Table of Contents

Chúng tôi, **Bùi Minh Huy**, **Trần Lê Vân**, **Nguyễn Thị Thanh Tâm** xin cam đoan rằng:

Tất cả thông tin và phân tích trình bày trong báo cáo này được thực hiện một cách chính xác và trung thực. Mọi dữ liệu, nhận định hoặc ý kiến được trích dẫn từ các nguồn khác đều đã được nêu rõ nguồn gốc và trích dẫn đúng quy định. Chúng tôi cam đoan rằng không có bất kỳ hành vi sao chép hoặc sử dụng thông tin không hợp pháp nào từ các nguồn khác. Bài báo cáo này là kết quả của công trình nghiên cứu độc lập của chúng tôi và chưa từng được công bố tại bất kỳ nơi nào khác. Chúng tôi cam đoan đã tuân thủ nghiêm ngặt các quy tắc và quy định của môn học, bao gồm việc tham khảo và áp dụng các công cụ nghiên cứu một cách hợp lệ. Nếu phát hiện có bất kỳ sự gian lận nào, chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo của mình. Chúng tôi hy vọng rằng bài báo cáo này sẽ cung cấp những thông tin hữu ích cho các nhà nghiên cứu, doanh nghiệp, góp phần vào việc hiểu rõ hơn về mạng xã hội ngày nay.

# 1 GIỚI THIỆU TỔNG QUAN

Trong thời đại của điện toán di động và thiết bị thông minh, việc theo dõi và nhận dạng hoạt động con người (Human Activity Recognition - HAR) đã trở thành một lĩnh vực nghiên cứu đóng vai trò quan trọng trong nhiều ngành như trí tuệ nhân tạo, khoa học dữ liệu, y học và công nghệ cảm biến. HAR đóng vai trò cốt lõi trong các ứng dụng như giám sát sức khỏe, phát hiện té ngã, điều khiển nhà thông minh. Ngày nay, nhu cầu càng ngày gia tăng về các thiết bị công nghệ có thể hiểu hành vi con người dẫn đến việc phát triển các mô hình HAR là chính xác, hiệu quả và có khả năng triển khai thực tế là vô cùng cần thết.

Một trong những yếu tố chính thúc đẩy sự phát triển của HAR là sự phổ biến của các thiết bị di động thông minh và đồng hồ thông minh, vốn đợc trang bị sẵn các cảm biến quán tính bao gồm gia tốc kế (accelerometer) và con quay hồi chuyển (gyroscope). Những cảm biến này cho phép thu thập dữ liệu về chuyển động của người sử dụng với độ chính xác cao, chi phí thấp và tính khả dụng cao trong đời sống hàng ngày. Nhờ vậy, hệ thống HAR có thể được lắp đặt mà không cần sử dụng các thiết bị đắt tiền hoặc lắp đặt phức tạp.

Bên cạnh tiềm năng ứng dụng rộng rãi, việc xây dựng các mô hình HAR vẫn có nhiều khó khăn thách thức như dữ liệu cảm biến thường có số chiều lớn, có nhiều dữ liệu nhiễu và có tính biến động cao do phụ thuộc vào thói quen và hình thể của mỗi người. Bên cạnh đó, một số hoạt động có thể có mẫu tín hiệu tương tự nhau khiến cho các bài toán phân loại trở nên khó khăn hơn. Vì vậy, cần có một quy trình xử lý dữ liệu bài bản bao gồm các bước tiền xử lý dữ liệu, giảm chiều dữ liệu và huấn luyện mô hình học máy để có thể đặt được hiệu quả cao trong việc nhận dạng hoạt động con người.

Trong nghiên cứu này, chúng em đã tiến hành khai thác bộ dữ liệu “**Human Activity Recognition with Smartphones**” do UCI Machine Learning Repository cung cấp, một bộ dữ liệu được sử dụng rộng rãi trong cộng đồng nghiên cứu HAR. Chúng em đề xuất một quy trình học máy toàn diện bao gồm phân tích đặc trưng, giảm chiều dữ liệu bằng UMAP, PCA, TSNE và huấn luyện bằng các mô hình học máy như Random Forest, Decision Tree, Logistic Regression, Support Vectot Machine (SVM) để phân loại các hoạt động với mục tiêu là nâng cao độ chính xác và hiệu quả của mô hình. Những kết quả này sẽ cung cấp cái nhìn thực nghiệm rõ ràng cho các nhà nghiên cứu, đồng thời làm nền móng cho việc triển khai các hệ thống nhận dạng hoạt động trong thế giới thực.

# 2 CƠ SỞ LÝ THUYẾT

## 2.1 Hồi quy logistic (Logistic Regression)

### 2.1.1 Khái niệm

Hồi quy logistic (Logistic Regression) là một thuật toán học máy có giám sát (supervised learning), đồng thời cũng là một phương pháp thống kê phổ biến, được sử dụng rộng rãi trong việc phân tích và dự đoán dữ liệu phân loại. Mô hình này đặc biệt hiệu quả trong các bài toán phân loại nhị phân, nơi mà biến phụ thuộc chỉ có hai giá trị khả dĩ như có/không, đúng/sai hoặc 1/0.[1]

Hồi quy logistic thường được ưu tiên sử dụng trong các bài toán dự đoán khi biến phụ thuộc không phải là biến liên tục mà là biến nhị phân hoặc phân loại, giúp cung cấp cái nhìn định lượng và chính xác về mối quan hệ giữa các biến độc lập và xác suất xảy ra của sự kiện cần phân tích.

### 2.1.2 Các loại hồi quy logistic

**Hồi quy logistic nhị phân (Binary Logistic Regression)**. Hồi quy logistic nhị phân dự đoán mối quan hệ giữa các biến phụ thuộc nhị phân và độc lập . Một số ví dụ về đầu ra của loại hồi quy này có thể là thành công/thất bại, 0/1 hoặc đúng/sai.

**Hồi quy logistic đa thức : (Multinomial Logistic Regression)**. Biến phụ thuộc phân loại có hai hoặc nhiều kết quả rời rạc trong loại hồi quy đa thức. Hồi quy logistic đa thức có nhiều hơn hai kết quả có thể xảy ra .

**Hồi quy logistic thứ tự (Ordinal Logistic Regression)**. Hồi quy logistic thứ tự áp dụng khi biến phụ thuộc ở trạng thái có thứ tự (tức là thứ tự.

### 2.1.3 Hàm Sigmoid

Hồi quy logistic dự đoán xác suất rơi vào một trong hai lớp (binary classification), thường được ký hiệu là 0 hoặc 1.

Để biểu diễn xác suất này, sử dụng hàm sigmoid, có dạng S-shaped và giới hạn giá trị đầu ra trong khoảng từ 0 đến 1.

Công thức của hàm sigmoid:

Trong đó:

* s(z) = đầu ra trong khoảng từ 0 đến 1 (giá trị xác suất ước lượng).
* z = đầu vào của hàm (giá trị dự đoán của thuật toán, ví dụ như mx+b).
* e = hằng số Euler, và là cơ sở của logarithm tự nhiên.

**Đặc điểm mô hình:**

* Phân lớp và dự đoán: Dự đoán biến phụ thuộc nhị phân hoặc danh mục từ một hoặc nhiều biến độc lập.
* Xác định mức độ ảnh hưởng của biến độc lập: Xác định cách thức và mức độ mà các biến độc lập ảnh hưởng đến xác suất của sự kiện hoặc lớp mục tiêu.
* Tính toán xác suất sự kiện: Cung cấp ước lượng xác suất cho một sự kiện xảy ra dựa trên biến độc lập.

### 2.1.4 Cách thức hoạt động của mô hình

Hồi quy logistic sử dụng hàm logistic (còn gọi là hàm sigmoid) để chuyển đổi giá trị dự đoán thành xác suất. Hàm logistic có dạng:

Công thức của mô hình hồi quy logistic:

Trong đó:

*P(Y = 1)* là xác suất để sự kiện *Y = 1* xảy ra (ví dụ: sự kiện thành công, top 1,…). *X₁, X₂, …, Xₖ* là các biến độc lập. *β₀, β₁, …, βₖ* là hệ số mô hình cần được ước lượng. *e* là cơ số của logarithm tự nhiên.

**Cách hoạt động**

* **Ước lượng hệ số mô hình**: Hệ số β của mô hình được ước lượng thông qua quy trình tối ưu hóa, thường là phương pháp Maximum Likelihood Estimation (MLE).MLE tìm cách tối đa hóa xác suất của dữ liệu quan sát dựa trên hệ số β.
* **Phân lớp**: Dựa vào xác suất được dự đoán từ hàm logistic, quyết định phân loại một quan sát vào lớp 1 nếu P(Y = 1) ≥ một ngưỡng cụ thể (thường là 0.5) và ngược lại là lớp 0. Ví dụ: Nếu P(Y = 1) > 0.5, quan sát được phân loại là lớp 1.
* **Đánh giá mô hình:** Mô hình hồi quy logistic thường được đánh giá thông qua các chỉ số như độ chính xác (accuracy), precision, recall, điểm số F1, hoặc thống qua ROC và AUC.

### 2.1.5 Nhược điểm của hồi quy logistic:

* Không thích hợp với biến phụ thuộc liên tục.
* khó khăn trong việc mô hình hóa mối quan hệ phức tạp hoặc không tuyến tính mà không cần biến đổi dữ liệu.
* Không hiệu quả khi xử lý dữ liệu có nhiều biến độc lập hoặc có sự tương quan cao giữa các biến.

## 2.2 K-Nearest Neighbors (KNN)

### 2.2.1 Khái niệm

K-Nearest Neighbors (KNN) là một trong những thuật toán học máy đơn giản nhất nhưng hiệu quả, thuộc nhóm học có giám sát (supervised learning). Thuật toán được sử dụng cho cả hai bài toán phân lớp (classification) và hồi quy (regression), tuy nhiên nó phổ biến hơn trong các bài toán phân lớp. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.[2]

### 2.2.2 Cách thức hoạt động của thuật toán KNN

Bước 1: Chọn một số nguyên K (số lượng hàng xóm gần nhất cần xét).

Bước 2: Tính khoảng cách của data input với các data trong có trong tập data train, có các cách tính khoảng cách như: Minkowski, Euclid, Manhattan, … tùy mục đích sử dụng mà chúng ta sử dụng cách tính khoảng cách, thông dụng nhất là cách tính Euclid.

Bước 3: Sau khi tính khoảng cách từ data input tới toàn bộ data trong tập training, chọn ra K lân cận với khoảng cách ngắn nhất, với K được chọn ở bước số 1.

Bước 4: Thực hiện phân loại, kết quả sẽ theo label có tỉ lệ voting cao nhất.

**Lựa chọn giá trị K**

* Giá trị K quá nhỏ (ví dụ K = 1): mô hình nhạy cảm với nhiễu (noise), dễ bị overfitting.
* Giá trị K quá lớn: mô hình quá “mềm”, dễ bị underfitting

### 2.2.3 Ưu điểm

* Đơn giản, dễ hiểu: Thuật toán có nguyên lý hoạt động trực quan, dễ cài đặt.
* Không cần huấn luyện: Không tốn thời gian xây dựng mô hình.
* Hiệu quả với dữ liệu nhỏ: Hoạt động tốt với tập dữ liệu kích thước vừa phải.
* Ứng dụng đa dạng: Có thể áp dụng cho cả bài toán phân lớp và hồi quy.
* Không có giả định về dữ liệu: Là mô hình phi tham số, không yêu cầu dữ liệu tuân theo phân phối cụ thể.
* Thích ứng tốt với dữ liệu mới: Dễ dàng cập nhật mô hình bằng cách thêm mẫu mới.

### 2.2.4 Nhược điểm

* Chi phí dự đoán cao: Phải tính khoảng cách đến mọi mẫu dữ liệu, tốn kém với dữ liệu lớn.
* Nhạy cảm với quy mô dữ liệu: Các đặc trưng có phạm vi lớn sẽ áp đảo các đặc trưng có phạm vi nhỏ.
* Nhạy cảm với nhiễu và dữ liệu thừa: Kém hiệu quả khi có nhiều đặc trưng không liên quan.
* Vấn đề với dữ liệu không cân bằng: Các lớp thịnh hành có xu hướng áp đảo các lớp thiểu số.
* Khó chọn k tối ưu: Giá trị k thích hợp phụ thuộc vào dữ liệu cụ thể.

## 2.3 Support Vector Machine (SVM)

### 2.3.1 Khái niệm

SVM là một thuật toán học máy có giám sát (supervised learning), dùng để giải quyết các bài toán phân loại và hồi quy.Mục tiêu chính của SVM là tìm một siêu phẳng (hyperplane) tốt nhất để phân tách các điểm dữ liệu thuộc hai lớp khác nhau sao cho khoảng cách (margin) giữa siêu phẳng và các điểm gần nhất của mỗi lớp là lớn nhất.Điểm dữ liệu gần nhất đó được gọi là vector hỗ trợ (support vector). SVM sử dụng các hàm kernel để biểu diễn không gian dữ liệu ban đầu vào không gian cao chiều hơn,giúp phân loại tốt hơn đối với các bài toán phức tạp.[3]

### 2.3.2 Các loại SVM

Trong thuật toán SVM, có hai loại chính là Linear SVM (SVM tuyến tính) và Non-linear SVM (SVM phi tuyến).

Linear SVM là loại SVM mà ta có thể phân chia dữ liệu bằng một đường thẳng. Điều này áp dụng cho các bài toán có dữ liệu tuyến tính và có thể tách biệt bằng một đường thẳng.

Non-linear SVM được sử dụng khi không thể phân chia dữ liệu bằng một đường thẳng. Trong trường hợp này, ta sẽ sử dụng các phương pháp biến đổi dữ liệu sao cho chúng trở thành tuyến tính, sau đó áp dụng Linear SVM để giải quyết bài toán.

### 2.3.3 Cách hoạt động

* Chọn siêu phẳng phân tách: SVM tìm kiếm siêu phẳng sao cho có khoảng cách margin lớn nhất giữa hai lớp dữ liệu.
* Tối đa hóa margin: Cố gắng duy trì khoảng cách lớn nhất từ siêu phẳng tới các điểm dữ liệu gần nhất.
* Áp dụng kernel nếu cần: Nếu dữ liệu không phân tách tuyến tính, áp dụng kernel trick để biến đổi không gian dữ liệu.
* Đánh giá và điều chỉnh tham số để tối ưu mô hình.

### 2.3.4 Margin trong SVM

Margin là khoảng cách từ siêu phẳng phân tách (hyperplane) đến các điểm dữ liệu gần nhất thuộc hai lớp khác nhau – các điểm này được gọi là vector hỗ trợ (support vectors). Trong hình dung đơn giản, ví dụ như bài toán phân loại quả táo và quả lê đặt trên mặt bàn, margin chính là khoảng cách từ cây que (đại diện cho siêu phẳng) đến quả táo và quả lê gần cây que nhất. Điều quan trọng trong SVM là thuật toán luôn tìm cách tối đa hóa margin này, nhằm tạo ra một siêu phẳng phân tách có khoảng cách lớn nhất đến các điểm dữ liệu gần ranh giới nhất của mỗi lớp. Nhờ đó, SVM có thể tăng cường khả năng tổng quát hóa và giảm thiểu nguy cơ phân loại sai khi xử lý các điểm dữ liệu mới chưa từng thấy trước đó.

### 2.3.5 Thủ thuật Kernel

Kernel là một hàm ánh xạ dữ liệu từ không gian ít nhiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Một cách trực quan, kỹ thuật này giống như việc bạn gập tờ giấy lại để có thể dùng kéo cắt một lỗ tròn trên nó.

### 2.3.6 Ưu điểm

* Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn.
* Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới.
* Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiếncho hiệu suất phân loại lớn hơn.

### 2.3.7 Nhược điểm

* Thuật toán SVM có độ phức tạp tính toán cao khi số lượng dữ liệu lớn.
* SVM yêu cầu dữ liệu huấn luyện là tuyến tính hoặc phi tuyến tính.
* Thuật toán SVM cần lựa chọn tham số tốt để đạt được kết quả tốt nhất.

## 2.4 Decision Tree

### 2.4.1 Khái niệm

Cây quyết định (Decision Tree) là một trong những thuật toán phổ biến nhất trong lĩnh vực máy học (Machine Learning). Là một công cụ mạnh mẽ, thường được sử dụng để giải quyết các bài toán về phân loại (classification) và dự đoán (regression) trong khai phá dữ liệu.Cây quyết định là một thuật toán máy học dùng để dự đoán hoặc phân loại dữ liệu dựa trên các bước ra quyết định nối tiếp nhau.[4]

### 2.4.2 Cách hoạt động

Cách hoạt động của cây quyết định rất đơn giản:

* Đầu tiên, cây quyết định sẽ xem xét toàn bộ tập dữ liệu.
* Sau đó, chia dữ liệu thành các nhóm nhỏ dựa trên một số đặc điểm hoặc điều kiện nhất định (ví dụ như tuổi tác, thu nhập, màu sắc, nhiệt độ..).
* Việc chia nhỏ này sẽ tiếp tục thực hiện đến khi dữ liệu ở mỗi nhóm trở nên rõ ràng và dễ dàng để dự đoán hoặc phân loại.
* Trong cây quyết định, mỗi “nút” (node) đại diện cho một đặc điểm (ví dụ: “tuổi”, “mức thu nhập”, “thời tiết”). Các “nhánh” (branch) nối giữa các nút chính là các điều kiện để chia dữ liệu (ví dụ: tuổi lớn hơn hay nhỏ hơn 30). Cuối cùng, các “nút lá” (leaf node) là kết quả cuối cùng mà cây quyết định đưa ra, ví dụ như “mua hàng” hay “không mua hàng”, “đi chơi thể thao” hay “ở nhà”.
* Nói cách khác, cây quyết định giống như một chuỗi các câu hỏi đơn giản được sắp xếp theo từng bước, giúp ta dễ dàng đi đến một quyết định cuối cùng.

### 2.4.3 Công thức

**Gini Impurity**

Công thức Gini Impurity được sử dụng trong thuật toán cây quyết định để đo lường độ không chính xác của một dự đoán khi phân loại một tập dữ liệu.

Cần lưu ý là Gini Impurity càng nhỏ (gần 0) thì tập dữ liệu đó càng “thuần khiết”, nghĩa là các mẫu trong cùng một nhóm có xu hướng thuộc vào cùng một lớp. Ngược lại, nếu Gini Impurity cao (gần 1), thì việc phân loại các mẫu trong nhóm đó trở nên không chắc chắn.

Giả sử khi đang xem xét một tập dữ liệu chia thành K nhóm, mỗi nhóm chứa một phần tỷ lệ pi với i=1,2,…,K.

Công thức tính độ bất thuần Gini (Gini Impurity):

Trong đó:

* là Gini Impurity.
* là tỷ lệ các mẫu thuộc vào lớp .
* Khi xây dựng cây quyết định, chúng ta cần chọn thuộc tính và giá trị phân chia sao cho Gini Impurity sau phân chia là nhỏ nhất, tức là mức độ “thuần khiết” cho dữ liệu thuộc về từng nhóm con được tạo ra.

**Entropy**

Entropy trong cây quyết định là một khái niệm được sử dụng để đo lường sự không chắc chắn trong dữ liệu (Trung bình surprise). Trong ngữ cảnh của cây quyết định, entropy thường được sử dụng để đo lường mức độ không chắc chắn của phân phối lớp trong tập dữ liệu.

Entropy được tính bằng công thức sau:

Trong đó:

* là tập dữ liệu.
* là số lớp trong tập dữ liệu.
* là tỷ lệ của lớp trong tập dữ liệu.
* Entropy càng cao khi tỷ lệ của các lớp trong tập dữ liệu gần bằng nhau, và càng thấp khi một lớp chiếm đa số.
* Khi xây dựng cây quyết định, chúng ta cố gắng chia tập dữ liệu sao cho entropy sau khi chia là thấp nhất có thể. Điều này giúp cây quyết định có thể học được các quy tắc quyết định hiệu quả từ dữ liệu.
* Quyết định về cách chia tập dữ liệu dựa trên entropy thường được thực hiện bằng cách so sánh entropy trước và sau khi chia, và chọn cách chia mà giảm entropy nhiều nhất.

### 2.4.4 Ưu điểm

* Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
* Dữ liệu đầu vào có thể là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả.
* Có thể làm việc với cả dữ liệu số và dữ liệu phân loại.
* Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê.
* Có khả năng làm với dữ liệu lớn.

### 2.4.5 Nhược điểm

* Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu. Thậm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
* Cây quyết định hay gặp vấn đề overfitting

### 2.4.6 Các vấn đề sau khi thực hiện áp dụng mô hình vào dữ liệu cần dự đoán

**Underfitting:**là hiện tượng kết quả độ chênh lệch của mô hình được huấn luyện và kết quả độ chênh lệch của dữ liệu cần dự đoán đạt giá trị mức cao giống nhau, do mô hình chưa được huấn luyện đầy đủ. Cần xem lại cấu trúc của mô hình (tăng thêm độ phức tạp) để có thể huấn luyện các tập dữ liệu khó và tăng thêm dữ liệu huấn luyện để tăng hiệu suất của mô hình.

**Overfitting:**là hiện tượng kết quả của mô hình được huấn luyện quá tốt (độ chênh lệch thấp) nhưng khi áp dụng vào dữ liệu cần dự đoán thì mô hình đạt hiêu suất kém (độ chênh lệch cao) do mô hình đã học quát sát với dữ liệu huấn luyện và không có khả năng tổng quát hóa các dữ liệu cần dự đoán. Cần sử dụng một số các phương pháp tránh overfitting như tăng độ đa dạng của dữ liệu, giảm thiểu độ phức tạp của mô hình.

=> Giải pháp : Pruning solution là được áp dụng đối với trường hợp mô hình huấn uyện bị overfitting khi sử dụng mô hình Decision Tree bằng cách hạn chế kích thước, chiều sâu của mô hình này.

## 2.5 Random Forest

### 2.5.1 Khái niệm

Random Forest là một thuật toán học máy thuộc nhóm học có giám sát (supervised learning), được sử dụng phổ biến trong cả bài toán phân loại và hồi quy.Khác với Decision Tree chỉ dựa vào một cây duy nhất, Random Forest kết hợp nhiều cây quyết định để tạo ra một mô hình tổng hợp, có khả năng dự đoán chính xác và ổn định hơn.[5]

### 2.5.2 Cách hoạt động của Random Forest

* Tạo ra nhiều cây khác nhau từ các tập dữ liệu nhỏ, được chọn ngẫu nhiên từ tập dữ liệu gốc.
* Mỗi cây sẽ đưa ra dự đoán riêng.
* Mô hình sẽ lấy trung bình (với hồi quy) hoặc bỏ phiếu theo số đông (với phân loại) để đưa ra kết quả cuối cùng.

### 2.5.3 Công thức

Bài toán phân loại (classification) Bài toán phân loại, mỗi cây quyết định trong rừng sẽ đưa ra một nhãn dự đoán. Sau đó, mô hình Random Forest sẽ lấy nhãn xuất hiện nhiều nhất trong các dự đoán của từng cây làm kết quả cuối cùng.

Công thức được biểu diễn như sau:

Trong đó:

* là kết quả dự đoán cuối cùng của mô hình.
* là kết quả dự đoán của cây thứ với đầu vào .
* là tổng số cây trong mô hình Random Forest.
* *mode* là hàm chọn giá trị xuất hiện nhiều nhất.
* Bài toán hồi quy (Regression Với bài toán hồi quy, mỗi cây trong rừng sẽ đưa ra một giá trị số. Kết quả cuối cùng được tính bằng cách lấy trung bình cộng các giá trị dự đoán của tất cả các cây.

Được tính bằng công thức sau:

Trong đó:

* là giá trị dự đoán cuối cùng.
* là giá trị dự đoán từ cây thứ .
* là tổng số cây trong mô hình.

### 2.5.4 Ưu điểm

* Độ chính xác cao: mô hình tổng hợp nhiều cây, Random Forest thường cho kết quả chính xác hơn so với cây quyết định đơn lẻ.
* Chống overfitting tốt: kết hợp nhiều cây được huấn luyện từ dữ liệu và thuộc tính ngẫu nhiên giúp mô hình tránh học quá sát vào dữ liệu huấn luyện.
* Làm việc tốt với dữ liệu lớn và có nhiều đặc trưng: Random Forest xử lý hiệu quả dữ liệu có số chiều lớn và phức tạp.
* Không yêu cầu chuẩn hóa dữ liệu: Dữ liệu đầu vào không cần phải được chuẩn hóa hoặc xử lý đặc biệt như một số thuật toán khác.
* Có thể đo lường mức độ quan trọng của các thuộc tính (feature importance): Giúp phân tích và chọn ra các yếu tố ảnh hưởng nhiều nhất đến kết quả dự đoán.

### 2.5.5 Nhược điểm

* Khó giải thích mô hình: Random Forest gồm nhiều cây nên khó để xem toàn bộ quátrình mô hình đưa ra kết quả , khó để giải thích được mô hình .
* Thời gian huấn luyện và dự đoán lâu hơn: Mô hình gồm nhiều cây nên tốn nhiều thời gian và tài nguyên hơn khi xử lý dữ liệu lớn.
* Chiếm nhiều bộ nhớ: Lưu trữ nhiều cây có thể tốn nhiều RAM, đặc biệt khi số lượng cây lớn.
* Dễ bị bias nếu dữ liệu không cân bằng: Trong trường hợp dữ liệu bị lệch, Random Forest có thể dự đoán thiên lệch theo lớp đó nếu không xử lý cân bằng dữ liệu trước.

## 2.6 UMAP

### 2.6.1 Khái niệm

**Uniform Manifold Approximation and Projection (UMAP)** là một kỹ thuật giảm chiều dữ liệu, tương tự như t-SNE, thường được sử dụng để trực quan hóa dữ liệu.[6] Ngoài ra, UMAP còn có thể được sử dụng như một phương pháp giảm chiều phi tuyến tổng quát trong các bài toán học máy. Thuật toán UMAP được xây dựng dựa trên ba giả định chính về cấu trúc dữ liệu: 1. Dữ liệu phân bố đều trên một đa tạp Riemannian (Riemannian manifold); 2. Metric Riemannian là hằng số cục bộ (hoặc có thể được xấp xỉ là như vậy); 3. Ống phân phối (local connectivity) được kết nối cục bộ.

### 2.6.2 Mục tiêu của UMAP

• Giảm số lượng biến: UMAP chuyển đổi dữ liệu sang một không gian có số chiều thấp hơn, phù hợp cho các tác vụ xử lý và phân tích tiếp theo.

* Giữ lại cấu trúc dữ liệu: UMAP cố gắng bảo toàn cả cấu trúc cục bộ (local structure) lẫn cấu trúc tổng thể (global structure) của dữ liệu trong không gian mới.
* Bảo tồn mối quan hệ phi tuyến: Khác với PCA, UMAP có khả năng nắm bắt và biểu diễn các mối quan hệ phi tuyến giữa các điểm dữ liệu.
* Trực quan hóa dữ liệu: UMAP đặc biệt hiệu quả trong việc trực quan hóa dữ liệu nhiều chiều trong không gian 2D hoặc 3D, với độ chính xác và sắc nét cao hơn so với nhiều kỹ thuật khác như t-SNE.

### 2.6.3 Quy trình thực hiện UMAP

* Chuẩn hóa dữ liệu (nếu cần).
* Tìm k hàng xóm gần nhất cho mỗi điểm và xây dựng đồ thị fuzzy biểu diễn cấu trúc cục bộ.
* Tính toán xác suất kết nối giữa các điểm dựa trên khoảng cách và hàm kernel.
* Khởi tạo các điểm trong không gian thấp chiều và tối ưu hóa đồ thị sao cho bảo toàn cấu trúc so với đồ thị ban đầu.
* Chiếu dữ liệu sang không gian mới để phục vụ trực quan hóa hoặc các bước phân tích tiếp theo.

### 2.6.4 Ưu điểm

* Bảo toàn cấu trúc cục bộ tốt.
* Có thể dùng cho cả supervised & unsupervised.
* UMAP cho phép lưu và áp dụng mô hình lên dữ liệu mới.

### 2.6.5 Nhược điểm

* Không giải thích được biến gốc.
* Phụ thuộc vào tham số.

# 3 GIỚI THIỆU BỘ DỮ LIỆU

Bộ dữ liệu **Human Activity Recognition with Smartphones** được xây dựng nhằm phục vụ cho các nghiên cứu về nhận diện hành vi con người thông qua dữ liệu cảm biến thu thập từ thiết bị di động. Tập dữ liệu được thu thập từ 30 người tham gia (gọi là *subjects*) (15 nam và 15 nữ, độ tuổi từ 19 đến 48) thực hiện sáu hoạt động thường ngày như Đi bộ (Walking), đi lên cầu thang (Walking Upstairs), đi xuống cầu thang (WalkingDownstairs), ngồi (Sitting), đứng (Standing)và nằm (Laying).[7]

Dữ liệu được ghi lại bằng một điện thoại thông minh (Samsung Galaxy S II) đeo ở thắt lưng của người dùng. Thiết bị này sử dụng hai loại cảm biến là **accelerometer** và **gyroscope** được sử dụng để ghi lại chuyển động theo 3 trục X, Y, Z với tần số lấy mẫu 50Hz.

Mỗi chuỗi tín hiệu được chia thành các **cửa sổ trượt** có độ dài 2.56 giây, tương ứng với 128 lần đo. Từ mỗi cửa sổ, các đặc trưng (features) đã được trích xuất từ **miền thời gian** và **miền tần số** để tạo ra một tập hợp dữ liệu có cấu trúc sẵn sàng cho mô hình học máy.

Các tín hiệu chính được sử dụng để tạo đặc trưng bao gồm:

* tBodyAcc-XYZ: Gia tốc cơ thể theo 3 trục trong miền thời gian
* tGravityAcc-XYZ: Gia tốc do trọng lực
* tBodyGyro-XYZ: Tốc độ quay từ con quay hồi chuyển
* tBodyAccJerk-XYZ, tBodyGyroJerk-XYZ: Jerk - đo sự thay đổi đột ngột của chuyển động
* Mag: Độ lớn vector gia tốc, được tính bằng chuẩn Euclidean:
* fBodyAcc-XYZ, fBodyGyro-XYZ: Biến miền tần số được tạo từ FFT

Từ các tín hiệu trên, 561 đặc trưng thống kê đã được trích xuất, bao gồm:

* mean(), std(), mad(), max(), min(): Đặc trưng thông kê truyền thống
* sma(): Signal Magnitude Area
* energy(): Tổng bình phương chia số phần tử
* entropy(), iqr(), arCoeff(), correlation()
* meanFreq(), skewness(), kurtosis(), bandsEnergy(), angle()
* Toàn bộ bộ dữ liệu được chia thành hai phần: Tập huấn luyện (train.csv): bao gồm 7352 mẫu. Tập kiểm tra (test.csv): bao gồm 2947 mẫu. Mỗi mẫu tương ứng với một cửa sổ thời gian 2.56 giây, được biểu diễn bằng **561 đặc trưng đầu vào (features)**, cùng với một mã định danh người thực hiện (subject) và một nhãn hoạt động (Activity), các nhãn này được mã hóa từ 1 đến 6 tương ứng với:

| Giá trị nhãn | Hoạt động |
| --- | --- |
| 1 | WALKING |
| 2 | WALKING\_UPSTAIRS |
| 3 | WALKING\_DOWNSTAIRS |
| 4 | SITTING |
| 5 | STANDING |
| 6 | LAYING |

Đầu tiên ta tiến hành đọc dữ liệu

# train <-read.csv('/Users/huy/Documents/doanthaytung/archive/train.csv')  
# train <-read.csv("D:/BT/clonegit/doanthaytung/archive/train.csv")  
train <- read.csv('archive/train.csv')  
# test <-read.csv('/Users/huy/Documents/doanthaytung/archive/test.csv')  
# test <-read.csv("D:/BT/clonegit/doanthaytung/archive/test.csv")  
test <- read.csv("archive/test.csv")  
#head(train)  
dim(train)

## [1] 7352 563

#head(test)  
dim(test)

## [1] 2947 563

Kiểm tra kiểu dữ liệu của các biến trong tập huấn luyện để thống kê số lượng cột thuộc từng kiểu dữ liệu trong bảng dữ liệu train. Kết quả sẽ cung cấp một cái nhìn tổng quan về cấu trúc dữ liệu, giúp đánh giá mức độ sẵn sàng của tập dữ liệu cho các bước xử lý tiếp theo.

table(sapply(train, class))

##   
## character integer numeric   
## 1 1 561

Theo kết quả trả về, bộ dữ liệu train bao gồm: 561 biến kiểu numeric, 1 biến kiểu character, 1 biến kiểu integer.

Kiểm tra xem có biến nào được lưu dưới dạng chuỗi ký tự (character) hay không. Đảm bảo các biến phân loại sẽ được xử lý đúng cách trong mô hình học máy.

names(train)[sapply(train, class) == "character"]

## [1] "Activity"

Kết quả cho thấy chỉ có một biến duy nhất là “Activity” đang được lưu dưới dạng character. Cho thấy nhãn phân loại hiện tại vẫn chưa được chuyển đổi về dạng factor, cần được xử lý lại để đảm bảo phù hợp với các thuật toán phân loại.

Kiểm tra danh sách đầy đủ các giá trị duy nhất (tức là các lớp phân loại) mà biến này chứa.

unique(train$Activity)

## [1] "STANDING" "SITTING"   
## [3] "LAYING" "WALKING"   
## [5] "WALKING\_DOWNSTAIRS" "WALKING\_UPSTAIRS"

Tiến hành chuyển đổi kiểu dữ liệu này sang kiểu factor.

train$Activity <- as.factor(train$Activity)  
test$Activity <- as.factor(test$Activity)

Để đảm bảo chất lượng dữ liệu đầu vào cho mô hình học máy, chúng em thực hiện kiểm tra các giá trị bị thiếu (missing values) trong cả hai tập dữ liệu huấn luyện và kiểm tra.

cat("Giá trị thiếu ở tập train:", sum(is.na(train)), "\n")

## Giá trị thiếu ở tập train: 0

cat("Giá trị thiếu ở tập test:", sum(is.na(test)), "\n")

## Giá trị thiếu ở tập test: 0

Kết quả cho thấy không có giá trị thiếu trong cả tập huấn luyện (train) và tập kiểm tra (test), với tổng số lượng NA bằng 0.

Bên cạnh việc kiểm tra giá trị thiếu, chúng em đánh giá tính duy nhất của các dữ liệu bằng cách xác định xem có các dòng dữ liệu nào bị trùng lặp trong cả hai tập huấn luyện và kiểm tra hay không.

cat("Số dòng bị trùng lặp trong tập train:", sum(duplicated(train)), "\n")

## Số dòng bị trùng lặp trong tập train: 0

cat("Số dòng bị trùng lặp trong tập test :", sum(duplicated(test)), "\n")

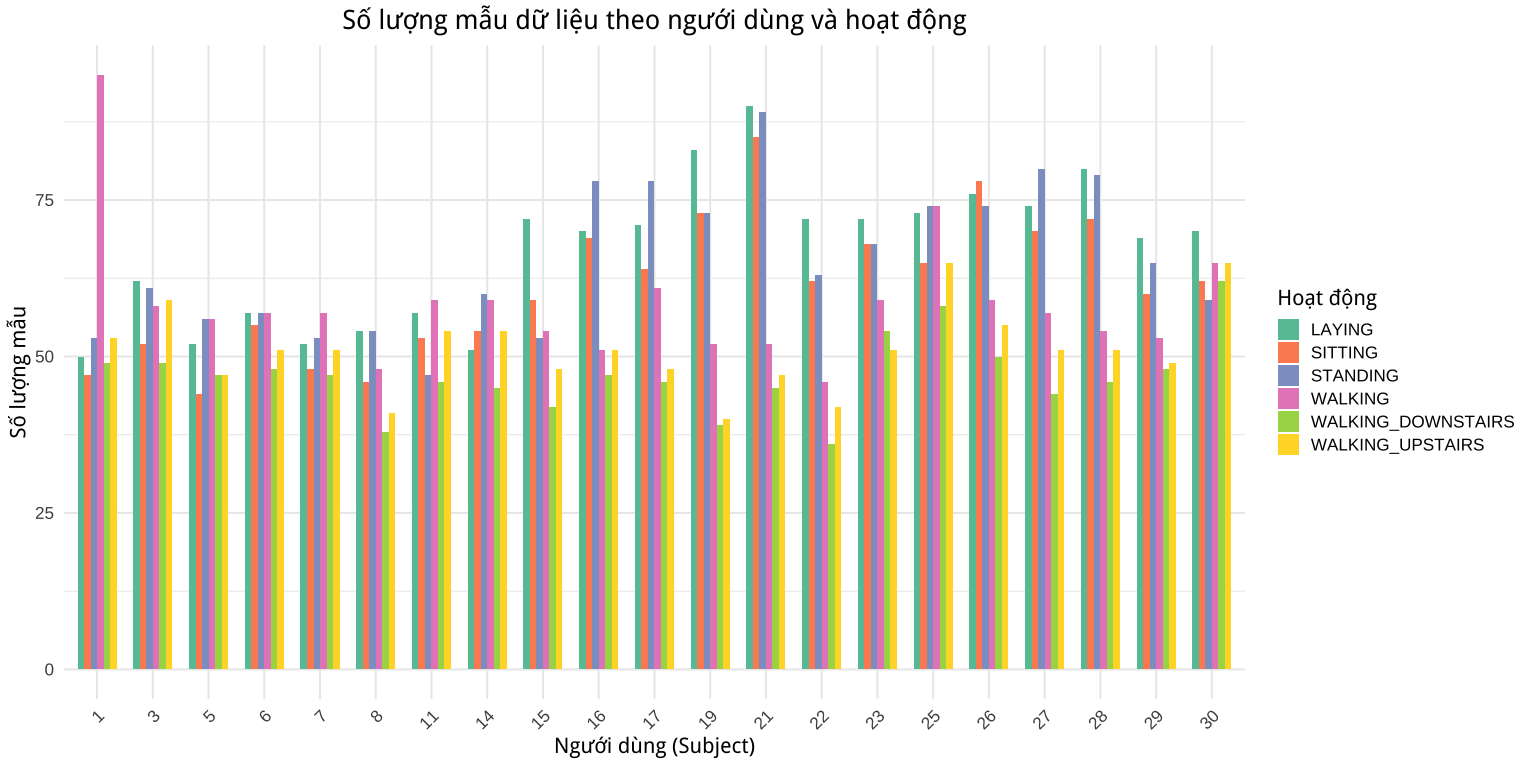
## Số dòng bị trùng lặp trong tập test : 0

Kết quả hiển thị không có dòng nào bị trùng lặp trong cả hai tập dữ liệu (train và test), với tổng số dòng trùng là 0.

columns <- colnames(train)  
columns <- gsub("\\.", "", columns)  
colnames(train) <- columns  
colnames(test) <- columns

Tiến hành xem phân phối dữ liệu qua biểu đồ sau:

library(ggplot2)  
library(showtext)  
showtext\_auto()  
  
ggplot(train, aes(x = factor(subject), fill = Activity)) +  
 geom\_bar(position = "dodge", width = 0.7) +  
 scale\_fill\_brewer(palette = "Set2") +   
 labs(  
 title = "Số lượng mẫu dữ liệu theo người dùng và hoạt động",  
 x = "Người dùng (Subject)",  
 y = "Số lượng mẫu",  
 fill = "Hoạt động"  
 ) +  
 theme\_minimal(base\_size = 16) +  
 theme(  
 plot.title = element\_text(hjust = 0.5, face = "bold", size = 20),  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 legend.position = "right"  
 )



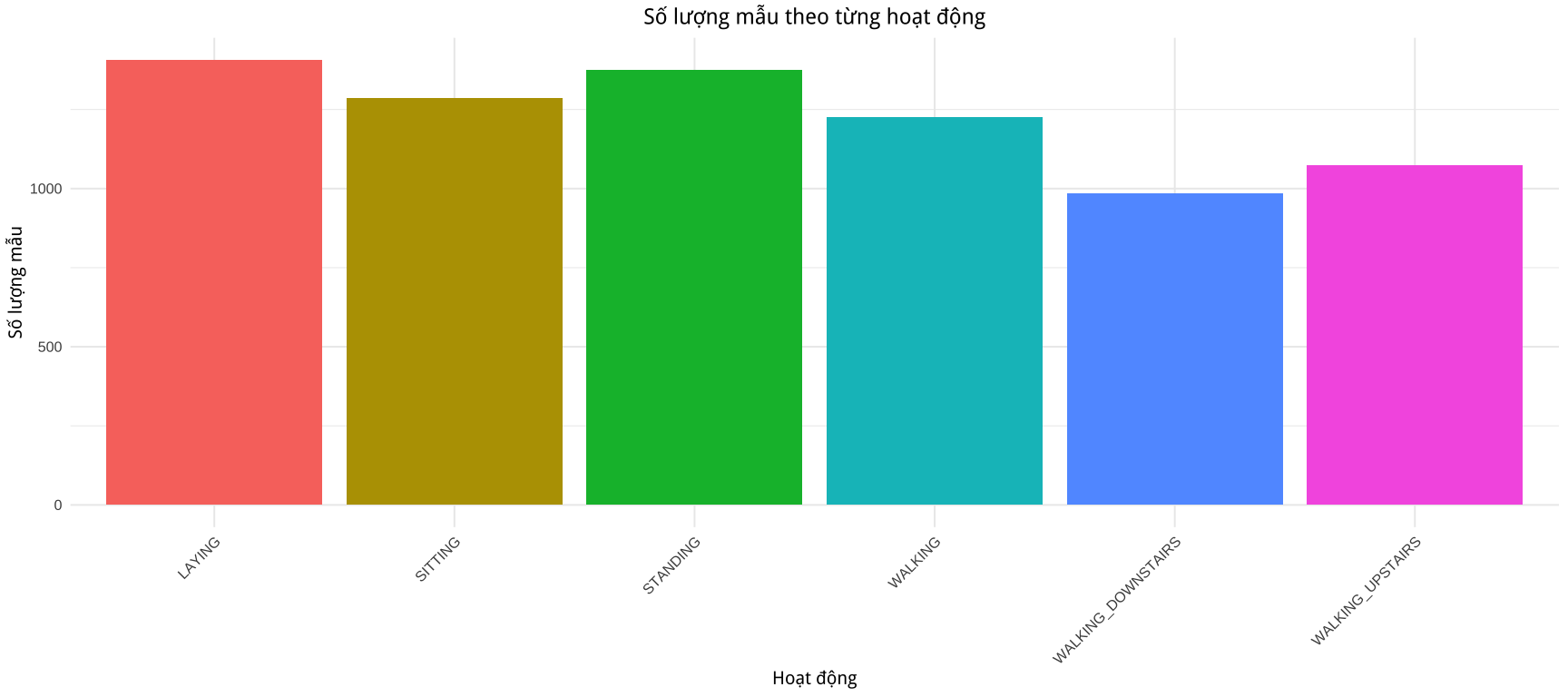
Số lượng mẫu dữ liệu theo người dùng và hoạt động

Nhận xét biểu đồ:

* Nhìn chung, số lượng mẫu giữa các người dùng tương đối đồng đều, thường dao động trong khoảng từ 50 đến 75 mẫu cho mỗi hoạt động.
* Riêng người dùng số 1 có hoạt động WALKING cao bất thường (trên 75 mẫu), vượt xa các hoạt động khác của cùng người dùng này cũng như so với các người dùng còn lại.
* Các hoạt động WALKING\_DOWNSTAIRS và WALKING\_UPSTAIRS thường có số lượng mẫu thấp hơn các hoạt động khác, đặc biệt là WALKING\_UPSTAIRS.
* Các hoạt động tĩnh như LAYING, SITTING và STANDING nhìn chung ổn định, không có sự chênh lệch đáng kể giữa các người dùng.

=> Dữ liệu khá cân bằng giữa các đối tượng và hoạt động, ngoại trừ trường hợp đặc biệt ở người dùng số 1 với hoạt động WALKING có thể cần xem xét thêm để tránh ảnh hưởng đến phân tích và xây dựng mô hình.

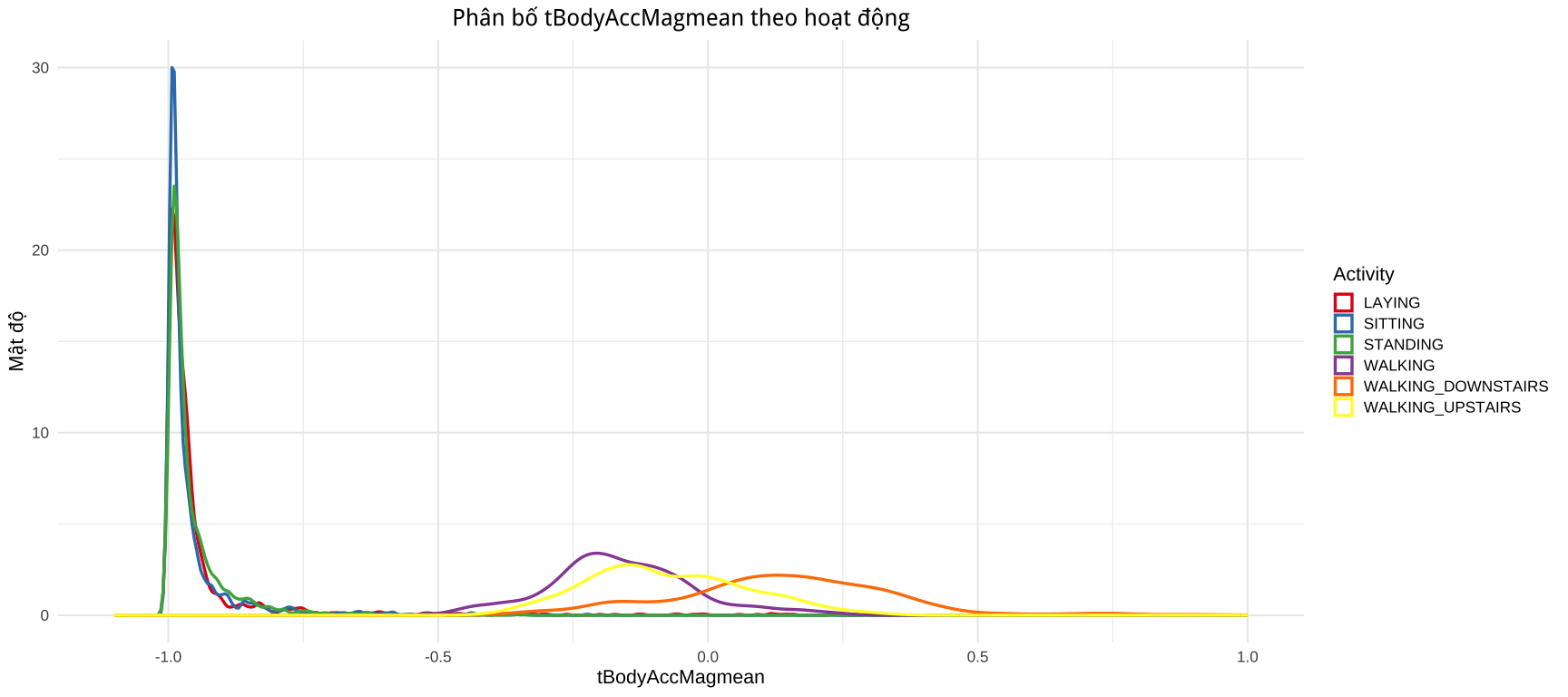
library(ggplot2)  
  
ggplot(train, aes(x = Activity, fill = Activity)) +  
 geom\_bar() +  
 labs(  
 title = "Số lượng mẫu theo từng hoạt động",  
 x = "Hoạt động",  
 y = "Số lượng mẫu"  
 ) +  
 theme\_minimal(base\_size = 15) +  
 theme(  
 axis.text.x = element\_text(angle = 45, vjust = 1, hjust=1),  
 plot.title = element\_text(hjust = 0.5, face = "bold")  
 ) +  
 guides(fill = "none")



Số lượng mẫu theo từng hoạt động

Số lượng mẫu cho mỗi hoạt động dao động trong khoảng 1000 mẫu đến 1200 mẫu. Các hoạt động tĩnh như laying, sitting, standing có xu hướng chiếm tỷ lệ cao hơn một chút so với các hoạt động di chuyển như walking, walking\_upstairs, walking\_downstairs. Mức độ chênh lệch không qua lớn giữa các nhóm, là điểm thuận lợi cho các mô hình học máy tránh bị lệch nhãn và đảm bảo khả năng học đều giữa các lớp.

library(ggplot2)  
  
ggplot(train, aes(x = tBodyAccMagmean, color = Activity)) +  
 geom\_density(size = 1.2) +  
 scale\_color\_brewer(palette = "Set1") +  
 scale\_x\_continuous(limits = c(-1.1, 1)) +   
 labs(  
 title = "Phân bố tBodyAccMagmean theo hoạt động",  
 x = "tBodyAccMagmean",  
 y = "Mật độ"  
 ) +  
 theme\_minimal(base\_size = 16) +  
 theme(  
 plot.title = element\_text(hjust = 0.5)  
 )

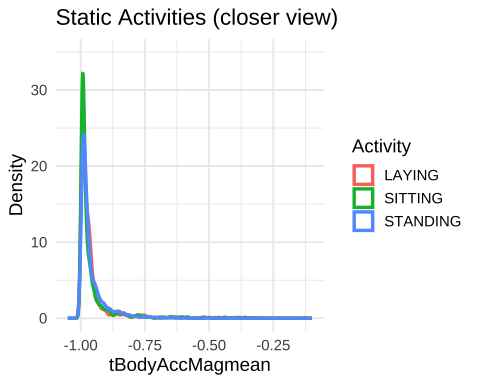


Phân bố tBodyAccMagmean theo hoạt động

Biểu đồ cho thấy biến tBodyAccMagmean phân biệt rõ giữa nhóm hoạt động tĩnh và hoạt động động:

* LAYING, SITTING, STANDING có mật độ tập trung mạnh quanh giá trị âm lớn (gần -1), cho thấy ít dao động, phù hợp với các hoạt động ít vận động.
* Trong khi đó, WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS phân bố lệch phải, trải dài và cao hơn ở các giá trị gần 0, biểu thị mức độ chuyển động nhiều hơn.
* Điều này cho thấy tBodyAccMagmean là biến có khả năng phân biệt tốt giữa hành vi tĩnh và động.

library(ggplot2)  
library(dplyr)  
library(gridExtra)  
  
p1 <- train %>%  
 filter(Activity %in% c("SITTING", "STANDING", "LAYING")) %>%  
 ggplot(aes(x = tBodyAccMagmean, color = Activity)) +  
 geom\_density(size = 1.2) +  
 labs(  
 title = "Static Activities (closer view)",  
 x = "tBodyAccMagmean",  
 y = "Density"  
 ) +  
 xlim(-1.05, -0.1) +  
 ylim(0, 35) +  
 theme\_minimal(base\_size = 14)  
p1



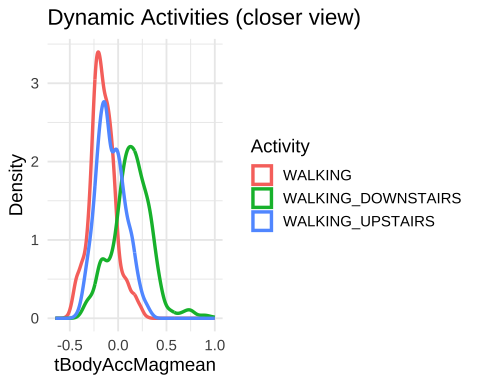
Phân bố tBodyAccMagmean theo hoạt động sitting, standing, laying

Biểu đồ cho thấy 3 hoạt động LAYING, SITTING, và STANDING đều có phân bố giá trị tBodyAccMagmean rất hẹp, tập trung mạnh quanh -1.

* SITTING có đỉnh mật độ cao nhất (~33), cho thấy dữ liệu phân bố rất dày và ổn định ở vùng gia tốc thấp.
* LAYING và STANDING có dạng phân bố tương tự nhưng thấp hơn, cũng tập trung gần -1.

=> Điều này phản ánh rằng các hoạt động tĩnh tạo ra rất ít chuyển động cơ thể, phù hợp với giá trị tBodyAccMagmean thấp và ổn định, tuy nhiên chúng khó phân biệt rõ ràng với nhau chỉ dựa rên đặc trưng này, vì có sự chồng lấn phân bố giữa ba hoạt động.

p2 <- train %>%  
 filter(Activity %in% c("WALKING", "WALKING\_DOWNSTAIRS", "WALKING\_UPSTAIRS")) %>%  
 ggplot(aes(x = tBodyAccMagmean, color = Activity)) +  
 geom\_density(size = 1.2) +  
 labs(  
 title = "Dynamic Activities (closer view)",  
 x = "tBodyAccMagmean",  
 y = "Density"  
 ) + xlim(-0.65, 1) +  
 theme\_minimal(base\_size = 14)  
  
p2



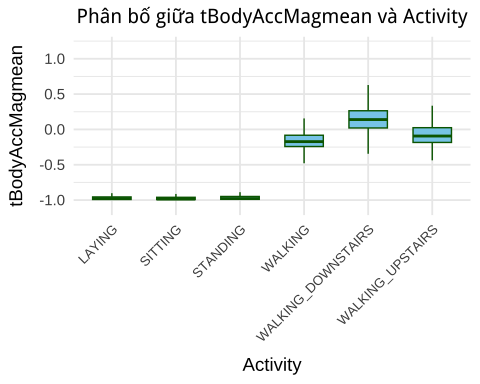
Phân bố tBodyAccMagmean theo hoạt động walking, walking\_dowstairs, walking\_upstairs

Biểu đồ cho thấy sự khác biệt rõ rệt về phân bố giá trị tBodyAccMagmean giữa các hoạt động động:

* WALKING có đỉnh mật độ cao nhất, tập trung quanh giá trị gần -0.1, cho thấy dao động đều và ổn định hơn so với hai hoạt động còn lại.
* WALKING\_UPSTAIRS có phân bố lệch trái hơn, cho thấy cường độ vận động ít hơn WALKING\_DOWNSTAIRS.
* WALKING\_DOWNSTAIRS có đuôi phân bố kéo dài và lan rộng hơn, với giá trị cao hơn về phía dương, phản ánh sự dao động mạnh và không đều khi đi xuống cầu thang.

=> Điều này cho thấy tBodyAccMagmean là một đặc trưng phân biệt tốt giữa các kiểu hoạt động động, đặc biệt là giữa đi lên và đi xuống cầu thang.

library(ggplot2)  
  
ggplot(train, aes(x = Activity, y = tBodyAccMagmean)) +  
 geom\_boxplot(  
 outlier.shape = NA,  
 fill = "skyblue",  
 color = "darkgreen",  
 width = 0.6  
 ) +  
 labs(  
 title = "Phân bố giữa tBodyAccMagmean và Activity",  
 y = "tBodyAccMagmean",  
 x = "Activity"  
 ) +  
 coord\_cartesian(ylim = c(-1.1, 1.2)) +   
 theme\_minimal(base\_size = 14) +  
 theme(  
 plot.title = element\_text(hjust = 0.5, size = 15, face = "bold"),  
 axis.text.x = element\_text(angle = 45, hjust = 1, size = 10),  
 axis.title.x = element\_text(margin = ggplot2::margin(t = 10)),  
 axis.title.y = element\_text(margin = ggplot2::margin(r = 10))  
 )



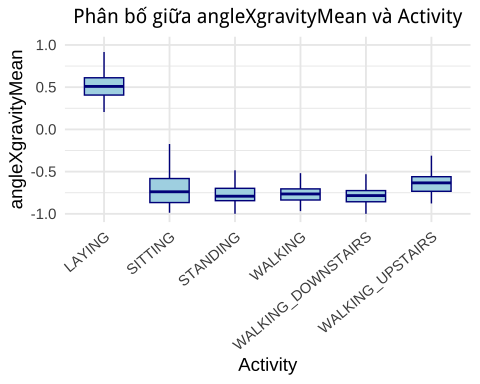
Phân bố giữa tBodyAccMagmean và Activity

Biểu đồ boxplot thể hiện rõ sự khác biệt giữa nhóm hoạt động tĩnh và hoạt động động:

* LAYING, SITTING, STANDING có giá trị tBodyAccMagmean gần như cố định quanh -1, với khoảng biến thiên cực nhỏ → cho thấy rất ít chuyển động.
* Ngược lại, WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS có median cao hơn nhiều và khoảng biến thiên rộng, đặc biệt WALKING\_DOWNSTAIRS có độ dao động lớn nhất → phản ánh cường độ hoạt động mạnh và không đều.

=> Từ đó ta thấy được Biến tBodyAccMagmean có khả năng phân biệt rõ ràng giữa hoạt động tĩnh và động, tuy nhiên khó phân biệt giữa các hoạt động tĩnh với nhau do phân bố quá giống nhau

library(ggplot2)  
  
ggplot(train, aes(x = Activity, y = angleXgravityMean)) +  
 geom\_boxplot(  
 fill = "lightblue",  
 color = "darkblue",  
 outlier.shape = NA,  
 width = 0.6  
 ) +  
 labs(  
 title = "Phân bố giữa angleXgravityMean và Activity",  
 x = "Activity",  
 y = "angleXgravityMean"  
 ) +  
 theme\_minimal(base\_size = 14) +  
 theme(  
 plot.title = element\_text(hjust = 0.5, size = 15, face = "bold"),  
 axis.text.x = element\_text(angle = 40, hjust = 1, size = 11)  
 )

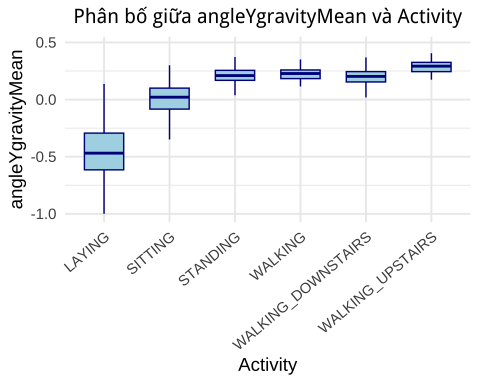


Phân bố giữa angleXgravityMean và Activity

* LAYING có giá trị angleXgravityMean cao vượt trội, median nằm khoảng 0.5 – 0.6, khác biệt rõ rệt so với các hoạt động còn lại.
* Các hoạt động còn lại (SITTING, STANDING, WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS) có giá trị nghiêng về âm, tập trung gần -0.8 đến -0.5, với độ dao động nhỏ và tương đối đồng đều.

=> Điều này cho thấy biến angleXgravityMean có khả năng phân biệt tốt hoạt động nằm (LAYING) so với các hoạt động khác. Tuy nhiên, nó không đủ để phân biệt rõ giữa các hoạt động còn lại, do các boxplot gần như chồng lấn.

library(ggplot2)  
  
ggplot(train, aes(x = Activity, y = angleYgravityMean)) +  
 geom\_boxplot(  
 fill = "lightblue",  
 color = "darkblue",  
 outlier.shape = NA,  
 width = 0.6  
 ) +  
 labs(  
 title = "Phân bố giữa angleYgravityMean và Activity",  
 x = "Activity",  
 y = "angleYgravityMean"  
 ) +  
 theme\_minimal(base\_size = 14) +  
 theme(  
 plot.title = element\_text(hjust = 0.5, size = 15, face = "bold"),  
 axis.text.x = element\_text(angle = 40, hjust = 1, size = 11)  
 )



Sau khi quan sát biểu đồ ta thấy được:

* LAYING có giá trị angleYgravityMean thấp nhất, tập trung quanh -0.5, và có phân bố lệch mạnh về phía âm.
* SITTING có median gần 0 và khoảng biến thiên rộng nhất trong tất cả các hoạt động.
* STANDING, WALKING, WALKING\_DOWNSTAIRS, WALKING\_UPSTAIRS đều có median gần 0.4–0.5, với phân bố rất hẹp và gần như trùng nhau.

=> angleYgravityMean phân biệt rất tốt LAYING với phần còn lại, và một phần phân biệt được SITTING.Tuy nhiên, nó không hiệu quả để phân biệt các hoạt động động vì các boxplot gần như giống nhau.

**Kết luận:**

Đặc trưng tBodyAccMagmean – đại diện cho độ lớn trung bình của gia tốc cơ thể trong mỗi phân đoạn tín hiệu. Biểu đồ mật độ cho thấy ba hoạt động tĩnh đều có phân phối rất hẹp, tập trung gần giá trị -1.0, với đỉnh phân phối cao và sắc. Điều này phản ánh rằng khi người dùng không di chuyển, thiết bị ở thắt lưng ghi nhận gia tốc gần như bằng 0 trong toàn bộ cửa sổ thời gian. Trong đó, hoạt động SITTING có mật độ tập trung cao nhất, cho thấy đây là tư thế có ít dao động nhất; STANDING có phân bố hơi rộng hơn một chút, có thể do người thực hiện có xu hướng chuyển trọng lượng nhẹ nhàng khi đứng. Ngược lại, hoạt động LAYING lại xuất hiện sự phân tán hơn – phản ánh trạng thái nằm có thể mang nhiều tư thế khác nhau (nằm nghiêng, nằm ngửa…).

AngleXgravityMean cho thấy sự khác biệt rõ giữa tư thế nằm và các tư thế còn lại: trong khi LAYING có giá trị trung bình dương lớn (≈ 0.5–0.7), các hoạt động khác có xu hướng âm rõ rệt, thường nằm trong khoảng -0.6 đến -1.0. Là chỉ báo mạnh về góc nghiêng của cơ thể theo trục trước–sau, rất hữu ích trong việc phân biệt trạng thái nghỉ ngơi so với trạng thái hoạt động.

AngleYgravityMean – thể hiện góc nghiêng cơ thể theo trục ngang. LAYING có giá trị âm lớn (≈ -0.5), SITTING nằm quanh giá trị 0, trong khi STANDING và các hoạt động di chuyển đều có giá trị dương và tập trung gần nhau, phản ánh trạng thái thẳng đứng của cơ thể. Các đặc trưng liên quan đến góc độ không gian của thiết bị có khả năng phân biệt rõ tư thế mà không cần đến các đặc trưng vận tốc hay gia tốc.

Phân tích các đặc trưng tBodyAccMagmean, angleXgravityMean và angleYgravityMean không chỉ làm rõ cấu trúc phân bố của dữ liệu, mà còn cho thấy mỗi hoạt động trong bộ dữ liệu HAR đều thể hiện dấu hiệu đặc trưng về mặt hình học hoặc vận động học, đóng vai trò quan trọng trong việc xây dựng các mô hình học máy chính xác nhằm phân loại hành vi người dùng.

# 4 THỰC NGHIỆM

## 4.1 Kiểm tra outlier

Tiến hành kiểm tra outlier trên tập train và tập test, xác định các điểm ngoại lai bằng phương pháp khoảng tứ phân vị (Interquartile Range – IQR). Loại bỏ biến subject và Activity để tạo ra tập dữ liệu sensor\_data, bao gồm 561 đặc trưng cảm biến, với mỗi cột trong tập dữ liệu xác định ranh giới bất thường dựa trên công thức:

Trong đó và là các tứ phân vị thứ nhất và thứ ba tương ứng, và . Mọi giá trị vượt ngoài khoảng này được coi là ngoại lai.

subject\_ids <- train$subject  
activity\_labels <- train$Activity  
sensor\_data <- train[, !(names(train) %in% c("Activity", "subject"))]  
subject\_ids\_test <- test$subject  
activity\_labels\_test <- test$Activity  
sensor\_data\_test <- test[, !(names(test) %in% c("Activity", "subject"))]  
  
is\_outlier\_iqr <- function(x) {  
 Q1 <- quantile(x, 0.25, na.rm = TRUE)  
 Q3 <- quantile(x, 0.75, na.rm = TRUE)  
 IQR\_val <- Q3 - Q1  
 lower\_bound <- Q1 - 1.5 \* IQR\_val  
 upper\_bound <- Q3 + 1.5 \* IQR\_val  
 x < lower\_bound | x > upper\_bound  
}  
  
outlier\_matrix <- mapply(is\_outlier\_iqr, sensor\_data)  
  
total\_outlier\_values <- sum(outlier\_matrix)  
total\_cells <- nrow(sensor\_data) \* ncol(sensor\_data)  
percent\_outlier\_cells <- round(total\_outlier\_values / total\_cells \* 100, 2)  
  
cat("Tổng số giá trị outlier:", total\_outlier\_values, "/", total\_cells, "\n")

## Tổng số giá trị outlier: 163682 / 4124472

cat("Tỷ lệ giá trị outlier:", percent\_outlier\_cells, "%\n")

## Tỷ lệ giá trị outlier: 3.97 %

Kết quả có tổng cộng 163,682 giá trị được đánh dấu là ngoại lai, trên tổng số 4,124,472 giá trị dữ liệu. Điều này tương ứng với tỷ lệ 3.97%, một mức thấp, dữ liệu phần lớn ổn định và sạch.

Tiến hành Chuẩn hóa outlier bằng kỹ thuật cắt ngưỡng (Percentile Capping) nhằm giảm ảnh hưởng của các outlier mà không loại bỏ chúng hoàn toàn khỏi tập dữ liệu. Thay thế các giá trị cực đoan bằng một ngưỡng giới hạn mềm dựa trên các phân vị phần trăm (percentiles) của phân phối dữ liệu. định nghĩa một hàm cap\_outliers() để áp dụng quy tắc sau trên từng cột trong bảng dữ liệu: giá trị nhỏ hơn phân vị 1% (1st percentile) được gán bằng chính giá trị tại phân vị này, giá trị lớn hơn phân vị 99% (99th percentile) được gán bằng giá trị tại phân vị đó, các giá trị nằm giữa hai ngưỡng này được giữ nguyên.

cap\_outliers <- function(df, lower = 0.01, upper = 0.99) {  
 for (col in names(df)) {  
 q\_low <- quantile(df[[col]], lower, na.rm = TRUE)  
 q\_high <- quantile(df[[col]], upper, na.rm = TRUE)  
 df[[col]] <- pmin(pmax(df[[col]], q\_low), q\_high)  
 }  
 df  
 }  
  
 sensor\_data\_capped <- cap\_outliers(sensor\_data)  
 sensor\_data\_test\_capped <- cap\_outliers(sensor\_data\_test)  
  
 train\_capped <- cbind(sensor\_data\_capped, subject = subject\_ids, Activity = activity\_labels)  
 test\_capped <- cbind(sensor\_data\_test\_capped, subject = subject\_ids\_test, Activity = activity\_labels\_test)

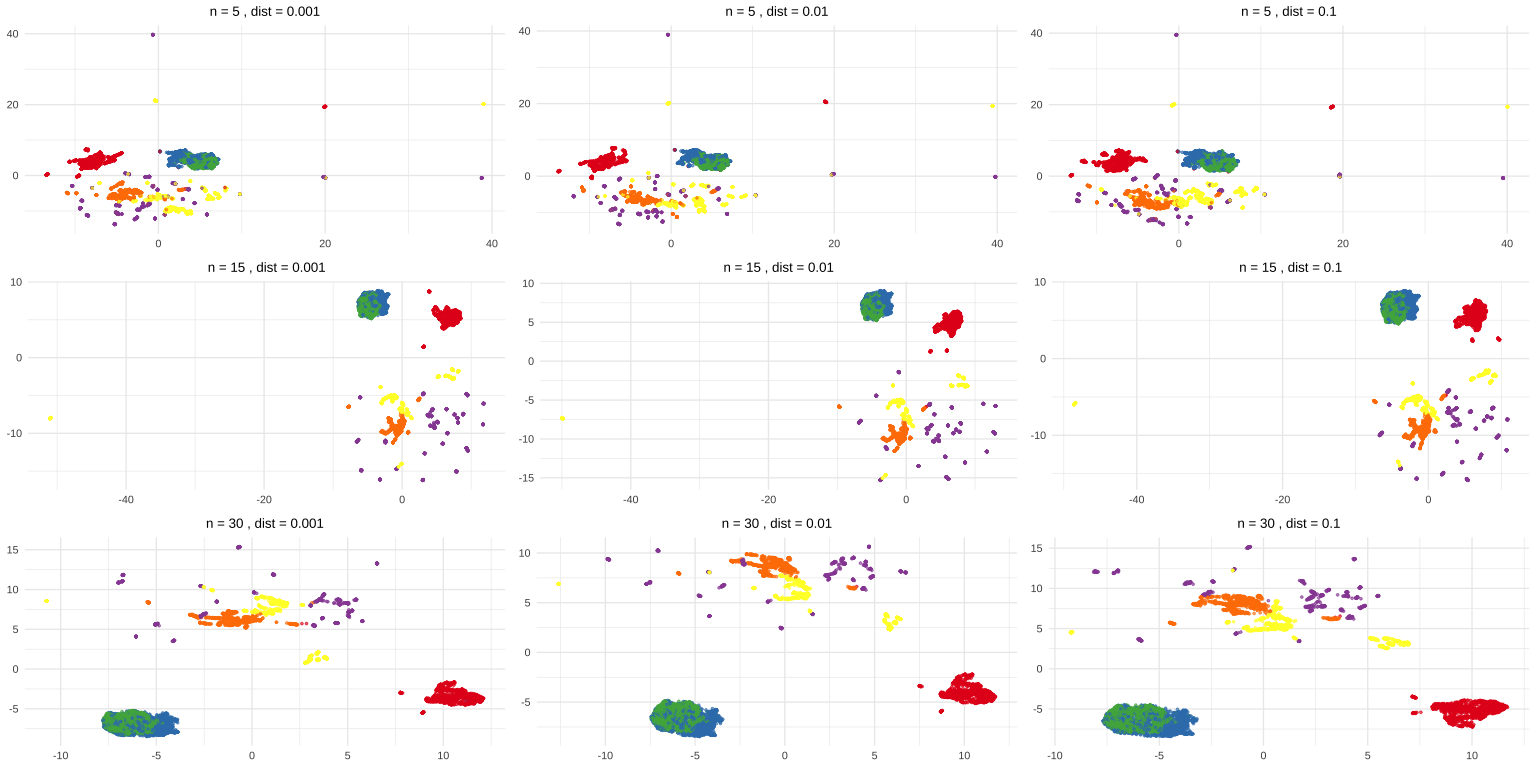
## 4.2 Giảm chiều dữ liệu bằng UMAP

Kỹ thuật UMAP sử dụng để giảm chiều dữ liệu nhằm mục tiêu trực quan hóa và đánh giá khả năng phân tách giữa các nhãn hoạt động khác nhau. Việc giảm chiều không gian về 2 chiều giúp quan sát cấu trúc không gian nhúng của dữ liệu gốc gồm 561 đặc trưng đầu vào.

Thử nghiệm hệ thống trên lưới các tổ hợp siêu tham số khác nhau, số lượng lân cận (n\_neighbors): 5, 15, 30 thể hiện mức độ chú trọng vào cấu trúc cục bộ hay toàn cục. Giá trị nhỏ nhấn mạnh mối quan hệ địa phương, giá trị lớn mang tính toàn cục cao hơn. Khoảng cách tối thiểu (min\_dist): 0.001, 0.01, 0.1 điều chỉnh mức độ nén các cụm dữ liệu trong không gian nhúng. Giá trị càng nhỏ thì các cụm càng bị nén lại gần nhau, cho phép bảo toàn mật độ; giá trị lớn làm dàn trải không gian nhúng.

Kết quả được trình bày thông qua ma trận trực quan gồm 9 biểu đồ, tương ứng với 9 tổ hợp siêu tham số khác nhau. Mỗi điểm trên biểu đồ đại diện cho một mẫu dữ liệu cảm biến, được tô màu theo nhãn Activity.

library(umap)  
library(ggplot2)  
library(gridExtra)  
  
perform\_umap\_grid <- function(X\_data, y\_data,  
 neighbors\_list = c(5, 15, 30),  
 min\_dist\_list = c(0.001, 0.01, 0.1)) {  
 plots <- list()  
 index <- 1  
  
 for (n in neighbors\_list) {  
 for (d in min\_dist\_list) {  
 config <- umap.defaults  
 config$n\_neighbors <- n  
 config$min\_dist <- d  
 config$n\_components <- 2  
  
 umap\_result <- umap(X\_data, config = config)  
 df <- as.data.frame(umap\_result$layout)  
 colnames(df) <- c("x", "y")  
 df$label <- y\_data  
  
 p <- ggplot(df, aes(x = x, y = y, color = label)) +  
 geom\_point(size = 0.6, alpha = 0.6) +  
 scale\_color\_brewer(palette = "Set1") +  
 theme\_minimal(base\_size = 10) +  
 labs(title = paste("n =", n, ", dist =", d), x = NULL, y = NULL) +  
 theme(legend.position = "none",  
 plot.title = element\_text(size = 10, hjust = 0.5))  
  
 plots[[index]] <- p  
 index <- index + 1  
 }  
 }  
  
 do.call(grid.arrange, c(plots, ncol = length(min\_dist\_list)))  
}  
  
perform\_umap\_grid(  
 X\_data = sensor\_data\_capped,  
 y\_data = train$Activity,  
 neighbors\_list = c(5, 15, 30),  
 min\_dist\_list = c(0.001, 0.01, 0.1)  
)



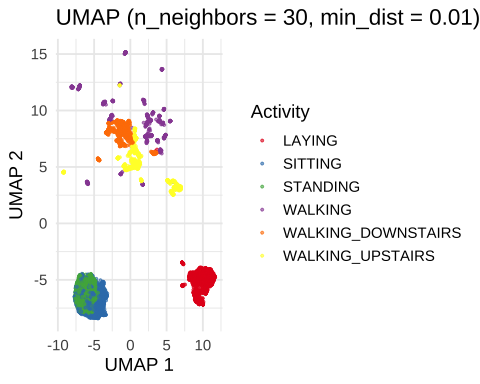
trực quan hóa UMAP

Khi **n\_neighbors = 5** mạng lưới lân cận nhỏ, cấu trúc cục bộ lỏng lẻo và thiếu kết nối giữa các cụm, các cụm co mạnh nhưng phân bố không đồng đều. **n neighbors = 15** cấu trúc dữ liệu trở nên rõ ràng hơn. Khi kết hợp với min dist = 0.01, các cụm đại diện cho các chức năng khác nhau bắt đầu được hình thành rõ ràng và phân chia khá mạch lạc, thể hiện sự phân cấp cao giữa các đối tượng, các cụm nhỏ như sitting và lay ing bắt đầu được gom sát nhau. **n\_neighbors = 30** thể hiện cấu trúc dữ liệu ổn định hơn. Dữ liệu được sắp xếp hợp lý, thấy được sự ổn định trong phân cụm. Đặc biệt với min\_dist = 0.1, dữ liệu dàn rộng hơn trong không gian nhúng, phù hợp cho việc phân tích tổng thể.

Kết quả phân tích thấy được các siêu tham số n neighbors = 30 và min dist = 0.1 tạo ra hình ảnh không gian nhúng chính xác và có sự phân biệt tốt nhất giữa các hoạt động. Đây là cơ sở quan trọng giúp xác định cấu hình tối ưu khi dùng UMAP như một bước tiền xử lý trong pipeline học máy.

=> Dựa vào kết phân tích hệ thống các siêu tham số trong kỹ thuật giảm chiều UMAP thì cấu hình n\_neighbors = 30 và min\_dist = 0.01 đã được xác định là một trong những cấu hình hiệu quả nhất để đảm bảo cấu trúc dữ liệu và phân tách các nhãn rõ ràng. Chúng em sẽ tiến hành minh họa kết quả giảm chiều dữ liệu cảm biến từ 561 đặc trưng về 2 chiều, áp dụng các cấu hinhg tối ưu nêu trên, mỗi điểm là đại diện cho một mẫu dữ liệu và được tô mày theo loại hoạt động được gán nhãn.

library(umap)  
library(ggplot2)  
  
config <- umap.defaults  
config$n\_neighbors <- 30  
config$min\_dist <- 0.1  
config$n\_components <- 2  
  
umap\_result <- umap(sensor\_data\_capped, config = config)  
  
umap\_df <- as.data.frame(umap\_result$layout)  
colnames(umap\_df) <- c("UMAP\_1", "UMAP\_2")  
umap\_df$Activity <- train$Activity  
  
ggplot(umap\_df, aes(x = UMAP\_1, y = UMAP\_2, color = Activity)) +  
 geom\_point(size = 0.7, alpha = 0.6) +  
 scale\_color\_brewer(palette = "Set1") +  
 theme\_minimal(base\_size = 14) +  
 labs(  
 title = "UMAP (n\_neighbors = 30, min\_dist = 0.01)",  
 x = "UMAP 1", y = "UMAP 2"  
 )



trực quan hóa kết quả giảm chiều dữ liệu bằng UMAP

Các hoạt động như LAYING, SITTING và STANDING tạo thành ba cụm rõ ràng, thấy được tín hiệu cảm biến ở trong thái tĩnh có sự ổn định cao và dễ phân biệt.

Các hoạt động như WALKING, WALKING\_DOWNSTAIRS, WALKING\_UPSTAIRS có xu hướng gần nhau hơn trong không gian 2 chiều, mặc dù vẫn tạo thành các cụm riêng biệt, nhưng vẫn có vẻ chồng lấn, cho thấy được biến thiên phức tạp của vận động trong thực tế. Thấy được sự tách biệt rõ ràng giữa các nhóm động và tĩnh. Đây là bước tiền xử lý và kiểm định trực quan quan trọng trước khi đưa dữ liệu vào các thuật toán học máy như Random Forest, SVM, Logistic Regression và k-NN.

## 4.3 Chuẩn bị dữ liệu cho mô hình học máy

Sau khi hoàn thành các bước tiền xử lý, tập dữ liệu được chia thành hai thành phần chính: ma trận đặc trưng đầu vào và nhãn đầu ra tương ứng. Các biến định danh như subject và Activity được loại bỏ khỏi ma trận đặc trưng, nhằm đảm bảo mô hình chỉ học từ các tín hiệu cảm biến mà không bị ảnh hưởng bởi thông tin định danh.

X\_train <- train\_capped[, !(names(train\_capped) %in% c("subject", "Activity"))]  
y\_train <- train\_capped$Activity  
  
X\_test <- test\_capped[, !(names(test\_capped) %in% c("subject", "Activity"))]  
y\_test <- test\_capped$Activity  
  
dim(X\_train)

## [1] 7352 561

length(y\_train)

## [1] 7352

dim(X\_test)

## [1] 2947 561

length(y\_test)

## [1] 2947

## 4.4 xây dựng hàm đánh giá tổng quá mô hình học máy

Để đảm bảo sự thống nhất và tái sử dụng trong việc đào tạo và đánh giá các mô hình học máy, một hàm tổng quát có tên train\_and\_evaluate\_model đã được thiết lập.Hàm thực hiện quy trình huấn luyện, dự đoán, đánh giá và trực quan hóa kết quả của một mô hình phân loại với cấu trúc linh hoạt, cho phép sử dụng nhiều thuật toán khác nhau như Random Forest, SVM, Logistic Regression, hoặc k-NN.

Hàm thiết ké với đầu vào linh hoạt bao gồm tên mô hình (model\_name), dữ liệu huấn luyện và kiểm tra (X\_train, y\_train, X\_test, y\_test), danh sách nhãn phân loại (class\_labels), bảng tham số tinh chỉnh (tuneGrid) và số lượng folds trong quy trình kiểm định chéo (k\_folds). Chuyển đổi các nhãn phân loại về kiểu factor để tương tích với các mô hình phân loại đa lớp trong môi trường R.

Quá trình huấn luyện mô hình được thực hiện thông qua hàm train() từ thưu viện cảet, cho phép sử dụng kiểm định chéo k-fold đồng thời tối ưu hóa các tham số theo lưới định nghĩa trước đó. Sau khi huấn luyện xong, mô hình được sử dụng để dự đoán trên dữ liệu kiểm tra, sau đó tính toán độ chính xác tổng thể và tạo ra ma trận nhầm lẫn để đánh giá mô hình. Bên cạnh đó, hàm còn cung cấp chi tiết các chỉ số quan trọng như recision, Recall, F1-score, được tính thông qua hàm confusionMatrix() từ gói caret.

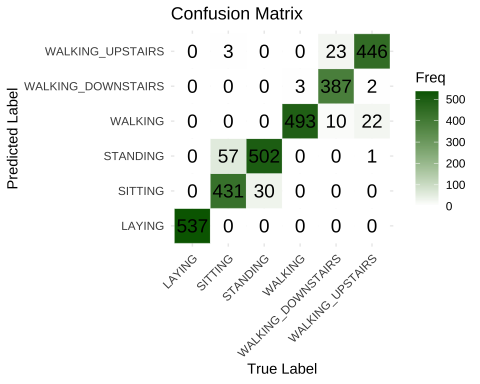
Ngoài việc đánh giá hiệu suất cho từng lớp riêng biệt, hàm train\_and\_evaluate\_model còn tiến hành tính toán các chỉ số trung bình ở cả hai mức độ: macro average (trung bình không trọng số giữa các lớp) và weighted average (trung bình có trọng số theo số mẫu trên mỗi lớp). Điều này giúp cung cấp một cái nhìn toàn diện hơn về hiệu suất mô hình, đặc biệt hữu ích trong các bài toán phân loại không cân bằng. Hàm còn sử dụng ggplot2 để sinh ra ma trận nhầm lẫn dưới dạng đồ họa giúp dễ dàng phát hiện những nhóm hành vi dễ bị mô hình nhầm lẫn.

train\_and\_evaluate\_model <- function(  
 model\_name,  
 X\_train, y\_train, X\_test, y\_test,  
 class\_labels,  
 tuneGrid = NULL,  
 k\_folds = 5  
) {  
 if (!require(caret)) install.packages("caret", quiet = TRUE)  
 if (!require(ggplot2)) install.packages("ggplot2", quiet = TRUE)  
  
 library(caret)  
 library(ggplot2)  
  
 y\_train <- as.factor(y\_train)  
 y\_test <- as.factor(y\_test)  
  
 ctrl <- trainControl(method = "cv", number = k\_folds)  
  
 set.seed(123)  
 model\_fit <- train(  
 x = X\_train,  
 y = y\_train,  
 method = model\_name,  
 trControl = ctrl,  
 tuneGrid = tuneGrid  
 )  
  
 best\_param <- model\_fit$bestTune  
  
 final\_model <- train(  
 x = X\_train,  
 y = y\_train,  
 method = model\_name,  
 trControl = trainControl(method = "none"),  
 tuneGrid = best\_param  
 )  
  
 y\_pred <- predict(final\_model, X\_test)  
  
 cm <- confusionMatrix(y\_pred, y\_test)  
 report <- cm$byClass[, c("Precision", "Recall", "F1")]  
 test\_acc <- mean(y\_pred == y\_test)  
  
 macro\_precision <- mean(report[, "Precision"], na.rm = TRUE)  
 macro\_recall <- mean(report[, "Recall"], na.rm = TRUE)  
 macro\_f1 <- mean(report[, "F1"], na.rm = TRUE)  
  
 support <- as.numeric(table(y\_test))  
 weighted\_precision <- weighted.mean(report[, "Precision"], w = support, na.rm = TRUE)  
 weighted\_recall <- weighted.mean(report[, "Recall"], w = support, na.rm = TRUE)  
 weighted\_f1 <- weighted.mean(report[, "F1"], w = support, na.rm = TRUE)  
  
 cat('--------------------------\n')  
 cat('| Best Parameters |\n')  
 cat('--------------------------\n')  
 for (param in names(best\_param)) {  
 cat(sprintf("%s = %s\n", param, best\_param[[param]]))  
 }  
  
 cat('--------------------------\n')  
 cat('| Best CV Accuracy |\n')  
 cat('--------------------------\n')  
 print(round(max(model\_fit$results$Accuracy), 4))  
  
 cat('--------------------------\n')  
 cat('| Test Accuracy |\n')  
 cat('--------------------------\n')  
 print(round(test\_acc, 4))  
  
 cat('--------------------------\n')  
 cat('| Precision / Recall / F1 |\n')  
 cat('--------------------------\n')  
 print(round(report, 4))  
  
 cat('------------------------------\n')  
 cat('| Macro Avg (All Classes) |\n')  
 cat('------------------------------\n')  
 cat(sprintf("Precision: %.4f\nRecall : %.4f\nF1-score : %.4f\n",  
 macro\_precision, macro\_recall, macro\_f1))  
  
 cat('------------------------------\n')  
 cat('| Weighted Avg (By Support) |\n')  
 cat('------------------------------\n')  
 cat(sprintf("Precision: %.4f\nRecall : %.4f\nF1-score : %.4f\n",  
 weighted\_precision, weighted\_recall, weighted\_f1))  
  
 cm\_table <- as.data.frame(cm$table)  
 colnames(cm\_table) <- c("Predicted", "Actual", "Freq")  
  
 p <- ggplot(data = cm\_table, aes(x = Actual, y = Predicted, fill = Freq)) +  
 geom\_tile(color = "white") +  
 geom\_text(aes(label = Freq), color = "black", size = 5) +  
 scale\_fill\_gradient(low = "white", high = "darkgreen") +  
 labs(title = "Confusion Matrix", x = "True Label", y = "Predicted Label") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
 print(p)  
  
 return(list(  
 model = final\_model,  
 best\_parameters = best\_param,  
 best\_cv\_accuracy = max(model\_fit$results$Accuracy),  
 test\_accuracy = test\_acc,  
 report = report,  
 macro\_metrics = c(Precision = macro\_precision, Recall = macro\_recall, F1 = macro\_f1),  
 weighted\_metrics = c(Precision = weighted\_precision, Recall = weighted\_recall, F1 = weighted\_f1),  
 confusion\_matrix = cm$table,  
 prediction = y\_pred  
 ))  
}

Trước tiên mô hình hồi quy logistic được triển khai thông qua gói glmnet, với việc tinh chỉnh lưới tham số alpha và lambda, tham số alpha kiểm soát mức độ pha trộn giữa Lasso (L1) và Ridge (L2), trong khi lambda điều chỉnh cường độ của hình phạt. Thử nghiệm với nhiều giá trị lambda đảo ngược (1/λ) cho phép kiểm tra ảnh hưởng của việc điều chỉnh trọng số lên khả năng phân loại.

grid\_logistic <- expand.grid(  
 alpha = c(0, 1),  
 lambda = 1 / c(0.01, 0.1, 1, 10, 20, 30)  
)  
  
results\_log <- train\_and\_evaluate\_model(  
 model\_name = "glmnet",  
 X\_train = X\_train,  
 y\_train = y\_train,  
 X\_test = X\_test,   
 y\_test = y\_test,  
 class\_labels = labels,  
 tuneGrid = grid\_logistic,  
 k\_folds = 5  
)

## --------------------------  
## | Best Parameters |  
## --------------------------  
## alpha = 0  
## lambda = 0.0333333333333333  
## --------------------------  
## | Best CV Accuracy |  
## --------------------------  
## [1] 0.9689  
## --------------------------  
## | Test Accuracy |  
## --------------------------  
## [1] 0.9488  
## --------------------------  
## | Precision / Recall / F1 |  
## --------------------------  
## Precision Recall F1  
## Class: LAYING 1.0000 1.0000 1.0000  
## Class: SITTING 0.9349 0.8778 0.9055  
## Class: STANDING 0.8964 0.9436 0.9194  
## Class: WALKING 0.9390 0.9940 0.9657  
## Class: WALKING\_DOWNSTAIRS 0.9872 0.9214 0.9532  
## Class: WALKING\_UPSTAIRS 0.9449 0.9469 0.9459  
## ------------------------------  
## | Macro Avg (All Classes) |  
## ------------------------------  
## Precision: 0.9504  
## Recall : 0.9473  
## F1-score : 0.9483  
## ------------------------------  
## | Weighted Avg (By Support) |  
## ------------------------------  
## Precision: 0.9496  
## Recall : 0.9488  
## F1-score : 0.9486



confusion matrix Logistic Regression

Mô hình Logistic Regression với lựa chọn siêu tham số tối ưu (alpha = 0, lambda = 0.0333) đạt độ chính xác rất cao trên tập kiểm tra (94.88%) và kiểm định chéo (96.89%), cho thấy mô hình tổng quát hóa tốt và không bị overfitting.

Nhìn vào confusion matrix, phần lớn các dự đoán đều nằm trên đường chéo chính, thể hiện mô hình đã phân loại chính xác đa số các trường hợp. Một số điểm đáng chú ý:

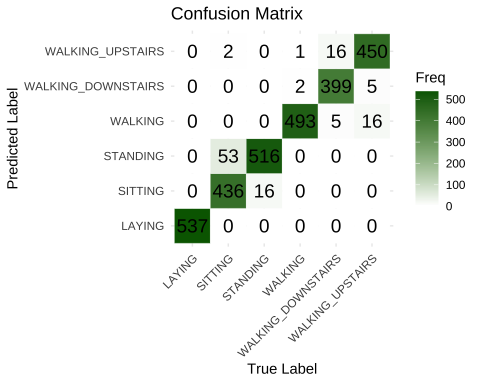
* LAYING được mô hình nhận diện chính xác tuyệt đối với 537/537 mẫu, phản ánh đặc trưng tín hiệu riêng biệt và dễ nhận dạng.
* WALKING cũng đạt độ chính xác rất cao, chỉ nhầm lẫn rất ít sang WALKING\_UPSTAIRS (22 mẫu) và WALKING\_DOWNSTAIRS (10 mẫu), phù hợp với sự tương đồng trong hoạt động vận động.
* SITTING và STANDING là hai lớp dễ nhầm lẫn lẫn nhau: 57 mẫu STANDING bị dự đoán nhầm sang SITTING, và 30 mẫu SITTING nhầm sang STANDING, điều này là hợp lý do hai tư thế này có tín hiệu khá giống nhau.
* WALKING\_UPSTAIRS vẫn là lớp có nhiều nhầm lẫn nhất, với 446 mẫu bị dự đoán nhầm sang các hành vi khác như WALKING (23 mẫu) hoặc WALKING\_DOWNSTAIRS (3 mẫu), cho thấy mô hình gặp khó khăn trong việc phân biệt các hoạt động di chuyển gần giống nhau.

=> Tổng thể, Logistic Regression cho kết quả rất ấn tượng trên tập dữ liệu này, đặc biệt là ở các lớp có đặc trưng rõ ràng. Tuy nhiên, vẫn tồn tại sự nhầm lẫn ở các lớp có hành vi tương đồng, điều này có thể được cải thiện bằng cách bổ sung thêm đặc trưng hoặc thử các mô hình phi tuyến khác.

Sau khi đánh giá Logistic Regression, ta tiếp tục thử nghiệm với mô hình SVM tuyến tính để so sánh hiệu quả giữa hai phương pháp phân loại tuyến tính. Mô hình SVM tuyến tính được thực hiện với lưới tham số c. Các giá trị C thấp khuyến khích mô hình đơn giản, trong khi giá trị lớn hướng tới việc giảm thiểu lỗi huấn luyện. Lựa chọn tập các giá trị c từ 0.125 đến 16 giúp đánh giá độ nhạy của SVM với mức độ điều chuẩn khác nhau.

grid\_linear\_svc <- expand.grid(C = c(0.125, 0.5, 1, 2, 8, 16))  
  
results\_svc <- train\_and\_evaluate\_model(  
 model\_name = "svmLinear",  
 X\_train = X\_train, y\_train = y\_train,  
 X\_test = X\_test, y\_test = y\_test,  
 class\_labels = labels,  
 tuneGrid = grid\_linear\_svc,  
 k\_folds = 5  
)

## --------------------------  
## | Best Parameters |  
## --------------------------  
## C = 0.125  
## --------------------------  
## | Best CV Accuracy |  
## --------------------------  
## [1] 0.9845  
## --------------------------  
## | Test Accuracy |  
## --------------------------  
## [1] 0.9606  
## --------------------------  
## | Precision / Recall / F1 |  
## --------------------------  
## Precision Recall F1  
## Class: LAYING 1.0000 1.0000 1.0000  
## Class: SITTING 0.9646 0.8880 0.9247  
## Class: STANDING 0.9069 0.9699 0.9373  
## Class: WALKING 0.9591 0.9940 0.9762  
## Class: WALKING\_DOWNSTAIRS 0.9828 0.9500 0.9661  
## Class: WALKING\_UPSTAIRS 0.9595 0.9554 0.9574  
## ------------------------------  
## | Macro Avg (All Classes) |  
## ------------------------------  
## Precision: 0.9621  
## Recall : 0.9595  
## F1-score : 0.9603  
## ------------------------------  
## | Weighted Avg (By Support) |  
## ------------------------------  
## Precision: 0.9615  
## Recall : 0.9606  
## F1-score : 0.9605



confusion matrix SVC

Mô hình SVM tuyến tính cho kết quả rất tốt trên tập kiểm tra, với độ chính xác đạt 96.06%. Đồng thời, các chỉ số đánh giá F1-score trung bình cũng rất cao (Macro F1 = 0.9603, Weighted F1 = 0.9605), cho thấy mô hình hoạt động hiệu quả và khá ổn định trên cả các lớp phổ biến lẫn các lớp ít xuất hiện.

Khi quan sát ma trận nhầm lẫn, ta thấy phần lớn các điểm dữ liệu đều được mô hình phân loại đúng, thể hiện qua các giá trị lớn nằm trên đường chéo chính. Tuy nhiên, vẫn còn một số lỗi nhầm lẫn nhỏ:

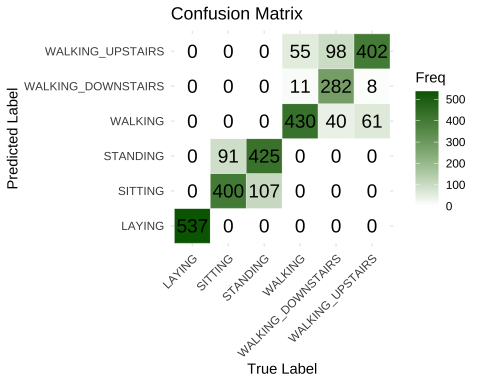
* SITTING và STANDING đôi lúc bị nhầm lẫn cho nhau (ví dụ: có 16 mẫu SITTING bị nhầm thành STANDING, và 53 mẫu STANDING bị dự đoán thành SITTING). Điều này dễ hiểu vì hai hoạt động này đều thuộc nhóm tĩnh và có tín hiệu khá giống nhau.
* Một vài mẫu WALKING bị nhầm sang WALKING\_UPSTAIRS hoặc WALKING\_DOWNSTAIRS, phản ánh sự tương đồng giữa các hành vi di chuyển.
* Các lớp như LAYING, WALKING và WALKING\_DOWNSTAIRS được mô hình phân loại rất chính xác, hầu như không có nhầm lẫn.

=> Tổng thể, mô hình SVM tuyến tính phân loại khá tốt và ổn định, đặc biệt hiệu quả với các hành vi có đặc trưng rõ ràng. Những lỗi nhỏ chủ yếu xảy ra ở các hành vi có tín hiệu tương tự nhau, điều này là hợp lý trong bài toán phân loại hoạt động con người.

Tiếp theo, ta tiến hành xây dựng và đánh giá mô hình Decision Tree Classifier nhằm so sánh hiệu quả với các mô hình tuyến tính đã triển khai trước đó. Mô hình Decision Tree Classifier được thực hiện thông qua rpart2, một biến thể của cây quyết định truyền thống cho phép tối ưu độ sâu cây (maxdepth). Điều chỉnh độ sâu của cây thông qua grid\_dt giúp kiểm soát mức độ phức tạp, tránh việc cây học thuộc dữ liệu huấn luyện. Các giá trị từ 3 đến 9 được thử nghiệm để tìm ra cấu trúc cây phù hợp nhất với độ phức tạp của dữ liệu. Quá trình huấn luyện và đánh giá được thực hiện đồng bộ trong hàm train\_and\_evaluate\_model() với xác thực chéo k-fold (k = 5), cho phép đo lường hiệu suất mô hình một cách khách quan.

grid\_dt <- expand.grid(maxdepth = seq(3, 9, by = 2))  
  
results\_dt <- train\_and\_evaluate\_model(  
 model\_name = "rpart2",  
 X\_train = X\_train,  
 y\_train = y\_train,  
 X\_test = X\_test,  
 y\_test = y\_test,  
 class\_labels = labels,  
 tuneGrid = grid\_dt,  
 k\_folds = 5  
)

## --------------------------  
## | Best Parameters |  
## --------------------------  
## maxdepth = 7  
## --------------------------  
## | Best CV Accuracy |  
## --------------------------  
## [1] 0.886  
## --------------------------  
## | Test Accuracy |  
## --------------------------  
## [1] 0.8402  
## --------------------------  
## | Precision / Recall / F1 |  
## --------------------------  
## Precision Recall F1  
## Class: LAYING 1.0000 1.0000 1.0000  
## Class: SITTING 0.7890 0.8147 0.8016  
## Class: STANDING 0.8236 0.7989 0.8111  
## Class: WALKING 0.8098 0.8669 0.8374  
## Class: WALKING\_DOWNSTAIRS 0.9369 0.6714 0.7822  
## Class: WALKING\_UPSTAIRS 0.7243 0.8535 0.7836  
## ------------------------------  
## | Macro Avg (All Classes) |  
## ------------------------------  
## Precision: 0.8473  
## Recall : 0.8342  
## F1-score : 0.8360  
## ------------------------------  
## | Weighted Avg (By Support) |  
## ------------------------------  
## Precision: 0.8479  
## Recall : 0.8402  
## F1-score : 0.8399



confusion matrix decision tree

Mô hình Decision Tree với độ sâu tối ưu là maxdepth = 7 đạt độ chính xác kiểm định chéo ở mức 88.60% và độ chính xác trên tập kiểm tra là 84.02%. So với Logistic Regression và SVM, độ chính xác của mô hình này thấp hơn, cho thấy khả năng tổng quát hóa của cây quyết định có phần hạn chế hơn trong bài toán này.

Quan sát confusion matrix cho thấy một số xu hướng nhầm lẫn điển hình:

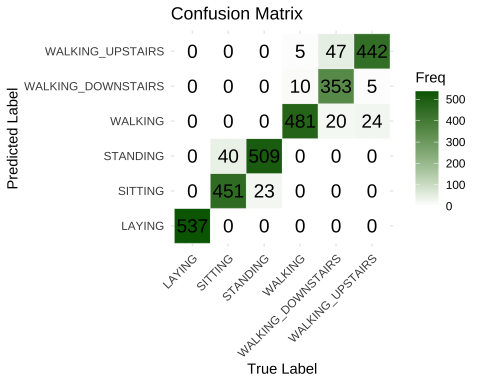
* LAYING tiếp tục là lớp được phân loại chính xác tuyệt đối với 537/537 mẫu, khẳng định đặc trưng tín hiệu rõ ràng của hoạt động này.
* WALKING có hiệu suất khá tốt, tuy có 40 mẫu nhầm sang WALKING\_DOWNSTAIRS và 61 mẫu nhầm sang WALKING\_UPSTAIRS phản ánh sự chồng lấn về đặc trưng giữa các hoạt động di chuyển.
* SITTING và STANDING tiếp tục là cặp lớp dễ nhầm lẫn nhất: 91 mẫu STANDING bị nhầm sang SITTING, và 107 mẫu SITTING bị nhầm thành STANDING.
* WALKING\_UPSTAIRS là lớp có tỉ lệ nhầm lẫn cao nhất, với hơn 400 mẫu bị dự đoán sai, chủ yếu sang WALKING và WALKING\_DOWNSTAIRS.
* Về mặt chỉ số:
* F1-score các lớp dao động từ 0.78 đến 1.00, với điểm trung bình Macro F1 đạt 0.8360 và Weighted F1 đạt 0.8399, phản ánh mô hình vẫn giữ được độ chính xác tương đối ổn định dù không cao bằng các mô hình trước đó.

=> Tổng thể, Decision Tree là mô hình trực quan, dễ giải thích nhưng hiệu suất phân loại thấp hơn đáng kể so với Logistic Regression và SVM trong trường hợp này. Những sai số đáng kể giữa các lớp tĩnh và động có thể được giảm thiểu nếu kết hợp với các kỹ thuật như pruning, boosting hoặc sử dụng các mô hình phức tạp hơn như Random Forest.

Sau khi đánh giá hiệu quả của mô hình Decision Tree, ta tiếp tục triển khai mô hình Random Forest – một phương pháp tổ hợp (ensemble) giúp khắc phục hiện tượng overfitting thường gặp ở cây quyết định đơn lẻ. Mô hình Random Forest trên bộ dữ liệu nhận diện hành vi người dùng từ cảm biến điện thoại thông minh, với mục tiêu là tìm ra số mtry tối ưu là số lượng biến ngẫu nhiên được chọn tại mỗi nút phân tách trong cây. Việc chọn mtry trong khoảng 16 đến 32 là phù hợp vì xét trên tổng thể số đặc trưng đầu vào 561 để đảm bảo sự đa dạn và ngẫu nhiên trong rừng cây. Cross-validation 5-fold đảm bảo rằng kết quả không bị lệ thuộc vào một phân phối dữ liệu cụ thể, tăng tính tổng quát hóa cho mô hình.

library(randomForest)  
grid\_rf <- expand.grid(mtry = c(16, 20, 24, 28, 32))  
results\_rf <- train\_and\_evaluate\_model(  
 model\_name = "rf",  
 X\_train = X\_train,  
 y\_train = y\_train,  
 X\_test = X\_test,  
 y\_test = y\_test,  
 class\_labels = labels,  
 tuneGrid = grid\_rf,  
 k\_folds = 5  
)

## --------------------------  
## | Best Parameters |  
## --------------------------  
## mtry = 16  
## --------------------------  
## | Best CV Accuracy |  
## --------------------------  
## [1] 0.983  
## --------------------------  
## | Test Accuracy |  
## --------------------------  
## [1] 0.941  
## --------------------------  
## | Precision / Recall / F1 |  
## --------------------------  
## Precision Recall F1  
## Class: LAYING 1.0000 1.0000 1.0000  
## Class: SITTING 0.9515 0.9185 0.9347  
## Class: STANDING 0.9271 0.9568 0.9417  
## Class: WALKING 0.9162 0.9698 0.9422  
## Class: WALKING\_DOWNSTAIRS 0.9592 0.8405 0.8959  
## Class: WALKING\_UPSTAIRS 0.8947 0.9384 0.9161  
## ------------------------------  
## | Macro Avg (All Classes) |  
## ------------------------------  
## Precision: 0.9415  
## Recall : 0.9373  
## F1-score : 0.9384  
## ------------------------------  
## | Weighted Avg (By Support) |  
## ------------------------------  
## Precision: 0.9420  
## Recall : 0.9410  
## F1-score : 0.9406



confusion matrix random forest

Mô hình Random Forest đạt độ chính xác kiểm tra là 94.1%, cùng với Macro F1-score = 0.9384 và Weighted F1-score = 0.9406, cho thấy đây là một mô hình mạnh, có khả năng phân loại tổng thể tốt và tương đối đồng đều giữa các lớp.

Dựa vào ma trận nhầm lẫn, ta thấy hầu hết các hoạt động như LAYING, SITTING, STANDING, và WALKING đều được mô hình nhận diện khá tốt, với số lượng mẫu dự đoán đúng lớn nằm trên đường chéo chính. Tuy nhiên, mô hình vẫn mắc một số lỗi phân loại:

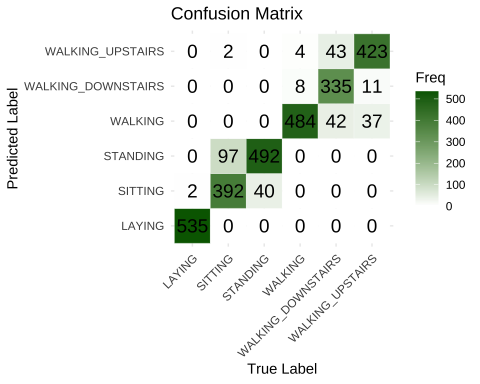
* SITTING bị nhầm sang STANDING 23 mẫu, và chiều ngược lại STANDING nhầm 40 mẫu sang SITTING. Đây là điều dễ hiểu vì hai hoạt động đều là trạng thái tĩnh, có đặc điểm tín hiệu gần giống nhau.
* WALKING\_UPSTAIRS có mức độ nhầm lẫn cao nhất, với 47 mẫu bị nhầm sang WALKING\_DOWNSTAIRS và 24 mẫu sang WALKING. Điều này phản ánh rằng các hành vi di chuyển tương tự như lên cầu thang, xuống cầu thang và đi bộ có tín hiệu khá giống nhau nên mô hình dễ nhầm lẫn.
* WALKING\_DOWNSTAIRS cũng bị ảnh hưởng, với 10 mẫu bị nhầm sang WALKING, cho thấy cần cải thiện khả năng phân biệt giữa các hành vi động học này.

=> Về tổng thể, mô hình Random Forest có khả năng nhận diện tốt các hành vi rõ ràng như LAYING (F1 = 1.000), WALKING (F1 = 0.9422), và STANDING (F1 = 0.9417). Tuy nhiên, vẫn còn không ít nhầm lẫn ở các lớp có ranh giới mờ hoặc tín hiệu gần nhau như WALKING\_UPSTAIRS và WALKING\_DOWNSTAIRS.

Sau đó chúng em tiếp tục triển khai K-Nearest Neighbors (KNN), một lưới các giá trị siêu tham số k đã được thiết lập từ 3 đến 30, với bước nhảy 2. Việc giới hạn lựa chọn k là các số lẻ nhằm tránh tình trạng hòa phiếu trong quá trình bỏ phiếu đa số của KNN. Sử dụng hàm train\_and\_evaluate\_model() để huấn luyện và đánh giá mô hình KNN ứng với từng giá trị k trong lưới thông qua quy trình cross-validation 5-fold. Sau quá trình huấn luyện, giá trị k tối ưu được trích xuất từ mô hình đã hiệu chỉnh thông qua thuộc tính bestTune, cho phép xác định cấu hình KNN có hiệu suất dự đoán tốt nhất trên tập kiểm tra.

grid\_knn <- expand.grid(k = seq(3, 30, by = 2))   
  
results\_knn <- train\_and\_evaluate\_model(  
 model\_name = "knn",  
 X\_train = X\_train,   
 y\_train = y\_train,  
 X\_test = X\_test,   
 y\_test = y\_test,  
 class\_labels = labels,  
 tuneGrid = grid\_knn,  
 k\_folds = 5  
)

## --------------------------  
## | Best Parameters |  
## --------------------------  
## k = 5  
## --------------------------  
## | Best CV Accuracy |  
## --------------------------  
## [1] 0.9675  
## --------------------------  
## | Test Accuracy |  
## --------------------------  
## [1] 0.903  
## --------------------------  
## | Precision / Recall / F1 |  
## --------------------------  
## Precision Recall F1  
## Class: LAYING 1.0000 0.9963 0.9981  
## Class: SITTING 0.9032 0.7984 0.8476  
## Class: STANDING 0.8353 0.9248 0.8778  
## Class: WALKING 0.8597 0.9758 0.9141  
## Class: WALKING\_DOWNSTAIRS 0.9463 0.7976 0.8656  
## Class: WALKING\_UPSTAIRS 0.8962 0.8981 0.8971  
## ------------------------------  
## | Macro Avg (All Classes) |  
## ------------------------------  
## Precision: 0.9068  
## Recall : 0.8985  
## F1-score : 0.9001  
## ------------------------------  
## | Weighted Avg (By Support) |  
## ------------------------------  
## Precision: 0.9063  
## Recall : 0.9030  
## F1-score : 0.9021



confusion matrix KNN

Mô hình KNN mặc dù đơn giản, nhưng cho thấy hiệu quả phân loại khá cao trên tập dữ liệu hoạt động con người. Với độ chính xác kiểm tra đạt 90.3% và Macro F1-score 90.01%, KNN đứng trong nhóm mô hình có hiệu suất tốt nhất, chỉ xếp sau các mô hình tuyến tính như SVM và Logistic Regression.

Tuy nhiên, một số điểm cần lưu ý:

* Mô hình nhạy cảm với nhiễu và có thể gặp khó khăn với các lớp có ranh giới mờ như SITTING và STANDING điều thể hiện rõ qua tỷ lệ nhầm lẫn cao giữa hai lớp này.
* Các hoạt động động như WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS vẫn còn nhầm lẫn lẫn nhau, phản ánh tính chất tín hiệu gần giống trong không gian đặc trưng.
* Mô hình yêu cầu dữ liệu được chuẩn hóa tốt để đảm bảo khoảng cách Euclidean phản ánh đúng sự tương đồng giữa các điểm.

=> Tóm lại, KNN là một lựa chọn hiệu quả cho bài toán phân loại này, đặc biệt khi không cần mô hình phức tạp. Tuy nhiên, hiệu năng của KNN phụ thuộc nhiều vào việc chọn số k phù hợp và đặc điểm phân bố dữ liệu trong không gian.

# 5 KẾT QUẢ THỰC NGHIỆM

## 5.1 Tổng hợp kết quả mô hình

Trong nghiên cứu này, chúng em đã thực hiện một quy trình phân tích và xây dựng mô hình học máy nhằm phân loại hành vi vận động của con người dựa trên dữ liệu cảm biến từ bộ dữ liệu Human Activity Recognition with Smartphones. Bộ dữ liệu chứa các đặc trưng trích xuất từ gia tốc kế và con quay hồi chuyển gắn trên điện thoại thông minh, với 561 đặc trưng đầu vào đại diện cho các thông tin trong miền thời gian và tần số.

Trước khi đưa vào mô hình, dữ liệu được xử lý kỹ lưỡng bao gồm: kiểm tra dữ liệu thiếu, loại bỏ dòng trùng lặp, xử lý các giá trị ngoại lai thông qua kỹ thuật capping theo ngưỡng phần trăm vị (percentile). Sau đó, nhóm đã áp dụng các kỹ thuật giảm chiều dữ liệu như UMAP để trực quan hóa dữ liệu. Kết quả trực quan cho thấy các nhóm hành vi có xu hướng phân tách rõ ràng, đặc biệt là các hoạt động như LAYING, WALKING\_UPSTAIRS và WALKING\_DOWNSTAIRS.Phần xây dựng mô hình, nhóm đã thử nghiệm và đánh giá hiệu quả của các thuật toán học máy khác nhau

Tiếp theo chúng em tiến hành tổng hợp và so sánh hiệu suất của các mô hình dựa trên các chỉ số đánh giá quan trọng như Accuracy, Macro F1 và Weighted F1.

model\_summary <- data.frame(  
 Model = c("Logistic Regression", "SVM (Linear)", "Decision Tree", "Random Forest", "KNN"),  
 Accuracy = c(  
 results\_log$test\_accuracy,  
 results\_svc$test\_accuracy,  
 results\_dt$test\_accuracy,  
 results\_rf$test\_accuracy,  
 results\_knn$test\_accuracy  
 ),  
 Macro\_F1 = c(  
 results\_log$macro\_metrics["F1"],  
 results\_svc$macro\_metrics["F1"],  
 results\_dt$macro\_metrics["F1"],  
 results\_rf$macro\_metrics["F1"],  
 results\_knn$macro\_metrics["F1"]  
 ),  
 Weighted\_F1 = c(  
 results\_log$weighted\_metrics["F1"],  
 results\_svc$weighted\_metrics["F1"],  
 results\_dt$weighted\_metrics["F1"],  
 results\_rf$weighted\_metrics["F1"],  
 results\_knn$weighted\_metrics["F1"]  
 )  
)  
  
print(model\_summary)

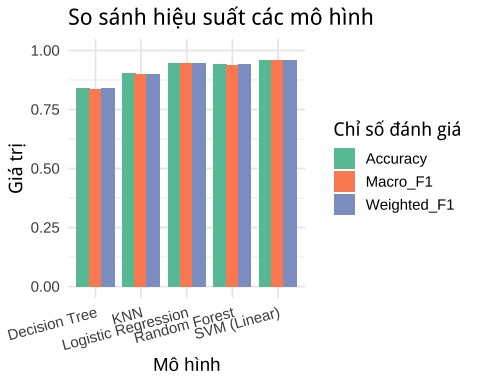
## Model Accuracy Macro\_F1 Weighted\_F1  
## 1 Logistic Regression 0.9487615 0.9482859 0.9486186  
## 2 SVM (Linear) 0.9606379 0.9603040 0.9605108  
## 3 Decision Tree 0.8401765 0.8359892 0.8398550  
## 4 Random Forest 0.9409569 0.9384417 0.9406305  
## 5 KNN 0.9029522 0.9000549 0.9021489

SVM tuyến tính là mô hình tốt nhất với độ chính xác và F1-score cao nhất trên cả hai tiêu chí macro và weighted. Điều này chứng tỏ SVM không chỉ hoạt động tốt trên toàn tập dữ liệu mà còn duy trì độ ổn định giữa các lớp phân loại.Logistic Regression cũng thể hiện hiệu quả rất tốt, cho thấy các đặc trưng đầu vào phù hợp với mô hình tuyến tính. Random Forest đạt hiệu suất cao, đặc biệt thích hợp với các lớp hành vi chuyển động rõ ràng như WALKING hay LAYING. Tuy KNN bị ảnh hưởng bởi độ lớn của chiều dữ liệu nhưng kết quả vẫn đạt ở mức ổn định khi k = 5. Decision Tree tuy không tối ưu về độ chính xác nhưng lại đóng vai trò baseline quan trọng và giúp giải thích được logic phân loại.

## 5.2 Phân tích và so sánh mô hình

Hình dưới đây minh họa trực quan mức độ hiệu quả của các mô hình dựa trên ba chỉ số đánh giá chính. Có thể thấy rõ rằng SVM đạt điểm cao nhất ở cả ba thước đo, trong khi các mô hình như Logistic Regression và Random Forest bám sát theo sau. Decision Tree là mô hình có hiệu suất thấp nhất trong nhóm, đặc biệt trên chỉ số Macro F1.

library(ggplot2)  
library(tidyr)  
library(dplyr)  
  
model\_summary\_long <- model\_summary %>%  
 pivot\_longer(cols = c("Accuracy", "Macro\_F1", "Weighted\_F1"),  
 names\_to = "Metric", values\_to = "Value")  
  
ggplot(model\_summary\_long, aes(x = Model, y = Value, fill = Metric)) +  
 geom\_bar(stat = "identity", position = position\_dodge(width = 0.8)) +  
 scale\_fill\_brewer(palette = "Set2") +  
 labs(  
 title = "So sánh hiệu suất các mô hình",  
 x = "Mô hình",  
 y = "Giá trị",  
 fill = "Chỉ số đánh giá"  
 ) +  
 ylim(0, 1) +  
 theme\_minimal(base\_size = 14) +  
 theme(axis.text.x = element\_text(angle = 15, hjust = 1))



SVM tuyến tính là mô hình có hiệu suất cao nhất trên cả ba chỉ số, thể hiện khả năng phân loại mạnh mẽ và ổn định giữa các lớp.

Logistic Regression và Random Forest cũng cho kết quả rất tốt, chỉ xếp sau SVM một cách sát sao. Logistic hoạt động đặc biệt hiệu quả trong không gian tuyến tính, còn Random Forest thể hiện ưu thế ở các lớp có tính phi tuyến cao.

KNN, mặc dù đơn giản, vẫn đạt hiệu suất đáng kể với Accuracy > 90%, cho thấy phù hợp với dữ liệu đã được chuẩn hóa và có ranh giới rõ ràng.

Decision Tree là mô hình có độ chính xác thấp nhất, chủ yếu do nhầm lẫn giữa các lớp tĩnh và giới hạn về khả năng khái quát khi không được kết hợp với các kỹ thuật tổ hợp như Random Forest.

=> Tổng thể, mô hình SVM tuyến tính được đánh giá là lựa chọn tối ưu nhất cho bài toán phân loại này, trong khi các mô hình khác như Logistic Regression và Random Forest vẫn có thể được cân nhắc tùy theo yêu cầu cụ thể về tốc độ, độ phức tạp và khả năng giải thích.

# 6 KẾT LUẬN

Qua quá trình thực hiện đã cho thấy tiềm năng lớn của các mô hình học máy trong việc nhận diện hành vi từ dữ liệu cảm biến, đồng thời khẳng định vai trò quan trọng của tiền xử lý, giảm chiều, lựa chọn mô hình và đánh giá hiệu năng chi tiết theo từng lớp.

Trong tương lai, nhóm mong muốn mở rộng nghiên cứu theo các hướng như: Tích hợp thêm các thuật toán học sâu (Deep Learning) như RNN/LSTM để tận dụng tính liên tục theo thời gian trong tín hiệu. Nghiên cứu các kỹ thuật tăng cường dữ liệu và học chuyển giao (Transfer Learning). Ứng dụng vào các hệ thống nhận dạng thời gian thực cho thiết bị đeo thông minh hoặc giám sát sức khỏe từ xa.

# Tài liệu tham khảo

[1] “FrontMatter,” in *Applied Logistic Regression*, John Wiley & Sons, Ltd, 2013, pp. i–xvi. doi: [10.1002/9781118548387.fmatter](https://doi.org/10.1002/9781118548387.fmatter).

[2] “Nearest neighbor pattern classification IEEE Journals & Magazine IEEE Xplore.” Accessed: Apr. 15, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/1053964>

[3] “1.4. Support Vector Machines,” *scikit-learn*. Accessed: Apr. 15, 2025. [Online]. Available: <https://scikit-learn/stable/modules/svm.html>

[4] “A Complete Guide to Decision Trees Paperspace Blog,” *Paperspace by DigitalOcean Blog*. Feb. 2020. Accessed: Apr. 15, 2025. [Online]. Available: <https://blog.paperspace.com/decision-trees/>

[5] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324).

[6] L. McInnes, J. Healy, and J. Melville, “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.” arXiv, Sep. 2020. doi: [10.48550/arXiv.1802.03426](https://doi.org/10.48550/arXiv.1802.03426).

[7] “Human Activity Recognition with Smartphones.” Accessed: Apr. 15, 2025. [Online]. Available: <https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones>