**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**MÔN: HỌC MÁY**

**ĐỀ TÀI**

**DỰ BÁO THỜI TIẾT BẰNG THUẬT TOÁN HỌC MÁY**

*Giảng viên hướng dẫn: ThS.Trần Anh Đạt*

***Nhóm sinh viên thực hiện: Nhóm 8\_64CNTT3***

|  |  |
| --- | --- |
| **Họ tên sinh viên:** | **Mã sinh viên:** |
| Trần Đăng Hiếu | 2251061779 |
| Nguyễn Khắc Trung | 2251061905 |
| Nguyễn Thị Phương | 2251162117 |
| Nguyễn Trí Duy | 2251061760 |

**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc181017036)

[CHƯƠNG I Quan sát dữ liệu, lựa chọn mô hình 4](#_Toc181017037)

[1 Quan sát dữ liệu. 4](#_Toc181017038)

[2 Phân tích dữ liệu. 5](#_Toc181017039)

[2.1 Chỉ số thống kê mô tả. 5](#_Toc181017040)

[2.2 Phân phối dữ liệu. 7](#_Toc181017041)

[2.3 Phân tích giá trị ngoại lệ. 8](#_Toc181017042)

[2.4 Phân tích ma trận tương quan. 9](#_Toc181017043)

[3 Nhận xét dữ liệu. 10](#_Toc181017044)

[4 Lựa chọn mô hình. 10](#_Toc181017045)

[CHƯƠNG II Đánh giá mô hình. 11](#_Toc181017046)

[1 Các mô hình được sử dụng. 11](#_Toc181017047)

[1.1 Perceptron. 11](#_Toc181017048)

[1.2 ID3. 23](#_Toc181017049)

[1.3 Neural Network. 37](#_Toc181017050)

[1.4 Stacking. 51](#_Toc181017051)

[2 Các tham số đánh giá. 61](#_Toc181017052)

[3 Đánh giá mô hình. 63](#_Toc181017053)

[3.1 Đánh giá qua các tham số đánh giá. 63](#_Toc181017054)

[3.2 Nhận xét. 64](#_Toc181017055)

[3.3 Kết luận. 65](#_Toc181017056)

[CHƯƠNG III Chương trình DEMO. 66](#_Toc181017057)

[1 Tổng quan chương trình 66](#_Toc181017058)

[2 Chạy demo chương trình 66](#_Toc181017059)

[CHƯƠNG IV Nhận xét tổng thể 69](#_Toc181017060)

[1 Nhận xét tổng quan 69](#_Toc181017061)

[2 Hướng cải thiện 69](#_Toc181017062)

[KẾT LUẬN 71](#_Toc181017063)

# LỜI NÓI ĐẦU

Trong bối cảnh Cách mạng công nghiệp 4.0, công nghệ thông tin và trí tuệ nhân tạo đang ngày càng trở thành nền tảng cho các lĩnh vực như kinh tế, khoa học, và đời sống. Với sự phát triển của các công nghệ học máy hiện đại, chúng ta có thể dự đoán và xử lý một khối lượng lớn dữ liệu, từ đó giúp giải quyết các vấn đề phức tạp với độ chính xác cao và nhanh chóng. Một trong những lĩnh vực được hưởng lợi nhiều nhất từ học máy là dự báo thời tiết, vốn có ảnh hưởng trực tiếp tới nhiều khía cạnh của cuộc sống con người, từ nông nghiệp, giao thông, đến quản lý thiên tai.

Ứng dụng học máy trong dự báo thời tiết giúp cải thiện khả năng dự báo với các thuật toán mạnh mẽ, hỗ trợ việc ra quyết định chính xác và kịp thời. Hệ thống dự báo thời tiết không chỉ hỗ trợ công tác dự phòng thiên tai mà còn góp phần nâng cao chất lượng cuộc sống và hiệu quả trong quản lý xã hội. Nhận thức được tầm quan trọng của đề tài, nhóm chúng em quyết định chọn “Dự báo thời tiết bằng các thuật toán học máy” cho bài tập lớn môn Học máy, nhằm áp dụng kiến thức đã học vào thực tiễn và đáp ứng nhu cầu phân tích, dự báo chính xác trong bối cảnh biến đổi khí hậu ngày càng phức tạp.

Là sinh viên ngành Công nghệ Thông tin, chúng em mong muốn được sử dụng các kiến thức về thuật toán và phân tích dữ liệu để triển khai và thử nghiệm hệ thống dự báo thời tiết hiệu quả. Tuy nhiên, do kiến thức còn hạn chế và thời gian nghiên cứu có hạn, bài làm của nhóm chúng em không tránh khỏi những sai sót. Chúng em rất mong nhận được sự góp ý từ thầy cô để có thể hoàn thiện đề tài tốt nhất.

Chúng em xin chân thành cảm ơn!

# Quan sát dữ liệu, lựa chọn mô hình

## Quan sát dữ liệu.

* Dữ liệu được tham khảo từ Kaggle, có thể truy cập dữ liệu bằng đường dẫn này:

[**https://www.kaggle.com/datasets/ruchikathakur07/wheather**](https://www.kaggle.com/datasets/ruchikathakur07/wheather)

****

* Tập dữ liệu bao gồm các đặc trưng liên quan đến thời tiết:
* Precipitation (lượng mưa). Đơn vị đo: mm.
* Temp\_max (nhiệt độ tối đa). Đơn vị đo: °C.
* Temp\_min (nhiệt độ thấp nhất). Đơn vị đo: °C.
* Wind (sức gió). Đơn vị đo: m/s .
* Weather chia thành 5 lớp: drizzle (mưa phùn), fog (sương mù), rain (mưa), snow (tuyết), và sun (nắng).

A screenshot of a weather report

Description automatically generated

* Thông tin dữ liệu:
* Dữ liệu có tổng cộng 1461 hàng.
* Không có giá trị nào bị null trong tất cả các cột.
* Cột “date” và cột “weather” có kiểu dữ liệu là string.
* Cột “precipitation”, “temp\_max”, “temp\_min”, “wind” có kiểu dữ liệu là float.

A screen shot of a computer

Description automatically generated

* Mục tiêu của bài toán: Dự đoán “weather” dựa trên các đặc trưng còn lại, vì thế cột “date” là không cần thiết nên sẽ được loại bỏ.

## Phân tích dữ liệu.

### Chỉ số thống kê mô tả.

A screenshot of a computer screen

Description automatically generated

* Giá trị trung bình:
  + Precipitation: Lượng mưa trung bình: 3.03 mm.
  + Temp\_max: Nhiệt độ tối đa trung bình: 16.44 °C.
  + Temp\_min: Nhiệt độ tối thiểu trung bình: 8.23 °C
  + Wind: tốc độ gió trung bình: 3.24 m/s.
* Độ lệch chuẩn: Đo lường mức độ phân tán của dữ liệu so với giá trị trung bình.
  + Precipitation: có độ lệch chuẩn cao là 6.88 cho thấy sự biến thiên lớn trong lượng mưa.
  + Temp\_max và temp\_min có độ lệch chuẩn tương đối thấp (7.35 và 5.02), cho thấy nhiệt độ dao động trong một khoảng không quá rộng.
  + Wind có độ lệch chuẩn 1.44, cho thấy tốc độ gió ít thay đổi so với các biến số khác.
* Giá trị nhỏ nhất:
  + Precipitation : Lượng mưa nhỏ nhất là 0 mm (không mưa).
  + Temp\_max : Nhiệt độ tối đa nhỏ nhất là -1.6 °C.
  + Temp\_min : Nhiệt độ tối thiểu thấp nhất là -7.1 °C.
  + Wind : Tốc độ gió nhỏ nhất là 0.4 m/s.
* Phân vi dữ liệu:

|  |  |  |  |
| --- | --- | --- | --- |
| Phân vị | Q1 - 25% | Q2 - 50% | Q3 - 75% |
| Precipitation - Lượng mưa | ≤ 0 mm | 0.8 mm | ≤ 2.8 mm. |
| Temp\_max - Nhiệt độ tối đa | ≤ 10.6 °C | 15.6 °C | ≤ 22.2 °C |
| Temp\_min - Nhiệt độ tối thiểu | ≤ 4.4 °C. | 8.3 °C | ≤ 12.2 °C |
| Wind - Tốc độ gió | ≤ 2.2 m/s. | 3.1 m/s | ≤ 4.0 m/s |

* Giá trị lớn nhất:
  + Precipitation : Lượng mưa lớn nhất là 55.9 mm..
  + Temp\_max : Nhiệt độ tối đa cao nhất là 35.6 °C..
  + Temp\_min : Nhiệt độ tối thiểu cao nhất là 18.3 °C.
  + Wind : Tốc độ gió cao nhất là 9.5 m/s.

### Phân phối dữ liệu.

A group of graphs with numbers

Description automatically generated with medium confidence

* Precipitation (lượng mưa):
  + Có dạng phân phối lệch phải. Đa số các ngày có lượng mưa rất thấp hoặc bằng 0.
  + Phần lớn dữ liệu tập trung ở lượng mưa thấp (0-5 mm). Có rất ít ngày có lượng mưa lớn (vượt quá 20 mm), và lượng mưa lớn rất hiếm gặp (lượng mưa lớn nhất đạt tới 55.9 mm).
* Temp\_max (Nhiệt độ tối đa):
  + Dạng phân phối này gần như là phân phối chuẩn, lệch trái nhẹ.
  + Nhiệt độ tối đa phổ biến nằm trong khoảng 10-25°C. Nhiệt độ thấp (dưới 0°C) và cao (trên 30°C) hiếm gặp.
* Temp\_min (Nhiệt độ tối thiểu):
  + Phân phối này có dạng gần phân phối chuẩn nhưng vẫn có một chút phân tán.
  + Nhiệt độ tối thiểu chủ yếu nằm trong khoảng 0-15°C. Có một số giá trị thấp hơn -5°C.
* Wind (Tốc độ gió):
  + Phân phối này có dạng phân phối chuẩn lệch phải nhẹ.
  + Phần lớn tốc độ gió nằm trong khoảng 1-5 m/s, một số ngày có gió mạnh hơn (trên 6 m/s).

### Phân tích giá trị ngoại lệ.

A comparison of a graph

Description automatically generated with medium confidence

* Precipitation (lượng mưa):
  + Có rất nhiều giá trị ngoại lai (outliers) ở phía trên, với một số giá trị vượt quá 40mm.
  + Phần lớn lượng mưa dường như khá thấp (dưới 10mm), phản ánh qua sự phân bố tập trung gần 0, trong khi các giá trị ngoại lai nằm rất xa giá trị trung bình.
* Temp\_max (Nhiệt độ tối đa):
  + Nhiệt độ cao nhất dao động trong khoảng từ -1.6°C đến 35.6°C, và không có ngoại lai.
  + Biểu đồ khá cân đối, với giá trị trung vị (median) nằm gần giữa hộp, cho thấy nhiệt độ cao nhất có phân phối tương đối đồng đều.
  + Phần lớn nhiệt độ cao nhất nằm trong khoảng từ khoảng 10°C đến 22°C (khoảng từ phần tư thứ nhất đến phần tư thứ ba).
* Temp\_min (Nhiệt độ tối thiểu):
  + Tương tự như temp\_max, biểu đồ không có ngoại lai.
  + Nhiệt độ thấp nhất dao động từ -7.1°C đến 18.3°C, với giá trị trung vị khoảng 8°C.
  + Phân phối nhiệt độ thấp nhất cũng khá đều, phần lớn giá trị nằm trong khoảng từ khoảng 4°C đến 12°C.
* Wind (Tốc độ gió):
  + Biểu đồ cho thấy có một số giá trị ngoại lai (outliers) với tốc độ gió vượt quá 7m/s.
  + Phần lớn giá trị của tốc độ gió nằm trong khoảng từ 2m/s đến 5m/s, và giá trị trung vị khoảng 3m/s.
  + Tốc độ gió trong các ngày phần lớn ở mức thấp đến trung bình, nhưng cũng có một số ngày có tốc độ gió khá cao.

### Phân tích ma trận tương quan.

A screenshot of a screen

Description automatically generated

* Precipitation và Temp\_max: Hệ số tương quan là -0.23, khi lượng mưa tăng, nhiệt độ cao nhất có xu hướng giảm nhẹ, nhưng mối quan hệ này không rõ ràng hoặc chặt chẽ vì hệ số tương quan không quá mạnh.
* Temp\_max và Temp\_min: Hệ số tương quan là 0.88, khi nhiệt độ cao nhất tăng, nhiệt độ thấp nhất cũng tăng theo. Mối quan hệ rất rõ ràng và mạnh, cho thấy sự đồng biến giữa nhiệt độ cao nhất và thấp nhất trong cùng một ngày.
* Wind và Temp\_max: Hệ số tương quan là -0.16, khi tốc độ gió tăng, nhiệt độ cao nhất có xu hướng giảm nhẹ, nhưng tương tác giữa hai yếu tố này không mạnh mẽ.
* Precipitation và Wind: Hệ số tương quan là 0.33, khi lượng mưa tăng, tốc độ gió có xu hướng tăng, nhưng mối tương quan này vẫn còn yếu. Điều này có thể giải thích rằng trong một số trường hợp, mưa lớn có thể đi kèm với gió mạnh hơn, nhưng không phải lúc nào cũng vậy.

## Nhận xét dữ liệu.

* **Precipitation** có phân phối lệch phải và nhiều giá trị ngoại lai, nhưng những ngoại lệ này là đặc trưng tự nhiên của dữ liệu thời tiết, đặc biệt là lượng mưa, vốn có xu hướng biến động mạnh theo mùa và điều kiện thời tiết cụ thể.
* **Wind** có một số giá trị ngoại lai, nhưng các giá trị này không quá xa so với phần còn lại của dữ liệu và không có sự chênh lệch quá lớn trong tốc độ gió.
* **Temp\_max** và **Temp\_min** có phân phối gần chuẩn, không có giá trị ngoại lai lớn và mức độ phân tán dữ liệu không quá lớn.
* Các đặc trưng trong tập dữ liệu này đã có sự phân bố tương đối phù hợp với bối cảnh phân tích thời tiết. Mặc dù có một số giá trị ngoại lai trong lượng mưa và tốc độ gió nhưng chúng không gây ảnh hưởng đáng kể đến hiệu quả của các mô hình phân tích hoặc dự đoán.
* VÌ vậy dữ liệu đã khá tốt để huấn luyện các mô hình nên chỉ cần loại bỏ cột dư thừa là cột “date” và có thể trực tiếp đưa vào huấn luyện.

## Lựa chọn mô hình.

Dữ liệu phù hợp với các mô hình Perceptron, ID3 và Neural Network:

* Perceptron: có thể học tốt từ các đặc trưng có phân phối chuẩn và gần chuẩn, nhờ vào sự phù hợp về mặt phân phối. Khả năng học nhanh của perceptron cũng là một yếu tố quan trọng trong việc phát hiện các mẫu và đưa ra dự đoán kịp thời cho các biến động thời tiết. Tuy nhiên vẫn có 2 đặc trưng chứa các giá trị ngoại lai dẫn đến kết quả dự đoán chính xác của mô hình này thấp hơn so với 2 mô hình còn lại.
* ID3: với khả năng xử lý các giá trị ngoại lai và tạo ra các quyết định phân nhánh rõ ràng, rất phù hợp để phân loại các tình huống thời tiết khác nhau dựa trên lượng mưa. Điều này giúp người dùng dễ dàng hiểu được các quyết định được đưa ra từ mô hình.
* Neural Network: có khả năng học từ những tương tác phức tạp giữa các đặc trưng như lượng mưa, nhiệt độ và tốc độ gió, giúp nâng cao độ chính xác trong dự đoán.

# Đánh giá mô hình.

## Các mô hình được sử dụng.

### Perceptron.

#### Lý thuyết.

* Khái niệm: Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất: chỉ có hai class (lớp) (bài toán với chỉ hai class được gọi là binary classification) và cũng chỉ hoạt động được trong một trường hợp rất cụ thể.
* Cách thực hiện:

B1: Chọn ngẫu nhiên một vector hệ số và bias ­­ (thường bằng 0 hoặc giá trị nhỏ ngẫu nhiên)

B2: Duyệt ngẫu nhiên từng , tính được

+Nếu được phân lớp đúng, tức là , chúng ta không cần làm gì

+Nếu được phân lớp lỗi, cập nhật và theo công thức:

hay

B3: Kiểm tra xem có bao nhiêu điểm bị phân lớp lỗi. Nếu không còn điểm nào, dừng thuật toán. Nếu còn, quay lại bước 2.

#### Input và Output.

* **Input**
* Mô tả chi tiết về dữ liệu:
* Dữ liệu đầu vào được đọc từ tệp seattle-weather.csv. Đây là bộ dữ liệu chứa thông tin về thời tiết ở Seattle. Dữ liệu ban đầu bao gồm các cột như: date (ngày tháng), precipitation (lượng mưa), temp\_max (nhiệt độ tối đa), temp\_min (nhiệt độ tối thiểu), wind (tốc độ gió), và weather (loại thời tiết).
* Các cột được chọn làm đầu vào của mô hình bao gồm:
* precipitation (lượng mưa), temp\_max (nhiệt độ tối đa), temp\_min (nhiệt độ tối thiểu), wind: (tốc độ gió).
* **Output.**
* Kết quả dự đoán là nhãn chữ tương ứng với loại thời tiết trong ngày:
* sun (nắng), rain (mưa), drizzle (mưa phùn), snow (tuyết), fog (sương mù).

#### Giải thích code.

* **Thư viện cần thiết:**

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.linear\_model import Perceptron

from sklearn.metrics import (ConfusionMatrixDisplay, accuracy\_score,

classification\_report, confusion\_matrix, hinge\_loss)

from sklearn.model\_selection import GridSearchCV, StratifiedKFold, learning\_curve, train\_test\_split

from sklearn.preprocessing import StandardScaler

* **matplotlib.pyplot (plt)**: Thư viện vẽ đồ thị và biểu đồ, hỗ trợ trực quan hóa kết quả của mô hình và dữ liệu.
* **numpy (np)**: Thư viện toán học và xử lý mảng, giúp thao tác với các cấu trúc dữ liệu dạng mảng và thực hiện các phép toán.
* **pandas (pd)**: Thư viện xử lý dữ liệu dạng bảng, thường dùng để đọc dữ liệu từ file CSV và thực hiện các thao tác trên DataFrame.
* **sklearn.linear\_model - Perceptron**: Mô hình phân loại tuyến tính đơn giản, thường dùng cho dữ liệu có thể phân tách tuyến tính.
* **sklearn.metrics - ConfusionMatrixDisplay**: Công cụ hiển thị ma trận nhầm lẫn, giúp quan sát kết quả phân loại của mô hình.
* **sklearn.metrics - accuracy\_score**: Hàm tính độ chính xác của mô hình dựa trên tỷ lệ các mẫu phân loại đúng.
* **sklearn.metrics - classification\_report**: Tạo báo cáo chi tiết gồm các chỉ số như độ chính xác (precision), độ nhạy (recall), và F1-score cho từng lớp.
* **sklearn.metrics - confusion\_matrix**: Tạo ma trận nhầm lẫn, hiển thị số lượng phân loại đúng và sai của từng lớp.
* **sklearn.metrics - hinge\_loss**: Hàm tính toán sai số hinge, đánh giá khoảng cách từ các điểm dữ liệu đến biên phân cách trong bộ phân loại tuyến tính.
* **sklearn.model\_selection - GridSearchCV**: Phương pháp tìm tham số tối ưu cho mô hình bằng cách thử nhiều tổ hợp tham số và đánh giá hiệu suất.
* **sklearn.model\_selection - StratifiedKFold**: Phương pháp chia dữ liệu kiểm tra chéo, đảm bảo tỷ lệ lớp trong mỗi phần chia, hữu ích cho dữ liệu mất cân đối.
* **sklearn.model\_selection - learning\_curve**: Hàm tạo biểu đồ learning curve, giúp đánh giá hiệu suất của mô hình theo kích thước tập huấn luyện.
* **sklearn.model\_selection - train\_test\_split**: Chia dữ liệu thành tập huấn luyện và tập kiểm tra, giúp đánh giá hiệu suất mô hình.
* **sklearn.preprocessing - StandardScaler**: Chuẩn hóa dữ liệu đầu vào về cùng một thang đo, giúp giảm chênh lệch giữa các đặc trưng và tăng hiệu quả huấn luyện.
* **Xử lý dữ liệu đầu vào:**

data = pd.read\_csv('./seattle-weather.csv')

* Đọc tệp dữ liệu đầu vào

data = weather\_data.dropna()

* Loại bỏ các dòng dữ liệu thiếu, giữ lại các dòng đầy đủ thông tin.

data.drop(['date'], axis=1, inplace=True)

* Loại bỏ cột "date" vì không có giá trị dự đoán.

le = LabelEncoder()

* Tạo bộ mã hóa để chuyển giá trị chuỗi (text) thành số nguyên.

data['weather\_encoded'] = le.fit\_transform(data['weather'])

* Mã hóa cột "weather" thành cột số "weather\_encoded".

X = data[["precipitation", "temp\_max", "temp\_min", "wind"]]

* Ở bước này, các cột đặc trưng đầu vào (independent variables) được chọn ra để làm tập dữ liệu cho việc huấn luyện mô hình. Các cột được chọn bao gồm:
* precipitation: Lượng mưa (mm).
* temp\_max: Nhiệt độ cao nhất trong ngày (°C).
* temp\_min: Nhiệt độ thấp nhất trong ngày (°C).
* wind: Tốc độ gió (m/s). Đây là những đặc trưng có thể ảnh hưởng đến tình trạng thời tiết trong ngày và sẽ được dùng làm input để dự đoán.

y = data['weather\_encoded']

* Cột weather\_encoded chứa các giá trị đã được mã hóa của cột weather được chọn làm biến mục tiêu (dependent variable) để dự đoán.

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, train\_size=0.7, test\_size=0.3, random\_state=42)

* train\_test\_split được sử dụng để chia dữ liệu thành tập huấn luyện (70%) và tập tạm thời (30%). random\_state=42 giúp chia dữ liệu ngẫu nhiên nhưng có thể tái lập kết quả.

X\_valid, X\_test, y\_valid, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42)

* Tập tạm thời (30%) được chia thành hai phần: tập xác thực (15%) và tập kiểm tra (15%). Tập xác thực dùng để điều chỉnh mô hình trong quá trình huấn luyện, còn tập kiểm tra dùng để đánh giá hiệu suất mô hình sau khi hoàn thành huấn luyện.

scaler = StandardScaler()

* StandardScaler được khởi tạo để chuẩn hóa các đặc trưng. Điều này giúp các đặc trưng như nhiệt độ, lượng mưa và tốc độ gió có cùng thang đo, giúp mô hình học tốt hơn và tránh việc các đặc trưng với giá trị lớn chi phối kết quả.

X\_train = scaler.fit\_transform(X\_train)

* Chuẩn hóa tập huấn luyện bằng cách tính toán trung bình và độ lệch chuẩn của các đặc trưng trong tập huấn luyện, sau đó áp dụng phép chuẩn hóa lên tập này. Sau khi chuẩn hóa, các giá trị sẽ có trung bình bằng 0 và độ lệch chuẩn bằng 1.

X\_valid = scaler.transform(X\_valid)

X\_test = scaler.transform(X\_test)

* Áp dụng phép chuẩn hóa tương tự đã tính toán từ tập huấn luyện lên tập xác thực và tập kiểm tra, giúp đảm bảo rằng các đặc trưng trong các tập dữ liệu này cũng nằm trên cùng thang đo với tập huấn luyện.
* **Xây dựng mô hình:**

best\_model = Perceptron(max\_iter=1000, eta0=0.01, penalty='l2', alpha=0, tol=1e-4, random\_state=42)

* Khởi tạo mô hình với các tham số tốt nhất tìm được từ Cross Validation:
* **best\_model**: Đây là tên biến được sử dụng để lưu mô hình Perceptron.
* **Perceptron()**: Đây là lớp mô hình hồi quy phân loại (classification model) trong Scikit-learn, dựa trên thuật toán Perceptron, một thuật toán học máy cổ điển cho bài toán phân loại nhị phân.
* **Tham số cụ thể**:
  + **max\_iter=1000**: Số lần tối đa để mô hình huấn luyện. Ở đây, mô hình sẽ chạy tối đa 1000 vòng lặp.
  + **eta0=0.01**: Tốc độ học (learning rate). Đây là tốc độ mà tại đó mô hình điều chỉnh các trọng số trong quá trình huấn luyện.
  + **penalty='l2'**: Đây là loại hình phạt (regularization) áp dụng cho mô hình. L2 penalty hay Ridge Regularization giúp tránh overfitting bằng cách phạt các trọng số lớn.
  + **alpha=0**: Điều này thể hiện rằng mức độ phạt (regularization strength) được thiết lập bằng 0, tức là không áp dụng mức phạt trực tiếp.
  + **tol=1e-4**: Ngưỡng dừng cho quá trình tối ưu hóa. Nếu sự thay đổi trong lỗi huấn luyện nhỏ hơn 1e-4, thì quá trình huấn luyện dừng lại.
  + **random\_state=42**: Giúp kết quả của quá trình huấn luyện có thể lặp lại được (deterministic). Mọi lần chạy với cùng dữ liệu và cùng random\_state sẽ cho kết quả giống nhau.

best\_model.fit(X\_train, y\_train)

* Huấn luyện mô hình tốt nhất trên toàn bộ tập huấn luyện

#### Biểu đồ và các tham số mô hình.

##### Giải thích tham số thiết lập mô hình.

* Giải thích khởi tạo các tham số với Cross Validation.

perceptron\_model = Perceptron(max\_iter=1000, random\_state=42)

* Khởi tạo mô hình Perceptron với số vòng lặp tối đa (max\_iter=1000) để mô hình có đủ thời gian hội tụ trong quá trình huấn luyện.

param\_grid = {

    'max\_iter': [1000, 2000, 3000],

    'tol': [1e-4, 1e-5],

    'eta0': [0.1, 0.01],

    'penalty': ['l2', 'elasticnet'],

    'alpha': [0.01, 0.1, 1,0]

}

* param\_grid chứa các giá trị để GridSearchCV có thể thử nghiệm nhằm tìm ra bộ siêu tham số tốt nhất cho mô hình Perceptron. Các tham số bao gồm:
* **max\_iter**: Số vòng lặp tối đa.
* **tol**: Ngưỡng dừng lại cho sự hội tụ.
* **eta0**: Tốc độ học ban đầu.
* **penalty**: Loại regularization để tránh overfitting.
* **alpha**: Hệ số điều chỉnh cho regularization.

grid\_search = GridSearchCV(estimator=perceptron\_model, param\_grid = param\_grid, cv=kf, scoring='accuracy', n\_jobs=-1)

* Khởi tạo GridSearchCV để thử nghiệm các tổ hợp tham số khác nhau từ param\_grid với phương pháp kiểm tra chéo kf. Phép đánh giá được sử dụng là độ chính xác (accuracy), và quá trình tìm kiếm tham số chạy song song trên tất cả các nhân xử lý (n\_jobs=-1).

grid\_search.fit(X\_train, y\_train)

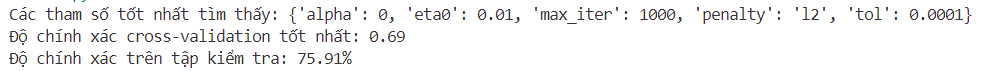
* Thực hiện huấn luyện với GridSearchCV trên tập huấn luyện và tìm kiếm bộ tham số tốt nhất từ param\_grid.

best\_model = grid\_search.best\_estimator\_

* Lưu lại mô hình với bộ tham số tốt nhất tìm được từ GridSearchCV.

best\_model.fit(X\_train, y\_train)

* Huấn luyện mô hình tốt nhất (best\_model) trên toàn bộ tập huấn luyện.
* **Kết quả:**

****

##### Đánh giá mô hình và tham số.

* **Confusion Matrix (Ma trận nhầm lẫn).**

A blue and white graph

Description automatically generated

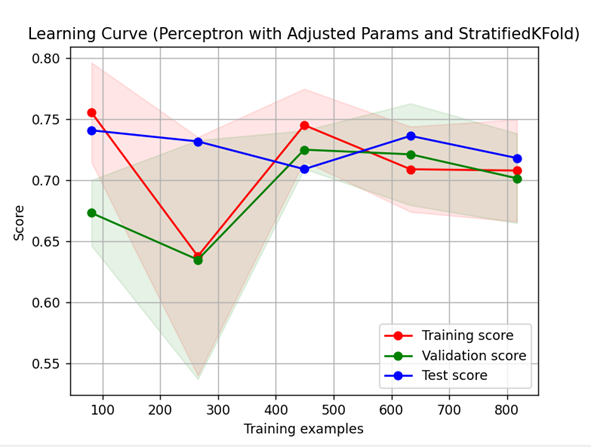
**Nhận xét:**

* **Lớp Drizzle**: Mô hình dự đoán tất cả 7 mẫu của lớp drizzle thành sun. Điều này cho thấy rằng mô hình không thể phân biệt rõ ràng giữa drizzle (mưa phùn) và sun (trời nắng). Một phần lý do có thể là do sự tương đồng giữa các đặc trưng thời tiết giữa hai lớp hoặc do mô hình không được cung cấp đủ dữ liệu từ lớp drizzle trong quá trình huấn luyện.
* **Lớp Fog**: Tương tự như lớp drizzle, tất cả 17 mẫu của lớp fog (sương mù) bị mô hình dự đoán thành sun. Điều này cho thấy mô hình không học đủ đặc trưng để phân biệt lớp fog. Lớp này có thể có quá ít dữ liệu hoặc các đặc trưng không đủ mạnh để mô hình có thể nhận biết và phân loại chính xác.
* **Lớp Rain và Sun**: Mô hình hoạt động rất tốt đối với hai lớp rain (mưa) và sun (trời nắng). Với 79/99 mẫu của lớp rain và 88/109 mẫu của lớp sun được dự đoán đúng. Tuy nhiên, một số mẫu của lớp rain bị nhầm lẫn với lớp sun (18 mẫu), điều này có thể là do các đặc trưng thời tiết ở các trường hợp đó khá giống nhau và gây nhầm lẫn.
* **Lớp Snow**: Mô hình chỉ dự đoán đúng 1/5 mẫu của lớp snow, còn lại 4 mẫu bị nhầm lẫn thành lớp rain. Điều này có thể xuất phát từ sự thiếu cân bằng trong tập dữ liệu, khi lớp snow có quá ít dữ liệu khiến mô hình không đủ dữ liệu để học và phân biệt lớp này.

**Kết luận**:

* **Lỗi nhầm lẫn giữa các lớp nhỏ**: Mô hình gặp khó khăn trong việc phân loại các lớp có ít dữ liệu như drizzle, fog và snow. Điều này có thể do mô hình chưa có đủ dữ liệu để học các đặc trưng của các lớp này.
* **Hiệu suất tốt trên các lớp phổ biến**: Mô hình hoạt động rất tốt với các lớp có số lượng mẫu lớn như rain và sun, cho thấy rằng với nhiều dữ liệu, mô hình có khả năng học và phân loại chính xác.
* **Mất cân bằng dữ liệu**: Các lớp có số lượng mẫu nhỏ (drizzle, fog) bị dự đoán sai hoàn toàn, điều này nhấn mạnh rằng việc xử lý dữ liệu mất cân bằng là cần thiết. Giải pháp như **SMOTE** hoặc tăng cường dữ liệu có thể được xem xét để cải thiện hiệu suất của mô hình.

* **Learning Curve (Biểu đồ đường học).**



**Nhận xét:**

* **Training Score (Điểm số trên tập huấn luyện)**:
* Đường đỏ (training score) thể hiện điểm số của mô hình trên tập huấn luyện. Khi số lượng mẫu huấn luyện nhỏ (khoảng 100 đến 200 mẫu), điểm số trên tập huấn luyện cao hơn so với tập xác thực và tập kiểm tra. Điều này cho thấy rằng mô hình đang gặp tình trạng **overfitting** khi học quá kỹ từ tập huấn luyện nhưng không tổng quát tốt trên dữ liệu chưa thấy.
* **Validation Score (Điểm số trên tập xác thực)**:
* Đường xanh (validation score) cho thấy hiệu suất tổng quát của mô hình. Khi số lượng mẫu tăng lên (400 mẫu trở lên), điểm số trên tập xác thực tăng lên và tiệm cận với điểm số của tập huấn luyện. Điều này cho thấy rằng khi có nhiều dữ liệu hơn, mô hình học tốt hơn và khả năng tổng quát hóa trên tập dữ liệu mới cũng cải thiện.
* **Test Score (Điểm số trên tập kiểm tra)**:
* Đường xanh dương (test score) biểu thị điểm số trên tập kiểm tra và khá ổn định quanh mức 0.7. Điều này chỉ ra rằng mô hình đã học tốt và không có sự biến động lớn khi kiểm tra trên dữ liệu chưa thấy trước.
* **Khoảng bóng mờ (Standard Deviation)**:
* Các vùng bóng mờ đại diện cho độ lệch chuẩn của điểm số tại mỗi kích thước tập huấn luyện. Khi số lượng mẫu nhỏ (100, 200 mẫu), khoảng bóng mờ lớn, điều này cho thấy mô hình hoạt động không ổn định. Tuy nhiên, khi số lượng mẫu tăng lên, vùng bóng mờ nhỏ dần, cho thấy mô hình trở nên ổn định hơn.

**Kết luận**:

* **Hiện tượng underfitting/overfitting**: Khi quan sát các đường cong học tập, khoảng cách giữa đường huấn luyện và đường xác thực là rất nhỏ khi kích thước tập huấn luyện tăng lên. Điều này cho thấy mô hình đã không gặp phải hiện tượng overfitting hay underfitting khi số lượng mẫu huấn luyện đủ lớn.
* **Sự hội tụ của các đường cong**: Khi tập huấn luyện tăng lên, cả đường huấn luyện và đường xác thực đều tiệm cận nhau, cho thấy hiệu suất của mô hình cải thiện khi dữ liệu tăng lên.
* **Dao động khi có ít mẫu**: Ở các kích thước nhỏ (100-200 mẫu), mô hình có sự biến động lớn, cho thấy sự không ổn định và khó khăn trong việc học đặc trưng từ dữ liệu ít. Điều này có thể được cải thiện bằng cách sử dụng kỹ thuật dữ liệu nhiều hơn hoặc áp dụng các phương pháp xử lý dữ liệu mất cân bằng.
* **Hinge Loss (Hàm mất mát qua các Epochs).**

A graph with blue lines

Description automatically generated

**Nhận xét:**

* **Phân tích biểu đồ Hinge Loss**:
  + Biểu đồ này thể hiện quá trình tối ưu hóa của mô hình qua các epoch. Hinge loss dao động từ mức 1.2 đến 2.2 và có nhiều dao động lớn trong quá trình huấn luyện.
* **Sự giảm dần của Loss**:
  + Mặc dù có sự dao động, ta vẫn thấy xu hướng giảm dần của loss sau một số epoch, điều này chỉ ra rằng mô hình đang dần cải thiện và hội tụ theo thời gian.

**Kết luận:**

* **Sự dao động lớn**: Sự dao động lớn của hàm mất mát qua các epoch cho thấy rằng mô hình gặp khó khăn trong việc đạt được hội tụ ổn định. Điều này có thể giải thích lý do vì sao mô hình không thể phân loại chính xác các lớp nhỏ như drizzle hay fog.
* **Giảm nhẹ**: Mặc dù có dao động, mô hình vẫn cho thấy sự tiến triển trong quá trình học khi loss có xu hướng giảm qua các epoch. Tuy nhiên, cần điều chỉnh thêm để giảm bớt sự dao động và cải thiện quá trình hội tụ.
* **Classification report.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a report

Description automatically generated

**Nhận xét:**

* **Hiệu suất tốt đối với các lớp phổ biến:** Mô hình hoạt động tốt trên các lớp có nhiều dữ liệu như **rain** (mưa) và **sun** (trời nắng), với F1-score cao (từ 0.79 đến 0.87). Điều này cho thấy mô hình có thể nhận diện tốt các lớp có số lượng mẫu lớn và các đặc trưng rõ ràng.
* **Vấn đề với các lớp có ít dữ liệu:** Các lớp có ít mẫu như **drizzle** (mưa phùn), **fog** (sương mù), và **snow** (tuyết) có precision, recall, và F1-score đều bằng 0. Điều này chứng tỏ mô hình không thể học được hoặc dự đoán chính xác các lớp này do sự thiếu hụt dữ liệu hoặc do các đặc trưng không đủ mạnh để phân biệt giữa các lớp này.
* **Sự mất cân bằng dữ liệu: Macro avg** luôn thấp (dao động từ 0.32 đến 0.36) cho thấy mô hình không thể dự đoán tốt đối với tất cả các lớp, đặc biệt là các lớp có ít mẫu. Ngược lại, **weighted avg** cao hơn, cho thấy mô hình tập trung dự đoán đúng các lớp có nhiều dữ liệu, điều này giúp cải thiện độ chính xác tổng thể.
* **Độ chính xác: Accuracy** trên tập huấn luyện, xác thực, và kiểm tra dao động từ 75% đến 79%, chứng tỏ mô hình có khả năng tổng quát hóa tốt, tuy nhiên, kết quả này chủ yếu đến từ việc dự đoán chính xác các lớp lớn hơn.

**Kết luận:**

* Dữ liệu về thời tiết Seattle là **phi tuyến tính**, vì vậy mô hình **Perceptron**, vốn là mô hình tuyến tính, gặp nhiều khó khăn trong việc nắm bắt các mối quan hệ phức tạp giữa các đặc trưng đầu vào và nhãn dự đoán.

### ID3.

#### Lý thuyết.

* Khái niệm: ID3 là một thuật toán phân loại theo cách tiếp cận tham lam bằng cách chọn thuộc tính tốt nhất nhằm mang lại Information Gain (lợi ích của thông tin) tối đa hoặc Entropy tối thiểu.
* Cách thực hiện:

Sử dụng hàm “entropy“ để đo độ tinh khiết (purity) / độ vẩn đục (impurity) của 1 phép phân chia

B1: Tại root node (nút gốc), tính entropy để đo lường mức độ hỗn loạn của toàn bộ tập dữ liệu trước khi nó được phân chia theo bất kỳ thuộc tính nào.

B2: Khi đã phân chia tập dữ liệu 𝑆 dựa trên một thuộc tính x, chúng ta sẽ có K tập con (child nodes), mỗi tập tương ứng với một giá trị cụ thể của thuộc tính x. Ta tính entropy tại child node:

B3: Information Gain(IG) là lượng thông tin thu được sau khi phân chia tập dữ liệu dựa trên một thuộc tính x, tức là nó đo lường sự giảm entropy khi chia dữ liệu theo x. Công thức tính IG dựa trên x là:

Và tại mỗi node thuộc tính được chọn được xác định dựa vào:

#### Input và Output.

* **Input**
  + Input của mô hình là tập dữ liệu seattle-weather bao gồm các đặc trưng của từng ngày về thời tiết, trong đó mỗi đặc trưng là một cột trong bảng dữ liệu. Các đặc trưng này bao gồm:
    - precipitation (lượng mưa), temp\_max (nhiệt độ cao nhất), temp\_min (nhiệt độ thấp nhất), wind (tốc độ gió).
  + Xử lý dữ liệu:
  + Cột “date” được loại bỏ vì không dùng tới trong quá trình train model.
  + Mô hình ID3 sử dụng entropy để xây dựng cây quyết định và chỉ quan tâm đến mối quan hệ giữa các giá trị của một đặc trưng và nhãn kết quả (lớp phân loại) để tạo ra các nút phân chia. Vì vậy đối với tập dữ liệu trên thì mô hình ID3 không cần chuẩn hóa dữ liệu.
    - Chia tập dữ liệu thành 3 phần:
* 70% dữ liệu cho training(X\_train, y\_train).
* 15% dữ liệu cho validation (X\_valid, y\_valid).
* 15% dữ liệu cho test (X\_test, y\_test).
* **Output.**
* Đầu ra của là các nhãn của cột dự đoán (weather) dựa trên dữ liệu đầu vào. Giá trị trả về là 1 trong các giá trị:
  + sun (nắng), rain (mưa), drizzle (mưa phùn), snow (tuyết), fog (sương mù).
* Sau khi huấn luyện mô hình ID3 với tập X\_train và y\_train kết quả là một mô hình cây quyết định được lưu vào decison\_tree.pkl để tái sử dụng và phục vụ cho việc tạo ứng dụng demo.

#### Giải thích code.

* **Thư viện cần thiết:**

import joblib

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

* **joblib**: Dùng để lưu và tải mô hình đã huấn luyện để có thể sử dụng lại sau này.
* **matplotlib.pyplot as plt** và **numpy**: Các thư viện phổ biến trong Python, ở đây plt và np không được dùng trong đoạn code này, nhưng thường được sử dụng để trực quan hóa dữ liệu và xử lý mảng số học.
* **pandas**: Được dùng để đọc và xử lý dữ liệu dưới dạng bảng, như dữ liệu từ tệp CSV.
* **train\_test\_split** từ **sklearn.model\_selection**: Dùng để chia dữ liệu thành tập huấn luyện, tập kiểm tra, và tập xác thực.
* **DecisionTreeClassifier** từ **sklearn.tree**: Thuật toán cây quyết định, được dùng để huấn luyện mô hình phân loại.
* **Xử lý dữ liệu đầu vào:**

data = pd.read\_csv("./seattle-weather.csv")

* Đọc tệp dữ liệu: Đọc tệp seattle-weather.csv bằng pandas và lưu vào biến data

data = data.dropna()

* Loại bỏ bất kỳ hàng nào trong data có chứa giá trị null để tránh lỗi khi huấn luyện mô hình.

data.drop(['date'], axis=1, inplace=True)

* Bỏ cột date vì ngày không ảnh hưởng đến dự đoán thời tiết.
* Chia dữ liệu thành đặc trưng và nhãn mục tiêu:

X = data[["precipitation", "temp\_max", "temp\_min", "wind"]]

* + Biến X lưu  lưu trữ các cột đặc trưng được chọn để huấn luyện mô hình (precipitation, temp\_max, temp\_min, và wind).\

y = data["weather"]

* + Biến y lưu trữ cột nhãn mục tiêu (weather) mà mô hình sẽ dự đoán.
* Chia dữ liệu thành các tập huấn luyện, xác thực và kiểm tra:

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, train\_size=0.7, test\_size=0.3, random\_state=42)

* + Chia dữ liệu thành 2 tập:
  + X\_train, y\_train: 70% dữ liệu dùng để huấn luyện mô hình.
  + X\_temp, y\_temp: 30% dữ liệu còn lại để chia tiếp thành tập xác thực và kiểm tra.

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, train\_size=0.7, test\_size=0.3, random\_state=42)

* + Chia X\_temp và y\_temp thành hai tập:
    - X\_valid, y\_valid: 15% dữ liệu dùng để xác thực mô hình.
    - X\_test, y\_test: 15% dữ liệu dùng để kiểm tra mô hình.
  + random\_state = 42: Đảm bảo tính ngẫu nhiên được cố định để kết quả phân chia nhất quán mỗi khi chạy.
  + train\_size=0.7: Chỉ định rằng 70% dữ liệu được sử dụng làm tập huấn luyện (X\_train, y\_train).
  + test\_size=0.3: Chỉ định rằng 30% dữ liệu sẽ được tạm thời chia vào X\_temp, y\_temp.
* **Xây dựng mô hình:**
* Khởi tạo và thiết lập mô hình cây quyết định ID3:

dt\_model = DecisionTreeClassifier(criterion='entropy', random\_state=42,

                                  class\_weight = None,

                                  max\_depth = 3,

                                  max\_features = 'sqrt',

                                  max\_leaf\_nodes = None,

                                  min\_samples\_leaf = 1,

                                  min\_samples\_split = 2,

                                  min\_weight\_fraction\_leaf = 0.0,

                                  splitter = 'best')

* **criterion='entropy'**: Sử dụng Entropy để xác định thuộc tính nào tốt nhất để phân chia tại mỗi nút của cây
* **random\_state=42**: Đảm bảo tính ngẫu nhiên được cố định để có kết quả nhất quán khi chạy lại.
* **class\_weight=None**: Không áp dụng trọng số cho các lớp, mặc định các lớp có trọng số bằng nhau.
* **max\_depth=3**: Giới hạn độ sâu tối đa của cây là 3 để tránh quá khớp.
* **max\_features='sqrt'**: Số lượng đặc trưng tối đa để xem xét tại mỗi nút là căn bậc hai của tổng số đặc trưng.
* **max\_leaf\_nodes=None**: Không giới hạn số lượng nút lá.
* **min\_samples\_leaf=1**: Số lượng mẫu tối thiểu trong một nút lá là 1.
* **min\_samples\_split=2**: Số lượng mẫu tối thiểu cần có để chia nút là 2.
* **min\_weight\_fraction\_leaf=0.0**: Không giới hạn tỷ lệ trọng số của các mẫu trong một nút lá.
* **splitter='best'**: Chọn thuộc tính tốt nhất để chia tại mỗi nút.
* Huấn luyện mô hình trên tập dữ liệu huấn luyện

dt\_model.fit(X\_train, y\_train)

* Huấn luyện mô hình với tập dữ liệu huấn luyện X\_train và y\_train. Mô hình sẽ tìm hiểu quy tắc từ dữ liệu đầu vào để dự đoán nhãn weather
* Lưu mô hình đã huấn luyện:

valueSend = {

    'model': dt\_model

}

joblib.dump(valueSend, 'decision\_tree.pkl')

* valueSend = {'model': dt\_model}: Đưa mô hình đã huấn luyện vào dictionary để lưu trữ.
* joblib.dump(valueSend, 'decision\_tree.pkl'): Lưu valueSend vào tệp decision\_tree.pkl, cho phép mô hình đã huấn luyện được sử dụng lại và cùng để phục vụ cho việc tạo sản phẩm demo.

#### Biểu đồ và các tham số mô hình.

##### Giải thích tham số thiết lập mô hình.

* Sử dụng cross-validation để tìm ra tham số tối ưu cho mô hình

param\_grid = {

    'max\_depth': [1, 2, 3, 5, 7, 9, None],

    'min\_samples\_split': [2, 3, 5, 10],

    'min\_samples\_leaf': [1, 2, 3, 4, 5],

    'max\_features': [None, 'sqrt', 'log2'],

    'class\_weight': [None, 'balanced'],

    'splitter': ['best', 'random'],

    'max\_leaf\_nodes': [None, 10, 20, 30, 40, 50],

    'min\_weight\_fraction\_leaf': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]

}

* **param\_grid:** Định nghĩa một từ điển chứa các tham số khác nhau cho cây quyết định để thực hiện tìm kiếm lưới (Grid Search) nhằm tìm ra các siêu tham số tốt nhất.
* **max\_depth***:* Độ sâu tối đa của cây quyết định. Tham số giúp giới hạn số lượng cấp độ cây có thể phát triển.
* **min\_samples\_split***:* Số mẫu tối thiểu cần để một nút có thể được chia ra thành hai nút con.
* **min\_samples\_leaf***:* Số mẫu tối thiểu cần có tại một nút lá.
* **max\_features***:* Số đặc trưng tối đa được xem xét khi chia một nút. (None là tất cả các đặc trưng; 'sqrt' là căn bậc hai của số lượng đặc trưng; 'log2' là logarit cơ số 2 của số lượng đặc trưng)
* **class\_weight***:* Khối lượng của các lớp trong dữ liệu. (None là không có khối lượng; 'balanced' tự động tính toán khối lượng dựa trên số lượng mẫu).
* **splitter***:* Phương pháp để chia các nút trong cây. 'best' chọn phương pháp chia tốt nhất, trong khi 'random' chọn chia ngẫu nhiên.
* **max\_leaf\_nodes***:* Giới hạn số lượng nút lá tối đa trong cây quyết định.
* **min\_weight\_fraction\_leaf***:* Phần trăm trọng số tối thiểu của các mẫu cần có tại một nút lá.

dt\_model = DecisionTreeClassifier(criterion='entropy', random\_state=42)

* Khởi tạo mô hình cây quyết định với tham số criterion = ‘entropy’

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

* Khởi tạo K-Fold cross-validation với n\_splits=5 nghĩa là dữ liệu sẽ được chia thành 5 phần và dữ liệu được Xáo trộn trước khi chia để đảm bảo tính ngẫu nhiên.

grid\_search = GridSearchCV(estimator=dt\_model, param\_grid=param\_grid, cv=kf, scoring='accuracy', n\_jobs=-1)

* Sử dụng để tìm các siêu tham số tốt nhất cho mô hình bằng cách thực hiện tìm kiếm lưới.
  + estimator=dt\_model: Mô hình cây quyết định.
  + param\_grid=param\_grid: Tham số sẽ được tìm kiếm.
  + cv=kf: Sử dụng K-Fold cross-validation đã định nghĩa.
  + scoring='accuracy': Sử dụng độ chính xác để đánh giá hiệu suất mô hình.
  + n\_jobs=-1: Sử dụng tất cả các luồng (core) của CPU để tăng tốc độ tính toán.

grid\_search.fit(X\_train, y\_train)

* Huấn luyện mô hình với tập huấn luyện X\_train và y\_train, thực hiện việc tìm kiếm lưới trong quá trình huấn luyện.

**print**(f"Best parameters found: {grid\_search.best\_params\_}")

**print**(f"Best cross-validation accuracy: {grid\_search.best\_score\_:.2f}")

* In ra các siêu tham số tốt nhất đã được tìm thấy và độ chính xác tốt nhất trong quá trình cross-validation. grid\_search.best\_params\_ chứa các tham số tốt nhất và grid\_search.best\_score\_ chứa độ chính xác cao nhất.
* **Kết luận:**
* Best parameters found: {'class\_weight': None, 'max\_depth': 3, 'max\_features': 'sqrt', 'max\_leaf\_nodes': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'splitter': 'best'}
* Best cross-validation accuracy: 0.86

##### Đánh giá mô hình và tham số.

* **Confusion Matrix (Ma trận nhầm lẫn).**

A blue and white diagram

Description automatically generated

**Nhận xét:**

* Nhãn dizzle: Mô hình đoán dự đoán đúng 1 lần và dự đoán nhầm thành nhãn sun 6 lần. Mô hình chỉ nhầm lẫn với lớp sun, điều này có thể là do các đặc trưng để phân biệt "drizzle" và "sun" không đủ rõ ràng, hoặc các đặc trưng thời tiết có phần trùng lặp.
* Nhãn fog: Mô hình không dự đoán đúng lần nào mà bị dự đoán nhầm tới 17 lần nhãn sun. Điều này có thể do đặc trưng thời tiết của "fog" chưa đủ rõ ràng hoặc có đặc trưng giống "sun" trong bộ dữ liệu, làm cho mô hình gặp khó khăn trong việc nhận diện lớp này.
* Nhãn rain: Mô hình dự đoán đúng 90 lần, dự đoán nhầm thành nhãn snow 1 lần và nhãn sun 7 lần. Mô hình có độ chính xác cao khi dự đoán "rain" vì các đặc trưng như độ ẩm và nhiệt độ có thể giúp phân biệt rõ "rain" với các loại thời tiết khác. Nhầm lẫn với "sun" có thể xảy ra trong các trường hợp trời vừa mưa xong và có nắng, hoặc mưa rất nhẹ, tạo điều kiện để mô hình khó phân biệt rõ ràng.
* Nhãn snow: Mô hình dự đoán đúng 2 lần, dự đoán nhầm sang rain 3 lần. "Snow" và "rain" đều có thể có các đặc trưng chung như nhiệt độ thấp hoặc độ ẩm cao, điều này có thể gây ra sự nhầm lẫn giữa hai loại.
* Nhãn sun: Mô hình dự đoán đúng 93 lần và không có lần nào bị dự đoán nhầm. Mô hình hoạt động rất tốt với lớp "sun" vì thời tiết nắng có các đặc trưng dễ nhận diện.

**Kết luận:**

* Mô hình đạt độ chính xác cao với các nhãn "rain" và "sun". Điều này cho thấy rằng các đặc trưng đầu vào, chẳng hạn như độ sáng, độ ẩm, hoặc nhiệt độ, có khả năng phân biệt rõ giữa các điều kiện thời tiết khi trời nắng và khi có mưa.
* Mô hình đạt độ chính xác khá kém với các nhan "drizzle", "fog", và "snow". Do các mẫu không đủ đa dạng để mô hình học được các đặc trưng phong phú. Điều này dẫn đến hiệu suất kém khi phân loại các lớp này, vì mô hình ID3 (một dạng cây quyết định) dễ bị ảnh hưởng bởi sự mất cân đối dữ liệu.
* **Learning Curve (Biểu đồ đường học).**

A graph with lines and numbers

Description automatically generated with medium confidence

**Nhận xét:**

* + **Training Score:**
    - **Giai đoạn đầu (ít mẫu huấn luyện)**: Đường **training score** bắt đầu rất cao, gần mức 0.95, cho thấy mô hình có khả năng khớp rất tốt với dữ liệu huấn luyện. Điều này phản ánh rằng mô hình đang bị **overfitting** – tức là mô hình học quá kỹ các chi tiết của tập huấn luyện nhưng không tổng quát tốt cho dữ liệu mới.
    - **Khi tăng số lượng mẫu huấn luyện**: Training score giảm xuống và bắt đầu ổn định quanh mức 0.85 khi số lượng mẫu vượt quá khoảng 200 mẫu. Điều này cho thấy khi có thêm dữ liệu, mô hình bớt phụ thuộc vào các mẫu huấn luyện ban đầu và học được cách tổng quát hơn.
    - **Cuối cùng (số lượng mẫu lớn hơn 400)**: Training score duy trì ở mức khoảng 0.85 và có xu hướng hội tụ, cho thấy mô hình đạt độ chính xác tối đa với khả năng hiện tại của nó và thêm dữ liệu không giúp tăng thêm độ chính xác trên tập huấn luyện.
  + **Validation Score:**
    - **Giai đoạn đầu (ít mẫu huấn luyện)**: Validation score bắt đầu thấp, dưới 0.8, và có sự biến động khá lớn, cho thấy mô hình không có khả năng tổng quát tốt với dữ liệu mới khi số lượng mẫu huấn luyện còn ít.
    - **Khi tăng số lượng mẫu huấn luyện**: Validation score tăng dần và hội tụ quanh mức 0.85 khi số lượng mẫu đạt khoảng 400. Điều này cho thấy rằng khi có nhiều dữ liệu hơn, mô hình học được đặc trưng tốt hơn và bớt overfitting, giúp tăng khả năng tổng quát hóa.
    - **Cuối cùng (số lượng mẫu lớn hơn 400)**: Validation score ổn định ở mức khoảng 0.85 và nằm rất gần với đường training score, cho thấy mô hình đã đạt mức ổn định. Sự hội tụ của hai đường này phản ánh mô hình không còn overfitting và đã học được cách tổng quát hóa tốt nhất có thể với dữ liệu hiện tại.
  + **Test Score:**
    - **Giai đoạn đầu (ít mẫu huấn luyện)**: Test score ban đầu thấp, khoảng 0.825, và có biến động khi số lượng mẫu huấn luyện tăng lên. Điều này tương tự như validation score, do test score cũng bị ảnh hưởng bởi khả năng tổng quát hóa yếu của mô hình khi số lượng mẫu nhỏ.
    - **Khi tăng số lượng mẫu huấn luyện**: Test score dao động và cũng có xu hướng tăng lên cùng với validation score khi có thêm dữ liệu. Tuy nhiên, nó không hoàn toàn trùng với validation score và có một chút thấp hơn, cho thấy mô hình chưa đạt độ chính xác tối đa trên tập test.
    - **Cuối cùng (số lượng mẫu lớn hơn 400)**: Test score dao động xung quanh mức 0.85 và không cải thiện thêm, cho thấy mô hình đã đạt giới hạn khả năng dự đoán trên tập kiểm tra với dữ liệu hiện tại.

**Kết luận:**

* + **Sự hội tụ giữa Training Score và Validation Score**: Khi số lượng mẫu lớn hơn 400, khoảng cách giữa training score và validation score hầu như biến mất, cho thấy mô hình đã giảm được overfitting và học tốt các đặc trưng cần thiết để tổng quát hóa.
  + **Khoảng cách nhỏ giữa Test Score và Validation Score**: Test score có xu hướng thấp hơn một chút so với validation score, nhưng nhìn chung là ổn định, cho thấy mô hình có khả năng tổng quát hóa tương đối tốt trên dữ liệu chưa thấy trước đó.
  + **Giới hạn độ chính xác của mô hình**: Cả ba đường đều hội tụ quanh mức 0.85, cho thấy mô hình ID3 hiện tại đã đạt giới hạn về độ chính xác với dữ liệu và cấu trúc mô hình đang sử dụng.
* **Classification report.**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Nhận xét:**

* + **Tập trainning:**
    - Mô hình đạt **accuracy** là 0.86, cho thấy mô hình có khả năng dự đoán khá tốt trên tập huấn luyện.
    - **Rain** và **Sun** có **precision**, **recall** và **f1-score** tương đối cao (0.98 và 0.77 với lớp sun). Điều này cho thấy mô hình có thể phân biệt khá tốt giữa các điều kiện này trong tập huấn luyện.
    - **Drizzle** và **Fog** có **precision** và **recall** rất thấp (drizzle có f1-score là 0.24 và fog là 0.00). Điều này cho thấy mô hình gặp khó khăn khi phân biệt các tình huống này.
    - **Snow** có precision cao (1.00) nhưng recall thấp (0.56), cho thấy mô hình có thể phát hiện chính xác những mẫu được dự đoán là snow nhưng bỏ sót nhiều trường hợp thực sự thuộc lớp này.
  + **Tập validation**
    - Accuracy đạt 0.85, thấp hơn một chút so với tập huấn luyện nhưng vẫn ở mức khá cao.
    - **Rain** và **Sun** có hiệu suất cao nhất với f1-score lần lượt là 0.95 và 0.87, cho thấy mô hình nhận diện tốt hai loại thời tiết này trên dữ liệu chưa từng thấy.
    - **Drizzle** và **Fog** có f1-score là 0.00, cho thấy mô hình gần như không nhận diện được các mẫu của hai lớp này.
    - **Snow** có precision là 1.00 nhưng recall là 0.33, điều này cho thấy có rất ít mẫu snow được phát hiện đúng.
  + **Tập test**
* Mô hình đạt **accuracy** là 0.85, tương tự với tập validation.
* **Rain** và **Sun** duy trì mức độ chính xác cao nhất với f1-score lần lượt là 0.94 và 0.86. Điều này cho thấy mô hình có khả năng ổn định trong việc dự đoán hai lớp này ngay cả trên dữ liệu hoàn toàn mới.
* **Drizzle** có precision là 1.00 nhưng recall rất thấp (0.14), cho thấy mô hình chỉ nhận diện được một phần rất nhỏ các mẫu thực sự thuộc lớp này.
* **Fog** có recall và precision là 0, điều này nghĩa là mô hình hoàn toàn không thể dự đoán đúng bất kỳ mẫu nào của lớp này.
* **Snow** có precision là 0.67 nhưng recall chỉ đạt 0.40, thể hiện rằng mô hình vẫn chưa nhận diện tốt các mẫu snow.

**Kết luận:**

* + **Hiệu suất tốt trên lớp Rain và Sun**: Mô hình có khả năng phân biệt tốt giữa các điều kiện thời tiết phổ biến, đặc biệt là **Rain** và **Sun** trên cả ba tập dữ liệu.
  + **Hiệu suất thấp trên các lớp ít dữ liệu**: Mô hình gặp khó khăn lớn với các lớp có số lượng mẫu nhỏ như **Drizzle**, **Fog**, và **Snow**, dẫn đến precision và recall rất thấp cho các lớp này.
  + Nguyên nhân do sự chênh lệch lớn giữa các lớp có thể là nguyên nhân chính dẫn đến hiệu suất kém của mô hình trên các lớp ít mẫu.
* **Loss function.**
* Do trong bài toán này entropy có xu hướng ziczac vì phân chia không đồng đều nên thay vào đó sử dụng biểu đồ ROC (Receiver Operating Characteristic curve) để đánh giá hiệu suất của mô hình phân loại.

A graph of different colored lines

Description automatically generated

**Nhận xét:**

* + **Lớp Drizzle**: Đường ROC của lớp Drizzle có diện tích dưới đường cong (AUC) là 0.81, cho thấy mô hình có khả năng phân biệt khá tốt đối với lớp này. Tuy nhiên, AUC 0.81 cũng chỉ ra rằng mô hình có khả năng phân biệt chưa hoàn toàn tối ưu cho lớp này, và có thể còn nhầm lẫn giữa lớp Drizzle với các lớp khác.
  + **Lớp Fog**: Đường ROC của lớp Fog có AUC là 0.79, thấp nhất trong các lớp, cho thấy khả năng phân loại cho lớp này là yếu nhất trong các lớp đã được đánh giá.
  + **Lớp Rain**: Đường ROC của lớp Rain có AUC là 0.95. Đường ROC tăng dần và ổn định, thể hiện hiệu suất rất cao, gần đạt mức hoàn hảo trong việc phân loại đúng các trường hợp thuộc lớp này.
  + **Lớp Snow**: Đường ROC của lớp Snow có AUC là 0.98, cao nhất trong các lớp, cho thấy.
  + **Lớp Sun**: Đường ROC của lớp Sun có AUC là 0.90. Đường ROC tăng dần và ổn định thể hiện hiệu suất tốt trong việc phân loại các trường hợp của lớp này.

**Kết luận:**

* + Đường ROC cho lớp Rain và lớp Sun khá cao, cho thấy mô hình có độ chính xác cao và khả năng phân biệt mạnh mẽ giữa các lớp này với các lớp còn lại.
  + Đường ROC lớp Snow (AUC 0.98) rất cao, gần 1.0 gần như hoàn hảo. Tuy nhiên khi xem xét tổng quan tất cả các biểu đồ thì thấy rằng lớp Snow đang bị overfitting do dữ liệu của lớp này khá ít so với các lớp khác.
  + Drizzle (AUC 0.81) và lớp Fog (AUC 0.79) có hiệu suất phân loại yếu nhất.

### Neural Network.

#### Lý thuyết.

* Khái niệm: Neural Network là một mô hình mô phỏng cách mà hệ thống thần kinh con người hoạt động. Neural Network gồm các nơ-ron kết nối với nhau thông qua các trọng số. Khi một dữ liệu được đưa vào, nó được lan truyền qua các lớp nơ-ron, qua quá trình huấn luyện, mô hình học cách phân loại và dự đoán dữ liệu mới.
* Hàm kích hoạt (Activation Function):
  + ReLU (Rectified Linear Unit): Thường được sử dụng trong các mạng nơ-ron có nhiều lớp vì khả năng khắc phục hiện tượng mất mát gradient trong quá trình huấn luyện.
  + Tanh (Hàm tanh): Là hàm kích hoạt hyperbolic tangent, thường được sử dụng trong các mạng nơ-ron truyền thống.
* Hàm mất mát (Loss Function):
  + Hàm mất mát là hàm giúp đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực tế.
  + Đối với bài toán phân loại, đặc biệt là phân loại đa lớp, hàm mất mát thường được sử dụng là Cross-Entropy Loss. Hàm này giúp tối ưu hóa mạng bằng cách giảm thiểu sự khác biệt giữa nhãn dự đoán và nhãn thực tế.
* Lan truyền ngược (Backpropagation):
  + Đây là quá trình tối ưu hóa mạng nơ-ron, giúp mạng điều chỉnh trọng số của các liên kết giữa các nơ-ron dựa trên giá trị của hàm mất mát. Lan truyền ngược sử dụng thuật toán Gradient Descent để điều chỉnh trọng số nhằm giảm thiểu hàm mất mát sau mỗi lần huấn luyện.

#### Input và Output.

* **Input.**
* Dữ liệu đầu vào là các đặc trưng được lấy ra từ dữ liệu thời tiết, bao gồm các thông tin đo đạc thực tế về điều kiện khí hậu:
  + precipitation (lượng mưa), temp\_max (nhiệt độ cao nhất), temp\_min (nhiệt độ thấp nhất), wind (tốc độ gió).
* Chuẩn hóa dữ liệu:
  + Mô hình **Neural Network** sẽ nhận các đặc trưng trên dưới dạng số học và được chuẩn hóa để đảm bảo các giá trị nằm trong 1 phạm vi giúp cho mô hình tiết kiệm thời gian học dữ liệu.
* **Output.**
* Dữ liệu đầu ra là các nhãn mà mô hình dự đoán cho dữ liệu đầu vào thể hiện loại thời tiết của ngày đó. Các nhãn gồm:
  + sun (nắng), rain (mưa), drizzle (mưa phùn), snow (tuyết), fog (sương mù).
* **Neural Network** sẽ đưa ra kết quả dự đoán dưới dạng 1 số nguyên mã hóa cho từng loại thời tiết. Sau đó, kết quả sẽ được giải mã trở lại thành nhãn dạng chữ

Ví dụ: Số 0 thì là sunny, Số 1 sẽ là rainy,.....

#### Giải thích code.

* **Thư viện cần thiết.**

**import** joblib

**import** pandas as pd

**import** numpy as np

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** LabelEncoder, StandardScaler

**from** sklearn.neural\_network **import** MLPClassifier

* **joblib**: Dùng để lưu và tải các mô hình, đối tượng nhằm triển khai dễ dàng.x
* **pandas (pd)**: Thư viện hỗ trợ xử lý dữ liệu dạng bảng, chủ yếu là đọc dữ liệu từ file CSV và thao tác trên DataFrame.
* **numpy (np)**: Thư viện toán học và xử lý mảng, giúp thao tác trên các cấu trúc dữ liệu mảng.
* **sklearn.model\_selection- train\_test\_split**: Chia dữ liệu huấn luyện thành 3 tập là training, test và validation.
* **sklearn.preprocessing- LabelEncoder**: Mã hóa các đặc trưng thành số để mô hình dễ học.
* **sklearn.preprocessing- StandardScaler**: Chuẩn hóa dữ liệu đầu vào để giảm chênh lệch giữa các đặc trưng.
* **sklearn.neural\_network- MLPClassifier**: Xây dựng và huấn luyện mô hình mạng nơ-ron đa lớp (Multi-layer Perceptron).
* **Xử lý dữ liệu đầu vào.**

df = pd.read\_csv('./seattle-weather.csv')

* Đọc tệp dữ liệu: Đọc tệp seattle-weather.csv vào DataFrame (df) bằng pandas.

df = df.dropna()

* Xử lý dữ liệu thiếu: Kiểm tra và loại bỏ bất kỳ hàng nào trong df có chứa giá trị null để tránh lỗi khi huấn luyện mô hình.

le = LabelEncoder()

df['weather\_encoded'] = le.fit\_transform(df['weather'])

* Mã hóa cột phân loại: *LabelEncoder* được sử dụng để chuyển đổi cọt *weather* từ giá trị dạng chữ (sun, rainy,...) thành các số nguyên (0,1,2,...) để lưu cột mới *weather\_encoded*.

joblib.dump(le, 'label\_encoder.pkl')

* Lưu đối tượng mã hóa: *le* được lưu vào tệp *label\_encoder.pkl* để sử dụng trong việc giải mã nhãn khi triển khai lên web.

X\_data = df[['precipitation', 'temp\_max', 'temp\_min', 'wind']].values

* Chọn các cột đặc trưng

scaler = StandardScaler()

X\_data = scaler.fit\_transform(X\_data)

* Chuẩn hóa đầu vào:
  + StandardScaler chuẩn hóa *X\_data* để đưa tất cả các đặc trưng về trung bình 0 và độ lệch chuẩn 1.
  + *fit\_transform* vừa tính toán các tham số chuẩn hóa (fit) vừa áp dụng chuẩn hóa (transform) lên *X\_data*.

joblib.dump(scaler, 'scaler.pkl')

* Lưu đối tượng scaler: Lưu vào file scaler.pkl.

y\_data = df['weather\_encoded'].values

* Tạo biến nhãn: y\_data chứa các nhãn đã được mã hóa từ weather\_encoded dưới dạng số, đại diện cho mục tiêu dự đoán của mô hình.
* **Xây dựng mô hình.**

# Chia tập dữ liệu thành 3 tập: training (70%), validation (15%), test (15%)

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X\_data, y\_data, test\_size=0.3, shuffle=True, random\_state=42)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, shuffle=True, random\_state=42)

* Chia dữ liệu thành các tập: *train\_test\_split* chia dữ liệu *x\_data* và *y\_data* thành 3 tệp:
  + Training (70%): Để huấn luyện mô hình.
  + Validation (15%): Để điều chỉnh mô hình.
  + Test (15%): Để kiểm tra hiệu suất mô hình.
  + Shuffle=True: Trộn dữ liệu ngẫu nhiên để đảm bảo phân phối đồng đều.
  + random\_state=42: Đảm bảo kết quả chia dữ liệu luôn giống nhau.

X\_train\_sample, \_, y\_train\_sample, \_ = train\_test\_split(X\_train, y\_train, test\_size=0.5, random\_state=42)

* Lấy mẫu nhỏ từ dữ liệu huấn luyện: Tạo 1 tập dữ liệu huấn luyện nhỏ hơn để giảm thời gian huấn luyện và thử nghiệm nhanh hơn.

clf = MLPClassifier(

    hidden\_layer\_sizes=(40, 20, 10),

    max\_iter=500,

    activation='tanh',

    solver='adam',

    random\_state=42,

    early\_stopping=True,

    learning\_rate\_init=0.01

)

* Khởi tạo mô hình nơ-ron đa lớp (MLPClassifier): Lấy cấu hình đã kiểm tra.

clf.fit(X\_train\_sample, y\_train\_sample)

* Huấn luyện mô hình: Sử dụng dữ liệu mẫu X\_train\_sample và Y\_train\_sample để huấn luyện mô hình MLP.

joblib.dump(clf, 'neural\_network\_model\_best.pkl')

* Lưu mô hình

#### Biểu đồ và các tham số mô hình.

##### Giải thích biểu đồ và tham số.

* Định nghĩa mạng nơ-ron và tham số cần tìm:

param\_grid\_nn = {

    'hidden\_layer\_sizes': [

        (10,), (20,), (30,),

        (20, 10), (30, 20, 10),

       (40, 20, 10), (50, 30, 10)],

    'activation': ['relu', 'tanh'],

    'solver': ['adam', 'sgd'],

    'learning\_rate\_init': [0.1, 0.01, 0.001, 0.0001],

    'max\_iter': [500, 1000],

    'early\_stopping': [True]

}

* + param\_grid\_nn: Định nghĩa từ điển *param\_grid\_nn* chứa các tham số khác nhau để thử nghiệm cho mạng nơ-ron
  + 'hidden\_layer\_sizes': Danh sách các cấu hình kích thước tầng ẩn khác nhau.
  + 'activation': Các hàm kích hoạt để thử nghiệm: 'relu' (Rectified Linear Unit) và 'tanh':
    - ‘relu’ (Rectified Linear Unit): Đây là một trong những hàm kích hoạt phổ biến nhất, đặc biệt hiệu quả với các mạng sâu. Nó giúp giảm thiểu vấn đề gradient biến mất trong quá trình huấn luyện.
    - ‘tanh’ (Hàm tanh): Là hàm kích hoạt hyperbolic tangent, thường được sử dụng trong các mạng nơ-ron truyền thống.
  + 'solver': Các thuật toán tối ưu hóa: 'adam' và 'sgd':
    - ‘adam’:Một thuật toán tối ưu hóa phổ biến, thích hợp với mạng nơ-ron có rất nhiều tham số. Adam sử dụng cả mô-men và hệ số learning rate thích ứng, giúp mô hình hội tụ nhanh hơn.
    - ‘sgd’: Stochastic Gradient Descent, một thuật toán tối ưu hóa dựa trên gradient descent, có thể giúp tránh việc rơi vào các điểm cực tiểu cục bộ nhưng tốc độ chậm hơn Adam.
  + 'learning\_rate\_init': Các tốc độ học khởi đầu khác nhau.
  + 'max\_iter': Số lần lặp tối đa cho mỗi lần huấn luyện.
  + 'early\_stopping': Ngừng huấn luyện nếu không có cải thiện nào trong một số vòng lặp nhất định để tránh overfitting.
* Khởi tạo MLPClassifier và GridSearchCV:

mlp\_model = MLPClassifier(random\_state=42)

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

grid\_search\_nn = GridSearchCV(estimator=mlp\_model, param\_grid=param\_grid\_nn, cv=kf, scoring='accuracy', n\_jobs=-1)

* + mlp\_model: Khởi tạo một mô hình mạng nơ-ron MLPClassifier.
  + kf: Sử dụng KFold để thực hiện chia tập dữ liệu thành 5 tập, trộn thứ tự các mẫu và đảm bảo độ lặp lại bằng cách cố định *random\_state*.
  + grid\_search\_nn: Sử dụng GridSearchCV để tìm kiếm tổ hợp tham số tốt nhất cho mạng nơ-ron bằng cách huấn luyện trên các tập dữ liệu được chia bởi kf. scoring='accuracy' chỉ định sử dụng độ chính xác để đánh giá mô hình, n\_jobs=-1 để sử dụng tất cả các CPU.

grid\_search\_nn.fit(X\_train, y\_train)

* Huấn luyện mô hình với Cross-validation: Thực hiện tìm kiếm theo lưới để huấn luyện mô hình mạng nơ-ron trên *X\_train* và *y\_train* với các tham số khác nhau.

print(f"Best parameters found for Neural Network: {grid\_search\_nn.best\_params\_}")

print(f"Best cross-validation accuracy for Neural Network: {grid\_search\_nn.best\_score\_:.4f}")

* Hiển thị các thông số tốt nhất

best\_nn\_model = grid\_search\_nn.best\_estimator\_

y\_train\_pred\_nn = best\_nn\_model.predict(X\_train)

y\_val\_pred\_nn = best\_nn\_model.predict(X\_val)

y\_test\_pred\_nn = best\_nn\_model.predict(X\_test)

* Đánh giá mô hình với tham số tốt nhất trên tập validation và test:
  + Trích xuất mô hình có tham số tốt nhất (best\_nn\_model) từ quá trình tìm kiếm.
  + Dùng best\_nn\_model để dự đoán trên các tập X\_train, X\_val, và X\_test.

train\_accuracy = round(accuracy\_score(y\_train, y\_train\_pred\_nn), 4)

val\_accuracy = round(accuracy\_score(y\_val, y\_val\_pred\_nn), 4)

test\_accuracy = round(accuracy\_score(y\_test, y\_test\_pred\_nn), 4)

* Đánh giá độ chính xác: Sử dụng accuracy\_score để tính độ chính xác cho từng tập train (train\_accuracy), validation (val\_accuracy), test (test\_accuracy), sau đó làm tròn đến 4 chữ số.

print("\nClassification Report on Training Set")

print(classification\_report(y\_train, y\_train\_pred\_nn, target\_names=le.classes\_, zero\_division=0))

print("\nClassification Report on Validation Set")

print(classification\_report(y\_val, y\_val\_pred\_nn, target\_names=le.classes\_, zero\_division=0))

print("\nClassification Report on Test Set")

print(classification\_report(y\_test, y\_test\_pred\_nn, target\_names=le.classes\_, zero\_division=0))

* In báo cáo phân loại cho từng tập dữ liệu:

##### Đánh giá mô hình và tham số.

* **Confusion matrix (Ma trận nhầm lẫn)**:

A blue and white chart with a blue square

Description automatically generated

**Nhận xét:**

* Dự đoán đúng:
  + Rain: Mô hình dự đoán chính xác 91 lần.
  + Sun: Mô hình dự đoán đúng 93 lần.
  + Snow: Mô hình dự đoán chính xác 1 lần.
* Dự đoán sai:
  + Drizzle: Tất cả 7 trường hợp drizzle đều bị dự đoán nhầm là sun.
  + Fog: Tất cả 17 trường hợp fog đều bị dự đoán nhầm thành sun.
  + Rain: Có 7 trường hợp rain bị nhầm thành sun.
  + Snow: Có 4 trường hợp snow bị dự đoán nhầm là rain

**Kết luận**:

* Mô hình hoạt động tốt nhất với các điều kiện mưa (rain) và nắng (sun), vì các giá trị trên đường chéo lớn nhất ở hai nhãn này.
* Mô hình cần cải thiện khả năng phân biệt giữa các loại thời tiết ít xuất hiện như drizzle, fog, và snow.
* Có thể cần thêm dữ liệu hoặc điều chỉnh các thông số mô hình để giảm thiểu việc nhầm lẫn giữa drizzle và sun, hoặc fog và sun.
* **Learning curve (Biểu đồ đường học):**

A graph showing the growth of a network

Description automatically generated

**Nhận xét:**

* Training score:
* Độ chính xác trên tập huấn luyện bắt đầu tăng khi số lượng mẫu huấn luyện tăng từ 100 đến hơn 300 mẫu.
* Sau khi đạt hơn 300 mẫu, độ chính xác ổn định ở mức khoảng 0.8. Điều này cho thấy rằng mô hình học tốt từ dữ liệu huấn luyện và đạt được mức hiệu suất ổn định khi có đủ số lượng mẫu.
  + Đường này ổn định sớm, điều đó cho thấy mô hình không gặp vấn đề về underfitting.
  + Validation score:
  + Đường này biểu diễn độ chính xác trên tập vali (validation set), và ban đầu thấp hơn so với đường huấn luyện (do dữ liệu kiểm định không được dùng để học trực tiếp).
  + Xu hướng tăng: Độ chính xác trên tập kiểm định tăng dần khi số lượng mẫu huấn luyện tăng lên. Điều này chứng tỏ rằng mô hình có khả năng tổng quát hóa tốt hơn với nhiều dữ liệu huấn luyện hơn.
  + Hội tụ: Đường validation score dần tiến gần đến đường training score, cho thấy mô hình tổng quát hóa tốt hơn khi có nhiều dữ liệu và không còn hiện tượng overfitting rõ rệt.
  + Test score:
    - Đây là độ chính xác trên tập test (test set), đại diện cho dữ liệu hoàn toàn chưa được thấy trước.
    - Ban đầu test score thấp (xấp xỉ 0.6) khi số lượng mẫu nhỏ, nhưng sau đó tăng lên khi mô hình được huấn luyện với nhiều mẫu hơn.
    - Sự hội tụ: Test score cũng tiến dần đến các đường khác khi số lượng mẫu đạt khoảng 700-800. Tuy nhiên, có một sự chênh lệch nhẹ giữa test score và validation/training score, cho thấy mô hình có thể chưa hoàn toàn tối ưu hóa tốt trên dữ liệu chưa thấy trước.

**Kết luận:**

* + **Training score** ổn định ở mức cao sau khi có đủ dữ liệu (trên 300 mẫu), điều này cho thấy mô hình học tốt từ tập huấn luyện.
  + **Validation score** và **Test score** có xu hướng cải thiện và hội tụ về các giá trị của training score khi số lượng mẫu tăng, điều này cho thấy mô hình không bị overfitting quá mức và có khả năng tổng quát hóa tốt khi huấn luyện với nhiều dữ liệu.
  + Khoảng cách nhỏ giữa test score và các giá trị khác cho thấy mô hình vẫn có tiềm năng cải thiện thêm, có thể qua việc điều chỉnh các siêu tham số hoặc bổ sung thêm dữ liệu huấn luyện để cải thiện khả năng dự đoán trên dữ liệu mới.
* **Loss Curve:**

A graph with a line graph

Description automatically generated

**Nhận xét:**

* Mất mát giảm nhanh ban đầu (Iterations 0 - 5):
* Trong giai đoạn đầu, mất mát giảm rất mạnh, từ giá trị khoảng 1.3 xuống còn 0.7 trong vài lần lặp đầu tiên.
* Điều này là do mô hình nhanh chóng học được các đặc trưng cơ bản của dữ liệu trong những bước đầu tiên, giúp cải thiện đáng kể độ chính xác và giảm mất mát.
* Giai đoạn giảm từ từ (Iterations 5 - 15):
  + - Sau khoảng 5 lần lặp, tốc độ giảm mất mát bắt đầu chậm lại, cho thấy mô hình đã học được hầu hết các đặc trưng quan trọng.
    - Mặc dù mất mát vẫn tiếp tục giảm, nhưng mức độ cải thiện không còn đáng kể như trước, do mô hình đã gần đạt trạng thái tối ưu hóa và cần nhiều lần lặp hơn để cải thiện nhỏ hơn.
* Xu hướng phẳng dần (Iterations 15 - 35):
  + - Từ khoảng 15 đến 35 lần lặp, đường mất mát có xu hướng phẳng dần, với giá trị xoay quanh mức 0.5. Đây là dấu hiệu cho thấy mô hình đang dần hội tụ và tiến gần đến điểm mà việc giảm mất mát không còn dễ dàng nữa.
* Dao động nhẹ ở cuối (Iterations 30 - 35):
  + - Mô hình đã gần đạt điểm tối ưu, và việc học thêm không còn giúp giảm mất mát nhiều hơn.
    - Quá trình điều chỉnh: Những dao động này có thể là dấu hiệu của overfitting nhẹ, khi mô hình cố gắng học những chi tiết nhỏ của tập huấn luyện, dẫn đến những thay đổi không còn giúp ích cho việc tối ưu hóa tổng thể.

**Kết luận:**

* Mô hình học tốt: Đường mất mát giảm mạnh ở những lần lặp đầu cho thấy mô hình học tốt từ dữ liệu.
* Hội tụ ổn định: Sau khoảng 20 lần lặp, mô hình hội tụ và mất mát đạt mức ổn định. Đây là dấu hiệu tốt, cho thấy mô hình đang tối ưu hóa mà không gặp nhiều vấn đề về underfitting hay overfitting nghiêm trọng.
* Cần chú ý giai đoạn dao động: Dao động nhỏ ở cuối có thể cho thấy cần theo dõi thêm, để đảm bảo mô hình không bị overfitting. Một cách kiểm tra là sử dụng early stopping để ngăn việc học quá mức nếu thấy hiệu suất trên tập kiểm định bắt đầu xấu đi.
* **Classification Report:**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a test

Description automatically generated

**Nhận xét:**

* + Training set:
    - Accuracy: Mô hình đạt độ chính xác là 0.85 (85%), nghĩa là mô hình phân loại đúng 85% các mẫu trong tập huấn luyện.
    - Precision và Recall cho từng lớp:
* **Drizzle và Fog:** Độ chính xác và độ nhạy đều là 0, cho thấy mô hình không thể phân loại chính xác bất kỳ trường hợp nào thuộc hai lớp này. Điều này có thể là do số lượng mẫu trong hai lớp này quá ít (39 cho drizzle và 69 cho fog), khiến mô hình không đủ dữ liệu để học tốt.
* **Rain:** Có độ chính xác là 0.97 và độ nhạy là 0.93, cho thấy mô hình phân loại tốt các trường hợp mưa. Tuy nhiên, vẫn có một số ít trường hợp bị nhầm lẫn.
* **Snow:** Mặc dù có độ chính xác là 1.00, nhưng độ nhạy chỉ là 0.44, nghĩa là mô hình chỉ nhận diện được 44% các trường hợp tuyết, dẫn đến sự bỏ sót lớn.
* **Sun:** Có độ chính xác và độ nhạy cao (0.76 và 1.00), cho thấy mô hình hoạt động rất tốt trong việc nhận diện điều kiện trời nắng.
  + - Macro avg và Weighted avg: Các chỉ số này cho thấy sự chênh lệch giữa các lớp. Chỉ số macro trung bình thấp hơn, do mô hình có hiệu suất kém ở các lớp như drizzle và fog.
* Validation set:
  + - Accuracy: Mô hình duy trì độ chính xác là 0.85 (85%), tương tự như trên tập huấn luyện, cho thấy khả năng tổng quát hóa của mô hình không bị giảm sút đáng kể.
    - Precision và Recall cho từng lớp:
      * Drizzle, Fog, và Snow: Vẫn có độ chính xác và độ nhạy là 0, cho thấy mô hình gặp khó khăn trong việc phân loại đúng các lớp này, có thể là do số lượng mẫu trong các lớp này quá ít (7, 15, và 3).
      * Rain và Sun: Vẫn giữ được hiệu suất tốt, với các chỉ số precision, recall và f1-score đều cao. Điều này cho thấy mô hình ổn định trong việc nhận diện các điều kiện mưa và nắng.
    - Macro avg và Weighted avg: Các chỉ số này tiếp tục cho thấy sự mất cân bằng trong hiệu suất giữa các lớp.
  + Test set:
    - Accuracy: Độ chính xác giảm nhẹ xuống còn 0.84 (84%), nhưng vẫn giữ được mức tương đối ổn định
    - Precision và Recall cho từng lớp:
      * **Drizzle và Fog:** Vẫn không có sự cải thiện, với các chỉ số precision, recall và f1-score đều bằng 0.
      * **Rain:** Hiệu suất gần tương tự như trên các tập trước, với precision là 0.96 và recall là 0.93.
      * **Snow:** Chỉ có 5 mẫu, mặc dù precision là 1.00 nhưng recall chỉ là 0.20, cho thấy mô hình chỉ nhận diện được rất ít trường hợp tuyết.
      * **Sun:** Duy trì hiệu suất tốt với các chỉ số cao, cho thấy mô hình vẫn nhận diện điều kiện trời nắng tốt.
    - Macro avg và Weighted avg: Các chỉ số này vẫn thấp do hiệu suất kém của mô hình đối với các lớp ít dữ liệu như drizzle và fog.

**Kết luận:**

* + Mô hình có khả năng phân loại tốt với các lớp phổ biến hơn (rain và sun), nhưng gặp khó khăn trong việc nhận diện các lớp ít dữ liệu (drizzle, fog, snow).
  + Sự chênh lệch trong số lượng mẫu giữa các lớp có thể là nguyên nhân chính dẫn đến hiệu suất kém đối với các lớp ít xuất hiện. Việc thu thập thêm dữ liệu cho các lớp này hoặc sử dụng các kỹ thuật xử lý mất cân bằng dữ liệu (như oversampling, undersampling hoặc sử dụng hàm mất mát có trọng số) có thể giúp cải thiện kết quả.

### Stacking.

#### Khái niệm.

Stacking là một kỹ thuật học máy cho kết hợp nhiều mô hình học máy khác nhau để tạo ra một mô hình tổng hợp mạnh mẽ hơn. Nó giúp tận dụng ưu điểm của từng mô hình cơ sở, từ đó tạo ra dự đoán chính xác hơn. Cơ bản thường gồm 2 phần: base model (lớp cơ sở) và meta model (lớp meta).

#### Input và Output.

* **Input.**
* Mô tả chi tiết về dữ liệu:
  + - Dữ liệu đầu vào được đọc từ tệp seattle-weather.csv. Đây là bộ dữ liệu chứa thông tin về thời tiết ở Seattle. Dữ liệu ban đầu bao gồm các cột như: date (ngày tháng), precipitation (lượng mưa), temp\_max (nhiệt độ tối đa), temp\_min (nhiệt độ tối thiểu), wind (tốc độ gió), và weather (loại thời tiết).
* Các cột được chọn làm đầu vào của mô hình bao gồm:
  + - precipitation (lượng mưa), temp\_max (nhiệt độ cao nhất), temp\_min (nhiệt độ thấp nhất), wind (tốc độ gió).
* **Output.**
* Kết quả dự đoán là nhãn chữ tương ứng với loại thời tiết trong ngày:
  + - sun (nắng), rain (mưa), drizzle (mưa phùn), snow (tuyết), fog (sương mù).

#### Giải thích code.

* **Thư viện cần thiết.**

import numpy as np

import pandas as pd

from sklearn import tree

from sklearn.linear\_model import Perceptron

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import train\_test\_split, learning\_curve

from sklearn.ensemble import StackingClassifier

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.neural\_network import MLPClassifier

import matplotlib.pyplot as plt

* **numpy (np)**: Thư viện hỗ trợ các phép toán học và xử lý mảng, giúp thao tác trên các cấu trúc dữ liệu mảng một cách nhanh chóng và hiệu quả.
* **pandas (pd)**: Thư viện xử lý dữ liệu dạng bảng, hỗ trợ đọc dữ liệu từ file CSV, Excel và thực hiện các thao tác trên DataFrame như lọc, biến đổi dữ liệu.
* **sklearn.tree - tree**: Thư viện chứa các thuật toán cây quyết định (Decision Tree) dùng để phân loại hoặc dự đoán dựa trên cấu trúc cây.
* **sklearn.linear\_model - Perceptron**: Mô hình phân loại tuyến tính đơn giản, huấn luyện để phân loại các điểm dữ liệu thành các lớp khác nhau dựa trên biên phân cách.
* **sklearn.metrics - accuracy\_score**: Tính tỷ lệ các mẫu được phân loại đúng để đánh giá độ chính xác của mô hình.
* **sklearn.metrics - classification\_report**: Tạo báo cáo chi tiết về các chỉ số như độ chính xác (precision), độ nhạy (recall), và F1-score cho từng lớp.
* **sklearn.metrics - confusion\_matrix**: Tạo ma trận nhầm lẫn, giúp đánh giá số lượng phân loại đúng và sai của từng lớp.
* **sklearn.metrics - ConfusionMatrixDisplay**: Công cụ trực quan hóa ma trận nhầm lẫn, giúp dễ dàng quan sát kết quả phân loại của mô hình.
* **sklearn.model\_selection - train\_test\_split**: Chia dữ liệu thành tập huấn luyện và tập kiểm tra để đánh giá mô hình.
* **sklearn.model\_selection - learning\_curve**: Tạo biểu đồ learning curve để đánh giá hiệu suất của mô hình khi thay đổi kích thước tập huấn luyện.
* **sklearn.ensemble - StackingClassifier**: Tạo mô hình phân loại kết hợp (ensemble) bằng cách chồng nhiều mô hình để tăng độ chính xác và giảm thiểu sai số.
* **sklearn.preprocessing - StandardScaler**: Chuẩn hóa dữ liệu đầu vào bằng cách đưa các đặc trưng về cùng một thang đo để mô hình huấn luyện hiệu quả hơn.
* **sklearn.preprocessing - LabelEncoder**: Mã hóa các đặc trưng phân loại (categorical) thành số nguyên, làm cho dữ liệu trở nên dễ học hơn với mô hình.
* **sklearn.neural\_network - MLPClassifier**: Mô hình mạng nơ-ron đa lớp (Multi-layer Perceptron), dùng để phân loại dữ liệu phi tuyến tính.
* **matplotlib.pyplot (plt)**: Thư viện vẽ đồ thị và biểu đồ, hỗ trợ trực quan hóa dữ liệu và kết quả của mô hình.
* **Xử lý dữ liệu đầu vào.**

data = pd.read\_csv('./seattle-weather.csv')

* Đọc tệp dữ liệu đầu vào

data = weather\_data.dropna()

* Loại bỏ các dòng dữ liệu thiếu, giữ lại các dòng đầy đủ thông tin.

data.drop(['date'], axis=1, inplace=True)

* Loại bỏ cột "date" vì không có giá trị dự đoán.

le = LabelEncoder()

* Tạo bộ mã hóa để chuyển giá trị chuỗi (text) thành số nguyên.

data['weather\_encoded'] = le.fit\_transform(data['weather'])

* Mã hóa cột "weather" thành cột số "weather\_encoded".

X = data[["precipitation", "temp\_max", "temp\_min", "wind"]]

* Ở bước này, các cột đặc trưng đầu vào (independent variables) được chọn ra để làm tập dữ liệu cho việc huấn luyện mô hình. Các cột được chọn bao gồm:
* precipitation: Lượng mưa (mm).
* temp\_max: Nhiệt độ cao nhất trong ngày (°C).
* temp\_min: Nhiệt độ thấp nhất trong ngày (°C).
* wind: Tốc độ gió (m/s). Đây là những đặc trưng có thể ảnh hưởng đến tình trạng thời tiết trong ngày và sẽ được dùng làm input để dự đoán.

y = data['weather\_encoded']

* Cột weather\_encoded chứa các giá trị đã được mã hóa của cột weather được chọn làm biến mục tiêu (dependent variable) để dự đoán.

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, train\_size=0.7, test\_size=0.3, random\_state=42)

* Dữ liệu đầu vào được chia thành hai phần: **70%** cho tập huấn luyện (X\_train, y\_train) và **30%** cho tập tạm thời (X\_temp, y\_temp). Tập huấn luyện được sử dụng để mô hình học và phát triển khả năng dự đoán. Tham số random\_state=42 đảm bảo rằng việc chia dữ liệu sẽ được thực hiện một cách ngẫu nhiên nhưng có thể tái tạo lại (kết quả luôn giống nhau mỗi khi chạy lại code).

X\_valid, X\_test, y\_valid, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42)

* Tiếp tục chia tập dữ liệu tạm thời (30% còn lại) thành hai tập: **15%** cho tập xác thực (X\_valid, y\_valid) và **15%** cho tập kiểm tra (X\_test, y\_test). Tập xác thực được dùng để kiểm tra hiệu suất của mô hình trong quá trình huấn luyện và điều chỉnh các tham số. Tập kiểm tra dùng để đánh giá hiệu suất cuối cùng của mô hình sau khi đã huấn luyện.

scaler = StandardScaler()

* Để chuẩn hóa các đặc trưng đầu vào, StandardScaler được khởi tạo. Điều này là quan trọng vì các đặc trưng như nhiệt độ, lượng mưa, và tốc độ gió có thể có các đơn vị và thang đo khác nhau. Chuẩn hóa dữ liệu giúp mọi đặc trưng nằm trong cùng một phạm vi giá trị, giúp mô hình học tốt hơn và tránh các giá trị lớn chi phối kết quả dự đoán.

X\_train = scaler.fit\_transform(X\_train)

* Phép chuẩn hóa (scaling) được áp dụng lên tập huấn luyện (X\_train). fit\_transform tính toán trung bình và độ lệch chuẩn của từng đặc trưng trong tập huấn luyện, sau đó chuẩn hóa các giá trị của tập này. Quá trình này đảm bảo rằng tất cả các đặc trưng đều có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp cải thiện hiệu suất mô hình.

X\_valid = scaler.transform(X\_valid)

X\_test = scaler.transform(X\_test)

* Sau khi đã chuẩn hóa tập huấn luyện, các đặc trưng trong tập xác thực (X\_valid) và tập kiểm tra (X\_test) cũng được chuẩn hóa theo cùng quy tắc (tức là cùng trung bình và độ lệch chuẩn đã tính từ tập huấn luyện). Điều này đảm bảo tính đồng nhất trong dữ liệu và giúp đánh giá hiệu suất mô hình chính xác.
* **Xây dựng mô hình.**
* Khởi tạo các mô hình cơ bản:

base\_learners = [

    ('decision\_tree', tree.DecisionTreeClassifier(max\_depth=5, random\_state=42)),

    ('perceptron', Perceptron(max\_iter=1000, random\_state=42)),

    ('neural\_network', MLPClassifier(hidden\_layer\_sizes=(5,), alpha=0.01, max\_iter=2000, random\_state=42))

]

* + **DecisionTreeClassifier**: Cây quyết định có **max\_depth=5**, tức là cây chỉ có thể phân nhánh tối đa 5 cấp độ. Điều này giúp tránh quá khớp dữ liệu (overfitting) và giữ cho mô hình đơn giản hơn. random\_state=42 đảm bảo kết quả có thể tái tạo.
  + **Perceptron**: Mô hình tuyến tính với tối đa 1000 vòng lặp (**max\_iter=1000**). Đây là phiên bản đơn giản nhất của mạng nơron, thường dùng cho các bài toán phân loại tuyến tính.
  + **MLPClassifier**: Mạng nơron nhiều lớp (**Multi-layer Perceptron**) có một lớp ẩn với 5 nút (**hidden\_layer\_sizes=(5,)**). Tham số **alpha=0.01** được sử dụng để điều chỉnh regularization, giúp mô hình tránh quá khớp. **max\_iter=2000** chỉ ra số vòng lặp tối đa cho việc học.
* Khởi tạo mô hình Stacking

stacking\_model=StackingClassifier(estimators=base\_learners, final\_estimator=MLPClassifier(hidden\_layer\_sizes=(5,), alpha=0.01, max\_iter=1000, random\_state=42))

* + **StackingClassifier** là một kỹ thuật ensemble, kết hợp nhiều mô hình con (estimators) để cải thiện hiệu suất. Trong trường hợp này, ba mô hình cơ bản được sử dụng: cây quyết định, perceptron, và mạng nơron.
  + Mô hình tổng hợp (meta learner) cũng là một **MLPClassifier**, hoạt động dựa trên kết quả của các mô hình con và đưa ra quyết định cuối cùng. Mạng nơron này có cùng kiến trúc với mạng nơron trong các mô hình con, nhưng nó học từ kết quả của các mô hình con thay vì học từ dữ liệu ban đầu.
* Huấn luyện mô hình Stacking

stacking\_model.fit(X\_train, y\_train)

* + Quá trình **fit()** huấn luyện mô hình Stacking bằng cách sử dụng tập dữ liệu huấn luyện (X\_train, y\_train). Mỗi mô hình con trong base\_learners được huấn luyện trước, sau đó kết quả của chúng được tổng hợp lại và mô hình tổng hợp (final\_estimator) sẽ học từ kết quả đó.

#### Biểu đồ và các tham số mô hình.

* **Confusion Matrix (Ma trận nhầm lẫn).**

**A blue and white diagram

Description automatically generated**

**Nhận xét:**

* **Lớp Drizzle**: Mô hình dự đoán tất cả 7 mẫu của lớp **drizzle** thành **sun**. Điều này cho thấy mô hình không thể phân biệt được **drizzle** (mưa phùn) và **sun** (trời nắng). Một phần có thể là do sự tương tự giữa các đặc trưng thời tiết trong các mẫu giữa hai lớp, hoặc có thể lớp drizzle có quá ít dữ liệu trong tập huấn luyện.
* **Lớp Fog**: Tương tự với lớp drizzle, tất cả 17 mẫu của lớp **fog** (sương mù) cũng bị mô hình dự đoán thành **sun**. Điều này cho thấy mô hình không đủ khả năng để học từ dữ liệu của lớp fog, có thể vì lớp này có quá ít dữ liệu hoặc các đặc trưng không đủ phân biệt.
* **Lớp Rain và Sun**: Mô hình hoạt động rất tốt đối với hai lớp **rain** (mưa) và **sun** (trời nắng), với 90 mẫu của mỗi lớp được dự đoán đúng. Tuy nhiên, có 7 mẫu của lớp **rain** bị nhầm lẫn thành **sun**, và 3 mẫu của **sun** bị nhầm lẫn thành **drizzle**. Điều này có thể là do các đặc trưng thời tiết trong những trường hợp này rất giống nhau, gây khó khăn cho mô hình khi phân biệt.
* **Lớp Snow**: Mô hình dự đoán 4/5 mẫu của lớp **snow** thành **rain**, chỉ có 1 mẫu được dự đoán đúng. Điều này có thể xuất phát từ sự tương đồng trong đặc trưng thời tiết giữa hai lớp này, hoặc do số lượng mẫu của lớp snow quá ít, khiến mô hình không đủ dữ liệu để học.

**Kết luận:**

* **Lỗi nhầm lẫn giữa các lớp**: Mô hình gặp vấn đề với các lớp có ít dữ liệu (drizzle, fog, snow). Điều này có thể do mô hình chưa có đủ dữ liệu để học các đặc trưng của các lớp này, dẫn đến việc dự đoán nhầm lẫn.
* **Hiệu suất tốt cho lớp phổ biến**: Mô hình hoạt động rất tốt đối với các lớp có số lượng mẫu lớn như **rain** và **sun**, với số lượng dự đoán đúng cao. Điều này cho thấy mô hình có khả năng phân loại tốt khi có đủ dữ liệu huấn luyện.
* **Vấn đề về sự mất cân bằng dữ liệu**: Các lớp có ít dữ liệu như drizzle và fog hoàn toàn bị nhầm lẫn, cho thấy rằng sự mất cân bằng dữ liệu có thể ảnh hưởng nghiêm trọng đến hiệu suất của mô hình. Giải pháp có thể là áp dụng kỹ thuật tăng cường dữ liệu (oversampling) hoặc cân bằng dữ liệu (SMOTE) để giúp mô hình học tốt hơn với các lớp nhỏ.
* **Learning Curve (Biểu đồ đường học)**

A graph with red and green lines

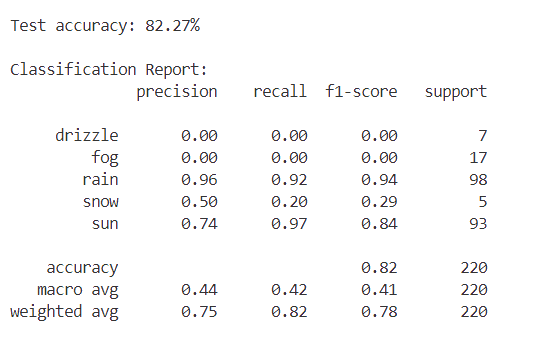
Description automatically generated

**Nhận xét:**

* **Training Score (đường đỏ):** Đây là điểm số của mô hình trên tập huấn luyện. Ở các kích thước nhỏ (như 100 hoặc 200 mẫu), điểm số của tập huấn luyện có xu hướng cao hơn so với tập xác thực, điều này thường là dấu hiệu của overfitting (quá khớp), khi mô hình học quá kỹ trên dữ liệu huấn luyện nhưng không tổng quát hóa tốt trên dữ liệu mới.
* **Validation Score (đường xanh):** Đây là điểm số của mô hình trên tập xác thực. Đường này cho thấy hiệu suất tổng quát của mô hình trên dữ liệu mà nó chưa thấy trước đó. Khi kích thước của tập huấn luyện tăng lên, điểm số trên tập xác thực cũng tăng dần và tiệm cận với điểm số của tập huấn luyện. Điều này cho thấy rằng khi có nhiều dữ liệu hơn, mô hình học tốt hơn và khả năng generalization (tổng quát hóa) cũng cải thiện.
* **Vùng bóng (độ lệch chuẩn):** Biểu đồ cũng hiển thị các vùng bóng cho cả tập huấn luyện và tập xác thực, biểu thị độ lệch chuẩn (standard deviation) của điểm số tại mỗi kích thước tập huấn luyện. Vùng bóng lớn có nghĩa là có nhiều biến động trong hiệu suất mô hình, cho thấy rằng ở các kích thước nhỏ, mô hình có thể hoạt động không ổn định. Khi tập huấn luyện lớn hơn, vùng bóng nhỏ dần, cho thấy hiệu suất của mô hình trở nên ổn định hơn.

**Kết luận:**

* **Underfitting/Overfitting**: Khi quan sát các đường cong học tập, khoảng cách giữa đường huấn luyện và đường xác thực là rất nhỏ khi kích thước tập huấn luyện tăng lên, cho thấy rằng mô hình không bị **overfitting** hay **underfitting**. Điều này có nghĩa là mô hình có khả năng tổng quát hóa tốt trên các dữ liệu chưa từng thấy trước đó.
* **Sự hội tụ của các đường cong**: Đường cong học tập cho thấy rằng khi tăng kích thước tập huấn luyện, mô hình học tốt hơn và điểm số của tập huấn luyện và tập xác thực tiệm cận nhau. Điều này cho thấy rằng với nhiều dữ liệu hơn, mô hình sẽ tiếp tục cải thiện hiệu suất. Tuy nhiên, điểm số dừng lại quanh mức 0.84, cho thấy rằng hiệu suất của mô hình đã đạt đến ngưỡng tối đa với bộ dữ liệu hiện tại.
* **Biến động trong tập xác thực**: Các điểm số trên tập xác thực có xu hướng dao động nhẹ, đặc biệt ở các kích thước nhỏ hơn. Điều này có thể là do sự phân phối không đồng đều của các lớp trong tập huấn luyện, làm cho mô hình gặp khó khăn trong việc học các đặc trưng một cách ổn định.
* **Classification report.**



**Nhận xét:**

* **Độ chính xác tổng quát (Test Accuracy: 82.27%)**:
* Độ chính xác 82.27% là kết quả khá tốt cho một bài toán phân loại với dữ liệu thời tiết. Tuy nhiên, kết quả này không đồng đều giữa các lớp.
* **Hiệu suất theo từng lớp**:
  + **Rain và Sun**: Đây là hai lớp có kết quả tốt nhất. Precision và Recall đều cao với lớp **Rain** (Precision: 0.96, Recall: 0.92) và **Sun** (Precision: 0.74, Recall: 0.97). Điều này cho thấy mô hình hoạt động hiệu quả với các lớp phổ biến và có dữ liệu phong phú.
  + **Drizzle, Fog, và Snow**: Các lớp có số lượng mẫu ít như **Drizzle**, **Fog**, và **Snow** có kết quả rất kém với Precision, Recall, và F1-Score đều bằng 0 cho **Drizzle** và **Fog**. Điều này cho thấy mô hình không thể học được đủ đặc trưng từ dữ liệu của các lớp này do có số lượng mẫu ít.
* **Macro và Weighted Averages**:
  + **Macro avg**: Thấp (Precision: 0.44, Recall: 0.42, F1-Score: 0.41), phản ánh rằng hiệu suất trung bình của mô hình trên tất cả các lớp là khá kém do sự mất cân bằng giữa các lớp.
  + **Weighted avg**: Tương đối cao (Precision: 0.75, Recall: 0.82, F1-Score: 0.78) do lớp **Rain** và **Sun** có nhiều mẫu hơn và được dự đoán tốt, kéo theo chỉ số tổng thể của mô hình.

**Kết luận:**

* **Hiệu suất tổng quát:**
  + Mô hình Stacking đạt được độ chính xác **82.27%**, cho thấy khả năng dự đoán tốt, đặc biệt đối với các lớp phổ biến như *rain* và *sun*. Điều này khẳng định rằng việc kết hợp nhiều mô hình lại với nhau (Stacking) đã giúp mô hình học được nhiều thông tin hữu ích từ các mô hình con.
* **Mất cân bằng dữ liệu:**
  + Mô hình gặp khó khăn lớn với các lớp có ít dữ liệu như *drizzle*, *fog*, và *snow*. Precision, recall và f1-score của các lớp này đều rất thấp (thậm chí bằng 0), chứng tỏ rằng mô hình không đủ khả năng để phân loại các lớp này một cách chính xác. Điều này là do sự mất cân bằng dữ liệu giữa các lớp.
* **Khả năng phân loại của các lớp phổ biến:**
  + Mô hình hoạt động rất tốt đối với lớp *rain* (độ chính xác f1-score là 0.94) và *sun* (f1-score là 0.84), cho thấy rằng với các lớp có số lượng mẫu lớn, mô hình có thể học và phân loại chính xác. Điều này cho thấy tính hữu dụng của mô hình Stacking trong việc xử lý dữ liệu lớn và đa dạng.
* **Overfitting/Underfitting:**
  + Qua biểu đồ Learning Curve, mô hình không bị vấn đề nghiêm trọng về overfitting hay underfitting, vì các đường huấn luyện và xác thực gần như tiệm cận nhau. Tuy nhiên, ở các kích thước dữ liệu nhỏ, có sự biến động nhẹ, điều này có thể do thiếu dữ liệu hoặc sự mất cân bằng dữ liệu giữa các lớp.

## Các tham số đánh giá.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Tên chỉ số** | Độ chính xác | Độ chính xác theo nhãn | Độ hồi đáp | Chỉ số F1 |
| **Mục đích** | Đánh giá tỷ lệ dự đoán đúng trên tổng số dự đoán | Đánh giá khả năng mô hình chỉ ra nhãn đúng trong số các dự đoán đúng | Đánh giá khả năng mô hình nhận diện đúng các nhãn thực | Đánh giá sự cân bằng giữa Precision và Recall |
| **Giá trị** | [0,1] | [0,1] | [0,1] | [0,1] |
| **Ý nghĩa** | -Giá trị Accuracy = 1 nghĩa là mô hình dự đoán hoàn toàn chính xác (tất cả dự đoán đều đúng).  - Nếu Accuracy = 0, mô hình hoàn toàn sai. Giá trị thấp có thể chỉ ra sự mất cân bằng dữ liệu (bias) trong tập dự đoán. | - Precision = 1 cho thấy mô hình không có lỗi dương tính giả (FP = 0), tức là tất cả các mẫu được dự đoán là dương đều thực sự là dương.  - Precision thấp chỉ ra rằng mô hình hay gán nhầm các mẫu âm thành dương (nhiều FP). | - Recall = 1 có nghĩa là mô hình không bỏ sót mẫu dương nào (FN = 0), tức là tất cả các mẫu dương đều được nhận diện đúng.  - Recall thấp nghĩa là mô hình bỏ sót nhiều nhãn dương, không nhận diện đúng các mẫu dương thực sự. | - F1 Score = 1 nghĩa là mô hình hoàn hảo trong việc cân bằng giữa Precision và Recall.  - F1 Score càng gần 0 nghĩa là mô hình kém, có sự mất cân bằng lớn giữa Precision và Recall (một trong hai có thể rất thấp). |
| **Lưu ý** | * Accuracy cao không phải lúc nào cũng tốt nếu dữ liệu bị lệch * Precision cao nhưng Recall thấp nghĩa là mô hình rất chính xác khi dự đoán nhãn dương nhưng bỏ sót nhiều nhãn dương thực tế. * Recall cao nhưng Precision thấp có nghĩa là mô hình phát hiện được nhiều nhãn dương, nhưng dự đoán nhiều nhãn dương sai (nhiều FP) | | | |
| **Công thức tính** |  |  |  |  |

Chú thích:

* TP (True Positive): Dự đoán đúng nhãn dương.
* TN (True Negative): Dự đoán đúng nhãn âm.
* FP (False Positive): Dự đoán sai nhãn dương.
* FN (False Negative): Dự đoán sai nhãn âm.

## Đánh giá mô hình.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Perceptron** | 0,74 | 0,70 | 0,74 | 0,70 |
| **ID3** | 0,85 | 0,80 | 0,85 | 0,80 |
| **Neural Network** | 0,84 | 0,77 | 0,84 | 0,79 |
| **Stacking** | 0,82 | 0,75 | 0,80 | 0,78 |

### Đánh giá qua các tham số đánh giá.

* Accuracy:
  + Mô hình **ID3** có độ chính xác cao nhất, chứng tỏ mô hình đưa ra dự đoán đúng nhất.
  + **Neural Network** có độ chính xác chỉ kém ID3 0.1, cho thấy rằng đây cũng là mô hình hoạt động tốt.
  + **Perceptron** cho thấy rằng đây là một mô hình không thích hợp để đánh giá với bộ dữ liệu khi có độ chính xác thấp nhất trong 4 mô hình.
  + **Stacking** dù là mô hình tổng hợp nhưng vẫn không có chính xác nhỉnh hơn so với **ID3** và **Neural Network**.
* Precision
  + **ID3** vẫn là mô hình có tỉ lệ cao nhất cho thấy rằng mô hình thường dự đoán dương chính xác.
  + Ở đây, cả **Neural Network** lẫn **Stacking** có tỉ lệ không chênh lệch nhau nhưng lại khá thấp so với **ID3.**
  + Trong khi đó, Perceptron lại là mô hình có thông số thấp nhất trong các mô hình, cho thấy rằng mô hình có 1 lượng dự đoán dương sai còn khá lớn (30%).
* Recall
  + Thông số của mô hình **ID3** tiếp tục là cao nhất trong 4 mô hình.
  + **Neural Network** chỉ kém so với thông số tốt nhất với 1 đơn vị rất nhỏ cho thấy mô hình vẫn xác định được mẫu dương ngang với **ID3**.
  + **Stacking** vẫn chỉ là mô hình có tỉ lệ đứng sau 2 mô hình trên dù là mô hình tổng hợp.
  + Cuối cùng, Perceptron vẫn tiếp tục là mô hình kém nhất khi không đạt được đến tỉ lệ 0.8-tỉ lệ thấp nhất trong 3 mô hình trên.
* F1 Score
  + Vì đây là chỉ số đánh giá dựa trên Precision và Recall nên ta có thể dễ dàng thấy được **ID3** là mô hình có F1 Score cao nhất trong 4 mô hình.
  + Ngay sau đó sẽ là **Neural Network** với F1 Score cũng gần bằng **ID3** cho thấy rằng mô hình cũng hoạt động khá hiệu quả.
  + **Stacking** là mô hình có chỉ số chỉ nhỉnh hơn so với **Perceptron** nhưng vẫn ổn định.
  + Perceptron là mô hình có thông số và tỉ lệ kém nhất trong 4 mô hình dù đã đạt F1 Score là 70%.

### Nhận xét.

* **Perceptron:** Là mô hình được cho là đơn giản nhất so với 3 mô hình còn lại cho thấy được 1 phần lí do vì sao mô hình có các kết quả thấp hơn các mô hình còn lại khi không đạt được đến 80% trong cả 4 tiêu chí. Điều này cho thấy rằng **Perceptron** có khả năng không phải là lựa chọn tốt cho bài toán Dự đoán thời tiết này.
* **ID3:** Là mô hình có các thông số dẫn đầu ở cả 4 tiêu chí đánh giá và khá ổn định khi luôn đạt được kết quả trên 80%. Từ đó cho ta thấy rằng đây là mô hình tối ưu cho bài toán thời tiết khi có khả năng phân tích được dữ liệu khá tốt và đưa ra quyết định rõ ràng.
* **Neural Network:** Có kết quả sau cùng khá sát so với **ID3** khi không có tiêu chí nào kết quả chênh lệch không quá 3%. Việc có hiệu suất khá cao và gần tương đương **ID3** chứng tỏ rằng **Neural Network** là 1 lựa chọn tốt, đặc biệt là khi cần sự linh hoạt trong xử lý dữ liệu.
* **Stacking:** Chính là sự kết hợp giữa 3 mô hình kể trên nhưng lại đưa ra mức kết quả trung bình khi không vượt trội so với Mạng nơ-ron nhân tạo, Cây quyết định và cũng chỉnh nhỉnh hơn 1 chút so với Perceptron. Vì vậy đây chưa phải lựa chọn tối ưu và tốt nhất trong 4 mô hình đã sử dụng.

### Kết luận.

* **ID3** sẽ là lựa chọn đầu tiên để dự đoán khi đều đứng đầu trong 4 chỉ số đánh giá. Mô hình đã đạt được hiệu quả cao trong việc phân loại chính xác các mẫu cùng với tỷ lệ dự đoán đúng cao.
* **Neural Network** cũng là 1 mô hình khá tốt khi các thông số đánh giá không quá kém so với **ID3** và có thể sử dụng để phân tích các dữ liệu phức tạp hơn trong tương lai.
* **Stacking** tuy mang tính linh hoạt nhờ việc đây là sự kết hợp giữa 3 mô hình nhưng lại không có kết quả thực sự xuất sắc, có thể là thất vọng so về kết quả từ 1 mô hình tổng hợp.
* **Perceptron** là mô hình đơn giản và có hiệu suất thấp. Đây sẽ chỉ là mô hình với sự lựa chọn để đánh giá cuối cùng khi cho thấy được sự không phù hợp với bài toán Dự báo thời tiết yêu cầu tỉ lệ chính xác khá cao.

# Chương trình DEMO.

## Tổng quan chương trình

* Link demo: <https://danghieu19224.pythonanywhere.com/>

A qr code on a white background

Description automatically generated

* Giới thiệu sản phẩm demo:
  + Chương trình demo sử dụng các mô hình đã huấn luyện để dự đoán thời tiết dựa trên các thông tin nhập vào
  + Chương trình gồm form nhập thông tin thời tiết như: precipitation, temp\_max, temp\_min, wind.
  + Sau khi nhấn nút Dự đoán và chọ mô hình dự đoán. Kết quả sẽ được hiển thị lên giao diện

## Chạy demo chương trình

* Giao diện chương trình

A screenshot of a computer

Description automatically generated

* Chạy demo:
  + Sử dụng dòng dữ liệu sau để kiểm tra mô hình:



* + Kết quả dự đoán trên mô hình Perceptron:

A screenshot of a computer

Description automatically generated

* + Kết quả dự đoán trên mô hình ID3:

A screenshot of a computer

Description automatically generated

* + Kết quả dự đoán trên mô hình Neural Network:

A screenshot of a computer

Description automatically generated

* + Kết quả dự đoán trên mô hình Stacking:

A screenshot of a computer

Description automatically generated

# Nhận xét tổng thể

## Nhận xét tổng quan

Các mô hình đã triển khai khá hoàn chỉnh với quy trình tiền xử lý dữ liệu, chia tách thành các tập huấn luyện, xác thực và kiểm tra. Các kỹ thuật tối ưu hóa siêu tham số (Cross-Validation, GridSearch) và các thước đo đánh giá như ma trận nhầm lẫn, báo cáo phân loại và learning curve đã được sử dụng để đảm bảo tính tổng quát của mô hình. Tuy nhiên, một số hạn chế còn tồn tại:

* **Quá khớp trên các mô hình phức tạp**: Mô hình như MLP và Decision Tree dễ bị quá khớp trên các lớp phổ biến khi dữ liệu không cân bằng.
* **Hiệu suất của các mô hình đơn giản**: Perceptron và các mô hình đơn giản có thể không đủ mạnh để nhận diện tốt các mẫu phức tạp, dẫn đến hiệu suất chưa cao.
* **Chưa tận dụng tối đa kỹ thuật xử lý dữ liệu và regularization**: Các kỹ thuật như SMOTE cho dữ liệu mất cân bằng hoặc regularization mạnh hơn có thể giúp mô hình ổn định và tổng quát hơn.

## Hướng cải thiện

* **Xử lý dữ liệu và tăng cường dữ liệu (Data Augmentation)**
* Sử dụng các kỹ thuật tăng cường dữ liệu như SMOTE để cân bằng lại các lớp, đặc biệt hữu ích cho Decision Tree và Perceptron khi đối mặt với dữ liệu không cân bằng.
* Tăng cường đặc trưng bằng cách tạo thêm các biến mới từ các đặc trưng hiện có, giúp mô hình có thêm thông tin và cải thiện khả năng dự đoán.
* **Kỹ thuật Regularization mạnh hơn**
* Áp dụng dropout hoặc tăng mức l2 regularization cho MLP để giảm thiểu hiện tượng quá khớp.
* Với Perceptron, thử nghiệm thêm các kỹ thuật regularization như l1, l2, hoặc elastic net để cải thiện khả năng tổng quát hóa.
* **Áp dụng các mô hình mạnh hơn hoặc kết hợp nhiều mô hình (Ensemble)**
  + Kết hợp thêm các mô hình như Random Forest, Gradient Boosting trong mô hình Stacking để tăng tính đa dạng và chính xác trong dự đoán.
  + Thử nghiệm với các mô hình như XGBoost hoặc CatBoost, vốn có khả năng vượt trội với dữ liệu không cân bằng và phức tạp.
* **Điều chỉnh cấu trúc mạng cho MLP**
  + Thay đổi số lớp ẩn và số nơ-ron trong mỗi lớp, hoặc thử nghiệm với các hàm kích hoạt khác như ReLU và ELU để tối ưu hóa hiệu suất.
  + Sử dụng Early Stopping để tránh việc huấn luyện quá mức, đặc biệt hữu ích khi số epoch lớn.
* **Kết hợp thêm chỉ số đánh giá**
  + Bổ sung các chỉ số AUC và F1-score để có cái nhìn toàn diện hơn về hiệu suất mô hình, đặc biệt quan trọng trong các bài toán mất cân bằng lớp.

# KẾT LUẬN

1. **Nội dung đạt được.**

Trong báo cáo này, chúng em đã hoàn thành quy trình phân tích và dự báo thời tiết dựa trên các mô hình học máy, từ việc quan sát và xử lý dữ liệu, lựa chọn mô hình, đến đánh giá hiệu quả của các mô hình Perceptron, ID3, Neural Network và Stacking. Các bước thực hiện gồm:

1. **Quan sát và phân tích dữ liệu**: Xử lý các đặc trưng quan trọng như lượng mưa, nhiệt độ và tốc độ gió để dự đoán điều kiện thời tiết.
2. **Lựa chọn và huấn luyện mô hình**: Mô hình ID3 và Neural Network được chọn nhờ khả năng phân loại và dự báo dựa trên các đặc trưng phi tuyến tính, trong khi Stacking được dùng để kết hợp và tăng cường độ chính xác.
3. **Đánh giá mô hình**: Các chỉ số như ma trận nhầm lẫn và F1-score giúp chúng em đánh giá chính xác mức độ phân biệt của các mô hình giữa các loại thời tiết.

Qua quá trình này, chúng em đã hiểu rõ hơn về cách các mô hình hoạt động trên dữ liệu thời tiết phức tạp, đồng thời nhận thấy rằng việc xử lý dữ liệu và cân bằng lớp là quan trọng để tăng độ chính xác.

1. **Lời kết.**

Chúng em xin chân thành cảm ơn ThS. Trần Anh Đạt đã tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện bài tập lớn này. Những kiến thức và chỉ dẫn quý báu của thầy đã giúp chúng em không chỉ hoàn thành đề tài mà còn hiểu sâu hơn về các thuật toán học máy và cách ứng dụng vào các vấn đề thực tiễn. Đồng thời, chúng em cũng xin cảm ơn các thầy cô trong Khoa Công nghệ Thông tin đã trang bị cho chúng em những nền tảng kiến thức vững chắc, là tiền đề để chúng em phát triển các kỹ năng phân tích và xử lý dữ liệu.

Do kiến thức và kinh nghiệm còn hạn chế nên nhóm em không thể tránh khỏi những sai sót. Chúng em rất mong nhận được sự đóng góp ý kiến từ thầy cô và các bạn để có thể hoàn thiện hơn nữa trong các nghiên cứu sau. Một lần nữa, chúng em xin chân thành cảm ơn.