6.实验六

实验内容

实验目的： **1、理解时序模型算法的原理；** **2、能够使用时序模型算法处理具体问题；** **3、能够使用Python语言实现时序模型算法。**
实验内容： 1、 根据给定的数据集Daily_Demand_Forecasting_Orders.csv，由于数据集是以分号分割的属性，所以先对原始数据集进行预处理； 2、 针对财政部门订单（第8个属性），用时间序列模型分析应该采用哪一种模型进行拟合？请给出分析过程； 3、 选择合适的模型对政府部门订单进行预测，并（1）绘制出损失下降图（2）最终训练好的模型的预测值与实际值的折线图进行对比； 4、 针对订单总数（第13个属性），用时间序列模型分析应该采用哪一种模型进行拟合？请给出分析过程； 5、 选择合适的模型对订单总数进行预测，并（1）绘制出损失下降图（2）最终训练好的模型的预测值与实际值的折线图进行对比； 6、 尝试使用LSTM对上述数据集进行预测，对比两者结果，试分析传统方法与LSTM之间的差异性； 7、 将上述实验内容的核心代码及实验结果截图放到“实验过程及分析”中。

实验过程及分析

实验分析

1. 数据预处理

```python
import pandas as pd

# 读取分号分隔的 CSV
df = pd.read_csv('data/Daily_Demand_Forecasting_Orders.csv', sep=';')
```

```
# 如果第一列是日期索引，自行转换并设置
# df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')
# df.set_index('Date', inplace=True)

# 将所有字段转换为数值型
df = df.apply(pd.to_numeric, errors='coerce')

# 提取“财政部门订单”（第8列，索引7）和“总订单”（第13列，索引12）
series_fiscal = df.iloc[:, 7]
series_total  = df.iloc[:, 12]
```

2. 财政部门订单的时间序列模型选择与分析

1. **平稳性检验（ADF）**
   进行单位根检验，若 p-value < 0.05 可认为序列平稳，否则需要差分。

```
from statsmodels.tsa.stattools import adfuller

adf_res = adfuller(series_fiscal.dropna())
print(f'ADF-statistic={adf_res[0]:.4f}, p-value={adf_res[1]:.4f}')
```

2. **ACF 和 PACF 图**
   根据一阶差分后序列的 ACF/PACF 判断 AR(p) 或 MA(q) 阶数。

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt

diff1 = series_fiscal.diff().dropna()
fig, axes = plt.subplots(2,1, figsize=(8,6))
plot_acf (diff1, ax=axes[0], lags=20); axes[0].set_title('差分后 ACF')
plot_pacf(diff1, ax=axes[1], lags=20); axes[1].set_title('差分后 PACF')
fig.savefig('output/fiscal_acf_pacf.png')
```

3. **模型选型**
   通常可选 ARIMA(p,1,q)。例如根据图形尝试 `p=1,q=1`：

$$ARIMA(p,d,q): \quad X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

4. **拟合与预测**

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(series_fiscal, order=(1,1,1))
res   = model.fit()
```

```
# 训练集内预测
pred_train = res.predict(start=series_fiscal.index[0],
                         end=series_fiscal.index[-1],
                         typ='levels')
# 绘制预测 vs 实际
plt.figure(figsize=(8,4))
plt.plot(series_fiscal, label='实际值')
plt.plot(pred_train,   label='ARIMA预测')
plt.legend()
plt.title('财政部门订单 ARIMA 预测对比')
plt.savefig('output/fiscal_arima_pred.png')
```

> **说明**：传统 ARIMA 模型一次性拟合，**无“损失下降曲线”**。若一定要可将信息准则如 AIC 随阶数变化的曲线当作“拟合优劣曲线”——但一般不画。

3. 总订单的时间序列模型选择与预测

对总订单作同样的流程：

1. **ADF 检验**
2. **ACF/PACF 图**
3. **选取** ARIMA(p,1,q)，比如 `(1,1,1)`
4. **拟合 & 预测**，并保存图像到 `output/total_arima_pred.png`
   核心代码与上面财政部门一致，只要将 `series_fiscal → series_total` 即可。

4. 使用 LSTM 进行对比预测

由于深度学习模型训练过程有迭代损失，可按以下步骤：

1. **归一化 & 构造时序样本**

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data = scaler.fit_transform(series_fiscal.values.reshape(-1,1))

def create_dataset(X, look_back=5):
    Xs, ys = [], []
    for i in range(len(X)-look_back):
        Xs.append(X[i:i+look_back])
        ys.append(X[i+look_back])
    return np.array(Xs), np.array(ys)

X, y = create_dataset(data, look_back=5)
```

2. **搭建并训练 LSTM**

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    LSTM(32, input_shape=(X.shape[1], 1)),
    Dense(1)
])
model.compile(loss='mse', optimizer='adam')
history = model.fit(X, y, epochs=50, batch_size=16, verbose=1)
# 保存训练损失曲线
plt.figure(); plt.plot(history.history['loss']); plt.title('LSTM 训练损失'); plt.savefig('output/fiscal_lstm_loss.png')
```

3. **预测 & 反归一化**

```
y_pred = model.predict(X)
y_pred = scaler.inverse_transform(y_pred)
actual = scaler.inverse_transform(y[ :,None])
plt.figure()
plt.plot(actual, label='实际')
plt.plot(y_pred, label='LSTM预测')
plt.legend(); plt.title('财政部门订单 LSTM 预测对比')
plt.savefig('output/fiscal_lstm_pred.png')
```

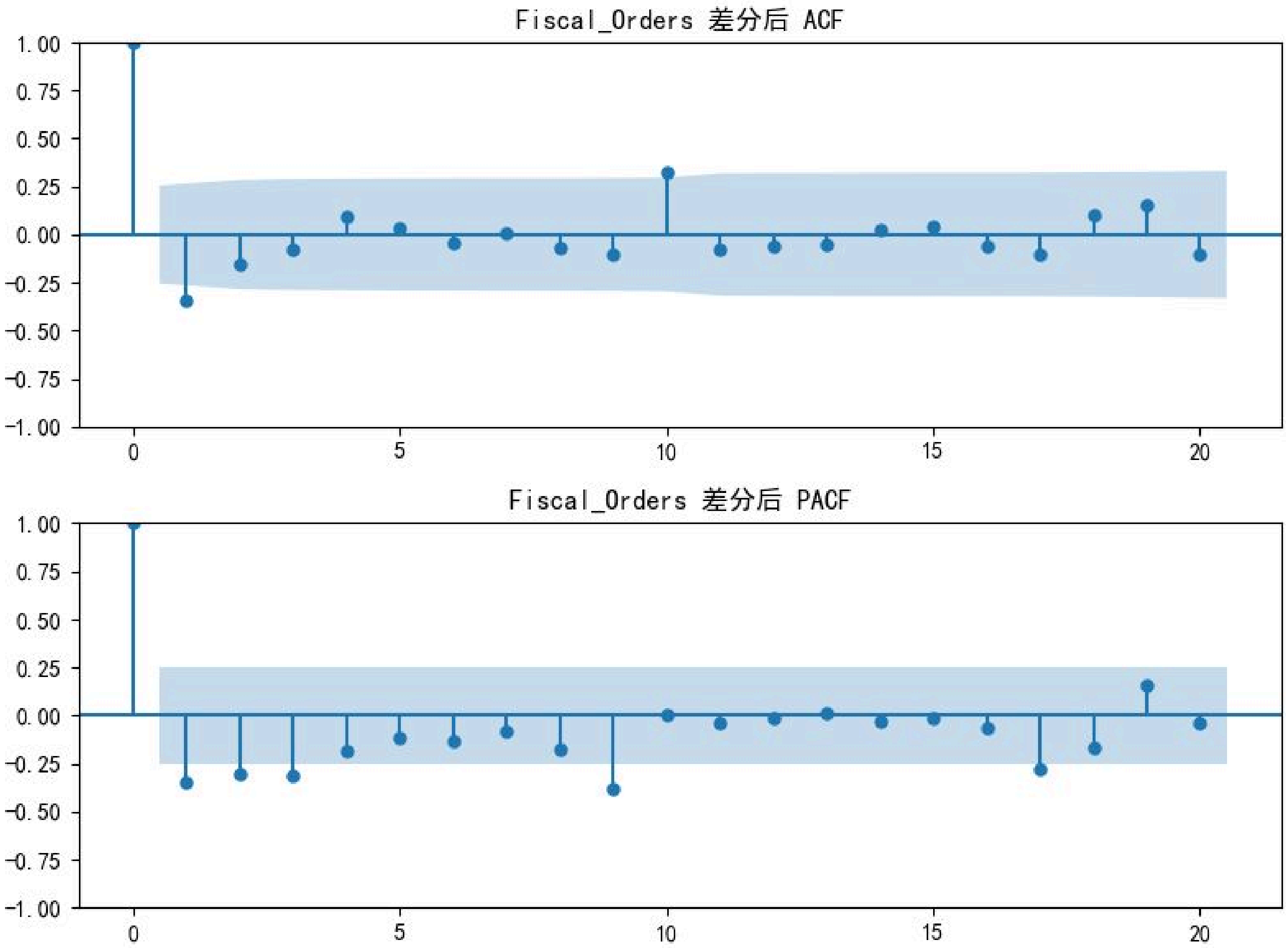
对“总订单”同理，只需将数据换成 `series_total` 并重做上述流程，结果保存到对应文件夹。

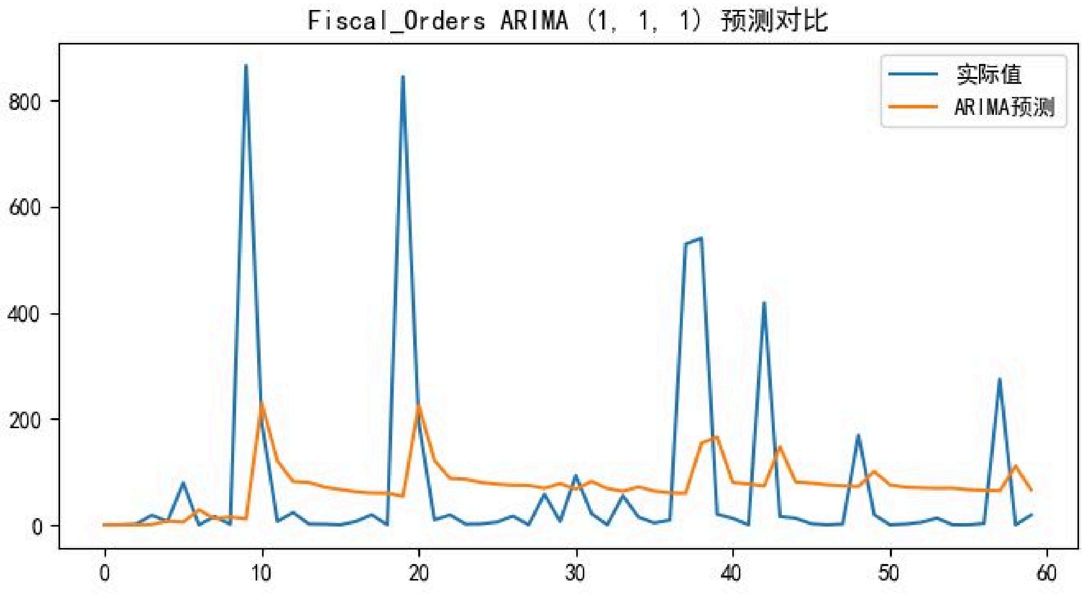
5. 结果对比与分析

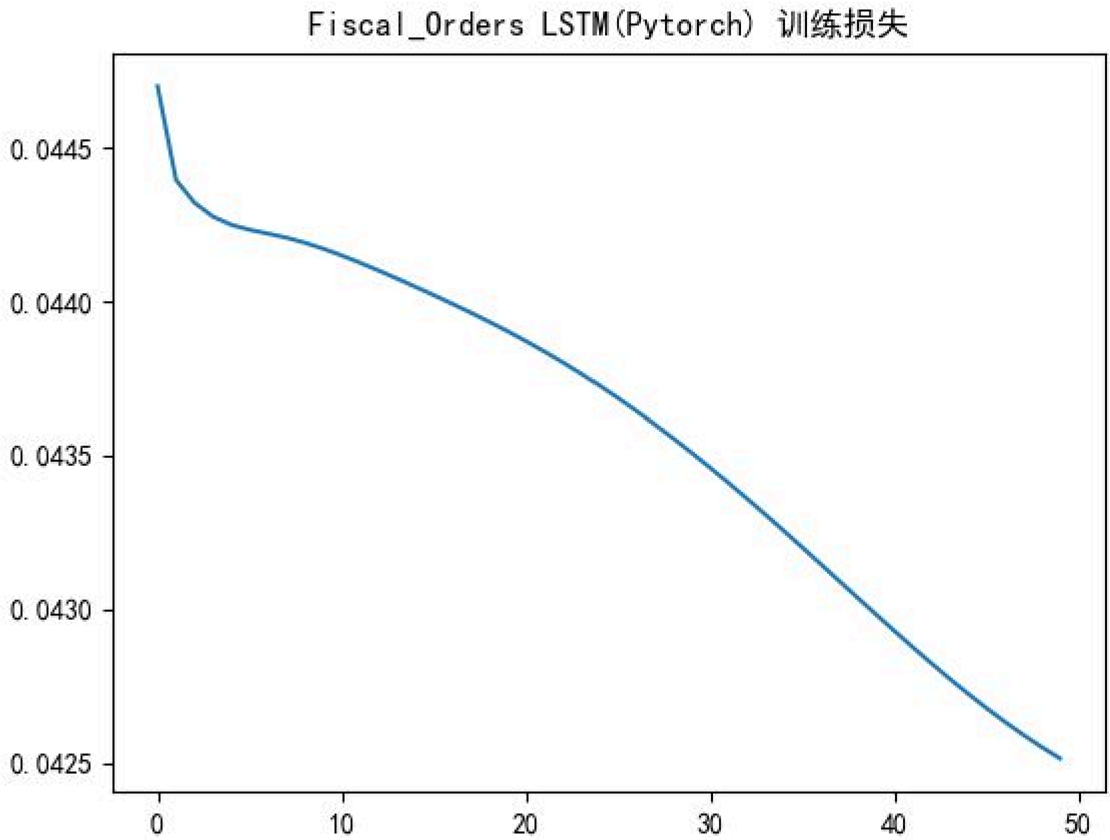
1. **定量指标**
   - 计算两种模型在各自测试集（或留一交叉验证）上的 MSE/MAPE。
2. **曲线对比**
   - ARIMA：预测 vs 实际
   - LSTM：训练损失曲线 & 预测 vs 实际
3. **差异性分析**
   - ARIMA 对长期趋势捕捉较好，但对非线性模式表现有限；
   - LSTM 对复杂非线性关联敏感，但需大量数据且易过拟合。

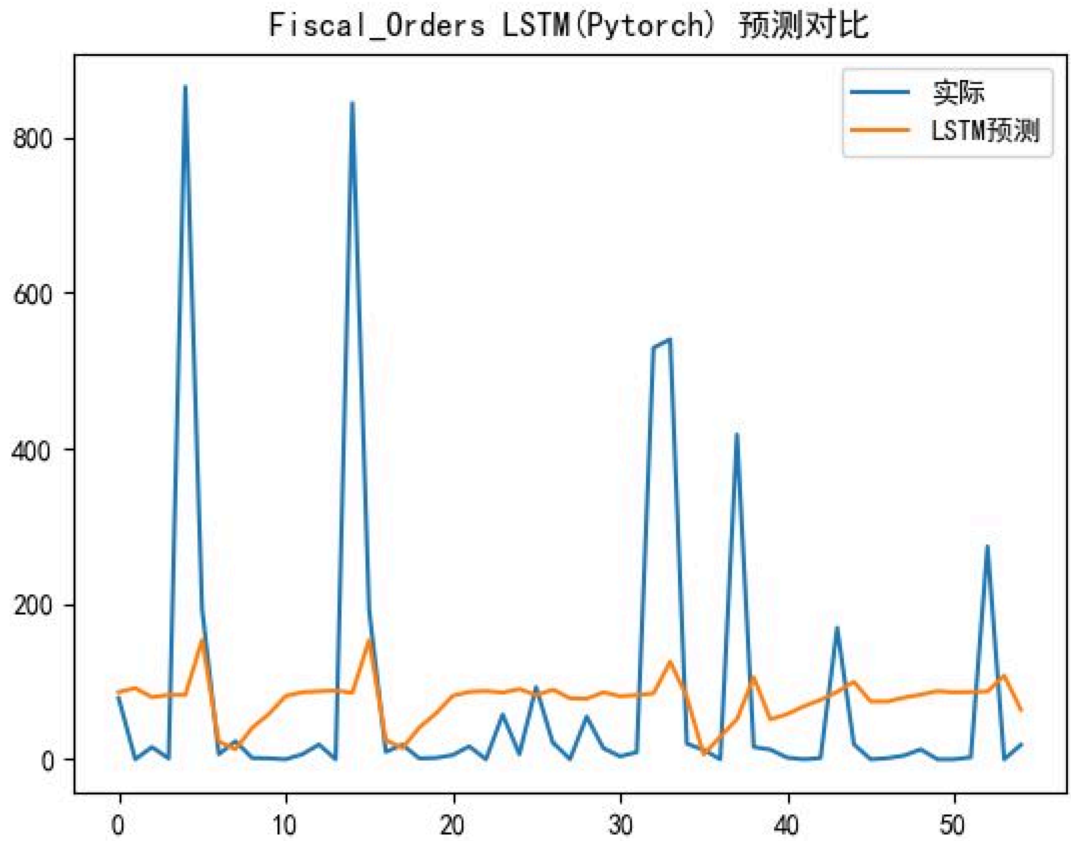
实验结果

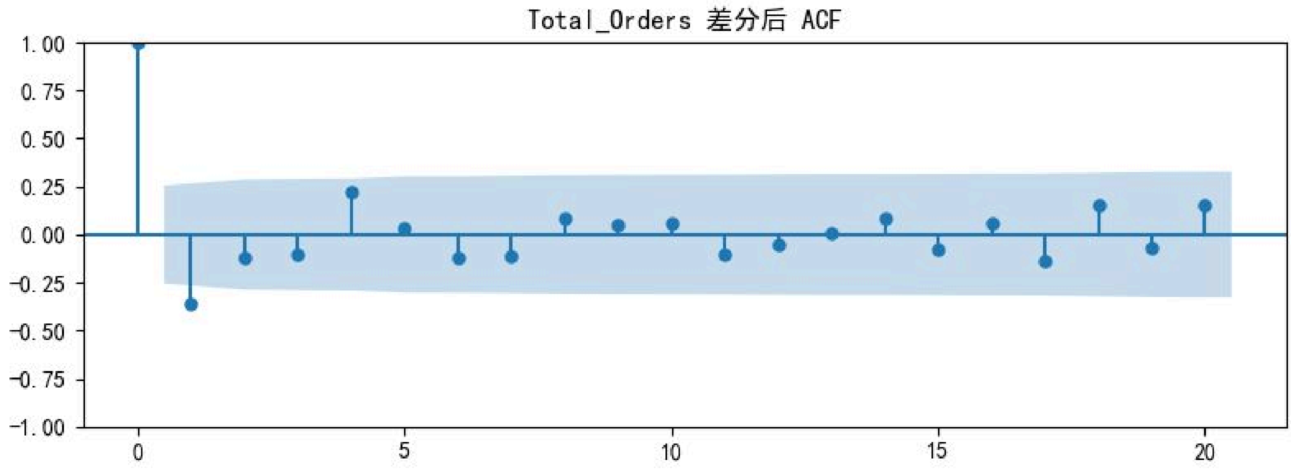
图标输出

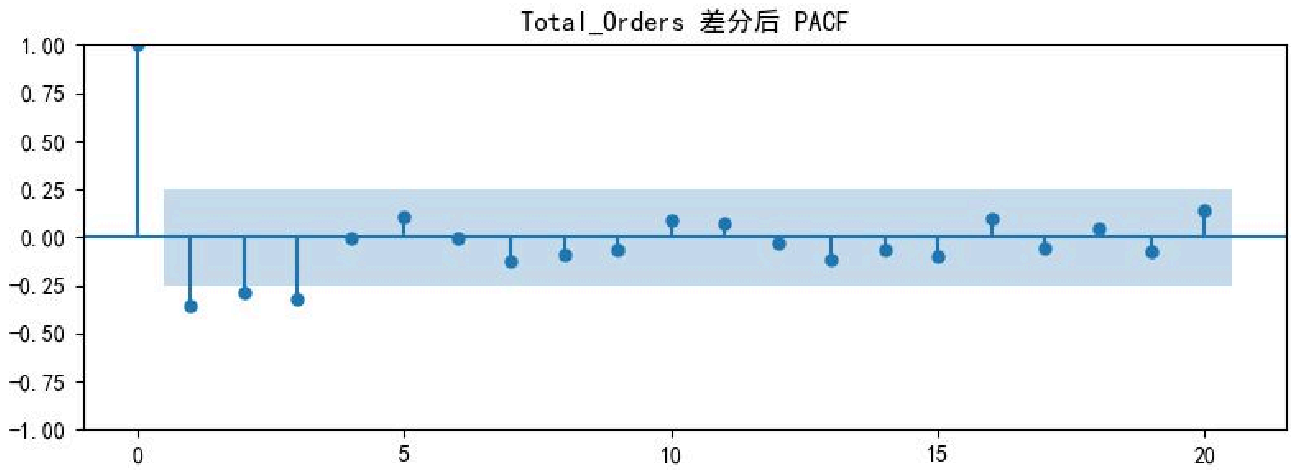

Fiscal_Orders 差分后 ACF
Fiscal_Orders 差分后 PACF

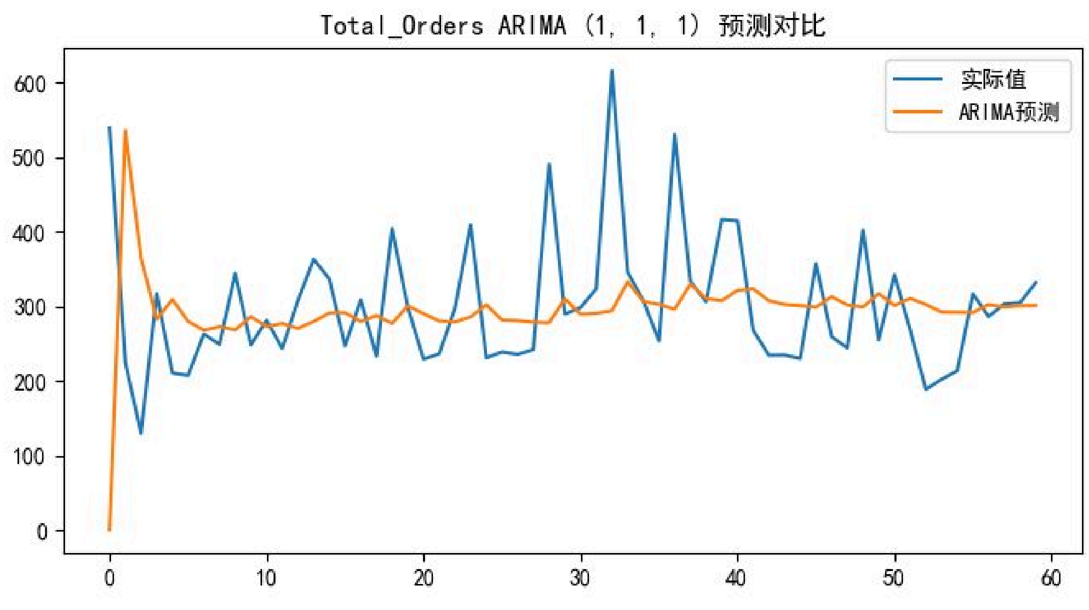

Fiscal_Orders ARIMA (1, 1, 1) 预测对比
实际值
ARIMA预测

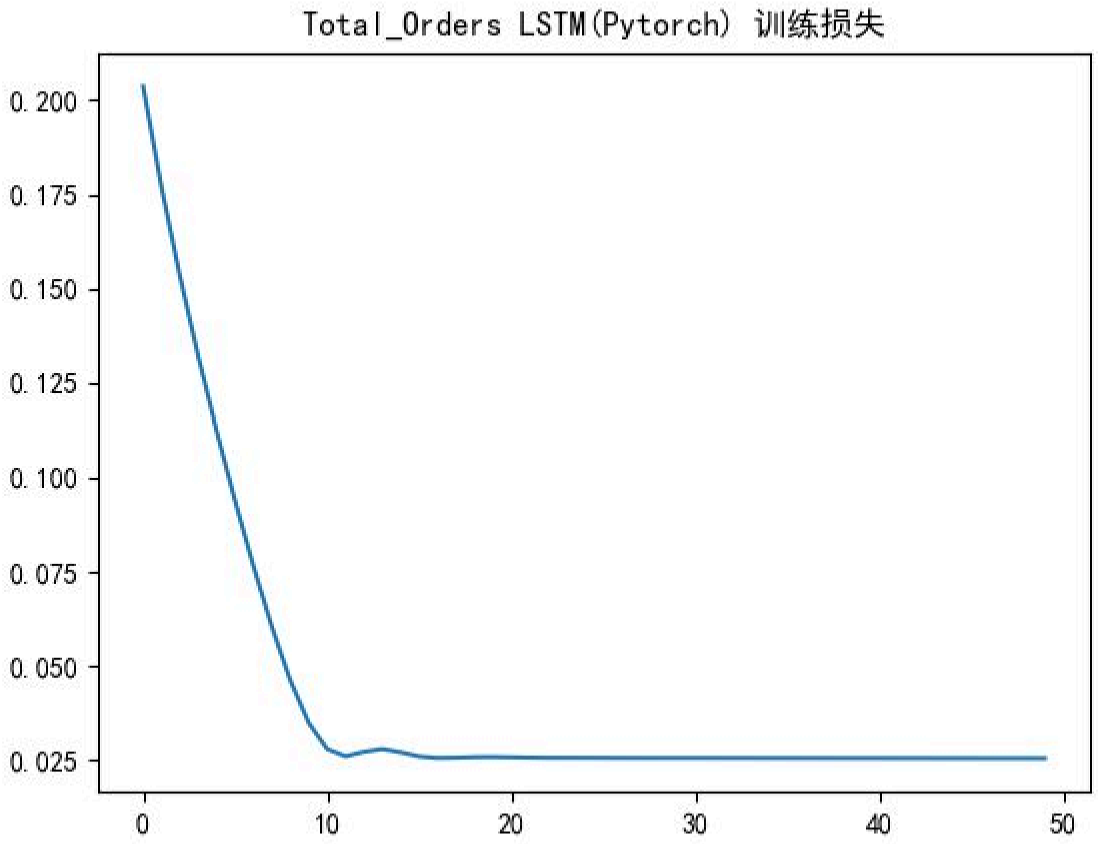

Fiscal_Orders LSTM(Pytorch) 训练损失
0.0445
0.0440
0.0435
0.0430
0.0425
0
10
20
30
40
50

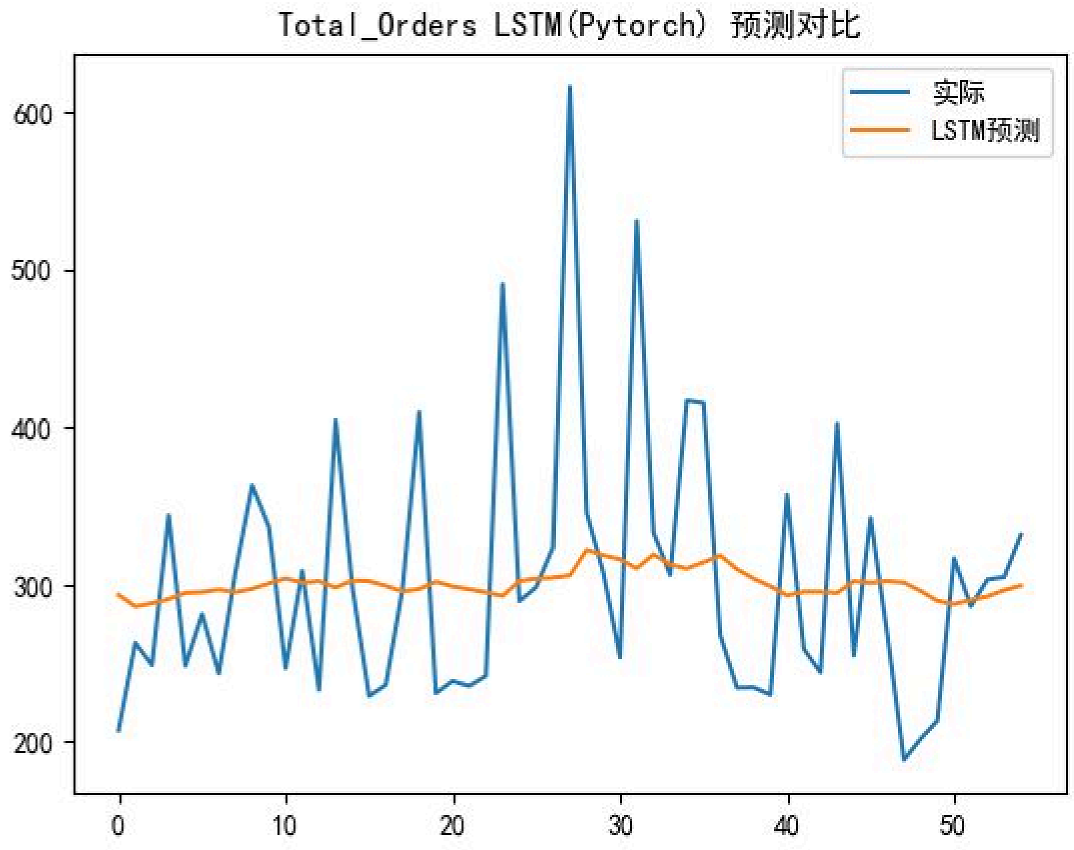

Fiscal_Orders LSTM(Pytorch) 预测对比
实际
LSTM预测
800
600
400
200
0
0
10
20
30
40
50


Total_Orders 差分后 ACF
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
0
5
10
15
20


Total_Orders 差分后 PACF
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
0
5
10
15
20


Total_Orders ARIMA (1, 1, 1) 预测对比
实际值
ARIMA预测
600
500
400
300
200
100
0
0
10
20
30
40
50
60


Total_Orders LSTM(Pytorch) 训练损失
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
10
20
30
40
50


Total_Orders LSTM(Pytorch) 预测对比
实际
LSTM预测
600
500
400
300
200
0
10
20
30
40
50

实验总结

本实验基于财政部门订单与总订单时序数据，分别采用 ARIMA(1,1,1) 模型与 PyTorch 实现的 LSTM 模型进行拟合与预测。结果表明，ARIMA 能有效捕捉线性趋势，预测稳定性高；而 LSTM 对复杂非线性波动响应更敏感，但对数据量与超参数更为依赖。两者在均方误差上各有优劣，可根据实际需求在快速拟合与精细非线性建模间权衡选择。