

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN KỸ THUẬT DỮ LIỆU



LÝ QUỐC DŨNG - 19133015

BÙI THỊ NGÂN TUYÊN - 19133066

Đề Tài:

**DỰ BÁO CHUỖI THỜI GIAN BẰNG MÔ HÌNH LAI GHÉP
FEEDFORWARD NEURAL NETWORK VÀ K-NEAREST NEIGHBORS**

LUẬN VĂN TỐT NGHIỆP

GIÁO VIÊN HƯỚNG DẪN

TS. NGUYỄN THÀNH SƠN

KHÓA 2019 – 2023

TP. Hồ Chí Minh, tháng 07 năm 2023

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN KỸ THUẬT DỮ LIỆU



LÝ QUỐC DŨNG - 19133015

BÙI THỊ NGÂN TUYỀN - 19133066

Đề Tài:

**DỰ BÁO CHUỖI THỜI GIAN BẰNG MÔ HÌNH LAI GHÉP
FEEDFORWARD NEURAL NETWORK VÀ K-NEAREST NEIGHBORS**

LUẬN VĂN TỐT NGHIỆP

GIÁO VIÊN HƯỚNG DẪN

TS. NGUYỄN THÀNH SƠN

KHÓA 2019 – 2023

TP. Hồ Chí Minh, tháng 07 năm 2023

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: **Lý Quốc Dũng**

MSSV 1: **19133015**

Họ và tên Sinh viên 2: **Bùi Thị Ngân Tuyền**

MSSV 2: **19133066**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Dự báo chuỗi thời gian bằng mô hình lai ghép FFNN và KNN.**

Họ và tên giảng viên hướng dẫn: **TS. Nguyễn Thành Sơn**

NHẬN XÉT

1. Về nội dung đề tài và khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

TP. Hồ Chí Minh, tháng 07 năm 2023
Giảng viên hướng dẫn
(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: **Lý Quốc Dũng**

MSSV 1: **19133015**

Họ và tên Sinh viên 2: **Bùi Thị Ngân Tuyền**

MSSV 2: **19133066**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Dự báo chuỗi thời gian bằng mô hình lai ghép FFNN và KNN.**

Họ và tên giảng viên phản biện: **ThS. Lê Thị Minh Châu**

NHẬN XÉT

1. Về nội dung đề tài và khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

TP. Hồ Chí Minh, tháng 07 năm 2023
Giảng viên phản biện
(Ký & ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên nhóm xin phép được gửi lời cảm ơn sâu sắc nhất đến với Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện cho nhóm chúng em được học tập, phát triển nền tảng kiến thức thực hiện đề tài này.

Bên cạnh đó nhóm chúng em xin gửi đến thầy Nguyễn Thành Sơn lời cảm ơn chân thành nhất, nhóm chúng em rất biết ơn khi thầy đã nhận lời hướng dẫn và giúp đỡ nhóm chúng em trong suốt quá trình thực hiện luận văn tốt nghiệp của mình. Trong quá trình thực hiện thầy đã tận tâm chỉ bảo, cho nhóm những lời khuyên hữu ích nhất từ lúc bắt đầu cũng như kết thúc đề tài này.

Với kinh nghiệm chuyên môn cao cũng như với những kinh nghiệm thực tế của các thầy cô Khoa Công Nghệ Thông Tin đặc biệt là thầy Nguyễn Thành Sơn đã giúp chúng em tiếp thu thêm những kiến thức mới, kinh nghiệm về chuyên ngành để thực hiện luận văn tốt nghiệp của mình cũng như cho công việc sau này. Chúng em thật sự biết ơn những gì mà thầy cô đã chỉ bảo, đó là hành trang, động lực cho chúng em khi bước ra một môi trường mới.

Tuy nhiên với khả năng của mình, chúng em khó có thể tiếp thu hết nguồn kiến thức vô tận, cũng như với khả năng chuyên môn còn hạn chế nên khó tránh khỏi xảy ra những thiếu sót trong quá trình thực hiện đồ án này. Chúng em hi vọng nhận được sự thông cảm và góp ý của quý thầy cô, nhờ những góp ý đó chúng em có thể học được những kinh nghiệm quý báu để hoàn thành và nâng cấp luận văn tốt hơn.

Sau tất cả, chúng em xin gửi lời cảm ơn sâu sắc nhất đến với thầy Nguyễn Thành Sơn và quý thầy, cô Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh vì tất cả những điều thầy cô đã gửi gắm và chỉ dạy chúng em. Nhóm xin kính chúc các thầy cô luôn có sức khỏe thật tốt và luôn thành công trong cuộc sống.

ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP

Họ và tên sinh viên thực hiện 1: **Lý Quốc Dũng** Mã số sinh viên: **19133015**

Họ và tên sinh viên thực hiện 2: **Bùi Thị Ngân Tuyền** Mã số sinh viên: **19133066**

Thời gian làm luận văn: Từ ngày 01/02/2023 đến ngày 13/05/2023

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Dự báo chuỗi thời gian bằng mô hình lai ghép FFNN và KNN.**

Giáo viên hướng dẫn: **TS. Nguyễn Thành Sơn**

Nhiệm vụ của luận văn:

Lý thuyết:

- Nghiên cứu tối ưu mô hình KNN cho bài toán dự báo chuỗi thời gian.
- Nghiên cứu tối ưu hóa mô hình FFNN cho bài toán dự báo chuỗi thời gian.
- Nghiên cứu cải tiến kết hợp hai mô hình lai ghép FFNN và KNN cho bài toán dự báo chuỗi thời gian.

Thực hành:

- Sử dụng python để phục vụ cho đề tài dự báo chuỗi thời gian bằng mô hình lai ghép FFNN và KNN.
- Tiền xử lý dữ liệu:
 - MinMaxScaler: lớp dùng để chuẩn hóa dữ liệu trong khoảng giá trị cụ thể.
 - Thay thế các giá trị Nan, Null, Empty, Outlier.
- Sử dụng các thư viện cho mô hình KNN:
 - sklearn.neighbors.KNeighborsRegressor: thư viện của KNN trong sklearn.
 - dtw.dtw: thư viện Dynamic Time Warping (DTW) trong python dùng để tính khoảng cách giữa hai chuỗi thời gian. DTW được sử dụng rộng rãi trong phân tích chuỗi thời gian và các bài toán liên quan đến phát hiện, phân loại, tìm kiếm và so khớp các chuỗi thời gian.

- Sử dụng các thư viện cho mô hình FFNN:
 - `keras.models.Sequential` và `keras.layers.Dense`: sử dụng để tạo một mô hình mạng nơ-ron tuần tự.
 - `keras.callbacks.EarlyStopping`: lớp được sử dụng để dừng huấn luyện mô hình nếu sự khác biệt giữa giá trị mất mát trên tập huấn luyện và tập xác thực không cải thiện sau một số lần lặp.
 - `sklearn.model_selection.GridSearchCV`: lớp được sử dụng để tìm kiếm lưới các tham số tốt nhất cho mô hình.
 - `keras.wrappers.scikit_learn.KerasRegressor`: lớp được sử dụng để tạo một mô hình mạng nơ-ron sử dụng cho huấn luyện mạng nơ-ron dựa trên Scikit-Learn.

KẾ HOẠCH THỰC HIỆN

STT	Thời gian	Công việc	Ghi chú
1	01/02/2023- 08/02/2023	Nghiên cứu và tìm hiểu mô hình k-nearest neighbors, Feed Forward Neural Network, Viết đề cương đề tài	
2	08/02/2023- 15/02/2023	Thiết kế mô hình k-nearest neighbors và tiến hành dự đoán tìm ra tham số	
3	15/02/2023- 22/02/2023	Thiết kế Feed Forward Neural Network và tiến hành dự đoán tìm ra tham số tốt nhất	
4	22/02/2023- 08/03/2023	Tìm hiểu, nghiên cứu và xây dựng mô hình lai ghép tuần tự hai mô hình k-nearest neighbors, Feed Forward Neural Network	
5	8/02/2023- 20/03/2023	Tìm hiểu, nghiên cứu và xây dựng mô hình lai ghép song song hai mô hình k-nearest neighbors, Feed Forward Neural Network	
6	20/03/2023- 15/04/2023	Thực hiện trên nhiều tập dữ liệu khác nhau	
7	16/04/2023- 06/05/2023	Điều chỉnh và đánh giá kết quả của mô hình lai ghép	
8	06/05/2023- 13/05/2023	Hoàn thành luận văn	

Ngày 13 tháng 05 năm 2023

Người viết đề cương

Lý Quốc Dũng

Bùi Thị Ngân Tuyền

Ý kiến của giáo viên hướng dẫn

(ký và ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN.....	5
ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP	6
KẾ HOẠCH THỰC HIỆN.....	8
MỤC LỤC	9
DANH SÁCH HÌNH.....	13
DANH SÁCH BẢNG.....	16
DANH SÁCH TỪ VIẾT TẮT	17
PHẦN MỞ ĐẦU	18
1. TÍNH CẤP THIẾT CỦA ĐỀ TÀI.....	18
2. MỤC TIÊU ĐỀ TÀI.....	18
3. PHƯƠNG PHÁP THỰC HIỆN	18
4. PHẠM VI THỰC HIỆN.....	18
PHẦN NỘI DUNG.....	19
CHƯƠNG 1: PHẦN NỘI DUNG TỔNG QUAN CHUỖI THỜI GIAN VÀ DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN.....	19
1.1. CHUỖI THỜI GIAN	19
1.1.1. Khái niệm.....	19
1.1.2. Đặc điểm của chuỗi thời gian	20
1.2. ĐỘ ĐO TRONG BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN	23
1.2.1. Độ đo Euclidean.....	23
1.2.2. Độ đo Dynamic Time Warping	24
1.2.3. Tầm quan trọng của dự báo và chuỗi thời gian	25
1.3. BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN.....	26

1.3.1.	Tiền xử lý dữ liệu.....	26
1.3.2.	Xây dựng mô hình	27
1.3.3.	Đánh giá mô hình.....	27
1.3.4.	Dự đoán.....	29
1.3.5.	Đánh giá kết quả	29
1.4.	NGHIÊN CỨU THỰC HIỆN TRAINING VÀ TESTING CHO MÔ HÌNH	30
1.4.1.	Quá trình training.....	30
1.4.2.	Quá trình testing.....	30
1.5.	PHÂN BIỆT OVERFITTING VÀ UNDERFITTING	30
1.5.1.	Overfitting.....	30
1.5.2.	Underfitting.....	31
1.5.3.	Cách khắc phục Overfitting và Underfitting	32
1.6.	CÁC MÔ HÌNH DỰ BÁO CHUỖI THỜI GIAN	32
1.6.1.	Phương pháp học sâu	33
1.6.2.	Phương pháp tuyến tính.....	35
1.6.3.	Phương pháp phi tuyến tính.....	38
CHƯƠNG 2: ỨNG DỤNG MÔ HÌNH KNN VÀ FFNN VÀO BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN		40
2.1.	FEED FORWARD NEURAL NETWORK (FFNN)	40
2.1.1.	Định nghĩa.....	40
2.1.2.	Nguyên lý hoạt động.....	41
2.2.	K-NEAREST NEIGHBORS (KNN)	42
2.2.1.	Mô hình KNN	42
2.2.2.	Hoạt động.....	43
2.2.3.	Ứng dụng mô hình KNN	44

CHƯƠNG 3: MÔ HÌNH KẾT HỢP K-NEAREST NEIGHBORS VÀ FEEDFORWARD NEURAL NETWORK.....	45
3.1. MÔ HÌNH LAI GHÉP GIỮA KNN VÀ FFNN CHO BÀI TOÁN DỰ BÁO TRÊN CHUỖI THỜI GIAN	45
3.2. NGHIÊN CỨU MÔ HÌNH LAI GHÉP BẰNG CÁCH THỰC HIỆN SONG SONG HAI MÔ HÌNH KNN VÀ FFNN	46
3.3. TUẦN TỰ THEO MÔ HÌNH CỘNG GIỮA KNN VÀ FFNN	47
CHƯƠNG 4: CÀI ĐẶT.....	49
4.1. CÀI ĐẶT PHẦN MỀM.....	49
4.1.1. Cài đặt Python.....	49
4.1.2. Cài đặt Anaconda.....	50
4.2. THƯ VIỆN.....	52
4.2.1. Loại bỏ Warning	52
4.2.2. Đọc dữ liệu.....	52
4.2.3. Tiền xử lý dữ liệu.....	53
4.2.4. Trích xuất chuỗi con	54
4.2.5. Mô hình K-nearest Neighbors	55
4.2.6. Mô hình FeedForward Neural Network	57
4.2.7. Mô hình lai ghép song song.....	59
4.2.8. Mô hình lai ghép tuần tự cộng.....	60
4.2.9. Đánh giá độ chính xác	60
4.2.10. Thời gian thực thi.....	62
CHƯƠNG 5: ĐÁNH GIÁ BẰNG THỰC NGHIỆM PHƯƠNG PHÁP DỰ BÁO CHUỖI THỜI GIAN KNN - FFNN.....	63
5.1. MÔI TRƯỜNG THỰC NGHIỆM.....	63
5.1.1. Phần cứng	63

5.1.2.	Phần mềm	63
5.2.	DỮ LIỆU THỰC NGHIỆM	63
5.3.	TIÊU CHÍ ĐÁNH GIÁ	66
5.3.1.	Đánh giá độ chính xác mô hình	67
5.3.2.	Đánh giá về thời gian thực thi.....	69
5.4.	CÁC TRƯỜNG HỢP THỰC NGHIỆM	70
5.4.1.	Mô hình FFNN.....	70
5.4.2.	Mô hình KNN	71
5.4.3.	So sánh mô hình.....	71
5.5.	KẾT QUẢ THỰC NGHIỆM	71
5.5.1.	Mô hình FFNN.....	71
5.5.2.	Mô hình KNN	78
5.5.3.	So sánh mô hình.....	83
PHẦN KẾT LUẬN		86
DANH SÁCH TÀI LIỆU THAM KHẢO.....		88

DANH SÁCH HÌNH

Hình 1.1 Hình ảnh minh họa chuỗi thời gian ngẫu nhiên	19
Hình 1.2 Phương pháp cửa sổ trượt cho dữ liệu đường huyết theo chuỗi thời gian	20
Hình 1.3 Hình minh họa thành phần Trend	20
Hình 1.4 Hình minh họa thành phần Seasonal	21
Hình 1.5 Hình ảnh minh họa thành phần Cycle	21
Hình 1.6 Hình ảnh minh họa thành phần Noise	22
Hình 1.7 Hình ảnh minh họa độ đo Euclide trong tọa độ Descartes	23
Hình 1.8 Hình ảnh minh họa độ đo Euclide trong chuỗi thời gian	24
Hình 1.9 Hình ảnh minh họa độ đo Dynamic Time Warping trong chuỗi thời gian	24
Hình 1.10 Hình minh họa Overfitting	30
Hình 1.11 Hình minh họa Underfitting	31
Hình 1.12 Mô hình Recurrent Neural Network.....	33
Hình 1.13 Mô hình Long Short-Term Memory	34
Hình 1.14 Mô hình Long Feedforward Neural Network.....	35
Hình 1.15 Mô hình Linear Regression	36
Hình 1.16 Mô hình Support Vector Machines	36
Hình 1.17 Mô hình K-Nearest Neighbors	37
Hình 1.18 Mô hình Decision Tree	38
Hình 2.1 Hình minh họa mô hình FFNN với nhiều lớp ẩn	40
Hình 2.2 Hình minh họa mô hình FFNN với một lớp ẩn	41
Hình 2.3 Mô hình thuật toán KNN	43
Hình 2.4 Hình minh họa về cách tiếp cận dựa trên KNN.....	44
Hình 3.1 Sơ đồ KNN cho bài toán dự báo chuỗi thời gian	45
Hình 3.2 Sơ đồ mô hình FFNN cho bài toán dự báo chuỗi thời gian.....	45
Hình 3.3 Mô hình dự đoán Hybrid	46
Hình 3.4 Sơ đồ mô hình lai ghép tuần tự KNN-FFNN	48
Hình 4.1 Trang download Python	49
Hình 4.2 Trang chọn bộ cài đặt	49
Hình 4.3 Màn hình cài đặt Python.....	49

Hình 4.4 Màn hình kiểm tra phiên bản Python	50
Hình 4.5 Màn hình kiểm tra phiên bản Pip trong Python.....	50
Hình 4.6 Trang tải Anaconda	51
Hình 4.7 Mở Jupyter notebook bằng cmd	51
Hình 4.8 Tạo thư mục trên Jupyter notebook.....	51
Hình 4.9 Chạy thử chương trình trên Jupyter notebook.....	52
Hình 5.1 Mô tả dữ liệu chứng khoán AGLE	64
Hình 5.2 Mô tả dữ liệu chứng khoán GDDY	65
Hình 5.3 Mô tả dữ liệu chứng khoán AVAL.....	65
Hình 5.4 Mô tả dữ liệu chứng khoán MDLY	66
Hình 5.5 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập AVAL	74
Hình 5.6 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập AGLE.....	74
Hình 5.7 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập MDLY	75
Hình 5.8 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập GDDY	75
Hình 5.9 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập AVAL	76
Hình 5.10 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập AGLE.....	77
Hình 5.11 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập MDLY.....	77
Hình 5.12 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập GDDY	77
Hình 5.13 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu AVAL	79
Hình 5.14 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu AGLE.....	80

Hình 5.15 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu MDLY	81
Hình 5.16 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu GDDY.....	82

DANH SÁCH BẢNG

Bảng 5.1 Tham số đầu vào tốt nhất đối với mô hình FFNN đối với tỷ lệ 80% train và 20% test	72
Bảng 5.2 Tham số đầu vào tốt nhất đối với mô hình FFNN đối với tỷ lệ 70% train và 30% test	72
Bảng 5.3 Tham số đầu vào tốt nhất của tập dữ liệu AVAL đối với từng giá trị lớp ẩn khác nhau	72
Bảng 5.4 Tham số đầu vào tốt nhất của tập dữ liệu AGLE đối với từng giá trị lớp ẩn khác nhau	73
Bảng 5.5 Tham số đầu vào tốt nhất của tập dữ liệu MDLY đối với từng giá trị lớp ẩn khác nhau	73
Bảng 5.6 Tham số đầu vào tốt nhất của tập dữ liệu GDDY đối với từng giá trị lớp ẩn khác nhau	73
Bảng 5.7 Trình bày tham số tốt nhất ở 4 tập dữ liệu	74
Bảng 5.8 Mô tả Số lượng lớp ẩn cùng với các tham số tốt nhất	76
Bảng 5.9 Trình bày giá trị k tốt nhất theo từng độ đo	78
Bảng 5.10 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu AVAL	83
Bảng 5.11 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu AGLE	84
Bảng 5.12 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu MDLY	84
Bảng 5.13 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu GDDY	85

DANH SÁCH TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
FFNN	Feed Forward Neural Network
KNN	K-Nearest Neighbors
DTW	Dynamic Time Warping
MAE	Mean Absolute Error
MSE	Mean squared error
RMSE	Root mean squared error
MAPE	Mean absolute percentage error
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machines
AGLE	Aeglea BioTherapeutics
GDDY	GoDaddy Inc
AVAL	Grupo Aval Acciones
MDLY	Medley Management Inc

PHẦN MỞ ĐẦU

1. TÍNH CẤP THIẾT CỦA ĐỀ TÀI

Trong thời đại hiện nay sự phát triển mạnh mẽ của công nghệ và khối lượng thông tin ngày càng đa dạng góp phần phát triển xã hội, đất nước. Nhưng đi chung với việc phát triển nhanh chóng khiến cho các hệ thống khai thác và lưu trữ thông tin không còn hữu dụng để có thể đáp ứng đủ các nhu cầu cần thiết hiện nay. Từ đó buộc các hệ thống lưu trữ và khai thác thông tin phải phát triển theo tốc độ của công nghệ để có thể đáp ứng nhu cầu sử dụng dữ liệu, từ việc khai phá các mối liên hệ và các mối tương quan giữa chúng để phát hiện xu hướng, dự báo các hành vi tiếp theo và các tính năng khác mà dữ liệu có thể mang lại.

Khai phá dữ liệu có ý nghĩa lớn đối với công nghệ hiện nay là các bài toán dự báo, đặc biệt bài toán dự báo chuỗi thời gian đang giữ vai không nhỏ trong việc đóng góp vào sự tồn tại và phát triển của hầu hết các lĩnh vực của xã hội. Dự báo chuỗi thời gian giúp phát hiện ra các xu hướng cũ, dự báo xu hướng trong thời gian tiếp theo từ đó giúp các nhà phân tích có thể đưa ra các quyết định đúng đắn trong tương lai.

2. MỤC TIÊU ĐỀ TÀI

Nghiên cứu phương pháp lai ghép mô hình FFNN và KNN vào bài toán dự báo chuỗi thời gian để nâng cao hiệu quả của bài toán dự báo.

3. PHƯƠNG PHÁP THỰC HIỆN

Tìm hiểu các kiến thức cần thiết để thực hiện đề tài:

- Nghiên cứu lý thuyết liên quan đến chuỗi thời gian và các mô hình kết hợp (FFNN và KNN).
- Xây dựng mô hình lai ghép FFNN và KNN.
- Xem xét và đánh giá mô hình bằng thực nghiệm.

4. PHẠM VI THỰC HIỆN

- Đối tượng nghiên cứu của đề tài là mô hình dự báo chuỗi thời gian sử dụng phương pháp cửa sổ trượt nhằm hỗ trợ thuật toán lai ghép FFNN và KNN.
- Phạm vi nghiên cứu: dự báo trên chuỗi thời gian bằng mô hình lai ghép KNN và FFNN.

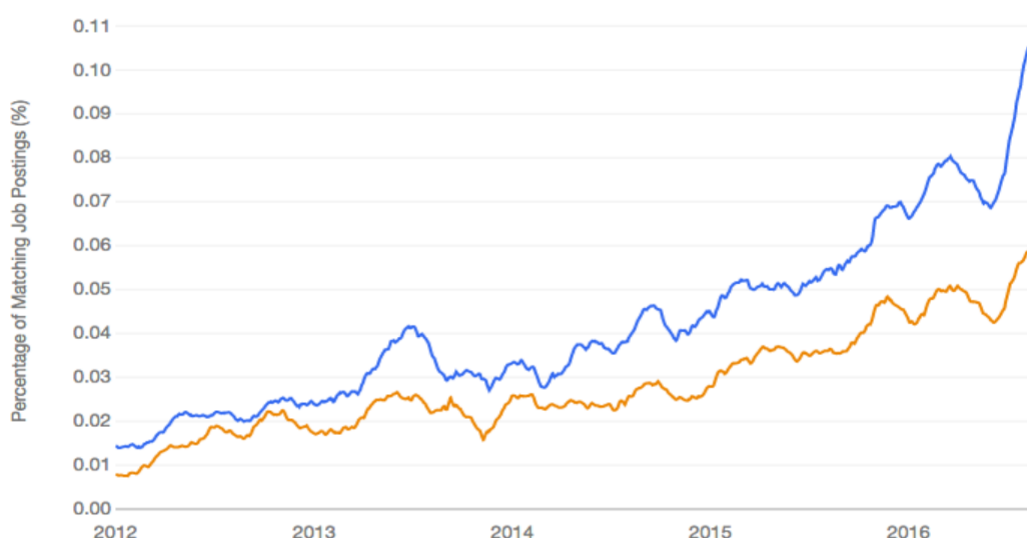
PHẦN NỘI DUNG

CHƯƠNG 1: PHẦN NỘI DUNG TỔNG QUAN CHUỖI THỜI GIAN VÀ DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN

1.1. CHUỖI THỜI GIAN

1.1.1. Khái niệm

Chuỗi thời gian (time series) là một chuỗi các điểm dữ liệu xảy ra theo thứ tự thời gian, một chuỗi thời gian sẽ theo dõi chuyển động của các điểm dữ liệu đã chọn như: sóng điện trong não, đo nhiệt độ, lượng mưa, dự báo giá cổ phiếu...[\[4\]](#)

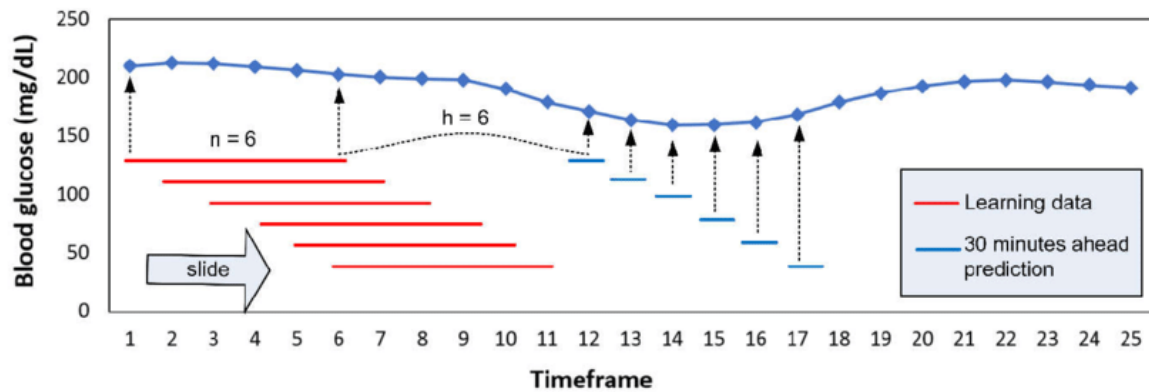


Hình 1.1 Hình ảnh minh họa chuỗi thời gian ngẫu nhiên

Dữ liệu chuỗi thời gian (time series data) là dữ liệu được đánh dấu thời gian, một chuỗi các điểm dữ liệu được lập chỉ mục theo thứ tự thời gian. Các điểm dữ liệu này thường bao gồm các phép đo liên tiếp được thực hiện từ cùng một nguồn trong một khoảng thời gian cố định và được sử dụng để theo dõi sự thay đổi theo thời gian [\[4\]](#).

Cửa sổ trượt là một kỹ thuật tính toán giúp tạo các chuỗi con liên tiếp trong chuỗi thời gian, nhằm mục đích giảm việc dùng vòng lặp lồng nhau và thay thế bằng một vòng lặp đơn các bước thời gian trước để giảm độ phức tạp thời gian. Kỹ thuật này sử dụng một cửa sổ có kích thước cố định di chuyển theo dữ liệu chuỗi thời gian. Cửa sổ này có thể chứa một số lượng giá trị dữ liệu được xác định trước hoặc có thể thay đổi để phù

hợp với từng mục đích sử dụng. Cửa sổ trượt là cơ sở để biến bất kỳ tập dữ liệu nào thành một bài toán học có giám sát [6].



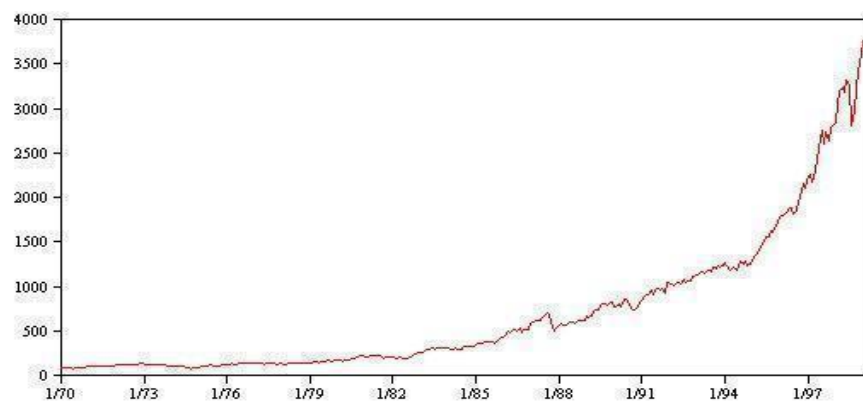
Hình 1.2 Phương pháp cửa sổ trượt cho dữ liệu đường huyết theo chuỗi thời gian

1.1.2. Đặc điểm của chuỗi thời gian

Để có thể phân tích và đưa ra các dự báo tốt từ việc nghiên cứu các hành vi trong quá khứ của một chuỗi thời gian cũng như xem xét và đánh giá các biến động bằng biểu đồ để xác định dữ liệu hiện tại thuộc những thành phần nào của chuỗi thời gian từ đó tìm ra được phương pháp thích hợp cho bài toán.

- Trend

Trend (xu hướng) là sự thay đổi dần dần lên hoặc xuống của chuỗi có xu hướng tăng hoặc giảm giá trị theo thời gian [6].

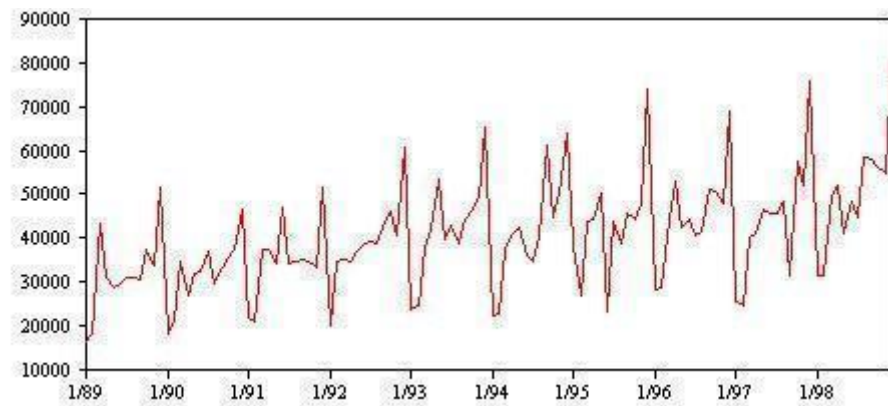


Hình 1.3 Hình minh họa thành phần Trend

Xu hướng có 2 loại là tuyến tính và phi tuyến tính. Xu hướng tuyến tính là xu hướng tăng hoặc giảm của dữ liệu được mô tả bằng đường thẳng, ngược lại xu hướng phi tuyến tính được thể hiện ở dạng đường cong [6].

- Seasonality

Seasonality (tính thời vụ) là một đặc điểm của chuỗi thời gian mà dữ liệu trải qua các thay đổi có tính lặp lại thường xuyên và có thể dự đoán được theo hàng năm [6].

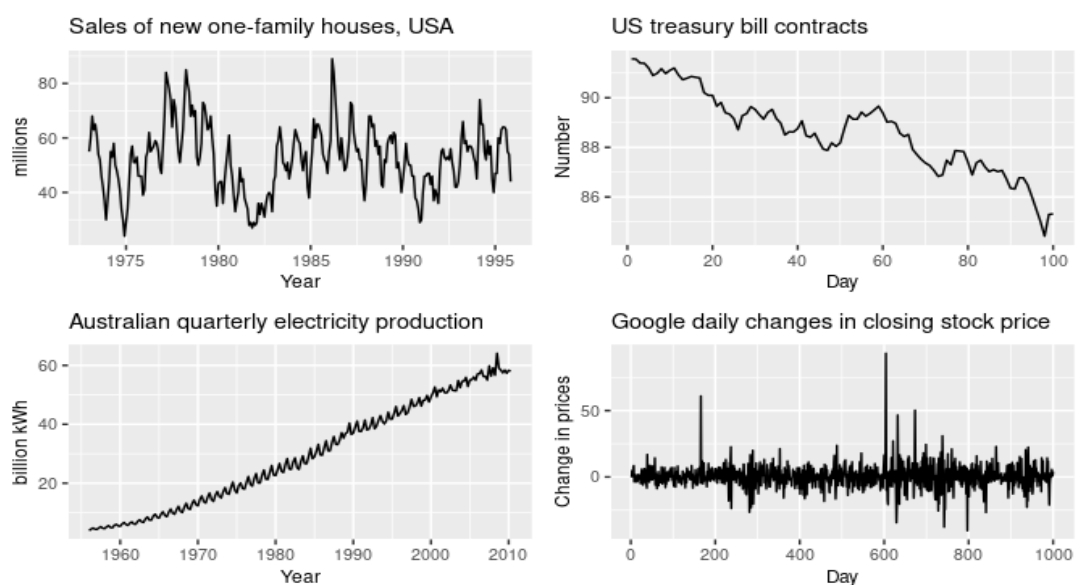


Hình 1.4 Hình minh họa thành phần Seasonal

Tính thời vụ thể hiện những biến động định kỳ các chu kỳ có thể diễn ra một mùa cụ thể như xuân, hạ, thu, đông hoặc cũng có thể là mùa thương mại, mùa nghỉ lễ [6].

- Cycle:

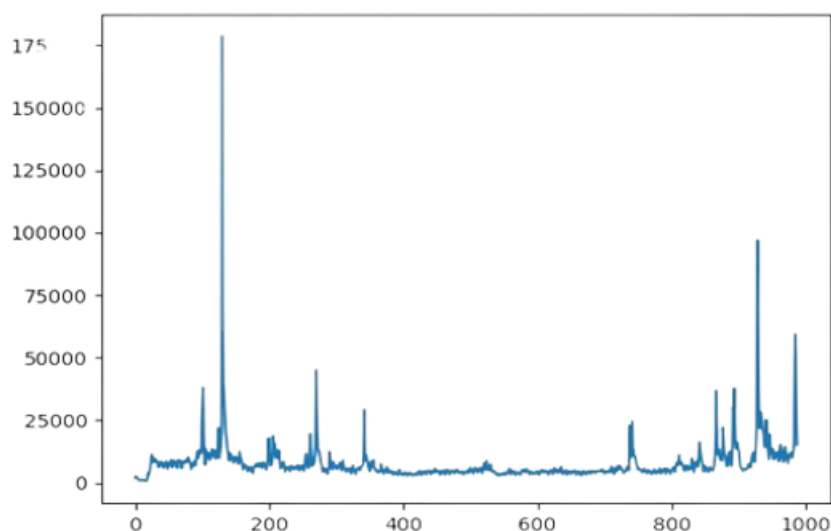
Cycle (chu kỳ): là một sự tồn tại tuần hoàn mà dữ liệu tăng hoặc giảm theo một chu kỳ cố định. Thời gian của một chu kỳ có thể 1 tháng, 1 quý, 1 năm, 2 năm, 10 năm... [6].



Hình 1.5 Hình ảnh minh họa thành phần Cycle

- Noise

Noise là sự thay đổi đột ngột trong việc quan sát mô hình không thể giải thích được. Những thành phần này không phù hợp với phần còn lại của chuỗi và có thể gây ảnh hưởng đến việc phân tích, do đó ảnh hưởng đến khả năng dự báo của mô hình chuỗi thời gian [6].



Hình 1.6 Hình ảnh minh họa thành phần Noise

Chuỗi thời gian dừng (Stationary) là một chuỗi không phụ thuộc vào thời điểm mà chuỗi được quan sát đồng thời không có các yếu tố xu hướng hoặc thời vụ - tính xu hướng và tính thời vụ sẽ ảnh hưởng đến giá trị của chuỗi thời gian tại thời gian khác nhau.. [6]

Một chuỗi thời gian dừng nếu phương sai (variance) và trung bình (mean) không thay đổi trong suốt thời gian quan sát và giá trị hiệp phương sai (covariance) giữa hai giai đoạn chỉ phụ thuộc vào khoảng cách giữa hai giai đoạn ấy [12].

Ví dụ: Các hệ số tự tương quan giữa $y_t, y_{t-1}, y_{t-2}, \dots, y_{t-p}$ sẽ là như nhau; nhưng khác với các hệ số tự tương quan giữa y_t và $y_{t-1}, y_{t-2}, \dots, y_{t-p}$. Nếu ta chia dữ liệu thành các giai đoạn khác nhau, thì giản đồ tự tương quan của các giai đoạn đó là như nhau [12].

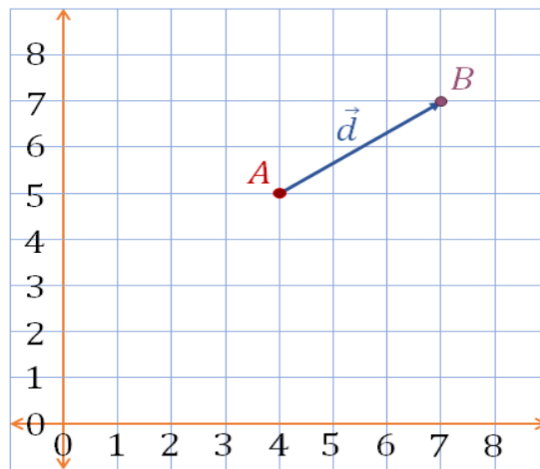
Trong nghiên cứu về chuỗi thời gian người ta thường giả định thời gian có tính dừng phần lớn do các công cụ sử dụng cho phân tích và dự báo chuỗi thời gian giả định

thời gian có tính dừng. Chuỗi thời gian không cố định thì phân phối sẽ thay đổi liên tục theo thời gian và nếu chỉ quan sát một lần cho mỗi khoảng thời gian thì rất khó để đưa ra các suy luận về chuỗi thời gian đó [12].

1.2. ĐỘ ĐO TRONG BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN

1.2.1. Độ đo Euclidean

Khoảng cách giữa hai điểm trong không gian Euclidean là độ dài đoạn thẳng nối hai điểm với nhau. Khoảng cách giữa các điểm này sẽ được đưa vào tọa độ Descartes và sử dụng định lý Pythagore để tính khoảng cách giữa các điểm [11].



Hình 1.7 Hình ảnh minh họa độ đo Euclide trong tọa độ Descartes

Công thức:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1.1)$$

Trong đó:

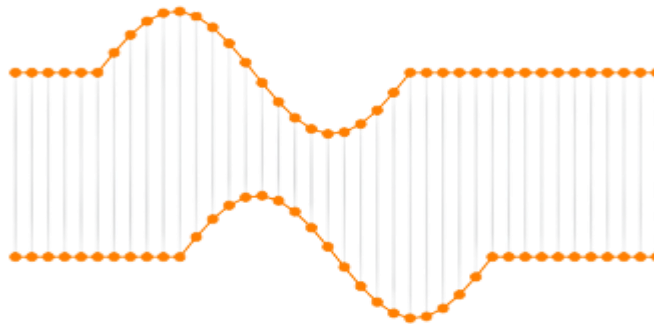
x_1, y_1 : tọa độ điểm thứ nhất

x_2, y_2 : tọa độ điểm thứ hai

d : khoảng cách giữa (x_1, y_1) và (x_2, y_2)

Trong time series, khoảng cách giữa hai chuỗi thời gian là khoảng cách giữa các điểm trong không gian n chiều, với n là số lượng thời điểm trong chuỗi thời gian. Euclidean thường được dùng làm độ đo khoảng cách trong các bài toán phân tích chuỗi thời gian. Tuy nhiên trong không gian đa chiều, việc sử dụng Euclidean là độ đo chuẩn thì

không phù hợp vì khó có thể quan sát một các trực quan và sự bùng nổ theo cấp số nhân do số lượng chiều quan sát dẫn đến việc khó tính khoảng cách Euclide [11].

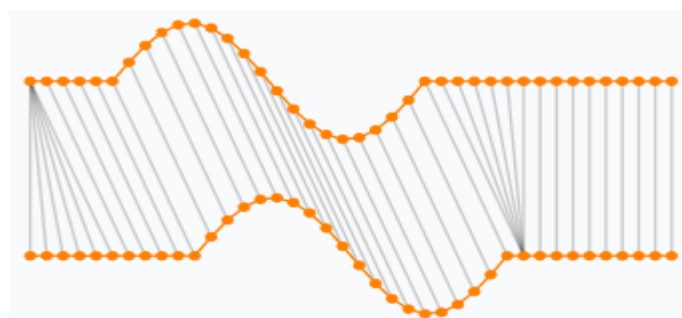


Hình 1.8 Hình ảnh minh họa độ đo Euclide trong chuỗi thời gian

Khoảng cách Euclide được dùng để so sánh các chuỗi thời gian, nếu khoảng cách giữa hai chuỗi thời gian càng nhỏ càng giống nhau và ngược lại. Tuy nhiên Euclide không phù hợp với các chuỗi thời gian có thay đổi nhanh chóng trong một khoảng thời gian ngắn hoặc các chuỗi có độ dài khác nhau [11].

1.2.2. Độ đo Dynamic Time Warping

Dynamic Time Warping (DTW) là thước đo độ tương đồng giữa hai chuỗi thời gian không đồng nhất với nhau về chiều dài và hình dạng. DTW tìm kiếm đường có độ dài ngắn nhất để nối các điểm trong chuỗi thời gian, những đường này được gọi là đường cong tối ưu. Đường cong tối ưu được tìm thấy bằng cách sử dụng Dynamic Programming để lưu trữ các giá trị khoảng cách tối ưu giữa hai chuỗi thời gian [11].



Hình 1.9 Hình ảnh minh họa độ đo Dynamic Time Warping trong chuỗi thời gian

Công thức:

$$D(i, j) = |x_i - y_j| + \min\{D(i-1, j); D(i-1, j-1); D(i, j-1)\} \quad (1.2)$$

Trong đó:

x, y : là hai chuỗi thời gian cần tính khoảng cách

i, j : là vị trí điểm dữ liệu trên hai chuỗi thời gian

Ưu điểm:

DTW là một thước đo linh hoạt khác với thước đo Euclide có thể so sánh chuỗi thời gian có độ dài khác và hình dạng khác nhau bằng cách làm cong một chuỗi dọc theo trục thời gian để khớp với chuỗi còn lại. DTW là một phương pháp phi tham số phù hợp để phân tích dữ liệu chuỗi thời gian có xu hướng phi tuyến tính hoặc phân phối không bình thường [13].

1.2.3. Tầm quan trọng của dự báo và chuỗi thời gian

Dự báo có các ứng dụng trong các ngành công nghiệp khác nhau. Có rất nhiều ứng dụng thực tế bao gồm: dự báo thời tiết, dự báo kinh tế, dự báo chăm sóc sức khỏe, dự báo tài chính, dự báo kinh doanh, v.v. Về cơ bản, bất kỳ ai có dữ liệu lịch sử nhất quán đều có thể phân tích dữ liệu đó bằng các phương pháp phân tích chuỗi thời gian, sau đó lập mô hình, dự báo và dự đoán. Đối với một số ngành, toàn bộ mục đích của phân tích chuỗi thời gian là để hỗ trợ dự báo. Một số công nghệ, chẳng hạn như phân tích tăng cường, thậm chí có thể tự động chọn dự báo trong số các thuật toán thống kê khác nếu nó mang lại sự chắc chắn nhất.

- Chuỗi thời gian được nhiều tổ chức, công ty sử dụng để dự báo xu hướng lãi, lỗ trong kinh doanh của họ từ đó có thể đưa ra các quyết định kinh doanh quan trọng trong quá trình phát triển.
- Được sử dụng để so sánh xu hướng hiện tại với xu hướng trong quá khứ đã xảy ra để có thể ước tính và chuẩn bị xu hướng trong tương lai.
- Các biến thể của chu kỳ trong một khoảng thời gian bằng cách sử dụng chuỗi thời gian sẽ cho phép ta hiểu được chu kỳ kinh doanh một cách khá hiệu quả.

- Được sử dụng để hiểu một sự kiện có thể thay đổi tính năng của nó trong một khoảng thời gian và do đó có thể xác định độ tin cậy, tính linh hoạt và các tính năng quan trọng khác.

Các mẫu tính hiệu phức có thể áp dụng một số phép biến đổi như phân tích Fourier để khử nhiễu biểu đồ và chia mẫu phức tạp thành một loạt các mẫu đơn giản hơn và do đó có thể hiểu rõ hơn.

1.3. BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN

Dự báo chuỗi thời gian (time series forecasting) là một kỹ thuật sử dụng dữ liệu trong quá khứ và hiện tại để dự đoán các giá trị tương lai trong một khoảng thời gian hoặc một điểm cụ thể [8]. Tuy nhiên một dự đoán không phải lúc nào cũng là chính xác và khả năng các dự báo có thể rất khác nhau đặc biệt là khi xử lý các biến thường dao động trong dữ liệu chuỗi thời gian cũng như các yếu tố nằm ngoài tầm kiểm soát.

1.3.1. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một kỹ thuật xử lý các dữ liệu thô vì dữ liệu trong thực tế thường bị thiếu giá trị không có đặc tính hay xu hướng cụ thể, không nhất quán và tiềm ẩn nhiều lỗi gây ảnh hưởng đến quá trình phân tích và dự báo.

Quá trình tiền xử lý:

- Làm sạch dữ liệu: dữ liệu được làm sạch thông qua việc xử lý các missing có thể dùng Interpolate để điền vào các giá trị bị thiếu bằng cách lấy trung bình của các giá trị trước và sau đó. Loại bỏ các giá trị bị nhiễu bằng cách thay thế các giá trị nằm ngoài khoảng x độ lệch chuẩn với giá trị trung bình của dữ liệu với x là giá trị quy định phù hợp với tập dữ liệu .
- Tích hợp dữ liệu: dữ liệu với được biểu diễn khác nhau cùng ở trong một thuộc tính sẽ gây ra sự xung đột ví dụ giá trị null xuất hiện trong cột dữ liệu giá trị tạo sự khác biệt nên cần xử lý bằng cách thay các giá trị null bằng 0.
- Chuẩn hóa dữ liệu:

Chuẩn hóa trung bình zero (Zero-Mean normalization): là phương pháp đưa dữ liệu trượt theo chiều dọc sao cho mức trung bình bằng 0 [11].

Công thức:

$$Q^1[i] = (Q[i] - \text{mean}(Q)) / \text{var}(Q) \quad (1.3)$$

Trong đó:

$\text{mean}(Q)$ là giá trị trung bình của Q .

$\text{var}(Q)$ là độ lệch chuẩn của Q .

Chuẩn hóa min max (min/max normalization): là phương pháp dùng để đưa tập dữ liệu về phạm vi $[0, 1]$ hoặc $[-1, 1]$. Công thức:

$$Q'[i] = \frac{Q[i] - \text{Min}_{old}}{\text{Max}_{old} - \text{Min}_{old}} (\text{Max}_{new} - \text{Min}_{new}) + \text{Min}_{new} \quad (1.4)$$

Trong đó:

Min_{old} và Max_{old} là giá trị nhỏ nhất và lớn nhất của chuỗi ban đầu.

Min_{new} và Max_{new} là giá trị nhỏ nhất và lớn nhất của chuỗi sau khi được chuẩn hóa.

1.3.2. Xây dựng mô hình

Xây dựng mô hình là quá trình tạo ra mô hình máy học từ dữ liệu đã tiền xử lý để dự đoán giá trị tương lai được áp dụng trong nhiều lĩnh vực như dự báo kinh tế, thời tiết... Việc đầu tiên cần làm khi xây dựng mô hình là chọn mô hình phù hợp với tập dữ liệu cần phân tích và mục đích dự đoán của bài toán.

1.3.3. Đánh giá mô hình

Đánh giá mô hình dự đoán chuỗi thời gian có vai trò rất quan trọng vì xác định được độ chính xác của mô hình, đảm bảo tính ứng dụng không chỉ với các dữ liệu đã được train. Khi biết được độ chính xác của mô hình sẽ đưa ra những cải tiến mô hình thích hợp. Có nhiều phương pháp để đánh giá mô hình dự đoán chuỗi thời gian như: MAE, MSE, RMSE,...

MAE (mean absolute error) lỗi tuyệt đối trung bình được tính bằng giá trị trung bình của các giá trị sai số dự báo, mà trong đó các giá trị dự báo bắt buộc dương, đều này được thể hiện bằng việc sử dụng hàm giá trị tuyệt đối [8].

Công thức:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n} \quad (1.5)$$

Trong đó:

n : số lỗi

y_i : giá trị quan sát thứ i

x_i : giá trị dự đoán tương ứng

e_i sai số tuyệt đối

MSE (mean squared error) lỗi bình phương trung bình được tính bằng giá trị trung bình của lỗi dự báo bình phương các giá trị. Bình phương các giá trị lỗi dự báo buộc là dương. Các lỗi dự báo rất lớn hoặc các ngoại lệ được bình phương, trong đó các lượt có các dụng kéo giá trị trung bình của các lỗi dự báo bình phương ra ngoài dẫn đến điểm lỗi bình phương trung bình lớn hơn. Thực tế, các giá trị đem lại hiệu suất kém hơn cho những mô hình đưa ra dự báo sai lớn [\[8\]](#).

Công thức:

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (1.6)$$

Trong đó:

n : số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

RMSE (root mean squared error) lỗi bình phương trung bình gốc xuất phát từ việc lỗi bình phương được mô tả theo đơn vị bình phương của cả dự đoán và nó có thể chuyển đổi trở lại vị trí ban đầu của dự đoán bằng cách lấy căn bậc hai của điểm lỗi bình phương trung bình. Các giá trị RMSE có cùng đơn vị với các dự đoán. Nếu RMSE bằng 0 thì mean_squared_error (bình phương trung bình lỗi) không có lỗi [\[8\]](#).

Công thức:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (1.7)$$

Trong đó:

n : số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

MAPE (Mean absolute percentage error) là phần trăm sai số trung bình tuyệt đối đo mức độ chính xác của mô hình dự đoán. MAPE được tính bằng sai số phần trăm trung bình tuyệt đối của dữ liệu dự đoán so với dữ liệu thực tế và sau đó tính trung bình sai số.

Công thức:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{|y_i|} \quad (1.8)$$

Trong đó:

n : số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

1.3.4. Dự đoán

Sử dụng các mô hình như KNN, FFNN, ... để dự báo các giá trị tương lai của tập dữ liệu chuỗi thời gian. Ngoài việc sử dụng các mô hình đơn lẻ có thể kết hợp hai hoặc ba mô hình lại để tăng độ chính xác của dự báo.

1.3.5. Đánh giá kết quả

So sánh kết quả dự báo với kết quả trong thực tế, đánh giá kết quả chính xác như thế nào. Nếu dự báo cho ra kết quả có độ chính xác kém từ đó thay đổi các siêu tham số,

thay đổi các mô hình dự báo hoặc kết hợp các mô hình khác nhau để tăng độ chính xác cho bài toán dự báo.

1.4. NGHIÊN CỨU THỰC HIỆN TRAINING VÀ TESTING CHO MÔ HÌNH

Đối với các bài toán dự báo thông qua các mô hình dự báo, để có thể giảm thiểu tối đa độ lỗi cho mô hình và cung cấp cho ra kết quả dự báo chính xác nhất, phần lớn đều sẽ phải thực hiện hai quá trình gọi là quá trình training và quá trình testing.

1.4.1. Quá trình training

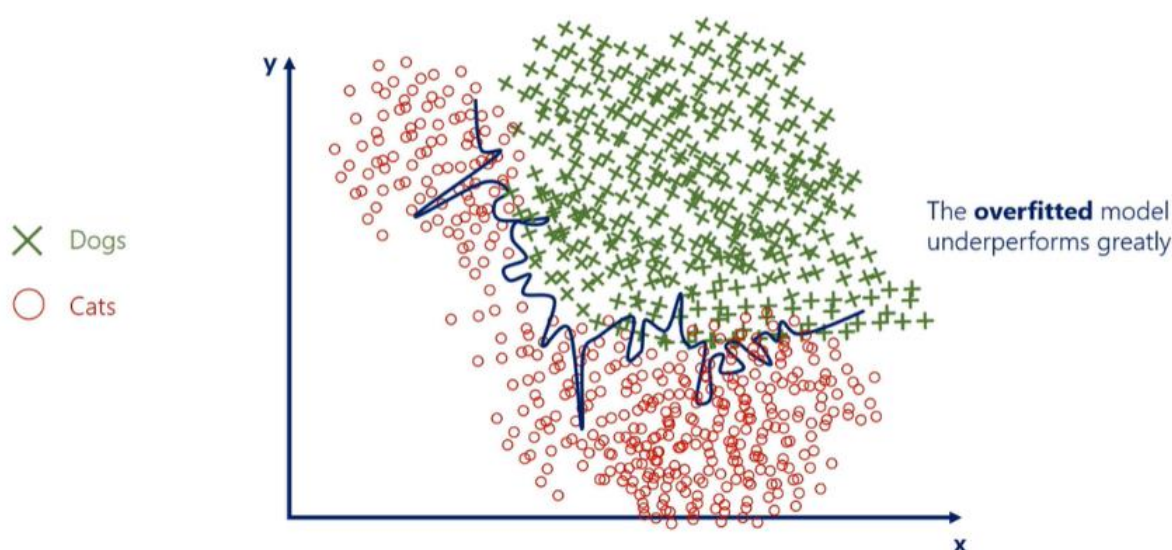
Quá trình training là quá trình đưa dữ liệu huấn luyện vào mô hình với các tham số khởi tạo ngẫu nhiên. Kết quả dự báo từ mô hình được so sánh với giá trị thực để tính toán lỗi. Sau đó, mô hình sẽ điều chỉnh các trọng số bằng cách lặp lại quá trình này cho đến khi tìm được giá trị trọng số tối ưu, đạt được giá trị lỗi dự báo (MSE) nhỏ nhất.

1.4.2. Quá trình testing

Quá trình testing được sử dụng để đánh giá hiệu quả của mô hình trên dữ liệu kiểm tra. Kết quả dự báo được tính toán từ tham số mô hình đã được điều chỉnh bởi quá trình training và sau đó so sánh với giá trị thực để tính toán lỗi dự báo. Kết quả này sẽ được sử dụng để đánh giá độ chính xác và hiệu suất của mô hình trên dữ liệu kiểm tra.

1.5. PHÂN BIỆT OVERFITTING VÀ UNDERFITTING

1.5.1. Overfitting

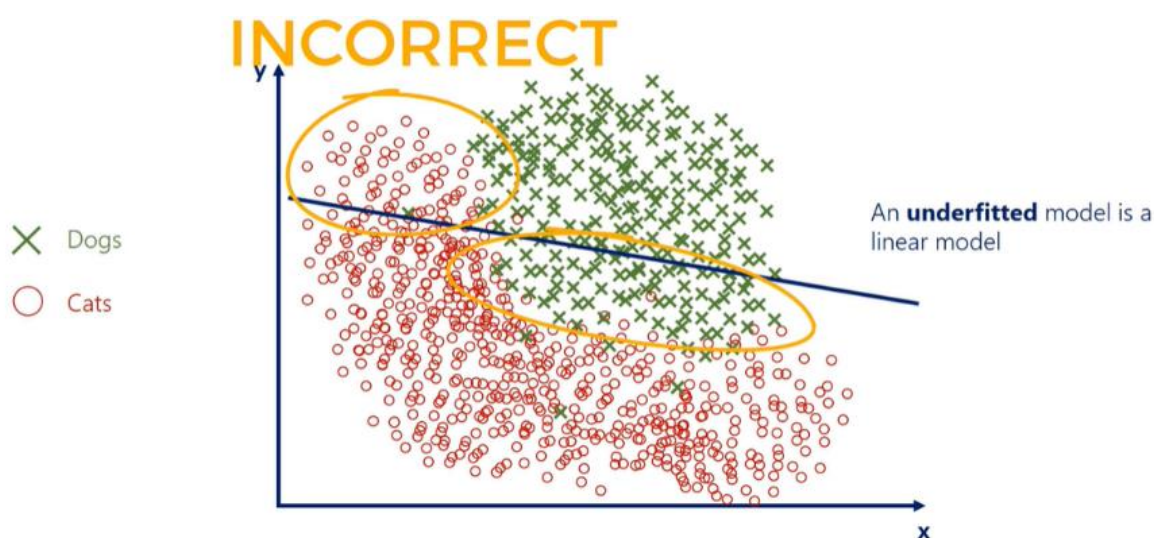


Hình 1.10 Hình minh họa Overfitting

Overfitting là một hành vi học máy xảy ra khi một mô hình dự đoán chính xác dữ liệu được huấn luyện dẫn đến việc mô hình không khái quát hóa tốt cho dữ liệu mới. Khi sử dụng các thuật toán máy học để huấn luyện mô hình, nếu mô hình quá phức tạp hoặc được đào tạo quá lâu trên tập dữ liệu mẫu, thì dẫn đến việc xuất hiện các noise “tiếng ồn”. Khi mô hình đã ghi nhớ các “tiếng ồn”, các thông tin không liên quan hoặc quá khớp với tập huấn luyện mô hình sẽ không khái quát hóa tốt dữ liệu mới, ảnh hưởng đến các công việc phân loại hoặc dự đoán của mô hình [15].

Tỉ lệ lỗi thấp và phương sai cao là những chỉ báo tốt nhưng ở quá mức để khái quát tập dữ liệu, để phòng trường hợp này xảy ra nên chia tập dữ liệu thành hai phần là tập “train” và tập “test”, “train” dùng cho việc huấn luyện, “test” dùng để kiểm tra mô hình có khớp quá mức hay không [15].

1.5.2. Underfitting



Hình 1.11 Hình minh họa Underfitting

Underfitting là một mô hình thống kê không thể nắm bắt xu hướng cơ bản của dữ liệu, không thể nắm bắt chính xác các mối quan hệ giữa các biến đầu vào, tỉ lệ lỗi cao trên tập train và test [14]. Underfitting xảy ra khi mô hình cần nhiều thời gian đào tạo, mô hình quá đơn giản, cần nhiều tính năng đầu vào hơn. Khác với overfitting, underfitting không thể xác định chính xác xu hướng và khái quát tập dữ liệu dẫn đến hiệu suất dự đoán và phân loại của mô hình không chính xác [15].

Độ lệch cao và phương sai thấp là nguyên nhân dẫn đến việc underfitting, làm cho một mô hình không thể khái quát hóa dữ liệu mới, không hoạt động tốt trên tập train nên underfitting sẽ dễ phát hiện hơn overfitting thuận tiện cho việc thay đổi các thuật toán học máy đưa vào mô hình tránh tình trạng underfitting [14]. Để tránh underfitting nên duy trì độ phức tạp của mô hình, thiết lập các mối quan hệ phi phối các biến đầu vào để đưa ra các dự đoán chính xác hơn [15].

1.5.3. Cách khắc phục Overfitting và Underfitting

Overfitting:

- Bổ sung thêm dữ liệu huấn luyện: Khi mô hình bị overfitting không hoạt động tốt trên các tập dữ liệu mới do dữ liệu mà mô hình đào tạo không đáp ứng yêu cầu của tập dữ liệu test. Nên đào tạo thuật toán tập dữ liệu lớn, phong phú và đa dạng hơn sẽ cải thiện hiệu suất của mô hình. [16].
- Dừng sớm: Cho mô hình dừng trước khi học các nhiễu trong dữ liệu nhưng việc này có một khuyết điểm là việc chọn thời điểm thích hợp để dừng mô hình [16].
- Đơn giản hóa mô hình: Bằng cách loại bỏ ngẫu nhiên các tính năng không cần thiết, giảm số lượng tham số để độ phức tạp của mô hình giảm đi [16].

Underfitting:

- Giảm số lượng dữ liệu: Loại bỏ một số tính năng và làm cho dữ liệu đơn giản hơn có thể giúp giảm tình trạng thừa [15].
- Tăng độ phức tạp của mô hình: Mô hình đào tạo có thể không phù hợp vì nó không đủ phức tạp để nắm bắt các mẫu trong dữ liệu. Mô hình có thể chuyển từ tuyến tính sang phi tuyến tính, tăng các siêu tham số [14].

1.6. CÁC MÔ HÌNH DỰ BÁO CHUỖI THỜI GIAN

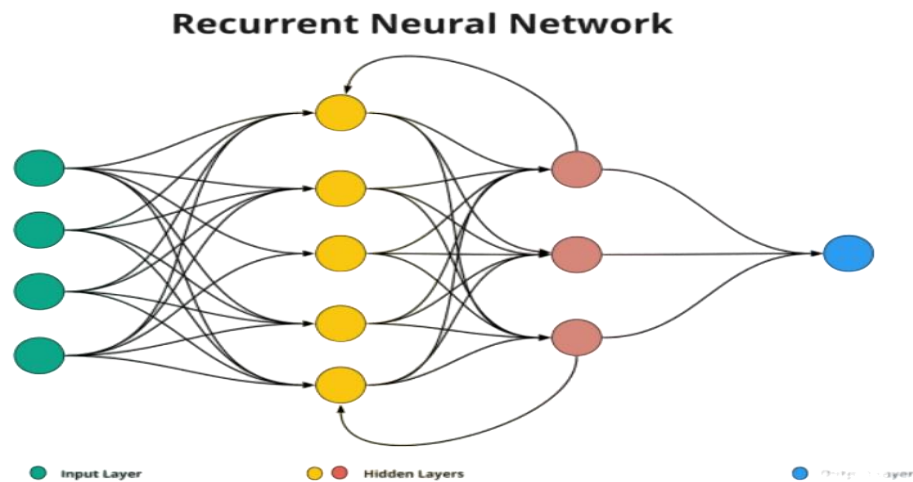
Dự báo chuỗi thời gian cũng là một lĩnh vực quan trọng của học máy thuộc học tập có giám sát. Dự báo chuỗi thời gian được dùng để phân tích dữ liệu chuỗi thời gian thuộc thành phần nào cộng với việc kết hợp với các phương pháp học máy như: Regression, Neural Networks, Support Vector Machines, Random Forests và XGBoost, KNN, RNN, LSTM... để áp dụng vào bài toán từ đó có thể dự báo chính xác hơn.

1.6.1. Phương pháp học sâu

1.6.1.1. Mô hình Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) là một dạng mạng nơ-ron truyền thẳng (feedforward neural network) được sử dụng để xử lý dữ liệu chuỗi (sequential data) như chuỗi thời gian, văn bản, âm thanh. Khác với mạng nơ-ron truyền thẳng thông thường, RNN có khả năng lưu giữ thông tin của chuỗi vào bộ nhớ ẩn (hidden state) và sử dụng lại thông tin đó cho các phân tích và dự đoán sau này [9].

Cấu trúc của RNN bao gồm các đơn vị đơn giản gọi là cell, mỗi cell có một bộ nhớ ẩn và một hàm trạng thái (state function) để tính toán trạng thái ẩn của cell từ trạng thái của cell trước đó và đầu vào hiện tại. Các cell trong RNN được kết nối với nhau theo chuỗi thời gian để tạo thành một mạng RNN [9].



Hình 1.12 Mô hình Recurrent Neural Network

Công thức:

$$h_t = f(w_{hh}h_{t-1} + w_{xh}x_t + b_h) \quad (1.9)$$

Trong đó:

h_t là trạng thái của nơ-ron đơn vị bộ nhớ tại thời điểm t

h_{t-1} là trạng thái của nơ-ron đơn vị bộ nhớ tại thời điểm trước đó $t-1$

x_t là đầu vào tại thời điểm t

W_{hh} là ma trận trọng số liên kết giữa các nơ-ron đơn vị bộ nhớ

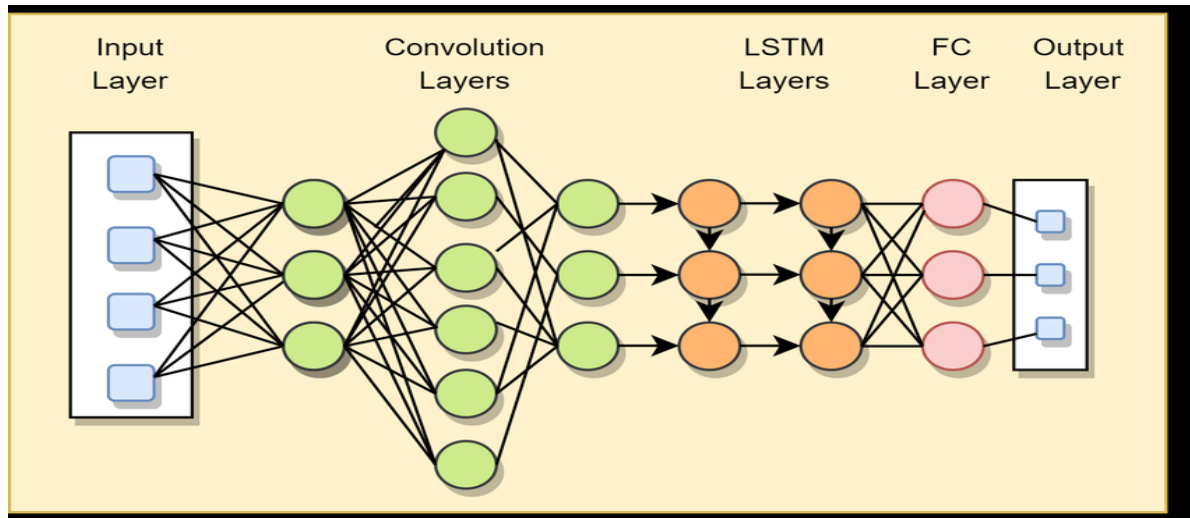
W_{xh} là ma trận trọng số liên kết giữa đầu vào x_t và nơ-ron đơn vị bộ nhớ

b_h là vector độ lệch của nơ-ron đơn vị bộ nhớ

f là hàm kích hoạt của nơ-ron đơn vị bộ nhớ

1.6.1.2. Mô hình Long Short-Term Memory (LSTM)

Mô hình LSTM là một dạng đặc biệt được cải tiến từ mô hình mạng RNN được Hochreiter và Schmidhuber đề xuất vào năm 1997 nhằm khắc phục vấn đề phụ thuộc xa của mô hình RNN. Mô hình LSTM được xây dựng từ 4 thành phần quan trọng (cell, forget gate, input gate, output gate) [10].



Hình 1.13 Mô hình Long Short-Term Memory

Công thức:

$$i_t = \sigma(x_t * U_i + H_{t-1} * w_1) \quad (1.10)$$

Trong đó:

x_t : mốc thời gian hiện tại

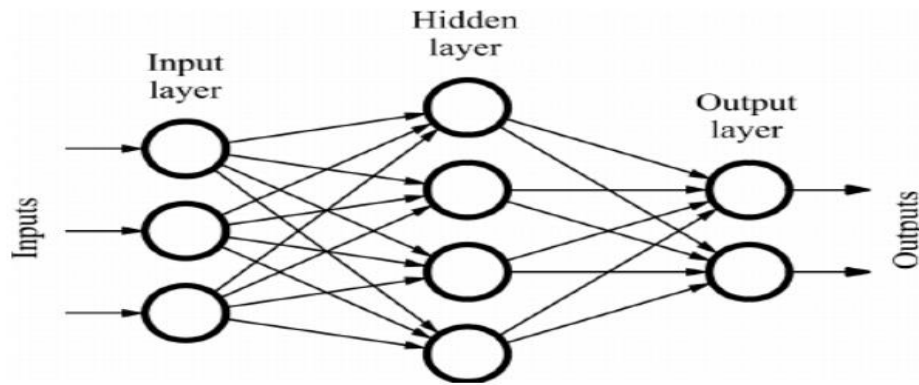
U_i : trọng số liên quan đến đầu vào

H_{t-1} : trạng thái ẩn của đầu thời gian trước đó

W_1 : Ma trận trọng số của đầu vào liên quan đến trạng thái ẩn

1.6.1.3. Mô hình Feedforward Neural Network(FFNN)

Feedforward neural network (FFNN) hay còn gọi là mạng thần kinh truyền thẳng, trong mỗi liên kết giữa các nút của mạng thần kinh hình thành một chu kỳ. Mạng thần kinh truyền thẳng là mô hình xử lý dữ liệu đơn giản nhất vì dữ liệu sẽ được truyền đi theo một hướng và không bao giờ phản hồi ngược lại [5].



Hình 1.14 Mô hình Long Feedforward Neural Network

Công thức:

$$f(x, w) = g \left(\sum_{i=1}^n w_{ji} \times x_i + b_j \right) \quad (1.11)$$

Trong đó:

$X = [x_0, x_1, \dots, x_n]$: là vector của các quan sát đầu vào chuỗi thời gian.

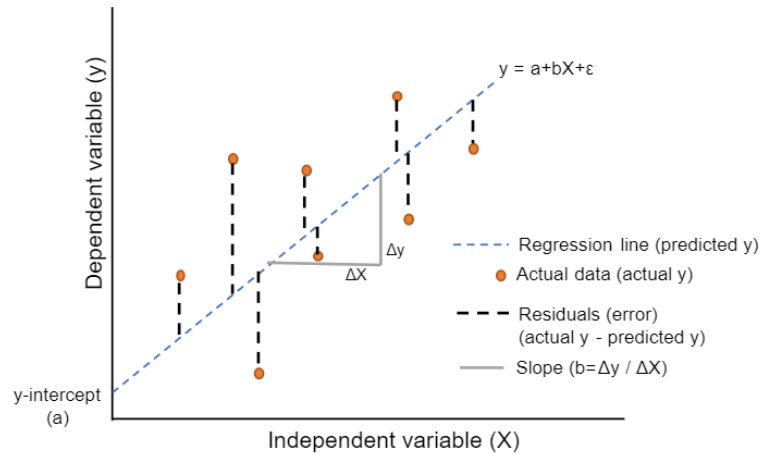
$w = (\beta, w)$: trọng lượng mạng

$g()$: hàm truyền phi tuyến tính.

1.6.2. Phương pháp tuyến tính

1.6.2.1. Mô hình Linear Regression

Regression (hồi quy) là một phương pháp thống kê để tìm ra mối quan hệ giữa các biến độc lập và kết quả phụ thuộc. Hồi quy là một nghiên cứu thống kê và là một thành phần chính của mô hình dự đoán nên được áp dụng nhiều trong các ứng dụng học máy. Hồi quy được sử dụng trong các lĩnh vực như: tài chính, chứng khoán, y tế, ... để xác định cường độ và đặc điểm các mối quan hệ giữa các biến. Hồi quy được sử dụng để tiếp cận các dự đoán các kết quả liên tục trong mô hình dự đoán. Hồi quy thường liên quan đến mô hình được vẽ một đường phù hợp nhất thông qua các điểm dữ liệu.



Hình 1.15 Mô hình Linear Regression

Công thức:

$$y = a + bx + \varepsilon \quad (1.12)$$

Trong đó:

a: giao điểm (giá trị của y khi x = 0)

b: hệ số góc của đường thẳng

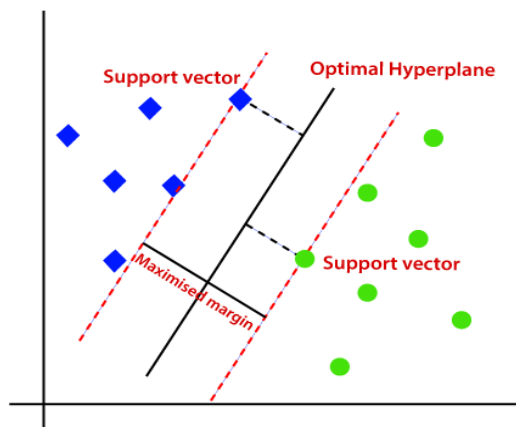
x: biến giải thích

y: biến phụ thuộc

ε: lỗi ngẫu nhiên

1.6.2.2. Mô hình Support Vector Machines (SVM)

Support Vector Machines (SVM) là một trong những thuật toán Supervised Learning phổ biến nhất được sử dụng để phân loại, hồi quy và phát hiện giá trị ngoại lệ. Thuật toán SVM tạo ra đường hoặc ranh giới quyết định tốt nhất có thể phân tách không gian thành các lớp để có thể đặt điểm dữ liệu mới vào vị trí mong muốn trong tương lai.



Hình 1.16 Mô hình Support Vector Machines

Ưu điểm của SVM là:

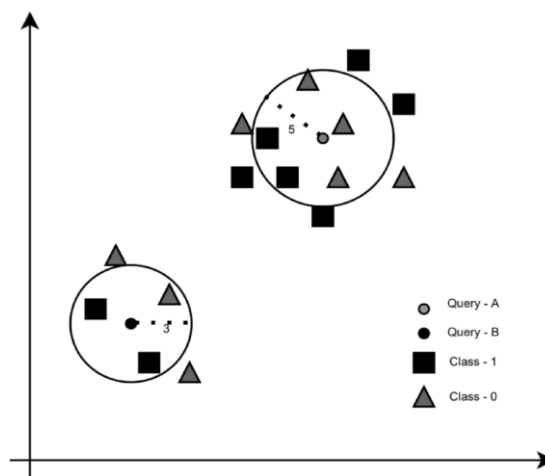
- Hiệu quả trong không gian n chiều.
- Hiệu quả trong trường hợp số chiều lớn hơn số mẫu.
- Có thể xử lý các bộ dữ liệu bị nhiễu hoặc dữ liệu bị lỗi theo một cách hiệu quả.

Nhược điểm của SVM:

- Yêu cầu sự việc cân nhắc kỹ lưỡng về việc lựa chọn kernel hàm và các tham số tương ứng.
- Khó hiểu và khó giải thích kết quả.
- Cần có sự cân bằng giữa lượng mẫu trong mỗi lớp để tránh sự phức tạp trong phân phối dữ liệu và tăng độ chính xác của mô hình.

1.6.2.3. Mô hình *K-Nearest Neighbors (KNN)*

KNN (K-Nearest Neighbor) là một thuật toán học máy đơn giản được sử dụng để dự đoán hoặc phân loại các điểm dữ liệu mà không cần thông qua quá trình train. Thuật toán K-NN sẽ đánh giá sự giống nhau giữa mẫu dữ liệu mới với các mẫu dữ liệu có sẵn trong tập dữ liệu dựa trên các độ đo thích hợp để tiến hành dự đoán hoặc phân loại. [7].



Hình 1.17 Mô hình *K-Nearest Neighbors*

Các thuật toán KNN hoạt động như sau:

Với mỗi điểm dữ liệu mới, tính khoảng cách từ điểm đó đến tất cả các điểm trong tập huấn luyện.

Chọn ra các điểm gần với điểm dữ liệu mới.

Dự đoán loại cho điểm dữ liệu mới bằng cách sử dụng k điểm đã chọn và phương thức biểu thị tần số của chúng.

Lựa chọn lựa k là một tham số quan trọng của thuật toán, và cần được xác định trước khi áp dụng thuật toán. Nếu k quá nhỏ, dự đoán có thể bị ảnh hưởng do nhiễu và các giá trị lạ, nếu k quá lớn, việc phân loại có thể trở nên phức tạp hơn.

KNN có thể được áp dụng cho cả bài toán phân loại và dự đoán, tùy thuộc vào loại đầu ra của mô hình.

Công thức:

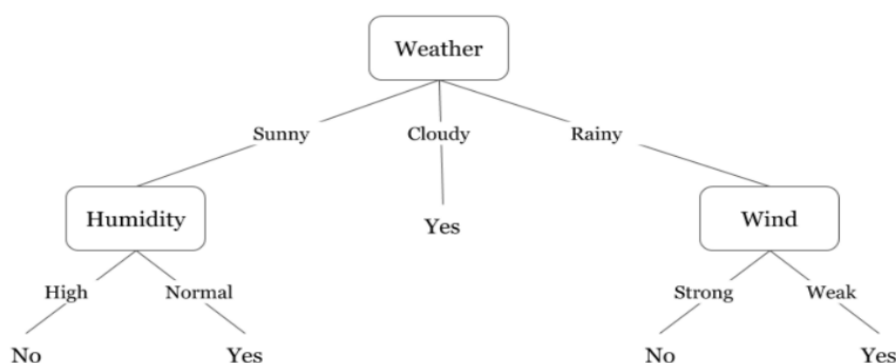
Trong độ đo Euclidean

$$D = \sqrt{\sum_{i=1}^k (x_i - y_i)^2} \quad (1.13)$$

Một trường hợp được phân loại theo đa số phiếu bầu của các điểm dữ liệu lân cận, với trường hợp được gán cho lớp phổ biến nhất trong số k hàng xóm gần nhất được đo bằng hàm khoảng cách.

1.6.3. Phương pháp phi tuyến tính

Mô hình cây quyết định là một trong những mô hình học máy phổ biến được sử dụng trong phân loại và dự đoán. Mô hình này được xây dựng trên cơ sở quyết định, trong đó mỗi nút đại diện cho một biến và các nhánh liên kết đại diện cho các giá trị có thể có của biến đó. Từ nút gốc, cây được xây dựng thông qua quá trình chia nhỏ dữ liệu thành các tập con nhỏ hơn, tối đa hóa mức độ tinh khiết của mỗi tập con. Quá trình này được lặp lại cho đến khi các tập con không thể được chia nhỏ nữa hoặc khi đạt được một điều kiện dừng khác.



Hình 1.18 Mô hình Decision Tree

Một trong những ưu điểm của mô hình Cây quyết định là tính dễ giải thích. Bằng cách theo dõi các quyết định được đưa ra trên các nút, ta có thể hiểu được cách mà mô

hình ra quyết định phân loại hoặc dự đoán. Ngoài ra, Decision Tree còn cho phép xử lý các biến đầu vào có kiểu dữ liệu khác nhau, bao gồm cả các biến dạng rời và liên tục.

Tuy nhiên, mô hình cây quyết định cũng có một số nhược điểm. Một trong những nhược điểm chính của mô hình là rất dễ bị khớp. Khi mô hình được xây dựng trên tập dữ liệu đào tạo, nó có thể hoàn toàn ghi nhớ các mẫu đào tạo được tạo thay vì học cách phân loại dữ liệu chung. Điều này dẫn đến hiện tượng mô hình không tốt trong dự đoán công việc trên các tập dữ liệu mới. Một cách để giải quyết vấn đề này là sử dụng các kỹ thuật như cắt cây (cắt tỉa) hoặc sử dụng các mô hình Ensemble như Random Forest.

Trong cây quyết định, đầu ra chủ yếu là “có” hoặc “không”

Công thức Entropy :

$$E(S) = -p_{(+)} \log \log p_{(+)} - p_{(-)} \log \log p_{(-)} \quad (1.14)$$

Trong đó:

entropy Thuật toán cây quyết định

p_{+} là xác suất của lớp tích cực

p_{-} là xác suất của hạng âm

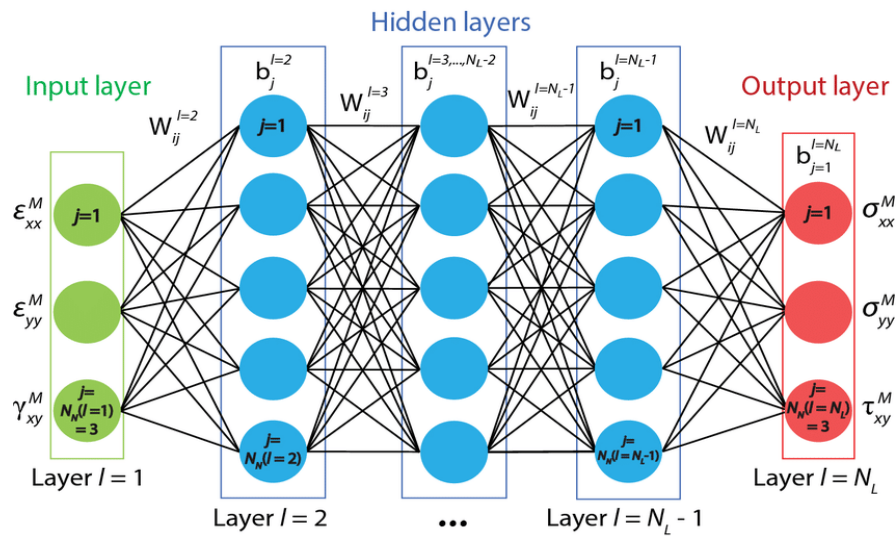
S là tập con của ví dụ huấn luyện

CHƯƠNG 2: ỨNG DỤNG MÔ HÌNH KNN VÀ FFNN VÀO BÀI TOÁN DỰ BÁO CHUỖI THỜI GIAN

2.1. FEED FORWARD NEURAL NETWORK (FFNN)

2.1.1. Định nghĩa

Feed Forward Neural Network là một mạng thần kinh nhân tạo trong đó các kết nối giữa các nút không tạo thành một chu kỳ. Mô hình chuyển tiếp nguồn cấp dữ liệu là dạng mạng thần kinh đơn giản nhất vì thông tin chỉ được xử lý theo một hướng. Mặc dù dữ liệu đi qua nhiều nút ẩn nhưng nó luôn di chuyển theo một hướng và không bao giờ quay ngược lại [5].



Hình 2.1 Hình minh họa mô hình FFNN với nhiều lớp ẩn

Trong bài toán dự báo chuỗi thời gian người ta thường dùng FFNN đơn giản 1 input layer - 1 hidden layer - 1 output layer trong đó số nút input do người thực hiện xác định, số nút ẩn do thực nghiệm để tìm ra, số nút output người ta có thể chọn 1 neural (Onestep) hoặc nhiều neural (Multistep).

FFNN là một phương pháp hiệu quả để xử lý dữ liệu chuỗi thời gian không có định, khả năng dự báo gần đúng và khái quát hóa bất kỳ mối quan hệ tuyến tính hoặc phi tuyến tính với bất kỳ độ chính xác nào [5].

Công thức:

$$f(x, w) = g \left(\sum_{i=1}^n w_{ji} \times x_i + b_j \right) \quad (2.1)$$

Trong đó:

$X = [x_0, x_1, \dots, x_n]$: là vector của các quan sát đầu vào bị trễ của chuỗi thời gian.

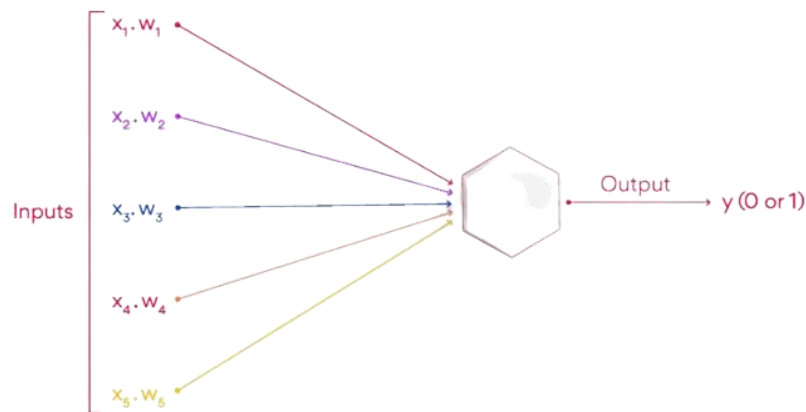
$w = (\beta, \gamma)$: trọng lượng mạng

I và H: số lượng các đơn vị đầu vào và ẩn trong mạng.

$g()$: hàm truyền phi tuyến tính.

2.1.2. Nguyên lý hoạt động

Khi mạng nơ ron chuyển tiếp nguồn cấp dữ liệu được đơn giản hóa, nó có thể xuất hiện dưới dạng một lớp tri giác đơn.



Hình 2.2 Hình minh họa mô hình FFNN với một lớp ẩn

Mô hình này nhân các đầu vào với trọng số khi chúng vào lớp. Sau đó, các giá trị đầu vào có trọng số được cộng lại với nhau để có được tổng. Miễn là tổng các giá trị tăng trên một ngưỡng nhất định, được đặt ở mức 0, thì giá trị đầu ra thường là 1, trong khi nếu nó giảm xuống dưới ngưỡng, nó thường là -1.

2.2. K-NEAREST NEIGHBORS (KNN)

2.2.1. Mô hình KNN

K-nearest neighbors hay còn gọi là k-NN là một thuật toán phân loại học có giám sát. KNN sẽ lưu trữ các điểm dữ liệu sẵn có và phân loại các điểm dữ liệu mới dựa trên sự tương đồng của các điểm dữ liệu có khoảng cách gần để phân loại dữ liệu thành một danh mục gần giống với dữ liệu mới. Ngoài ra KNN còn được gọi là lazy learner algorithm (thuật toán học lười) vì KNN không cần học từ tập huấn luyện thay vào đó KNN lưu trữ tập dữ liệu tại thời điểm phân loại và thực hiện các thao tác ngay tại thời điểm đó [7].

Thường được sử dụng để phân loại dựa trên các điểm gần nhau có xu hướng như thế nào từ đó tiến hành phân loại, ngoài ra KNN có thể được dùng cho các bài toán hồi quy. KNN là một thuật toán linh hoạt để xử lý các giá trị bị thiếu, thuật toán sẽ lấy các giá trị k lân cận để dự đoán các giá trị liên tục cho điểm dữ liệu mới [7].

Công thức:

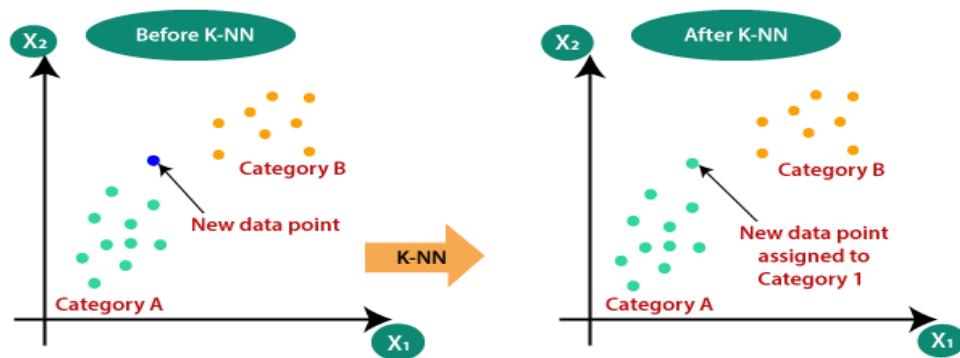
$$d_{(x,y)} = \sqrt{\sum_{i=1}^n (y_i - x_i)^2} \quad (2.2)$$

Trong đó:

n: số lượng điểm dữ liệu

x_i : vị trí điểm dữ liệu lân cận

y_i : vị trí điểm dữ liệu mới



Hình 2.3 Mô hình thuật toán KNN

Trong thuật toán KNN giá trị k dùng để xác định số lượng điểm dữ liệu sẽ được kiểm tra phân loại. Phải lựa chọn k sao cho phù hợp vì dữ liệu có thể chứa nhiều ngoại lệ hoặc nhiễu có khả năng làm giảm độ chính xác của dự đoán. Nên chọn k là một số lẻ để tránh ràng buộc trong phân loại và các xác thực có thể chọn ra k tối ưu cho tập dữ liệu [7].

Ứng dụng của KNN:

- Tiền xử lý dữ liệu: tập dữ liệu đầu vào thường gặp tình trạng bị thiếu các giá trị, thuật toán KNN có thể ước tính các giá trị đó bằng cách quy nạp dữ liệu bị thiếu.
- Chăm sóc sức khỏe: KNN đưa ra các dự đoán chuẩn đoán bệnh ... bằng cách tính toán và xem các biểu hiện gen có khả năng gây bệnh nhất.
- Nhận dạng: KNN có khả năng nhận dạng phân loại văn bản để xác định các chữ cái viết tay trên các tập phong bì.

2.2.2. Hoạt động

Cách hoạt động của KNN:

- Bước 1: Cung cấp tập huấn luyện

Tập huấn luyện là một tập hợp dữ liệu được sử dụng để huấn luyện một mô hình. Đề tài này sử dụng tỷ lệ với trường hợp 80% train 20% test.

- Bước 2: Chọn giá trị k :

Tìm k lân cận để tính toán khoảng cách và tìm ra những phần tử giống có tính chất giống k .

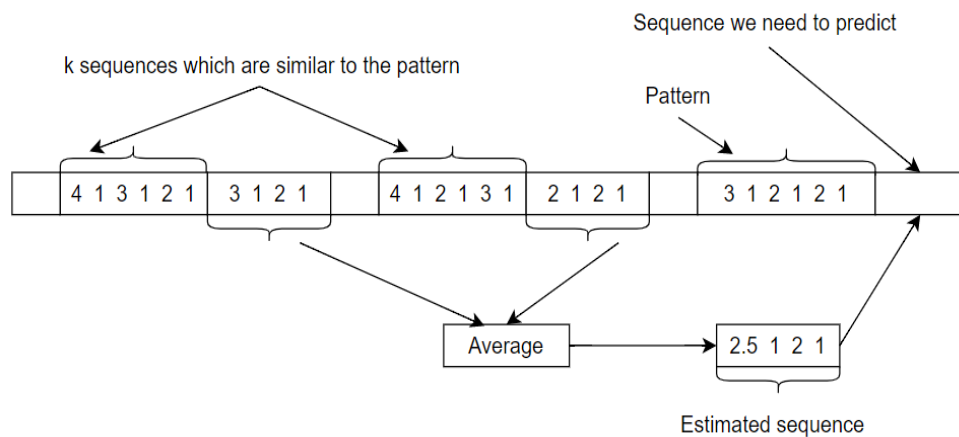
- Bước 3: Phân loại điểm dữ liệu

Với bài toán phân loại, dựa trên thành phần của phần tử xảy ra nhiều nhất. Với bài toán hồi quy thì sử dụng giá trị trung để xác định k gần nhất.

- Bước 4: Đánh giá:

Bằng các so sánh độ chính xác của kết quả bằng cách xem các dự đoán và ước tính của mô hình khớp với các lớp đã biết trong tập thử nghiệm đến mức nào.

2.2.3. Ứng dụng mô hình KNN



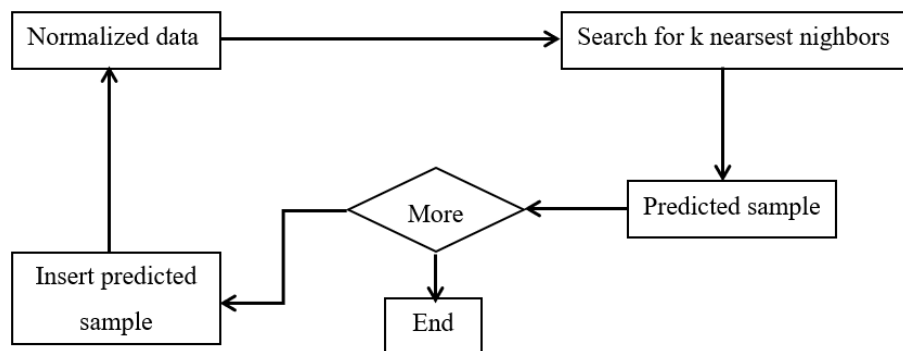
Hình 2.4 Hình minh họa về cách tiếp cận dựa trên KNN

Hình 2.4 Trong mô hình KNN, để có thể tiến hành dự đoán chuỗi thời gian, người dùng sẽ cần nhập vào k hàng xóm gần nhất, với giá trị k được nhập vào mô hình sẽ được tiến hành xem xét sự tương đồng giữa chuỗi mẫu với các chuỗi thời gian trong quá khứ có cùng chiều dài và tìm ra được k chuỗi giống với chuỗi mẫu nhất và bằng các độ đo Euclidean và Dynamic time warping, tiến hành chọn ra k chuỗi hậu tố có chiều dài bằng với <chiều dài của số dự đoán> của k chuỗi giống vừa tìm được và trung bình cộng các giá trị theo ngày trong k chuỗi hậu tố để cho ra kết quả output chuỗi cần dự đoán.

CHƯƠNG 3: MÔ HÌNH KẾT HỢP K-NEAREST NEIGHBORS VÀ FEEDFORWARD NEURAL NETWORK

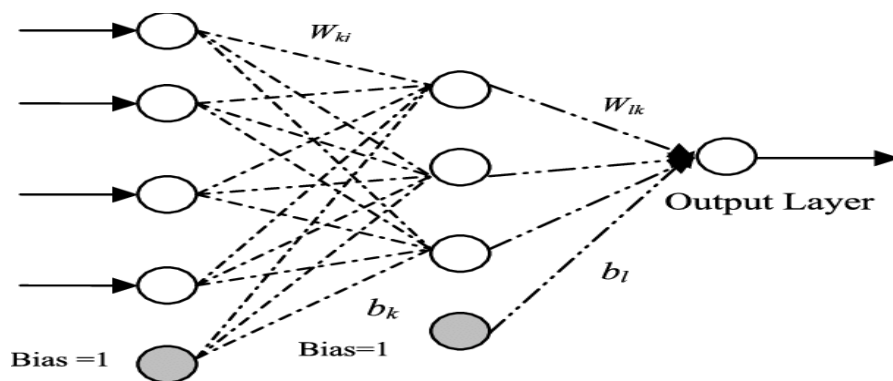
3.1. MÔ HÌNH LẠI GHÉP GIỮA KNN VÀ FFNN CHO BÀI TOÁN DỰ BÁO TRÊN CHUỖI THỜI GIAN

Trong lai ghép mô hình KNN và FFNN này, đề tài sử dụng mô hình KNN, [8] ý tưởng chính của kỹ thuật này là nhận dạng các mẫu trong quá khứ khớp với mẫu hiện hành và dùng tri thức về cách mà chuỗi thời gian biến đổi trong quá khứ trong những tình huống tương tự để dự báo về biến đổi trong tương lai.



Hình 3.1 Sơ đồ KNN cho bài toán dự báo chuỗi thời gian

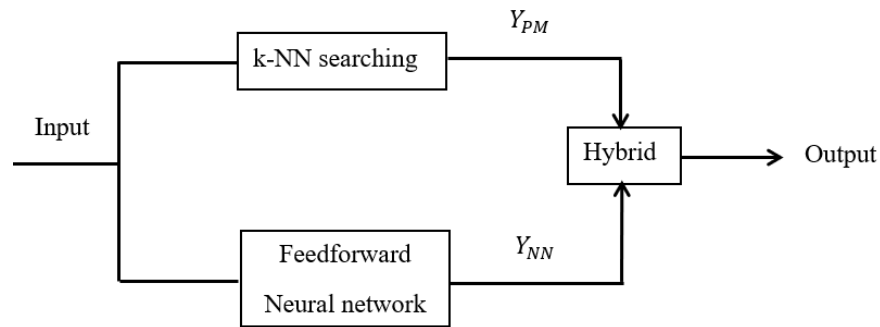
Mô hình FFNN, ý tưởng chính của kỹ thuật này là dự đoán giá trị của một mẫu theo thời gian, dựa vào các giá trị quá khứ của mẫu đó, các mẫu đầu vào được đưa vào thông qua các lớp đầu vào (Input-Layer), sau đó được xử lý thông qua các Hidden Layer và cuối cùng được dự đoán bởi lớp đầu ra, với các ý tưởng trên ta sẽ tiến hành xem xét đề xuất tính khả thi lai ghép hai mô hình dựa trên cách thực hiện lai ghép song song và lai ghép tuần tự.



Hình 3.2 Sơ đồ mô hình FFNN cho bài toán dự báo chuỗi thời gian

3.2. NGHIÊN CỨU MÔ HÌNH LAI GHÉP BẰNG CÁCH THỰC HIỆN SONG SONG HAI MÔ HÌNH KNN VÀ FFNN

Trong phương pháp này, kết quả dự đoán sẽ được thực hiện cho ra cùng lúc bằng cách thực thi song song hai mô hình K-Nearest Neighbors và Feedforward Neural Network và kết hợp kết quả dự đoán của cả hai mô hình bằng mô hình The hybrid predicting model.



Hình 3.3 Mô hình dự đoán Hybrid

Theo hình 3.3, dữ liệu đầu vào sẽ được cho vào cả hai mô hình KNN và FFNN. với mô hình KNN sẽ cho ra kết quả dự đoán chuỗi thời gian Forecast 1 và mô hình FFNN cùng lúc sẽ cho ra chuỗi thời gian Forecast 2 với hai kết quả của mô hình sẽ được cùng lúc cho vào mô hình hybrid và kết quả của hybrid là chuỗi thời gian dự đoán chính là output của toàn mô hình lai ghép. tại mô hình hybrid sẽ tiến hành kết hợp hai kết quả của mô hình KNN và FFNN [3].

$$Y_{hybird} = \omega Y_{NN} + (1 - \omega) Y_{PM} \quad (3.1)$$

Với Y_{PM} là kết quả của mô hình KNN, Y_{NN} là kết quả của mô hình FFNN và ω là trọng số của mô hình. nếu ω bằng không hoặc gần bằng không thì kết quả dự báo của KNN sẽ chiếm ưu thế hơn, nếu ω bằng một hoặc gần bằng một thì kết quả dự báo của FFNN sẽ chiếm ưu thế hơn [3].

Để có thể tìm ra được trọng số tốt nhất, ta sẽ tìm giá trị của ω để cho giá trị MSE đạt nhỏ nhất

$$MSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.2)$$

trong đó Y_i là giá trị thực tế ngày thứ i , $Y_{NN,i}$ là giá trị dự đoán ngày thứ i do mô hình FFNN dự đoán và $Y_{PM,i}$ là giá trị dự đoán ngày thứ i do mô hình KNN dự đoán, với phương trình bậc hai này sẽ cung cấp giá trị của ω làm cho lỗi dự báo MSE đạt mức nhỏ nhất [3].

$$\omega = \frac{\sum_{i=1}^n (Y_{NN,i} - Y_{PM,i})(Y_i - Y_{PM,i})}{\sum_{i=1}^n (Y_{NN,i} - Y_{PM,i})^2} \quad (3.3)$$

Trong đó:

Y_i : giá trị thực tế của một ngày trên chuỗi thời gian

$Y_{NN,i}$: giá trị dự đoán ngày thứ i của mô hình FFNN

$Y_{PM,i}$: giá trị dự đoán ngày thứ i của mô hình KNN

3.3. TUẦN TỰ THEO MÔ HÌNH CỘNG GIỮA KNN VÀ FFNN

3.3.1. KNN và FFNN

Zhang [2] đề xuất mô hình lai ghép hybrid với giả định chuỗi thời gian y_t là tổng của kết quả mô hình dự đoán tuyến tính và phi tuyến.

$$y_t = L_t + N_t \quad (3.4)$$

KNN được sử dụng trong mô hình này đại diện cho phần tuyến tính \hat{L}_t của \hat{y}_t và dự đoán \hat{L}_t là

$$\hat{L}_t = \frac{\sum_{i=1}^n (Y_i)}{n} \quad (3.5)$$

Các dự báo tuyến tính từ mô hình KNN được trừ khỏi chuỗi gốc y_t đã được chỉ ra ở phương trình (3.5). Zhang [2] nhấn mạnh rằng phần chênh lệch giữa chuỗi gốc với chuỗi

dự báo KNN này là các giá trị phi tuyến tính, vì mô hình KNN chỉ khớp với các giá trị tuyến tính.

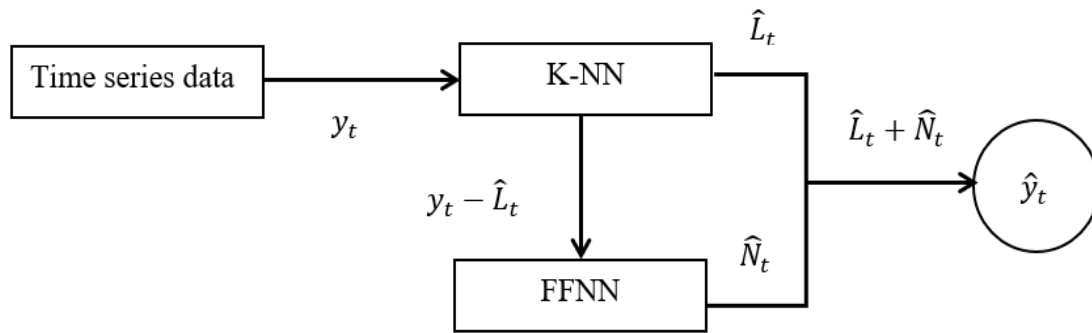
$$N_t = y_t - \hat{L}_t \quad (3.6)$$

Trong đó giá trị N_t chuỗi chênh lệch và sẽ được dự đoán bằng FFNN và kết quả dự đoán sẽ thu được bằng cách sử dụng phương trình.

$$\hat{N}_t = g \left(\sum_{i=1}^n w_{ji} \times x_i + b_j \right) \quad (3.7)$$

Kết quả dự báo mô hình lai ghép sẽ thu được bằng tổng các dự báo của KNN (3.5) và dự đoán FFNN phần chênh lệch ở phương trình (3.7).

$$\hat{y}_t = \hat{L}_t + \hat{N}_t \quad (3.8)$$



Hình 3.4 Sơ đồ mô hình lai ghép tuần tự KNN-FFNN

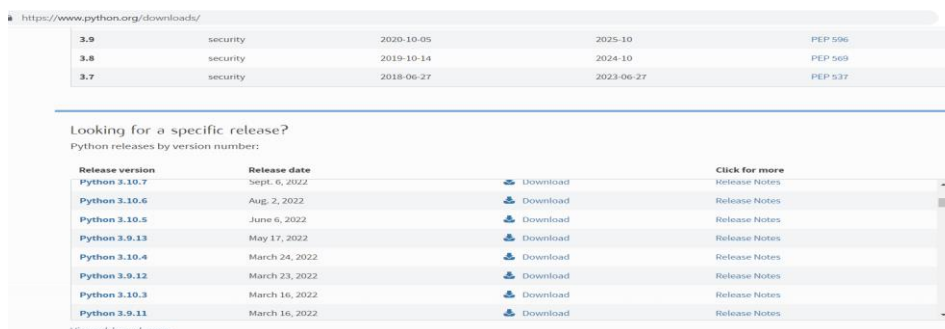
CHƯƠNG 4: CÀI ĐẶT

4.1. CÀI ĐẶT PHẦN MỀM

4.1.1. Cài đặt Python

- Bước 1: Tải Python executable

Vào link <https://www.python.org/downloads/> chọn file Python 3.9.13 vào tải về máy.



Hình 4.1 Trang download Python

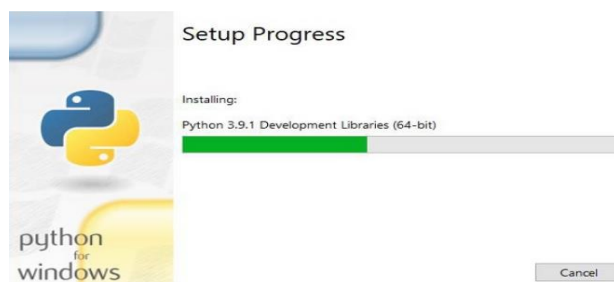
Chọn bộ cài đặt phù hợp với phiên bản hệ điều hành của máy đang sử dụng (Windows 10 64 bit).

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball		Source release	ea1da83543bad127c4def4d288fdab87	26355887	SIG
XZ compressed source tarball		Source release	5e2411217b0060828d5f923eb422a3b8	19754368	SIG
macOS 64-bit Intel-only installer	macOS	for macOS 10.9 and later, deprecated	671848930809decf27f586ddf98c6e9b	30997161	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	76b63cf623e32cdf27c5033434bd69ce	38821163	SIG
Windows embeddable package (32-bit)	Windows		fec0bc06857502a56dd1aeaea6488ef8	7729405	SIG
Windows embeddable package (64-bit)	Windows		57731cf80b1c429a0be7133266d7d7cf	8570740	SIG
Windows help file	Windows		c86feba059b340a1de2a9d2ee7059a6d	8953644	SIG
Windows installer (32-bit)	Windows		46c35b0a2a4325c275b2ed3187b08ac4	28096840	SIG
Windows installer (64-bit)	Windows	Recommended	e7062b85c3624af82079794729618eca	29235432	SIG

Hình 4.2 Trang chọn bộ cài đặt

Bước 2: Chạy file executable

Chạy các file đã tải về, chọn vào mục Python 3.9 to PATH sau đó chọn Install để cài đặt vào máy.

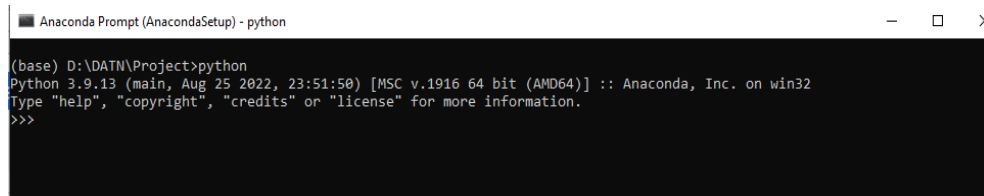


Hình 4.3 Màn hình cài đặt Python

Sau khi cài đặt thành công màn hình sẽ hiện như sau:

- Bước 3: Xác minh Python được cài đặt

Kiểm tra Python cài đặt trên máy tính một cách chắc chắn thực hiện câu lệnh ‘python’ trên cmd, nếu cài đặt chính xác thì màn hình cmd hiển thị như sau:

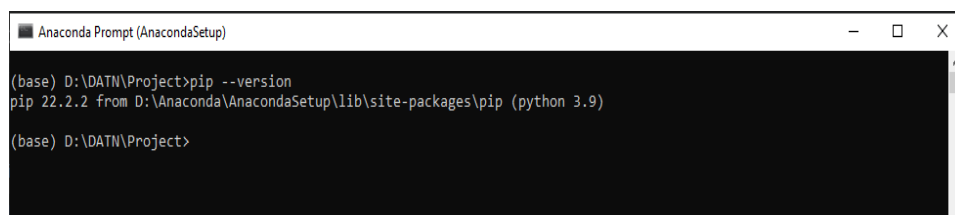


```
Anaconda Prompt (AnacondaSetup) - python
(base) D:\DATN\Project>python
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Hình 4.4 Màn hình kiểm tra phiên bản Python

- Bước 4: Xác minh Pip được cài đặt

Pip là một hệ thống quản lý package mạnh mẽ vì vậy nên đảm bảo Pip đã được cài đặt bằng cách thực hiện câu lệnh ‘pip -v’ trên cmd, nếu đã có pip trên máy thì màn hình cmd hiển thị như sau:



```
Anaconda Prompt (AnacondaSetup)
(base) D:\DATN\Project>pip --version
pip 22.2.2 from D:\Anaconda\AnacondaSetup\lib\site-packages\pip (python 3.9)
(base) D:\DATN\Project>
```

Hình 4.5 Màn hình kiểm tra phiên bản Pip trong Python

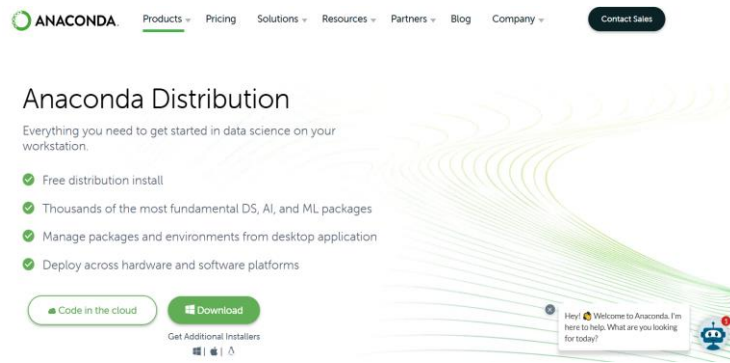
4.1.2. Cài đặt Anaconda

Anaconda là nền tảng phân phối các ngôn ngữ lập trình như Python, R phục vụ cho việc tính toán khoa học (Data science, machine learning, big data processing, predictive analytics,...). Phục vụ trên nhiều hệ điều hành như: Windows, MacOS và Linux.

Cài đặt Anaconda:

- Đến trang web Anaconda theo đường link:

<https://www.anaconda.com/products/distribution> chọn phiên bản phù hợp rồi nhấn Download.



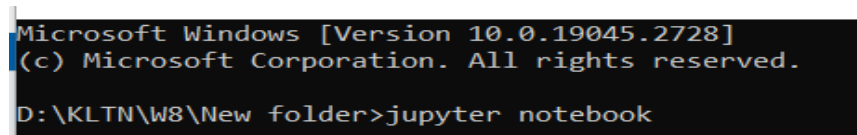
Hình 4.6 Trang tải Anaconda

- Mở file Setup vừa tải từ trang web và cài đặt theo trình tự Next → Agree → Next → Install → Finish.

Jupyter Notebook là nền tảng tính toán khoa học với khả năng nổi bật cho phép tương tác với từng dòng code, hỗ trợ kernel cho các ngôn ngữ khác nhau như: Python, R, Julia...

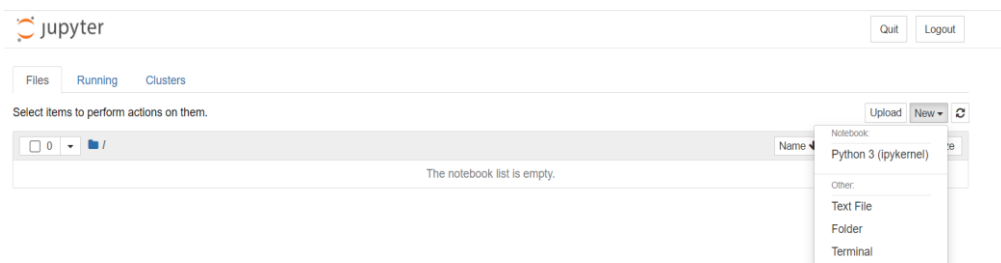
Cài đặt Jupyter Notebook:

- Sau khi cài đặt Anaconda người dùng có thể chạy Jupyter Notebook đã được thêm vào menu Start của Windows hoặc chạy trực tiếp trên cmd bằng lệnh jupyter notebook.



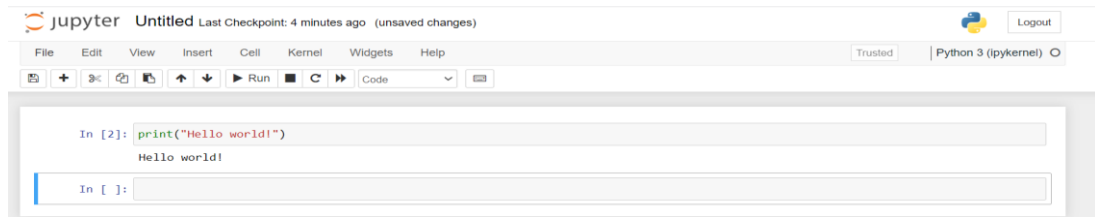
Hình 4.7 Mở Jupyter notebook bằng cmd

- Sau khi chạy jupyter notebook nhấn vào New chọn Python 3 để tạo file Python.



Hình 4.8 Tạo thư mục trên Jupyter notebook

- Sau khi tạo file và gõ câu lệnh `print("Hello world!")`, chương trình chạy cho ra kết quả “Hello world!” thì thành công.



Hình 4.9 Chạy thử chương trình trên Jupyter notebook

4.2. THU VIỆN

Các thư viện sử dụng để cài đặt demo chương trình gồm:

4.2.1. Loại bỏ Warning

Trong quá trình sử dụng các hàm, các thư viện của python trên jupyter notebook sẽ thường xuất hiện các warning nhưng không ảnh hưởng đến code nên để loại bỏ các warning này sử dụng hàm `filterwarnings()` trong thư viện `warnings`

```
import warnings

warnings.filterwarnings('ignore')
```

Với hàm `filterwarnings()` trong thư viện `warnings`, truyền vào chuỗi string là `ignore`, python sẽ tự động loại bỏ tất cả các warnings.

4.2.2. Đọc dữ liệu

Để có thể tiến hành đưa dữ liệu vào mô hình, cần phải chuyển đổi dữ liệu sang định dạng thích hợp cho quá trình dự đoán, để giải quyết vấn đề này sẽ được hàm `read_csv()` trong thư viện `pandas` hỗ trợ

```
import pandas as pd

dataCSV = pd.read_csv(<filePath>)
```

Việc sử dụng hàm `read_csv()` sẽ được truyền vào đường dẫn của tập dữ liệu dưới dạng CSV và python sẽ đọc file csv sau đó chuyển đổi sang dataframe, dưới dạng dataframe do thư viện hỗ trợ này người dùng sẽ dễ dàng sử dụng và thao tác hơn.

4.2.3. Tiền xử lý dữ liệu

Đối với các dữ liệu được truyền vào python sẽ được gọi là các dữ liệu thô, các dữ liệu này sẽ chứa rất nhiều dữ liệu chưa được sạch như Null, empty,... ảnh hưởng đến quá trình dự đoán, để khắc phục được các trường hợp này sẽ tiến hành sử dụng các hàm trong thư viện pandas để làm sạch dữ liệu bao gồm:

- Hàm `fillna()`:

```
import pandas as pd

dataCSV.fillna(0, inplace=True)
```

Sử dụng hàm `fillna(<param replace>, inplace=True)` trong thư viện pandas, python sẽ tiến hành thay đổi các giá trị NAN thành giá trị 0. Điều này sẽ đảm bảo rằng ta có đủ dữ liệu để làm việc, và không bị ảnh hưởng bởi dữ liệu NAN

- Hàm `std()` và `mean()`

```
import pandas as pd

std = data[<ColumnTimeSeries>].std()

mean = data[<ColumnTimeSeries>].mean()

data[<ColumnTimeSeries>]=np.where(data[<ColumnTimeSeries>]>
(mean + <n>*std), mean, data[<ColumnTimeSeries>])

data[<ColumnTimeSeries>]=np.where(data[<ColumnTimeSeries>]<
(mean - <n>*std), mean, data[<ColumnTimeSeries>])
```

Sử dụng hàm `std()` để tiến hành lấy ra được giá trị độ lệch chuẩn trên toàn tập dữ liệu, tiếp theo sử dụng hàm `mean()` để lấy ra giá trị trung bình của toàn tập dữ liệu, với hai giá trị vừa tìm được sẽ được tiến hành khử các giá trị outlier bằng cách thay thế các giá trị nằm ngoài khoảng `<n>` độ lệch chuẩn của dữ liệu với giá trị trung bình của dữ liệu.

- Hàm `MinMaxScaler()`:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
```

```
data[<ColumnTimeSeries>].scaler.fit_transform(data[<ColumnTimeSeries>].values.reshape(-1, 1))
```

Trên tập dữ liệu thô, các giá trị của tập dữ liệu sẽ có sự chênh lệch rất lớn, do đó hàm `MinMaxScaler().fit_transform` của thư viện `sklearn` của thư viện `python` sẽ hỗ trợ được sử dụng để đưa toàn bộ các giá trị trong tập dữ liệu về cùng một phạm vi giá trị từ không tới một điều này sẽ mang lại tính nhất quán cho tập dữ liệu và làm tăng khả năng dự đoán của các mô hình.

- Hàm `interpolate()`:

```
data[<ColumnTimeSeries>]=data[<ColumnTimeSeries>].interpolate(method='linear')
```

Với hàm `interpolate()` này được sử dụng để interpolate các giá trị còn thiếu trong tập dữ liệu bằng phương pháp tuyến tính (linear). Nếu một giá trị bị thiếu, `interpolate()` sẽ tính toán giá trị tại vị trí đó dựa trên các giá trị xung quanh bằng cách sử dụng phương pháp tuyến tính.

4.2.4. Trích xuất chuỗi con

Dữ liệu sau khi được thư viện `pandas` đọc vào sẽ là một chuỗi thời gian kéo dài từ ngày đầu tiên của tập dữ liệu đến ngày cuối cùng, với dữ liệu như vậy sẽ không thể tiến hành đưa vào mô hình để cho ra dự đoán chính xác, để khắc phục điều này ta sẽ sử dụng phương pháp trích xuất chuỗi con trên tập dữ liệu và sử dụng hàm `array` của thư viện `numpy` để đưa các chuỗi con được trích xuất vào mảng

```
import numpy as np

def prepare_data(<tập dữ liệu>, <chiều dài của sổ>,<chiều dài của sổ dự đoán>,<số ngày trượt>):

    X, y = [], []

    startWindow = 0

    for i in range(len(data) - size_window - 1):

        if (len(data[(startWindow + size_window):(startWindow + size_window + size_predict) , 0]) != size_predict):
```

```

        break

    X.append(data[startWindow:(startWindow + size_window),
:]))

    y.append(data[(startWindow + size_window):(startWindow +
size_window + size_predict) , 0])

    startWindow += stepWindow

return np.array(X), np.array(y)

```

Hàm `prepare_data()` sẽ được truyền vào các tham số như chiều dài cửa sổ, chiều dài ngày dự đoán, số lượng ngày trượt, cùng với các tham số hàm sẽ được thực thi tiến hành trích xuất các chuỗi con tương ứng và chuyển các chuỗi con trích xuất được vào mảng bằng cách sử dụng hàm `array` của thư viện `numpy`.

4.2.5. Mô hình K-nearest Neighbors

Trong mô hình KNN, input đầu vào sẽ là mảng các chuỗi con được trích xuất và sử dụng các độ đo Ecclidean và DTW để tiến hành đánh giá sự tương đồng giữa chuỗi mẫu với các chuỗi trong quá khứ để tìm ra được k chuỗi tương đồng sau đó lấy n ngày dự đoán sau k chuỗi tương đồng trong quá khứ để tính trung bình cộng, kết quả phép tính này sẽ là kết quả dự đoán của KNN, để thực hiện được quá trình trên cần sử dụng các hàm của thư viện sau:

- Hàm `dtw()`:

```

import numpy as np

from dtw import dtw

dist, _, _, _ = dtw(<chuỗi mẫu>, <chuỗi trong quá khứ>,
dist=lambda <chuỗi mẫu>, <chuỗi trong quá khứ>: np.abs(<chuỗi mẫu> -
<chuỗi trong quá khứ>))

```

Trong hàm `dtw()` của thư viện `dtw` sẽ nhận vào các tham số sẽ hỗ trợ tính khoảng cách giữa 1 điểm trên <chuỗi mẫu> với các điểm trên <chuỗi trong quá khứ> và liên tục lặp lại đến khi tìm ra được đường đi ngắn nhất giữa 2 chuỗi.

- Hàm `flatten()` và `euclidean()`:

```
from scipy.spatial.distance import euclidean

import numpy as np

def euclidean_distance(<chuỗi 1>, <chuỗi 2>):
    ts1= <chuỗi 1>.flatten()
    ts2= <chuỗi 2>.flatten()

    return euclidean(ts1,ts2)
```

Trong hàm `euclidean_distance()` sẽ nhận vào hai tham số chuỗi mẫu và chuỗi được xét trong quá khứ, hai tham số này đang ở dạng nhiều chiều (list lồng list) cần chuyển sang dạng một chiều bằng hàm `flatten()` của thư viện `numpy`, với hai tham số sẽ được chuyển đổi về dạng một chiều này và tiến hành tính khoảng cách giữa chúng bằng hàm `euclidean()` của thư viện `scipy`, hàm `euclidean()` này sẽ thực hiện tính khoảng cách Euclidean bằng cách:

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (4.1)$$

Trong đó: x_i : giá trị ngày thứ i của chuỗi trong quá khứ

y_i : giá trị ngày thứ i của chuỗi trong mẫu

d : khoảng cách (độ tương đồng giữa hai chuỗi)

- Hàm `argsort()` :

```
import numpy as np

def kSimilarityTimeSeries(k, <mảng khoảng cách>):
    distances = np.array(<mảng khoảng cách>)

    return distances.argsort()[:k]
```

Trong Hàm `kSimilarityTimeSeries()` sẽ nhận vào hai tham số k chuỗi trong quá khứ tương đồng và `<mảng khoảng cách>` được tính bằng Eculidean hoặc DTW của chuỗi mẫu với các chuỗi được xét trong quá khứ, với hai tham số này sẽ được hàm

`argsort()` của thư viện `numpy` cho ra kết quả output k chuỗi trong quá khứ tương đồng với chuỗi mẫu nhất.

- Hàm `mean()` :

```
import numpy as np

y_pred = np.mean(<mảng vị trí <chiều dài của số dự đoán> ngày
sau của k chuỗi tương đồng>)
```

Trong hàm `mean()` của thư viện `numpy` sẽ được truyền vào tham số mảng vị trí của <chiều dài của số dự đoán> ngày sau của k chuỗi tương đồng với chuỗi mẫu và tiến hành tính trung bình cộng.

4.2.6. Mô hình FeedForward Neural Network

4.2.6.1. Quá trình Train:

Trong quá trình train mô hình FFNN sẽ tiến hành train trên tập dữ liệu và tự cập nhật trọng số sau mỗi lần lặp để tìm ra các tham số tốt nhất giúp tăng khả năng mô dự đoán chính xác cho mô hình. Đối với quá trình train mô hình FFNN sẽ sử dụng các hàm của thư viện sau:

+ Hàm `Sequential()` của thư viện `keras`:

```
from keras.models import Sequential

def create_model(num_layers_hidden=1, num_neurons_hidden=1):

    model = Sequential()

    for i in range(num_layers_hidden):

        if i == 0:

            model.add(Dense(num_neurons_hidden, input_dim=size_window, activation='sigmoid'))

        else:

            model.add(Dense(num_neurons_hidden, activation='sigmoid'))

    model.add(Dense(1))
```

```
model.compile(loss='mean_squared_error',optimizer='adam')

return model
```

Trong hàm `create_model()` khi được gọi tới sẽ nhận vào các tham số chính là số lượng lớp ẩn, số lượng neural đối với một lớp ẩn và tiến hành tạo mô hình FFNN với ba layer được thêm vào bởi hàm `add()` của thư viện `keras`, trong đó input layer sẽ có <kích thước cửa sổ> neural, hidden layer sẽ có <số lượng neural Hidden> neural và có thể tăng xác định số lớp ẩn theo tham số truyền vào, output layer sẽ là một neural.

- Hàm `GridSearchCV()`:

```
from sklearn.model_selection import GridSearchCV

grid_FFNN = GridSearchCV(estimator=<mô hình FFNN>, param_grid=<tham số cần duyệt>,
cv=3)
```

Trong hàm `GridSearchCV()` của thư viện `sklearn`, tham số truyền vào sẽ gồm <mô hình FFNN>, <tham số cần duyệt>, và sử dụng phương pháp cross validation = 3 để chia tập dữ liệu làm 3 phần sau đó thực hiện train và test 3 lần, trong đó mỗi lần sử dụng một phần dữ liệu khác nhau làm tập kiểm tra và các phần còn lại làm tập huấn luyện. Kết quả đánh giá của ba lần kiểm tra chéo này sẽ được kết hợp lại để đưa ra đánh giá tổng thể về hiệu suất của mô hình đối với tham số đang xét.

- Hàm `save_weights()`:

```
from keras.models import Sequential

<Mô hình>.save_weights('<filePath>.h5')
```

Trong hàm `Sequential()` của thư viện `keras` hỗ trợ hàm `save_weights()` cho người dùng lưu dữ liệu trọng số tốt nhất vào đường dẫn thư mục được chỉ định.

4.2.6.2. Quá trình test

Đối với quá trình test của mô hình FFNN sẽ nhận input đầu vào là dữ liệu cần dự đoán cùng với các tham số, trọng số tốt nhất tìm được ở quá trình train, để load lại được các trọng số ở quá trình train sẽ sử dụng hàm `load_weights()` của thư viện `keras`

- Hàm `load_weights()`:

```
from keras.models import Sequential

<Mô hình>.load_weights('<filePath>.h5')
```

Trong hàm `Sequential()` của thư viện `keras` hỗ trợ hàm `load_weights()` cho người dùng load lại dữ liệu trọng số tốt nhất dựa theo đường dẫn thư mục được chỉ định.

4.2.7. Mô hình lai ghép song song

Trong mô hình lai ghép song song này mục tiêu là tìm ra ω để lỗi dự báo của mô hình đạt giá trị nhỏ nhất, với giá trị ω tìm được sẽ được sử dụng để tiến hành kết hợp hai mô hình lại theo công thức (3.1) và sử dụng hàm `array` của thư viện `numpy` để lưu giá trị dự báo.

```
import numpy as np

def predictHybrid(y_pred_FFNN, y_pred_KNN, y_test):

    FFNNSubKNN=[]

    TestSubKNN=[]

    weightEl=[]

    for i in range(len(y_pred_FFNN)):

        FFNNSubKNN.append(y_pred_FFNN[i]-
y_pred_KNN[i])

        TestSubKNN.append(y_test[i]-y_pred_KNN[i])

    for j in range(len(FFNNSubKNN)):

        weightEl.append(((FFNNSubKNN[j]*TestSubKNN[j])/
(FFNNSubKNN[j]*FFNNSubKNN[j])))

    weight = np.array(weightEl)

    y_pred_combine=[]

    for i in range(len(weight)):

        y_pred_combine.append(weight[i]*y_pred_FFNN[i]+(1-
weight[i])*y_pred_KNN[i])

    return np.array(y_pred_combine)
```

Trong đoạn code trên là quá trình tính ω được thực hiện dựa theo công thức:

$$\omega = \frac{\sum_{i=1}^n (Y_{NN,i} - Y_{PM,i})(Y_i - Y_{PM,i})}{\sum_{i=1}^n (Y_{NN,i} - Y_{PM,i})^2} \quad (4.2)$$

Trong đó:

Y_i : giá trị thực tế của một ngày trên chuỗi thời gian

$Y_{NN,i}$: giá trị dự đoán ngày thứ i của mô hình FFNN

$Y_{PM,i}$: giá trị dự đoán ngày thứ i của mô hình KNN

kết quả ω sẽ được thêm vào mảng array nhờ vào hàm `append()` của thư viện `numpy` và mảng ω này sẽ được đưa vào để dự đoán công thức:

$$Y_{hybird} = \omega Y_{NN} + (1 - \omega) Y_{PM} \quad (4.3)$$

Kết quả của công thức trên cũng chính là kết quả output cho mô hình lai ghép song song, giúp cho mô hình lai ghép đạt được giá trị nhỏ nhất [3].

4.2.8. Mô hình lai ghép tuần tự cộng

Trong mô hình lai ghép tuần tự cộng này sẽ sử dụng hàm array của thư viện `numpy` để lưu trữ các chuỗi dự báo do mô hình lai ghép dự báo.

```
import numpy as np

for i in range(len(<mảng >)):

    pred_knn_ffnn.append(<mảng chuỗi dự báo KNN>[i]+<mảng chuỗi
    dự báo lỗi FFNN>[i])

pred_knn_ffnn=np.array(pred_knn_ffnn)
```

Với mảng chuỗi giá trị dự báo của KNN và mảng chuỗi giá trị dự báo lỗi của FFNN sẽ được tiến hành cộng với nhau, kết quả cộng cũng chính là kết quả dự báo của mô hình lai ghép sẽ được đưa vào mảng bằng cách sử dụng hàm `array` trong thư viện `numpy`.

4.2.9. Đánh giá độ chính xác

Sau khi các mô hình hoạt động và cho ra các giá trị dự đoán, việc quan trọng tiếp theo là đánh giá độ chính xác sẽ cho biết mức độ chính xác và tối ưu của mô hình. Để

đảm bảo đánh giá mô hình một cách tổng quát và khách quan nên đánh giá mô hình trên nhiều chỉ số như: MAE, MSE, RMSE, MAPE.

- Hàm `mean_squared_error()`:

```
from sklearn.metrics import mean_squared_error

mse_KNN_FFNN= mean_squared_error(<giá trị thực tế>, <giá trị dự đoán>)
```

Sử dụng hàm `mean_squared_error()` để ước lượng sai số bình phương giữa giá trị thực tế và giá trị dự đoán bằng cách lấy khoảng cách từ các điểm thực tế đến các điểm dự đoán (khoảng cách này gọi là sai số) và bình phương.

- Hàm `mean_absolute_error()`:

```
from sklearn.metrics import mean_absolute_error

mae_KNN_FFNN= mean_absolute_error(<mảng chuỗi giá trị thực tế>,<mảng chuỗi giá trị dự đoán>)
```

Hàm `mean_absolute_error()` đo độ chênh lệch trung bình của giá trị thực tế với của giá trị dự đoán bằng cách lấy giá trị trung bình của khoảng cách các giá trị thực tế và giá trị dự đoán của mô hình.

-Hàm `sqrt()` và `mean_squared_error()`:

```
from sklearn.metrics import mean_squared_error

from math import *

rmse_KNN_FFNN= sqrt(mean_squared_error(<mảng chuỗi giá trị thực tế>,<mảng chuỗi giá trị dự đoán>))
```

RMSE là độ đo đánh giá sự chênh lệch của giá trị thực tế với giá trị dự đoán bằng cách thực hiện căn bậc hai của sai số bình phương giá trị thực tế và giá trị dự đoán, trong đó hàm `mean_squared_error()` của thư viện `sklearn` sẽ nhận vào hai tham số là chuỗi dự đoán và chuỗi thực tế để tiến hành tính sai số bình phương giữa hai tham số này, kết quả sẽ được căn bậc hai bằng hàm `sqrt()` của thư viện `math` kết quả của căn bậc hai này sẽ là kết quả `sqrt()`.

- Hàm `mean_absolute_percentage_error()` :

```
from sklearn.metrics import mean_absolute_percentage_error

mape_combine = mean_absolute_percentage_error(<mảng chuỗi giá trị thực tế>,<mảng chuỗi giá trị dự đoán>)
```

Hàm `mean_absolute_percentage_error()` của thư viện `sklearn` hỗ trợ tính toán MAPE đây là phần trăm sai số trung bình tuyệt đối, chỉ số này giúp đo độ chính xác theo phần trăm và có thể được tính bằng cách lấy sai số phần trăm tuyệt đối trung bình cho mỗi khoảng thời gian trừ đi các giá trị thực chia cho các giá trị thực.

4.2.10. Thời gian thực thi

Trong bài toán dự đoán để đánh giá một mô hình được áp dụng một cách hiệu quả hay không ngoài việc dựa vào đánh giá độ chính xác của mô hình thì còn dựa vào thời gian thực thi của mô hình. Để tính toán thời gian thực thi mô hình một cách chính xác thì việc dùng thư viện `time` trong python sẽ là một sự lựa chọn phù hợp.

```
import time

start_<mô hình> = time.time()

end_<mô hình> = time.time()
```

Hàm `time()` trong thư viện `time` của python hỗ trợ người dùng trong việc xác định thời gian, trong mỗi mô hình hàm `time()` sẽ được gọi hai lần khi thực thi một mô hình dự báo, trong đó lần gọi hàm `time()` đầu tiên sẽ được xem xét là thời gian bắt đầu dữ liệu được đưa vào mô hình và lần gọi hàm `time()` thứ hai sẽ được xem xét là thời gian kết thúc quá trình dự đoán, với hai kết quả thời gian từ hai lần gọi trừ nhau sẽ là tổng thời gian thực thi của một mô hình.

CHƯƠNG 5: ĐÁNH GIÁ BẰNG THỰC NGHIỆM PHƯƠNG PHÁP DỰ BÁO CHUỖI THỜI GIAN KNN - FFNN

Mô hình kết hợp cải tiến song song và tuần từ này được sẽ được thực nghiệm trên hai bộ dữ liệu gồm bốn tập dữ liệu thực tế. Qua phần thực nghiệm, ta sẽ so sánh kết quả dự báo của mô hình kết hợp liệu có cải tiến hơn so với các mô hình bình thường.

5.1. MÔI TRƯỜNG THỰC NGHIỆM

5.1.1. Phần cứng

Trong đề tài này, các mô hình sẽ được thực hiện đánh giá thực nghiệm trên máy tính của hãng Toshiba, tên thiết bị là DESKTOP-DAF3E09, với bộ lõi xử lý intel(R) Core(TM) i7-6500U CPU, tốc độ xung nhịp (tốc độ xử lý) đạt 2.60 GHz, thông số ram lên tới 12.0 GB, gồm 2 core và số luồng xử lý 4.

5.1.2. Phần mềm

Với quá trình thực nghiệm mô hình kết hợp sẽ được sử dụng thực nghiệm trên ngôn ngữ lập trình Python đây là một ngôn ngữ lập trình cho phép người dùng làm việc nhanh chóng, tích hợp đa dạng các thư viện hỗ trợ hiệu quả cho quá trình thực nghiệm và code sẽ được thực hiện trên phần mềm jupyter notebook chạy bằng python được cung cấp bởi Anaconda đây là nền tảng phân phối các ngôn ngữ lập trình như Python, R, là nền tảng phục vụ cho việc tính toán khoa học và hỗ trợ trên nhiều hệ điều hành.

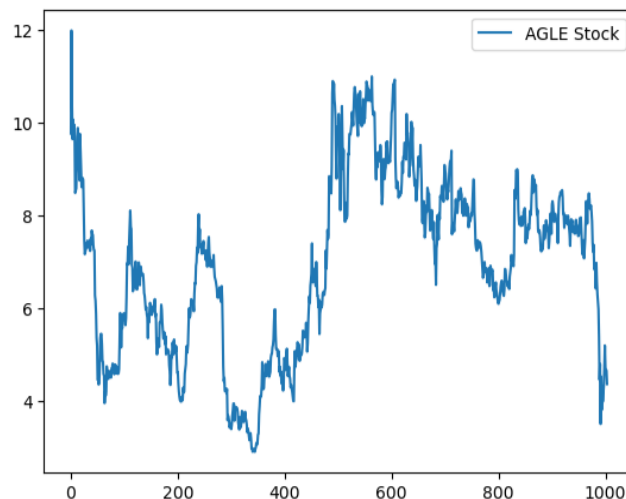
5.2. DỮ LIỆU THỰC NGHIỆM

Trong đề tài này, dữ liệu thực nghiệm sẽ gồm 4 tập dữ liệu AGLE (Aeglea BioTherapeutics), GDDY (GoDaddy Inc), AVAL (Grupo Aval Acciones), MDLY (Medley Management Inc). Các tập dữ liệu này được sử dụng rộng rãi trên trang kaggle, đây là một trong những bộ dữ liệu được cộng đồng mạng đánh giá cao và tích cực tham gia [\[1\]](#).

Về tổng quan Bộ dữ liệu này chứa giá lịch sử hàng ngày của tất cả các mã hiện đang giao dịch trên NASDAQ và được cập nhật từ nasdaqtrader.com. Các dữ liệu sẽ được lấy từ Yahoo tài chính thông qua gói python yfinance. Bộ dữ liệu lưu trữ thông tin theo ngày và có 7 thuộc tính:

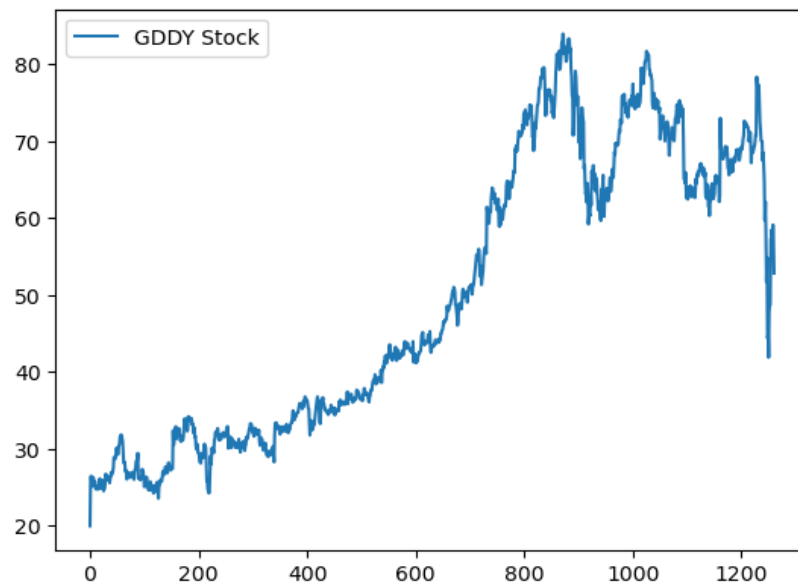
- Date: xác định ngày giao dịch.
- Open: giá mở cửa.
- High: giá tối đa trong ngày.
- Low: giá tối thiểu trong ngày.
- Close: giá đóng được điều chỉnh cho các lần chia tách.
- Adj Close: giá đóng được điều chỉnh cho cả cổ tức và chia tách.
- Volume: số lượng cổ phiếu đã đổi chủ trong một ngày nhất định.

Tập dữ liệu AGLE (Aeglea BioTherapeutics) chứa lịch sử giá cổ phiếu gồm 1004 ngày trong 4 năm liên theo từng ngày, bắt đầu từ ngày 07/04/2016 đến ngày 01/04/2020, tại hình 5.1 đang thể hiện mã chứng khoán có sự biến động tăng giảm liên tục không theo bất kỳ tính chất nào.



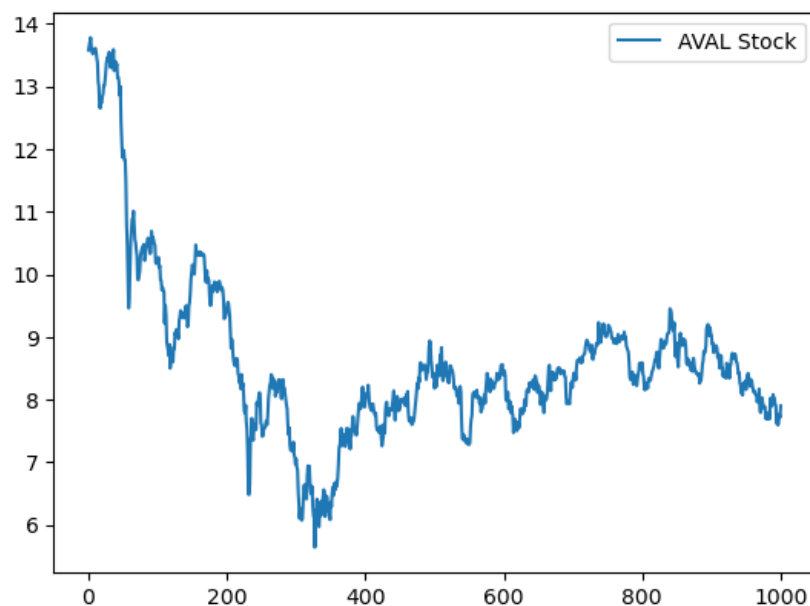
Hình 5.1 Mô tả dữ liệu chứng khoán AGLE

Tập dữ liệu GDDY (GoDaddy Inc) chứa lịch sử giá cổ phiếu gồm 1261 ngày trong 5 năm liên theo từng ngày, bắt đầu từ ngày 31/03/2015 đến ngày 01/04/2020, tại hình 5.2 đang thể hiện mã chứng khoán có xu hướng đi lên và biến động trong tăng giảm liên tục.



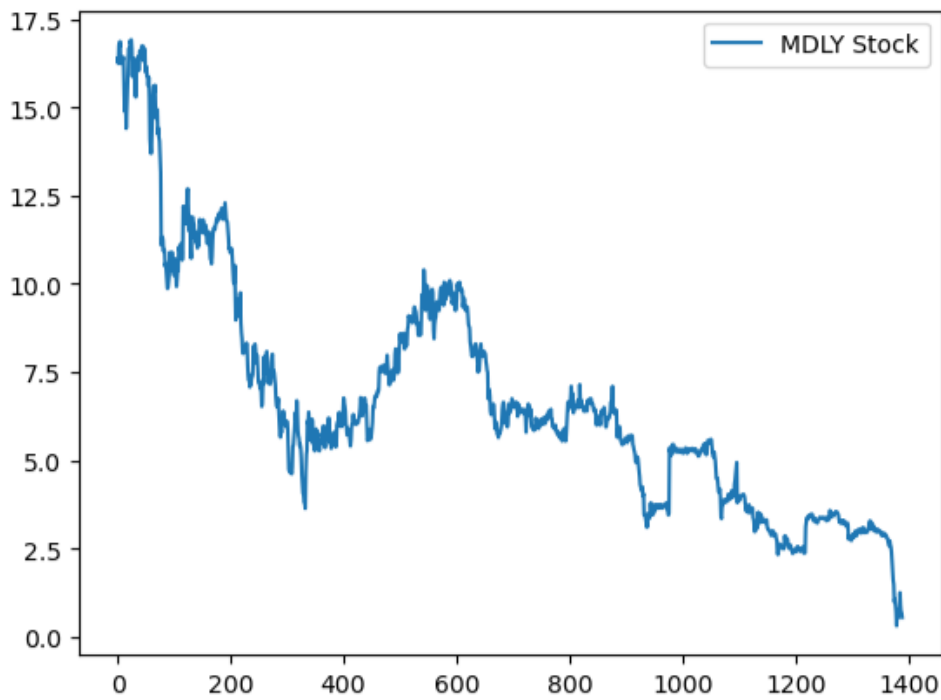
Hình 5.2 Mô tả dữ liệu chứng khoán GDDY

Tập dữ liệu AVAL (Grupo Aval Acciones) chứa lịch sử giá cổ phiếu gồm 1001 ngày trong 4 năm liên tiếp theo từng ngày, bắt đầu từ ngày 23/09/2014 đến ngày 12/09/2018, tại hình 5.3 đang thể hiện mã chứng khoán có xu hướng đi giảm dần và giữ ổn định.



Hình 5.3 Mô tả dữ liệu chứng khoán AVAL

Tập dữ liệu MDLY (Medley Management Inc) chứa lịch sử giá cổ phiếu gồm 1930 ngày trong 6 năm liên tiếp theo từng ngày, bắt đầu từ ngày 24/09/2014 đến ngày 01/04/2020, tại hình 5.4 đang thể hiện mã chứng khoán có xu hướng đi giảm dần.



Hình 5.4 Mô tả dữ liệu chứng khoán MDLY

Dựa vào bốn tập dữ liệu với sự biến động khác nhau sẽ được tiến hành thực nghiệm thuật toán KNN, FFNN, mô hình lai ghép song song và mô hình lai ghép tuần tự cộng để đưa ra dự đoán cho tập dữ liệu và đánh giá mức độ hiệu quả của việc lai ghép có được cải thiện hơn so với mô hình đơn và tìm ra được phương pháp lai ghép tốt, mang đến kết quả dự đoán sát với thực tế nhất.

5.3. TIÊU CHÍ ĐÁNH GIÁ

Một mô hình dự báo được đánh giá tốt khi sai số dự báo nhỏ. Ngoài ra tính ngẫu nhiên của sai số cũng là một tham số quan trọng để đánh giá độ chính xác của dự báo. Khi tiến hành dự báo người ta thường giả định dữ liệu ban đầu ngẫu nhiên; các tính toán, đánh giá, kiểm định cũng đều dựa trên giả định này (ngẫu nhiên, phân phối chuẩn) nên nếu mô hình đúng thì sai số cũng phải không theo một chiều hướng nào cả [\[11\]](#).

Trong đề tài này, thực nghiệm sẽ được đánh giá dựa trên các tiêu chí về độ chính xác và thời gian thực thi.

5.3.1. Đánh giá độ chính xác mô hình

Các tiêu chí đánh giá thường được sử dụng trong thực tế để đánh giá một mô hình dự báo như sau:

MAE (mean absolute error) lỗi tuyệt đối trung bình được tính bằng giá trị trung bình của các giá trị sai số dự báo dùng để so sánh mức độ khác biệt giữa dự đoán của một quan sát và giá trị thực của quan sát đó [8].

Công thức:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n} \quad (5.1)$$

Trong đó:

n: số lỗi

y_i : giá trị quan sát thứ i

x_i : giá trị dự đoán tương ứng

e_i sai số tuyệt đối

MSE (mean squared error) lỗi bình phương trung bình được tính bằng giá trị trung bình của lỗi dự báo bình phương các giá trị giúp so sánh các mô hình hồi quy khác nhau hoặc để điều chỉnh các tham số thông qua tìm kiếm lưới và xác thực chéo [8].

Công thức:

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (5.2)$$

Trong đó:

n: số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

RMSE (root mean squared error) lỗi bình phương trung bình gốc xuất phát từ việc lỗi bình phương được mô tả theo đơn vị bình phương của dự đoán cho biết mức độ tập trung của dữ liệu quanh đường phù hợp nhất [8].

Công thức:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (5.3)$$

Trong đó:

n : số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

MAPE (Mean absolute percentage error) là phần trăm sai số trung bình tuyệt đối, đây là thước đo mức độ chính xác của một mô hình dự báo. Nó đo độ chính xác này theo phần trăm và có thể được tính là sai số phần trăm tuyệt đối trung bình cho mỗi khoảng thời gian trừ đi các giá trị thực chia cho các giá trị thực.

Công thức:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{|y_i|} \quad (5.4)$$

Trong đó:

n : số lượng quan sát

y_i : giá trị quan sát thứ i

\hat{y}_i : giá trị dự đoán tương ứng

5.3.2. Đánh giá về thời gian thực thi

5.3.2.1. Mô Hình K-nearest neighbors

Đối với mô hình KNN thời gian thực thi được tính từ lúc dữ liệu bắt đầu được đưa vào mô hình KNN cho đến khi ra được kết quả dự báo

thời gian thực thi = thời gian dữ liệu vào mô hình + thời gian mô hình cho ra dự đoán.

5.3.2.2. Mô Hình FeedFoward NeuralNetwork

Đối với mô hình FFNN thời gian thực thi được tính từ lúc dữ liệu bắt đầu được đưa vào mô hình FFNN cho đến khi ra được kết quả dự báo

thời gian thực thi = thời gian dữ liệu vào mô hình + thời gian mô hình cho ra dự đoán.

5.3.2.3. Mô hình lai ghép song song

Đối với mô hình lai ghép song song thời gian thực thi được tính từ lúc dữ liệu bắt đầu được đưa vào mô hình KNN cho đến khi ra được kết quả dự báo, tương tự với mô hình FFNN được tính từ lúc dữ liệu đi vào mô hình cho đến khi ra kết quả, so sánh thời gian của hai mô hình để tìm ra thời gian lớn nhất. thời gian thực thi sẽ được tính bằng cách lấy thời gian lớn nhất vừa tìm được cộng với thời gian tổng hợp.

thời gian thực thi = $\max(\text{thời gian KNN}, \text{FFNN})$ + thời gian tổng hợp.

5.3.2.4. Mô hình lai ghép tuần tự cộng

Đối với mô hình tuần tự cộng thời gian thực thi sẽ được tính bằng cách lấy tổng của thời gian dự đoán của mô hình KNN với thời gian dự đoán của FFNN và thời gian tổng hợp.

thời gian thực thi = thời gian dự đoán KNN + thời gian dự đoán FFNN + thời gian tổng hợp.

5.4. CÁC TRƯỜNG HỢP THỰC NGHIỆM

5.4.1. Mô hình FFNN

5.4.1.1. *Đánh giá ảnh hưởng của tỷ lệ tập dữ liệu đến việc lựa chọn tham số mô hình*

Trong thực nghiệm này, sẽ đánh giá sự ảnh hưởng của tỷ lệ tập dữ liệu đến việc lựa chọn tham số đầu vào tốt nhất của mô hình, tập dữ liệu sẽ chia theo 80% train và 20% test, tiến hành huấn luyện mô hình trên tập train nhận được các tham số (số neural hidden, batch size, epochs) tốt nhất của mô hình đối với tỷ lệ 80:20 này. Tiếp tục thực nghiệm tương tự trên tỷ lệ chia tập dữ liệu thành 70% train và 30% test để lấy được các tham số tốt nhất của mô hình đối với tỷ lệ 70:30 này, với các kết quả thực nghiệm nhận được các tham số đầu vào tốt nhất đối với tỷ lệ khác nhau sẽ tiến hành xem xét, đánh giá sự ảnh hưởng của tập dữ liệu đến việc lựa chọn tham số cho mô hình.

5.4.1.2. *Đánh giá sự ảnh hưởng số lớp ẩn đối với việc lựa chọn mô hình*

Trong thực nghiệm này, sẽ đánh giá sự ảnh hưởng của số lớp ẩn đến việc lựa chọn tham số đầu vào tốt nhất của mô hình, mô hình sẽ được thực nghiệm trên số lượng lớp ẩn khác nhau, thông qua quá trình train sẽ nhận được các tham số (số neural hidden, batch size, epochs) tốt nhất của mô hình đối với từng số lượng lớp ẩn khác nhau, kết quả thực nghiệm nhận tham số đầu vào tốt nhất đối với từng số lớp ẩn khác nhau sẽ tiến hành xem xét, đánh giá sự ảnh hưởng của số lớp ẩn đến việc lựa chọn tham số cho mô hình.

5.4.1.3. *Thực nghiệm tìm số lượng neural tốt nhất trong mô hình FFNN với một lớp ẩn:*

Trong thực nghiệm này, số neural của lớp input sẽ được cố định là 7 ngày, số neural của lớp output là 1 ngày và số neural của lớp ẩn sẽ được thay đổi từ 1 tới 20, với các tham số này sẽ được truyền vào mô hình FFNN và xem xét đánh giá biểu đồ lỗi đối với sự thay đổi số neural của lớp ẩn để tìm ra số neural tốt nhất.

5.4.1.4 *Thực nghiệm tìm số lượng lớp ẩn tốt nhất trong mô hình FFNN:*

Trong thực nghiệm này, số neural của lớp input sẽ được cố định là 7 ngày, số neural của lớp output là 1 ngày, cố định số neural tốt nhất của một lớp ẩn (thực nghiệm 5.4.1.1), với các tham số này sẽ được truyền vào mô hình FFNN cùng với sự thay đổi

số lớp ẩn tăng dần từ một và xem xét đánh giá biểu đồ lỗi đối với sự thay đổi số lớp ẩn để tìm ra số lớp ẩn tốt nhất.

5.4.2. Mô hình KNN

5.4.2.1 Thực nghiệm tìm K tốt nhất:

Trong thực nghiệm này, chiều dài chuỗi mẫu sẽ được cố định là 7 ngày và chiều dài chuỗi dự đoán là 1 ngày, và k hàng xóm lân cận sẽ được thay đổi tăng dần từ 1, với các tham số này sẽ được đưa vào mô hình KNN để tiến hành dự đoán, và xem xét đánh giá biểu đồ lỗi tương ứng với từng k để tìm ra giá trị k tốt nhất.

5.4.3. So sánh mô hình

Trong thực nghiệm này, sẽ tiến hành so sánh đánh giá các mô hình đơn như KNN và FFNN với các mô hình lai ghép song song và tuần tự cộng dựa trên các tiêu chí đánh giá, xem xét sự ảnh hưởng của việc lai ghép có tốt hơn so với việc dự báo dùng các mô hình đơn, đưa ra các kết luận tổng quát về đề tài này.

5.5. KẾT QUẢ THỰC NGHIỆM

5.5.1. Mô hình FFNN

5.5.1.1. Đánh giá ảnh hưởng của tỷ lệ tập dữ liệu đến việc lựa chọn tham số mô hình:

Trong thực nghiệm này, ta sẽ xem xét ảnh hưởng của tỷ lệ tập dữ liệu train/test đối với việc lựa chọn tham số đầu vào tốt nhất, tăng khả năng dự đoán cho mô hình. Lưu ý rằng với cả 4 tập dữ liệu mẫu đều chọn số neural input là 7, số neural lớp output là 1, và 1 lớp ẩn, tập dữ liệu sẽ được thực nghiệm trên hai trường hợp chia theo tỷ lệ 80% train 20% test và 70% train 30% test, để tiến hành chọn ra các tham số tốt nhất cho mô hình (số lớp neural ẩn, batch size, epoch) sẽ được quá trình training tiến hành lặp để tìm ra được tham số tốt nhất. Bảng 5.1 và bảng 5.2 trình bày các tham số đầu vào của mô hình dự báo được thực nghiệm trên 4 tập dữ liệu (AVAL, MDLY, GDDY, AGEL), với hai tỷ lệ tập dữ liệu 80% train và 20% test, 70% và 30% test.

Bảng 5.1 Tham số đầu vào tốt nhất đối với mô hình FFNN đối với tỷ lệ 80% train và 20% test

	Neural Hidden	Batch Size	Epochs
AVAL	13	8	200
AGLE	19	8	300
MDLY	19	8	300
GDDY	20	8	300

Bảng 5.2 Tham số đầu vào tốt nhất đối với mô hình FFNN đối với tỷ lệ 70% train và 30% test

	Neural Hidden	Batch Size	Epochs
AVAL	19	16	200
AGLE	20	8	250
MDLY	20	8	300
GDDY	14	8	300

Kết quả thực nghiệm cho thấy rằng các tham số đầu vào thay đổi với các giá trị tỷ lệ khác nhau, nên có thể kết luận rằng với từng tỷ lệ khác nhau dựa trên việc chia tập dữ liệu các tham số đầu vào cũng sẽ bị thay đổi do mô hình sẽ tự học lỗi trên tỷ lệ tập train khác nhau và cập nhật các tham số liên tục để đạt kết quả dự báo tốt nhất.

5.5.1.2 Đánh giá sự ảnh hưởng của số lượng lớp ẩn trong mô hình FFNN:

Trong thực nghiệm này, sẽ đánh giá sự ảnh hưởng của số lớp ẩn đến việc lựa chọn tham số đầu vào tốt nhất của mô hình, Lưu ý rằng với cả 4 tập dữ liệu mẫu đều chọn số neural input là 7, số neural lớp output là 1, tập dữ liệu sẽ được chia theo tỷ lệ 80% train 20 % test cùng với số lớp ẩn khác nhau, để tiến hành chọn ra các tham số tốt nhất cho mô hình(số lớp neural ẩn, batch size, epoch) sẽ được quá trình training tiến hành lặp để tìm ra được tham số tốt nhất. Bảng 5.3, 5.4, 5.5, 5.6 trình bày các tham số đầu vào của mô hình dự báo được thực nghiệm trên 4 tập dữ liệu (AVAL, MDLY, GDDY, AGEL), với hai tỷ lệ tập dữ liệu 80% train 20% test và số lượng lớp ẩn khác nhau.

Bảng 5.3 Tham số đầu vào tốt nhất của tập dữ liệu AVAL đối với từng giá trị lớp ẩn khác nhau

	Neural Hidden	Batch Size	Epochs
1	16	8	250
2	16	16	300
3	20	16	300
4	20	16	300
5	19	8	250

Bảng 5.4 Tham số đầu vào tốt nhất của tập dữ liệu AGLE đối với từng giá trị lớp ẩn khác nhau

	Neural Hidden	Batch Size	Epochs
1	15	8	300
2	20	8	250
3	19	8	200
4	15	8	250
5	18	8	250

Bảng 5.5 Tham số đầu vào tốt nhất của tập dữ liệu MDLY đối với từng giá trị lớp ẩn khác nhau

	Neural Hidden	Batch Size	Epochs
1	19	8	300
2	16	8	300
3	16	8	300
4	20	8	250
5	16	8	300

Bảng 5.6 Tham số đầu vào tốt nhất của tập dữ liệu GDDY đối với từng giá trị lớp ẩn khác nhau

	Neural Hidden	Batch Size	Epochs
1	18	8	300
2	19	8	250
3	15	16	300
4	20	8	300
5	19	8	250

Kết quả thực nghiệm cho thấy rằng các tham số đầu vào thay đổi với các số lượng lớp ẩn khác nhau, nên có thể kết luận rằng với các số lượng lớp ẩn thay đổi thì các tham số đầu vào cũng sẽ bị thay đổi do cấu trúc của mô hình đã bị thay đổi dẫn đến việc quá trình học lỗi sẽ khác nhau, nên việc train sẽ cho ra các tham số khác nhau.

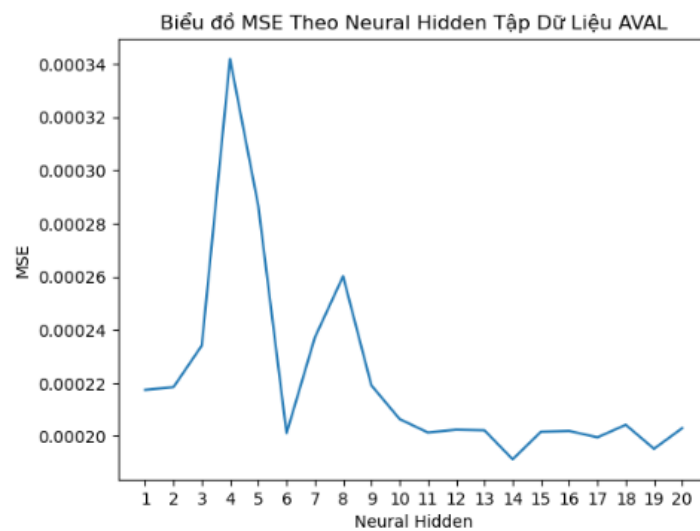
5.5.1.3. Thực nghiệm tìm số lượng neural tốt nhất trong mô hình FFNN với một lớp ẩn:

Trong thực nghiệm này, 4 tập dữ liệu mẫu đều chọn số neural input là 7, số neural output là 1 và cố định số lớp input, số lớp ẩn, số lớp output đều bằng 1 lớp. Các tham số (batch size, epoch, trọng số) sẽ được quá trình training chọn ra tham số tốt nhất, với các tham số đầu vào tốt nhất tìm được nhờ quá trình train cùng với số neural lớp ẩn thay đổi liên tục tăng dần từ một để tìm ra được số lượng neural lớp ẩn tốt nhất bằng cách đánh giá độ lỗi (MSE) và xem xét xu hướng của độ lỗi. Bảng 5.7 trình bày tham số tốt nhất ở 4 tập dữ liệu cùng với lỗi dự báo (MSE).

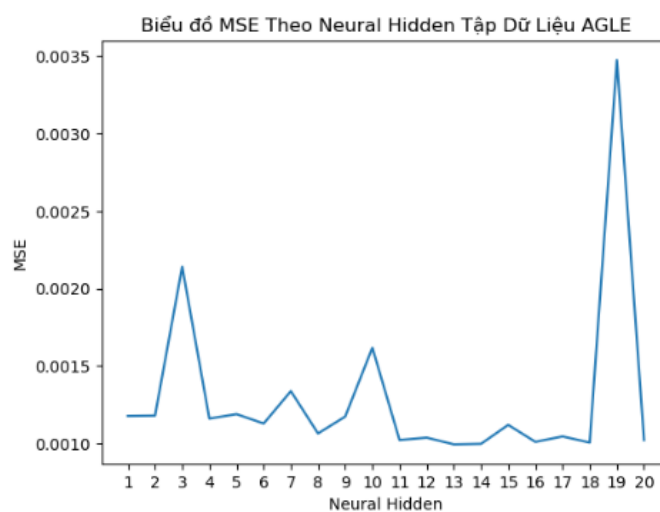
Bảng 5.7 Trình bày tham số tốt nhất ở 4 tập dữ liệu

	Neural Hidden	Batch Size	Epochs	Time Train
AVAL	14.0	8.0	200.0	1269.38
AGLE	13.0	8.0	200.0	1321.6
MDLY	17.0	8.0	300.0	1169.9
GDDY	12.0	8.0	300.0	1722.03

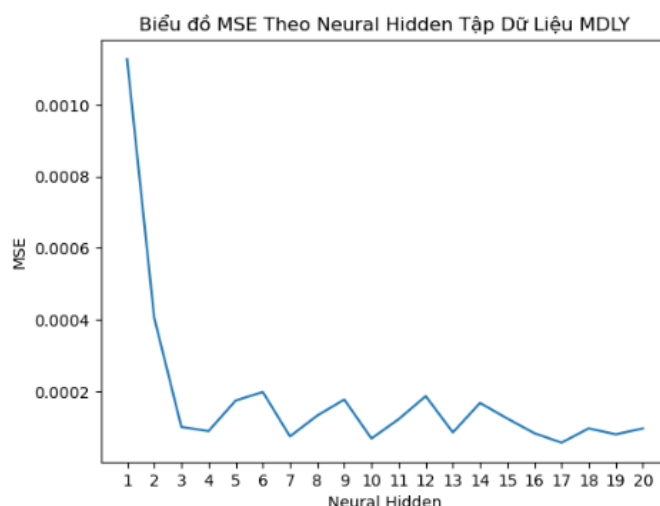
Với Bảng 5.7 đã tìm ra được số neural lớp ẩn và các tham số đầu vào tốt nhất khiến cho tập dữ liệu đạt độ lỗi ít nhất, nhưng để củng cố nhận định trên, ta vẫn sẽ tiếp tục xem xét tổng quát để đánh giá xu hướng lỗi tại 4 tập dữ liệu AVAL, AGLE, MDLY, GDDY lần lượt qua các hình 5.5, 5.6, 5.7, 5.8.



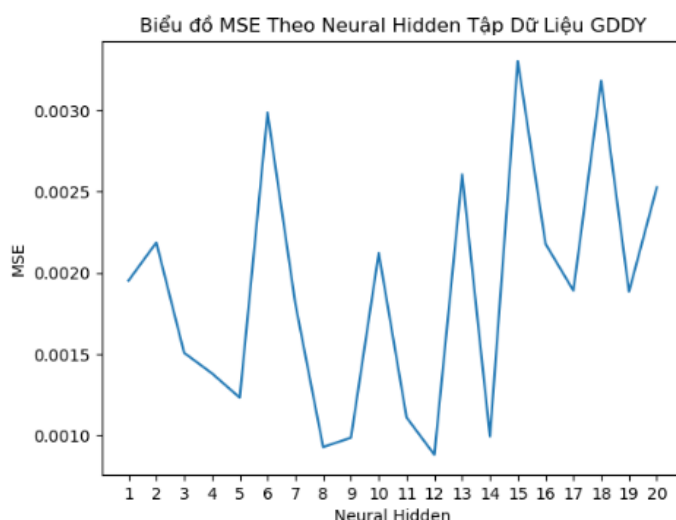
Hình 5.5 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập AVAL



Hình 5.6 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập AGLE



Hình 5.7 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập MDLY



Hình 5.8 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số neural lớp ẩn của tập GDDY

Từ Bảng 5.7 đã tìm ra được lỗi dự báo của tập 4 tập dữ liệu nhỏ nhất khi số neural lớp ẩn của 4 tập dữ liệu AVAL, AGLE, MDLY và GDDY lần lượt là 14, 13, 17, 12 neural và củng cố nhận định thông qua Hình 5.5, Hình 5.6, Hình 5.7, Hình 5.8 đang thể hiện xu hướng lỗi của 4 tập dữ liệu AVAL, AGLE, MDLY và GDDY đều đạt giá trị nhỏ nhất khi số lượng neural lớp ẩn lần lượt là 14, 13, 17, 12 neural.

5.5.1.4 Thực nghiệm tìm số lượng lớp ẩn tốt nhất trong mô hình FFNN:

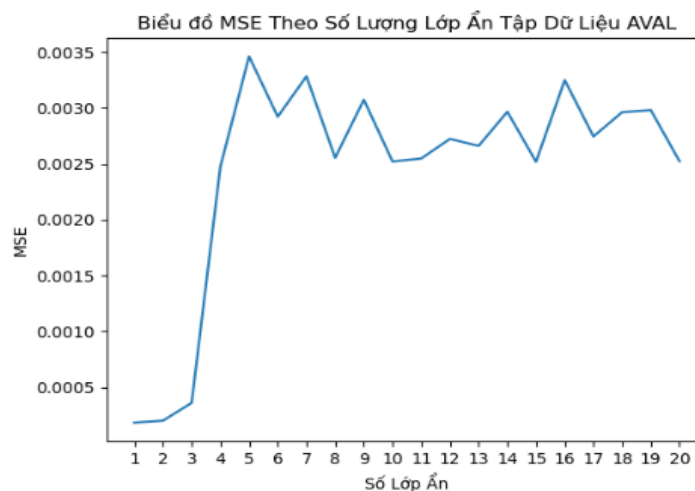
Trong thực nghiệm này, 4 tập dữ liệu mẫu đều chọn chiều dài của chuỗi mẫu là 7, các tham số (batch size, epoch, trọng số, số neural của một lớp ẩn) sẽ được tìm ra trong quá trình thực nghiệm (thực nghiệm 5.5.1.3) chọn ra tham số tốt nhất, với các tham

số đầu vào tốt nhất tìm được nhờ quá trình thực nghiệm(thực nghiệm 5.5.1.3) cùng với số lượng lớp ẩn thay đổi liên tục tăng dần từ một để tìm ra được số lượng lớp ẩn tốt nhất bằng cách đánh giá độ lỗi (MSE) và xem xét xu hướng của độ lỗi. Bảng 5.10 trình bày số lượng lớp ẩn cùng với các tham số đầu vào tốt nhất được thực nghiệm trên 4 tập dữ liệu AVAL, AGLE, MDLY và GDDY, ở thực nghiệm này các tập dữ liệu đều đạt độ lỗi nhỏ nhất khi số lượng lớp ẩn bằng 1.

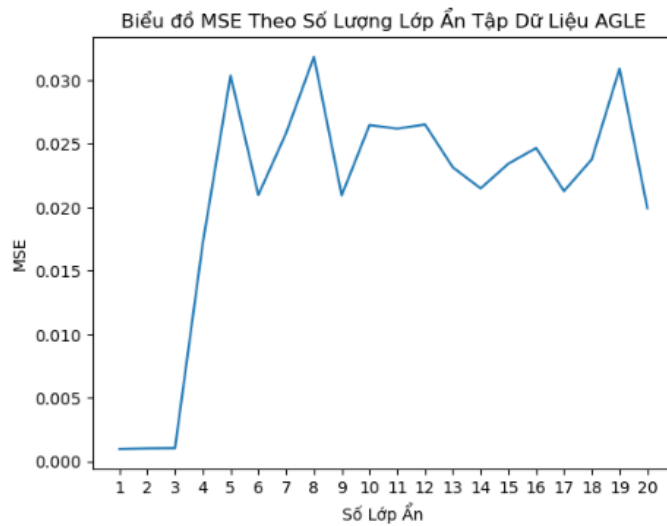
Bảng 5.8 Mô tả Số lượng lớp ẩn cùng với các tham số tốt nhất

	Hidden Layer	Neural Hidden	Batch Size	Epochs	Time Train
AVAL	1	14.0	8.0	200.0	29.054
AGLE	1	13.0	8.0	200.0	25.36
MDLY	1	17.0	8.0	300.0	58.732
GDDY	1	12.0	8.0	300.0	43.009

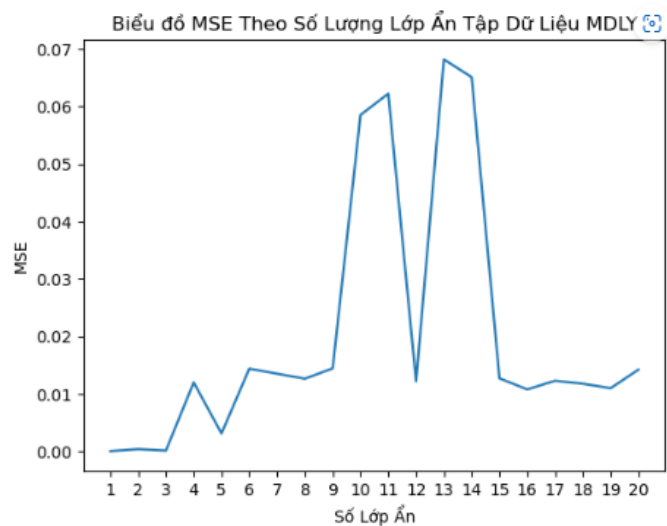
Với Bảng 5.8 đã tìm ra được số lượng lớp ẩn tốt nhất khiến cho tập dữ liệu đạt độ lỗi nhỏ nhất, nhưng để củng cố nhận định này, thực nghiệm vẫn sẽ tiếp tục xem xét tổng quát để đánh giá xu hướng lỗi tại 4 tập dữ liệu AVAL, AGLE, MDLY, GDDY lần lượt qua các hình 5.9, 5.10, 5.11, 5.12.



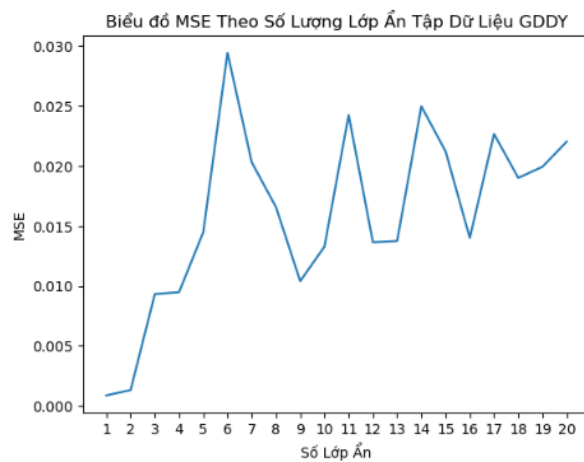
Hình 5.9 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập AVAL



Hình 5.10 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập AGLE



Hình 5.11 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập MDLY



Hình 5.12 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi số lượng lớp ẩn của tập GDDY

Từ Bảng 5.8 đã tìm ra được số lượng lớp ẩn của 4 tập AVAL, AGLE, MDLY, GDDY đều đạt giá trị MSE nhỏ nhất khi số lượng lớp ẩn bằng 1 và củng cố lại nhận định qua Hình 5.9, 5.10, 5.11, 5.12 đang thể hiện xu hướng lỗi và giá trị lỗi dự báo của 4 tập dữ liệu AVAL, AGLE, MDLY, GDDY đều đạt nhỏ nhất khi số lượng lớp ẩn bằng 1.

5.5.2. Mô hình KNN

5.5.2.1 Thực nghiệm tìm K tốt nhất:

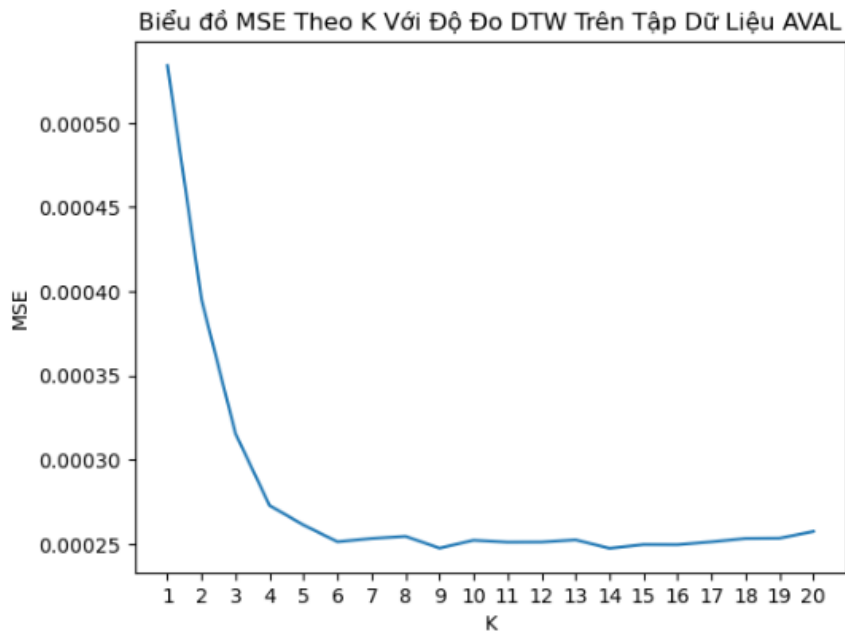
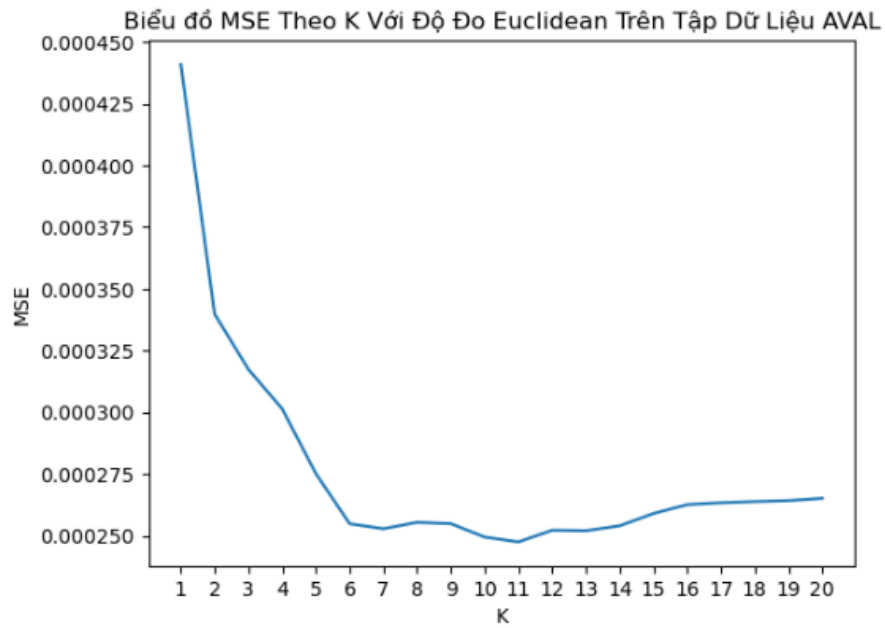
Trong thực nghiệm này, 4 tập dữ liệu mẫu đều chọn chiều dài của chuỗi mẫu là 7 và được thực nghiệm đánh giá dựa trên 2 độ đo DTW (Dynamic time warping) và độ đo euclidean, trong đó số K thay đổi liên tục tăng dần từ một để tìm ra được giá trị k tốt nhất bằng cách đánh giá độ lỗi (MSE), so sánh giữa 2 độ đo và xem xét xu hướng của độ lỗi. Bảng 5.15 trình bày giá trị k tốt nhất đối với 4 tập dữ liệu AVAL, AGLE, MDLY, GDDY đều đạt giá trị lỗi MSE nhỏ nhất theo từng độ đo.

Bảng 5.9 Trình bày giá trị k tốt nhất theo từng độ đo

	Độ Đo	K	MSE	Time
AVAL	Euclidean	11	0.0002474	6.545
	DTW	14	0.0002473	96.883
AGLE	Euclidean	5	0.001547	5.592
	DTW	11	0.001559	76.132
MDLY	Euclidean	4	0.000433	13.944
	DTW	3	0.000379	185.468
GDDY	Euclidean	5	0.001706	10.7999
	DTW	13	0.001978	156.18

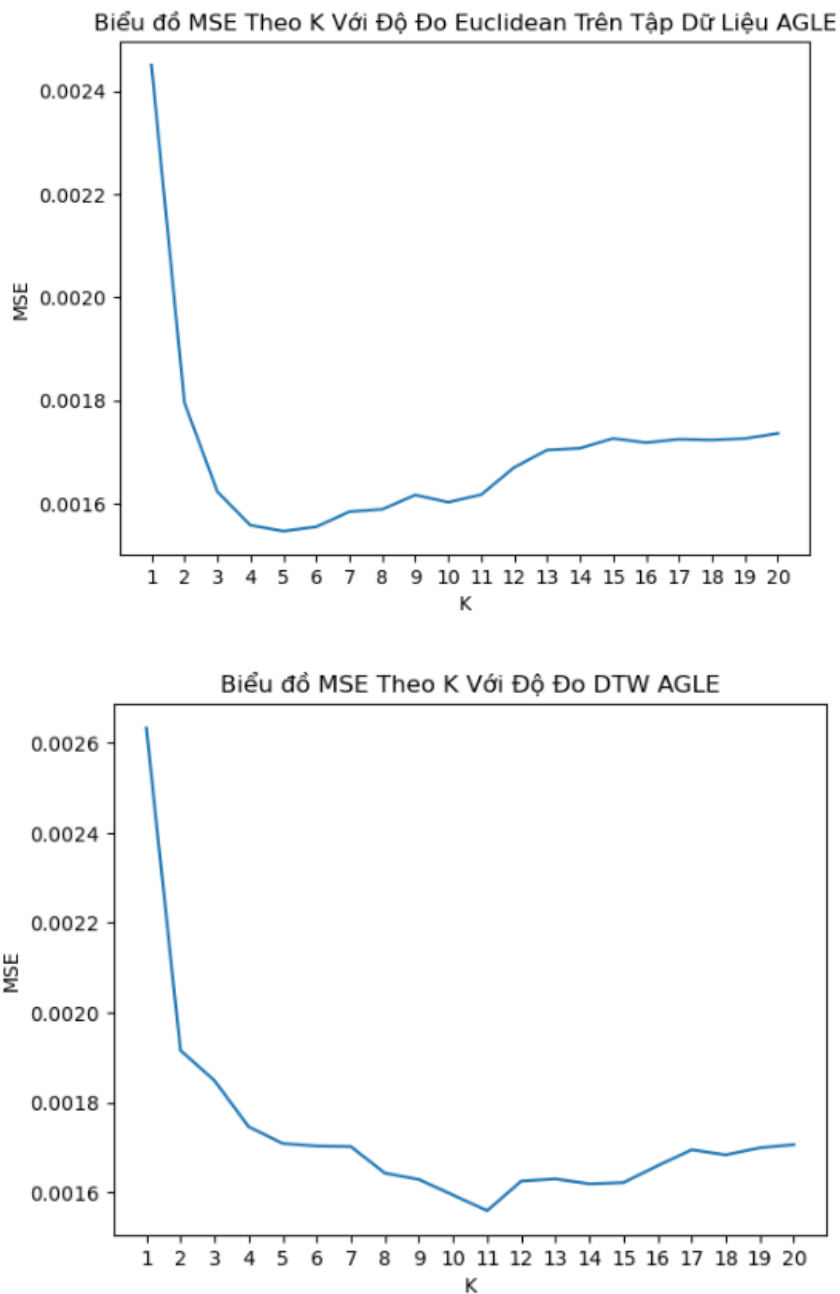
Với Bảng 5.9 tìm ra được giá trị K tốt nhất khiến cho tập dữ liệu đạt độ lỗi ít nhất, nhưng để củng cố lại nhận định thực nghiệm vẫn sẽ được tiếp tục xem xét tổng quát để đánh giá xu hướng lỗi đối với 4 tập dữ liệu tại Hình 5.13, 5.14, 5.15, 5.16

Tại hình 5.13 đối với tập dữ liệu AVAL dựa trên 2 độ đo Euclidean và DTW đã thể hiện xu hướng lỗi MSE theo k tăng dần từ 1 đã đạt giá trị MSE nhỏ nhất tại vị trí k bằng 11 đối với độ đo Euclidean và k bằng 14 đối với độ đo DTW



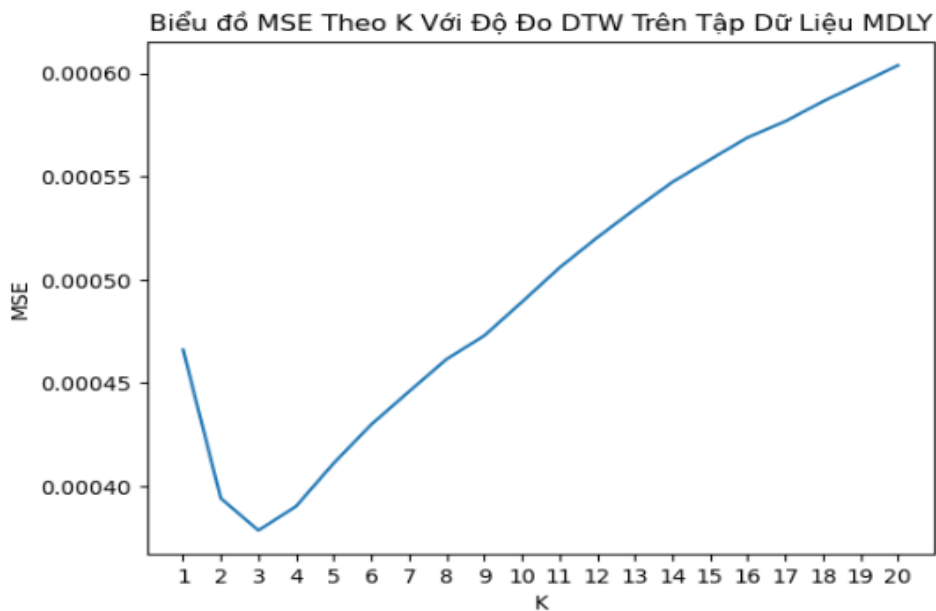
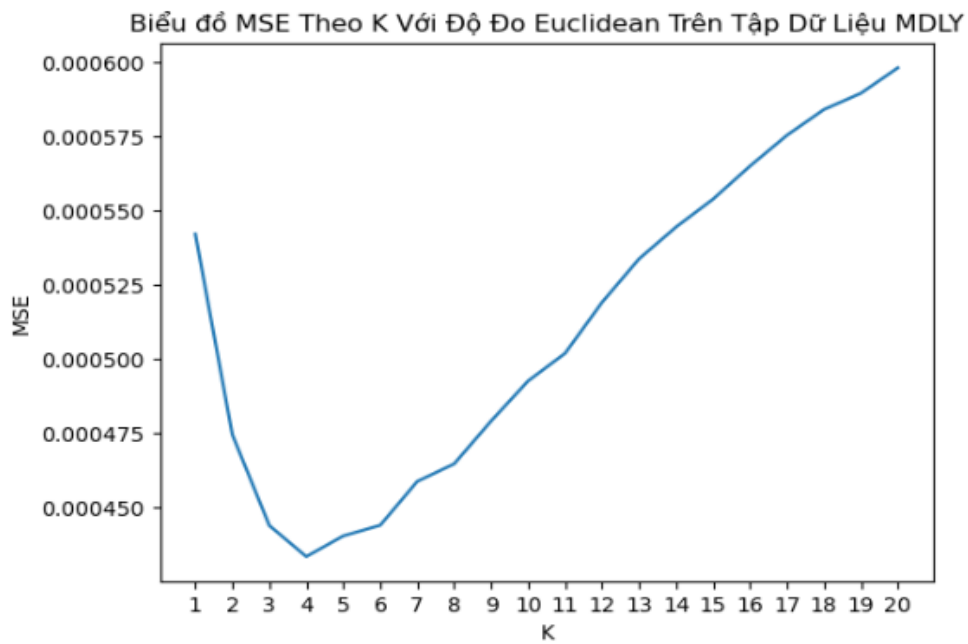
Hình 5.13 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu AVAL

Tại hình 5.14 đối với tập dữ liệu AGLE dựa trên 2 độ đo Euclidean và DTW đã thể hiện xu hướng lỗi MSE theo k tăng dần từ 1 đã đạt giá trị MSE nhỏ nhất tại vị trí k bằng 5 đối với độ đo Euclidean và k bằng 11 đối với độ đo DTW.



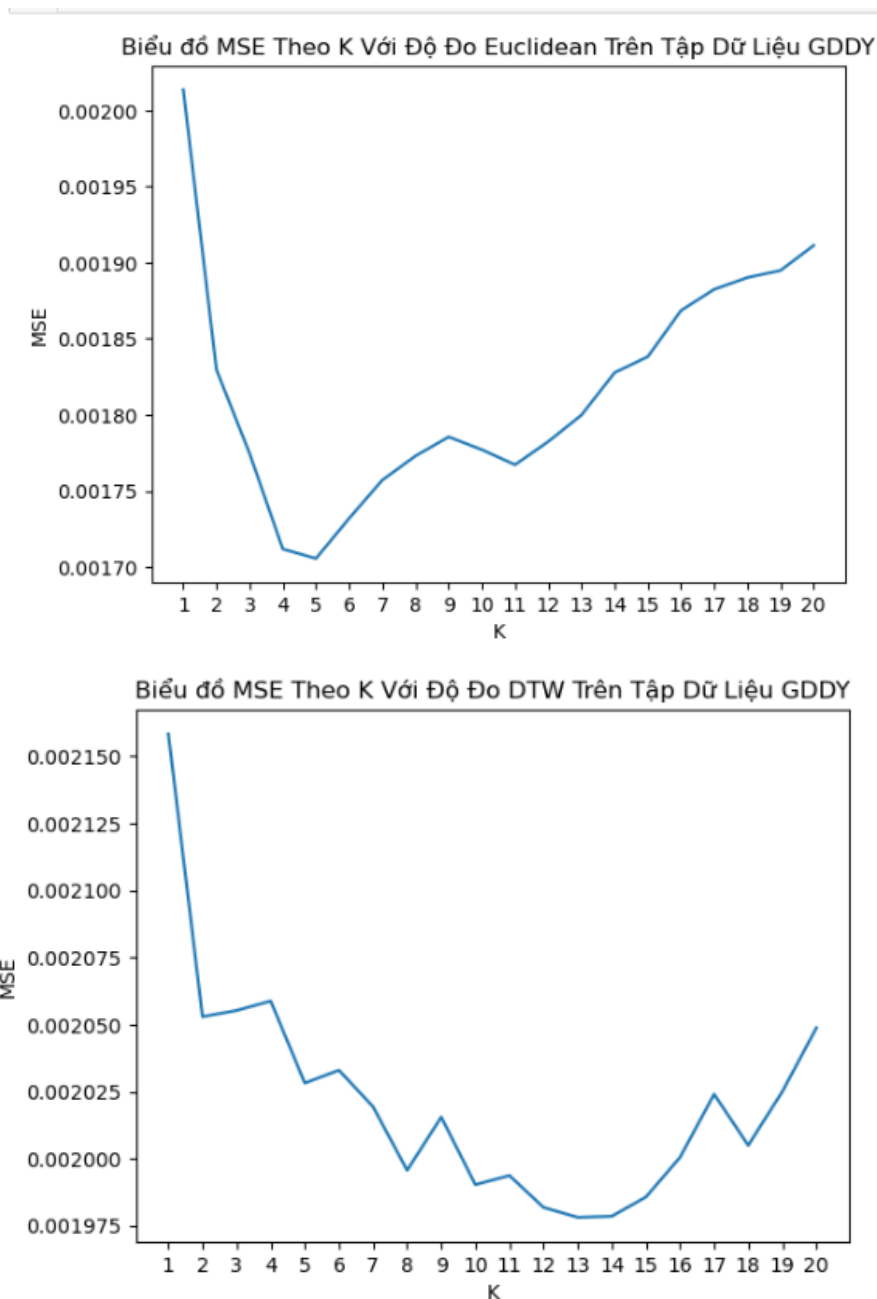
Hình 5.14 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu AGLE

Tại hình 5.15 đối với tập dữ liệu MDLY dựa trên 2 độ đo Euclidean và DTW đã thể hiện xu hướng lỗi MSE theo k tăng dần từ 1 đã đạt giá trị MSE nhỏ nhất tại vị trí k bằng 4 đối với độ đo Euclidean và k bằng 3 đối với độ đo DTW.



Hình 5.15 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu MDLY

Tại hình 5.16 đối với tập dữ liệu GDDY dựa trên 2 độ đo Euclidean và DTW đã thể hiện xu hướng lỗi MSE theo k tăng dần từ 1 đã đạt giá trị MSE nhỏ nhất tại vị trí k bằng 5 đối với độ đo Euclidean và k bằng 13 đối với độ đo DTW.



Hình 5.16 Biểu đồ mô tả xu hướng lỗi dự báo đối với sự thay đổi K của tập dữ liệu GDDY

Từ Bảng 5.9 đã chứng tỏ được rằng lỗi dự báo dữ liệu AGLE, GDDY có giá trị MSE nhỏ nhất với k lần lượt là 5, 5 dựa trên độ đo Euclidean, và tập dữ liệu AVAL, MDLY có giá trị MSE nhỏ nhất với k lần lượt là 14, 3 dựa trên độ đo DTW và cũng có nhận định thông qua hình 5.13, 5.14, 5.15, 5.16 đã thể hiện cả 4 tập dữ liệu tại vị trí k đạt MSE nhỏ nhất ở bảng 5.9 thì cũng chính là k đạt được MSE nhỏ nhất dựa trên xu hướng lỗi.

5.5.3. So sánh mô hình

Trong thực nghiệm này sẽ tổng hợp lại các kết quả của trường hợp thực nghiệm để xem xét đánh giá việc dự đoán trên mô hình lai ghép có tốt hơn so với việc dự đoán trên các mô hình đơn, tại bảng 5.10, 5.11, 5.12, 5.13 trình bày các tiêu chí đánh giá của các mô hình dựa trên các tham số đầu vào tốt nhất lần lượt trên các tập dữ liệu AVAL, AGLE, MDLY, GDDY.

Tại bảng 5.10, dựa trên các tiêu chí đánh giá đối với tập dữ liệu AVAL tại mô hình KNN độ đo DTW có thời gian thực thi lâu hơn so với độ đo Euclidean nhưng độ đo DTW mang đến kết quả dự đoán tốt hơn so với độ đo Euclidean. Tại thực nghiệm mô hình lai ghép song song và mô hình lai ghép tuần tự cộng sẽ sử dụng độ đo DTW cùng với các tham số đầu vào tốt nhất tìm được tại thực nghiệm mô hình KNN và mô hình FFNN, kết quả thực nghiệm hai mô hình lai ghép cho thấy mô hình lai ghép song song cho ra kết quả tốt hơn so với mô hình lai ghép tuần tự. Tổng quát thực nghiệm các mô hình trên tập dữ liệu AVAL, ta thấy mô hình lai ghép song song đã cho ra kết quả tốt nhất, với tổng thời gian thực hiện là 76.21 giây.

Bảng 5.10 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu AVAL

		MSE	RMSE	MAE	MAPE	Time (s)	
						Train	Test
KNN	Euclidean	0.000248	0.01573	0.0121	0.0348	0	6.55
	DTW	0.000247	0.01572	0.0122	0.0351	0	96.9
FFNN		0.000184	0.0136	0.0105	0.0301	29.05	0.35
Song song		0.000000000000001	0.0000001	0.00000004	0.0000001	26.91	49.05
Tuần tự		0.000387	0.0197	0.0171	0.0495	11.85	44.27

Tại Bảng 5.11, dựa trên các tiêu chí đánh giá đối với tập dữ liệu AGLE tại mô hình KNN thì độ đo Euclidean mang đến kết quả dự đoán tốt hơn cùng với thời gian thực thi ngắn hơn so với độ đo DTW. Tại thực nghiệm mô hình lai ghép song song và mô hình lai ghép tuần tự cộng sẽ sử dụng độ đo Euclidean cùng với các tham số đầu vào tốt nhất tìm được tại thực nghiệm mô hình KNN và mô hình FFNN, kết quả thực nghiệm hai mô hình lai ghép cho thấy mô hình lai ghép song song cho ra kết quả tốt hơn so với mô hình lai ghép tuần tự. Tổng quát thực nghiệm các mô hình trên tập dữ liệu AGLE,

ta thấy mô hình lai ghép song song đã cho ra kết quả tốt nhất, với tổng thời gian thực hiện là 29.87 giây.

Bảng 5.11 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu AGLE

		MSE	RMSE	MAE	MAPE	Time (s)	
						Train	Test
KNN	Euclidean	0.00155	0.00155	0.0291	0.0733	0	5.6
	DTW	0.00156	0.03948	0.0289	0.0736	0	76.1
FFNN		0.001	0.03153	0.02293	0.0643	25.4	0.1
Song song		0.0000000000 003	0.000000 6	0.0000000 9	0.00000 02	25.77	3.99
Tuần tự		0.005	0.067738	0.04933	0.13396	8.995	3.95

Tại bảng 5.12, dựa trên các tiêu chí đánh giá đối với tập dữ liệu MDLY tại mô hình KNN độ đo DTW có thời gian thực thi lâu hơn so với độ đo Euclidean nhưng độ đo DTW mang đến kết quả dự đoán tốt hơn so với độ đo Euclidean. Tại thực nghiệm mô hình lai ghép song song và mô hình lai ghép tuần tự cộng sẽ sử dụng độ đo DTW cùng với các tham số đầu vào tốt nhất tìm được tại thực nghiệm mô hình KNN và mô hình FFNN, kết quả thực nghiệm hai mô hình lai ghép cho thấy mô hình lai ghép song song cho ra kết quả tốt hơn so với mô hình lai ghép tuần tự. Tổng quát thực nghiệm các mô hình trên tập dữ liệu MDLY, ta thấy mô hình lai ghép song song đã cho ra kết quả tốt nhất, với tổng thời gian thực hiện là 196.21 giây.

Bảng 5.12 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu MDLY

		MSE	RMSE	MAE	MAPE	Time (s)	
						Train	Test
KNN	Euclidean	0.00043	0.0208	0.0113	211337564 7460.3	0	13.9
	DTW	0.00038	0.0195	0.01098	186494019 9063.8	0	185.5
FFNN		0.000054	0.0073	0.00464	423807078 551.8	58.7	0.2
Song song		0.000000000 0000003	0.000000 02	0.000000 008	64435.8	58.2	137.82
Tuần tự		0.00115	0.0339	0.0155	364674463 4375.8	16.2	147.93

Tại Bảng 5.13, dựa trên các tiêu chí đánh giá đối với tập dữ liệu GDDY tại mô hình KNN thì độ đo Euclidean mang đến kết quả dự đoán tốt hơn cùng với thời gian thực thi ngắn hơn so với độ đo DTW. Tại thực nghiệm mô hình lai ghép song song và mô hình lai ghép tuần tự cộng sẽ sử dụng độ đo Euclidean cùng với các tham số đầu vào

tốt nhất tìm được tại thực nghiệm mô hình KNN và mô hình FFNN, kết quả thực nghiệm hai mô hình lai ghép cho thấy mô hình lai ghép song song cho ra kết quả tốt hơn so với mô hình lai ghép tuần tự. Tổng quát thực nghiệm các mô hình trên tập dữ liệu GDDY, ta thấy mô hình lai ghép song song đã cho ra kết quả tốt nhất, với tổng thời gian thực hiện là 11.691 giây.

Bảng 5.13 Trình bày các kết quả thực nghiệm được thực hiện trên tập dữ liệu GDDY

		MSE	RMSE	MAE	MAPE	Time (s)	
						Train	Test
KNN	Euclidean	0.0017	0.0017	0.0262	0.0389	0	10.8
	DTW	0.00198	0.0445	0.026	0.03994	0	156.19
FFNN		0.0009	0.0296	0.018	0.0264	43.01	0.125
Song song		0.000000000 0009	0.000000 9	0.00000011 2	0.00000 015	5.265	6.275
Tuần tự		0.0039	0.063	0.0365	0.056	10.81	7.998

Tại Bảng 5.10, 5.11, 5.12, 5.13 đã chứng tỏ rằng tại mô hình KNN tùy vào tập dữ liệu sẽ có các độ đo phù hợp khác nhau để tính độ tương đồng giữa 2 chuỗi, và độ đo DTW sẽ luôn có thời gian cao hơn so với độ đo Euclidean vì độ đo DTW sẽ thực hiện tính khoảng cách từ 1 điểm trên chuỗi mẫu đến tất cả các điểm trên chuỗi xét để tìm ra được khoảng cách nhỏ nhất.

So sánh giữa hai mô hình FFNN và KNN thì mô hình FFNN thể hiện khả năng dự báo chính xác hơn so với mô hình KNN, vì mô hình FFNN sẽ có quá trình train, giúp mô hình luôn luôn tự cập nhật các bộ trọng số cùng với các tham số đầu vào tốt nhất để cho ra kết quả dự đoán chính xác nhất, còn ở mô hình KNN là quá trình thực thi xét độ tương đồng giữa chuỗi mẫu với các chuỗi trong quá khứ và lấy các ngày sau của các chuỗi tương đồng, có chiều dài tương ứng với chiều dài chuỗi dự đoán và tính trung bình cộng các ngày sau của chuỗi tương đồng đó.

So sánh giữa các mô hình lai ghép với các mô hình đơn, thì ở mô hình lai ghép song song trên cả 4 tập dữ liệu đều thể hiện khả năng vượt trội trong quá trình dự đoán, ở mô hình song song này đã cải thiện khả năng dự báo hơn so với các mô hình đơn, còn ở mô hình tuần tự lại không cải thiện khả năng dự báo so với cả hai mô hình đơn.

PHẦN KẾT LUẬN

Đề tài này đã mang lại nhiều kiến thức quan trọng giúp nhóm tiếp thu được nhiều kiến thức mới về lĩnh vực dự đoán chuỗi thời gian và có thể áp dụng rộng rãi các kiến thức này với các mô hình, bài toán khác nhau.

- Hiểu được rõ cách xây dựng mô hình FFNN, KNN, mô hình lai ghép song song FFNN và KNN, mô hình lai ghép tuần tự FFNN và KNN.
- Học được các phương pháp tìm trọng số tốt nhất để áp dụng vào mô hình.
- Học hỏi được cách debug để tìm lỗi và giải quyết các lỗi xảy ra trong quá trình phát triển đề tài.

Sau quá trình tìm hiểu các kiến thức liên quan đến đề tài, tiếp thu các nhận xét của giáo viên hướng dẫn nhóm đã xây dựng được:

- Các mô hình dự đoán chuỗi thời gian bằng FFNN, KNN, lai ghép song song FFNN với KNN, lai ghép tuần tự FFNN với KNN cho độ chính xác cao.
- Giao diện cho người dùng truyền tham số, thực hiện train, test và cho ra kết quả so sánh giữa dự đoán và thực tế.

Nhờ sự hướng dẫn tận tình của thầy Nguyễn Thành Sơn nhóm đã học được nhiều kiến thức mới và áp dụng vào đề tài này, nhằm mang lại những hiệu quả sau:

- Đề tài sử dụng nhiều mô hình khác nhau để tìm ra phương pháp thích hợp và hiệu quả trong việc dự báo chuỗi thời gian.
- Đề tài sử dụng nhiều độ đo, nhiều tham số đầu vào giúp thống kê tìm ra những trọng số tối ưu.
- Đề tài sử dụng nhiều tiêu chí đánh giá khác nhau: thời gian thực thi, độ chính xác mô hình (MAE, MSE, RMSE, MAPE).

Khó khăn:

Trong quá trình thực hiện đề tài, do thời gian và kinh nghiệm của nhóm còn hạn chế nên gặp không ít khó khăn trong việc triển khai đề tài.

- Cấu hình máy còn yếu nên chưa đánh giá mô hình trên các tập dữ liệu có kích thước lớn với các hình dạng chuỗi khác nhau.

- Thời gian còn hạn chế nên chưa đánh giá sự ảnh hưởng của kích thước cửa sổ đầu vào, kích thước cửa sổ dự đoán đối với các mô hình.
- Thời gian còn hạn chế nên chưa hoàn thiện app với các tính năng filter, các hiệu ứng,.... để tăng trải nghiệm người dùng.

Hướng phát triển:

Do các khó khăn khách quan về thời gian, kinh nghiệm của nhóm trong quá trình thực hiện đề tài này, vì vậy nhóm đã đề ra hướng phát triển cho đề tài như sau:

- Thực nghiệm trên nhiều dữ liệu với các kích thước dữ liệu lớn và hình dạng chuỗi thời gian khác nhau để có được độ tin cậy cao hơn trong mô hình lai ghép.
- Đánh giá ảnh hưởng của các kích thước cửa sổ mẫu và kích thước cửa sổ dự đoán đối với các mô hình.
- Cải thiện app với các chức năng và phát triển các tiện ích (thống kê có filter, các hiệu ứng, ...) để tăng trải nghiệm và hiệu suất cho người dùng.

DANH SÁCH TÀI LIỆU THAM KHẢO

- [1] Oleh Onyshchak (April 2020), Stock market dataset, Kaggle.com.
- [2] P. G. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, *Neurocomputing* 50 (2003) 159–175.
- [3] Nguyen Thanh Son (2018), ‘Hybridising neural network and pattern matching under dynamic time series prediction’, *Int.J.Business Intelligence and Data Mining*, Vol.17, No.1, 2020.Pp54-75 (Scopus Journal, RG Journal Impact: 0.67)
- [4] Jason Brownlee (2020), Introduction to Time Series Forecasting with Python, All Rights Reserved.
- [5] Roya Asadi and Sameem Abdul Kareem (2014), Review of Feed Forward Neural Network classification preprocessing techniques, Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, 50603, Malaysia
- [6] Jason Brownlee on December 5, 2016 Time Series Forecasting as Supervised Learning in Time Series
- [7] Michael Steinbach and Pang-Ning Tan (2009), The top Ten Algorithms in Data Mining, Chapter 8: kNN: k-Nearest Neighbors., by Taylor & Francis Group, LLC.
- [8] Ivan Svetunkov 2023 Forecasting and Analytics with ADAM Arne Niklas Jansson
- [9] KE-LIN DU and M. N. S. SWAMY (April 28, 2013), Neural networks and Statistical Learning, Chapter 11: Recurrent Neural Network, Enjoyor Labs, Enjoyor Inc., China, Concordia University, Canada.
- [10] Ralf C. Staudemeyer, Eric Rothstein Morris, Understanding LSTM, A tutorial into Long Short-Term Memory, Recurrent Neural Networks, Echamkaden University, Singapore University, September 23, 2019.
- [11] Nguyễn Thành Sơn (6/2014- 10/2015). *Dự báo dữ liệu chuỗi thời gian có tính xu hướng hoặc mùa sử dụng giải thuật k lân cận gần nhất*, Công trình nghiên cứu khoa học cấp trường, Trường Đại học Sư Phạm Kỹ Thuật Hồ Chí Minh.
- [12] Damodar Gujarati Chapter 13 (2011), *Econometrics by example*, Bloomsbury.
- [13] Romain Tavenard (Sakoe & Chiba, 1978), An introduction to Dynamic Time Warping,
- [14] Vaibhav Verbhane (October 2020), Supervised Learning with Python: Concepts and Practical Implementation Using Python, Apress.
- [15] Fabio Manganiello (February 2021), Computer Vision with Maker Tech: Detecting People With a Raspberry Pi, a Thermal Camera, and Machine Learning, Apress.
- [16] Xue Ying (2019), An Overview of Overfitting and its Solutions, *Journal of Physics: Conference Series*