BÀI KIỂM TRA PHÂN TÍCH CHUỖI THỜI GIAN

Đề 3: Phân tích Sarima, Arimax

Họ Và Tên: Đỗ Duy Đức

MSV: 2151264650

Bài 1:

1. ARIMAX

ARIMAX là một phương pháp dự báo thống kê giữa mô hình ARIMA và các biến số mở rộng X từ các chuỗi thời gian khác. Nó cho phép dự báo chuỗi thời gian dựa trên sự ảnh hưởng của các biến số bên ngoài. Điều này làm tăng tính linh hoạt của mô hình ARIMA và thích hợp cho các tình huống khi chuỗi thời gian có thể bị ảnh hưởng bởi các yếu tố bên ngoài

- Ưu điểm:

- Giúp tăng tính linh hoạt
- Có khả năng xử lý nhiễu
- Có thể áp dụng trong nhiều lĩnh vực như tài chính, kinh tế, ….

- Nhược điểm:

- Yêu cầu một lượng dữ liệu lớn
- Khó khăn trong việc lựa chọn tham số

2. SARIMA

SARIMA là một phương pháp dự báo chuỗi thời gian kết hợp giữa mô hình ARIMA và các thành phần mùa vụ. Nó được sử dụng để dự báo các chuỗi thời gian có mùa vụ như doanh số bán hàng hàng tháng hoặc doanh thu hàng quý. SARIMA cho phép mô hình hóa sự biến đổi theo mùa vụ và các yếu tố khác trong dữ liệu, giúp cải thiện độ chính xác trong dự báo

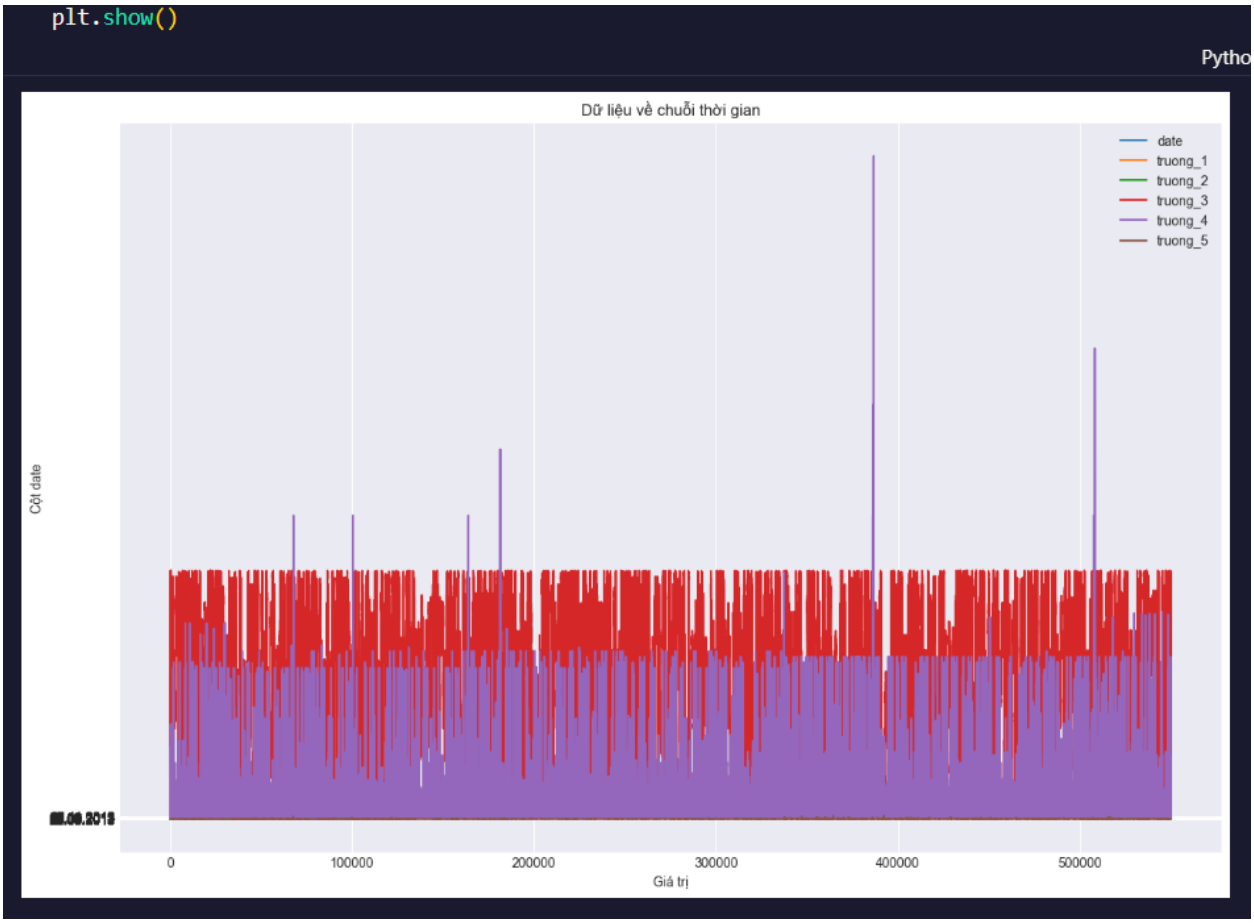
- Ưu điểm:

- Xử lý tốt các chuỗi thời gian có tính mùa vụ
- Phù hợp với nhiều loại chuỗi thời gian khác nhau
- Dự báo với độ chính xác cao

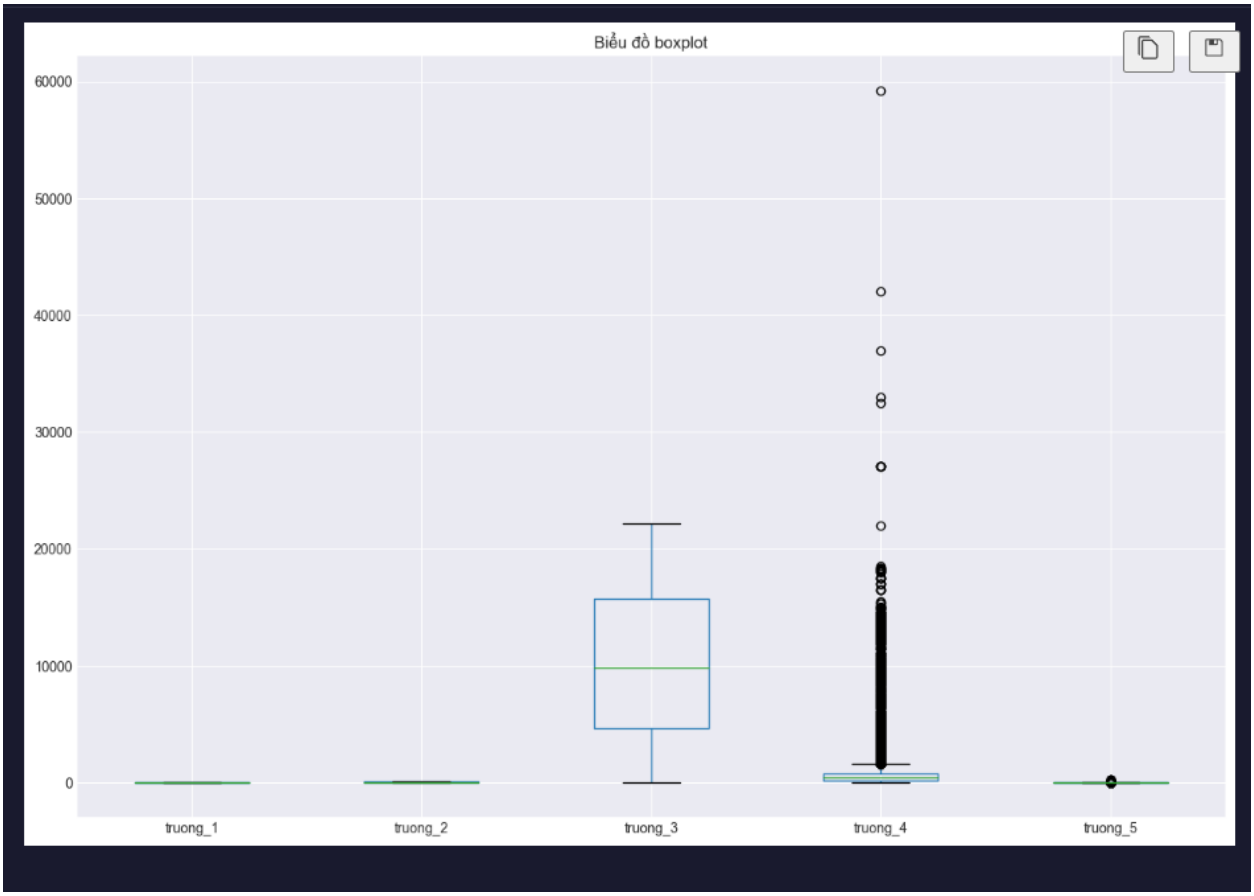
- Nhược điểm

- Yêu cầu lượng dữ liệu đủ lớn để xây dựng một mô hình có độ chính xác cao
- Khó khăn trong lựa chọn tham số cho mô hình

Bài 2:



Biểu đồ này là một biểu đồ đường (line chart) biểu thị dữ liệu của một chuỗi thời gian cho các cột trong dataframe df. Mỗi cột được hiển thị trên cùng một biểu đồ, với trục x biểu thị các giá trị của cột "date" và trục y biểu thị các giá trị của các cột khác. Biểu đồ này giúp bạn quan sát sự biến động của các biến theo thời gian và cũng có thể giúp phát hiện các mẫu, xu hướng và biên độ trong dữ liệu.



Biểu đồ boxplot để thể tim ra được outliers trong tập dữ liệu

SỬ DỤNG KỸ THUẬT IQR ĐỂ XỬ LÝ CÁC OUTLIERS

```
def cap_outliers_iqr(df):
    capped_df = df.copy()
    for column in df.columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        capped_df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
        capped_df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
    return capped_df

data_capped = cap_outliers_iqr(df)
```

Em sử dụng kỹ thuật IQR để xử lý dữ liệu ouliers

ÁP DỤNG TRUNG BÌNH TRƯỢT ĐỂ GIẢM NHIỄU

+ Code + Markdown

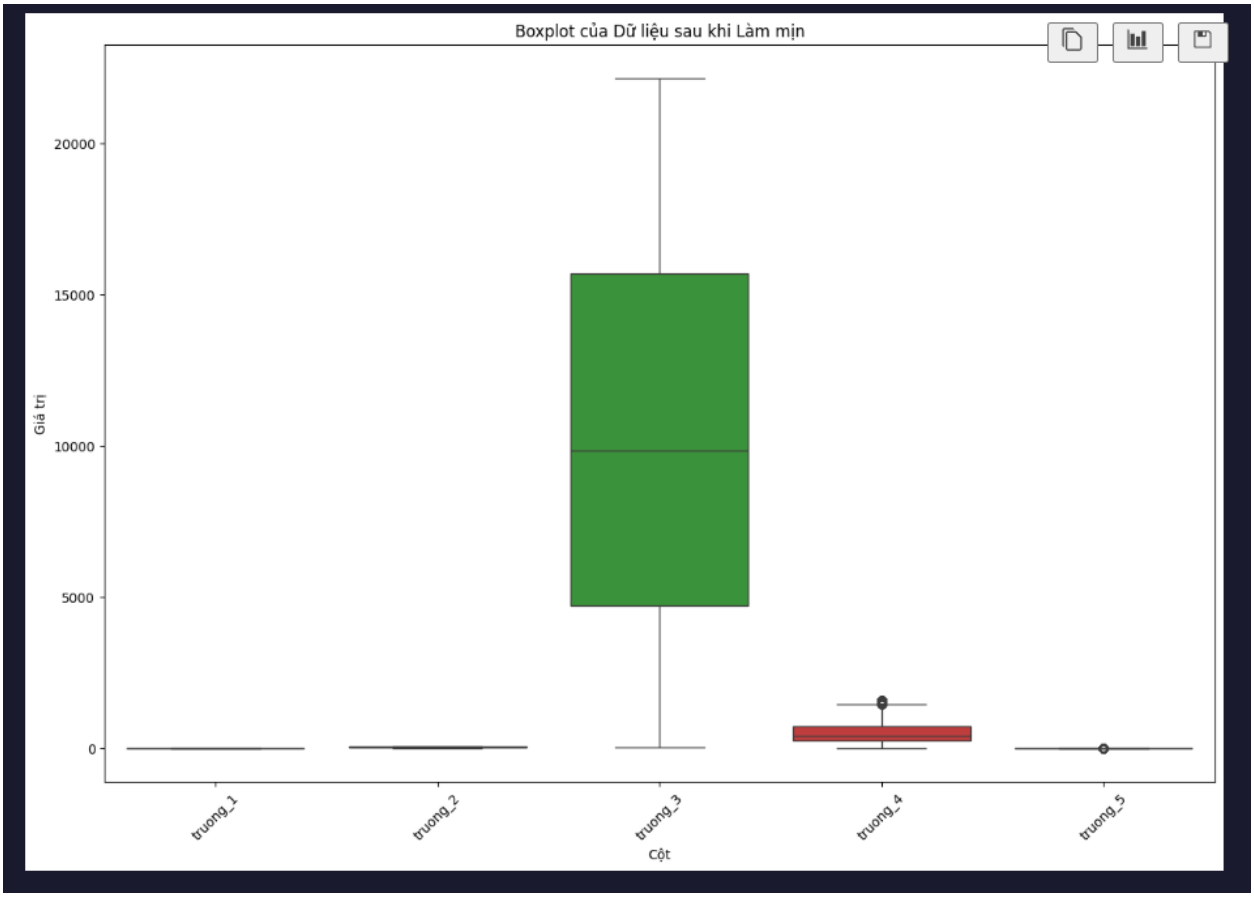
```
data_smoothed = data_capped.rolling(window=3, min_periods=1).mean()

print(data_smoothed.describe())

# Vẽ biểu đồ boxplot của dữ liệu đã làm mịn
plt.figure(figsize=(15, 10))
sns.boxplot(data=data_smoothed)
plt.title('Boxplot của Dữ liệu sau khi Làm mịn')
plt.xlabel('Cột')
plt.ylabel('Giá trị')
plt.xticks(rotation=45)
plt.show()
```

[6] ✓ 7.2s

Sau đó e áp dụng trung bình trượt để xử lý dữ liệu nhiễu



Kết quả sau khi xử lý outliers và làm mịn dữ liệu

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))

data_min_max_scaled = pd.DataFrame(scaler.fit_transform(data_capped), columns=df.columns, index=(
```

Tiếp theo, em chuẩn hóa dữ liệu bằng phương pháp MinMaxScaler

Phương pháp chuẩn hóa bằng MinMaxScaler là một kỹ thuật chuẩn hóa dữ liệu trong quy trình tiền xử lý dữ liệu. Cụ thể, nó chuyển đổi các giá trị của biến thành một phạm vi cụ thể, thường là từ 0 đến 1

Quá trình này giúp đưa các giá trị của biến về cùng một phạm vi, giúp dễ dàng so sánh và hiểu được ý nghĩa của chúng. MinMaxScaler thường được sử dụng trong các trường hợp khi phân phối của các biến không quá lệch và khi có sự cần thiết đảm bảo các giá trị nằm trong một phạm vi cụ thể, chẳng hạn như trong việc huấn luyện mô hình máy học hoặc các thuật toán dự báo. Tuy nhiên, nếu dữ liệu có nhiễu nhiều hoặc có phân

phối không đồng đều, MinMaxScaler có thể không phù hợp và các phương pháp chuẩn hóa khác như StandardScaler thường được ưu tiên.

Kết quả sau chuẩn hóa

```
data_min_max_scaled
```

date	truong_1	truong_2	truong_3	truong_4	truong_5
2013-05-10	0.0	0.982456	0.169188	0.189926	1.0
2013-05-26	0.0	0.982456	0.168963	0.158158	1.0
2013-05-19	0.0	0.982456	0.181067	0.266168	1.0
2013-05-25	0.0	0.982456	0.580416	0.094623	1.0
2013-05-15	0.0	0.982456	0.580733	0.093988	1.0
...	...	...	...	...	...
2013-11-07	1.0	0.614035	0.833160	0.126391	1.0
2013-11-18	1.0	0.614035	0.833160	0.126391	1.0
2013-11-24	1.0	0.614035	0.833612	0.126391	1.0
2013-11-11	1.0	0.614035	0.890836	0.062855	1.0
2013-11-26	1.0	0.614035	0.834244	0.126391	1.0

ARIMAX

ARIMAX

CHIA DỮ LIỆU

```
# Chia dữ liệu thành phần huấn luyện và phần kiểm tra
train_size = int(len(data_min_max_scaled) * 0.8)  # 80% dữ liệu cho huấn luyện
train_data_min_max_scaled = data_min_max_scaled.iloc[:train_size]
test_data_min_max_scaled = data_min_max_scaled.iloc[train_size:]
```

52] Python

TRAIN MODEL

```
train_data_min_max_scaled['date'] = train_data_min_max_scaled.index.astype(int) // 10**9
test_data_min_max_scaled['date'] = test_data_min_max_scaled.index.astype(int) // 10**9

# Huấn luyện mô hình ARIMAX
exog_train_data_min_max_scaled = train_data_min_max_scaled[['date', 'truong_1', 'truong_2', 'truo
arimax_model = ARIMA(train_data_min_max_scaled['truong_3'], exog=exog_train_data_min_max_scaled,
arimax_result = arimax_model.fit()


print(arimax_result.summary())
```

60]

```
Dự đoán và đánh giá mô hình ARIMAX trên dữ liệu kiểm tra
c:\Users\DELL\AppData\Local\Programs\Python\Python310\lib\site-pack
  return get_prediction_index(
c:\Users\DELL\AppData\Local\Programs\Python\Python310\lib\site-pack
  return get_prediction_index(
ARIMAX Mean Squared Error: 0.26545941846378807
ARIMAX Mean Absolute Error: 0.42698140540453045
```

Kết quả của các chỉ số dự đoán độ chính xác của mô hình, với kết quả này ta thấy được mô hình có kết quả dự đoán khá tốt

SARIMA

SARIMA

CHIA DỮ LIỆU

```
# Chia dữ liệu thành phần huấn luyện và phần kiểm tra
train_size = int(len(data_min_max_scaled) * 0.6)
train_data_min_max_scaled = data_min_max_scaled.iloc[:train_size]
test_data_min_max_scaled = data_min_max_scaled.iloc[train_size:]
```

[9] ✓ 0.0s Pyth

TRAIN MODEL

```
# Chọn cột chuỗi thời gian mục tiêu và các biến giải thích nếu có
y = train_data_min_max_scaled[['truong_3']]
from sklearn.preprocessing import StandardScaler

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
y_scaled = scaler.fit_transform(y)


# Huấn luyện mô hình SARIMA
sarima_model = SARIMAX(y_scaled, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
sarima_result = sarima_model.fit()
```

Kết quả của SARIMA

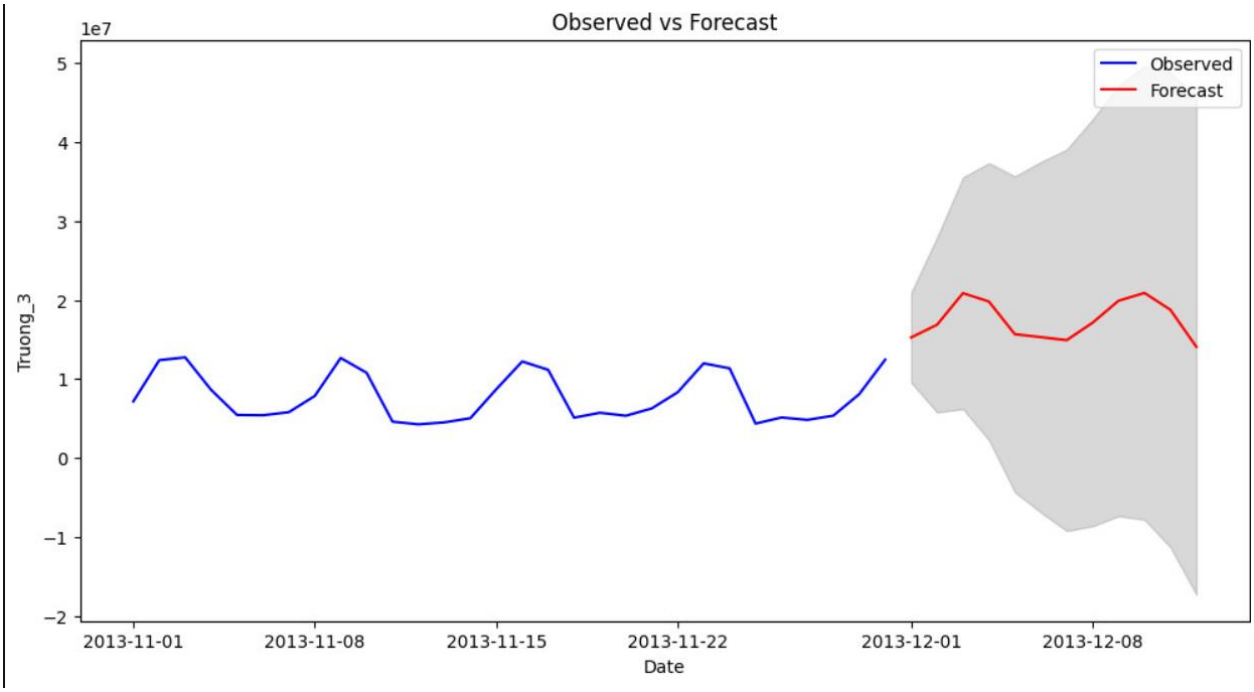
SARIMAX Results

Dep. Variable:	truong_3	No. Observations:	30
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	Log Likelihood	13.046
Date:	Tue, 04 Jun 2024	AIC	-12.092
Time:	04:44:49	BIC	-6.260
Sample:	0	HQIC	-11.512
	- 30		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
truong_1	1.0212	0.126	8.111	0.000	0.774	1.268
truong_5	-0.0386	0.123	-0.313	0.754	-0.280	0.203
ar.L1	-0.9862	4.195	-0.235	0.814	-9.209	7.237
ma.L1	0.4386	0.752	0.583	0.560	-1.036	1.913
ar.S.L12	-0.6882	2.625	-0.262	0.793	-5.834	4.457
ma.S.L12	-0.7927	49.408	-0.016	0.987	-97.630	96.045
sigma2	0.0035	0.149	0.024	0.981	-0.288	0.295

Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	5.61
Prob(Q):	0.90	Prob(JB):	0.06
Heteroskedasticity (H):	0.28	Skew:	1.13
Prob(H) (two-sided):	0.15	Kurtosis:	4.67

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



https://github.com/ducjr/TH5_PhanTichChuoiThoiGian