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

TIỂU LUẬN MÔN HỌC
PHÂN TÍCH DỮ LIỆU CHUỖI THỜI GIAN VÀ ỨNG DỤNG

ỨNG DỤNG MÔ HÌNH ARIMA VÀ KỸ THUẬT PHÂN LỚP ĐỂ DỰ BÁO VÀ PHÂN LOẠI LỢI NHUẬN

Giảng viên giảng dạy : THS.ĐẶNG NHƯ PHÚ
Sinh viên thực hiện : BÙI TIẾN SANG
MSSV : 2000004684
Chuyên ngành : KHOA HỌC DỮ LIỆU
Môn học : PHÂN TÍCH DỮ LIỆU CHUỖI THỜI GIAN VÀ ỨNG DỤNG
Khóa : 2020

Tp.HCM, ngày 11 tháng 5 Năm 2023

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

TIỂU LUẬN MÔN HỌC
PHÂN TÍCH DỮ LIỆU CHUỖI THỜI GIAN VÀ ỨNG DỤNG

ỨNG DỤNG MÔ HÌNH ARIMA VÀ KỸ THUẬT PHÂN LỚP ĐỂ DỰ BÁO VÀ PHÂN LOẠI LỢI NHUẬN

Giảng viên giảng dạy : THS.ĐẶNG NHƯ PHÚ

Sinh viên thực hiện : BÙI TIẾN SANG

MSSV : 2000004684

Chuyên ngành : KHOA HỌC DỮ LIỆU

Môn học : PHÂN TÍCH DỮ LIỆU CHUỖI THỜI GIAN VÀ ỨNG DỤNG

Khóa : 2020

Tp.HCM, ngày 11 tháng 5 Năm 2023

LỜI MỞ ĐẦU

Trong lĩnh vực kinh tế và đầu tư, dự báo và phân loại lợi nhuận là những khía cạnh vô cùng quan trọng để tối ưu hóa hoạt động kinh doanh và đầu tư của các doanh nghiệp. Tuy nhiên, việc dự báo và phân loại này đòi hỏi sự chính xác và nhạy bén, điều đó không phải lúc nào cũng dễ dàng. Vì thế, nhóm em đã tìm tòi và áp dụng nhiều công nghệ mới để giúp cho việc này trở nên dễ dàng và hiệu quả hơn. Trong đó, mô hình ARIMA và kỹ thuật phân lớp là hai trong những phương pháp được áp dụng rộng rãi, giúp cho việc dự báo và phân loại lợi nhuận trở nên chính xác và đáng tin cậy hơn bao giờ hết. Hãy cùng tìm hiểu sâu hơn về ứng dụng của chúng trong các hoạt động kinh doanh và đầu tư của doanh nghiệp của nhóm chúng em qua tiểu luận môn học phân tích chuỗi thời gian và ứng dụng.

LỜI CẢM ƠN

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý Thầy Cô ở Khoa Công Nghệ Thông Tin Trường Đại Học Nguyễn Tất Thành đã truyền đạt vốn kiến thức quý báu của quý thầy cô cho chúng em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của thầy cô nên đề tài ứng dụng mô hình ARIMA và kỹ thuật phân lớp để dự báo và phân loại lợi nhuận thành công tốt đẹp.

Một lần nữa, em xin chân thành cảm ơn thầy Ths.ĐẶNG NHƯ PHÚ – người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn em hoàn thành tốt sản phẩm và bài báo cáo này trong thời gian qua.

Bài báo cáo của em còn hạn chế và còn nhiều bỡ ngỡ nên không tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao ý thức và trình độ của mình của mình.

Em xin chân thành cảm ơn các quý thầy cô rất nhiều!

Sinh viên thực hiện

Bùi Tiến Sang

BM-ChT-

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
TRUNG TÂM KHẢO THÍ

KỲ THI KẾT THÚC HỌC PHẦN
HỌC KỲ II NĂM HỌC 2022 - 2023

PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN

Môn thi:Phân tích chuỗi thời gian và ứng dụng..... Lớp học phần: 20DTH1D.....

Nhóm sinh viên thực hiện: Bùi Tiến Sang +Nguyễn Khánh Toàn .....

1..... Tham gia đóng góp:.....

2..... Tham gia đóng góp:.....

3. ..... Tham gia đóng góp:.....

4. ..... Tham gia đóng góp:.....

5..... Tham gia đóng góp:.....

6..... Tham gia đóng góp:.....

7..... Tham gia đóng góp:.....

8..... Tham gia đóng góp:.....

Ngày thi: 11/05/2023 ..... Phòng thi:L.510.....

Đề tài tiểu luận/báo cáo của sinh viên : ỨNG DỤNG MÔ HÌNH ARIMA VÀ KỸ THUẬT PHÂN LỚP ĐỂ DỰ BÁO VÀ PHÂN LOẠI LỢI NHUẬN

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

Tiêu chí (theo CĐR HP)	**Đánh giá của GV**	**Điểm tối đa**	**Điểm đạt được**
Cấu trúc của báo cáo	....	1,5	
Nội dung			
- Các nội dung thành phần	....	5	
- Lập luận	....	2	
- Kết luận	.....	0.5	
Trình bày	.....	1	
TỔNG ĐIỂM		**10**	

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

NHẬN XÉT CỦA GIẢNG VIÊN GIẢNG DẠY

Tp.HCM, Ngày . . . . tháng . . . . năm . . . .

Giảng viên giảng dạy

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
LTSM	Long Short-Term Memory
ARIMA	AutoRegressive Integrated Moving Average
ML	Machine learning
MA	Moving Average
AR	Autoregressive
GARCH	Generalized Autoregressive Conditional Heteroskedasticity

CHƯƠNG 1: GIỚI THIỆU

1.Giới thiệu đề tài.

2.Lý do chọn đề tài.

Chuỗi thời gian đang được sử dụng như một công cụ hữu hiệu để phân tích và dự báo trong kinh tế xã hội cũng như trong nghiên cứu khoa học.Chính do tầm quan trọng của phân tích chuỗi thời gian, rất nhiều nghiên cứu đã đề xuất các công cụ để phân tích và dự báo chuỗi thời gian. Trong những năm trước, công cụ để phân tích chuỗi thời gian là sử dụng các công cụ thống kê như hồi quy, phân tích Furie và một vài công cụ khác.Nhưng hiệu quả nhất là mô hình ARIMA của Box-Jenkins. Từ các công trình ban đầu về chuỗi thời gian, hiện nay mô hình này đang được dùng rất nhiều để phân tích và dự báo trong các ngành kinh tết tài chính chứng khoáng , giá vàng.

Nghiên cứu phân tích và dự báo chuỗi thời gian luôn là một bài toángây được sự chú ý của các nhà toán học, kinh tế, xã hội học,... Các quan sát trong thực tế thường được thu thập dưới dạng chuỗi số liệu. Từ những số liệu này, người ta có thế rút ra được những quy luật của một quá trìnhđược mô tả thông qua chuỗi số liệu.Xuất phát từ thực tế ứng dụng lớn của mô hình ARIMA, em chọn đề tài nghiên cứu về: "Ứng dụng mô hình ARIMA và kỹ thuật phân lớp để dự báo và phân loại lợi nhuận" để là về bài tiểu luận của mình.

3.Mục tiêu đề tài.

Nghiên cứu một số khái niệm và tính chất cơ bản về chuỗi thời gian; các quá trình trung bình (MA), quá trình tự hồi quy (AR), quá trình trùng bình trượt tự hồi quy (ARMA) và quá trình trung bình trượt, tích hợp tự hồi quy (ARIMA).

Đối tượng nghiên cứu: Mô hình ARIMA

Phạm vi nghiên cứu: Mô hình ARIMA, phương pháp Box - Jenkins,

Ứng dụng mô hình ARIMA và kỹ thuật phân lớp để dự báo và phân loại lợi nhuận

4.Phương pháp đề tài.

- Phương pháp so sánh, phân tích, tổng hợp kiến thức.
- Phương pháp phân tích thực nghiệm với dữ liệu thực tế.
- Sử dụng phần mềm Excel, GGColab.

5.Đối tượng và phạm vi nghiên cứu.

- Chuẩn bị và trình bày đầy đủ một số khái niệm và kiến thức cơ bản sẽ được sử dụng trong lúc thực hiện báo cáo và đồ án.
- Mô hình ARIMA và ứng dụng áp dụng mô hình và nghiên cứu trình bay cơ bản về bài báo cáo về "ứng dụng mô hình ARIMA và kỹ thuật phân lớp để dự báo và phân loại lợi nhuận".
- Đối tượng của ứng dụng mô hình ARIMA và kỹ thuật phân lớp để dự báo và phân loại lợi nhuận là tập dữ liệu chứa thông tin về lợi nhuận của doanh nghiệp trong quá khứ. Mô hình ARIMA (Autoregressive Integrated Moving Average) được sử dụng để dự báo lợi nhuận tiếp theo dựa trên các giá trị lịch sử. Kỹ thuật phân lớp được sử dụng để phân loại dữ liệu lợi nhuận thành các nhóm tương đương với nhau, để giúp phân tích và quản lý dữ liệu dễ dàng hơn. Từ đó, các quyết định kinh doanh có thể được đưa ra dựa trên dự báo và phân loại lợi nhuận này, giúp doanh nghiệp quản lý tài chính và tối ưu hóa kết quả kinh doanh.

CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT

1.Giới thiệu về mô hình ARIMA.

1.1 Giới thiệu về chuỗi thời gian.

Dự báo chuỗi thời gian là một lớp mô hình quan trọng trong thống kê, kinh tế lượng và machine learning. Sở dĩ chúng ta gọi lớp mô hình này là chuỗi thời gian (*time series*) là vì mô hình được áp dụng trên các chuỗi đặc thù có yếu tố thời gian. Một mô hình chuỗi thời gian thường dự báo dựa trên giả định rằng các qui luật trong quá khứ sẽ lặp lại ở tương lai. Do đó xây dựng mô hình chuỗi thời gian là chúng ta đang mô hình hóa mối quan hệ trong quá khứ giữa biến độc lập (biến đầu vào) và biến phụ thuộc (biến mục tiêu). Dựa vào mối quan hệ này để dự đoán giá trị trong tương lai của biến phụ thuộc.

Vai trò của chuỗi thời gian rất quan trọng đối với nền kinh tế và hoạt động của doanh nghiệp nên trong machine learning và thống kê có những ngành học nghiên cứu chuyên sâu về chuỗi thời gian như kinh tế lượng, định giá tài sản tài chính.

Khác với các mô hình dự báo thông thường trong machine learning, các mô hình trong dự báo chuỗi thời gian trong kinh tế lượng có những đặc trưng rất riêng. Đòi hỏi phải tuân thủ nghiêm ngặt các điều kiện về chuỗi dừng, nhiễu trắng và tự tương quan. Có rất nhiều lớp mô hình chuỗi thời gian khác nhau và mỗi một lớp mô hình sẽ có một tiêu chuẩn áp dụng cụ thể. Ở đây ta sài mô hình ARIMA:

Mô hình ARIMA: Dựa trên giả thuyết chuỗi dừng và phương sai sai số không đổi. Mô hình sử dụng đầu vào chính là những tín hiệu quá khứ của chuỗi được dự báo để dự báo nó. Các tín hiệu đó bao gồm: chuỗi tự hồi qui AR (auto regression) và chuỗi trung bình trượt MA (moving average). Hầu hết các chuỗi thời gian sẽ có xu hướng tăng hoặc giảm theo thời gian, do đó yếu tố chuỗi dừng thường không đạt được. Trong trường hợp chuỗi không dừng thì ta sẽ cần biến đổi sang chuỗi dừng bằng sai phân. Khi đó tham số đặc trưng của mô hình sẽ có thêm thành phần bậc của sai phân d và mô hình được đặc tả bởi 3 tham số ARIMA(p, d, q).

1.2. Mô hình ARIMA.

Hiện tại cả R và python đều support xây dựng các mô hình chuỗi thời gian ARIMA, SARIMA, ARIMAX, GARCH,…. Trên R chúng ta có thể sử dụng các packages

như forecast và lmtest để xây dựng các mô hình này khá dễ dàng. Đối với thống kê và các mô hình chuỗi thời gian R đang support tốt hơn python. Một lý do đó là các nhà thống kê và kinh tế lượng ưa chuộng sử dụng R hơn. Hướng dẫn xây dựng mô hình ARIMA trên R các bạn có thể xem tại ARIMA turtorial và GARCH time series model.

1.3. Lý thuyết mô hình ARIMA.

Lý thuyết: Chúng ta biết rằng hầu hết các chuỗi thời gian đều có sự tương quan giữa giá trị trong quá khứ đến giá trị hiện tại. Mức độ tương quan càng lớn khi chuỗi càng gần thời điểm hiện tại. Chính vì thể mô hình ARIMA sẽ tìm cách đưa vào các biến trễ nhằm tạo ra một mô hình dự báo fitting tốt hơn giá trị của chuỗi.

ARIMA model là viết tắt của cụm từ Autoregressive Intergrated Moving Average. Mô hình sẽ biểu diễn phương trình hồi qui tuyến tính đa biến (multiple linear regression) của các biến đầu vào (còn gọi là biến phụ thuộc trong thống kê) là 2 thành phần chính:

Auto regression: Kí hiệu là AR. Đây là thành phần tự hồi qui bao gồm tợp hợp các độ trễ của biến hiện tại. Độ trễ bậc p chính là giá trị p lùi về quá khứ p bước thời gian của chuỗi. Độ trễ dài hoặc ngắn trong quá trình AR(p) phụ thuộc vào tham số trễ Xt . Cụ thể, quá trình AR của chuỗi được biểu diễn như bên dưới:

$$\mathrm{AR}(p) = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p}$$

Moving average: Qúa trình trung bình trượt được hiểu là quá trình dịch chuyển hoặc thay đổi giá trị trung bình của chuỗi theo thời gian. Do chuỗi của chúng ta được giả định là dừng nên quá trình thay đổi trung bình dường như là một chuỗi nhiễu trắng. Qúa trình moving average sẽ tìm mối liên hệ về mặt tuyến tính giữa các phần tử ngẫu nhiên ϵ_t (stochastic term). Chuỗi này phải là một chuỗi nhiễu trắng thỏa mãn các tính chất:

$$\begin{cases} \mathrm{E}(\epsilon_t) & = 0 & (1) \\ \sigma(\epsilon_t) & = \alpha & (2) \\ \rho(\epsilon_t, \epsilon_{t-s}) & = 0, \forall s <= t & (3) \end{cases}$$

Vế (1) có nghĩa rằng kì vọng của chuỗi bằng 0 để đảm bảo chuỗi dừng không có sự thay đổi về trung bình theo thời gian. Vế (2) là phương sai của chuỗi không đổi. Do kì vọng và

phương sai không đổi nên chúng ta gọi phân phối của nhiễu trắng là phân phối xác định (identical distribution) và được kí hiệu là MA.Nhiễu trắng là một thành phần ngẫu nhiên thể hiện cho yếu tố không thể dự báo của model và không có tính qui luật. Qúa trình trung bình trượt được biểu diễn theo nhiễu trắng như sau:

$$\mathrm{MA}(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Qúa trình này có thể được biểu diễn theo dịch chuyển trễ - backshift operator *B* như sau:

$$\mathrm{MA}(q) = \mu + (1 + \theta_1 B + \cdots + \theta_q B^q)\epsilon_t$$

Như vậy bạn đọc đã hình dung ra moving average là gì rồi chứ? Về mặt ý tưởng thì đó chính là quá trình hồi qui tuyến tính của giá trị hiện tại theo các giá trị hiện tại và quá khứ của sai số nhiễu trắng (white noise error term) đại diện cho các yếu tố shock ngẫu nhiên, những sự thay đổi không lường trước và giải thích bởi mô hình.

Intergrated: Là quá trình đồng tích hợp hoặc lấy sai phân. Yêu cầu chung của các thuật toán trong time series là chuỗi phải đảm bảo tính dừng. Hầu hết các chuỗi đều tăng hoặc giảm theo thời gian. Do đó yếu tố tương quan giữa chúng chưa chắc là thực sự mà là do chúng cùng tương quan theo thời gian. Khi biến đổi sang chuỗi dừng, các nhân tố ảnh hưởng thời gian được loại bỏ và chuỗi sẽ dễ dự báo hơn. Để tạo thành chuỗi dừng, một phương pháp đơn giản nhất là chúng ta sẽ lấy sai phân. Một số chuỗi tài chính còn qui đổi sang logarit hoặc lợi suất. Bậc của sai phân để tạo thành chuỗi dừng còn gọi là bậc của quá trình đồng tích hợp (order of intergration). Qúa trình sai phân bậc *d* của chuỗi được thực hiện như sau:

- Sai phân bậc 1: $\mathrm{I}(1) = \Delta(x_t) = x_t - x_{t-1}$
- Sai phân bậc d: $\mathrm{I}(d) = \Delta^d(x_t) = \underbrace{\Delta(\Delta(\ldots\Delta(x_t)))}_{\text{d times}}$

Thông thường chuỗi sẽ dừng sau quá trình đồng tích hợp I(0) hoặc I(1). Rất ít chuỗi chúng ta phải lấy tới sai phân bậc 2. Một số trường hợp chúng ta sẽ cần biến đổi logarit hoặc căn bậc 2 để tạo thành chuỗi dừng. Phương trình hồi qui ARIMA(p, d, q) có thể

được biểu diễn dưới dạng:

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + \ldots + \phi_p \Delta x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

Trong đó Δxt là giá trị sai phân bậc d và ϵ_t là các chuỗi nhiễu trắng.

Như vậy về tổng quát thì ARIMA là mô hình kết hợp của 2 quá trình tự hồi qui và trung bình trượt. Dữ liệu trong quá khứ sẽ được sử dụng để dự báo dữ liệu trong tương lai. Trước khi huấn luyện mô hình, cần chuyển hóa chuỗi sang chuỗi dừng bằng cách lấy sai phân bậc 1 hoặc logarit. Ngoài ra mô hình cũng cần tuân thủ điều kiện ngặt về sai số không có hiện tượng tự tương quan và phần dư là nhiễu trắng. Đó là lý thuyết của kinh tế lượng. Còn theo trường phái machine learning thì tôi chỉ cần quan tâm đến làm sao để lựa chọn một mô hình có sai số dự báo là nhỏ nhất. Tiếp theo chúng ta sẽ sử dụng package vnquant, một package được tôi viết để hỗ trợ cộng đồng khai thác dữ liệu chứng khoán thuận tiện hơn.

2.Mô tả thuật toán.

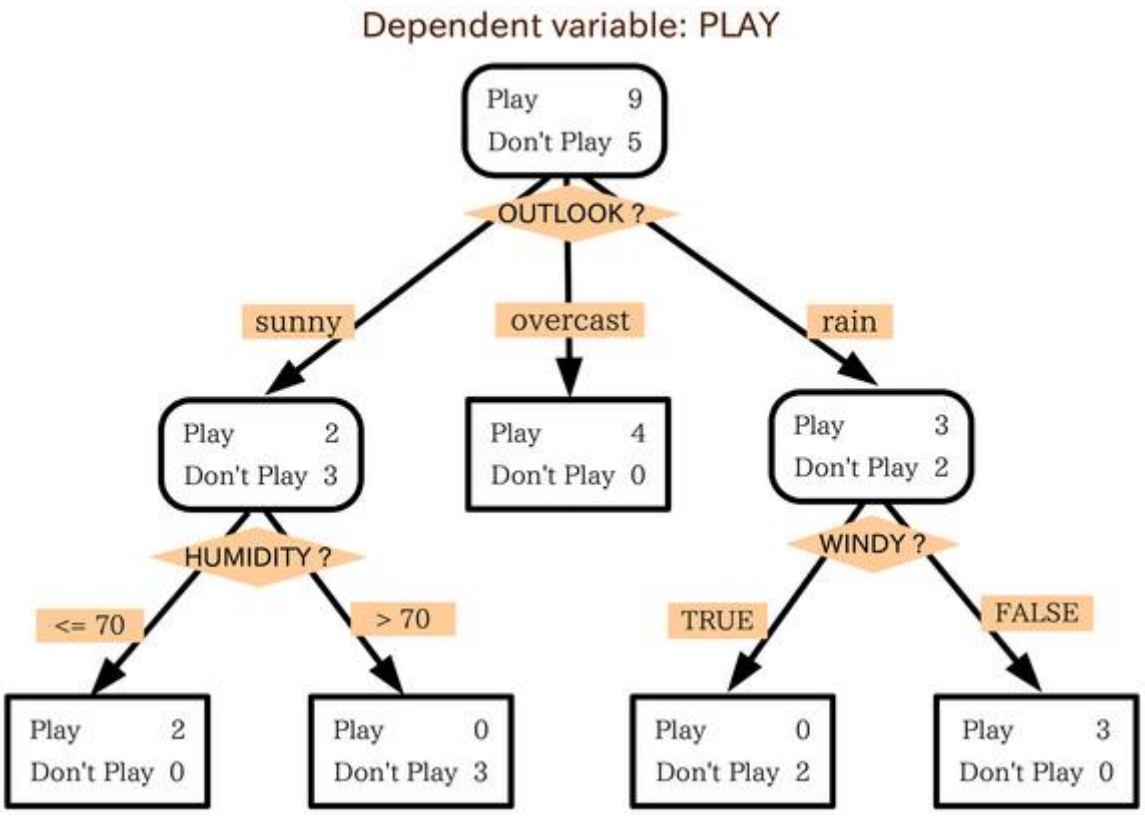
2.1.Cây quyết định.

Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán phân loại và dự báo của học có giám sát. Khác với những thuật toán khác trong học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập huấn luyện và sử dụng cây quyết định này dự báo trên tập kiểm tra.

Vậy một cây quyết định sẽ được xây dựng như thế nào? Điều gì ẩn chứa sau thuật toán cây quyết định? Để trả lời cho câu hỏi này, chúng ta cùng lấy ví dụ về áp dụng mô hình cây quyết định cho bài toán phân loại giá nhà trên bộ dữ liệu boston.

2.1.1.Mô hình cây quyết định.

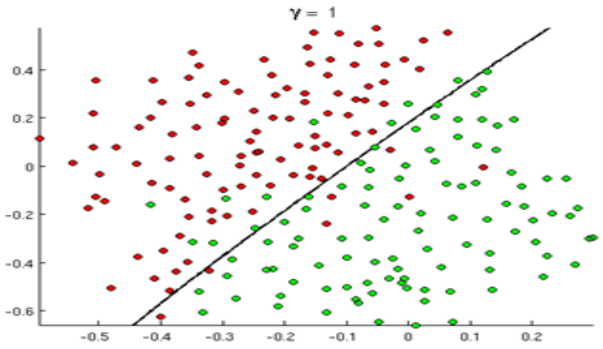
Xuất phát từ nút gốc (root) tới nút trung gian (nút nội – internal node) và kết thúc ở nút lá (leaf node). Trong đó: Nút gốc và nút nội chứa các biểu thức điều kiện rẽ nhánh



Hình 1 : Cây quyết định.

2.2.Hồi quy Logistic

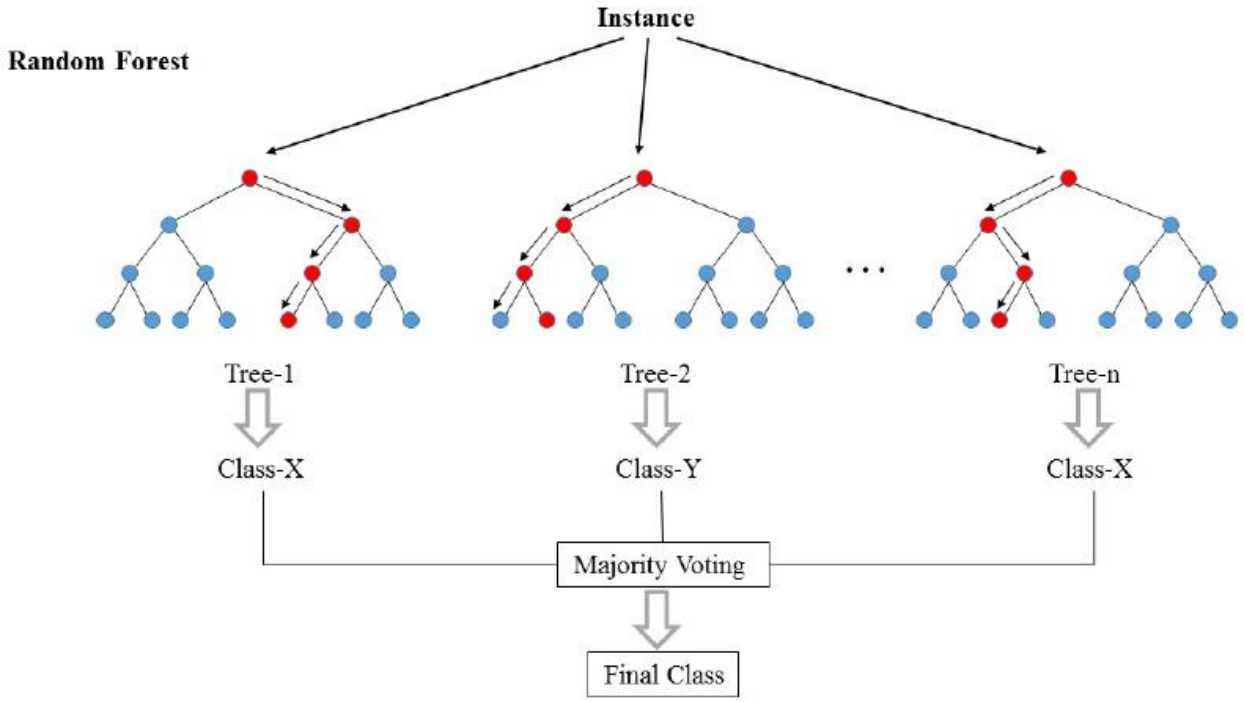
Hồi quy logistic là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.



Hình 2: Hồi Quy Logistic

2.3. Random Forest.

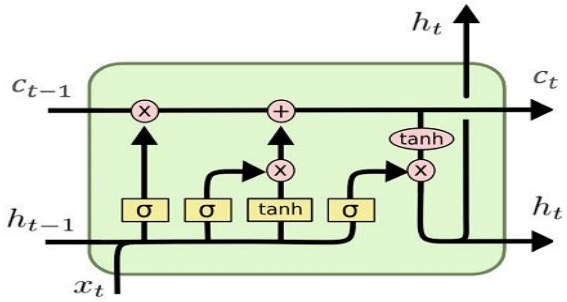
Random forest là một phương pháp thống kê mô hình hóa bằng máy (machine learning statistic) dùng để phục vụ các mục đích phân loại, tính hồi quy và các nhiệm vụ khác bằng cách xây dựng nhiều cây quyết định (Decision tree).



Hình 3: Random Forest

2.4. LTSM

LSTM là một mạng thần kinh hồi quy (RNN) nhân tạo được sử dụng trong lĩnh vực học sâu. Không giống như các mạng thần kinh truyền thẳng (FNN) tiêu chuẩn, LSTM có chứa



Hình 4:LTSM

các kết nối phản hồi. Mạng không chỉ xử lý các điểm dữ liệu đơn lẻ (như các hình ảnh), mà còn xử lý toàn bộ chuỗi dữ liệu (chẳng hạn như lời nói hoặc video). Ví dụ, LSTM có thể áp dụng cho các tác vụ nhận dạng chữ viết tay, nhận dạng tiếng nói và phát hiện bất thường có tính chất kết nối, không phân đoạn trong giao thông mạng hoặc các IDS (hệ thống phát hiện xâm nhập).

2.5 Xây dựng bộ dữ liệu

Dữ liệu được thống kê và lấy trên kaggle

	A	B	C	D	E	F	G
1	NgayThang	Gia_MoCua	Gia_Tran	Gia_San	Gia_DongCua	KhoiLuongGD	LoiNhuan_Hay_Khor
2	12-12-1980	0.1003	0.1007	0.1003	0.1003	469033600	0
3	15-12-1980	0.0955	0.0955	0.0951	0.0951	175884800	1
4	16-12-1980	0.0885	0.0885	0.0881	0.0881	105728000	1
5	17-12-1980	0.0902	0.0907	0.0902	0.0902	86441600	0
6	18-12-1980	0.0929	0.0933	0.0929	0.0929	73449600	
7	19-12-1980	0.0985	0.099	0.0985	0.0985	48630400	0
8	22-12-1980	0.1034	0.1038	0.1034	0.1034	37363200	0
9	23-12-1980	0.1077	0.1081	0.1077	0.1077	46950400	0
10	24-12-1980	0.1134	0.1138	0.1134	0.1134	48003200	0
11	26-12-1980	0.1238	0.1243	0.1238	0.1238	55574400	0
12	29-12-1980	0.1256	0.126	0.1256	0.1256	93161600	0
13	30-12-1980	0.123	0.123	0.1225	0.1225	68880000	1
14	31-12-1980	0.1195	0.1195	0.1191	0.1191	35750400	1

Hình 5: Bảng thống kê dữ liệu

Theo dữ liệu trên lấy và thực hiện kê khai các mục về thụp nhập đầu ra vào áp dụng để thống kê lợi nhuận và làm bài ARIMA.

CHƯƠNG 3. THỰC NGHIỆM

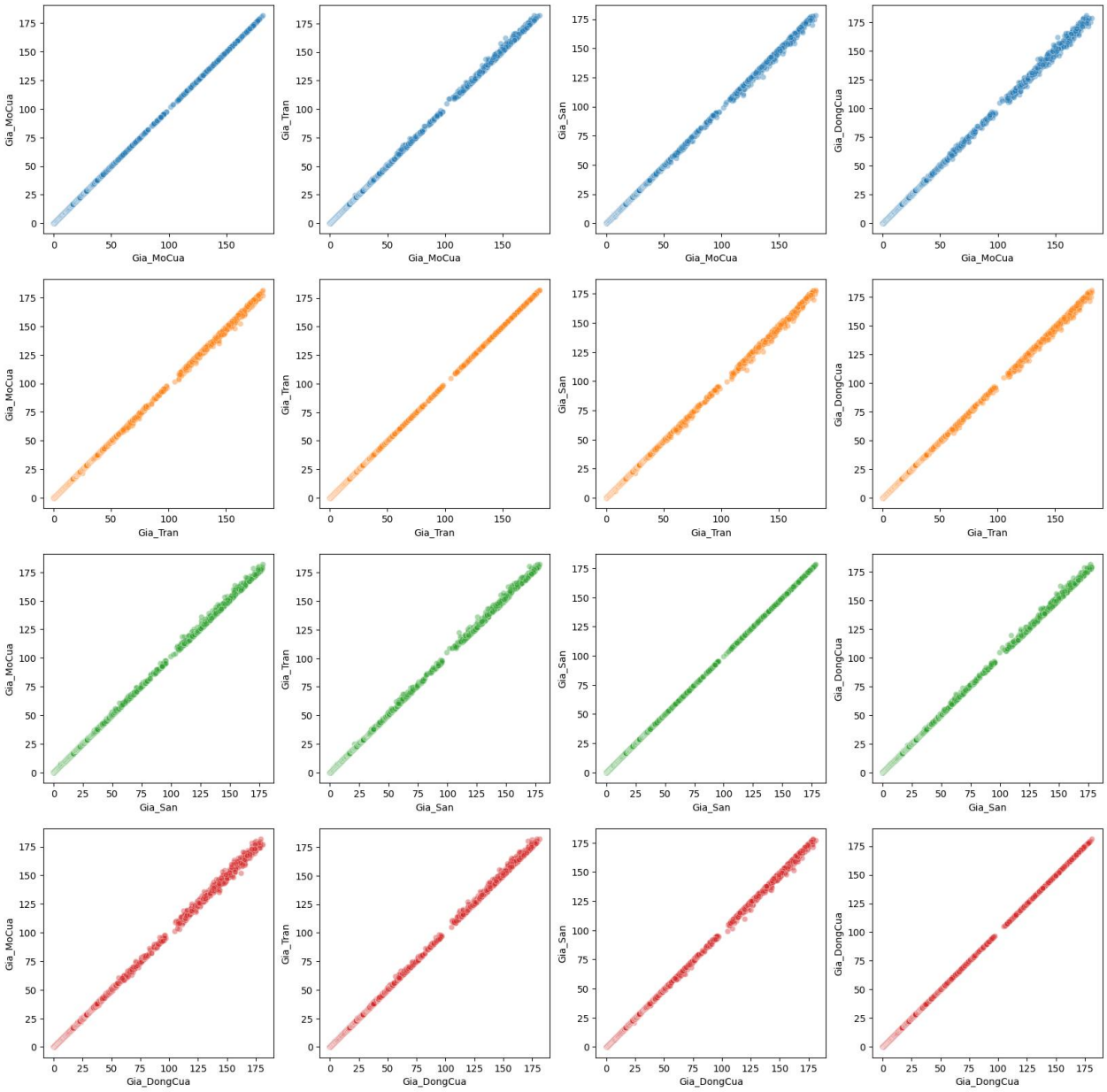
1.Trực quan dữ liệu bằng các biểu đồ.

Sau khi loading file thì em đã tiến hành kiểm tra các dữ liệu xem có đồng bộ không,có chỗ nào bị null hay là bị thiếu xót gì không,để tránh dài dòng nên em sẽ không đề cập ở đây mà sẽ đưa ra một bảng xử lý dữ liệu hoàn tất và xây dựng mối tương quan giữa các cột bằng tree map.

Hình 6 : Biểu đồ Treemap

Ta có thể thấy sự tương quan của các dòng dữ liệu rất là đồng đều với nhau khi chỉ giao động từ 0-1 của các cột Gia_MoCua,Gia_Tran,Gia_San,Gia_DongCua,KhoiLuongGD và LoiNhuan_Hay_Không nên có thể thấy việc xử lý dữ liệu đã vô cùng ổn định.

Tiếp theo thì tiến hành vẽ suplots để hể hiện độ tương quan của đồ thị.



Hình 7: Suplot tương quan

Với sự tương quan của mình thì ta có thể nhận định được đây chính là dữ liệu rất đẹp có sự tương quan đồng đều với nhau.

2.Xây dựng chương trình ứng dụng.

2.1 Xây dựng các biến độc lập.

```
# Lấy 2  Cột làm biến độc lập
inputs = data[['Gia_MoCua','Gia_DongCua']]
# Lấy 1 Cột làm biến phụ thuộc
targets = data['LoiNhuan_Hay_Khong']
# Lấy ngẫu nhiên 80% dòng để máy học và 20% dòng để kiểm tra
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( inputs, targets, train_
size = 0.8, test_size=0.2, random_state=16,)
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)
```

Để tăng sự chính xác của mô hình thì ta lấy 2 cột làm biến độc lập là Gia_MoCua, Gia_DongCua làm biến độc lập và dùng LoiNhuan_Hay_Khong để phụ thuộc sau đó Lấy ngẫu nhiên 80% dòng để máy học và 20% dòng để kiểm tra.

2.2 Cây quyết đinh.

```
# Cây Quyết Định
from sklearn import datasets
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Triển khai cây quyết định với độ đo là Entropy và độ sâu
model_DecisionTree = DecisionTreeClassifier(criterion='entropy')
model_DecisionTree.fit(X_train, y_train)
prediction_DecisionTree = model_DecisionTree.predict(X_test)
DecisionTree = accuracy_score(y_test, prediction_DecisionTree)

print(f'Từ bộ dữ liệu có {len(data)} dòng ta lấy ra {len(X_train)} dòng cho
máy học và dùng {len(X_test)} dòng để kiểm tra')
print(f'Cây Quyết Định cho ra kết quả: {DecisionTree*100}% Độ Chính Xác')
#Kết quả đạt được.
Từ bộ dữ liệu có 10559 dòng ta lấy ra 8447 dòng cho máy học và dùng 2112
dòng để kiểm tra
Cây Quyết Định cho ra kết quả: 87.7840909090909% Độ Chính Xác
```

Sau khi sử dụng các thư viện cần thiết thì tiến hành thực hiện thuật toán cây quyết định với độ đo là Entropy và độ sâu của thuật toán rồi tiến hành thực hiện mô hình và kết quả thì ra độ chính xác khá cao xấp xỉ là : 87,78%.

2.3 Hồi quy Logistic.

Tương tự với cây quyết định hồi quy logistic cũng dùng với mục đích là dự đoán có lợi nhuận hay không.

```
# Hồi Quy Logistic from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
model_Logistic = LogisticRegression()
model_Logistic.fit(X_train, y_train)
prediction_Logistic = model_Logistic.predict(X_test)
Logistic_Acc = accuracy_score(y_test, prediction_Logistic)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( inputs, targets, train_
size = 0.8, test_size=0.2,)
#In dữ liệu của mô hình .
print(f'Từ bộ dữ liệu có {len(data)} dòng ta lấy ra {len(X_train)} dòng cho
máy học và dùng {len(X_test)} dòng để kiểm tra')
print(f'Hồi Quy Logistic cho ra kết quả: {Logistic_Acc*100}% Độ Chính Xác')
#Kết quả của mô hình.

Từ bộ dữ liệu có 10559 dòng ta lấy ra 8447 dòng cho máy học và dùng 2112
dòng để kiểm tra
Hồi Quy Logistic cho ra kết quả: 82.85984848484848% Độ Chính Xác
```

Hồi Quy Logistic cho ra kết quả: 82.85984848484848% Độ Chính Xác.

2.4 Random Forest.

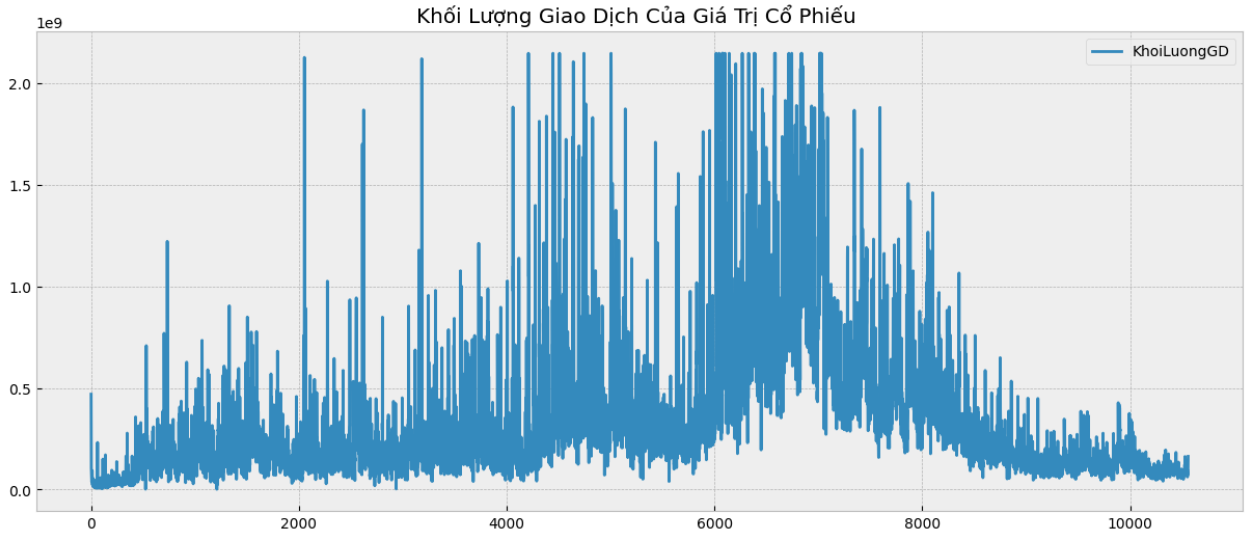
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Create a Random Forest classifier
rfc = RandomForestClassifier(n_estimators=100)
# Train the classifier using the training data
rfc.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = rfc.predict(X_test)
# Calculate the accuracy of the classifier
Random = accuracy_score(y_test, y_pred)
print(f'Từ bộ dữ liệu có {len(data)} dòng ta lấy ra {len(X_train)} dòng cho
máy học và dùng {len(X_test)} dòng để kiểm tra')
print(f' Random Forest cho ra kết quả: {Random*100}% Độ Chính Xác')
#In kết quả của mô hình.

Từ bộ dữ liệu có 10559 dòng ta lấy ra 8447 dòng cho máy học và dùng 2112
dòng để kiểm tra
Random Forest cho ra kết quả: 89.15719696969697% Độ Chính Xác
```

Có thể thấy Random Forest sẽ cho ra kết quả cao hơn so với cây quyết định và Logistic.

3.Mô hình ARIMA.

Ở đây thì chúng ta sẽ bỏ hết các cột còn lại và chỉ lấy cột KhoiLuongGD làm biến độc lập mà thôi dưới đây là biểu đồ thể hiện khối lượng giao dịch.

```
import matplotlib.pyplot as plt
df.plot(figsize=(15, 6))
plt.title('Khối Lượng Giao Dịch Của Giá Trị Cổ Phiếu ')
plt.show()
```



Hình 8: Khối Lượng Giao Dịch

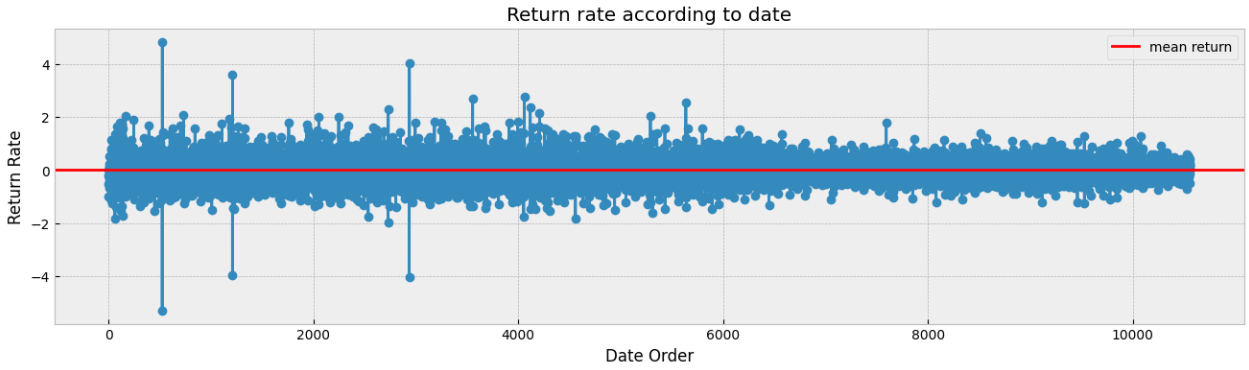
Tiến hành tính chuỗi return.

```
import numpy as np
# Tính chuỗi return
import numpy as np
r_t = np.log(df['KhoiLuongGD'] / df['KhoiLuongGD'].shift(1))
mean = np.nanmean(r_t)
r_t[0] = mean  # sử dụng mean thay vì mean
r_t[:5]
#In ra kết quả.
0 -0.000100
1 -0.980845
2 -0.508959
3 -0.201401
4 -0.162870
Name: KhoiLuongGD, dtype: float64
```

Tiếp tục vẽ biểu đồ Return rate according to date.

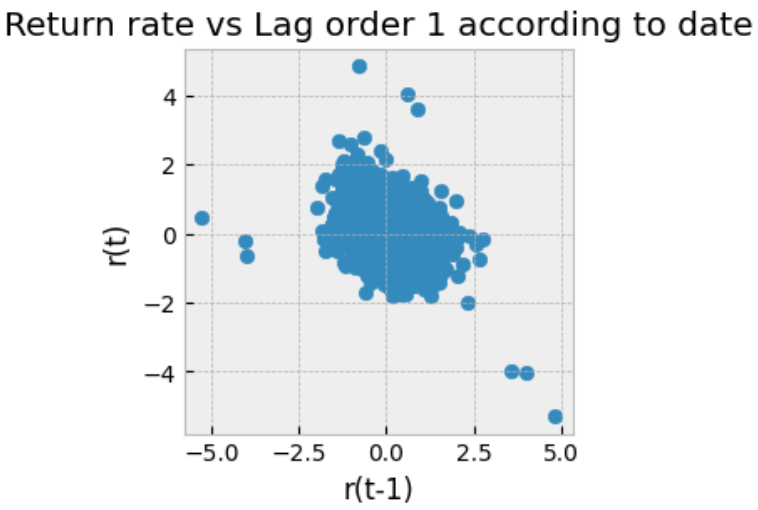
```
import matplotlib.pyplot as plt
plt.figure(figsize=(16, 4))
plt.plot(np.arange(r_t.shape[0]), r_t, '-o')
plt.axhline(y=mean, label='mean return', c='red')
12
plt.title('Return rate according to date')
plt.xlabel('Date Order')
plt.ylabel('Return Rate')
plt.legend()
plt.show()
```

Biểu đồ được thể hiện như sau:



Hình 9:Return rate according to date

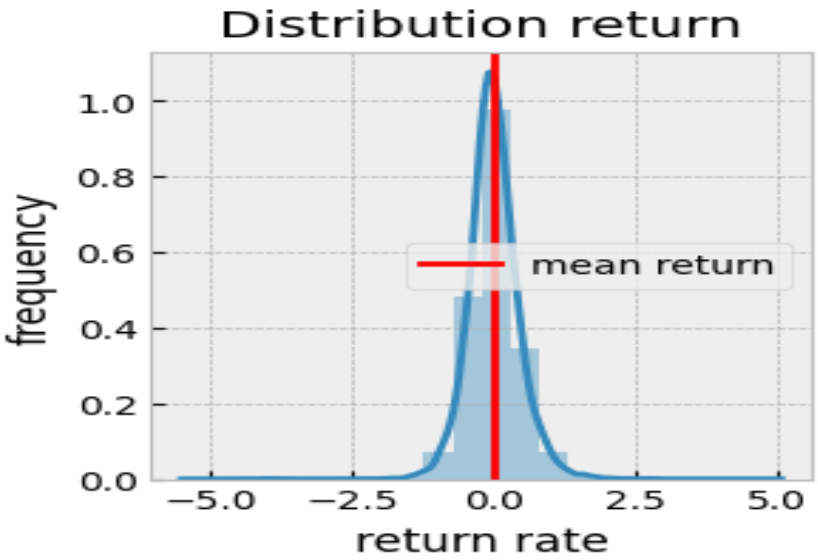
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 8))
plt.scatter(x=r_t[1:], y=r_t[:-1])
plt.title('Return rate vs Lag order 1 according to date')
plt.xlabel('r(t-1)')
plt.ylabel('r(t)')
plt.show()
```



Hình 10: Return rate vs Lag order 1 according to date

Biểu đồ Distribution return.

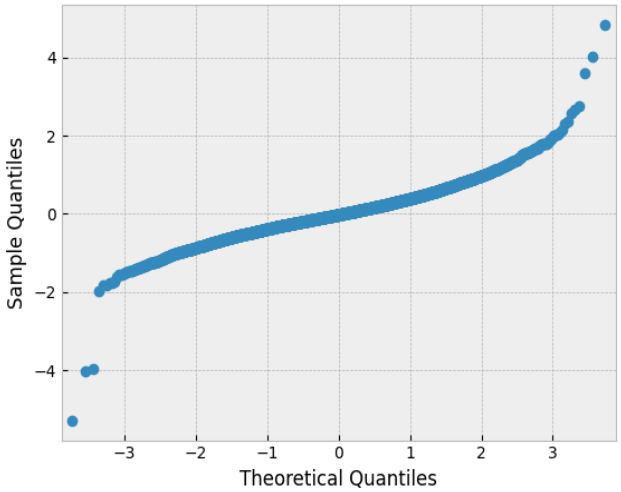
```
import seaborn as sns
plt.figure(figsize = (3, 3))
sns.distplot(r_t, bins = 20)
plt.axvline(x=mean, label='mean return', c='red')
plt.title('Distribution return')
plt.legend()
plt.xlabel('return rate')
plt.ylabel('frequency')
```



Hình 11:Distribution return

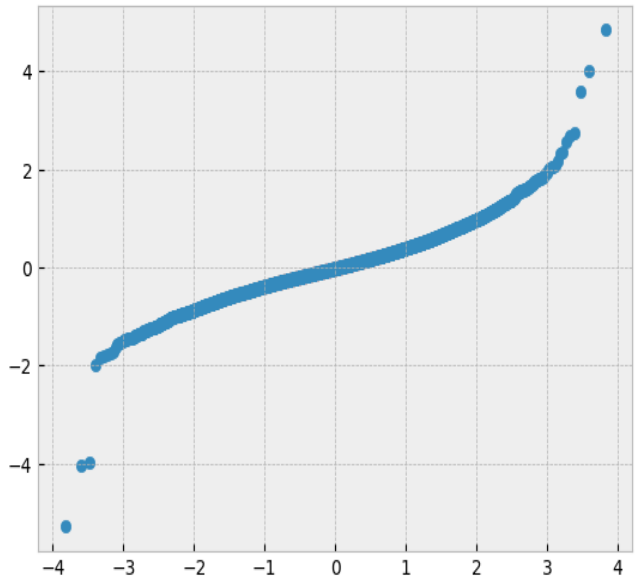
Biểu đồ qqplot.

```
# vẽ biểu đồ qqplot
import statsmodels.api as sm
sm.qqplot(r_t)
plt.show()
```



Hình 12:Theoretical Quantiles

```
from scipy import stats
tq = stats.probplot(r_t)
plt.scatter(x=tq[0][0], y = tq[0][1])
plt.show()
```



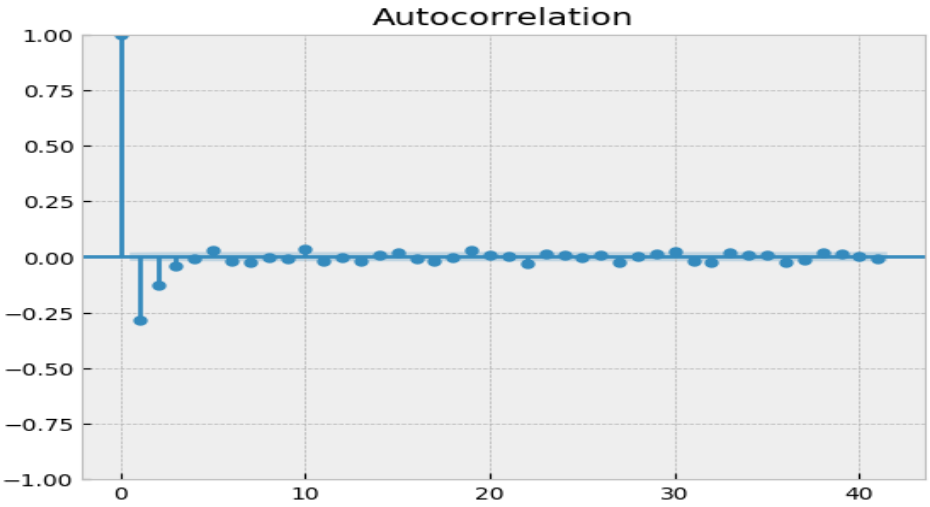
Hình 13:Theoretical Quantiles(tt)

Kiểm định cho chuỗi lợi suất .

```
#kiểm định ADF cho chuỗi lợi suất.
from statsmodels.tsa.stattools import adfuller
result = adfuller(r_t)
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))
#In ra kết quả :
ADF Statistic: -24.669973
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567
```

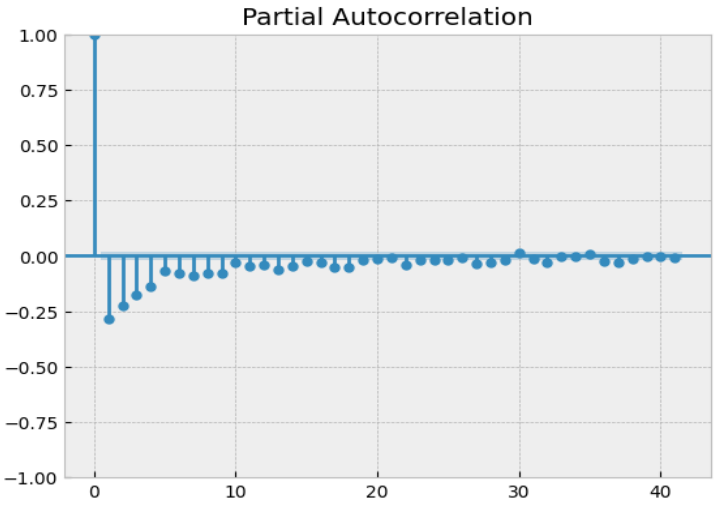
Biểu đồ hệ số tự tương quan ACF.

```
#vẽ biểu đồ các hệ số tự tương quan ACF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

import matplotlib.pyplot as plt
plt.figure(figsize = (8, 6))
ax1 = plot_acf(r_t)
```



Hình 14:Hệ số tự tương quan ACF

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
ax2 = plot_pacf(r_t)
import matplotlib.pyplot as plt
plt.figure(figsize = (8,6))
plt.show()
```



Hình 15:hệ số tự tương quan ACF(tt)

3.1 SARIMAX từ ARIMA.

```
from statsmodels.tsa.arima_model import ARIMA
import warnings
warnings.filterwarnings("ignore")
from statsmodels.tsa.arima.model import ARIMA

model_arima = ARIMA(r_t, order = (2, 0, 2))
model_fit = model_arima.fit()
print(model_fit.summary())
                               SARIMAX Results
==============================================================================
======
Dep. Variable:            KhoiLuongGD   No. Observations:
10559
Model:                 ARIMA(2, 0, 2)   Log Likelihood                -
5277.864
Date:                Tue, 02 May 2023   AIC
10567.729
Time:                        08:48:15   BIC
10611.317
Sample:                             0   HQIC
10582.443
                                - 10559
Covariance Type:                  opg
==============================================================================
======
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const       1.899e-05      0.001      0.036      0.971      -0.001
0.001
ar.L1         -0.3099      0.725     -0.428      0.669      -1.730
1.110
ar.L2          0.3665      0.333      1.099      0.272      -0.287
1.020
ma.L1         -0.1522      0.725     -0.210      0.834      -1.573
1.269
ma.L2         -0.7213      0.672     -1.073      0.283      -2.039
0.597
sigma2         0.1591      0.001    144.907      0.000       0.157
0.161
==============================================================================
===========
Ljung-Box (L1) (Q):                   1.78   Jarque-Bera (JB):
18824.37
Prob(Q):                              0.18   Prob(JB):
0.00
Heteroskedasticity (H):               0.34   Skew:
0.25
Prob(H) (two-sided):                  0.00   Kurtosis:
9.52
```

```
from statsmodels.tsa.arima_model import ARIMA
import warnings
warnings.filterwarnings("ignore")
from statsmodels.tsa.arima.model import ARIMA
def _arima_fit(orders, data):
  models = dict()
  for order in orders:
    model = ARIMA(data, order = order).fit()
    model_name = 'ARIMA({},{},{})'.format(order[0], order[1], order[2])
    print('{} --> AIC={}; BIC={}'.format(model_name, model.aic, model.bic))
    models[model_name] = model
  return models

orders = [(2, 0, 2), (2, 0, 0), (5, 0, 0), (0, 0, 5)]
models = _arima_fit(orders, r_t)
```

#In ra kết quả

```
ARIMA(2,0,2) --> AIC=10567.7288351746; BIC=10611.317238309439
ARIMA(2,0,0) --> AIC=11635.19955986652; BIC=11664.258495289745
ARIMA(5,0,0) --> AIC=11055.45986365329; BIC=11106.313000643935
ARIMA(0,0,5) --> AIC=10591.08289026568; BIC=10641.936027256324
```

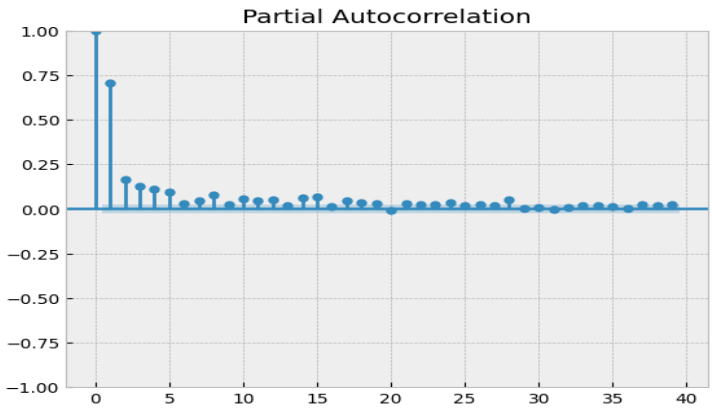
3.2 Xây dựng hồi quy theo AUTO ARIMA.

```
#Xây dựng phương trình hồi qui theo phương pháp Auto ARIMA
from pmdarima import auto_arima
model = auto_arima(r_t, start_p=0, start_q=0,
                           max_p=5, max_q=5, m=12,
                           start_P=0, seasonal=False,
                           d=0, D=0, trace=True,
                           error_action='ignore',
                           suppress_warnings=True,
                           stepwise=True)

print(model.aic())
#in ra kết quả :
Best model:  ARIMA(5,0,1)(0,0,0)[0]
Total fit time: 95.535 seconds
10465.819120831376
```

Vậy mô hình tốt nhất là ARIMA(5,0,1).

Vẽ Parial Auto corelation.

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



Hình 16:Parial Auto corelation.

Xây dựng phương trình hồi qui theo phương pháp Auto ARIMA.

```
from pmdarima.arima import auto_arima
model_sarima = auto_arima(train, start_p=0, start_q=0,
                          max_p=5, max_q=5, m=5,
                          start_P=0, seasonal=True,
                          d=1, D=1, trace=True,
                          error_action='ignore',
                          suppress_warnings=True,
                          stepwise=True)

print(model_sarima.aic())
#In ra kết quả
Best model:  ARIMA(4,1,0)(2,1,0)[5]
Total fit time: 320.512 seconds
323992.81950704346
```

Vậy tốt nhất với câu lệnh trên là ARIMA(4,1,0)(2,1,0)[5].

4.LTSM.

Import các thư viện LTSM cần thiết.

```
#LSTM Network for Regression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

Tiếp theo thì chúng ta sửa dữ liệu cho nó ngẫu nhiên và bình thường hóa lại tập dữ liệu sau đó tiến hành chia tập dữ liệu thành 80% train và 20% test để tiến hành cho máy học

```
# fix random seed for reproducibility
tf.random.set_seed(7)

# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
df = scaler.fit_transform(df)

# split into train and test sets
train_size = int(len(df) * 0.8)
test_size = len(df) - train_size
train, test = df[0:train_size,:], df[train_size:len(df),:]
print(len(train), len(test))
```

Sau đó tiến hành chuyển đổi một mảng giá trị thành ma trận tập dữ liệu và định hình lại dữ liệu X=t và Y=t+1 và định hình lại đầu vào thành [mẫu, bước thời gian, tính năng].

```
# reshape into X=t and Y=t+1
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Xây dựng matrix dữ liệu .

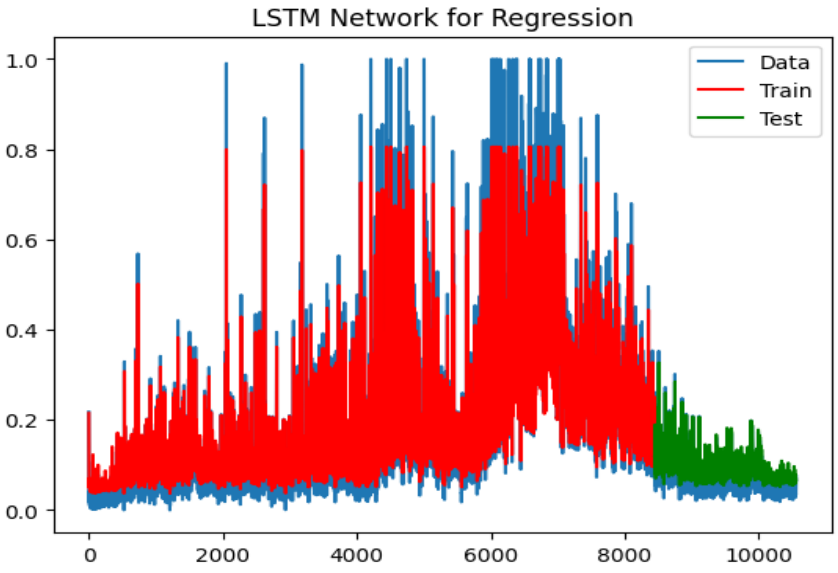
```
# Convert an array of values into a dataset matrix
def create_dataset(df, look_back=1):
  dataX, dataY = [], []
  for i in range(len(df)-look_back-1):
    a = df[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(df[i + look_back, 0])
  return np.array(dataX), np.array(dataY)
```

Tiến hành tạo các thuộc tính phù hợp của mô hình mạng LTSM sau đó thì tiến hành tạo các thuộc tính dự đoán và đảo ngược chúng và áp dụng tính sai số bình phương trung bình gốc.

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=5, batch_size=32, verbose=2)
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = np.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

Sau đó dự đoán của tập train và vẽ biểu đồ thay đổi của tập test và tập train và tiến hành dự đoán của mạng LTSM.

```
# shift train predictions for plotting
trainPredictPlot = np.empty_like(df)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
# shift test predictions for plotting
testPredictPlot = np.empty_like(df)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(df)-
1, :] = testPredict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(df),label = 'Data')
plt.plot(trainPredictPlot,label = 'Train',color = 'r')
plt.plot(testPredictPlot,label = 'Test',color = 'green')
plt.title('LSTM Network for Regression')
plt.legend()
plt.show()
#Kết quả thực nghiệm.

264/264 - 1s - loss: 0.0087 - 510ms/epoch - 2ms/step
Epoch 5/5
264/264 - 1s - loss: 0.0086 - 608ms/epoch - 2ms/step
264/264 [==============================] - 1s 2ms/step
66/66 [==============================] - 0s 3ms/step
Train Score: 0.09 RMSE
Test Score: 0.03 RMSE
```



Hình 17:LSTM Network for Regression

Thêm "valid" được sử dụng khi áp dụng phép tích chập (convolution) trên đầu vào (input) hoặc trạng thái ẩn trước đó (hidden state) trong quá trình huấn luyện.

Import các thư viện LTSM cần thiết.

```
#LSTM Network for Regression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

Tiếp theo thì chúng ta sửa dữ liệu cho nó ngẫu nhiên và bình thường hóa lại tập dữ liệu sau đó tiến hành chia tập dữ liệu thành 80% train và 20% test để tiến hành cho máy học

```
# Fix random seed for reproducibility
tf.random.set_seed(7)
# Normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
df = scaler.fit_transform(df)

# Split into train, validation and test datasets
train_size = int(len(df) * 0.6)
val_size = int(len(df) * 0.2)
test_size = len(df) - train_size - val_size
train, val, test = df[0:train_size,:], df[train_size:train_size+val_size,:],
df[train_size+val_size:len(df),:]
```

Xây dựng matrix dữ liệu.

```
# Convert an array of values into a dataset matrix
def create_dataset(df, look_back=1):
  dataX, dataY = [], []
  for i in range(len(df)-look_back-1):
    a = df[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(df[i + look_back, 0])
  return np.array(dataX), np.array(dataY)
```

Sau đó tiến hành chuyển đổi một mảng giá trị thành ma trận tập dữ liệu và định hình lại dữ liệu X=t và Y=t+1 và định hình lại đầu vào thành [mẫu, bước thời gian, tính năng].

```
# Reshape into X=t and Y=t+1
look_back = 1
trainX, trainY = create_dataset(train, look_back)
valX, valY = create_dataset(val, look_back)
testX, testY = create_dataset(test, look_back)
# Reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
valX = np.reshape(valX, (valX.shape[0], 1, valX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Tiến hành tạo các thuộc tính phù hợp của mô hình mạng LTSM sau đó thì tiến hành tạo các thuộc tính dự đoán và đảo ngược chúng và áp dụng tính sai số bình phương trung bình gốc.

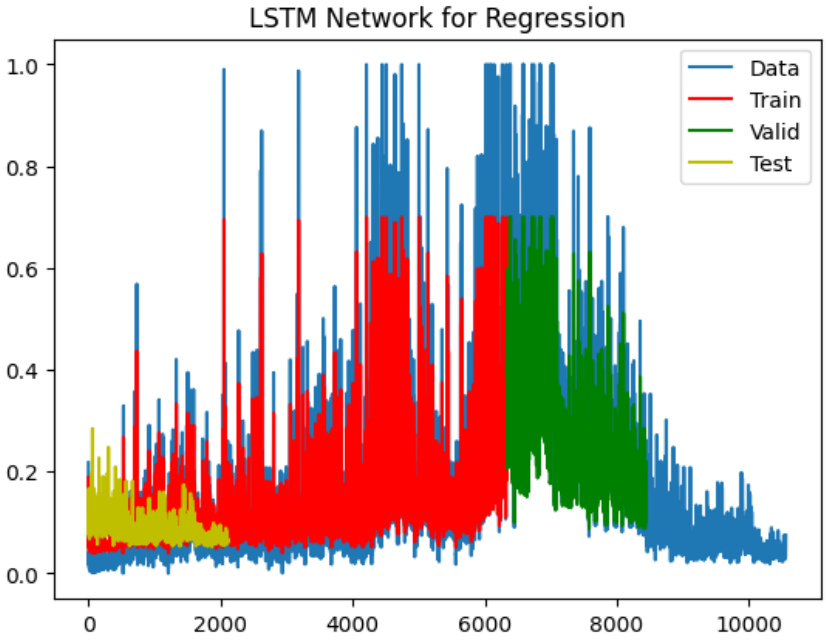
```
# Create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(trainX, trainY, epochs=5, batch_size=32, verbose=2, val
idation_data=(valX, valY))

# Make predictions
trainPredict = model.predict(trainX)
valPredict = model.predict(valX)
testPredict = model.predict(testX)

# Invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
valPredict = scaler.inverse_transform(valPredict)
valY = scaler.inverse_transform([valY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])

# Calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
valScore = np.sqrt(mean_squared_error(valY[0], valPredict[:,0]))
print('Validation Score: %.2f RMSE' % (valScore))
testScore = np.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

Sau đó dự đoán của tập train và vẽ biểu đồ thay đổi của tập test và tập train, valid và tiến hành dự đoán của mạng LTSM.

```
# Shift train predictions for plotting
trainPredictPlot = np.empty_like(df)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict

# Shift validation predictions for plotting
valPredictPlot = np.empty_like(df)
valPredictPlot[:, :] = np.nan
valPredictPlot[len(trainPredict)+(look_back*2)+1:len(trainPredict)+len(valPr
edict)+(look_back*2)+1, :] = valPredict

# Shift test predictions for plotting
testPredictPlot = np.empty_like(df)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPredict)+len(valPredict)+(look_back*2)+1:len(df)-
1, :]
testPredictPlot = testPredict
# Plot baseline and predictions
plt.plot(scaler.inverse_transform(df),label = 'Data')
plt.plot(trainPredictPlot,label = 'Train',color = 'red')
plt.plot(valPredictPlot,label = 'Valid',color = 'green')
plt.plot(testPredictPlot,label = 'Test',color = 'y')
plt.title('LSTM Network for Regression')
plt.legend()
plt.show()
#in ra sai số và kết quả
Epoch 5/5
198/198 - 1s - loss: 0.0074 - val_loss: 0.0150 - 898ms/epoch - 5ms/step
198/198 [==============================] - 1s 2ms/step
66/66 [==============================] - 0s 1ms/step
66/66 [==============================] - 0s 1ms/step
Train Score: 0.09 RMSE
Validation Score: 0.12 RMSE
Test Score: 0.03 RMSE
```

Với mô hình LTSM với mức train_size = int(len(df) * 0.6) và val_size = int(len(df) * 0.2) test_size = len(df) - train_size - val_size cho ra kết quả rất là đẹp với chỉ số loss: 0.0074 - val_loss: 0.0150.

Train Score: 0.09 RMSE Validation Score: 0.12 RMSE Test Score: 0.03 RMSE.



Hình 18:LSTM Network for Regression(tt)

CHƯƠNG 4 : TỔNG KẾT

1.Kết quả đạt được.

Dự báo cho ta thấy được giá trị xấp xỉ với giá trị thực tế về giá cả lợi nhuận với mô hình trên điều này chứng to độ chính xác và tin cậy về Mô Hình chuỗi ARIMA khá là cao. Nhưng trong điểm mạnh thì dự báo của Mô Hình ARIMA cũng cho kết quả dự đoán và tham khảo khi dự liệu hoặc các thay đổi thì sẽ ảnh hưởng đến kết quả nên ta cần cập nhập thông tin kịp thời. có thể nói Mô Hình ARIMA rất tốt và rất tối ưu trong việc dự báo ngắn hạn.

2.Hạn chế và hướng phát triển .

Song theo những mặt tích cực thì trong phần code thì cũng gặp rất nhiều lỗi và sự cố phát sinh trong quá trình xây dựng thuật toán cũng như chưa lưu ý chi tiết những phần code thật sự dễ đọc và dễ hiểu, nhưng song theo đó thì chúng em sẽ cố gắng phát triển hơn , chi tiết hơn trong quá trình xây dựng thuật toán cũng như ứng dụng để đáp ứng được yêu cầu cũng như là nâng cao trình độ để sau này có thể thực chiến tại các công ty một cách thật chuyên nghiệp.

TÀI LIỆU THAM KHẢO

[1] Ths.Đặng Như Phú(2023),Phân tích chuỗi thời gian và ứng dụng Khoa Công nghệ thông tin trường ĐH Nguyễn Tất Thành : Phân Tích Chuỗi Thời Gian Và Ứng Dụng

[2] Nguyễn Thanh Tuấn(2009), Deeplearning cơ bản The Legrand Orange Book Template by Mathias Legrand is used : https://nttuan8.com/sach-deep-learning-co-ban/

[3] Ths.Hồ Khôi(2022),Lecturer in deep learning at the Faculty of Information Technology, Nguyen Tat Thanh University.

[4] Từ DeepLearning.AI (10/5/2023) : Start or Advance Your Career in AI.

[5] Đắm chìm vào học sâu (10/5/2023) :https://d2l.aivivn.com/

[6]Tham khảo sửa chữa bài (10/5/2023) : https://stackoverflow.com/

[7]File csv Apple revenue from 1980 to 2022 | Apple revenue from 1980 to 2022 | Kaggle

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770–778).

[9] Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

[10] Phạm Đình Khanh(2023) : Khoa học dữ liệu (phamdinhkhanh.github.io).