Phần 1: Perceptron

1. Cơ chế hoạt động của code

-LabelEncoder(): mã hóa cột phân loại (cột “weather”) thành các số nguyên để sử dụng được cho mô hình vì các thuật toán như Perceptron chỉ làm việc với dữ liệu số. Ví dụ: "rainy" có thể được mã hóa thành 0, "sunny" là 1,...

-train\_test\_split(): chia dữ liệu thành hai phần: tập huấn luyện và tập kiểm tra. Tập huấn luyện được sử dụng để huấn luyện mô hình, còn tập kiểm tra dùng để đánh giá hiệu quả dự đoán.

-Perceptron(): là một mô hình phân loại tuyến tính, sử dụng các trọng số để phân loại dữ liệu, nó dựa vào 1 hàm kích hoạt sign(w.X+b) để dự đoán nhãn của DL

-fit(): Hàm này huấn luyện mô hình bằng cách tìm ra các trọng số tối ưu để phân loại dữ liệu. Nó nhận vào hai tham số: dữ liệu đầu vào X\_train và nhãn y\_train. Mô hình cập nhật trọng số mỗi khi dự đoán sai dựa trên các quy tắc cập nhật như sau:

Trong đó là learning rate, y là nhãn thực tế, là nhãn dự đoán

-predict():Hàm này sử dụng các trọng số đã được học để dự đoán nhãn của dữ liệu mới dựa trên công thức:

Trong đó w là trọng số, b là bias và X là mẫu DL mới

-ConfusionMatrixDisplay(): Hiển thị ma trận nhầm lẫn, biểu diễn số lượng mẫu dự đoán đúng và sai cho từng nhãn.

2. Các tham số tham gia

-Accuracy (Độ chính xác): Đo lường tỷ lệ các dự đoán đúng trên tổng số các dự đoán

Trong đó: TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative).

Ví dụ: Nếu mô hình dự đoán đúng 80 trong tổng số 100 mẫu, độ chính xác sẽ là 80%

-Precision: tỷ lệ các dự đoná dương đúng trong tổng số các dự đoán dương

Ví dụ: Nếu có 10 dự đoán dương và 7 trong số đó là đúng, precision sẽ là 70%.

-Recall (Độ phủ): Tỷ lệ dự đoán dương đúng trên tổng các mẫu dương thật sự:

Ví dụ: Nếu có 20 mẫu dương thực sự và mô hình dự đoán đúng 15 mẫu, recall sẽ là 75%.

-F1-score: Trung bình điều hòa giữa precision và recall, giúp cân bằng giữa precision và recall

Ví dụ: Nếu precision = 70% và recall = 80%, F1-score sẽ là 74.6%.

3. Cách đánh giá phổ biến nhất của thuật toán Perceptron

Các phương pháp đánh giá phổ biến cho thuật toán Perceptron gồm:

-Confusion Matrix: Đây là một phương pháp trực quan hóa kết quả phân loại của mô hình. Ma trận nhầm lẫn cho biết số lượng mẫu đúng và sai trong từng lớp, giúp phân tích rõ ràng mô hình có bị nhầm lẫn ở đâu.

-Precision-Recall Curve: Đặc biệt hữu ích khi có sự mất cân bằng giữa các lớp, đồ thị này vẽ mối quan hệ giữa precision và recall ở các ngưỡng phân loại khác nhau.

-ROC Curve và AUC (Area Under Curve): Dùng để đánh giá khả năng phân loại của mô hình ở các ngưỡng khác nhau. Đồ thị này vẽ mối quan hệ giữa True Positive Rate (độ phủ) và False Positive Rate.

-Biểu đồ các trọng số: Đối với các bài toán đơn giản với 2 đặc trưng, có thể vẽ đồ thị của các trọng số và siêu phẳng tuyến tính để trực quan hóa cách Perceptron phân loại dữ liệu.

4. Các hàm loss function của Perceptron và hiện tượng overfitting

-Loss function trong Perceptron: Mặc dù Perceptron không có hàm loss phức tạp như các thuật toán khác, quá trình huấn luyện của Perceptron có thể được hiểu là tối thiểu hóa số lượng các mẫu dữ liệu bị phân loại sai. Khi dự đoán sai, trọng số được cập nhật để sửa sai.

Nếu số lượng các mẫu sai giảm về 0, quá trình huấn luyện dừng lại.

-Overfitting trong Perceptron:

+Perceptron có thể dẫn đến overfitting nếu dữ liệu huấn luyện không thể phân tách tuyến tính rõ ràng và mô hình cố gắng điều chỉnh trọng số để phân loại toàn bộ các mẫu huấn luyện một cách hoàn hảo. Điều này dẫn đến việc mô hình học thuộc nhiễu và không thể tổng quát hóa khi dự đoán trên dữ liệu mới.

+Để giảm thiểu overfitting, có thể sử dụng các phương pháp như:

* Regularization: Áp dụng thêm regularization để hạn chế việc cập nhật trọng số quá mức.
* Sử dụng tập dữ liệu lớn và đa dạng hơn.
* Cross-validation: Đánh giá mô hình bằng cách chia tập dữ liệu thành nhiều phần nhỏ và thực hiện huấn luyện, kiểm tra trên các tập con đó.

Phần 2: ID3

1. Cơ chế hoạt động trong CODE

Thuật toán ID3 (Iterative Dichotomiser 3) là một thuật toán cây quyết định dựa trên entropy và information gain. Trong đoạn code bạn đưa ra, thuật toán được triển khai bằng cách sử dụng DecisionTreeClassifier của thư viện sklearn, với criterion='entropy', tương ứng với thuật toán ID3. Dưới đây là cơ chế hoạt động của từng phần trong code:

* DecisionTreeClassifier(criterion='entropy'): Thuật toán xây dựng cây quyết định bằng cách tìm cách chia dữ liệu dựa trên tiêu chí giảm thiểu entropy lớn nhất (hay còn gọi là tối đa hóa information gain). Với mỗi node (nút) trong cây, nó tìm cột đặc trưng nào (ví dụ: "precipitation", "temp\_max", "temp\_min", "wind") sẽ chia dữ liệu tốt nhất để giảm bớt sự hỗn loạn của thông tin (entropy).
* fit(X\_train, y\_train): Hàm này tiến hành huấn luyện mô hình ID3. Cụ thể, nó tìm kiếm các node và phân tách dữ liệu theo đặc trưng, chọn ra đặc trưng nào giúp tối ưu giảm entropy (hoặc tối đa hóa thông tin). Khi không còn cải thiện được thông tin hoặc đạt được điều kiện dừng (như không còn đặc trưng để phân chia), nó sẽ dừng lại và tạo ra một node lá với nhãn dự đoán.
* predict(): Sau khi huấn luyện, mô hình cây sẽ được sử dụng để dự đoán nhãn của dữ liệu kiểm tra. Quá trình này thực hiện bằng cách đi từ gốc cây và duyệt qua các node cho đến khi đến node lá để xác định nhãn dự đoán.
* ConfusionMatrixDisplay và classification\_report(): Đây là các công cụ để đánh giá hiệu suất của mô hình. Confusion matrix sẽ hiển thị ma trận dự đoán đúng/sai của từng lớp, còn classification report cung cấp các chỉ số như precision, recall và F1-score.

2. Các tham số đánh giá phù hợp nhất với thuật toán ID3

Các tham số đánh giá phù hợp nhất để đánh giá thuật toán ID3 giống với Perceptron​

3. Cách đánh giá phổ biến nhất (mô hình hay sơ đồ) của thuật toán ID3

* Confusion Matrix: Hiển thị trực quan số lượng dự đoán đúng và sai của từng lớp, giúp bạn thấy rõ các lớp mà mô hình đang phân loại tốt hoặc kém.
* Classification Report: Cung cấp một cái nhìn tổng quan về các chỉ số đánh giá như precision, recall và F1-score cho từng lớp. Đây là công cụ phổ biến để đánh giá mô hình phân loại như ID3.
* Cây quyết định: Với thuật toán ID3, một cách đánh giá trực quan rất phổ biến là vẽ cây quyết định. Thư viện sklearn.tree cung cấp hàm plot\_tree() để vẽ cây, cho thấy cách mà các đặc trưng được phân chia và kết quả ở từng node.

4. Các hàm loss function và khi nào xảy ra overfitting trong ID3

* Loss function trong ID3: ID3 không trực tiếp sử dụng hàm loss như các thuật toán hồi quy, mà nó sử dụng khái niệm entropy và information gain để đánh giá độ "hỗn loạn" của dữ liệu trong mỗi lần phân tách.
  + Entropy: Đo lường mức độ không chắc chắn trong một tập hợp nhãn:

Trong đó, p\_i là xác suất của từng nhãn trong tập dữ liệu S.

* + Information Gain: Đo lường sự giảm entropy sau khi dữ liệu được phân tách theo một đặc trưng nào đó:

Trong đó, A là một đặc trưng, và S\_v là tập con của S tương ứng với giá trị v của đặc trưng A.

* Overfitting trong ID3: Thuật toán ID3 rất dễ bị overfitting nếu cây được phát triển quá sâu, tức là nó học thuộc các chi tiết và nhiễu trong dữ liệu huấn luyện thay vì học quy luật tổng quát. Một số nguyên nhân dẫn đến overfitting trong ID3 bao gồm:
  + Cây quyết định có quá nhiều node.
  + Không áp dụng pruning (cắt tỉa cây), tức là bỏ đi các nhánh cây không cần thiết.

Cách phòng tránh overfitting:

* Pruning: Cắt bỏ các nhánh cây không cung cấp nhiều thông tin (tức là không làm giảm entropy đáng kể).
* Giới hạn độ sâu của cây (max\_depth): Quy định cây không được phát triển sâu quá một mức nhất định.
* Sử dụng dữ liệu kiểm tra: Đánh giá mô hình trên dữ liệu chưa được sử dụng trong quá trình huấn luyện.

Phần 3: Neural Networks

**1. Cơ chế hoạt động trong CODE**

Thuật toán **Neural Networks** (ở đây là **Multilayer Perceptron - MLP**) trong đoạn code được triển khai như sau:

* **Đọc và xử lý dữ liệu**: Dữ liệu thời tiết từ file CSV được đọc và xử lý để loại bỏ các giá trị null, mã hóa thuộc tính phân loại weather thành số (bằng LabelEncoder), và chuẩn hóa các thuộc tính đầu vào (như precipitation, temp\_max, temp\_min, wind) bằng StandardScaler để đảm bảo rằng các giá trị có độ lớn tương tự nhau, phù hợp cho mạng neuron.
* **train\_test\_split**: Dữ liệu được chia ngẫu nhiên thành 70% cho huấn luyện và 30% cho kiểm tra. Điều này giúp đánh giá tính tổng quát của mô hình sau khi huấn luyện.
* **Xây dựng mô hình MLP**:
  + **MLPClassifier**: Mạng neuron được xây dựng với 2 tầng ẩn (hidden\_layer\_sizes=(100, 50)), tương ứng với 100 và 50 neuron cho mỗi lớp.
  + **max\_iter=1000**: Số vòng lặp tối đa cho việc huấn luyện. Mô hình sẽ dừng lại khi đạt mức hội tụ hoặc đạt đến số vòng lặp này.
  + **activation='relu'**: Hàm kích hoạt ReLU (Rectified Linear Unit) được sử dụng trong các neuron của lớp ẩn. Hàm ReLU trả về giá trị dương của input hoặc 0 nếu input âm.
  + **solver='adam'**: Thuật toán Adam là một phương pháp tối ưu hóa được sử dụng để điều chỉnh trọng số của mạng neuron trong quá trình huấn luyện.
* **Huấn luyện mô hình**: Mô hình được huấn luyện với clf.fit(X\_train, y\_train) sử dụng dữ liệu huấn luyện đã chia ở bước trước.
* **Dự đoán và đánh giá mô hình**: Mô hình dự đoán các nhãn cho tập dữ liệu kiểm tra (X\_test), sau đó các kết quả dự đoán được so sánh với giá trị thực tế để tính các chỉ số đánh giá như **accuracy**, **precision**, **recall**, và **F1-score** thông qua hàm classification\_report().

**2. Các tham số đánh giá phù hợp nhất, phổ biến nhất với thuật toán Neural Networks**

​

**3. Cách đánh giá phổ biến nhất (mô hình hay sơ đồ) của thuật toán Neural Networks**

* **Confusion Matrix (Ma trận nhầm lẫn)**: Đây là cách trực quan phổ biến nhất để đánh giá một mô hình phân loại như mạng neuron. Ma trận nhầm lẫn hiển thị số lượng dự đoán đúng và sai cho từng lớp.
* **Learning Curve (Đường cong học tập)**: Đường cong học tập cho biết sự thay đổi về độ lỗi của mô hình theo thời gian (số vòng lặp huấn luyện). Nếu mô hình đạt đến hội tụ và không cải thiện, thì đường cong học tập sẽ dừng lại.
* **Loss Curve (Đường cong lỗi)**: Đối với mạng neuron, một cách phổ biến khác để đánh giá quá trình huấn luyện là vẽ biểu đồ đường cong lỗi theo số epoch (vòng lặp). Đường cong lỗi giúp hiểu rõ hơn về quá trình tối ưu hóa và có thể phát hiện overfitting khi lỗi trên tập kiểm tra tăng lên trong khi lỗi trên tập huấn luyện tiếp tục giảm.

**4. Các hàm loss function và khi nào xảy ra overfitting**

**Loss Function trong Neural Networks:**

* **Cross-Entropy Loss (Log Loss)**: Đối với bài toán phân loại nhiều lớp, hàm loss được sử dụng thường là **Cross-Entropy Loss** (hay còn gọi là log loss), đo lường sự khác biệt giữa nhãn thực và nhãn dự đoán. 1

Trong đó:

* + N là số mẫu,
  + M là số lớp,
  + y\_{ij} là nhãn thực tế (một-hot encoding),
  + \hat{y}\_{ij} là xác suất dự đoán của lớp j cho mẫu i.

**Overfitting và cách phát hiện:**

**Overfitting** xảy ra khi mô hình quá phù hợp với dữ liệu huấn luyện, dẫn đến khả năng tổng quát kém khi làm việc với dữ liệu kiểm tra. Các dấu hiệu của overfitting bao gồm:

* **Loss trên tập kiểm tra tăng lên trong khi loss trên tập huấn luyện giảm**: Điều này có nghĩa là mô hình đang học quá chi tiết các mẫu trong tập huấn luyện, bao gồm cả nhiễu, thay vì học quy tắc tổng quát.
* **Sự khác biệt lớn giữa accuracy trên tập huấn luyện và tập kiểm tra**: Nếu accuracy trên tập huấn luyện rất cao nhưng trên tập kiểm tra rất thấp, đó là dấu hiệu của overfitting.

**Các biện pháp ngăn chặn overfitting:**

* **Regularization**: Áp dụng các kỹ thuật như **L2 regularization** (trong mạng neuron gọi là weight decay) hoặc **Dropout** để giảm bớt sự phụ thuộc vào các trọng số cụ thể.
* **Early Stopping**: Theo dõi lỗi trên tập kiểm tra và dừng quá trình huấn luyện sớm khi lỗi không còn giảm.
* **Tăng kích thước tập dữ liệu**: Cung cấp nhiều dữ liệu hơn để giúp mô hình học được các quy tắc tổng quát tốt hơn.
* **Giảm độ phức tạp của mạng**: Giới hạn số lớp ẩn hoặc số neuron trong mỗi lớp để tránh học quá chi tiết dữ liệu huấn luyện.