PREDICT SALES OF PRODUCT - TIME SERIES FORECASTING

Business

Name: DO TAN THANH NGUYEN - GROUP 4

Table of contents

01 Overview

About data

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Kaggle

Link

Dictionary

Column	Description
InvoiceNo	Số hóa đơn
StockCode	Mã sản phẩm
Description	Tên sản phẩm
Quantity	Số lượng bán
InvoiceDate	Ngày hóa đơn
UnitPrice	Đơn giá ($)
CustomerID	Mã khách hàng
Country	Khu vực

Insight

Lựa chọn sản phẩm có doanh thu cao nhất trong khu vực đứng top 1 về doanh thu -> dự đoán tình hình kinh doanh của SP này trong 17 tuần tiếp theo

02 Exploration & Preprocessing

```
df = df.drop(['CustomerID'] , axis = 1)
```

```
[136] df[pd.isnull(df['Description'])]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country
622	536414	22139	NaN	56	01-12-2010 11:52	0.0	United Kingdom
1970	536545	21134	NaN	1	01-12-2010 14:32	0.0	United Kingdom
1971	536546	22145	NaN	1	01-12-2010 14:33	0.0	United Kingdom
1972	536547	37509	NaN	1	01-12-2010 14:33	0.0	United Kingdom
1987	536549	85226A	NaN	1	01-12-2010 14:34	0.0	United Kingdom
...	...	...	...	...	...	...	...
535322	581199	84581	NaN	-2	07-12-2011 18:26	0.0	United Kingdom
535326	581203	23406	NaN	15	07-12-2011 18:31	0.0	United Kingdom
535332	581209	21620	NaN	6	07-12-2011 18:35	0.0	United Kingdom
536981	581234	72817	NaN	27	08-12-2011 10:33	0.0	United Kingdom
538554	581408	85175	NaN	20	08-12-2011 14:06	0.0	United Kingdom

1454 rows × 7 columns

```
[137] df = df.dropna(subset= ['Description'])
```

```
df['Price'] = df['Quantity'] * df['UnitPrice']
```

```
df
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Price
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.75	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.39	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.39	United Kingdom	20.34
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	09-12-2011 12:50	0.85	France	10.20
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	09-12-2011 12:50	2.10	France	12.60
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	09-12-2011 12:50	4.15	France	16.60
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	09-12-2011 12:50	4.15	France	16.60
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	09-12-2011 12:50	4.95	France	14.85

540455 rows × 8 columns

Xóa cột Customer

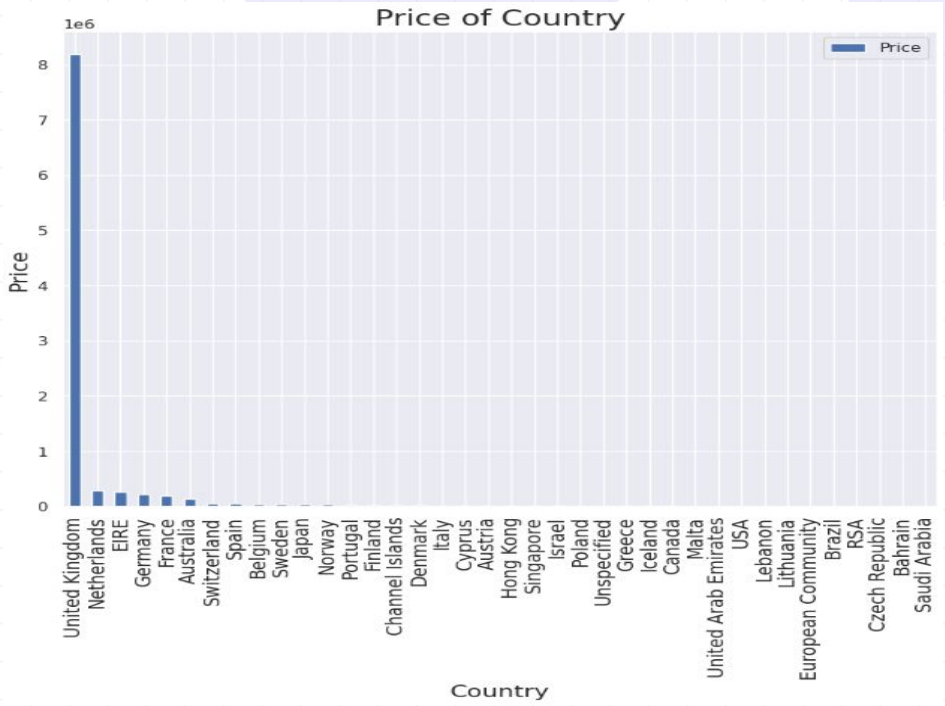
Xóa các dòng null ở cột Description

Price = Quantity x UnitPrice

02 Exploration & Preprocessing

```
df_country = df.groupby(by=[
    'Country'
]).aggregate({
    #'Quantity': sum,
    'Price': sum
}).sort_values(by=['Price'], ascending=False)
df_country
```

Country	Price
United Kingdom	8187806.364
Netherlands	284661.540
EIRE	263276.820
Germany	221698.210
France	197403.900
Australia	137077.270
Switzerland	56385.350
Spain	54774.580
Belgium	40910.960
Sweden	36595.910

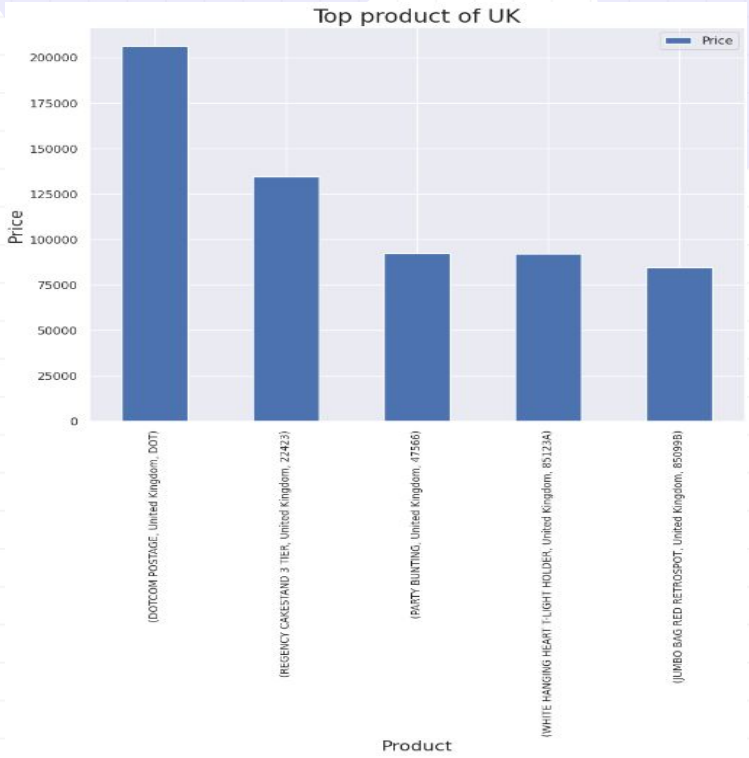


—--> Doanh thu cao nhất: UK

02 Exploration & Preprocessing

```
df_product = df1.groupby(by=[
    'Description',
    'Country',
    'StockCode'
]).aggregate({
    'Quantity': sum,
    'InvoiceNo': lambda x: len(set(x)),
    'Price': sum
}).sort_values(by=['Price','Quantity'], ascending=False).head(5)
df_product
```

Description	Country	StockCode	Quantity	InvoiceNo	Price
DOTCOM POSTAGE	United Kingdom	DOT	707	709	206245.48
REGENCY CAKESTAND 3 TIER	United Kingdom	22423	10376	1832	134405.94
PARTY BUNTING	United Kingdom	47566	16709	1613	92501.73
WHITE HANGING HEART T-LIGHT HOLDER	United Kingdom	85123A	32901	2139	92000.59
JUMBO BAG RED RETROSPOT	United Kingdom	85099B	43167	1978	84516.44



—----> Dự đoán doanh thu của SP Dotcom Postage ở khu vực UK

02 Exploration & Preprocessing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Before

```
dcp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 709 entries, 2010-12-01 14:32:00 to 2011-12-09 10:26:00
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Price   709 non-null    float64
dtypes: float64(1)
memory usage: 11.1 KB
```

After

```
dcp_df
```

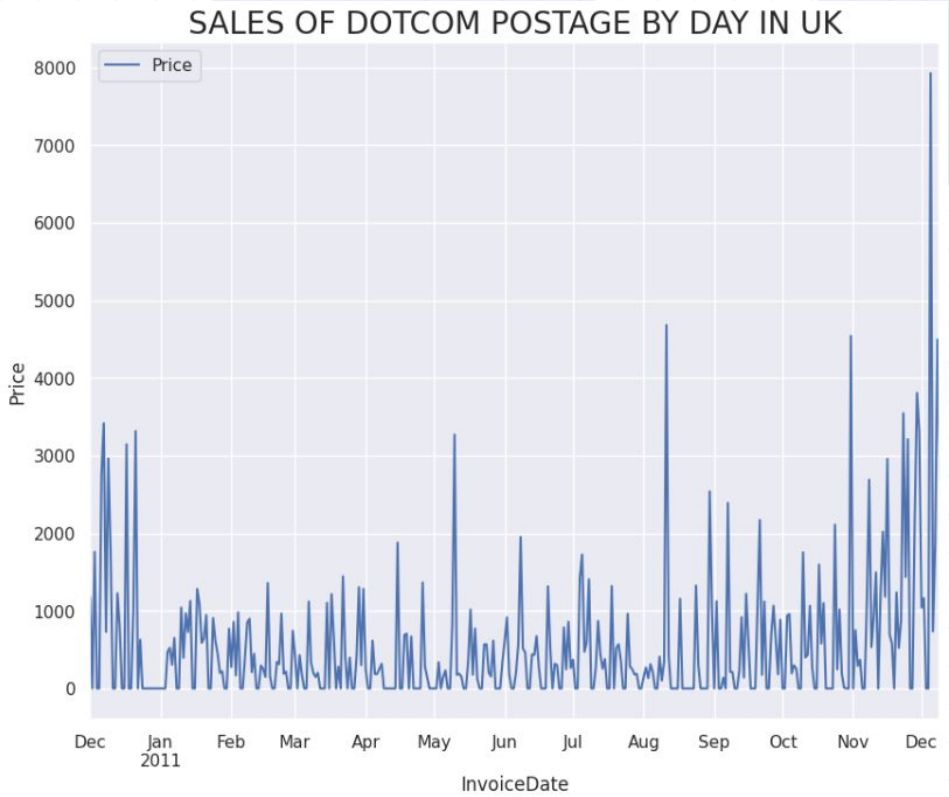
InvoiceDate	Price
2010-12-01 14:32:00	569.77
2010-12-01 17:06:00	607.49
2010-12-03 11:13:00	254.43
2010-12-03 11:27:00	121.06
2010-12-03 11:28:00	498.47
...	...
2011-12-08 09:28:00	1008.96
2011-12-08 10:53:00	1683.75
2011-12-08 16:30:00	938.59
2011-12-09 10:03:00	933.17
2011-12-09 10:26:00	1714.17

709 rows × 1 columns

Data

02 Exploration & Preprocessing

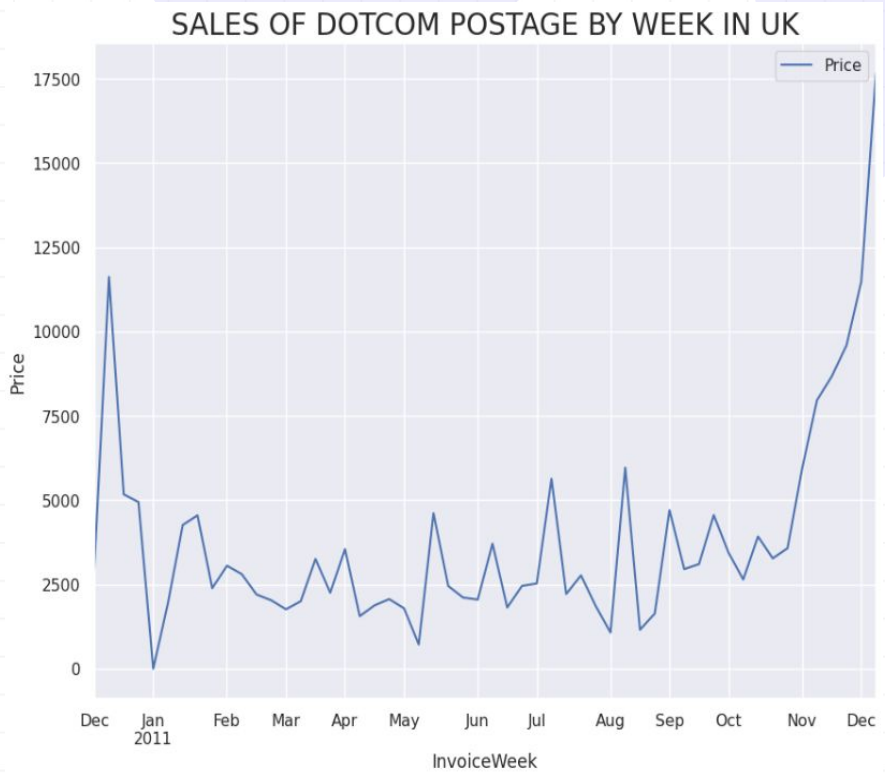
```
dcp_df.resample('D').sum().plot()
plt.title("SALES OF DOTCOM POSTAGE BY DAY IN UK",fontsize=20)
plt.ylabel('Price', fontsize=12)
plt.xlabel('InvoiceDate', fontsize=12)
plt.show()
```



02 Exploration & Preprocessing

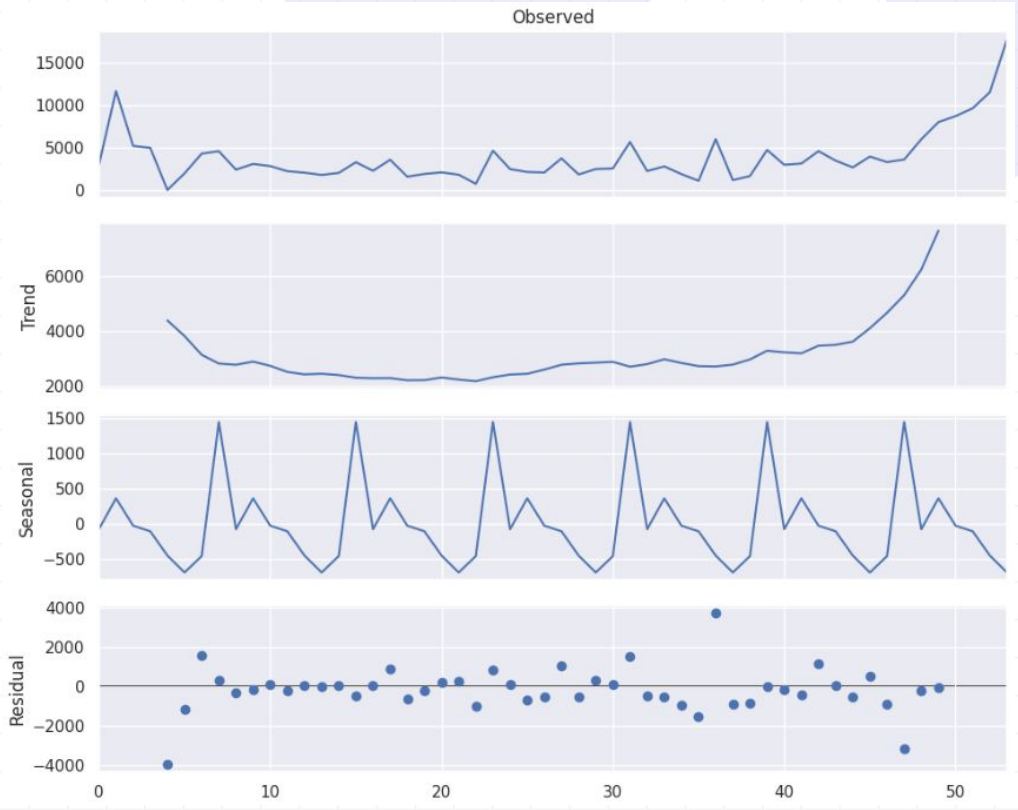
```
dcp_df.resample('W').sum().plot()
plt.title("SALES OF DOTCOM POSTAGE BY WEEK IN UK",fontsize=20)
plt.ylabel('Price', fontsize=12)
plt.xlabel('InvoiceWeek', fontsize=12)
plt.show()
```

—----> Dự đoán doanh thu của SP Dotcom Postage ở khu vực UK theo tuần



02 Exploration & Preprocessing

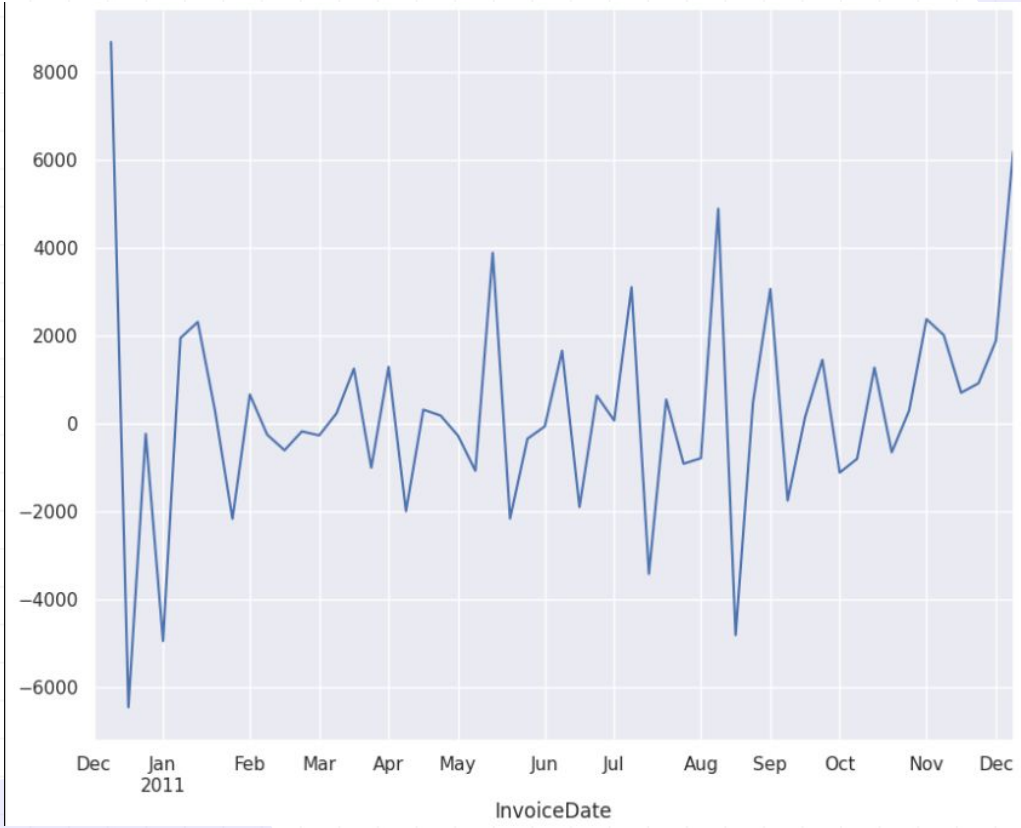
```
# seasonal decompose
from statsmodels.tsa.seasonal import seasonal_decompose

decomposed = seasonal_decompose(
    dcp_w_df['Price'].values,
    model='additive',
    filt=None,
    period=8,
    two_sided=True,
    extrapolate_trend=0,
)
decomposed.plot();
```



Train Test Split Model

```
# split train test with test = 0.3 * train
split = int(0.7 * len(dcp_w_df))
ts_data = dcp_w_df.iloc[:, 0].values
train = ts_data[:split]
test = ts_data[split:]
```

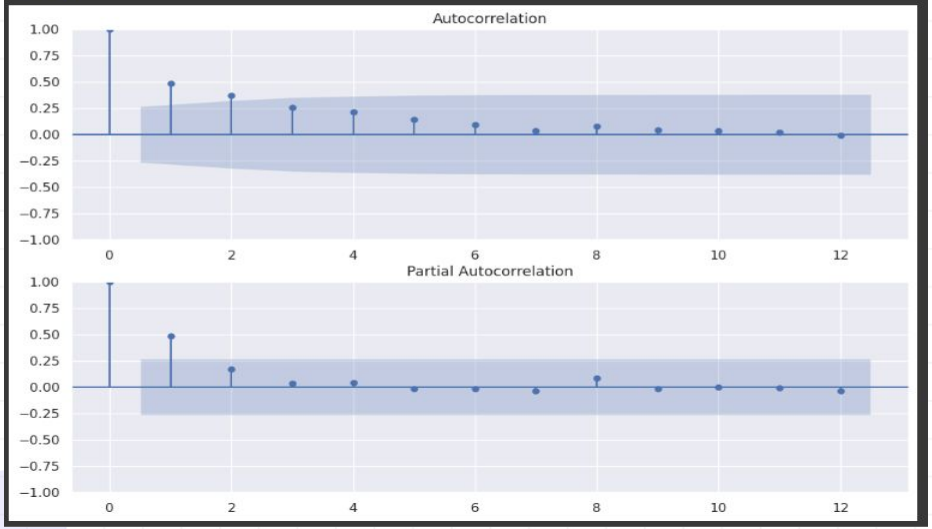
03 Build Model




```
dcp_w_df['Price'].diff().plot()
```

03 Build Model

```python
# plot acf and pacf (no diff)
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
data = dcp_w_df['Price'].copy()
fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(211)
fig = plot_acf(data, lags=12, ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(data, lags=12, ax=ax2)
plt.show();
```

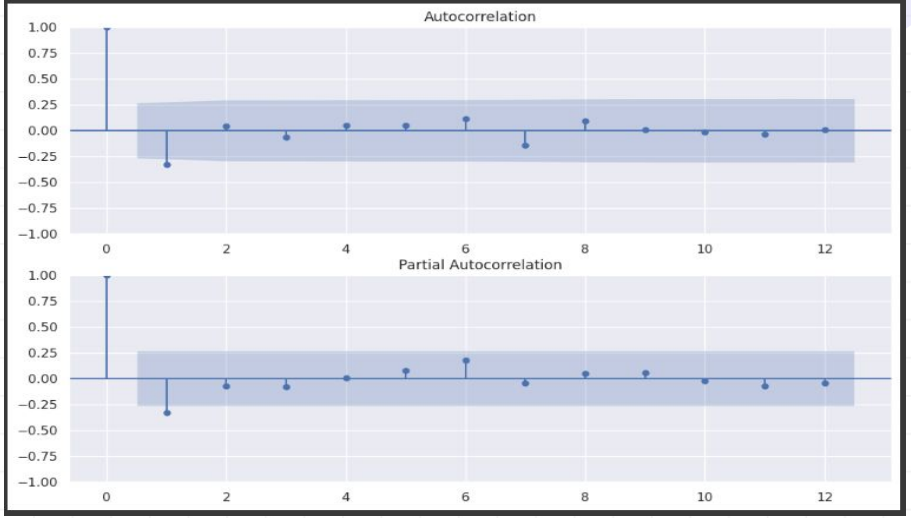


With no diff

ARIMA (2,1,2)
ARIMA(3,1,2)

With diff

```python
# plot acf and pacf (with diff)
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
data = dcp_w_df['Price'].diff()[1:].copy()
fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(211)
fig = plot_acf(data, lags=12, ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(data, lags=12, ax=ax2)
plt.show();
```



03 Build Model

```
# ARIMA(2,1,2)
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(
    train,
    order=(2, 1, 2),
    seasonal_order=(0, 0, 0, 0),
)  # define ARIMA model
fitted_model = model.fit()  # fit ARIMA model
fitted_model.predict()  # make prediction on training
```

```
array([   0.        , 1869.02560464, 6349.27043107, 5015.17820929,
       4958.79634798, 3353.93935895, 3469.97762168, 3953.46919278,
       4057.88869385, 3547.80832617, 3604.25038609, 3506.4909452 ,
       3300.28356657, 3190.52474864, 3044.66232363, 3027.34757862,
       3237.39896909, 3028.40323459, 3257.57879024, 2858.15449528,
       2853.43921634, 2858.96995067, 2757.65981503, 2508.93644736,
       3216.87276132, 2848.22083833, 2751.18960289, 2730.17228087,
       3021.15778677, 2686.20838512, 2767.77382228, 2787.90298681,
       3365.46430275, 2790.6286859 , 2861.21638326, 2703.61981422,
       2512.18633819])
```

ARIMA (2,1,2)

```
# ARIMA(3,1,2)
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(
    train,
    order=(3, 1, 2),
    seasonal_order=(0, 0, 0, 0),
)  # define ARIMA model
fitted_model = model.fit()  # fit ARIMA model
fitted_model.predict()  # make prediction on training
```

```
array([   0.        , 2232.00393306, 9028.19665353, 4138.53927304,
        472.46761859, 3159.97513079, 3308.0393118 , 6341.46594692,
       4726.59485005, 2953.11110677, 3192.54460056, 4168.06361133,
       3429.83809599, 3491.05956475, 3605.03552467, 3608.93627642,
       3940.14585353, 3428.87874415, 3122.02438848, 3149.00145011,
       2546.32159196, 3626.9958437 , 3196.05121614, 2829.63462806,
       3865.65131728, 3681.45140023, 1649.6093754 , 2989.97778963,
       3372.61922661, 2877.70791646, 2255.35306714, 3293.54528281,
       3522.67187   , 2643.46404143, 1382.80600481, 3095.92589367,
       2459.71826376])
```

ARIMA (3,1,2)

04 Evaluation

ARIMA (2,1,2)

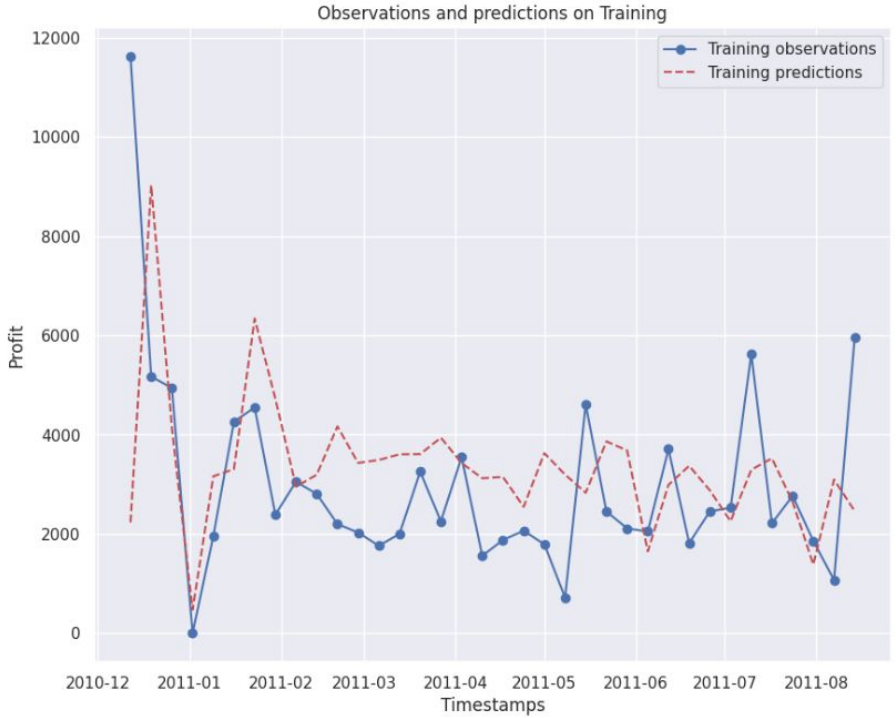

```python
timestamps = dcp_w_df.index[1:split]
train_obs = train[1:]
train_pred = fitted_model.predict()[1:]

fig, ax = plt.subplots()
ax.plot(
    timestamps, train_obs, '-o',
    label='Training observations',
    color='b'
)
ax.plot(
    timestamps, train_pred, '--',
    label='Training predictions',
    color='r'
)
ax.legend()
ax.set_xlabel('Timestamps')
ax.set_ylabel('Profit')
ax.set_title('Observations and predictions on Training')
plt.show()
```

```python
from sklearn.metrics import mean_squared_error

train_mse = mean_squared_error(train_obs, train_pred)
print('MSE: {:.2f}'.format(train_mse))
print('RMSE: {:.2f}'.format(train_mse**(1/2)))
```

```
MSE: 4964166.17
RMSE: 2228.04
```

04 Evaluation

ARIMA (3,1,2)




```python
timestamps = dcp_w_df.index[1:split]
train_obs = train[1:]
train_pred = fitted_model.predict()[1:]

fig, ax = plt.subplots()
ax.plot(
    timestamps, train_obs, '-o',
    label='Training observations',
    color='b'
)
ax.plot(
    timestamps, train_pred, '--',
    label='Training predictions',
    color='r'
)
ax.legend()
ax.set_xlabel('Timestamps')
ax.set_ylabel('Profit')
ax.set_title('Observations and predictions on Training')
plt.show()
```

```python
from sklearn.metrics import mean_squared_error

train_mse = mean_squared_error(train_obs, train_pred)
print('MSE: {:.2f}'.format(train_mse))
print('RMSE: {:.2f}'.format(train_mse**(1/2)))
```

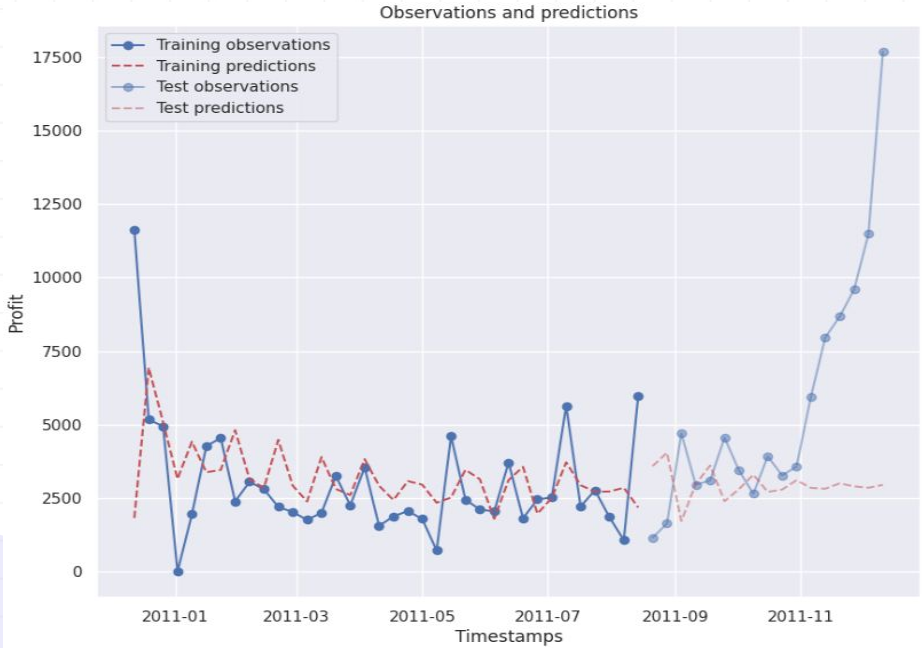
```
MSE: 4944910.42
RMSE: 2223.72
```

—----> Qua MSE và RMSE: lựa chọn mô hình ARIMA với p=3, i=1, q=2

05 Forecasting

★ Dự đoán doanh thu SP “Dotcom Postage tại UK cho 17 tuần tới (tương ứng với tập test)

- Forecasting without refit




```python
## Forecast n steps ahead without refit
train_timestamps = dcp_w_df.index[1:split]
train_obs = train[1:]
train_pred = fitted_model.predict()[1:]

test_timestamps = dcp_w_df.index[split:]
test_obs = test
test_pred = fitted_model.forecast(steps=len(test))

fig, ax = plt.subplots()
ax.plot(
    train_timestamps, train_obs, '-o',
    label='Training observations',
    color='b'
)
ax.plot(
    train_timestamps, train_pred, '--',
    label='Training predictions',
    color='r'
)
ax.plot(
    test_timestamps, test_obs, '-o',
    label='Test observations',
    color='b',
    alpha=0.5,
)
ax.plot(
    test_timestamps, test_pred, '--',
    label='Test predictions',
    color='r',
    alpha=0.5,
)
ax.legend()
ax.set_xlabel('Timestamps')
ax.set_ylabel('Profit')
ax.set_title('Observations and predictions')
plt.show()
```

Giá trị về sau có xu hướng không thay đổi

05 Forecasting

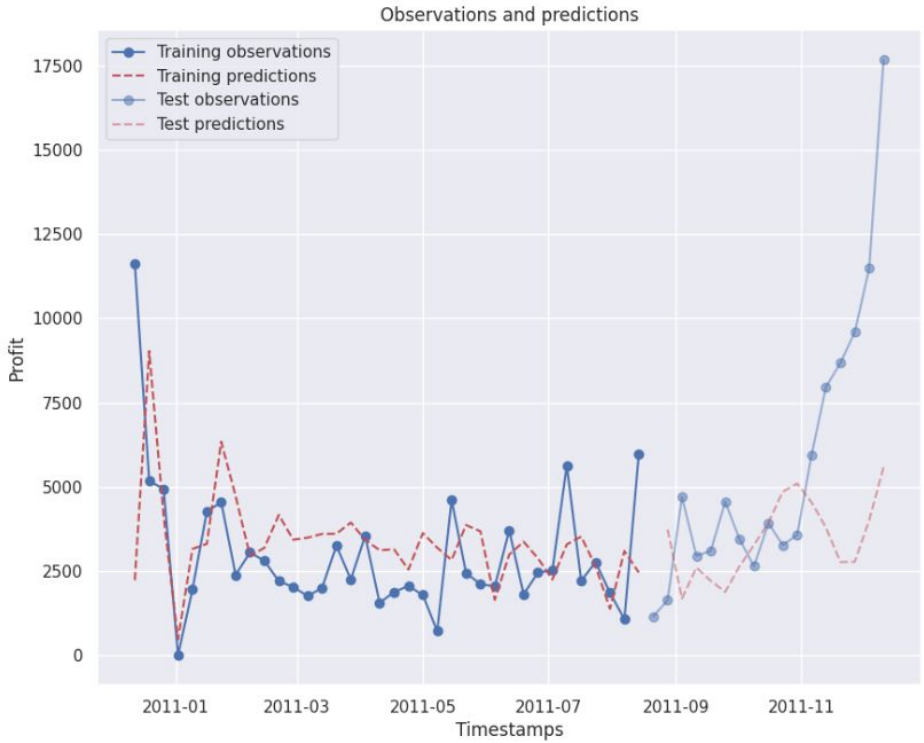
- Forecasting with update:

```python
## Forecast n step ahead with update
def arima_forecast_with_update(model_params,
                               train, test,
                               step=1, keep_history=True):
    if not isinstance(train, list):
        history = list(train)
    forecast = list()

    for t in range(len(test) - step + 1):
        model = ARIMA(
            history,
            order=model_params.get('order', (1, 0, 0)),
            seasonal_order=model_params.get('seasonal_order', (0, 0, 0, 0)),
        )  # define ARIMA model
        fitted_model = model.fit()  # fit ARIMA model
        f = fitted_model.forecast(steps=step)[-1]
        forecast.append(f)
        history.append(test[t])
        if not keep_history:
            keep_history.pop(0)
    return forecast
```

```python
# filter out warning
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter('ignore', ConvergenceWarning)
warnings.simplefilter('ignore', UserWarning)


step = 2
model_params = {'order': (3, 1, 2),
                'seasonal_order': (0, 0, 0, 0)}
forecast = arima_forecast_with_update(
    model_params, train, test, step
)
```

```python
step = 2
model_params = {'order': (3, 1, 2),
                'seasonal_order': (0, 0, 0, 0),}

model = ARIMA(
    train,
    order=model_params.get('order', (1, 0, 0)),
    seasonal_order=model_params.get('seasonal_order', (0, 0, 0, 0)),
)  # define ARIMA model
fitted_model = model.fit()  # fit ARIMA model

train_timestamps = dcp_w_df.index[1:split]
train_obs = train[1:]
train_pred = fitted_model.predict()[1:]

test_obs_timestamps = dcp_w_df.index[split:]
test_obs = test
test_pred_timestamps = dcp_w_df.index[split + step - 1:]
test_pred = forecast = arima_forecast_with_update(
    model_params, train, test, step
)
```

05 Forecasting




```python
fig, ax = plt.subplots()
ax.plot(
    train_timestamps, train_obs, '-o',
    label='Training observations',
    color='b'
)
ax.plot(
    train_timestamps, train_pred, '--',
    label='Training predictions',
    color='r'
)
ax.plot(
    test_obs_timestamps, test_obs, '-o',
    label='Test observations',
    color='b',
    alpha=0.5,
)
ax.plot(
    test_pred_timestamps, test_pred, '--',
    label='Test predictions',
    color='r',
    alpha=0.5,
)
ax.legend()
ax.set_xlabel('Timestamps')
ax.set_ylabel('Profit')
ax.set_title('Observations and predictions')
plt.show()
```

Giá trị về sau có xu hướng xu hướng đi lên ở cuối giai đoạn dự đoán

Thanks for listening!