Applicazione del modello ARIMA in ambiente Python

L'acronimo ARIMA sta per Auto-Regressive Integrated Moving Average. È uno degli strumenti più comuni per la previsione di una serie temporale nell'ambiente dei data scientist. ARIMA è utilizzato ai fini previsionali nei più svariati ambiti. In questo documento viene trattata la codifica in linguaggio Python del modello, lasciando libero il lettore di approfondire in maniera autonoma la già vasta documentazione teorica disponibile.

Per l'esecuzione ho utilizzato un notebook jupiter (file .ipynb) ma può essere utilizzato qualsiasi ambiente di sviluppo Python (file .py).

Descrizione e codifica del modello

Per prima cosa carichiamo le librerie utili allo sviluppo del nostro algoritmo.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from statsmodels.tools.eval_measures import rmse
import seaborn as sns
import statsmodels.api as sm
import itertools
from statsmodels.tsa.arima_model import ARIMA, ARMA
import warnings
warnings.filterwarnings("ignore")
```

Con le seguenti istruzioni importiamo il set di dati e lo visualizziamo. Per questa operazione utilizziamo la funzione di pandas: "read_csv". All'interno dei due apici va inserito il percorso che consente di accedere al file. Il file deve essere di tipo CSV (Comma-separated values), un file di testo con un formato specifico che consente di salvare i dati in maniera strutturata in tabella. I dati all'interno della tabella sono separati da una virgola.

```
data = pd.read_csv('mio path/mio file.csv')
data.head()
```

Per acquisire i dati in formato CSV è possibile utilizzare il servizio gratuito di Yahoo Finance (https://it.finance.yahoo.com/), che consente di scaricare le serie storiche dei sottostanti desiderati. Yahoo Finance permette di impostare il periodo storico interessato, e la frequenza dei dati. In alcuni casi i dati devono essere "puliti" eliminando eventuali campi vuoti (null), per farlo basta semplicemente aprire il file con un editor di testo e eliminare la riga. Certamente eliminare il record non è la strada migliore, l'informazione mancante andrebbe trattata in modo diverso ma non è lo scopo di questo lavoro.

Il file contiene nella prima colonna il campo "date" nel

Maurizio Michele Zuzzaro

Maurizio Michele Zuzzaro è il fondatore del sito finanziario www.performancetrading.it. Ha oltre 18 anni di esperienza come analista finanziario nel mercato delle valute e delle materie prime energetiche. È membro della SIAT (Società Italiana per Analista Tecnica) e dell'Associazione dei mercati finanziari italiani (Assiom Forex).

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-01-02	2419.0	2424.0	2367.0	2394.0	2394.0	22875
1	2019-01-03	2396.0	2423.0	2379.0	2400.0	2400.0	21118
2	2019-01-04	2399.0	2402.0	2314.0	2361.0	2361.0	22432
3	2019-01-07	2362.0	2413.0	2356.0	2410.0	2410.0	16914
4	2019-01-08	2389.0	2399.0	2355.0	2382.0	2382.0	19167

formato ISO 8601 cioè yyyy-mm-dd, i dati OHLC (open, High, Low, Close), il prezzo di chiusura rettificato Adj Close (Adjusted Closing Price) e infine il volume. I dati riportati nella tabella riguardano il future della commodity Cocoa.

Con le seguenti istruzioni impostiamo il frame di dati sulla "data" e sul "prezzo di chiusura", ovviamente questi valori possono essere sostituiti da altri che si desiderano analizzare.

```
df = data[['Date','Close']]
df.Date = pd.to_datetime(df.Date)
df = df.set_index("Date")
```

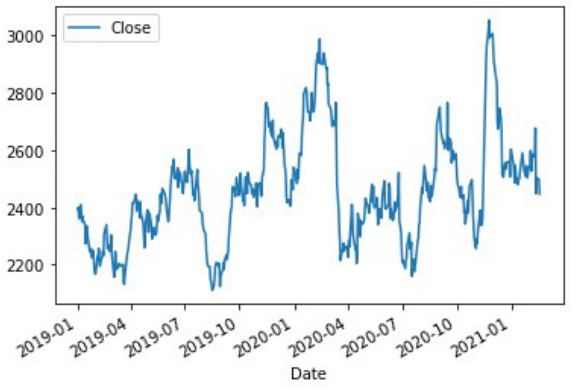
Per tracciare il grafico a linea utilizzando la funzione plot().

```
df.plot(style="-")
```

Otteniamo la rappresentazione grezza dei valori.

Quando si adatta un modello ARIMA, bisogna impostare i valori dei parametri P, D e Q che ottimizzano o minimizzano una certa metrica di interesse. Esistono molti metodi per trovare i giusti valori per la corretta parametrizzazione dei modelli ARIMA. Questo processo può essere noioso e

```
<matplotlib.axes._subplots.AxesSubplot at 0x26309d84760>
```



richiede tempo e competenza statistica.

In questo documento, cerchiamo di superare questo problema scrivendo un algoritmo di ricerca a griglia in Python, che permette di selezionare i valori dei parametri ottimali per il nostro modello in modo automatico.

L'uso di una "ricerca a griglia" consiste nell'esplorare in modo iterativo diverse combinazioni di parametri. Per ogni combinazione, adattiamo un modello ARIMA con la funzione SARIMAX (), e ne valutiamo le prestazioni complessive. Dopo aver esplorato l'intero dominio dei parametri, il nostro insieme ottimale sarà quello che fornisce le migliori prestazioni per i nostri criteri di interesse. In questo scenario, il nostro criterio di interesse è l'informazione Akaike (AIC). L'AIC misura quanto un modello si adatta ai dati tenendo conto della complessità generale del modello. Noi ovviamente siamo interessati a trovare un modello che restituisca il valore AIC più basso.

Nel codice seguente definiamo i parametri e generiamo tutte le possibili combinazioni.

Applichiamo la funzione SARIMAX () a tutte le combinazioni di parametri, infine viene stampato il modello con l'AIC più basso.

```
p = d = q = range(0, 3)
pdq = list(itertools.product(p, d, q))
```

```
warnings.filterwarnings("ignore")
aic= []
parameters = []
for param in pdq:
  #for param in pdq:
      try:
          mod = sm.tsa.statespace.SARIMAX(df, order=param,enforce_stationarity=True, enforce_invertibility=True)
          results = mod.fit()
          # save results in lists
          aic.append(results.aic)
          parameters.append(param)
          #seasonal_param.append(param_seasonal)
          print('ARIMA{} - AIC:{}'.format(param, results.aic))
      except:
          continue
# find lowest aic
index_min = min(range(len(aic)), key=aic.__getitem__)

print('The optimal model is: ARIMA{} -AIC{}'.format(parameters[index_min], aic[index_min]))
```

```
ARIMA(0, 0, 0) - AIC:9801.440891494258
ARIMA(0, 0, 1) - AIC:9087.042555305728
ARIMA(0, 0, 2) - AIC:8835.681278435539
ARIMA(0, 1, 0) - AIC:5570.21172723429
ARIMA(0, 1, 1) - AIC:5572.020510603941
ARIMA(0, 1, 2) - AIC:5572.65768414109
ARIMA(0, 2, 0) - AIC:5916.034399350021
ARIMA(0, 2, 1) - AIC:5569.019975702116
ARIMA(0, 2, 2) - AIC:5570.782407087731
ARIMA(1, 0, 0) - AIC:5590.626843012112
ARIMA(1, 0, 1) - AIC:5592.485639483846
ARIMA(1, 0, 2) - AIC:5593.103590329714
ARIMA(1, 1, 0) - AIC:5571.998753725232
ARIMA(1, 1, 1) - AIC:5572.024885966345
ARIMA(1, 1, 2) - AIC:5572.048391870499
ARIMA(1, 2, 0) - AIC:5754.36819466128
ARIMA(1, 2, 1) - AIC:5570.759677397451
ARIMA(1, 2, 2) - AIC:5571.0508628362895
ARIMA(2, 0, 0) - AIC:5592.400417847459
ARIMA(2, 0, 1) - AIC:5592.654887359722
ARIMA(2, 0, 2) - AIC:5592.401023920929
ARIMA(2, 1, 0) - AIC:5572.572639573444
ARIMA(2, 1, 1) - AIC:5572.467734986294
ARIMA(2, 1, 2) - AIC:5572.200408069755
ARIMA(2, 2, 0) - AIC:5674.194701004144
ARIMA(2, 2, 1) - AIC:5571.256594344164
ARIMA(2, 2, 2) - AIC:5573.045945228648
The optimal model is: ARIMA(0, 2, 1) -AIC5569.019975702116
```

Il passaggio successivo consiste nell'adattare il modello ARIMA (0,2,1) alla nostra serie temporale in modo automatico.

```
model_fit = model.fit(disp=0)
```

Stampiamo il report con l'istruzione print().

```
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                D2.Close   No. Observations:                  529
Model:                 ARIMA(0, 2, 1)   Log Likelihood               -2782.482
Method:                       css-mle   S.D. of innovations             46.294
Date:                Sat, 20 Feb 2021   AIC                           5570.963
Time:                        09:25:16   BIC                           5583.776
Sample:                             2   HQIC                          5575.979
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0014      0.013     -0.109      0.913      -0.027       0.024
ma.L1.D2.Close    -1.0000      0.005   -196.897      0.000      -1.010      -0.990
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            1.0000           +0.0000j            1.0000            0.0000
-----------------------------------------------------------------------------
```

Infine, settiamo i parametri per creare un grafico con la previsione del prossimo periodo, impostando la dimensione grafica.

```
plt.rcParams['figure.figsize'] = [20, 10]
```

Tracciamo e salviamo il grafico in formato png.

```
model_fit.plot_predict(start=2, end=len(df)+12)
plt.savefig('miofile.png', dpi=300)
plt.show()
```

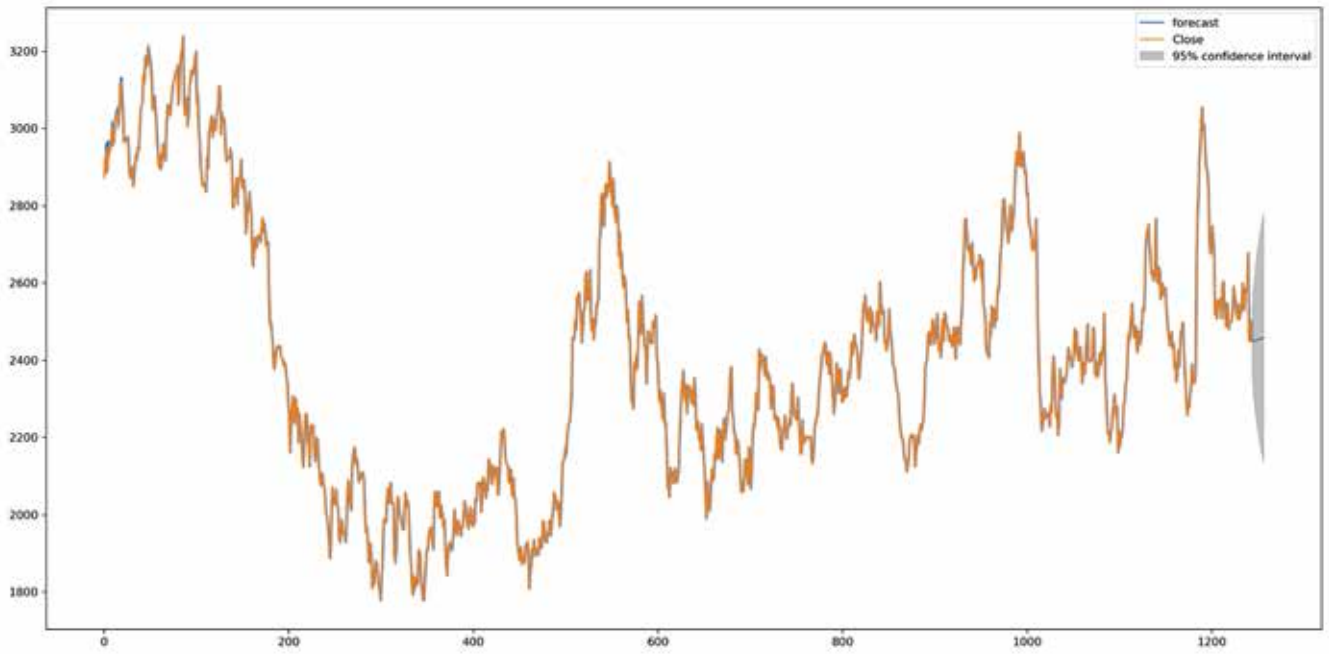
Nelle immagini vediamo alcuni plottaggi ottenuti (vedi figure da 1 a 6).

L'obbiettivo di questo articolo è di fornire spunti pratici nella applicazione di metodi statistici in Python. Per maggiori informazioni sul linguaggio rimando alla ricca comunità italiana per lo sviluppo:

https://www.python.it/

Per acquisire i dati in formato CSV è possibile utilizzare il servizio gratuito di Yahoo Finance!

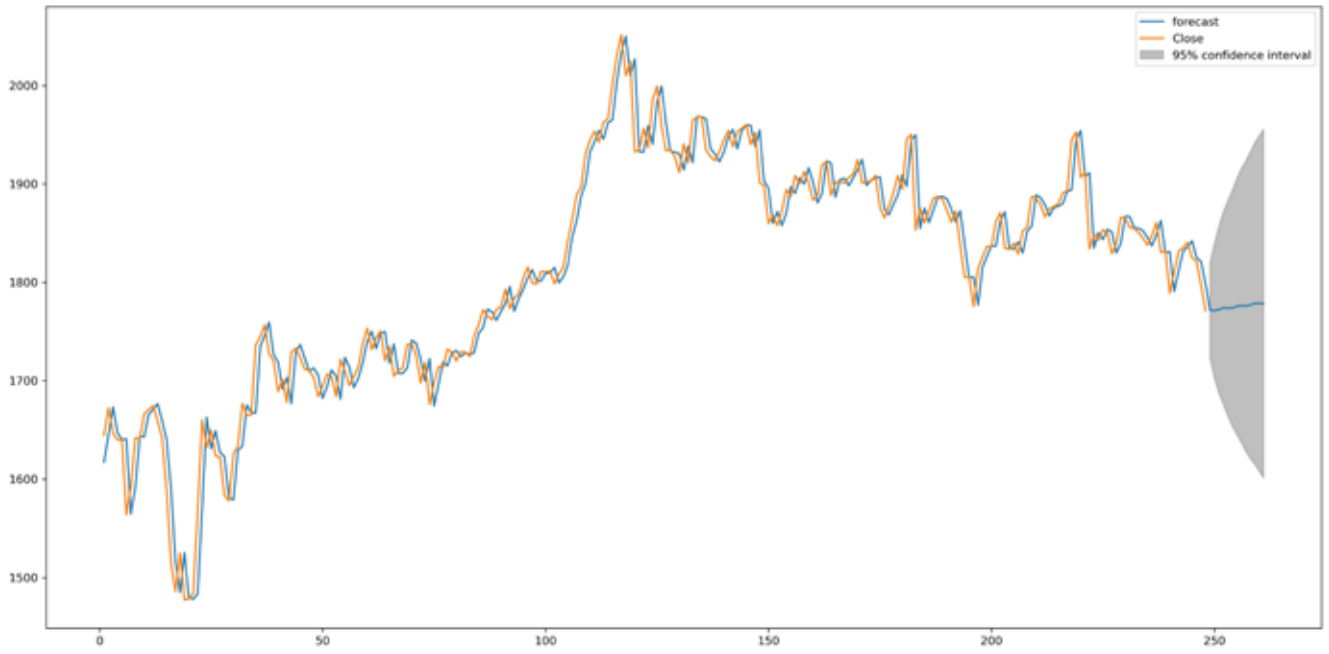
F1) Cocoa_5Y



Cocoa serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro

F2) Copper_5Y

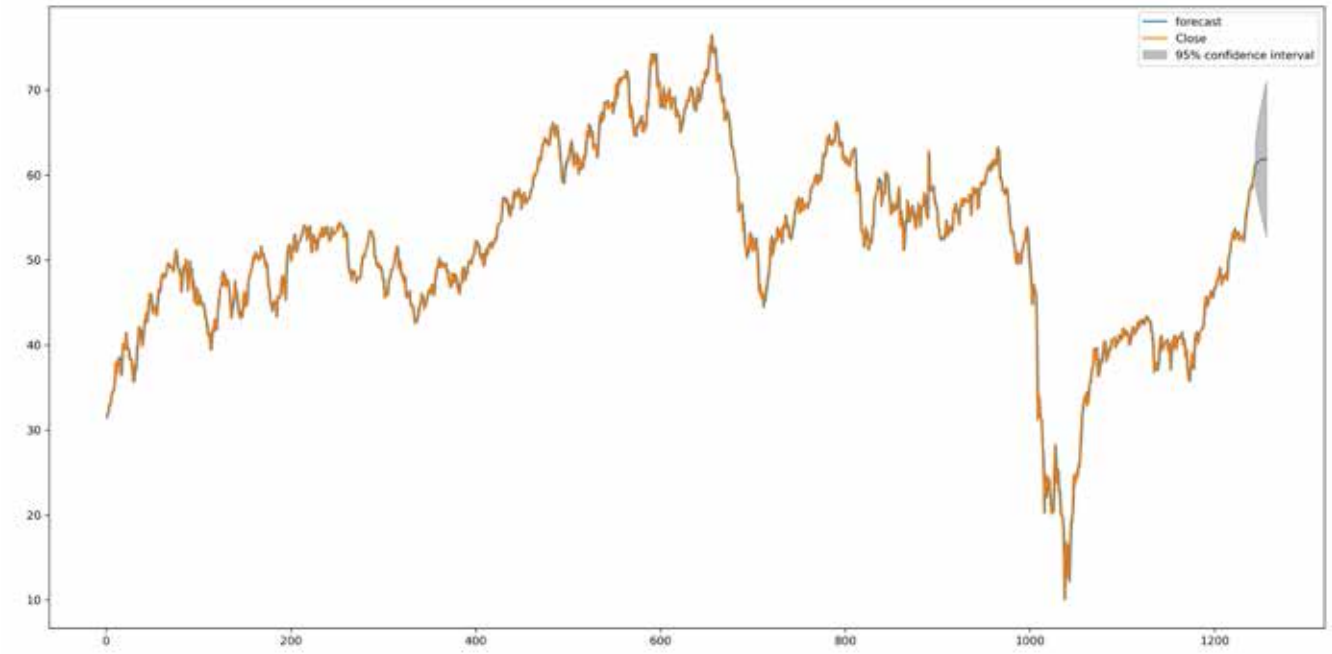


Copper serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro

» L'uso di una "ricerca a griglia" consiste nell'esplorare in modo iterativo diverse combinazioni di parametri! «

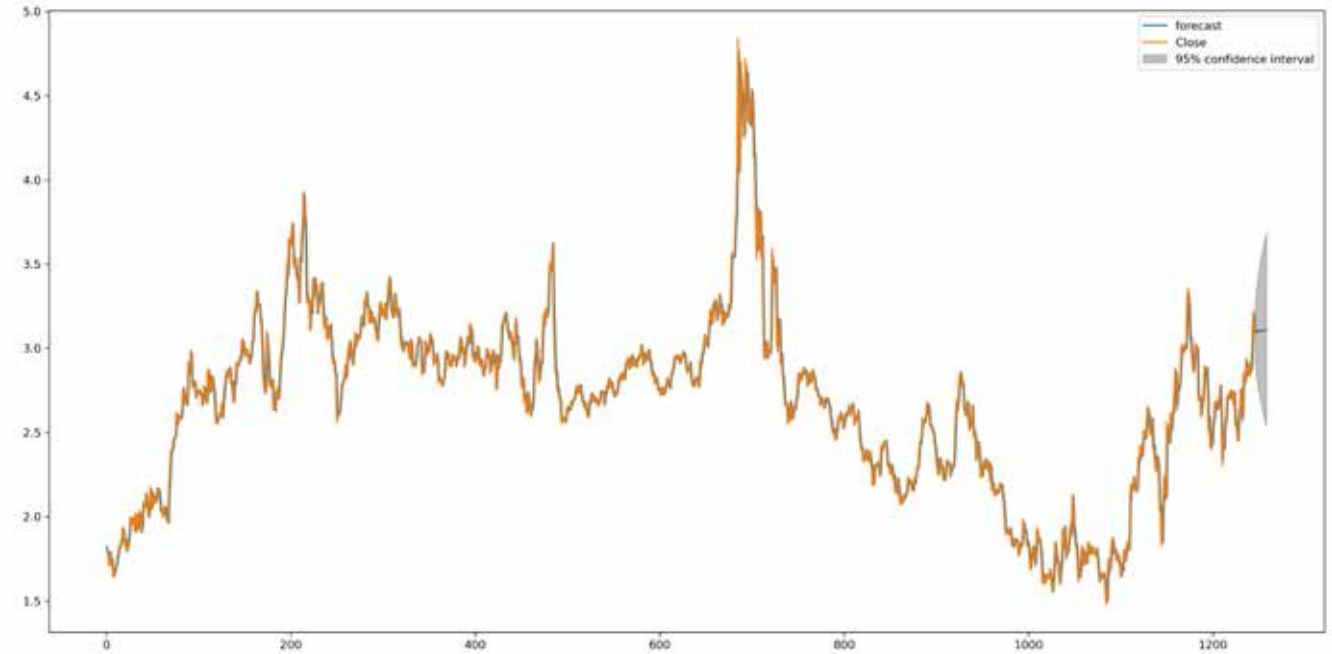
F3) Cotton_5Y



Cotton serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro

F4) CrudeOil_5Y

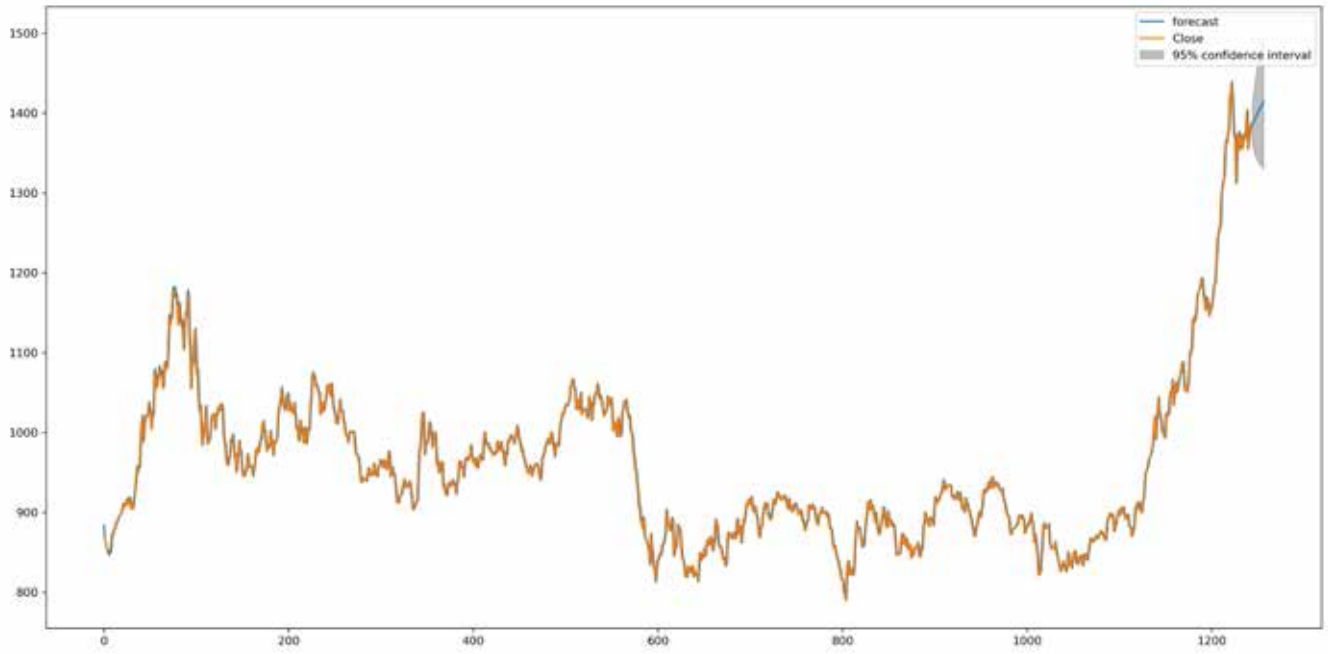


Crude Oil serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro

L'obbiettivo di questo articolo è di fornire spunti pratici nella applicazione di metodi statistici in Python!

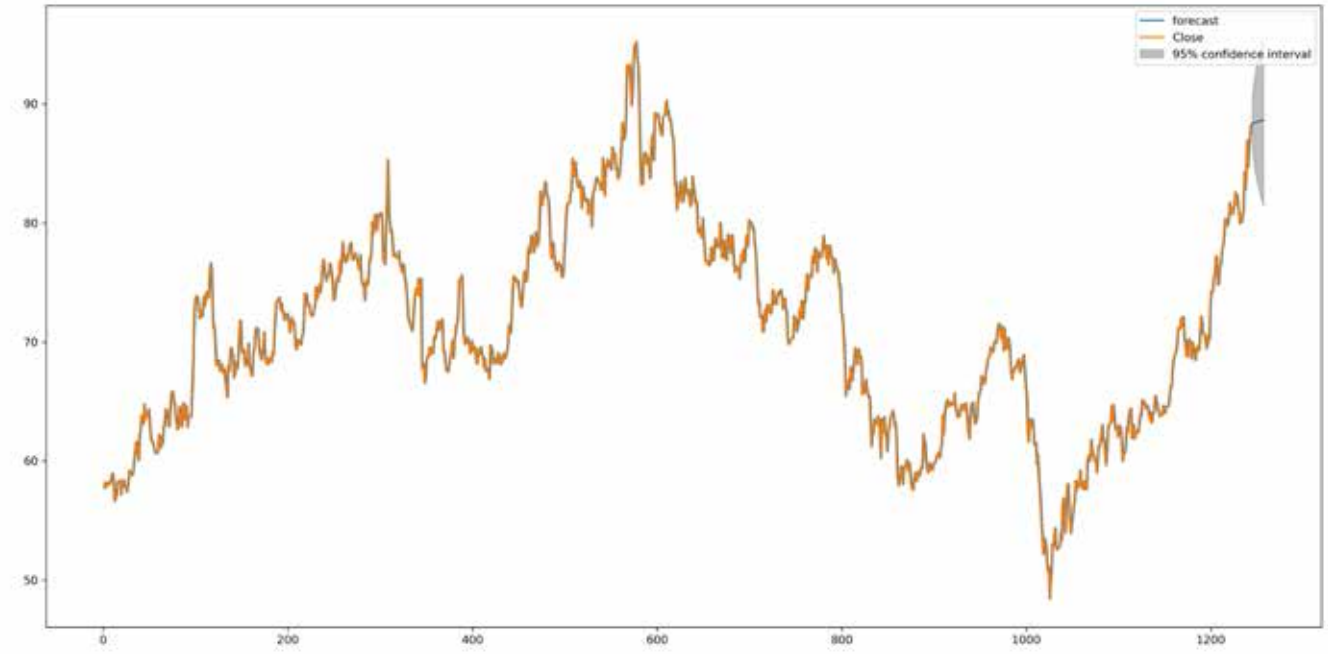
F5) NaturalGas_5Y



Natural Gas serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro

F6) Soybean_5Y



Soybean serie storica 5 anni.

Fonte: dati Yahoo finance elaborazione e plottaggio grafico Maurizio Michele Zuzzaro