

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN

**BÁO CÁO MÔN HỌC  
TOÁN CHO TRÍ TUỆ NHÂN  
TẠO**

**DỰ ĐOÁN LỖI CÁNH QUẠT ĐIỆN  
GIÓ BẰNG SUPPORT VECTOR  
MACHINE**

**HỌC KỲ 2 – NĂM HỌC: 2024-2025**

**MÃ LỚP HỌC PHẦN: MAAI330985\_01CLC**

**Giảng viên hướng dẫn:** TS. Bùi Mạnh Quân

**Nhóm sinh viên thực hiện:** Nguyễn Trung Hiếu MSSV: 22110138

Lê Hoàng Bảo Phúc MSSV: 22110200

Phạm Anh Quân MSSV: 22110215

TP. HCM, tháng 05 năm 2025

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC  
TOÁN CHO TRÍ TUỆ NHÂN  
TẠO

DỰ ĐOÁN LỖI CÁNH QUẠT ĐIỆN  
GIÓ BẰNG SUPPORT VECTOR  
MACHINE

HỌC KỲ 2 – NĂM HỌC: 2024-2025

MÃ LỚP HỌC PHẦN: MAAI330985\_01CLC

Giảng viên hướng dẫn: TS. Bùi Mạnh Quân

Nhóm sinh viên thực hiện: Nguyễn Trung Hiếu MSSV: 22110138  
Lê Hoàng Bảo Phúc MSSV: 22110200  
Phạm Anh Quân MSSV: 22110215

TP. HCM, tháng 05 năm 2025

## **NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

# MỤC LỤC

<b>DANH MỤC HÌNH ẢNH .....</b>	<b>1</b>
<b>DANH MỤC BẢNG BIỂU .....</b>	<b>2</b>
<b>DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT .....</b>	<b>3</b>
<b>LỜI NÓI ĐẦU .....</b>	<b>4</b>
<b>CHƯƠNG 1. GIỚI THIỆU .....</b>	<b>5</b>
1.1. Lý do chọn đề tài .....	5
1.2. Mục tiêu đề tài .....	5
1.3. Phạm vi đề tài .....	7
1.4. Đối tượng nghiên cứu .....	7
1.5. Phương pháp nghiên cứu .....	8
1.6. Bố cục đề tài .....	10
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT .....</b>	<b>12</b>
2.1. Định nghĩa SVM .....	12
2.2. Cách SVM phân loại dữ liệu .....	14
2.3. Bài toán phân loại với SVM .....	18
2.4. SVM Kernel RBF .....	20
2.4.1. Tại sao phải dùng RBF Kernel? .....	20
2.4.2. Định nghĩa .....	20
2.4.3. Biến Đổi Các Thuật Toán Tuyến Tính Thành Bộ Phân Loại và Hồi Quy Phi Tuyến .....	20
2.4.4. Vai trò của Kernel RBF trong SVM .....	21
2.4.5. Tham số $\gamma$ .....	21
<b>CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM .....</b>	<b>22</b>
3.1. Phân tích dữ liệu đầu vào .....	22
3.1.1. Giới thiệu tập dữ liệu .....	22
3.1.2. Mô tả về tập dữ liệu .....	23
3.2. Xây dựng mô hình dự đoán lỗi của cách quạt gió .....	29
3.2.1. Quy trình xây dựng mô hình .....	29
3.2.2. Kết quả thực nghiệm .....	46
<b>CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>51</b>
4.1. Kết luận .....	51
4.2. Hướng phát triển .....	52
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>54</b>

## **DANH MỤC HÌNH ẢNH**

Hình 2.1.1: Phân Chia Nhóm Dữ Liệu Bằng Siêu Phẳng.....	14
Hình 2.2.1: Các Điểm Dữ Liệu Trên Mặt Phẳng.....	15
Hình 2.2.2: Tác Động Của Tham Số C Lên Siêu Phẳng.....	17
Hình 2.2.3: Siêu Phẳng Không Phân Nhóm Được Dữ Liệu .....	18
Hình 3.2.1: Phân Bố Nhãn Ban Đầu.....	46
Hình 3.2.2: Phân Bố Nhãn Sau Khi Tăng Cường .....	46
Hình 3.2.3: Số Đặc Trưng Sau PCA.....	47
Hình 3.2.4: Tham Số Tối Ưu.....	47
Hình 3.2.5: Độ Chính Xác & Độ Lệch Chuẩn .....	47
Hình 3.2.6: Biểu Đồ Learning Curve Cho Mô Hình SVM Với Kernel RBF.....	48
Hình 3.2.7: Báo Cáo Phân Loại.....	49
Hình 3.2.8: Ma Trận Nhầm Lẫn Cho Mô Hình SVM với Kernel RBF .....	50

## **DANH MỤC BẢNG BIỂU**

Bảng 3.1.1: Phân Loại Các Tập Dữ Liệu .....	22
Bảng 3.1.2: Phân Loại Nhãn Đầu Ra .....	23
Bảng 3.1.3: Thuộc Tính Của Tập Dữ Liệu Tổng Hợp .....	28

## **DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT**

<b>Thuật ngữ</b>	<b>Ý nghĩa</b>
SVM	Support Vector Machine

## LỜI NÓI ĐẦU

Năng lượng gió đang trở thành trụ cột trong việc cung cấp nguồn năng lượng tái tạo, sạch và bền vững trên toàn cầu. Các tua-bin gió cần đảm bảo hiệu suất cao và độ tin cậy để giảm chi phí vận hành, nhưng cánh quạt dễ gặp các hư hỏng như xói mòn, nứt gãy, mất cân bằng hoặc xoắn cánh, gây ảnh hưởng nghiêm trọng nếu không được phát hiện kịp thời. Những thách thức này đòi hỏi các phương pháp giám sát tiên tiến để cải thiện hiệu quả chẩn đoán lỗi.

Việc phát hiện lỗi cánh quạt gấp khó khăn do thiết kế phức tạp và điều kiện vận hành thay đổi, như tốc độ gió. Các phương pháp truyền thống thường không đủ chính xác, đặc biệt với dữ liệu bị nhiễu. Kỹ thuật học máy, cụ thể là Support Vector Machine (SVM), nổi bật nhờ khả năng phân loại tín hiệu rung động, mang lại giải pháp hiệu quả cho việc dự đoán lỗi. Báo cáo này khai thác tập dữ liệu rung động từ nghiên cứu của Ogaili và cộng sự (2023) trên *Data in Brief* để xây dựng mô hình SVM, nhằm nâng cao khả năng giám sát và bảo trì dự đoán.

Nghiên cứu tập trung phân loại các lỗi cánh quạt dựa trên dữ liệu rung động thu thập ở nhiều tốc độ gió, từ trạng thái khỏe mạnh đến các trạng thái lỗi. Mục tiêu là đề xuất một hệ thống dự đoán chính xác, hỗ trợ ngành công nghiệp điện gió tối ưu hóa vận hành và phát triển bền vững. Báo cáo được tổ chức thành các phần: cơ sở lý thuyết, phương pháp SVM, phân tích dữ liệu, và kết luận, mang đến cái nhìn toàn diện về tiềm năng ứng dụng của học máy trong lĩnh vực này.

# CHƯƠNG 1. GIỚI THIỆU

## 1.1. Lý do chọn đề tài

Năng lượng gió ngày càng trở thành nguồn lực quan trọng trong chiến lược phát triển bền vững toàn cầu, với công suất điện gió tăng hơn 10% mỗi năm theo Cơ quan Năng lượng Tái tạo Quốc tế (IRENA). Tuy nhiên, ngành công nghiệp điện gió đối mặt với thách thức lớn về độ tin cậy và chi phí vận hành, đặc biệt liên quan đến cánh quạt – bộ phận cốt lõi của tua-bin gió. Cánh quạt thường xuyên chịu tác động từ gió mạnh, mưa, bụi, và biến đổi nhiệt độ, dẫn đến các hư hỏng như xói mòn bề mặt, nứt vỡ, mất cân bằng khối lượng, hoặc biến dạng góc xoắn. Những lỗi này không chỉ làm giảm hiệu suất phát điện mà còn gây ra sự cố nghiêm trọng, như hỏng hệ thống hoặc tai nạn lao động, kéo theo chi phí bảo trì cao. Trong bối cảnh các trang trại điện gió ngày càng mở rộng, việc chẩn đoán lỗi cánh quạt trở nên cấp thiết do thiết kế phức tạp và điều kiện vận hành biến đổi, trong khi các phương pháp giám sát truyền thống như kiểm tra trực quan thường thiếu chính xác và không kịp thời, đặc biệt khi dữ liệu bị nhiễu.

Việc giải quyết bài toán chẩn đoán lỗi cánh quạt mang lại lợi ích thiết thực cho các nhà vận hành trang trại điện gió bằng cách giảm thời gian ngừng hoạt động, tối ưu hóa bảo trì, và kéo dài tuổi thọ tua-bin, từ đó tiết kiệm chi phí. Đối với ngành năng lượng tái tạo, nâng cao độ tin cậy của tua-bin gió đảm bảo nguồn cung điện ổn định, củng cố vai trò của năng lượng gió như giải pháp thay thế nhiên liệu hóa thạch. Hơn nữa, các kỹ thuật phân tích tín hiệu rung động trong đề tài có tiềm năng ứng dụng trong các lĩnh vực như hàng không (giám sát cánh quạt), sản xuất (phát hiện lỗi máy móc), hoặc năng lượng thủy điện (theo dõi turbine). Đề tài “Hệ thống dự đoán lỗi cánh quạt điện gió bằng Support Vector Machine” được chọn để khai thác tín hiệu rung động, đáp ứng nhu cầu thực tiễn và mở ra cơ hội nghiên cứu liên ngành, góp phần vào sự phát triển của công nghệ giám sát thông minh trong thời đại công nghiệp 4.0.

## 1.2. Mục tiêu đề tài

Đề tài “Hệ thống dự đoán lỗi cánh quạt điện gió bằng Support Vector Machine” hướng tới xây dựng một hệ thống thông minh nhằm phát hiện và phân loại các lỗi trên cánh quạt tua-bin gió dựa trên phân tích tín hiệu rung động. Hệ thống được thiết kế để tự động nhận diện các trạng thái lỗi phổ biến, bao gồm xói mòn bề mặt, nứt gãy, mất cân bằng khói lượng, và biến dạng góc xoắn, từ dữ liệu rung động thu thập trong các điều kiện vận hành khác nhau, như thay đổi tốc độ gió. Mục tiêu chính là cung cấp một công cụ hỗ trợ giám sát tình trạng hiệu quả, giúp tối ưu hóa vận hành và bảo trì các tua-bin gió, từ đó nâng cao độ tin cậy và hiệu suất của các trang trại điện gió.

Hệ thống này chủ yếu phục vụ các đối tượng trong ngành công nghiệp năng lượng tái tạo, bao gồm các nhà vận hành trang trại điện gió, kỹ sư bảo trì, và các nhà quản lý hệ thống năng lượng. Cụ thể, hệ thống sẽ hỗ trợ các kỹ sư trong việc phát hiện sớm các dấu hiệu hư hỏng, từ đó lập kế hoạch bảo trì dự đoán, giảm thiểu thời gian ngừng hoạt động và chi phí sửa chữa. Đối với các nhà vận hành, hệ thống cung cấp thông tin chính xác về tình trạng sức khỏe của tua-bin, giúp đảm bảo nguồn cung cấp điện ổn định và tối ưu hóa hiệu suất phát điện. Ngoài ra, các nhà nghiên cứu trong lĩnh vực học máy và kỹ thuật cơ khí cũng có thể tận dụng hệ thống như một nền tảng để thử nghiệm và phát triển các thuật toán chẩn đoán lỗi tiên tiến hơn.

Về kết quả cần đạt được, đề tài đặt mục tiêu phát triển một hệ thống ứng dụng có khả năng phân loại chính xác các loại lỗi cánh quạt với độ tin cậy cao, dựa trên tập dữ liệu rung động từ nghiên cứu của Ogaili và cộng sự (2023). Hệ thống sẽ sử dụng kỹ thuật Support Vector Machine (SVM) để xử lý và phân tích dữ liệu, tận dụng khả năng phân loại mạnh mẽ của thuật toán này để nhận diện các mẫu rung động đặc trưng cho từng trạng thái lỗi. Kết quả mong đợi là một mô hình SVM được huấn luyện và kiểm định, đạt độ chính xác cao trong việc phân biệt giữa trạng thái khỏe mạnh và các trạng thái lỗi, ngay cả trong điều kiện dữ liệu bị nhiễu hoặc tốc độ gió thay đổi. Về công nghệ ứng dụng, đề tài nhằm đóng góp vào việc thúc đẩy sử dụng học máy trong giám sát tình trạng công nghiệp, mở ra tiềm năng áp dụng kỹ thuật SVM vào các lĩnh vực khác như giám sát máy móc quay trong sản xuất hoặc phân tích tín hiệu trong hàng không.

Kết quả này không chỉ mang lại giá trị thực tiễn cho ngành điện gió mà còn đặt nền tảng cho các nghiên cứu liên ngành trong thời đại công nghiệp 4.0.

### **1.3. Phạm vi đề tài**

Để đảm bảo tính khả thi và tập trung, phạm vi nghiên cứu được giới hạn rõ ràng về không gian, thời gian, lĩnh vực, và các khía cạnh cụ thể của đề tài, tránh mở rộng quá mức hoặc thu hẹp không cần thiết.

Về không gian, nghiên cứu sử dụng tập dữ liệu rung động từ bài báo của Ogaili và cộng sự (2023), được thu thập tại Phòng thí nghiệm Năng lượng Tái tạo, Khoa Kỹ thuật Cơ khí, Đại học Mustansiriyah, Baghdad, Iraq. Dữ liệu này bao gồm tín hiệu rung động từ một tua-bin gió quy mô phòng thí nghiệm, mô phỏng các điều kiện thực tế với tốc độ gió từ 1.3 m/s đến 5.6 m/s. Do đó, phạm vi không gian của đề tài giới hạn ở môi trường phòng thí nghiệm, không mở rộng sang các tua-bin gió thương mại hoặc điều kiện thực địa phức tạp hơn, như trang trại điện gió ngoài khơi hoặc trên núi.

Về thời gian, đề tài dựa trên tập dữ liệu được công bố năm 2023, phản ánh các phép đo rung động trong một khoảng thời gian thử nghiệm cụ thể tại phòng thí nghiệm. Nghiên cứu không xem xét các yếu tố thay đổi theo thời gian dài, như sự xuống cấp dần của cánh quạt qua nhiều năm hoặc tác động của các mùa trong năm. Thay vào đó, đề tài tập trung vào phân tích dữ liệu tĩnh tại thời điểm thu thập, với mục tiêu xây dựng mô hình SVM dựa trên các phép đo ngắn hạn.

Về lĩnh vực, đề tài thuộc giao thoa giữa kỹ thuật cơ khí và học máy, cụ thể là giám sát tình trạng (condition monitoring) trong ngành năng lượng tái tạo. Nghiên cứu tập trung vào việc áp dụng SVM để phân loại các lỗi cánh quạt, bao gồm xói mòn bề mặt, nứt gãy, mất cân bằng khối lượng, và biến dạng góc xoắn. Các khía cạnh khác của tua-bin gió, như động cơ hoặc hệ thống truyền động, không nằm trong phạm vi nghiên cứu. Ngoài ra, đề tài chỉ sử dụng tín hiệu rung động đơn trực (uniaxial) từ cảm biến gia tốc, không xem xét các loại dữ liệu khác như tín hiệu âm thanh hoặc hình ảnh.

### **1.4. Đối tượng nghiên cứu**

Nghiên cứu tập trung vào các thành phần và dữ liệu liên quan đến việc phân tích trạng thái hoạt động của cánh quạt tua-bin gió thông qua tín hiệu rung động. Đối tượng nghiên cứu chính bao gồm tập hợp tín hiệu rung động đơn trực, được thu thập từ cảm biến gia tốc gắn trên nacelle gần trung tâm tua-bin gió trong môi trường phòng thí nghiệm. Những tín hiệu này phản ánh các trạng thái vận hành khác nhau của cánh quạt, bao gồm cả tình trạng không hư hỏng và các trạng thái lỗi cụ thể như bề mặt bị xói mòn, nứt gãy, mất cân bằng khối lượng, và biến dạng góc xoắn. Dữ liệu được lấy từ 35 tệp CSV, đại diện cho các phép đo tại nhiều tốc độ gió từ 1.3 m/s đến 5.6 m/s, cung cấp thông tin về biên độ rung động theo thời gian.

Bên cạnh tín hiệu rung động, nghiên cứu còn xem xét các đặc trưng thống kê được trích xuất từ dữ liệu này, chẳng hạn như giá trị trung bình, độ lệch chuẩn, và các thông số tần số, nhằm làm rõ mối liên hệ giữa các mẫu rung động và loại lỗi tương ứng. Các đặc trưng này đóng vai trò như đầu vào chính để phân tích và phân loại trạng thái của cánh quạt. Ngoài ra, cấu trúc vật lý của cánh quạt trong thí nghiệm, được chế tạo từ polymer gia cường sợi (FRP) với chiều dài 300 mm và góc lắp đặt cố định, cũng là một đối tượng nghiên cứu để hiểu cách các lỗi được mô phỏng ảnh hưởng đến tín hiệu rung động.

Nghiên cứu không mở rộng sang các bộ phận khác của tua-bin gió, như động cơ hoặc hệ thống truyền động, mà chỉ giới hạn ở cánh quạt và các tín hiệu rung động liên quan. Các điều kiện vận hành, cụ thể là tốc độ gió được mô phỏng trong phòng thí nghiệm, cũng được xem xét như một đối tượng phụ để đánh giá tác động của chúng đến đặc tính rung động. Thông qua việc tập trung vào các đối tượng này, nghiên cứu nhằm làm rõ các mẫu tín hiệu đặc trưng cho từng loại lỗi, tạo cơ sở cho việc phân loại và phát hiện hư hỏng một cách chính xác trong môi trường kiểm soát.

## 1.5. Phương pháp nghiên cứu

Nghiên cứu áp dụng một số phương pháp để thu thập, xử lý và phân tích dữ liệu nhằm phân loại các trạng thái lỗi của cánh quạt tua-bin gió dựa trên tín hiệu rung động. Các phương pháp được sử dụng bao gồm thu thập thông tin, xử lý thông tin định lượng, và

phân tích dữ liệu bằng kỹ thuật học máy, với trọng tâm là đảm bảo tính chính xác và khả thi trong môi trường phòng thí nghiệm.

Phương pháp thu thập thông tin: Dữ liệu chính được lấy từ tập dữ liệu rung động công bố trong nghiên cứu của Ogaili và cộng sự (2023), bao gồm 35 tệp CSV chứa tín hiệu rung động đơn trực từ cảm biến gia tốc PCB Piezotronics 352C65. Các tín hiệu này được ghi nhận tại Phòng thí nghiệm Năng lượng Tái tạo, Đại học Mustansiriyah, Baghdad, Iraq, với tốc độ gió mô phỏng từ 1.3 m/s đến 5.6 m/s. Ngoài ra, phương pháp đọc tài liệu được sử dụng để thu thập thông tin lý thuyết về phân tích tín hiệu rung động, kỹ thuật học máy, và các nghiên cứu liên quan đến giám sát tình trạng tua-bin gió. Các tài liệu tham khảo bao gồm bài báo khoa học, sách chuyên ngành, và báo cáo kỹ thuật từ các nguồn uy tín.

Phương pháp xử lý thông tin: Nghiên cứu áp dụng phương pháp định lượng để xử lý tín hiệu rung động. Các đặc trưng thống kê, như giá trị trung bình, độ lệch chuẩn, độ lệch, và các thông số tần số (qua biến đổi Fourier), được trích xuất từ dữ liệu thô để làm nổi bật các mẫu rung động đặc trưng cho từng trạng thái lỗi, bao gồm xói mòn bề mặt, nứt gãy, mất cân bằng khối lượng, và biến dạng góc xoắn. Dữ liệu sau đó được chuẩn hóa và tổ chức thành các tập huấn luyện và kiểm tra để phục vụ phân tích học máy. Phương pháp này đảm bảo rằng thông tin được xử lý một cách hệ thống và có thể sử dụng trực tiếp cho mô hình phân loại.

Phương pháp phân tích học máy: Kỹ thuật Support Vector Machine (SVM) được sử dụng làm công cụ chính để phân loại các trạng thái lỗi dựa trên các đặc trưng trích xuất. SVM được chọn vì khả năng xử lý dữ liệu phi tuyến và hiệu quả trong việc phân biệt các lớp dữ liệu phức tạp. Quá trình phân tích bao gồm huấn luyện mô hình trên tập dữ liệu đã được gắn nhãn, tối ưu hóa các tham số như hàm nhân (kernel) và tham số phạt, sau đó đánh giá hiệu suất thông qua các chỉ số như độ chính xác, độ nhạy, và độ đặc hiệu. Phương pháp này được thực hiện bằng các công cụ lập trình như Python với thư viện scikit-learn.

Phương pháp đánh giá: Hiệu quả của mô hình được đánh giá thông qua kỹ thuật kiểm định chéo (cross-validation) để đảm bảo tính tổng quát hóa trên các tập dữ liệu khác nhau. Các chỉ số định lượng, như ma trận nhầm lẫn (confusion matrix) và đường cong ROC, được sử dụng để so sánh hiệu suất phân loại giữa các trạng thái lỗi và trạng thái không hư hỏng. Phương pháp này giúp xác định độ tin cậy của mô hình trong việc nhận diện các mẫu rung động đặc trưng.

Bằng cách kết hợp các phương pháp trên, nghiên cứu đảm bảo dữ liệu được thu thập đầy đủ, xử lý khoa học, và phân tích chính xác, tạo nền tảng cho việc phát triển một hệ thống phân loại lỗi hiệu quả trong môi trường phòng thí nghiệm.

## 1.6. Bố cục đề tài

Phần còn lại của báo cáo tiểu luận môn học này được tổ chức như sau. Chương 2 trình bày cơ sở lý thuyết nền tảng liên quan đến việc phân tích tín hiệu rung động và ứng dụng học máy trong giám sát tình trạng tua-bin gió. Nội dung chương bao gồm tổng quan về các đặc trưng của tín hiệu rung động, chẳng hạn như biên độ, tần số, và các thông số thống kê, cùng với vai trò của chúng trong việc phản ánh trạng thái vận hành của cánh quạt. Đồng thời, chương này giới thiệu khái niệm và nguyên lý hoạt động của kỹ thuật Support Vector Machine (SVM), bao gồm các khía cạnh như hàm nhân, phân loại phi tuyến, và tối ưu hóa tham số, làm rõ lý do tại sao SVM phù hợp để phân loại các trạng thái lỗi. Ngoài ra, chương cũng đề cập đến các nghiên cứu liên quan trong lĩnh vực giám sát tình trạng, tập trung vào cách các phương pháp học máy đã được áp dụng để phát hiện hư hỏng trong các hệ thống cơ khí, đặc biệt là tua-bin gió. Những nội dung này cung cấp nền tảng lý thuyết cần thiết để hiểu và triển khai các phân tích trong chương tiếp theo.

Trong Chương 3, tôi giới thiệu quá trình phân tích hệ thống và xây dựng sản phẩm cốt lõi của nghiên cứu, đó là một mô hình học máy dựa trên Support Vector Machine để phân loại các trạng thái lỗi của cánh quạt tua-bin gió. Chương này bắt đầu bằng việc mô tả chi tiết tập dữ liệu rung động được sử dụng, bao gồm cách các tín hiệu được thu thập và các đặc trưng được trích xuất để làm đầu vào cho mô hình. Tiếp theo, chương

trình bày quy trình xây dựng mô hình SVM, từ việc tiền xử lý dữ liệu, lựa chọn tham số, đến huấn luyện và kiểm định mô hình trên các tập dữ liệu đại diện cho trạng thái không hư hỏng và các lỗi như xói mòn bề mặt, nứt gãy, mất cân bằng khối lượng, và biến dạng góc xoắn. Đặc biệt, chương phân tích hiệu suất của mô hình thông qua các chỉ số như độ chính xác, độ nhạy, và ma trận nhầm lẫn, đồng thời thảo luận về khả năng tổng quát hóa của mô hình trong các điều kiện vận hành khác nhau. Kết quả của chương là một hệ thống phân loại tự động, có khả năng hỗ trợ giám sát tình trạng trong môi trường phòng thí nghiệm, với tiềm năng mở rộng ứng dụng trong thực tế.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Nền tảng học máy dựa trên Support Vector Machine (SVM) cung cấp khung lý thuyết vững chắc để phân loại các trạng thái lỗi của cánh quạt tua-bin gió. Thuật toán này, với cơ chế tìm siêu phẳng tối ưu, tận dụng các khái niệm như vector hỗ trợ, lề, và hàm nhân để tách biệt các lớp dữ liệu. SVM xử lý hiệu quả bài toán phân loại nhị phân, đặc biệt với dữ liệu phi tuyến thường gặp trong tín hiệu phức tạp. Khả năng tối đa hóa khoảng cách lề giúp đảm bảo độ chính xác cao khi nhận diện các hư hỏng. Phương pháp này phù hợp để phân tích các mẫu tín hiệu có nhiều, hỗ trợ phát triển hệ thống phân loại đáng tin cậy. Lý thuyết SVM tạo cơ sở khoa học chặt chẽ cho việc xây dựng giải pháp nhận diện lỗi hiệu quả.

### 2.1. Định nghĩa SVM

SVM (Support Vector Machine) là một thuật toán học máy có giám sát, được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression), tuy nhiên, SVM được ứng dụng rộng rãi và hiệu quả nhất trong các bài toán phân loại.

Trong thuật toán này, dữ liệu được biểu diễn dưới dạng các điểm trong một không gian n chiều, trong đó n là số lượng các tính năng bạn có (hay còn gọi là các đặc trưng của dữ liệu). Mỗi đặc trưng tương ứng với một trục tọa độ, và một điểm dữ liệu sẽ có tọa độ dựa trên giá trị của các đặc trưng đó. Tập hợp các điểm dữ liệu này tạo thành một đồ thị trong không gian nhiều chiều.

Mục tiêu của thuật toán là tìm ra một siêu phẳng (hyperplane), hiểu đơn giản là một mặt phẳng (trong không gian 3 chiều) hoặc đường thẳng (trong không gian 2 chiều), sao cho siêu phẳng này có thể phân tách dữ liệu thuộc các lớp khác nhau một cách rõ ràng. Siêu phẳng được xem là tối ưu nếu nó phân chia hai lớp dữ liệu sao cho khoảng cách từ siêu phẳng đến các điểm dữ liệu gần nhất ở mỗi lớp là lớn nhất. Điều này giúp mô hình có khả năng phân loại tốt hơn và tổng quát hóa cao hơn trên dữ liệu chưa thấy.

Những điểm dữ liệu gần nhất với siêu phẳng, tức là những điểm “nằm sát biên” của hai lớp, được gọi là support vectors. Chúng đóng vai trò quan trọng trong việc xác định vị

trí và hướng của siêu phẳng phân chia, hoặc ta có thể nói Support Vectors là các điểm dữ liệu đặc biệt có ảnh hưởng trực tiếp đến việc xác định biên phân tách của mô hình.

Vì vậy, Support Vector Machine có thể được hiểu như là một mô hình xây dựng ra một đường biên tốt nhất giữa hai lớp bằng cách sử dụng chính các support vectors để phân chia khoảng cách giữa hai lớp một cách tốt nhất.

Các yếu tố được định nghĩa trong SVM:

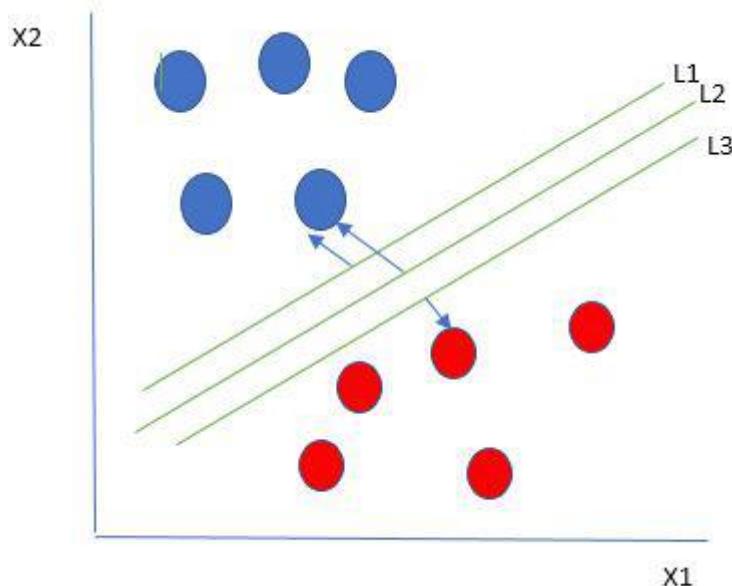
- Siêu mặt phẳng: Là một không gian con có chiều thấp hơn không gian đặc trưng một đơn vị.
- Vector hỗ trợ: Là những điểm dữ liệu gần nhất với siêu mặt phẳng phân chia.
- Lê: Là khoảng cách từ siêu mặt phẳng tới support vectors gần nhất của 2 lớp dữ liệu.
- Hard Margin: Là SVM yêu cầu phân chia dữ liệu một cách hoàn hảo, không cho phép bất kỳ điểm nào nằm sai phía của siêu mặt phẳng.
- Soft Margin: Cho phép một số điểm vi phạm margin.
- C: Tham số cân bằng giữa độ lớn của margin và lỗi phân loại.
- Kernel: dùng để biến đổi dữ liệu không tuyến tính thành một không gian có thể phân tách tuyến tính, các kernel phổ biến:
  - Linear Kernel: giữ nguyên không gian đặc trưng ban đầu của dữ liệu.
  - Polynomial Kernel: ánh xạ dữ liệu vào không gian bậc cao hơn.
  - RBF (Radial Basis Function) Kernel : ánh xạ điểm dữ liệu vào không gian vô hạn chiều.
  - Sigmoid Kernel: ánh xạ dữ liệu phi tuyến, dựa vào cơ chế hoạt động của một nơ-ron trong mạng nơ-ron nhân tạo.
- Hinge Loss: Là hàm mất mát dùng trong SVM: được tính theo:

$$\text{Hinge Loss} = \max(0, 1 - y(w \times x + b))$$

Với  $y$  là nhãn thực tế (+1 hoặc -1)

$(w^*x+b)$  là giá trị dự đoán

Ý tưởng chính của thuật toán SVM là tìm ra siêu mặt phẳng tách biệt tốt nhất 2 lớp bằng cách tối đa hóa biên độ giữa chúng. Lê chính là khoảng cách từ siêu mặt phẳng đến các điểm dữ liệu gần nhất (Vector hỗ trợ) ở mỗi bên.

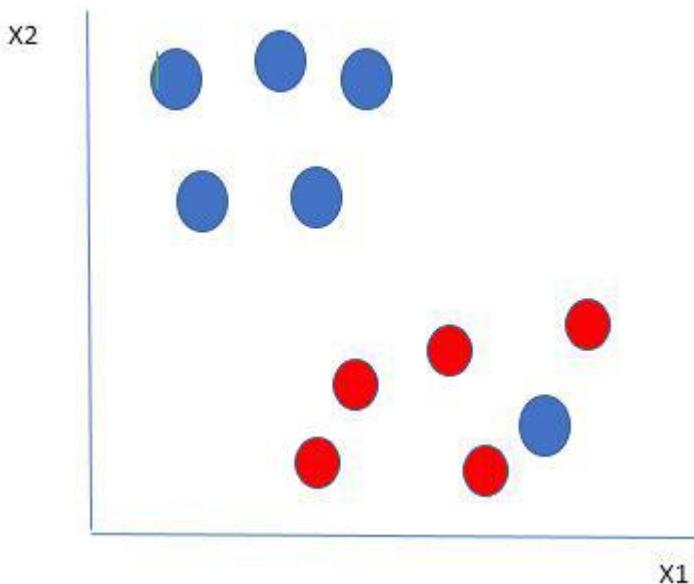


**Hình 2.1.1: Phân Chia Nhóm Dữ Liệu Bằng Siêu Phẳng**

Siêu mặt phẳng tốt nhất, hay còn gọi là Hard Margin, là siêu mặt phẳng tối đa hóa khoảng cách giữa siêu mặt phẳng và các điểm dữ liệu gần nhất từ cả hai lớp, như hình trên chúng ta sẽ chọn L2 làm hard margin.

## 2.2. Cách SVM phân loại dữ liệu

Giờ chúng ta sẽ xét bài toán:



**Hình 2.2.1: Các Điểm Dữ Liệu Trên Mặt Phẳng**

Ở tập hợp các chấm đỏ, chúng ta có thể thấy có một chấm màu xanh lam.

Chấm màu xanh lam trong tập hợp các chấm đỏ là một ngoại lệ. Thuật toán SVM có thể bỏ qua các điểm ngoại lệ và tìm siêu phẳng tốt nhất để tối đa hóa khoảng cách giữa các lớp.

Một Soft margin cho phép một số điểm dữ liệu bị phân loại sai hoặc vi phạm khoảng cách biên nhằm cải thiện khả năng tổng quát hóa của mô hình.

Thuật toán SVM tối ưu hóa phương trình để cân bằng giữa việc tối đa hóa biên và giảm thiểu hình phạt:

Công thức khái quát:

$$Objective\ Function = \frac{1}{margin} + \lambda \sum \text{penalty}$$

Với margin là khoảng cách biên,  $\lambda$  là hệ số điều chỉnh và penalty là điểm phạt, thường là hàm mất mát hinge loss:

- Nếu một điểm dữ liệu được phân loại đúng và nằm ngoài biên thì không bị phạt (mất mát = 0).

- Nếu một điểm bị phân loại sai hoặc vi phạm biên, thì hàm mất mát sẽ tăng tỷ lệ thuận với khoảng cách vi phạm.

Hệ số điều chỉnh  $\lambda$  là hệ số điều chỉnh có tác động với mô hình như sau:

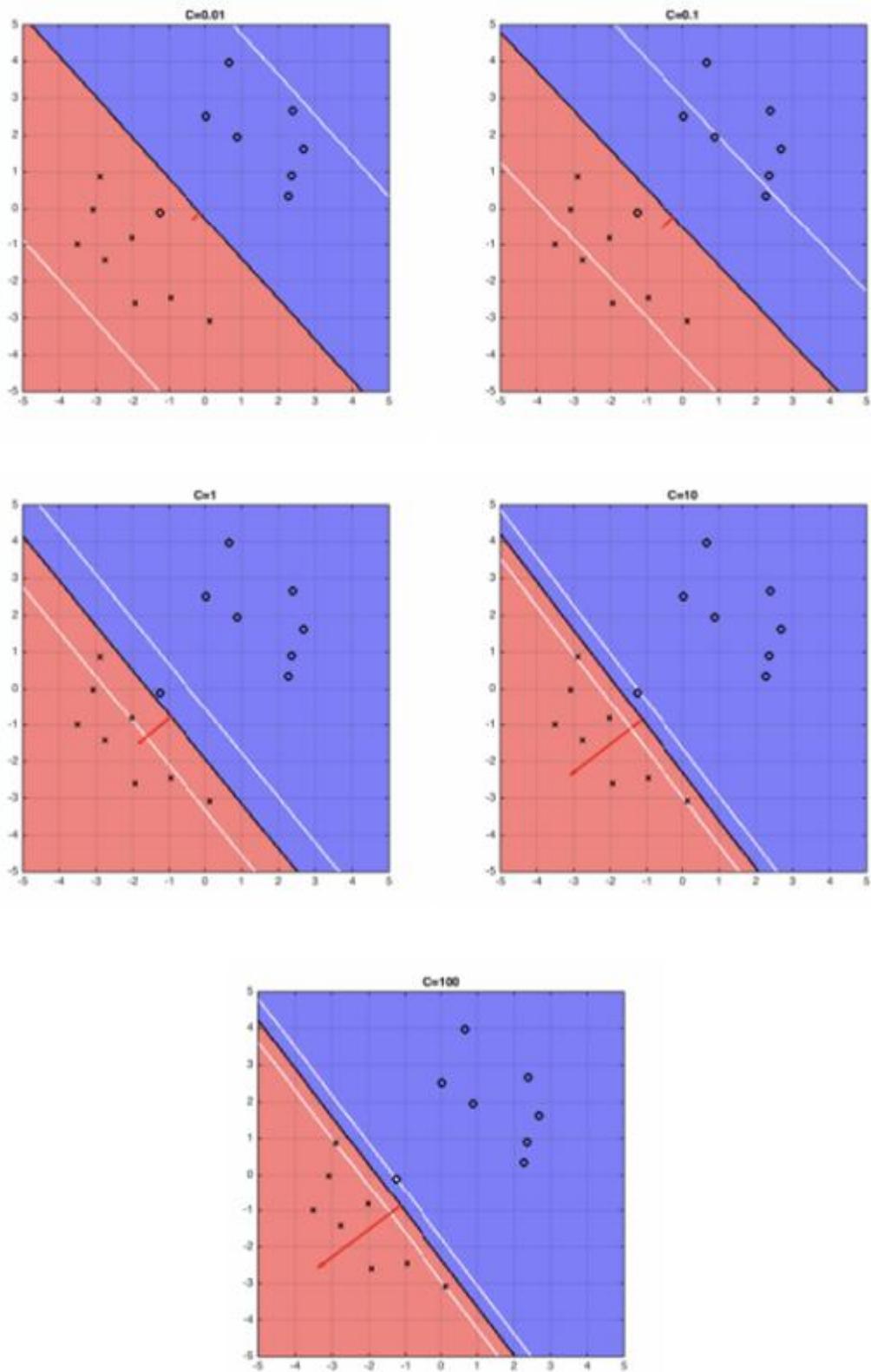
- $\lambda$  lớn: Mô hình ưu tiên việc có ít lỗi hơn, sẵn sàng chấp nhận biên nhỏ hơn để phân loại chính xác hơn.
- $\lambda$  nhỏ: Mô hình ưu tiên khoảng cách biên rộng, chấp nhận một số lỗi để cải thiện khả năng tổng quát hóa.

Công thức đầy đủ:

$$\min_{w, b, \varepsilon} \left( \frac{\|w\|^2}{2} + C \sum_{i=1}^n \varepsilon_i \right)$$

Trong đó:

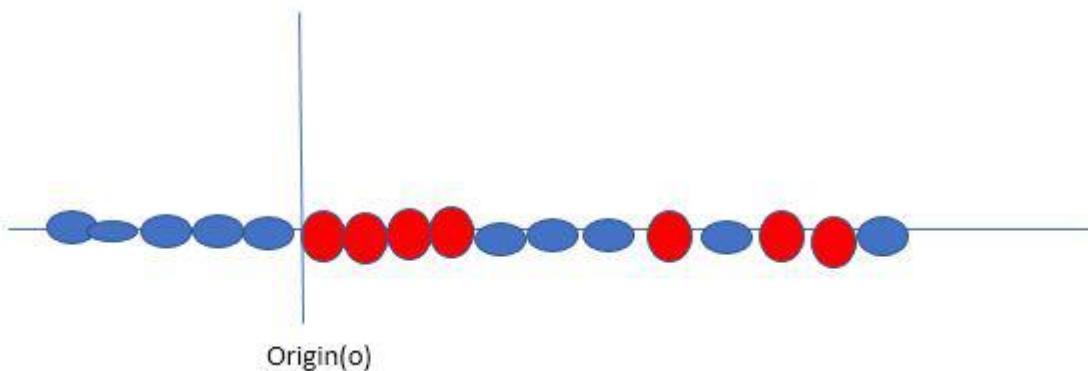
- $\frac{\|w\|^2}{2} = \frac{1}{margin}$
- $\varepsilon_i = \max(1 - y_i(w^T x_i + b), 0)$  : hinge – loss



**Hình 2.2.2: Tác Động Của Tham Số  $C$  Lên Siêu Phẳng**

Khi dữ liệu không thể phân tách bằng một đường thẳng, SVM sử dụng kỹ thuật Kernel(Hàm nhân) để ánh xạ dữ liệu sang không gian có số chiều cao hơn, nơi mà dữ liệu có thể được phân tách. Việc biến đổi này giúp SVM tìm được ranh giới quyết định(Decision boundary) kể cả với dữ liệu phi tuyến.

Kernel là gì là một hàm dùng để ánh xạ điểm dữ liệu vào một không gian có số chiều cao hơn mà không cần phải tính toán trực tiếp các tọa độ trong không gian đó nên SVM vẫn hoạt động hiệu quả với dữ liệu phi tuyến bằng cách ngầm thực hiện phép biến đổi mà không tốn nhiều tài nguyên.



*Hình 2.2.3: Siêu Phẳng Không Phân Nhóm Được Dữ Liệu*

### 2.3. Bài toán phân loại với SVM

Giả sử rằng bài toán phân loại nhị phân với 2 lớp dữ liệu được gán nhãn +1 và -1. Tập huấn luyện gồm các vector đặc trưng đầu vào X và nhãn lớp tương ứng Y.

Phương trình của Siêu phẳng tuyến tính có thể được viết là:

$$w^T \times x + b = 0$$

Với w là vector pháp tuyến với siêu phẳng(Tức là vector vuông góc với mặt phẳng phân chia).

B là hằng số dịch chuyển(bias) biểu thị khoảng cách từ gốc tọa độ đến siêu phẳng theo hướng vector w.

Ta có công thức tính khoảng cách từ một điểm dữ liệu đến siêu phẳng:

$$d_i = \frac{w^T x_i + b}{\|w\|}$$

Trong đó:

- $x_i$ : Là điểm dữ liệu cụ thể
- $w^T x_i + b$ : Là giá trị hàm phân biệt(Decision function) cho điểm đó.
- $\|w\|$ : Là chuẩn Euclid(độ dài) của vector w, tức là:

$$\|w\| = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$$

Điều kiện phân tách dữ liệu:

Để mô hình có thể phân loại đúng các điểm dữ liệu, cần thỏa mãn điều kiện phân tách sau đối với mỗi điểm dữ liệu  $(x_i, y_i)$ , trong đó  $y_i \in \{-1, +1\}$ :

$$y_i(w^T x_i + b) \geq 1$$

Chi tiết:

- Nếu  $y_i = +1$  thì  $w^T x_i + b \geq 1$
- Nếu  $y_i = -1$  thì  $w^T x_i + b \leq -1$

$\Rightarrow$  Điều này đảm bảo rằng tất cả các điểm đều nằm ở ngoài biên (margin) và không nằm giữa hai biên, từ đó tối đa hóa khoảng cách giữa hai lớp.

Nhận đầu ra  $\hat{y}$  cho một điểm dữ liệu được xác định bởi:

$$\hat{y} = \begin{cases} 1 & \text{nếu } w^T x_i + b \geq 0 \\ 0 & \text{nếu } w^T x_i + b < 0 \end{cases}$$

Nếu đầu ra của hàm phân biệt  $\geq 0$ , điểm đó thuộc lớp 1.

Nếu đầu ra  $< 0$ , điểm đó thuộc lớp 0.

## 2.4. SVM Kernel RBF

### 2.4.1. Tại sao phải dùng RBF Kernel?

Trong trường hợp dữ liệu không thể phân tách tuyến tính, SVM sử dụng kỹ thuật kernel để ánh xạ dữ liệu sang không gian đặc trưng có chiều cao hơn, nơi mà dữ liệu có thể được phân tách tuyến tính.

RBF Kernel hay còn được gọi là Gaussian kernel, là một trong những hàm hạt nhân được sử dụng rộng rãi nhất. Nó hoạt động bằng cách đo độ tương đồng giữa các điểm dữ liệu dựa trên khoảng cách Euclid của chúng trong không gian đầu vào.

### 2.4.2. Định nghĩa

Kernel RBF được định nghĩa như sau:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

Trong đó:

- $x, x'$  : Hai điểm dữ liệu trong không gian đầu vào.
- $\|x - x'\|^2$ : Bình phương khoảng cách Euclid giữa hai điểm.
- $\gamma$ : Tham số điều chỉnh độ ảnh hưởng của một điểm dữ liệu đơn lẻ. Giá trị nhỏ của  $\gamma$  tương ứng với ảnh hưởng lớn (hàm kernel rộng), trong khi giá trị lớn của  $\gamma$  tương ứng với ảnh hưởng nhỏ (hàm kernel hẹp).

### 2.4.3. Biến Đổi Các Thuật Toán Tuyến Tính Thành Bộ Phân Loại và Hồi Quy Phi Tuyến

Khi chúng ta áp dụng các thuật toán tuyến tính như Perceptron hoặc Hồi quy tuyến tính lên kernel RBF, thực chất ta đang áp dụng những thuật toán này lên các điểm dữ liệu đã được ánh xạ sang một không gian mới với số chiều là vô hạn. Trong không gian đặc trưng này, mô hình sẽ học được một siêu phẳng tuyến tính, nhưng khi chuyển ngược lại không gian ban đầu, nó trở thành một đường phân cách hoặc đường hồi quy phi tuyến cực mạnh.

$$a_1x^{\infty} + a_2x^{\infty-1} + \dots + a_2x + c$$

Nghĩa là, mặc dù bản chất mô hình vẫn là tuyến tính trong không gian đặc trưng, nhưng khi phản chiếu lại không gian gốc, kết quả là một hàm phi tuyến dưới dạng đa thức bậc vô hạn. Chính điều này khiến Radial Basis Function (RBF) kernel trở thành một công cụ cực kỳ mạnh mẽ, có khả năng khớp với những tập dữ liệu có ranh giới phức tạp và phi tuyến cao.

#### 2.4.4. Vai trò của Kernel RBF trong SVM

Kernel RBF cho phép SVM tạo ra siêu phẳng phân tách trong không gian đặc trưng có chiều cao hơn mà không cần tính toán trực tiếp tọa độ trong không gian đó. Điều này giúp SVM xử lý hiệu quả các bài toán phân loại dữ liệu.

#### 2.4.5. Tham số $\gamma$

$\gamma$ : điều chỉnh ảnh hưởng của mỗi điểm dữ liệu đến mô hình.

- Giá trị nhỏ của  $\gamma$ : Khi  $\gamma$  nhỏ, ảnh hưởng của mỗi điểm dữ liệu lan rộng ra, làm cho mô hình có lè rộng hơn. Điều này có thể dẫn đến underfitting vì mô hình không phân biệt rõ ràng các lớp và không phản ánh đầy đủ các đặc điểm của dữ liệu.
- Giá trị lớn của  $\gamma$ : Khi  $\gamma$  lớn, ảnh hưởng của mỗi điểm dữ liệu sẽ hẹp hơn, khiến mô hình tập trung vào các điểm gần nhau, tạo ra lè hẹp hơn. Điều này có thể dẫn đến overfitting vì mô hình quá nhạy cảm với các biến động nhỏ trong dữ liệu huấn luyện và có thể không tổng quát tốt được cho dữ liệu mới.

Tối ưu giá trị  $\gamma$  giúp ta tìm ra sự cân bằng giữa việc tránh underfitting và overfitting để mô hình hoạt động hiệu quả nhất.

## CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM

### 3.1. Phân tích dữ liệu đầu vào

#### 3.1.1. Giới thiệu tập dữ liệu

Tập dữ liệu được trình bày trong tạp chí Data in Brief số 49, tháng 8 năm 2023. Tập dữ liệu tập trung vào dữ liệu rung động của các cánh quạt gió bị lỗi, bao gồm các cơ chế rung động phổ biến liên quan đến nhiều loại lỗi và điều kiện vận hành như tốc độ gió. Các lỗi được đề cập gồm: Xói mòn bề mặt, cánh bị nứt, mất cân bằng khối lượng, cánh bị vặn. Dữ liệu được thu thập tại các tốc độ gió khác nhau, từ 1.3 đến 5.3 m/s.

Tập dữ liệu về rung động của các cánh quạt gió bị lỗi là tài nguyên quan trọng để xác thực các phương pháp giám sát tình trạng trong các ngành công nghiệp liên quan đến quạt gió. Ngoài ra còn giúp các nhà nghiên cứu tìm hiểu rõ hơn về đặc điểm của tín hiệu rung động liên quan đến từng loại lỗi. Từ đó cải thiện hiệu quả phân tích và giúp ích trong công việc bảo trì.

Các lỗi cụ thể trong tập dữ liệu:

- Xói mòn bề mặt (Surface erosion).
- Cánh bị nứt (cracked blade).
- Mất cân bằng khối lượng (mass imbalance).
- Cánh bị vặn (twist blade fault).

Tập dữ liệu bao gồm dữ liệu rung động một trực từ các quạt gió hoạt động ở nhiều điều kiện tải khác nhau do tốc độ gió thay đổi, tổng cộng 35 file. Bao gồm:

Bảng 3.1.1: Phân Loại Các Tập Dữ Liệu

Loại trạng thái	Tốc độ gió (m/s)	Số file
Bình thường	1.3,2.3,3.2,3.7,4.5,5,5.6	7
Cánh bị nứt	1.3,2.3,3.2,3.7,4.5,5,5.3	7
Xói mòn bề mặt	1.3,2.3,3.3,3.7,4.5,5,5.3	7
Mất cân bằng khối lượng	1.3,2.3,3.2,3.7,4.5,5,5.3	7

Cánh bị vỡ	1.3,2,3.2,4,4.7,5,5.3	7
------------	-----------------------	---

### 3.1.2. Mô tả về tập dữ liệu

Vì có 35 file, ta cần gộp lại 1 file csv để có thể dễ xử lý.

```
# Gán nhãn và lưu thông tin file
file_label_map = {}

for file in files:
    file_lower = file.lower()
    if "h-" in file_lower or "h for" in file_lower:
        label = 0 # Healthy
    elif "erosion" in file_lower:
        label = 1 # Surface Erosion
    elif "crack" in file_lower:
        label = 2 # Cracked Blade
    elif "unbalance" in file_lower or "unbalance" in file_lower:
        label = 3 # Mass Imbalance
    elif "twist" in file_lower or "twist" in file_lower:
        label = 4 # Twist Blade Fault
    else:
        label = -1 # Nhãn không xác định
    file_label_map[file] = label
```

Ta gán nhãn dựa trên tên file:

Bảng 3.1.2: Phân Loại Nhãn Đầu Ra

Tình trạng cánh quạt	Nhãn
Bình thường	0
Xói mòn bề mặt	1
Cánh bị nứt	2
Mất cân bằng khối lượng	3

Tiếp theo, ta đọc tất cả 35 file và gộp chúng lại thành 1:

```
data = []

for file, label in file_label_map.items():
    file_path = os.path.join(folder_path, file)
    if not os.path.exists(file_path):
        print(f"File không tồn tại: {file_path}")
        continue

    print(f"Đang xử lý file: {file}")
    # Đọc file
    try:
        if file.endswith('.csv'):
            df = pd.read_csv(file_path)
        elif file.endswith('.xlsx'):
            df = pd.read_excel(file_path)
        else:
            print(f"Không hỗ trợ định dạng file: {file}")
            continue
    except Exception as e:
        print(f"Lỗi khi đọc file {file}: {e}")
        continue
```

Sau khi đọc file để gộp, ta cần chuẩn hóa dữ liệu:

```
# Chuẩn hóa dữ liệu

# Kiểm tra các cột trong file
columns = df.columns.tolist()

print(f"Các cột trong file {file}: {columns}") # Thêm dòng này để kiểm tra cột
```

```

if "Time - Voltage_1" in df.columns and "Amplitude - Voltage_1" in df.columns:
    # File .xlsx có cột riêng lẻ (như twist.xlsx)
    df = df.rename(columns={"Time - Voltage_1": "time", "Amplitude - Voltage_1": "amplitude"})
    df["amplitude"] = df["amplitude"].astype(float) * 10 # Chuyển từ Voltage sang g

elif "Time - sec" in df.columns and "Amplitude - g" in df.columns:
    # File .csv có cột riêng lẻ
    df = df.rename(columns={"Time - sec": "time", "Amplitude - g": "amplitude"})
    df["amplitude"] = df["amplitude"].astype(float) # Đã ở đơn vị g

elif "Time - Voltage_1;Amplitude - Voltage_1" in df.columns:
    # File có cột ghép (dạng chuỗi)
    time_amp = df["Time - Voltage_1;Amplitude - Voltage_1"].str.split(";", expand=True)
    time_amp = time_amp.dropna() # Loại bỏ dòng có giá trị NaN
    time_amp = time_amp[time_amp[0].str.strip() != ""] # Loại bỏ dòng có time rỗng
    time_amp = time_amp[time_amp[1].str.strip() != ""] # Loại bỏ dòng có amplitude rỗng

    if time_amp.empty:
        print(f"File {file} không có dữ liệu hợp lệ sau khi lọc.")
        continue

    try:
        df = df.loc[time_amp.index] # Chỉ giữ các dòng hợp lệ
        df["time"] = time_amp[0].astype(float)
        df["amplitude"] = time_amp[1].astype(float) * 10 # Chuyển từ Voltage sang g
    except Exception as e:

```

```

print(f'Lỗi khi chuẩn hóa dữ liệu trong file {file}: {e}')
continue

elif "Time - sec;Amplitude - g" in df.columns:
    # File có cột ghép (dạng chuỗi)
    time_amp = df["Time - sec;Amplitude - g"].str.split(";", expand=True)
    time_amp = time_amp.dropna()
    time_amp = time_amp[time_amp[0].str.strip() != ""]
    time_amp = time_amp[time_amp[1].str.strip() != ""]
    if time_amp.empty:
        print(f'File {file} không có dữ liệu hợp lệ sau khi lọc.')
        continue

    try:
        df = df.loc[time_amp.index]
        df["time"] = time_amp[0].astype(float)
        df["amplitude"] = time_amp[1].astype(float) # Đã ở đơn vị g
    except Exception as e:
        print(f'Lỗi khi chuẩn hóa dữ liệu trong file {file}: {e}')
        continue

    else:
        print(f'File {file} không có cột dữ liệu phù hợp (Time và Amplitude).')
        continue

```

Cụ thể, trong file dataset gốc ta có 2 cột “Time-Voltage\_1” và “Amplitude – Voltage\_1”. Ta chuyển tên cột về time và amplitude, đồng thời nhân Amplitude với 10 để đổi từ đơn vị Volt sang gia tốc.

Đối với Time-sec và Amplitude – g, ta cũng để tên cột là time và amplitude nhưng giữ nguyên vì đã amplitude đã là gia tốc rồi.

Có những cột chứa 2 giá trị bị ghép vào một chuỗi, ta sẽ phải tách ra(Trong trường hợp đơn vị là voltage), sau đó cũng nhân amplitude với 10.

Còn với trường hợp Time-sec;Amplitude-g, ta cũng tách ra nhưng không nhân với 10.

Ở trong tài liệu về dataset có nhắc đến thông tin về cảm biến gia tốc (accelerometer): “It has a sensitivity of  $\pm 10\% 100mV/g$ ”. It ở đây là PCB Piezotronics 352C65 – Cảm biến gia tốc được sử dụng trong bài báo khoa học.

Sensitivity = 100mV/g có nghĩa là với 1g(gia tốc trọng trường) sẽ cho ra 0.1V(100mV). Hay ngược lại  $1V = 10 g$ . Hay ta có thể tính theo công thức:

$$a_g = \frac{V_{(mV)}}{100} = V_{(V)} \times 10$$

Sau đó, ta trích xuất đặc trưng tốc độ gió từ tên file và gộp thành một dataframe:

```
# Trích xuất tốc độ gió
match = re.search(r"vw(ind)?=([\d\.\.]+)", file.lower())
if match:
    wind_speed_str = match.group(2).rstrip('.')
    try:
        wind_speed = float(wind_speed_str)
    except ValueError:
        print(f"Không thể chuyển đổi tốc độ gió từ file {file}: {wind_speed_str}")
        wind_speed = 0.0
    else:
        print(f"Không tìm thấy tốc độ gió trong file {file}")
        wind_speed = 0.0
    df["wind_speed"] = wind_speed

# Chỉ giữ các cột cần thiết
df = df[["time", "amplitude", "label", "wind_speed"]]
data.append(df)
```

```

# Gộp thành một DataFrame
if data:
    merged_data = pd.concat(data, ignore_index=True)

    # Kiểm tra dữ liệu gộp
    print("\nThông tin dữ liệu gộp:")
    print(merged_data.info())
    print("\nMẫu dữ liệu đầu tiên:")
    print(merged_data.head())

# Lưu dữ liệu gộp trong cùng thư mục với file Python
output_path = os.path.join(script_dir, "merged_wind_turbine_data.csv")
merged_data.to_csv(output_path, index=False)
print(f"\nĐã lưu dữ liệu gộp vào '{output_path}'")

else:
    print("Không có dữ liệu nào được đọc. Vui lòng kiểm tra lại các file.")

```

Ta trích xuất tốc độ gió và kiểm tra lỗi, cuối cùng ta giữ lại các cột cần thiết bao gồm: time (thời gian đo), amplitude (biên độ), label (nhãn), wind\_speed (tốc độ gió). Sau đó ta gộp dữ liệu thành một dataframe lớn và xuất ra file CSV có tên là merge\_wind\_turbine\_data.csv.

Phân tích Merge\_wind\_turbine\_data.csv:

*Bảng 3.1.3: Thuộc Tính Của Tập Dữ Liệu Tổng Hợp*

Tên cột	Ý nghĩa	Nội dung
Time	Thời gian	Thời gian(giây) từ 0 giây đến 0.499 với bước nhảy 0.001 giây tương ứng 500 mẫu mỗi đoạn dữ liệu

Amplitude	Biên độ tín hiệu rung	Được chuyển đổi qua đơn vị gia tốc
Label	Nhãn phân loại dựa trên trạng thái của turbine	Là các giá trị số từ 0 đến 4 dựa trên tình trạng của cánh quạt
Wind_speed	Tốc độ gió	Có đơn vị m/s, được tính xuất từ tên file của các dataset gốc

### 3.2. Xây dựng mô hình dự đoán lỗi của cách quạt gió

#### 3.2.1. Quy trình xây dựng mô hình

Link GitHub của dự án: <https://github.com/HueyAnthonyDisward/Propeller-Fault-Prediction-Model-using-SVM-RBF>

Ta sử dụng dataset Merge\_wind\_turbine\_data.csv đã xử lý ở trên, đầu tiên ta sẽ khai báo các thư viện cần sử dụng.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV,
learning_curve, cross_val_score, StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.decomposition import PCA
from scipy import stats, fft
import random
import matplotlib.pyplot as plt
import seaborn as sns
```

pandas: Thư viện dùng để thao tác và phân tích dữ liệu, cung cấp các cấu trúc dữ liệu để xử lý dữ liệu dạng bảng.

numpy: Thư viện hỗ trợ tính toán số học trên mảng đa chiều, cung cấp các hàm toán học và thao tác ma trận nhanh.

sklearn.preprocessing.StandardScaler: Thư viện chuẩn hóa dữ liệu, đưa các đặc trưng về cùng thang đo để cải thiện hiệu suất mô hình.

sklearn.model\_selection:

- train\_test\_split: Chia dữ liệu thành tập huấn luyện và tập kiểm tra
- GridSearchCV: Tìm kiếm siêu tham số tối ưu cho mô hình bằng cách thử nghiệm tổ hợp các tham số
- learning\_curve: Đánh giá hiệu suất mô hình trên các kích thước tập huấn luyện khác nhau để vẽ đường cong học tập
- cross\_val\_score: Thực hiện kiểm tra chéo để đánh giá độ chính xác của mô hình.
- StratifiedKFold: Chia dữ liệu thành các fold có tỷ lệ lớp cân bằng để kiểm định chéo.

sklearn.svm.SVC: Lớp triển khai mô hình Support Vector Classifier (SVM) để phân loại dữ liệu.

sklearn.metrics:

- accuracy\_score: Tính độ chính xác của mô hình dựa trên nhãn dự đoán và nhãn thực tế.
- classification\_report: Tạo báo cáo chi tiết về các chỉ số(precision, recall, F1-Score) cho từng lớp.

sklearn.decomposition.PCA: Thực hiện phân tích các thành phần chính (PCA) để giảm chiều dữ liệu, giữ lại các thành phần quan trọng nhất.

scipy:

- stat: Cung cấp các hàm thống kê.

- Fft: Thực hiện biến đổi Fourier(Fast Fourier Transform) để phân tích tín hiệu trong miền tần số.

random: Tạo số ngẫu nhiên.

matplotlib.pyplot: Vẽ biểu đồ, trực quan hóa dữ liệu như đường cong học tập, biểu đồ phân tán,...

Tiếp theo, ta cần xây dựng hàm trích xuất đặc trưng từ chuỗi tín hiệu từ biên độ và tốc độ gió. Các đặc trưng gồm miền thời gian và miền tần số, được sử dụng trong phân tích tín hiệu.

Hàm trích xuất đặc trưng từ tín hiệu

```
def extract_features(sequence):
    amplitude = sequence[:, 0] # Cột amplitude
    wind_speed = sequence[:, 1] # Cột wind_speed

    # Đặc trưng miền thời gian
    features = []
    features.append(np.mean(amplitude))
    features.append(np.std(amplitude))
    features.append(np.max(amplitude))
    features.append(np.min(amplitude))
    features.append(stats.skew(amplitude))
    features.append(stats.kurtosis(amplitude))
    features.append(np.mean(wind_speed))
    features.append(np.std(wind_speed))

    # Thêm đặc trưng mới: Tỷ lệ biến thiên biên độ
    amplitude_diff = np.diff(amplitude)
    amplitude_variation_ratio = np.std(amplitude_diff) /
        (np.mean(np.abs(amplitude)) + 1e-10)
```

```

features.append(amplitude_variation_ratio)

# Đặc trưng miền tần số
fft_vals = np.abs(fft.fft(amplitude))
fft_vals = fft_vals[:len(fft_vals) // 2]
peak_freq = np.argmax(fft_vals)
features.append(peak_freq)
spectral_energy = np.sum(fft_vals ** 2)
features.append(spectral_energy)
fft_probs = fft_vals / np.sum(fft_vals)
spectral_entropy = -np.sum(fft_probs * np.log2(fft_probs + 1e-10))
features.append(spectral_entropy)

return np.array(features)

```

Đầu vào:

Sequence: Một mảng 2D(Numpy array) với 2 cột:

- Cột 0: amplitude(biên độ của tín hiệu).
- Cột 1: wind\_speed(tốc độ gió).

Hàm giả định rằng mỗi hàng của sequence đại diện cho một điểm dữ liệu theo thời gian.

Quá trình trích xuất đặc trưng:

Tách dữ liệu:

Ta tách 2 cột thành 2 mảng riêng biệt để xử lý độc lập.

Sau đó ta tạo danh sách các feature để lưu các đặc trưng, các đặc trưng của miền thời gian được tính như sau:

- np.mean(amplitude): Trung bình của biên độ, biểu thị mức độ trung tâm của tín hiệu
- np.std(amplitude): Độ lệch chuẩn của biên độ, đo lường độ phân tán của tín hiệu
- np.max(amplitude): Giá trị biên độ lớn nhất, biểu thị mức cao nhất của tín hiệu.
- np.min(amplitude): Giá trị biên độ nhỏ nhất, biểu thị mức thấp nhất của tín hiệu.
- stats.skew(amplitude): Độ lệch của biên độ, đo lường mức độ bất đối xứng của phân phối dữ liệu.
- stats.kurtosis(amplitude): Độ nhọn của biên độ, đo lường mức độ tập trung của dữ liệu quanh giá trị trung bình.
- np.mean(wind\_speed): Trung bình của tốc độ gió.
- np.std(wind\_speed): Độ lệch chuẩn của tốc độ gió.

Tỷ lệ biến thiên biên độ:

amplitude\_variation\_ratio: Tỷ lệ giữa độ lệch chuẩn của sai phân và trung bình của giá trị tuyệt đối biên độ (cộng 1e-10 để tránh chia cho 0). Đặc trưng này đo lường mức độ dao động của tín hiệu so với giá trị trung bình của nó.

Đầu tiên, chúng ta sẽ ký hiệu:

$$a = [a_1, a_2, \dots, a_n]: dãy biên độ(amplitude)$$

$$\Delta a_i = a_{i+1} - a_i: sai phân của biên độ$$

$$std(\Delta a): Độ lệch chuẩn của sai phân$$

$$mean(|a|): Giá trị trung bình của trị tuyệt đối biên độ$$

Công thức cho amplitude\_variation\_ratio là:

$$AVR = \frac{\sigma(\Delta a)}{\mu(|a|) + \varepsilon}$$

Trong đó:

$$\sigma(\Delta a) = \sqrt{\frac{1}{n-2} \sum_{i=1}^{n-1} (\Delta a_i - \Delta \bar{a})^2}$$

$$\Delta \bar{a} = \frac{1}{n-1} \sum_{i=1}^{n-1} \Delta a_i$$

$$\mu(|a|) = \frac{1}{n} \sum_{i=1}^n |a_i|$$

$\epsilon = 10^{-10}$ : Hàng số rất nhỏ để tránh chia cho 0

Đặc trưng miền tần số: Các đặc trưng miền tần số được tính dựa trên biến đổi Fourier nhanh(FFT) của biên độ:

Định nghĩa phép biến đổi Fourier nhanh:

Cho một tín hiệu biên độ rời rạc:

$$a = [a_0, a_1, \dots, a_{N-1}]$$

Phép biến đổi Fourier nhanh:

$$A_k = \sum_{n=0}^{N-1} a_n * e^{-2\pi * \frac{kn}{N}}, \quad k = 0, 1, N-1$$

Vậy ta sẽ có kết quả  $A = [A_0, A_1, \dots, A_{N-1}]$ .

Khi ta thực hiện biến đổi FFT trên tín hiệu thì FFT  $A_k$  là đối xứng hermitian.

$$A_k = \overline{A_{N-k}}$$

Nửa sau của phổ tần số là phản chiếu phức hợp của nửa đầu(Vì phổ tần số là số phức), tức là các thông tin về biên độ đã được đày đủ trong nửa đầu phổ. Do đó ta chỉ giữ lại nửa đầu để phân tích

Ở các miền tần số, ta tìm tần số đỉnh(Peak Frequency). Hay là tần số chính của tín hiệu.

Là chỉ số  $k$  mà tại đó độ lớn của phổ đạt cực đại:

$$peak\ frequency = \underset{k}{\operatorname{argmax}}(|A_k|), k \in \left[0, \left\lfloor \frac{N}{2} - 1 \right\rfloor\right]$$

Trong đó:  $A_k$  là độ lớn phô FFT tại tần số k.

Đặc trưng thứ 2 là năng lượng phô(Spectral Energy), đó chính là tổng bình phương các giá trị FFT, đo lường tổng năng lượng của tín hiệu trong miền tần số:

$$Spectral\ Energy = \sum_{k=0}^{\left\lfloor \frac{N}{2} \right\rfloor - 1} |A_k|^2$$

Đặc trưng thứ 3 chính là Entropy: Đầu tiên, ta chuẩn hóa các giá trị FFT thành xác suất(fft\_probs). Sau đó ta tính entropy của phô tần số để đo lường mức độ “hỗn loạn” hoặc phân bố năng lượng trong các tần số. Giá trị entropy cao hơn cho thấy năng lượng phân bố đều hơn.

$$p_k = \frac{|A_k|}{\sum_{j=0}^{\left\lfloor \frac{N}{2} \right\rfloor - 1} |A_j|}$$

$$Spectral\ Entropy = - \sum_{k=0}^{\left\lfloor \frac{N}{2} \right\rfloor - 1} p_k * \log_2(p_k + \varepsilon), \text{ với } \varepsilon = 10^{-10} \text{ để tránh log của 0}$$

Cuối cùng ta trả về một mảng 1 chiều chứa 12 đặc trưng(Bao gồm 9 đặc trưng về miền thời gian và 3 đặc trưng về miền tần số).

Tiếp theo, ta tiến hành đọc dữ liệu, tái cấu trúc dữ liệu và kiểm tra phân bố nhãn:

```
# Đọc dữ liệu
df = pd.read_csv("merged_wind_turbine_data.csv")

# Tái cấu trúc dữ liệu thành các chuỗi
sequences = []
labels = []
```

```

sequence_length = 500

for i in range(0, len(df), sequence_length):
    sequence_data = df.iloc[i:i + sequence_length]
    if len(sequence_data) == sequence_length:
        sequence = sequence_data[['amplitude', 'wind_speed']].values
        label = sequence_data['label'].iloc[0]
        sequences.append(sequence)
        labels.append(label)

sequences = np.array(sequences)
labels = np.array(labels)

# Kiểm tra phân bố nhãn ban đầu
print("Phân bố nhãn ban đầu:")
print(pd.Series(labels).value_counts())

```

Ta tạo sequences: Danh sách lưu các chuỗi dữ liệu, mỗi chuỗi là một mảng 2D chứa các giá trị amplitude và wind\_speed.

Labels: Danh sách lưu nhãn tương ứng với mỗi chuỗi.

Sequence\_length = 500: độ dài cố định của mỗi cho mỗi chuỗi(500 điểm dữ liệu liên tiếp).

Tiếp theo ta tạo vòng lặp for để duyệt qua DataFrame với 500 bước, kiểm tra đoạn dữ liệu có đúng 500 hàng. Sau đó ta trích xuất đặc trưng bằng cách lấy các cột amplitude và wind\_speed của đoạn dữ liệu và chuyển thành mảng NumPy.

label = sequence\_data['label'].iloc[0]: Lấy nhãn của điểm dữ liệu đầu tiên trong đoạn làm nhãn cho toàn bộ chuỗi. Sau đó ta chuyển danh sách thành các mảng numpy và kiểm tra phân bố nhãn ban đầu.

Vì đã biến đổi dữ liệu thành các chuỗi, nên ta cần phải tăng cường dữ liệu trước khi đưa vào mô hình:

```
# Tăng cường dữ liệu
augmented_sequences = []
augmented_labels = []
noise_factor = 0.02
shift_max = 50
scale_factor_range = (0.9, 1.1)

for i in range(len(sequences)):
    for _ in range(50): # Tăng lên 50 biến thể
        # Thêm nhiễu
        noise = np.random.normal(0, noise_factor, sequences[i].shape)
        seq_noisy = sequences[i] + noise
        # Dịch chuyển thời gian
        shift = random.randint(-shift_max, shift_max)
        seq_shifted = np.roll(seq_noisy, shift, axis=0)
        # Phóng to/thu nhỏ biên độ
        scale = random.uniform(scale_factor_range[0], scale_factor_range[1])
        seq_scaled = seq_shifted.copy()
        seq_scaled[:, 0] = seq_scaled[:, 0] * scale
        # Đảo ngược tín hiệu
        if random.random() > 0.5:
            seq_scaled = np.flip(seq_scaled, axis=0)
        augmented_sequences.append(seq_scaled)
        augmented_labels.append(labels[i])

augmented_sequences = np.array(augmented_sequences)
augmented_labels = np.array(augmented_labels)
```

Đầu tiên, ta khởi tạo:

Augmented\_sequences: Danh sách lưu các chuỗi dữ liệu sau khi được tăng cường.

Augmented\_labels: Danh sách lưu nhãn tương ứng với các chuỗi tăng cường.

Tham số tăng cường:

- noise\_factor = 0.02: Độ lớn của nhiễu Gaussian được thêm vào tín hiệu.
- shift\_max = 50: Giới hạn tối đa cho số điểm dịch chuyển (thời gian).
- scale\_factor\_range = (0.9,1.1): Phạm vi tỷ lệ phóng to/thu nhỏ biên độ (90% đến 110% giá trị gốc).

Mỗi chuỗi gốc, ta tạo ra 50 biến thể, trong vòng lặp ta thực hiện:

Thêm nhiễu (Add noise):

Ta tạo nhiễu Gaussian với trung bình 0 và độ lệch chuẩn noise\_factor = 0.02 có cùng kích thước với chuỗi gốc sequences.

Seq\_noisy = sequences[i] + noise: Thêm nhiễu vào chuỗi gốc (Cả amplitude và wind\_speed).

Việc thêm nhiễu mô phỏng các biến động ngẫu nhiên trong dữ liệu thực tế, giúp mô hình học cách bỏ qua các biến động nhỏ và tập trung vào các đặc trưng chính của tín hiệu.

Dịch chuyển thời gian (Time shifting):

Ta chọn ngẫu nhiên một giá trị dịch chuyển trong khoảng [-50,50]. Sau đó dịch chuyển chuỗi theo trục thời gian với số bước là shift.

Việc dịch chuyển thời gian giúp mô hình không phụ thuộc vào vị trí chính xác của các mẫu trong chuỗi.

Phóng to/thu nhỏ biên độ (Amplitude Scaling):

Ta chọn ngẫu nhiên tỷ lệ phóng to/thu nhỏ trong khoảng tỉ lệ, nhân cột amplitude với tỷ lệ scale đã chọn ngẫu nhiên, cột wind\_speed sẽ không bị thay đổi.

Việc phóng to/thu nhỏ biên độ sẽ mô phỏng các biến đổi về cường độ tín hiệu, giúp mô hình tập trung vào hình dạng của tín hiệu hơn là giá trị tuyệt đối

Đảo ngược tín hiệu (Signal Reversal):

Với xác suất 50%, chuỗi sẽ đảo ngược thứ tự theo trực thời gian, đảo ngược thứ tự các hàng của chuỗi (Từ đầu đến cuối).

Cuối cùng ta lưu trữ chuỗi đã tăng cường và nhãn tương ứng. Sau đó ta kiểm tra phân bố nhãn sau tăng cường.

Tiếp theo, ta trích xuất đặc trưng và chuẩn hóa cùng với giảm chiều dữ liệu với PCA:

```
X_features = np.array([extract_features(seq) for seq in augmented_sequences])

# Chuẩn hóa và giảm chiều dữ liệu với PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_features)
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X_scaled)
print(f"Số đặc trưng sau PCA: {X_reduced.shape[1]}")

# Chia dữ liệu
X_train, X_test, y_train, y_test = train_test_split(X_reduced, augmented_labels,
test_size=0.2, random_state=42, stratify=augmented_labels)
```

Ta sử dụng hàm extract\_features ở đầu chương trình để trích xuất các đặc trưng giúp mô hình dễ xử lý hơn. Các đặc trưng này nắm bắt các thuộc tính thống kê và tần số của tín hiệu, làm tăng khả năng phân tách giữa các lớp(nhãn).

Tiếp theo ta chuẩn hóa dữ liệu bằng StandardScaler() và giảm chiều dữ liệu với PCA, khởi tạo PCA để giữ lại các thành phần chính sao cho tổng phương sai được giải thích đạt ít nhất 95% phương sai của dữ liệu gốc. sau đó áp dụng PCA lên dữ liệu đã được chuẩn hóa để tạo X\_reduced là mảng với số cột(Đặc trưng giảm xuống). Sau đó ta chia dữ liệu thành tập huấn luyện(80%) và tập kiểm tra(20%).

Ta định nghĩa param\_grid và khởi tạo mô hình SVM với kernel RBF để sử dụng với GridCV tìm ra tổ hợp các tham số tối ưu:

```
# Huấn luyện SVM với GridSearchCV (kernel RBF)
param_grid = {
    'C': [0.01, 0.1, 1, 10], # Mở rộng phạm vi C
    'gamma': ['scale', 0.001, 0.01, 0.1], # Mở rộng phạm vi gamma
    'kernel': ['rbf']
}
svm = SVC()
```

Trong phần định nghĩa lưới tham số(Param\_grid), có các tham số được đặt chi tiết:

- C: Tham số điều chỉnh mức độ phạt cho các điểm dữ liệu bị phân loại sai (regularization parameter). Ta chọn các giá trị C nhỏ đến C lớn. Với trường hợp C nhỏ - Ưu tiên lè phân cách lớn(margin lớn), chấp nhận một số điểm bị phân loại sai, dẫn đến mô hình đơn giản hơn nhưng có thể bị underfitting. Trường hợp C lớn ưu tiên phân loại chính xác các điểm dữ liệu, dẫn đến lè nhỏ hơn và mô hình phức tạp hơn, có thể gây overfitting.
- Gamma: Tham số kiểm soát độ rộng của kernel RBF(Radial Basis Function), ảnh hưởng đến mức độ ảnh hưởng của một điểm dữ liệu lên các điểm khác. Đối với gamma nhỏ, Kernel RBF có ảnh hưởng rộng, dẫn đến ranh giới quyết định mượt mà hơn, phù hợp với dữ liệu phân bố rộng. Đối với Gamma lớn, Kernel RBF có ảnh hưởng hẹp dẫn đến ranh giới quyết định phức tạp hơn, phù hợp với dữ liệu có cấu trúc nhưng dễ quá khớp. Scale là giá trị mặc định của scikit-learn, được tính tự động dựa trên dữ liệu:

$$\gamma = \frac{1}{n_{features} * var}$$

Sau đó ta khởi tạo mô hình SVM.

Ta tạo ra hàm get\_valid\_folds để tạo các fold hợp lệ cho kiểm định chéo, hàm đảm bảo rằng mỗi Fold(Tập huấn luyện và kiểm tra) chứa ít nhất 2 lớp khác nhau.

```
def get_valid_folds(X, y, n_splits=5, random_state=42, verbose=False):
```

```
    """
```

Trả về danh sách các fold hợp lệ từ StratifiedKFold,

mỗi fold đảm bảo có ít nhất 2 lớp trong cả tập train và test.

```
    """
```

```
def is_valid_fold(y_subset):
```

```
    return len(np.unique(y_subset)) > 1
```

```
skf = StratifiedKFold(n_splits=n_splits, shuffle=True,  
random_state=random_state)
```

```
valid_cv = []
```

```
for i, (train_idx, test_idx) in enumerate(skf.split(X, y)):
```

```
    y_train_fold = y[train_idx]
```

```
    y_test_fold = y[test_idx]
```

```
    if is_valid_fold(y_train_fold) and is_valid_fold(y_test_fold):
```

```
        valid_cv.append((train_idx, test_idx))
```

```
    if verbose:
```

```
        print(f" ✅ Fold {i + 1} hợp lệ.")
```

```
else:
```

```
    if verbose:
```

```
print(f" X Fold {i + 1} bị loại: thiếu lớp.")
```

```
if verbose:
```

```
    print(f"\nTổng số fold hợp lệ: {len(valid_cv)}/{n_splits}")
```

```
return valid_cv
```

```
# Sử dụng hàm trên để tạo fold hợp lệ
```

```
valid_cv = get_valid_folds(X_train, y_train, n_splits=5, verbose=True)
```

Trong hàm này ta có hàm `is_valid_fold` để kiểm tra tập nhãn con có chứa ít nhất 2 lớp khác nhau hay không, nếu một lớp thì sẽ gây lỗi vì thuật toán SVM yêu cầu ít nhất 2 lớp để phân loại.

Ta tiến hành tạo các fold kiểm định chéo sao cho tỷ lệ các lớp trong mỗi fold gần giống với tỷ lệ trong toàn bộ tập dữ liệu. sau đó ta kiểm tra xem fold có đủ lớp không để tạo fold hợp lệ.

Ta huấn luyện mô hình SVM bằng `GridSearchCV`, sử dụng các fold hợp lệ từ hàm `get_valid_folds` để kiểm định chéo và in ra số tối đa sau khi tìm kiếm:

```
# Huấn luyện GridSearchCV với fold hợp lệ
```

```
grid_search = GridSearchCV(svm, param_grid, cv=valid_cv, scoring='accuracy',  
n_jobs=-1)
```

```
grid_search.fit(X_train, y_train)
```

```
# In tham số tốt nhất
```

```
print("Tham số tốt nhất:", grid_search.best_params_)
```

Các tham số của `GridSearchCV`:

- svm: Mô hình SVM đã được khởi tạo trước đó (SVC()):
- param\_grid: Lưới tham số đã được định nghĩa trước.
- cv: Danh sách các fold hợp lệ đã kiểm tra ở trên
- scoring: Sử dụng độ chính xác (accuracy) Làm chỉ số để đánh giá và chọn mô hình tốt nhất.
- n\_jobs = -1: Sử dụng tất cả các lõi CPU có sẵn để song song hóa quá trình huấn luyện.

Sau đó ta thử nghiệm tất cả các tổ hợp tham số trong param\_grid, và chọn tổ hợp tham số có độ chính xác trung bình cao nhất. Cuối cùng là in ra tham số tốt nhất.

Ta đánh giá mô hình từ Cross-Validation:

```
# Đánh giá bằng cross-validation
best_model = grid_search.best_estimator_
cv_scores = cross_val_score(best_model, X_reduced, augmented_labels,
cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
scoring='accuracy', n_jobs=-1)
print(f"\nĐộ chính xác trung bình (Cross-validation): {cv_scores.mean() * 100:.2f}%")
print(f"Độ lệch chuẩn (Cross-validation): {cv_scores.std() * 100:.2f}%")

# Dự đoán trên tập kiểm tra
y_pred = best_model.predict(X_test)

# Tính độ chính xác trên tập kiểm tra
accuracy = accuracy_score(y_test, y_pred)
print(f"\nĐộ chính xác trên tập kiểm tra: {accuracy * 100:.2f}%")

# In báo cáo phân loại
print("\nBáo cáo phân loại (tập kiểm tra):")
```

```

target_names = ["Healthy", "Surface Erosion", "Cracked Blade", "Mass
Imbalance", "Twist Blade Fault"]
print(classification_report(y_test, y_pred, target_names=target_names))

```

Ta lấy mô hình SVM với tổ hợp tham số tối ưu nhất, thực hiện tính độ chính xác của best\_model bằng kiểm định chéo trên toàn bộ dữ liệu tăng cường. Sau đó tính độ chính xác trung bình.

- Độ chính xác trung bình cho biết hiệu suất tổng quát của mô hình trên toàn bộ dữ liệu.
- Độ lệch chuẩn thể hiện mức độ ổn định của mô hình qua các fold: Độ lệch chuẩn nhỏ thì mô hình ổn định ít biến động giữa các fold, độ lệch chuẩn lớn thì mô hình có thể không ổn định, phụ thuộc vào cách chia dữ liệu.

Sau đó ta tiến hành dự đoán nhãn trên tập kiểm tra, tính độ chính xác trên tập kiểm tra và in báo cáo phân loại.

Cuối cùng ta vẽ Learning curve để đánh giá hiệu suất của mô hình SVM khi số lượng mẫu huấn luyện tăng, vẽ ma trận nhầm lẫn:

```

# Vẽ Learning Curve
train_sizes, train_scores, test_scores = learning_curve(
    grid_search.best_estimator_, X_reduced, augmented_labels,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42), n_jobs=-1,
    train_sizes=np.linspace(0.1, 1.0, 10), scoring="accuracy"
)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

```

```

# Vẽ ma trận nhầm lẫn

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target_names,
            yticklabels=target_names)

plt.title('Ma trận nhầm lẫn cho SVM với kernel RBF')

plt.xlabel('Nhận dự đoán')

plt.ylabel('Nhận thực')

plt.show()

plt.figure(figsize=(10, 6))

plt.plot(train_sizes, train_mean, label="Training Accuracy", color="blue",
         marker="o")

plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
                 color="blue", alpha=0.1)

plt.plot(train_sizes, test_mean, label="Cross-validation Accuracy",
         color="green", marker="o")

plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std,
                 color="green", alpha=0.1)

plt.xlabel("Training Examples")

plt.ylabel("Accuracy")

plt.title("Learning Curve for SVM with RBF Kernel")

plt.legend(loc="best")

plt.grid()

plt.show()

```

Cuối cùng, ta lưu mô hình đã huấn luyện lại:

```

#Lưu mô hình

import joblib

```

```

joblib.dump(best_model, "wind_turbine_svm_rbf_model_optimized_v5.pkl")
joblib.dump(scaler, "wind_turbine_svm_scaler_optimized_v5.pkl")
joblib.dump(pca, "wind_turbine_pca_optimized_v5.pkl")
print("Đã lưu mô hình, scaler và PCA vào
'wind_turbine_svm_rbf_model_optimized_v5.pkl',
'wind_turbine_svm_scaler_optimized_v5.pkl' và
'wind_turbine_pca_optimized_v5.pkl'")

```

### 3.2.2. Kết quả thực nghiệm

Sau khi kiểm tra số lượng nhãn, ta được:

Phân bố nhãn ban đầu:	
2	7
1	7
0	7
4	7
3	7
Name: count, dtype: int64	

*Hình 3.2.1: Phân Bố Nhãn Ban Đầu*

Đây là số lượng nhãn ban đầu, tổng cộng là 35 nhãn – khớp với số lượng ban đầu.

Tiếp tục kiểm tra phân bố nhãn sau tăng cường:

Phân bố nhãn sau tăng cường:	
2	350
1	350
0	350
4	350
3	350

*Hình 3.2.2: Phân Bố Nhãn Sau Khi Tăng Cường*

Ta trích xuất đặc trưng được 12 đặc trưng, tuy nhiên chúng ta dùng PCA để giảm chiều, sau khi dùng PCA thì chỉ còn 9 đặc trưng:

Thêm vào đó, ta kiểm tra Các Fold xem có đủ 2 lớp trở lên không:

```
Số đặc trưng sau PCA: 9
✓ Fold 1 hợp lệ.
✓ Fold 2 hợp lệ.
✓ Fold 3 hợp lệ.
✓ Fold 4 hợp lệ.
✓ Fold 5 hợp lệ.
```

*Hình 3.2.3: Số Đặc Trưng Sau PCA*

Sau đó, ta tìm được các tham số tối ưu với mô hình SVM:

```
Tổng số fold hợp lệ: 5/5
Tham số tốt nhất: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

*Hình 3.2.4: Tham Số Tối Ưu*

Sau khi huấn luyện mô hình, ta có báo cáo phân loại và chỉ số về độ chính xác cũng như độ lệch chuẩn:

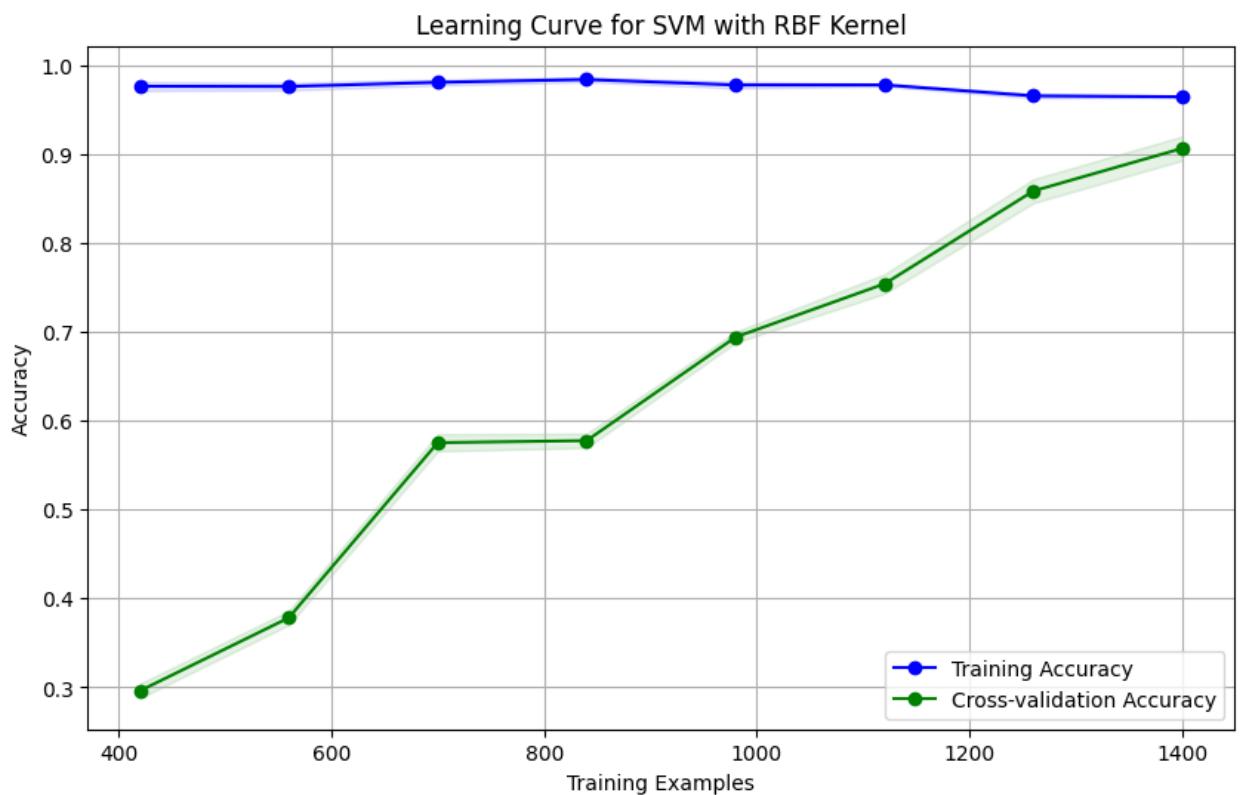
```
Độ chính xác trung bình (Cross-validation): 90.69%
Độ lệch chuẩn (Cross-validation): 1.36%

Độ chính xác trên tập kiểm tra: 88.86%
```

*Hình 3.2.5: Độ Chính Xác & Độ Lệch Chuẩn*

Từ kết quả này cho thấy, mô hình SVM dùng Kernel RBF hoạt động khá tốt và không bị overfitting, bởi vì độ lệch chuẩn rất ít và chỉ số độ chính xác trung bình cao và gần giống với độ chính xác trên tập kiểm tra.

Với biểu đồ Learning Curve cho mô hình SVM với Kernel RBF:



**Hình 3.2.6: Biểu Đồ Learning Curve Cho Mô Hình SVM Với Kernel RBF**

Training Accuracy khá cao, mô hình khớp rất tốt với tập huấn luyện, tuy nhiên đây là dấu hiệu của overfitting. Nên ta cần xét Cross – validation Accuracy (Đường màu xanh lá):

Ban đầu Cross-validation Accuracy rất thấp – mô hình tổng quát hóa kém với tập dữ liệu nhỏ, tuy nhiên khi số lượng mẫu tăng thì accuracy cũng tăng đều đặn, đến cuối cùng khoảng cách dưới 2 đường đã là rất nhỏ.

Điều đó thể hiện rằng mô hình không bị Overfitting.

Ta có báo cáo phân loại các tình trạng của cánh quạt gió:

Báo cáo phân loại (tập kiểm tra):				
	precision	recall	f1-score	support
Healthy	0.99	1.00	0.99	70
Surface Erosion	0.78	0.91	0.84	70
Cracked Blade	0.89	0.84	0.87	70
Mass Imbalance	0.95	0.83	0.89	70
Twist Blade Fault	0.86	0.86	0.86	70
accuracy			0.89	350
macro avg	0.89	0.89	0.89	350
weighted avg	0.89	0.89	0.89	350

Hình 3.2.7: Báo Cáo Phân Loại

Ý nghĩa của các chỉ số:

- **Precision (Độ chính xác):** Tỉ lệ dự đoán đúng trong số các mẫu được dự đoán là lớp đó.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Ví dụ trường hợp “Mass Imbalance là” 0.95, có nghĩa là 95% mẫu được dự đoán là “Mass Imbalance” thực sự đúng

- **Recall (Độ nhạy):** Tỉ lệ mẫu thực tế thuộc lớp đó được dự đoán đúng.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Ví dụ Recall của Healthy là 1.00, nghĩa là tất cả các mẫu “Healthy” thực tế đều được dự đoán đúng.

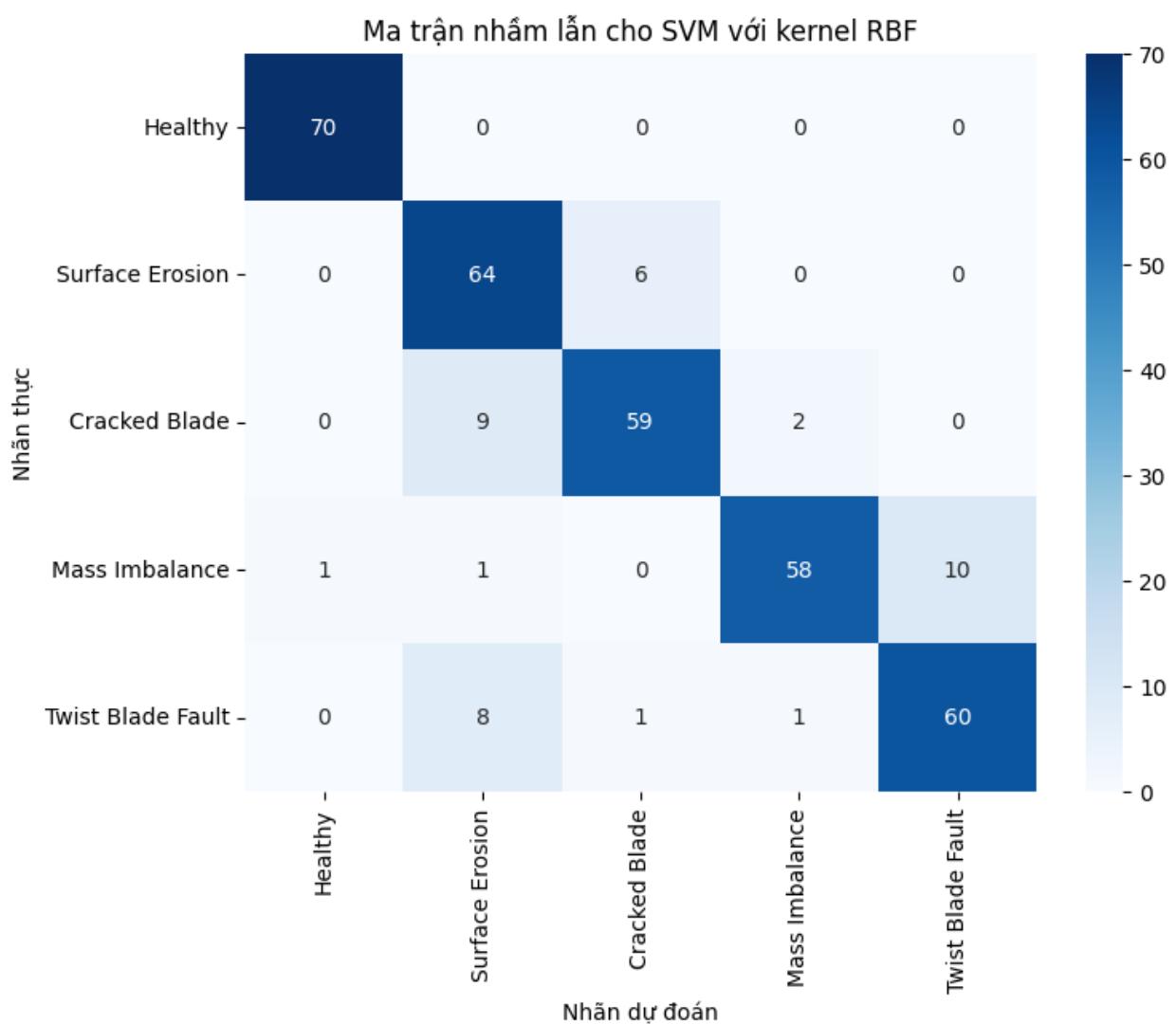
- F1-Score:

Trung bình điều hòa của Precision và recall, cân bằng giữa 2 chỉ số.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Support: Số lượng mẫu thực tế của mỗi lớp trong tập kiểm tra. Kết quả mỗi lớp có 70 mẫu, tổng cộng 350 mẫu, dữ liệu cân bằng hoàn hảo.

Sau đó, ta vẽ ma trận nhầm lẫn:



**Hình 3.2.8: Ma Trận Nhầm Lẩn Cho Mô Hình SVM với Kernel RBF**

Nhận xét về Ma trận nhầm lẩn:

- Nhận Healthy: Phân loại hoàn toàn chính xác (70/70), không nhầm sang bất kỳ lớp nào khác.
- Twist Blade Fault: Phân loại tốt với 60/70, nhầm 8 nhãn sang Surface Erosion, và 1 nhãn cho Cracked Blade và Mass Imbalance.
- Mass Imbalance: Tương đối tốt với 58/70, có một số nhầm lẩn với Twist Blade Fault (10 trường hợp).
- Surface Erosion: Phân loại tốt với 64/70 – Độ chính xác tốt nhưng 6 trường hợp bị nhầm sang Cracked Blade.
- Cracked Blade: Phân loại tương đối tốt với 59/70, đa số bị nhầm sang Surface Erosion.

## CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 4.1. Kết luận

Đồ án này đã thành công trong việc xây dựng một hệ thống phân loại tự động các trạng thái lỗi của cánh quạt tua-bin gió, đáp ứng mục tiêu đề ra là phát triển một giải pháp nhận diện chính xác các hư hỏng dựa trên tín hiệu rung động. Hệ thống tận dụng thuật toán Support Vector Machine (SVM) với kernel RBF, chứng minh được khả năng phân tích tín hiệu phức tạp và đưa ra kết quả đáng tin cậy trong môi trường phòng thí nghiệm. So với mục tiêu ban đầu, nghiên cứu này không chỉ đạt được độ chính xác cao mà còn thể hiện tính ổn định, với khả năng nhận diện trạng thái bình thường gần như hoàn hảo và các trạng thái lỗi với độ tin cậy đáng kể. Khi đặt trong bối cảnh các nghiên cứu tương tự, chẳng hạn nghiên cứu của Ogaili và cộng sự (2023) sử dụng Random Forest, đồ án này nổi bật nhờ khả năng xử lý hiệu quả dữ liệu phi tuyến, mang lại hiệu suất vượt trội trong việc phân tách các lớp lỗi. Tuy nhiên, so với các hệ thống giám sát tình trạng tiên tiến trong công nghiệp, vốn kết hợp nhiều nguồn dữ liệu như rung động, âm thanh, và hình ảnh, hệ thống trong nghiên cứu này vẫn còn giới hạn do chỉ dựa vào tín hiệu rung động đơn trực. Điều này khiến khả năng ứng dụng trong các kịch bản thực tế phức tạp chưa được đảm bảo, nhưng vẫn khẳng định giá trị của nghiên cứu trong việc cung cấp một giải pháp hiệu quả với nguồn lực hạn chế.

Trong quá trình thực hiện đồ án, ta đã hoàn thành một quy trình khép kín, từ việc xử lý dữ liệu thô đến triển khai mô hình học máy. Ta đã gộp và chuẩn hóa 35 tập dữ liệu từ các định dạng khác nhau, trích xuất các đặc trưng thống kê và tần số, đồng thời áp dụng các kỹ thuật tăng cường dữ liệu để nâng cao khả năng tổng quát hóa của mô hình. Việc tối ưu hóa thông qua GridSearchCV và sử dụng các công cụ đánh giá như ma trận nhầm lẫn, đường cong học tập đã giúp củng cố chất lượng của hệ thống. Tuy nhiên, ta gặp khó khăn trong việc xử lý dữ liệu không đồng nhất ban đầu, đòi hỏi nhiều lần thử nghiệm để đảm bảo tính nhất quán. Hơn nữa, mô hình vẫn chưa khắc phục được hoàn toàn nhầm lẫn giữa các trạng thái lỗi tương đồng, như mất cân bằng khối lượng và vặn cánh, do sự tương đồng trong đặc trưng tín hiệu. Việc thiếu dữ liệu thực địa cũng là một hạn chế, khiến ta chưa thể đánh giá hiệu quả của hệ thống trong điều kiện

vận hành thực tế. Những khó khăn này phản ánh sự phức tạp của bài toán phân tích tín hiệu và yêu cầu cao về chất lượng dữ liệu.

Đóng góp nổi bật của đồ án này nằm ở việc thiết lập một quy trình phân loại lỗi hoàn chỉnh, từ tiền xử lý dữ liệu đến đánh giá mô hình, có thể tái sử dụng trong các bài toán giám sát tình trạng khác. Hệ thống mang lại giá trị thực tiễn bằng cách cung cấp một công cụ hỗ trợ cho các kỹ sư bảo trì, giúp phát hiện sớm hư hỏng và tối ưu hóa hoạt động của tua-bin gió. Nghiên cứu cũng góp phần làm sáng tỏ tiềm năng của SVM trong xử lý tín hiệu rung động, mở ra cơ hội ứng dụng trong các lĩnh vực tương tự. Bài học kinh nghiệm rút ra bao gồm tầm quan trọng của việc kiểm tra kỹ lưỡng cấu trúc dữ liệu trước khi xử lý, bởi bất kỳ sai sót nào ở giai đoạn đầu đều có thể ảnh hưởng đến kết quả cuối cùng. Kỹ thuật tăng cường dữ liệu đã chứng minh giá trị trong việc cải thiện hiệu suất mô hình, đặc biệt khi dữ liệu gốc hạn chế. Việc sử dụng các công cụ trực quan như ma trận nhầm lẫn và đường cong học tập không chỉ giúp đánh giá mô hình mà còn cung cấp cái nhìn sâu sắc về cách tinh chỉnh hệ thống. Quá trình thực hiện đồ án đã rèn luyện khả năng phân tích và giải quyết vấn đề, đồng thời nhấn mạnh sự cần thiết của tư duy phản biện khi đối mặt với các thách thức kỹ thuật. Những kinh nghiệm này sẽ là nền tảng vững chắc cho các nghiên cứu hoặc dự án tương lai, đặc biệt trong lĩnh vực học máy và phân tích tín hiệu.

## 4.2. Hướng phát triển

Để hoàn thiện các chức năng của đồ án này, ta cần tập trung khắc phục những hạn chế đã xác định trong quá trình thực hiện. Trước tiên, cần cải thiện khả năng phân biệt giữa các trạng thái lỗi tương đồng, như mất cân bằng khối lượng và vặn cánh, vốn gây ra nhầm lẫn do đặc trưng tín hiệu tương tự. Ta có thể bổ sung các đặc trưng miền thời gian-tần số, chẳng hạn sử dụng biến đổi wavelet, để nắm bắt tốt hơn các khác biệt tinh vi giữa các lỗi. Việc này đòi hỏi thử nghiệm và đánh giá kỹ lưỡng để đảm bảo các đặc trưng mới không làm tăng độ phức tạp tính toán quá mức. Thứ hai, ta cần kiểm chứng mô hình trên dữ liệu thực địa từ các tua-bin gió đang hoạt động, thay vì chỉ dựa vào dữ liệu phòng thí nghiệm. Điều này yêu cầu hợp tác với các đơn vị vận hành tua-bin để

thu thập tín hiệu rung động trong điều kiện thực tế, nơi nhiễu và biến động môi trường có thể ảnh hưởng đến hiệu suất mô hình. Ngoài ra, ta cần tối ưu hóa quy trình xử lý dữ liệu để giảm thời gian chuẩn hóa và gộp dữ liệu, đặc biệt khi làm việc với các tập dữ liệu lớn hơn. Tự động hóa các bước tiền xử lý, chẳng hạn bằng cách phát triển một pipeline tích hợp, sẽ giúp tăng hiệu quả và giảm nguy cơ sai sót do thao tác thủ công.

Về các hướng đi mới, ta có thể nâng cấp hệ thống bằng cách tích hợp thêm các nguồn dữ liệu đa dạng, như tín hiệu âm thanh từ micro hoặc hình ảnh từ camera giám sát, để tăng cường khả năng nhận diện lỗi. Ví dụ, âm thanh phát ra từ các lỗi như nứt gãy có thể cung cấp thông tin bổ sung, giúp phân biệt rõ hơn với các lỗi khác. Việc này đòi hỏi phát triển các thuật toán tổng hợp đa phương thức (multimodal fusion) để kết hợp hiệu quả các loại dữ liệu. Một hướng khác là thử nghiệm các mô hình học máy tiên tiến hơn, như mạng nơ-ron sâu hoặc học tăng cường (reinforcement learning), để so sánh và cải thiện hiệu suất so với SVM. Những mô hình này có thể phù hợp hơn khi làm việc với dữ liệu thực địa phức tạp, nhưng cần cân nhắc về yêu cầu tính toán và dữ liệu huấn luyện. Ta cũng có thể phát triển một giao diện người dùng trực quan, cho phép các kỹ sư bảo trì nhập dữ liệu và nhận kết quả phân loại theo thời gian thực, từ đó tăng tính thực tiễn của hệ thống. Hơn nữa, việc triển khai mô hình trên các thiết bị nhúng (embedded devices) sẽ hỗ trợ giám sát liên tục tại hiện trường, giảm sự phụ thuộc vào các hệ thống máy tính mạnh. Cuối cùng, ta có thể mở rộng nghiên cứu bằng cách áp dụng hệ thống cho các loại máy móc khác, như động cơ hoặc máy nén, để kiểm chứng tính linh hoạt của quy trình. Những định hướng này không chỉ giúp khắc phục các hạn chế hiện tại mà còn đưa nghiên cứu tiến gần hơn đến các ứng dụng công nghiệp thực tế, đồng thời mở ra cơ hội khám phá các phương pháp phân tích tín hiệu và học máy tiên tiến hơn.

## **TÀI LIỆU THAM KHẢO**

- [1] C. C. Staff, "Lecture Note 9: Support Vector Machines," Cornell University, Department of Computer Science, 2018. [Online]. Available:  
<https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html>.
- [2] T. Nguyen, "Support Vector Machine với Kernel Trick," Machine Learning Cơ Bản, 2017. [Online]. Available: <https://machinelearningcoban.com/2017/04/22/kernelsmv/>.