

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**  
**KHOA CÔNG NGHỆ THÔNG TIN**

---



**ĐỒ ÁN MÔN HỌC**

**NGHIÊN CỨU ĐIỂN HÌNH VỀ DỰ BÁO CHUỖI THỜI GIAN  
VỚI PYTHON: CƯỚP CÓ VŨ TRANG HÀNG THÁNG Ở  
BOSTON**

**Giảng viên giảng dạy: TS Hà Minh Tân**

**Sinh viên thực hiện : Chu Doãn Đức**

**MSSV : 2000003917**

**Môn học : Phân tích chuỗi thời gian và ứng dụng**

**Khóa : 2020**

**Tp.HCM, tháng 5 năm 2023**

## LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Trường Đại học Nguyễn Tất Thành đã đưa môn học Phân tích chuỗi thời gian và ứng dụng vào trường trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – Thầy Hà Minh Tân đã dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc và đã cho em chắc chắn được hoạch định tương lai của mình.

Bộ môn Phân tích chuỗi thời gian và ứng dụng là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên nói chung và riêng bản thân em nói riêng. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ và hạn hẹp. Mặc dù em đã cố gắng hết sức nhưng chắc chắn bài báo của em khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong các thầy/cô chấm bài xem xét và góp ý để bài tiểu luận của em được hoàn thiện hơn.

Kính chúc thầy có nhiều sức khỏe, hạnh phúc, thành công trên con đường giảng dạy  
Em xin chân thành cảm ơn!

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH

KỲ THI KẾT THÚC HỌC PHẦN

TRUNG TÂM KHẢO THÍ

HỌC KỲ ..... NĂM HỌC ..... - .....

**PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN**

Môn thi: Phân tích dữ liệu chuỗi thời gian và ứng dụng. Lớp học phần: 20DTH2A .....

Nhóm sinh viên thực hiện: .....

1. Chu Doãn Đức ..... Tham gia đóng góp: 100% .....

2. Phạm Phước Long ..... Tham gia đóng góp: 100% .....

3. Huỳnh Thị Huyền Trân..... Tham gia đóng góp: 100% .....

Ngày thi: Ngày 18 tháng 5 năm 2023 ..... Phòng thi: L.501 .....

Đề tài tiểu luận/báo cáo của sinh viên: Nghiên cứu điển hình về dự báo chuỗi thời gian với python: Cướp có vũ trang hàng tháng ở Boston.

Phản đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Tiêu chí (theo CDR HP)	Đánh giá của GV	Điểm tối đa	Điểm đạt được
Cấu trúc của báo cáo	..... .....		
Nội dung			
- Các nội dung thành phần	..... .....		
- Lập luận	..... .....		
- Kết luận	.....		
Trình bày	.....		
<b>TỔNG ĐIỂM</b>			

**Giảng viên chấm thi***(ký, ghi rõ họ tên)*

## LỜI MỞ ĐẦU

Ngày nay, ứng dụng công nghệ thông tin và việc tin học hóa được xem là một trong những yếu tố mang tính quyết định trong hoạt động của các chính phủ, tổ chức, cũng như các công ty, cửa hàng. Nó đóng vai trò hết sức quan trọng, có thể tạo ra những bước đột phá mạnh mẽ.

Đi cùng với đó dự đoán chuỗi thời gian cũng là một lĩnh vực quan trọng. Nó sử dụng để dự báo các giá trị trong tương lai dựa trên các dữ liệu quá khứ.

Việc ứng dụng dự báo chuỗi thời gian giúp các quyết định kinh doanh quan trọng có thể thúc đẩy sự phát triển. So sánh xu hướng hiện tại với xu hướng quá khứ đã xảy ra để có thể ước tính xu hướng tương lai và đưa ra các chiến lược phù hợp. Phát hiện và giải thích các yếu tố ảnh hưởng đến biến số theo thời gian, như mùa vụ, chu kỳ kinh tế, hoặc các sự kiện bất ngờ. Kiểm tra và đánh giá hiệu quả của các chính sách hoặc can thiệp có tác động đến biến số theo thời gian, như chính sách tiền tệ, thuế, hoặc giáo dục.

Chính vì lẽ đó, trong báo cáo này chúng ta sẽ áp dụng các kỹ thuật phân tích time series để nghiên cứu và dự báo số vụ cướp vũ trang hàng tháng tại Boston, Mỹ. Bộ dữ liệu cung cấp số vụ cướp vũ trang hàng tháng từ tháng 1 năm 1966 đến tháng 10 năm 1975, tức là gần 10 năm dữ liệu. Chúng sẽ sử dụng mô hình ARIMA (0, 1, 2) để khớp với bộ dữ liệu và đánh giá hiệu suất của mô hình bằng sai số bình phương trung bình căn (RMSE). Và đi cùng với đó cũng sẽ kiểm tra tính ổn định, tính dừng và tính mùa vụ của chuỗi thời gian và áp dụng biến đổi Box-Cox để làm cho phân phối của dữ liệu gần với phân phối chuẩn hơn. Mục tiêu của báo cáo này là để minh họa các bước và công cụ cho một dự án dự báo chuỗi thời gian bằng Python.

Hoàn thành xong đề tài, em vô cùng biết ơn thầy Hà Minh Tân đã hướng dẫn nhiệt tình cho chúng em trong suốt quá trình thực hiện đề tài này.

[illegible]

**Giảng viên giảng dạy**  
(Ký tên và ghi rõ họ tên)

# MỤC LỤC

LỜI CẢM ƠN.....	
LỜI MỞ ĐẦU.....	
NHẬN XÉT CỦA GIẢNG VIÊN GIẢNG DẠY .....	
MỤC LỤC .....	
CHƯƠNG 1: GIỚI THIỆU .....	1
1. Giới thiệu đề tài:.....	1
2. Lý do chọn đề tài:.....	1
3. Mục tiêu của đề tài: .....	1
4. Công nghệ áp dụng:.....	2
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....	3
2.1 – Tổng quan về dự báo chuỗi thời gian:.....	3
2.1.1 – Giới thiệu: .....	3
2.1.2 - Ứng dụng: .....	3
2.1.3 – Mô hình ARIMA (Auto-Regression Integrated Moving Average): .....	4
CHƯƠNG 3: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM .....	6
3.1 – Xác thực và phân chia bộ dữ liệu:.....	6
3.2 – Đánh giá mô hình: .....	7
3.2.1 – Phương pháp đo hiệu năng (Performance Measure). .....	7
3.2.2 – Chiến lược kiểm tra (Test Strategy). .....	7
3.3 – Mô hình Persistence: .....	8
3.4 – Phân tích dữ liệu:.....	10
3.4.1 – Summary Statistics (Thống kê tóm tắt): .....	10
3.4.2 – Line Plot: .....	11
3.4.3 – Density Plots: .....	12
3.4.4 – Box and Whisker Plot: .....	13
3.5 – Mô hình ARIMA:.....	14
3.5.1 – Mô hình ARIMA thủ công:.....	15
3.5.2 - Lưới tìm kiếm Siêu tham số ARIMA:.....	18
3.5.3 – Xem lại các lỗi dư:.....	20
3.5.4 – Chuyển đổi tập dữ liệu Box-Cox:.....	22
3.6 – Kiểm định mô hình:.....	24
3.6.1 – Hoàn thiện mô hình: .....	24
3.6.2 – Dự đoán:.....	25
3.6.3 – Kiểm định mô hình: .....	26
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	28
1. Kết luận:.....	28
2. Hướng phát triển: .....	28
Tài liệu tham khảo .....	29

# CHƯƠNG 1: GIỚI THIỆU

## 1. Giới thiệu đề tài:

Trong những năm gần đây, dự báo chuỗi thời gian (Time Series Forecasting) là một trong những kỹ thuật cực kỳ quan trọng tổng nhiều lĩnh vực như kinh tế, tài chính, y tế, thời tiết và nhiều hơn nữa. Nó giúp chúng ta dự đoán các sự kiện trong tương lai dựa trên các xu hướng và mẫu trong quá khứ. Trong đề tài lần này chúng ta sẽ tìm hiểu cách sử dụng Python trong dự báo chuỗi thời gian để “Nghiên cứu điển hình về dự báo chuỗi thời gian với Python: Cướp có vũ trang hàng tháng ở Boston” từ đó đưa ra dự báo số lượng cướp có vũ trang hàng tháng ở Boston - Mỹ, từ tháng 1 năm 1966 đến tháng 10 năm 1975. Đề tài này sử dụng các thư viện pandas, numpy và matplotlib để xử lý và trực quan hóa dữ liệu chuỗi thời gian. Nó cũng giải thích các khái niệm cơ bản về chuỗi thời gian, như tính dừng yên, tự tương quan và xu hướng. Áp dụng mô hình ARIMA (Tự hồi quy - Trung bình động - Tích phân) để xây dựng mô hình dự báo và đánh giá hiệu suất của nó.

## 2. Lý do chọn đề tài:

Em đã chọn đề tài “Nghiên cứu điển hình về dự báo chuỗi thời gian với Python: Cướp có vũ trang hàng tháng ở Boston” vì em muốn học cách sử dụng Python để giải quyết một bài toán dự báo chuỗi thời gian thực tế. Em muốn khám phá các kỹ thuật phân tích và mô hình hóa dữ liệu chuỗi thời gian, cũng như các mô hình học máy truyền thống và học sâu. Em hy vọng rằng bằng cách làm đề tài này, tôi sẽ nâng cao kỹ năng lập trình Python của mình và hiểu sâu hơn về dự báo chuỗi thời gian. Em cũng mong muốn áp dụng những kiến thức và kinh nghiệm từ đề tài này vào các bài toán khác trong tương lai. Em tin rằng dự báo chuỗi thời gian là một kỹ thuật có giá trị cao và có nhiều ứng dụng trong nhiều lĩnh vực khác nhau.

## 3. Mục tiêu của đề tài:

Học cách sử dụng Python để dự báo số lượng cướp có vũ trang hàng tháng ở Boston - Mỹ, trong một thập kỷ. Đây là một bài toán dự báo chuỗi thời gian thực tế, có tính ứng

dụng cao trong lĩnh vực an ninh công cộng. Bằng cách làm đề tài này, chúng ta sẽ nâng cao kỹ năng lập trình Python của bản thân và hiểu sâu hơn về các kỹ thuật phân tích và mô hình hóa dữ liệu chuỗi thời gian. Chúng ta cũng sẽ có cơ hội so sánh hiệu suất của mô hình ARIMA với các mô hình học sâu khác nhau và khám phá các ứng dụng khác của dự báo chuỗi thời gian.

#### **4. Công nghệ áp dụng:**

- Pandas: một thư viện phổ biến cho thao tác và phân tích dữ liệu, bao gồm các hàm cho phân tích chuỗi thời gian.
- NumPy: một thư viện cho tính toán khoa học, bao gồm các hàm cho xử lý mảng và các phép toán toán học.
- Matplotlib: một thư viện cho trực quan hóa dữ liệu, bao gồm các hàm để vẽ biểu đồ đường, biểu đồ cột, biểu đồ hộp và râu và nhiều hơn nữa.
- ARIMA: một mô hình học máy truyền thống cho dự báo chuỗi thời gian, viết tắt của Autoregressive Integrated Moving Average.
- Darts: một thư viện Python mới được phát triển bởi Unit8 cho việc thao tác và dự báo chuỗi thời gian dễ dàng. Ý tưởng của Darts là làm cho nó đơn giản để sử dụng như sklearn cho chuỗi thời gian. Darts cố gắng làm mịn quá trình sử dụng chuỗi thời gian trong học máy. Nó chứa nhiều mô hình, từ những kinh điển như ARIMA đến các mạng nơ-ron sâu.
- scikit-learn: một thư viện cho học máy trong Python, bao gồm các hàm cho huấn luyện và đánh giá các mô hình học máy, cũng như các hàm cho tiền xử lý dữ liệu, chọn lựa mô hình và tối ưu hóa tham số.
- statsmodels: một thư viện cho phân tích dữ liệu thống kê trong Python, bao gồm các hàm cho kiểm định giả thuyết, mô hình hồi quy, phân tích chuỗi thời gian và nhiều hơn nữa.



## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1 – Tổng quan về dự báo chuỗi thời gian:

#### 2.1.1 – Giới thiệu:

Dự báo chuỗi thời gian (Time Series Forecasting) là một kỹ thuật dự đoán các sự kiện thông qua một chuỗi thời gian. Nó giả định rằng các xu hướng trong quá khứ sẽ lặp lại trong tương lai và phân tích chúng để dự đoán. Đây cũng là một lĩnh vực học máy quan trọng và có thể xem như một bài toán học có giám sát. Các phương pháp học máy như Regression, Neural Networks, Support Vector Machines, Random Forests và XGBoost có thể được sử dụng cho bài toán này. Dự báo chuỗi thời gian kết hợp với phân tích để xây dựng các mô hình khám phá tri thức từ dữ liệu. Sau đó, các mô hình này được dùng để làm cơ sở cho các quan sát và quyết định chiến lược trong tương lai.

#### 2.1.2 - Ứng dụng:

Dự báo chuỗi thời gian là một trong những kỹ thuật khoa học dữ liệu phổ biến trong lập kế hoạch kinh doanh, sản xuất và hàng tồn kho, tài chính và quản lý chuỗi cung ứng. Một số trường hợp sử dụng dự báo chuỗi thời gian:

- Dự đoán giá và nhu cầu:

Việc dự đoán nhu cầu hoặc kỳ vọng của khách hàng là một thách thức lớn cho các doanh nghiệp liên quan đến nguồn cung ứng và mua sắm. Một ứng dụng khác là dự đoán giá hoặc tỷ giá của sản phẩm/dịch vụ để tự động điều chỉnh giá hoặc tỷ lệ theo mục tiêu doanh thu hoặc nhu cầu. Các khái niệm dự đoán giá trong dự báo chuỗi thời gian sẽ tạo ra nhiều cơ hội để cải thiện và cá nhân hóa trải nghiệm của khách hàng.

- Phát hiện gian lận:

Phát hiện bất thường trong học máy bao gồm việc quan sát các giá trị ngoại lệ trong phân phối điểm dữ liệu. Nói ngắn gọn, nó nhận ra các đợt tăng đột ngột bất thường, chênh lệch so với cả xu hướng và mùa vụ. Phát hiện gian lận là một vấn đề lớn cho bất kỳ lĩnh vực nào liên quan đến hoạt động tài chính và thanh toán. Phân tích chuỗi thời gian kết hợp với học máy có thể phát hiện các hoạt

động khả nghi như thay đổi địa chỉ giao hàng hoặc rút số tiền lớn để chỉ ra các giao dịch gian lận.

### 2.1.3 – Mô hình ARIMA (Auto-Regression Integrated Moving Average):

Mô hình ARIMA là mô hình dự báo cho chuỗi thời gian, mô hình này chỉ dựa trên bản thân chuỗi dữ liệu theo thời gian của chỉ tiêu muốn dự báo. Một điều đáng lưu ý là để có thể áp dụng mô hình ARIMA thì dãy số liệu nghiên cứu của chúng ta đòi hỏi phải có tính dừng. Tuy nhiên, trong thực tế đa số các dãy số liệu kinh tế thường không có tính dừng, chính vì vậy chúng ta phải chuyển chúng sang dừng trước khi áp dụng các mô hình này.

Về cơ bản tạo ra một phương trình tuyến tính trong đó mô tả và dự báo dữ liệu chuỗi thời gian của bạn. Phương trình này được tạo ra thông qua ba phần riêng biệt có thể được mô tả như sau:

- AR - tự động hồi quy: các thuật ngữ phương trình được tạo dựa trên các điểm dữ liệu trong quá khứ.
- I - tích hợp hoặc khác biệt: tính đến "xu hướng" tổng thể trong dữ liệu.
- MA - đường trung bình động: thuật ngữ phương trình về lỗi hoặc nhiễu dựa trên các điểm dữ liệu trong quá khứ.

Mô hình ARIMA thường được viết dưới dạng ARIMA (p, d, q) với mỗi chữ cái tương ứng với một trong ba phần đã nêu ở trên. Ba chữ cái này biểu thị cho các tham số mà bạn cần cung cấp và có ý nghĩa như sau:

- p: cho biết số lượng các thuật ngữ tự hồi quy (AR).
- d: cho biết bậc của sự khác biệt.
- q: cho biết số lượng các điều kiện của đường trung bình động (MA).

Áp dụng mô hình ARIMA với phương pháp Box-Jenkins (Phương pháp xác định, ước lượng, kiểm định và dự báo mô hình ARIMA) và chuyển đổi Box-Cox (Một phương pháp biến đổi dữ liệu để làm cho nó tuân theo giả định về phân phối chuẩn) vào Monthly Armed Robberies in Boston:

- Cải thiện độ chính xác của dự báo bằng cách làm cho phân phối của dữ liệu gần với phân phối chuẩn, giảm thiểu ảnh hưởng của các giá trị ngoại lai và giảm thiểu sai số trong quá trình ước lượng.

- Giúp xác định các giá trị  $p$ ,  $d$ ,  $q$  cho mô hình ARIMA một cách dễ dàng hơn bằng cách sử dụng các lược đồ ACF và PACF<sup>2</sup>.
- Giúp kiểm tra tính phù hợp của mô hình bằng cách kiểm tra xem phần dư có phải là nhiễu trắng hay không.
- Giúp phát hiện các xu hướng, chu kỳ và tính dừng của chuỗi thời gian.
- Giúp phân tích sự tác động của các yếu tố kinh tế, chính sách và xã hội đến số lượng vụ cướp có vũ trang.

## CHƯƠNG 3: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM

Tải các thư viện cần thiết cho mô hình.

```
[ ] import warnings
    from sklearn.metrics import mean_squared_error
    from math import sqrt
    from math import log
    from math import exp

    import numpy as np
    import pandas as pd

    from matplotlib import pyplot

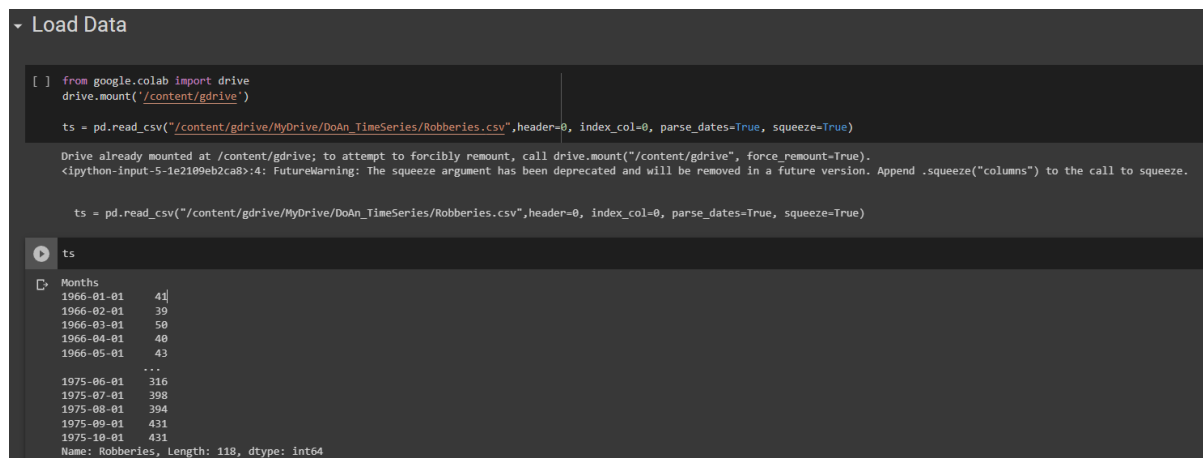
    from pandas import DataFrame
    from pandas import Grouper

    from statsmodels.tsa.stattools import adfuller
    from statsmodels.tsa.arima_model import ARIMA
    from statsmodels.graphics.tsaplots import plot_acf
    from statsmodels.graphics.tsaplots import plot_pacf

    from scipy.stats import boxcox
    from statsmodels.graphics.gofplots import qqplot

    from statsmodels.tsa.arima_model import ARIMAResults
```

Tải dữ liệu đã lấy về từ kaggle.



The screenshot shows a Jupyter Notebook interface. The top section is titled 'Load Data' and contains a code cell with the following Python code:

```
[ ] from google.colab import drive
    drive.mount('/content/gdrive')

    ts = pd.read_csv("/content/gdrive/MyDrive/DoAn_TimeSeries/Robberies.csv", header=0, index_col=0, parse_dates=True, squeeze=True)
```

Below the code cell, there is a warning message: "Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount(\"/content/gdrive\", force\_remount=True)." and a FutureWarning: "The squeeze argument has been deprecated and will be removed in a future version. Append .squeeze(\"columns\") to the call to squeeze."

The bottom section shows the variable 'ts' as a pandas Series. The first few rows of the series are displayed:

Months	ts
1966-01-01	41
1966-02-01	39
1966-03-01	58
1966-04-01	48
1966-05-01	43
...	...
1975-06-01	316
1975-07-01	398
1975-08-01	394
1975-09-01	431
1975-10-01	431

At the bottom, it says: "Name: Robberies, Length: 118, dtype: int64".

### 3.1 – Xác thực và phân chia bộ dữ liệu:

Dữ liệu không được cập nhật thường xuyên. Điều này khiến cho việc thu thập dữ liệu mới nhất để kiểm tra mô hình trở nên khó khăn. Vì vậy, chúng ta sẽ giả định rằng hiện tại là tháng 10 năm 1974 và loại bỏ một năm dữ liệu cuối cùng khỏi quá trình phân tích và lựa chọn mô hình. Năm dữ liệu cuối cùng này sẽ được dùng để kiểm tra mô hình hoàn thiện. Đoạn mã sau đây sẽ đọc bộ dữ liệu làm một Series Pandas và chia làm hai

phần, một phần để xây dựng mô hình (dataset.csv) và phần còn lại để kiểm tra (validation.csv).

### Split Train/Validation

```
[ ] split_point = len(ts) - 12
    dataset, validation = ts[0:split_point], ts[split_point:]
    print('Dataset %d, Validation %d' % (len(dataset), len(validation)))
    dataset.to_csv('dataset.csv', header=False)
    validation.to_csv('validation.csv', header=False)
```

Dataset 106, Validation 12

## 3.2 – Đánh giá mô hình:

Đánh giá mô hình chỉ được thực hiện trên dữ liệu trong dataset.csv được chuẩn bị trong phần trước. Đánh giá mô hình bao gồm hai yếu tố:

- Phương pháp đo hiệu năng (Performance Measure).
- Chiến lược kiểm tra (Test Strategy).

### 3.2.1 – Phương pháp đo hiệu năng (Performance Measure).

Phương pháp đo hiệu năng Các quan sát là số lượng vụ cướp. Chúng ta sẽ đánh giá hiệu năng của các dự đoán bằng cách sử dụng sai số bình phương trung bình căn bậc hai (RMSE). Điều này sẽ cho trọng số nhiều hơn cho những dự đoán sai lệch nhiều và sẽ có cùng đơn vị với dữ liệu gốc. Bất kỳ biến đổi nào đối với dữ liệu cũng phải được đảo ngược trước khi tính toán và báo cáo RMSE để làm cho hiệu năng giữa các phương pháp khác nhau có thể so sánh trực tiếp.

Chúng ta có thể tính RMSE bằng cách sử dụng hàm trợ giúp từ thư viện scikit-learn `mean_squared_error()` để tính sai số bình phương trung bình giữa một danh sách các giá trị mong đợi và các dự đoán. Sau đó chúng ta có thể lấy căn bậc hai của giá trị này để cho chúng ta một điểm số RMSE.

### 3.2.2 – Chiến lược kiểm tra (Test Strategy).

Các mô hình ứng viên sẽ được đánh giá bằng cách sử dụng kiểm tra tiến lên. Điều này là do một mô hình dự báo di chuyển được yêu cầu từ định nghĩa vấn đề. Đây là nơi cần

có các dự báo một bước cho biết tất cả dữ liệu có sẵn. Kiểm tra tiến lên sẽ hoạt động như sau:

1. 50% đầu tiên của bộ dữ liệu sẽ được giữ lại để huấn luyện mô hình.
2. 50% còn lại của bộ dữ liệu sẽ được lặp lại và kiểm tra mô hình
3. Đối với mỗi bước trong bộ dữ liệu kiểm tra:
  - a. Một mô hình sẽ được huấn luyện.
  - b. Một dự báo một bước (one-step) được thực hiện và dự báo được lưu trữ để đánh giá sau này.
  - c. Quan sát thực tế từ bộ dữ liệu kiểm tra sẽ được thêm vào bộ dữ liệu huấn luyện cho lần lặp tiếp theo.
4. Các dự báo được thực hiện trong quá trình lặp lại của bộ dữ liệu kiểm tra sẽ được đánh giá và một điểm số RMSE được báo cáo.

Do kích thước nhỏ của dữ liệu, chúng ta sẽ cho phép mô hình được huấn luyện lại với tất cả dữ liệu có sẵn trước mỗi dự báo. Chúng ta có thể viết mã cho khung kiểm tra bằng cách sử dụng mã NumP và Python đơn giản. Đầu tiên, chúng ta có thể chia bộ dữ liệu thành các tập huấn luyện và kiểm tra trực tiếp. Chúng ta cần thận luôn chuyển đổi bộ dữ liệu đã tải thành float32 trong trường hợp dữ liệu đã tải vẫn có một số loại dữ liệu Chuỗi hoặc Số nguyên.

#### ▾ Chuẩn bị dữ liệu

```
[ ] X = ts.values
    X = X.astype('float32')
    train_size = int(len(X) * 0.50)
    train, test = X[0:train_size], X[train_size:]
```

### 3.3 – Mô hình Persistence:

Bước đầu tiên trước khi bị mắc kẹt trong phân tích dữ liệu và mô hình hóa là thiết lập một hiệu năng cơ sở. Điều này sẽ cung cấp cho chúng ta một mẫu để đánh giá các mô hình sử dụng bộ kiểm tra đề xuất và một phương pháp đo hiệu năng để so sánh tất cả các mô hình dự báo phức tạp hơn. Dự báo cơ sở cho dự báo chuỗi thời gian được gọi

là dự báo ngẫu thơ, hoặc tính bền vững. Đây là nơi quan sát từ bước thời gian trước được sử dụng làm dự báo cho quan sát ở bước thời gian tiếp theo. Chúng ta có thể áp dụng nó trực tiếp vào bộ kiểm tra được xác định trong phần trước. Danh sách mã hoàn chỉnh được cung cấp bên dưới.

```
# Kiểm tra tiên bộ
history = [x for x in train]
predict = list()

for i in range(len(test)):
    # Dự đoán
    yhat = history[-1]
    predict.append(yhat)
    # quan sát
    obs = test[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%.3f' % (yhat, obs))

# báo cáo hiệu suất
rmse = sqrt(mean_squared_error(test, predict))
print('RMSE: %.3f' % rmse)
```

```
>Predicted=174.000, Expected=178.000
>Predicted=178.000, Expected=136.000
>Predicted=136.000, Expected=161.000
>Predicted=161.000, Expected=171.000
>Predicted=171.000, Expected=149.000
>Predicted=149.000, Expected=184.000
>Predicted=184.000, Expected=155.000
>Predicted=155.000, Expected=276.000
>Predicted=276.000, Expected=224.000
>Predicted=224.000, Expected=213.000
>Predicted=213.000, Expected=279.000
>Predicted=279.000, Expected=268.000
>Predicted=268.000, Expected=287.000
>Predicted=287.000, Expected=238.000
>Predicted=238.000, Expected=213.000
>Predicted=213.000, Expected=257.000
>Predicted=257.000, Expected=293.000
>Predicted=293.000, Expected=212.000
>Predicted=212.000, Expected=246.000
>Predicted=246.000, Expected=353.000
>Predicted=353.000, Expected=339.000
>Predicted=339.000, Expected=308.000
>Predicted=308.000, Expected=247.000
>Predicted=247.000, Expected=257.000
>Predicted=257.000, Expected=322.000
>Predicted=322.000, Expected=298.000
>Predicted=298.000, Expected=273.000
>Predicted=273.000, Expected=312.000
>Predicted=312.000, Expected=249.000
>Predicted=249.000, Expected=286.000
```

```
>Predicted=286.000, Expected=279.000
>Predicted=279.000, Expected=309.000
>Predicted=309.000, Expected=401.000
>Predicted=401.000, Expected=309.000
>Predicted=309.000, Expected=328.000
>Predicted=328.000, Expected=353.000
>Predicted=353.000, Expected=354.000
>Predicted=354.000, Expected=327.000
>Predicted=327.000, Expected=324.000
>Predicted=324.000, Expected=285.000
>Predicted=285.000, Expected=243.000
>Predicted=243.000, Expected=241.000
>Predicted=241.000, Expected=287.000
>Predicted=287.000, Expected=355.000
>Predicted=355.000, Expected=460.000
>Predicted=460.000, Expected=364.000
>Predicted=364.000, Expected=487.000
>Predicted=487.000, Expected=452.000
>Predicted=452.000, Expected=391.000
>Predicted=391.000, Expected=500.000
>Predicted=500.000, Expected=451.000
>Predicted=451.000, Expected=375.000
>Predicted=375.000, Expected=372.000
>Predicted=372.000, Expected=302.000
>Predicted=302.000, Expected=316.000
>Predicted=316.000, Expected=398.000
>Predicted=398.000, Expected=394.000
>Predicted=394.000, Expected=431.000
>Predicted=431.000, Expected=431.000
RMSE: 54.191
```

Việc chạy kiểm tra sẽ in ra dự báo và quan sát cho mỗi lần lặp lại của bộ dữ liệu kiểm tra. RMSE của mô hình Persistence ở đây là 54.191. Điều này có nghĩa là trung bình, mô hình sai khoảng 54 vụ cướp cho mỗi dự báo được thực hiện.


### 3.4 – Phân tích dữ liệu:

Chúng ta có thể sử dụng số liệu thống kê tóm tắt và sơ đồ dữ liệu để nhanh chóng tìm hiểu thêm về cấu trúc của vấn đề dự đoán. Trong phần này, chúng ta sẽ xem xét dữ liệu từ bốn khía cạnh:

1. Summary Statistics (Thống kê tóm tắt).
2. Line Plot.
3. Density Plots.
4. Box and Whisker Plot.

#### 3.4.1 – Summary Statistics (Thống kê tóm tắt):

Thống kê tóm tắt cung cấp một cái nhìn nhanh về các giới hạn của các giá trị được quan sát. Có thể giúp hiểu nhanh về những gì chúng ta đang làm việc. Đây là các số liệu thể hiện những đặc điểm chung của một tập dữ liệu, như giá trị trung bình, độ lệch chuẩn, phân vị, phạm vi, và tần suất.

Summary Statistics(Thống kê tóm tắt)	
	<code>print(ts.describe())</code>
count	118.000000
mean	196.288136
std	128.043602
min	29.000000
25%	85.500000
50%	166.000000
75%	296.750000
max	500.000000
Name: Robberies, dtype: float64	

Nhìn bảng thống kê trên chúng ta có thể dễ dàng thấy được một số quan sát:

- Số lượng quan sát (count) phù hợp với kỳ vọng của chúng ta, nghĩa là chúng ta đang xử lý dữ liệu chính xác.



- Giá trị trung bình là khoảng 196, mà chúng ta có thể coi là cấp độ của chúng tôi trong loạt bài này.
- Độ lệch chuẩn tương đối lớn là 128 vụ cướp.
- Các phần trăm cùng với độ lệch chuẩn cho thấy mức độ lây lan lớn đối với dữ liệu.
- Mức chênh lệch lớn trong chuỗi này sẽ khó đưa ra dự đoán có độ chính xác cao nếu nó được gây ra bởi biến động ngẫu nhiên.

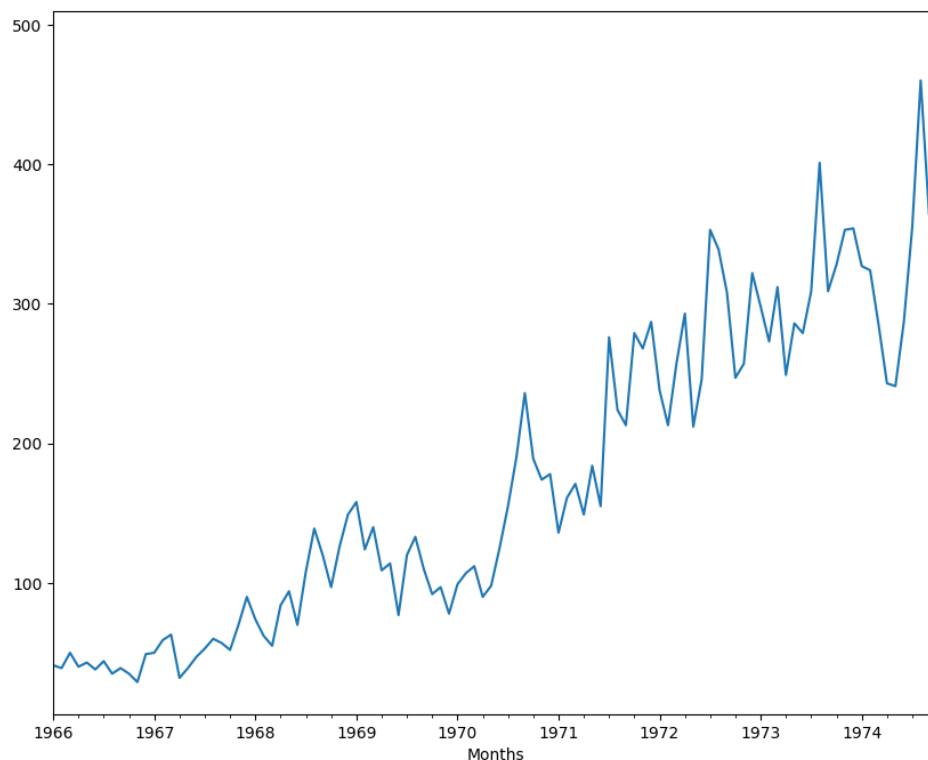
### 3.4.2 – Line Plot:

Line Plot là Biểu đồ Đường. Đây là một loại đồ thị biểu diễn sự thay đổi của một biến số theo một biến số khác, thường là thời gian. Đồ thị đường được vẽ bằng cách nối các điểm dữ liệu bằng các đoạn thẳng. Đồ thị đường có thể dùng để phân tích xu hướng, chu kỳ, hoặc mối quan hệ giữa hai biến số.

### Line Plot

```

series = dataset
pyplot.figure(figsize=(10,8))
series.plot()
pyplot.show()
```

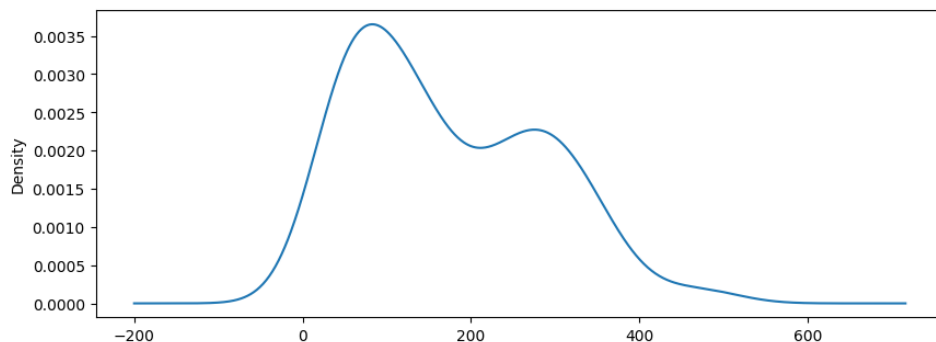
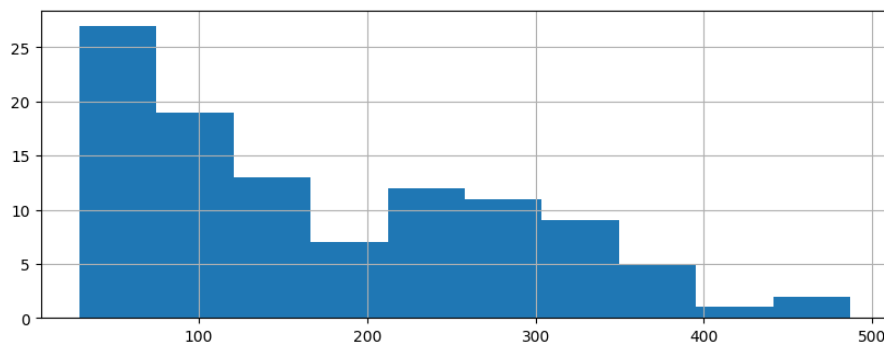
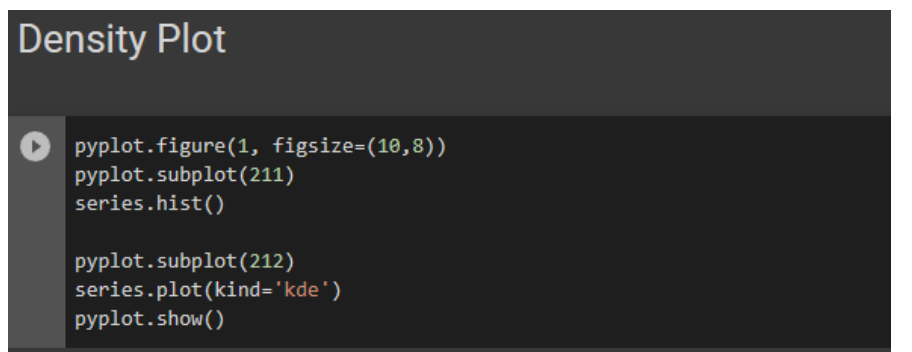


Nhìn vào biểu đồ Line Plot trên chúng ta có thể quan sát thấy:

- Biểu đồ có xu hướng gia tăng của các vụ cướp theo Months (Tháng).
- Có vẻ như không có bất kỳ ngoại lệ rõ ràng.
- Có sự biến động tương đối lớn từ năm này sang năm khác (lên và xuống).
- Biến động vào cuối năm có vẻ lớn hơn biến động vào những năm trước đó.
- Xu hướng này cho thấy là tập dữ liệu gần như chắc chắn không cố định và sự thay đổi dao động rõ ràng cũng có thể ảnh hưởng.

### 3.4.3 – Density Plots:

Density Plots (Biểu đồ mật độ) một loại đồ thị biểu diễn phân bố của một biến số liên tục, bằng cách ước lượng hàm mật độ xác suất của biến số đó. Đồ thị mật độ có thể dùng để so sánh phân bố của nhiều nhóm dữ liệu, hoặc để khám phá các đặc điểm của phân bố như đối xứng, lệch, hoặc đa đỉnh.



Từ biểu đồ Density Plots (Biểu đồ mật độ) chúng ta có thể đưa ra nhận xét như sau:

- Phân bố không tuân theo phân phối chuẩn.
- Phân bố bị lệch sang trái và có thể là phân phối mũ hoặc phân phối chuẩn kép.

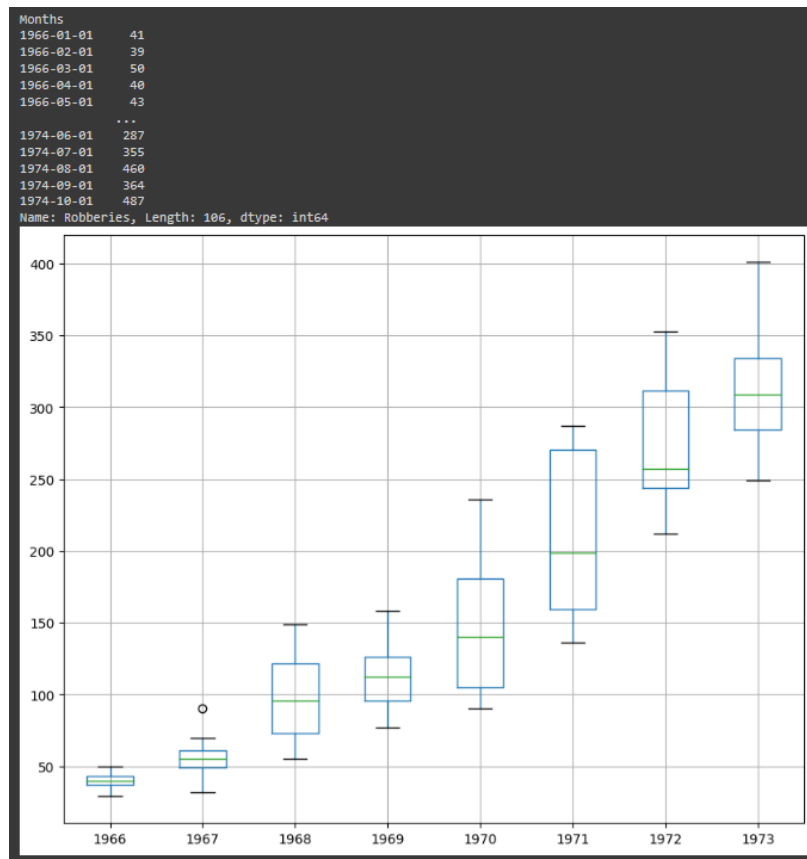
#### 3.4.4 – Box and Whisker Plot:

Đây là một loại đồ thị biểu diễn phân bố của một biến số liên tục, bằng cách sử dụng các phân vị và các giá trị cực trị của dữ liệu. Đồ thị hộp và râu có thể dùng để so sánh phân bố của nhiều nhóm dữ liệu, hoặc để phát hiện các giá trị ngoại lai trong dữ liệu.

Để hiểu rõ hơn về sự biến đổi của các quan sát theo từng năm, chúng ta có thể phân loại dữ liệu hàng tháng theo năm. Chúng ta hy vọng sẽ nhận ra một xu hướng nào đó (giá trị trung bình hoặc trung vị tăng dần), nhưng cũng có thể khám phá xem phân bố của dữ liệu có gì thay đổi hay không. Chúng ta sẽ tạo một biểu đồ Box and Whisker cho mỗi năm dựa trên các quan sát được nhóm theo năm. Vì năm 1974 chỉ có 10 tháng dữ liệu, nên chúng ta không so sánh nó với các năm còn lại có 12 tháng dữ liệu. Chúng ta chỉ vẽ biểu đồ cho dữ liệu từ năm 1966 đến 1973.

```
Box and Whisker Plots

print(series)
groups = series['1966':'1973'].groupby(Grouper(freq='A'))
years = DataFrame()
for name, group in groups:
    years[name.year] = group.values
years.boxplot()
pyplot.show()
```



Nhìn vào kết quả trên chúng ta có thể đưa ra nhận xét như sau:

- Các giá trị trung vị của mỗi năm (đường màu xanh lá cây) cho thấy một xu hướng có thể không phải là đường thẳng.
- Độ phân tán, hay 50% dữ liệu ở giữa (hộp màu xanh lam), có sự khác biệt, nhưng có lẽ không ổn định theo thời gian.
- Các năm đầu tiên, có thể là 2 năm đầu, rất khác biệt so với phần còn lại của tập dữ liệu.

### 3.5 – Mô hình ARIMA:

Trong phần này, chúng ta sẽ xây dựng các mô hình Trung bình động tích hợp Tự hồi quy cho vấn đề. Chúng ta sẽ thực hiện điều này qua 4 bước:

1. Xây dựng một mô hình ARIMA thủ công.
2. Sử dụng lưới tìm kiếm siêu tham số của ARIMA để tìm một mô hình được tối ưu hóa.
3. Phân tích các sai số dự báo để đánh giá bất kỳ độ lệch nào trong mô hình.
4. Khám phá cải tiến cho mô hình bằng cách sử dụng các biến đổi lũy thừa.

### 3.5.1 – Mô hình ARIMA thủ công:

Để phân tích dữ liệu chuỗi thời gian, ta cần sử dụng mô hình ARIMA trái mùa với ba tham số  $p$ ,  $d$ ,  $q$ . Tham số này thường được xác định thủ công. Ta giả sử rằng chuỗi thời gian là ổn định, nhưng thực tế có thể không phải vậy. Ta có thể làm cho chuỗi thời gian ổn định bằng cách lấy sai phân bậc nhất của nó và kiểm tra tính ổn định bằng một bài kiểm tra thống kê. Dưới đây tạo ra một phiên bản ổn định của chuỗi thời gian và lưu nó vào tệp stationary.csv.

```
[23] #tạo ra một chuỗi thời gian khác biệt
def difference(dataset):
    diff = list()
    for i in range(1, len(dataset)):
        value = dataset[i] - dataset[i - 1]
        diff.append(value)
    return pd.Series(diff)

[24] #kiểm tra thống kê cho tính dừng của chuỗi thời gian
series = dataset

X = series.values

# difference data
stationary = difference(X)
stationary.index = series.index[1:]

# kiểm tra xem có đứng yên không
result = adfuller(stationary)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value)) # save

stationary.to_csv('stationary.csv', header=False)
```

```
ADF Statistic: -3.980946
p-value: 0.001514
Critical Values:
    1%: -3.503
    5%: -2.893
   10%: -2.584
```

Đoạn mã trên cho ta kết quả của bài kiểm tra thống kê để kiểm tra tính ổn định của chuỗi sai phân bậc nhất. Đó là bài kiểm tra Dickey-Fuller mở rộng. Kết quả cho thấy giá trị thống kê kiểm định là **-3.980946** bé hơn giá trị tới hạn ở mức 5% là **-2.893**. Điều

này có nghĩa là ta có thể loại bỏ giả thuyết không với mức ý nghĩa nhỏ hơn 5%. Loại bỏ giả thuyết không có nghĩa là quá trình không có căn đơn vị và do đó chuỗi sai phân bậc nhất là ổn định hoặc không có cấu trúc thay đổi theo thời gian.

Điều này cho thấy cần ít nhất một bậc sai phân. Tham số  $d$  trong mô hình ARIMA của chúng ta ít nhất phải bằng 1.

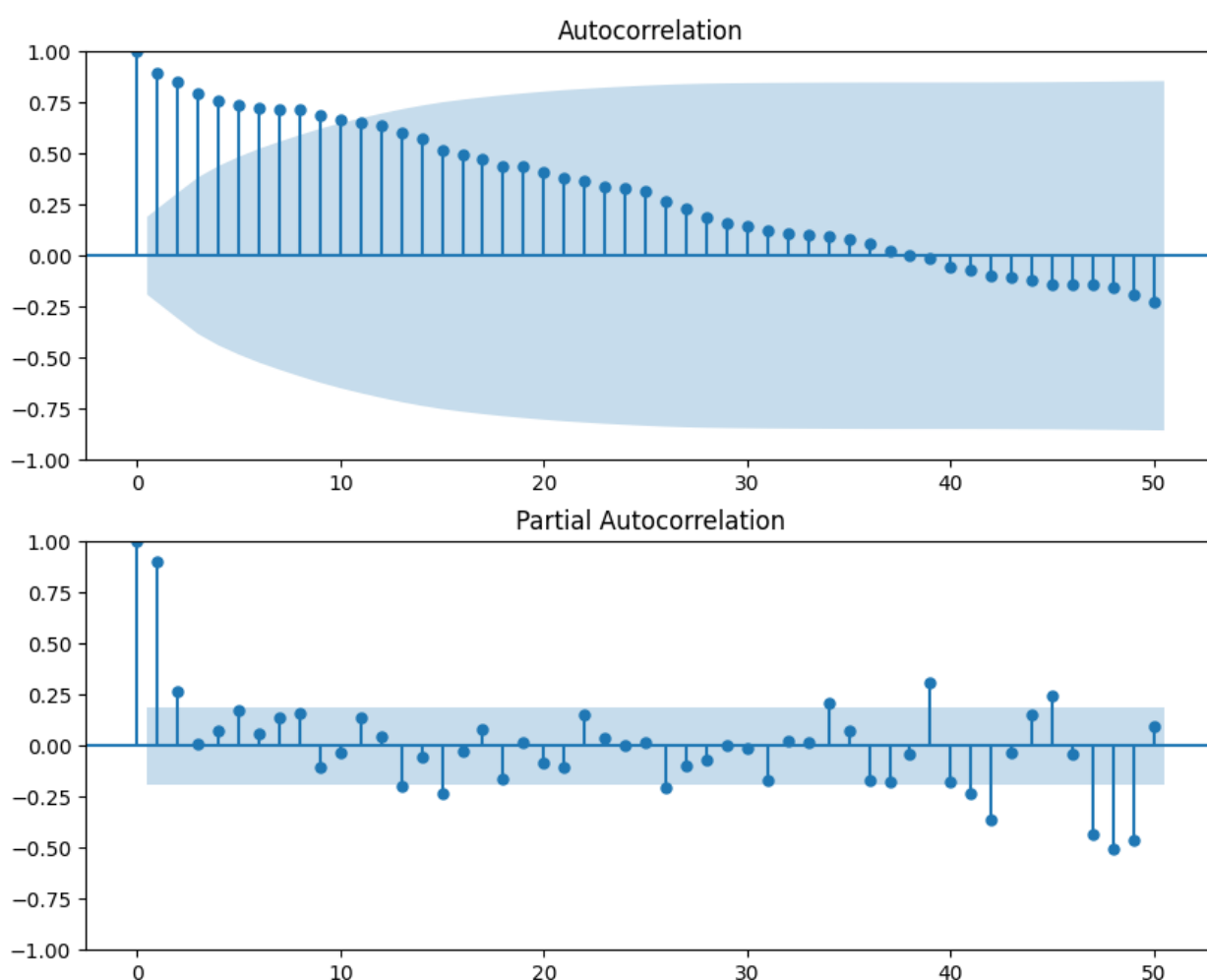
Bước tiếp theo là chọn các giá trị độ trễ cho các tham số AR và MA,  $p$  và  $q$  tương ứng.

Ta có thể làm điều này bằng cách xem các biểu đồ ACF và PACF.

### ACF and PACF plots of time series(Biểu đồ ACF và PACF của chuỗi thời gian)

```
#series = read_csv('dataset.csv', header=None, index_col=0, parse_dates=True, squeeze=True)
pyplot.figure()

series = dataset
pyplot.figure(figsize=(10,8))
pyplot.subplot(211)
plot_acf(series, lags=50, ax=pyplot.gca())
pyplot.subplot(212)
plot_pacf(series, lags=50, ax=pyplot.gca())
pyplot.show()
```



Từ 2 biểu đồ trên chúng ta có thể quan sát thấy:

- ACF cho thấy một độ trễ đáng kể trong khoảng 10-11 tháng.
- PACF cho thấy một độ trễ đáng kể trong khoảng 2 tháng.
- Cả ACF và PACF đều cho thấy sự giảm ở cùng một điểm, có thể gợi ý về sự pha trộn giữa AR và MA.

Một điểm khởi đầu tốt cho giá trị p và q là 1 hoặc 2.

Phân tích nhanh này cho thấy một mô hình ARIMA (1, 1, 2) trên dữ liệu gốc có thể là một điểm khởi đầu tốt. Tuy nhiên, thực nghiệm cho thấy cấu hình ARIMA này không hội tụ và dẫn đến lỗi bởi thư viện bên dưới, giống như các giá trị AR lớn tương tự. Một số thực nghiệm cho thấy mô hình không có vẻ ổn định, với các chỉ số AR và MA không bằng không cùng một lúc. Mô hình có thể được đơn giản hóa thành ARIMA (0, 1, 2).

Đoạn mã bên dưới mô tả hiệu suất của mô hình ARIMA này trên bộ kiểm tra.

#### Evaluate manually configured ARIMA model(Đánh giá mô hình ARIMA cấu hình thủ công)

```
# load data
#series = read_csv('dataset.csv', header=None, index_col=0, parse_dates=True, squeeze=True) # prepare data

from statsmodels.tsa.arima.model import ARIMA

X = ts.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)
train, test = X[0:train_size], X[train_size:]

# xác thực chuyển tiếp
history = [x for x in train]
predict = list()

for i in range(len(test)):
    # Dự báo
    model = ARIMA(history, order=(0,1,2))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
    predict.append(yhat)
    # quan sát
    obs = test[i]
    history.append(obs)
    print('>Dự đoán=%.3f, Dự kiến=%.3f' % (yhat, obs))
```

#### Report Performance(Báo cáo hiệu suất)

```
[30] rmse = sqrt(mean_squared_error(test, predict))
    print('RMSE: %.3f' % rmse)
```

RMSE: 52.494

RMSE là 49.821, thấp hơn so với mô hình Persistence. Đây là một khởi đầu tốt, nhưng chúng ta có thể nhận được kết quả cải thiện với mô hình ARIMA được định cấu hình tốt hơn.

### 3.5.2 - Lưới tìm kiếm Siêu tham số ARIMA:

Ban đầu dự kiến sẽ sử dụng lưới tìm kiếm siêu tham số để khám phá tất cả các tổ hợp trong một tập con của các giá trị nguyên của tổ hợp  $p(0,12)$ ,  $d(0,3)$ ,  $q(0,12)$ . Nhưng thời gian chạy khá dài có thể mất hơn 2 tiếng nên chúng ta sẽ đánh giá mức độ hiệu quả của một mô hình ARIMA dựa vào dự đoán các giá trị trong tương lai của dữ liệu chuỗi thời gian dựa trên các giá trị trong quá khứ của nó.

Đánh giá một mô hình ARIMA cho một thứ tự cho trước  $(p,d,q)$  và trả về RMSE.

Evaluate an ARIMA model for a given order  $(p,d,q)$  and return RMSE(Đánh giá một mô hình ARIMA cho một thứ tự cho trước  $(p,d,q)$  và trả về RMSE)

```
def evaluate_arma_model(X, arma_order): # chuẩn bị tập dữ liệu huấn luyện
    X = X.astype('float32')
    train_size = int(len(X) * 0.50)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # Dự đoán
    predict = list()

    for t in range(len(test)):
        model = ARIMA(history, order=arma_order)
        model_fit = model.fit(dispatch=0)
        yhat = model_fit.forecast()[0]
        predict.append(yhat)
        history.append(test[t])

    # tính toán lỗi mẫu
    rmse = sqrt(mean_squared_error(test, predict))
    return rmse
```

Đoạn mã bên dưới chúng ta sẽ đánh giá các tổ hợp giá trị  $p$ ,  $d$  và  $q$  cho một mô hình ARIMA.

Evaluate combinations of  $p$ ,  $d$  and  $q$  values for an ARIMA model(Đánh giá các tổ hợp giá trị  $p$ ,  $d$  và  $q$  cho một mô hình ARIMA)

```
[32] def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype("float32")
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p, d, q)
                try:
                    rmse = evaluate_arma_model(dataset, order)
                    if rmse < best_score:
                        best_score, best_cfg = rmse, order
                    print("ARIMA%s RMSE=%.3f" % (order, rmse))
                except:
                    continue
    print("Best ARIMA%s RMSE=%.3f" % (best_cfg, best_score))
```



Hàm này sử dụng vòng lặp lồng nhau để lặp qua tất cả các giá trị p, d, q đã cho và gọi hàm `evaluate_arima_model()` để đánh giá mô hình ARIMA được tạo ra từ các giá trị p, d, q này. Nếu độ đo RMSE (Root Mean Squared Error) của mô hình tốt hơn so với các mô hình trước đó, nó sẽ lưu trữ các giá trị p, d, q tương ứng như là giá trị tốt nhất cho đến hiện tại và in ra RMSE và giá trị của các tham số p, d, q đó. Cuối cùng, nó sẽ in ra giá trị tốt nhất của RMSE và giá trị tốt nhất của p, d, q tương ứng.

Evaluate(Đánh giá)

```
%%time
# series = read_csv('dataset.csv', header=None, index_col=0, parse_dates=True, squeeze=True) # evaluate parameters
series = dataset
p_values = range(0,13)
d_values = range(0, 4)
q_values = range(0, 13)
warnings.filterwarnings("ignore")
#evaluate_models(series.values, p_values, d_values, q_values)
```

CPU times: user 55 µs, sys: 0 ns, total: 55 µs  
Wall time: 60.8 µs

Đoạn mã trên sử dụng tập dữ liệu `dataset` và tìm kiếm giá trị tốt nhất cho các tham số p, d và q bằng cách duyệt qua các giá trị trong phạm vi `p_values`, `d_values` và `q_values`. Mỗi lần với các giá trị p, d, q khác nhau, đoạn mã sử dụng hàm `evaluate_arima_model()` để đánh giá mô hình ARIMA với các tham số đó. Nếu RMSE (Root Mean Square Error) của mô hình được tính toán nhỏ hơn giá trị tốt nhất hiện tại, thì giá trị RMSE và các tham số p, d, q đó sẽ được lưu trữ là giá trị tốt nhất hiện tại.

**`warnings.filterwarnings("ignore")`** được sử dụng để tắt thông báo cảnh báo trong quá trình chạy đoạn mã, do đó không hiển thị các thông báo không cần thiết trong quá trình tìm kiếm giá trị tốt nhất.

**`%%time`** là một phương thức trong Jupyter Notebook, để đo thời gian chạy của đoạn mã và hiển thị kết quả.

### 3.5.3 – Xem lại các lỗi dư:

#### Review Residual Errors(Xem xét các sai số dư)

```
[34] # series = read_csv('dataset.csv', header=None, index_col=0, parse_dates=True, squeeze=True) # prepare data
series = dataset
X = series.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)
train, test = X[0:train_size], X[train_size:]

# walk-forward validation
history = [x for x in train]
predictions = list()

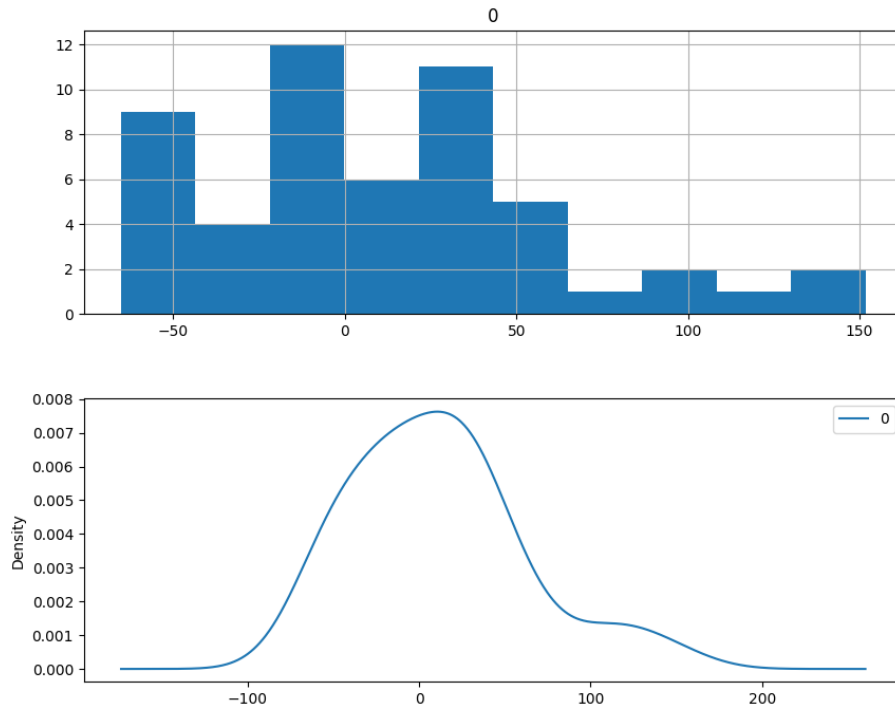
for i in range(len(test)):
    # predict
    model = ARIMA(history, order=(0,1,2))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
    predictions.append(yhat)
    # observation
    obs = test[i]
    history.append(obs)

# errors
residuals = [test[i]-predictions[i] for i in range(len(test))]
residuals = DataFrame(residuals)
```

Một bước kiểm tra cuối cùng tốt cho một mô hình là xem xét các lỗi dự báo dư. Lý tưởng nhất là phân phối các lỗi dư phải là Gaussian với giá trị trung bình bằng không. Chúng ta có thể kiểm tra điều này bằng cách vẽ biểu đồ phân dư với biểu đồ biểu đồ và mật độ. Đoạn mã phía trên tính toán các lỗi còn lại cho các dự đoán trên tập kiểm tra và tạo các biểu đồ mật độ.

#### Plot Residuals

```
pyplot.figure(figsize=(10,8))
pyplot.subplot(211)
residuals.hist(ax=pyplot.gca())
pyplot.subplot(212)
residuals.plot(kind='kde', ax=pyplot.gca())
pyplot.show()
```



Các đồ thị cho thấy một phân phối giống như Gaussian với đuôi phải dài hơn. Đây có thể là một dấu hiệu cho thấy các dự đoán bị thiên lệch và trong trường hợp này có thể là một biến đổi dựa trên lũy thừa của dữ liệu thô trước khi mô hình hóa có thể hữu ích. Cũng là một ý tưởng tốt để kiểm tra chuỗi thời gian của các lỗi dư cho bất kỳ loại tự tương quan nào. Nếu có, điều này sẽ cho thấy mô hình có nhiều cơ hội để mô hình hóa cấu trúc thời gian trong dữ liệu. Đoạn mã dưới đây tính toán lại các lỗi dư và tạo ra các biểu đồ ACF và PACF để kiểm tra bất kỳ tự tương quan đáng kể.

ACF and PACF plots of forecast residual errors(Sơ đồ ACF và PACF của sai số dự báo)

```

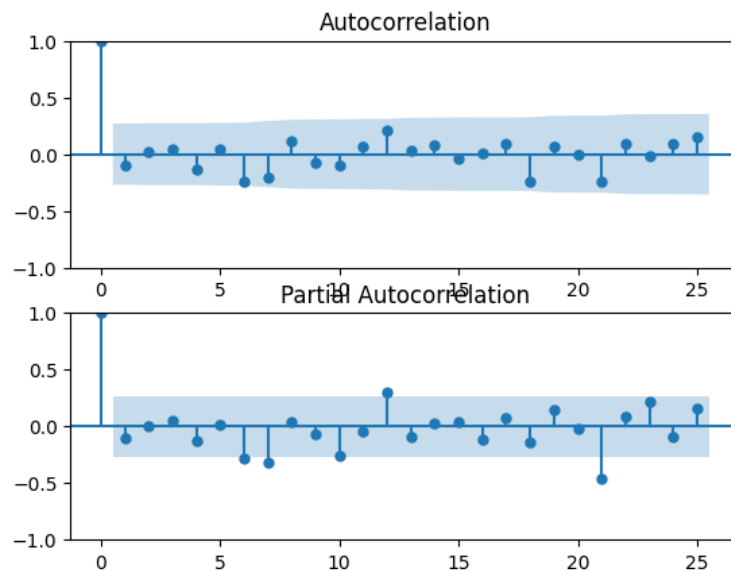
series = dataset
X = series.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)
train, test = X[0:train_size], X[train_size:]
# walk-forward validation
history = [x for x in train]
predictions = list()

for i in range(len(test)):
    # predict
    model = ARIMA(history, order=(0,1,2))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
    predictions.append(yhat)

    # observation
    obs = test[i]
    history.append(obs)

# errors
residuals = [test[i]-predictions[i] for i in range(len(test))]
residuals = DataFrame(residuals)
pyplot.figure()
pyplot.subplot(211)
plot_acf(residuals, lags=25, ax=pyplot.gca())
pyplot.subplot(212)
plot_pacf(residuals, lags=25, ax=pyplot.gca())
pyplot.show()

```



Kết quả cho thấy rằng một chút tự tương quan hiện diện trong chuỗi thời gian đã được mô hình nắm bắt.

### 3.5.4 – Chuyển đổi tập dữ liệu Box-Cox:

#### Box-Cox Transformed Dataset(Tập dữ liệu được biến đổi Box-Cox)

```

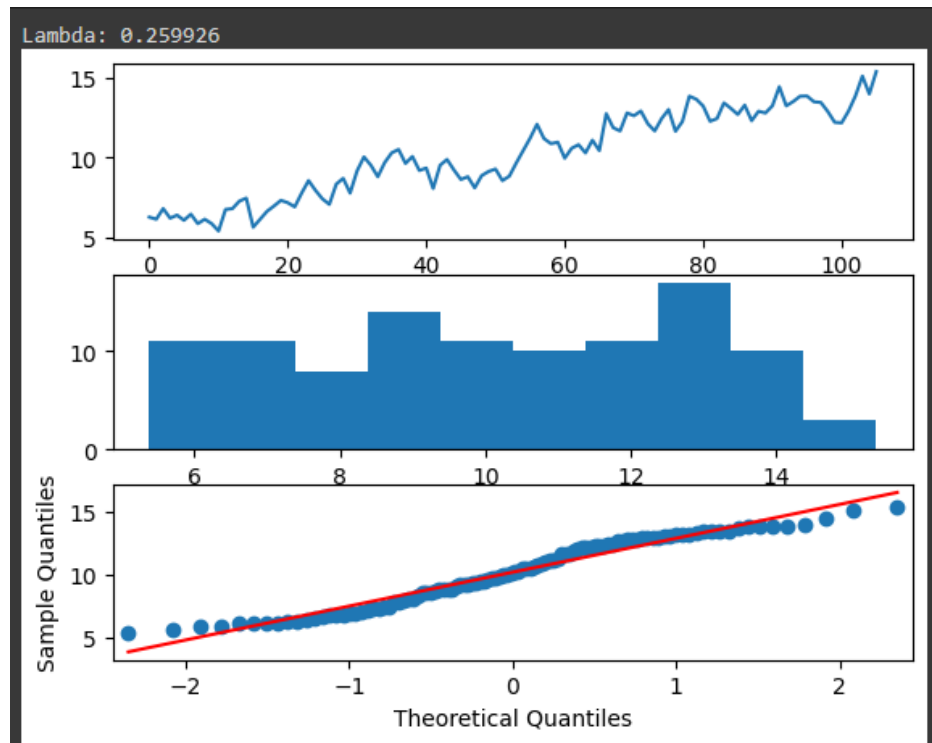
series = dataset
transformed, lam = boxcox(X)
print('Lambda: %f' % lam)
pyplot.figure(1)

#Line Plot
pyplot.subplot(311)
pyplot.plot(transformed)

#Histogram Plot
pyplot.subplot(312)
pyplot.hist(transformed)

#q-q Plot
pyplot.subplot(313)
qqplot(transformed, line='r', ax=pyplot.gca())
pyplot.show()

```



Nhìn vào 3 biểu đồ trên chúng ta ddauw ra nhận xét như sau:

- Những biến động lớn đã được loại bỏ khỏi biểu đồ line của chuỗi thời gian.
- Biểu đồ histogram cho thấy một phân phối phẳng hơn hoặc đồng đều hơn của các giá trị.
- Biểu đồ Q-Q hợp lý, nhưng vẫn chưa phù hợp hoàn hảo với phân phối Gauss.

### Đảo ngược biến đổi Box-Cox:

Invert Box-Cox transform(Đảo ngược phép biến đổi Box-Cox))

```
[79] def boxcox_inverse(value, lam):
    if lam == 0:
        return exp(value)
    return exp(log(lam * value + 1) / lam)

series = dataset
X = series.values
X = X.astype('float32')
train_size = int(len(X) * 0.50)

train, test = X[0:train_size], X[train_size:]

# Kiểm tra tiên bộ
history = [x for x in train]
predictions = list()

for i in range(len(test)):
    # biến đổi
    transformed, lam = boxcox(history)
    if lam < -5:
        transformed, lam = history, 1

    # dự đoán
    model = ARIMA(transformed, order=(0,1,2))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]

    # đảo ngược dự đoán biến đổi
    yhat = boxcox_inverse(yhat, lam)
    predictions.append(yhat)

    # quan sát
    obs = test[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%.3f' % (yhat, obs))
```

```
Report Performance(Báo cáo hiệu suất)

rmse = sqrt(mean_squared_error(test, predictions))
print('RMSE: %.3f' % rmse)

RMSE: 50.223
```

Sai số RMSE cuối cùng của mô hình trên dữ liệu biến đổi là 50.223. Đây là một sai số nhỏ hơn so với mô hình ARIMA dữ liệu không biến đổi, nhưng chỉ nhỏ hơn một chút, và có thể có hoặc không có sự khác biệt về mặt thống kê.

### 3.6 – Kiểm định mô hình:

Sau khi các mô hình đã được phát triển và chọn một mô hình cuối cùng, nó phải được kiểm định và hoàn thiện. Kiểm định là một phần tùy chọn của quá trình, nhưng là một phần cung cấp một kiểm tra cuối cùng để đảm bảo chúng ta không bị lừa hoặc nói dối bản thân. Phần này bao gồm các bước sau:

- Hoàn thiện mô hình: Huấn luyện và lưu mô hình cuối cùng.
- Dự đoán: Tải mô hình đã hoàn thiện và thực hiện dự đoán.
- Kiểm định mô hình: Tải và kiểm định mô hình cuối cùng.

#### 3.6.1 – Hoàn thiện mô hình:

Hoàn thiện mô hình bao gồm việc khớp một mô hình ARIMA trên toàn bộ tập dữ liệu, trong trường hợp này, trên một phiên bản biến đổi của toàn bộ tập dữ liệu. Sau khi khớp, mô hình có thể được lưu vào tệp để sử dụng sau này. Bởi vì một phép biến đổi Box-Cox cũng được thực hiện trên dữ liệu, chúng ta cần biết lambda được chọn để bất kỳ dự đoán nào từ mô hình có thể được chuyển đổi trở lại quy mô gốc, không biến đổi.

#### Finalize Model(Hoàn thiện mô hình)

```
[82] # monkey patch around bug in ARIMA class
def __getnewargs__(self):
    return ((self.endog),(self.k_ar, self.k_diff, self.k_ma))

ARIMA.__getnewargs__ = __getnewargs__

# load data
series = dataset
# prepare data

X = series.values
X = X.astype('float32')

# transform data
transformed, lam = boxcox(X)
```

#### Fit Model(Huấn luyện mô hình)

```
[83] model = ARIMA(transformed, order=(0,1,2))
model_fit = model.fit()
```

#### Save Model(Lưu mô hình)

```
[84] model_fit.save('model.pkl')
np.save('model_lambda.npy', [lam])
```

### 3.6.2 – Dự đoán:

#### Make Prediction(Dự đoán)

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

# Tải mô hình từ tập tin
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

# Tải tham số lambda
lam = np.load('model_lambda.npy')

# Đưa ra dự báo
yhat = model.forecast()[0]

# Đảo ngược biến đổi Box-Cox
if lam == 0:
    yhat = np.exp(yhat)
else:
    yhat = np.exp(np.log(lam*yhat + 1)/lam)

print('Predicted: %.3f' % yhat)
```

Predicted: 452.365

Một trường hợp tự nhiên có thể là tải mô hình và đưa ra một dự báo duy nhất. Điều này liên quan đến việc khôi phục lại mô hình đã lưu và lambda và gọi hàm forecast().

### 3.6.3 – Kiểm định mô hình:

```
# đảo ngược biến đổi box-cox
def boxcox_inverse(value, lam):
    if lam == 0:
        return exp(value)
    return exp(log(lam * value + 1) / lam)

X = dataset.values.astype('float32')
history = [x for x in X]
y = validation.values.astype('float32')

# Tải mô hình
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)
# Tải tham số lambda
lam = np.load('model_lambda.npy')

# đưa ra dự đoán đầu tiên
predictions = list()
yhat = model_fit.forecast()[0]
yhat = boxcox_inverse(yhat, lam)
predictions.append(yhat)
history.append(y[0])
print('>Predicted=%.3f, Expected=%.3f' % (yhat, y[0]))

# Dự báo
for i in range(1, len(y)):
    # Biến đổi
    transformed, lam = boxcox(history)
    if lam < -5:
        transformed, lam = history, 1
    # dự báo
    model = ARIMA(transformed, order=(0,1,2))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
    # đảo ngược dự đoán biến đổi
    yhat = boxcox_inverse(yhat, lam)
    predictions.append(yhat)

    # quan sát
    obs = y[i]
    history.append(obs)
    print('>Predicted=%.3f, Expected=%.3f' % (yhat, obs))
```

```
>Predicted=452.365, Expected=452.000
>Predicted=429.900, Expected=391.000
>Predicted=401.393, Expected=500.000
>Predicted=476.461, Expected=451.000
>Predicted=442.454, Expected=375.000
>Predicted=397.527, Expected=372.000
>Predicted=390.647, Expected=302.000
>Predicted=329.393, Expected=316.000
>Predicted=334.171, Expected=398.000
>Predicted=382.819, Expected=394.000
>Predicted=378.508, Expected=431.000
>Predicted=413.525, Expected=431.000
```



## ▼ Report Performance(Báo cáo hiệu suất)

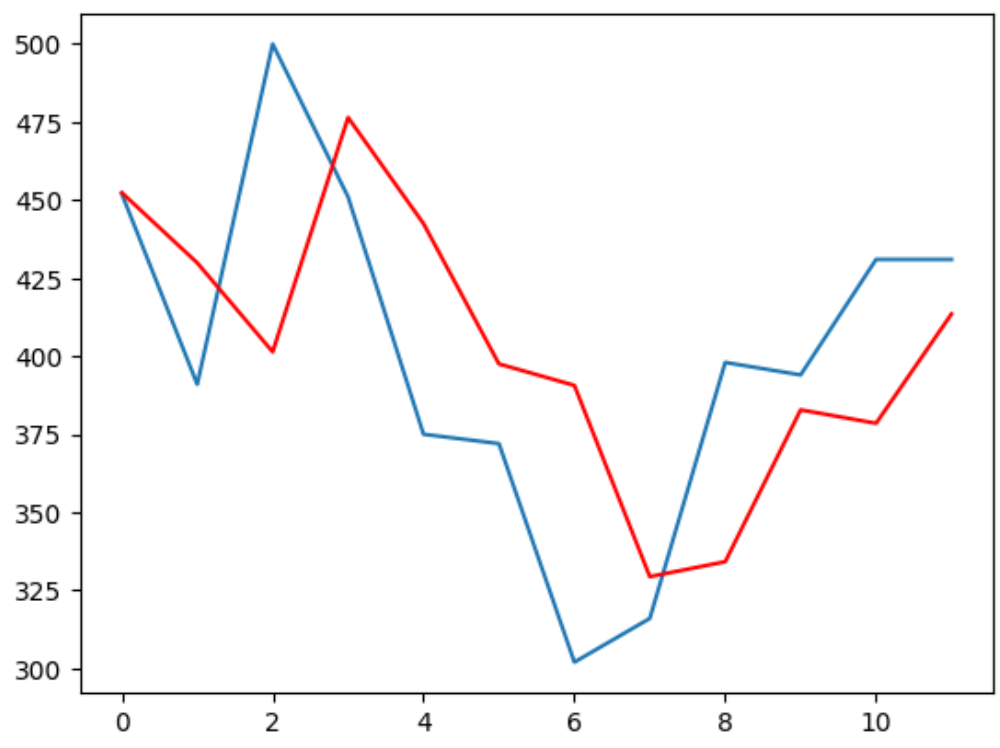
✓  
1  
giây



```
rmse = sqrt(mean_squared_error(y, predictions))  
print('RMSE: %.3f' % rmse)  
pyplot.plot(y)  
pyplot.plot(predictions, color='red')  
pyplot.show()
```



RMSE: 51.949



## CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 1. Kết luận:

Dự đoán tội phạm hàng tháng ở Boston là một bài toán quan trọng trong việc đảm bảo an ninh và trật tự công cộng. Trong bài này, chúng ta đã sử dụng mô hình ARIMA để dự đoán số vụ cướp hàng tháng trong Boston từ năm 1966 đến năm 1975.

Qua quá trình huấn luyện và đánh giá mô hình ARIMA, chúng ta đã chọn ra mô hình ARIMA(0,1,2) là mô hình tốt nhất để dự đoán số vụ cướp hàng tháng trong tương lai. Mô hình này cho thấy độ chính xác khá cao với giá trị RMSE là 51.949.

Ngoài ra, chúng ta đã tìm hiểu về các khái niệm quan trọng trong phân tích chuỗi thời gian, bao gồm chuỗi tăng trưởng và chuỗi dừng, kiểm định Dickey-Fuller để xác định tính dừng của chuỗi, và các phương pháp để xác định các siêu tham số của mô hình ARIMA.

Tuy nhiên, cần lưu ý rằng mô hình ARIMA không thể dự đoán chính xác tất cả các biến đổi của dữ liệu, và việc đưa ra dự đoán phụ thuộc vào chất lượng và tính đại diện của dữ liệu được sử dụng. Do đó, cần tiếp tục nghiên cứu và cải tiến mô hình để đạt được dự đoán chính xác hơn.

### 2. Hướng phát triển:

Trong tương lai, có thể nghiên cứu thêm các phương pháp khác nhau để cải thiện mô hình, chẳng hạn như sử dụng mô hình deep learning như LSTM hoặc CNN để xử lý dữ liệu chuỗi thời gian. Ngoài ra, có thể cải thiện mô hình bằng cách kết hợp với các thông tin khác như thông tin về các sự kiện xảy ra trong khu vực và các yếu tố kinh tế xã hội. Các nghiên cứu khác có thể tập trung vào dự báo tốc độ tăng trưởng của tội phạm trong tương lai và những biện pháp đối phó phù hợp.

Tuy nhiên, việc sử dụng các phương pháp học máy sẽ đòi hỏi một lượng dữ liệu lớn hơn và phức tạp hơn trong quá trình huấn luyện và kiểm định mô hình. Do đó, việc tăng cường thu thập và xử lý dữ liệu sẽ là một yêu cầu quan trọng để đạt được độ chính xác và tính linh hoạt tốt nhất cho mô hình.

## TÀI LIỆU THAM KHẢO

- <https://machinelearningmastery.com/time-series-forecast-case-study-python-monthly-armed-robberies-boston/>
- <https://www.kaggle.com/code/mmellinger66/forecasting-boston-monthly-robberies>
- <https://www.sciencedirect.com/science/article/abs/pii/S027861252200111X>