

Міністерство освіти та науки України  
Львівський національний університет імені Івана Франка

# Звіт

Про виконання лабораторної роботи №1  
“Метод Arima”

Виконав: студент групи Фес-31

Мацевко Олександр

Перевірив: Сінкевич О.О.

Львів 2019

## Мета :

“Реалізувати на мові програмування Python метод ARIMA”

## Теорія:

Модель *ARIMA* певною мірою є розширеною версією моделі *ARMA*. Символ *I* ( *Integrated* ) відповідає за порядок оператора послідовної різниці. Справа в тому, що далеко не всі ряди є стаціонарними, але деякі з них можуть бути приведені до стаціонарних шляхом взяття послідовної різниці. Якщо, наприклад, часовий ряд  $y$ , став стаціонарним після взяття послідовної різниці порядку  $s$  і для опису вже стаціонарного ряду може бути використана модель *ARMA* ( $p, q$ ), то процесу, називається інтегрованим процесом авторегресії і ковзної середньої (*ARIMA* ( $p, s, q$ )).

При цьому для оцінки моделі можна користуватися кілька модифікованим підходом Боксу - Дженкінса. Безпосередньо перед першим етапом необхідно буде привести досліджуваний ряд до стаціонарного шляхом взяття послідовних різниць. Решта етапи підходу не зміняться, після цього стаціонарний ряд потрібно буде ідентифікувати, оцінити, діагностувати і використовувати

Модель *ARIMA* (Autoregressive Integrated Moving Average) - один з найбільш поширених методів аналізу і прогнозування часових рядів. Ця модель дозволяє обробити дані про послідовні, щоб краще зрозуміти цей ряд або передбачити його розвиток.

*ARIMA* використовує три основних параметри ( $p, d, q$ ), які виражаються цілими числами. Тому модель також записується як *ARIMA* ( $p, d, q$ ). Разом ці три параметри враховують сезонність, тенденцію і шум в наборах даних:

$p$  - порядок авторегресії (AR), який дозволяє додати попередні значення часового ряду. Цей параметр можна проілюструвати твердженням «завтра, ймовірно, буде тепло, якщо в останні три дні було тепло».  $d$  - порядок інтегрування (I; т. е. порядок різниць вихідного часового ряду). Він додає в модель поняття різниці часових рядів (визначає кількість минулих тимчасових точок, які потрібно вилучити з поточного значення). Цей параметр ілюструє таке твердження: «завтра, ймовірно, буде така ж температура, якщо різниця в температурі за останні три дні була дуже мала».  $q$  - порядок змінного середнього (MA), який дозволяє встановити похибка моделі як лінійну комбінацію спостерігалися раніше значень помилок. Для відстеження сезонності використовується сезонна модель *ARIMA* - *ARIMA* ( $p, d, q$ ) ( $P, D, Q$ )  $s$ . Тут ( $p, d, q$ ) - несезонні параметри, описані вище, а ( $P, D, Q$ ) слідує тим же визначенням, але застосовуються до сезонної складової часового ряду. Параметр  $s$  визначає періодичність тимчасового ряду (4 - квартальні періоди, 12 - річні періоди і т.д.).

Сезонна модель ARIMA може здатися складною через численні параметрів. У наступному розділі ви дізнаєтеся, як автоматизувати процес визначення оптимального набору параметрів для сезонної моделі часових рядів ARIMA.

## Дослідження програми:

### 1. Підключені бібліотеки

```
[ ] import numpy as np
import pandas as pd
from statsmodels.tsa.arima_model import ARIMA
%matplotlib inline
import matplotlib.pyplot as plt
from pmdarima.arima.utils import ndiffs
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
import warnings
```

### 2. Стаціонарність

```
[ ] # let's check the order of parameter d
def test_stationarity(timeseries, window = 12, cutoff = 0.01):

    #Determining rolling statistics
    rolmean = timeseries.rolling(window).mean()
    rolstd = timeseries.rolling(window).std()

    #Plot rolling statistics:
    fig = plt.figure(figsize=(12, 8))
    orig = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show()

    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries.iloc[:,0].values)
    pvalue = dftest[1]
    if pvalue < cutoff:
        print('p-value = %.4f. The series is likely stationary.' % pvalue)
    else:
        print('p-value = %.4f. The series is likely non-stationary.' % pvalue)
```

### 3.Виконання Arima

```
[ ] # evaluate an ARIMA model for a given order (p,d,q)
def check_arima_model(ts, arima_order):
    # prepare training dataset
    train_size = int(len(ts) * 0.6)
    train, test = ts[0:train_size], ts[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    #for t in range(len(test)):
    #    model = ARIMA(history, order=arima_order)
    #    model_fit = model.fit(dispatch=0)
    #    yhat = model_fit.forecast()[0]
    #    predictions.append(yhat)
    #    history.append(test[t])
    model = ARIMA(history, order=arima_order)
    model_fit = model.fit(dispatch=0)
    yhats = model_fit.forecast(len(test))[0]
    predictions = [x for x in yhats]
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    return error
```

### 4.Параметри

```
[ ] # evaluate combinations of p and q values for an ARIMA model
def get_arima_order(ts, p_vector, q_vector, d=0):
    best_score, best_cfg = float("inf"), None
    for p in p_vector:
        for q in q_vector:
            order = (p, d, q)
            try:
                mse = check_arima_model(ts, order)
                if mse < best_score:
                    best_score, best_cfg = mse, order
                    print('ARIMA%s MSE=%.3f' % (order,mse))
            except:
                continue
    print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))

[ ] # let's generate the ranges of values
p_vector = range(0, 3)
q_vector = range(0, 3)
warnings.filterwarnings("ignore")
get_arima_order(ts_train.iloc[:,0].values, p_vector, q_vector, d=0)

ARIMA(0, 0, 0) MSE=4.745
Best ARIMA(0, 0, 0) MSE=4.745
```

## 5. Навчання

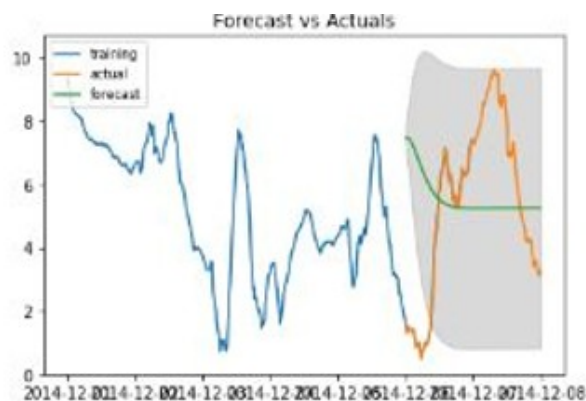
```
[ ] # Build Model
model = ARIMA(ts_train.iloc[:,0].values, order=(2, 0, 1))
fitted = model.fit(disp=-1)
print(fitted.summary())

# Forecast
fc, se, conf = fitted.forecast(len(ts_test), alpha=0.05) # 95% conf
# Make as pandas series
fc_series = pd.Series(fc, index=ts_test.index)
lower_series = pd.Series(conf[:, 0], index=ts_test.index)
upper_series = pd.Series(conf[:, 1], index=ts_test.index)

# Plot
plt.figure()
plt.plot(ts_train, label='training')
plt.plot(ts_test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series, color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

ARMA Model Results						
Dep. Variable:	y	No. Observations:	576			
Model:	ARMA(2, 1)	Log Likelihood	337.402			
Method:	css-mle	S.D. of innovations	0.134			
Date:	Sun, 13 Oct 2019	AIC	-664.804			
Time:	09:42:15	BIC	-643.024			
Sample:	0	HQIC	-656.310			
	coef	std err	z	P> z	[0.025	0.975]
const	5.2391	0.654	8.010	0.000	3.957	6.521
ar.L1.y	1.8840	0.027	69.702	0.000	1.831	1.937
ar.L2.y	-0.8883	0.027	-32.874	0.000	-0.941	-0.835
ma.L1.y	-0.4780	0.052	-9.151	0.000	-0.580	-0.376

## 6. Результат



**Висновок :** На цій лабораторній роботі я вивчив початки роботи з Python та навчився працювати з модель Arima, для прогнозування часових рядів, також навчився працювати з базами даних температур та використовувати Jupyter notebook.