Progetto Intelligenza Artificiale

Gianluca Parpanesi

22 febbraio 2024

Sommario

"Confronto tra ARIMA e Tensorflow LSTM su titoli azionari"

Introduzione

Il progetto ha lo scopo di mettere a confronto due tecniche di regressione per apprendere e predire un'andamento di una specifica azione (scelta dall'utente). Le tecniche utilizzate sono le seguenti:

- **ARIMA**: modello matematico-statistico basato su serie temporali di dati. La parte Autoregressiva (AR) indica che la variabile di interesse viene regredita su valori precedenti (numero di intervalli di tempo). L'integrazione (I) indica che i valori dei dati sono stati sostituiti con la differenza tra i loro valori e i valori dei precedenti (volte in cui i dati sono stati sottratti ai valori passati) . Infine la media mobile (MA) indica che l'errore di regressione è in realtà una combinazione lineare di termini di errore i cui valori si sono verificati contemporaneamente e in tempi differenti in passato. ARIMA combina le caratteristiche autoregressive con quelle delle medie mobili. Un processo autoregressivo AR(1), ad esempio, è quello in cui il valore corrente è basato sul valore immediatamente precedente, mentre un processo AR(2) è quello in cui il valore corrente è basato sui due valori precedenti. Una media mobile è un calcolo utilizzato per analizzare i punti dati creando una serie di medie di diversi sottoinsiemi dell'intero set di dati per attenuare l'influenza dei valori anomali. Come risultato di questa combinazione di tecniche, i modelli ARIMA possono tenere conto di tendenze, cicli, stagionalità e altri tipi di dati non statici quando si effettuano previsioni. [1]

- LSTM: rete neurale ricorrente (*RNN - Recurrent Neural Network*) applicabile a processi di classificazione, serie temporali e predizione. Una unità LSTM (*LSTM - Long Short Term Memory*) è composta da una singola cella con un input, un output e un *forget gate*. La cella ricorda i valori di una serie temporale arbitraria e l'input e il forget regolano il flusso di informazioni entranti nella singola cella. Il forget decide quale informazione deve essere scartata confrontando lo stato precedente con il valore attuale ed assegnandoli un valore [0, 1]. L'input decide quale parte di informazioni mantenere mentre l'output (come detto dal nome stesso), lascia passare un valore (basato sull'input e sul forget).[2]

[1] fonte: Wikipedia/ARIMA, Investopedia/ARIMA

[2] fonte: Wikipedia/LSTM

Struttura del codice

Il codice, scritto in Python, permette di selezionare un'azione (o un ETF / fondo) che si desidera analizzare. All'interno del codice è possibile regolare degli *iperpametri* in modo da regolare in base alle proprie esigenze la quantità dei dati da scaricare, regolare ARIMA e regolare Tensorflow (compresi i singoli parametri della RNN).

1 Avvio del programma

La parte iniziale ci permette di scaricare un titolo di mercato a nostra scelta mediante la libreria yfinance[3], la quale scarica i dati direttamente dalle API pubbliche di Yahoo Finance i dati del titolo da noi richiesto. Il titolo viene ricercato tramite il Ticker (abbreviazione utilizzata per identificare un titolo, *es: Apple Inc. = AAPL, Microsoft = MSFT*), scaricando i dati in un lasso di tempo identificato da un iperparametro PERIOD.

```
...
yf.Ticker(input("Quale titolo vuoi scaricare?\n"))
...
history = data.history(PERIOD)
```

2 Struttura del codice

Una volta scaricati i dati del titolo richiesto verranno passati a due metodi, uno che avvierà l'analisi tramite ARIMA mentre l'altro tramite Tensorflow (LSTM). Per entrambi è possibile impostare un iperparametro SIZE il quale identifica come verranno suddivisi i dati. Precisamente SIZE è un valore in percentuale che identifica la porzione di dati riservata alla fase di *training* del modello (ARIMA/Tensorflow LSTM), il restante verrà utilizzato come porzione di *test*.

2.1 ARIMA

Il file arima.py dispone di tre iperparametri per il settaggio delle tre componenti ARIMA ovvero, come spiegato in precedenza, AR - Autoregressive, I - Integrated, MA - Moving Average, strettamente maggiori di 0. Il modello è costruito tramite la libreria stastmodel, importando il modello ARIMA[5].

[3] pypi.org/project/yfinance/

[5] statsmodels.tsa.arima.model.ARIMA

```
def ARIMA_Predictions(data: pd.DataFrame):

    values = data.values
    size = int(len(values)*SIZE)
    train = values[0:size]
    test = values[size:len(values)]
    history = [x for x in train]
    predictions = list()

    for x in range(len(test)):

        model_fit = create_model(history)
        output = model_fit.forecast()
        pred = output[0]
        predictions.append(pred)
        obs = test[x]
        history.append(obs)


    sqm = sqrt(mean_squared_error(test, predictions
        ))

    return predictions, sqm
```

2.2 Tensorflow LSTM

La rete neurale LSTM viene costruita tramite la libreria Tensorflow[6] basata su Keras, ovvero una libreria open source di apprendimento automatico. La rete è multilivello, strutturata nel seguente modo:

- LSTM
- DROPOUT
- LSTM
- DROPOUT
- LSTM

[6] www.tensorflow.org

- DROPOUT
- DENSE

La rete è stata sviluppata multilivello per rendere più solido l'addestramento, soprattutto con pochi dati, a discapito di una computazione maggiore. Anche il file tens.py dispone di alcuni iperparametri per regolare tutti i settaggi della rete. Gli iperparametri utilizzati per la rete neurale sono i seguenti:

- EPOCHS: il numero di passi di addestramento effettuati dalla rete.
- PREDICTION DAY: indica la granularità sulla quale la rete effettua l'addestramento, un valore più alto comporta una precisione minore.
- UNITS: indica la dimensione interna di una cella LSTM.
- DROPOUT: valore compreso tra 0 e 1 che permette di decidere la percentuale di nodi interni scartati in una rete neurale in fase di addestramento.
- DENSE: indica la dimensionalità dell'output.

I dati vengono normalizzati e suddivisi in dati di train e di test basandosi sulla proporzione scelta dall'utente, dettata dall'iperparametro SIZE. Una volta suddivisi verranno create varie coppie (input, output) in modo da apprendere il modello e testarlo successivamente con i dati di test. La dimensione del blocco di dati assegnato all'input è determinato, come detto in precedenza, da PREDICITON DAY.

```
for a in range(PREDICTION_DAY, len(train)):

    #es: da 0 - 30, 30, coppia (x, y) dato x ->
        risultato y
    #es: da 1 - 31, 31

    train_x.append(train[a-PREDICTION_DAY:a, 0])
    train_y.append(train[a, 0])
```

Successivamente viene creato il modello utilizzato nella creazione della rete neurale multilivello. Il modello è di tipo Sequential(), ovvero un modello adatto per una pila di strati piana dove ogni strato ha esattamente un tensore di ingresso ed uno di uscita. Creato quindi il modello base, andremo ad aggiungere i vari strati della rete nell'ordine specificato in precedenza:

```
model = Sequential()
model.add(LSTM(units = UNITS, return_sequences =
    True, input_shape=(train_x.shape[1], 1)))
model.add(Dropout(DROPOUT))
model.add(LSTM(units = UNITS, return_sequences=True
    ))
model.add(Dropout(DROPOUT))
model.add(LSTM(units = UNITS))
model.add(Dropout(DROPOUT))
model.add(Dense(units = DENSE))
model.summary()
model.compile(optimizer = "adam", loss="
    mean_squared_error")
```

Come funzione di loss utilizziamo la stessa utilizzata per il modello ARIMA. Successivamente passeremo i dati suddivisi in precedenza, avendo quindi la rete pronta ad essere addestrata e valutata.

3 Inizio analisi

Confronteremo ora l'analisi effettuata da ARIMA e Tensorflow LSTM sull'azione Microsoft in un periodo pari a 5 anni (PERIOD = "5Y"). Una volta scaricati i dati verrà avviata automaticamente l'analisi prima di ARIMA e successivamente di Tensorflow. Ai fini di un confronto più accurato proveremo vari settaggi di vari iperparametri per entrambi. Partiremo con dei valori di default per poi incrementarli ad ogni confronto effettuato. Entrambi utilizzeranno un rapporto di size uguale a 0.70.

3.1 Primo confronto

3.1.1 Analisi ARIMA

I valori utilizzati di default sono i segueti:

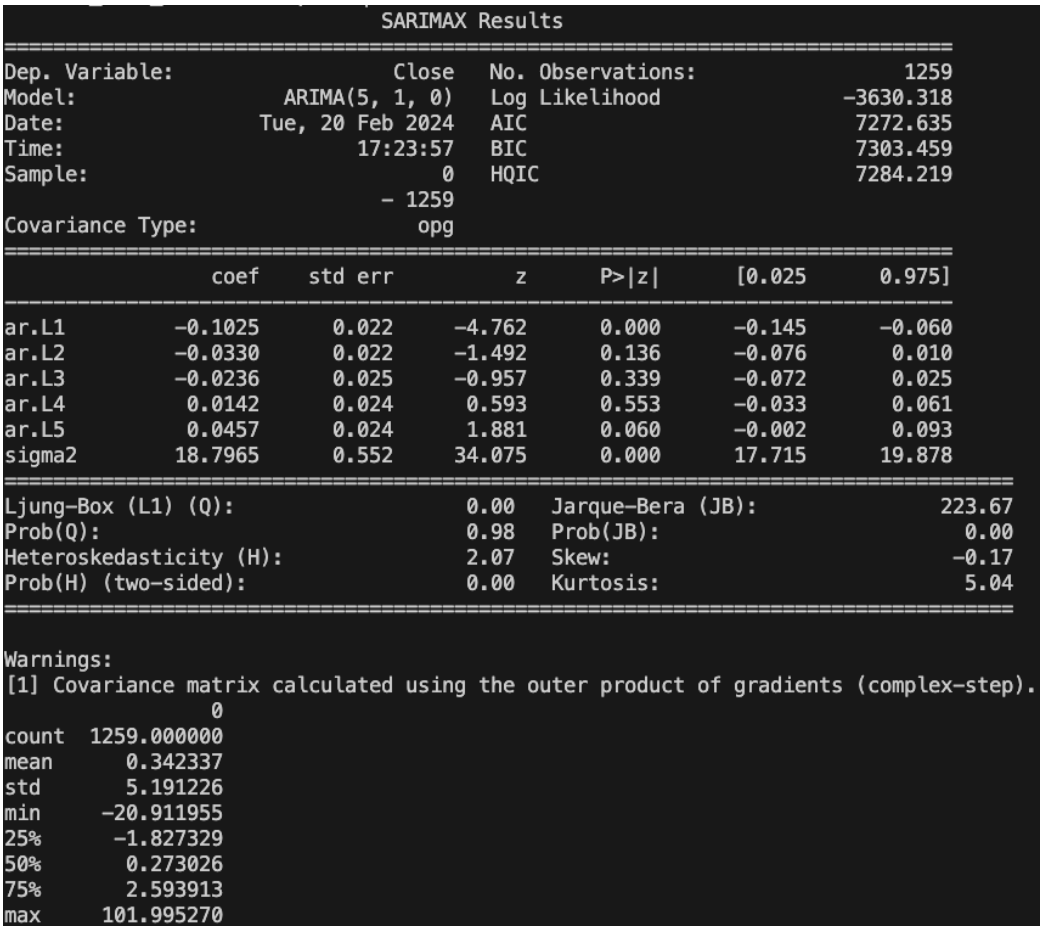
- Autoregressive: 5
- Integrated: 1
- Moving Average: 0

3.1.2 Analisi Tensorflow LSTM

I valori utilizzati di default sono i seguenti:

- EPOCHS: 5
- PREDICTION DAY: 30
- UNITS: 64
- DROPOUT: 0.5
- DENSE: 1 (rimarrà tale per tutti gli altri addestramenti)

3.1.3 Confronto



SARIMAX Results

Dep. Variable:	Close	No. Observations:	1259
Model:	ARIMA(5, 1, 0)	Log Likelihood	-3630.318
Date:	Tue, 20 Feb 2024	AIC	7272.635
Time:	17:23:57	BIC	7303.459
Sample:	0	HQIC	7284.219
	- 1259		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1025	0.022	-4.762	0.000	-0.145	-0.060
ar.L2	-0.0330	0.022	-1.492	0.136	-0.076	0.010
ar.L3	-0.0236	0.025	-0.957	0.339	-0.072	0.025
ar.L4	0.0142	0.024	0.593	0.553	-0.033	0.061
ar.L5	0.0457	0.024	1.881	0.060	-0.002	0.093
sigma2	18.7965	0.552	34.075	0.000	17.715	19.878

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	223.67
Prob(Q):	0.98	Prob(JB):	0.00
Heteroskedasticity (H):	2.07	Skew:	-0.17
Prob(H) (two-sided):	0.00	Kurtosis:	5.04

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

	0
count	1259.000000
mean	0.342337
std	5.191226
min	-20.911955
25%	-1.827329
50%	0.273026
75%	2.593913
max	101.995270

Figura 1: Risultati analisi preliminare della libreria ARIMA

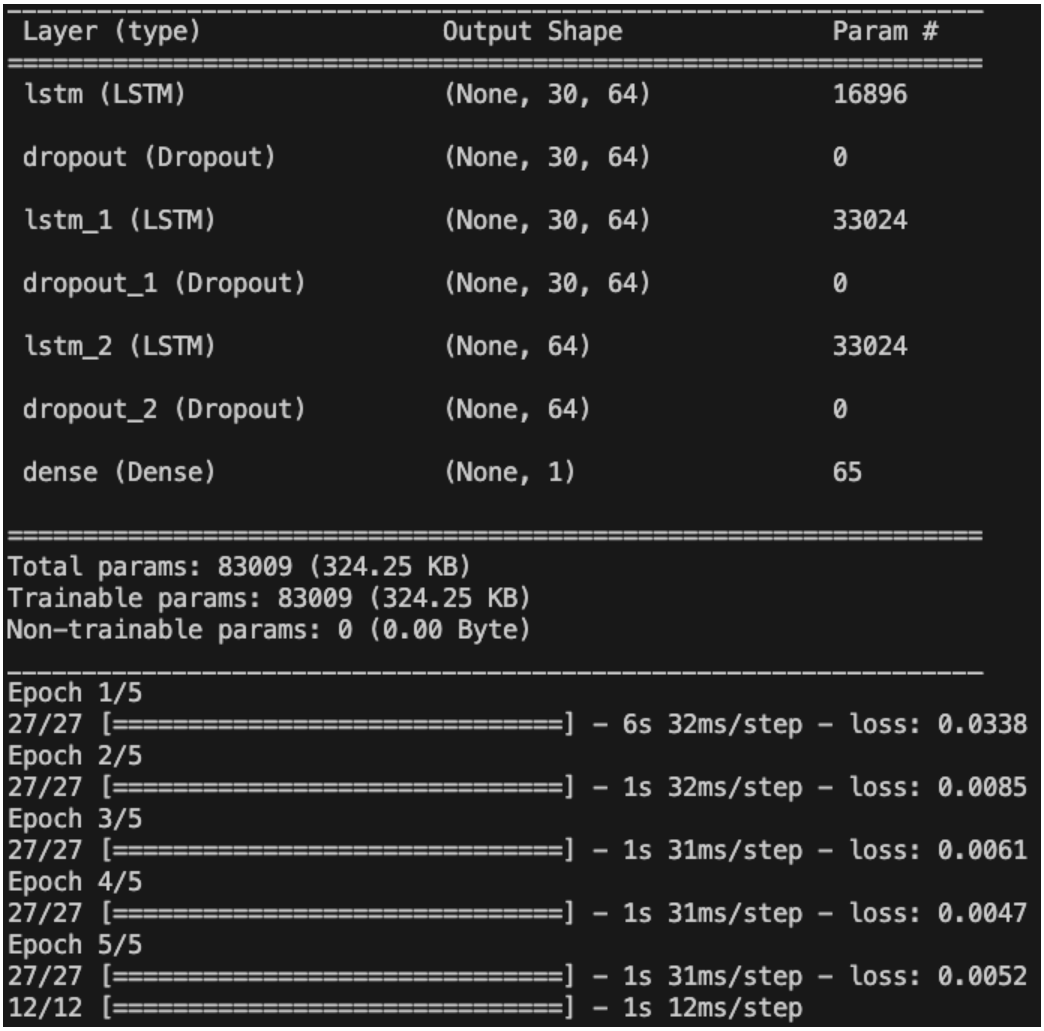


Figura 2: Struttura Rete Neurale e passi di apprendimento

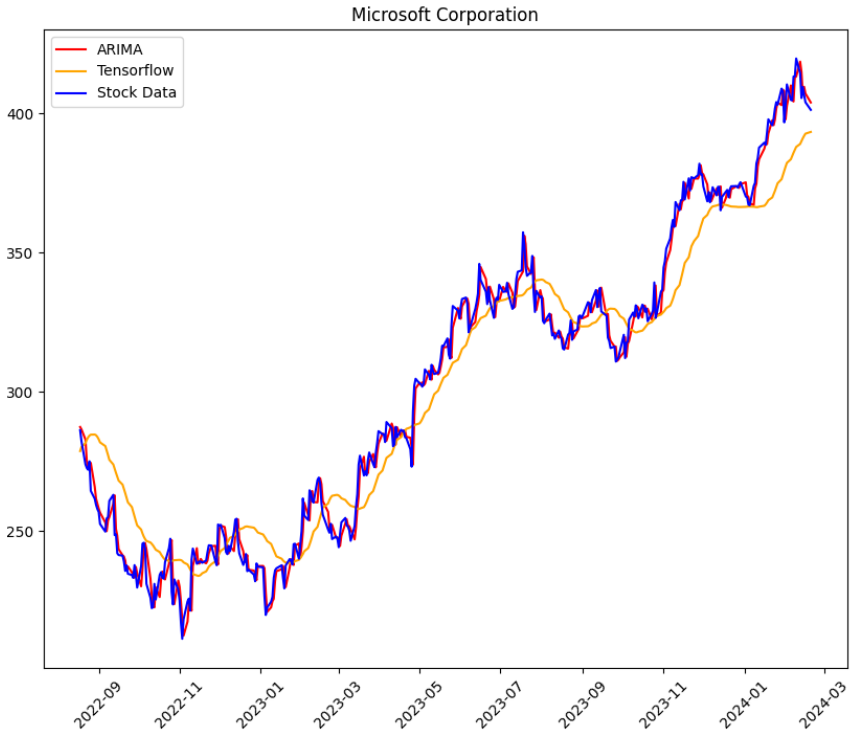


Figura 3: Confronto tra apprendimento ARIMA e LSTM

I risultati ottenuti, confrontati tramite scarto quadratico medio sono i seguenti:

```
Scarto quadratico medio

ARIMA:                  5.000594417640255
TENSORFLOW:             14.269615876183487
```

Figura 4: Confronti tra gli scarti quadratici medi

3.2 Secondo confronto

3.2.1 Analisi ARIMA

I valori utilizzati di default sono i segueti:

- Autoregressive: 10
- Integrated: 2
- Moving Average: 0

3.2.2 Analisi Tensorflow LSTM

I valori utilizzati di default sono i seguenti:

- EPOCHS: 10
- PREDICTION DAY: 15
- UNITS: 128
- DROPOUT: 0.3

3.2.3 Confronto

SARIMAX Results

Dep. Variable:	Close	No. Observations:	1259
Model:	ARIMA(10, 2, 0)	Log Likelihood	-3668.421
Date:	Tue, 20 Feb 2024	AIC	7358.842
Time:	17:29:55	BIC	7415.343
Sample:	0	HQIC	7380.076
	- 1259		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9978	0.023	-44.318	0.000	-1.042	-0.954
ar.L2	-0.9389	0.034	-27.862	0.000	-1.005	-0.873
ar.L3	-0.8463	0.040	-21.279	0.000	-0.924	-0.768
ar.L4	-0.7299	0.042	-17.550	0.000	-0.811	-0.648
ar.L5	-0.5880	0.044	-13.355	0.000	-0.674	-0.502
ar.L6	-0.5529	0.044	-12.620	0.000	-0.639	-0.467
ar.L7	-0.4122	0.044	-9.463	0.000	-0.498	-0.327
ar.L8	-0.3646	0.041	-8.817	0.000	-0.446	-0.284
ar.L9	-0.1452	0.033	-4.351	0.000	-0.211	-0.080
ar.L10	-0.0546	0.024	-2.308	0.021	-0.101	-0.008
sigma2	20.0326	0.590	33.936	0.000	18.876	21.190

Ljung-Box (L1) (Q):	0.03	Jarque-Bera (JB):	202.22
Prob(Q):	0.86	Prob(JB):	0.00
Heteroskedasticity (H):	2.18	Skew:	-0.04
Prob(H) (two-sided):	0.00	Kurtosis:	4.96

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

	0
count	1259.000000
mean	0.028782
std	5.493923
min	-48.846270
25%	-2.308543
50%	-0.053139
75%	2.354398
max	101.995255

Figura 5: Risultati analisi preliminare della libreria ARIMA

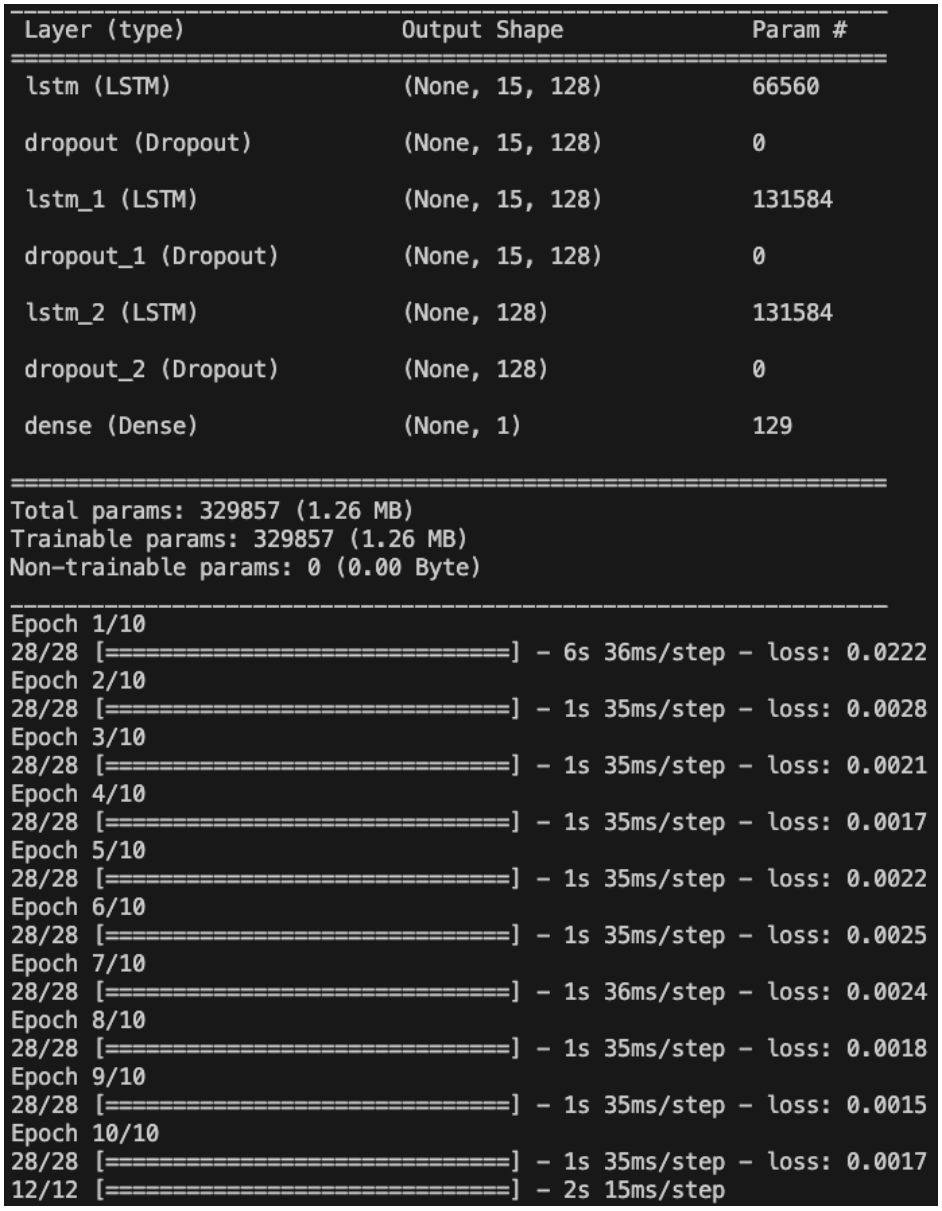


Figura 6: Struttura Rete Neurale e passi di apprendimento

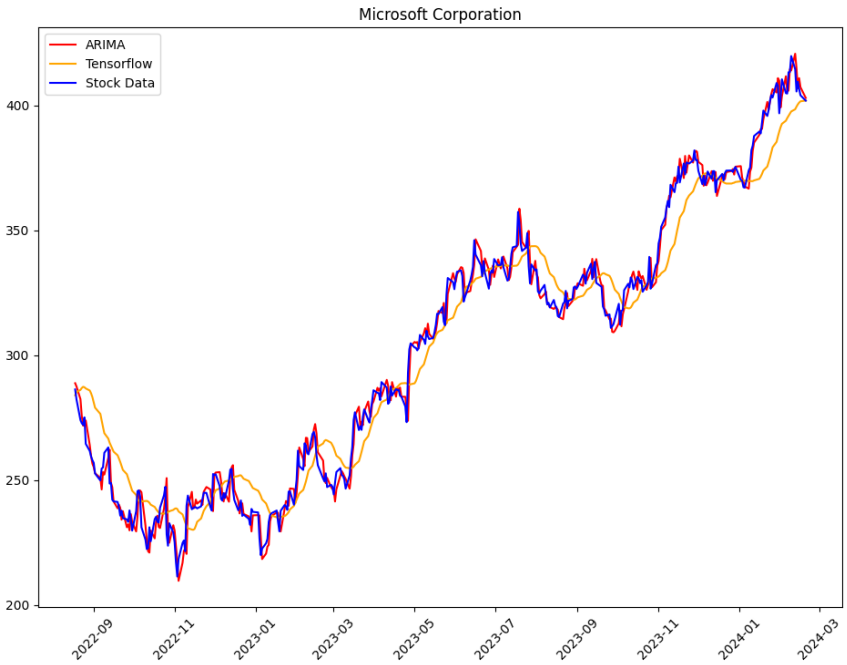


Figura 7: Confronto tra apprendimento ARIMA e LSTM

I risultati ottenuti, confrontati tramite scarto quadratico medio sono i seguenti:

```
Scarto quadratico medio

ARIMA:                  5.2100090030738855
TENSORFLOW:             11.65341443062873
```

Figura 8: Confronti tra gli scarti quadratici medi

3.3 Terzo confronto

3.3.1 Analisi ARIMA

I valori utilizzati di default sono i segueti:

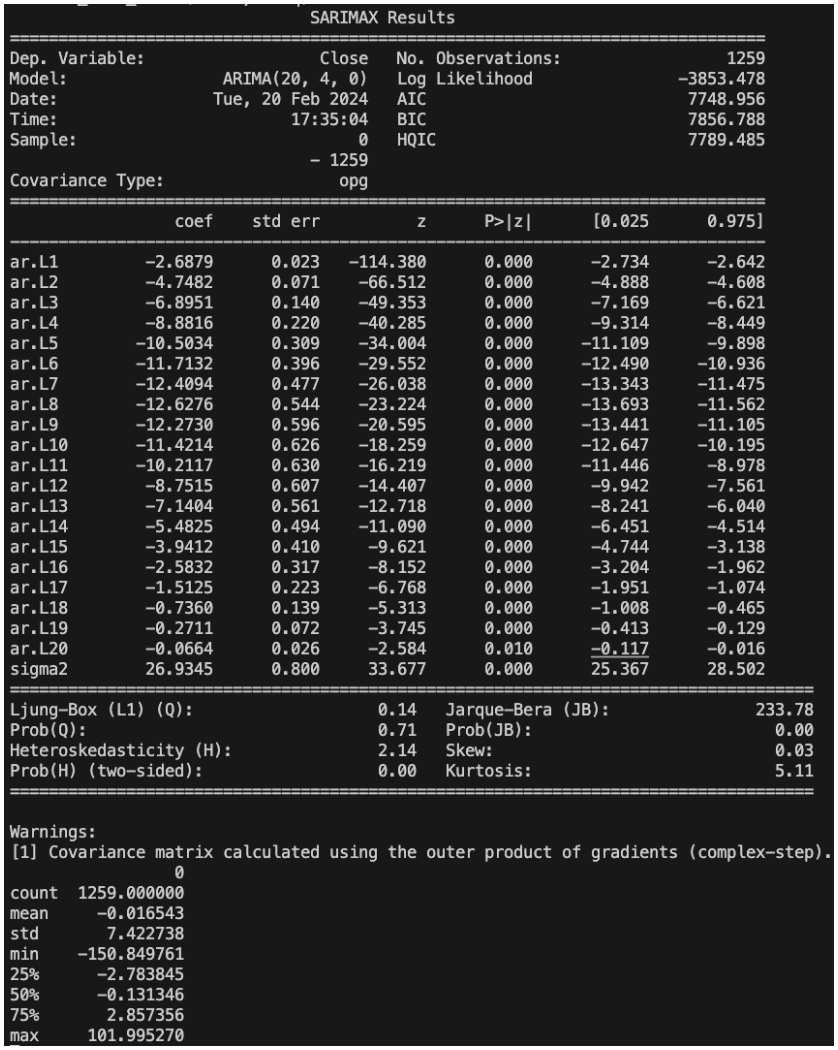
- Autoregressive: 20
- Integrated: 4
- Moving Average: 0

3.3.2 Analisi Tensorflow LSTM

I valori utilizzati di default sono i seguenti:

- EPOCHS: 20
- PREDICTION DAY: 7
- UNITS: 256
- DROPOUT: 0.2

3.3.3 Confronto



SARIMAX Results

Dep. Variable:	Close	No. Observations:	1259
Model:	ARIMA(20, 4, 0)	Log Likelihood	-3853.478
Date:	Tue, 20 Feb 2024	AIC	7748.956
Time:	17:35:04	BIC	7856.788
Sample:	0	HQIC	7789.485
	- 1259		
Covariance Type:	opg		

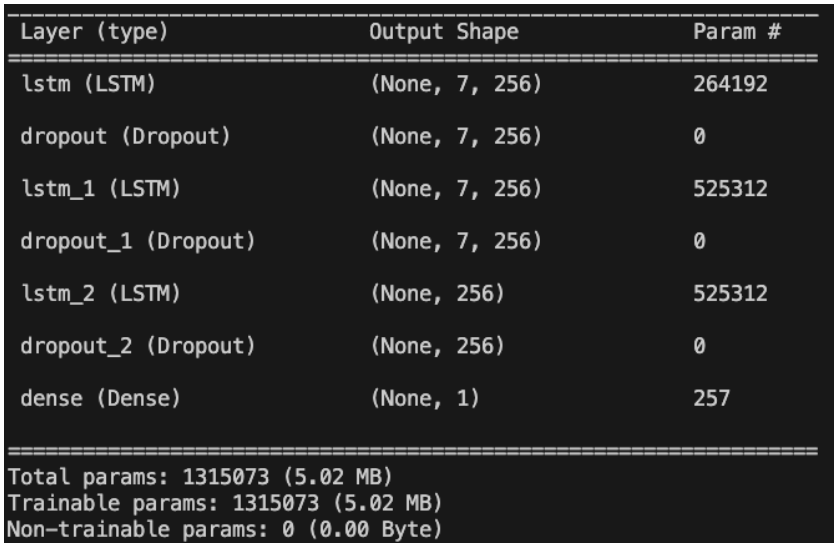
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-2.6879	0.023	-114.380	0.000	-2.734	-2.642
ar.L2	-4.7482	0.071	-66.512	0.000	-4.888	-4.608
ar.L3	-6.8951	0.140	-49.353	0.000	-7.169	-6.621
ar.L4	-8.8816	0.220	-40.285	0.000	-9.314	-8.449
ar.L5	-10.5034	0.309	-34.004	0.000	-11.109	-9.898
ar.L6	-11.7132	0.396	-29.552	0.000	-12.490	-10.936
ar.L7	-12.4094	0.477	-26.038	0.000	-13.343	-11.475
ar.L8	-12.6276	0.544	-23.224	0.000	-13.693	-11.562
ar.L9	-12.2730	0.596	-20.595	0.000	-13.441	-11.105
ar.L10	-11.4214	0.626	-18.259	0.000	-12.647	-10.195
ar.L11	-10.2117	0.630	-16.219	0.000	-11.446	-8.978
ar.L12	-8.7515	0.607	-14.407	0.000	-9.942	-7.561
ar.L13	-7.1404	0.561	-12.718	0.000	-8.241	-6.040
ar.L14	-5.4825	0.494	-11.090	0.000	-6.451	-4.514
ar.L15	-3.9412	0.410	-9.621	0.000	-4.744	-3.138
ar.L16	-2.5832	0.317	-8.152	0.000	-3.204	-1.962
ar.L17	-1.5125	0.223	-6.768	0.000	-1.951	-1.074
ar.L18	-0.7360	0.139	-5.313	0.000	-1.008	-0.465
ar.L19	-0.2711	0.072	-3.745	0.000	-0.413	-0.129
ar.L20	-0.0664	0.026	-2.584	0.010	-0.117	-0.016
sigma2	26.9345	0.800	33.677	0.000	25.367	28.502

Ljung-Box (L1) (Q):	0.14	Jarque-Bera (JB):	233.78
Prob(Q):	0.71	Prob(JB):	0.00
Heteroskedasticity (H):	2.14	Skew:	0.03
Prob(H) (two-sided):	0.00	Kurtosis:	5.11

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

	0
count	1259.000000
mean	-0.016543
std	7.422738
min	-150.849761
25%	-2.783845
50%	-0.131346
75%	2.857356
max	101.995270

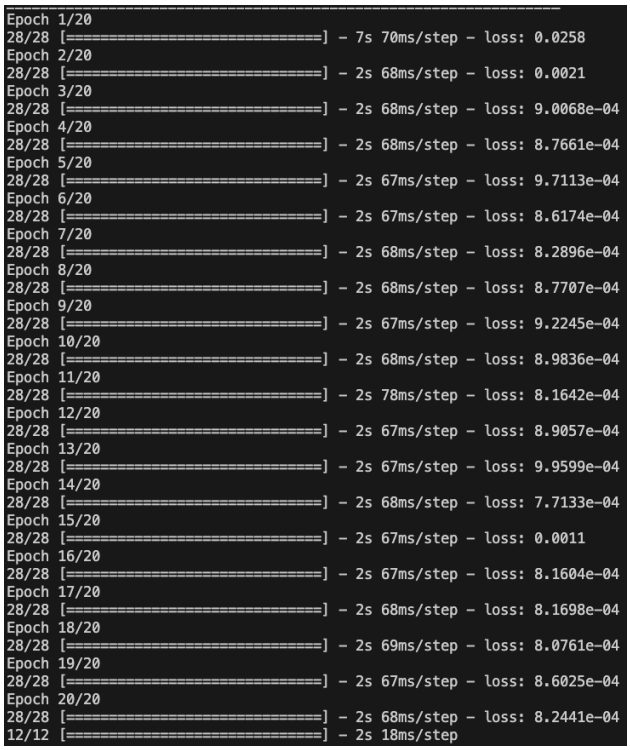
Figura 9: Risultati analisi preliminare della libreria ARIMA



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 7, 256)	264192
dropout (Dropout)	(None, 7, 256)	0
lstm_1 (LSTM)	(None, 7, 256)	525312
dropout_1 (Dropout)	(None, 7, 256)	0
lstm_2 (LSTM)	(None, 256)	525312
dropout_2 (Dropout)	(None, 256)	0
dense (Dense)	(None, 1)	257

Total params: 1315073 (5.02 MB)
Trainable params: 1315073 (5.02 MB)
Non-trainable params: 0 (0.00 Byte)

Figura 10: Struttura Rete Neurale



```
Epoch 1/20
28/28 [==============================] - 7s 70ms/step - loss: 0.0258
Epoch 2/20
28/28 [==============================] - 2s 68ms/step - loss: 0.0021
Epoch 3/20
28/28 [==============================] - 2s 68ms/step - loss: 9.0068e-04
Epoch 4/20
28/28 [==============================] - 2s 68ms/step - loss: 8.7661e-04
Epoch 5/20
28/28 [==============================] - 2s 67ms/step - loss: 9.7113e-04
Epoch 6/20
28/28 [==============================] - 2s 67ms/step - loss: 8.6174e-04
Epoch 7/20
28/28 [==============================] - 2s 68ms/step - loss: 8.2896e-04
Epoch 8/20
28/28 [==============================] - 2s 68ms/step - loss: 8.7707e-04
Epoch 9/20
28/28 [==============================] - 2s 67ms/step - loss: 9.2245e-04
Epoch 10/20
28/28 [==============================] - 2s 68ms/step - loss: 8.9836e-04
Epoch 11/20
28/28 [==============================] - 2s 78ms/step - loss: 8.1642e-04
Epoch 12/20
28/28 [==============================] - 2s 67ms/step - loss: 8.9057e-04
Epoch 13/20
28/28 [==============================] - 2s 67ms/step - loss: 9.9599e-04
Epoch 14/20
28/28 [==============================] - 2s 68ms/step - loss: 7.7133e-04
Epoch 15/20
28/28 [==============================] - 2s 67ms/step - loss: 0.0011
Epoch 16/20
28/28 [==============================] - 2s 67ms/step - loss: 8.1604e-04
Epoch 17/20
28/28 [==============================] - 2s 68ms/step - loss: 8.1698e-04
Epoch 18/20
28/28 [==============================] - 2s 69ms/step - loss: 8.0761e-04
Epoch 19/20
28/28 [==============================] - 2s 67ms/step - loss: 8.6025e-04
Epoch 20/20
28/28 [==============================] - 2s 68ms/step - loss: 8.2441e-04
12/12 [==============================] - 2s 18ms/step
```

Figura 11: Passi di addestramento

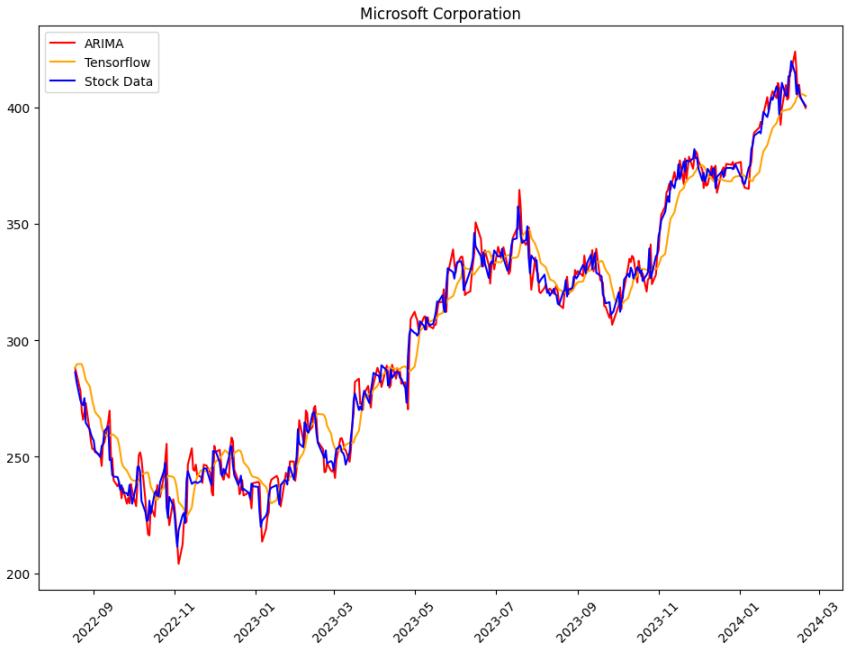


Figura 12: Confronto tra apprendimento ARIMA e LSTM

I risultati ottenuti, confrontati tramite scarto quadratico medio sono i seguenti:

```
Scarto quadratico medio

ARIMA:                  6.051355881692028
TENSORFLOW:             9.137515005237795
```

Figura 13: Confronti tra gli scarti quadratici medi

4 Analisi risultati

Come possiamo notare, fin dai primi valori ottenuti tramite ARIMA, lo scarto quadratico medio è davvero basso, ottenendo una curva di predizione quasi identica alla curva dei dati originale. Incrementando i valori, per semplicità raddoppiati ad ogni addestramento, il valore dello scarto quadratico medio aumenta anche se di poco.
L'addestramento mediante Rete Neurale (Tensorflow LSTM), inizialemente ottiene prestazioni nettamente inferiori rispetto ad ARIMA. Questo possiamo

tranquillamente notarlo con il divario che abbiamo tra i due scarti quadratici medi. Tuttavia notiamo come all'aumentare della complessità nella rete neurale, lo scarto quadratico medio diminuisce notevolemente.

5 Conclusioni

ARIMA applica una regressione specifica per le serie temporali, di conseguenza possiamo notare come fin dai primi risultati otteniamo previsioni ottime con scarti quadratici medi davvero bassi.
LSTM invece utilizza comunque regressione ma per raggiungere la stessa precisione (se non migliore di ARIMA), necessita di una complessità della rete maggiore e una notevole quantità di potenza di calcolo (oltre che di memoria) essendo una rete neurale applicabile sia a casi di regressione su serie temporali, ma anche in riconoscimento della scrittura, delle parole e molti altri ambiti di ricerca.
Nel caso di ARIMA, aumentando principalmente il valore di Autoregressione (AR), otteniamo una complessità del polinomio maggiore aumentando di conseguenza lo scarto dai dati originali. Nella Rete Neurale abbiamo quindi un comportamento opposto dove, aumentando la complessità e la potenza di calcolo richiesta, aumentiamo la precisione della nostra previsione facendo diminiure lo scarto.
Di fatti la Rete Neurale pur ricevendo un gran contributo da iperparametri come EPOCHS e SIZE, gran parte della precisione è affidata all'iperparametro UNITS (che regola la dimensione di una singola cella LSTM): notiamo difatti come aumentando la UNITS, a discapito della stessa SIZE per ogni confronto, la dimensione degli input aumenta (da KB a MB) dato che, aumentando la complessità della cella, aumenta la complessità generale di come gli input vengono combinati ottenendo cammini computazionali più complessi e aumentando la precisione finale. L'iperparametro DROPOUT permette di scartare una serie di dati espressa in percentuale [0, 1] alleggerendo l'addestramento: esso è comunque strettamente legato a UNITS difatti riducendo il suo valore verranno scartati meno elementi. Ricollegandosi quindi a quanto detto precedentemente, scartando meno dati, manteniamo cammini computazionali più complessi aumentando la precisione dell'addestramento (e di conseguenza il risultato finale).