

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT
KHOA HỆ THỐNG THÔNG TIN



ĐỒ ÁN CUỐI KỲ
“Áp dụng và so sánh hiệu quả của các thuật toán
KNN, Linear Regression, Decision Tree &
Random Forest vào bài toán dự đoán
doanh số bán hàng”

Môn học: Phân tích dữ liệu với R/Python

Mã học phần: 222IS2901

Giảng viên: Thầy Nguyễn Phát Đạt

Trợ giảng: Anh Trần Lê Tấn Thịnh

Nhóm sinh viên thực hiện: Nhóm 07

Thành phố Hồ Chí Minh, ngày 09 tháng 4 năm 2023

TÓM TẮT

Đề tài “Áp dụng và so sánh hiệu quả của KNN, Linear Regression, Decision Tree & Random Forest vào bài toán dự đoán doanh số bán hàng” tập trung nghiên cứu về việc áp dụng các mô hình hồi quy học máy giám sát để dự báo doanh số bán hàng trong lĩnh vực bán lẻ. Nghiên cứu được thực hiện với ba mục tiêu cơ bản là: (1) Nắm được quy trình, cách thức và thực hành khai phá dữ liệu và tiền xử lý dữ liệu; (2) Tìm ra thuộc tính quan trọng ảnh hưởng đến doanh số bán hàng; (3) So sánh các mô hình và tìm ra mô hình hoạt động hiệu quả nhất. Trong bài nghiên cứu, nhóm tác giả đã sử dụng dữ liệu của Rossmann - chuỗi cửa hàng thuốc lớn thứ hai của Đức với 3.000 cửa hàng tại Châu Âu để xây dựng mô hình dự đoán doanh số bán lẻ dựa trên 4 thuật toán KNN, Linear Regression, Decision Tree và Random Forest. Nhóm tác giả sử dụng ngôn ngữ Python và công cụ hỗ trợ Google Colab để tiến hành phân tích dữ liệu và xây dựng mô hình. Kết quả nghiên cứu cho thấy mô hình Random Forest là mô hình hoạt động tốt nhất (R- square đạt 0.971, RMSE đạt 698.3, RMPSE đạt 0.10782), tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression. Bên cạnh đó, nghiên cứu cũng chỉ ra 5 features quan trọng nhất tác động đến hiệu quả mô hình là: Competition Distance, Store, Promo, Day Of Week, Competition Open Since Month.

LỜI CAM ĐOAN

Tôi tên là: Vũ Thị Phương Anh - trưởng nhóm thực hiện đồ án với đề tài “Áp dụng và so sánh hiệu quả của KNN, Linear Regression, Decision Tree & Random Forest vào bài toán dự đoán doanh số bán hàng” trong môn học Phân tích dữ liệu bằng R/Python do thầy – ThS. Nguyễn Phát Đạt hướng dẫn, chuyên ngành Thương mại điện tử, Khoa Hệ thống thông tin kinh doanh, Trường Đại học Kinh tế - Luật, Đại học Quốc gia Thành phố Hồ Chí Minh.

Tôi xin cam đoan đây là đề tài nghiên cứu do chính nhóm tôi thực hiện dưới sự hướng dẫn của Th.S Nguyễn Phát Đạt và trợ giảng là anh Trần Lê Tấn Thịnh. Các số liệu, kết quả nêu trên trong bài nghiên cứu là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Nhóm sinh viên thực hiện

Vũ Thị Phương Anh

Đinh Thị Thúy An

Nguyễn Thị Thanh Bình

Lê Thị Thùy Linh

Mai Lê Ngọc Trâm

MỤC LỤC

TÓM TẮT	1
LỜI CAM ĐOAN	2
MỤC LỤC	3
DANH MỤC HÌNH ẢNH	6
DANH MỤC THUẬT NGỮ VIẾT TẮT	9
CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI	10
1.1. Lý do lựa chọn đề tài.....	10
1.2. Mục tiêu của đề tài	10
1.3. Đối tượng và phạm vi nghiên cứu.....	11
1.3.1. Đối tượng nghiên cứu	11
1.3.2. Phạm vi nghiên cứu	11
1.4. Phương pháp nghiên cứu.....	11
1.4.1. Phương pháp xử lý dữ liệu.....	11
1.4.2. Phương pháp học máy	12
1.5. Kết cấu bài nghiên cứu.....	12
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÁC NGHIÊN CỨU LIÊN QUAN	13
2.1. Giới thiệu Python	13
2.1.1. Python và phân tích dữ liệu	13
2.1.2. Phân tích dữ liệu với Python.....	14
2.2. Dự đoán doanh số.....	15
2.2.1. Khái niệm.....	15
2.2.2. Ý nghĩa và tầm quan trọng [5]	16
2.2.3. Những yếu tố ảnh hưởng đến dự đoán doanh số [6].....	16
2.2.4. Cách đưa ra dự đoán hiệu quả và chính xác [7].....	17
2.2.5. Những thuật toán dùng để dự đoán doanh số	17
2.3. Lý thuyết ứng dụng trong đề tài	18
2.3.1. K-Neighbors Nearest (KNN)	18
2.3.1.1. Khái niệm	18

2.3.1.2. Thuật toán.....	18
2.3.1.3. Phương pháp cải thiện	22
2.3.2. Linear Regression	24
2.3.2.1. Khái niệm	24
2.3.2.2. Thuật toán.....	24
2.3.2.3. Phương pháp cải thiện	29
2.3.3. Decision Tree	31
2.3.3.1. Khái niệm	31
2.3.3.2. Thuật toán.....	33
2.3.3.3. Phương pháp cải thiện	38
2.3.4. Random Forest.....	41
2.3.4.1. Khái niệm	41
2.3.4.2. Thuật toán.....	42
2.3.4.3. Phương pháp cải thiện	51
2.3.5. Các chỉ số đánh giá	53
2.3.5.1. Mean Absolute Error - MAE	53
2.3.5.2. Mean Squared Error - MSE.....	53
2.3.4.3. Root Mean Square Error - RMSE	54
2.3.5.4. Root Mean Square Percentage Error - RMPSE	55
2.3.5.5. R Squared - R2	55
2.4. Các nghiên cứu liên quan.....	55
2.4.1. Sales Forecasting for Retail Chains (Ankur Jain, Manghat Nitish Menon, Saurabh Chandra, 2015)	55
2.4.2. Walmart Sales Prediction Based on Decision Tree, Random Forest, and K Neighbors Regressor (Bo Yao, 2022)	57
CHƯƠNG 3. MÔ HÌNH DỰ ĐOÁN DOANH THU	59
3.1. Quy trình phân tích và xây dựng mô hình.....	59
3.1.1. Xác định mục tiêu phân tích	59
3.1.2. Thu thập dữ liệu	59

3.1.3. Phân tích khám phá dữ liệu.....	60
3.1.4. Tiền xử lý dữ liệu.....	61
3.1.5. Xây dựng mô hình phân tích.....	61
3.1.6. Đánh giá mô hình và kết luận	61
3.2. Phân tích khám phá dữ liệu	62
3.2.1 Trực quan hóa dữ liệu	62
3.2.2 Xử lý giá trị null.....	66
3.2.3 Xử lý giá trị ngoại lai	68
3.3 Tiền xử lý dữ liệu	70
3.3.1 Chuyển đổi kiểu dữ liệu phân loại sang số nhị phân	70
3.3.2 Xây dựng model Regression.....	72
3.4. Mô hình đề xuất	74
CHƯƠNG 4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT	77
4.1. Kết quả nghiên cứu	77
4.1.1. KNN.....	77
4.1.2. Linear Regression	80
4.1.3. Decision Tree	83
4.1.4. Random Forest.....	86
4.2. Thảo luận.....	92
4.2.1 Báo cáo kết quả.....	92
4.2.2 Features quan trọng.....	95
4.3. Đề xuất mô hình áp dụng	98
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	99
5.1 Kết luận	99
5.2 Hạn chế.....	100
5.3 Hướng phát triển trong tương lai	100
TÀI LIỆU THAM KHẢO.....	102
MÃ NGUỒN.....	104

DANH MỤC HÌNH ẢNH

Hình 1: Mô tả kết quả phân cụm của KNN cho bài toán phân loại ba màu sắc “Đỏ - Xanh - Vàng”	18
Hình 2: Norm 1 và norm 2 trong không gian hai chiều.....	21
Hình 3: Hình mô tả công dụng của Linear Regression	24
Hình 4: Đồ thị thuật toán Linear Regression.....	25
Hình 5: Ba trường hợp của learning_rate	28
Hình 6: Hình ảnh minh họa cho việc nhạy cảm với dữ liệu nhiễu	30
Hình 7: Biểu diễn kết quả dự đoán của Decision Tree Regression	32
Hình 8: Kết quả phân chia của Decision Tree Regression	32
Hình 9: Cây quyết định không phải cây nhị phân	33
Hình 10: Cây nhị phân.....	33
Hình 11: Mô tả cây quyết định	33
Hình 12: Minh họa cây quyết định dự đoán số điểm của người chơi golf (Decision Tree)	35
Hình 13: Mô tả kết quả của thuật toán cây quyết định hồi quy	36
Hình 14: So sánh model thường, bagging và boosting. Nguồn: Viblo.....	40
Hình 15: Các phương pháp giảm chiều dữ liệu. Nguồn: Dataaspirant.....	41
Hình 16: Biểu diễn mô hình Random Forest cơ bản	42
Hình 17: Ý tưởng thuật toán Random Forest	43
Hình 18: So sánh model thường, bagging và boosting.....	44
Hình 19: Minh họa bagging.....	45
Hình 20: Minh họa cách lấy mẫu tái lập.....	46
Hình 21: Minh họa cách xây dựng Random Forest.....	48
Hình 22: Quy chế voting trong Random Forest	48
Hình 23: Biểu đồ thể hiện doanh thu theo năm	62
Hình 24: Biểu đồ thể hiện doanh thu theo ngày trong tuần.....	63
Hình 25: Biểu đồ thể hiện doanh thu theo khuyến mãi	63

Hình 26: Biểu đồ thể hiện doanh thu theo ngày lễ	64
Hình 27: Biểu đồ thể hiện doanh thu theo ngày nghỉ ở trường	65
Hình 28: Biểu đồ thể hiện doanh thu theo loại cửa hàng	65
Hình 29: Biểu đồ thể hiện doanh thu theo Assortment.....	66
Hình 30: Biểu diễn giá trị null của tập dữ liệu	66
Hình 31: Biểu diễn giá trị null sau khi xử lý	67
Hình 32: Biểu đồ thể hiện doanh thu theo CompetitionDistance	67
Hình 33: Biểu đồ histogram của biến sales	68
Hình 34: Biểu đồ phân phối của những giá trị ngoại lai	69
Hình 35: Biểu diễn kiểu dữ liệu của các cột.....	70
Hình 36: Bảng dữ liệu sau khi dùng label encoder và dummy encoding.....	71
Hình 37: Bảng correlation thể hiện chỉ số tương quan giữa các biến	72
Hình 38: Heatmap thể hiện mối tương quan giữa các biến	72
Hình 39: Mô hình nghiên cứu đề xuất.....	74
Hình 40: So sánh chỉ số R2 của mô hình KNN khi thay đổi k.....	78
Hình 41: Kết quả R2 của mô hình khi thay đổi chiều dữ liệu	79
Hình 42: Xây dựng mô hình Linear Regression cho tập dữ liệu.....	80
Hình 43: Kết quả giảm dần của hàm chi phí khi chạy thuật toán Gradient Descent.....	82
Hình 44: Đồ thị thể hiện mức giảm dần của hàm chi phí.....	82
Hình 45: Mô hình Decision Tree chưa điều chỉnh	83
Hình 46: Kết quả chạy GridSearchCV trên Decision Tree.....	84
Hình 47: Kiểm tra max_features và hiệu quả của mô hình Decision Tree.....	85
Hình 48: Mô hình Decision Tree đã điều chỉnh	85
Hình 49: 10 thuộc tính quan trọng nhất trong việc hình thành Decision Tree	86
Hình 50: Kiểm tra n_estimators trên mô hình Random Forest	88
Hình 51: Kiểm tra max_features trên mô hình Random Forest	90
Hình 52: Các thuộc tính quan trọng nhất trong hoạt động mô hình Random Forest	91
Hình 53: Dataframe kết quả các mô hình.....	92
Hình 54: Biểu đồ thể hiện chỉ số R2.....	92

Hình 55: Biểu đồ thể hiện chỉ số MAE, MSE và RMSE	93
Hình 56: Biểu đồ thể hiện chỉ số RMPSE	94
Hình 57: Mức độ tương quan của features đối với biến Sales	95
Hình 58: Mức độ quan trọng của features trong mô hình Decision Tree	96
Hình 59: Mức độ quan trọng của features trong mô hình Random Forest.....	97

DANH MỤC THUẬT NGỮ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ viết đầy đủ	Ý nghĩa tiếng Việt (nếu có)
1	KNN	K-Nearest Neighbors	K- Hệ số láng giềng gần
2	LDA	Linear Discriminant Analysis	Phân tích biệt thức tuyến tính
3	MAE	Mean Absolute Error	Sai số trung bình tuyệt đối
4	MSE	Mean Squared Error	Sai số toàn phương trung bình
5	PCA	Principal Components Analysis	Phép phân tích thành phần chính
6	R ²	R Squared	R bình phương
7	RMPSE	Root Mean Square Percentage Error	
8	RMSE	Root Mean Square Error	Căn bậc hai của sai số toàn phương trung bình
9	SVD	Singular Value Decomposition	Phân tích suy biến
10	VIF	Variance Inflation Factors	Hệ số phóng đại phương sai

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI

Tóm tắt chương 1: Giới thiệu tổng quan về bài nghiên cứu, bối cảnh và tính cấp thiết của đề tài, đặt ra vấn đề, mục tiêu, đối tượng và phạm vi nghiên cứu, phương pháp cũng như kết cấu của bài nghiên cứu. Trong chương này, nhóm tác giả đã xác định được ba mục tiêu chính của bài nghiên cứu là: (1) Nắm được quy trình, cách thức và thực hành khai phá dữ liệu và tiền xử lý dữ liệu; (2) Tìm ra được thuộc tính quan trọng ảnh hưởng đến doanh số bán hàng; (3) So sánh các mô hình và tìm ra mô hình hoạt động hiệu quả nhất.

1.1. Lý do lựa chọn đề tài

Dự đoán doanh số đóng vai trò quan trọng trong kinh doanh. Dựa vào các dự đoán doanh số trong bán hàng, doanh nghiệp có thể xây dựng các chiến lược phát triển. Nhờ vào đó, các doanh nghiệp cũng có thể giảm chi phí hoạt động và cải thiện doanh số đồng thời tăng sự hài lòng của khách hàng [1].

Nhưng cho đến thời điểm hiện tại, nhiều nhà quản lý vẫn tự đưa ra dự báo một cách cảm tính và kinh nghiệm cá nhân. Khi chỉ có những nhà quản lý thực hiện công việc này, nếu thiếu đi họ, công ty phải đối mặt với vấn đề thiếu hụt nhân sự dự đoán và phải đào tạo nhân sự mới. Điều này gây ra sự phụ thuộc vào những nhân sự cụ thể và những sai sót, chậm trễ trong quá trình đưa ra dự đoán đến từ các cá nhân.

Từ những yêu cầu và thực trạng đó, mô hình hóa kỹ năng của các nhà quản lý bằng các mô hình học máy để dự đoán doanh thu tốt hơn trở thành phương pháp được đón nhận và nghiên cứu nhiều hơn [2]. Vì vậy đề tài này sẽ nghiên cứu về các mô hình học máy áp dụng vào bài toán dự báo doanh số trong kinh doanh, sau đó so sánh và đưa ra một số đề xuất về mô hình dự báo doanh thu hiệu quả cho doanh nghiệp.

1.2. Mục tiêu của đề tài

Đề tài “Áp dụng và so sánh hiệu quả của KNN, Linear Regression, Decision Tree & Random Forest vào bài toán dự đoán doanh số bán hàng” với mục tiêu chính là đề xuất

một cách tiếp cận để dự đoán nhu cầu và doanh số bán lẻ tại mỗi cửa hàng từ 04 mô hình học máy khác nhau dựa trên ngôn ngữ lập trình Python.

Trong đó, một số mục tiêu cần đạt được trong và sau khi thực hiện đề tài như sau:

- Nắm được quy trình, cách thức và thực hành chuyển đổi dữ liệu thành dạng thích hợp bằng các kỹ thuật tiền xử lý khác nhau để triển khai các thuật toán Học máy.
- Phân tích khai phá bộ dữ liệu và tìm ra được các thuộc quan trọng có ảnh hưởng đến doanh số bán hàng.
- So sánh các mô hình và xác định mô hình hiệu quả cho việc dự đoán doanh số bán hàng.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Nghiên cứu lý thuyết của mô hình học máy nói chung và đi sâu vào nghiên cứu bốn mô hình học máy bao gồm Linear Regression, Decision Tree, Random Forest và K-nearest Neighbor; nghiên cứu về cách thức xây dựng và điều chỉnh những thông số của mô hình để xây dựng mô hình tối ưu nhất vào việc dự đoán doanh số bán hàng; so sánh hiệu quả của các mô hình.

1.3.2. Phạm vi nghiên cứu

- Lĩnh vực: Dự đoán doanh số bán hàng bằng các mô hình học máy.
- Không gian: Không giới hạn.
- Thời gian: Những bài nghiên cứu được thực hiện từ tháng 02/2023 đến tháng 4/2023 và các bài báo liên quan có thời gian từ năm 2015 đến nay.

1.4. Phương pháp nghiên cứu

1.4.1. Phương pháp xử lý dữ liệu

- Phương pháp phân tích khám phá dữ liệu: sử dụng bar chart, scatter chart, heatmap, distribution plots để trực quan hóa những phân tích về dữ liệu.

- Phương pháp tiền xử lý dữ liệu: sử dụng label encoder, dummy encoding, chuyển đổi kiểu dữ liệu, tiến hành chia tập train - test và sử dụng StandardScaler để phục vụ việc dựng mô hình hồi quy.

1.4.2. Phương pháp học máy

- Nghiên cứu sử dụng thuật toán KNN, Decision Tree, Random Forest, Linear Regression để huấn luyện mô hình học máy nhằm dự đoán doanh thu.
- Nghiên cứu sử dụng các chỉ số đánh giá sau để đánh giá chất lượng của mô hình: R^2 , MAE, MSE, MRSE, RMPSE.

1.5. Kết cấu bài nghiên cứu

Đồ án gồm có 5 chương:

- Chương 1: Giới thiệu tổng quan đề tài
- Chương 2: Cơ sở lý thuyết và các nghiên cứu liên quan
- Chương 3: Mô hình dự đoán doanh thu
- Chương 4: Kết quả thực nghiệm và đề xuất
- Chương 5: Kết luận và hướng phát triển

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÁC NGHIÊN CỨU LIÊN QUAN

Tóm tắt chương 2: Chương 2 nhằm mục đích giới thiệu tổng quan về Python, những lợi ích của Python trong việc xử lý dữ liệu và những ứng dụng của nó. Bên cạnh đó còn giới thiệu tổng quan về Dự đoán doanh số, cung cấp những khái niệm, tầm quan trọng và những phương pháp của dự đoán doanh số. Từ đó đưa ra những phương pháp ứng dụng trong đề tài bao gồm 4 phương pháp: Linear Regression, Decision Tree, Random Forest và K-Nearest Neighbor, tiến hành đi sâu vào phân tích khái niệm, thuật toán và phương pháp cải thiện của từng thuật toán.

2.1. Giới thiệu Python

2.1.1. Python và phân tích dữ liệu

Python là ngôn ngữ lập trình bậc cao dành cho các mục đích lập trình đa dạng, lập trình hướng đối tượng có cấu trúc dữ liệu cấp cao, mạnh mẽ và hệ thống thư viện lớn [3].

Phân tích dữ liệu bao gồm việc thu thập dữ liệu từ nhiều nguồn khác nhau và sử dụng phân tích thống kê và học máy trên dữ liệu đó để trích xuất thông tin chi tiết có giá trị từ dữ liệu đó. Đây là một khái niệm phổ biến, đặc biệt là trong lĩnh vực thương mại vì nó cho phép các tổ chức đưa ra quyết định dựa trên dữ liệu dựa trên kết quả của phân tích dữ liệu.

Ngày nay, Phân tích dữ liệu và Python là hai thuật ngữ không thể tách rời vì Python hoạt động tốt trên mọi giai đoạn phân tích dữ liệu [4]. Nhờ vào các thư viện Python việc phân tích dữ liệu trở nên thuận tiện và dễ dàng hơn. Khai thác dữ liệu, xử lý dữ liệu và mô hình hóa – trực quan hóa dữ liệu là 3 ví dụ phổ biến nhất về cách Python đang được sử dụng để phân tích dữ liệu.

2.1.2. Phân tích dữ liệu với Python

Phân tích dữ liệu có thể là một quá trình phức tạp đối với người mới bắt đầu, nhưng chúng ta có thể dễ dàng hiểu các khía cạnh quan trọng của việc triển khai phân tích dữ liệu với Python bằng cách làm việc cùng với các bước sau [4]:

Bước 1: Thiết lập môi trường Python

Điều cần thiết cơ bản để làm việc trong Phân tích dữ liệu với Python là phải có một nền tảng nơi mà chúng ta có thể viết mã của mình và thực thi nó. Có nhiều nền tảng trực tuyến miễn phí có thể cung cấp môi trường lập trình cần thiết, như là nền tảng Python Anaconda, Google Colab,...

Bước 2: Tìm hiểu các khái niệm cơ bản về Python

Điều cần thiết là trước tiên chúng ta phải hiểu các khái niệm cơ bản của Python trước khi chuyển sang bất kỳ loại phân tích dữ liệu nào với Python. Chúng ta không cần phải trở thành một chuyên gia về ngôn ngữ lập trình này, chỉ cần bao gồm các chủ đề quan trọng sau đây là đủ:

- Triển khai cấu trúc dữ liệu
- Các loại dữ liệu khác nhau
- Tạo các chức năng
- Sử dụng vòng lặp
- Sử dụng câu lệnh có điều kiện

Bước 3: Hiểu hoạt động của thư viện Python

Một tính năng chính của Python là nó có rất nhiều thư viện có thể đơn giản hóa công việc của bạn ở một mức độ lớn. Nếu bạn muốn thực hiện Phân tích dữ liệu với Python, thì bạn phải tự làm quen với một số thư viện Python được sử dụng chính. Các thư viện Python thiết yếu liên quan đến khoa học dữ liệu là:

- Pandas
- Numpy

- Scikit-learning
- Matplotlib

Bước 4: Thực hành làm việc với tập dữ liệu

Chúng ta có thể tham khảo các bài phân tích dữ liệu mẫu trên nền tảng Kaggle. Sau đây là quy trình của một bài phân tích dữ liệu với Python:

1. Làm sạch dữ liệu: Nó liên quan đến việc tìm kiếm và sửa chữa bất kỳ điểm nào không chính xác hoặc không rõ ràng có trong dữ liệu được lưu trữ.
2. Tiền xử lý dữ liệu: Là quá trình sửa đổi dữ liệu thành các định dạng phù hợp hơn để thực hiện Phân tích dữ liệu với Python.
3. Thao tác dữ liệu: Là quá trình thực hiện các mô hình học máy trên dữ liệu để thu được kết quả mong muốn. Các tác vụ như phân cụm, phân loại, hồi quy, v.v.
4. Trực quan hóa dữ liệu: Kết quả thu được bởi bất kỳ quy trình nào trong số 3 quy trình trên của phân tích dữ liệu với Python được trình bày theo cách dễ hiểu hơn bằng Hình ảnh hóa dữ liệu. Nó bao gồm đồ thị thanh, biểu đồ hình tròn, bản đồ nhiệt, v.v.

2.2. Dự đoán doanh số

2.2.1. Khái niệm

Dự báo doanh số hay dự báo bán hàng chỉ sự ước tính về các chỉ số bán hàng trong tương lai. Các chỉ số bán hàng đó được đo lường bằng doanh thu hoặc số lượng sản phẩm. Và thời gian dự báo có thể trong ngắn hạn theo tuần, theo tháng, theo từng quý hoặc dài hạn hơn theo năm [5].

Ba điều kiện cần để cho ra một “Dự đoán doanh số” chuẩn xác:

- Dữ liệu bán hàng của doanh nghiệp trong quá khứ (Báo cáo bán hàng)
- So sánh dữ liệu bán hàng toàn ngành (Báo cáo thị phần, độ phủ)
- Xu hướng thị trường hiện tại (Báo cáo thị trường)

2.2.2. Ý nghĩa và tầm quan trọng [5]

Dự báo đóng vai trò rất quan trọng trong doanh nghiệp. Đặc biệt, công tác dự báo của phòng kinh doanh đóng vai trò chính trong mọi hoạt động của doanh nghiệp. Trưởng phòng kinh doanh phải dự báo doanh số trong ngắn hạn (3 tháng - 6 tháng), dự báo trung hạn (6 tháng – 1 năm), dự báo dài hạn (trên 1 năm). Dự báo là chìa khóa để thành công của doanh nghiệp, có khi là đề tồn tại trong giai đoạn khó khăn.

Trưởng phòng kinh doanh phải dự báo số lượng hàng bán ra, hoặc lượng tiền có thể thu về. Căn cứ vào đó, các phòng ban khác mới có kế hoạch cho mình. Và dự báo luôn có xác suất, không bao giờ chính xác 100%.

Dự đoán doanh số giữ vai trò then chốt trong việc lập kế hoạch và điều khiển hoạt động bán hàng.

- Dự đoán doanh số là nền tảng để thiết lập và duy trì sản xuất. Dự đoán doanh số giúp xác định sản lượng sản xuất xét theo sự sẵn có của các yếu tố về cơ sở vật chất như thiết bị, nhân lực, không gian, thời gian, nguồn vốn,...
- Dự đoán doanh số là cơ sở đưa ra các quyết định về kế hoạch mở rộng hay thay đổi sản xuất, hoặc có nên chuyển sang sản xuất sản phẩm khác hay không.
- Dự đoán doanh số bán hàng là một cam kết từ phía bộ phận kinh doanh sẽ đạt được mức chỉ tiêu nào trong 1 khoảng thời gian nhất định.

2.2.3. Những yếu tố ảnh hưởng đến dự đoán doanh số [6]

Những yếu tố cần được xem xét khi thực hiện dự đoán doanh số:

- **Đối thủ cạnh tranh:** Để đánh giá nhu cầu, đây là yếu tố chính để biết về các đối thủ cạnh tranh đã và đang tồn tại trên thị trường cũng như các chương trình tương lai, chất lượng sản phẩm, doanh số bán hàng của họ. Ý kiến của khách hàng về các sản phẩm của các đối thủ khác có liên quan đến sản phẩm do công ty sản xuất cũng phải được xem xét.
- **Thay đổi trong công nghệ:** Với sự tiến bộ của công nghệ, các sản phẩm mới đang xuất hiện trên thị trường và sở thích của người tiêu dùng cũng thay đổi theo sự tiến bộ và thay đổi của công nghệ.

- **Hành động của chính phủ:** Khi chính phủ sản xuất hoặc mua và sau đó tùy thuộc vào chính sách và quy tắc của chính phủ, doanh số bán hàng cũng bị ảnh hưởng.
- **Các yếu tố liên quan đến việc sản xuất:** Những yếu tố này liên quan đến sự thay đổi công suất của nhà máy, thay đổi giá do thay đổi chi tiêu, thay đổi trong hỗn hợp sản phẩm, v.v.

2.2.4. Cách đưa ra dự đoán hiệu quả và chính xác [7]

- Thống kê và nắm chắc con số cửa hàng trên địa bàn cụ thể. Phân loại các cửa hàng này dựa trên các mặt hàng mà công ty hiện bán vào, mặt hàng công ty đối thủ bán, hàng hóa tiềm năng có thể vào cửa hàng này, vị trí... Dựa trên con số này sẽ nắm căn bản về lực đẩy ra thị trường tiềm năng.
- Dựa vào số lượng salesman, mức tăng trưởng salesman hàng năm, effective calls để đo lường khả năng bao phủ các cửa hàng và doanh số. Dựa trên thông tin bước trên để điều chỉnh cho hiệu quả nhất độ bao phủ và dựa trên năng lực của nhân viên sales và số lượng cũng như địa hình để xác định khả năng dự đoán
- Dựa trên dữ liệu sales năm trước, quý trước, mức tăng trưởng mỗi năm và sử dụng các hỗ trợ từ Trade Marketing, Marketing.
- Dựa trên dòng hàng mới thêm vào, việc phát hành sản phẩm mới... ước tính ra con số tăng thêm.

2.2.5. Những thuật toán dùng để dự đoán doanh số

Để dự đoán doanh số, ta có thể dùng những thuật toán hồi quy trong Machine learning:

- Linear Regression
- Decision Tree
- Random Forest
- K-nearest Neighbor
- Logistic Regression

- Ordinary Least Squares Regression (OLSR)
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

2.3. Lý thuyết ứng dụng trong đề tài

2.3.1. K-Neighbors Nearest (KNN)

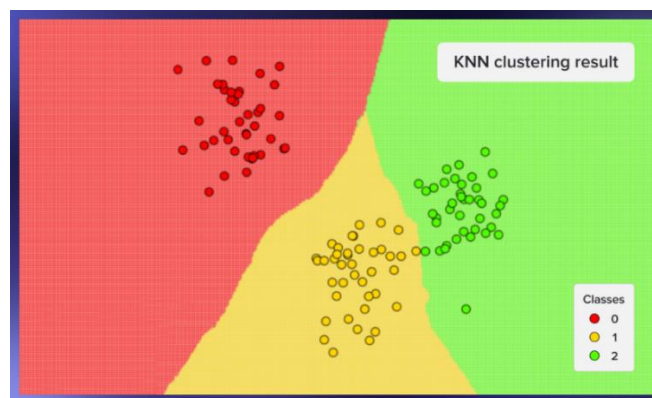
2.3.1.1. Khái niệm

K-Nearest neighbor (KNN) là một trong những thuật toán học có giám sát đơn giản nhất trong Machine Learning. Ý tưởng của KNN là tìm ra đầu ra của dữ liệu dựa trên thông tin của những dữ liệu training gần nó nhất [8].

2.3.1.2. Thuật toán

a) Ý tưởng chung của thuật toán

KNN sẽ đi tìm đầu ra của một điểm giá trị mới bằng cách tính toán khoảng cách với k điểm dữ liệu gần nó nhất. K được hiểu là “số điểm lân cận” hay “số điểm hàng xóm”. KNN sẽ không quan tâm đến việc các điểm dữ liệu trong k điểm này có phải là giá trị nhiều hay không [9].



Hình 1: Mô tả kết quả phân cụm của KNN cho bài toán phân loại ba màu sắc “Đỏ - Xanh - Vàng”

b) Khoảng cách trong không gian vector

Trong không gian một chiều, việc đo khoảng cách giữa hai điểm đã rất quen thuộc: lấy trị tuyệt đối của hiệu giữa hai giá trị đó. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính:

Distance functions

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k x_i - y_i $
Minkowski	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

Trong đó:

- k là số chiều dữ liệu
- x_i, y_i là điểm dữ liệu
- Minkowski là công thức tổng quát của khoảng cách Euclidean và Manhattan.

Trong không gian hai chiều, tức mặt phẳng, chúng ta thường dùng khoảng cách Euclid để đo khoảng cách giữa hai điểm. Việc đo khoảng cách giữa hai điểm dữ liệu nhiều chiều, tức hai vector, là rất cần thiết trong Machine Learning. Chúng ta cần đánh giá xem điểm nào là điểm gần nhất của một điểm khác; và quan tâm đến độ chính xác của việc ước lượng.

Để xác định khoảng cách giữa hai vector y và z , người ta thường áp dụng một hàm số lên vector hiệu $x = y - z$. Một hàm số được dùng để đo các vector cần đáp ứng được một số điều kiện cụ thể.

c) Định nghĩa về norm [10]

Một hàm số $f()$ ánh xạ một điểm x từ không gian n chiều sang tập số thực một chiều được gọi là norm nếu nó thỏa mãn ba điều kiện sau đây:

- $F(x) \geq 0$. Dấu bằng xảy ra $x = 0$.
- $F(\alpha x) = |\alpha|f(x)$, $\forall \alpha \in \mathbb{R}$.
- $F(x_1) + f(x_2) \geq f(x_1 + x_2)$, $\forall x_1, x_2 \in \mathbb{R}$

d) Một số norm thường dùng

Giả sử các vector $x = [x_1; x_2 \dots x_n]$, $y = [y_1; y_2 \dots y_n]$.

Nhận thấy khoảng cách Euclid chính là một norm, thường được gọi là norm 2:

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots x_n^2} \quad (1)$$

Với p là một số không nhỏ hơn 1 bất kỳ, hàm số sau đây:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \dots |x_n|^p)^{\frac{1}{p}} \quad (2)$$

Được chứng minh thỏa mãn ba điều kiện trên, và được gọi là norm p .

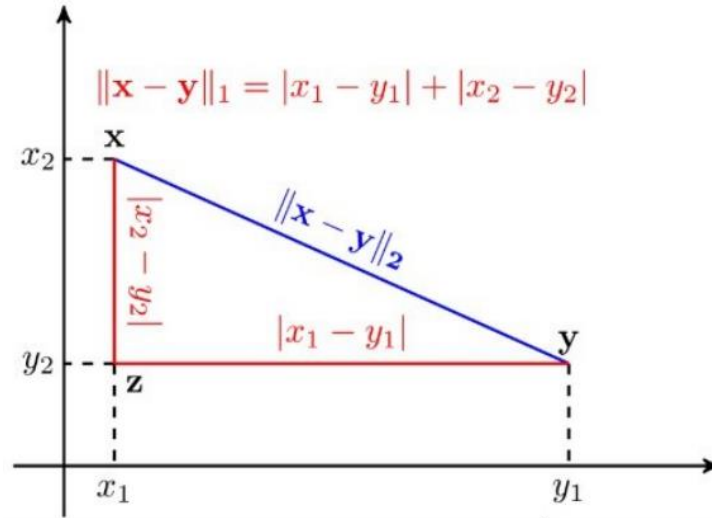
Nhận thấy rằng khi $p \rightarrow 0$ thì biểu thức bên trên trở thành số các phần tử khác 0 của x . Hàm số (2) khi $p=0$ được gọi là giả chuẩn (pseudo norm) 0. Nó không phải là norm vì nó không thỏa mãn điều kiện 2 và 3 của norm. Giả chuẩn này, thường được ký hiệu là $\|x\|_0$, khá quan trọng trong học máy vì trong nhiều bài toán, chúng ta cần có ràng buộc “sparse”, tức số lượng thành phần “active” của x là nhỏ.

Có một vài giá trị của p thường được dùng:

- Khi $p = 2$ chúng ta có norm2 như ở trên.
- Khi $p = 1$ chúng ta có:

$$\|x\|_1 = |x_1| + |x_2| + |x_3| + \dots |x_n| \quad (3)$$

Là tổng các giá trị tuyệt đối của từng phần tử của x . Norm 1 thường được dùng như xấp xỉ của norm 0 trong các bài toán có ràng buộc. Dưới đây là một ví dụ so sánh norm 1 và norm 2 trong không gian hai chiều:



Hình 2: Norm 1 và norm 2 trong không gian hai chiều

Norm 2 (màu xanh) chính là đường chim bay nối giữa vector x và vector y .

Khoảng cách norm 1 giữa hai điểm này (màu đỏ) có thể diễn giải như là đường đi từ x đến y trong một thành phố mà thành phố được tạo hình bàn cờ, chúng ta chỉ có thể đi theo dọc bàn cờ chứ không thể đi theo đường thẳng.

Khi $p > \infty$, ta có norm p chính là trị tuyệt đối của phần tử lớn nhất của vector đó:

$$\|x\|_{\infty} = \max_{i=1,2,\dots,n} |x_i|$$

e) Các chỉ số, siêu tham số quan trọng trong KNN

Câu lệnh mẫu chuẩn trong thư viện Scikit Learn:

```
KNeighborsRegressor(n_neighbors=5, *, weights='uniform', algorithm='auto',
leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None).
```

Trong đó:

n_neighbors: int, default=5

Số láng giềng gần

metric: str or callable, default='minkowski'

Số liệu để sử dụng để tính toán khoảng cách. Mặc định là “minkowski”, dẫn đến khoảng cách Euclidean tiêu chuẩn khi $p = 2$. Có thể đặt là ‘manhattan’ hoặc ‘euclid’ và khi này chúng ta không cần chọn p

p: int, default=2

Tham số công suất cho số liệu Minkowski. Khi $p = 1$, điều này tương đương với việc sử dụng manhattan và euclidean cho $p = 2$. Đối với p tùy ý, minkowski được sử dụng.

weights: {'uniform', 'distance'}, callable or None, default='uniform'

Weight dùng để đánh trọng số. Các giá trị có ý nghĩa như sau:

- 'uniform' : trọng lượng thống nhất. Tất cả các điểm trong mỗi vùng lân cận đều có trọng số như nhau.
- 'distance': điểm trọng lượng bằng nghịch đảo của khoảng cách. trong trường hợp này, các hàng xóm gần điểm test data hơn sẽ có ảnh hưởng lớn hơn các hàng xóm ở xa.

n_jobs: int, default=None

Số lượng công việc chạy song song để tìm kiếm hàng xóm. *None* có nghĩa là 1. Còn -1 có nghĩa là tất cả.

algorithm: {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'

Thuật toán được sử dụng để tính toán các hàng xóm gần nhất:

- 'ball_tree' sẽ sử dụng thuật toán BallTree
- 'kd_tree' sẽ sử dụng thuật toán KDTree
- 'brute' sẽ sử dụng tìm kiếm brute-force
- 'auto' sẽ cố gắng quyết định thuật toán phù hợp nhất dựa trên các giá trị được truyền cho phương thức phù hợp

leaf_size: int, default=30

Kích thước lá được chuyển đến BallTree hoặc KDTree. Điều này có thể ảnh hưởng đến tốc độ xây dựng và truy vấn, cũng như bộ nhớ cần thiết để lưu trữ cây. Giá trị tối ưu phụ thuộc vào bản chất của vấn đề

2.3.1.3. Phương pháp cải thiện

Phương pháp cải thiện thứ nhất: giá trị k cần được tính toán kỹ càng. Giá trị k thấp sẽ dẫn đến overfitting. Giá trị k lớn hơn sẽ giảm rủi ro overfitting, nhưng nếu k quá lớn sẽ

dẫn đến việc phân lớp quá rộng và không đạt hiệu quả cao [11]. Không có quy tắc cụ thể nào cho việc lựa chọn giá trị k tốt nhất. Một số mẹo chọn k tốt hơn:

- Chọn $k =$ căn bậc 2 của n (Với n là tổng mẫu huấn luyện). Nên chọn k là số lẻ để tránh xảy ra trường hợp số lượng 2 lớp bằng nhau.
- Thực hiện thay đổi k và đánh giá từng mô hình, sau đó chọn k có kết quả tốt nhất.

Phương pháp cải thiện thứ hai: thuật toán này cần nhiều dung lượng và thời gian để thực hiện. KNN là một thuật toán “lười” (lazy algorithm) vì nó không trải qua việc huấn luyện mô hình mà thay vào đó là “ghi nhớ” tập dữ liệu huấn luyện [12]. Chính vì vậy mà KNN sẽ cần nhiều bộ nhớ hơn để lưu trữ dữ liệu huấn luyện. Bên cạnh đó, thời gian xử lý của KNN cũng khá lâu do thuật toán này phải thực hiện tính toán khoảng cách của tất cả giá trị trong tập dữ liệu mỗi lần chạy mô hình.

Phương pháp cải thiện thứ ba: KNN chịu ảnh hưởng của “lời nguyền của chiều dữ liệu”: Với những bộ dữ liệu nhiều chiều thì việc sử dụng KNN sẽ không hiệu quả. Khi số lượng tính năng (feature) tăng lên, khoảng cách giữa các điểm dữ liệu trở nên khó phân biệt hơn, tổng diện tích thuật toán cần bao phủ để tìm k giá trị tăng lên. Từ đó dễ xảy ra hiện tượng overfitting. Lúc này chúng ta có thể sử dụng các kỹ thuật về giảm chiều để hạn chế ảnh hưởng từ lời nguyền của chiều dữ liệu.

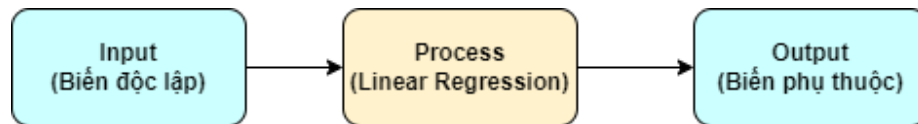
Phương pháp khắc phục: sử dụng các phương pháp giảm chiều dữ liệu bao gồm cả feature selection và dimensionality reduction tùy theo ngữ cảnh từng bài toán.

Phương pháp cải thiện thứ tư: Khó có thể áp dụng KNN lên tập dữ liệu có phân loại (categorical features), loại dữ liệu không phải là số. Nếu giá trị là “Thấp”, “Trung bình”, “Cao” thì việc mã hóa chúng thành các giá trị “1”, “2”, “3” có thể cân nhắc. Tuy nhiên với một bộ giá trị như “Đỏ”, “Vàng”, “Xanh” thì việc mã hóa thành số không còn phù hợp. Chúng ta có thể mã hóa chúng trong không gian với tọa độ (1;0;0), (0;1;0), (0;0;1) để đảm bảo khoảng cách giữa các giá trị bằng nhau. Tuy nhiên việc tính toán khoảng cách sẽ trở nên phức tạp khi số giá trị tăng lên. Câu hỏi cốt lõi ở đây sẽ là “Khoảng cách trong bộ dữ liệu của bạn được định nghĩa như thế nào?”.

2.3.2. Linear Regression

2.3.2.1. Khái niệm

Hồi quy tuyến tính (Linear Regression) là một phương pháp thống kê để hồi quy dữ liệu giữa biến phụ thuộc có giá trị liên tục với một hoặc nhiều biến độc lập có giá trị liên tục hoặc phân loại [13]. Hay nói cách khác, hồi quy tuyến tính là phương pháp dự đoán biến phụ thuộc dựa trên giá trị của biến độc lập.



Hình 3: Hình mô tả công dụng của Linear Regression

Thuật toán hồi quy tuyến tính thuộc nhóm học có giám sát và có thể sử dụng cho các trường hợp dự đoán đầu ra là một giá trị liên tục. Ví dụ như dự đoán doanh số cửa hàng, dự đoán số người truy cập website,...

2.3.2.2. Thuật toán

a) Ý tưởng chung

Thuật toán tìm một đường thẳng để đường thẳng đó gần với tất cả các điểm trên đồ thị của chúng ta nhất có thể (khoảng cách từ đường thẳng đó đến các điểm là nhỏ nhất).

b) Các loại Linear Regression

Hồi quy tuyến tính được phân ra làm 2 loại chính là: Simple linear regression và Multiple linear regression [14].

Simple linear regression

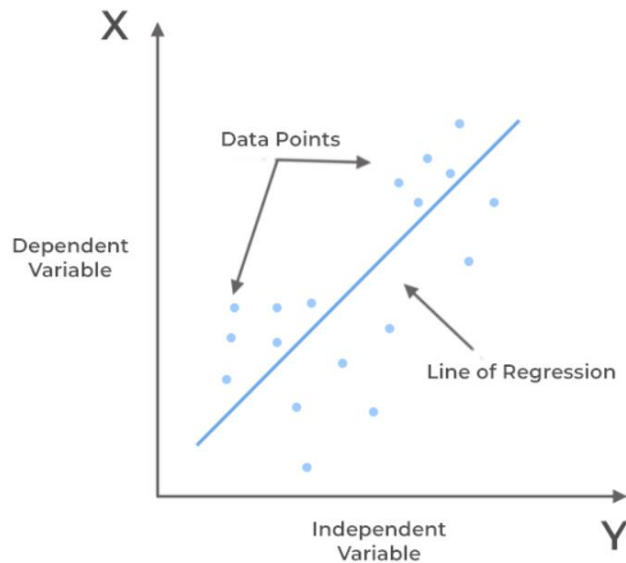
Hồi quy tuyến tính đơn biến cho thấy mối tương quan giữa một biến phụ thuộc và một biến độc lập. Khi đó, đồ thị biểu diễn mối quan hệ giữa 2 biến là một đường thẳng.

Phương trình hồi quy có dạng: $Y = \beta_0 + \beta_1 X$

Trong đó:

- β_0, β_1 là các hằng số

- Y: Biến phụ thuộc (biến mục tiêu)
- X: Biến độc lập
- β_1 : Độ dốc của đường hồi quy
- β_0 : Hệ số chặn (intercept)



Hình 4: Đồ thị thuật toán Linear Regression

Multiple linear regression

Hồi quy tuyến tính đa biến là phần mở rộng của hồi quy tuyến tính đơn biến và được sử dụng khi chúng ta muốn dự đoán giá trị của một biến dựa trên giá trị của hai hoặc nhiều biến. Nó được sử dụng để xác định mối quan hệ giữa tập biến này.

Phương trình hồi quy có dạng: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \varepsilon$

Trong đó:

- Y: Biến phụ thuộc (biến mục tiêu)
- X, X_1 , X_2 , X_n : Biến độc lập
- β_1 , β_2 , β_n : Hệ số hồi quy, hay còn được gọi là hệ số góc. Chỉ số này cho chúng ta biết về mức thay đổi của Y gây ra bởi X tương ứng

- ϵ : Sai số (phần dư). Chỉ số này càng lớn càng khiến cho khả năng dự đoán của hồi quy trở nên kém chính xác hơn hoặc sai lệch nhiều hơn so với thực tế.

$$\epsilon_i = Y_{\text{predicted}} - Y_i$$

c) Cost Function (Hàm chi phí)

Mô hình hồi quy sử dụng hàm chi phí để đi tìm các trọng số tối ưu. Hàm chi phí dùng để đo lường hiệu suất của một mô hình máy học, đây là phép tính sai số giữa giá trị dự đoán và giá trị thực tế, được biểu diễn dưới dạng một số thực duy nhất. Trong Linear regression, hàm chi phí Mean Squared Error (MSE) thường được sử dụng [14] [15].

Mean Squared Error (MSE)

MSE là sai số bình phương trung bình giữa các giá trị được dự đoán và giá trị thực tế. MSE là thước đo chất lượng của một công cụ ước tính, nó luôn không âm và các giá trị càng gần 0 càng tốt [16]. Công thức tính MSE như sau:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Trong đó:

- n : Số dòng dữ liệu của tập test
- y_i : Giá trị thực (giá trị thực của biến mục tiêu trong tập test)
- \hat{y}_i : Giá trị dự đoán (giá trị dự đoán của biến mục tiêu trong tập test)

Tính chất: Chỉ số MSE luôn không âm và nếu chỉ số này càng thấp thì sự chênh lệch giữa giá trị dự đoán và giá trị thực tế càng nhỏ và mô hình sẽ có tính chính xác cao.

Root Mean Squared Error (RMSE)

RMSE là căn bậc hai mức trung bình của các sai số bình phương. RMSE là độ lệch chuẩn của các phần dư (sai số dự đoán). Nó chỉ định mức độ phù hợp tuyệt đối của mô hình với dữ liệu, tức là mức độ gần của các điểm dữ liệu được quan sát với các giá trị dự đoán [16].

Công thức:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Tính chất: RMSE luôn không âm và giá trị 0 (hầu như không bao giờ đạt được trong thực tế) sẽ chỉ ra sự phù hợp hoàn hảo với dữ liệu. RMSE càng thấp thì độ tin cậy của mô hình càng cao.

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) đo độ lớn trung bình của các lỗi trong một tập hợp các dự đoán mà không cần xem xét hướng của chúng. MAE được định nghĩa là trung bình tổng trị tuyệt đối sai số giữa đầu ra dự đoán và kết quả thực [16].

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

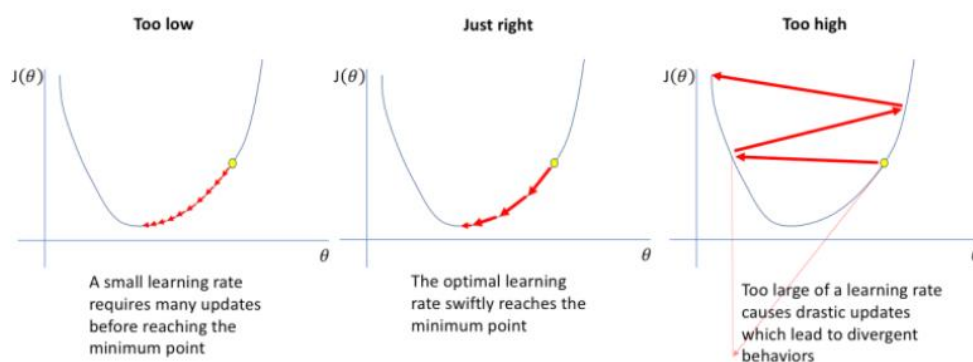
d) Thuật toán Gradient Descent

Thuật toán Gradient Descent dùng để tối ưu hóa hàm chi phí MSE để độ chính xác của mô hình cao hơn. Gradient descent là thuật toán tìm giá trị nhỏ nhất của hàm số $f(x)$ dựa trên đạo hàm [17]. Thuật toán được thực hiện như sau:

- Bước 1: Khởi tạo giá trị $x=x_0$ tùy ý
- Bước 2: Gán $x = x - \text{learning_rate} * f'(x)$ (learning_rate là hằng số không âm)
- Bước 3: Tính lại $f(x)$. Nếu $f(x)$ đủ nhỏ thì dừng lại, ngược lại tiếp tục bước 2

Trong đó, learning_rate là **tốc độ học** và xảy ra 3 trường hợp như sau [18]:

- Nếu `learning_rate` nhỏ: Mỗi lần hàm số giảm rất ít nên cần rất nhiều lần thực hiện bước 2 để hàm số đạt giá trị nhỏ nhất.
- Nếu `learning_rate` hợp lý: Sau một số lần lặp bước 2 vừa phải thì hàm sẽ đạt giá trị đủ nhỏ.
- Nếu `learning_rate` quá lớn: Sẽ gây hiện tượng overshoot và không bao giờ đạt được giá trị nhỏ nhất của hàm.



Hình 5: Ba trường hợp của `learning_rate`

Ta sẽ lặp lại bước 2 một số lần đủ lớn (100 hoặc 1000 lần tùy vào bài toán và hệ số `learning_rate`) cho đến khi $f(x)$ đạt giá trị đủ nhỏ.

Nhận xét:

- Thuật toán hoạt động rất tốt trong trường hợp không thể tìm giá trị nhỏ nhất bằng đại số tuyến tính.
- Việc quan trọng nhất của thuật toán là tính đạo hàm của hàm số theo từng biến sau đó lặp lại bước 2.

Áp dụng vào bài toán cho Linear Regression: Chúng ta sẽ áp dụng thuật toán Gradient Descent vào hàm chi phí, các giá trị được cập nhật trong mỗi lần lặp lại. Chúng ta sẽ thực hiện thuật toán Gradient Descent cho đến khi tìm được các hệ số của phương trình tối ưu (weight và bias) [16].

$$B_0 = B_0 - \alpha \cdot \frac{2}{n} \sum_{i=1}^n (pred_i - y_i)$$

$$B_1 = B_1 - \alpha \cdot \frac{2}{n} \sum_{i=1}^n (pred_i - y_i) \cdot x_i$$

e) Các chỉ số, siêu tham số quan trọng trong mô hình

Bạn có thể cung cấp một số tham số tùy chọn cho LinearRegression [19]:

fit_intercept: bool, default=True

Việc tính toán hệ số chặn cho mô hình, nếu đặt bằng false thì sẽ coi hệ số chặn bằng 0. Tham số này được mặc định là True.

n_jobs: int, default=None

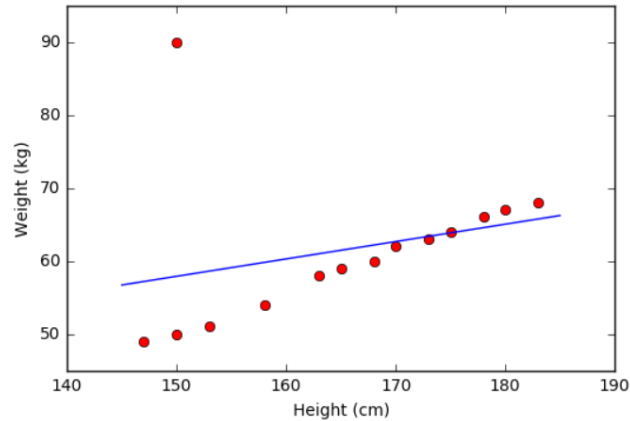
Đại diện cho số lượng công việc được sử dụng trong tính toán song song (*nếu như có nhiều hơn 1 biến mục tiêu*). None thường có nghĩa là một công việc và -1 sử dụng tất cả các bộ xử lý. Tham số này được mặc định là None.

positive: bool, default=False

Nếu chuyển thành True, nó sẽ bắt buộc các hệ số phải dương. Mặc định của tham số này là False.

2.3.2.3. Phương pháp cải thiện

Nhạy cảm với các giá trị ngoại lai, dữ liệu nhiễu => Phải xử lý giá trị ngoại lai trước khi chạy, đây là bước tiền xử lý dữ liệu



Hình 6: Hình ảnh minh họa cho việc nhảy cảm với dữ liệu nhiễu

Vấn đề overfitting (low bias, high variance): Vấn đề mô hình phù hợp với quá mức làm cho mô hình trở nên cụ thể hơn là chung chung [20]. Điều này thường dẫn đến độ chính xác đào tạo cao và độ chính xác kiểm tra rất thấp. Để giải quyết vấn đề này, chúng ta, có các cách sau:

- K-fold cross-validation: Chia tập dữ liệu thành k tập con, huấn luyện trên các tập dữ liệu khác nhau k lần, và từ đó, xây dựng ước lượng độ chính xác của mô hình học máy với dữ liệu mới.
- Nếu dữ liệu huấn luyện nhỏ thì thêm dữ liệu sạch và phù hợp hơn
- Nếu dữ liệu huấn luyện quá lớn, hãy thực hiện một số lựa chọn tính năng và xóa các tính năng không cần thiết.
- Regularization: Giúp chúng ta tránh overfit nhưng vẫn giữ nguyên tính bao quát của thuật toán (không cần phải loại bỏ bất kỳ feature nào của training set). Bằng cách thêm vào hàm chi phí một đại lượng \Rightarrow làm giảm độ lớn hàm và giảm sức ảnh hưởng của các feature lên để hàm đơn giản hơn.

Để xảy ra hiện tượng đa cộng tuyến (biến độc lập có mối quan hệ với nhau): Thông thường, các đầu vào không độc lập với nhau và do đó phải loại bỏ bất kỳ đa cộng tuyến nào trước khi áp dụng hồi quy tuyến tính, giải quyết bằng các cách sau [21]: Dựa vào ma trận tương quan và giá trị VIF. Thường thì, hệ số phóng đại phương sai (Variance Inflation

Factors) được sử dụng để đo lường mối tương quan và độ mạnh của mối tương quan giữa các biến dự báo trong mô hình hồi quy. Những cách khắc phục hiện tượng đa cộng tuyến:

Giải pháp 1: Loại bỏ biến độc lập có hệ số VIF vượt qua giá trị tiêu chuẩn. Bạn nên bỏ biến có VIF lớn nhất rồi chạy lại phân tích hồi quy xem thử có còn hiện tượng đa cộng tuyến hay không.

Giải pháp 2: Có thể đa cộng tuyến xảy ra do cỡ mẫu thu thập nhỏ. Bạn hãy thử thu thập thêm phiếu trả lời để tăng cỡ mẫu lên khoảng gấp 1,5 đến 2 lần. Khi cỡ mẫu lớn hơn sẽ làm giảm phương sai và ý nghĩa các kiểm định cũng sẽ có giá trị hơn.

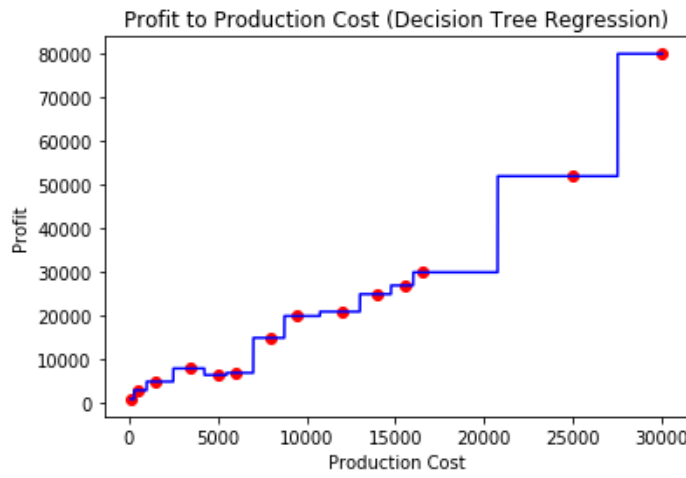
Giải pháp 3: Nếu vấn đề xuất phát từ chính bước chọn mô hình nghiên cứu và lập bảng khảo sát. Bạn có thể sẽ phải hủy bỏ dữ liệu thu thập và điều chỉnh lại mô hình, tiến hành khảo sát lại. Cho nên, bước lập cơ sở lý luận để đưa ra mô hình đề xuất và bảng khảo sát là rất quan trọng, các bạn nên làm cho thật tốt phần này qua sự hướng dẫn của giảng viên, những người có chuyên môn.

2.3.3. Decision Tree

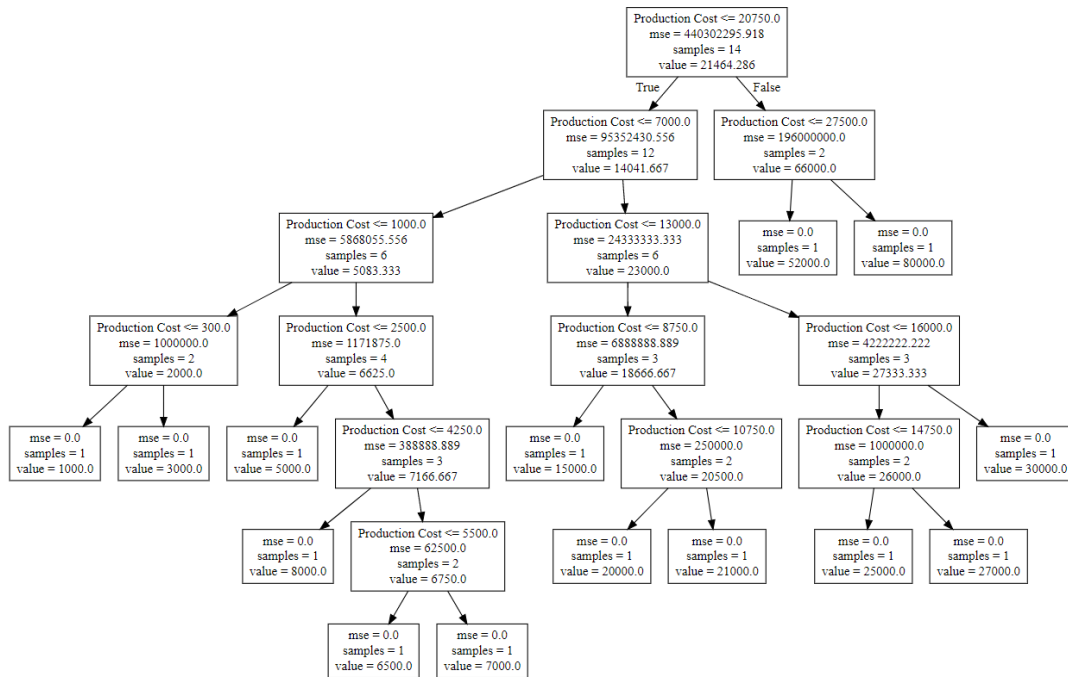
2.3.3.1. Khái niệm

Decision Tree (Cây quyết định) là một loại học máy có giám sát (Supervised Learning) được sử dụng để phân loại hoặc đưa ra dự đoán dựa vào việc trả lời một loạt các câu hỏi khác nhau.

Cây quyết định thực chất là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào các thuật toán cụ thể. Có thể xem cây quyết định là một hình thức biểu diễn các các câu lệnh điều kiện liên kết với nhau. Cây quyết định được sử dụng cho cả bài toán phân lớp và hồi quy. Với bài toán phân lớp, leaf node là một phân loại, còn với bài toán hồi quy, leaf node là một giá trị thực.

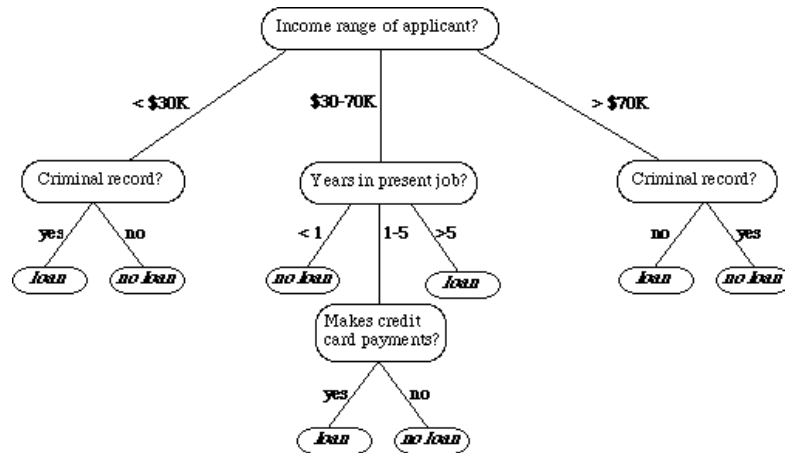


Hình 7: Biểu diễn kết quả dự đoán của Decision Tree Regression

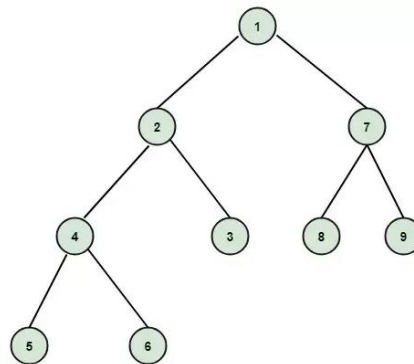


Hình 8: Kết quả phân chia của Decision Tree Regression

Decision Tree không phải là cây nhị phân (binary tree). Chỉ trong trường hợp các node không phải leaf node có tối đa 2 node con thì decision tree được coi là cây nhị phân.



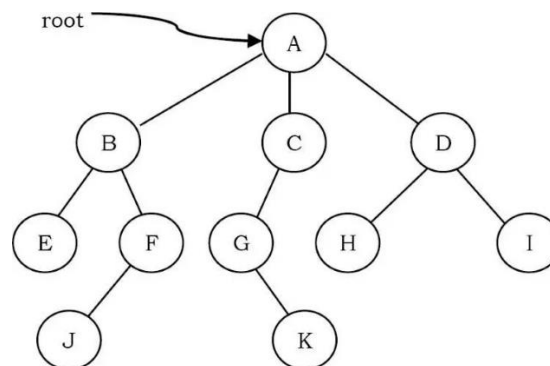
Hình 9: Cây quyết định không phải cây nhị phân



Hình 10: Cây nhị phân

2.3.3.2. Thuật toán

a) Ý tưởng chung



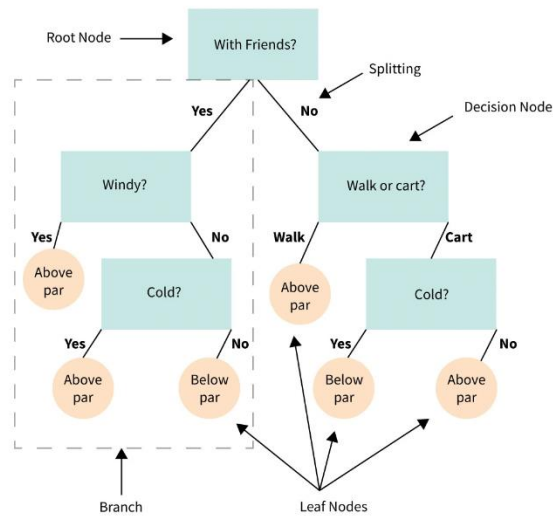
Hình 11: Mô tả cây quyết định

Cấu trúc của cây quyết định gồm các thành phần cơ bản sau [22]:

- Root node (nút gốc) là nút không có nút cha. Theo hình là nút A.
- Branch (nhánh) biểu diễn sự liên hệ giữa nút cha và nút con
- Leaf node (nút lá) là nút không có nút con. Theo hình là nút J, K, H, I.
- Những node con có cùng node cha được gọi là siblings (anh chị em ruột). Theo hình thì B, C, D là siblings của A và E, F là anh chị em của B.
- Một node p là ancestor (tổ tiên) của node q nếu tồn tại một đường đi từ gốc đến q và p xuất hiện trên đường đi. Node q được gọi là descendant (con cháu) của p. Ví dụ, A, C và G là tổ tiên của K.
- Tập hợp tất cả các node ở một độ sâu nhất định được gọi là level của cây (B, C và D là cùng một mức). Nút Root ở mức không.
- Độ sâu (depth) của node là độ dài của đường đi từ nút gốc đến node (độ sâu của G là 2, A - C - G).
- Chiều cao của một node là độ dài của đường đi từ nút đó đến nút sâu nhất. Chiều cao của cây là chiều dài của đường đi từ root đến node sâu nhất trong cây. Một cây chỉ có một node (gốc) có chiều cao bằng không. Theo hình trên, chiều cao của B là 2 (B - F - J).
- Kích thước của một node là số lượng nút con mà nó có bao gồm cả chính nó (kích thước của cây con C là 3).

Cây quyết định là sự kết hợp có cấu trúc của các điều kiện. Trong đó kết quả là nội dung của nút lá (node leaf) và các điều kiện dọc theo nhánh và nút tạo thành một liên kết trong mệnh đề if. Các điều kiện sẽ có dạng:

Nếu điều kiện 1 và điều kiện 2 và điều kiện n thì kết quả (leaf node).



Hình 12: Minh họa cây quyết định dự đoán số điểm của người chơi golf (Decision Tree)

Quy trình xây dựng một cây quyết định [23]:

Bước 1: Chọn lựa thuộc tính của data để chia data bằng các chỉ số ASM (Attribute Selection Measures: Chỉ số đánh giá lựa chọn thuộc tính).

Bước 2: Tạo decision node với feature và điều kiện.

Bước 3: Rẽ nhánh tạo các nút con và lặp lại tiến trình ở trên cho đến khi một trong các điều kiện sau thỏa mãn, ta sẽ có leaf node thỏa mãn các điều kiện dừng (sẽ được chỉ ra ở phần sau).

Bằng việc phân tích dữ liệu đầu vào, ASM đưa ra xếp hạng hiệu quả cho việc phân tách theo các thuộc tính của data, sau đó lựa chọn thuộc tính tốt nhất làm điểm để chia nhánh dữ liệu. Một số ASM phổ biến được giới thiệu ở phần sau đây.

b) Các chỉ số ASM phổ biến

Các chỉ số ASM phổ biến của thuật toán cây quyết định trong bài toán phân loại là: Information gain, GINI index, Gain Ratio. Đối với bài toán hồi quy, hiệu quả của sự phân tách nhánh dựa vào mức độ chênh lệch giữa dự đoán và kết quả thực tế. Mức độ chênh lệch này thường được thể hiện qua chỉ số ASM là Mean Square Error (MSE). Công thức của MSE như sau:

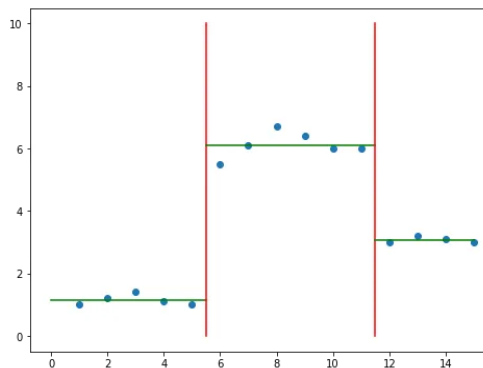
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Trong đó:

- Y là giá trị biến phụ thuộc thực tế
- Y mũ là giá trị biến phụ thuộc được dự đoán từ mô hình
- n là số mẫu của bộ dữ liệu đầu vào

Trong bài toán hồi quy, thuật toán sẽ tìm một điểm trong các biến độc lập để chia tập dữ liệu thành 2 phần, sao cho MSE đạt thấp nhất tại điểm đó. Thuật toán thực hiện điều này theo kiểu lặp đi lặp lại và tạo thành một cấu trúc dạng cây.

Chỉ số MSE nhỏ hơn đồng nghĩa với việc chênh lệch giữa kết quả dự đoán và thực tế càng nhỏ.



Hình 13: Mô tả kết quả của thuật toán cây quyết định hồi quy

Bên cạnh MSE, cây quyết định hồi quy còn có thể sử dụng các chỉ số ASM khác như Friedman_MSE, MAE, Poisson variance.

c) Tiêu chuẩn dừng

Decision Tree trong bài toán hồi quy có thể phân tách cho đến khi chỉ số MSE của các nút đạt về 0 hoặc chỉ còn 1 điểm dữ liệu trong một nút. Khi các nút đạt đến mức MSE = 0 và dừng lại, tất cả các điểm trong tập dữ liệu tập huấn đều được dự đoán đúng, cây quyết định cũng sẽ trở nên phức tạp và có nguy cơ overfitting cao.

Vì vậy, người ta đề ra một số tiêu chuẩn dừng trong việc phân nút cho cây quyết định. Nhưng không có một tiêu chuẩn dừng nào là đúng trong mọi trường hợp, vì vậy việc quyết định tiêu chuẩn dừng phù hợp phụ thuộc nhiều vào kinh nghiệm của người phân tích và bối cảnh của việc phân tích.

d) Các chỉ số, siêu tham số quan trọng

Mô hình Decision Tree có một số siêu tham số đầu vào quan trọng có ảnh hưởng đáng kể đến kết quả dự đoán của mô hình (*sklearn.tree.DecisionTreeRegressor*) được cung cấp bởi thư viện Scikit-learn như sau:

criterion: {"squared_error", "friedman_mse", "absolute_error", "poisson"}, default="squared_error"

Chỉ số quyết định sự phân tách nhánh trong cây quyết định. "squared_error" là chỉ số MSE, "friedman_mse" là chỉ số nâng cao của MSE trong một vài trường hợp. "absolute_error" là chỉ số MAE, "poisson" sử dụng độ lệch của phân phối Poisson để phân tách nhánh.

splitter: {"best", "random"}, default="best"

Quyết định cách phân chia tại mỗi nút. Các lựa chọn đầu vào của splitter là "best" để chọn cách phân chia tốt nhất hoặc "random" để chọn cách phân chia ngẫu nhiên tốt nhất.

max_depth: int, default=None

Độ sâu tối đa của cây. Nếu "None", thì các nút được mở rộng cho đến khi tất cả các lá đều sạch hoặc cho đến khi tất cả các lá chứa ít hơn min_samples_split mẫu.

min_samples_split: int hoặc float, default=2

Số lượng mẫu tối thiểu cần thiết để tách một nút.

- Nếu kiểu dữ liệu là int, thì coi min_samples_split là số mẫu tối thiểu.
- Nếu kiểu dữ liệu là float, thì min_samples_split là một phân số và $\text{ceil}(\text{min_samples_split} * n_samples)$ là số lượng mẫu tối thiểu cho mỗi lần phân chia. (ceil là một hàm làm tròn)

min_samples_leaf: int hoặc float, default=1

Số lượng mẫu tối thiểu cần có tại một nút lá. Một điểm phân chia ở bất kỳ độ sâu nào sẽ chỉ được xem xét nếu nó để lại ít nhất `min_samples_leaf` mẫu huấn luyện trong mỗi nhánh trái và phải. Điều này có thể có tác dụng làm mô hình “mượt” (smooth) hơn, đặc biệt là trong hồi quy[9].

max_features: int, float or {“auto”, “sqrt”, “log2”}, default=None

Số lượng feature cần xem xét khi thực hiện phân chia. Kiểu dữ liệu cho `max_feature` thường được dùng là int.

random_state: int, giá trị RandomState hoặc None, default=None

Về cơ bản, sự thay đổi các thông số nhập vào cho `random_state` không có ảnh hưởng đến kết quả cuối cùng [11].

max_leaf_nodes: int, default=None

Số lượng nút lá tối đa cho phép.

min_impurity_decrease: float, default=0.0

Một nút sẽ bị phân tách nếu sự phân tách này giúp làm giảm tạp chất (impurity) lớn hơn hoặc bằng giá trị này.

Trong các thông số trên, người ta thường tập trung và điều chỉnh các siêu tham số là `max_depth`, `min_samples_split`, `min_samples_leaf`, `max_features`, `min_impurity_decrease`[10].

2.3.3.3. Phương pháp cải thiện

Decision tree có nhiều điểm mạnh đáng chú ý như không phụ thuộc vào loại dữ liệu đầu vào là gì, phân bố ra sao. Thuật toán này có thể biểu diễn khá dễ hiểu về mặt kỹ thuật (ví dụ so với KNN và một số thuật toán phân loại khác). Ngoài ra, thuật toán này cũng không yêu cầu chuẩn hóa dữ liệu (Normalization) và scaling [24].

Nhưng bên cạnh đó, decision tree cũng có những hạn chế, có thể kể đến:

- Dễ bị overfitting;
- Tính toán lâu và phức tạp nếu bộ dữ liệu quá lớn hay có nhiều chiều;
- Dành nhiều thời gian để huấn luyện mô hình;
- Chưa tối ưu để áp dụng cho các bài toán hồi quy và dự đoán ra giá trị liên tục;

- Nếu có sự thay đổi trong bộ dữ liệu, dù là nhỏ cũng có thể ảnh hưởng đến toàn bộ cây quyết định;
- Không được thư viện Scikit-learn hỗ trợ trong việc xử lý dữ liệu phân loại (categorical variables).

Vấn đề đáng quan tâm nhất và có thể cải thiện là overfitting. Sau đây là một số phương pháp đề xuất để giảm overfitting ở cây quyết định.

a) Đặt ra một tiêu chuẩn dừng - Prunning

Prunning là quá trình cắt tỉa một cây quyết định để nó trở nên đơn giản và bao quát hơn. Trước khi bắt đầu prunning, một cây quyết định sẽ được xây dựng để phân lớp đúng mọi điểm trong tập huấn luyện. Kỹ thuật prunning sẽ cắt tỉa các leaf-node có chung một non-leaf-node, non-leaf-node vì vậy mà trở thành leaf-node. Phân lớp của leaf node mới sẽ tương ứng với class chiếm đa số trong số mọi điểm được phân vào node đó. Điều này cũng đồng nghĩa với việc đặt ra một số tiêu chuẩn dừng cho cây quyết định.

Người ta đưa ra một số tiêu chí cụ thể để đặt điều kiện dừng cho việc phân tách các nút trong cây quyết định để tránh overfitting. Tuy nhiên, những tiêu chuẩn này chỉ mang tính tham khảo, người phụ trách phân tích sẽ phải dựa vào bối cảnh phân tích, tính chất của tập dữ liệu và kinh nghiệm của bản thân để đưa ra lựa chọn phù hợp:

- Nếu nút đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Lớp của nút lá này có thể được xác định dựa trên lớp chiếm đa số trong nút.
- Nếu độ sâu nút gốc đến nút lá đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của cây này làm giảm độ phức tạp của cây và phần nào giúp tránh overfitting.
- Nếu tổng số nút lá vượt quá một ngưỡng nào đó.
- Nếu việc phân chia node đó giúp giảm tạp chất (impurity) đến một ngưỡng nào đó.

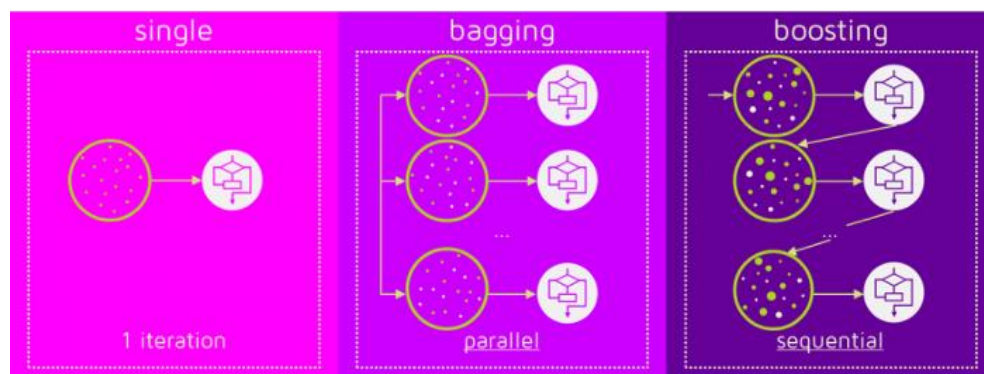
b) Ensemble methods: Random forest

Ensemble methods là phương pháp tổng hợp kết quả dự đoán của nhiều model để đưa ra model cuối cùng. Đối với cây quyết định, ensemble methods sẽ tập hợp kết quả của nhiều cây quyết định để đưa ra một kết quả tối ưu nhất.

Đối với các bài toán phân loại, ensemble methods sẽ chọn mode của lớp thường được dự đoán nhất. Đối với bài toán hồi quy, ensemble methods sẽ lấy kết quả là giá trị trung bình của các model.

Đối với các nhiệm vụ phân loại, đầu ra của Random forest là loại được chọn bởi hầu hết các cây. Đối với các tác vụ hồi quy, giá trị trung bình hoặc dự đoán trung bình của các cây riêng lẻ được trả về.

Các thuật toán ensemble có thể được chia thành 03 nhóm chính là: bagging, boosting và stacking [25].



Hình 14: So sánh model thường, bagging và boosting. Nguồn: Viblo.

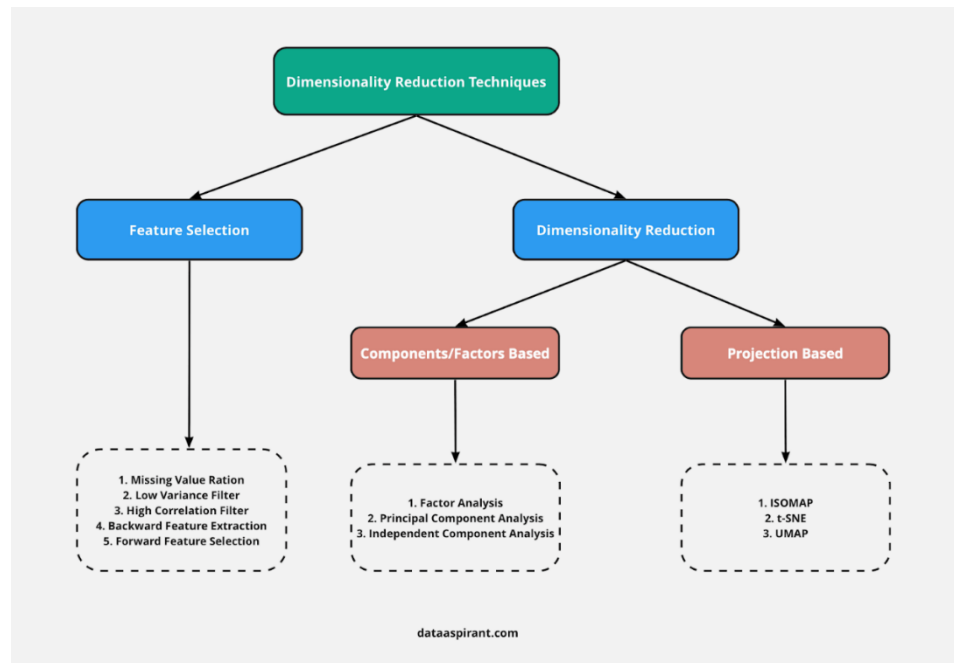
Một thuật toán tập hợp khá nổi tiếng thuộc nhóm bagging và cải thiện decision tree là random forest.

c) Feature selection hoặc dimensionality reduction

Feature selection và dimensionality reduction được sử dụng để giảm số lượng feature trong tập dữ liệu. Hai phương pháp này hoạt động trên những nguyên tắc khác nhau như sau:

- Feature selection sử dụng các phương pháp tính điểm hoặc thống kê để chọn feature nào cần giữ và feature nào cần xóa. Một số phương pháp có thể sử dụng là Filter methods, Wrapper methods, Embedded methods.

- Dimensionality reduction là quá trình giảm số lượng feature (hoặc số chiều) trong tập dữ liệu đảm bảo giữ lại càng nhiều thông tin càng tốt. Một số thuật toán có thể tham khảo là PCA, SVD, LDA.



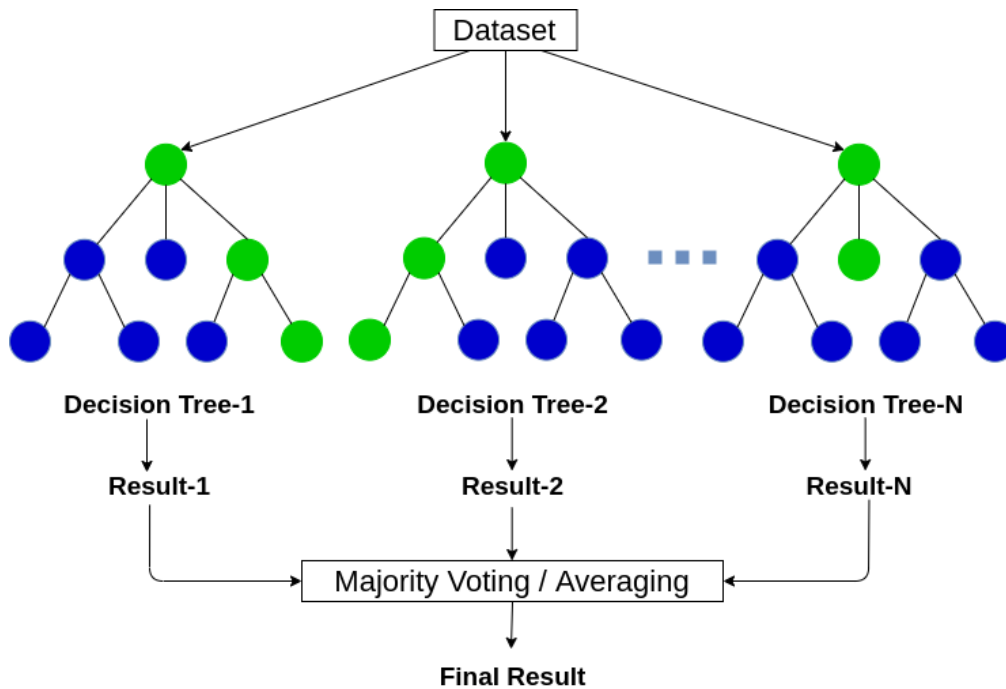
Hình 15: Các phương pháp giảm chiều dữ liệu. Nguồn: Dataaspirant

2.3.4. Random Forest

2.3.4.1. Khái niệm

Random Forest là một thuật toán học máy có giám sát được dùng để phục vụ cho mục đích phân loại, hồi quy,... bằng cách xây dựng và kết hợp đầu ra của nhiều cây quyết định (Decision tree) để đạt được một kết quả duy nhất.

Random forest có thể sử dụng cho cả bài toán Phân loại (Classification) và Hồi quy (Regression) với nhiều ứng dụng như công cụ đề xuất, phân loại hình ảnh, phân loại nhãn lớp, dự đoán kết quả,... [26]



Hình 16: Biểu diễn mô hình Random Forest cơ bản

2.3.4.2. Thuật toán

a) Ý tưởng chung

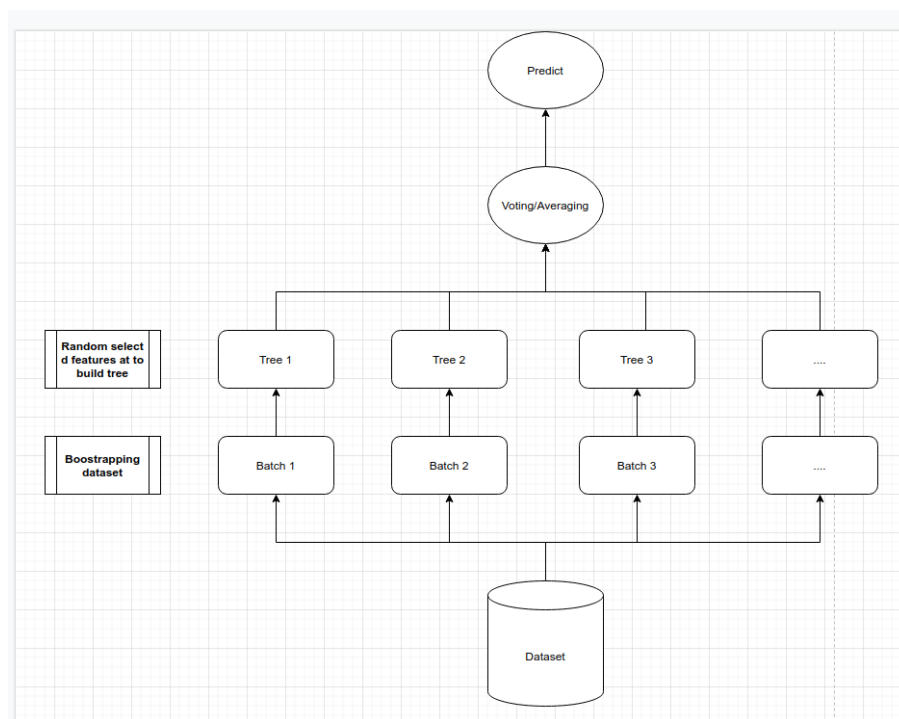
Mô hình Random Forest hoạt động dựa trên sự phối hợp giữa mô hình kết hợp (ensembling) và quá trình lấy mẫu tái lập (bootstrapping). Trong đó, một cây quyết định sẽ biểu diễn một kế hoạch, một quy trình trả lời cho một câu hỏi mà đầu ra của mỗi cây là câu trả lời tương ứng [27]. Mỗi Node của cây sẽ là các thuộc tính, và các nhánh là giá trị lựa chọn của thuộc tính đó, để đưa ra kết quả cuối cùng, cây quyết định sẽ đi qua, thu thập và xử lý các giá trị thuộc tính trên cây [28].

Ý tưởng của thuật toán này là nếu như sức mạnh của một cây quyết định là yếu thì hợp sức của nhiều cây quyết định sẽ trở nên mạnh mẽ hơn (một cây làm chẳng nên non, ba cây chụm lại nên hòn núi cao). Trong mô hình Random forest, một tập hợp nhiều cây quyết định sẽ được tạo ra dựa trên các mẫu dữ liệu được chọn ngẫu nhiên, các kết quả từ các cây quyết định sẽ được tổng hợp lại lấy giá trị trung bình hoặc bỏ phiếu đa số để chọn ra cây quyết định cuối cùng đưa ra kết quả tốt nhất. Như vậy một kết quả dự báo được tổng hợp

từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình [2].

Quy tắc chọn kết quả cuối cùng:

- Trong bài toán phân loại, rừng ngẫu nhiên thực hiện bỏ phiếu đa số: mỗi cây đưa ra kết quả phân loại lớp, sau đó tập hợp tất cả các kết quả thu được tính tỉ lệ/tần suất xuất hiện của từng phân loại lớp trên các cây. Kết quả cuối cùng lớp được phân loại của mô hình là lớp có tỉ lệ/tần suất xuất hiện lớn hơn.
- Trong bài toán hồi quy, Rừng ngẫu nhiên thực hiện lấy giá trị trung bình: mỗi cây đưa ra kết quả dự đoán, sau đó tập hợp tất cả các kết quả thu được tính giá trị trung bình là kết quả cuối cùng.



Hình 17: Ý tưởng thuật toán Random Forest

b) Mô hình kết hợp (ensemble model)

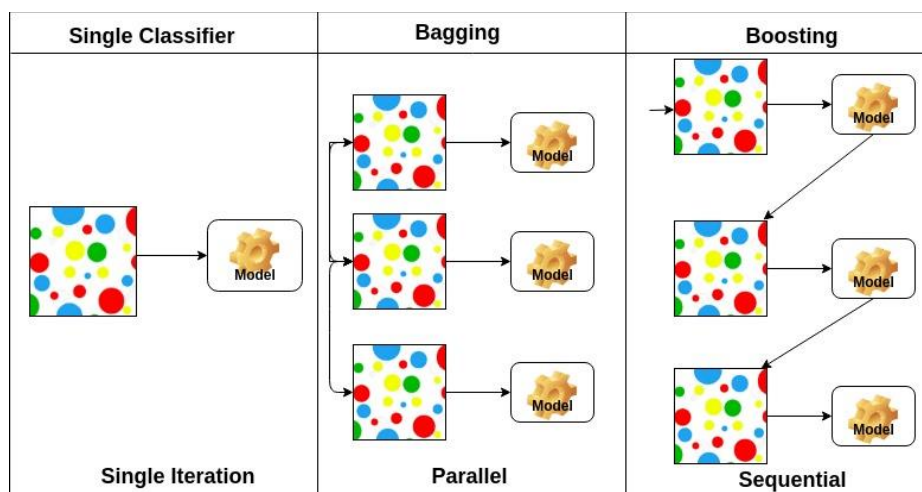
Để chắc chắn hơn về kết quả và độ chính xác của mô hình thì người ta thường tham vấn thêm kết quả từ nhiều mô hình khác hơn và tiến hành bầu cử/lấy giá trị trung bình kết

quả trả về giữa chúng. Những kết quả từ mô hình kết hợp được cho rằng tốt hơn so với kết quả chỉ từ một mô hình do chịu ảnh hưởng bởi trí thông minh đám đông (The wisdom of crowds) [27].

Ensemble methods là phương pháp tổng hợp kết quả dự đoán của nhiều model để đưa ra model cuối cùng. Đối với Random Forest, ensemble methods sẽ tập hợp kết quả của nhiều cây quyết định (Decision Tree) để đưa ra một kết quả tối ưu nhất.

Đối với các bài toán phân loại, ensemble methods sẽ chọn mode của lớp thường được dự đoán nhiều nhất. Đối với bài toán hồi quy, ensemble methods sẽ lấy kết quả là giá trị trung bình của các model

Các thuật toán ensemble có thể được chia thành 03 nhóm chính là: bagging, boosting và stacking.



Hình 18: So sánh model thường, bagging và boosting

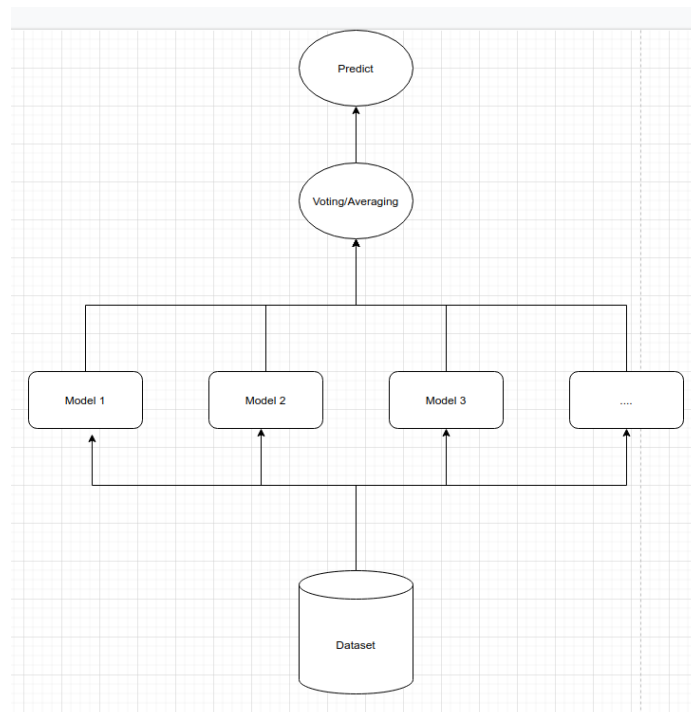
- **Bagging:** Xây dựng một lượng lớn các model (thường là cùng loại) dựa trên những subsamples khác nhau từ tập training dataset theo quy tắc chọn ngẫu nhiên [25]. Những model này sẽ được huấn luyện độc lập và song song với nhau cho ra kết quả đầu ra là giá trị trung bình hoặc kết quả được bầu chọn đa số.
- **Boosting:** Xây dựng một lượng lớn các model (thường là cùng loại). Mỗi model sau sẽ học cách sửa những lỗi errors của các model trước tạo thành một chuỗi các model

mà model sau sẽ tốt hơn model trước [25]. Như vậy, kết quả của model cuối cùng trong chuỗi model này sẽ là kết quả tốt nhất trả về.

- **Stacking:** Xây dựng mô hình cơ sở (base model) gồm một số loại model thường khác loại và một siêu mô hình (meta model) thường là supervisor model, huấn luyện những model này độc lập và song song, sau đó meta model sẽ kết hợp kết quả của một số mô hình để cho ra kết quả tốt nhất [25]. Mô hình cơ sở sử dụng nhiều loại mô hình khác nhau thường phức tạp và đa dạng để đưa ra các giả định rất khác nhau về cách giải quyết cùng một nhiệm vụ, có thể là mô hình tuyến tính, cây quyết định, rừng ngẫu nhiên,... Trong khi, siêu mô hình thường đơn giản, cung cấp sự giải thích trôi chảy về các dự đoán do các mô hình cơ sở đưa ra.

+ Siêu mô hình hồi quy: Hồi quy tuyến tính Linear Regression (dự đoán một giá trị số);

+ Siêu mô hình phân loại: Hồi quy logistic (dự đoán nhãn lớp).

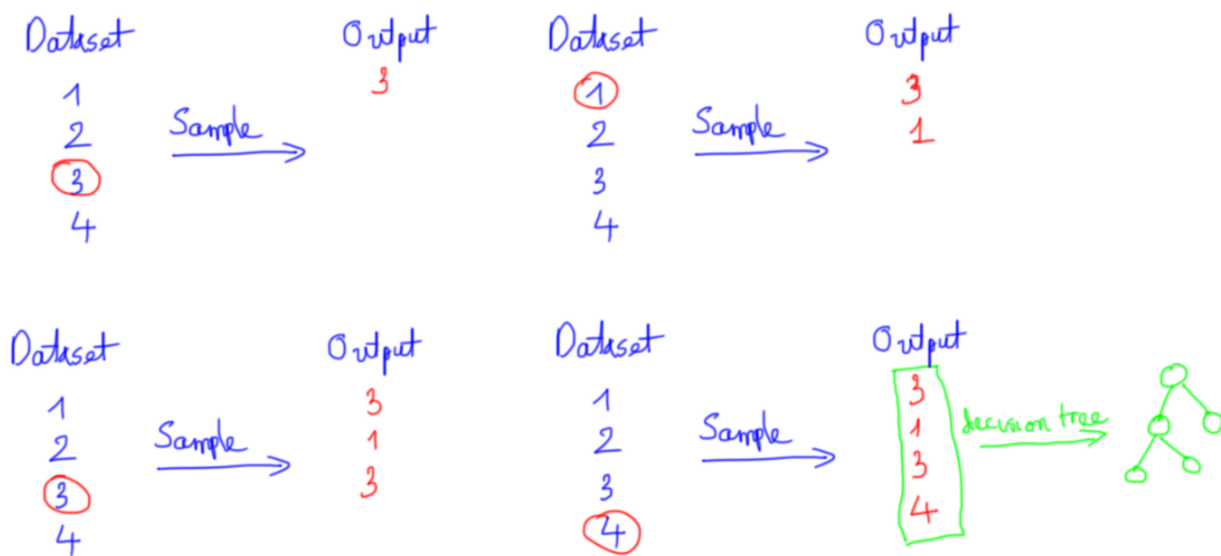


Hình 19: Minh họa bagging

Lấy mẫu tái lập (bootstrapping)

Là một phương pháp lấy mẫu dùng để ước lượng lỗi chuẩn (standard errors), độ lệch (bias) và tính toán khoảng tin cậy (confidence interval) cho các tham số được giới thiệu bởi Bradley Efron vào năm 1979 [29]. Phương pháp này được thực hiện như sau: Giả định dữ liệu huấn luyện mô hình là một tập D bao gồm N quan sát. Thuật toán rừng cây sẽ sử dụng phương pháp lấy mẫu tái lập để tạo thành tập dữ liệu con B . Quá trình lấy mẫu tái lập này còn gọi là bỏ túi (bagging). Tức là chúng ta sẽ thực hiện M lượt nhặt các mẫu từ tổng thể và bỏ vào túi để tạo thành tập dữ liệu B_i [27].

$$B_i = \{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)}), \dots, (x_M^{(i)}, y_M^{(i)})\}.$$



Hình 20: Minh họa cách lấy mẫu tái lập

Bagging

Phương pháp này được xem như là một phương pháp tổng hợp kết quả có được từ các bootstrap. Ý tưởng chính của phương pháp này như sau: Cho một tập huấn luyện $B_i = \{(x_i, y_i) : i=1, 2, \dots, n\}$ và giả sử chúng ta muốn có một dự đoán nào đó đối với biến x . Sau khi lấy mẫu, với mỗi tập dữ liệu B_i chúng ta xây dựng và tập huấn một mô hình cây quyết định và lần lượt thu thập kết quả dự báo trả về là $y_j(i) = f_i(x_j)$. Trong đó $y_j(i)$ là dự

báo của quan sát thứ j từ mô hình thứ i), x_j là giá trị véc tơ đầu vào, f_i là hàm dự báo của mô hình thứ i . Mô hình dự báo từ cây quyết định là giá trị trung bình hoặc bầu cử của B cây quyết định [29].

Đối với mô hình hồi quy: Chúng ta tính giá trị trung bình của các dự báo từ mô hình con.

$$y_j = \frac{1}{B} \sum_{i=1}^B y_{ji}(i) = \frac{1}{B} \sum_{i=1}^B f_i(x_j)$$

Trong đó, độ lệch chuẩn của dự đoán từ tất cả các cây hồi quy riêng lẻ trên x' được tính bằng công thức:

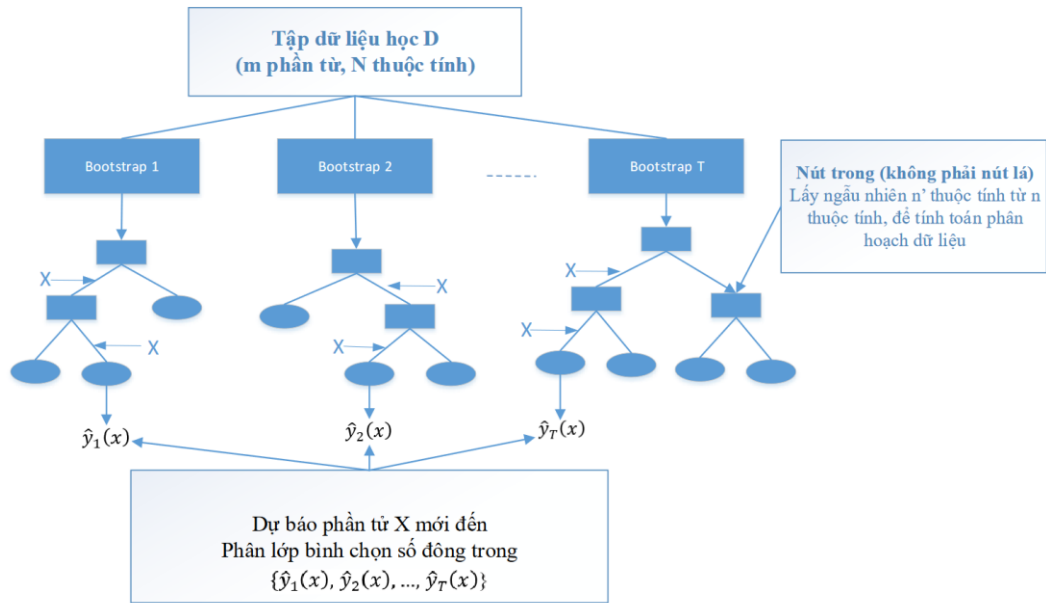
$$\sigma = \sqrt{\frac{\sum_{b=1}^B (f_b(x') - \hat{f})^2}{B - 1}}.$$

Đối với mô hình phân loại: Chúng ta thực hiện bầu cử từ các mô hình con để chọn ra nhãn dự báo có tần suất lớn nhất.

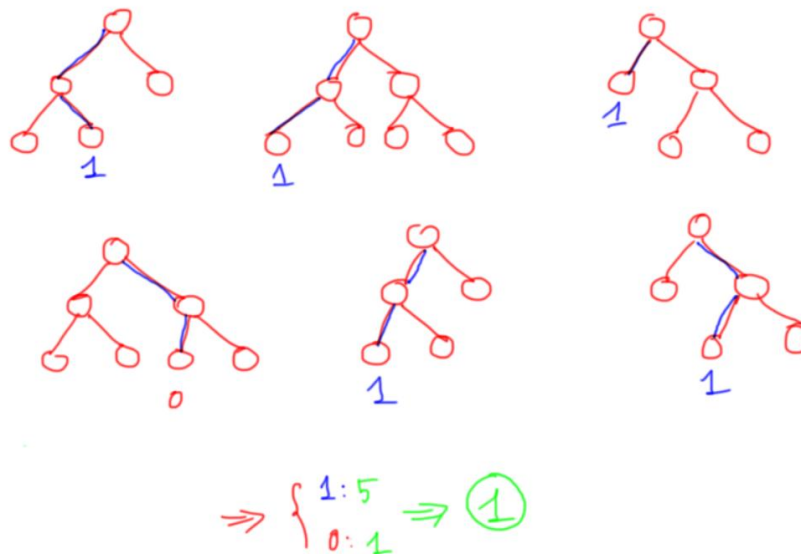
$$y_j = \operatorname{argmax}_c \sum_{i=1}^B \mathbb{1}_{p(y_{ji}=c)}$$

c) Xây dựng thuật toán *Random Forest*

1. Chọn T là số lượng các cây thành phần sẽ được xây dựng.
2. Từ tập dữ liệu D , lấy mẫu tái lập Bootstrapping m dữ liệu ngẫu nhiên để tạo thành một tập dữ liệu con.
3. Sau khi lấy được m dữ liệu từ bước 1 thì chọn ra ngẫu nhiên n thuộc tính tạo thành bộ dữ liệu mới gồm m dữ liệu và mỗi dữ liệu có n thuộc tính.
4. Dùng thuật toán Decision Tree để xây dựng mô hình cây quyết định với bộ dữ liệu ở bước 2. Lặp lại T lần để xây dựng T cây quyết định cho mô hình Random Forest.
5. Thực hiện *bầu cử* hoặc lấy *trung bình* giữa các cây quyết định để đưa ra kết quả dự báo cuối cùng.



Hình 21: Minh họa cách xây dựng Random Forest



Hình 22: Quy chế voting trong Random Forest

Trong Random Forest, các phương sai (variance) được giảm thiểu do kết quả của RF được tổng hợp thông qua nhiều bộ học (learner) và việc chọn ngẫu nhiên tại mỗi bước trong RF sẽ làm giảm mối tương quan (correlation) giữa các bộ phận lớp trong việc tổng hợp các kết quả tránh tình trạng lỗi chung.

d) Các chỉ số, siêu tham số quan trọng

Tương tự với thuật toán cây quyết định (Decision Tree), Random Forest cũng kế thừa những chỉ số và siêu tham số tương ứng. Ngoài những chỉ số lựa chọn đánh giá lựa chọn thuộc tính như Decision Tree, một số siêu tham số trong Random Forest quan trọng cần chú ý như:

Siêu tham số **n_estimators (int, default=100)** chỉ số cây quyết định mà thuật toán xây dựng trước khi lấy phiếu bầu tối đa hoặc lấy giá trị trung bình của các dự đoán. Số lượng cây cao hơn làm tăng hiệu suất của mô hình và làm cho các dự đoán ổn định hơn, nhưng nó cũng làm chậm quá trình tính toán và tăng chi phí tính toán cho đào tạo và dự đoán.

Siêu tham số **bootstrap (bool, default=True)** chọn True tương ứng với sử dụng phương pháp lấy mẫu tái lập khi xây dựng các cây quyết định. Ngược lại thì chúng ta sử dụng toàn bộ dữ liệu.

Siêu tham số **max_depth (int, default=None)** là độ sâu tối đa của mỗi cây quyết định trong rừng. Mặc định là None, các nút được mở rộng cho đến khi ra tất cả các lá hoặc cho đến khi tất cả các lá chứa ít hơn min_samples_split mẫu tối thiểu. Đặt giá trị cao hơn cho max_depth có thể dẫn đến trang bị thừa trong khi đặt giá trị quá thấp có thể dẫn đến trang bị thiếu. Đối với mô hình bị quá khớp thì chúng ta cần giảm độ sâu và vị khớp thì gia tăng độ sâu.

Siêu tham số **criterion ({“squared_error”, “absolute_error”, “friedman_mse”, “poisson”}, default=“squared_error”)** tiêu chí xác định chức năng đo lường chất lượng của một sự phân tách. Với RandomForestClassifier sẽ có các chỉ số như “gini”, “entropy”, “log_loss”, trong đó mặc định là “gini”. Các tiêu chí được hỗ trợ là “gini” đối với tạp chất Gini và “log_loss” và “entropy” đối với cả thông tin thu được của Shannon. Với RandomForestRegressor sẽ có các chỉ số như “squared_error”, “absolute_error”, “friedman_mse”, “poisson”, trong đó mặc định là “squared_error”. Tiêu chí được hỗ trợ là “squared_error” cho lỗi bình phương trung bình, bằng với giảm phương sai làm tiêu chí lựa chọn features, “friedman_mse” sử dụng lỗi bình phương trung bình với điểm cải thiện của Friedman cho các phân tách tiềm năng, “absolute_error” cho lỗi tuyệt đối trung bình

sử dụng trung vị của mỗi nút đầu cuối và “poisson” sử dụng giảm độ lệch Poisson để tìm sự phân tách.

Siêu tham số **n_jobs (int, default=None)** cho động cơ biết nó được phép sử dụng bao nhiêu bộ xử lý để kiểm soát tính song song của mô hình. Vì mỗi cây được tạo độc lập từ các dữ liệu và thuộc tính khác nhau nên chúng ta hoàn toàn có thể sử dụng CPU để chạy song song quy trình xây dựng các cây quyết định, huấn luyện, dự đoán và đưa ra quyết định trên các cây. Nếu nó có giá trị là một, nó chỉ có thể sử dụng một bộ xử lý. Giá trị “-1” có nghĩa là không có giới hạn.

Siêu tham số **max_features ({“sqrt”, “log2”, None}, int or float, default=1.0)** là số lượng tối đa các đối tượng mà rừng ngẫu nhiên xem xét để tách một nút. Mặc định là toàn bộ các features đầu vào $\text{max_features} = \text{n_features}$, nếu là “sqrt”, thì $\text{max_features} = \sqrt{\text{n_features}}$ còn “log2”, thì $\text{max_features} = \log_2(\text{n_features})$.

Siêu tham số **min_samples_leaf (int or float, default=1)** xác định số lượng lá tối thiểu cần thiết để tiếp tục phân chia đối với node quyết định. Được sử dụng để tránh kích thước của node lá quá nhỏ nhằm giảm thiểu hiện tượng quá khớp.

Siêu tham số **max_leaf_nodes (int, default=None)** là số lượng các node lá tối đa của cây quyết định. Thường được thiết lập khi muốn kiểm soát hiện tượng quá khớp.

Siêu tham số **max_samples (int or float, default=None)** là số lượng mẫu cần lấy từ tập dữ liệu D để huấn luyện. Mặc định là $D.\text{shape}[0]$ nghĩa là số lượng mẫu cần lấy bằng với số hàng, số lượng điểm quan sát trong tập dữ liệu D.

Siêu tham số **min_impurity_decrease (float, default=0.0)**: Ngưỡng để dừng sớm - threshold (early stopping) quá trình phát triển của cây quyết định. Một nút sẽ bị phân tách nếu độ vẩn đục (impurity) lớn hơn hoặc bằng giá trị ngưỡng threshold.

Siêu tham số **min_samples_split (int or float, default=2)**: Số lượng mẫu tối thiểu cần thiết để phân chia một internal node. Nếu kích thước mẫu ở một internal node nhỏ hơn ngưỡng thì ta sẽ không rẽ nhánh internal node.

Siêu tham số **random_state (int, default=None)** kiểm soát cả tính ngẫu nhiên của việc khởi động các mẫu được sử dụng khi xây dựng cây (nếu `bootstrap=True`) và lấy mẫu các tính năng cần xem xét khi tìm kiếm sự phân chia tốt nhất tại mỗi nút (nếu `max_features`

< n_features). Mô hình sẽ luôn tạo ra cùng một kết quả khi nó có một giá trị xác định là random_state, trong đó, random_state còn là bước nhảy giữa các tham số với nhau, các điểm quan sát dữ liệu sẽ cách nhau 1 khoảng random state. Với random state = 0, chúng ta nhận được cùng một tập dữ liệu huấn luyện và thử nghiệm trên các lần thực thi khác nhau. Với random state = 42, chúng ta nhận được cùng một tập dữ liệu huấn luyện và thử nghiệm trên các lần thực thi khác nhau, nhưng lần này tập dữ liệu huấn luyện và thử nghiệm khác với trường hợp trước với random state = 0.

Cuối cùng là **oob_score (bool, default=False)** (còn được gọi là lấy mẫu oob) là một phương pháp xác thực chéo rừng ngẫu nhiên để tính số lượng hàng được dự đoán chính xác từ mẫu out of bag và số lần phân loại sai mẫu oob (Obb_Error). Khi sử dụng phương pháp bootstrap lấy mẫu tái lập ngẫu nhiên nên trong mỗi lần lấy thu được tập dữ liệu con có khoảng 2/3 các mẫu không trùng nhau dùng để xây dựng cây, các mẫu này được gọi là in-bag. Khoảng một phần ba dữ liệu còn lại gọi là out-of-bag không được sử dụng để đào tạo mô hình nhưng có thể được sử dụng để đánh giá hoạt động của mô hình, tính toán độ quan trọng thuộc tính của các cây quyết định trong rừng cũng như sử dụng để ước lượng lỗi tạo ra từ việc kết hợp các kết quả từ các cây tổng hợp trong Random Forest.

2.3.4.3. Phương pháp cải thiện

Random Forest chậm tạo dự đoán bởi vì mô hình khó hiểu và phức tạp hơn so với cây quyết định rất nhiều, thời gian huấn luyện của rừng có thể kéo dài tùy số cây và số thuộc tính phân chia. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây quyết định trong Random Forest đều phải chạy lại và đưa ra dự đoán của từng cây để sau đó kết tiến hành bỏ phiếu hoặc lấy giá trị trung bình trên đó làm kết quả cuối cùng. Vì vậy cần tối ưu hóa các siêu tham số để cải thiện độ chính xác, thời gian thực thi cũng như tài nguyên hệ thống của thuật toán bằng cách tối ưu bộ dữ liệu và sử dụng các phương pháp điều chỉnh tham số thích hợp để tìm ra mô hình nào tối ưu nhất như GridSearchCV, RandomizeSearchCV,...

Thuật toán Random Forest trong bài toán phân loại có những hạn chế với các lớp mất cân bằng vì nó sử dụng một mẫu bootstrap của tập huấn luyện để tạo thành từng cây, vì thế mà khả năng các mẫu bootstrap chứa ít hoặc không có lớp thiểu số tăng lên đáng kể,

dẫn đến kết quả mô hình không được khách quan. Việc không cân bằng nhãn lớp khiến kết quả dự đoán của thuật toán có thể lệch về số đông nhãn lớp bởi đa phần kết quả dự báo ra thường thiên về 1 nhóm là nhóm đa số và rất kém trên nhóm thiểu số [30]. Để giải quyết vấn đề mất cân bằng dữ liệu, số nhãn lớp, có một số phương pháp như sau:

- Under sampling giảm số lượng các quan sát của nhóm đa số để nó trở nên cân bằng với số quan sát của nhóm thiểu số. Ưu điểm là làm cân bằng mẫu nhanh chóng, không cần thuật toán giả lập mẫu, dễ dàng thực hiện. Nhược điểm là kích thước mẫu sẽ bị giảm đi, không phù hợp với tập dữ liệu nhỏ [30].
- Over sampling gia tăng kích thước mẫu thuộc nhóm thiểu số bằng các phương pháp khác nhau để giải quyết tình trạng mất cân bằng nhãn. Trong đó, 2 phương pháp chính là: Lựa chọn mẫu có tái lập (Naive random Over sampling) & Tạo mẫu tổng hợp (SMOTE - Synthetic Minority Over-sampling và ADASYN - Adaptive synthetic sampling) [30].
- Sử dụng các chỉ số đánh giá khác khi hiện tượng mất cân bằng dữ liệu nghiêm trọng xảy ra vì lúc này chỉ số đánh giá mô hình accuracy - độ chính xác thường không hiệu quả. Một số chỉ số đánh giá và phương pháp thay thế như precision, recall, f1-score, gini, confusion matrix, auc,... [30]

Tương tự thuật toán Decision Tree, Random Forest cũng khó giải quyết những bài toán có dữ liệu chuỗi thời gian hay phụ thuộc thời gian liên tục, các thuộc tính dự đoán có tương quan tuyến tính với biến mục tiêu, có nhiều biến phân loại dễ xảy ra lỗi khi có quá nhiều tính toán để xây dựng mô hình cây quyết định [31]. Cách giải quyết là sử dụng Feature selection (Filter, Wrapper, Embedded), Dimensionality reduction (Linear discriminant analysis - LDA và principal components analysis - PCA) để giảm số lượng feature trong tập dữ liệu.

Mô hình Random Forest không giỏi tổng quát hóa các trường hợp với dữ liệu hoàn toàn mới trong bài toán hồi quy. Ví dụ: Nếu chúng ta biết rằng giá của một cây kem là 1 đô la, 2 cây kem có giá 2 đô la và 3 cây kem có giá 3 đô la, thì 10 cây kem có giá bao nhiêu? Trong những trường hợp như vậy, các mô hình hồi quy tuyến tính Linear Regression

có thể dễ dàng tìm ra điều này, trong khi Rừng ngẫu nhiên Random Forest thì lại gặp khó khăn để tìm câu trả lời.

2.3.5. Các chỉ số đánh giá

2.3.5.1. Mean Absolute Error - MAE

Mean Absolute Error (MAE) đo độ lớn trung bình của các lỗi trong một tập hợp các dự đoán mà không cần xem xét hướng của chúng. Đó là giá trị trung bình trên mẫu thử nghiệm về sự khác biệt tuyệt đối giữa dự đoán và quan sát thực tế, trong đó tất cả các khác biệt riêng lẻ có trọng số bằng nhau [32].

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

Trong đó: n là số điểm dữ liệu, x_i là giá trị thực và y_i là giá trị dự đoán.

MAE được biết đến là mạnh mẽ hơn đối với các yếu tố ngoại lai so với MSE. Lý do chính là trong MSE bằng cách bình phương các sai số, các giá trị ngoại lai (thường có sai số cao hơn các mẫu khác) được chú ý nhiều hơn và chiếm ưu thế trong sai số cuối cùng và tác động đến các tham số của mô hình.

2.3.5.2. Mean Squared Error - MSE

Mean Squared Error (MSE) có lẽ là số liệu phổ biến nhất được sử dụng cho các bài toán hồi quy. Về cơ bản, nó tìm thấy sai số bình phương trung bình giữa các giá trị được dự đoán và thực tế. MSE là thước đo chất lượng của một công cụ ước tính - nó luôn không âm và các giá trị càng gần 0 càng tốt [32].

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Trong đó: n là số điểm dữ liệu, y_i là giá trị quan sát và \hat{y}_i là giá trị dự đoán.

Trong phân tích hồi quy, vẽ biểu đồ là một cách tự nhiên hơn để xem xu hướng chung của toàn bộ dữ liệu. Đơn giản MSE cho bạn biết mức độ gần của đường hồi quy với một tập hợp các điểm. Nó thực hiện điều này bằng cách lấy khoảng cách từ các điểm đến đường hồi quy (những khoảng cách này là “sai số”) và bình phương chúng. Bình phương là rất quan trọng để giảm độ phức tạp với các dấu hiệu tiêu cực. Nó cũng tạo ra nhiều trọng lượng hơn cho sự khác biệt lớn hơn. MSE càng thấp thì dự báo càng tốt.

2.3.4.3. Root Mean Square Error - RMSE

Root Mean Square Error (RMSE) hoặc Root Mean Square Deviation (RMSD) là căn bậc hai của mức trung bình của các sai số bình phương. RMSE là độ lệch chuẩn của các phần dư (sai số dự đoán) [32].

Phần dư là thước đo khoảng cách từ các điểm dữ liệu đường hồi quy; RMSE là thước đo mức độ dàn trải của những phần dư này, nói cách khác, nó cho bạn biết mức độ tập trung của dữ liệu xung quanh đường phù hợp nhất.

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{N}}$$

RMSD = root-mean-square deviation

i = variable *i*

N = number of non-missing data points

x_i = actual observations time series

\hat{x}_i = estimated time series

Ảnh hưởng của mỗi lỗi đối với RMSE tỷ lệ với kích thước của lỗi bình phương; do đó các sai số lớn hơn có ảnh hưởng lớn đến RMSE một cách không cân xứng. Do đó, RMSE nhạy cảm với các yếu tố ngoại lai. Sai số bình phương trung bình gốc thường được sử dụng trong khí hậu học, dự báo và phân tích hồi quy để xác minh kết quả thực nghiệm. RMSD thấp hơn sẽ tốt hơn RMSD cao hơn.

2.3.5.4. Root Mean Square Percentage Error - RMPSE

RMSPE dùng để xác định lỗi phần trăm bình phương trung bình gốc, được tính theo công thức [33]:

$$RMPSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{y_i} \right)^2}$$

Chỉ số RMPSE thấp hơn sẽ tốt hơn RMPSE cao hơn.

2.3.5.5. R Squared - R2

R squared phản ánh mức độ phù hợp của mô hình, được tính bằng công thức [34]:

$$R^2 = 1 - \frac{ESS}{TSS}$$

Trong đó:

- ESS là viết tắt của Residual Sum of Squares, tức là tổng các độ lệch bình phương của phần dư.
- TSS là viết tắt của Total Sum of Squares, tức là tổng độ lệch bình phương của toàn bộ các nhân tố nghiên cứu.

Chỉ số R squared càng cao thì mô hình càng chính xác.

2.4. Các nghiên cứu liên quan

2.4.1. Sales Forecasting for Retail Chains (Ankur Jain, Manghat Nitish Menon, Saurabh Chandra, 2015)

Bài báo “Sales Forecasting for Retail Chains” của nhóm tác giả Ankur Jain, Manghat Nitish Menon, Saurabh Chandra nghiên cứu về một số cách khai thác dữ liệu để dự báo doanh số bán hàng trong lĩnh vực bán lẻ. Trong bài báo, nhóm tác giả đã sử dụng dữ liệu của Rossmann - chuỗi cửa hàng thuốc lớn thứ hai của Đức với 3.000 cửa hàng tại

Châu Âu để xây dựng mô hình dự đoán doanh số bán lẻ dựa trên ba thuật toán XGBoost, Linear Regression, Random Forest. Nhóm tác giả sử dụng công cụ như là Google Trend để phát hiện xu hướng sau đó tiến hành đánh giá mức độ quan trọng của các biến để lựa chọn biến xây dựng mô hình, cuối cùng dùng chỉ số RMPSE để đánh giá và so sánh hiệu quả của các mô hình. Từ kết quả nghiên cứu, nhóm tác giả cũng đưa ra các kiến nghị cho Rossman để cải tiến kế hoạch kinh doanh nhằm tăng doanh thu.

Mục tiêu của bài nghiên cứu là xây dựng mô hình dự đoán doanh thu bán lẻ bằng các thuật toán XGBoost, Linear Regression, Random Forest và so sánh kết quả của từng mô hình. Từ đó rút ra kết luận về mô hình hoạt động tốt nhất, đề xuất một số kiến nghị cho Rossman để cải thiện năng suất và tăng doanh thu.

Dữ liệu của bài nghiên cứu là tập dữ liệu Rossmann Store Sales, được lấy từ trang web Kaggle với 2 tập dữ liệu được tổng hợp thành một khuôn dữ liệu có tổng cộng 15 biến. Tập dữ liệu Rossmann chứa thông tin về 1115 cửa hàng từ ngày 1 tháng 1 năm 2013 đến ngày 31 tháng 7 năm 2015 (942 ngày). Tổng cộng có 1017209 dòng dữ liệu.

Có ba phương pháp được sử dụng trong bài nghiên cứu cụ thể là: thuật toán chuỗi thời gian của Microsoft (Microsoft Time Series Algorithm) để phát hiện xu hướng; Mô hình để dự đoán bán hàng: XGboost, Linear Regression, Random Forest; Chỉ số đánh giá RMPSE.

Kết quả nghiên cứu cho thấy XGBoost - thuật toán tăng cường độ dốc được cải tiến đã được quan sát - là thuật toán xây dựng được mô hình hoạt động tốt nhất trong các mô hình mà tác giả thực hiện. Mô hình Random Forest Regression đứng thứ nhì về mức độ hiệu quả, cuối cùng là Linear Regression.

Trong suốt bài nghiên cứu tác giả chỉ sử dụng chỉ số RMPSE để đánh giá và so sánh mà bỏ qua một số chỉ số quan trọng khác như R-Squared, MSE, RMSE,... bên cạnh đó tác giả đã bỏ đi một số biến, điều này có thể dẫn đến việc lược nhảm biến quan trọng, bỏ lỡ cơ hội xây dựng được mô hình tối ưu hơn.

2.4.2. Walmart Sales Prediction Based on Decision Tree, Random Forest, and K Neighbors Regressor (Bo Yao, 2022)

Dự báo bán hàng là một hướng nghiên cứu rất quan trọng trong lĩnh vực kinh doanh và học thuật, và các phương pháp dự báo bán hàng cũng ngày càng đa dạng và phổ biến, chẳng hạn như mô hình chuỗi thời gian, mô hình học máy và mô hình mạng nơ-ron nhân tạo. Bài báo này sử dụng ba mô hình học máy chính là Decision Tree Regressor, Random Forest Regressor và K Neighbors Regressor để dự đoán bán hàng trên tập dữ liệu Walmart Recruiting - Store Sales kết hợp sử dụng các chỉ số hệ số tương quan, sai số tuyệt đối trung bình và sai số bình phương trung bình để đánh giá kết quả dự đoán của ba mô hình.

Mục tiêu của bài nghiên cứu là so sánh kết quả dự đoán của các mô hình, tìm ra mô hình nào tốt nhất và chính xác trên dữ liệu bán hàng, đồng thời đưa ra một số gợi ý về dự đoán doanh số các chuỗi cửa hàng bán lẻ của Walmart, từ đó có những biện pháp nâng cao khả năng hoạt động của chính cửa hàng và cải thiện khả năng luân chuyển hàng hóa. Ngoài ra, bài nghiên cứu này cũng cung cấp một khung tham khảo bao gồm các mô hình và các phương pháp đánh giá các dự đoán mô hình cho các nghiên cứu dự báo doanh số khác.

Kết quả nghiên cứu cho thấy hiệu quả dự đoán của Random Forest Regressor tốt nhất trong ba mô hình trên, Decision Tree Regressor đứng thứ hai và K Neighbors Regressor hoạt động kém nhất. Lý do cho hiệu ứng kém của K Neighbors Regressor có thể là các chỉ số định lượng khoảng cách được sử dụng không phù hợp. Một số thuộc tính có ảnh hưởng quan trọng để dự đoán doanh số hàng tuần như Temperature, CPI, Unemployment, Fuel_Price,.. Tuy nhiên, bài nghiên cứu vẫn còn tồn tại một số hạn chế như sau:

Đầu tiên là các chỉ số đánh giá của mô hình. Mỗi mô hình chỉ sử dụng một chỉ báo đánh giá khi xây dựng. Trong tương lai, nghiên cứu có thể sử dụng các chỉ số đánh giá khác và so sánh các chỉ số đánh giá phù hợp với từng mô hình.

Thứ hai là cài đặt tham số của mô hình "n_Estimators" được thiết lập trong mô hình Random Forest trong quá trình tập huấn chỉ bằng 20. Trong tương lai có thể sử dụng GridSearchCV để thử nghiệm và chọn ra giá trị tham số tốt hơn vì việc thiết lập các thông số khác nhau có thể nhận được kết quả dự báo bán hàng tốt hơn.

Cuối cùng, về mặt lựa chọn mô hình, ba mô hình được chọn trong bài báo này là các mô hình học máy cổ điển và mô hình hồi quy tuyến tính tổng quát chứ chưa sử dụng các mô hình học máy chuyên sâu khác. Trong tương lai, có thể sử dụng mô hình mạng nơ-ron nhân tạo để có thể được so sánh với các mô hình này cùng một lúc để tìm ra mô hình tốt nhất để dự đoán doanh số.

CHƯƠNG 3. MÔ HÌNH DỰ ĐOÁN DOANH THU

Tóm tắt chương 3: Trước khi tiến hành chạy mô hình thực nghiệm, nhóm tiến hành đưa ra quy trình phân tích và xây dựng mô hình bao gồm các 6 bước là xác định mục tiêu phân tích của đề tài nghiên cứu, thu thập dữ liệu, phân tích và khám phá dữ liệu, tiền xử lý dữ liệu, đánh giá mô hình và kết luận. Cuối chương, nhóm đã đưa ra mô hình nghiên cứu đề xuất cho bài đồ án.

3.1. Quy trình phân tích và xây dựng mô hình

3.1.1. Xác định mục tiêu phân tích

Hiện tại, Rossman điều hành hơn 3000 cửa hàng thuốc ở Châu Âu. Trong đó, những người quản lý các cửa hàng của Rossman sẽ được giao nhiệm vụ dự đoán doanh số hàng ngày trước tối đa 6 tuần. Doanh số bán hàng của cửa hàng hiện đang bị ảnh hưởng bởi nhiều yếu tố như khuyến mãi, đối thủ cạnh tranh, ngày lễ của trường học và tiểu bang, tính thời vụ và địa phương. Việc dự đoán doanh số của mỗi nhà quản lý trên các trường hợp riêng biệt có thể cho ra kết quả khác nhau. Mục tiêu chính của chúng ta là xác định các yếu tố ảnh hưởng chính và xây dựng mô hình để dự đoán hiệu quả doanh số bán hàng của cửa hàng trong tương lai.

3.1.2. Thu thập dữ liệu

Hai bộ dữ liệu được cung cấp: một bộ chứa dữ liệu cửa hàng và một bộ chứa dữ liệu bán hàng lịch sử của 1115 cửa hàng thuốc Rossman từ tháng 1 năm 2013 đến tháng 7 năm 2015. Bộ dữ liệu sau khi hợp nhất bao gồm 18 cột và 1017209 dòng. Mô tả các biến của dữ liệu:

- Store: ID của mỗi cửa hàng;
- DayOfWeek: Ngày thứ bao nhiêu trong tuần (thứ 2 đến chủ nhật), thông thường các cửa hàng đóng cửa vào thứ bảy và chủ nhật;
- Date: Thời gian diễn ra hoạt động của cửa hàng;
- Sales: Doanh thu cho bất kỳ ngày nào (Giá trị chúng ta cần dự đoán);
- Customers: Số lượng khách hàng vào một ngày;

- Open: Chỉ báo xem cửa hàng có mở hay không (0: đóng cửa, 1: mở cửa);
- Promo: Cho biết cửa hàng có khuyến mãi vào ngày đó không (0: không có, 1: có);
- StateHoliday: Cho biết ngày lễ của tiểu bang. Trừ một vài ngoại lệ, thông thường các cửa hàng đều đóng cửa vào các ngày lễ của tiểu bang (a: nghỉ lễ, b: lễ Phục sinh, c: Giáng sinh, 0: không);
- SchoolHoliday: Cho biết liệu cửa hàng có bị ảnh hưởng bởi việc đóng cửa, nghỉ lễ của trường công lập không (0: không, 1: có);
- StoreType: Chỉ ra 4 mô hình cửa hàng khác nhau gồm a, b, c, d. Các loại cửa hàng khác nhau thì bán khác sản phẩm;
- Assortment: Mô tả cấp độ phân loại (a: cơ bản, b: bổ sung, c: mở rộng). Cho biết sự đa dạng trong các mặt hàng của cửa hàng;
- CompetitionDistance: Khoảng cách đến đối thủ cạnh tranh gần nhất (tính bằng mét);
- CompetitionOpenSinceMonth: Tháng mà đối thủ cạnh tranh gần nhất được mở
- CompetitionOpenSinceYear: Năm mà đối thủ cạnh tranh gần nhất được mở;
- Promo2: Chương trình khuyến mãi liên tục, liên tiếp cho một số cửa hàng (1: cửa hàng đang tiến hành khuyến mãi, 0: cửa hàng không tiến hành khuyến mãi);
- Promo2SinceWeek: Tuần khi cửa hàng bắt đầu tham gia Promo2;
- Promo2SinceYear: Năm khi cửa hàng bắt đầu tham gia Promo2;
- PromoInterval: Cho biết Promo2 chạy trong những tháng nhất định nào trong năm (khoảng thời gian liên tiếp promo2 được bắt đầu lại).

3.1.3. Phân tích khám phá dữ liệu

Trước tiên, chúng ta cần hiểu được bộ dữ liệu và đánh giá về sự đầy đủ và hữu ích của dữ liệu. Tại bước này, chúng ta sẽ sử dụng các phương pháp phân tích và trực quan hóa dữ liệu, cụ thể gồm những công việc sau:

- Kiểm tra giá trị thiếu của dữ liệu;
- Sơ đồ cấu trúc cơ bản về sự phân bố dữ liệu;

- Kiểm tra giá trị dị thường và ngoại lệ;
- Xác định các biến quan trọng có ảnh hưởng đến cột Sales và cách chúng ảnh hưởng lên nó.

3.1.4. Tiền xử lý dữ liệu

Sau khi phân tích và khám phá dữ liệu, chúng ta sẽ tiến hành xử lý dữ liệu trước khi chạy mô hình dự đoán doanh số, cụ thể trong bài toán này bao gồm những công việc sau:

- Xử lý giá trị bị thiếu;
- Xử lý giá trị ngoại lệ;
- Chuyển đổi biến phân loại thành số (dummy và label encoder);
- Chia biến độc lập và biến phụ thuộc;
- Ước lượng và chia tập train, test phục vụ cho việc chạy mô hình;
- Dùng Standard Scaler để chuyển đổi dữ liệu.

3.1.5. Xây dựng mô hình phân tích

Khi dữ liệu đã được xử lý, chúng ta sẽ xây dựng mô hình để thực hiện dự đoán doanh số. Sau quá trình nghiên cứu, nhóm quyết định chọn 4 loại mô hình gồm: Decision Tree, Random Forest, Linear Regression và KNN. Và nhóm sẽ sử dụng các chỉ số đánh giá mô hình như MAE, MSE, RMSPE, RMSE, R^2 để xem độ chính xác của mô hình.

Tiếp theo, nhóm sẽ tiến hành điều chỉnh các tham số của mô hình để nâng cao độ chính xác của mô hình.

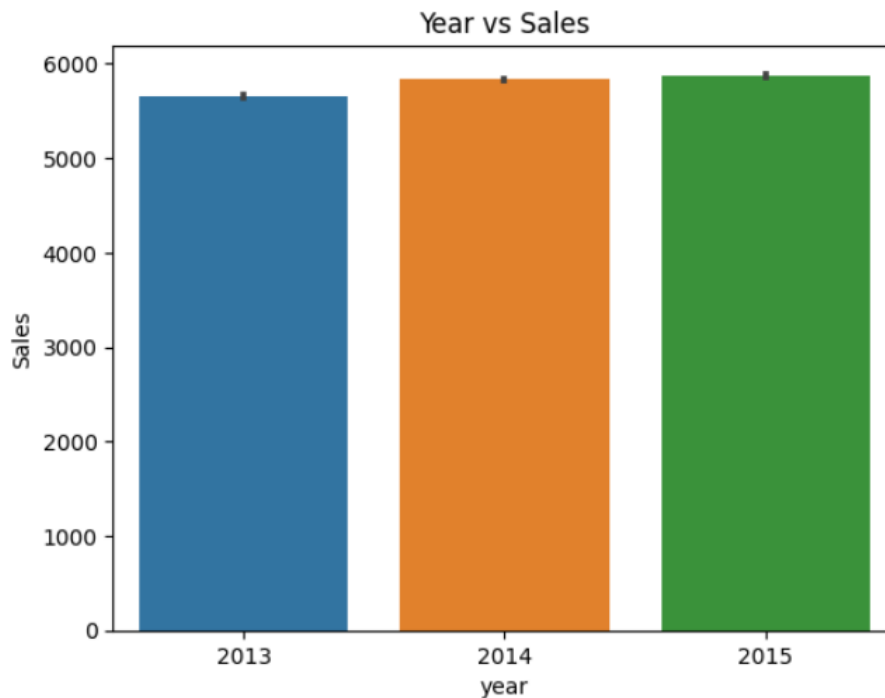
3.1.6. Đánh giá mô hình và kết luận

Nhóm sẽ tiến hành trực quan hóa để so sánh các mô hình và đưa ra mô hình phù hợp nhất cho bài toán dự đoán doanh số cho cửa hàng thuốc Rossman. Cuối cùng, nhóm sẽ đưa ra kiến nghị nhằm thúc đẩy doanh số cho cửa hàng.

3.2. Phân tích khám phá dữ liệu

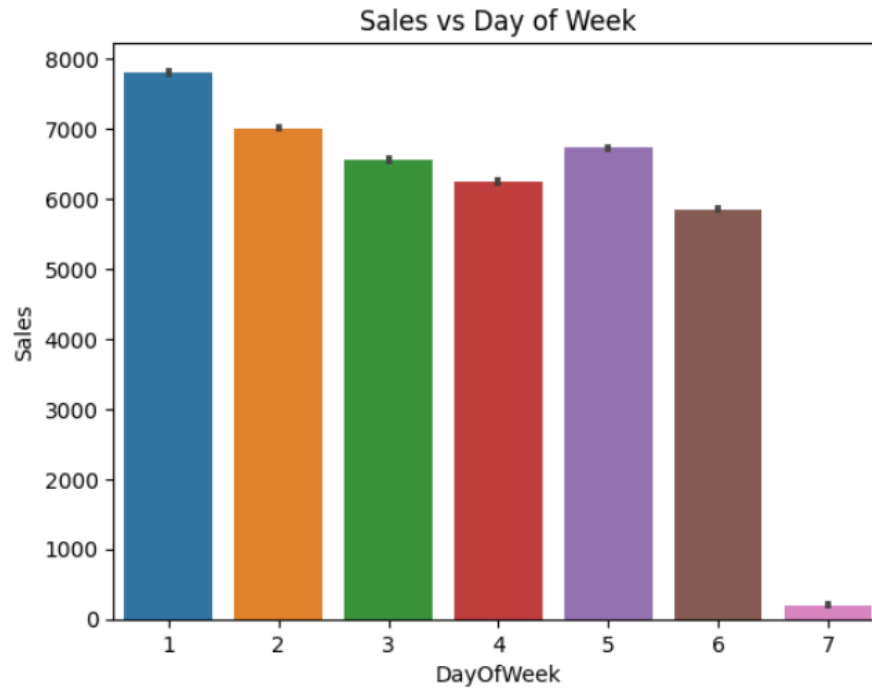
Trước khi xây dựng mô hình dự đoán cho bài toán, ta phải tiến hành xem xét và hiểu rõ về bộ dữ liệu trước. Nhóm sẽ tiến hành phân tích và khai thác dữ liệu qua các chỉ số đo, độ phân tán của biến và biểu đồ thể hiện mối tương quan giữa các biến. Qua đó, chúng ta sẽ biết được dữ liệu đã sạch hay chưa và tiến hành làm sạch dữ liệu nếu chưa đáp ứng.

3.2.1 Trực quan hóa dữ liệu



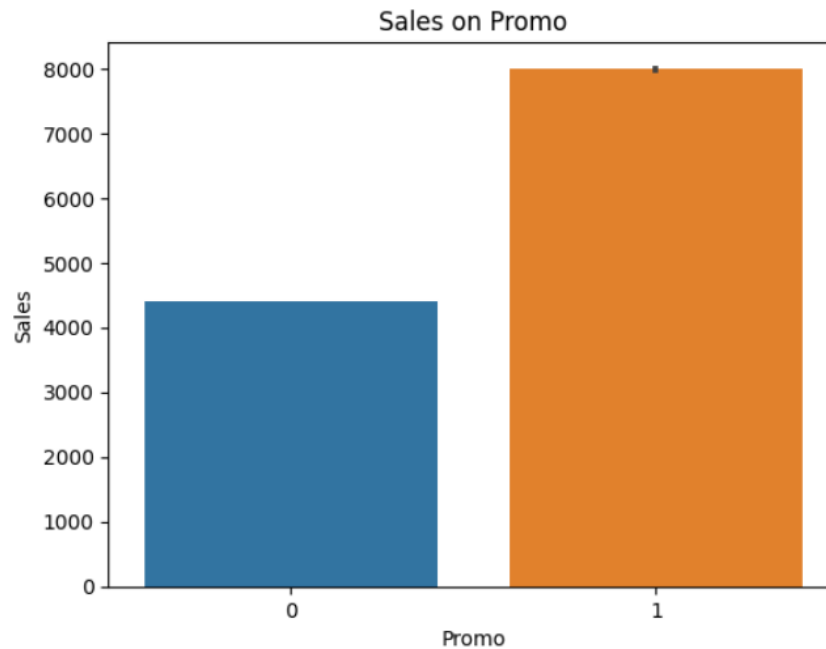
Hình 23: Biểu đồ thể hiện doanh thu theo năm

Từ biểu đồ trên ta có thể dễ dàng thấy doanh thu tăng dần theo năm, từ năm 2013 đến 2015.



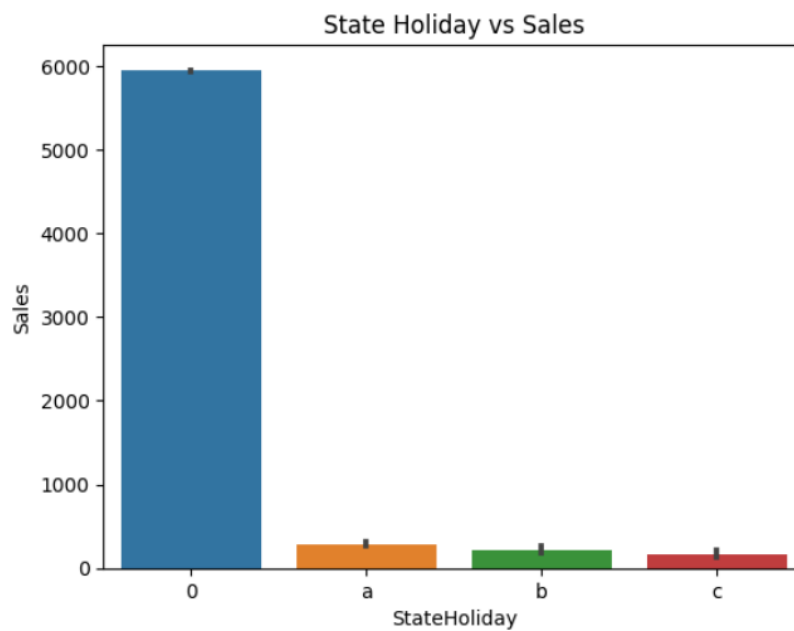
Hình 24: Biểu đồ thể hiện doanh thu theo ngày trong tuần

Từ biểu đồ ta có thể thấy rằng doanh thu vào thứ 2 là cao nhất và doanh thu vào thứ 7 là thấp nhất. Doanh thu còn lại của những ngày trong tuần xấp xỉ nhau, cao nhất là thứ 5.



Hình 25: Biểu đồ thể hiện doanh thu theo khuyến mãi

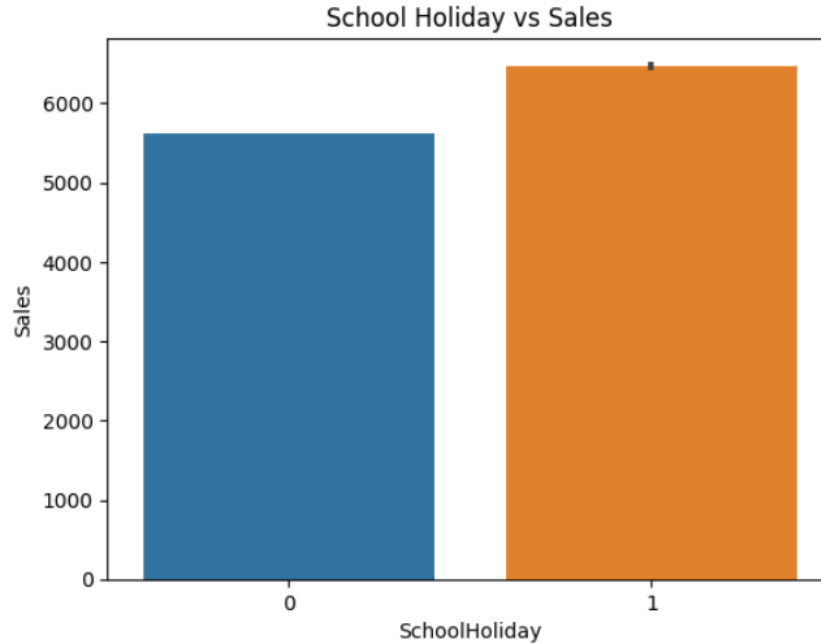
Dễ dàng thấy được doanh thu có khuyến mãi cao hơn so với không khuyến mãi, từ đó có thể rút ra rằng khách hàng bị thu hút bởi sự khuyến mãi.



Hình 26: Biểu đồ thể hiện doanh thu theo ngày lễ

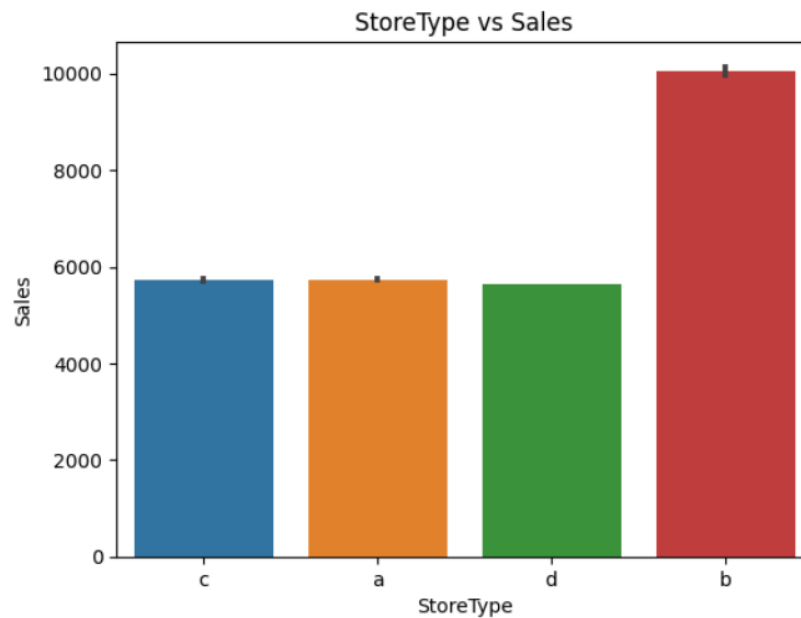
Hầu hết các cửa hàng đều đóng cửa vào các ngày lễ của Nhà nước, đó là lý do tại sao chúng ta có thể thấy rằng có rất ít doanh số bán hàng trong a, b, c. Trong đó:

- a = Ngày lễ
- b = Kỳ nghỉ lễ Phục sinh
- c = Giáng sinh
- 0 = Không có ngày nghỉ, Ngày làm việc.



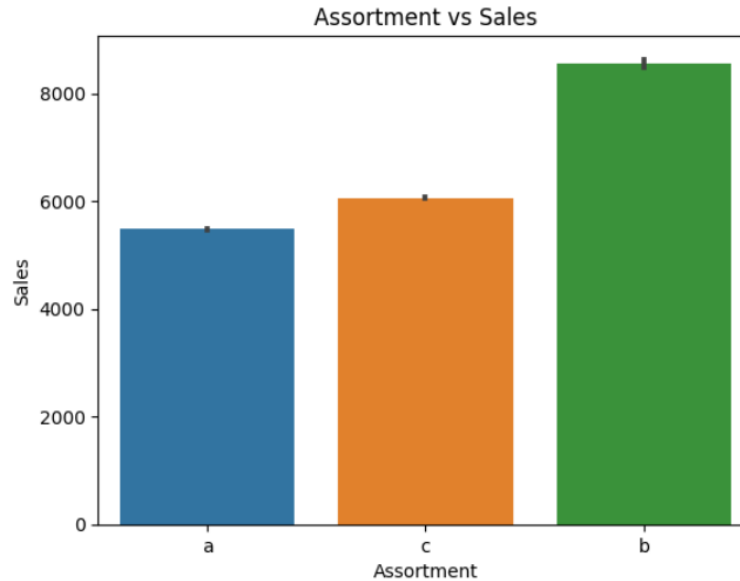
Hình 27: Biểu đồ thể hiện doanh thu theo ngày nghỉ ở trường

Từ biểu đồ có thể thấy rằng vào ngày nghỉ lễ ở trường thì doanh thu cao hơn.



Hình 28: Biểu đồ thể hiện doanh thu theo loại cửa hàng

Trong 4 loại cửa hàng a,b,c,d thì b là loại cửa hàng có doanh thu cao nhất, 3 loại cửa hàng còn lại có mức doanh thu gần như bằng nhau.



Hình 29: Biểu đồ thể hiện doanh thu theo Assortment

Có thể thấy rằng, trong 3 loại assortment, assortment b có doanh thu cao nhất.

3.2.2 Xử lý giá trị null

Kiểm tra giá trị null ở file store và file train.

```
store_data.isnull().sum()

Store                0
StoreType            0
Assortment           0
CompetitionDistance   3
CompetitionOpenSinceMonth  354
CompetitionOpenSinceYear  354
Promo2               0
Promo2SinceWeek      544
Promo2SinceYear      544
PromoInterval        544
dtype: int64
```

Hình 30: Biểu diễn giá trị null của tập dữ liệu

File store có 3 giá trị null ở cột CompetitionDistance, 354 giá trị null ở cột CompetitionOpenSinceMonth, 354 giá trị null ở cột CompetitionOpenSinceYear, Và ở mỗi cột Promo2SinceWeek, Promo2SinceYear và PromoInterval đều có 544 giá trị null.

```
train_data.isnull().sum()
```

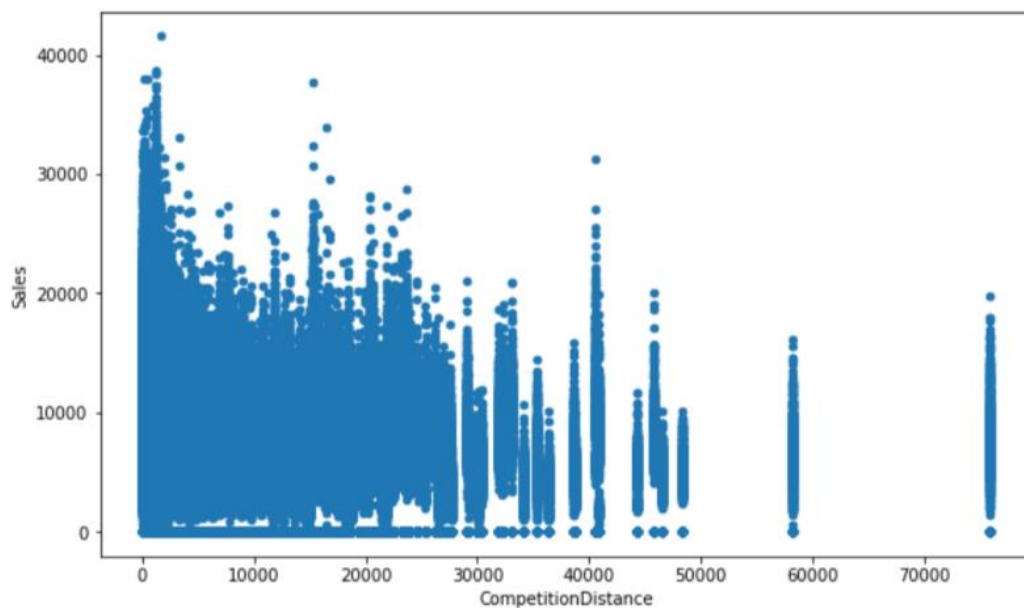
```
Store          0
DayOfWeek      0
Date           0
Sales          0
Customers      0
Open           0
Promo          0
StateHoliday   0
SchoolHoliday  0
dtype: int64
```

Hình 31: Biểu diễn giá trị null sau khi xử lý

Không có giá trị null ở tập train.

Tiến hành fill null:

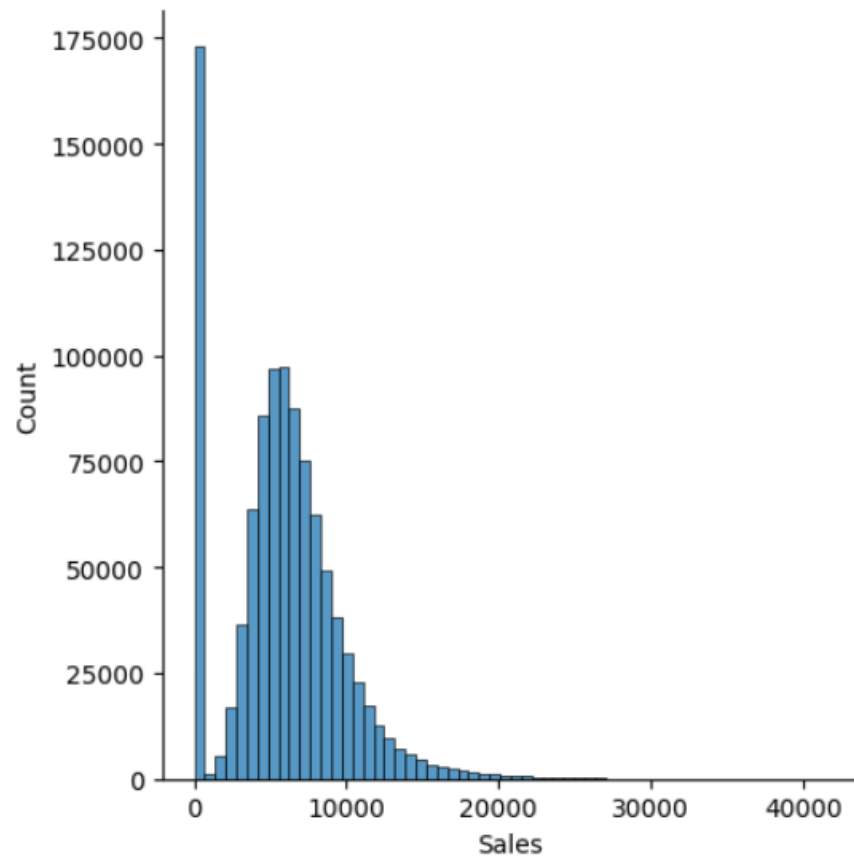
- Giá trị null ở những cột Promo2SinceWeek, Promo2SinceYear, PromoInterval được fill null bằng 0.
- Giá trị null ở cột CompetitionDistance được fill null bằng giá trị trung bình.
- Giá trị null ở cột CompetitionOpenSinceMonth, CompetitionOpenSinceYear được điền bằng giá trị tháng, năm hiện tại.



Hình 32: Biểu đồ thể hiện doanh thu theo CompetitionDistance

Từ biểu đồ có thể thấy rằng, càng gần cửa hàng đối thủ thì doanh số bán hàng trong các cửa hàng Rossmann càng cao.

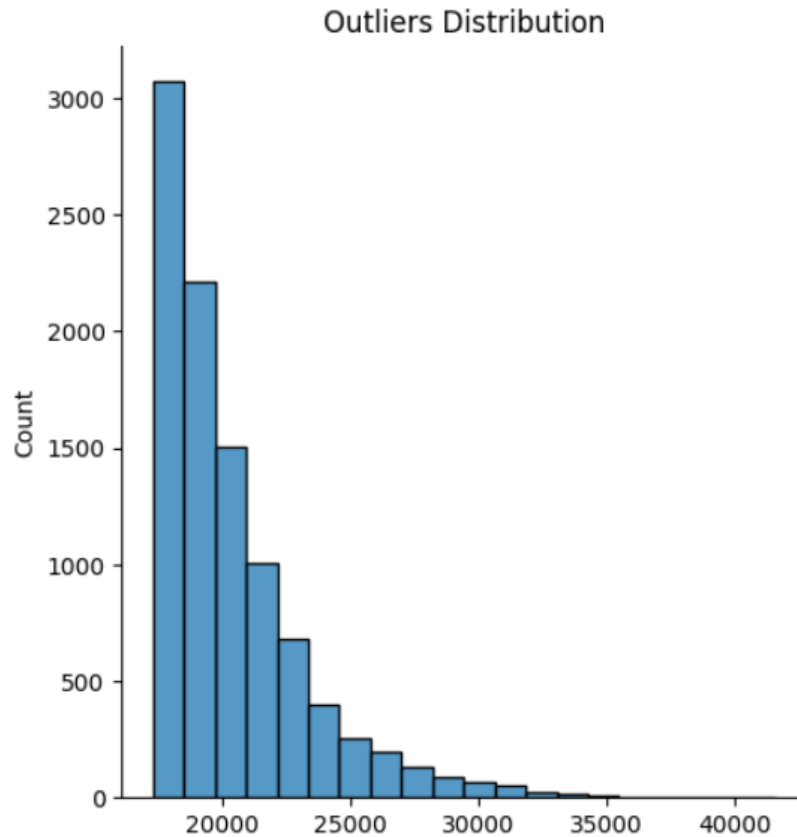
3.2.3 Xử lý giá trị ngoại lai



Hình 33: Biểu đồ histogram của biến sales

Từ biểu đồ có thể thấy có rất ít biến doanh số lớn hơn 25 000, do đó chúng có thể là ngoại lệ.

Z-Score : Nếu Z-Score của bất kỳ điểm dữ liệu nào lớn hơn 3(ngưỡng) thì đó có thể được coi là Ngoại lệ.



Hình 34: Biểu đồ phân phối của những giá trị ngoại lai

Tính phần trăm những giá trị ngoại lệ

```
# Percentage of Outliers
```

```
zero_sales = combined_data.loc[combined_data['Sales']==0]
```

```
sales_greater_than_25k = combined_data.loc[combined_data['Sales'] >
25000]
```

```
print('Length of the dataset:', len(combined_data))
```

```
print('Percentage of Zeros in dataset: %.3f%%'
```

```
%((len(zero_sales)/len(combined_data))*100))
```

```
print('Percentage of sales greater than 25k in dataset: %.3f%% '
```

```
%((len(sales_greater_than_25k)/len(combined_data))*100))
```

```
Length of the dataset: 1017209
```

```
Percentage of Zeros in dataset: 16.995%
```

```
Percentage of sales greater than 25k in dataset: 0.075%
```

Chúng ta có thể loại bỏ các điểm dữ liệu bán hàng lớn hơn 25 nghìn vì chúng chiếm tỷ lệ phần trăm rất ít trong tập dữ liệu và có thể là ngoại lệ.

```
combined_data.drop(combined_data.loc[combined_data['Sales'] >
25000].index,inplace=True)
```

Loại bỏ những ngày có doanh thu lớn hơn 25 000.

3.3 Tiền xử lý dữ liệu

3.3.1 Chuyển đổi kiểu dữ liệu phân loại sang số nhị phân

Tiến hành tách cột 'Date' thành 2 cột 'Year' và 'Month', đồng thời xóa cột 'Date'.

Kiểm tra kiểu dữ liệu của bộ data:

```
combined_data.dtypes
```

Store	int64
StoreType	object
Assortment	object
CompetitionDistance	float64
CompetitionOpenSinceMonth	float64
CompetitionOpenSinceYear	float64
Promo2	int64
Promo2SinceWeek	float64
Promo2SinceYear	float64
PromoInterval	object
DayOfWeek	int64
Sales	int64
Customers	int64
Open	int64
Promo	int64
StateHoliday	object
SchoolHoliday	int64
Year	int64
Month	int64
dtype:	object

Hình 35: Biểu diễn kiểu dữ liệu của các cột

Có 4 cột có kiểu dữ liệu object là: StoreType, Assortment, PromoInterval và StateHoliday.

Sử dụng label encoder để chuyển đổi dữ liệu cột PromoInterval và StateHoliday về dạng 0,1.

```
# encoding all categorical varibale to numeric values
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

combined_data['StateHoliday'] =
label_encoder.fit_transform(combined_data['StateHoliday'])
combined_data['PromoInterval'] =
label_encoder.fit_transform(combined_data['PromoInterval'])

combined_data.head()
```

Sử dụng dummy encoding để chuyển đổi kiểu dữ liệu có m phân nhóm sang (m-1) đặc trưng nhị phân.

Chuyển đổi cột StoreType thành 4 cột 'StoreType_a', 'StoreType_b', 'StoreType_c', 'StoreType_d'.

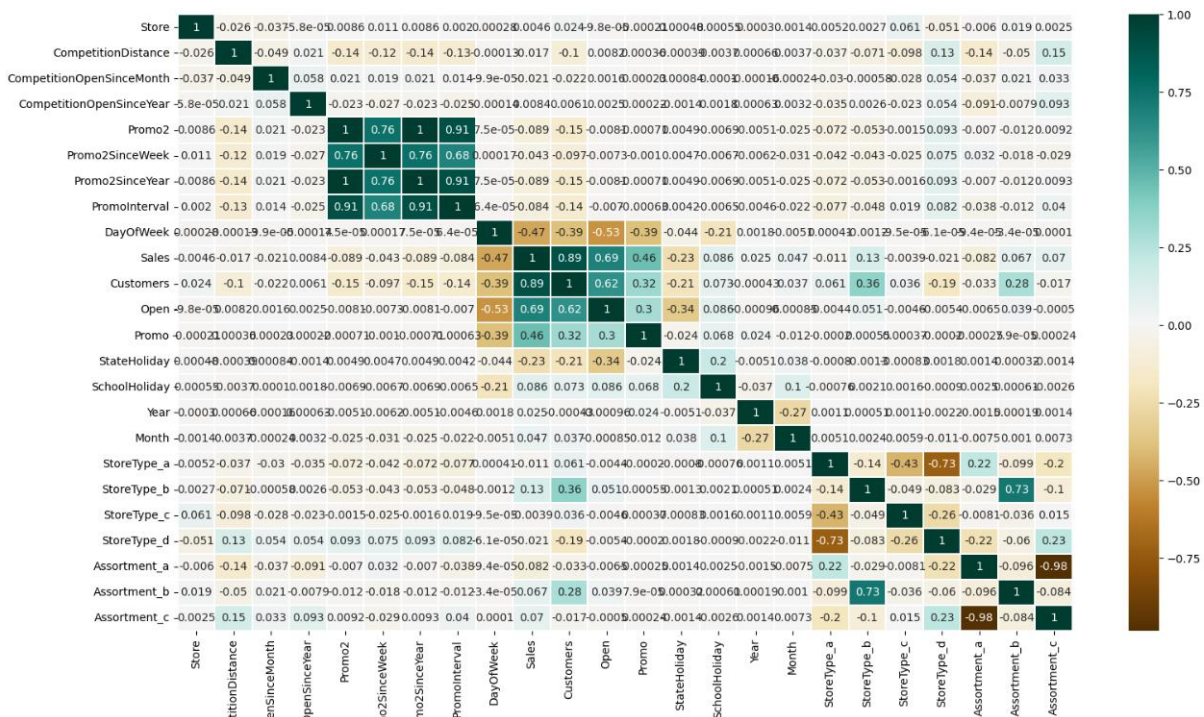
Chuyển đổi cột Assortment thành 3 cột 'Assortment_a', 'Assortment_b', 'Assortment_c'.

SchoolHoliday	Year	Month	StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
1	2015	7	0	0	1	0	1	0	0
...
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1
1	2013	1	0	0	0	1	0	0	1

Hình 36: Bảng dữ liệu sau khi dùng label encoder và dummy encoding

	Store	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval	DayOfWeek	Sales	...	SchoolHoliday	Year	Month	StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c
Store	1.000000	-0.026302	-0.017453	-0.000058	0.008997	0.011117	0.008630	0.001996	0.000283	0.004987	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487
CompetitionDistance	-0.026302	1.000000	-0.048701	0.020918	-0.140106	-0.129399	-0.140170	-0.129408	-0.000127	-0.017230	...	-0.003663	0.000664	0.003693	-0.037084	-0.070836	-0.098043	0.131818	-0.141198	-0.049927	0.190355
CompetitionOpenSinceMonth	-0.017453	-0.048701	1.000000	0.086235	0.021135	0.019334	0.021145	0.014308	-0.000099	-0.021271	...	-0.000101	-0.000161	-0.000238	-0.030165	-0.000579	-0.028430	0.033735	-0.037163	0.021000	0.003814
CompetitionOpenSinceYear	-0.000058	0.020918	0.086235	1.000000	-0.022552	-0.027398	-0.022572	-0.022576	-0.000135	0.008423	...	0.001841	0.000627	0.003232	-0.035031	0.002381	-0.028482	0.054013	-0.091339	-0.007931	0.002867
Promo2	0.008997	-0.140106	0.021135	-0.022552	1.000000	0.799134	0.799134	0.999999	0.905080	-0.008949	...	-0.006892	-0.005955	-0.023138	-0.071922	-0.053090	0.092925	-0.006870	-0.012485	0.006226	0.002624
Promo2SinceWeek	0.011117	-0.129399	0.019334	-0.027398	0.799134	1.000000	0.798948	0.678920	0.000146	-0.042964	...	-0.006679	-0.006235	-0.031097	-0.041672	-0.042694	-0.024873	0.074758	0.031813	-0.017887	-0.028624
Promo2SinceYear	0.008630	-0.017453	0.021145	-0.022572	0.999999	0.798948	1.000000	0.905143	0.000168	-0.008905	...	-0.006895	-0.005956	-0.023144	-0.071867	-0.053093	-0.001577	0.092886	-0.007021	-0.012427	0.002970
PromoInterval	0.001996	-0.129408	-0.022576	0.905080	0.678920	0.905143	1.000000	0.000094	-0.083996	...	-0.006458	-0.004554	-0.022484	-0.077239	-0.048026	0.019196	0.082006	-0.038024	-0.011639	0.048465	0.000100
DayOfWeek	0.000283	-0.000127	-0.000099	-0.000135	0.000075	0.000198	0.000094	1.000000	-0.469695	...	-0.255405	0.001838	-0.005363	0.000414	-0.001162	-0.000095	-0.000061	-0.000094	-0.000094	-0.000094	0.000100
Sales	0.004987	-0.017230	-0.000127	-0.000135	-0.000135	-0.000135	-0.000135	-0.000135	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
Customers	0.004987	-0.017230	-0.000127	-0.000135	-0.000135	-0.000135	-0.000135	-0.000135	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
Open	-0.000099	0.000146	0.000146	0.000146	0.000146	0.000146	0.000146	0.000146	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
Promo	-0.008949	-0.042964	-0.000146	-0.000146	-0.000146	-0.000146	-0.000146	-0.000146	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
StateHoliday	0.000443	-0.000393	0.000340	-0.000447	0.000487	0.000487	0.000487	0.000487	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
SchoolHoliday	0.000554	-0.003663	0.000664	-0.000101	-0.000161	-0.000238	-0.000238	-0.000238	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
Year	0.000302	-0.000664	-0.000369	0.000627	-0.000595	-0.000623	-0.000623	-0.000623	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
Month	0.001445	-0.005182	-0.002663	0.000105	-0.000579	-0.002843	-0.002843	-0.002843	1.000000	...	0.008124	0.022435	0.067228	-0.011960	0.154623	-0.009772	-0.020044	0.361952	0.066713	0.170023	0.000000
StoreType_a	-0.051445	-0.005970	-0.019345	-0.000058	-0.008997	-0.011117	-0.008630	-0.001996	-0.000283	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
StoreType_b	-0.002663	-0.070836	-0.098043	0.131818	-0.141198	-0.049927	-0.049927	-0.049927	-0.049927	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
StoreType_c	0.001051	-0.000095	-0.000061	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
StoreType_d	-0.051445	-0.005970	-0.019345	-0.000058	-0.008997	-0.011117	-0.008630	-0.001996	-0.000283	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
Assortment_a	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
Assortment_b	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867
Assortment_c	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	-0.000094	...	0.000554	0.000302	0.001445	0.005182	0.002663	0.001051	-0.051445	-0.005970	0.019345	0.002487	0.002867

Hình 37: Bảng correlation thể hiện chỉ số tương quan giữa các biến



Hình 38: Heatmap thể hiện mối tương quan giữa các biến

Từ bảng correlation và heatmap có thể thấy biến Sales có mối tương quan mạnh với biến Customer, Open và Promo.

3.3.2 Xây dựng model Regression

Chia biến, chia tập train - test và loại bỏ biến tương quan mạnh: Customer, Open.

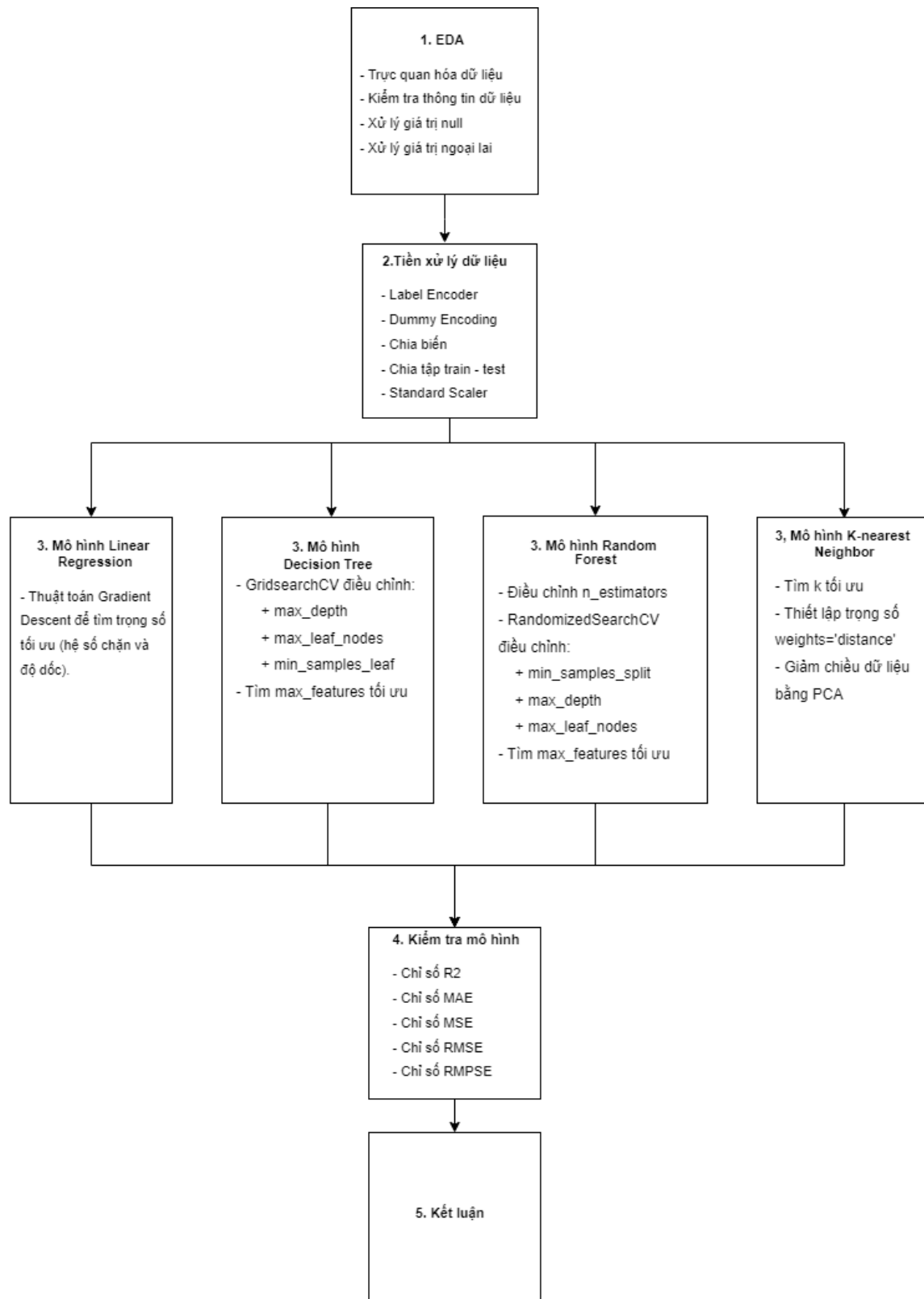
```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import
r2_score, mean_squared_error, mean_absolute_error
import math

X_train, X_test, y_train, y_test_open =
train_test_split(combined_data_open.drop(['Sales', 'Customers', 'Open'
], axis=1), combined_data_open['Sales'], test_size=0.2,
random_state=23)
```

StandardScaler sẽ chia tỷ lệ cho tập huấn luyện và kiểm tra các biến độc lập để giảm kích thước xuống các giá trị nhỏ hơn phù hợp với việc chạy mô hình.

3.4. Mô hình đề xuất



Hình 39: Mô hình nghiên cứu đề xuất

Bước 1: Khai phá dữ liệu EDA

- Kiểm tra thông tin kiểu dữ liệu và chuyển kiểu dữ liệu cho phù hợp (datetime, int,...);
- Tiến hành trực quan hóa dữ liệu;
- Xử lý giá trị null;
- Xử lý giá trị ngoại lệ cột Sales.

Bước 2: Tiền xử lý dữ liệu

- Sử dụng Label Encoder và Dummy encoding: chuyển kiểu dữ liệu phân loại sang kiểu dữ liệu số;
- Chia biến;
- Chia tập train - test với test_size=0.2, random_state=23;
- Standard Scaler để quy đổi tỷ lệ của các giá trị khác nhau để so sánh.

Bước 3: Chạy mô hình, chạy song song 4 mô hình Linear Regression, Decision Tree, Random Forest và K-Nearest Neighbor.

- Linear: Sử dụng thuật toán Gradient descent để tìm trọng số tối ưu: hệ số chặn và độ dốc.
- Decision Tree
 - + GridsearchCV điều chỉnh thông số cho mô hình: max depth, max leaf nodes, min samples leaf;
 - + Điều chỉnh chiều dữ liệu bằng max_features.
- Random Forest
 - + Điều chỉnh n_estimators;
 - + RandomizedSearchCV điều chỉnh thông số cho mô hình: min_samples_split, max_depth, max leaf nodes;
 - + Điều chỉnh chiều dữ liệu bằng max_features.
- K-Neighbors Nearest
 - + Thực hiện tìm k tối ưu để nâng cao độ chính xác cho mô hình;
 - + Thiết lập trọng số weights='distance' vào mô hình;
 - + Dùng PCA giảm chiều dữ liệu.

Bước 4: Kiểm tra mô hình

Sử dụng 5 chỉ số để kiểm tra kết quả của 4 mô hình:

- Chỉ số R^2
- Chỉ số MAE
- Chỉ số MSE
- Chỉ số RMSE
- Chỉ số RMPSE

Bước 5: Kết luận

Dựa vào những chỉ số đã tính ở trên, tiến hành so sánh những chỉ số đó để tìm ra mô hình tối ưu.

CHƯƠNG 4. KẾT QUẢ THỰC NGHIỆM VÀ ĐỀ XUẤT

Tóm tắt chương 4: Chương 4 sẽ trình bày kết quả sau khi chạy thực nghiệm các mô hình KNN, Linear Regression, Decision Tree và Random Forest. Từ đó, tiến hành tổng quan và thảo luận về hiệu quả của 4 mô hình, xác định đâu là mô hình cho kết quả cao nhất, đâu là mô hình cho kết quả thấp nhất để đưa ra mô hình đề xuất cho bài toán.

4.1. Kết quả nghiên cứu

4.1.1. KNN

Để tiến hành đào tạo được mô hình tối ưu nhất, nhóm đã xuất phát từ mô hình thô sau:

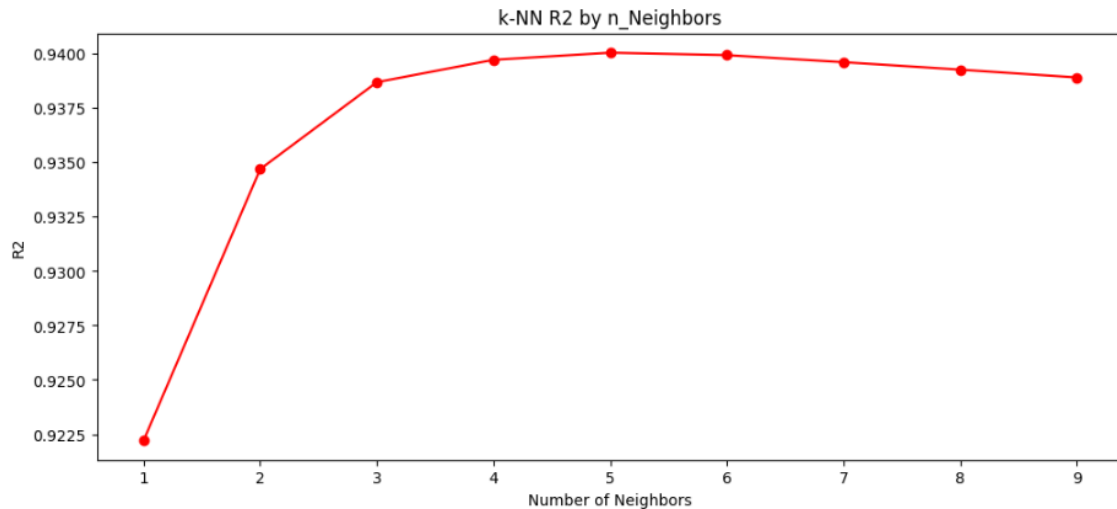
```
KNN_original = KNeighborsRegressor(n_neighbors=5, algorithm='auto',  
leaf_size=30, p=2, metric='minkowski')
```

Các thông số algorithm, leaf_size được đưa vào nhằm mục đích tăng tốc độ tính toán khoảng cách của thuật toán. Vì vậy nhóm sẽ không điều chỉnh những thông số này. Bên cạnh đó, nhóm nhận thấy thông số metric và p cũng không ảnh hưởng quá nhiều đến sự hiệu quả của mô hình nên nhóm cũng sẽ không điều chỉnh chúng.

Sau đây nhóm sẽ tiến hành tìm kiếm k tối ưu, thử đánh trọng số và thực hiện thay đổi chiều dữ liệu. Từ đó, tìm được mô hình hoạt động tốt nhất.

a) Tiến hành tìm k tối ưu

Nhóm thực hiện đào tạo mô hình với k chạy từ 1-9 đồng thời tiến hành đo chỉ số R2 để so sánh hiệu quả. Kết quả được trực quan hóa thành biểu đồ đường để tiện cho việc so sánh:



Hình 40: So sánh chỉ số R2 của mô hình KNN khi thay đổi k

Từ biểu đồ ta thấy, mô hình đạt tối ưu nhất tại k=5 và có xu hướng giảm sau đó. Vì vậy nhóm quyết định chọn k=5 để tiếp tục điều chỉnh mô hình.

Sau đây là kết quả đánh giá của mô hình KNN với k=5

```
r2_score in testing data: 0.9307036485422675
RMPSE in testing data: 0.2317600892601289
MAE: 487.238095238095298
MSE: 1,149,970.281460895435885
RMSE: 1072.3666730465357
```

b) Tiến hành đánh trọng số

```
KNN_original = KNeighborsRegressor(n_neighbors=5,
algorithm='auto', leaf_size=30, p=2, metric='minkowski',
weights='distance')
```

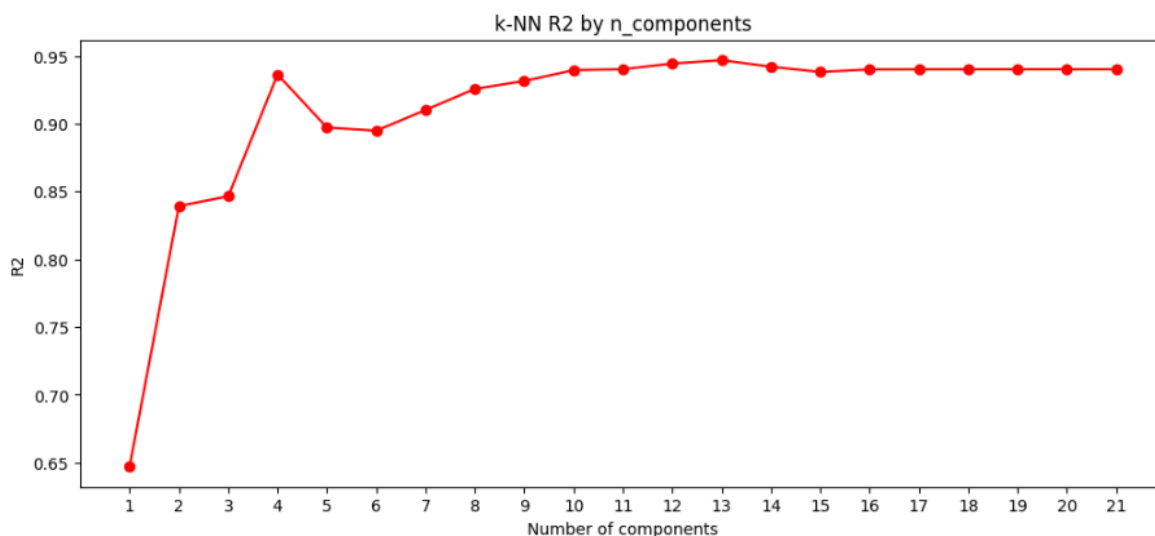
Kết quả đánh giá của mô hình sau khi được đánh trọng số là:

```
r2_score in testing data: 0.9400508510439197
RMPSE in testing data: 0.23253306852736833
MAE: 460.885002978934779
MSE: 994,853.816227463656105
RMSE: 997.4235891673425
```

Từ kết quả, ta có thể thấy mô hình sau khi đánh trọng số có R^2 tăng từ 0.9307 lên 0.9401 tuy nhiên chỉ số RMPSE lại tăng từ 0.2318 lên 0.2325, còn các chỉ số như MAE, MSE, RMSE thì có chuyển biến tích cực hơn. Tổng quan mà nói, mô hình sau khi đánh trọng số vẫn là hoạt động hiệu quả hơn. Vì vậy nhóm quyết định giữ mô hình này để tiếp tục điều chỉnh.

c) Thực hiện thay đổi chiều dữ liệu.

Để tiến hành thay đổi chiều dữ liệu, nhóm chọn phương pháp PCA. Và cho chạy vòng lặp với $n_components$ thuộc đoạn $[1;21]$. Tiến hành đo chỉ số R^2 để so sánh. Kết quả được trực quan hóa lên biểu đồ để tiện so sánh:



Hình 41: Kết quả R^2 của mô hình khi thay đổi chiều dữ liệu

Từ biểu đồ ta thấy, kết quả R^2 đạt cao nhất tại $n_components = 13$ và có xu hướng bão hòa sau đó. Kết quả đánh giá của mô hình đã giảm chiều xuống còn 13 là:

```
r2_score in training data: 0.9639713714600899
r2_score in testing data: 0.9467745673700014
RMPSE in testing data: 0.22318051214960927
MAE: 441.651308398020547
MSE: 883,274.002957157325000
RMSE: 939.826581320808
```


Từ kết quả đánh giá mô hình ta thấy, tất cả các chỉ số đều có xu hướng chuyển biến tích cực hơn. R2 tăng từ 0.9401 lên 0.9468, RMPSE giảm từ 0.2325 xuống còn 0.2232, MAE giảm từ 461 xuống còn 442, MSE giảm từ 994854 xuống còn 883274, RMSE giảm từ 997 còn 939.

4.1.2. Linear Regression

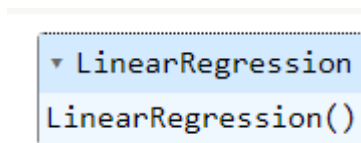
a) Mô hình chưa điều chỉnh

Đầu tiên, ta cần import thư viện cần thiết:

```
from sklearn.linear_model import LinearRegression
```

Thực hiện xây dựng và chạy mô hình:

```
saleLinear = LinearRegression()  
saleLinear.fit(X_train, y_train)
```



Hình 42: Xây dựng mô hình Linear Regression cho tập dữ liệu

Sau khi chạy mô hình, tiến hành in hệ số hồi quy của các biến tương ứng bằng phương thức `coef_` và in hệ số chặn bằng phương thức `intercept_`

```
saleLinear.coef_.round(3)
```

```
array([ 1.58980000e+01, -1.78246000e+02, -1.07431000e+02, -5.92900000e+00,  
        4.29725320e+04,  2.96792000e+02, -4.34251900e+04, -1.50337000e+02,  
       -2.52369000e+02,  1.06716800e+03, -6.03200000e+00,  2.97760000e+01,  
        1.53579000e+02,  2.48359000e+02, -5.85124680e+13, -1.57157896e+13,  
       -3.99629828e+13, -5.41461245e+13, -7.36672493e+15, -1.44644032e+15,  
       -7.35750072e+15])
```

```
saleLinear.intercept_.round(3)
```

```
6937.121
```

Thực hiện dự đoán giá trị biến Sales trên tập test dựa vào model đã xây dựng trên tập train

```

prediction_open_linear = saleLinear.predict(X_test)
prediction_closed_linear =
np.zeros(combined_data_closed.shape[0])
y_predict_linear =
np.append(prediction_open_linear,prediction_closed_linear)

```

Kết quả mô hình chưa điều chỉnh cho ra các thông số đánh giá như sau:

```

R2 in training data:  0.21721974680640443
R2 in test data:    0.7842788134831783
MAE:   984.2487711607647
MSE:   3579884.7754219854
RMSE:  1892.058343556558
RMSPE:  0.34173537896224254

```

Có thể thấy mô hình hoạt động ở mức hiệu quả trung bình với độ chính xác R bình phương là 78%.

b) Sử dụng thuật toán Gradient Descent cho mô hình Linear Regression

Sử dụng thuật toán Gradient Descent cho mô hình Linear Regression để tìm hệ số hồi quy và hệ số chặn tối ưu. Xây dựng thuật toán Gradient Descent như sau:

```

def CostFunction(x, y, w, b):
    cost = np.sum((((x.dot(w) + b) - y) ** 2) / (2*len(y)))
    return cost

def GradientDescent(x, y, w, b, learning_rate, epochs):
    cost_list = [0] * epochs
    for epoch in range(epochs):
        z = x.dot(w) + b
        loss = z - y
        weight_gradient = x.T.dot(loss) / len(y)
        bias_gradient = np.sum(loss) / len(y)
        w = w - learning_rate*weight_gradient
        b = b - learning_rate*bias_gradient
        cost = CostFunction(x, y, w, b)
        cost_list[epoch] = cost

```

```

if (epoch%(epochs/10)==0):
    print("Cost is:",cost)
return w, b, cost_list

```

Tính toán các trọng số của mô hình với learning rate = 0.556 và epochs = 900000 (số vòng lặp tối đa)

```

w, b, c= GradientDescent(X_train, y_train, np.zeros(X_train.shape[1]), 0,
0.556, epochs=900000)

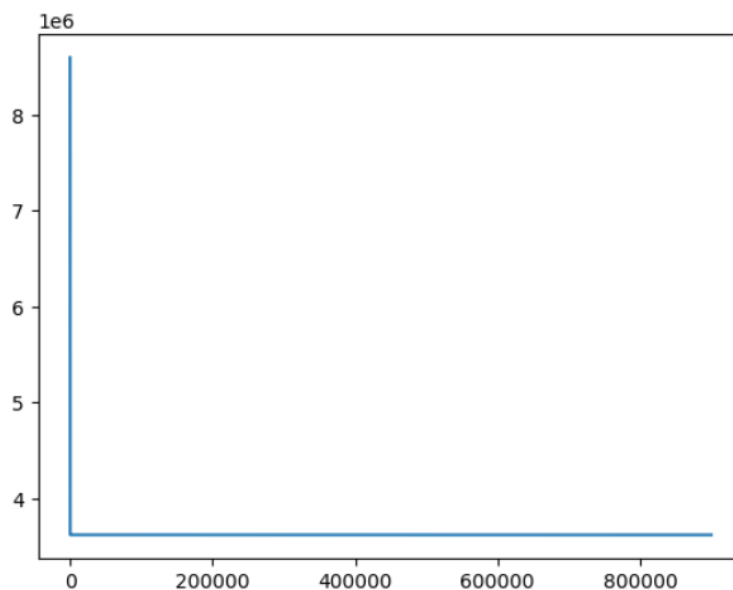
```

```

Cost is: 8593986.492160352
Cost is: 3616048.7993090674
Cost is: 3615981.8394388487
Cost is: 3615918.91830362
Cost is: 3615859.7923040404
Cost is: 3615804.2325336444
Cost is: 3615752.0238926336
Cost is: 3615702.9642551155
Cost is: 3615656.863686576
Cost is: 3615613.5437085396

```

Hình 43: Kết quả giảm dần của hàm chi phí khi chạy thuật toán Gradient Descent



Hình 44: Đồ thị thể hiện mức giảm dần của hàm chi phí

Ta thu được hệ số hồi quy và hệ số chặn có kết quả như sau:

Hệ số hồi quy:

```
array([ 1.47002157e+01, -1.74732672e+02, -1.07032819e+02, -5.36671200e+00,  
       1.14284572e+04,  3.09294501e+02, -1.18801866e+04, -1.60310286e+02,  
      -2.53568680e+02,  1.06667811e+03, -5.44722032e+00,  2.92811234e+01,  
       1.52342285e+02,  2.47851618e+02, -3.07150989e+01,  6.19889673e+02,  
      -4.62800017e+01, -1.12572277e+02, -1.91057595e+02, -2.81769640e+02,  
       2.46691340e+02])
```

Hệ số chặn:

6936.608734240404

Kết quả của mô hình Linear Regression sau khi chạy Gradient Descent:

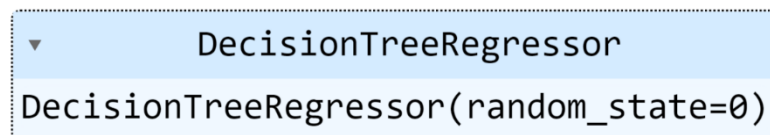
```
R2 in test data:  0.7842575321431153  
MAE:  984.2987440998945  
MSE:  3580237.9384400547  
RMSE: 1892.1516689842954  
RMSPE:  0.3417489957429222
```

Nhận thấy, sau khi chạy thuật toán, độ chính xác của mô hình Linear Regression dường như không thay đổi.

4.1.3. Decision Tree

a) Mô hình chưa điều chỉnh

Đầu tiên, nhóm thực hiện kiểm tra hiệu quả của mô hình trên bộ dữ liệu đã qua tiền xử lý với các thông số mặc định.



```
▼ DecisionTreeRegressor  
DecisionTreeRegressor(random_state=0)
```

Hình 45: Mô hình Decision Tree chưa điều chỉnh

Kết quả mô hình chưa điều chỉnh cho ra các thông số đánh giá như sau:

```
R2 in training data:  0.9639683306788563  
R2 in test data:  0.9515685306466164  
MAE:  419.85406380046703  
MSE:  803718.3671617471  
RMSE: 896.5034116843879
```

RMSPE: 0.180575137303045

Có thể thấy mô hình này hoạt động khá hiệu quả và có độ chính xác cao khi chỉ số R bình phương lên đến hơn 0.95. Nhóm sẽ thực hiện điều chỉnh tăng độ chính xác và giảm rủi ro overfitting của mô hình.

b) Điều chỉnh các thông số

Trong phần điều chỉnh thông số, nhóm tiến hành cắt tỉa Decision Tree bằng cách đặt ra các giới hạn cho cây. Để chọn ra cách cắt tỉa tốt nhất, nhóm tiến hành kiểm tra mô hình với các thông số ngẫu nhiên sau đó chọn ra khoảng giá trị phù hợp để kiểm tra kỹ lưỡng hơn.

Sau khi đi tìm các khoảng giá trị bằng cách thay các giá trị ngẫu nhiên, nhóm đưa ra được các giá trị đưa vào hàm GridSearchCV như sau:

```
'max_depth': [3000, 4000, 5000, 6000, 7000],  
'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 10, 15, 20],  
'max_leaf_nodes' : [10000, 11000, 12000, 13000, 14000, 15000]
```

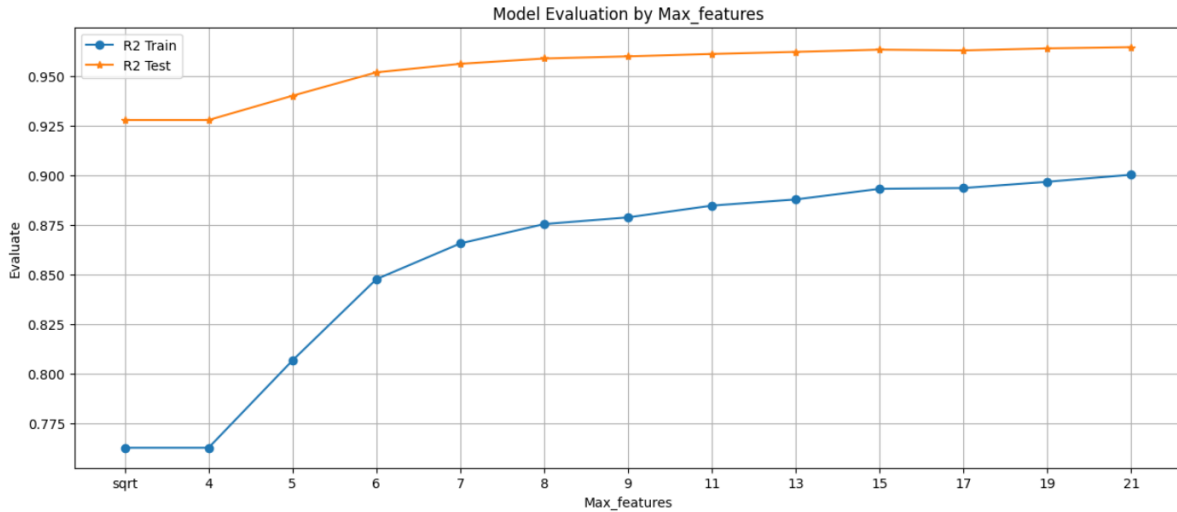
GridSearchCV được chạy với khoảng giá trị trên cùng với chỉ số đánh giá là “R2”. Kết quả chạy thu được là:

- max_depth = 3000
- min_samples_leaf = 5
- max_leaf_nodes = 15000

```
{ 'max_depth': 3000, 'max_leaf_nodes': 15000, 'min_samples_leaf': 5 }  
  
DecisionTreeRegressor(max_depth=3000, max_leaf_nodes=15000, min_samples_leaf=5,  
                      random_state=42)
```

Hình 46: Kết quả chạy GridSearchCV trên Decision Tree

Tiếp theo, nhóm tiếp tục tìm ra max_features nhập vào phù hợp thông qua biểu đồ thể hiện kết quả đo lường R2 của tập dữ liệu huấn luyện và tập dữ liệu kiểm tra. Từ biểu đồ, ta có thể thấy rằng max_feature = 15 là điểm bắt đầu bão hòa về giá trị của R2. Vì vậy, nhóm chọn max_feature = 15 để đảm bảo mức độ chính xác và hiệu quả cho mô hình.



Hình 47: Kiểm tra $\max_features$ và hiệu quả của mô hình Decision Tree

Có một điểm đáng lưu ý trong biểu đồ này là chỉ số R2 trong tập huấn luyện luôn thấp hơn tập kiểm tra. Điều này có thể giải thích thông qua sự khác biệt trong việc tách các tập huấn luyện và kiểm tra theo thuộc tính có mở cửa hay không của cửa hàng. Thêm vào đó, mô hình chưa gặp phải vấn đề overfitting hay underfitting nên kết quả này có thể chấp nhận được.

c) Mô hình đã điều chỉnh thông số

Sau khi tìm các chỉ số tối ưu cho mô hình, nhóm thực hiện chạy lại mô hình với các thông số đã được điều chỉnh. Trong đó: **random_state = 42**, **max_depth = 3000**, **max_leaf_nodes = 15000**, **min_samples_leaf = 5** và **max_features = 15**.

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3000, max_features=15, max_leaf_nodes=15000,
                      min_samples_leaf=5, random_state=42)
```

Hình 48: Mô hình Decision Tree đã điều chỉnh

Kết quả mô hình đã qua điều chỉnh các thông số như sau:

```
R2 in training data: 0.8930158622068994
R2 in test data: 0.9631723412467204
MAE: 364.9851298074085
MSE: 611153.5775139137
```

RMSE: 781.7631211012155

RMSPE: 0.11078514717434555

Các chỉ số đánh giá cao hơn chứng tỏ mô hình đã được cải thiện, chỉ số R2 cho tập kiểm tra tăng từ 0.9515 lên 0.9631, cùng với đó chỉ số RMPSE cũng giảm từ 0.1805 xuống 0.1107. Chỉ số R2 cho tập dữ liệu huấn luyện giảm từ 0.9639 xuống 0.8930 cho thấy khả năng overfitting của mô hình có giảm.

Bên cạnh đó, nhóm cũng sử dụng feature_importance của mô hình để khám phá các thuộc tính có ảnh hưởng mạnh lên việc phân tách nhánh của mô hình.

	Feature	Importance
1	CompetitionDistance	0.198299
0	Store	0.164238
9	Promo	0.144933
8	DayOfWeek	0.077947
3	CompetitionOpenSinceYear	0.074483
2	CompetitionOpenSinceMonth	0.071056
13	Month	0.058147
6	Promo2SinceYear	0.040562
5	Promo2SinceWeek	0.030060
15	StoreType_b	0.026183

Hình 49: 10 thuộc tính quan trọng nhất trong việc hình thành Decision Tree

Từ kết quả phân tích, chúng ta có thể thấy 05 thuộc tính quan trọng nhất lần lượt là: CompetitionDistance, Store, Promo, DayofWeek, CompetitionOpenSinceYear.

4.1.4. Random Forest

a) Mô hình chưa điều chỉnh

Nhóm tiến hành xây dựng mô hình Random Forest và thực hiện kiểm tra hiệu quả của mô hình trên bộ dữ liệu đã qua tiền xử lý với các thông số mặc định:

```

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
prediction_open = reg.predict(X_test)
prediction_closed = np.zeros(combined_data_closed.shape[0])
y_predict_reg =
np.append(prediction_open, prediction_closed)

```

Kết quả mô hình chưa điều chỉnh cho ra các thông số đánh giá như sau:

```

R2 in test data:  0.9652749579165507
R2 in training data:  0.9581272901136981
MAE:  356.6870206604133
MSE:  576260.7349219919
RMSE: 759.118393218075
RMSPE:  0.1255416568838386

```

Có thể thấy mô hình này hoạt động khá hiệu quả và có độ chính xác cao, bằng chứng là chỉ số R bình phương cao trên tập test lên đến 0.96527.

b) Mô hình điều chỉnh thông số

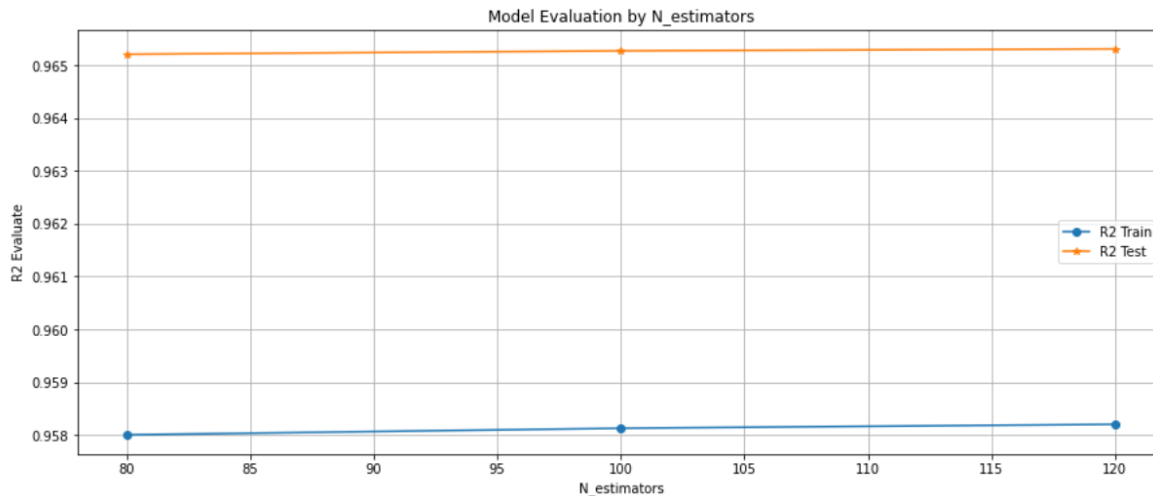
Tại đây, nhóm sẽ thực hiện điều chỉnh một số thông số, kiểm tra và so sánh hiệu quả mô hình nhằm tối ưu mô hình đạt hiệu quả nhất có thể.

Đầu tiên, nhóm thực hiện điều chỉnh `n_estimators` - chỉ số cây quyết định của mô hình với các thông số ngẫu nhiên tương ứng là 80, 100 và 120:

```

estimators = [80, 100, 120]
for i in estimators:
    regressor = RandomForestRegressor(random_state=42, n_estimators=i)

```

Hình 50: Kiểm tra $n_estimators$ trên mô hình Random Forest

Từ biểu đồ, ta có thể thấy rằng hiệu quả của mô hình có xu hướng tăng khi tăng chỉ số cây quyết định. Chỉ số cây quyết định tốt nhất trong những thông số thử nghiệm là 120 - chỉ số cao nhất với kết quả mô hình đạt 0.96531 tăng không quá đáng kể so với mô hình ban đầu. Dự đoán là sẽ tăng mạnh hơn khi tăng số cây lớn hơn nhưng vì hạn chế về tài nguyên nên nhóm sẽ chỉ dừng lại ở số cây là 120.

```
Best R2 of model with n_estimators: 120
R2 test: 0.9653113732829767
R2 train: 0.958201315840973
MAE: 356.508385267222
MSE: 575656.4233197584
RMSE: 758.7202536638641
RMSPE: 0.12452298810473188
```

Tiếp theo, nhóm tiến hành điều chỉnh các thông số về cây quyết định trong mô hình Random Forest như `max_depth`, `min_samples_split`, `max_leaf_nodes` bằng `RandomizedSearchCV`:

```
param_dist = {
    'max_depth': [4000, 3000],
    'min_samples_split': [10, 2],
    'max_leaf_nodes': [12000, None]
}
```

```

regressor = RandomForestRegressor(n_estimators=120,
random_state=42)

rand_search = RandomizedSearchCV(regressor,
                                param_distributions =
param_dist,
                                n_iter = 3,
                                n_jobs = -1,
                                verbose = 2,
                                cv = 3)

```

Kết quả chạy thu được là:

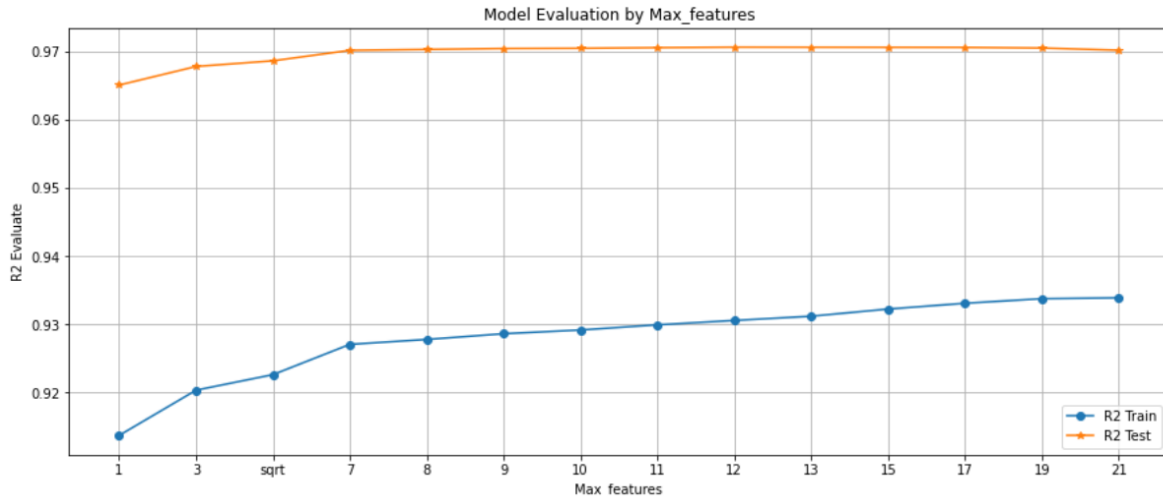
- Best estimator: RandomForestRegressor(max_depth=4000, min_samples_split=10, n_estimators=120, random_state=42)
- Best parameter: {'min_samples_split': 10, 'max_leaf_nodes': None, 'max_depth': 4000}
- R2 score: 0.9701706801461882

Cuối cùng, nhóm tiến hành điều chỉnh max_features để tìm ra giá trị max_features phù hợp của mô hình:

```

features = [1, 3, 'sqrt', 7, 8, 9, 10, 11, 12, 13, 15, 17, 19,
21]
for i in features:
    regressor = RandomForestRegressor(max_depth=4000,
random_state=42, max_leaf_nodes=None, min_samples_split=10,
n_estimators=120, max_features=i)

```



Hình 51: Kiểm tra max_features trên mô hình Random Forest

Thông qua biểu đồ thể hiện kết quả đo lường R2 của tập dữ liệu huấn luyện và tập dữ liệu kiểm tra, ta có thể thấy rằng max_features càng tăng thì R2 trên tập huấn luyện càng tăng. Với tập kiểm tra, ban đầu khi max_features tăng thì giá trị R2 cũng tăng theo, chỉ khi bắt đầu từ max_features = 12 là điểm cho kết quả R2 cao nhất thì các giá trị max_features càng về sau cho kết quả trên R2 thấp dần xuống. Vì vậy, nhóm chọn max_features = 12 cho hiệu quả ổn định và tốt nhất trên mô hình Random Forest.

Best R2 of model with max_feature: 12

R2 test: 0.9706198721824455

Với kết quả mô hình sau khi điều chỉnh thông số (max_depth=4000, random_state=42, min_samples_split=10, max_leaf_nodes=None, n_estimators=120, max_features=12):

R2 test: 0.9706198721824455

R2 train: 0.9305522470742541

MAE: 325.0529503168332

MSE: 487562.0886955109

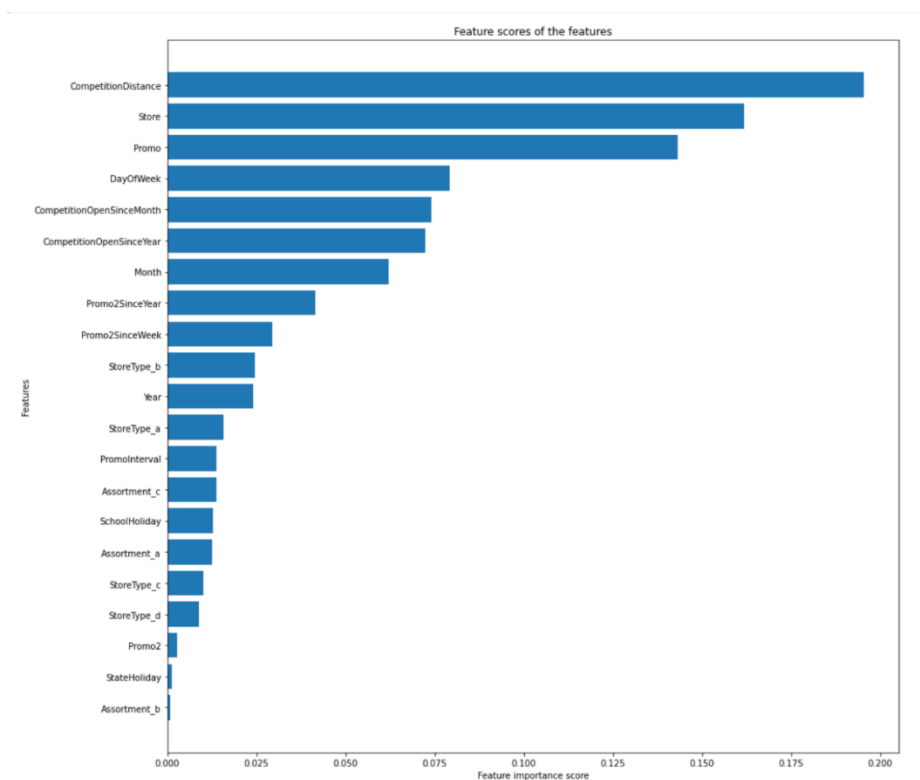
RMSE: 698.256463411196

RMSPE: 0.10782265846883857

So sánh với kết quả chạy mô hình chưa điều chỉnh, ta thấy hiệu quả của mô hình có cải thiện khi chỉ số R2 từ 0.96527 tăng lên 0.97062, các chỉ số error như MAE, MSE, RMSE, RMSPE cũng giảm đáng kể so với mô hình ban đầu, dù mức độ cải thiện lên không quá rõ rệt nhưng vẫn có giá trị tham khảo nhất định.

```
Optimal R2 score compared to raw
model: 0.005344914265894785
Optimal MAE compared to raw model: -31.634070343580106
Optimal MSE compared to raw model: -88698.64622648107
Optimal RMSE compared to raw model: -60.86192980687895
Optimal RMSPE compared to raw model: -0.01771899841500002
```

Bên cạnh đó, nhóm cũng sử dụng feature_importance của mô hình để khám phá các thuộc tính có ảnh hưởng mạnh lên hoạt động của mô hình.



Hình 52: Các thuộc tính quan trọng nhất trong hoạt động mô hình Random Forest

Từ kết quả phân tích, chúng ta có thể thấy 05 thuộc tính quan trọng nhất lần lượt là: CompetitionDistance, Store, Promo, DayOfWeek, CompetitionOpenSinceYear.

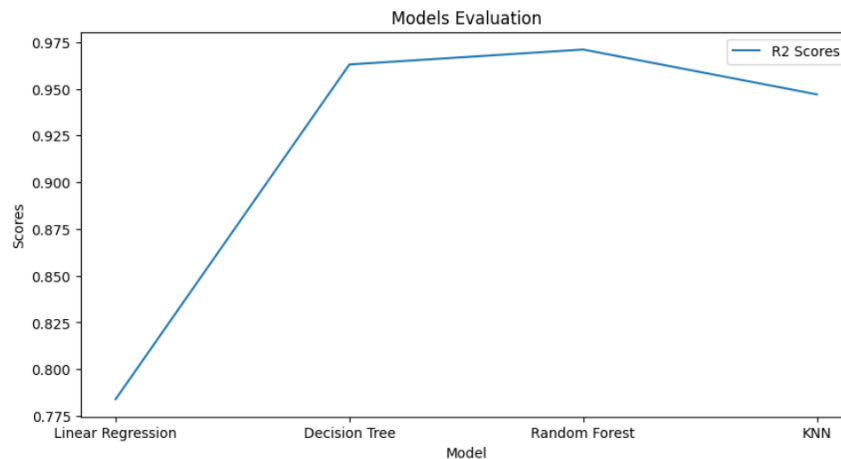
4.2. Thảo luận

4.2.1 Báo cáo kết quả

	Model1	R2	MAE	MSE	RMSE	RMSPE
0	Linear Regression	0.784	984.4	3580618	1892.2	0.34175
1	Desicion Tree	0.963	365.0	611154	781.8	0.11079
2	Random Forest	0.971	325.1	487562	698.3	0.10782
3	K-Nearest Neighbor	0.947	441.7	883274	939.8	0.22318

Hình 53: Dataframe kết quả các mô hình

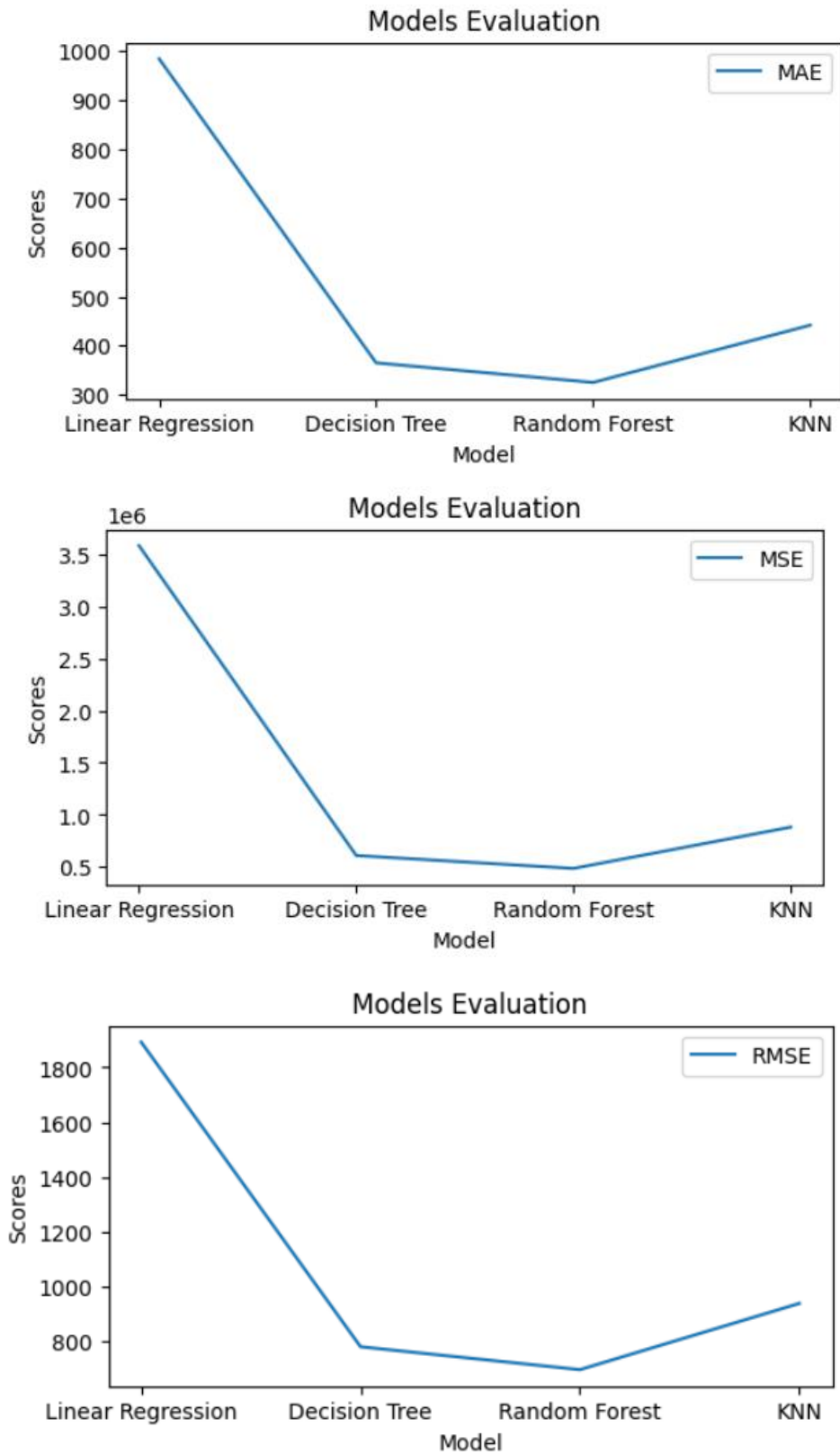
Để có một cách nhìn khái quát, nhóm đã tiến hành trực quan hóa kết quả như sau:



Hình 54: Biểu đồ thể hiện chỉ số R2

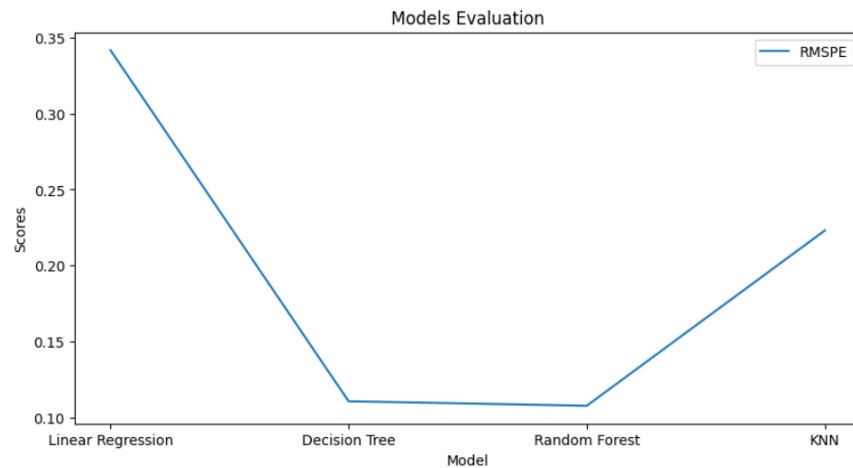
Từ biểu đồ trên có thể thấy chỉ số R2 cao nhất khi sử dụng model Random Forest (0.971) và thấp nhất khi sử dụng Linear Regression (0.784). Hai model còn lại là Decision Tree và KNN có chỉ số R2 khá cao, lần lượt là 0.963 và 0.947.

Nếu chỉ xét về chỉ số R2 thì Random Forest là mô hình tốt nhất, tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression.



Hình 55: Biểu đồ thể hiện chỉ số MAE, MSE và RMSE

Cả 3 chỉ số trên đều có xu hướng giống nhau. Cao nhất là Linear Regression, tiếp đến là KNN, Decision Tree và thấp nhất là Random Forest. Vì những chỉ số này để chỉ mức độ lỗi của mô hình nên nếu chỉ xét 3 chỉ số này thì Random Forest là mô hình tốt nhất, tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression.



Hình 56: Biểu đồ thể hiện chỉ số RMPSE

RMPSE dùng để xác định lỗi phần trăm bình phương trung bình gốc, chỉ số RMPSE càng thấp chứng tỏ mô hình càng tốt. Từ biểu đồ trên có thể thấy được Random Forest là mô hình có chỉ số RMPSE thấp nhất, tiếp đến là Decision Tree, KNN và cao nhất là Linear Regression. Từ chỉ số này cho biết rằng Random Forest là mô hình tốt nhất, tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression.

Kết luận chung: Tất cả 5 chỉ số R^2 , MAE, MSE, RMSE, RMPSE đều cho ra một kết quả mô hình giống nhau: Random Forest là mô hình tốt nhất, tiếp đến là Decision Tree, KNN và cuối cùng là Linear Regression.

4.2.2 Features quan trọng

DayOfWeek	-0.181738
Promo2SinceYear	-0.125911
Promo2	-0.125886
PromoInterval	-0.119290
Assortment_a	-0.116823
Promo2SinceWeek	-0.056701
CompetitionDistance	-0.034005
CompetitionOpenSinceMonth	-0.033747
StoreType_d	-0.026050
StoreType_a	-0.012072
StoreType_c	-0.001078
Store	0.007035
CompetitionOpenSinceYear	0.010111
StateHoliday	0.014028
Year	0.037990
SchoolHoliday	0.039528
Assortment_b	0.055543
Month	0.072109
Assortment_c	0.106033
StoreType_b	0.137502
Promo	0.373611
Sales	1.000000

Hình 57: Mức độ tương quan của features đối với biến Sales

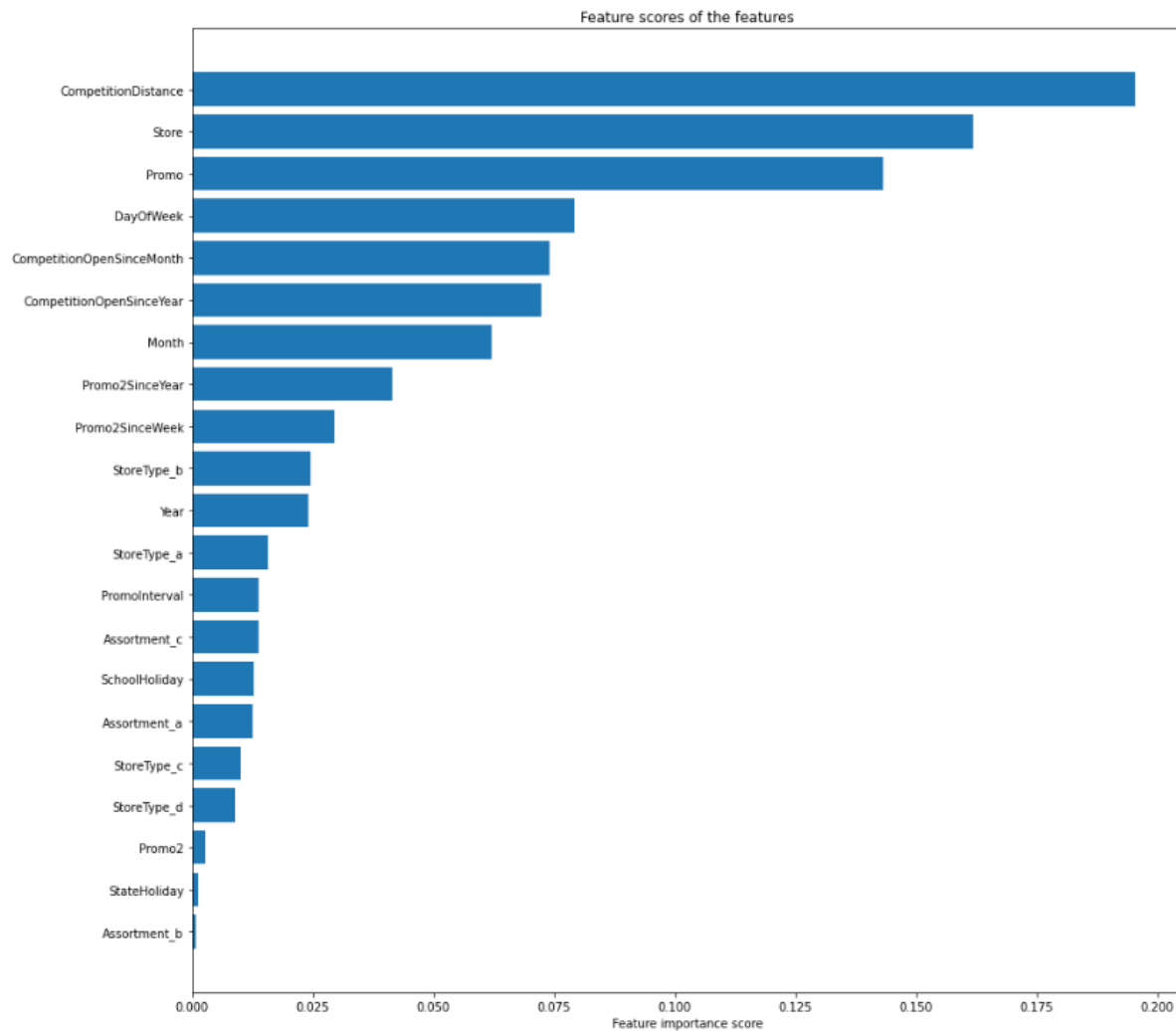
Có thể thấy rằng, 10 feature có độ tương quan lớn với biến Sales là: Promo, DayOfWeek, StoreType_b, Promo2SinceYear, Promo2, PromoInterval, Assortment_a, Assortment_c, Month, Promo2SinceWeek.

	Feature	Importance
1	CompetitionDistance	0.198299
0	Store	0.164238
9	Promo	0.144933
8	DayOfWeek	0.077947
3	CompetitionOpenSinceYear	0.074483
2	CompetitionOpenSinceMonth	0.071056
13	Month	0.058147
6	Promo2SinceYear	0.040562
5	Promo2SinceWeek	0.030060
15	StoreType_b	0.026183
18	Assortment_a	0.022100
12	Year	0.022041
14	StoreType_a	0.018030
7	PromoInterval	0.013004
11	SchoolHoliday	0.012598
17	StoreType_d	0.008711
16	StoreType_c	0.008279
20	Assortment_c	0.005264
4	Promo2	0.002446
10	StateHoliday	0.001208
19	Assortment_b	0.000411

Hình 58: Mức độ quan trọng của features trong mô hình Decision Tree

Từ bảng trên có thể thấy 10 features quan trọng nhất tác động đến hiệu quả mô hình Decision Tree là: Competition Distance, Store, Promo, Day Of Week, Competition Open

Since Year, Competition Open Since Month, Month, Promo 2 Since Year, Promo 2 Since Week, StoreType_b.



Hình 59: Mức độ quan trọng của features trong mô hình Random Forest

Từ mô hình có thể thấy 10 features quan trọng nhất tác động đến hiệu quả mô hình Random Forest là: Competition Distance, Store, Promo, Day Of Week, Competition Open Since Month, Competition Open Since Year, Month, Promo 2 Since Year, Promo 2 Since Week, StoreType_b.

Kết luận chung: Dựa vào việc mô hình Random Forest và Decision Tree cho kết quả tối ưu vượt trội, nên ta có thể rút ra 5 features quan trọng nhất tác động đến hiệu quả

mô hình là: Competition Distance, Store, Promo, Day Of Week, Competition Open Since Month.

4.3. Đề xuất mô hình áp dụng

Sau quá trình đào tạo và đánh giá từng mô hình, nhóm nhận thấy mô hình Random Forest đạt hiệu quả tốt nhất với R^2 đạt 0.971, RMSE đạt 698.3, RMPSE đạt 0.10782. Vì vậy nhà thuốc Rossman nên áp dụng mô hình Random Forest để dự đoán doanh thu các cửa hàng nhằm có những điều chỉnh thích hợp, kịp thời trên kế hoạch kinh doanh của mình.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Tóm tắt chương 5: Trong chương này, các kết luận về kết quả của đồ án sẽ được trình bày bao gồm kết quả so sánh các mô hình và các biến có ảnh hưởng đáng kể đến việc đưa ra dự đoán. Mô hình Random Forest có hiệu quả cao nhất phù hợp với kết quả của nhiều nghiên cứu khác. Cùng với đó, một số hạn chế của đồ án sẽ được chỉ ra, từ đó nhóm tác giả đưa ra một số hướng phát triển trong tương lai cho đồ án.

5.1 Kết luận

Trong đồ án này, nhóm đã thực hiện dự báo doanh thu bán hàng cho các cửa hàng thuộc thương hiệu Rossmann. Trước khi huấn luyện mô hình, tập dữ liệu đã được khai phá và xử lý làm sạch. Sau khi huấn luyện mô hình, nhóm thực hiện tính toán các chỉ số MSE, MAE, R2, RMSE. Đây là các chỉ số đánh giá thường gặp để đánh giá các thuật toán hồi quy. Bên cạnh đó, chỉ số RMPSE được thêm vào theo nghiên cứu liên quan để so sánh kết quả [1].

Năm thuộc tính của bộ dữ liệu có ảnh hưởng mạnh mẽ nhất đến doanh thu xếp từ cao đến thấp bao gồm: Competition Distance, Store, Promo, Day Of Week, Competition Open Since Month. Từ đây, các cửa hàng có thể xem xét các chiến lược để thúc đẩy doanh thu cửa hàng thông qua việc tác động đến các yếu tố ảnh hưởng.

Kết quả huấn luyện 04 mô hình là K-Nearest Neighbors, Linear Regression, Decision Tree, Random Forest cho thấy Random Forest mang lại độ chính xác cao nhất. Tiếp theo đó là các mô hình Decision Tree, K-Nearest Neighbors và Linear Regression. Điều này phù hợp với các nghiên cứu trước đây khi so sánh hiệu suất có hai trong bốn mô hình kể trên [1] [2] [35] [36]. Tuy nhiên, mô hình K-Nearest Neighbors và Decision Tree có độ chính xác chênh lệch không quá lớn và đều tương đối cao là một sự cải thiện so với nghiên cứu trước đó [37]. Thêm vào đó, mô hình Random Forest cho ra chỉ số RMPSE nhỏ hơn bài nghiên cứu trước đó cùng sử dụng bộ dữ liệu [1], điều này chứng tỏ có sự cải thiện trong các bước xây dựng mô hình.

Từ việc so sánh mức độ hiệu quả của các mô hình trên, các cửa hàng có thể xem xét mô hình Random Forest như một phương pháp học máy để áp dụng vào quá trình dự đoán doanh số nhằm đưa ra dự đoán chính xác hơn.

5.2 Hạn chế

Hạn chế về thời gian và tài nguyên máy nên việc thử nghiệm và tối ưu các mô hình chưa được toàn diện. Trong đó, thuật toán Random Forest và KNN mất 2 đến 3 tiếng trong việc huấn luyện mô hình và khi chạy mô hình điều chỉnh các tham số cho hai mô hình này, nhóm không nhận được kết quả vì tài nguyên đã hết. Ngoài ra, khi chạy thuật toán Gradient Descent cải thiện độ chính xác Linear Regression, thời gian chạy rất lâu (lâu nhất đến 10 tiếng), vì thế nhóm gặp trục trặc trong việc chạy để đưa ra hiệu suất cao hơn cho mô hình này.

Nền tảng hỗ trợ việc chạy mô hình chưa phù hợp. Cụ thể, nhóm sử dụng nền tảng Google Colab và Ubuntun nhưng vẫn không quá trình chạy code mô hình tối ưu vẫn khá lâu và lặp lại rất nhiều lần chạy.

Tập dữ liệu lớn với 1017209 dòng và 18 cột làm cho việc phân tích dự đoán với các mô hình gặp trục trặc. Bên cạnh đó, đây là bộ dữ liệu cũ chưa được cập nhật.

Nhiều siêu tham số được tìm hiểu nhưng chưa áp dụng vào bài. Cụ thể, trong mô hình Decision tree không dùng `min_impurity_decrease`, `min_samples_split`.

Nhóm tập trung vào mô hình học máy với những mô hình quen thuộc và cơ bản, chưa mở rộng và so sánh với các mô hình khác như mô hình times series để tìm ra mô hình tốt nhất cho dự đoán doanh số.

5.3 Hướng phát triển trong tương lai

Từ những kết quả đạt được và những hạn chế nêu trên, nhóm đề xuất một số hướng phát triển trong tương lai cho đề án như sau:

Mở rộng so sánh hiệu quả giữa các loại mô hình và đề xuất so sánh hiệu quả giữa phân tích thời gian và học máy. Sau khi tham khảo một số nghiên cứu liên quan, nhóm nhận thấy hướng nghiên cứu dự báo doanh số bán hàng đã có từ rất sớm với mô hình chuỗi

thời gian nhưng mô hình này chỉ tập trung vào chính khối lượng bán hàng mà không xem xét các biến liên quan khác trong cùng thời kỳ nên hiệu quả cho ra sẽ khác nhau [37] [1]. Hiện tại nhóm chỉ mới dừng ở việc thử nghiệm và so sánh các mô hình học máy để dự đoán doanh số, trong tương lai, nếu có thể nhóm sẽ mở rộng sử dụng các mô hình chuỗi thời gian ARIMA và các mô hình mạng nơ-ron nhân tạo artificial neural network để có cái nhìn toàn diện hơn về bài toán.

Ứng dụng thêm các giải pháp tối ưu hiệu suất mô hình hiệu quả hơn. Đầu tiên là tìm hiểu và thực hiện các giải pháp giảm thời gian chạy và tài nguyên máy như tối giản hóa tập dữ liệu, loại bỏ các thuộc tính không quan trọng, giảm chiều dữ liệu,... Tiếp theo là, thay thế các phương pháp tối ưu tham số không hiệu quả, tốn nhiều tài nguyên bằng một số phương pháp khác như Hyperopt, Bayesian Optimization và Genetic Algorithms [38]. Hiện tại, nhóm có sử dụng hai phương pháp tối ưu tham số là GridSearchCV và RandomizedSearchCV nhưng đều không hiệu quả khi áp dụng vào mô hình Random Forest trên tập dữ liệu của nhóm. Trong khi GridSearchCV thử mọi kết hợp trong danh sách giá trị của các siêu tham số và đưa ra mô hình tốt nhất dựa trên điểm xác thực chéo nhưng lại tốn rất nhiều thời gian, tài nguyên máy; RandomizedSearchCV thử kết hợp ngẫu nhiên nhiều loại giá trị, có thể sử dụng trên tập dữ liệu lớn nhưng nó không đảm bảo đưa ra mô hình tốt nhất vì không phải tất cả các giá trị tham số đều được thử [39] [40].

Mở rộng thử nghiệm nhiều siêu tham số khác nhau có tiềm năng chưa được thử nghiệm như `min_impurity_decrease`, `criterion`, `min_samples_leaf` ở cả hai mô hình Decision Tree và Random Forest,... Và một số mô hình hồi quy dự đoán doanh số bán hàng khác để có cái nhìn khách quan hơn và xác định được đâu là mô hình tốt nhất và yếu nhất trong bài toán dự đoán doanh số bán hàng. Trong đó, một số mô hình tiềm năng được nhóm đề xuất có kết quả tốt trong các nghiên cứu liên quan về dự báo doanh số bán hàng như Gradient Boosted [2], XGBoost [1], Support Vector Machines (SVM) [2].

TÀI LIỆU THAM KHẢO

- [1] Ankur Jain, Manghat Nitish Menon, Saurabh Chandr, "Sales Forecasting for Retail Chains," *Semantic Scholar*, 2015.
- [2] Sai Nikhil Boyapati, Ramesh Mummidi, "Predicting sales using Machine Learning Techniques," Blekinge Institute of Technology, Sweden, 2020.
- [3] "Python là gì? Các kiến thức cần biết về lập trình Python," Mất Bảo, 15 10 2021. [Online]. Available: <https://wiki.matbao.net/python-la-gi-cac-kien-thuc-can-biet-ve-lap-trinh-python/>. [Accessed 12 3 2023].
- [4] Huy, "Phân tích dữ liệu với Python: 4 bước đơn giản," COLE.VN, 25 06 2022. [Online]. Available: <https://blog.cole.vn/phan-tich-du-lieu-voi-python-4-buoc-don-gian/>. [Accessed 12 3 2023].
- [5] Phu, "Dự báo bán hàng là gì? 7 sale forecast method tốt nhất dành cho nhà quản lý," Mobi Work, 13 9 2021. [Online]. Available: <https://mobiwork.vn/du-bao-ban-hang-la-gi-7-sale-forecast-method-tot-nhat-danh-cho-nha-quan-ly/>. [Accessed 14 3 2023].
- [6] Nam, "DỰ BÁO DOANH SỐ: Ý NGHĨA, TẦM QUAN TRỌNG VÀ CÁC PHƯƠNG PHÁP," SAGA, 8 7 2020. [Online]. Available: https://docs.google.com/document/d/1N9W_IF4iRfIY9GoSdLl6dxXb-QCD85DSctWDp8p2Ec/edit. [Accessed 14 3 2023].
- [7] Lyn, "Bài Toán Dự Báo Bán Hàng (Sales Forecast)," Magestore, 26 4 2020. [Online]. Available: <https://insights.magestore.com/posts/du-bao-ban-hang-sales-forecast>. [Accessed 14 3 2023].
- [8] Hop, "KNN (K-Nearest Neighbors) #1," Viblo, 16 7 2019. [Online]. Available: <https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz>. [Accessed 20 3 2023].
- [9] "Bài 6: K-nearest neighbors," Machine learning cơ bản, 8 1 2017. [Online]. Available: <https://machinelearningcoban.com/2017/01/08/knn/>. [Accessed 20 3 2023].
- [10] "Một số kiến thức về Đại Số Tuyến Tính, Xác Suất Thống Kê, Toán Tối Ưu cần thiết cho Machine Learning.," Machine learning cơ bản, [Online]. Available: <https://machinelearningcoban.com/math#dinh-nghia>. [Accessed 20 3 2023].
- [11] "K-Nearest Neighbors Algorithm," IBM , [Online]. Available: <https://www.ibm.com/topics/knn>. [Accessed 20 3 2023].
- [12] S. Raschka, "Why is Nearest Neighbor a Lazy Algorithm?," Sebastian Raschka, [Online]. Available: <https://sebastianraschka.com/faq/docs/lazy-knn.html>. [Accessed 20 3 2023].
- [13] Uyên, "[Machine Learning] Linear Regression và ứng dụng cho bài toán dự đoán điểm Nhập môn Lập trình," AI CLUB TUTORIALS, 21 4 2021. [Online]. Available: <http://tutorials.aiclub.cs.uit.edu.vn/index.php/2021/04/24/linear-regression/>. [Accessed 20 3 2023].
- [14] "Linear Regression in Machine Learning," Java Point, [Online]. Available: <https://www.javatpoint.com/linear-regression-in-machine-learning>. [Accessed 20 3 2023].

- [15] "Cost Function in Machine Learning," Java Point, [Online]. Available: <https://www.javatpoint.com/cost-function-in-machine-learning>. [Accessed 20 4 2023].
- [16] K. MALI, "Everything you need to Know about Linear Regression!," Analytics Vidhya, 4 10 2021. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>. [Accessed 20 3 2023].
- [17] Quy, "4. Gradient Descent," Quy's Blog, 3 4 2021. [Online]. Available: <https://ndquy.github.io/posts/gradient-descent-2/>. [Accessed 20 3 2023].
- [18] Tuan, "Bài 1: Linear Regression và Gradient descent," nttuan8, 24 2 2019. [Online]. Available: <https://nttuan8.com/bai-1-linear-regression-va-gradient-descent/>. [Accessed 20 3 2023].
- [19] "sklearn.linear_model.LinearRegression," Scikit Learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. [Accessed 20 3 2023].
- [20] Huy, "Regularized linear regression," How K team, 2020. [Online]. Available: <https://howkteam.vn/course/machine-learning-co-ban-voi-numpy/regularized-linear-regression-4037>. [Accessed 20 3 2023].
- [21] Loc, "Đa cộng tuyến: Nguyên nhân, dấu hiệu nhận biết và cách khắc phục," Pham Loc Blog, 2 10 2021. [Online]. Available: <https://www.phamlocblog.com/2018/07/da-cong-tuyen-nhan-biet-khac-phuc.html>. [Accessed 20 3 2023].
- [22] An, "Chương 6: TREES - 1. Tree là gì? Lý thuyết về Binary Tree.," Viblo, 26 11 2022. [Online]. Available: <https://viblo.asia/p/chuong-6-trees-1-tree-la-gi-ly-thuyet-ve-binary-tree-obA46PM9LKv>. [Accessed 20 3 2023].
- [23] "Decision Tree và ý nghĩa của các chỉ số Gini Impurity và Entropy," Vietnamlab, 09 10 2019. [Online]. Available: <https://blog.vietnamlab.vn/decision-tree/>. [Accessed 20 3 2023].
- [24] "Decision Tree Advantages and Disadvantages," Educba, [Online]. Available: <https://www.educba.com/decision-tree-advantages-and-disadvantages/>. [Accessed 20 3 2023].
- [25] Trung, "Gradient Boosting - Tất tần tít về thuật toán mạnh mẽ nhất trong Machine Learning," Viblo, 28 5 2021. [Online]. Available: <https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0>. [Accessed 20 3 2023].
- [26] Ánh, "Phương pháp phát hiện tấn công web ứng dụng kỹ thuật phân tích hành vi," HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG, Hà Nội, 2021.
- [27] Khánh, "9.1. Ý tưởng của mô hình rừng cây," Deep AI KhanhBlog, [Online]. Available: https://phamdinhhkhanh.github.io/deepai-book/ch_ml/RandomForest.html. [Accessed 20 3 2023].
- [28] PHẠM MINH HẢI, NGUYỄN NGỌC QUANG, "KHÁI NIỆM VỀ PHƯƠNG PHÁP RANDOM FOREST TRONG CUỘC CÁCH MẠNG MACHINE LEARNING VÀ ĐỊNH HƯỚNG ỨNG DỤNG TRONG LĨNH VỰC VIỄN THÁM," *TẠP CHÍ KHOA HỌC ĐO ĐẠC VÀ BẢN ĐỒ*, Vols. 39-3/2019, 2019.
- [29] Thanh, "Kỹ thuật học máy phối hợp và tiền xử lý dữ liệu trong việc nâng cao chất lượng phân lớp của các hệ thống phát hiện xâm nhập mạng," TRƯỜNG ĐẠI HỌC LẠC HỒNG, Đồng Nai, 2022.
- [30] Khánh, "Bài 24 - Mất cân bằng dữ liệu (imbalanced dataset)," Khoa học dữ liệu - Khanh's blog, 17 2 2020. [Online]. Available: <https://phamdinhhkhanh.github.io/2020/02/17/ImbalancedData.html>. [Accessed 20 3 2023].

- [31] Đỗ, "Tìm hiểu và tối ưu các tham số trong thuật toán Random Forest," Trường Đại học Nha Trang, Nha Trang, 2015.
- [32] Thịnh, "Đánh giá model trong Machine Learning," Viblo, 29 5 2022. [Online]. Available: <https://viblo.asia/p/danh-gia-model-trong-machine-learning-RnB5pAq7KPG>. [Accessed 20 3 2023].
- [33] Ankur Jain, Manghat Nitish Menon, Saurabh Chandra, "Sales Forecasting for Retail Chains," *Semantic Scholar*, 2015.
- [34] "R Squared là gì? Công thức và ý nghĩa của R Squared," DINHNGHIA.VN, [Online]. Available: <https://dinhnghia.vn/r-square-la-gi-cong-thuc-r-square.html>. [Accessed 20 3 2023].
- [35] Chenghao Wang, Yang Li, "Drug Store Sales Prediction".
- [36] Stuti Raizada, Jatinderkumar R. Saini, "Comparative Analysis of Supervised Machine," (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 12, 2021.
- [37] B. Yao, "Walmart Sales Prediction Based on Decision Tree, Random Forest, and K Neighbors Regressor," *International Conference on Financial Technology and Market Management*, vol. 5, no. 978-1-62437-674-0, 2022.
- [38] P. Y. Wicaksana, "Hyperparameter Optimization on Random Forest Classifier," Medium, 9 3 2022. [Online]. Available: <https://medium.com/@prabowoyogawicaksana/hyperparameter-optimization-random-forest-classifier-550fd5ed8e14>. [Accessed 5 4 2023].
- [39] W. Koehrsen, "Hyperparameter Tuning the Random Forest in Python," Towards Data Science, 10 1 2018. [Online]. Available: <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>. [Accessed 5 4 2023].
- [40] "Alternative Hyperparameter Optimization Technique You need to Know – Hyperopt," Analytics Vidhya, 18 9 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/09/alternative-hyperparameter-optimization-technique-you-need-to-know-hyperopt/>. [Accessed 5 4 2023].
- [41] Kaggle, "Rossmann Store Sales," 2015.

MÃ NGUỒN

Đường dẫn đến mã nguồn:

https://colab.research.google.com/drive/1zYSZB8HNU_EsmINb8Lscza6uQO2q6UMb?usp=sharing