TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

ଌ📖ଌ

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG

ĐỀ TÀI: PHÂN LOẠI BỆNH TRÊN CÂY DƯA LEO SỬ DỤNG MÔ HÌNH SVM

GVHD:	TS. Nguyễn Mạnh Cường
Nhóm:	14
Mã lớp độc lập:	20231IT6075002
Người thực hiện:	Nguyễn Minh Đức
Mã sinh viên:	2020601169

Hà Nội – 2023

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

ଌ📖ଌ

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG

ĐỀ TÀI: PHÂN LOẠI BỆNH TRÊN CÂY DƯA LEO SỬ DỤNG MÔ HÌNH SVM

GVHD:	TS. Nguyễn Mạnh Cường
Nhóm:	14
Mã lớp độc lập:	20231IT6075002
Người thực hiện:	Nguyễn Minh Đức
Mã sinh viên:	2020601169

Hà Nội – 2023

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

SVM	Support Vector Machine
MSVM	Multiclass Support Vector Machines
SVMs	Support Vector Machines for Multiclass Classification
CNN	Convolutional neural network
KNN	Graphical User Interface
NBC	Navie Bayes Classification
LBP	Local Binary Patterns
HOG	Histogram of Oriented Gradients
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
RBF	Gaussian kernel
JPEG	Joint Photographic Experts Group

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

LỜI CẢM ƠN

Trước tiên, với tình cảm chân thành và sâu sắc nhất, cho phép em được bày tỏ lòng biết ơn sâu sắc đến các thầy cô của trường Đại học Công Nghiệp Hà Nội, đặc biệt là các thầy cô khoa Công Nghệ Thông Tin của trường đã tạo điều kiện cho em có một kỳ học tập vô cùng bổ ích và đáng nhớ.

Em xin chân thành cảm ơn thầy giáo Tiến Sĩ Nguyễn Mạnh Cường – người đã tận tâm hướng dẫn chúng em qua từng buổi nói chuyện, hướng dẫn, thảo luận về các lĩnh vực trong đề tài. Thầy đã truyền đạt cho chúng em những kiến thức quý báu, những kinh nghiệm quý giá, giúp chúng em có thể hoàn thành tốt báo cáo của mình.

Em cũng xin cảm ơn các thầy cô trong khoa đã luôn quan tâm, hỗ trợ chúng em trong suốt quá trình học tập. Những lời động viên, chia sẻ của thầy cô đã giúp chúng em vượt qua những khó khăn và hoàn thành tốt nhiệm vụ được giao.

Trong quá trình học tập, cũng như là trong quá trình làm bài báo cáo, khó tránh khỏi sai sót. Em rất mong nhận được ý kiến đóng góp từ thầy cô để học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo sắp tới.

Em xin chúc thầy cô luôn dồi dào sức khỏe, luôn vui vẻ và thành công trong cuộc sống.

Thay lời kết, em xin gửi lời chào trân trọng và lời chúc tốt đẹp nhất đến tất cả các thầy cô.

Sinh viên thực hiện

Nguyễn Minh Đức

LỜI NÓI ĐẦU

Nông nghiệp là một ngành sản xuất quan trọng, đóng góp trực tiếp vào sự phát triển kinh tế - xã hội của đất nước. Tuy nhiên, sản xuất nông nghiệp vẫn đang đối mặt với nhiều khó khăn, thách thức, trong đó có vấn đề dịch bệnh cây trồng.

Các bệnh cây trồng có thể gây ra những thiệt hại nghiêm trọng về năng suất, chất lượng nông sản, thậm chí là dẫn đến chết cây. Việc phát hiện và xử lý bệnh cây trồng kịp thời là vô cùng cần thiết để đảm bảo năng suất và chất lượng nông sản.

Hiện nay, việc phát hiện và xử lý bệnh cây trồng chủ yếu dựa trên kinh nghiệm của người nông dân hoặc các chuyên gia nông nghiệp. Tuy nhiên, cách làm này còn nhiều hạn chế, như:

- Tốn nhiều thời gian và công sức.
- Không chính xác, đặc biệt là đối với các loại bệnh khó nhận biết.
- Không có khả năng dự báo bệnh.

Để khắc phục những hạn chế trên, việc ứng dụng trí tuệ nhân tạo (AI) trong phát hiện và xử lý bệnh cây trồng là một hướng đi đầy tiềm năng. AI có thể giúp tự động hóa quá trình phát hiện và xử lý bệnh cây trồng, từ đó nâng cao hiệu quả và chính xác.

Trên cơ sở đó, em đã lựa chọn đề tài "Phân loại bệnh trên cây dưa leo" nhằm góp phần giải quyết bài toán phát hiện và xử lý bệnh cây trồng một cách hiệu quả và chính xác.

Dưa leo là một loại rau ăn quả được trồng phổ biến ở Việt Nam. Đây là loại cây có giá trị kinh tế cao, tuy nhiên cũng là loại cây dễ bị mắc bệnh. Một số bệnh phổ biến trên cây dưa leo bao gồm bệnh phấn trắng, bệnh sương mai, bệnh héo

xanh,... Những bệnh này gây ảnh hưởng nghiêm trọng đến năng suất và chất lượng của cây trồng.

Việc phát hiện và phân loại bệnh trên cây dưa leo là vô cùng cần thiết để có biện pháp phòng trừ kịp thời và hiệu quả. Hiện nay, có nhiều phương pháp để phân loại bệnh trên cây trồng, trong đó phương pháp phân loại bằng máy tính là một phương pháp có nhiều ưu điểm như độ chính xác cao, tốc độ nhanh,...

Trong đồ án này, em đã sử dụng mô hình SVM để phân loại bệnh trên cây dưa leo. Mô hình SVM là một mô hình phân loại mạnh, có khả năng phân biệt tốt các lớp đối tượng.

Mục tiêu của đồ án là xây dựng mô hình SVM phân loại bệnh trên cây dưa leo với độ chính xác cao. Để đạt được mục tiêu này, em đã thực hiện các bước sau:

Bài báo cáo cả em gồm có 4 phần chính

Chương 1: Tổng quan về bài toán phân loại bệnh trên cây dưa leo

Chương 1 em thực hiện nêu khái quát bài toán phân loại trong trí tuệ nhận tạo và học máy, khái quát chung về SVM và áp dụng SVM trong bài toán nhận diện bệnh trên cây dưa leo. Ngoài ra

Chương 2: Các kỹ thuật giải quyết bài toán

Ở chương này em giới thiệu về mô hình SVM, các đặc trưng hình ảnh và các kỹ thuật xử lý dữ liệu hình ảnh. Trình bày phương pháp thu thập, xử lý dữ liệu hình ảnh của cây dưa leo và xây dựng mô hình SVM phân loại bệnh trên cây dưa leo.

Chương 3: Đánh giá hiệu quả của mô hình.

Tại chương 3 trình bày kết quả đánh giá hiệu quả của mô hình.

Kết quả nghiên cứu cho thấy, mô hình SVM phân loại bệnh trên cây dưa leo có độ chính xác cao. Đây là một kết quả khả quan, cho thấy mô hình có thể được ứng dụng trong thực tế để phát hiện và phân loại bệnh trên cây dưa leo.

Phần kết luận

Cuối cùng trong phần kết luật, em tổng hợp các kết quả đạt được, các hướng phát triển và mở rộng đề tài nghiên cứu trong tương lai.

Em xin chân thành cảm ơn các thầy đã tận tình hướng dẫn và giúp đỡ nhóm hoàn thành đồ án này.

NỘI DUNG

CHƯƠNG 1 TỔNG QUAN VỀ BÀI TOÁN BÀI TOÁN PHÂN LOẠI BỆNH TRÊN CÂY DƯA LEO

1.1. Tổng quan về khai phá dữ liệu

Thuật ngữ Khai phá dữ liệu (Data mining) ra đời vào cuối những năm 80 thế kỷ trước. Có nhiều định nghĩa khác nhau về khai phá dữ liệu, nhưng diễn đạt một cách dễ hiểu thì khai phá dữ liệu là quá trình tìm kiếm những thông tin(tri thức), mẫu và xu hướng có ích, tiềm từ dữ liệu thô. Mục đích của nó là làm cho dữ liệu trở nên quan trọng hơn và có thể sử dụng được.

Ngày nay phân lớp dữ liệu (classification) là một trong những hướng nghiên cứu chính của khai phá dữ liệu. Thực tế đặt ra nhu cầu là từ một cơ sở dữ liệu với nhiều thông tin ẩn con người có thể trích rút ra các quyết định nghiệp vụ thông minh. Phân lớp và dự đoán là hai dạng của phân tích dữ liệu nhằm trích rút ra một mô hình mô tả các lớp dữ liệu quan trọng hay dự đoán xu hướng dữ liệu tương lai. Phân lớp dự đoán giá trị của những nhãn xác định (categorical label) hay những giá trị rời rạc (discrete value), có nghĩa là phân lớp thao tác với những đối tượng dữ liệu mà có bộ giá trị là biết trước.

Trong khi đó, dự đoán lại xây dựng mô hình với các hàm nhận giá trị liên tục. Ví dụ mô hình phân lớp dự báo thời tiết có thể cho biết thời tiết ngày mai là mưa, hay nắngdựa vào những thông số về độ ẩm, sức gió, nhiệt độ,… của ngày hôm nay và các ngày trước đó. Hay nhờ các luật về xu hướng mua hàng của khách hàng trong siêu thị, các nhân viên kinh doanh có thể ra những quyết sách đúng đắn về lượng mặt hàng cũng như chủng loại bày bán… Một mô hình dự đoán có thể dự đoán được lượng tiền tiêu dùng của các khách hàng tiềm năng dựa trên những thông tin về thu nhập và nghề nghiệp của khách hàng. Trong những năm qua, phân lớp dữ liệu đã thu hút sự quan tâm các nhà nghiên cứu trong nhiều lĩnh vực khác

nhau như học máy (machine learning), hệ chuyên gia (expert system), thống kê (statistics)

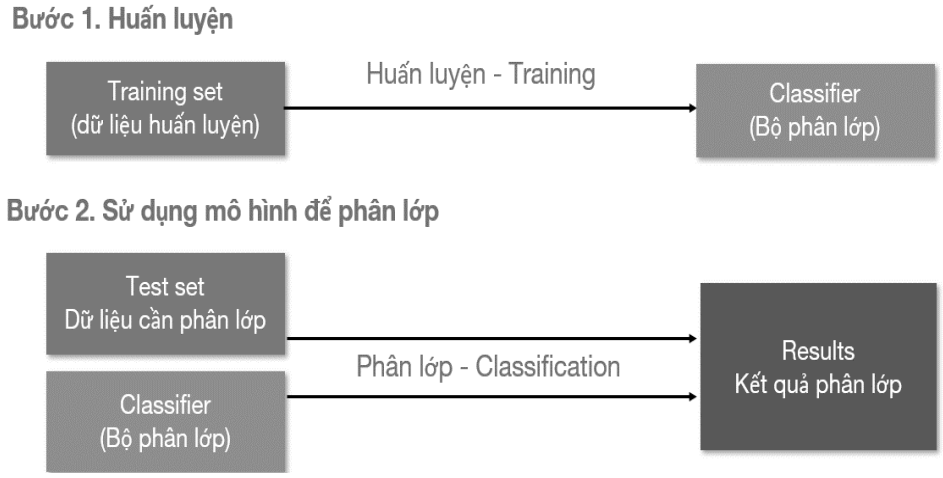
Hiện nay kỹ thuật khai phá dữ liệu đang được áp dụng một cách rộng rãi trong nhiều lĩnh vực kinh doanh và đời sống khác nhau, như: Thương mại (phân tích dữ liệu bán hàng và thị trường, phân tích đầu tư, quyết định cho vay, phát hiện gian lận,...); Thông tin sản xuất (điều khiển và lập kế hoạch, hệ thống quản lý, phân tích kết quả thử nghiệm…); Thông tin khoa học (dự báo thời tiết,...); CSDL sinh học (ngân hàng gen,…); Khoa học địa lý (dự báo động đất,…); Trong y tế, marketing, ngân hàng, viễn thông, du lịch, internet…

1.2. Tổng quan về bài toán phân loại

Bài toán phân loại là một trong những bài toán cơ bản trong trí tuệ nhân tạo và học máy. Nó được định nghĩa là việc phân loại các đối tượng vào các lớp khác nhau dựa trên các đặc trưng của chúng. Các thuật toán phân loại được sử dụng rộng rãi trong các ứng dụng thực tế như phân loại email, phân loại ảnh, phân loại văn bản, và nhiều ứng dụng khác. Học máy là một phương pháp thuộc lĩnh vực trí tuệ nhân tạo, trong đó các máy tính được lập trình để tự học từ dữ liệu mà không cần phải được lập trình cụ thể. Học máy cho phép máy tính nhận biết mẫu và rút ra các thông tin, kết quả mà không cần sự can thiệp trực tiếp từ con người. Học máy được sử dụng rộng rãi trong việc phân loại, gom nhóm, dự đoán, và khám phá thông tin từ dữ liệu. Ví dụ, học máy được áp dụng trong bài toán lọc thư rác, gợi ý sản phẩm, phân tích tình cảm người dùng, và nhiều ứng dụng khác.

Phân loại là quá trình xác định nhãn lớp của một điểm dữ liệu dựa trên các đặc trưng đã xác định trước đó. Trong trí tuệ nhân tạo và học máy, phân loại là một trong những tác vụ quan trọng, mô hình học máy được huấn luyện để thực hiện dự đoán hoặc gán nhãn dựa trên các dữ liệu đã được biết trước đó.

Các bước thực hiện bài toán phân loại:



Hình 1: Các bước tổng quát thực hiện bài toán phân loại

1.2.1. Một số thuật toán phân loại phổ biến

Cây quyết định (Random Forest)

Cây quyết định phân loại dữ liệu bằng cách xây dựng một cây có cấu trúc, mỗi nút trên cây đại diện cho một quyết định dựa trên một đặc trưng của dữ liệu. Quá trình xây dựng cây được thực hiện bằng cách chọn đặc trưng tốt nhất để phân chia dữ liệu ở mỗi nút sao cho sự không chắc chắn (entropy) hoặc gini impurity giảm đáng kể.

SVM (Super Vector Machine)

SVM tìm siêu phẳng phân chia các điểm dữ liệu của hai lớp sao cho khoảng cách giữa siêu phẳng và các điểm gần nhất của hai lớp là lớn nhất. Điều này thường được thực hiện bằng cách tối ưu hóa margin.

KNN (K-nearest Neightbors)

KNN dựa trên giả sử rằng các điểm gần nhau trong không gian đặc trưng có xu hướng thuộc cùng một lớp. Khi có một điểm cần phân loại, nó sẽ xem xét các điểm gần nhất và gán nhãn dựa trên đa số k điểm gần nhất.

Naïve Bayes

Naïve Bayes dựa trên Định lý Bayes và giả định rằng các đặc trưng là độc lập giữa nhau khi đã biết lớp của dữ liệu. Nó ước lượng xác suất của lớp dựa trên xác suất của các đặc trưng.

1.2.2. Ứng dụng chính của bài toán phân loại

Y tế và y học: Hệ thống phân loại được ứng dụng để chẩn đoán bệnh từ hình ảnh y tế và dự đoán, theo dõi các bệnh dịch, phân loại bệnh nhân

Tài chính và ngân hàng: Phân loại giao dịch tài chính, kiểm soát rủi ro trong giao dịch, dự đoán xu hướng tài chính…

Bán lẻ và thương mại điện tử: Phân loại sản phẩm, dự đoán xu hướng mua sắm, trending, ..

Tìm kiếm và tổ chức thông tin: Phân loại tài liệu, văn bản, cải thiện hiệu suất tìm kiếm, lọc các thông tin không an toàn..

Giáo dục: Phân loại học sinh, tự động đánh giá bài kiểm tra,..

Nông nghiệp: Nhận dạng và phân loại cây trồng, bệnh cây trồng, phân loại đất đai,…

1.3. Bài toán phân loại bệnh trên cây dưa leo

1.3.1. Mô tả chung

Bài toán phân loại bệnh trên cây dưa leo là một bài toán phân loại đa lớp, trong đó mỗi lớp tương ứng với một loại bệnh trên cây dưa leo. Bài toán này có thể được giải quyết bằng các phương pháp học máy như học sâu, học máy SVM, học máy Naive Bayes, học máy Random Forest, học máy KNN, học máy Decision Tree, và nhiều phương pháp khác.

Các bệnh phổ biến như vàng lá, phấn trắng, đốm nâu, sương mai, thối rễ có thể làm suy giảm chất lượng quả, thậm chí làm chết cả cây. Do đó, bài toán phân loại các bệnh trên cây dưa leo dựa trên hình ảnh lá là hết sức cấp thiết. Mô hình phân loại hiệu quả có thể được tích hợp vào các hệ thống cảnh báo sớm hoặc ứng dụng chẩn đoán hỗ trợ người dùng.

1.3.2. Đầu vào của bài toán (Input)

Đầu vào: Tập hình ảnh lá cây dưa leo được chụp từ các góc độ và điều kiện ánh sáng khác nhau. Có thể là toàn bộ cây hoặc tập trung chủ yếu vào lá. Ảnh có thể bao gồm các triệu chứng bệnh hoặc lá bình thường.

Nhãn (loại bệnh)	Số lượng hình ảnh
Bệnh thán thư (Anthracnose)	800
Bệnh héo vi khuẩn (Bacterial Wilt)	800
Bệnh bụng thối (Belly Rot)	800
Bệnh lá tươi (Fresh Leaf)	800
Bệnh thối quả (Pythium Fruit Rot)	800
Bệnh bạc lá (Gummy Stem Blight)	800
Nấm mốc (Downy Mildew)	800
Bình thườn (normal)	800

Bảng 1: Mô tả dữ liệu đầu vào

1.3.3. Đầu ra của bài toán

Dự đoán nhãn (label): Output sẽ là một dự đoán nhãn cho ảnh, xác địnhliệu nó thuộc nhóm bệnh nào hoặc không có bệnh

Độ chính xác (accuracy): Output có thể cung cấp thông tin về độ chính xác của mô hình phân loại. Đây là tỷ lệ phần trăm của các ảnh bệnh mà mô hình dự

đoán đúng. Ví dụ, output có thể cho biết rằng mô hình đạt được độ chính xác là 85%, tức là mô hình dự đoán chính xác cho 85% các mẫu tế bào trong bộ dữ liệu.

Các chỉ số đánh giá (evaluation metrics): Output có thể cung cấp các chỉ số đánh giá như độ chính xác (accuracy), độ nhạy (sensitivity), độ đặc hiệu (specificity), độ đo F1 (F1 score) và các chỉ số khác. Các chỉ số này cung cấp thông tin chi tiết về hiệu suất của mô hình trong việc phân loại các mẫu

1.3.4. Một số khó khăn thách thức của bài toán

Mất cân bằng lớp - Các lớp bệnh khác nhau có thể có số lượng mẫu mắc bệnh không cân bằng nhau. Điều này có thể làm sai lệch mô hình về phía các lớp đa số.

Dữ liệu không đầy đủ - Các triệu chứng của bệnh có thể không được ghi lại đầy đủ trong tập dữ liệu. Thiếu dữ liệu sẽ ảnh hưởng tới khả năng học và dự đoán của mô hình.

Sai sót trong gán nhãn - Các bệnh có triệu chứng tương tự nhau có thể bị nhầm lẫn khi gán nhãn. Điều này làm nhiễu dữ liệu và ảnh hưởng tới độ chính xác của mô hình.

Quá khớp - Do số lượng mẫu hạn chế nên mô hình dễ bị overfitting, khả năng khái quát hóa kém. Cần sử dụng kỹ thuật regularization để khắc phục.

Đa dạng sinh học - Cây dưa leo của cùng một giống nhưng ở các địa điểm khác nhau, môi trường khác nhau (điều kiện ánh sáng, thiết bị chụp ảnh..) có thể cho phản ứng khác nhau với cùng một loại bệnh. Điều này gây khó khăn cho việc khái quát hóa

CHƯƠNG 2 CÁC KĨ THUẬT GIẢI QUYẾT BÀI TOÁN

2.1. Phương hướng tiếp cận bài toán

Đối với việc tiếp cận và giải quyết bài toán phân loại bệnh trên cây dưa leo, trước tiên ta cần tiền xử lý và quan sát bộ dữ liệu đầu vào. Phương hướng giải quyết bài toán là:

- Thu thập dữ liệu
- Tiền xử lí dữ liệu (nếu cần thiết)
- Thực hiện trích chọn đặc trưng của ảnh
- Sử dụng mô hình học máy để thực hiện train dữ liệu
- Thực nghiệm với dữ liệu test

2.2. Trích chọn đặc trưng

2.2.1. Trích chọn đặc trưng HOG

Nguyên lý hoạt động:

HOG là một phương pháp trích chọn đặc trưng phổ biến trong thị giác máy tính và nhận diện hình ảnh. Nó tập trung vào việc mô tả cấu trúc và hình dạng của đối tượng trong ảnh bằng cách tính toán và xây dựng histogram của các hướng gradient trong ảnh.

Quá trình này bao gồm các bước như chuyển đổi ảnh thành ảnh xám, tính toán đạo hàm theo hướng x và y, chia ảnh thành các ô (cells) nhỏ, tính toán hướng gradient và cuối cùng là xây dựng histogram các gradient.

Ưu điểm và Ứng dụng:

HOG phù hợp để trích chọn đặc trưng trong nhận diện vật thể, phân loại hình ảnh và nhận dạng người.

Đặc điểm của HOG là có thể biểu diễn thông tin hình dạng của đối tượng mà không quan tâm đến màu sắc, độ sáng.

2.2.2. Trích chọn đặc trưng LBP

Nguyên lý hoạt động:

LBP cũng là một phương pháp phổ biến để trích chọn đặc trưng từ hình ảnh. Nó tập trung vào việc mô tả đặc điểm cục bộ của texture trong ảnh bằng cách so sánh độ sáng của các điểm ảnh xung quanh một điểm ảnh trung tâm.Các giá trị nhị phân được tạo thành từ việc so sánh các điểm xung quanh với điểm trung tâm để tạo ra một mã LBP cho từng điểm ảnh.

Ưu điểm và Ứng dụng:

LBP thường được sử dụng trong nhận diện texture, gặp ứng dụng trong phân loại hình ảnh, nhận diện khuôn mặt, và phân loại vật liệu.

2.2.3. Trích chọn đặc trưng sử dụng mô hình mạng

Nguyên lý hoạt động:

CNN là một mô hình học sâu được thiết kế để tự động học các đặc trưng từ dữ liệu hình ảnh thông qua mạng các lớp convolution, pooling và fully connected.

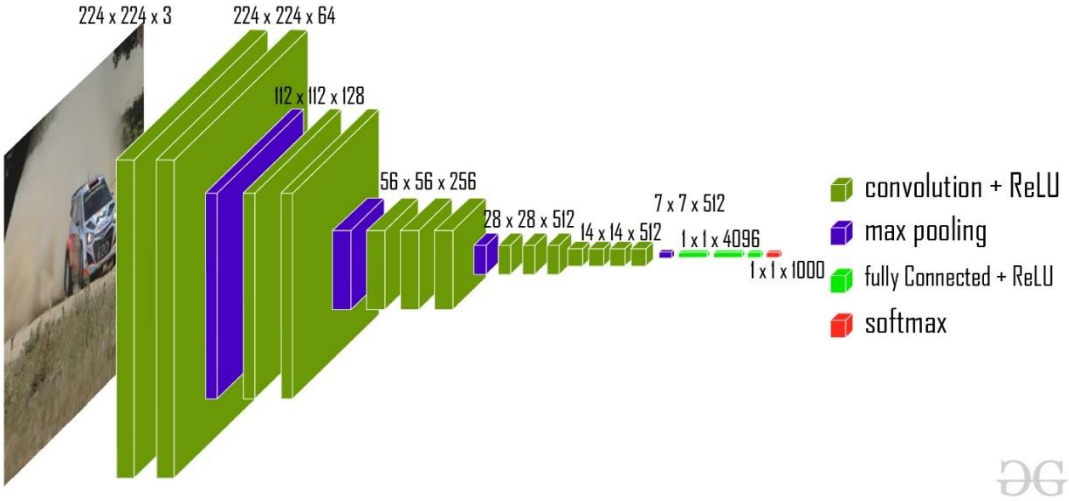
CNN có khả năng học các đặc trưng cấp cao từ dữ liệu hình ảnh mà không cần phải xác định trước các đặc trưng cụ thể.

Các lớp convolution trong CNN tự động học các bộ lọc để trích xuất các đặc trưng từ dữ liệu ảnh.

Ưu điểm và Ứng dụng:

CNN là mô hình mạnh mẽ trong thị giác máy tính, đặc biệt là trong việc nhận diện vật thể, phân loại hình ảnh, nhận diện khuôn mặt, và nhiều ứng dụng khác trong xử lý hình ảnh.

2.2.4. VGG-16 | Mô hình CNN



Hình 2: Kiến trúc VGG-16

Mô hình này đạt được độ chính xác kiểm tra top 5 92,7% trên tập dữ liệu ImageNet, chứa 14 triệu hình ảnh thuộc 1000 lớp.

Mục tiêu: Tập dữ liệu ImageNet chứa hình ảnh có kích thước cố định 224*224 và có các kênh RGB. Vì vậy, chúng ta có một tenxơ (224, 224, 3) làm đầu vào của chúng ta. Mô hình này xử lý hình ảnh đầu vào và xuất ra một vectơ 1000 giá trị.

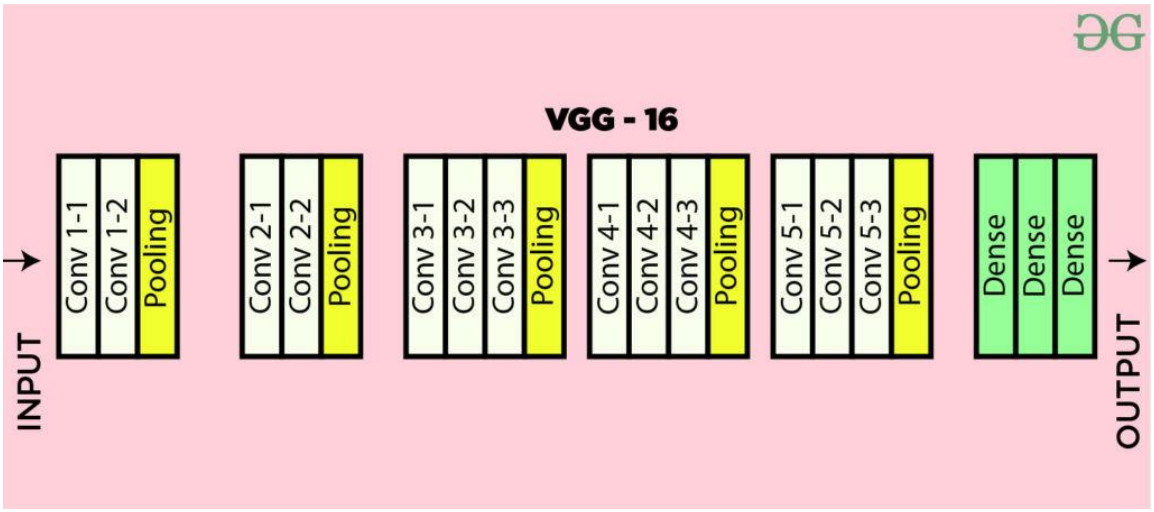
$$\hat{y} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_{9999} \end{bmatrix}$$

Vectơ này đại diện cho xác suất phân loại cho lớp tương ứng. Giả sử chúng ta có một mô hình dự đoán rằng hình ảnh thuộc về lớp 0 với xác suất 1, lớp 1 với xác suất 0,05, lớp 2 với xác suất 0,05, lớp 3 với xác suất 0,03, lớp 780 với xác suất 0,72, lớp 999 với xác suất 0,05 và tất cả các lớp khác với 0. Vì vậy, vectơ phân loại cho điều này sẽ là:

$$\hat{y} = \begin{bmatrix} \hat{y}_0 = 0.1 \\ 0.05 \\ 0.05 \\ 0.03 \\ . \\ . \\ . \\ \hat{y}_{780} = 0.72 \\ . \\ . \\ \hat{y}_{9999} = 0.05 \end{bmatrix}$$

Để đảm bảo các xác suất này thêm vào 1, chúng tôi sử dụng hàm softmax.

Kiến trúc VGG: Đầu vào mạng là hình ảnh có kích thước (224, 224, 3). Hai lớp đầu tiên có 64 kênh có kích thước bộ lọc 3 * 3 và cùng một lớp đệm. Sau đó, sau một lớp sải chân hồ bơi tối đa (2, 2), hai lớp có các lớp tích chập có kích thước bộ lọc 128 và kích thước bộ lọc (3, 3). Tiếp theo là một lớp sải chân gộp tối đa (2, 2) giống như lớp trước. Sau đó, có 2 lớp tích chập có kích thước bộ lọc (3, 3) và 256 bộ lọc. Sau đó, có 2 bộ gồm 3 lớp tích chập và một lớp hồ bơi tối đa. Mỗi bộ lọc có 512 bộ lọc có kích thước (3, 3) với cùng một lớp đệm. Hình ảnh này sau đó được chuyển đến ngăn xếp của hai lớp tích chập. Trong các lớp tích chập và tổng hợp tối đa này, các bộ lọc chúng tôi sử dụng có kích thước 3 * 3 thay vì 11 * 11 trong AlexNet và 7 * 7 trong ZF-Net. Trong một số lớp, nó cũng sử dụng 1 * 1 pixel được sử dụng để thao tác số lượng kênh đầu vào. Có một lớp đệm 1 pixel (cùng một lớp đệm) được thực hiện sau mỗi lớp tích chập để ngăn chặn đặc điểm không gian của hình ảnh.



Hình 3: Bản đồ kiến trúc VGG-16

Sau khi xếp chồng lớp tích chập và lớp tổng hợp tối đa, chúng tôi có một bản đồ tính năng (7, 7, 512). Chúng tôi làm phẳng đầu ra này để biến nó thành vectơ tính năng (1, 25088). Sau này có 3 lớp được kết nối đầy đủ, lớp đầu tiên lấy đầu vào từ vectơ tính năng cuối cùng và xuất ra vectơ (1, 4096), lớp thứ hai cũng xuất ra một vectơ có kích thước (1, 4096) nhưng lớp thứ ba xuất ra 1000 kênh cho 1000 lớp thử thách ILSVRC, tức là lớp được kết nối đầy đủ thứ 3 được sử dụng để thực hiện chức năng softmax để phân loại 1000 lớp. Tất cả các lớp ẩn sử dụng ReLU làm chức năng kích hoạt của nó. ReLU hiệu quả hơn về mặt tính toán vì nó dẫn đến việc học nhanh hơn và nó cũng làm giảm khả năng biến mất các vấn đề về độ dốc.

Cấu hình: Bảng dưới đây liệt kê các kiến trúc VGG khác nhau. Chúng ta có thể thấy rằng có 2 phiên bản VGG-16 (C và D). Không có nhiều sự khác biệt giữa chúng ngoại trừ một số lớp tích chập, (3, 3) tích chập kích thước bộ lọc được sử dụng thay vì (1, 1). Hai thông số này chứa lần lượt 134 triệu và 138 triệu tham số.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Hình 4: Cấu hình VGG khác nhau

Bản địa hóa đối tượng trong hình ảnh: Để thực hiện bản địa hóa, chúng ta cần thay thế điểm lớp bằng cách giới hạn tọa độ vị trí hộp. Vị trí hộp giới hạn được biểu thị bằng vectơ 4-D (tọa độ trung tâm (x, y), chiều cao, chiều rộng). Có hai phiên bản của kiến trúc bản địa hóa, một là bounding box được chia sẻ giữa các ứng cử viên khác nhau (đầu ra là 4 parameter vector) và phiên bản còn lại là bounding box dành riêng cho lớp (đầu ra là vectơ tham số 4000). Bài báo đã thử nghiệm cả hai cách tiếp cận trên kiến trúc VGG -16 (D). Ở đây chúng ta cũng cần thay đổi tổn thất từ tổn thất phân loại sang các hàm tổn thất hồi quy (chẳng hạn như MSE) để phạt độ lệch của tổn thất dự đoán so với sự thật cơ bản.

Kết quả: VGG-16 là một trong những kiến trúc hoạt động tốt nhất trong thử thách ILSVRC 2014.It là á quân trong nhiệm vụ phân loại với sai số phân loại top 5 là

7,32% (chỉ sau GoogLeNet với sai số phân loại là 6,66%). Nó cũng là người chiến thắng trong nhiệm vụ bản địa hóa với 25,32% lỗi bản địa hóa.

Hạn chế của VGG 16:

- Nó rất chậm để đào tạo (mô hình VGG ban đầu đã được đào tạo trên GPU Nvidia Titan trong 2-3 tuần).
- Kích thước của VGG-16 training: imageNet weights là 528 MB. Vì vậy, nó chiếm khá nhiều dung lượng đĩa và băng thông khiến nó không hiệu quả.
- 138 triệu tham số dẫn đến vấn đề gradient nổ.

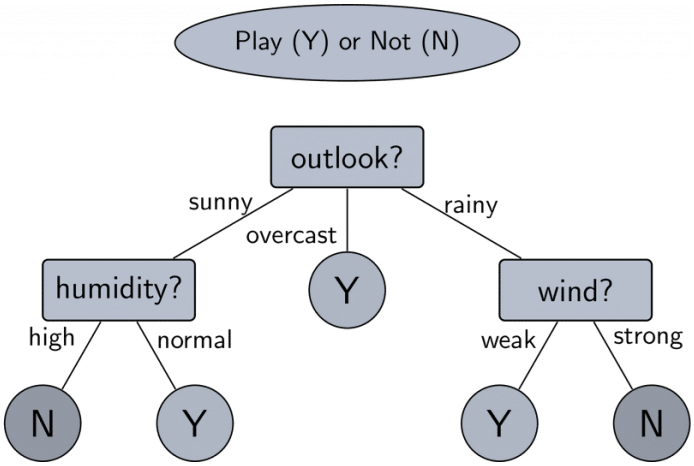
2.3. Kĩ thuật phân loại

2.3.1. Cây quyết định

**** Khái niệm***

Cây quyết định (Decision tree) là một đồ thị của các quyết định và các kết quả của nó. Nó có dạng một sơ đồ dòng chảy, bắt đầu từ một nút gốc và phân nhánh theo các điều kiện khác nhau, cho đến khi đạt được các kết quả cuối cùng. Nó giúp chúng ta nhìn thấy rõ các bước logic và xác suất trong quá trình ra quyết định.

Hay nói cách khác cây quyết định là cây mà mỗi nút biểu diễn một đặc trưng (tính chất), mỗi nhánh (branch) biểu diễn một quy luật (rule) và mỗi lá biểu biễn một kết quả (giá trị cụ thể hay một nhánh tiếp tục).



Hình 5: Ví dụ minh họa cây quyết định

****Các bước thuật toán***

Để phân lớp dựa trên cây quyết định bao gồm 2 giai đoạn:

Giai đoạn huấn luyện: Từ tập dữ liệu huấn luyện, tiến hành tạo cây quyết đinh theo các bước sau:

Bước 1: Lựa chọn thuộc tính "tốt nhất" theo một độ đo lựa chọn chotrước.

Bước 2: Mỗi giá trị trên thuộc tính được chọn sẽ thành một nhánh.

Bước 3: Sắp xếp các mẫu dữ liệu tương ứng với các nhánh

Bước 4: Lặp lại bước 1 đến 3 trên các nhánh nếu các mẫu của nhánh chưa được phân lớp rõ.

Bước 5: Dừng thuật toán khi tất cả các mẫu đã được phân lớp rõ.

Giai đoạn phân lớp: Sử dụng cây quyết định vừa xây dựng được để tiến hành phân lớp dữ liệu mới. Đối tượng mới sẽ được đưa qua cây quyết định, bắt đầu từ nút gốc và theo dõi các nhánh dựa trên giá trị của các thuộc tính, cho đến khi đạt đến một nút lá, nơi mà nó sẽ được gán vào một lớp.

***Ưu điểm**

Đơn giản, dễ hiểu và dễ giải thích: Cấu trúc cây quyết định dễ hiểu cho người dùng, dễ dàng giải thích lý do đưa ra các dự đoán.

Ít tốn kém tài nguyên tính toán: Thuật toán huấn luyện và dự đoán của cây quyết định đơn giản, hiệu quả, ít tốn kém tài nguyên.

Khả năng xử lý dữ liệu lớn: Cây quyết định có thể xử lý hiệu quả cả dữ liệu có số chiều lớn.

Có khả năng xử lý cả dữ liệu danh nghĩa và số liệu: Cây quyết định có thể xử lý được cả các thuộc tính danh nghĩa và số liệu.

Khả năng song song hóa tốt trong huấn luyện và dự đoán: Cây quyết định có thể được huấn luyện và dự đoán song song, giúp tăng tốc độ xử lý.

***Nhược điểm**

Dễ bị overfitting: Cây quyết định có xu hướng phù hợp quá mức với dữ liệu huấn luyện, dẫn đến hiệu suất kém trên dữ liệu mới.

Khó hiểu với cây phức tạp: Cây quyết định với độ sâu và chi nhánh lớn trở nên khó hiểu và giải thích cho người dùng.

Dễ bị ảnh hưởng bởi dữ liệu nhiễu: Do dựa trên các ngưỡng split đơn giản nên cây quyết định dễ bị các điểm nhiễu hoặc outliers làm sai lệch quá trình huấn luyện.

Yêu cầu chuẩn bị dữ liệu kỹ lưỡng: Cây quyết định đòi hỏi phải xử lý các giá trị thiếu, ngoại lai, chuẩn hoá dữ liệu trước khi huấn luyện.

2.3.2. Naïve bayes

***Khái niệm**

Navie Bayes Classification (NBC) là một thuật toán thuộc vào nhóm học máycó giám sát, dựa trên định lý Bayes về lý thuyết xác suất, giả định các thuộctính là độc lập mạnh mẽ (naïve) với nhau. Thuật toán này tính xác suất cho cácyếu tố, sau đó chọn kết quả với xác xuất cao nhất. Naïve bayes được ứng dụngrất nhiều trong các lĩnh vực Machine learning dùng để đưa các dự đoán có độ chính xác cao, dựa trên một tập dữ liệu đã được thu thập.

Cơ sở lý thuyết

$$P(H|X) = \frac{P(X|H) * P(H)}{P(X)}$$

Trong đó:

P(H| X): Xác suất xảy ra giả thiết H với điều kiện X

P(X| H): Xác suất để có điều kiện X trong số các biến cố thuộc H

P(H): Xác suất để một biến cố thuộc H

P(X): Xác suất để một biến cố có điều kiện X

Một đối tượng có đặc điểm X sẽ thuộc vào lớp k nếu xác suất để nó rơi vào lớp k là lớn nhất:

$$P(Hx|X) = Max_i\, P(H_i|X)) = Max_i \frac{P(X|H_i) * P(H_i)}{P(X)}$$

Cách hoạt động

Naïve bayes Classification hoạt động như sau:

- Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n giá trị thuộc tính A1, A2, … An = $\{x_1, x_2, x_3,\ldots,x_n\}$

- Giả sử có m lớp $C_1, C_2, C_3, ..., C_m$. cho một phần tử dữ liệu X, bộ phận lớp sẽ gán nhãn cho X là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán X thuộc vào lớp C_i nếu và chỉ nếu:

$$P = \pi r^2$$

Giá trị này sẽ dựa trên định lí Bayes

- Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(X) là các giá trị giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của P(X|Ci) * P(Ci) Chú ý rằng P(Ci) được ước lượng bằng |Di|/|D|, trong đó Di là tập các phần tử dữ liệu thuộc lớp Ci. Nếu xác suất tiền nghiệm P(Ci) cũng không xác định được thì ta coi chúng bằng nhau P(C1) = P(C2) = ... = P(Cm), khi đó ta chỉ cần tìm giá trị P(X|Ci) lớn
- Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toán P(X|Ci) là rất lớn, dó đó có thể giảm độ phức tạp của thuật toán Naïve Bayes giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:

P(X|Ci) = P(x1|Ci)...P(xn|Ci)

Ưu điểm

Đơn giản, dễ hiểu và dễ giải thích: Cấu trúc của Naive Bayes dễ hiểu cho người dùng, dễ dàng giải thích lý do đưa ra các dự đoán.

Ít tốn kém tài nguyên tính toán: Thuật toán huấn luyện và dự đoán của Naïve Bayes đơn giản, hiệu quả, ít tốn kém tài nguyên.

Khả năng xử lý dữ liệu lớn: Naive Bayes có thể xử lý hiệu quả cả dữ liệu có số chiều lớn.

Có khả năng xử lý cả dữ liệu danh nghĩa và số liệu: Naive Bayes có thể xử lý được cả các thuộc tính danh nghĩa và số liệu.

Ổn định và ít bị overfitting hơn các mô hình khác: Nếu được chọn tham số phù hợp, Naive Bayes sẽ ổn định và ít bị overfitting.

Nhược điểm

Giả định độc lập: Naive Bayes giả định rằng các thuộc tính là độc lập với nhau. Giả định này có thể không đúng với thực tế trong nhiều trường hợp. Điều này có thể dẫn đến giảm hiệu suất của Naive Bayes.

Khả năng dự đoán dữ liệu chưa từng thấy: Naive Bayes phụ thuộc nhiều vào dữ liệu huấn luyện nên có thể khó dự đoán chính xác các điểm dữ liệu mới hoàn toàn khác với dữ liệu đã học.

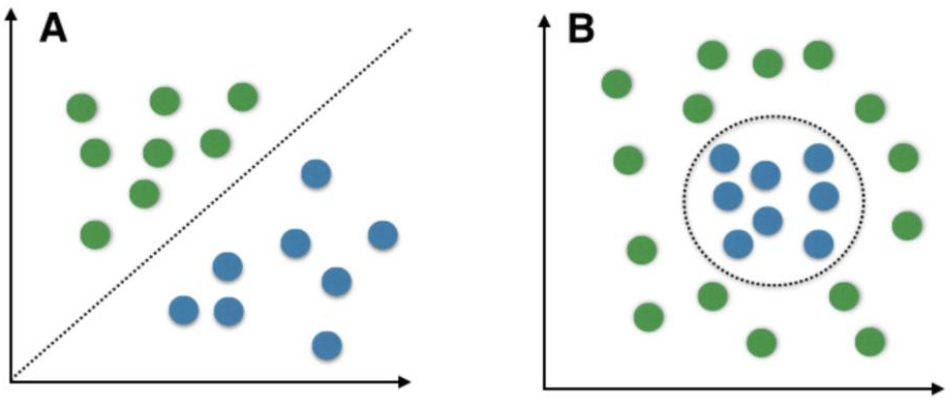
2.3.3. Máy vector hỗ trợ (SVM)

**** Khái niệm***

Support Vector Machine (SVM) là một trong những thuật toán học có giám sát phổ biến nhất. Tuy nhiên, chủ yếu, nó được sử dụng cho các vấn đề phân loại trong các bài toán học máy

SVM là một mô hình phân loại hoạt động bằng việc xây dựng một siêu phẳng (hyperplane) có (n - 1) chiều trong không gian n chiều của dữ liệu sao cho siêu phẳng này phân loại các lớp một cách tối ưu nhất (Khoảng cách từ các đối tượng gần nhất tới siêu phẳng là cực đại). Nói cách khác, cho một tập dữ liệu có nhãn (học có giám sát), thuật toán sẽ dựa trên dữ liệu học để xây dựng một siêu phẳng tối ưu được sử dụng để phân loại dữ liệu mới. Ở không gian 2 chiều thì siêu phẳng này là 1 đường thẳng phân cách chia mặt phẳng không gian thành 2 phần tương ứng 2 lớp với mỗi lớp nằm ở 1 phía của đường thẳng.

Siêu phẳng tạo ra biên giới phân chia 2 lớp của dữ liệu



Hình 6: Minh họa siêu phẳng trong mô hình SVM

***Cách hoạt động**

Với một tập dữ liệu huấn luyện cho trước ban đầu gồm n mẫu trong không gian $\mathbf{R}^{d+1}$ (d thuộc tính dữ liệu và 1 thuộc tính lớp) với các mẫu dữ liệu thuộc vào 1 trong 2 lớp (tạm ký hiệu là lớp +1 và lớp -1). Kỹ thuật SVM bao gồm 2 giai đoạn như sau:

Giai đoạn huấn luyện:

Giai đoạn này là quá trình đi tìm một siêu phẳng phân tách tốt nhất tập dữ liệu huấn luyện thành hai lớp +1 và -1. Trong không gian hai chiều, siêu phẳng là một đường thẳng, trong không gian ba chiều là một mặt phẳng. Một cách tổng quát, trong không gian d chiều ta gọi chúng là siêu phẳng.

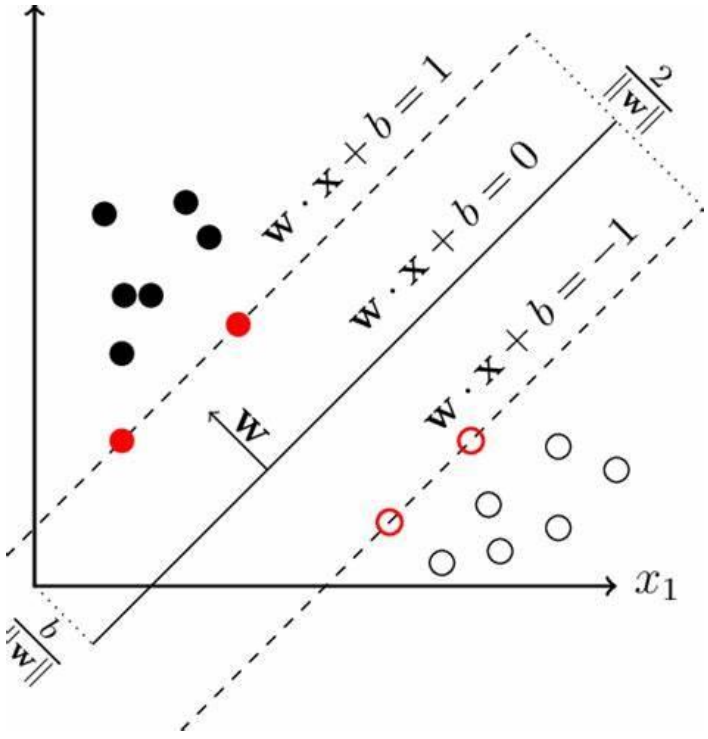
Gọi $x \in R^d$ là các điểm trong không gian d chiều và một siêu phẳng sẽ có dạng: $<w, x>+ b = 0$, trong đó $w \in R^d$ là véc tơ chuẩn của siêu phẳng và $b \in R$ là ngưỡng (bias). Với $<w, x>$ là tích vô hướng của hai véc tơ được định nghĩa trong không gian R^d Dễ thấy rằng để xác định một siêu phẳng như vậy, chúng ta cần xác định bộ các hệ số {w, b} tương ứng.

Một siêu phẳng phân tách bộ dữ liệu thành hai miền (+1 và -1) một cách tốt nhất được định nghĩa là siêu phẳng có khoảng cách lề (margin) giữa hai lớp đạt

cực đại. hình dưới đây chỉ ra ví dụ về một siêu phẳng phân tách hai lớp dữ liệu một cách tốt nhất trong đó lề giữa hai lớp được tính bằng 2 / ||w||. ||w|| là Norm 2, là khoảng các euclidean từ w tới O và bằng căn bậc hai của tổng bình phương các Một siêu phẳng phân tách bộ dữ liệu thành hai miền (+1 và -1) một cách tốt nhất được định nghĩa là siêu phẳng có khoảng cách lề (margin) giữa hai lớp đạt cực đại. Hình sau chỉ ra ví dụ về một siêu phẳng phân tách hai lớp dữ liệu một cách tốt nhất trong đó lề giữa hai lớp được tính bằng 2 / ||w||. ||w|| là Norm 2, là khoảng các euclidean từ w tới O và bằng căn bậc hai của tổng bình phương các giá trị.

Giai đoạn dự đoán

Sau khi đã tìm ra siêu phẳng tối ưu, SVM sẽ sử dụng nó để phân loại dữliệu mới dựa vào vị trí của điểm dữ liệu mới so với siêu phẳng. Nếu điểm dữ liệu nằm ở một phía của siêu phẳng, nó sẽ được phân vào lớp tương ứng.



Hình 7: Minh họa siêu phẳng phân tách dữ liệu tốt nhất

Ưu điểm

Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệuquả trong không gian chiều cao, đặc biệt là áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn.Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định.

Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng

Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

Nhược điểm

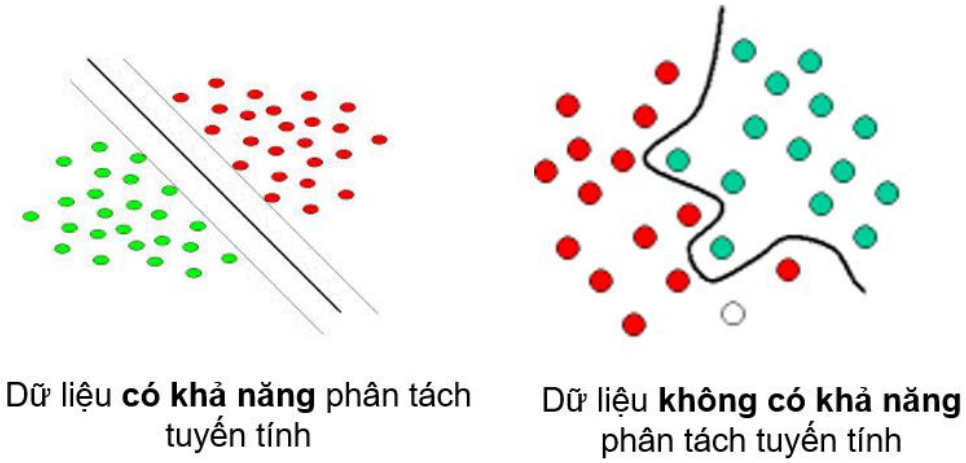
Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu thì SVM cho kết quả khá tồi.

Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM, chưa giải thích được xác suất xuất hiện của một thuộc tính trong một nhóm thuộc tính như thế nào. Hiệu quả của việc phân lớp xác định dựa vào khái niệm

Margin từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã nói đến ở phía trên.

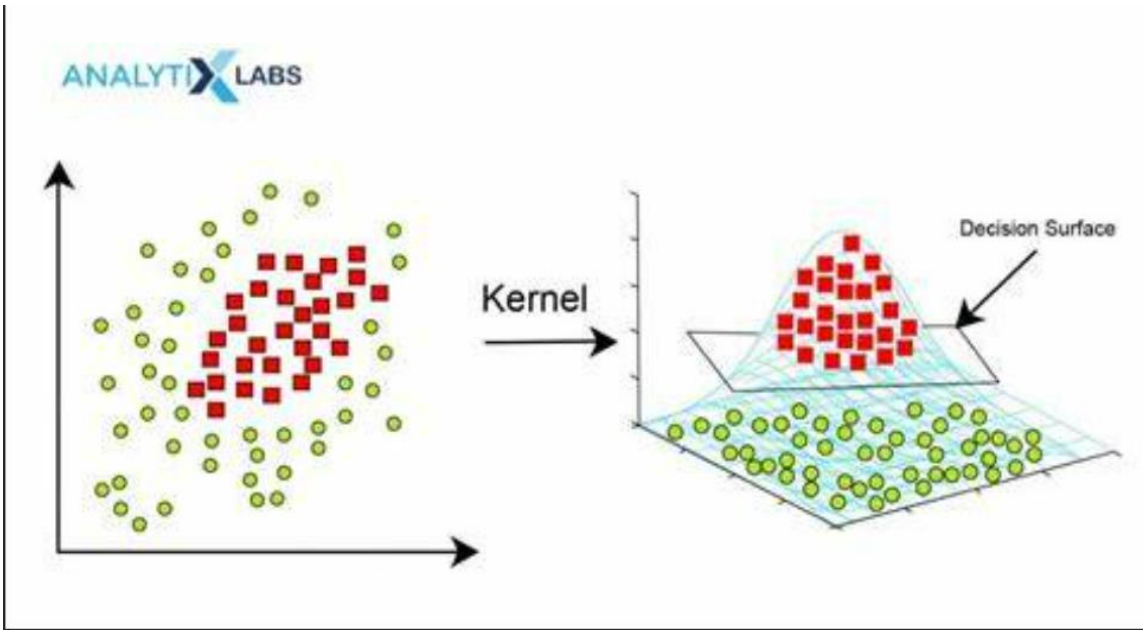
2.3.4. SVM với các nhân kernel

Trên thực tế, dữ liệu có thể phân tách tuyến tính hoặc không phân tách tuyến tính. SVM có thể hoạt động tốt trên dữ liệu phân tác tuyến tính.



Hình 8: Minh họa dữ liệu thực tế trong phân loại khi sử dụng SVM

Trường hợp dữ liệu không phân tách tuyến tính, ta sử dụng các hàm kernel. Các hàm kernel ánh xạ các đối tượng sang một không gian với số chiều lớn hơn. Trong không gian mới này, dữ liệu có nhiều khả năng phân tác tuyến tính hơn, từ đó kết quả phân lớp có thể sẽ tốt hơn. Có rất nhiều hàm kernel khác nhau được định nghĩa. Trong đó phổ biến là hàm kernel Poly, Gaussian kernel (RBF),…



Hình 9: Minh họa kernel trong thực tế

2.3.5. SVM và bài toán phân loại đa lớp.

Thuật toán SVM gốc chỉ có thể phân loại hai lớp. Để giải quyết bài toán phân loại đa lớp, có hai phương pháp chính:

SVMs (Support Vector Machines for Multiclass Classification): Sử dụng nhiều bộ phân loại nhị phân để phân loại đa lớp. Có hai chiến lược chính:

Chiến lược One vs All: Xây dựng n mô hình phân loại nhị phân, với mỗi mô hình phân biệt một lớp với tất cả các lớp còn lại.Chiến lược One vs One: Xây dựng n*(n-1)/2 mô hình phân loại nhị phân, với mỗi mô hình phân biệt hai lớp với nhau.

MSVM (Multiclass Support Vector Machines): Tìm ra một tập N siêu phẳng để phân loại đa lớp. Có một số mô hình MSVM phổ biến, bao gồm:

Mô hình MSVM của Weston & Watkin: Sử dụng phương pháp đối ngẫu để tối ưu hóa hàm mục tiêu.Mô hình MSVM của Cramer & Singer: Sử dụng phương pháp trực tiếp để tối ưu hóa hàm mục tiêu.Mô hình MSVM của Lee & Lin: Sử dụng phương pháp cộng hưởng để tối ưu hóa hàm mục tiêu.Mô hình MSVM tổng quát của Yann Guermeur: Sử dụng phương pháp tối ưu lồi để tối ưu hóa hàm mục tiêu.

Ưu nhược điểm của hai phương pháp

SVMs

Ưu điểm:

Dễ dàng triển khai và huấn luyện.Có thể đạt được độ chính xác cao.

Nhược điểm:

Sử dụng nhiều mô hình, có thể dẫn đến việc huấn luyện chậm và tốn tài nguyên.

MSVM

Ưu điểm:

Chỉ sử dụng một mô hình, có thể huấn luyện nhanh và tiết kiệm tài nguyên.Có thể đạt được độ chính xác cao.Có thể áp dụng cho các bài toán phân loại có số lớp lớn.

Nhược điểm:

Khó triển khai và huấn luyện hơn SVMs.Có thể không đạt được độ chính xác cao như SVMs với các tập dữ liệu nhỏ.

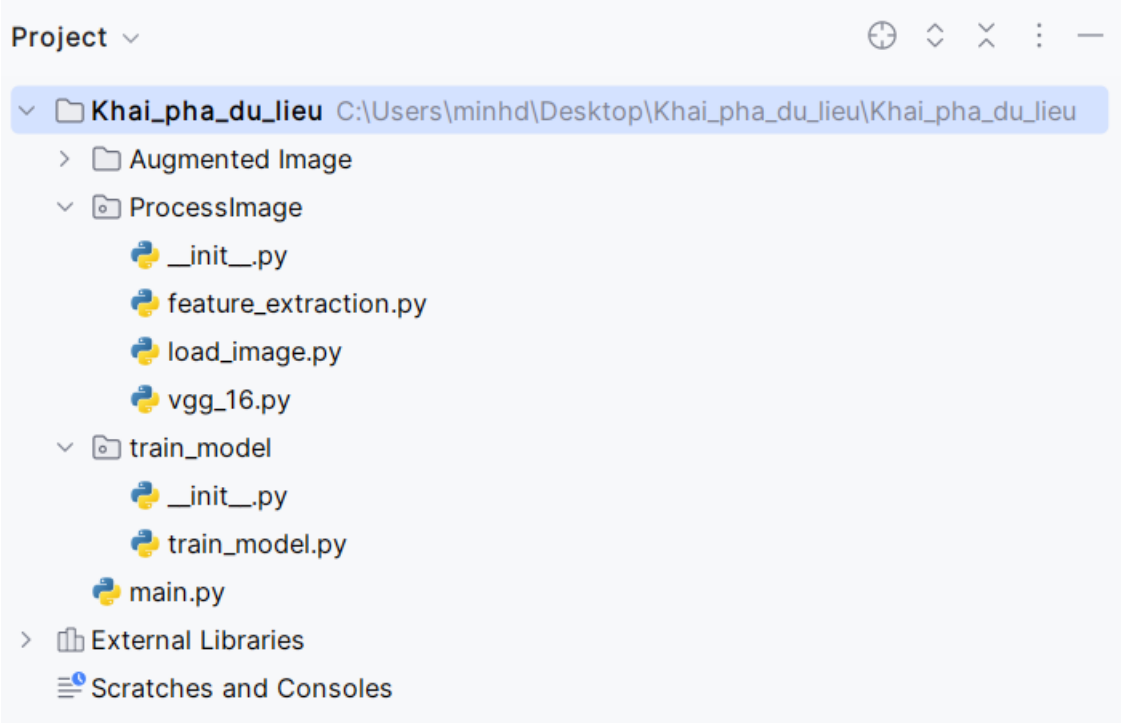
CHƯƠNG 3 THỰC NGHIỆM

3.1. Dữ liệu thực nghiệm

Tập dữ liệu này chứa 6400 - hình ảnh ở định dạng JPEG với chú thích khung giới hạn của biển số xe ô tô trong hình ảnh. Mỗi ảnh được đánh tên với bênh/ bình thường tương ứng và theo số thứ tự.

3.2. Triển khai thực nghiệm

3.2.1. Thực hiện triển khai thành các package và module



Hình 10: Tổ chức package và module

Package ProcessImage: gồm các module xử lí và hình ảnh, trích chọn đặc trưng, bao gồm cả module vgg16

Package Train model: gồm các module train và load data

3.2.2. Package ProcessImage

Trong pakage này, chúng em thực hiện triển khai các module

3.2.2.1. Module vgg_16

Trong module này thực hiện khởi tạo mô hình vgg16 nhằm thực hiện trích chọn đặc trưng

Code chương trình:

```
from keras.applications.vgg16 import VGG16

vgg_model = VGG16(weights='imagenet', include_top=False)
```

Đoạn mã này sử dụng thư viện Keras để tải mô hình VGG16 đã được huấn luyện trước trên tập dữ liệu ImageNet. Hãy phân tích mã:

```
from keras.applications.vgg16 import VGG16
```

Dòng này nhập lớp mô hình VGG16 từ module applications của thư viện Keras. Keras cung cấp nhiều mô hình học sâu đã được huấn luyện trước, và VGG16 là một trong số đó. VGG16 là một kiến trúc mạng nơ-ron tích chập đã chứng minh hiệu suất tốt trong nhiều nhiệm vụ thị giác máy tính.

```
vgg_model = VGG16(weights='imagenet', include_top=False)
```

Ở đây, một thể hiện của mô hình VGG16 được tạo ra. Tham số weights được đặt là 'imagenet', có nghĩa là mô hình sẽ được tải với trọng số được huấn luyện trước từ tập dữ liệu ImageNet. ImageNet là một tập dữ liệu lớn với hàng triệu hình ảnh được gán nhãn trong hàng nghìn lớp, và các mô hình được huấn luyện trên nó có thể hữu ích cho nhiều nhiệm vụ liên quan đến hình ảnh.

Tham số include_top được đặt là False, điều này có nghĩa là các lớp kết nối đầy đủ (lớp top) của mô hình VGG16, chịu trách nhiệm cho việc phân loại, không được bao gồm. Điều này hữu ích nếu bạn muốn sử dụng mô hình VGG16 như một trích xuất đặc trưng cho một nhiệm vụ khác, chẳng hạn như học chuyển giao.

3.2.2.2. Module load_image

Code chương trình

```
import numpy as np
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image


def load_and_preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)
    return img_array
```

Đoạn mã này sử dụng thư viện Keras để tải và tiền xử lý một hình ảnh theo chuẩn của mô hình VGG16. Cụ thể:

1. `import numpy as np`: Import thư viện NumPy để làm việc với mảng số.
2. `from keras.applications.vgg16 import preprocess_input`: Import hàm `preprocess_input` từ module VGG16 để tiền xử lý dữ liệu hình ảnh trước khi đưa vào mô hình.
3. from keras.preprocessing import image`: Import module `image` từ Keras để xử lý hình ảnh.
4. Định nghĩa hàm `load_and_preprocess_image(img_path)`: Hàm này nhận đường dẫn của hình ảnh, tải và tiền xử lý nó để chuẩn bị cho việc đưa vào mô hình VGG16.
    - `img = image.load_img(img_path, target_size=(224, 224))`: Tải hình ảnh và resize về kích thước 224x224 pixel.
    - `img_array = image.img_to_array(img)`: Chuyển hình ảnh thành mảng số.
    - `img_array = np.expand_dims(img_array, axis=0)`: Mở rộng chiều để tạo thành batch với một hình ảnh.

- `img_array = preprocess_input(img_array)`: Tiền xử lý dữ liệu theo chuẩn của mô hình VGG16.
- `return img_array`: Trả về mảng số đã được tiền xử lý.

3.2.2.3. Module feature_extraction

Code chương trình

```
from ProcessImage.load_image import load_and_preprocess_image
from ProcessImage.vgg_16 import vgg_model


def extract_features(img_path):
    img = load_and_preprocess_image(img_path)
    vgg_features = vgg_model.predict(img)
    vgg_features = vgg_features.flatten()
    return vgg_features
```

Đoạn mã này sử dụng hai module, `load_image` và `vgg_16`, để trích xuất đặc trưng từ một hình ảnh bằng mô hình VGG16:

1. `from ProcessImage.load_image import load_and_preprocess_image`: Import hàm `load_and_preprocess_image` từ module `load_image` để tải và tiền xử lý hình ảnh.
2. `from ProcessImage.vgg_16 import vgg_model`: Import mô hình VGG16 từ module `vgg_16`.
3. Định nghĩa hàm `extract_features(img_path)`: Hàm này nhận đường dẫn của hình ảnh, tải và tiền xử lý nó bằng hàm `load_and_preprocess_image`, sau đó trích xuất đặc trưng sử dụng mô hình VGG16.
   - `img = load_and_preprocess_image(img_path)`: Sử dụng hàm `load_and_preprocess_image` để tải và tiền xử lý hình ảnh.
   - `vgg_features = vgg_model.predict(img)`: Sử dụng mô hình VGG16 để dự đoán đặc trưng của hình ảnh.

- `vgg_features = vgg_features.flatten()`: Chuyển mảng đặc trưng về dạng mảng một chiều.
- `return vgg_features`: Trả về đặc trưng đã được trích xuất.

3.2.3. Package train_model

Hàm *plot_metrics_across_folds()*: Vẽ đồ thị các độ đo (F1-score, Accuracy, Recall, Precision) qua từng fold trong quá trình cross-validation.

```
def plot_metrics_across_folds(f1_scores, accuracies, recalls,
precisions):
    plt.figure(figsize=(10, 6))

    plt.plot(f1_scores, label='F1-score', marker='o')
    plt.plot(accuracies, label='Accuracy', marker='o')
    plt.plot(recalls, label='Recall', marker='o')
    plt.plot(precisions, label='Precision', marker='o')

    plt.title('Metrics Across Folds')
    plt.xlabel('Fold')
    plt.ylabel('Score')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Hàm *load_data()*: Tải dữ liệu hình ảnh từ thư mục *'dataset_path'* và trả về dữ liệu và nhãn đã được tiền xử lý.

```
def load_data(dataset_path):
    labels = os.listdir(dataset_path)
    data = []
    target = []

    for label in labels:
        label_path = os.path.join(dataset_path, label)
        for img_name in os.listdir(label_path):
            img_path = os.path.join(label_path, img_name)
            features = extract_features(img_path)
            data.append(features)
            target.append(label)

    data = np.array(data)
    target = np.array(target)
```

```
    label_to_numeric = {label: idx for idx, label in
enumerate(np.unique(target))}
    target = np.array([label_to_numeric[label] for label in
target])

    return data, target
```

Hàm *train_test_split_data()*: Chia dữ liệu thành tập huấn luyện và tập kiểm thử.

```
def train_test_split_data(data, target, test_size=0.2,
random_state=42):
    return train_test_split(data, target, test_size=test_size,
random_state=random_state)
```

Hàm *perform_k_fold_cross_validation()*: Thực hiện quá trình k-fold cross-validation với mô hình SVM và trả về mô hình có độ chính xác cao nhất.

```
def perform_k_fold_cross_validation(data, target, C_value):
    kf = KFold(n_splits=5, shuffle=True, random_state=42)

    best_accuracy = 0
    best_model = None
    fold = 1

    f1_scores = []
    accuracies = []
    recalls = []
    precisions = []

    for train_index, test_index in kf.split(data):
        X_train, X_test = data[train_index], data[test_index]
        y_train, y_test = target[train_index],
target[test_index]

        svm_model = LinearSVC(multi_class='crammer_singer',
C=C_value, max_iter=1000)
        svm_model.fit(X_train, y_train)

        y_pred = svm_model.predict(X_test)

        # Calculate metrics
```

```
        accuracy = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred,
average='weighted')
        precision = precision_score(y_test, y_pred,
average='weighted')

        accuracies.append(accuracy)
        f1_scores.append(f1)
        recalls.append(recall)
        precisions.append(precision)

        print(f"Metrics for Fold {fold}:")
        print(f"Accuracy: {accuracy}")
        print(f"Precision: {precision}")
        print(f"Recall: {recall}")
        print(f"F1-score: {f1}")
        print(f"Classification report for Fold
{fold}:\n{classification_report(y_test, y_pred)}")

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_model = svm_model
            print(f"Updating best model for Fold {fold}")

        fold += 1

    return best_model, best_accuracy, f1_scores, accuracies,
recalls, precisions
```

Hàm *train_and_evaluate_svm()*: Huấn luyện và đánh giá mô hình SVM với các giá trị tham số C khác nhau, chọn giá trị C tối ưu.

```
def train_and_evaluate_svm(X_train, X_test, y_train, y_test,
C_values):
    best_accuracy = 0
    best_C = None

    for C in C_values:
        svm_model = LinearSVC(multi_class='crammer_singer',
C=C, max_iter=1000)
        svm_model.fit(X_train, y_train)

        y_pred = svm_model.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
```

```
        f1 = f1_score(y_test, y_pred, average='weighted')
        precision = precision_score(y_test, y_pred,
average='weighted')
        recall = recall_score(y_test, y_pred,
average='weighted')

        print(f"Accuracy for C={C}: {accuracy}")
        print(f"F1 score for C={C}: {f1}")
        print(f"Precision for C={C}: {precision}")
        print(f"Recall for C={C}: {recall}")
        print(f"Classification report for
C={C}:\n{classification_report(y_test, y_pred)}")

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_C = C

    print(f"Best accuracy: {best_accuracy} achieved with
C={best_C}")
    return best_C
```

Hàm *save_best_model()*: Lưu trữ mô hình tốt nhất vào một tệp tin sử dụng thư viện joblib.

```
def save_best_model(model, accuracy):
    if model is not None:
        best_model_filename = "best_svm_model.joblib"
        joblib.dump(model, best_model_filename)
        print(f"Best model saved as {best_model_filename} with
accuracy: {accuracy}")
    else:
        print("No best model found.")
```

3.2.4. Hàm main()

Code chương trình:

```
from joblib import load

# Đường dẫn đến file chứa model
file_path = 'best_svm_model.joblib'

# Tải model từ file
model = load(file_path)
```

```
if model is not None:
    print("Load model thành công")
else:
    print("Không thể load được model")
```

Đoạn mã này sử dụng thư viện joblib để tải một mô hình máy học đã được lưu trữ từ một tệp tin

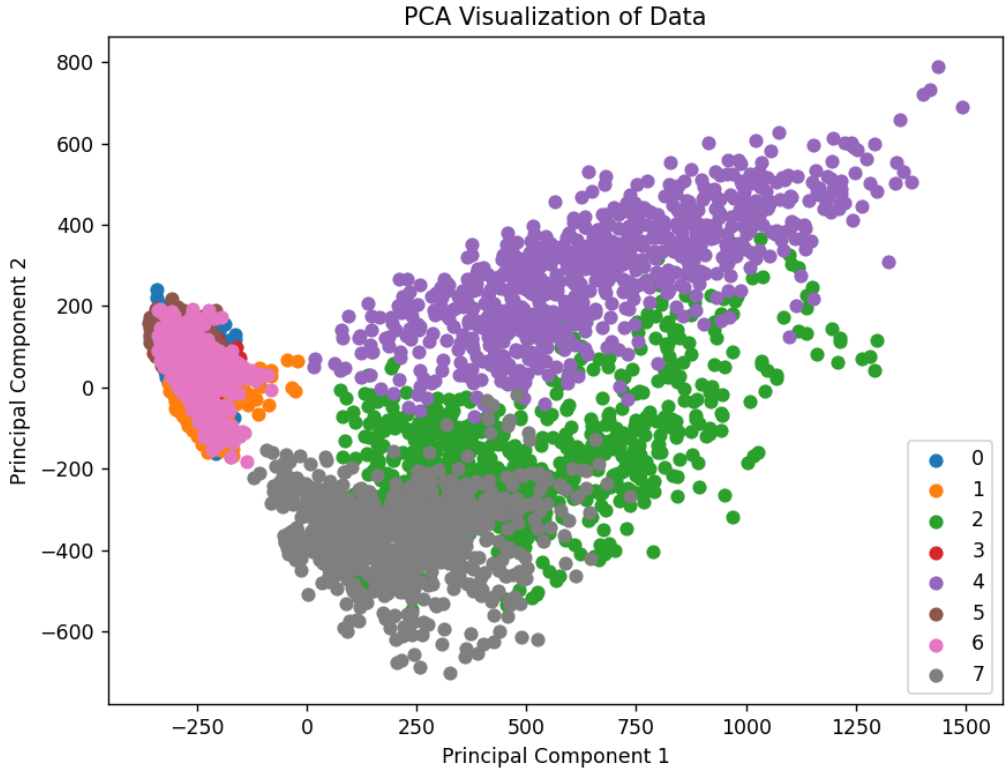
Đoạn mã này sử dụng thư viện `joblib` để tải một mô hình máy học đã được lưu trữ từ một tệp tin. Dưới đây là giải thích:

1. Import thư viện joblib: Dòng này import hàm `load` từ thư viện `joblib`, giúp chúng ta tải các đối tượng đã được lưu trữ.
2. Đường dẫn đến file chứa model: `file_path` là đường dẫn tới tệp tin chứa mô hình SVM đã được lưu trữ.
3. Tải model từ file: Dòng này sử dụng hàm `load` để tải mô hình từ tệp tin được chỉ định vào biến `model`.
4. Kiểm tra việc tải model thành công: Kiểm tra xem việc tải mô hình đã thành công hay không. Nếu `model` khác `None`, tức là mô hình đã được tải thành công, sẽ in ra thông báo "Load model thành công", ngược lại sẽ in ra "Không thể load được model".

3.3. Các kết quả thực nghiệm

Thực hiện trực quan hóa dữ liệu và nhãn

Dữ liệu được trực quan hóa thông qua giảm chiều dữ liệu, với các nhãn được chuẩn hóa thành số (đánh số thứ tự từ 0 đến 7)



Hình 11: Trực quan hóa dữ liệu

***Thực hiện train model với mục đích tìm tham số "C" tốt nhất phù hợp với bài toán và lưu lại tham số "C" tốt nhất**

- Với C = 0.001

```
Accuracy for C=0.001: 0.965625
F1 score for C=0.001: 0.965519320578149
Precision for C=0.001: 0.9655892818871162
Recall for C=0.001: 0.965625
Classification report for C=0.001:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 12: Bảng kết quả train mô hình với tham số C = 0.001

- Với C = 0.01

```
Accuracy for C=0.01: 0.965625
F1 score for C=0.01: 0.965519320578149
Precision for C=0.01: 0.9655892818871162
Recall for C=0.01: 0.965625
Classification report for C=0.01:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 13: Bảng kết quả train mô hình với tham số C = 0.01

- Với C = 0.1

```
Accuracy for C=0.1: 0.965625
F1 score for C=0.1: 0.965519320578149
Precision for C=0.1: 0.9655892818871162
Recall for C=0.1: 0.965625
Classification report for C=0.1:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 14: Bảng kết quả train mô hình với tham số C = 0.1

- Với C = 1

```
Accuracy for C=1: 0.965625
F1 score for C=1: 0.965519320578149
Precision for C=1: 0.9655892818871162
Recall for C=1: 0.965625
Classification report for C=1:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 15: Bảng kết quả train mô hình với tham số C = 1

- Với C = 10

```
Accuracy for C=10: 0.965625
F1 score for C=10: 0.965519320578149
Precision for C=10: 0.9655892818871162
Recall for C=10: 0.965625
Classification report for C=10:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 16: Bảng kết quả train mô hình với tham số C = 10

Độ chính xác tốt nhất và tham số C tốt nhất là:

```
Best accuracy: 0.965625 achieved with C=0.001
```

Hình 17: Kết quả mô hình tốt nhất

Thực hiện đánh giá model với duyệt K-fold

- Fold 1

```
Metrics for Fold 1:
Accuracy: 0.965625
Precision: 0.9655892818871162
Recall: 0.965625
F1-score: 0.965519320578149
Classification report for Fold 1:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       185
           1       0.96      0.98      0.97       162
           2       1.00      1.00      1.00       159
           3       0.97      0.95      0.96       161
           4       1.00      1.00      1.00       156
           5       0.96      0.99      0.97       153
           6       0.94      0.92      0.93       153
           7       1.00      1.00      1.00       151

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 18: Bảng kết quả train mô hình với Fold 1

- Fold 2

```
Metrics for Fold 2:
Accuracy: 0.959375
Precision: 0.9595395220420976
Recall: 0.959375
F1-score: 0.959067882554027
Classification report for Fold 2:
              precision    recall  f1-score   support

           0       0.93      0.85      0.89       175
           1       0.98      0.95      0.96       157
           2       1.00      1.00      1.00       138
           3       0.91      0.95      0.93       169
           4       1.00      1.00      1.00       167
           5       0.94      0.99      0.97       157
           6       0.93      0.95      0.94       158
           7       1.00      1.00      1.00       159

    accuracy                           0.96      1280
   macro avg       0.96      0.96      0.96      1280
weighted avg       0.96      0.96      0.96      1280
```

Hình 19: Bảng kết quả train mô hình với Fold 2

- Fold 3

```
Metrics for Fold 3:
Accuracy: 0.96953125
Precision: 0.9694328224696701
Recall: 0.96953125
F1-score: 0.9693824043202532
Classification report for Fold 3:
              precision    recall  f1-score   support

           0       0.90      0.87      0.88       135
           1       0.99      0.98      0.99       172
           2       1.00      0.98      0.99       174
           3       0.95      0.98      0.96       145
           4       0.98      1.00      0.99       153
           5       0.97      0.98      0.98       161
           6       0.95      0.95      0.95       166
           7       1.00      1.00      1.00       174

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 20: Bảng kết quả train mô hình với Fold 3

- Fold 4

```
Metrics for Fold 4:
Accuracy: 0.96640625
Precision: 0.9667431087985916
Recall: 0.96640625
F1-score: 0.9664148044860019
Classification report for Fold 4:
              precision    recall  f1-score   support

           0       0.92      0.89      0.91       166
           1       0.98      0.98      0.98       165
           2       1.00      1.00      1.00       163
           3       0.91      0.97      0.94       152
           4       1.00      1.00      1.00       162
           5       0.98      0.96      0.97       169
           6       0.94      0.94      0.94       154
           7       1.00      1.00      1.00       149

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 21: Bảng kết quả train mô hình với Fold 4

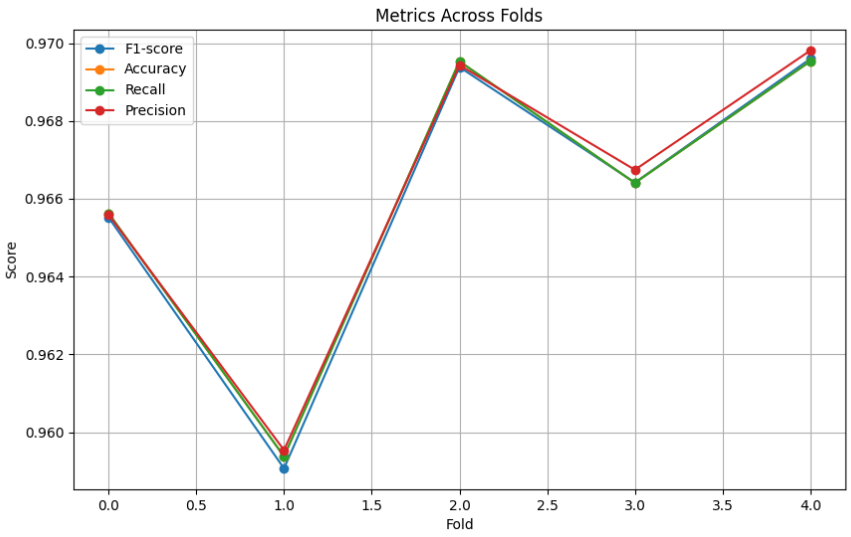
- Fold 5:

```
Metrics for Fold 5:
Accuracy: 0.96953125
Precision: 0.9698082547018613
Recall: 0.96953125
F1-score: 0.969595196232274
Classification report for Fold 5:
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       139
           1       0.99      0.99      0.99       144
           2       0.99      1.00      1.00       166
           3       0.97      0.96      0.96       173
           4       0.99      1.00      1.00       162
           5       0.97      0.96      0.97       160
           6       0.94      0.92      0.93       169
           7       1.00      0.99      0.99       167

    accuracy                           0.97      1280
   macro avg       0.97      0.97      0.97      1280
weighted avg       0.97      0.97      0.97      1280
```

Hình 22: Bảng kết quả train mô hình với Fold 5

Sau khi chạy 5 fold và Cập nhất Model ở fold 3 với độ chính xác tốt nhất.

Đồ thị tổng quan về các chỉ số metrics sau khi thực hiên chạy 5 fold:



Hình 23: Đồ thị biểu diễn các chỉ số metrics khi thực hiện 5 Folds

KẾT LUẬN

Trong báo cáo này, chúng em đã thực hiện và nghiên cứu mô hình SVM để thực hiện giải quyết bài toán: Nhận diện biển số xe

**Sau đây là các kết quả mà chúng em đã thu được:*

- Học được cách xây dựng, huấn luyện và triển khai mô hình SVM cho bài toán phân loại đa lớp
- Hiểu rõ cách vận dụng mô hình SVM để giải quyết bài toán phân loại bệnh trên cây trồng
- Nắm được quy trình xử lý dữ liệu, huấn luyện, đánh giá và triển khai mô hình học máy.
- Trực quan hóa dữ liệu và quá trình huấn luyện

**Bên cạnh đó báo cáo của em còn nhiều hạn chế như:*

- Chưa xây dựng được tập dữ liệu đầy đủ về các loại bệnh và chỉ giới hạn trên cây dưa leo, cũng chưa thu thập được đa dạng các loại bênh trên cây dưa leo trong các điều kiện khí hậu khác nhau
- Vấn đề đọc ảnh cũng khá tốn thời gian và chưa được tối ưu hóa
- Chưa triển khai thành một ứng dụng cụ thể (bao gồm giao diện, cơ sở dữ liệu…)

Nhóm em hy vọng những nghiên cứu và kết quả trong báo cáo này sẽ góp phần giải quyết hiệu quả bài toán phân loại bệnh trên cây dưa leo nói chung các loại cây khác nói riêng, từ đó ứng dụng công nghệ vào ngành nông nghiệp . Đây cũng là hướng nghiên cứu tiềm năng để áp dụng trí tuệ nhân tạo trong ngành nông nghiệp.

TÀI LIỆU THAM KHẢO

1. Tác phẩm sách

[1] Lê Thị Thuỷ, Trần Hùng Cường, Ngô Thị Bích Thuý. (2019). Giáo trình xử lý ảnh.

[2] Phạm Ngọc Hùng, Trương Anh Hoàng, Đặng Văn Hưng (2014). Giáo trình kiểm thử phần mềm.

[3] Richard Szeliski. (2022). Computer Vision.

[4] Anirudh Koul, Siddha Ganju, Meher Kasam. (2019). Practical Deep Learning for Cloud, Mobile, and Edge.

2. Website

[1] https://www.lisado.vn/cac-loai-benh-dua-leo-va-cach-dieu-tri/

[2] https://hatgiongtruongphuc.vn/blogs/news/tat-tan-tat-ve-cay-dua-leo

[3] https://tapchi.vaas.vn/sites/tapchi.vaas.vn/files/tapchi/2021-08/tc5_3.pdf

[4] https://ohstem.vn/hoc-co-giam-sat-machine-learning/

[5] https://aiwithmisa.com/2020/12/12/aml-bai5/

[6] https://www.geeksforgeeks.org/image-classification-using-support-vector-machine-svm-in-python/

[7] VGG-16 | Mô hình CNN - GeeksforGeeks