ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO ĐỒ ÁN: PHÂN LOẠI VĂN BẢN (TEXT CLASSIFICATION)

Môn học: Nhập môn xử lý ngôn ngữ tự nhiên

Giáo viên hướng dẫn: Nguyễn Hồng Bửu Long

Lê Thanh Tùng

Lương An Vinh

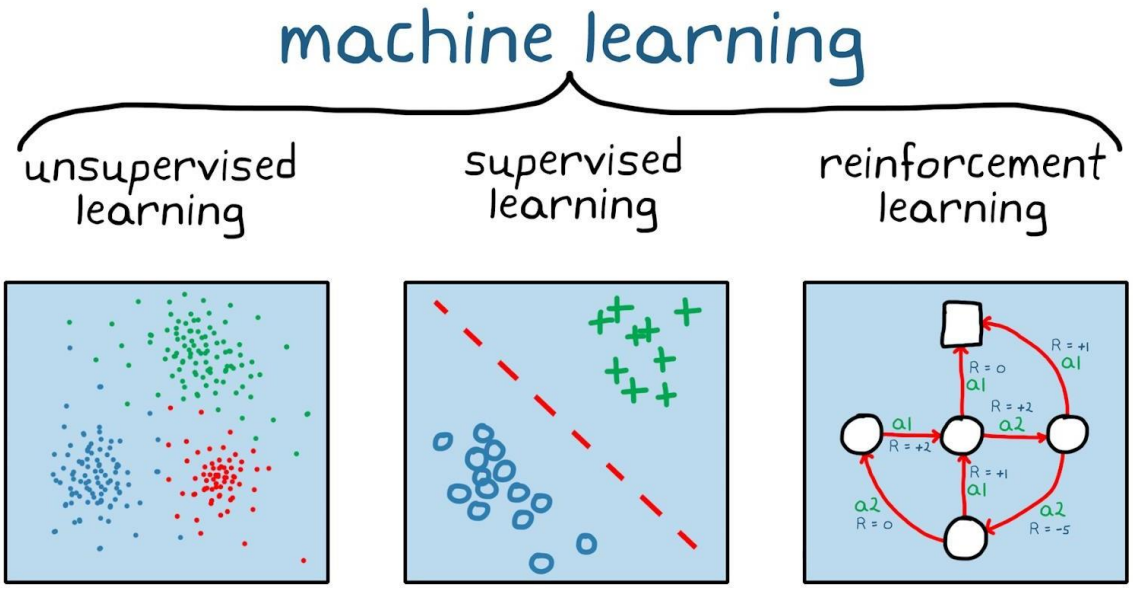
Thành viên:

21127056 – Lâm Thiều Huy

Mục lục

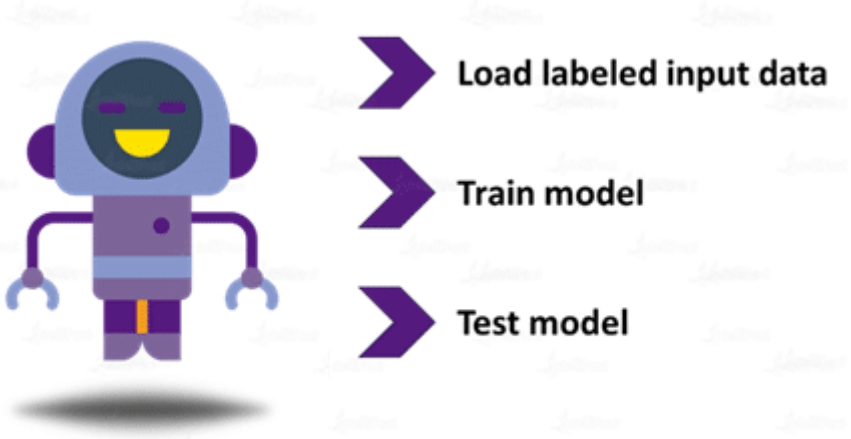
Phần Lý Thuyết:

1/ Sự khác nhau giữa học có giám sát và học không giám sát và học tăng cường và chúng khác nhau như thế nào trong phương pháp học tập?



Đầu tiên, để tìm ra sự khác nhau của từng loại cần phải hiểu được khái niệm của từng loạihọc.

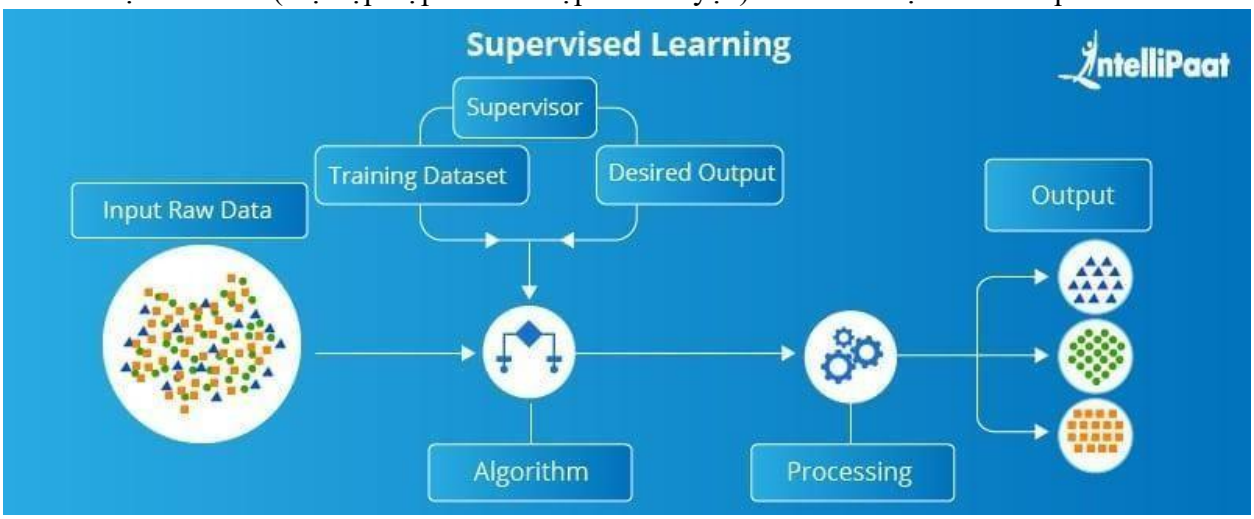
- **Supervised Learning (Học có giám sát):** trong học có giám sát, mô hình AI được đào tạo dựa trên những cái đầu vào được đưa vào nhất định và đầu ra dự kiến của nó, tức là nhãn đầu vào của nó. Mô hình tạo ra một phương trình ánh xạ dựa trên đầu vào và đầu ra, đồng thời dự đoán nhãn của đầu vào trong tương lai dựa trên phương trình ánh xạ đó.Ví dụ, giả sử chúng ta phải phát triển một mô hình có thể phân biệt giữa mèo và chó. Để đào tạo mô hình, chúng tôi đưa nhiều hình ảnh về mèo và chó vào mô hình vớ nhãn dán cho biết hình ảnh đó là của mèo hay chó. Mô hình cố gắng phát triển một phương trình giữa hình ảnh đầu vào và nhãn của chúng. Sau khi đào tạo, mô hình có thể dự đoán hình ảnh là của mèo hay chó ngay cả khi mô hình chưa nhìn thấy hình ảnh đó trước đó. Nói chung, cái đầu vào của học tập được giám sát được cung cấp dưới dạng **tập dữ liệu được dán nhãn**, một mô hình có thể học hỏi từ đó để đưa ra kết quả của vấn đề một cách dễ dàng.



Hình ảnh minh họa Supervised Learning

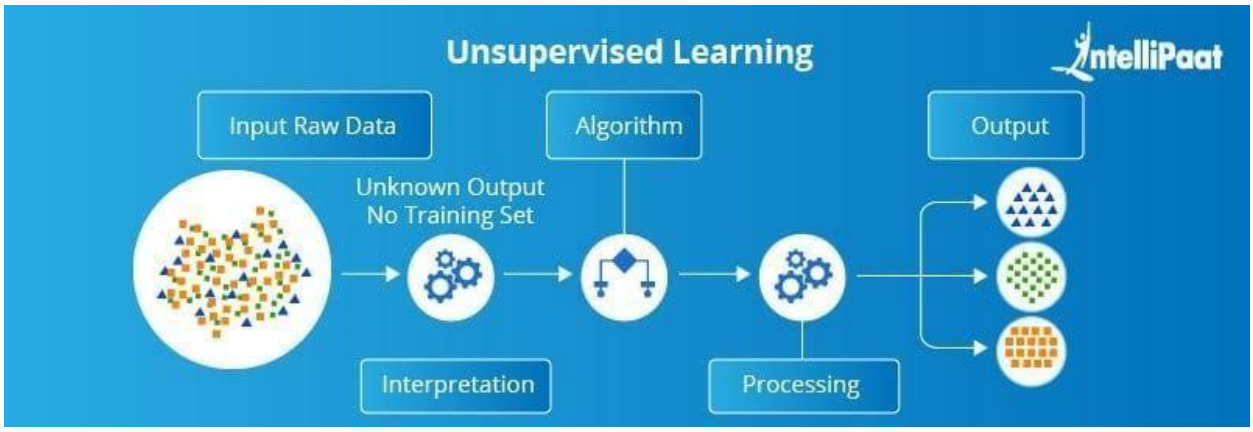
- **Supervised Learning (Học có giám sát):** giải quyết hai loại vấn đề - vấn đề phân loại và vấn đề hồi quy.
    - **Vấn đề phân loại:** Thuật toán này giúp dự đoán một giá trị rời rạc. Có thể coi dữ liệu đầu vào là thành viên của một lớp hoặc nhóm cụ thể. Ví dụ: lấy các bức ảnh của tập dữ liệu trái cây, mỗi bức ảnh được gắn nhãn là xoài, táo, v.v. Ở đây, thuật toán phải phân loại các hình ảnh mới thành bất kỳ danh mục nào trong số này. Ví dụ một số thuật toán: Naive Bayes Classifier, Support Vector Machines, Logistic Regression.
    - **Vấn đề hồi quy:** Những vấn đề này được sử dụng cho dữ liệu liên tục. Ví dụ: dự đoán giá một mảnh đất trong thành phố, dựa trên diện tích, vị trí, số phòng, v.v. Sau đó, dữ liệu đầu vào được gửi đến máy để tính giá đất theo các ví dụ trước. Ví dụ tên một số thuật toán: Linear Regression, Nonlinear Regression, Bayesian Linear Regression.
- **Cách làm việc của Supervised Learning (Học có giám sát):** Trong học có giám sát, các mô hình được đào tạo bằng cách sử dụng tập dữ liệu có nhãn, trong đó mô hình tìm hiểu

về từng loại dữ liệu. Sau khi quá trình đào tạo hoàn tất, mô hình sẽ được kiểm tra dựa trên dữ liệu kiểm tra (một tập hợp con của tập huấn luyện) và sau đó dự đoán kết quả đầu ra.

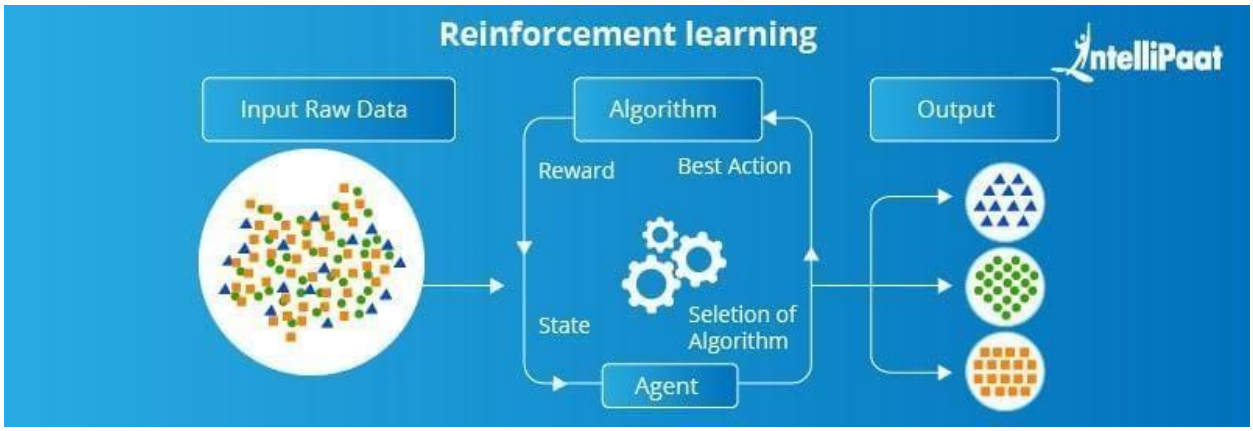


Hình ảnh minh họa cách làm việc của Supervised ML

- **Unsupervised learning (Học không có giám sát):** Trong học tập không giám sát, mô hình AI chỉ được đào tạo dựa trên đầu vào mà không có nhãn của chúng. Mô hình phân loại dữ liệu đầu vào thành các lớp có tính năng tương tự. Nhãn của đầu vào sau đó được dự đoán trong tương lai dựa trên sự giống nhau về các tính năng của nó với một trong các lớp. Ví dụ, có một bộ sưu tập các quả bóng màu đỏ và xanh và chúng ta phải phân loại chúng thành hai lớp. Giả sử tất cả các đặc điểm khác của quả bóng đều giống nhau ngoại trừ màu sắc của chúng. Mô hình cố gắng tìm ra những đặc điểm khác nhau giữa các quả bóng trên cơ sở cách mô hình có thể phân loại các quả bóng thành hai loại. Sau khi các quả bóng được phân thành hai loại tùy theo màu sắc của chúng, chúng ta sẽ có được hai cụm bóng, một cụm màu xanh và một cụm màu đỏ. Nói tóm lại, học không có giám sát không có tập dữ liệu được dán nhãn đầy đủ và rõ ràng. Học không giám sát là **học tự tổ chức**. Mục đích chính của nó là khám phá các mô hình cơ bản và dự đoán kết quả đầu ra. Về cơ bản, ở đây chúng tôi cung cấp cho máy dữ liệu và yêu cầu tìm kiếm các tính năng ẩn và phân cụm dữ liệu theo cách có ý nghĩa. Ví dụ: K – Means clustering, Neural Networks, Principal Component Analysis.
- **Cách làm việc của Unsupervised learning (Học không có giám sát):** Học không giám sát hoạt động bằng cách phân tích dữ liệu mà không có nhãn cho các cấu trúc ẩn bên trong nó và thông qua việc xác định mối tương quan cũng như các tính năng tương quan với hai mục dữ liệu. Nó đang được sử dụng để phân cụm, giảm kích thước, học tính năng, ước tính mật độ, v.v.



- **Reinforcement learning (Học tăng cường):** Trong học tăng cường, mô hình AI cố gắng thực hiện hành động tốt nhất có thể trong một tình huống nhất định để tối đa hóa cái kết quả đạt được. Mô hình học bằng cách nhận phản hồi về kết quả trong quá khứ của nó.Ví dụ: một robot được yêu cầu chọn một con đường giữa A và B. Ban đầu, robot chọn một trong hai con đường vì nó chưa có kinh nghiệm trong quá khứ. Robot nhận được phản hồi về con đường nó chọn và học hỏi từ phản hồi này. Lần tới khi robot gặp tình huống tương tự, nó có thể sử dụng phản hồi để giải quyết vấn đề. Nếu robot chọn đường B và nhận được phần thưởng, tức là phản hồi tích cực, thì lần này robot biết rằng nó phải chọn đường B để tối đa hóa phần thưởng. Nói tóm lại, nó không dựa trên việc học có giám sát hay học không giám sát. Hơn nữa, ở đây các thuật toán học cách tự phản ứng với môi trường. Nó đang phát triển nhanh chóng và hơn nữa còn tạo ra nhiều thuật toán học tập khác nhau. Các thuật toán này rất hữu ích trong lĩnh vực Robotics, Gaming, v.v.
- **Cách làm việc của Reinforcement learning (Học tăng cường):** Vấn đề Học tăng cường liên quan đến việc một tác nhân khám phá một môi trường chưa xác định để đạt được mục tiêu. RL dựa trên giả thuyết rằng tất cả các mục tiêu có thể được mô tả bằng cách tối đa hóa phần thưởng tích lũy dự kiến. Tác nhân phải học cách cảm nhận và làm xáo trộn trạng thái của môi trường bằng cách sử dụng các hành động của mình để nhận được phần thưởng tối đa.



Bảng so sánh các đặc điểm Supervised ML, Unsupervised ML, Reinforcement ML

Tiêu chí	Supervised ML	Unsupervised ML	Reinforcement ML
Định nghĩa	Học bằng cách sử dụng những dữ liệu được gắn nhãn	Được đào tạo bằng những dữ liệu **KHÔNG** có gắn nhãn mà không cần bất kì hướng dẫn nào	Hoạt động trên tương tác với môi trường
Kiểu dữ liệu	Dữ liệu được gắn nhãn	Dữ liệu không được gắn nhãn	Dữ liệu không cần định nghĩa/xác định trước
Vấn đề giải quyết	Phân Loại && Hồi Quy	Liên Kết và Phân Cụm	Khai Thác và Thăm Dò
Sự giám sát	Giám sát bổ sung	Không có	Không có
Thuật toán	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Mục đích	Tính toán kết quả	Khám phá các mẫu cơ bản	Học hỏi thông qua hàng loạt những hoạt động (quá trình học tập và tương tác với môi trường)
Ứng dụng	Đánh giá rủi ro, Dự báo doanh số bán hàng	Hệ thống đề xuất, phát hiện những điều bất thường	Xe tự lái, Trò chơi, Chăm sóc sức khỏe

⇨ Tóm lại: Học có giám sát, mục tiêu là tạo ra công thức dựa trên các giá trị đầu vào và đầu ra. Trong học không giám sát là tìm ra mối liên hệ giữa các giá trị đầu vào và gom nhóm chúng lại. Trong học tập tăng cường, một tác nhân học thông qua phản hồi trễ (delayed feedback) bằng cách tương tác với môi trường.

2/ Khi đánh giá hiệu suất của mô hình phân loại văn bản, các chỉ số đánh giá thường được sử dụng là gì và làm thế nào chúng ta có thể đánh giá hiệu quả của nó?

- Các chỉ số phổ biến thường được sử dụng để đánh giá hiệu suất của mô hình phân loại, đó là:
    1. **Độ chính xác (Accuracy):** Tỉ lệ dự đoán đúng so với tổng số dự đoán.
    2. **Độ chính xác chính (Precision):** Tỉ lệ dự đoán đúng positive so với tổng số positive dự đoán (precision = true positives / (true positives + false positives)).
    3. **Recall (Độ nhạy hoặc Tỉ lệ True Positive):** Tỉ lệ dự đoán đúng positive so với tổng số positive thực tế (recall = true positives / (true positives + false negatives)).
    4. **Điểm F1 (F1 Score):** Trung bình điều hòa giữa precision và recall, cung cấp sự cân bằng giữa hai chỉ số này (F1 score = 2 * ((precision * recall) / (precision + recall))).
    5. **Area under the receiver operating characteristic curve (AUC-ROC):** Một chỉ số đánh giá sự đối đầu giữa tỷ lệ true positive và tỷ lệ false positive ở các ngưỡng khác nhau. Đặc biệt hữu ích cho các vấn đề phân loại nhị phân.

6. **Confusion Matrix (Ma trận nhầm lẫn):** Confusion Matrix hiển thị số lượng True Positive, True Negative, False Positive và False Negative. Nó cung cấp cái nhìn chi tiết về hiệu suất của mô hình trên từng lớp.
7. **Đánh giá thời gian thực hiện (Runtime Metrics):** Nếu như độ chính xác cũng như các yếu tố khác giữa các thuật toán hoặc các mô hình đều không có sự chênh lệc, chúng ta có thể chọn tiêu chí thời gian huấn luyện của từng mô hình để đưa ra sự lựa chọn thuật toán/mô hình nào tối ưu hơn.

- Để đánh giá hiệu quả của mô hình:
    1. **Chuẩn Bị Dữ Liệu Kiểm Tra:** Sử dụng một tập dữ liệu kiểm tra độc lập chưa từng được mô hình thấy để đảm bảo tính khách quan của quá trình đánh giá.
    2. **Dự Đoán và Thu Thập Kết Quả:** Sử dụng mô hình để dự đoán trên tập dữ liệu kiểm tra và thu thập kết quả.
    3. **Tính Toán Chỉ Số Đánh Giá:** Sử dụng các chỉ số như Accuracy, Precision, Recall và F1 Score để đánh giá hiệu suất của mô hình.
    4. **Confusion Matrix và Chi Tiết Theo Lớp:** Phân tích Confusion Matrix để xem xét các loại lỗi mà mô hình có thể gặp và sử dụng Classification Report để đánh giá chi tiết theo từng lớp (nếu là bài toán phân loại đa lớp).
    5. **ROC và Precision-Recall Curve (Nếu Có):** Nếu là bài toán phân loại nhị phân, xem xét ROC Curve để đánh giá tỷ lệ true positive và false positive ở các ngưỡng quyết định khác nhau. Precision-Recall Curve cung cấp cái nhìn về cân bằng giữa precision và recall.
    6. **Đánh Giá Đồng Nhất và Ổn Định:** Thực hiện kiểm định chéo để đánh giá sự ổn định và đồng nhất của mô hình trên các tập dữ liệu con khác nhau.
    7. **So Sánh với Baseline:** So sánh hiệu suất của mô hình với một mô hình cơ bản hoặc phương pháp ngẫu nhiên để đảm bảo rằng mô hình của bạn mang lại cải tiến thực sự.
    8. **Xem Xét Ngưỡng Quyết Định (Threshold):** Nếu cần, xem xét việc điều chỉnh ngưỡng quyết định để thay đổi trade-off giữa precision và recall.
    9. **Lưu Ý Điểm Mạnh và Yếu:** Ghi chép các điểm mạnh và điểm yếu của mô hình, giúp xác định những phần cần cải thiện.
    10. **Lặp Lại và Tối Ưu Hóa (Nếu Cần):** Nếu mô hình không đạt được hiệu suất mong muốn, lặp lại quá trình huấn luyện, điều chỉnh tham số hoặc thực hiện các biện pháp tối ưu hóa khác.

3/ Trong bối cảnh phân loại văn bản, những kỹ thuật nào có thể được sử dụng để xử lý dữ liệu không cân đối và giải quyết thách thức liên quan đến mất cân đối lớp?

Trong bối cảnh phân loại văn bản, việc xử lý dữ liệu không cân đối là một thách thức thường gặp. Có thể sử dụng nhiều kỹ thuật để giải quyết vấn đề này:

Kỹ thuật Lấy Mẫu Lại:Tăng cường lấy mẫu cho lớp thiểu số: Kỹ thuật này bao gồm việc tăng số lượng mẫu trong lớp thiểu số để cân xứng với lớp đa số. Điều này có thể được thực hiện bằng cách sao chép mẫu hiện có hoặc tạo ra mẫu tổng hợp mới, ví dụ như sử dụng kỹ thuật Synthetic Minority Over-sampling Technique (SMOTE). SMOTE tạo ra các mẫu tổng hợp mới, tương tự nhưng không giống hệt với các mẫu thiểu số hiện có bằng cách nội suy giữa nhiều mẫu thiểu số.

Giảm mẫu lớp đa số: Cách tiếp cận này giảm số lượng mẫu trong lớp đa số để phù hợp với lớp thiểu số. Phương pháp đơn giản nhất là giảm mẫu ngẫu nhiên, loại bỏ mẫu từ lớp đa số một cách ngẫu nhiên. Tuy nhiên, điều này có thể dẫn đến mất dữ liệu quan trọng. Các phương pháp tiên tiến hơn cố gắng chỉ loại bỏ những mẫu ít thông tin hoặc dư thừa.

Chỉnh sửa Trọng lượng Lớp: Trong nhiều thuật toán học máy, đặc biệt là những thuật toán dựa trên tối ưu hóa (như hồi quy logistic hoặc máy vector hỗ trợ), có thể điều chỉnh hàm chi phí để gán trọng lượng cao hơn cho lớp thiểu số. Điều này có nghĩa thuật toán sẽ phạt nặng hơn khi phân loại sai mẫu của lớp thiểu số so với lớp đa số, hiệu quả chuyển sự chú ý của nó về phía lớp thiểu số.

Phương pháp Ensemble: Các phương pháp này, như bagging và boosting, kết hợp nhiều mô hình để đạt được hiệu suất tốt hơn. Trong các bộ dữ liệu không cân đối, phương pháp ensemble có thể được thiết kế để tập trung nhiều hơn vào lớp thiểu số. Ví dụ, trong bagging, mỗi mô hình trong ensemble có thể được đào tạo trên một tập con cân đối khác nhau được tạo ra thông qua lấy mẫu lại. Boosting có thể được điều chỉnh để chú ý nhiều hơn đến các trường hợp mà các mô hình trước đó phân loại sai, thường là từ lớp thiểu số.

Kỹ thuật Phát hiện Bất thường: Đối với các bộ dữ liệu cực kỳ không cân đối, nơi mà lớp thiểu số rất hiếm, việc xử lý vấn đề như phát hiện bất thường có thể hiệu quả. Ở đây, trọng tâm không phải là phân biệt giữa các lớp mà là phát hiện các trường hợp hiếm của lớp thiểu số.

Điều chỉnh Thuật toán: Một số thuật toán tự nhiên tốt hơn trong việc xử lý dữ liệu không cân đối. Ví dụ, các phương pháp dựa trên cây như cây quyết định và rừng ngẫu nhiên có thể xử lý mất cân đối vì chúng phân chia không gian theo cách mà chúng có thể cô lập mẫu của lớp thiểu số trong một số nhánh của cây.

Sử dụng Chỉ số Đánh giá: Việc sử dụng các chỉ số đánh giá phản ánh đúng mức độ hiệu suất mô hình trên các bộ dữ liệu không cân đối là cực kỳ quan trọng. Độ chính xác có thể gây hiểu lầm vì nó có thể cao do phần lớn lớp đa số được dự đoán chính xác. Các chỉ số như F1-score, G-mean (trung bình hình học của độ chính xác và độ nhạy) và AUC-ROC (Diện tích dưới đường cong ROC) có thông tin hơn vì chúng tính đến cả độ chính xác và độ nhạy, và do đó hiệu suất trên cả hai lớp.

4: Máy vectơ hỗ trợ (SVM) được thiết kế chủ yếu để phân loại nhị phân. Bạn có thể giải thích cách tiếp cận được sử dụng để mở rộng SVM nhằm xử lý các vấn đề phân loại nhiều lớp và cách nó cho phép chúng phân loại dữ liệu thành nhiều lớp không?

-Mô hình Máy Vectơ Hỗ trợ (SVM) ban đầu được thiết kế cho bài toán phân loại nhị phân. Để mở rộng SVM cho bài toán phân loại đa lớp, có hai phương pháp chính:

1. **Phương pháp Một-đối-một (One-vs-One):** Trong phương pháp này, một mô hình SVM được huấn luyện cho mỗi cặp lớp. Với N lớp, có N(N−1)/2N(N−1)/2 mô hình SVM được huấn luyện. Khi phân loại, mỗi mô hình bỏ phiếu cho một lớp, và lớp nhận được số phiếu cao nhất là lớp dự đoán.Ví dụ, nếu có ba lớp (A, B, và C), ba bộ phân loại SVM sẽ được huấn luyện: một để phân biệt A với B và C, một để phân biệt B với A và C, và một để phân biệt C với A và B. Khi phân loại một điểm dữ liệu mới, tất cả các bộ phân loại sẽ được áp dụng, và lớp tương ứng với bộ phân loại có độ tin cậy cao nhất sẽ được chọn.

2. **Phương pháp Một-đối-tất cả (One-vs-All):** Mỗi lớp được xem là lớp dương và tất cả các lớp còn lại là lớp âm. Một mô hình SVM được huấn luyện cho mỗi lớp với cách tiếp cận này. Khi phân loại, mô hình với độ tin cậy cao nhất (tính theo khoảng cách từ mẫu tới đường biên quyết định) xác định lớp của mẫu.

Kết luận :

Cả hai phương pháp đều có ưu và nhược điểm. OvA đơn giản hơn và nhanh hơn khi huấn luyện vì chỉ cần xây dựng N bộ phân loại cho N lớp. Tuy nhiên, nó có thể không hiệu quả khi số lượng lớp tăng lên. Ngược lại, OvO cần nhiều bộ phân loại hơn, nhưng từng bộ phân loại chỉ cần xem xét một phần nhỏ của toàn bộ dữ liệu, có thể làm cho việc huấn luyện chính xác hơn trong một số trường hợp.Tùy thuộc vào bản chất của dữ liệu và số lượng lớp, một trong hai phương pháp này hoặc kết hợp của chúng có thể được sử dụng để giải quyết vấn đề phân loại nhiều lớp bằng SVM.

-Có một số phương pháp khác để mở rộng máy vectơ hỗ trợ (SVM) cho phân loại đa lớp . Dưới đây là một số phương pháp phụ:

1. **Phân loại Định hướng Đa lớp (Directed Acyclic Graph SVM, DAG SVM):** Phương pháp này kết hợp cách tiếp cận Một-Với-Một và áp dụng một đồ thị có hướng không chu kỳ (DAG) để thực hiện phân loại. Đối với N lớp, DAG SVM sử dụng N*(N-1)/2 bộ phân loại như trong OvO, nhưng trong quá trình phân loại, nó sử dụng một cấu trúc DAG để loại bỏ các lớp một cách hiệu quả, dẫn đến quyết định cuối cùng chỉ sau N-1 so sánh.

2. **Phân loại Dựa trên Cảm nhận (Perceptron-based Methods):** Các phương pháp này sử dụng các mô hình như máy vectơ hỗ trợ chuẩn hóa (SVM) hoặc perceptron để tạo ra một không gian đặc trưng mới, nơi việc phân loại đa lớp có thể thực hiện một cách hiệu quả hơn.

3.Phương pháp Đa lớp Kernel: Trong cách tiếp cận này, một hàm nhân (kernel) đa lớp được sử dụng để biến đổi không gian đặc trưng sao cho phân loại đa lớp trở nên hiệu quả hơn. Cách tiếp cận này cố gắng tận dụng ưu điểm của SVM trong việc tìm ra ranh giới quyết định tốt trong không gian đặc trưng cao chiều.

4. SVM Đa lớp Tích hợp (Integrated Multiclass SVM): Đây là một biến thể của SVM mà thay vì xây dựng nhiều bộ phân loại riêng lẻ, một bộ phân loại đa lớp duy nhất được xây dựng. Nó tìm cách tối ưu hóa một hàm mục tiêu tổng thể, có thể xem xét cả các lớp cùng một lúc.

Kết luận :

Mỗi phương pháp có những đặc điểm và ứng dụng riêng, và lựa chọn phụ thuộc vào bộ dữ liệu cụ thể và các yêu cầu của vấn đề. Trong thực tế, lựa chọn phương pháp phù hợp đòi hỏi phải xem xét cả về hiệu suất tính toán và độ chính xác của mô hình.

5/ Bạn định nghĩa kernel trong thuật toán SVM như thế nào? Hơn nữa, bạn có thể giới thiệu một số hàm kernel được cung cấp trong scikit-learn và giải thích chi tiết về các biến thể giữa chúng không?

Trong thuật toán SVM, hàm nhân là một công cụ quan trọng cho phép SVM hoạt động trong không gian đa chiều. Hàm nhân biến đổi dữ liệu từ không gian đầu vào sang không gian đặc trưng nhiều chiều hơn, giúp phân tách dữ liệu tốt hơn. Các hàm nhân phổ biến bao gồm:

1. **Hàm Nhân Tuyến Tính (Linear Kernel): $K(x,y)=x\cdot y$.** Đơn giản và nhanh chóng, thích hợp cho dữ liệu có thể tách tuyến tính.

2. **Hàm Nhân Đa Thức (Polynomial Kernel): $K(x,y)=(gamma\cdot x\cdot y+coef0)degree$**. Có khả năng mô hình hóa sự tương quan phi tuyến tính giữa các đặc trưng.

3. **Hàm Nhân RBF (Radial Basis Function Kernel): $K(x,y)=\exp(-gamma\cdot||x-y||2)$.** Rất mạnh mẽ trong việc xử lý dữ liệu phi tuyến tính.

4. **Hàm Nhân Sigmoit (Sigmoid Kernel): $K(x,y)=\tanh(gamma\cdot x\cdot y+coef0)$.** Lấy cảm hứng từ hàm kích hoạt neuron trong mạng nơ-ron.

Phần báo cáo kết quả khảo sát

Đối với phần code, nhóm em đã cấu hình cho lượng tỉ lệ positive, negative 50/50 như thầy đã yêu cầu, cũng như cũng có chia tỉ lệ nhiều dataset khác nhau, chi tiết được cung cấp ở mỗi thuật toán phía dưới.

Ban đầu, tỉ lệ phân bố các nhãn trong tập dataset con vừa trích xuất, đa số trong các lần nhóm generate sẽ ra được label '1': 50.21% và Label '0': '49,79%' (dựa trên tổng số samples là 10000). Sau khi thực hiện cấu hìn để chọn số lượng sample có số nhãn cân bằng thì số lượng mẫu cân bằng cho mỗi nhãn là 4979, tỉ lệ nhãn dán cũng được chia đều 50% cho label '1', label '0'

```
Số lượng mẫu cân bằng cho mỗi nhãn:
1    4979
0    4979
Name: label, dtype: int64

Tỉ lệ dán nhãn:
Label '1': 50.00%
Label '0': 50.00%
```

Thuật toán Decision Tree

Trong thời gian em tìm hiểu, cũng như khảo sát thuật toán thì em có tìm hiểu chung về phần lý thuyết của thuật toán cây quyết định. Đầu tiên là, về những mặt lợi em thấy là nổi bật khi sử dụng thuật toán cây quyết định là:

- Cây quyết định dễ diễn giải và dễ hình dung
- Nó có thể dễ dàng nắm bắt được các mẫu phi tuyến tính
- Nó có thể được sử dụng cho kỹ thuật tính năng như dự đoán các giá trị còn thiếu, phù hợp cho việc lựa chọn biến.
- Nó yêu cầu người dùng ít xử lý trước dữ liệu hơn, chẳng hạn như không cần chuẩn hóa các cột.

Về những mặt bất lợi:

- Nhạy cảm với dữ liệu nhiễu. Nó có thể phù hợp với noisy data (tạm dịch là dữ liệu ồn ào).
- Sự thay đổi nhỏ (hoặc phương sai) trong dữ liệu có thể dẫn đến cây quyết định khác nhau. Điều này có thể được giảm bớt bằng các thuật toán đóng bao và tăng cường.
- Cây quyết định bị sai lệch với tập dữ liệu mất cân bằng, vì vậy nên cân bằng tập dữ liệu trước khi tạo cây quyết định.

Câu hỏi tiếp theo em tìm hiểu đó chính là làm sao để cây quyết định biết cách để đưa ra một quyết định. Thuật toán sử dụng một số cách khác nhau để chia tập dữ liệu thành một loạt các quyết định. Một trong những cách đó là phương pháp đo Gini Impurity. Vậy thì Gini Impurity là gì ? Gini Impurity đề cập đến phép đo khả năng phân loại không chính xác một phiên bản mới của biến

ngẫu nhiên nếu phiên bản đó được phân loại ngẫu nhiên theo sự phân bổ nhãn lớp từ tập dữ liệu. Công thức:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Theo em tìm hiểu được thì tạp chất Gini được giới hạn thấp hơn bằng 0, nghĩa là giá trị càng gần 0 thì cái nó càng ít tạp chất

Trong thuật toán Decision Tree của scikit-learn, có một số siêu tham số quan trọng, việc điều chỉnh có thể ảnh hưởng đến việc phân loại văn bản. Dưới đâ là một số siêu tham số quan trọng và tác động của chúng:

1. max_depth: int, default=None (theo scikit-learn): Đây là độ sâu tối đa của cây quyết định. Nếu giảm max_depth, cây sẽ trở nên ngắn hơn và ít phức tạp hơn, có thể giảm nguy cơ quá mức điều chỉnh (overfitting).
2. min_samples_split int or float, default=2: Số lượng mẫu tối thiểu cần để một nút có thể chia thành nút con mới. Nếu giảm giá trị này, cây có thể trở nên rất sâu và phức tạp, có thể dẫn đến overfitting.
3. min_samples_leaf int or float, default=1: Số lượng mẫu tối thiểu cần để một lá có thể tồn tại. Giảm giá trị này có thể dẫn đến lá nhỏ và có thể gây overfitting.
4. max_features int, float or {“auto”, “sqrt”, “log2”}, default=None: Số lượng đặc trưng tối đa được xem xét cho việc tìm kiếm phân chia tốt nhất. Giảm giá trị này có thể làm giảm độ phức tạp của cây và nguy cơ overfitting.
5. criterion{“gini”, “entropy”, “log_loss”}, default=”gini”: Xác định phương pháp đo lường chất lượng của phân chia. "gini" sử dụng chỉ số Gini và "entropy" sử dụng độ thuần khiết thông tin (entropy).
6. random_state: Điều này là giống như một seed cho tạo số ngẫu nhiên. Nếu bạn cung cấp một giá trị cụ thể, mô hình sẽ học theo cách nhất định và sẽ không thay đổi kết quả nếu chạy lại với cùng một giá trị random_state.

Khi khảo sát và so sánh thuật toán Decision Tree, có một số tiêu chí quan trọng bạn có thể xem xét:

1. Độ chính xác (Accuracy): Đánh giá khả năng phân loại chính xác của mô hình trên tập dữ liệu kiểm thử. Đây là một tiêu chí quan trọng, nhưng cần lưu ý rằng nó không phản ánh đầy đủ thông tin nếu tỷ lệ các lớp trong tập dữ liệu là chênh lệch.

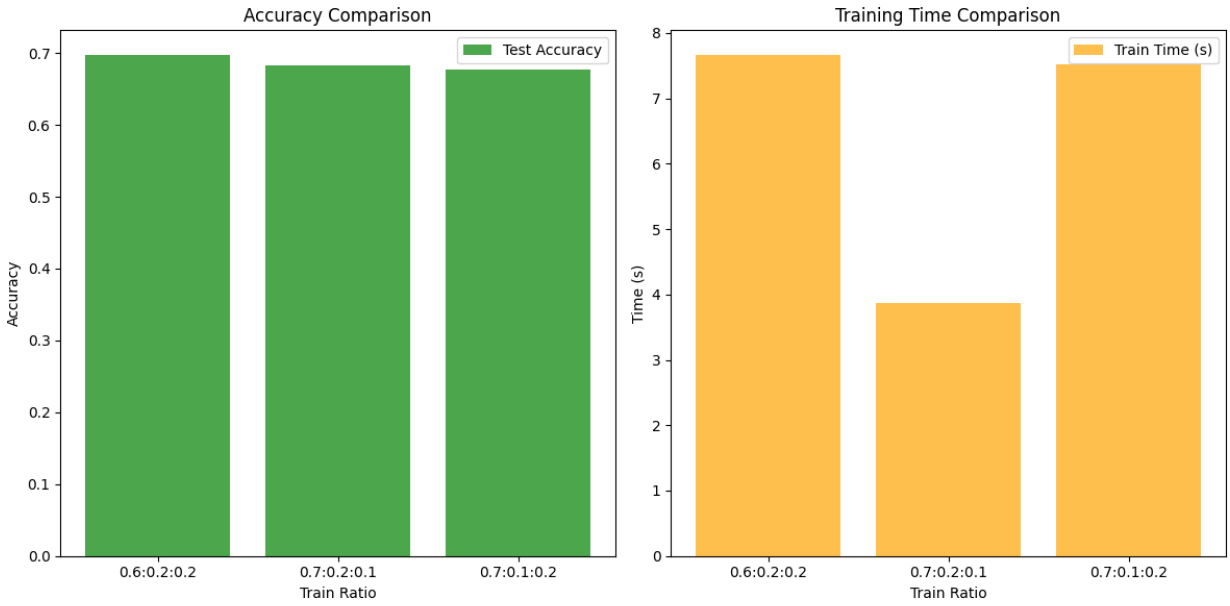
2. Thời gian huấn luyện và dự đoán: Đánh giá thời gian cần thiết để huấn luyện mô hình và thời gian dự đoán trên tập kiểm thử. Đối với các tập dữ liệu lớn, thời gian huấn luyện và dự đoán có thể trở thành yếu tố quan trọng.
3. Tinh chỉnh siêu tham số (Hyperparameter Tuning): Thử nghiệm và so sánh hiệu suất của mô hình trên các giá trị khác nhau của siêu tham số như max_depth, min_samples_split, min_samples_leaf, và max_features.
4. Ổn định của mô hình: Kiểm tra sự ổn định của mô hình trước các biến động nhỏ trong dữ liệu.

1.1/ Về sự phân bố tỉ lệ trong dataset

Khi thay đổi, tỉ lệ phân bố bên trong dataset (train/dev/test). Em chia làm 3 trường hợp phân bố tỉ lệ dữ liệu là 6/2/2, 7/2/1, 7/1/2. Về mặt so sánh độ chính xác của thuật toán ổn định, không có quá nhiều sự chênh lệch về kết quả của độ chính xác.

- Nếu dữ liệu huấn luyện ít: Nếu tập huấn luyện quá nhỏ, có thể xảy ra hiện tượng "underfitting" do mô hình không học được đủ thông tin từ dữ liệu. Mô hình có thể không đủ phức tạp để biểu diễn các mô hình phức tạp của dữ liệu thực tế.
- Dữ liệu kiểm thử ít: Nếu tập kiểm thử quá nhỏ, độ chính xác trên nó có thể không phản ánh chính xác thực sự của mô hình trên dữ liệu mới.

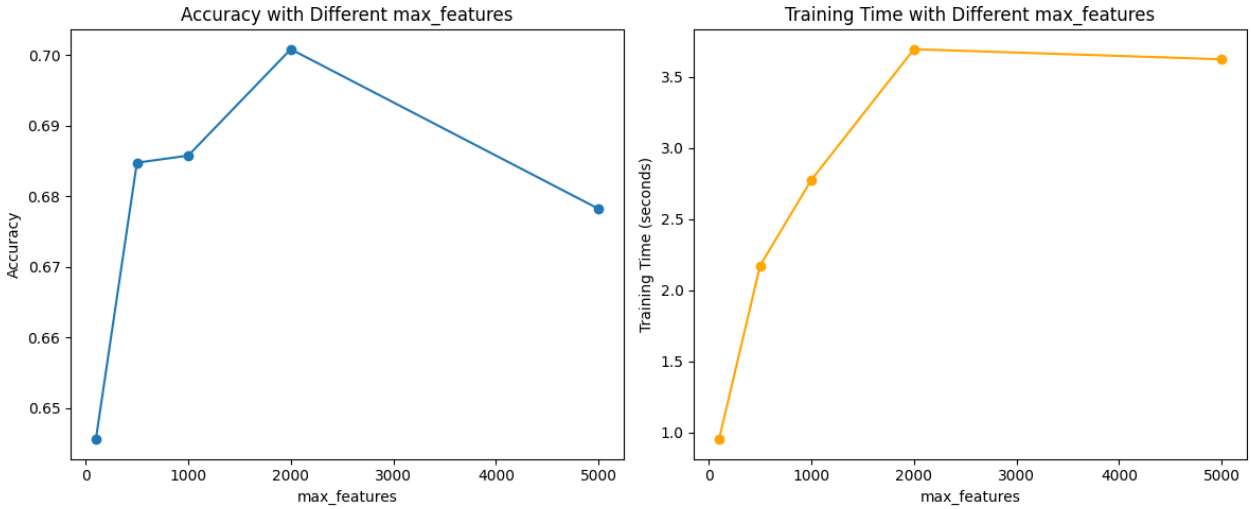
⇒ Kết luận: Để đảm bảo một ước lượng đáng tin cậy về hiệu suất của mô hình, quan trọng để sử dụng các tập huấn luyện, phát triển, và kiểm thử có kích thước đủ lớn để đại diện cho sự phân bố của dữ liệu thực tế.



1.2/Max Features

	max_features	accuracy	training_time
0	NaN	0.682731	5.852246

1	100.0	0.645582	0.952978
2	500.0	0.684739	2.174210
3	1000.0	0.685743	2.776768
4	2000.0	0.700803	3.695105
5	5000.0	0.678213	3.624446

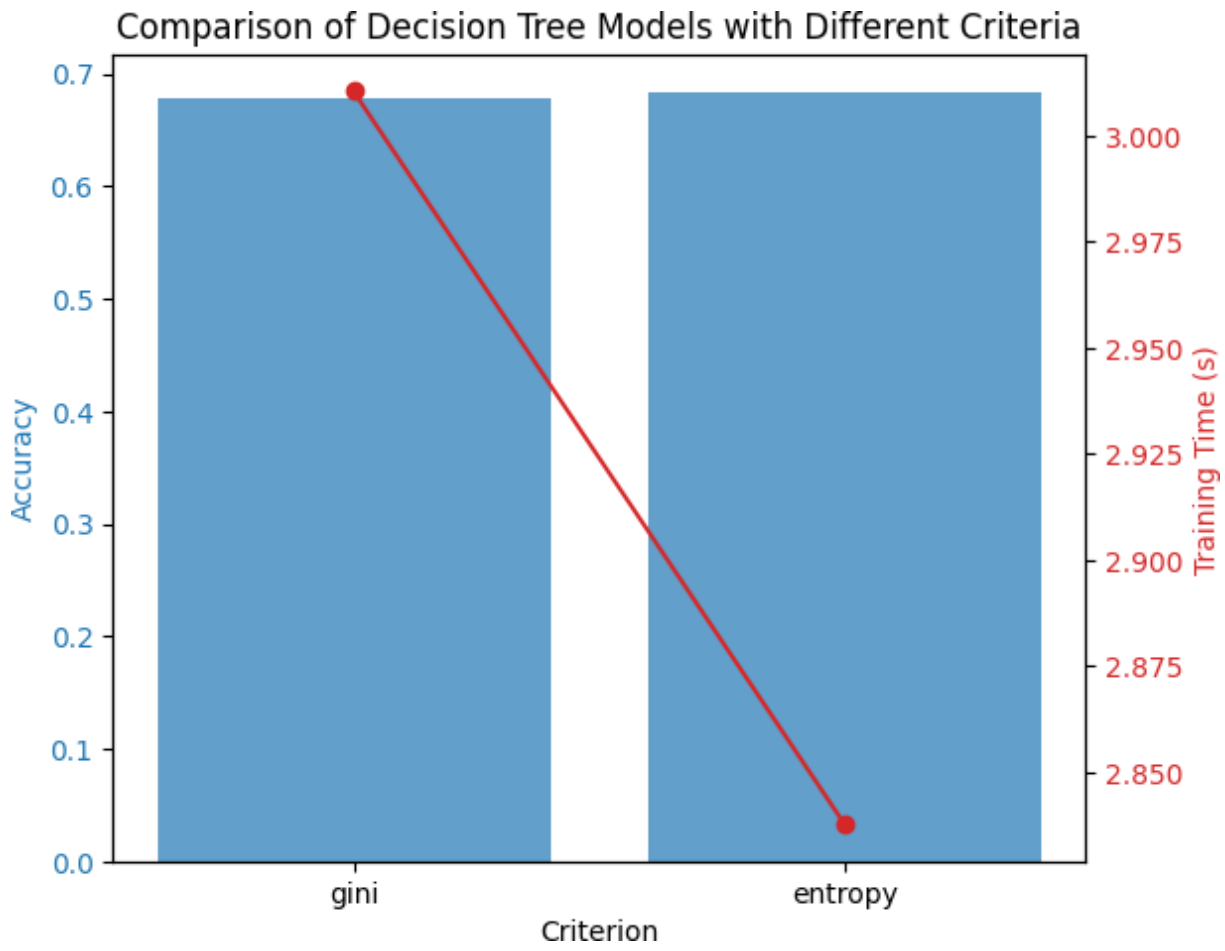


Nhận xét:

- Nhìn chung, độ chính xác của thuật toán đồng bộ với thời gian huấn luyện dữ liệu với các cùng một số lượng max_features. Trong đó, ở max_features có gì trị là 2000, đạt được độ chính xác cao nhất song đó thời gian huấn luyện cũng chỉ cao hơn một xíu so với lại max_features là 5000. Vì vậy, theo những thông số đã khảo sát phía trên với siêu tham số max_features thì, 2000 max_features có lẽ là sự lựa chọn tối ưu cho thuật toán.

1.3/ Criterion

	criterion	accuracy	training_time
0	gini	0.678213	3.010369
1	entropy	0.682731	2.837729

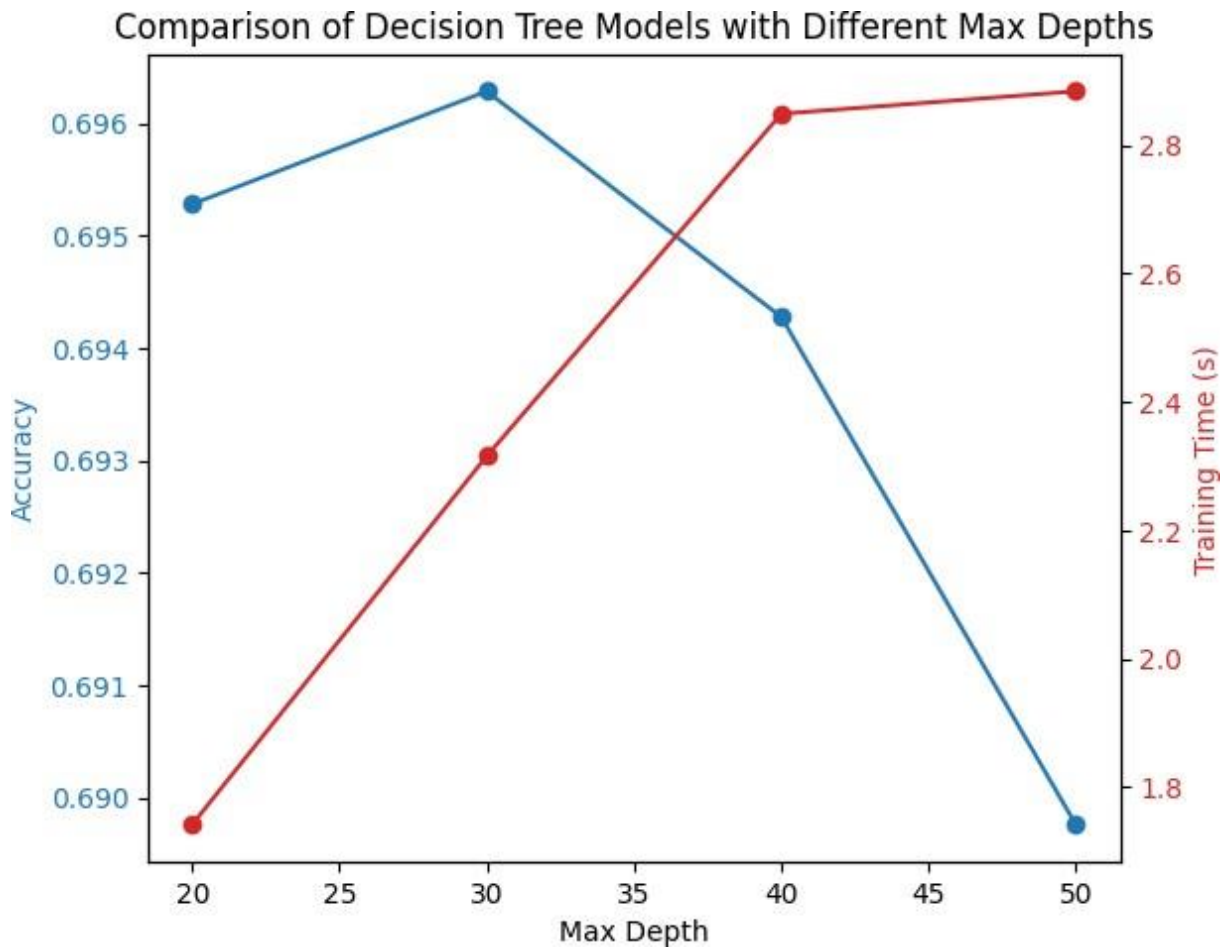


Nhận xét:

- Ở siêu tham số, có thể thấy rõ ràng, độ chính xác của thuật toán giữa critertion "gini"và criterion "entropy" đạt giá trị là gần 70%. Tuy nhiên, có sự chênh lệch về thời gian huấn luyện, thì khi criterion có giá trị là "entropy" thì thuật toán có thể được tối ưu nhất về thời gian và độ chính xác (thời gian huấn luyện của entropy, thấp hơn hẳn so với gini).

1.4/ Max Depth

	max_depth	accuracy	training_time
0	NaN	0.678213	2.800316
1	20.0	0.695281	1.741774
2	30.0	0.696285	2.315679
3	40.0	0.694277	2.848818
4	50.0	0.689759	2.883914



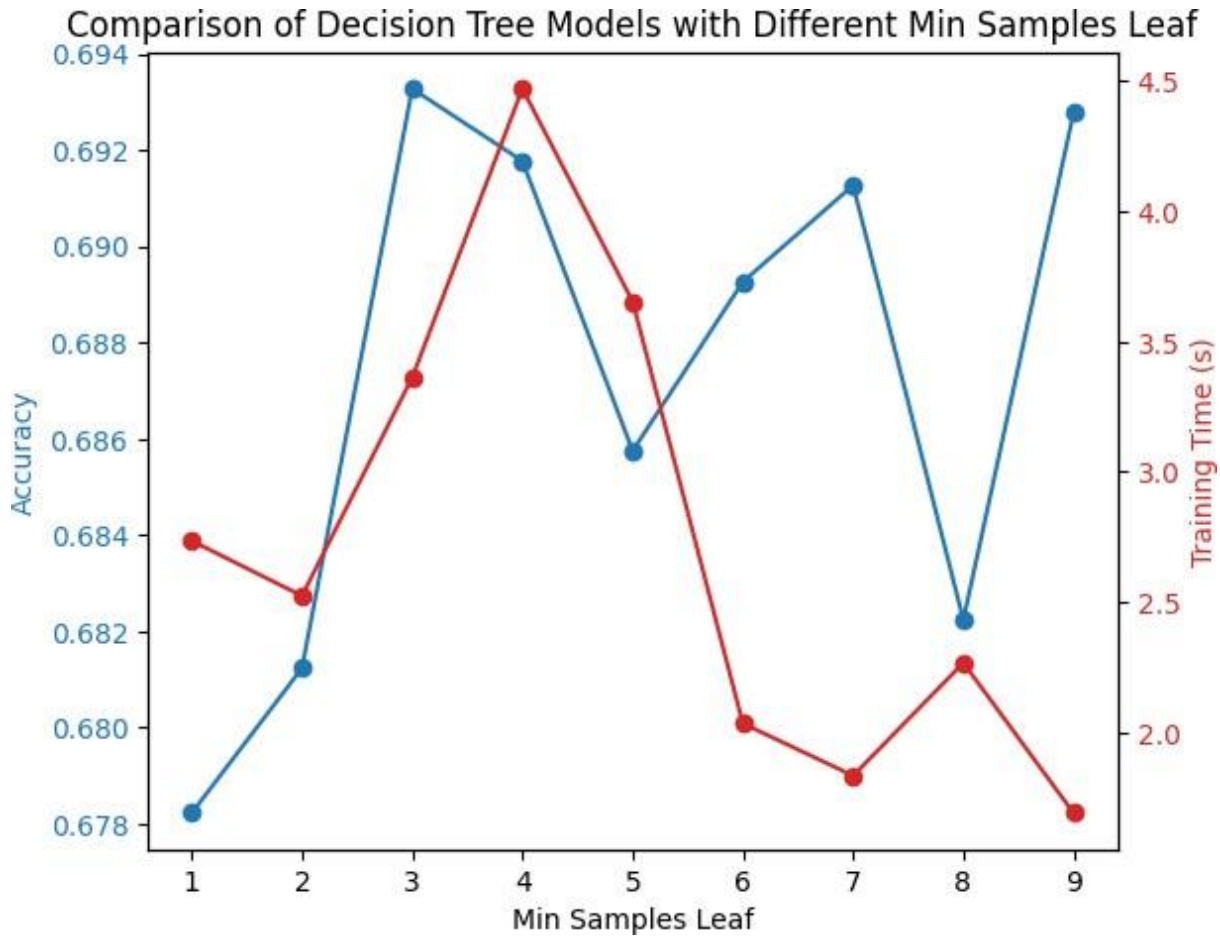
Nhận xét:

- Ở siêu tham số này, có thể thấy rõ ràng, độ chính xác của thuật toán có xu hướng là tăng nhẹ trước khi giảm mạnh về sau. Nói chung là độ chính xác theo biểu đồ, thì càng cái độ sâu tối đa càng sâu thì tỉ lệ chính xác của thuật toán. Song đó thì thời gian để huấn luyện dữ liệu độ sâu tối đa càng sâu thì thời gian cũng tăng theo chiều sâu tối đa.
- Ở max_depth đạt giá trị là 30 thì tỉ lệ chính xác của thuật toán đạt tối đa, trong khi đó thì thời gian huấn luyện cũng không phải quá cao, nằm ở ngưỡng trung bình so với nhữn max_depth còn lại.
- Còn nếu so sánh max_depth giữa 30 và 20 thì tại sao em không lấy max_depth là 20 mà em chọn là 30 vì em thiên về độ chính xác của thuật toán hơn so với thời gian huấn luyện. Một phần nữa nếu lấy tỉ lệ giữa độ chính xác và thời của hai max_depth này thì cái chênh lệch ở thời gian huấn luyện rất bé, bé hơn so với độ chính xác của chúng nên em nghĩ là max_depth = 30 sẽ là siêu tham số phù hợp cho thuật toán cây lựa chọn

1.5/ Min_sample_leaf

	min_samples_leaf	accuracy	training_time
0	1.0	0.678213	2.735450

1	2.0	0.681225	2.524250
2	3.0	0.693273	3.362485
3	4.0	0.691767	4.471026
4	5.0	0.685743	3.653712
5	6.0	0.689257	2.034379
6	7.0	0.691265	1.833813
7	8.0	0.682229	2.267006
8	9.0	0.692771	1.690818



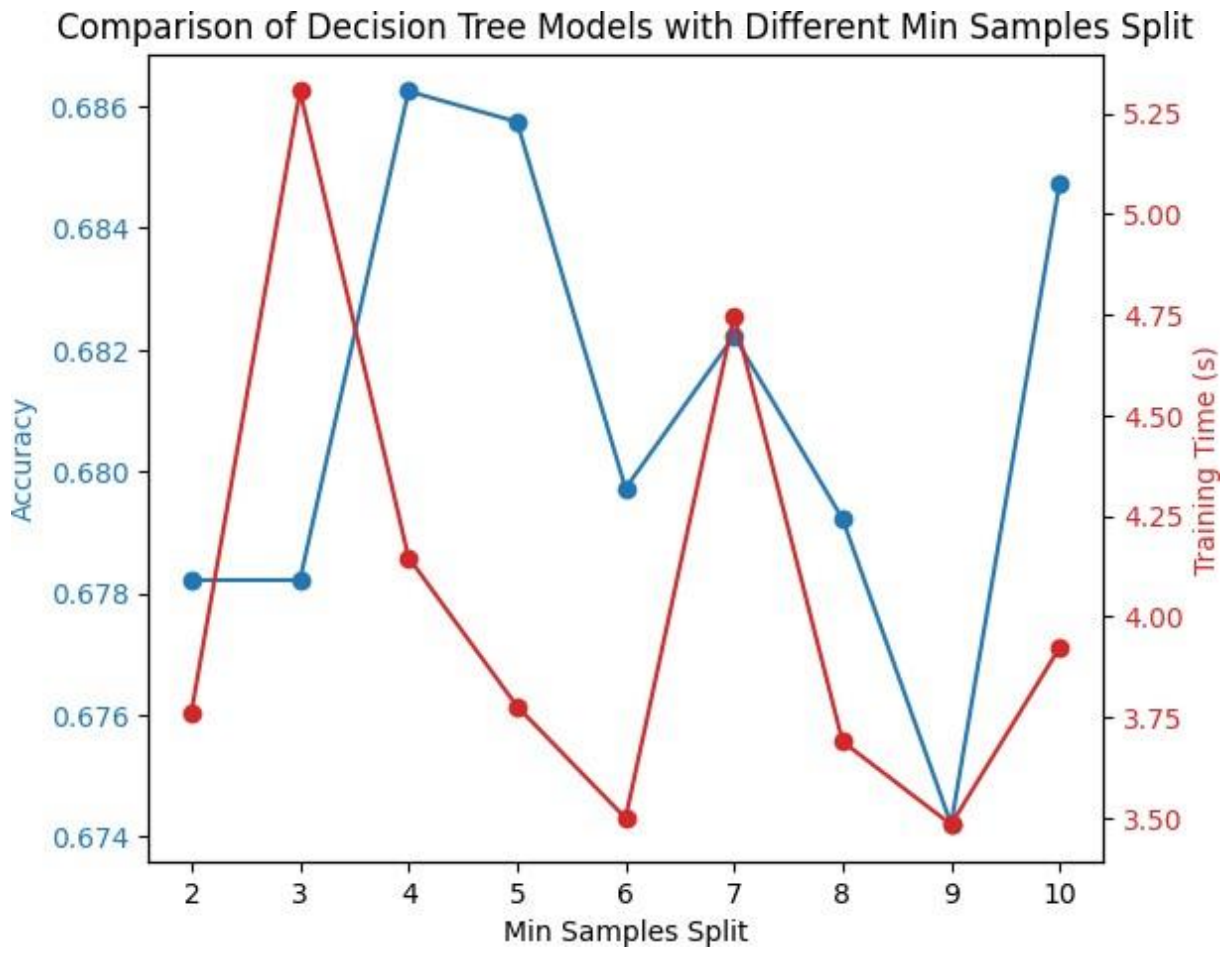
Nhận xét:

- Min_sample_leaf cho giá trị từ 1 đến 9 thì, độ chính xác và thời gian huấn luyện dữ liệu không ổn định. Nhìn kết quả thu được trên biểu đồ thì, siêu tham số tốt nhất là min_sample_leaf đạt giá trị bằng 9, có độ chính xác chỉ ít hơn không đáng kể so với khi min_sample_leaf đạt giá trị bằng 3. Nhưng xét về mặt thời gian huấn luyện thì lại ít hơn hẳn 2.78s.

- Ý nghĩa được rút ra dựa theo biểu, số lượng tối thiểu sample leaf càng cao thì cái độ chính xác của thuật toán không quá chênh lệch nhưng thời gian huấn luyện lại được giảm một cách đáng kể.

1.6 / Min_sample_split

	min_samples_split	accuracy	training_time
0	2.0	0.678213	3.763190
1	3.0	0.678213	5.304577
2	4.0	0.686245	4.147127
3	5.0	0.685743	3.777557
4	6.0	0.679719	3.499851
5	7.0	0.682229	4.747763
6	8.0	0.679217	3.689951
7	9.0	0.674197	3.483946
8	10.0	0.684739	3.923497



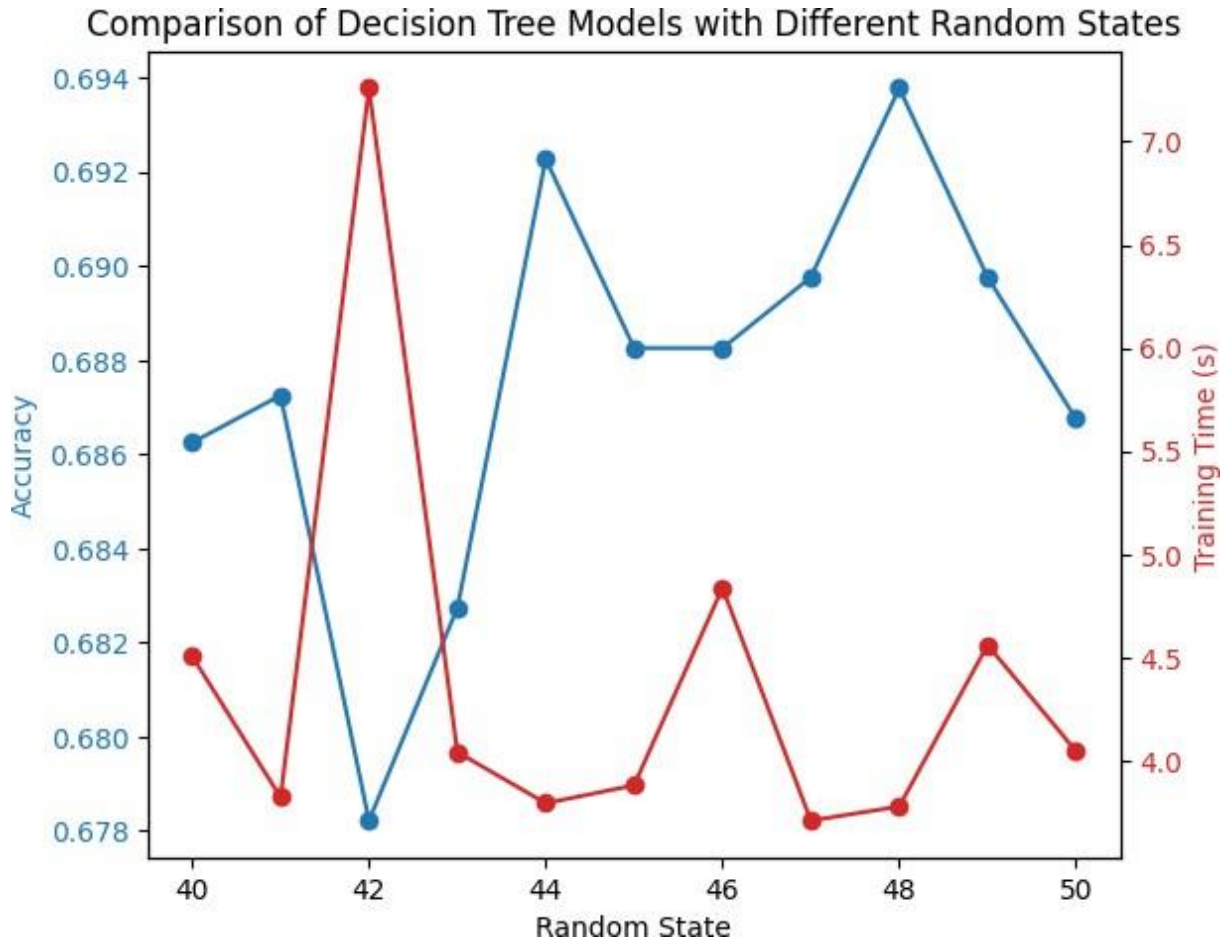
Nhận xét:

- Min_sample_split thì tương tự với min_sample_leaf, thì độ chính xác cũng như thời gian huấn luyện lại không ổn định. Nhìn kết quả thu được trên biểu đồ thì, siêu tham số tốt nhất là min_sample_split đạt giá trị bằng 10, có độ chính xác chỉ ít hơn 0.01 không đáng kể so với khi min_sample_split đạt giá trị bằng 4. Nhưng xét về mặt thời gian huấn luyện thì lại ít hơn hẳn khoảng 0.25s.

1.7/ Random state

	random_state	accuracy	training_time
0	40.0	0.686245	4.504205
1	41.0	0.687249	3.820551
2	42.0	0.678213	7.258939
3	43.0	0.682731	4.039776
4	44.0	0.692269	3.791799
5	45.0	0.688253	3.878876

6	46.0	0.688253	4.835895
7	47.0	0.689759	3.707655
8	48.0	0.693775	3.776013
9	49.0	0.689759	4.558872
10	50.0	0.686747	4.044908



<u>Nhận xét:</u>

- Trong khi random_state đạt giá trị 42 thì độ chính xác của thuật toán gần như là thấp nhất, nhưng thời gian huấn luyện lại là lâu nhất. Trong khi, khi random_states khi đạt những giá trị còn lại thì độ chính xác của thuật toán lại tỉ lệ nghịch với thời gian huấn luyện. Cụ thể hơn là khi mà độ chính xác càng thấp, thì thời gian huấn luyện lại cao hơn so với thời gian huấn luyện mà cho ra độ chính xác cao hơn
- Theo em tìm hiểu thì cần phải chọn một giá trị cụ thể cho random_states

⇒Ghi chú: theo sự quan sát của em mỗi khi chạy lại các thuật toán thì các kết quả cho ra lại có những sự chênh lệch không quá lớn. Và những nhận xét của em dựa trên kết quả mà biểu đồ tại thời điểm ấy đạt được.

Thuật toán Random Forest

Lý thuyết:

Định nghĩa: Random Forests là thuật toán học có giám sát (supervised learning). Random Forest có thể được sử dụng cho cả Phân lớp và Hồi quy trong Machine Learning. Như tên cho thấy, Random Forest là một bộ phân loại chứa một số cây quyết định trên các tập con khác nhau của tập dữ liệu đã cho và lấy giá trị trung bình để cải thiện độ chính xác dự đoán của tập dữ liệu đó. Thay vì dựa vào một cây quyết định, rừng ngẫu nhiên lấy dự đoán từ mỗi cây và dựa trên đa số phiếu dự đoán, và nó dự đoán kết quả cuối cùng.

Ưu điểm:

- Random Forest algorithm có thể sử dụng cho cả bài toán Classification và Regression.
- Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này, tránh vấn đề Overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán.
- Random forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu.
- Nó dự đoán đầu ra với độ chính xác cao, ngay cả đối với tập dữ liệu lớn, nó chạy hiệu quả.

Nhược điểm:

- Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian.
- Mô hình khó hiểu hơn so với cây quyết định, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

Các siêu tham số quan trọng:

Tăng sức mạnh dự đoán:

Siêu tham số **n_estimators**, chỉ là số cây mà thuật toán xây dựng trước khi lấy phiếu bầu tối đa hoặc lấy giá trị trung bình của các dự đoán. Nói chung, số lượng cây cao hơn làm tăng hiệu suất và làm cho các dự đoán ổn định hơn, nhưng nó cũng làm chậm quá trình tính toán.

Một siêu tham số quan trọng khác là **max_features**, là số lượng tối đa các đối tượng mà rừng ngẫu nhiên xem xét để tách một nút.

Siêu tham số quan trọng cuối cùng là **min_sample_leaf**. Điều này xác định số lượng lá tối thiểu cần thiết để tách một nút bên trong.

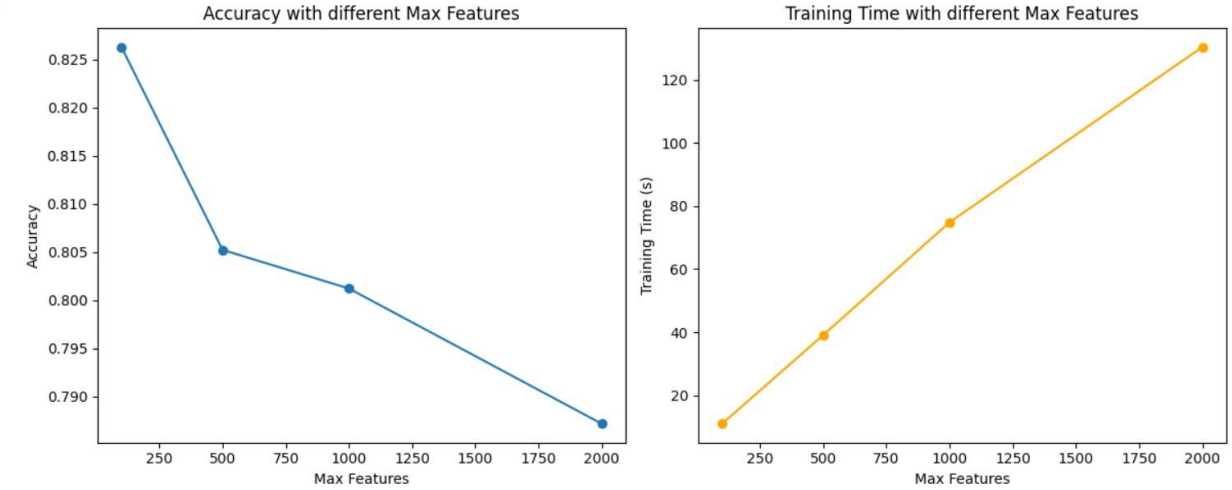
Tăng tốc độ của mô hình:

Siêu tham số **n_jobs** cho động cơ biết nó được phép sử dụng bao nhiêu bộ xử lý. Nếu nó có giá trị là "**1**", nó chỉ có thể sử dụng một bộ xử lý. Giá trị "**-1**" có nghĩa là không có giới hạn.

Siêu tham số **random_state** làm cho đầu ra của mô hình có thể sao chép được. Mô hình sẽ luôn tạo ra cùng một kết quả khi nó có một giá trị xác định là **random_state** và nếu nó được cung cấp cùng một siêu tham số và cùng một dữ liệu huấn luyện.

Khảo sát và nhận xét:

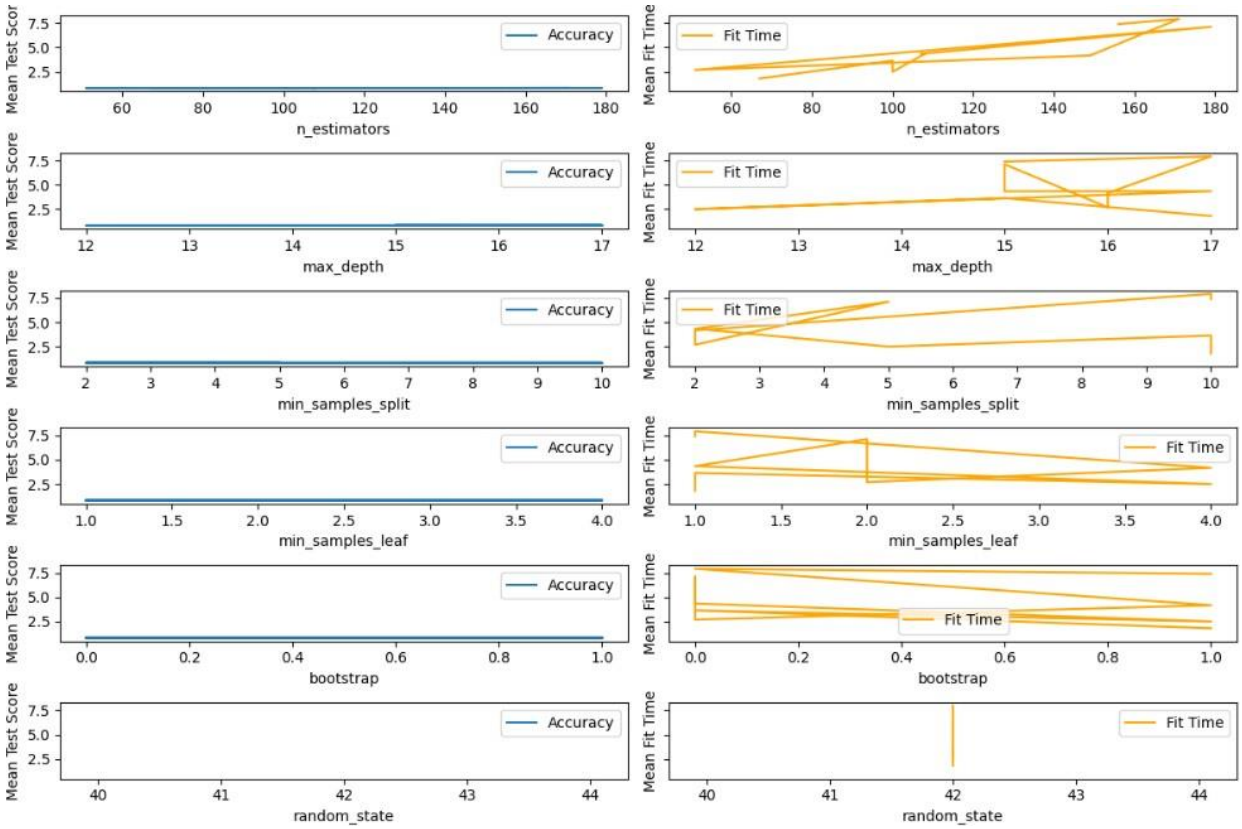
- **Khảo sát Accuracy và Training Time với các giá trị siêu tham số max_features khác nhau:**



Nhận xét:

- Độ chênh lệch **Accuracy** giữa max_feature nhỏ nhất (100) và max_feature lớn nhất (2000) không nhiều, cụ thể là không quá 0.1 => **Random Forest** dự đoán đầu ra với độ chính xác cao, ngay cả đối với tập dữ liệu lớn, nó chạy hiệu quả.
- Về **Training Time**, có sự chênh lệch lớn về thời gian training giữa max_feature nhỏ nhất (100) và max_feature lớn nhất (2000) => **Random Forest** sẽ đưa ra quyết định lâu khi tập dữ liệu lớn.

- **Khảo sát Accuracy và Training Time của các siêu tham số:**



Nhận xét:

Ở đây, vì chỉ số **Mean Test Score** là bằng nhau nên em sẽ dựa vào **Fit Time** - một chỉ số thời gian mà một mô hình máy học mất để được huấn luyện trên dữ liệu huấn luyện - để đưa ra kết luận về tầm ảnh hưởng của các siêu tham số với hiệu suất của mô hình. Ta có thể thấy **Mean Test Score** của các siêu tham số là như nhau nên siêu tham số nào có **Fit Time** giảm khi giá trị siêu tham số tăng thì em nghĩ đó là một siêu tham số hiệu quả.

Ngoài các tham số quan trọng đã được liệt kê ở phần lý thuyết như: **n_estimators, min_samples_leaf, random_state, n_jobs**, thì em có khảo sát thêm các siêu tham số: **max_depth, min_samples_split, bootstrap.**

Nhìn vào biểu đồ, có thể thấy rõ hai siêu tham số **bootstrap** và **min_samples_leaf** có xu hướng giảm thời gian **Fit Time** khi giá trị siêu tham số tăng.

=> Đây là hai siêu tham số quan trọng nên cân nhắc sử dụng trong quá trình xây dựng mô hình **Random Forest.** Việc cải thiện **Fit Time** nhưng không ảnh hưởng đến **Mean Test Score** cũng đem lại lợi ích trong việc xây dựng một mô hình học máy.

Kết luận:

Random Forest là một thuật toán đáng được sử dụng khi muốn xây dựng mô hình phân loại văn bản. Mặc dù có nhược điểm về thời gian khi gặp tập dữ liệu lớn, nhưng em nghĩ có thể điều chỉnh việc đó trong việc lựa chọn các siêu tham số. Còn lại với các ưu điểm của **Random Forest** như: là một phương pháp chính xác và mạnh mẽ, tránh vấn đề

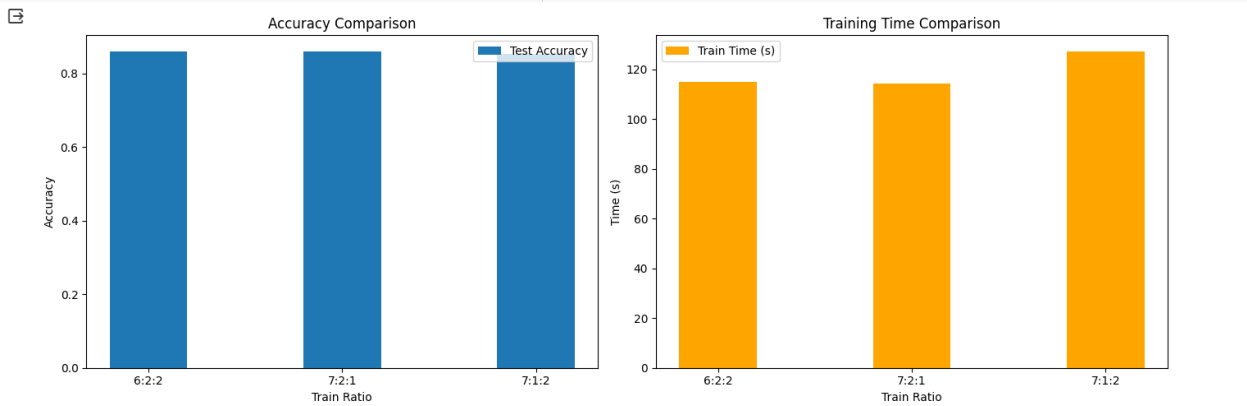
overfitting, có đầu ra với độ chính xác cao,... thì em tin đây là một thuật toán sáng giá để lựa chọn khi xây dựng mô hình phân loại văn bản.

Thuật toán CRF

Sau quá trình thử nghiệm, nhận thấy rằng mô hình CRF không phải là lựa chọn tối ưu cho việc phân loại cảm xúc từ văn bản đánh giá phim. Mô hình CRF được thiết kế để khai thác các mối quan hệ tuần tự trong dữ liệu chuỗi, như gán nhãn từng phần từ ngữ hoặc nhận dạng thực thể có tên. Tuy nhiên, trong bối cảnh của tác vụ phân loại cảm xúc từ văn bản đánh giá phim, các đặc điểm này không phát huy được hiệu quả:

1. Thời gian huấn luyện cao: CRF yêu cầu thời gian huấn luyện lớn, đặc biệt là khi xử lý bộ dữ liệu có kích thước lớn, như được thể hiện qua thời gian huấn luyện đáng kể so với các thuật toán khác như SVM, Decision Tree và Random Forest.

	Train Ratio	dev Accuracy	Test Accuracy	Train Time
0	(6, 2, 2)	0.861905	0.860738	114.953625
1	(7, 2, 1)	0.863173	0.858957	114.066797
2	(7, 1, 2)	0.871001	0.852890	127.249510



2. Bản chất của dữ liệu đánh giá phim: Đánh giá phim đặc trưng bởi văn bản có cấu trúc không đồng nhất và thường không có tính tuần tự mạnh mẽ giữa các nhãn, khiến cho CRF không phải là công cụ phù hợp.

3. Phù hợp với tác vụ gán nhãn chuỗi: CRF hiệu quả trong các tác vụ mà mối quan hệ giữa các phần tử liền kề có ý nghĩa, nhưng đối với phân loại văn bản cảm xúc, mỗi đoạn văn bản thường được xem xét độc lập, không cần đến việc mô hình hóa tuần tự.

4. Độ phức tạp tính toán cao: Việc xác định các đặc trưng cần thiết cho từng từ trong văn bản dài và phức tạp như đánh giá phim làm tăng khối lượng tính toán và không cần thiết khi mỗi đánh giá có thể được coi là một đơn vị độc lập.

5. Các mô hình phù hợp hơn: Các mô hình như SVM, Decision Tree, Random Forest và các thuật toán mạnh mẽ hơn như CNN, RNN, LSTM có thể nắm bắt mối quan hệ ngữ nghĩa phức tạp trong văn bản một cách hiệu quả hơn và phù hợp hơn cho tác vụ phân loại cảm xúc từ văn bản.

Kết luận:

Qua phân tích, khuyến nghị sử dụng các thuật toán khác thay cho CRF trong việc phân loại cảm xúc từ đánh giá phim. Những thuật toán này không chỉ cung cấp độ chính xác tương đương hoặc cao hơn mà còn giảm đáng kể thời gian huấn luyện và độ phức tạp trong việc cài đặt và tinh chỉnh mô hình.Phân tích này cung cấp một cái nhìn toàn diện về hiệu suất của CRF so với các thuật toán khác trong một tác vụ cụ thể và giải thích rõ lý do tại sao CRF không phải là sự lựa chọn tối ưu cho tác vụ này.

Thuật toán SVM

Mô hình Máy Vector Hỗ trợ (SVM) là một trong những phương pháp phân loại mạnh mẽ và linh hoạt, được sử dụng rộng rãi trong các tác vụ học có giám sát. Dưới đây là tổng kết về ưu và nhược điểm của SVM, cũng như tác động của các siêu tham số và `max_features`:

Ưu điểm của SVM:

1. Khả Năng Phân Tách Tốt: SVM tìm ra siêu phẳng tối ưu phân tách các lớp, thậm chí trong trường hợp các lớp không hoàn toàn tách biệt được trong không gian đầu vào.

2. Linh Hoạt: Các hàm nhân khác nhau (linear, rbf, poly) cho phép SVM hoạt động tốt trên dữ liệu tuyến tính lẫn không tuyến tính.

3. Hiệu Suất Cao: SVM thường cho hiệu suất phân loại cao và có khả năng chống lại việc overfitting, đặc biệt khi số chiều cao.

Nhược điểm của SVM:

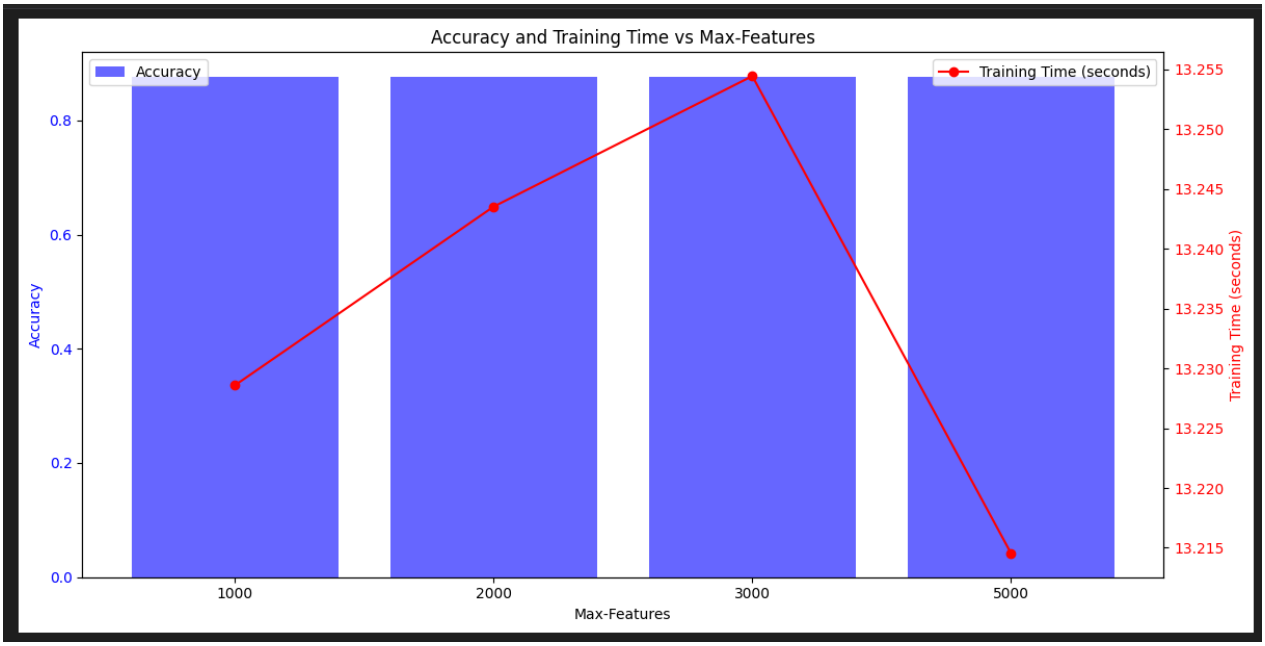
1. Lựa Chọn Hàm Nhân và Siêu Tham Số: Việc chọn hàm nhân và siêu tham số (như C, gamma) có thể khó khăn và cần phải dựa trên cross-validation, điều này tốn thời gian và tài nguyên.

2. Không Hiệu Quả với Dữ Liệu Lớn: SVM có thể trở nên không hiệu quả về mặt tính toán khi làm việc với dữ liệu lớn do nó yêu cầu giải quyết các bài toán tối ưu hóa phức tạp.

3. Không Cung Cấp Xác Suất: SVM thông thường không cung cấp dự đoán dưới dạng xác suất mà chỉ đưa ra lớp.

Đánh giá khảo sát Max-features:

```
Kết quả khảo sát Max-Features:
Max-Features: 1000 - Accuracy: 0.8755, Training Time: 13.23 seconds
Max-Features: 2000 - Accuracy: 0.8755, Training Time: 13.24 seconds
Max-Features: 3000 - Accuracy: 0.8755, Training Time: 13.25 seconds
Max-Features: 5000 - Accuracy: 0.8755, Training Time: 13.21 seconds
```



1. Độ Chính Xác không Thay Đổi: Độ chính xác không thay đổi theo số lượng đặc trưng, điều này có thể có nghĩa là:

- Các đặc trưng quan trọng nhất đã được bao gồm ngay cả ở mức max_features thấp nhất (1000).
- Mô hình có thể không nhạy cảm với sự thay đổi trong số lượng đặc trưng, có thể do bản chất của dữ liệu hoặc sự phân phối đặc trưng không thay đổi nhiều giữa các mức max_features.
- Có thể dữ liệu đã được làm sạch tốt và không chứa nhiều nhiễu, vì vậy việc tăng số lượng đặc trưng không làm tăng thông tin hữu ích.

2. Thời Gian Huấn Luyện ổn Định: Thời gian huấn luyện không giảm đáng kể khi tăng max_features. Điều này cho thấy rằng:

- Mô hình SVM đã được tối ưu hóa tốt về mặt tính toán, có khả năng xử lý số lượng đặc trưng tăng lên mà không làm tăng thời gian huấn luyện.

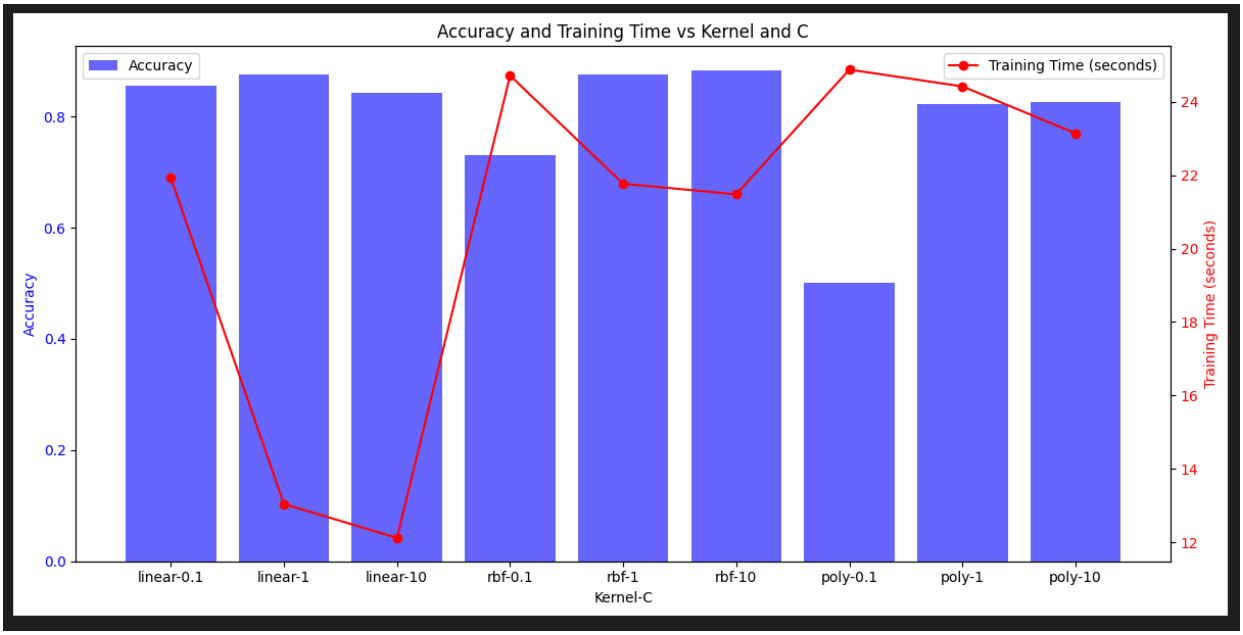
- Có thể dữ liệu của bạn không lớn đến mức sự tăng số lượng đặc trưng làm tăng thời gian tính toán.

3. Lựa Chọn Tốt Nhất: Dựa vào kết quả:

- Nếu bạn muốn mô hình tối ưu về mặt tài nguyên, max_features ở mức thấp nhất (1000) là lựa chọn tốt nhất, vì nó cung cấp độ chính xác tương tự nhưng có thể tiết kiệm bộ nhớ và tài nguyên tính toán.
- Nếu bạn không giới hạn bởi bộ nhớ hoặc tài nguyên tính toán, việc chọn một giá trị max_features cao hơn có thể giúp mô hình bạn có khả năng mở rộng tốt hơn khi thêm dữ liệu hoặc khi bạn muốn mô hình có khả năng nắm bắt thông tin chi tiết hơn.

4. Kết Luận: Trong tình huống của bạn, vì độ chính xác không thay đổi và thời gian huấn luyện không tăng, bạn có thể chọn mức max_features ở mức 1000 để giữ mô hình nhẹ nhàng và tốt cho việc triển khai trong môi trường hạn chế về tài nguyên. Tuy nhiên, bạn cũng nên kiểm tra độ chính xác trên tập phát triển hoặc sử dụng cross-validation để đảm bảo rằng mô hình không bị overfitting.

Đánh giá khảo sát siêu tham số:



```
Kết quả khảo sát Siêu Tham Số trong SVM:
Kernel: linear, C: 0.1 - Accuracy: 0.8555, Training Time: 21.93 seconds
Kernel: linear, C: 1 - Accuracy: 0.8755, Training Time: 13.04 seconds
Kernel: linear, C: 10 - Accuracy: 0.8430, Training Time: 12.12 seconds
Kernel: rbf, C: 0.1 - Accuracy: 0.7305, Training Time: 24.72 seconds
Kernel: rbf, C: 1 - Accuracy: 0.8760, Training Time: 21.77 seconds
Kernel: rbf, C: 10 - Accuracy: 0.8825, Training Time: 21.48 seconds
Kernel: poly, C: 0.1 - Accuracy: 0.5015, Training Time: 24.88 seconds
Kernel: poly, C: 1 - Accuracy: 0.8225, Training Time: 24.42 seconds
Kernel: poly, C: 10 - Accuracy: 0.8255, Training Time: 23.14 seconds
```

1. Hiệu Suất Độ Chính Xác:

- Kernel linear với C=1 cho độ chính xác tốt nhất (0.8755). Điều này có thể cho thấy rằng với dữ liệu của bạn, một siêu phẳng tuyến tính là đủ để phân loại hiệu quả.
- Khi C tăng lên 10 (với kernel linear), độ chính xác giảm xuống còn 0.8430, cho thấy rằng việc tăng độ phạt cho các điểm phân loại sai có thể không phải lúc nào cũng cải thiện được mô hình.
- Kernel rbf và poly với C=1 cho kết quả tương đối thấp hơn so với linear, điều này có thể chỉ ra rằng việc áp dụng phi tuyến không cần thiết hoặc không phải là cách tiếp cận tốt nhất với dữ liệu này.

2. Thời Gian Huấn Luyện:

- Thời gian huấn luyện dài nhất là với kernel poly và C=0.1, lên đến 24.48 giây. Điều này có thể do đặc tính tính toán phức tạp hơn của hàm nhân đa thức.
- Các mô hình sử dụng kernel rbf và poly với C cao hơn cũng cần thời gian huấn luyện nhiều hơn so với linear, điều này phản ánh độ phức tạp tính toán cao hơn của các hàm nhân không tuyến tính.
- Kernel linear với C=1 cung cấp một sự cân bằng tốt giữa độ chính xác và thời gian huấn luyện.

3. Lựa Chọn Tối Ưu:

- Mô hình SVM với kernel linear và C=1 có vẻ là lựa chọn tối ưu dựa trên kết quả này, vì nó cung cấp độ chính xác cao nhất và thời gian huấn luyện hợp lý.
- Các giá trị C khác và kernels không tuyến tính không cải thiện độ chính xác và thậm chí còn làm tăng thời gian huấn luyện, điều này không mong muốn trong môi trường sản xuất.

4. Kết Luận: Đối với dữ liệu của bạn, mô hình SVM sử dụng kernel tuyến tính và siêu tham số C=1 có thể là lựa chọn tốt nhất dựa trên cả độ chính xác và hiệu quả về mặt thời gian. Điều này

cho thấy rằng mô hình tuyến tính đã đủ để phân loại dữ liệu của bạn một cách hiệu quả mà không cần đến những mô hình phức tạp hơn.

Hiệu Quả và Độ Phức Tạp:

- SVM có thể đạt được độ chính xác cao với độ phức tạp hợp lý khi chọn đúng siêu tham số và `max_features`. Việc tối ưu hóa các siêu tham số dựa trên hiểu biết về dữ liệu và sử dụng cross-validation có thể giúp mô hình đạt được hiệu suất cao mà không làm tăng quá nhiều thời gian huấn luyện.

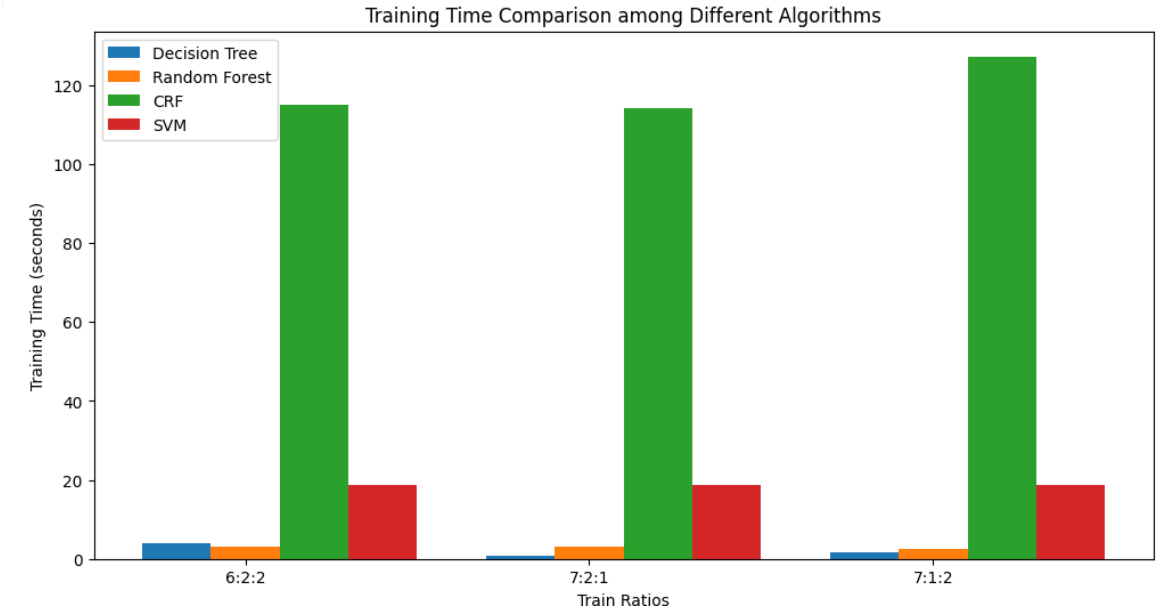
Kết Luận:

SVM có thể là một lựa chọn mạnh mẽ cho các tác vụ phân loại, nhưng việc chọn đúng siêu tham số và `max_features` là chìa khóa để đạt được hiệu suất tối ưu. Trong trường hợp cụ thể của bạn, kernel `linear` với C=1 và `max_features` ở mức thấp nhất cung cấp một sự cân bằng tốt giữa độ chính xác và hiệu quả tính toán, đồng thời cho thấy rằng việc tăng số lượng đặc trưng không cần thiết đối với bộ dữ liệu cụ thể này.

Trình bày kết quả đạt được

	Training Time	Test Accuracy	Test Precision	Test Recall	Test F1 Score
Decision Tree	5.581	0.678	0.679	0.678	0.678
SVM	41.748	0.87	0.871	0.87	0.87
Random Forest	9.354	0.833	0.834	0.833	0.833

Đánh giá kết quả



Dựa vào biểu đồ , có thể đưa ra một số nhận xét về thời gian huấn luyện của bốn thuật toán Decision Tree, Random Forest, CRF và SVM như sau :

+ **Decision Tree (Cây Quyết Định)**: Cây quyết định có thời gian huấn luyện ngắn nhất trong cả ba tỉ lệ tập huấn luyện. Điều này phản ánh tính đơn giản và hiệu quả về mặt tính toán của mô hình cây quyết định.
+ **Random Forest (Rừng Ngẫu Nhiên)**: Mô hình Random Forest có thời gian huấn luyện dài hơn so với Decision Tree, điều này có thể được giải thích bởi việc nó tạo ra nhiều cây quyết định để hình thành một mô hình tổng hợp, cần nhiều tính toán hơn.
+ **CRF (Conditional Random Field)**: CRF có thời gian huấn luyện dài nhất trong tất cả các thuật toán, đặc biệt là với tỉ lệ tập huấn luyện 7:1:2. Điều này cho thấy CRF có độ phức tạp tính toán cao, làm cho nó không phù hợp cho các tác vụ mà cần sự linh hoạt và nhanh nhẹn trong việc huấn luyện mô hình, hoặc khi cần đến việc thực hiện thử nghiệm nhanh với nhiều cấu hình khác nhau.
+ **SVM (Máy Vector Hỗ Trợ)**: SVM có thời gian huấn luyện dài hơn Decision Tree nhưng ngắn hơn CRF, đặc biệt là ở tỉ lệ tập huấn luyện 6:2:2 và 7:1:2. Điều này cho thấy SVM cân bằng khá tốt giữa hiệu suất và độ phức tạp tính toán, làm cho nó có thể là một lựa chọn hợp lý trong nhiều tình huống.

Kết luận :

Từ biểu đồ và các số liệu, chúng ta có thể kết luận rằng CRF có vẻ không phù hợp nhất cho việc phân tích dữ liệu này nếu xem xét thời gian huấn luyện là một yếu tố quan trọng. Điều này có thể bởi vì CRF thường được sử dụng cho các tác vụ gán nhãn tuần tự như nhận dạng thực thể trong

văn bản, nơi mà mối quan hệ giữa các đầu vào là quan trọng, chứ không phải cho việc phân loại toàn bộ tài liệu, nơi mà các mối quan hệ này không được nhấn mạnh.

Tài liệu tham khảo

Viblo. (2018, November 23). # Phân lớp bằng Random Forests trong Python. https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz

Couhp. (2018, January 24). Random Forest, Thế Nào Là Một Rừng Ngẫu Nhiên.

https://couhpcode.wordpress.com/2018/01/24/random-forest-the-nao-la-mot-rung-ngau-nhien/

ThongKe.Club. (2022, September 06). Ứng Dụng Của Thuật Toán Rừng Ngẫu Nhiên – Random Forests.
https://thongke.club/ung-dung-cua-thuat-toan-rung-ngau-nhien-random-forests/

Avinash Navlani. (2022). Decision Tree Classification in Python Tutorial.

https://www.datacamp.com/tutorial/decision-tree-classification-python?fbclid=IwAR1BAxFAd58PfWuRNvRvGC4b5D4DZnW9XXd-10qJcH-K_0A5akKGDTbNTP8

scikit-learn. (2023). sklearn.ensemble.RandomForestClassifier.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html