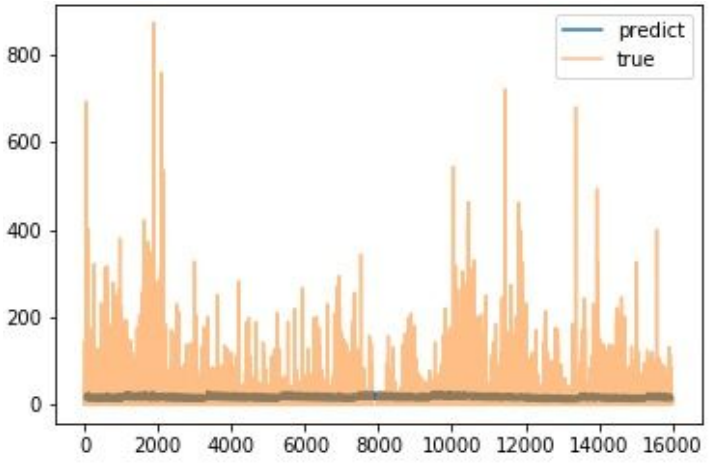
5.2 Architettura della rete

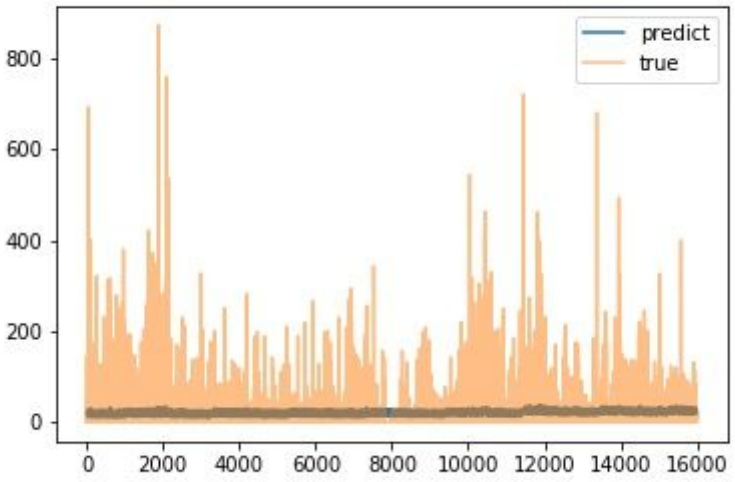
5.2.1 Strati LSTM

La rete è caratterizzata da un unico strato LSTM.

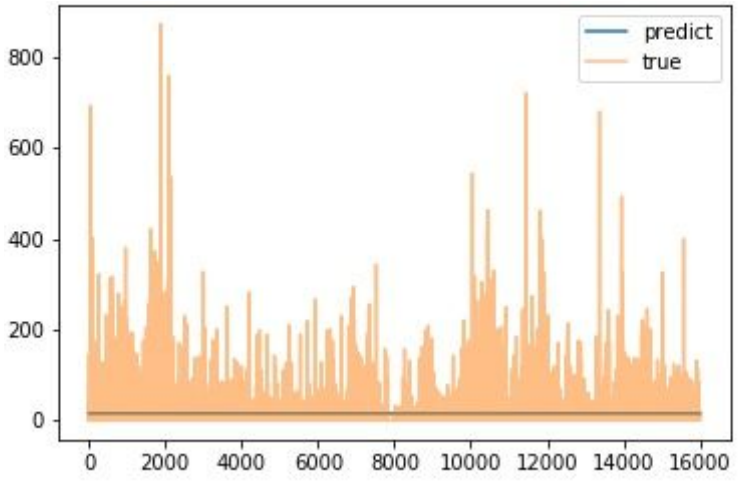
Durante le prove abbiamo anche analizzato la possibilità di aggiungerne altri; tuttavia, una volta adottata questa alternativa abbiamo riscontrato risultati peggiori di quelli ottenuti con un unico strato.

Esempi con uno strato aggiuntivo:





Con due strati aggiuntivi:



5.3 Tuning dei parametri della rete

Parametri come batch size, learning rate, numero di neuroni sono stati decisi in sede di test poiché abbiamo scelto la combinazione che portava ad una migliore predizione.Nella tabella seguente vengono riassunti i risultati dei training al variare dei seguenti parametri:

N=numero neuroni LSTM ; LB=look back
E=epoche ; BS=batch size
LR=learning rate ; A=accuracy
L=loss ; S=shuffle del dataset(1=true, 0=false)
FL=funzione attivazione LSTM (s=sigmoide, t=tanh) ;
FD=funzione attivazione Dense
O=funzione di ottimizzazione

risultati su subset ; risultati migliori ; risultati peggiori

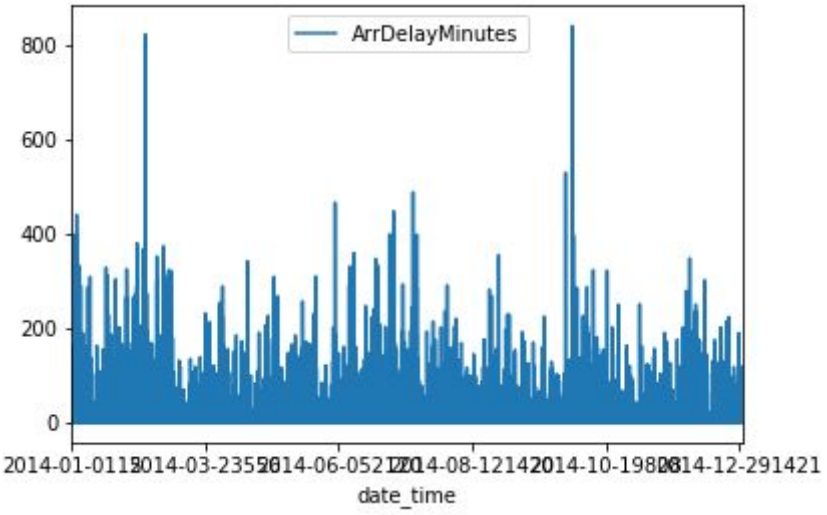
N	LB	E	BS	LR	L	S/FL/FD	O
50	70	15	400	0.01	-	1/t/-	sgd
50	70	15	1000	0.01	-	1/t/-	sgd
50	70	15	4000	0.01	-	1/t/-	sgd
50	70	15	4000	0.01	-	1/t/-	sgd
50	70	15	4000	0.01	-	1/s/-	sgd
50	70	50	4000	0.01	0.001	1/s/-	sgd
50	70	15	8000	0.01	0.001	1/s/-	sgd
50	70	15	4000	0.1	0.0044	1/t/-	sgd
50	70	15	1000	0.1	0.0038	1/t/-	sgd
50	70	20	4000	0.1	0.0044	0/t/-	sgd
50	70	15	4000	0.01	0.0039	0/t/-	sgd
50	70	70	4000	0.01	0.0039	0/t/-	sgd
50	70	70	32	0.01	0.0039	0/t/-	sgd
50	150	20	32	0.01	-	0/t/-	sgd
50	70	20	32	0.01	0.0034	0/t/l	sgd
50	70	110	32	0.01	0.0031	0/t/l	sgd
100	70	360	128	0.01	0.0032	0/t/l	sgd

N	LB	E	BS	LR	L	S/FL/FD	O
50	70	20	32	0.05	0.0035	0/t/l	sgd
50	70	20	32	0.001	0.0036	0/t/l	sgd
64	70	10	32	adam	0.0038	0/t/l	adam
64	70	10	32	adam	0.0033	0/t/l	adam
64	70	25	32	adam	0.0032	0/t/l	adam
42	70	25	32	adam	0.0032	0/t/l	adam
64	70	20	128	adam	0.0033	0/t/l	adam
25	70	15	32	0.01	0.0036	1/t/-	sgd
50	70	15	32	0.01	0.0035	1/t/-	sgd
100	70	15	32	0.01	0.0036	1/t/-	sgd
50	30	15	32	0.01	0.0035	1/t/-	sgd
50	150	15	32	0.01	0.0035	1/t/-	sgd
50	70	15	32	0.01	0.0035	0/t/-	sgd
100	70	25	32	0.01	0.0033	0/t/-	sgd
20	70	250	32	0.1	0.0031	0/t/-	sgd

5.6 DataSet non shufflato

Analizzando l'andamento dei ritardi negli anni e lavori di prediction in diversi campi applicativi abbiamo deciso di provare non shufflare i dati. Questo implica che il dataset viene diviso in train e test in maniera sequenziale. Il nostro obiettivo è quello di catturare, presi degli intervalli di qualche ora, i particolari pattern che li caratterizzano.Nei test si è notato che questa organizzazione dei dati porta ad una previsione più accurata.

Plot ritardi annuali giorno per giorno



Con il dataset non shufflato abbiamo allenato la rete con due approcci diversi:

1. ogni sequenza, prima di essere processata, viene shufflata in modo che la rete non apprenda pattern che legano le diverse sequenze ;
2. ogni sequenza viene processata dalla rete mantenendo l'ordine temporale.

Tra questi due modi di allenare la rete, il primo si è rivelato più efficace.

6. Test

6.1.1 Test con epoche, batch size e learning rate variabili.

Nei test di questa sezione rimarranno fissi il numero di neuroni (50) e l'architettura della rete(una cella LSTM e uno strato completamente connesso).

I valori del dataset prima di essere processati dalla rete vengono scalati tra -1 e 1.
La funzione di attivazione della cella LSTM è la tanh.

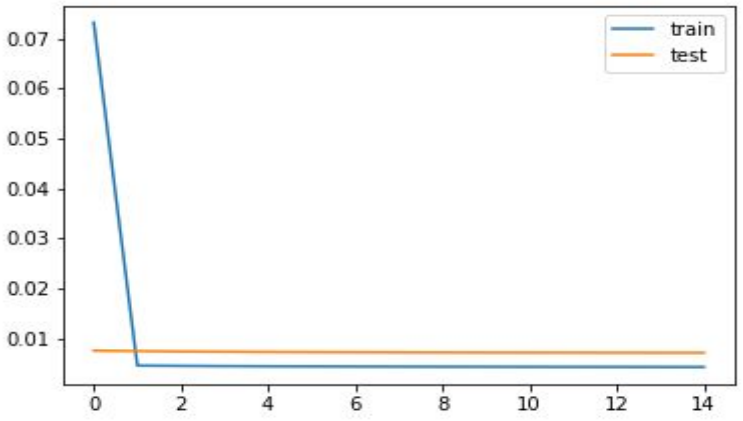
#1

DATI : batch_size = 400; epoche = 15;

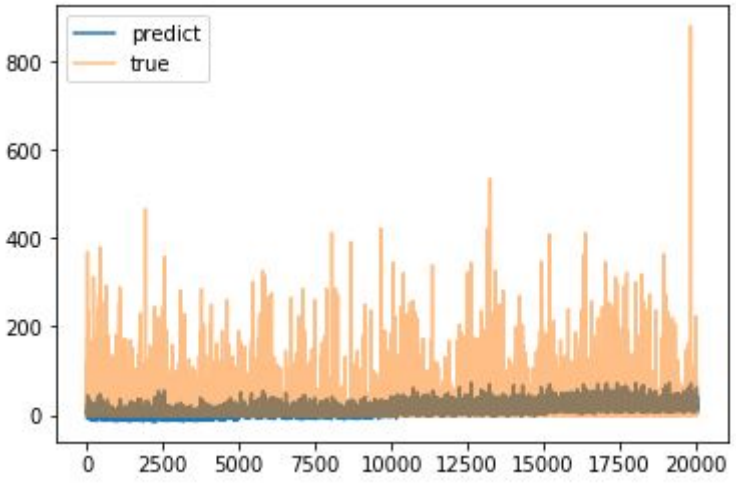
RISULTATO : accuracy = 0.6338

PLOT:

Loss



Predizione



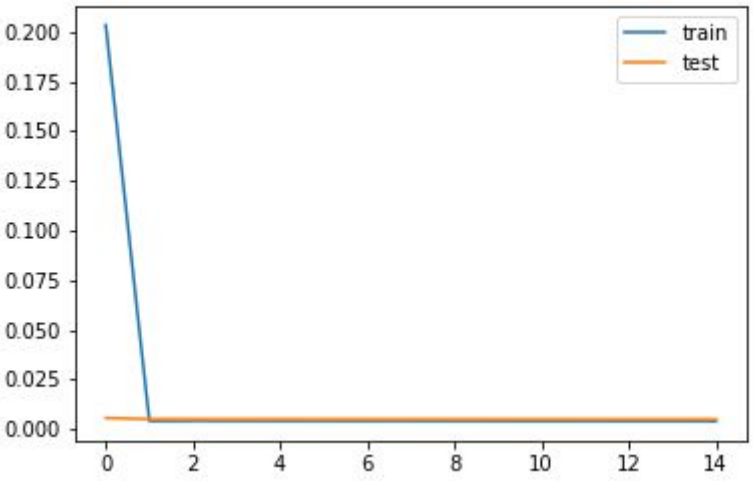
#2

DATI : batch_size = 1000; epoche = 15;

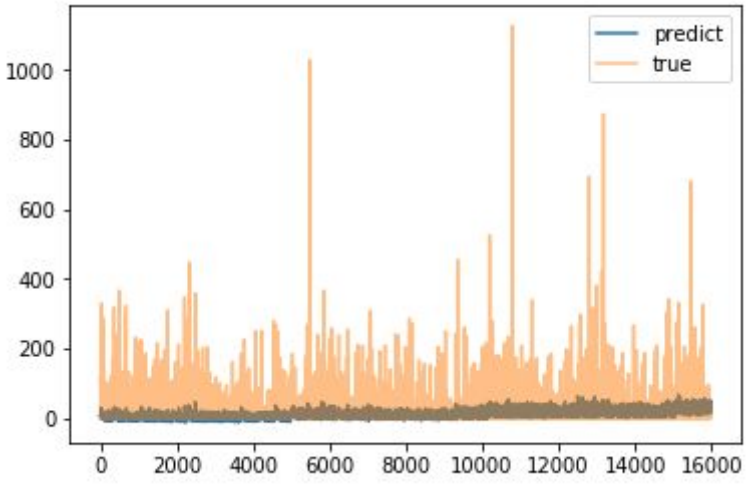
RISULTATO : accuracy = 0.6494

PLOT:

Loss



Predizione



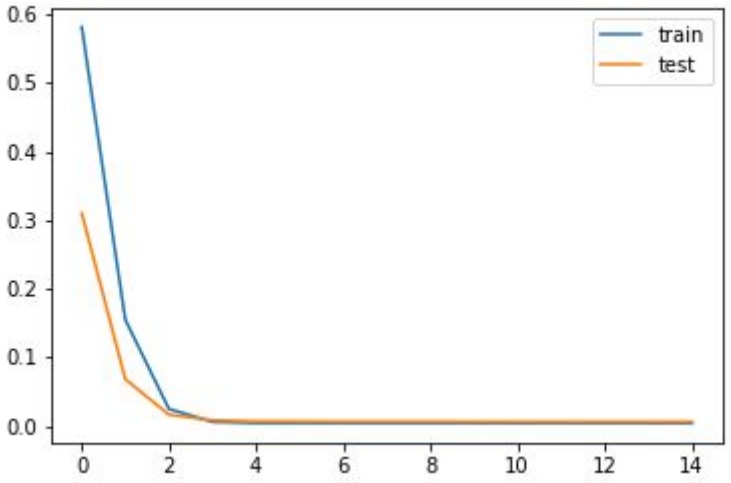
#3

DATI : batch_size = 4000; epoche = 15;

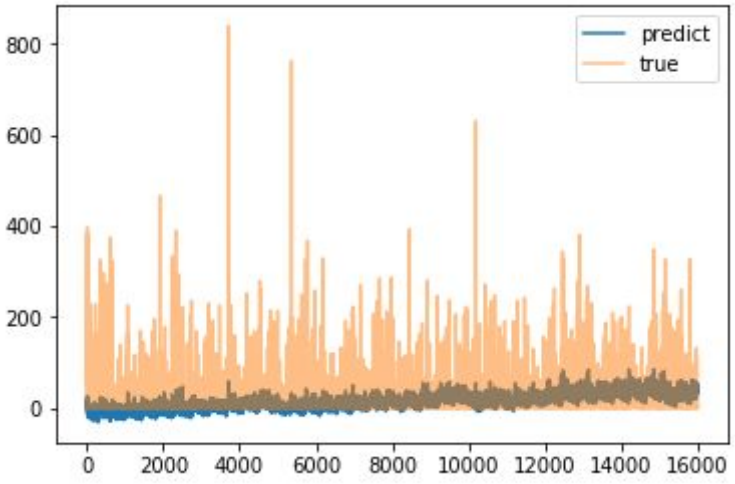
RISULTATO : accuracy = 0.6494

PLOT:

Loss



Predizione



6.1.2

Nei seguenti test i valori nel dataset vengono normalizzati e non scalati tra -1 e 1.

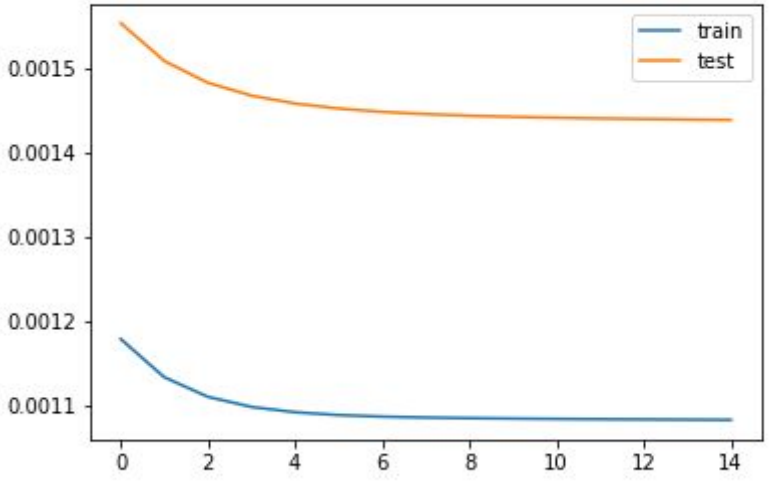
#4

DATI : batch_size = 4000; epoche = 15;

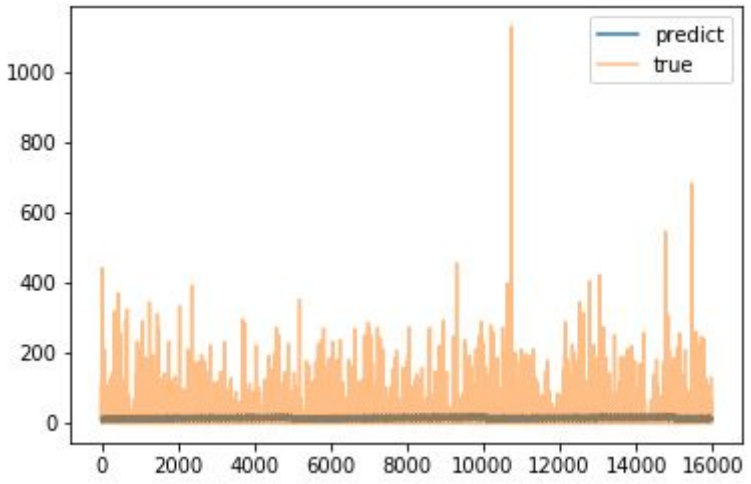
RISULTATO : accuracy = 0.6491

PLOT:

Loss



Predizione



6.1.3

I dati vengono normalizzati prima di essere processati dalla rete e viene usata la sigmoide come funzione di attivazione della cella LSTM.

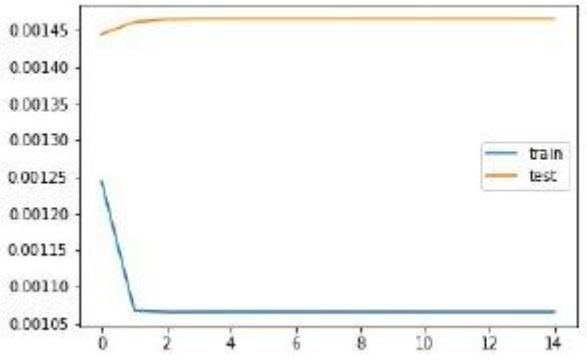
#5

DATI : batch_size = 4000; epoche = 15;

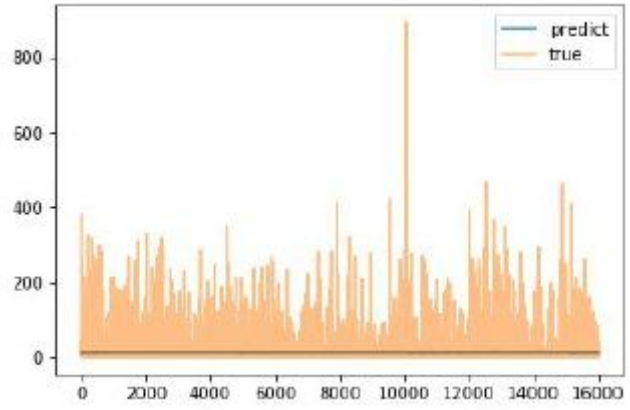
RISULTATO : accuracy = 0.6496

PLOT:

Loss



Predizione



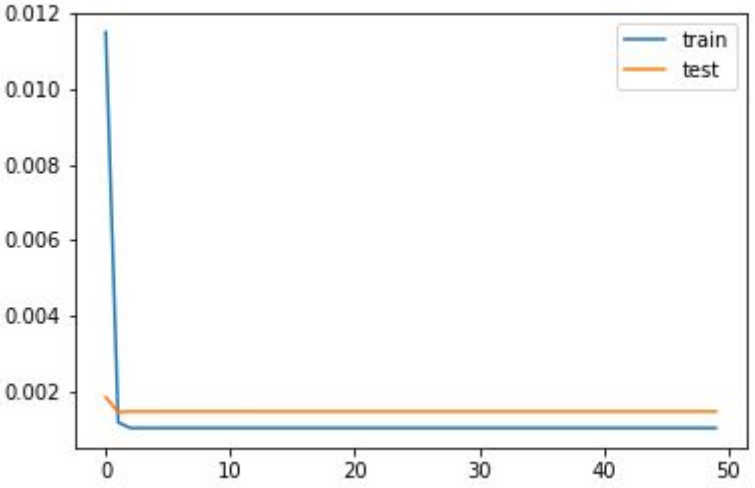
#6

DATI : batch_size = 4000; epoche = 50;

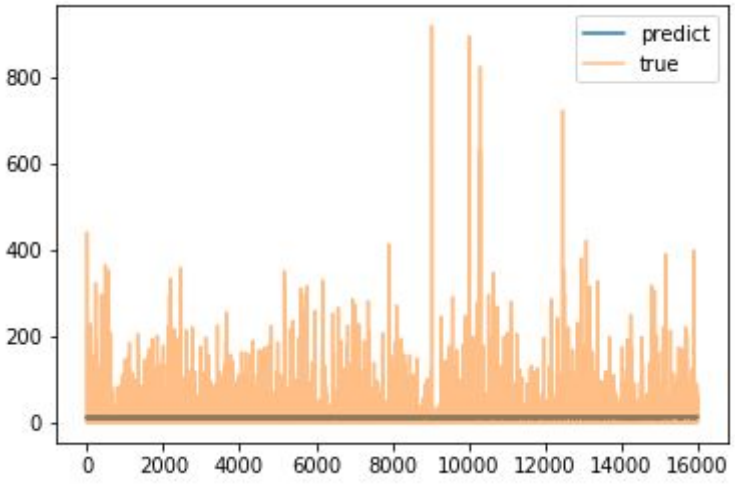
RISULTATO : loss: 0.001

PLOT:

Loss



Predizione



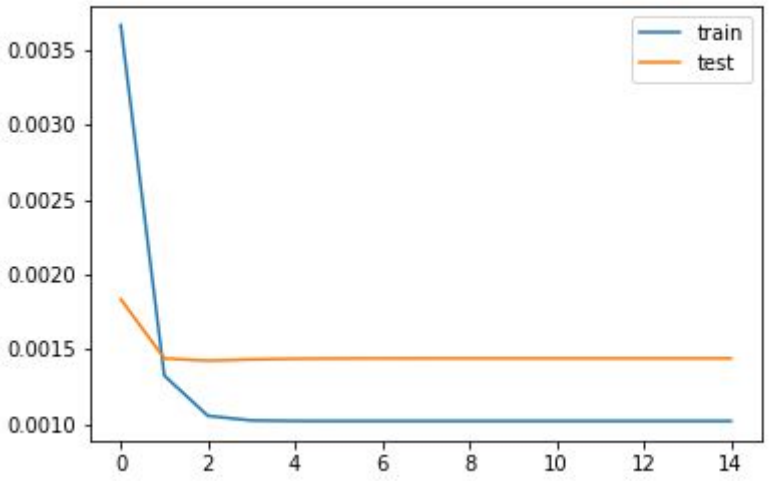
#7

DATI : batch_size = 8000; epoche = 15;

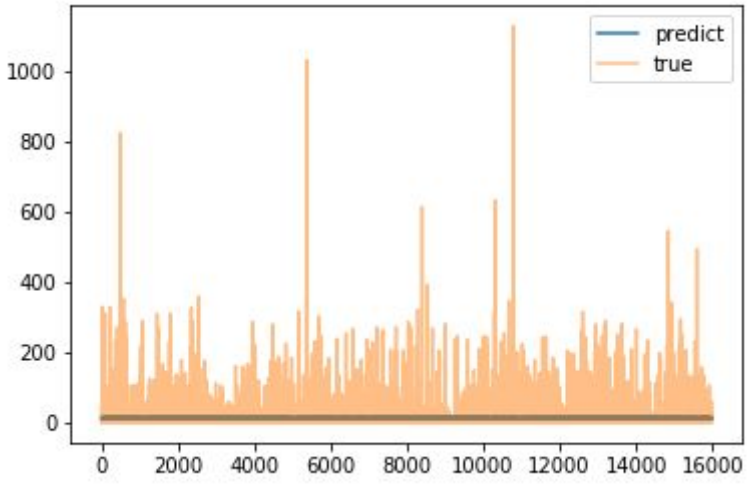
RISULTATO : accuracy = 0.6499

PLOT:

Loss



Predizione



6.1.4

I valori del dataset prima di essere processati dalla rete vengono scalati tra -1 e 1. La funzione di attivazione della cella LSTM è la tanh.
Viene modificato anche il learning rate(riferimento al codice:riga 113) :
sgd = optimizers.SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)

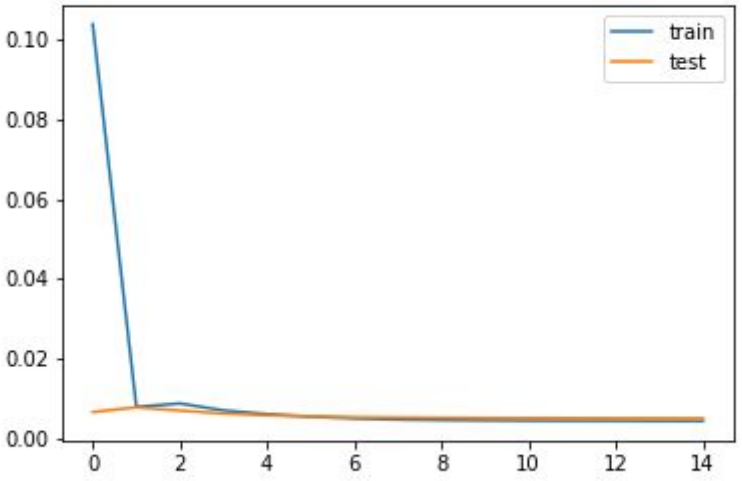
#8

DATI : batch_size = 4000; epoche = 15; learning_rate = 0.1

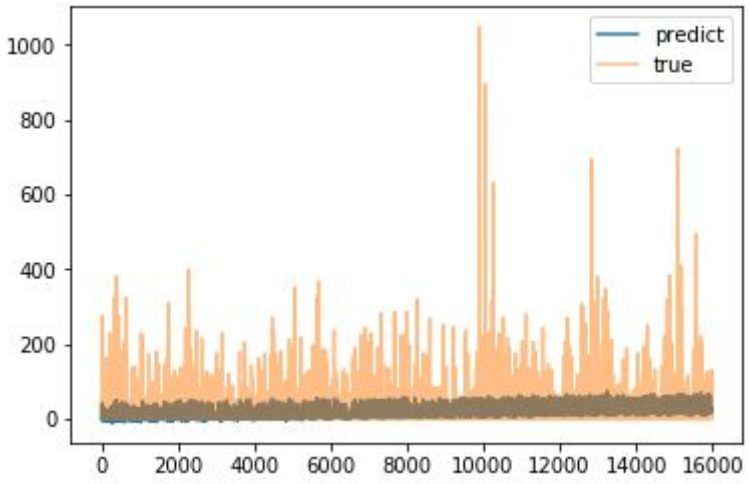
RISULTATO : accuracy = 0.6478

PLOT:

Loss



Predizione



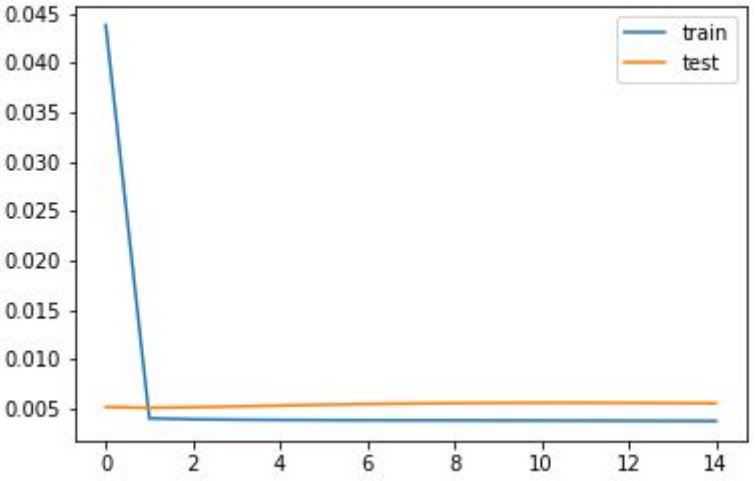
#9

DATI : batch_size = 1000; epoche = 15; learning_rate = 0.1

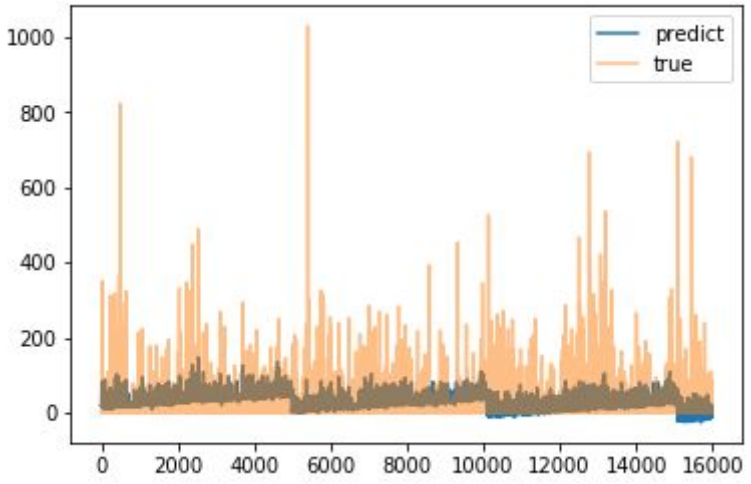
RISULTATO : accuracy = 0.6473

PLOT:

Loss



Predizione



6.1.5

Visti i numerosi problemi di prediction presenti in letteratura, si è deciso di dividere il dataset in training set e test set in modo sequenziale, ovvero non shufflando.
Dove non specificato, si ha look back = 70, batch size=32, learning rate=0.01, 50 neuroni dello strato LSTM e 1 dello strato totalmente connesso

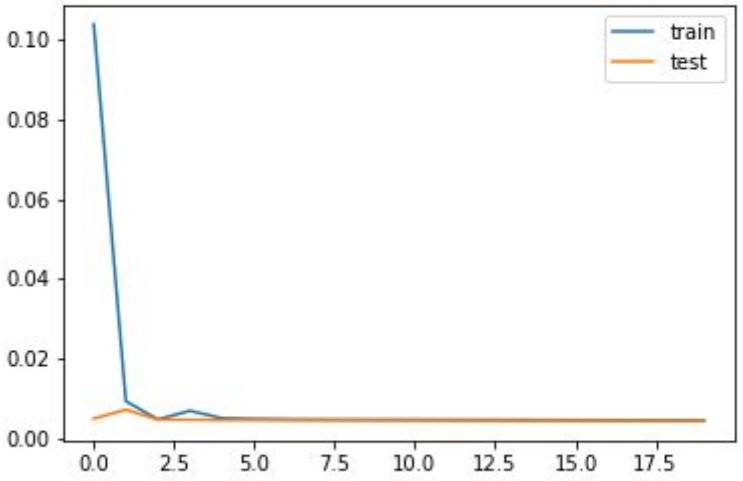
#10

DATI : batch_size = 4000; epoche = 20; learning_rate = 0.1

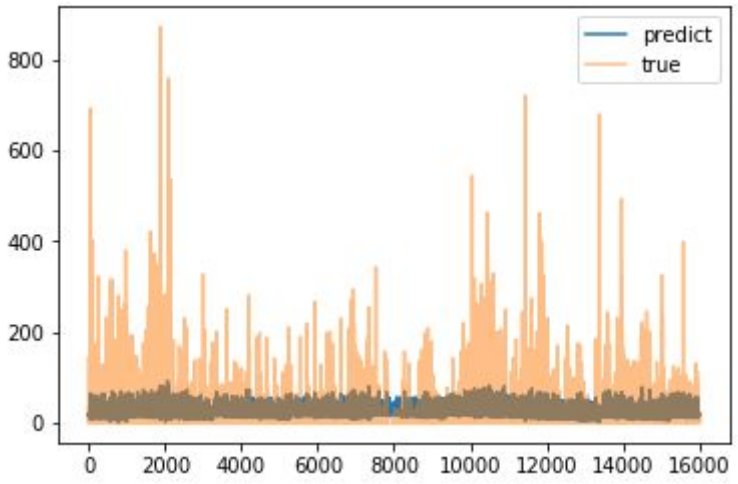
RISULTATO : accuracy = 0.6406

PLOT:

Loss



Predizione



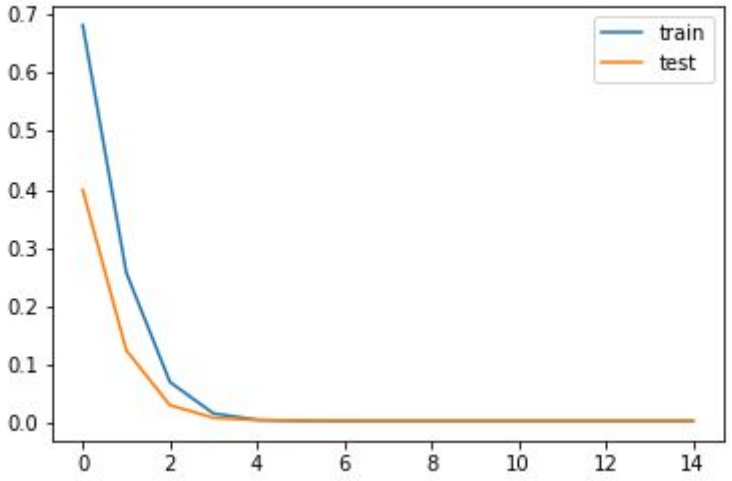
#11

DATI : batch_size = 4000; epoche = 15;

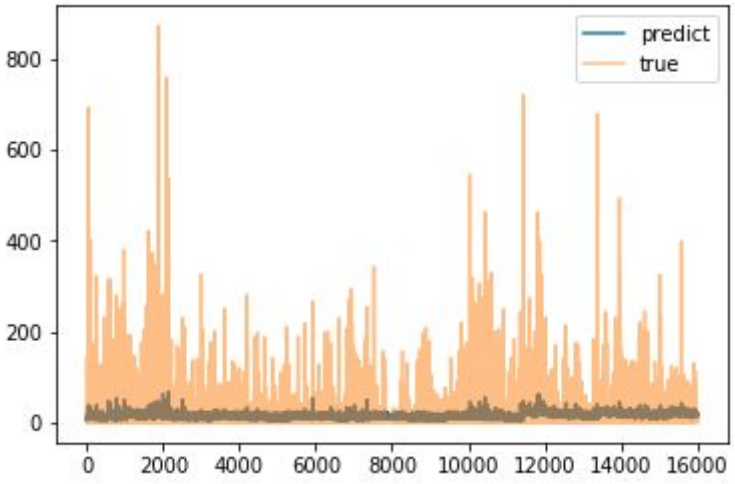
RISULTATO : accuracy = 0.6496

PLOT:

Loss



Predizione



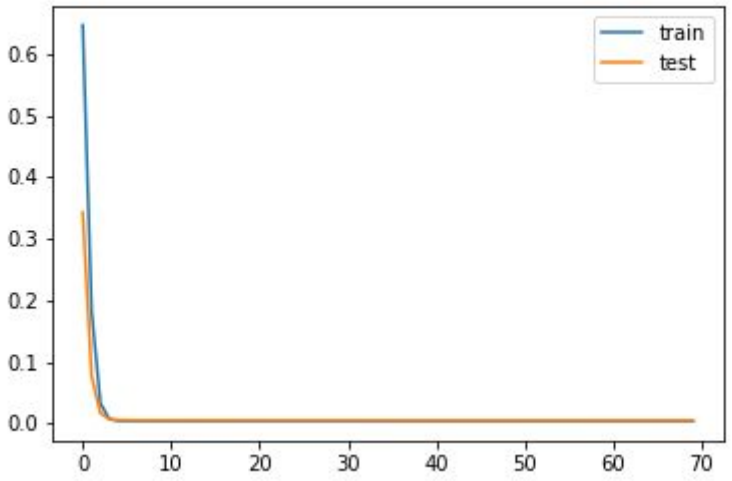
#12

DATI : batch_size = 4000; epoche = 70;

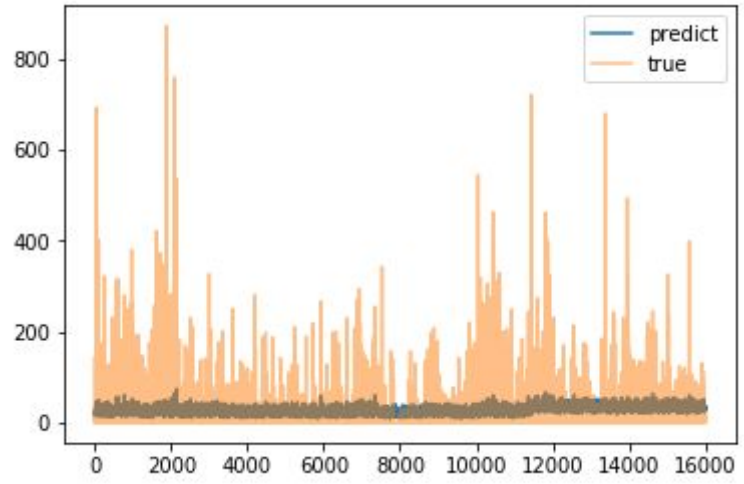
RISULTATO : accuracy = 0.6406

PLOT:

Loss



Predizione



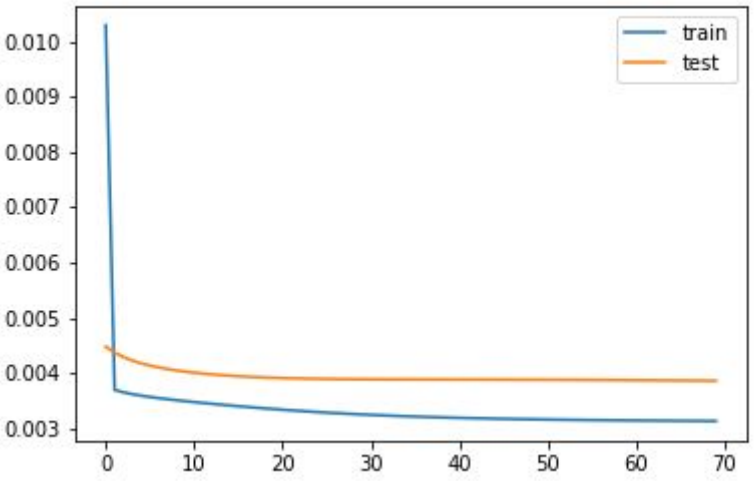
#13

DATI : epoche = 70; batch_size = 32;

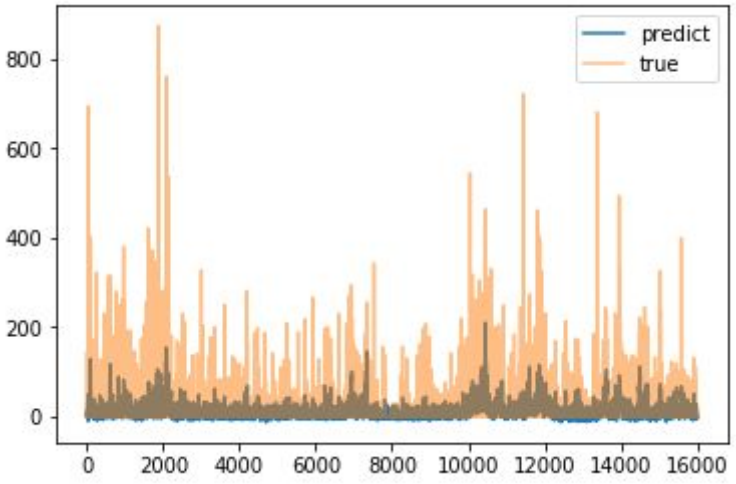
RISULTATO : accuracy = 0.6496

PLOT:

Loss



Predizione



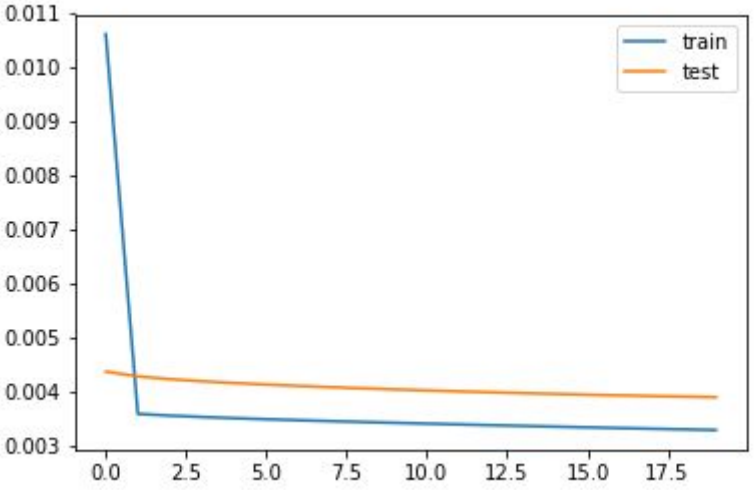
#14

DATI : epoche = 20; look_back = 150;

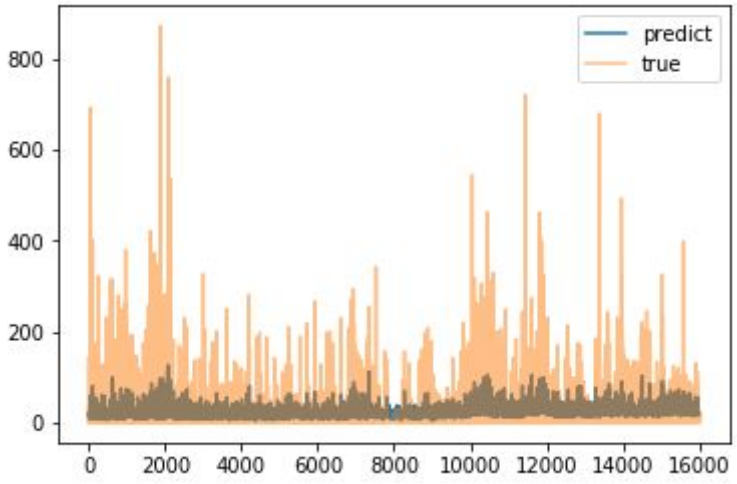
RISULTATO : accuracy = 0.6414

PLOT:

Loss



Predizione



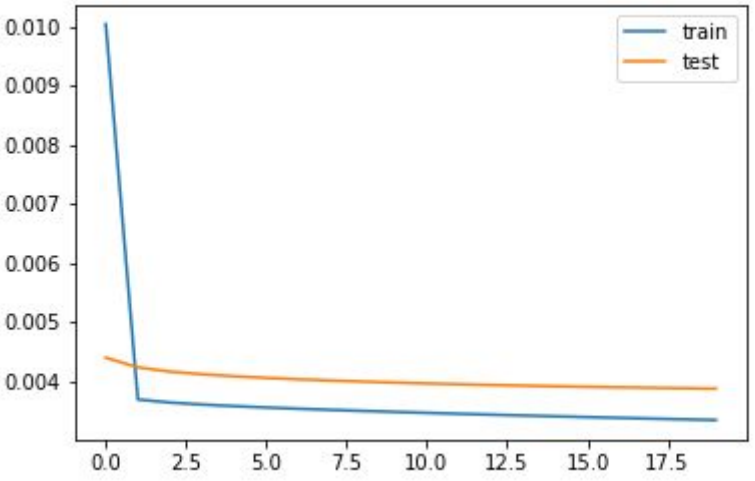
#15

DATI : epoche = 20; funzione di attivazione in dense= linear;

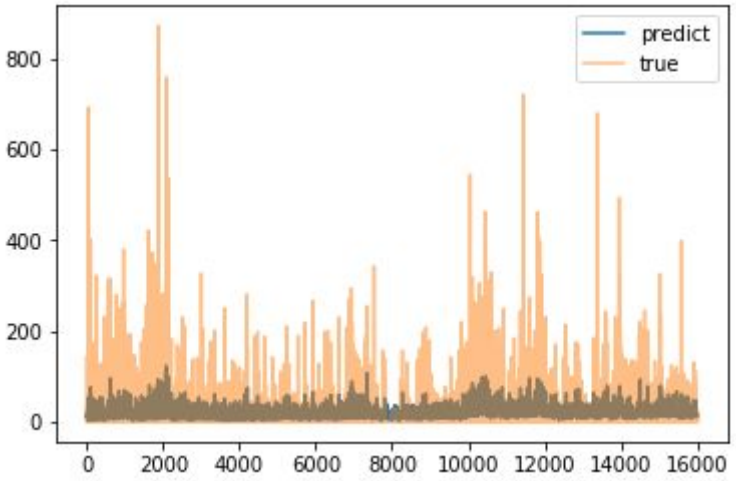
RISULTATO : loss: 0.0034; acc: 0.6406; val_loss: 0.0039; val_acc: 0.6818;
RMSE= 38.125

PLOT:

Loss



Predizione



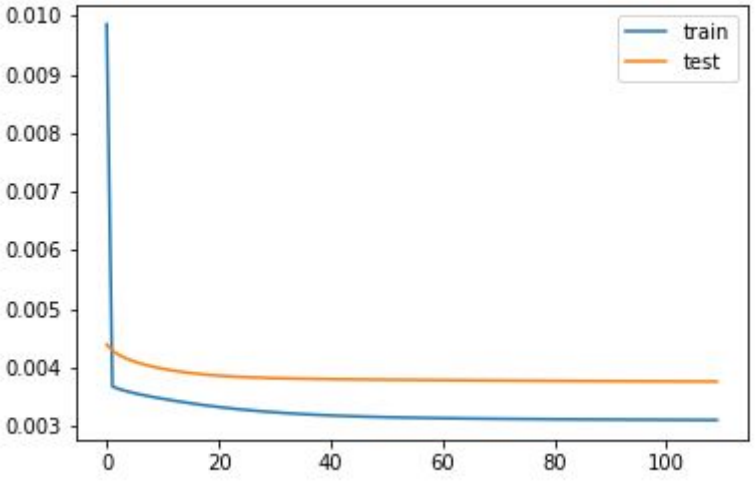
#16

DATI : epoche = 110; funzione di attivazione in dense= linear;

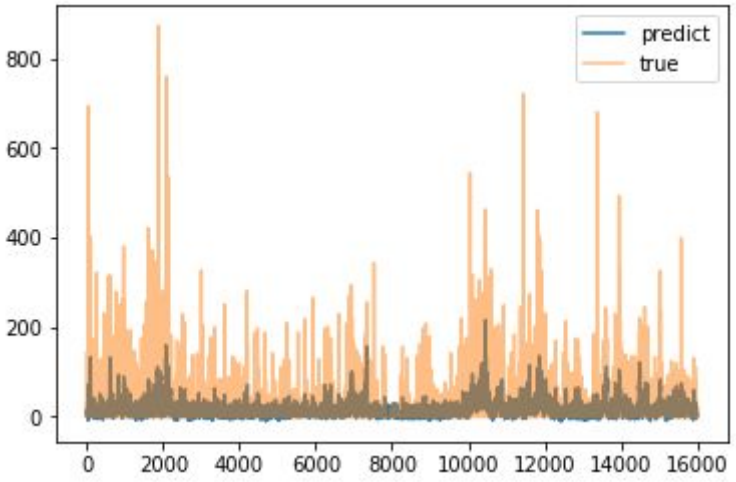
RISULTATO : loss: 0.0031; acc: 0.6406; val_loss: 0.0038; val_acc: 0.6818;
RMSE= 36.941

PLOT:

Loss



Predizione



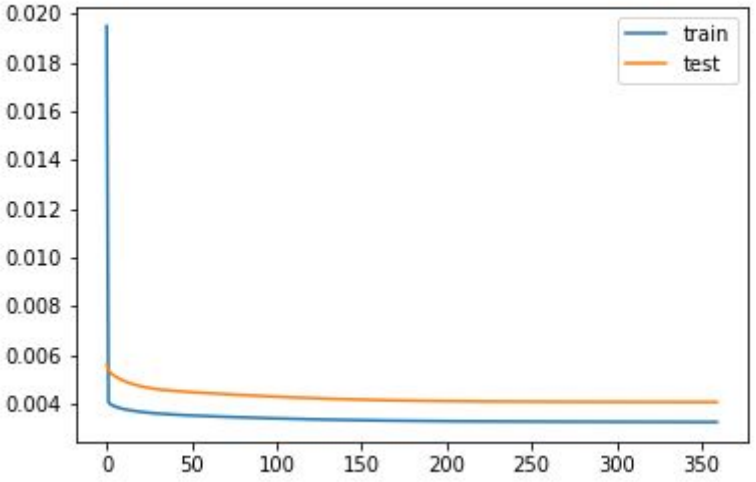
#17

DATI : LSTM_NEURONS = 100; batch_size = 128; epoche = 360; funzione di attivazione in dense= linear;

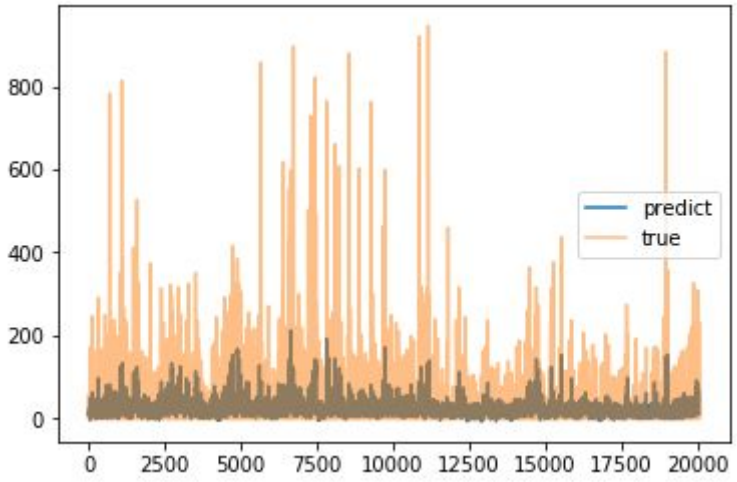
RISULTATO : loss: 0.0032; acc: 0.6485; val_loss: 0.0041; val_acc: 0.5721;
RMSE= 43.721

PLOT:

Loss



Predizione



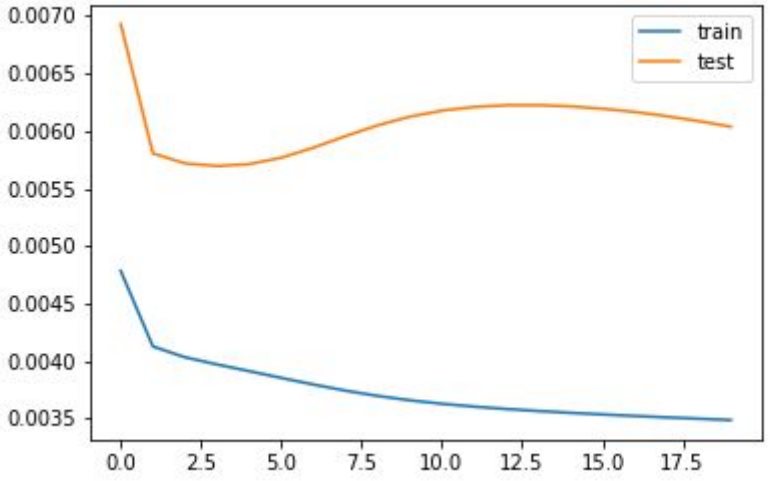
#18

DATI : epoche = 20; learning rate=0.05; funzione di attivazione in dense= linear;

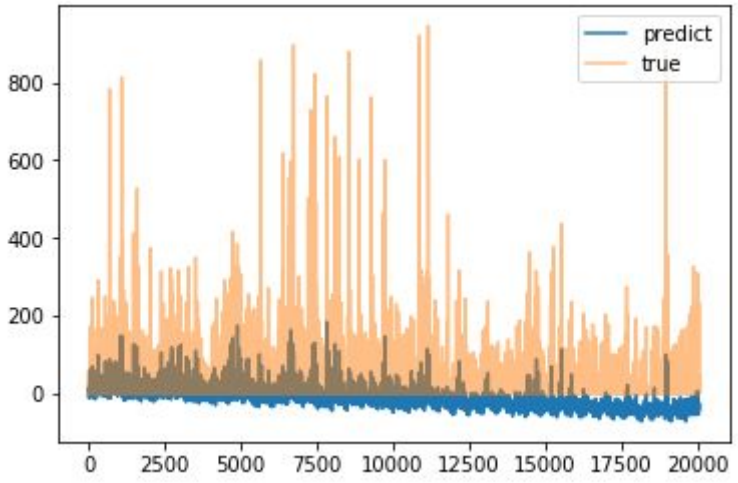
RISULTATO : loss: 0.0035; acc: 0.6485; val_loss: 0.0060; val_acc: 0.5721;
RMSE= 56.929

PLOT:

Loss



Predizione



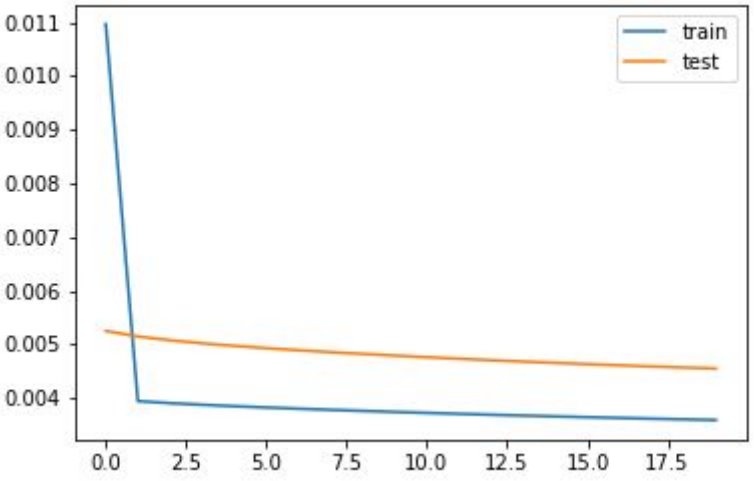
#19

DATI : epoche = 20; learning rate=0.001; funzione di attivazione in dense= linear;

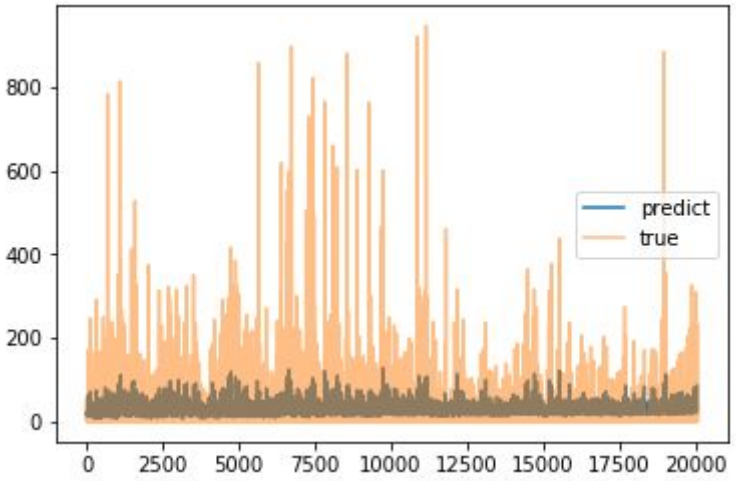
RISULTATO : loss: 0.0036; acc: 0.6485; val_loss: 0.0046; val_acc: 0.5721;
RMSE= 45.719

PLOT:

Loss



Predizione



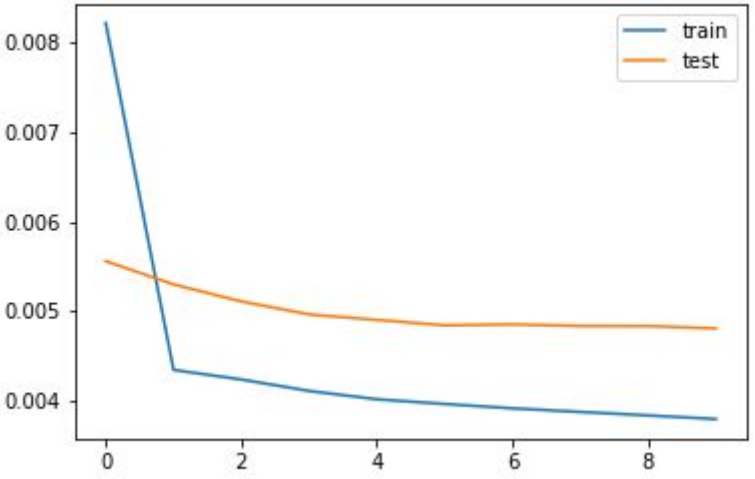
#20

DATI : neuroni= 64; epoche=10; funzione di attivazione in dense= linear;
funzione di ottimizzazione LSTM= adam;

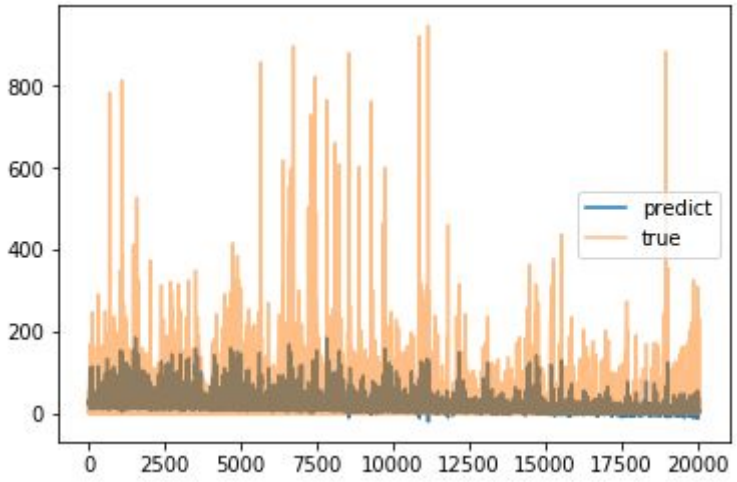
RISULTATO : loss: 0.0038; acc: 0.6485; val_loss: 0.0048; val_acc: 0.5721;
RMSE= 44.313, Train RMSE: 35.960

PLOT:

Loss



Prediction



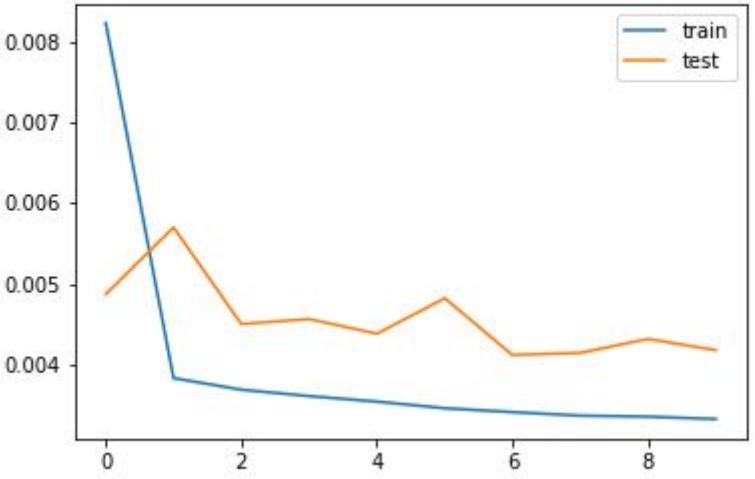
#21

DATI : neuroni= 64; epoche=10; funzione di attivazione in dense= linear;
funzione di ottimizzazione LSTM= adam; Shufflati i dati nel model.fit

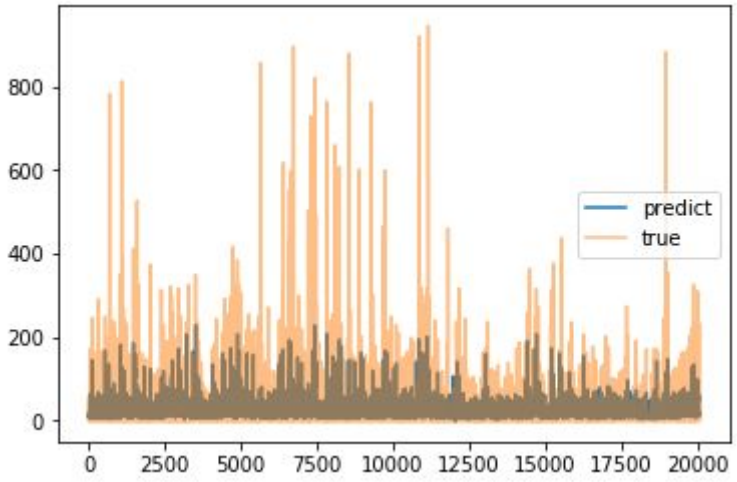
RISULTATO : loss: 0.0033; acc: 0.6485; val_loss: 0.0042; val_acc: 0.5721;
RMSE= 44.257, Train RMSE: 33.135

PLOT:

Loss



Predizione



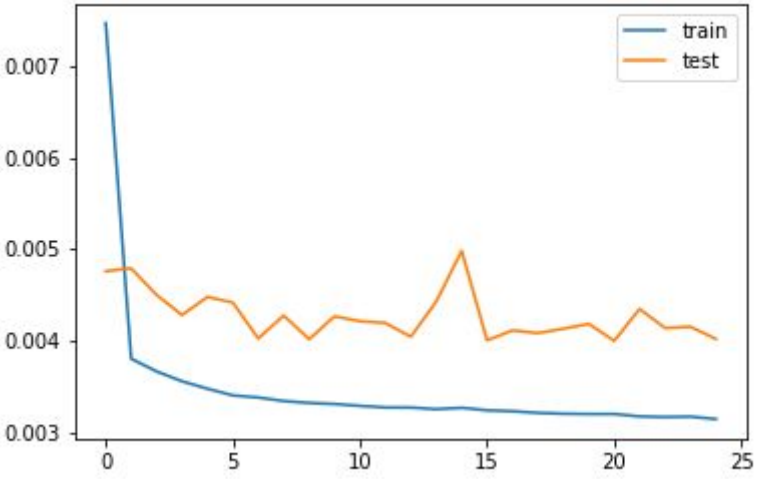
#22

DATI : neuroni= 64; epoche=25; funzione di attivazione in dense= linear;
funzione di ottimizzazione LSTM= adam; Shufflati i dati nel model.fit

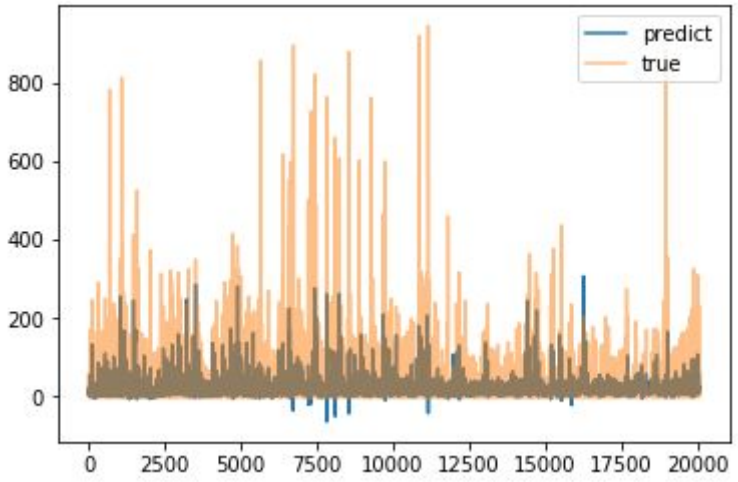
RISULTATO : loss: 0.0032; acc: 0.6485; val_loss: 0.0040; val_acc: 0.5721;
RMSE= 43.606, Train RMSE: 32.178;

PLOT:

Loss



Prediction



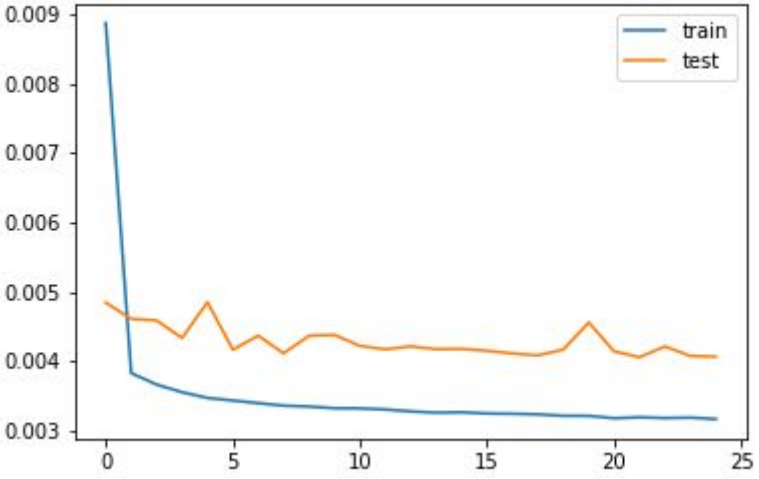
#23

DATI : neuroni= 42; epoche=25; funzione di attivazione in dense= linear;
funzione di ottimizzazione LSTM= adam; Shufflati i dati nel model.fit

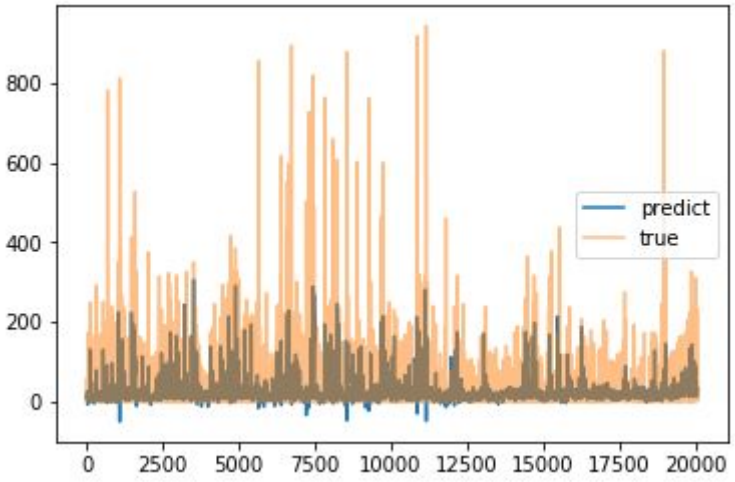
RISULTATO : loss: 0.0032; acc: 0.6485; val_loss: 0.0041; val_acc: 0.5721;
RMSE= 43.916, Train RMSE: 32.324;

PLOT:

Loss



Prediction



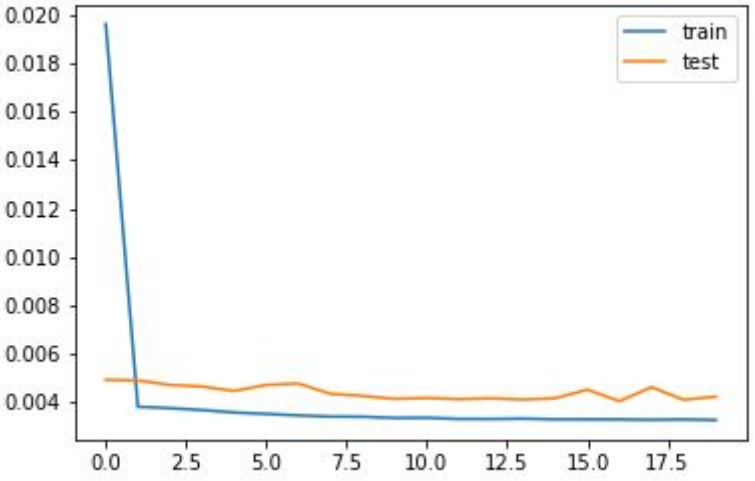
#24

DATI: neuroni LSTM: 64; epoche = 20; batch size = 128; funzione di attivazione in dense= linear; funzione di ottimizzazione LSTM= adam; Shufflati i dati nel model.fit

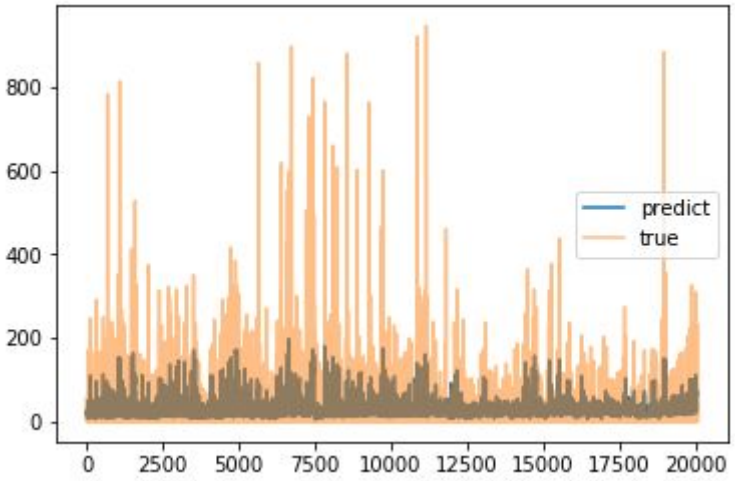
RISULTATO: loss: 0.0033 - acc: 0.6485 - val_loss: 0.0042 - val_acc: 0.5721

PLOT:

Loss



Prediction



6.2.1 Test con numero di neuroni e lookback variabili

Nei test di questa sezione rimarranno fissi il numero di epoche (15), il batch size (32) e l'architettura della rete(una cella LSTM e uno strato completamente connesso).
I valori del dataset prima di essere processati dalla rete vengono scalati tra -1 e 1.
La funzione di attivazione della cella LSTM è la tanh.

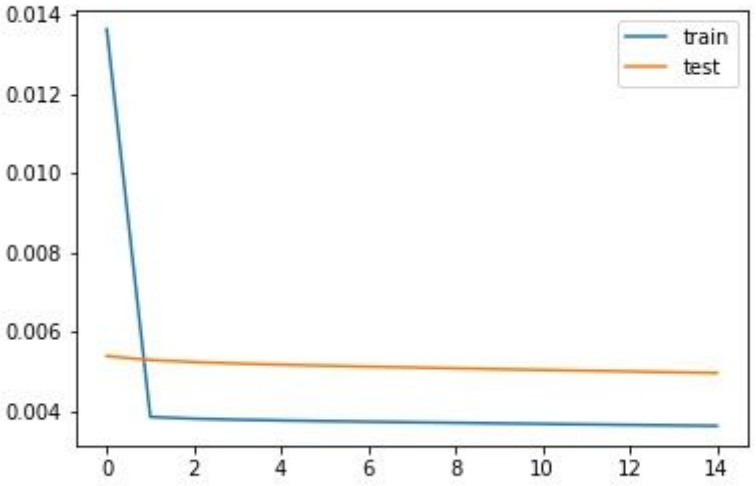
#1

DATI : neuroni LSTM = 25; lookback = 70;

RISULTATO : loss=0.0036; accuracy = 0.6485; val_loss=0.0050; val_acc=0.5731;

PLOT:

Loss



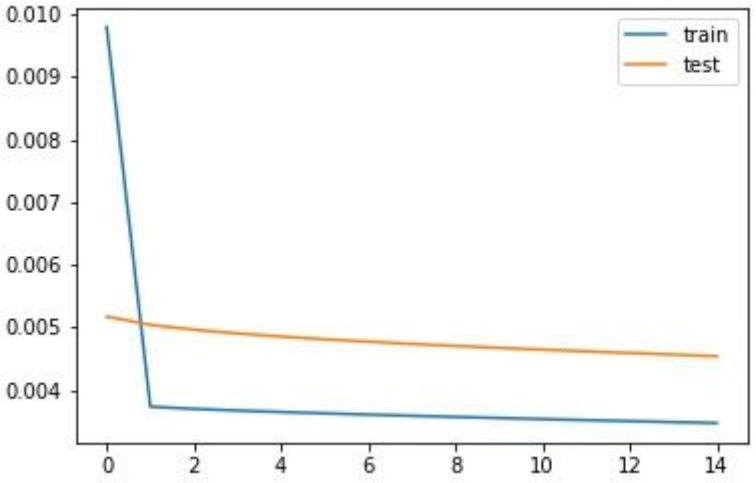
#2

DATI : neuroni LSTM = 50; lookback = 70;

RISULTATO : loss=0.0035; accuracy = 0.6491; val_loss=0.0045; val_acc=0.5713;

PLOT:

Loss



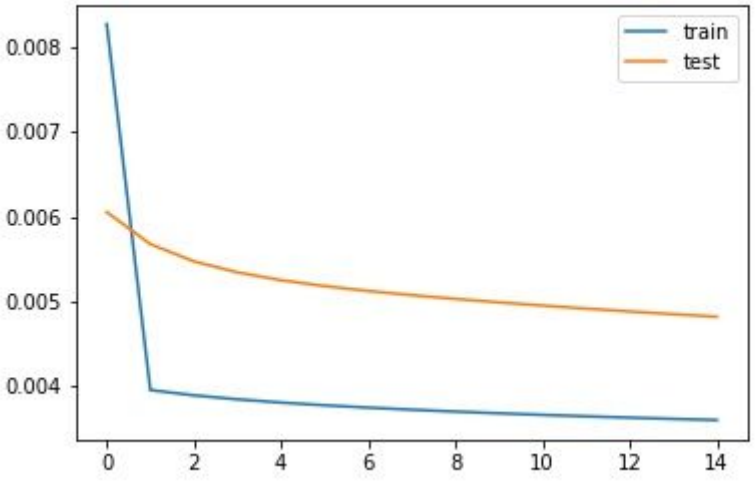
#3

DATI : neuroni LSTM = 100; lookback = 70;

RISULTATO : loss=0.0036; accuracy = 0.6496; val_loss=0.0048; val_acc=0.5696;
RMSE= 35.392

PLOT:

Loss



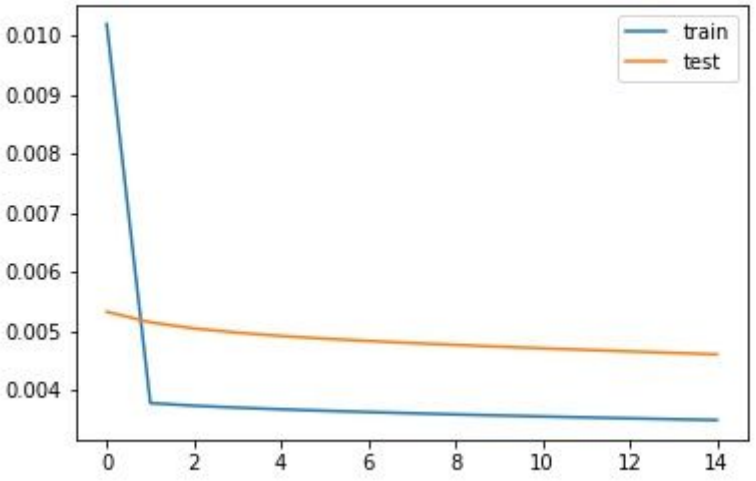
#4

DATI : neuroni LSTM = 50; lookback = 30;

RISULTATO : loss=0.0035; accuracy = 0.6490; val_loss=0.0046; val_acc=0.5708;
RMSE= 37,887

PLOT:

Loss



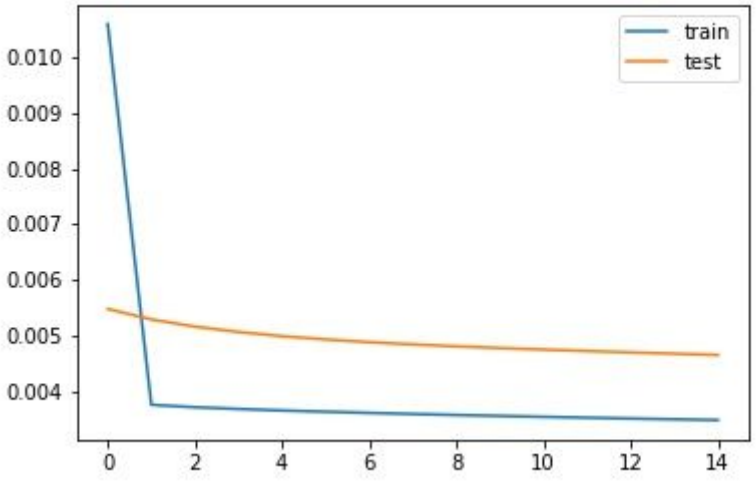
#5

DATI : neuroni LSTM = 50; lookback = 150;

RISULTATO : loss=0.0035; accuracy = 0.6486; val_loss=0.0047; val_acc=0.5697;
RMSE= 37,124

PLOT:

Loss



6.2.2 Test con dataset non shufflato

Rimangono fissi il batch size (32), il lookback (70) e l'architettura della rete(una cella LSTM e uno strato completamente connesso).
I valori del dataset prima di essere processati dalla rete vengono scalati tra -1 e 1.
La funzione di attivazione della cella LSTM è la tanh.

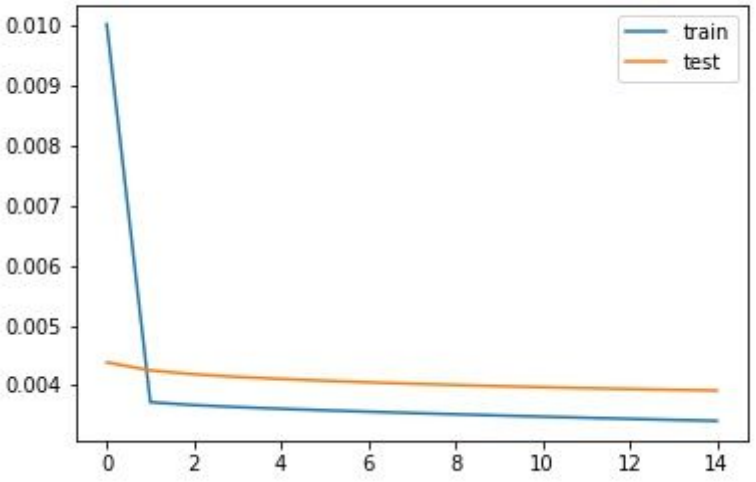
#6

DATI : neuroni LSTM = 50; epoche = 15;

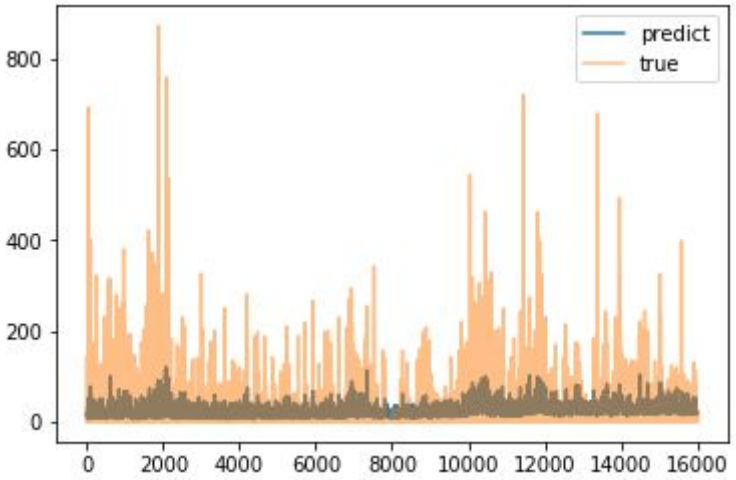
RISULTATO : loss=0.0034; accuracy = 0.6406; val_loss=0.0039; val_acc=0,6818;
RMSE= 38,543

PLOT:

Loss



Prediction



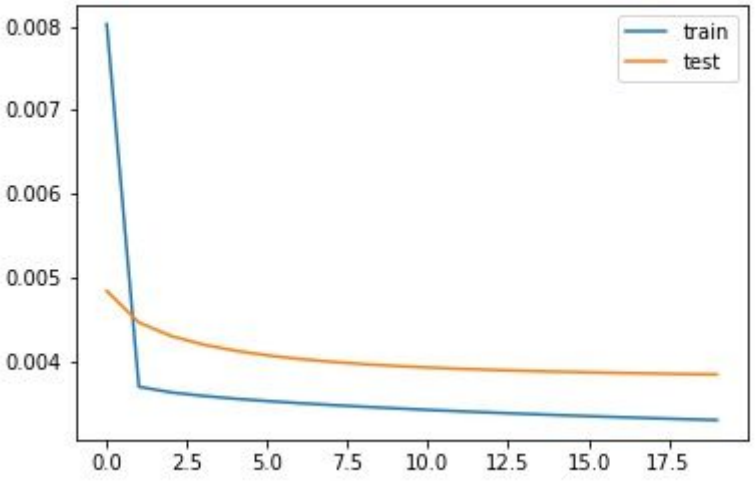
#7

DATI : neuroni LSTM = 100; epoche = 25;

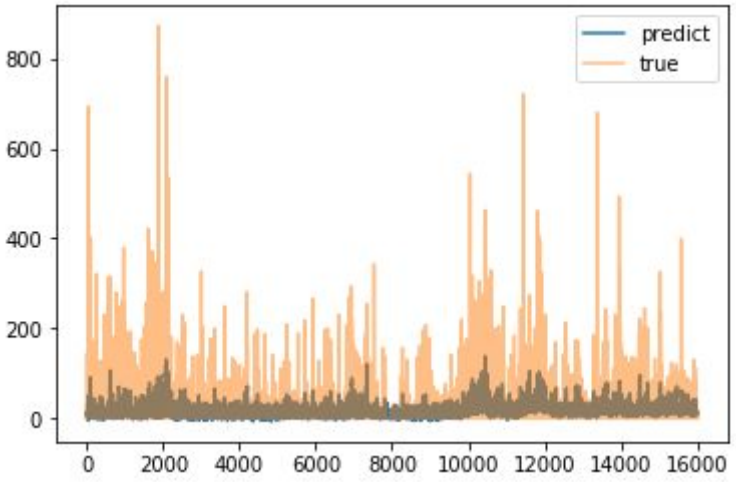
RISULTATO : loss=0.0033; accuracy = 0.6406; val_loss=0.0038; val_acc=0.6818;
RMSE= 37.546

PLOT:

Loss



Prediction



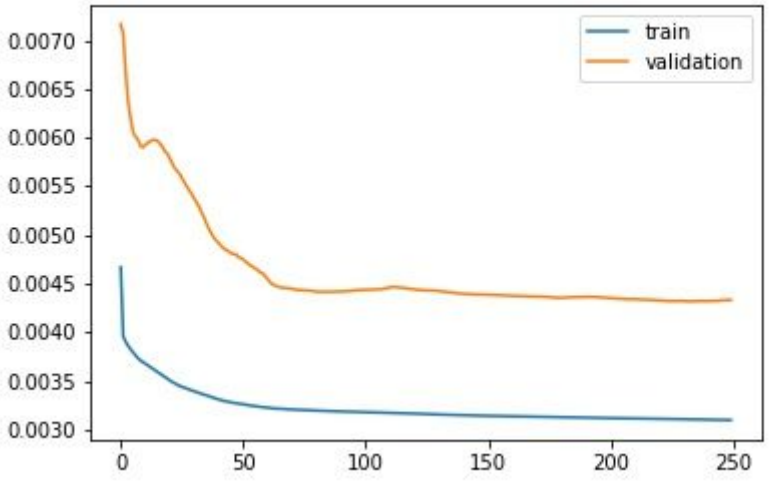
#8

DATI : neuroni LSTM = 20; epoche = 250; learning_rate = 0,1;

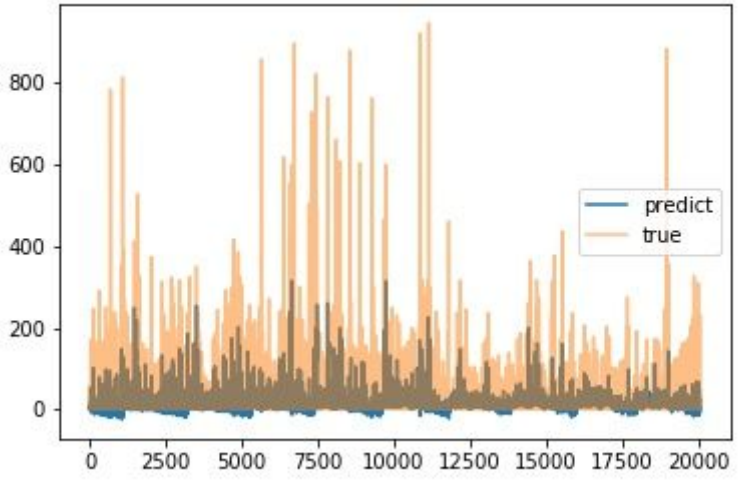
RISULTATO : loss=0.0031; accuracy = 0.6485; val_loss=0.0043; val_acc=0.5720;
RMSE= 44.037

PLOT:

Loss



Prediction



7. Bibliografia

1. Dataset :
    a. https://www.bts.gov/ ;
2. Lstm :
    a. http://adventuresinmachinelearning.com/keras-lstm-tutorial/ ;
3. Implementazione :
    a. https://www.kaggle.com/jphoon/bitcoin-time-series-prediction-with-lstm ;
    b. https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/ ;