1. Mô hình Arima

   Mô hình Arima dựa trên ý tưởng cho rằng chuỗi thời gian có thể được giải thích bằng cách kết hợp các hành vi hiện tại và trong quá khứ với các yếu tố ngẫu nhiên ( nhiễu) ở hiện tại và quá khứ.

   ARIMA là tổng hợp các mô hình:

   - AR (Mô hình tự hồi quy): Sử dụng giá trị của chuỗi thời gian trong quá khứ để dự đoán giá trị hiện tại
   - I (Mô hình tích hợp ): biến chuỗi thời gian thành dừng trước khi áp dụng các mô hình AR và MA
   - MA( mô hình trung bình trượt): sử dụng lỗi dự báo trong quá khứ để dự báo giá trị hiện tại → lỗi dự báo là khác biệt giữa giá trị thực tế và giá trị dự báo trong lịch sử.

   Ưu điểm:

   - Cho kết quả dự báo ngắn hạn đáng tin cậy
   - Dự báo chính xác ( trừ các trường hợp dữ liệu có biến động lớn )

   Nhược điểm:

   - Không có tính chất phản ứng nhanh
   - Chỉ dùng để dự báo ngắn hạn trong điều kiện tương đối ổn định

   Hiện nay do nền kinh tế hay có biến động lớn, như giá vàng ngày hôm qua giảm sâu nên mô hình ARIMA không phù hợp để dự đoán các biến số kinh tế trong giai đoạn hiện nay.

   Phương pháp BOX-JENKINS trong mô hình ARIMA gồm 6 bước cơ bản

   - Bước 1: Kiểm tra xem chuỗi thời gian có dừng hay không (Chọn cột cần kiểm tra, sử dụng kiểm định ADF để kiểm tra tính dừng của chuỗi thời gian)
     Kiểm định ADF trả về giá trị p, nếu p < 0.05 là chuỗi dừng và ngược lại
   - Bước 2: Xử lý chuỗi không dừng
     Dùng phương pháp lấy hiệu : Tạo ra một chuỗi mới trong đó mỗi giá tị là hiệu quả giá trị hiện tại và giá trị liền trước nó
   - Bước 3: Chọn bậc AR(p) tối ưu -> Có thể dùng hàm auto_arima tự động hóa quấ trình chọn tham số
   - Bước 4: Chọn bậc MA(q) tối ưu -> Có thể dùng hàm auto_arima tự động hóa quấ trình chọn tham số
   - Bước 5: Ước lượng mô hình ARIMA (p, q, d) và chọn mô hình tối ưu
   - Bước 6: Dự báo

2. Mô hình Garch

   Mô hình Garch hữu ích trong việc mô hình hóa và dự báo các biến động của chuỗi thời gian tài chính.

   Mô hình GARCH giúp nắm bắt các đặc điểm biến động theo thời gian của dữ liệu tài chính bằng cách mô hình hóa sự biến đổi của phương sai có điều kiện theo thời gian.

   - Heteroskedasticity: Phương sai của lỗi thay đổi theo thời gian, không cố định. Phương sai của lỗi là đo lường mức độ mà các giá trị lỗi ( sai lệch giữa giá trị thực tế và giá trị dự báo) phân tán quanh giá trị trung bình của chúng. Phương sai này thay đổi theo thời gian không cố định

   Cách mô hình GARCH hoạt động trong thực tế:

Bước 1: Tính tỉ lệ lợi suất hàng ngày bằng cách lấy tỉ lệ phần trăm thay đổi của giá cổ phiếu giữa các ngày liên tiếp
Bước 2: Tính Toán Lợi Suất Hàng Ngày
Lợi suất hàng ngày được tính bằng cách lấy tỷ lệ phần trăm thay đổi của giá cổ phiếu giữa các ngày liên tiếp
Bước 3: Khởi tạo mô hình GARCH

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

Trong đó:

- $\sigma_t^2$ là phương sai có điều kiện tại thời điểm $t$.
- $\epsilon_{t-i}^2$ là lỗi bình phương tại thời điểm $t - i$.
- $\sigma_{t-j}^2$ là phương sai có điều kiện tại thời điểm $t - j$.

Bước 4: Sử dụng dữ liệu lịch sử để ước lượng tham số
Bước 5: Dự báo sự thay đổi của lợi suất ( biến động ) trong tương lai

3. Chạy mô hình
   3.1 Tiền xử lý dữ liệu

| | date | truong_1 | truong_2 | truong_3 | truong_4 | truong_5 |
|---|---|---|---|---|---|---|
| 0 | 10.05.2013 | 4 | 58 | 3773 | 299.0 | 1 |
| 1 | 26.05.2013 | 4 | 58 | 3768 | 249.0 | 1 |
| 2 | 19.05.2013 | 4 | 58 | 4036 | 419.0 | 1 |
| 3 | 25.05.2013 | 4 | 58 | 12878 | 149.0 | 1 |
| 4 | 15.05.2013 | 4 | 58 | 12885 | 148.0 | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| 550033 | 07.11.2013 | 10 | 37 | 18474 | 199.0 | 1 |
| 550034 | 18.11.2013 | 10 | 37 | 18474 | 199.0 | 1 |
| 550035 | 24.11.2013 | 10 | 37 | 18484 | 199.0 | 1 |
| 550036 | 11.11.2013 | 10 | 37 | 19751 | 99.0 | 1 |
| 550037 | 26.11.2013 | 10 | 37 | 18498 | 199.0 | 1 |

550038 rows × 6 columns

Ta thấy dữ liệu chưa được sắp xếp theo cách ngày liên tiếp nhau, kiểm tra thấy trong dữ liệu chỉ có 214 ngày

```
# Kiểm tra số ngày duy nhất trong cột "date"
unique_dates = df['date'].nunique()

print(f"Số ngày duy nhất trong cột 'date': {unique_dates}")
```

✓ 0.0s

```
Số ngày duy nhất trong cột 'date': 214
```

Sau khi sắp xếp lại các ngày, thấy 1 ngày có nhiều giá trị ở các trường

```
df['date'] = pd.to_datetime(df['date'], format='%d.%m.%Y')

# Sắp xếp dữ liệu theo cột "date"
df = df.sort_values(by='date')
df
```

✓ 0.0s

| | date | truong_1 | truong_2 | truong_3 | truong_4 | truong_5 |
|---|---|---|---|---|---|---|
| 10740 | 2013-05-01 | 4 | 15 | 3476 | 799.0 | 1 |
| 13910 | 2013-05-01 | 4 | 10 | 1904 | 159.0 | 1 |
| 13907 | 2013-05-01 | 4 | 10 | 1915 | 154.0 | 1 |
| 28695 | 2013-05-01 | 4 | 7 | 12180 | 999.0 | 1 |
| 2691 | 2013-05-01 | 4 | 57 | 10928 | 199.0 | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| 537467 | 2013-11-30 | 10 | 25 | 20276 | 399.0 | 1 |
| 537478 | 2013-11-30 | 10 | 25 | 20448 | 399.0 | 1 |
| 537480 | 2013-11-30 | 10 | 25 | 20506 | 698.0 | 1 |
| 537326 | 2013-11-30 | 10 | 26 | 2808 | 999.0 | 1 |
| 537736 | 2013-11-30 | 10 | 26 | 7872 | 699.0 | 1 |

550038 rows × 6 columns

Ta thấy dữ liệu trên khá giống với giao dịch của cửa hàng trong các ngày:

`date`: Ngày giao dịch.

`truong_1`: Mã sản phẩm.

`truong_2`: Số lượng bán.

`truong_3`: Giá đơn vị.

`truong_4`: Lợi nhuận.

`truong_5`: Trạng thái giao dịch (hoàn thành, hủy bỏ,...).

Vậy ta sẽ dùng truong_4 và tính tổng lợi nhận cho mỗi ngày để tiến hành dự báo.

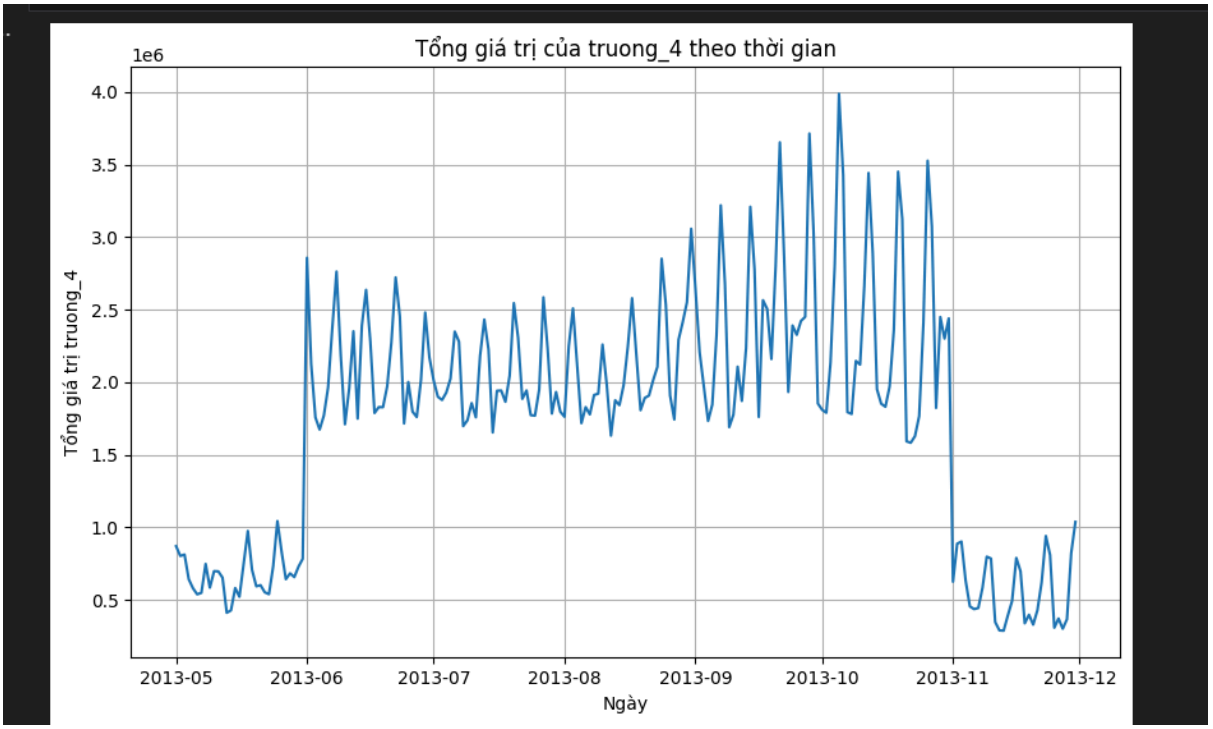
```
df_sum = df.groupby('date')['truong_4'].sum().reset_index()

# Đổi tên cột 'truong_4' thành 'sum_truong_4' để rõ ràng hơn
df_sum.rename(columns={'truong_4': 'sum_truong_4'}, inplace=True)

# Thiết lập định dạng hiển thị cho cột 'sum_truong_4'
pd.options.display.float_format = '{:.2f}'.format

# Hiển thị DataFrame mới
df_sum
```

[28] ✓ 0.0s

...

| | date | sum_truong_4 |
|---|---|---|
| 0 | 2013-05-01 | 869941.13 |
| 1 | 2013-05-02 | 802586.67 |
| 2 | 2013-05-03 | 812157.50 |
| 3 | 2013-05-04 | 642066.03 |
| 4 | 2013-05-05 | 580066.32 |
| ... | ... | ... |
| 209 | 2013-11-26 | 371267.00 |
| 210 | 2013-11-27 | 301100.17 |
| 211 | 2013-11-28 | 366603.90 |
| 212 | 2013-11-29 | 822282.69 |
| 213 | 2013-11-30 | 1037198.00 |

214 rows × 2 columns




3.2 Chạy mô hình ARIMA

Kiểm tra và chuyển dữ liệu sang chuỗi dừng

```
data = df_sum["sum_truong_4"]
data
from statsmodels.tsa.stattools import adfuller

result = adfuller(data)
print('ADF Statistic:', result[0])
print('p-value:', result[1])
```

✓ 0.6s

```
ADF Statistic: -1.907354187206557
p-value: 0.3286094174037726
```

```
data_diff = data.diff().dropna()

result = adfuller(data_diff)
print('ADF Statistic (differenced):', result[0])
print('p-value (differenced):', result[1])
```

✓ 0.0s

```
ADF Statistic (differenced): -3.7530194824900565
p-value (differenced): 0.0034262683981900574
```

Kịch bản: Sử dụng 15 ngày cuối trong data đưa vào tập test

```
train_data = data_diff.iloc[:-15]
test_data = data_diff.iloc[-15:]
```

Chọn tham số và chạy mô hình ARIMA

```
# Tự động chọn tham số ARIMA bằng auto_arima
model_auto = auto_arima(train_data, seasonal=False, trace=True, error_action='ignore', suppress_warnings=True)

# In ra các tham số được chọn
print(model_auto.summary())

# Khớp mô hình ARIMA với các tham số được chọn
model = ARIMA(train_data, order=model_auto.order)
```

```
from pmdarima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
p = 7
d = 1
q = 11

model = ARIMA(train_data, order=(p, d, q))
model_fit = model.fit()

print(model_fit.summary())
```

```
forecast = model_fit.forecast(steps=15)
forecast.index = test_data.index

plt.figure(figsize=(12, 6))
plt.plot(data_diff, label='Actual')
plt.plot(forecast, label='Forecast', color='red')
plt.legend()
plt.show()

forecast_df = pd.DataFrame({'Actual': test_data, 'Forecast': forecast})
print(forecast_df)
```