Oil & Gold Price Forecasting using VAR and Granger Causality

By GROUP 6*
JAIHIND (19BM6JP43), DEO ASHISH SAMANTA (19BM6JP44)
SARNESH KALUNDIA (19BM6JP45), VINEET KUMAR(19BM6JP46)
SOHAM MULLICK (19BM6JP47), KOUSTAV MITRA (19BM6JP48)
RAJAT GAUTAM (19BM6JP49), KOWSHIK MOYYA (19BM6JP50), AND
CHETAN MEHATRE (19BM6JP60)

Oil & Gold price forecasting has become an important problem for academics as well as other stakeholders of the market. In this project we revisit the research question of significant relationship between Indian economy and historical oil price. We aim to price forecast (in INR) and assess the interdependence & causality between IIP index, oil import & export by India, and precious metal price. We are using a vector auto-regressive model to forecast and extract relevant inferences. We are using a 12 year rich monthly data sourced from EPWRF database and evaluated our model on 15 step ahead forecast using standard metrics viz. RMSE, MAPE etc.

I. Introduction

Energy resources like oil occupy a crucial position in the modern economics as they are extensively used in the provisions of goods and services. That is why volatility in their prices has been a major concern for modern economies especially since World War II. Results from existing literature on the relationship between oil price and macro economy are mixed.

Gold is used as a risk management benchmark tool in hedging and diversifying commodity portfolios. Investors in advanced and emerging markets often switch between oil and gold or combine them to diversify their portfolios

Forecasting Gold and Oil has garnered major attention from academics, investors and Government agencies alike. These two products are known for their substantial influence on the global economy. However, what is relationship between oil price and macro economy in Indian context with updated data and methodology, What generally happens to Gold prices when Oil takes a plunge? In this project we use a vector auto-regressive model to do forecasting of future Gold & Oil prices.

* Submitted for Selected Aspects of Advanced Predictive Modelling Course Project 2020

II. Dataset

The dataset is downloaded from http://www.epwrfits.in, We visualize all the time series under consideration to get a preliminary understanding of their nature. From the plots we can see that most of the time series are having an increasing trend and thus are non stationary. We also see the presence of seasonality in the Mining Index. Also we observe that many of the series seem to have a positive correlation between pairs of such time series variables like Electricity and Coke Fuel Index, Oil Import and Export and Oil Price and WPI for mineral oils

The raw dataset is plotted and shown in Figure: 1

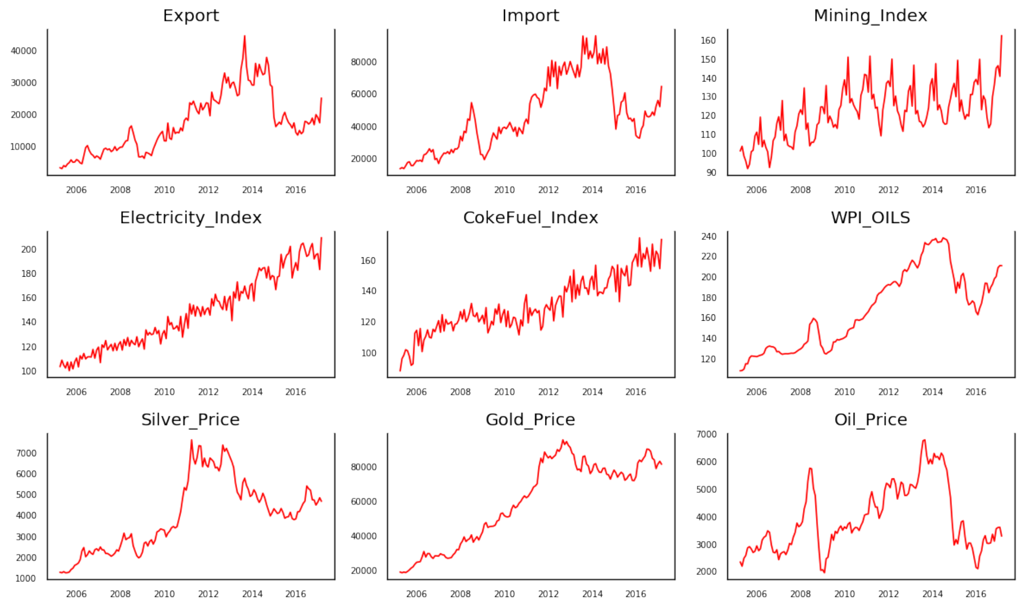


Figure 1. : Raw Time Series Data

A. Stationarity

Stationarity can be defined as a constant mean, variance or autocovariances for each given lag. There are several reasons why the concept of non-stationarity is important and why it is essential that variables that are non-stationary should be treated differently from variables that are stationary. These reasons are given as follows:

- The stationarity or otherwise of a series can strongly influence its behavior or properties. To offer one illustration, the word 'shock' usually used to denote a change or an unexpected change in a variable or perhaps simply the value of the error term during a particular time period. For a stationary

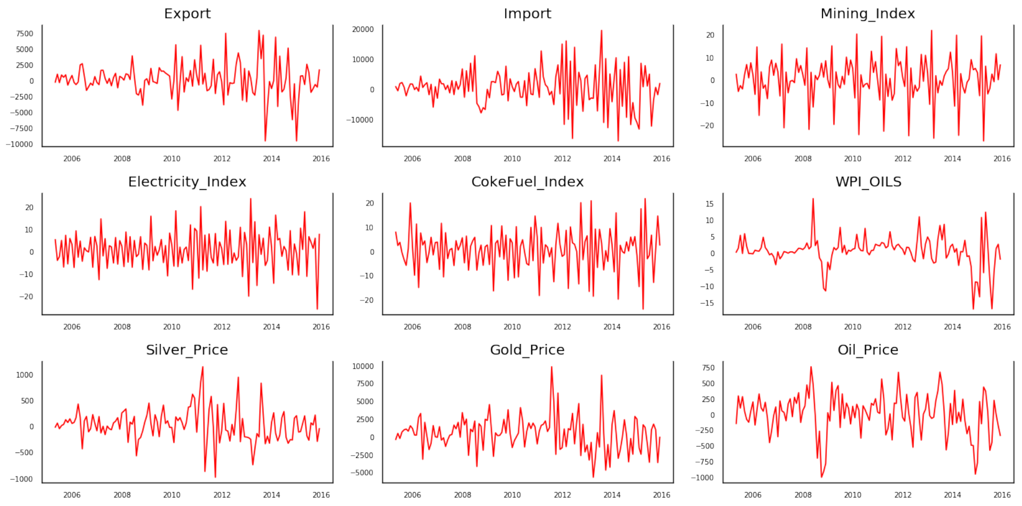


Figure 2. : Stationary Time Series after First Differencing

series, 'shocks' to the system will gradually die away. This can be contrasted with the case of non-stationary data, where the persistence of shocks will always be infinite.

- The use of non-stationarity data can lead to spurious regressions. If two stationary variables are generated as independent random series, when one of those variables is regressed on the other, the t-ratio on the slope coefficient would be expected not to be significantly different from zero, and the value of R^2 would be expected to be very low. So, if standard regression techniques are applied to non-stationary data, the end result could be a regression that 'looks' good under standard measures, but which is really valueless. Such a model would be termed a 'spurious' regression.

  First order difference makes the series stationary. The same is shown in Figure 2.

III. Causation Analysis

As we have included multiple time series we want to see how many of theses time series actually aid us in improving predictive performance of models.

For this, we use granger causality test — A time series X is said to Granger-cause Y if it can be shown, usually through a series of t-tests and F-tests on lagged values of X (and with lagged values of Y also included), that those X values provide statistically significant information about future values of Y.

Testing Granger Causality Idea. — The test is pretty simple we include another time series that we want to assess and test for corresponding coefficients of regression. If the p values is small we reject the null hypothesis that is X, do not cause other series, Y.

More formally, Let y and x be stationary time series. To test the null hypothesis that x does not Granger-cause y, one first finds the proper lagged values of y to include in a univariate autoregression of y:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_m y_{t-m} + \text{error}_t.$$

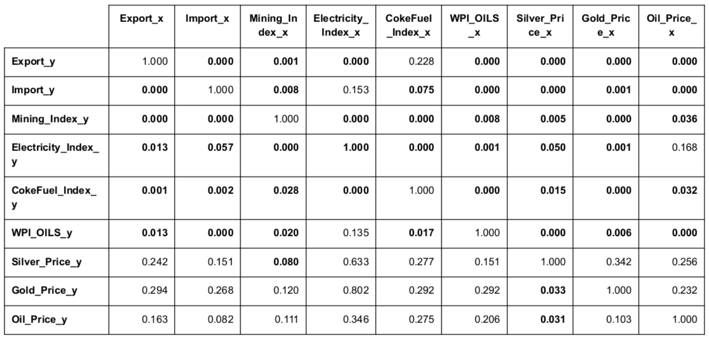
Next, the auto-regression is augmented by including lagged values of x:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_m y_{t-m} + b_p x_{t-p} + \cdots + b_q x_{t-q} + \text{error}_t.$$

One retains in this regression all lagged values of x that are individually significant according to their t-statistics, provided that collectively they add explanatory power to the regression according to an F-test (whose null hypothesis is no explanatory power jointly added by the x's). In the notation of the above augmented regression, p is the shortest, and q is the longest, lag length for which the lagged value of x is significant.

The null hypothesis that x does not Granger-cause y is accepted if and only if no lagged values of x are retained in the regression.

we must also run this model in reverse to verify that Y does not provide information about future value of X.If we find that this is the case, it is likely that there is some exogenous variable, Z, which needs to be controlled or could be a better candidate for Granger causation.



	Export_x	Import_x	Mining_Index_x	Electricity_Index_x	CokeFuel_Index_x	WPI_OILS_x	Silver_Price_x	Gold_Price_x	Oil_Price_x
Export_y	1.000	**0.000**	**0.001**	**0.000**	0.228	**0.000**	**0.000**	**0.000**	**0.000**
Import_y	**0.000**	1.000	**0.008**	0.153	**0.075**	**0.000**	**0.000**	**0.001**	**0.000**
Mining_Index_y	**0.000**	**0.000**	1.000	**0.000**	**0.000**	**0.008**	**0.005**	**0.000**	**0.036**
Electricity_Index_y	**0.013**	**0.057**	**0.000**	**1.000**	**0.000**	**0.001**	**0.050**	**0.001**	0.168
CokeFuel_Index_y	**0.001**	**0.002**	**0.028**	**0.000**	1.000	**0.000**	**0.015**	**0.000**	**0.032**
WPI_OILS_y	**0.013**	**0.000**	**0.020**	0.135	**0.017**	1.000	**0.000**	**0.006**	**0.000**
Silver_Price_y	0.242	0.151	**0.080**	0.633	0.277	0.151	1.000	0.342	0.256
Gold_Price_y	0.294	0.268	0.120	0.802	0.292	0.292	**0.033**	1.000	0.232
Oil_Price_y	0.163	0.082	0.111	0.346	0.275	0.206	**0.031**	0.103	1.000

Figure 3. : Granger Causation Matrix

In figure 4 we can see the table of p values; columns are predictors and rows are a response. Most of the variables are interchangeably causing each other. This justifies the VAR modeling approach.

IV. Co-integration and Johansen Test

Regression errors are a linear combination of two unit root series (X and Y) of the following equation:

$$\varepsilon = Y_t - \alpha - \beta X_t$$

If X_t and X_t have a unit root, this usually means ε_t also has a unit root. This is the cause of a spurious regression problem.

However, there are some cases when the unit root in Y and X cancel each other out and ε_t does not have any unit root. In this special case, the spurious regression problem vanishes. This phenomenon is referred to as cointegration, which measures the equilibrium spread between two non-stationary time series measured by constant terms α. If the error term is stationary then we can say that cointegration and an equilibrium spread exist.

Definition. The two $I(d)$ processes (Time Series) X_t and Y_t are said to have cointegration, if there exist a linear combination $aX_t + bY_t$ which is an $I(m)$ process for some $m < d$.

A. Johansen Test

Consider the joint hypothesis

$$H_0 : r = 0 \text{ vs } H_1 : 0 < r \leq g$$

$$\begin{gathered} H_0 : r = 1 \text{ vs } H_1 : 1 < r \leq g \\ H_0 : r = 2 \text{ vs } H_1 : 2 < r \leq g \\ \ldots \qquad \ldots \qquad \ldots \\ H_0 : r = g-1 \text{ vs } H_1 : r = g \end{gathered}$$

Test Statistics:

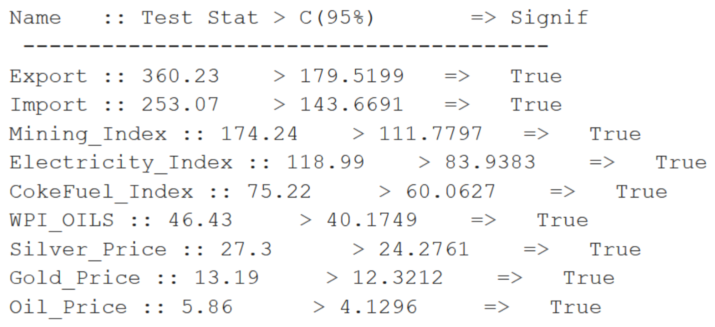
$$\lambda_{\text{trace}}(r) = -T \sum_{i=r+1}^{g} \ln\left(1 - \hat{\lambda}_i\right) \text{ and } \lambda_{\max}(r, r+1) = -T \ln\left(1 - \hat{\lambda}_{r+1}\right)$$

where, $\widehat{\lambda}_i$ is the estimated value for the ith ordered eigenvalue from the Π matrix. To use Johansen's method, we need to turn the VAR of the form

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_k y_{t-k} + u_t$$

into a VECM, which can be written as

$$\Delta y_t = \Pi y_{t-1} + \Gamma_1 \Delta y_{t-1} + \Gamma_2 \Delta y_{t-2} + \ldots + \Gamma_{k-1} \Delta y_{t-(k-1)} + u_t$$



```
Name   :: Test Stat > C(95%)       => Signif
 ----------------------------------------
Export :: 360.23    > 179.5199   =>   True
Import :: 253.07    > 143.6691   =>   True
Mining_Index :: 174.24    > 111.7797   =>   True
Electricity_Index :: 118.99    > 83.9383    =>   True
CokeFuel_Index :: 75.22      > 60.0627    =>   True
WPI_OILS :: 46.43      > 40.1749    =>   True
Silver_Price :: 27.3       > 24.2761    =>   True
Gold_Price :: 13.19      > 12.3212    =>   True
Oil_Price :: 5.86       > 4.1296     =>   True
```

Figure 4. : Johansen Test corresponding to each variable

Upon performing the test we could reject the Null Hypothesis, for $r = 0$, hence Π has rank 0. ***There are no cointegration variables.***

V. Vector Auto Regressor Model

As we can see in previous section, there is no co-integration in any of our variables. Now we can conduct Vector auto-regressor or VAR, which is essentially a multivariate forecasting algorithm used when two or more time-series influence one another.

In the VAR model, each variable is modeled as a linear combination of past values of itself and the past values of other variables in the system.

More formally, VAR models are characterized by their order, which refers to the number of earlier time periods the model will use. A lag is the value of a variable in a previous time period. So in general a pth-order VAR refers to a VAR model which includes lags for the last p time periods. A pth-order VAR is denoted "VAR(p)" and sometimes called "a VAR with p lags". A pth-order VAR model is written as

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

The variables of the form y_{t-i} indicate that variable's value i time periods earlier and are called the "ith lag" of y_t. The variable c is a $k-$vector of constants serving as the intercept of the model. A_i is a time-invariant $(k \times k)$ matrix and e_t is a $k-$vector of error terms.

The process of choosing the maximum lag p in the VAR model requires special attention because inference is dependent on correctness of the selected lag order.

As you increase the number of time series in the model, the system of equations become larger. What we will attempt to do is fit the VAR model on the training set and then use the fitted model to forecast the next 15 observations. These forecasts will be compared to the observations in the testing dataset. We have taken the maximum lag of 5 to identify the required lags in the VAR model.

VI. Forecasting

After deciding on the model we forecasted the price of Oil as well as Gold using the VAR model on the above mentioned test data. To evaluate the forecast in a more comprehensive manner, we will use mean absolute error, mean squared error, and root mean squared error. What are the difference between these three accuracy metrics:

- Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Root Mean Squared Error (RMSE) is a quadratic scoring rule that also measures the average magnitude of the error. It's the squared root of the average of squared differences between prediction and actual observation.,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  Taking the square root of the average squared errors has some interesting implications for RSME. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE sould be more useful when large errors are particularly undesirable.

- Mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

  where where A_t is the actual value and F_t is the forecast value.

Higher value of RMSE in gold is due to the higher magnitude of the original values, not because of forecast. Which is evident from the 2% MAPE value in Table 1.

We have also plotted our forecast value and overlayed it with original series in Figure 5.

	Gold Price	Oil Price
MAPE	2.067%	6.864%
MAE	1714.430	218.703
RMSE	1947.669	249.501

Table 1—: Forecast Performance

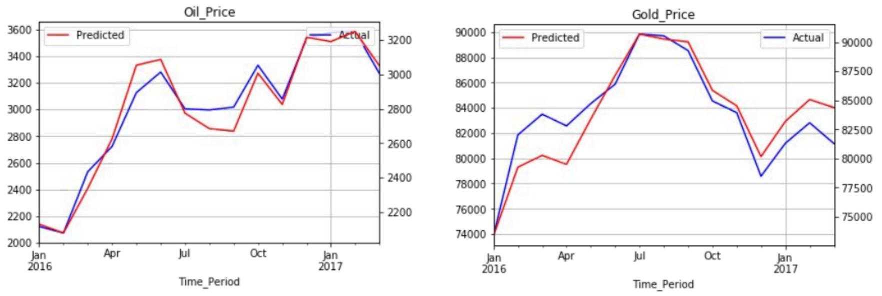


Figure 5. : Time Series Forecast

VII. Discussion and Insights

In the current study, we tried to identify and explain any causalities among gold and oil prices with other economic factors like exports and imports, WPI, electricity, and mining index values and mining index values for India, using Granger causality theory. Causal connectivity implies a connection that lies in the enhancement of forecasting between two time-series. That being said and according to our knowledge, the lack of similar research on non-market coupled countries makes the current work unique and the outcomes that arise should be taken under consideration

The model that was presented is the one that examines the interconnection between gold and oil prices and other econometric variables.

The results show that there were some causal connections among some of the electricity trading fundamentals which indicate that in order to facilitate the forecasting and minimize errors, we need to include them in the same model.

There are many different variables that can account for the changes in different financial markets. These changes may or may not always occur in financial markets. Nonetheless, this tutorial can demonstrate how we can use the relationship in certain financial markets to create models for forecasting.

Gold prices are closely related to other commodities. A hike in oil prices will have a positive impact on Gold prices and vice versa. Historically, we have seen that, when there is a hike in equity, Gold prices go down.

Varied sources of data had to be relied upon to collect the data related to Export and Import, WPI, and other Indices. All monthly data were collected from the said variables from EPWRF Data Set. These sources are vouched for their authenticity in the data world, however, vary in the presentation of the data. Some use averages of the daily data readings to provide each monthly datum, while some just provide the data observed on the first or last day of the month. The data was not adjusted for such discrepancies, therefore is likely to have some bearing on the empirical results.

The WPI, Export, and Import Values were taken as proxies for Price Levels. To what extent do these proxies stand as a true representation of respective sectors is questionable itself and hence a limitation. Wholesale Price Indices only index a few types of consumer goods prices, may not be an accurate representation of the general price levels in the economy. This warrants consideration in future researches to use data sets that remain more representative of the testable parameters.

References

1) Hiemstra, Craig, and Jonathan D. Jones. "Testing for linear and nonlinear Granger causality in the stock price- volume relation." The Journal of Finance 49.5 (1994): 1639-1664.

2) Johansen, S. (1995). Likelihood-based inference in cointegrated vector autoregressive models. Oxford University Press on Demand.

3) Lopez, J. Humberto. "The power of the ADF test." Economics Letters 57.1 (1997): 5-10.

4) Acharya, Rajesh H., and Anver C. Sadath. "Revisiting the relationship between oil price and macro economy: Evidence from India." ECONOMICS AND POLICY OF ENERGY AND THE ENVIRONMENT (2018).

Code Listing

Check jupyter notebook, dataset and other details https://github.com/vntkumar8/refactored-bassoon/.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import datetime


# In[2]:


df=pd.read_csv('Combined_data2.csv',parse_dates=['Time_Period'])


# In[3]:


df['Time_Period']=df['Time_Period'].apply(lambda x: datetime.datetime.
    strptime(x,'%b-%y'))


# In[4]:


df=df.set_index('Time_Period')


# In[5]:


df.head(5)


# In[169]:


from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# Plot
fig, axes = plt.subplots(nrows=3, ncols=3, dpi=120, figsize=(10,6))
for i, ax in enumerate(axes.flatten()):
    data = df[df.columns[i]]
    ax.plot(data, color='red', linewidth=1)
    # Decorations
    ax.set_title(df.columns[i])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines['top'].set_alpha(0)
    ax.tick_params(labelsize=6)
    plt.tight_layout();

```

```python

# In[170]:


from scipy import stats

# unpacks the test statistic and p-value from the normaltest of gold
stat, p = stats.normaltest(df.Oil_Price)

# prints both the test statistic and p-value
print('Statistics=%.3f, p=%.3f' % (stat, p))

# chooses an alpha of 0.05 and assigns to variable 'alpha'
alpha = 0.05

# creates an if statement for the hypothesis using the p-value test
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')


# In[172]:


print('Kurtosis of normal distribution: {}'.format(stats.kurtosis(df.
    Oil_Price)))
print('Skewness of normal distribution: {}'.format(stats.skew(df.Oil_Price)
    ))


# In[173]:


plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
df['Oil_Price'].hist(bins=50)
plt.title('Oil')
plt.subplot(1,2,2)
stats.probplot(df['Oil_Price'], plot=plt);
df.Oil_Price.describe().T


# In[174]:


for name in X_test.columns:
    plt.figure(figsize=(14,6))
    plt.subplot(1,2,1)
    df[name].hist(bins=50)
    plt.title(name)
    plt.subplot(1,2,2)
    stats.probplot(df[name], plot=plt);
    df.Oil_Price.describe().T


# In[179]:


for name in X_test.columns:
    print(name)
    # unpacks the test statistic and p-value from the normaltest of gold
    stat, p = stats.normaltest(df[name])

    # prints both the test statistic and p-value
    print('Statistics=%.3f, p=%.3f' % (stat, p))

    # chooses an alpha of 0.05 and assigns to variable 'alpha'
    alpha = 0.05

    # creates an if statement for the hypothesis using the p-value test
    if p > alpha:
        print('Data looks Gaussian (fail to reject H0)')
    else:
        print('Data does not look Gaussian (reject H0)')


# In[10]:


from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller

fig, ax = plt.subplots(2, figsize=(10,6))
ax[0] = plot_acf(df[name], ax=ax[0])
ax[1] = plot_pacf(df[name], ax=ax[1])


# In[8]:


def adfuller_test(series, signif=0.05, name='', verbose=False):
    """ Perform ADFuller to test for Stationarity of given series and print
     report"""
    r = adfuller(series, autolag = 'BIC')
    output = {'test_statistics':round(r[0], 4), 'pvalue':round(r[1], 4), '
    n_lags':round(r[2], 4), 'n_obs':r[3]}
    p_value = output['pvalue']
    def adjust(val, length=6): return str(val).ljust(length)

    # Print Summary
    print(f'   Augmented Dickey-Fuller Test on "{name}"', "\n   ", '-'*47)
    print(f' Null Hypothesis: Data has unit root. Non-Stationary.')
    print(f' Significance level    = {signif}')
    print(f' Test Statistics       = {output["test_statistics"]}')
    print(f' No. Lags Chosen       = {output["n_lags"]}')

    for key, val in r[4].items():
          print(f' Critical value {adjust(key)} = {round(val, 3)}')

    if p_value <= signif:
          print(f" => P-Value = {p_value}. Rejecting Null Hypothesis.")
          print(f" => Series is Stationary.")

    else:
          print(f" => P-Value = {p_value}. Weak evidence to reject the Null
     Hypothesis.")
          print(f" => Series in Non-Stationary.")


# In[11]:


for name, column in df.iteritems():
```

```
          adfuller_test(column, name=column.name)
          print('\n')


# In[12]:


nobs = 15

# assigning the last 15 observations to the variable 'X_test', while the
    rest will be assigned to the variable 'X_train'
X_train, X_test = df[0:-nobs], df[-nobs:]

#check size
print(X_train.shape)
print(X_test.shape)


# In[13]:


transform_data = X_train.diff().dropna()
transform_data.head()


# In[14]:


for name, column in transform_data.iteritems():
          adfuller_test(column, name=column.name)
          print('\n')


# In[15]:


fig, axes = plt.subplots(nrows=3, ncols=3, dpi=120, figsize=(12, 6))
for i, ax in enumerate(axes.flatten()):
    data = transform_data[transform_data.columns[i]]
    ax.plot(data, color='red', linewidth=1)
    # Decorations
    ax.set_title(transform_data.columns[i])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines['top'].set_alpha(0)
    ax.tick_params(labelsize=6)

plt.tight_layout()


# In[16]:


# importing the granger causality test from statsmodels
from statsmodels.tsa.stattools import grangercausalitytests

# creating a variable named 'maxlag' and assigning the integer 5
maxlag = 5

# assigning the string 'ssr_chi2test' to the variable 'test'
test = 'ssr_chi2test'

```

```python
# creating a function named 'granger_causation_matrix' with the arguments 'data', 'variables', 'test', and 'verbose'
def granger_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):

    # creating a dataframe with the same dimensions as number of variables entered, assigned to the variable 'X_train'
    X_train = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)

    # loops through the columns and the indexes
    for c in X_train.columns:
        for r in X_train.index:

            # conducts a granger causality test on a variable row and column using the 'maxlag' variable; assigns to
            # variable test_result
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)

            # locates the test result in the tuple 'test_result' and rounds the number by 4 digits; assigns to 'p_values'
            p_values = [round(test_result[i+1][0][test][1], 4) for i in range(maxlag)]

            # if the variable is 'verbose', print the y and x variables and the associated p-value
            if verbose:
                print(f'Y = {r}, X = {c}, P Values = {p_values}')

            # find the smallest p-value and assign to variable 'min_p_value'
            min_p_value = np.min(p_values)

            # find the smallest p-value and assign to its respective row and column
            X_train.loc[r, c] = min_p_value

    # remane the row and column names based on the relationship
    X_train.columns = [var + '_x' for var in variables]
    X_train.index = [var + '_y' for var in variables]
    return X_train


# In[17]:


granger_causation_matrix(X_train, variables = X_train.columns)


# In[25]:


from statsmodels.tsa.vector_ar.vecm import coint_johansen

def cointegration_test(transfrom_data, alpha=0.05):
    """Perform Johanson's Cointegration Test and Report Summary"""
    for i in range(0,8):
        print("for i="+str(i)+'\n')
        out = coint_johansen(transform_data, -1, i)
        d = {'0.90':0, '0.95':1, '0.99':2}
        traces = out.lr1
        cvts = out.cvt[:, d[str(1-alpha)]]
        def adjust(val, length=6):
            return str(val).ljust(length)

        # Summary
        print('Name   :: Test Stat > C(95%)       => Signif   \n', '--'*20)
        for col, trace, cvt in zip(transform_data.columns, traces, cvts):
            print(adjust(col), '::', adjust(round(trace, 2), 9), ">",
    adjust(cvt, 8), '  =>  ', trace >cvt)
#     print(traces)
#     print(cvts)

cointegration_test(X_train)


# In[191]:


import statsmodels.tsa.api as smt
from statsmodels.tsa.api import VAR
mod = smt.VAR(transform_data)
res = mod.fit(maxlags= 5, ic = 'aic')
print(res.summary())


# In[192]:


pred = res.forecast(transform_data.values[-9:], 15)
pred_df = pd.DataFrame(pred, index=df.index[-15:], columns = df.columns)
pred_df


# In[193]:


# reversing the difference and assigning to variable 'pred_inverse'
pred_inverse = pred_df.cumsum()

# inverse the difference values and assigning to variable 'f'
f = pred_inverse + X_test
print(f)


# In[194]:


plt.figure(figsize=(12,5))
plt.xlabel('Date')

ax1 = X_test.Gold_Price.plot(color='blue', grid=True, label='Actual Gold
    Price')
ax2 = f.Gold_Price.plot(color='red', grid=True, secondary_y=True, label='
    Predicted Gold Price')

ax1.legend(loc=1)
ax2.legend(loc=2)
plt.title('Prediction vs Actual Gold Price')
plt.show()


# In[195]:
```

```python


plt.figure(figsize=(12,5))
plt.xlabel('Date')

for name in X_test.columns:
    label=str(name)
    ax1 = X_test[name].plot(color='blue', grid=True, label='Actual')
    ax2 = f[name].plot(color='red', grid=True, secondary_y=True, label='
    Predicted')

    ax1.legend(loc=1)
    ax2.legend(loc=2)
    plt.title(label)
    plt.show()


# In[115]:


plt.figure(figsize=(12,5))
plt.xlabel('Date')

ax1 = X_test.Oil_Price.plot(color='blue', grid=True, label='Actual Oil
    Price')
ax2 = f.Oil_Price.plot(color='red', grid=True, secondary_y=True, label='
    Predicted Oil Price')

ax1.legend(loc=1)
ax2.legend(loc=2)
plt.title('Prediction vs Actual Oil Price')
plt.show()


# In[205]:


from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt

forecast_errors = [X_test.Oil_Price[i]-f.Oil_Price[i] for i in range(len(
    X_test.Oil_Price))]
bias = sum(forecast_errors) * 1.0/len(X_test.Oil_Price)
print('Bias: %f' % bias)

mae = mean_absolute_error(X_test.Oil_Price, f.Oil_Price)
print('MAE: %f' % mae)

mse = mean_squared_error(X_test.Oil_Price, f.Oil_Price)
print('MSE: %f' % mse)

rmse = sqrt(mse)
print('RMSE: %f' % rmse)


# In[207]:


for name in X_test.columns:
    print(name)
    forecast_errors = [X_test[name][i]-f[name][i] for i in range(len(X_test
    [name]))]
    bias = sum(forecast_errors) * 1.0/len(X_test[name])
    print('Bias: %f' % bias)

    mae = mean_absolute_error(X_test[name], f[name])
    print('MAE: %f' % mae)

    mse = mean_squared_error(X_test[name], f[name])
    print('MSE: %f' % mse)

    rmse = sqrt(mse)
    print('RMSE: %f' % rmse)
    print('-'*10)
```