Forecasting Time Series of Precious Metals

Tatev Baghdasaryan
Time Series, Spring 2024
ASDS, YSU

WEEKLY REPORT

Week 1

13.05-19.05

This week I've been working on these

- Data Choice
- Getting Familiar with the Data
- Data Visualizations

Data Choice

Precious metals have always played a pivotal role in economies worldwide, serving not only as currency, but also as a protection against economic ups and downs. This is the reason I found this topic quite interesting and chose it for my final project.

Getting Familiar with the Data

The dataset offers detailed, up-to-date information on precious metals futures. Futures are financial contracts obligating the buyer to purchase, and the seller to sell a particular precious metal (such as gold, silver, platinum, etc.) at a predetermined future date and price.

Here's the link to the dataset: Gold, Silver and Precious Metals Futures Daily Data

Data Visualizations

Stock Open/Close Prices
We can first investigate the trend of the stock open and close prices over time (Figure 1).
Observations: With the exception of Gold, the open/close prices for all the stocks have dips in their stock prices around the year 2008. We can speculate this is due to the 2008 financial crisis that occurred around that time.

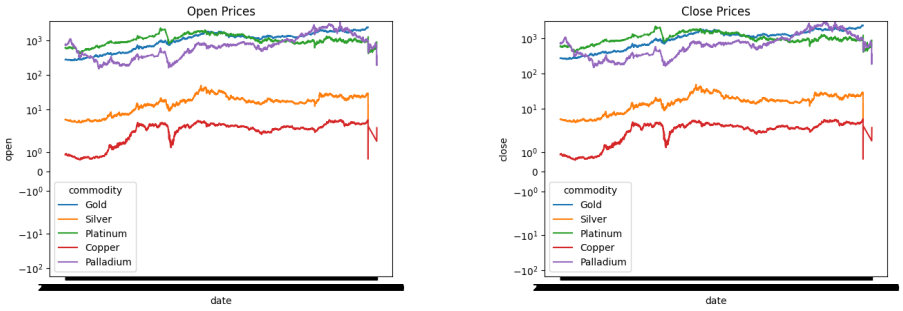

Figure 1: Open and Close Prices

Stock Volume
We can also investigate the trends of the stock volumes using a similar analysis.

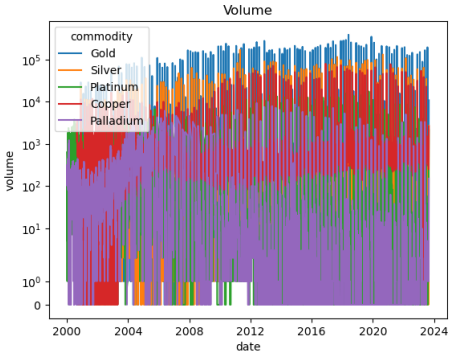

Figure 2: Volume

We see that there is a lot of variability in the data (Figure 2). So, instead, something we could do is plot the moving average of the data (Figure 3).

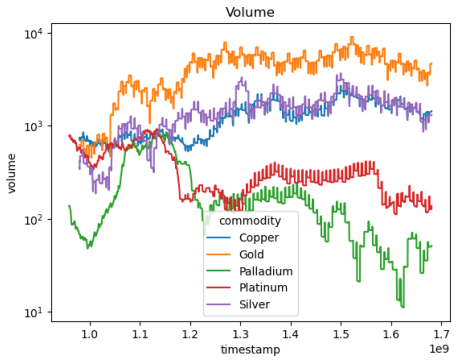

Figure 3: Moving Average

ACF
When dealing with time series data, it might be helpful to investigate the auto-correlation function (ACF) plots of the data to investigate any trends/seasonality in the data.

Now we can investigate the trends of the percentage change over time:

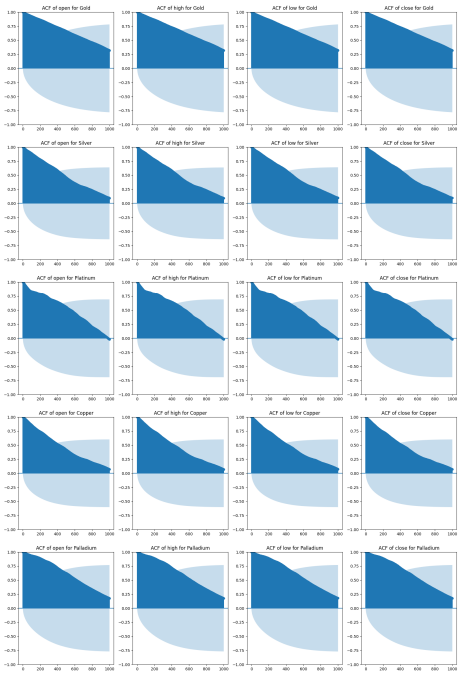

Figure 4: ACF

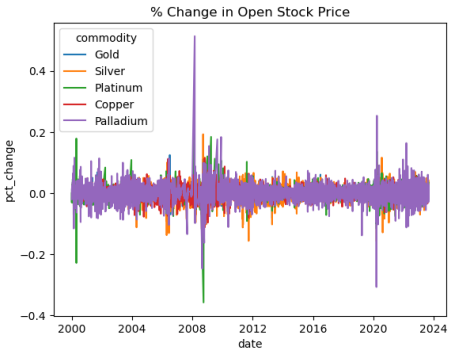

Figure 5: Percentage change

We see that there are two distinct spikes in this graph:

- one at 2008, which roughly corresponds to the 2008 financial crisis;
- and one at 2020, which roughly corresponds to the start of the COVID-19 pandemic.

Week 2

20.05-25.05

This week I've completed the following

- Stationary Test
- Feature Engineering: Adding New Features
- Making the series stationary

Stationary Test

Stationary test is a vital step in analyzing time series, so let's go on and analyze the results obtained from the Stationary Test.

Performing Augmented DF test for Gold
ADF Statistic: -0.199862
p-value: 0.938538
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for Silver
ADF Statistic: -1.573720
p-value: 0.496780
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for Copper
ADF Statistic: -1.614846
p-value: 0.475432
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for Platinum
ADF Statistic: -2.523577

p-value: 0.109843
Critical Values:
1%: -3.432
5%: -2.862
10%: -2.567

Performing Augmented DF test for Palladium
ADF Statistic: -1.483393
p-value: 0.541710
Critical Values:
1%: -3.432
5%: -2.862
10%: -2.567

We observe that the p-values of all the tests are greater than 0.05, which signifies that the time series data have unit roots, and so are non-stationary.

Feature Engineering: Adding New Features

Our goal will be to predict the (open) stock prices of the different commodities in 2023 using the previous stock data. To do this, we will need to add some features to our dataset before passing it into modelling (Listing 1):

- We should add extra features that incorporate more information about the date variate; for example, the day, week, month, day of week, etc.
- We should add columns for the open, close, low and high prices for each commodity in our dataset.

```
df["timestamp"] = df["date"].apply(lambda d: d.timestamp())
df["day"] = df["date"].dt.day
df["week"] = df["date"].dt.isocalendar().week
df["month"] = df["date"].dt.month
df["year"] = df["date"].dt.year
df["day_of_week"] = df["date"].dt.dayofweek

df.sample(5)
```

Listing 1: Python code for adding datetime specific features

Making the series stationary

To improve modelling, we should make the series stationary. To do this, we will do

- a log transformation to the open stock price
- compute the differences between the log open stock prices

```
df["log_open"] = np.log(df["open"])
for comm in commodities:
    data = df.loc[df["commodity"] == comm, "log_open"]
    df.loc[df["commodity"] == comm, "diff_log_open"] = data.diff()
    # set diff of first element to be 0, since it is currently NA
    df.loc[(df["commodity"] == comm).idxmax(), "diff_log_open"] = 0

df.head()
```

Listing 2: Python code for a log transformation

We can check whether this new variate `"diff_log_open"` is stationary:

Performing Augmented DF test for (diff_log) Gold
ADF Statistic: -33.106441
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for (diff_log) Silver
ADF Statistic: -13.614050
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for (diff_log) Copper
ADF Statistic: -17.369685
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Performing Augmented DF test for (diff_log) Platinum
ADF Statistic: -31.093501
p-value: 0.000000
Critical Values:
1%: -3.432
5%: -2.862
10%: -2.567

Performing Augmented DF test for (diff_log) Palladium
ADF Statistic: -25.070213
p-value: 0.000000
Critical Values:
1%: -3.432
5%: -2.862
10%: -2.567

As we can see, the p-values are now extremely small (basically zero), which means this transformed variate is now stationary.