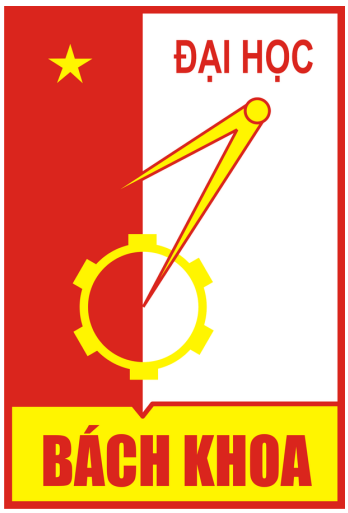
ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



Hệ Hỗ Trợ Quyết Định

Dự đoán tần suất mua hàng của khách hàng

Giảng viên hướng dẫn: TS. Trần Ngọc Thăng
Nhóm: 22
Sinh viên: Vũ Danh Trung Hiếu - 20216925
Đỗ Anh Duy - 20210278

Hà Nội, 06/2024

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1. LỜI NÓI ĐẦU

Trong thế kỷ 21, dữ liệu được ví như vàng với vai trò cực kỳ quan trọng trong mọi lĩnh vực, đặc biệt là trong kinh doanh. Dữ liệu không chỉ là nguồn thông tin mà còn là mũi nhọn để các doanh nghiệp cạnh tranh và tồn tại trong thị trường khốc liệt hiện nay.

Trong lĩnh vực kinh doanh và đặc biệt là lĩnh vực thương mại điện tử, dữ liệu đóng vai trò tiên quyết trong việc xây dựng chiến lược kinh doanh hiệu quả. Việc phân hạng khách hàng là một trong những công cụ mạnh mẽ để giúp các doanh nghiệp hiểu rõ hơn về các nhóm khách hàng tiềm năng từ đó đưa ra các tùy chỉnh chiến lược tiếp thị một cách phù hợp cho từng nhóm đối tượng khách hàng.

Với sự phát triển nhanh chóng của công nghệ, việc áp dụng các mô hình học máy, học sâu vào việc phân tích dữ liệu đã trở thành công cụ quan trọng không thể thiếu trong việc giải quyết các bài toán trong kinh doanh. Nhờ sức mạnh tính toán và khả năng học hỏi nhanh chóng, các mô hình học máy có thể xử lý và phân tích hàng tỉ dòng dữ liệu một cách nhanh chóng và hiệu quả.

Báo cáo này sẽ đi sâu vào việc nghiên cứu và áp dụng các mô hình học máy để phân cụm khách hàng và áp dụng các mô hình học máy đã tìm hiểu vào việc phân hạng khách hàng dựa trên lịch sử giao dịch của công ty thương mại điện tử. Từ đó khám phá những lợi ích to lớn mà học máy mang lại cho chiến lược kinh doanh và sự phát triển của doanh nghiệp.

CHƯƠNG 2. GIỚI THIỆU CHỦ ĐỀ

Trong bối cảnh của thị trường thương mại điện tử ngày nay, việc hiểu và phân tích hành vi mua sắm của khách hàng là một yếu tố quan trọng không thể thiếu đối với bất kỳ doanh nghiệp nào mong muốn thành công. Công ty FADO, công ty chuyên kinh doanh các sản phẩm quà tặng lưu niệm, đã nhận thức rõ ràng về sự cần thiết của việc phân nhóm khách hàng để hiểu rõ hơn về các xu hướng và nhu cầu của từng nhóm khách hàng khác nhau.

Dữ liệu từ các giao dịch chi tiết trong khoảng thời gian từ 01/12/2009 đến 09/12/2011 đóng vai trò quan trọng trong việc giúp FADO xây dựng các nhóm khách hàng có cùng hành vi mua sắm. Việc này không chỉ giúp công ty phát triển các chiến lược tiếp thị chính xác mà còn cải thiện sự phục vụ và trải nghiệm mua sắm của từng đối tượng khách hàng.

Mục tiêu của việc phân nhóm khách hàng này là tối ưu hóa các hoạt động tiếp thị, từ việc phát hành các mã giảm giá đến việc thiết kế sản phẩm tặng kèm và các chính sách khuyến mãi phù hợp. Nhờ vào việc áp dụng phân tích dữ liệu này, FADO hy vọng có thể tăng cường doanh thu và đồng thời giữ vững mối quan hệ với khách hàng trong một thị trường ngày càng cạnh tranh và đa dạng.

Qua việc sử dụng kết quả từ phân nhóm khách hàng, FADO không chỉ xác định được mối quan tâm và nhu cầu cụ thể của từng nhóm khách hàng mà còn cải thiện hiệu quả tổng thể của chiến lược kinh doanh, từ đó đem lại lợi ích bền vững và giúp mở rộng thị trường một cách hiệu quả.

CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

Phân cụm và phân lớp là kỹ thuật quan trọng trong lĩnh vực khoa học dữ liệu. Phân cụm là quá trình nhóm các đối tượng dữ liệu vào các nhóm (cụm) sao cho các đối tượng trong cùng một nhóm có tính chất tương đồng cao, nhằm khám phá cấu trúc dữ liệu mà không cần có sự giám sát. Các mô hình phân cụm phổ biến như K-means Clustering, Agglomerative Clustering,...

Phân lớp, là kỹ thuật gán nhãn cho các đối tượng dữ liệu dựa trên các đặc trưng đã biết từ các ví dụ huấn luyện. Mục tiêu của phân lớp là dự đoán nhãn cho các đối tượng mới. Một số mô hình phân lớp phổ biến như Logistic Regression, Support Vector Machines (SVM), Decision Trees...

Các kỹ thuật phân lớp và phân cụm được ứng dụng rộng rãi và quan trọng trong nhiều lĩnh vực từ khoa học dữ liệu đến các ứng dụng thực tế trong đời sống và kinh doanh.

3.1 Mô hình hồi quy logistic

3.1.1 Khái niệm

Hồi quy logistic là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.

Hồi quy logistic là một kỹ thuật quan trọng trong lĩnh vực trí tuệ nhân tạo và máy học (AI/ML). Mô hình ML là các chương trình phần mềm có thể được đào tạo để thực hiện các tác vụ xử lý dữ liệu phức tạp mà không cần sự can thiệp của con người. Mô hình ML được xây dựng bằng hồi quy logistic có thể giúp các tổ chức thu được thông tin chuyên sâu hữu ích từ dữ liệu của mình. Chúng có thể sử dụng những thông tin chuyên sâu này để phân tích dự đoán nhằm giảm chi phí hoạt động, tăng độ hiệu quả và điều chỉnh quy mô nhanh hơn.

3.1.2 Ưu điểm của mô hình hồi quy Logistic

- Đơn giản, dễ hiểu: Các hệ số trong mô hình hồi quy logistic có thể được diễn giải dễ dàng. Ví dụ, một hệ số dương chỉ ra rằng xác suất xảy ra của sự kiện tăng khi biến độc lập tăng, và ngược lại. Điều này giúp các nhà phân tích dễ dàng hiểu và giải thích kết quả của mô hình.
- Tốc độ cao: Các mô hình hồi quy logistic có thể xử lý khối lượng lớn dữ liệu ở tốc độ cao bởi chúng cần ít khả năng điện toán hơn, chẳng hạn như bộ nhớ và sức mạnh xử lý. Điều này khiến các mô hình hồi quy logistic trở nên lý

tưởng đối với những tổ chức đang bắt đầu với các dự án ML để đạt được một số thành tựu nhanh chóng.

- Dễ dàng triển khai: Hồi quy logistic được hỗ trợ mạnh mẽ bởi nhiều phần mềm phân tích và thư viện lập trình như R, Python (scikit-learn, statsmodels), SAS, SPSS, v.v. Điều này giúp các nhà phân tích dễ dàng triển khai và sử dụng mô hình.

3.1.3 Công thức tính

Hồi quy logistic là một mô hình thống kê sử dụng hàm logistic, hay hàm logit trong toán học làm phương trình giữa x và y. Hàm logit ánh xạ y làm hàm sigmoid của x.

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\intercal \mathbf{x} + b)$$

Trong đó:

- $P(y = 1 \mid \mathbf{x})$ là xác suất có điều kiện rằng $y = 1$ cho một vector đầu vào $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\intercal$.
- σ là hàm sigmoid, được định nghĩa là $\sigma(z) = \frac{1}{1+e^{-z}}$.
- $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\intercal$ là vector trọng số của các đặc trưng (hoặc biến độc lập) $\mathbf{x}$.
- b là hệ số chặn (bias term).

3.1.4 Phân loại

- Hồi Quy Logistic Nhị Phân (Binary Logistic Regression)

  Đây là dạng hồi quy logistic cơ bản và phổ biến nhất, được sử dụng khi biến phụ thuộc là nhị phân, tức là chỉ có hai giá trị có thể (ví dụ: 0 và 1, có và không, thành công và thất bại).

  Công thức:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_k X_k)}}$$

- Hồi Quy Logistic Đa Thức (Multinomial Logistic Regression)

  Loại hồi quy này được sử dụng khi biến phụ thuộc có nhiều hơn hai giá trị không có thứ tự (ví dụ: màu sắc, loại hình dịch vụ). Mô hình này là sự mở rộng của hồi quy logistic nhị phân để dự đoán một trong nhiều loại (kết quả).

  Công thức:

$$P(Y = i|X) = \frac{e^(\beta_{i0} + \beta_{i1}X_1 + \beta_{i2}X_2 + ... + \beta_{ik}X_k)}{1 + \sum_{j=1}^{m-1} e^(\beta_{j0} + \beta_{j1}X_1 + \beta_{j2}X_2 + ... + \beta_{jk}X_k)}$$

Trong đó $P(Y = i|X)$ là xác suất của lớp thứ i với m là số lớp

- Hồi Quy Logistic Thứ Tự (Ordinal Logistic Regression)

  Mô hình này được sử dụng khi biến phụ thuộc có nhiều hơn hai giá trị có thứ tự (ví dụ: mức độ hài lòng từ 1 đến 5). Hồi quy logistic thứ tự giúp dự đoán kết quả có thứ tự.

  Công thức:

$$\log(\frac{P(Y <= j)}{P(Y > j)}) = \beta_0^j + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

  Trong đó $P(Y <= j)$ là xác suất mà biên phụ thuộc Y nhỏ hơn hoặc bằng j

3.1.5 Các tiêu chí đánh giá mô hình

Sau khi xây dựng mô hình, chúng ta cần đánh giá độ chính xác và hiệu quả của nó. Một số phương pháp thường được sử dụng bao gồm:

- Ma Trận Nhầm Lẫn (Confusion matrix): Đo lường số lượng dự đoán đúng và sai.
- Độ Chính Xác (Accuracy): Tỷ lệ dự đoán đúng trên tổng số dự đoán.
- Báo Cáo Phân Loại (Classification Report): Bao gồm các chỉ số như Precision, Recall, F1-Score.
- AUC-ROC Curve: Đường cong ROC (Receiver Operating Characteristic) và diện tích dưới đường cong (AUC) đo lường khả năng phân biệt giữa hai lớp của mô hình.

Ma trận nhầm lẫn (Confusion matrix)

Ma trận nhầm lẫn là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:

		Dự đoán	
		Dương (+)	Âm (-)
Thực tế	Dương (+)	TP	FN
	Âm (-)	FP	TN

Trong đó:

- TP (True Positive): Số lượng mẫu dương được dự đoán chính xác là dương.
- FN (False Negative): Số lượng mẫu dương được dự đoán sai là âm.
- FP (False Positive): Số lượng mẫu âm được dự đoán sai là dương.
- TN (True Negative): Số lượng mẫu âm được dự đoán chính xác là âm.

Độ chính xác (Accuracy):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Độ chính xác biểu thị tỷ lệ dự đoán đúng trên tổng số dự đoán.

Độ nhạy (Sensitivity):

$$Sensitivity = \frac{TP}{TP + FN}$$

Độ nhạy biểu thị tỷ lệ mẫu dương được dự đoán chính xác.

Độ đặc hiệu (Specificity):

$$Spectificity = \frac{TN}{TN + FP}$$

Độ đặc hiệu biểu thị tỷ lệ mẫu âm được dự đoán chính xác.

Giá trị dự đoán dương (Precision):

$$Precision = \frac{TP}{TP + FP}$$

Giá trị dự đoán dương biểu thị tỷ lệ mẫu dự đoán dương đúng trên tổng số mẫu dự đoán dương.

F1 Score:

$$F1 = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$$

F1 Score là trung bình điều hòa của giá trị dự đoán dương và độ nhạy, cung cấp một thước đo cân bằng giữa hai chỉ số này.

Báo cáo phân loại (Classification report)

Báo cáo phân loại (classification report) là một tài liệu thống kê được tạo ra để đánh giá hiệu suất của mô hình phân loại. Nó thường bao gồm các chỉ số như độ chính xác (accuracy), độ nhạy (recall), giá trị dự đoán dương (precision) và F1 score cho mỗi lớp trong tập dữ liệu.

	Precision	Recall	F1 Score	Support
Lớp 0	0.75	0.80	0.77	50
Lớp 1	0.82	0.78	0.80	50
Trung bình (macro)	0.79	0.79	0.79	100
Trung bình (weighted)	0.79	0.79	0.79	100

Bảng 3.1: Báo cáo phân loại cho mô hình phân loại nhị phân

Đường cong ROC

Đường cong ROC (Receiver Operating Characteristic curve) là một công cụ quan trọng trong việc đánh giá hiệu suất của các mô hình phân loại, đặc biệt trong các bài toán phân loại nhị phân. Đường cong ROC biểu diễn mối quan hệ giữa Tỷ lệ dương giả (False Positive Rate - FPR) và Tỷ lệ dương đúng (True Positive Rate - TPR) ở các ngưỡng dự đoán khác nhau.

Các thành phần chính:

- Tỷ lệ dương đúng (True Positive Rate - TPR):

$$TPR = \frac{TP}{TP + FN}$$

  TPR còn được gọi là Độ nhạy (Sensitivity).

- Tỷ lệ dương giả (False Positive Rate - FPR):

$$FPR = \frac{FP}{FP + TN}$$

Thành phần của đường cong ROC

- Trục hoành (x-axis): Tỷ lệ dương giả (FPR).
- Trục tung (y-axis): Tỷ lệ dương đúng (TPR).

Đường cong ROC được vẽ bằng cách thay đổi ngưỡng phân loại từ 0 đến 1 và tính toán các giá trị FPR và TPR tại mỗi ngưỡng.

Diện tích dưới đường cong (AUC - Area Under the Curve): AUC là một giá trị tổng hợp của đường cong ROC, biểu thị khả năng phân biệt của mô hình. AUC dao động từ 0 đến 1, với giá trị 1 biểu thị mô hình hoàn hảo và giá trị 0.5 biểu thị mô hình ngẫu nhiên.

3.2 Mô hình K-Means

3.2.1 Khái niệm

K-Means là một phương pháp phân cụm đơn giản thuộc loại học không giám sát và được sử dụng rộng rãi để giải quyết bài toán phân cụm. Ý tưởng chính của K-Means là phân chia một bộ dữ liệu thành K cụm, với K là số lượng cụm biết trước. Các điểm dữ liệu trong cùng 1 cụm thì phải có những tính chất tương đồng nhất định. Mục tiêu của thuật toán là xác định trung tâm mỗi cụm và gán các điểm dữ liệu vào cụm tương ứng sao cho mỗi điểm chỉ thuộc một cụm. (Mỗi điểm chỉ thuộc 1 cụm duy nhất).

3.2.2 Tóm tắt thuật toán

Đầu vào: Dữ liệu X và số lượng cụm cần tìm K Đầu ra: Các trung tâm M và vector nhãn cho từng điểm dữ liệu Y Các bước thực hiện:

- Bước 1: Chọn K điểm ngẫu nhiên làm trung tâm ban đầu.
- Bước 2: Phân mỗi điểm dữ liệu vào cụm có trung tâm gần nhất.
- Bước 3: Cập nhật trung tâm mỗi cụm bằng cách lấy trung bình cộng các điểm trong cụm.
- Bước 4: Lặp lại Bước 2 và 3 cho đến khi không còn sự thay đổi trong việc gán cụm, hoặc đạt số lần lặp tối đa.

3.2.3 Ưu nhược điểm của thuật toán

Ưu điểm:

- Thuật toán đơn giản, dễ hiểu và dễ triển khai.
- Hiệu quả cao với các bộ dữ liệu có số lượng lớn.

Nhược điểm:

- Hiệu quả phụ thuộc nhiều vào sự lựa chọn số cụm K.
- Chưa tốt với các bộ dữ liệu có nhiều trường. Thông thường một đối tượng không phải bao giờ cũng thuộc một cụm.
- Không phù hợp với dữ liệu có phân phối không đồng đều hoặc có nhiều ngoại lệ. Với bộ dữ liệu ban đầu khi thêm một điểm dữ liệu mới nằm quá xa so với các cụm ban đầu sẽ dẫn tới các cụm ban đầu bị phân tán.
- Khó xác định số lượng cụm K một cách chính xác mà không có kiến thức nghiệp vụ cụ thể hoặc sử dụng phương pháp đánh giá thích hợp.

3.2.4 Công thức tính

Công thức cơ bản của thuật toán K-means bao gồm hai bước chính: gán cụm và cập nhật trung tâm cụm.

Gán cụm (Assign clusters):

Cho trước K cụm và các điểm dữ liệu $x^{(1)}$ (với i = 1,2,...,m):

- μ_k là trung tâm cụm thứ k
- $c^{(i)}$ là chỉ số các cụm mà điểm $x^{(i)}$ được gán vào

Công thức để gán điểm dữ liệu vào cụm được tính bằng cách trọn cụm có trung tâm gần nhất với điểm đó. Cụ thể:

$$c^{(i)} = argmin_k ||x^{(i)} - \mu_k||$$

Trong đó:

- $||x^{(i)} - \mu_k||$ là bình phương khoảng cách Euclid giữa điểm $x^{(i)}$ và trung tâm cụm μ_k.
- $argmin_k$ đánh chỉ số của cụm có khoảng cách nhỏ nhất đến điểm $x^{(i)}$.

Cập nhật trung tâm cụm (Update cluster centroids):

Sau khi gán các điểm dữ liệu vào các cụm, ta cần cập nhật lại trung tâm của từng cụm. Trung tâm mới μ_k của cụm thứ k được tính bằng trung bình của tất cả các điểm dữ liệu $x^{(i)}$ thuộc cụm đó:

$$\mu_k = \frac{1}{count_k} \sum_{i=1}^{m} 1\{c^{(i)} = k\} x^{(i)}$$

Trong đó:

- $count_k$ là số lượng điểm dữ liệu trong cụm thứ k.
- $1\{c^{(i)} = k\}x^{(i)}$ là hàm chỉ số, bằng 1 nếu $x^{(i)}$ thuộc cụm k và bằng 0 nếu ngược lại,

3.2.5 Tiêu chí đánh giá mô hình

- Chỉ số Silhouette (Silhouette Score):

  Áp dụng cho các bài toán phân cụm: Đánh giá mức độ tách biệt giữa các cụm. Điểm silhouette càng cao thể hiện rằng các điểm trong cùng một cụm gần nhau và các cụm cách xa nhau.

- Sai số bình phương trung bình (Mean Squared Error - MSE):

  Áp dụng cho các mô hình dự báo: Đánh giá sự sai lệch giữa giá trị dự báo và giá trị thực tế. Không áp dụng trực tiếp cho K-means: Vì K-means không dự đoán giá trị số mà chỉ phân cụm.

- Chỉ số Dunn:

  Áp dụng cho các bài toán phân cụm: Đánh giá mức độ tách biệt giữa các cụm và sự gần gũi bên trong từng cụm.

3.3 Mô hình phân cụm tổng hợp

3.3.1 Khái niệm

Agglomerative clustering là một kỹ thuật phân cụm từ dưới lên (bottom-up), trong đó mỗi điểm dữ liệu bắt đầu từ một cụm riêng lẻ và các cụm được hợp nhất lặp đi lặp lại dựa trên một tiêu chí tương đồng cho đến khi tất cả các điểm dữ liệu được gộp vào một cụm duy nhất hoặc đạt đến số lượng cụm mong muốn.

3.3.2 Tiêu chí hợp nhất cụm

Tiêu chí hợp nhất cụm là một khái niệm quan trọng trong Agglomerative clustering, nó quyết định cách thức hai cụm con sẽ được hợp nhất để tạo thành một cụm lớn hơn. Đây là một trong những yếu tố quyết định tính chất của kết quả phân cụm và có ảnh hưởng đáng kể đến cấu trúc cuối cùng của các cụm. Các tiêu chí hợp nhất cụm phổ biến bao gồm:

- Single Linkage (nearest neighbor): Single Linkage xem xét khoảng cách giữa hai điểm gần nhất trong hai cụm con khác nhau. Công thức tính khoảng cách giữa hai cụm C_i và C_j là khoảng cách giữa hai điểm gần nhất thuộc hai cụm

$$d_{\text{single}}(C_i, C_j) = \min_{x \in C_i, y \in C_j} |x - y|$$

  Trong đó:

  C_i, C_j : Hai cụm cần so sánh.

  x, y : Các điểm trong C_i và C_j tương ứng.

  $|x - y|$: Khoảng cách giữa hai điểm x và y.

  Single Linkage có xu hướng tạo ra các cụm dài và thưa hơn trong các dendrogram, và có thể bị ảnh hưởng nhiều khi dữ liệu xuất hiện các điểm ngoại lai hoặc điểm nhiễu.

- Complete Linkage (farthest neighbor): Complete Linkage xem xét khoảng

cách giữa hai điểm xa nhất trong hai cụm con khác nhau. Khoảng cách giữa hai cụm C_i và C_j là khoảng cách giữa hai điểm xa nhất thuộc hai cụm.

$$d_{\text{complete}}(C_i, C_j) = \max_{x \in C_i, y \in C_j} |x - y|$$

Trong đó:

C_i, C_j : Hai cụm cần so sánh.

x, y : Các điểm trong C_i và C_j tương ứng.

$|x - y|$: Khoảng cách giữa hai điểm x và y.

Complete Linkage đảm bảo rằng các cụm hợp nhất sẽ có khoảng cách lớn nhất có thể giữa các điểm trong hai cụm con. Tiêu chí này thường tạo ra các cụm đồng đều hơn và ít bị ảnh hưởng bởi nhiễu hơn so với Single Linkage.

- Average Linkage: Khoảng cách giữa hai cụm được tính bằng trung bình khoảng cách giữa tất cả các cặp điểm thuộc hai cụm khác nhau. Công thức tính khoảng cách giữa hai cụm C_i và C_j được xác định bằng công thức:

$$d_{\text{average}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} |x - y|$$

Trong đó:

$|C_i|, |C_j|$: Số lượng điểm trong C_i và C_j tương ứng.

$|x - y|$: Khoảng cách giữa hai điểm x và y.

- Ward's Method: Tiêu chí này dựa trên việc tối thiểu hóa sự gia tăng của tổng bình phương sai số (sum of squared errors - SSE) khi hợp nhất hai cụm. Khoảng cách giữa hai cụm C_i và C_j bằng Ward's Method được xác định bằng công thức:

$$d_{\text{Ward}}(C_i, C_j) = \frac{|C_i| + |C_j|}{|C_i| \cdot |C_j|} \cdot |\mu_i - \mu_j|$$

Trong đó:

$|C_i|, |C_j|$: Số lượng điểm trong C_i và C_j tương ứng.

μ_i, μ_j : Là trung bình của các điểm trong C_i và C_j tương ứng.

Ward's Method thường đưa ra các cụm đồng đều và ít bị ảnh hưởng bởi điểm

ngoại lai và điểm nhiễu hơn so với các phương pháp khác.

Việc lựa chọn tiêu chí sẽ ảnh hưởng đến cấu trúc của các cụm và kết quả cuối cùng của bài toán phân cụm. Lựa chọn tiêu chí hợp nhất cụm phù hợp là một bước quan trọng trong quá trình thiết kế và triển khai mô hình Agglomerative clustering.

3.3.3 Xác định đại diện cụm

Đại diện cụm là quá trình xác định một điểm dữ liệu hoặc một tập điểm dữ liệu đại diện cho mỗi cụm sau khi các cụm con được hợp nhất lại thành các cụm lớn hơn. Điểm đại diện của mỗi cụm được sử dụng để mô tả cụm đó và có thể được sử dụng trong các ứng dụng như phân loại, nhận dạng, hoặc để trực quan hóa kết quả phân cụm. Các phương pháp xác định đại diện cụm thường được dùng trong Agglomerative clustering bao gồm:

- Centroid thường được sử dụng trong các tập dữ liệu mà mỗi điểm dữ liệu là một vector số và được dùng để trực quan hóa kết quả phân cụm. Centroid của một cụm được tính bằng cách lấy trung bình của tất cả các điểm trong cụm đó. Điểm trọng tâm m_k của cụm C_k được tính theo công thức:

$$m_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$$

  Trong đó:

  $|C_k|$: Số lượng điểm trong cụm C_k.

  x : Các điểm dữ liệu trong cụm C_k

  Centroid thường được sử dụng phổ biến bởi tính đại diện trung bình của tất cả các điểm trong cụm, dễ tính toán và hiệu quả trong việc mô tả vị trí trung tâm của cụm.

- Medoid thường được sử dụng trong các tập dữ liệu mà mỗi điểm dữ liệu không phải là một vector, hoặc khi ta cần một điểm dữ liệu thực tế trong tập dữ liệu để đại diện cho cụm. Điểm trọng tâm m_k của cụm C_k được tính theo công thức:

$$m_k = argmin_{x \in C_k} \sum_{y \in C_k} ||x - y||$$

  Trong đó:

  $argmin$: Cho biết medoid là điểm trong cụm có tổng khoảng cách đến các điểm khác trong cụ

Medoid thường được sử dụng khi cần đại diện cho các cụm trong các tập dữ liệu có cấu trúc phức tạp hơn và khi các khoảng cách được đo không phải là khoảng cách Euclide.

3.3.4 Ưu nhược điểm của mô hình

Mô hình Agglomerative clustering là một trong những phương pháp phân cụm được sử dụng rộng rãi trong lĩnh vực học máy và khai phá dữ liệu. Tuy nhiên, như mọi mô hình học máy khác, nó cũng có những ưu điểm và nhược điểm riêng.

- Ưu điểm:
  - Dễ triển khai và hiểu: Agglomerative clustering là một phương pháp đơn giản và dễ hiểu. Quá trình phân cụm bắt đầu bằng việc coi mỗi điểm dữ liệu là một cụm riêng biệt và sau đó hợp nhất các cụm gần nhất cho đến khi chỉ còn lại một cụm duy nhất.
  - Phù hợp với các cấu trúc dữ liệu phức tạp: Agglomerative clustering có thể hoạt động tốt trên các cấu trúc dữ liệu phức tạp, ví dụ như các dạng cụm không đều (non-convex clusters).
  - Có thể đo lường sự tương đồng giữa các cụm con: Bằng cách sử dụng các phương pháp đo khoảng cách khác nhau như Euclidean, Manhattan, hay các độ đo khoảng cách ngẫu nhiên, Agglomerative clustering cho phép đo lường sự tương đồng giữa các cụm con và hợp nhất chúng dựa trên khoảng cách này.
- Nhược điểm:
  - Độ phức tạp tính toán: Agglomerative clustering có độ phức tạp tính toán cao hơn so với một số phương pháp phân cụm khác, đặc biệt là khi số lượng điểm dữ liệu lớn. Quá trình hợp nhất các cụm và tính toán khoảng cách giữa các cụm có thể tốn nhiều thời gian và bộ nhớ.
  - Nhạy cảm với nhiễu và điểm ngoại lai: Agglomerative clustering có thể nhạy cảm với nhiễu và điểm dữ liệu ngoại lai (outliers), vì khoảng cách giữa các cụm có thể bị ảnh hưởng bởi những điểm này.
  - Không thích hợp cho dữ liệu lớn: Khi dữ liệu lớn, việc tính toán ma trận khoảng cách và lưu trữ các cây phân cấp có thể gặp vấn đề về tài nguyên tính toán và bộ nhớ.
  - Không thích hợp cho cấu trúc cụm phi tuyến: Agglomerative clustering có thể không hiệu quả trên các cấu trúc cụm phi tuyến (non-hierarchical clusters) vì phương pháp này giả định rằng cấu trúc cụm là phân tầng

(hierarchical).

Agglomerative clustering là một phương pháp mạnh mẽ và linh hoạt trong việc phân cụm dữ liệu, nhưng cũng có những hạn chế riêng. Việc lựa chọn phương pháp phân cụm phù hợp cần dựa trên đặc tính của dữ liệu và mục tiêu của bài toán cụ thể.

3.3.5 Tiêu chí đánh giá mô hình

Để đánh giá hiệu suất của mô hình Agglomerative clustering trong việc phân cụm dữ liệu, chúng ta cần sử dụng các tiêu chí đánh giá đa dạng nhằm đo lường sự phân tách giữa các cụm, đồng thời đánh giá sự đồng đều và sự tách biệt giữa các cụm sau khi quá trình phân cụm hoàn thành. Các tiêu chí này cung cấp cái nhìn toàn diện về hiệu quả và độ phù hợp của mô hình trong từng hoàn cảnh và mục đích cụ thể.

- Silhouette Score Silhouette Score đánh giá mức độ phân tách của các cụm và đo đạc độ tương đồng giữa các điểm dữ liệu trong cùng một cụm so với các cụm khác. Silhouette Score được tính bằng công thức:

$$\text{Silhouette Score} = \frac{1}{N}\sum_{i=1}^{N} S(i)$$

$$\text{S(i)} = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Trong đó:

S(i) : Silhouette Score tại điểm *i*.

a(i) : Là trung bình khoảng cách từ điểm *i* đến tất cả các điểm khác trong cùng một cụm.

b(i) : Là trung bình khoảng cách từ điểm *i* đến tất cả các điểm trong cụm gần nhất (khác cụm của điểm *i*)

Giá trị của Silhouette Score nằm trong đoạn [-1;1]. Cụ thể:

- *S(i)* $\approx$ 1: Điểm dữ liệu đó được xem là rất "tốt" trong cụm của nó. Nó có khoảng cách lớn đến các điểm của cụm khác, và gần với các điểm trong cụm của nó. Điều này cho thấy mô hình đã phân cụm hiệu quả.
- *S(i)* $\approx$ 0: Cho thấy rằng điểm dữ liệu đó nằm gần biên của cụm. Điều này có thể xảy ra khi các cụm chồng lấn lẫn nhau hoặc khi mô hình phân cụm không tốt
- *S(i)* $\approx$ -1: Điểm dữ liệu đó được xem là rất "xấu" trong cụm của nó. Nó có

thể gần biên của cụm hơn là gần trung tâm của cụm. Điều này có thể xảy ra khi các điểm trong cụm phân bố không đồng đều.

- Dunn Index Dunn Index được sử dụng để đo lường sự phân tách giữa các cụm (between-cluster separation) so với sự phân tán bên trong các cụm (within-cluster dispersion). Điểm Dunn Index càng cao thì mô hình phân cụm càng tốt. Dunn Index được tính toán bằng cách lấy tỉ lệ giữa khoảng cách giữa hai cụm gần nhất (between-cluster separation) và đường kính của cụm lớn nhất (diameter of the largest cluster):

$$\text{Dunn Index} = \frac{\min_{1 \leq i < j \leq k} d(C_i, C_j)}{\max_{1 \leq i \leq k} \text{diameter}(C_l)}$$

Trong đó:

d(C_i, C_j) : Khoảng cách giữa hai cụm C_i và C_j.

diameter(C_l) : Đường kính của cụm C_l, tức là khoảng cách lớn nhất giữa hai điểm trong cụm C_l

Giá trị Dunn Index càng cao thì mô hình phân cụm càng tốt, tuy nhiên nó không cung cấp thông tin về cấu trúc nội bộ của từng cụm. Dunn Index là một tiêu chí hữu ích để đánh giá và tối ưu hóa mô hình phân cụm, đo lường sự phân tách giữa các cụm và sự đồng nhất bên trong từng cụm.

CHƯƠNG 4. XÂY DỰNG CHƯƠNG TRÌNH

4.1 Xử lý dữ liệu

Trong phần này, nhóm chúng em sẽ tiến hành sử dụng bộ dữ liệu Online Retail List, bao gồm thông tin về lịch sử giao dịch của một công ty thương mại điện tử. Đây là một tập dữ liệu phong phú, chứa đựng nhiều thông tin quan trọng liên quan đến các giao dịch bán lẻ trực tuyến của công ty. Trước khi áp dụng các mô hình học máy nhằm phân loại và phân cấp khách hàng, chúng em sẽ thực hiện các bước khám phá và xử lý dữ liệu.

Quá trình khám phá dữ liệu sẽ giúp hiểu rõ hơn về cấu trúc và đặc điểm của tập dữ liệu này, từ đó xác định được những thông tin quan trọng và những mối quan hệ tiềm ẩn giữa các biến số. Bước xử lý dữ liệu bao gồm việc làm sạch dữ liệu, xử lý các giá trị thiếu và các ngoại lệ, cũng như chuẩn hóa và chuyển đổi dữ liệu để chuẩn bị cho các bước phân tích tiếp theo.

Mục tiêu của việc này là tạo ra một tập dữ liệu tối ưu, chứa các trường dữ liệu cần thiết và có giá trị nhất để phục vụ cho việc phân tích và xây dựng các mô hình học máy. Việc xử lý dữ liệu một cách cẩn thận và kỹ lưỡng sẽ giúp cải thiện hiệu quả và độ chính xác của các mô hình học máy, đồng thời cung cấp những thông tin chi tiết và chính xác về phân cấp khách hàng, từ đó hỗ trợ công ty trong việc đưa ra các quyết định kinh doanh chiến lược.

```
import pandas as pd
import numpy as np
from scipy.spatial import ConvexHull
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering,
   KMeans
from sklearn.metrics import silhouette_score,
   accuracy_score, confusion_matrix,
   classification_report, roc_curve, auc,
   davies_bouldin_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
```

```
import time
```

Listing 4.1: "Import các thư viện cần dùng"

```
print(df.head(5))
print("=========")
print(df.info())
print("=========")
print(df.describe())
print("=========")
print(df.shape)
```

Listing 4.2: "Đọc dữ liệu và thực hiện khám phá dữ liệu"

```
  Invoice  Quantity      InvoiceDate  Price  Customer ID
0  489434        12  1.12.2009 07:45   6.95      13085.0
1  489434        12  1.12.2009 07:45   6.75      13085.0
2  489434        12  1.12.2009 07:45   6.75      13085.0
3  489434        48  1.12.2009 07:45   2.10      13085.0
4  489434        24  1.12.2009 07:45   1.25      13085.0
=========
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 5 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   Invoice      1048575 non-null  object
 1   Quantity     1048575 non-null  int64
 2   InvoiceDate  1048575 non-null  object
 3   Price        1048575 non-null  float64
 4   Customer ID  811893 non-null   float64
dtypes: float64(2), int64(1), object(2)
memory usage: 40.0+ MB
None
=========
           Quantity         Price    Customer ID
count  1.048575e+06  1.048575e+06  811893.000000
mean   9.957525e+00  4.627346e+00   15324.712265
std    1.335187e+02  1.228024e+02    1697.033034
min   -7.421500e+04 -5.359436e+04   12346.000000
25%    1.000000e+00  1.250000e+00   13971.000000
50%    3.000000e+00  2.100000e+00   15260.000000
75%    1.000000e+01  4.150000e+00   16795.000000
max    7.421500e+04  3.897000e+04   18287.000000
=========
(1048575, 5)
```

Hình 4.1: Các khám phá dữ liệu cơ bản

```
df = df.dropna()
# Convert "InvoiceDate" to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'],
    format='%d.%m.%Y %H:%M').dt.date
# Calculate revenue per order (cancellation revenue =
    0)
df['total_price']=df['Quantity']*df['Price']
df['total_price'] = df['total_price'].apply(lambda x:
    max(x, 0))
# Calculate the number of days from when the order is
    generated to the last day
max_day = df["InvoiceDate"].max() + timedelta(days
   =1)
df['diff']=(max_day-df.InvoiceDate)
print(df.head(5))
print("=========")
print(df.info())
```

Listing 4.3: "Làm sạch dữ liệu"

```
  Invoice  Quantity InvoiceDate  Price  Customer ID  total_price  \
0  489434        12  2009-12-01   6.95      13085.0         83.4
1  489434        12  2009-12-01   6.75      13085.0         81.0
2  489434        12  2009-12-01   6.75      13085.0         81.0
3  489434        48  2009-12-01   2.10      13085.0        100.8
4  489434        24  2009-12-01   1.25      13085.0         30.0

               diff
0  734 days, 0:00:00
1  734 days, 0:00:00
2  734 days, 0:00:00
3  734 days, 0:00:00
4  734 days, 0:00:00
=========
<class 'pandas.core.frame.DataFrame'>
Index: 811893 entries, 0 to 1048574
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      811893 non-null  object
 1   Quantity     811893 non-null  int64
 2   InvoiceDate  811893 non-null  object
 3   Price        811893 non-null  float64
 4   Customer ID  811893 non-null  float64
 5   total_price  811893 non-null  float64
 6   diff         811893 non-null  object
dtypes: float64(3), int64(1), object(3)
memory usage: 49.6+ MB
```

Hình 4.2: Làm sạch dữ liệu

```
data = df.groupby('Customer ID')['Invoice'].nunique()
    .reset_index(name='order_count')  #Total orders
    for each customer
data = pd.merge(data, df.groupby('Customer ID')['
    total_price'].sum().reset_index(name='total_price'
    ), on='Customer ID')  # total revenue from each
    customer
data = pd.merge(data, df.groupby('Customer ID')['
    InvoiceDate'].min().reset_index(name = 'begin_date
    '), on = 'Customer ID') # first day of purchase
data = pd.merge(data, df.groupby('Customer ID')['
    InvoiceDate'].max().reset_index(name = 'end_date')
    , on = 'Customer ID')     # last day of purchase
data['begin_date'] = pd.to_datetime(data['begin_date'
    ], format='%d.%m.%Y')
data['end_date'] = pd.to_datetime(data['end_date'],
    format='%d.%m.%Y')
data['avg_date'] = (((data['end_date'] - data['
    begin_date']).dt.days/ data['order_count'])).round
    (0) #Average number of days an order is generated
data= pd.merge(data, df.groupby('Customer ID')['diff'
    ].min().reset_index(name = 'recency'), on = '
    Customer ID')
data['recency'] = data['recency'].apply(lambda x: x.
    days)
data = data.drop(columns = ['begin_date', 'end_date'
    ])
# total cancel order
cancel_data = df[df['Invoice'].str.startswith('C', na
    =False)].drop_duplicates()
data = pd.merge(data, cancel_data.groupby('Customer
    ID')['Invoice'].nunique().reset_index(name = '
    cancel_order'), on = 'Customer ID')
# total successful order
data['success_order'] = data['order_count'] - data['
    cancel_order']
print(data.head())
```

Listing 4.4: "Chọn những trường dữ liệu quan trọng"

	Customer ID	order_count	total_price	avg_date	recency	cancel_order	success_order
0	12346.0	17	77556.46	24.0	321	5	12
1	12349.0	5	4428.69	143.0	14	1	4
2	12352.0	13	2849.84	27.0	32	3	10
3	12359.0	14	8935.94	52.0	3	4	10
4	12360.0	9	4252.89	67.0	48	1	8
5	12362.0	13	4827.19	56.0	7	3	10
6	12365.0	3	641.38	0.0	287	1	2
7	12379.0	5	1620.22	91.0	77	1	4
8	12380.0	15	9676.30	43.0	17	4	11
9	12381.0	5	1698.30	17.0	29	1	4

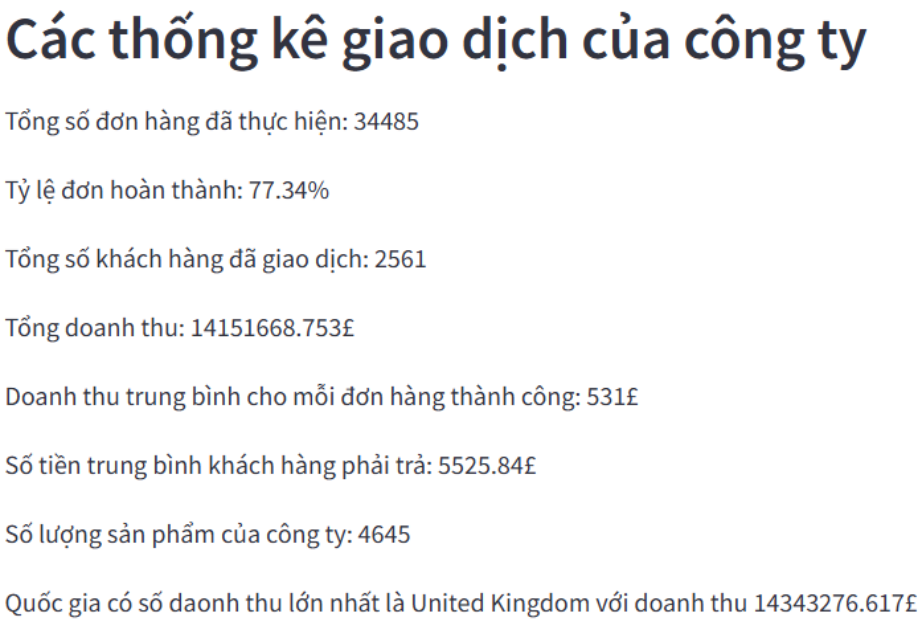
Hình 4.3: Dữ liệu cần thiết cho mô hình

4.2 Các thống kê phân tích cơ bản về giao dịch của công ty

```
print("Total number of orders executed: {}".format(
    data['order_count'].sum()))
print("Completion rate of orders: {}%".format(round
    (100 * (1 - data['cancel_order'].sum() / data['
    order_count'].sum()), 2)))
print("Total number of customers transacted: {}".
    format(data['Customer ID'].count()))
print("Total revenue: {}$\textsterling$".format(data[
    'total_price'].sum()))
print("Average revenue per successful order: {}$\
    textsterling$".format(round(data['total_price'].
    sum() / (data['order_count'].sum() - data['
    cancel_order'].sum()))))
print("Average amount customers have to pay: {}$\
    textsterling$".format(round(data['total_price'].
    sum() / data['Customer ID'].count(), 2)))
print("Number of company's products: {}".format(df['
    StockCode'].nunique()))
# Group data by country and calculate total revenue
    for each country
country_revenue = df.groupby('Country')['total_price'
    ].sum().reset_index()
# Determine the country with the highest revenue
max_revenue_country = country_revenue.loc[
```

```
    country_revenue['total_price'].idxmax()]
print("Country with the highest revenue is {} with
    revenue {}$\textsterling$".format(
    max_revenue_country['Country'],
    max_revenue_country['total_price']))
```

Listing 4.5: "Các thống kê phân tích cơ bản về giao dịch của công ty"



Hình 4.4: Các thống kê phân tích cơ bản về giao dịch của công ty

4.3 Phân lớp khách hàng bằng mô hình hồi quy logistic

Sau khi khám phá bộ dữ liệu ban đầu và có cái nhìn tổng quát về các giao dịch của công ty, chúng em sẽ tiến hành phân lớp khách hàng bằng mô hình hồi quy logistic. Bước đầu tiên là đánh nhãn cho từng phân lớp khách hàng. Ở đây, chúng em sẽ đánh nhãn khách hàng theo giá trị tổng tiền thanh toán, số lượng đơn hàng thành công.

```
total_price_quantiles = df['total_price'].quantile
    ([0.2, 0.4, 0.6, 0.8])
success_order_quantiles = df['order_count'].quantile
    ([0.2, 0.4, 0.6, 0.8])
def classify_customer(row):
    if (row['total_price'] <= total_price_quantiles
        [0.2] and row['order_count'] <=
```

```
            success_order_quantiles[0.2]) or (row['
            total_price'] > total_price_quantiles[0.2] and
             row['order_count'] <= success_order_quantiles
            [0.2]) or (row['total_price'] <=
            total_price_quantiles[0.2] and row['
            order_count'] > success_order_quantiles[0.2]):
             return 0
        elif (row['total_price'] <= total_price_quantiles
            [0.4] and row['order_count'] <=
            success_order_quantiles[0.4]) or (row['
            total_price'] > total_price_quantiles[0.4] and
             row['order_count'] <= success_order_quantiles
            [0.4]) or (row['total_price'] <=
            total_price_quantiles[0.4] and row['
            order_count'] > success_order_quantiles[0.4]):
             return 1
        elif (row['total_price'] <= total_price_quantiles
            [0.6] and row['order_count'] <=
            success_order_quantiles[0.6]) or (row['
            total_price'] > total_price_quantiles[0.6] and
             row['order_count'] <= success_order_quantiles
            [0.6]) or (row['total_price'] <=
            total_price_quantiles[0.6] and row['
            order_count'] > success_order_quantiles[0.6]):
             return 2
        elif (row['total_price'] <= total_price_quantiles
            [0.8] and row['order_count'] <=
            success_order_quantiles[0.8]) or (row['
            total_price'] > total_price_quantiles[0.8] and
             row['order_count'] <= success_order_quantiles
            [0.8]) or (row['total_price'] <=
            total_price_quantiles[0.8] and row['
            order_count'] > success_order_quantiles[0.8]):
             return 3
        else:
             return 4

df['loyal']= df.apply(classify_customer, axis=1)
```

Listing 4.6: "Đánh nhãn cho bộ dữ liệu"

Tiếp theo, chúng em tiến hành xác định các biến độc lập và biến phụ thuộc và chia dữ liệu thành các tập huấn luyện và tập kiểm tra.

```
# Determine independent variables (features) and
   dependent variables (target)
X = df[['total_price', 'avg_date', 'recency']]
y = df['loyal']

# Divide data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X
   , y, test_size=0.2, random_state=42)
```

Listing 4.7: "Xác định các biến và chia dữ liệu thành tập huấn luyện và tập kiểm tra"

Để đánh giá hiệu quả của mô hình, chúng em tiến hành xây dựng các hàm đánh giá liên quan cho mô hình hồi quy logistic như ma trận nhầm lẫn, biểu đồ đường cong ROC.

```
# Confusion matrix
def conf_matrix(y_testt, y_pred):
    conf_matrix = confusion_matrix(y_testt, y_pred)
    fig = plt.figure(figsize=(10, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d',
       cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    st.pyplot(fig)

# ROC chart
def ROC(y_testt, y_pred):
    # Calculate fpr, tpr and threshold values
          for    each class to plot ROC
    fpr = {}
    tpr = {}
    roc_auc = {}
    for i in range(len(np.unique(y))):
        fpr[i], tpr[i], _ = roc_curve(y_testt == i,
           y_pred == i)
        roc_auc[i] = auc(fpr[i], tpr[i])
```

```
    # Draw the ROC chart
    fig = plt.figure(figsize=(10,6))
    for i in range(len(np.unique(y))):
        plt.plot(fpr[i], tpr[i], label=f'ROC curve of
            class {i} (area = {roc_auc[i]:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC
       )')
    plt.legend(loc="lower right")
    st.pyplot(fig)
```

Listing 4.8: "Các hàm đánh giá hiệu quả của mô hình"

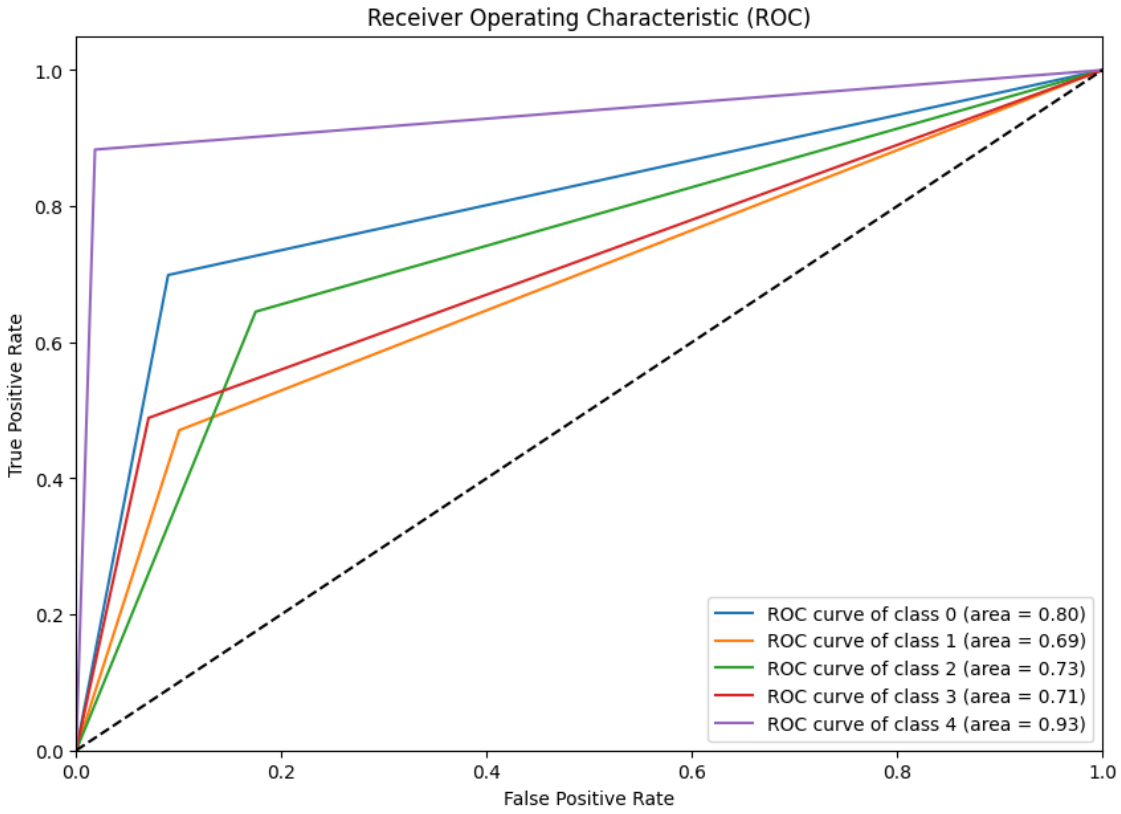
Sau khi đã xử lý dữ liệu và xây dựng các hàm đánh giá cần thiết. Chúng em sẽ tiến hành lựa chọn các mô hình và đánh giá kết quả huấn luyện. Mô hình đầu tiên chúng em sử dụng là mô hình mặc định của hàm LogisticRegression().

```
# Model 1: Default univariate logistic regression
   model
# Create a logistic regression model
model = LogisticRegression()
# Train the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
print("Model 1: Default univariate logistic
   regression model")
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(
   y_test, y_pred))
print("\nClassification Report:\n",
   classification_report(y_test, y_pred))
```

Listing 4.9: "Model 1: Default univariate logistic regression model"

```
Model 1: Default univariate logistic regression model

Accuracy: 0.6413255360623782

Confusion Matrix:
 [[109  28  15   3   1]
 [ 26  40  15   3   1]
 [  6  15  69  15   2]
 [  0   0  41  43   4]
 [  0   0   0   9  68]]

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       156
           1       0.48      0.47      0.48        85
           2       0.49      0.64      0.56       107
           3       0.59      0.49      0.53        88
           4       0.89      0.88      0.89        77

    accuracy                           0.64       513
   macro avg       0.65      0.64      0.64       513
weighted avg       0.65      0.64      0.64       513
```

Hình 4.5: Đánh giá mô hình 1



Hình 4.6: Đường cong ROC cho mô hình 1

Đánh giá về kết quả huấn luyện của mô hình 1:

- Mô hình đạt độ chính xác 64%, cho thấy mô hình đã phân loại đúng 64% tổng số trường hợp.
- Độ chính xác dao động từ 0.48 (Class 1) đến 0.89 (Class 4). Độ chính xác cao cho Class 4 và thấp nhất cho Class 1.
- F1-Score dao động từ 0.48 (Class 1) đến 0.89 (Class 4). Class 4 có điểm F1 cao nhất, cho thấy mô hình hoạt động rất tốt với lớp này.
- Biểu đồ ROC cho thấy mô hình có hiệu suất tốt nhất với Class 4 và kém nhất với Class 1, đây có thể là do sự chênh lệch về số lượng mẫu giữa các lớp hoặc độ phân tán của dữ liệu.

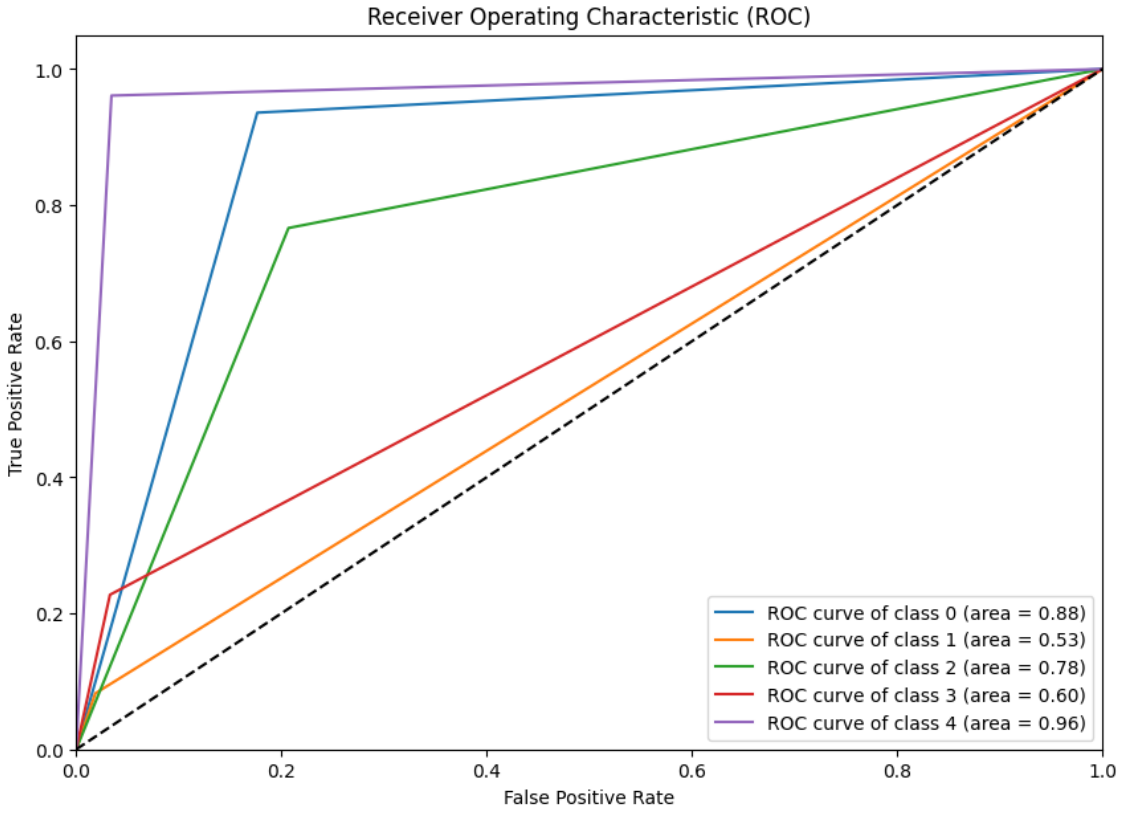
Với mô hình thứ 2, chúng em vẫn sử dụng mô hình logistic nhưng sử dụng tham số penalty = 'l1' và solver = 'liblinear' của hàm LogisticRegresstion(). Tham số L1 Regularization có xu hướng tạo ra các hệ số hồi quy bằng 0 cho một số biến, do đó dẫn đến mô hình thưa thớt. Tham số solver 'liblinear' sử dụng thuật toán tối ưu dựa trên phương pháp descent tuyến tính nhằm tăng hiệu quả phân lớp của mô hình.

```
# Model 2: Model with regularizer L1 (Lasso)
# Create a logistic regression model
model = LogisticRegression(penalty='l1', solver='
    liblinear')
# Train the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
print("Model 2: Model with regularizer L1 (Lasso)")
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(
    y_test, y_pred))
print("\nClassification Report:\n",
    classification_report(y_test, y_pred))
```

Listing 4.10: "Model 2: Model with regularizer L1 (Lasso)"

```
Model 2: Model with regularizer L1 (Lasso)

Accuracy: 0.6413255360623782

Confusion Matrix:
 [[146   0   5   5   0]
 [ 54   7  21   1   2]
 [  9   7  82   6   3]
 [  0   1  57  20  10]
 [  0   0   1   2  74]]

Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.94      0.80       156
           1       0.47      0.08      0.14        85
           2       0.49      0.77      0.60       107
           3       0.59      0.23      0.33        88
           4       0.83      0.96      0.89        77

    accuracy                           0.64       513
   macro avg       0.62      0.59      0.55       513
weighted avg       0.62      0.64      0.58       513
```

Hình 4.7: Đánh giá mô hình 2



Hình 4.8: Đường cong ROC cho mô hình 2

Kết quả huấn luyện của mô hình 2 cho được là khá tốt với Accuracy = 0.64 và

độ chính xác cao nhất đối với lớp thứ 4 (Class 4) là 0.83. Có thể đánh giá mô hình đang hoạt động tốt với một số lớp cụ thể. Tuy nhiên cần cải thiện đối với lớp thứ nhất (Class 1).

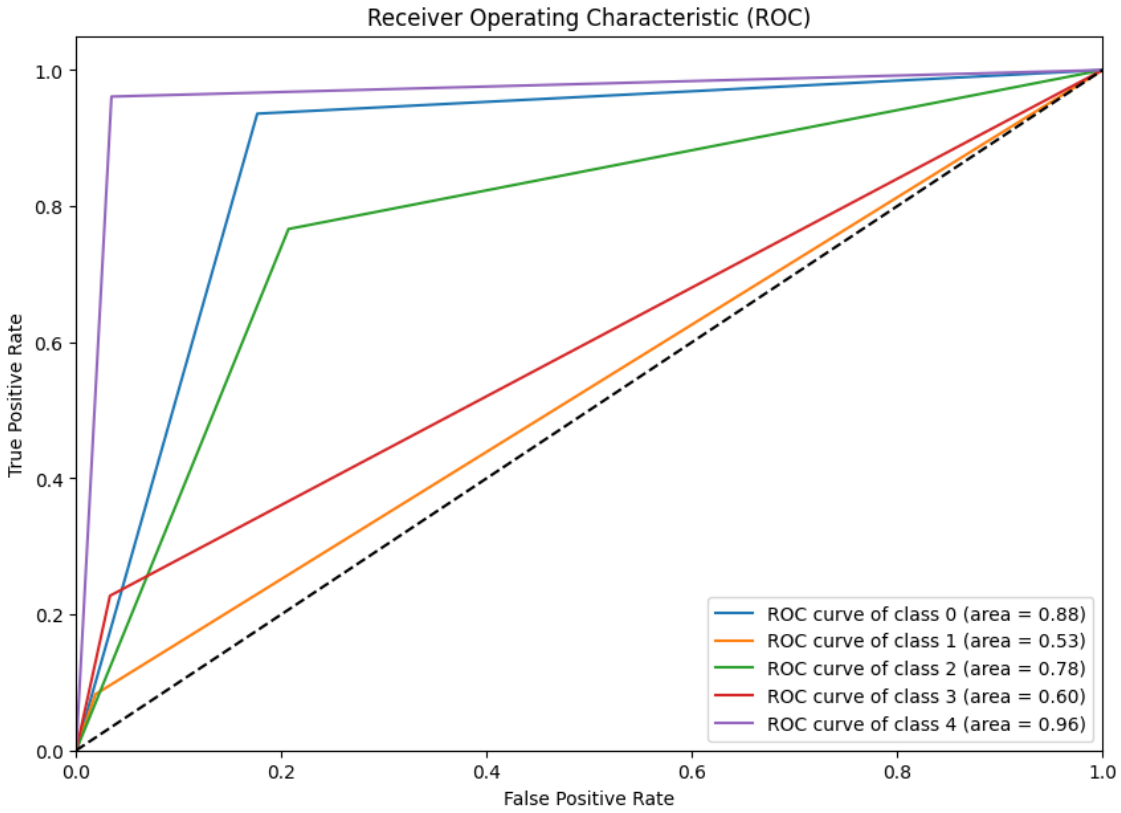
Với mô hình thứ 3, chúng em sử dụng tham số penalty = 'l2' để giúp mô hình hoạt động chính xác hơn và tham số điều chỉnh C= 0.1 giúp mô hình giảm Overfitting. Dưới đây là đoạn mã triển khai mô hình 3.

```
# Model 3: Model with regularizer L2 and C= 0.1 (
    Ridge)
# Create a logistic regression model
model = LogisticRegression(penalty='l2', solver='
    lbfgs', C= 0.1)
# Train the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
print("Model 3: Model with regularizer L2 and C= 0.1
    (Ridge)")
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(
    y_test, y_pred))
print("\nClassification Report:\n",
    classification_report(y_test, y_pred))
```

Listing 4.11: "Model 3: Model with regularizer L2 (Ridge) "

```
Model 3: Model with regularizer L2 and C= 0.1 (Ridge)

Accuracy: 0.6647173489278753

Confusion Matrix:
 [[118  19  13   5   1]
 [ 27  40  14   3   1]
 [  5  17  69  14   2]
 [  0   0  39  47   2]
 [  0   0   0  10  67]]

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.76      0.77       156
           1       0.53      0.47      0.50        85
           2       0.51      0.64      0.57       107
           3       0.59      0.53      0.56        88
           4       0.92      0.87      0.89        77

    accuracy                           0.66       513
   macro avg       0.67      0.66      0.66       513
weighted avg       0.67      0.66      0.67       513
```

Hình 4.9: Đánh giá mô hình 3



Hình 4.10: Đường cong ROC cho mô hình 3

Độ chính xác của mô hình 3 có cải thiện hơn hai mô hình trước với Accuracy = 0.66 và lớp thứ 4 (Class 4) được dự đoán chính xác hơn với Precision = 0.92.

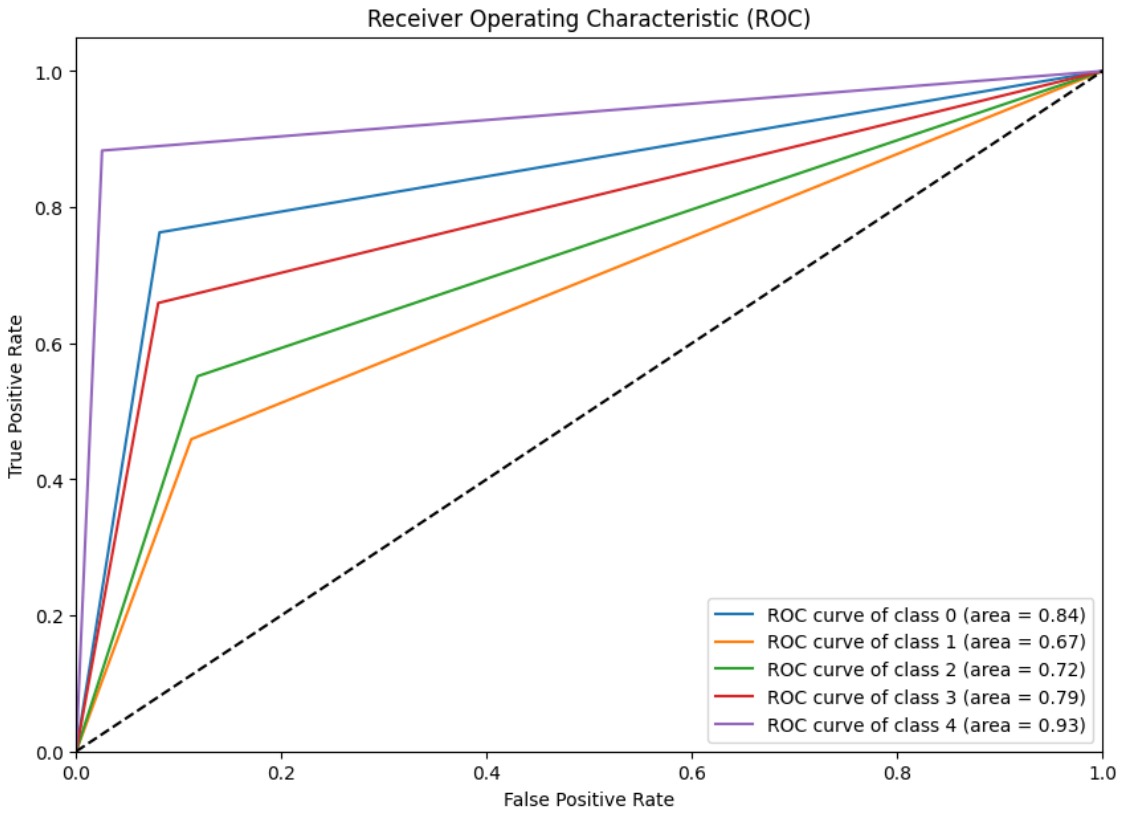
Mô hình 4 sử dụng class_weight='balanced' giúp giải quyết vấn đề dữ liệu mất cân bằng bằng cách điều chỉnh trọng số của các lớp sao cho các lớp ít mẫu hơn được ưu tiên hơn trong quá trình huấn luyện.

```
# Model 4: Model with class weight balancing (class
    weight balancing)
# Create a logistic regression model
model = LogisticRegression(class_weight='balanced')
# Train the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
print("Model 4: Model with class weight balancing (
    class weight balancing)")
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(
    y_test, y_pred))
print("\nClassification Report:\n",
    classification_report(y_test, y_pred))
```

Listing 4.12: "Model 4: Model with class weight balancing (class weight balancing)"

```
Model 4: Model with class weight balancing (class weight balancing)

Accuracy: 0.6686159844054581

Confusion Matrix:
 [[119  19  12   4   2]
 [ 29  39  12   4   1]
 [  0  28  59  17   3]
 [  0   1  24  58   5]
 [  0   0   0   9  68]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       156
           1       0.45      0.46      0.45        85
           2       0.55      0.55      0.55       107
           3       0.63      0.66      0.64        88
           4       0.86      0.88      0.87        77

    accuracy                           0.67       513
   macro avg       0.66      0.66      0.66       513
weighted avg       0.67      0.67      0.67       513
```

Hình 4.11: Đánh giá mô hình 4



Hình 4.12: Đường cong ROC cho mô hình 4

Bằng việc sử dụng class_weight = 'balanced', mô hình cho kết quả dự đoán giữa các lớp cân bằng hơn. Từ biểu đồ đường cong ROC có thể thấy Class 1 đã cải thiện độ chính xác với AUC tăng từ 0.53 lên 0.67.

Mô hình thứ 5 thay đổi số lần lặp tối đa max_iter lên 1000 của mô hình để tìm điểm hội tụ nhất.

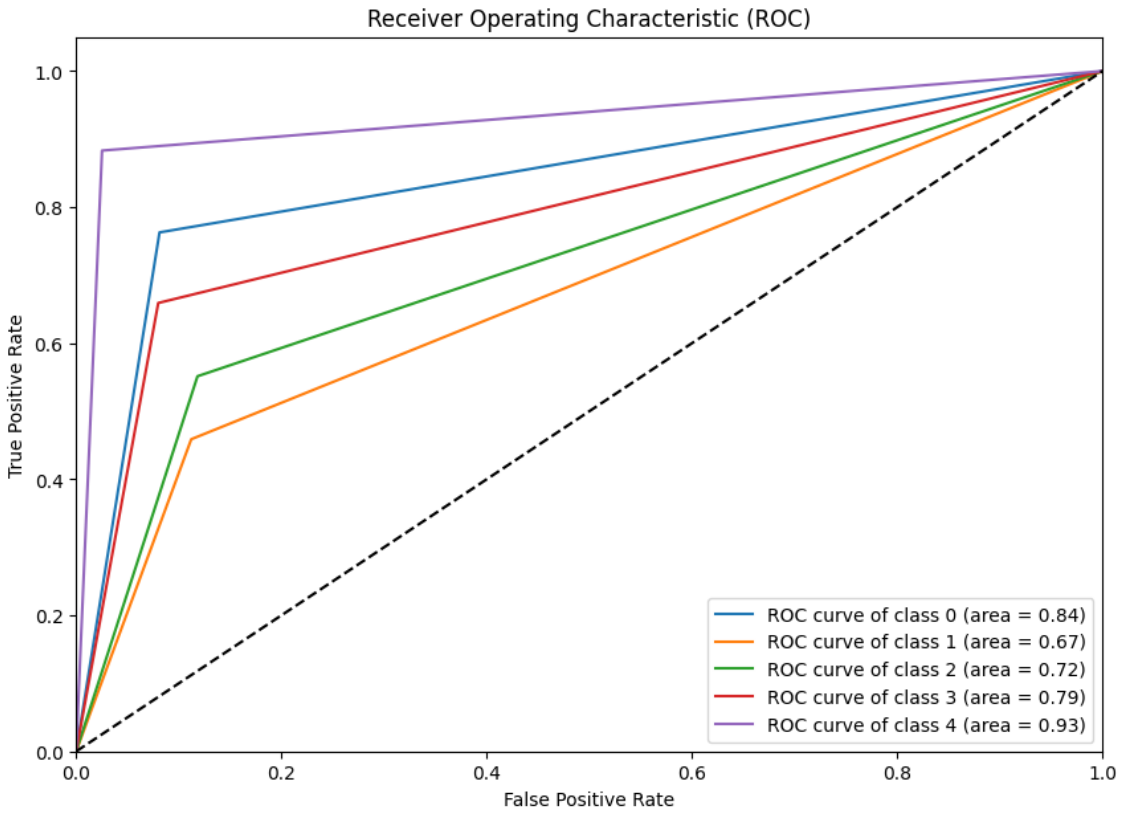
```
# Model 5: Model with limited number of iterations (
    maximum iterations)
# Create a logistic regression model
model = LogisticRegression(max_iter=1000)
# Train the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
print("Model 5: Model with limited number of
    iterations (maximum iterations)")
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(
```

```
    y_test, y_pred))
print("\nClassification Report:\n",
    classification_report(y_test, y_pred))
```

Listing 4.13: "Model 5: Model with limited number of iterations (maximum iterations)"

```
Model 5: Model with limited number of iterations (maximum iterations)

Accuracy: 0.6686159844054581

Confusion Matrix:
 [[119  19  12   4   2]
 [ 29  39  12   4   1]
 [  0  28  59  17   3]
 [  0   1  24  58   5]
 [  0   0   0   9  68]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       156
           1       0.45      0.46      0.45        85
           2       0.55      0.55      0.55       107
           3       0.63      0.66      0.64        88
           4       0.86      0.88      0.87        77

    accuracy                           0.67       513
   macro avg       0.66      0.66      0.66       513
weighted avg       0.67      0.67      0.67       513
```

Hình 4.13: Đánh giá mô hình 5



Hình 4.14: Đường cong ROC cho mô hình 5

Kết quả của mô hình 5 không có nhiều sự thay đổi với mô hình 4 sau khi thay đổi số lần lặp max. Có thể hiểu thuật toán đã hội tụ tại trước khi số lần lặp đạt tối đa. Dễ hiểu thì việc tăng số lần lặp max không thể tìm được kết quả tối ưu hơn.

4.4 Phân cụm khách hàng bằng mô hình K-Means

Trong thuật toán này, chúng em sẽ sử dụng mô hình phân cụm bằng K-Means được đưa ra trong bài báo cáo Customer Segmentation using K-means Clustering | IEEE Conference Publication | IEEE Xplore - 2019. Sau khi thiết lập nền tảng lý thuyết, chúng em sẽ áp dụng thuật toán K-Means trên một bộ dữ liệu mẫu để minh họa cách thức hoạt động và hiệu quả của thuật toán K-means. Quá trình này sẽ bao gồm các bước chuẩn hoá dữ liệu, triển khai thuật toán, và đánh giá kết quả phân cụm thông qua các chỉ số đánh giá như biểu đồ phân cụm và Silhouette Score, Davies-Bouldin Index. Dự liệu dùng để phân cụm khách hàng dựa trên các đặc trưng là tổng giá trị thanh toán và số đơn đặt hàng thành công.

```
# Select the columns to normalize
features = ['total_price', 'success_order']
X = df[features]
# Standardized data
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Listing 4.14: "Chuẩn hoá dữ liệu"

Tiến hành lựa chọn mô hình đầu tiên là mô hình mặc định của thuật toán với số lượng cụm n_cluster = 5.

```
# Model 1: Default base model with n_cluster = 5
kmeans = KMeans(n_clusters= 5, init='k-means++',
   max_iter=100, n_init=10, tol= 0.0001, random_state
   =42)
y_kmeans = kmeans.fit_predict(X_scaled)
# Add clustering results column to DataFrame
df['cluster'] = y_kmeans
# Evaluate the model
print('\nEvaluate clustering results:')
silhouette_avg = silhouette_score(X_scaled, df['
   cluster'])
davies_bouldin = davies_bouldin_score(X_scaled, df['
   cluster'])
print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {davies_bouldin}")
```

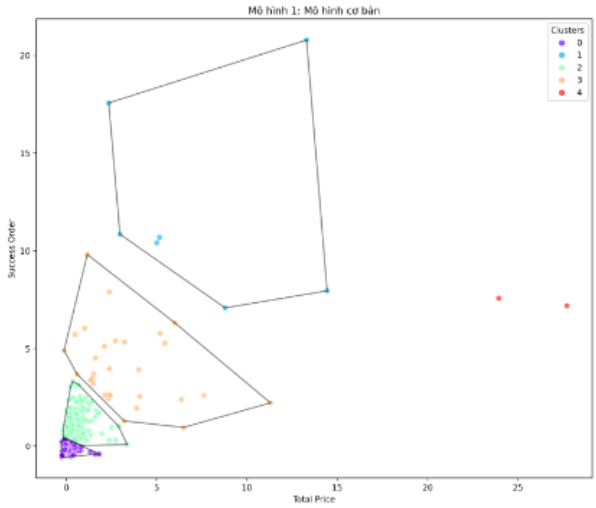
Listing 4.15: "Model 1: Default base model with 5 clusters"

Mô hình 1: Mô hình cơ bản

Đánh giá kết quả phân cụm:

Silhouette Score: 0.6927957140862854

Davies-Bouldin Index: 0.7402851839402337



Bảng dữ liệu sau khi phân cụm

	total_price	success_order	lable
0	77,556.46	12	2
1	4,428.69	4	0
2	2,849.84	10	0
3	8,935.94	10	0
4	4,252.89	8	0
5	4,827.19	10	0
6	641.38	2	0
7	1,620.22	4	0
8	9,676.3	11	0
9	1,698.3	4	0

Hình 4.15: Đánh giá mô hình 1

Sau khi triển khai mô hình, kết quả thu về khá tốt với Silhouette Score: 0.69 là một giá trị tương đối cao, cho thấy các điểm dữ liệu nằm gần trung tâm của cụm và cách xa các cụm khác. Tuy nhiên, Davies-Bouldin Index: 0.74 là một giá trị tương đối cao, cho thấy các cụm có thể có sự chồng chéo lên nhau và cần cải thiện hơn.

Tiếp theo chúng em sẽ cài đặt mô hình 2 với sự thay đổi số lần lặp tối đa max_iter từ 100 lên 300 vòng lặp để quan sát kết quả phân cụm.

```
# Model 2: Change number of 'max_iter' iterations
   from 100 to 300
kmeans = KMeans(n_clusters= 5, init='k-means++',
   max_iter=300, n_init=10, tol= 0.0001, random_state
   =42)
y_kmeans = kmeans.fit_predict(X_scaled)
# Add clustering results column to DataFrame
df['cluster'] = y_kmeans
# Evaluate the model
print('\nEvaluate clustering results:')
silhouette_avg = silhouette_score(X_scaled, df['
```

```
    cluster'])
davies_bouldin = davies_bouldin_score(X_scaled, df['
    cluster'])
print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {davies_bouldin}")
```

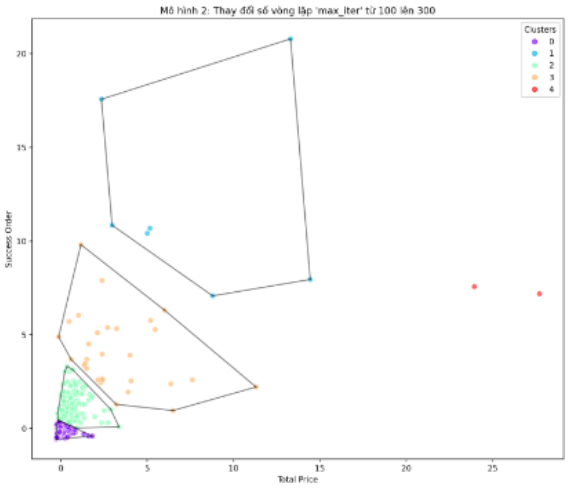
Listing 4.16: "Model 2: Change number of 'max_iter' iterations from 100 to 300"

Mô hình 2: Thay đổi số vòng lặp 'max_iter' từ 100 lên 300

Đánh giá kết quả phân cụm:

Silhouette Score: 0.6927957140862854

Davies-Bouldin Index: 0.7402851839402337



Bảng dữ liệu sau khi phân cụm

	total_price	success_order	lable
0	77,556.46	12	2
1	4,428.69	4	0
2	2,849.84	10	0
3	8,935.94	10	0
4	4,252.89	8	0
5	4,827.19	10	0
6	641.38	2	0
7	1,620.22	4	0
8	9,676.3	11	0
9	1,698.3	4	0

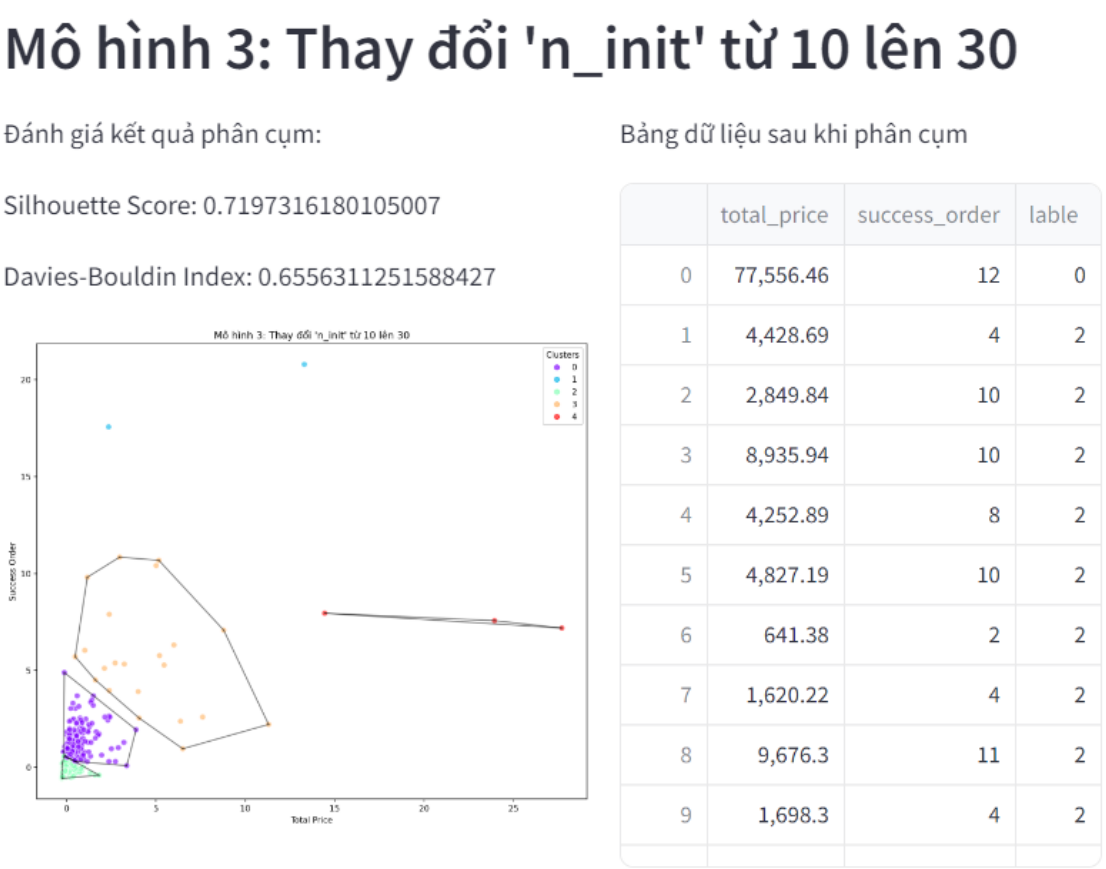
Hình 4.16: Đánh giá mô hình 2

Đối với mô hình 3, chúng em thay đổi n_init từ 10 lên 30 và quan sát kết quả phân cụm. Mục đích của tham số này là tăng số lượng lần khởi tạo ngẫu nhiên các centroids ban đầu mà thuật toán sẽ thử. Mỗi lần khởi tạo một tập hợp các centroids là một lần chạy K-means độc lập.

```
# Model 3: Change 'n_init' from 10 to 30
kmeans = KMeans(n_clusters= 5, init='k-means++',
    max_iter=300, n_init=30, tol= 0.0001, random_state
    =42)
```

```
y_kmeans = kmeans.fit_predict(X_scaled)
# Add clustering results column to DataFrame
df['cluster'] = y_kmeans
# Evaluate the model
print('\nEvaluate clustering results:')
silhouette_avg = silhouette_score(X_scaled, df['
    cluster'])
davies_bouldin = davies_bouldin_score(X_scaled, df['
    cluster'])
print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {davies_bouldin}")
```

Listing 4.17: "Model 3: Change 'n_init' from 10 to 30"



	total_price	success_order	lable
0	77,556.46	12	0
1	4,428.69	4	2
2	2,849.84	10	2
3	8,935.94	10	2
4	4,252.89	8	2
5	4,827.19	10	2
6	641.38	2	2
7	1,620.22	4	2
8	9,676.3	11	2
9	1,698.3	4	2

Hình 4.17: Đánh giá mô hình 3

Kết quả quan sát được cho thấy Silhouette Score đã tăng lên 0.71 và chỉ số Davies-Bouldin giảm xuống còn 0.66. Đây là kết quả tốt, thể hiện thuật toán đang hoạt động tốt và khả năng phân cụm có độ chính xác cao. Từ đó có thể nhận xét, khi tăng số lần khởi tạo thì kết quả của thuật toán cho ra tốt hơn.

Mô hình 4, chúng em thay tham số 'init' từ 'k-means++' thành 'random' với mục đích tạo các centroids ngẫu nhiên trong không gian dữ liệu.

```
# Model 4: Change 'init' from 'k-means++' to 'random'
kmeans = KMeans(n_clusters= 5, init='random',
   max_iter=100, n_init=10, tol= 0.0001, random_state
   =42)
y_kmeans = kmeans.fit_predict(X_scaled)
# Add clustering results column to DataFrame
df['cluster'] = y_kmeans
# Evaluate the model
print('\nEvaluate clustering results:')
silhouette_avg = silhouette_score(X_scaled, df['
   cluster'])
davies_bouldin = davies_bouldin_score(X_scaled, df['
   cluster'])
print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {davies_bouldin}")
```

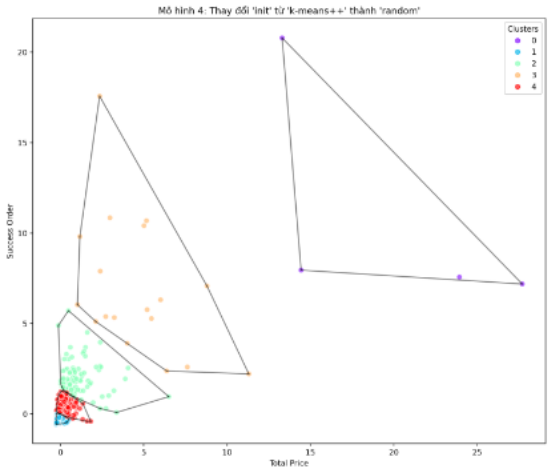
Listing 4.18: "Model 4: Change 'init' from 'k-means++' to 'random'"

Mô hình 4: Thay đổi 'init' từ 'k-means++' thành 'random'

Đánh giá kết quả phân cụm:

Silhouette Score: 0.5984964239869235

Davies-Bouldin Index: 0.7613655811030795



Bảng dữ liệu sau khi phân cụm

	total_price	success_order	lable
0	77,556.46	12	2
1	4,428.69	4	1
2	2,849.84	10	1
3	8,935.94	10	4
4	4,252.89	8	1
5	4,827.19	10	1
6	641.38	2	1
7	1,620.22	4	1
8	9,676.3	11	4
9	1,698.3	4	1

Hình 4.18: Đánh giá mô hình 4

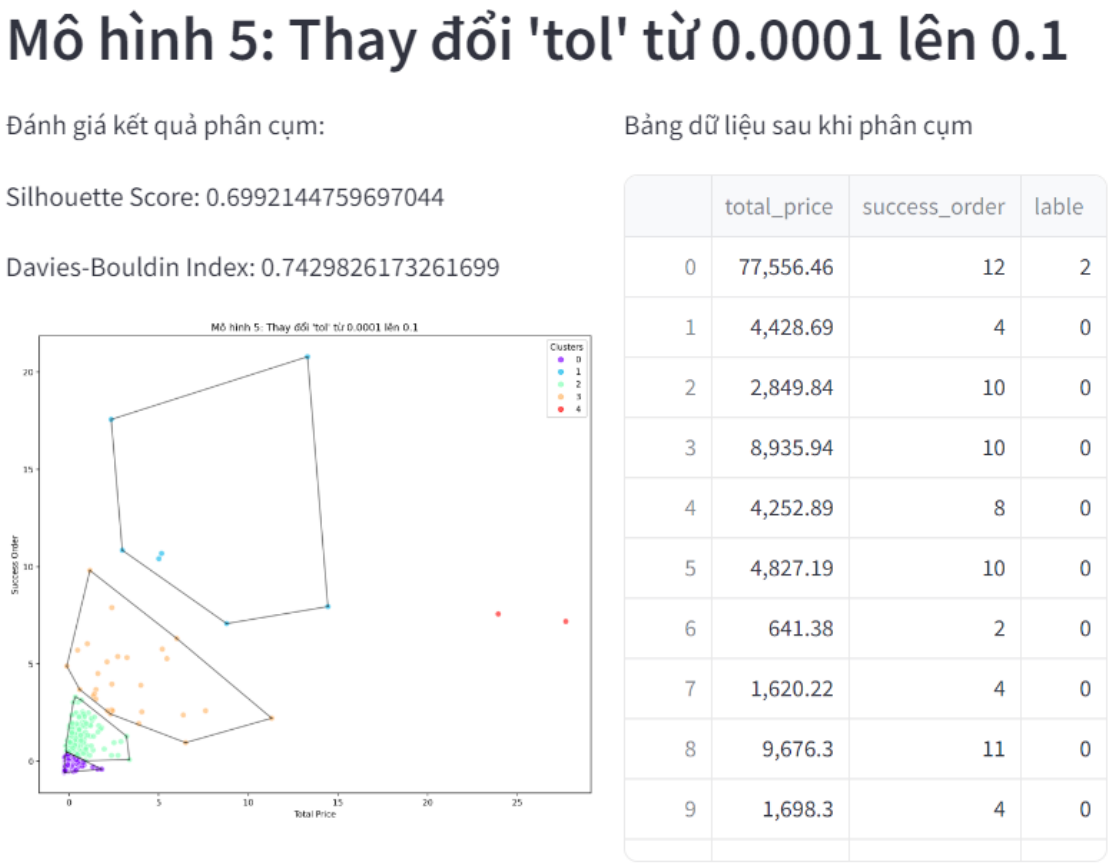
Có thể thấy phương pháp này cho kết quả không tối ưu hơn, vì có thể rơi vào trường hợp khởi tạo ban đầu không tốt. Với Silhouette Score = 0.59 và Davies-Bouldin Index = 0.76.

Mô hình 5, chúng em sẽ giới hạn điều kiện kết thúc bằng cách thay đổi tham số 'tol' từ mặc định lên 0.1 và đánh giá kết quả.

```
# Model 5: Change 'tol' from 0.0001 to 0.1
kmeans = KMeans(n_clusters= 5, init='random',
    max_iter=100, n_init=10, tol= 0.1, random_state
    =42)
y_kmeans = kmeans.fit_predict(X_scaled)
# Add clustering results column to DataFrame
df['cluster'] = y_kmeans
# Evaluate the model
print('\nEvaluate clustering results:')
silhouette_avg = silhouette_score(X_scaled, df['
    cluster'])
davies_bouldin = davies_bouldin_score(X_scaled, df['
```

```
    cluster'])
print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {davies_bouldin}")
```

Listing 4.19: "Model 5: Change 'tol' from 0.0001 to 0.1"



	total_price	success_order	lable
0	77,556.46	12	2
1	4,428.69	4	0
2	2,849.84	10	0
3	8,935.94	10	0
4	4,252.89	8	0
5	4,827.19	10	0
6	641.38	2	0
7	1,620.22	4	0
8	9,676.3	11	0
9	1,698.3	4	0

Hình 4.19: Đánh giá mô hình 5

Có thể hiểu khi giá trị của "tol" là 0.1, nghĩa là thuật toán sẽ dừng lại khi sự thay đổi giữa các lần cập nhật centroids (có thể là khoảng cách Euclidean) nhỏ hơn hoặc bằng 0.1. Điều này cho thấy rằng thuật toán không cần phải tiếp tục tối ưu hóa nếu sự thay đổi trong quá trình cập nhật centroids đã đạt được một mức chấp nhận được. Với kết quả thu được, khi điều chỉnh tol đã cho Silhouette Score là 0.699, đây là kết quả khá tốt cho thuật toán phân cụm.

4.5 Phân cụm khách hàng bằng mô hình phân cụm tổng hợp

Trong phần này, chúng em sẽ đưa ra mô hình phân cụm khách hàng với phương pháp phân cụm tổng hợp, một kỹ thuật mạnh mẽ cho việc nhóm các khách hàng dựa trên các đặc điểm chung của họ. Phân cụm tổng hợp không chỉ giúp chúng ta nhận biết, phân loại khách hàng một cách hiệu quả, mà còn cung cấp các thông tin

chi tiết để xây dựng các chiến lược tiếp thị và dịch vụ cá nhân hóa.

```
X = df[['total_price', 'order_count']]
# Remove outliers
z_scores = np.abs(stats.zscore(X))
X = X[(z_scores < 3).all(axis=1)]
# Standardized data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Listing 4.20: "Loại bỏ các điểm ngoại lai và chuẩn hóa dữ liệu"

```
# func evaluate clustering
def evaluate_clustering(X, labels):
    silhouette_avg = silhouette_score(X, labels)
    return silhouette_avg
# func plot cluster
def plot_clusters(X, labels, title):
    fig, ax = plt.subplots(figsize=(12, 10))
    scatter = ax.scatter(X[:, 0], X[:, 1], c=labels,
       cmap='rainbow', alpha=0.6, edgecolors='w', s
       =50)

    unique_labels = np.unique(labels)
    for label in unique_labels:
        points = X[labels == label]
        if len(points) >= 3:
            hull = ConvexHull(points)
            for simplex in hull.simplices:
                ax.plot(points[simplex, 0], points[
                   simplex, 1], 'k-', alpha=0.5)

    legend1 = ax.legend(*scatter.legend_elements(),
       title="Clusters")
    ax.add_artist(legend1)
    ax.set_title(title)
    ax.set_xlabel('Total Price')
    ax.set_ylabel('Order Count')
    plt.show()
```

```
# Display data after clustering
def data_after(lable):
    label = pd.DataFrame(lable, columns=['label'])
    df = pd.concat([data, label], axis = 1)
    df = df[["Customer ID", "order_count",      "
       total_price    avg_date",      "recency", "
       label"]]
    print("Data table after clustering")
    print(df)
```

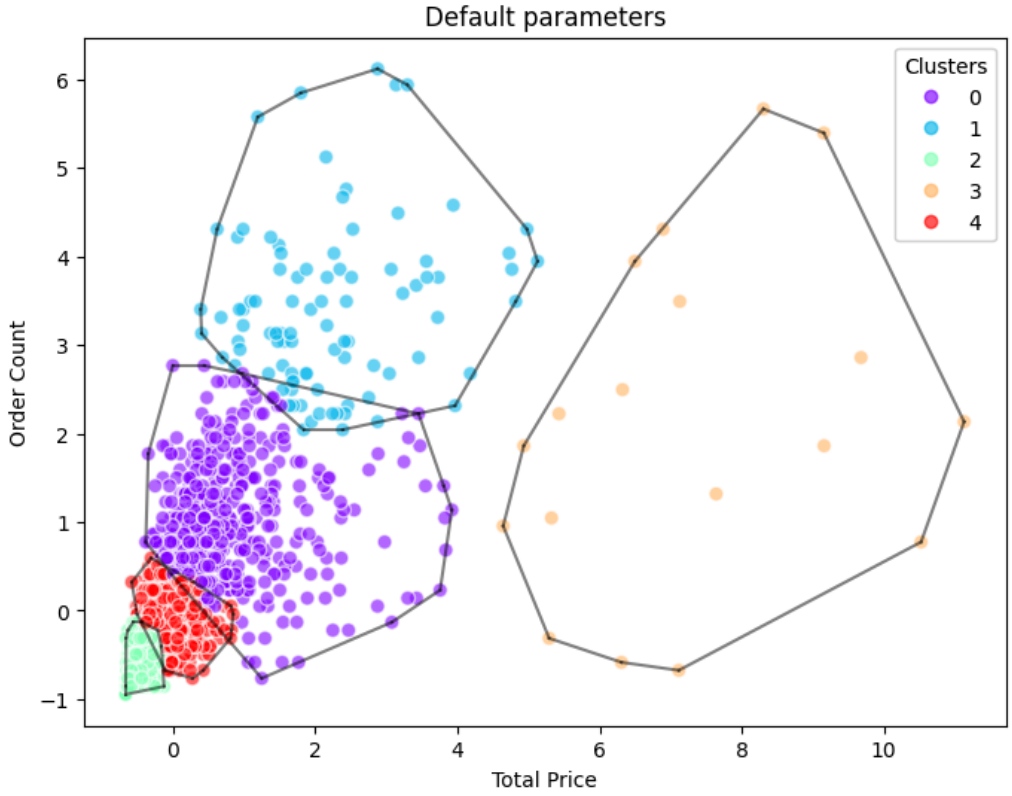
Listing 4.21: "Các hàm cần thiết"

Trong mô hình 1, chúng em sẽ sử dụng các tham số đầu vào mặc định của mô hình và đặt số lượng cụm **n_clusters = 5**. Dữ liệu đầu vào dùng để huấn luyện mô hình bao gồm dữ liệu về số lượng đơn hàng đã giao dịch và tổng doanh thu đối với từng khách hàng. Sau khi mô hình được huấn luyện sẽ tiến hành phân cụm và áp dụng mô hình đã huấn luyện để tìm mức độ phân cấp của từng khách hàng. Sau đó thực hiện các bước đánh giá mô hình bằng chỉ số silhouette và biểu đồ phân cụm dữ liệu.

```
print("Model 1: Default parameters")
model1 = AgglomerativeClustering(n_clusters=5)
labels1 = model1.fit_predict(X_scaled)
silhouette_avg1= evaluate_clustering(X_scaled,
   labels1)
print(f'Silhouette Score: {silhouette_avg1}')
data_after(labels1)
plot_clusters(X_scaled, labels1, "Default parameters"
   )
```

Listing 4.22: "Model 1"

```
Model 1: Default parameters
Silhouette Score: 0.44171071388795846
Data table after clustering
   Customer ID  order_count  total_price  avg_date  recency  label
0      12346.0           17     77556.46      24.0      321    4.0
1      12349.0            5      4428.69     143.0       14    4.0
2      12352.0           13      2849.84      27.0       32    0.0
3      12359.0           14      8935.94      52.0        3    4.0
4      12360.0            9      4252.89      67.0       48    4.0
```



Hình 4.20: Mô hình 1

Từ biểu đồ phân cụm và chỉ số Silhouette Score = 0.4417 có thể thấy các cụm được phân bố không hoàn toàn độc lập và các điểm dữ liệu trong cụm chưa thực sự tối ưu. Mô hình cần phải cải thiện thêm để kết quả phân cụm được tốt hơn.

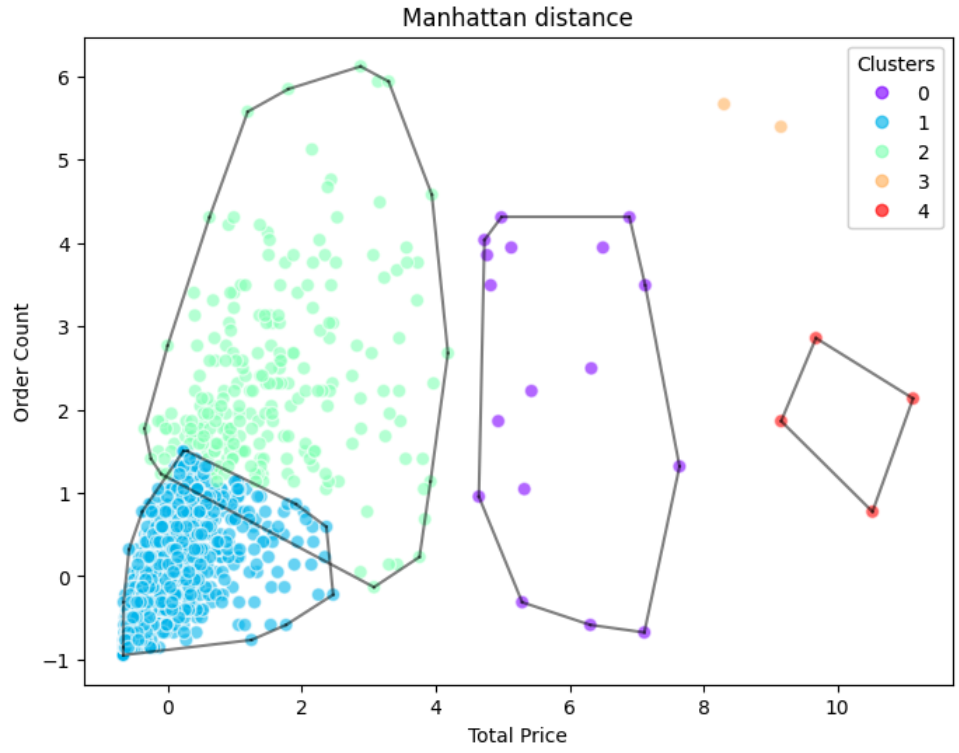
Trong mô hình này số lượng cụm không đổi **n_clusters = 5** nhưng phương thức tính khoảng cách giữa các nút được đổi sang thành **manhattan**. Khoảng cách manhattan giữa 2 điểm **p** = (p_1,p_2,....,p_n và **q** = (q_1,q_2,....,q_n được xác định bằng công thức:

$$d_{(\mathrm{p,q})} = \sum_{i=1}^{n} |p_i - q_i|$$

```
print("\nModel 2: Manhattan distance")
model2 = AgglomerativeClustering(n_clusters=5,
    affinity='manhattan', linkage='average')
labels2 = model2.fit_predict(X_scaled)
silhouette_avg2 = evaluate_clustering(X_scaled,
    labels2)
print(f'Silhouette Score: {silhouette_avg2}')
data_after(labels2)
plot_clusters(X_scaled, labels2, "Manhattan distance"
    )
```

Listing 4.23: "Model 2"

```
Model 2: Manhattan distance
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning: Attribute
  warnings.warn(
Silhouette Score: 0.6956214429389687
Data table after clustering
   Customer ID  order_count  total_price  avg_date  recency  label
0      12346.0           17     77556.46      24.0      321    1.0
1      12349.0            5      4428.69     143.0       14    1.0
2      12352.0           13      2849.84      27.0       32    1.0
3      12359.0           14      8935.94      52.0        3    1.0
4      12360.0            9      4252.89      67.0       48    1.0
```



Hình 4.21: Mô hình 2

Từ biểu đồ phân cụm và chỉ số Silhouette Score = 0.6956 có thể mô hình đưa ra

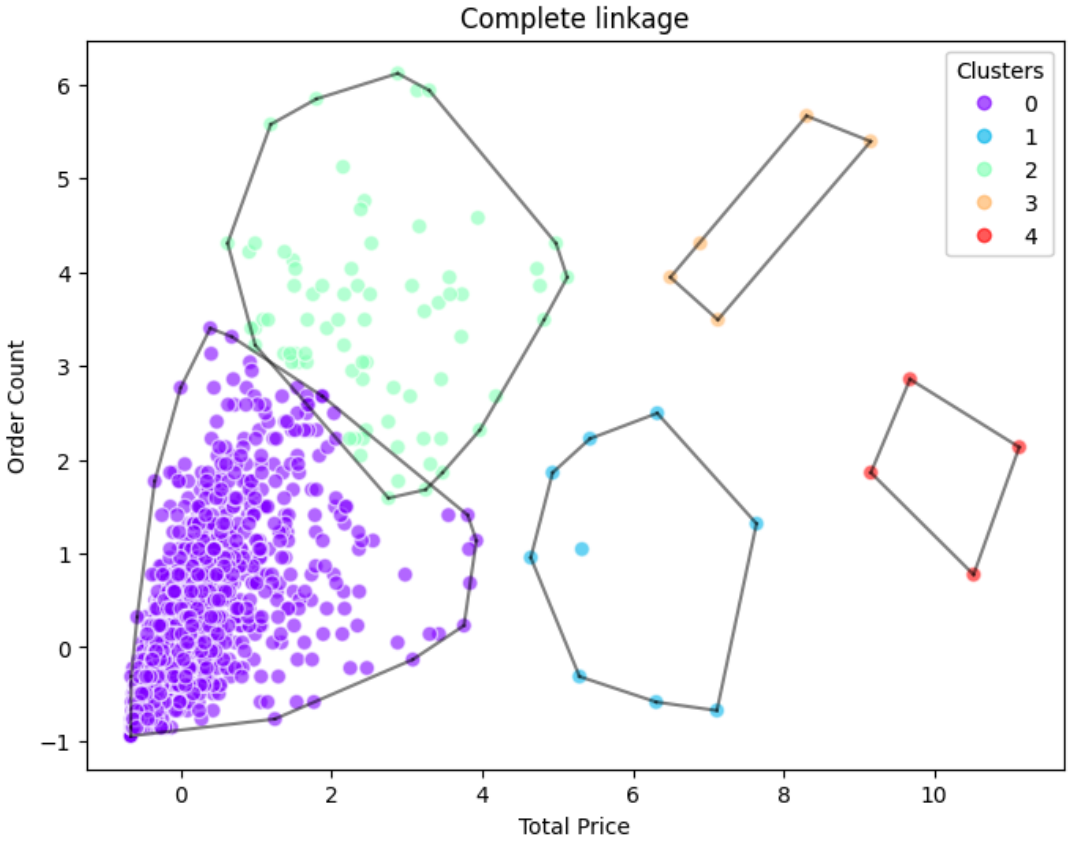
các phân cụm tương đối tốt và có sự phân tách rõ ràng giữa các cụm, tuy nhiên vẫn cần cải thiện thêm để có mô hình tốt hơn.

Mô hình 3 sẽ sử dụng tiêu chí hợp nhất cụm **linkage = 'complete'** đã được đề cập ở phần trên để thay đổi tiêu chí hợp nhất. Tiêu chí này đảm bảo rằng các cụm hợp nhất sẽ có khoảng cách lớn nhất có thể giữa các điểm trong hai cụm con. Tiêu chí này thường tạo ra các cụm đồng đều và ít bị ảnh hưởng bởi nhiễu.

```
print("\nModel 3: Complete linkage")
model3 = AgglomerativeClustering(n_clusters=5,
    linkage='complete')
labels3 = model3.fit_predict(X_scaled)
silhouette_avg3 = evaluate_clustering(X_scaled,
    labels3)
print(f'Silhouette Score: {silhouette_avg3}')
data_after(labels3)
plot_clusters(X_scaled, labels3, "Complete linkage")
```

Listing 4.24: "Model 3"

```
Model 3: Complete linkage
Silhouette Score: 0.7373135445185246
Data table after clustering
   Customer ID  order_count  total_price  avg_date  recency  label
0      12346.0           17     77556.46      24.0      321    0.0
1      12349.0            5      4428.69     143.0       14    0.0
2      12352.0           13      2849.84      27.0       32    0.0
3      12359.0           14      8935.94      52.0        3    0.0
4      12360.0            9      4252.89      67.0       48    0.0
```



Hình 4.22: Mô hình 3

Sau khi thực hiện huấn luyện mô hình, biểu đồ phân tán và chỉ số Silhouette Score = 0.7373 cho thấy mô hình hoạt động tương đối tốt với chỉ số Silhouette ≈ 1. Mô hình này có thể được áp dụng để phân cụm đối với bộ dữ liệu này.

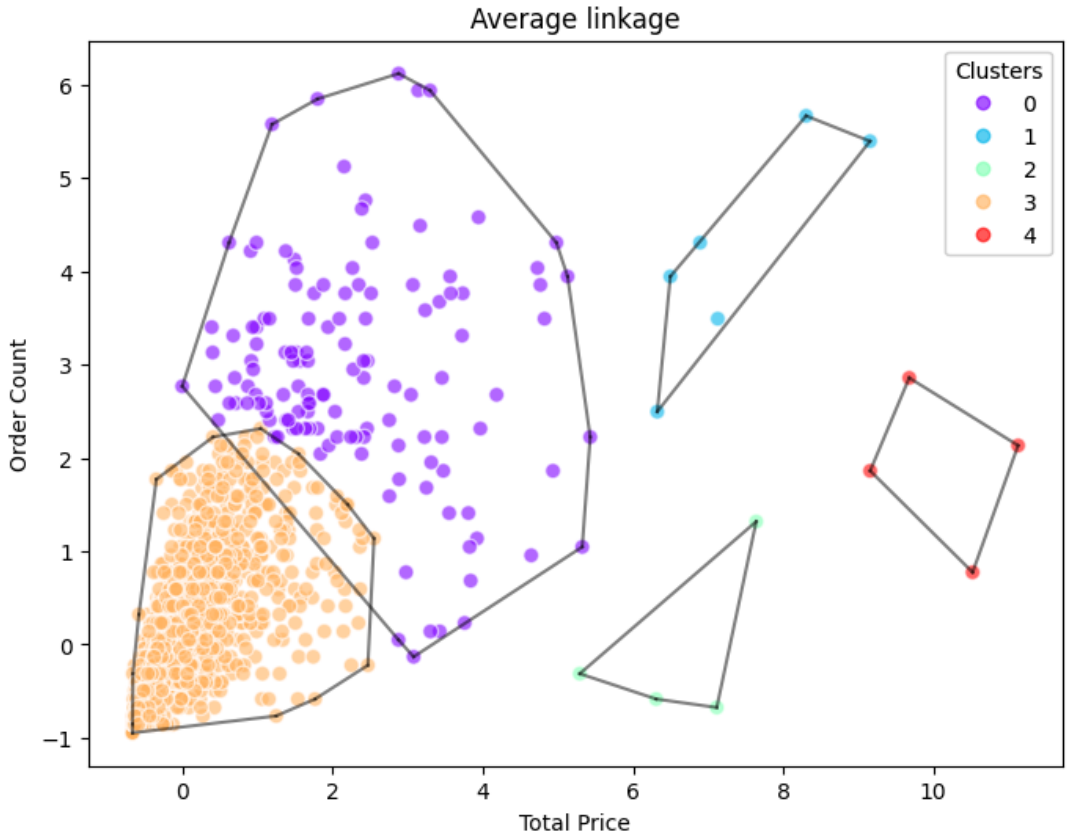
Trong mô hình này, tiêu chí hợp nhất cụm **linkage = 'average'** được sử dụng. Khoảng cách giữa hai cụm sẽ được tính bằng trung bình khoảng cách giữa tất cả các cặp điểm thuộc hai cụm khác nhau.

```
print("\nModel 4: Average linkage")
model4 = AgglomerativeClustering(n_clusters=5,
    linkage='average')
labels4 = model4.fit_predict(X_scaled)
silhouette_avg4 = evaluate_clustering(X_scaled,
    labels4)
```

```
print(f'Silhouette Score: {silhouette_avg4}')
data_after(labels4)
plot_clusters(X_scaled, labels4, "Average linkage")
```

Listing 4.25: "Model 4"

```
Model 4: Average linkage
Silhouette Score: 0.7320605176411057
Data table after clustering
   Customer ID  order_count  total_price  avg_date  recency  label
0      12346.0           17     77556.46      24.0      321    3.0
1      12349.0            5      4428.69     143.0       14    3.0
2      12352.0           13      2849.84      27.0       32    3.0
3      12359.0           14      8935.94      52.0        3    3.0
4      12360.0            9      4252.89      67.0       48    3.0
```



Hình 4.23: Mô hình 4

Tương tự như mô hình trên,kết quả đạt được của mô hình này là tương đối tốt và có thể áp dụng nếu không tìm được giải pháp tối ưu mô hình. Mô hình 5 sử dụng phương thức tính khoảng cách **cosine** và tiêu chí hợp nhất cụm **linkage = 'average'**.

Khoảng cách cosine giữa hai điểm p và q được xác định bằng công thức:

$$d(p,q) = \cos(\theta) = \frac{p \cdot q}{||p||||q||}$$

Trong đó:

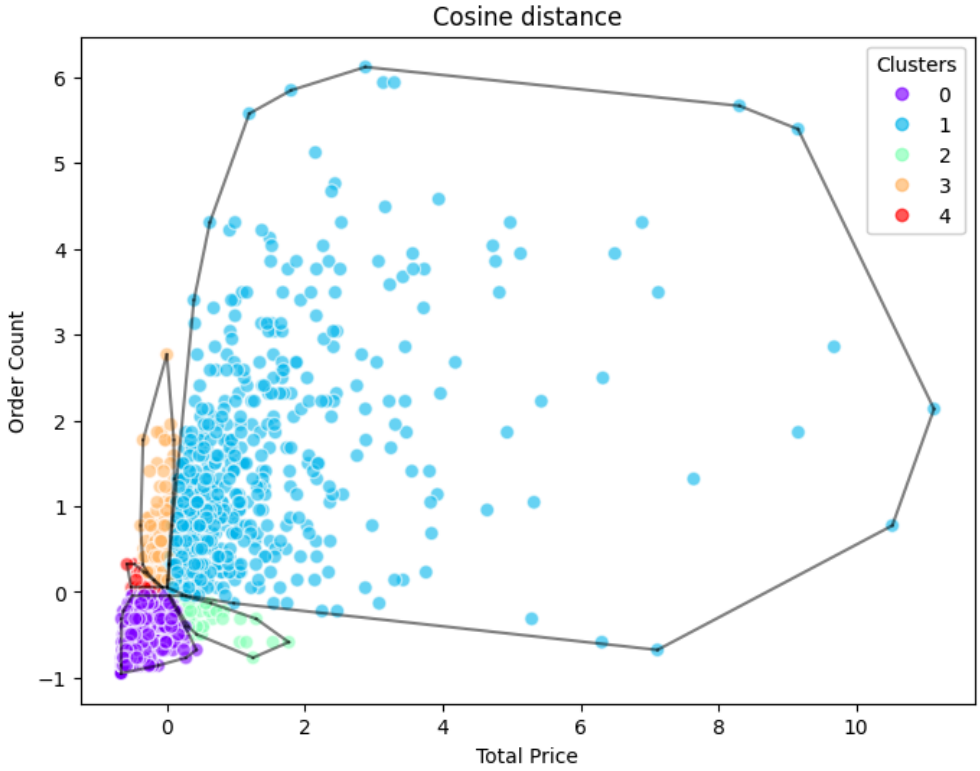
p · q : Tích vô hướng của hai vector p và q.

||p|| và ||q|| : Là độ dài của vector p và q tương ứng.

```
print("\nModel 5: Cosine distance")
model5 = AgglomerativeClustering(n_clusters=5,
   affinity='cosine', linkage='average')
labels5 = model5.fit_predict(X_scaled)
silhouette_avg5, ch_score5 = evaluate_clustering(
   X_scaled, labels5)
print(f'Silhouette Score: {silhouette_avg5}, Calinski
   -Harabasz Index: {ch_score5}')
data_after(labels5)
plot_clusters(X_scaled, labels5, "Cosine distance")
```

Listing 4.26: "Model 5"

```
Model 5: Cosine distance
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning: Attribu
  warnings.warn(
Silhouette Score: 0.2054359304404339
Data table after clustering
   Customer ID  order_count  total_price  avg_date  recency  label
0      12346.0           17     77556.46      24.0      321    0.0
1      12349.0            5      4428.69     143.0       14    3.0
2      12352.0           13      2849.84      27.0       32    1.0
3      12359.0           14      8935.94      52.0        3    0.0
4      12360.0            9      4252.89      67.0       48    1.0
```



Hình 4.24: Mô hình 5

Mô hình này có điểm Silhouette = 0.2054 ≈ 0 cho thấy mô hình này cho ra các cụm chồng lấn nhau và mô hình phân cụm này là không tốt.

CHƯƠNG 5. GIAO DIỆN THỰC HIỆN CHƯƠNG TRÌNH

Sau khi thực hiện xây dựng các mô hình, chúng em sẽ đi xây dựng giao diện web cho chương trình bằng thư viện *streamlit* của python. Giao diện thực hiện chương trình sẽ được đưa ra ở bên dưới và mã nguồn của chương trình sẽ được đưa ra ở đây

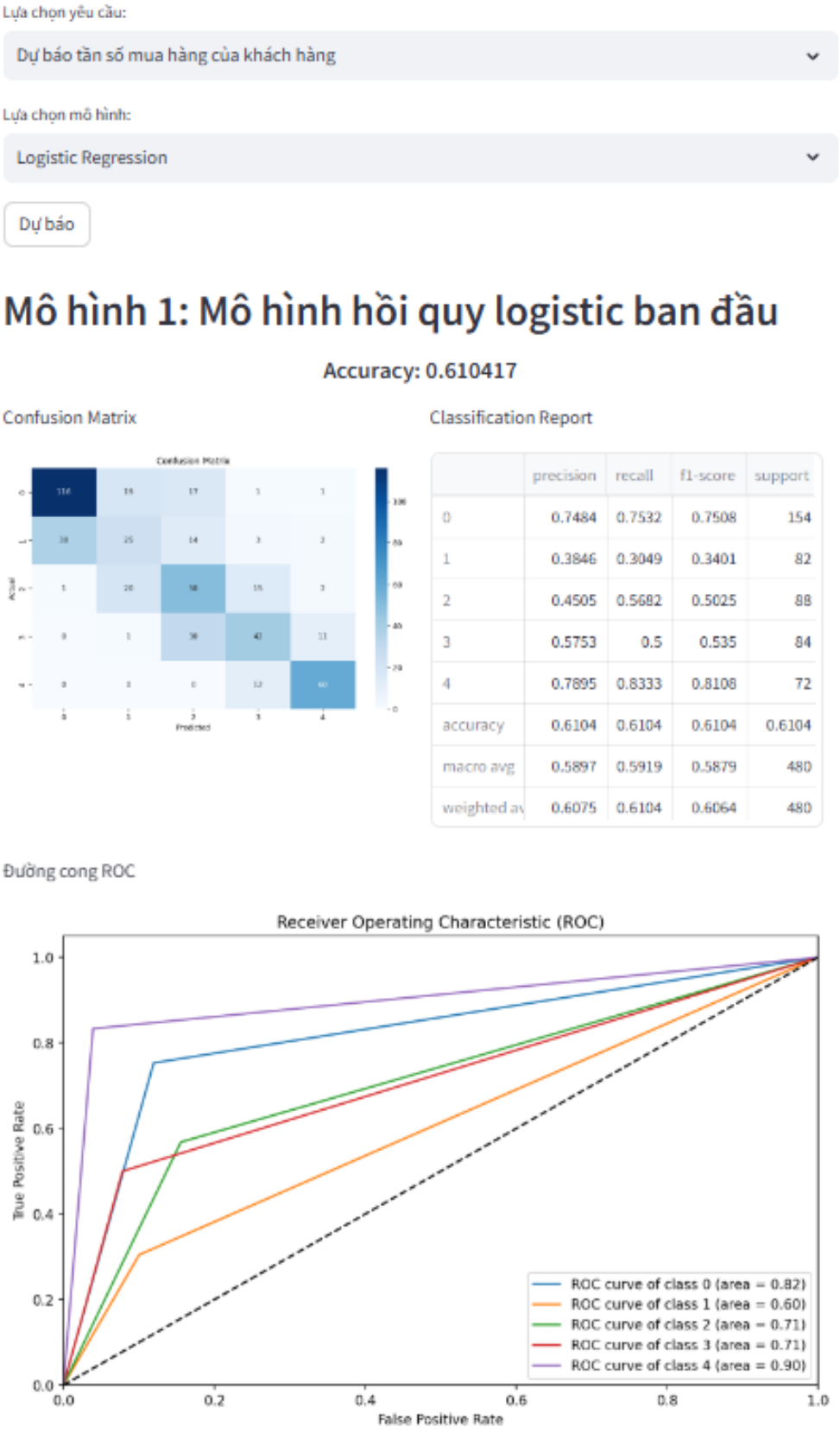
Giao diện ban đầu.

Hình 5.1: Giao diện chương trình

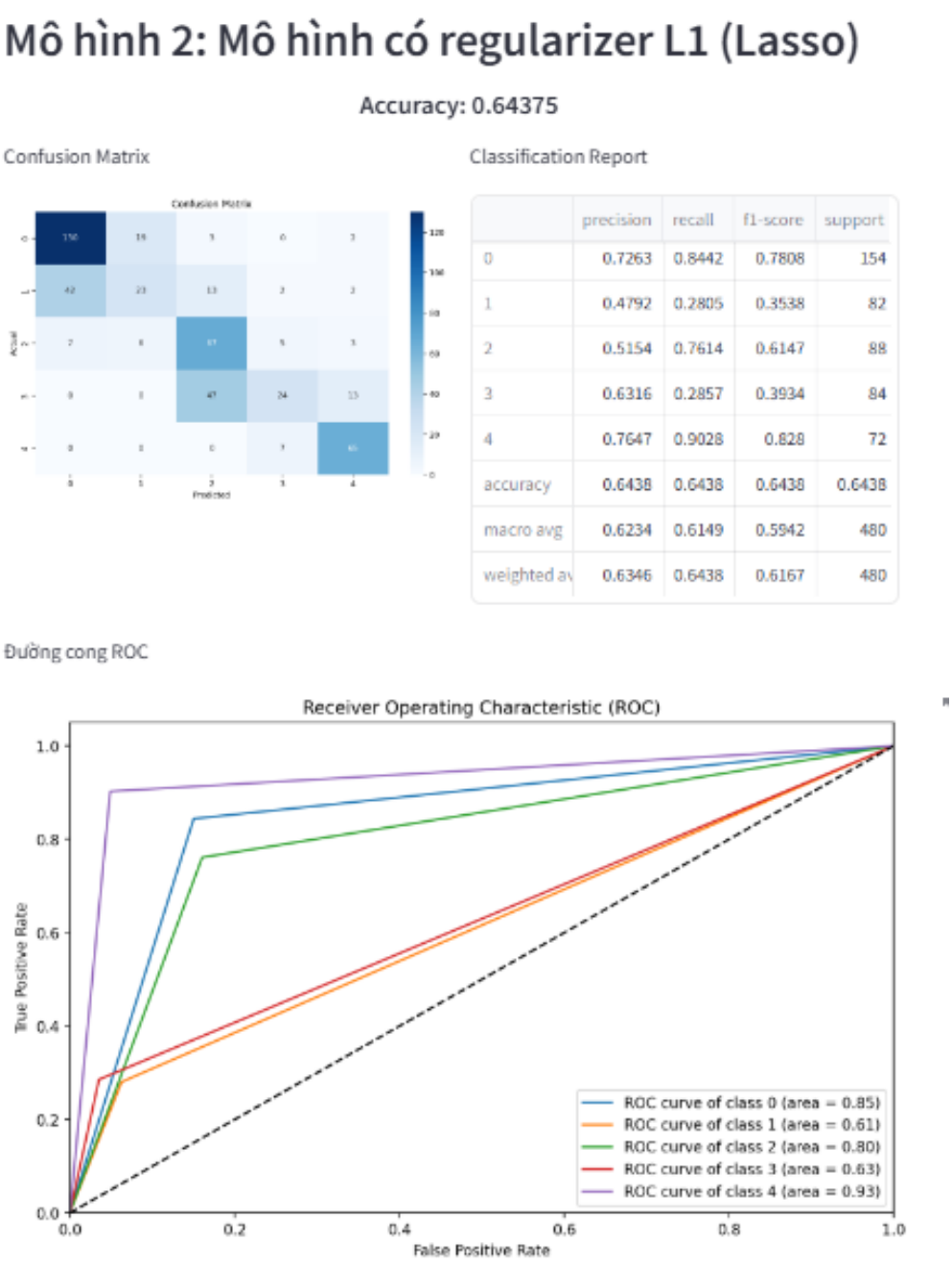
Giao diện thống kê dữ liệu.

Hình 5.2: Giao diện thống kê dữ liệu

Giao diện mô hình hồi quy Logistic.



Hình 5.3: Giao diện mô hình hồi quy Logistic



Hình 5.4: Giao diện mô hình hồi quy Logistic

Mô hình 4: Mô hình với tỷ lệ thay đổi weight class (class weight balancing)

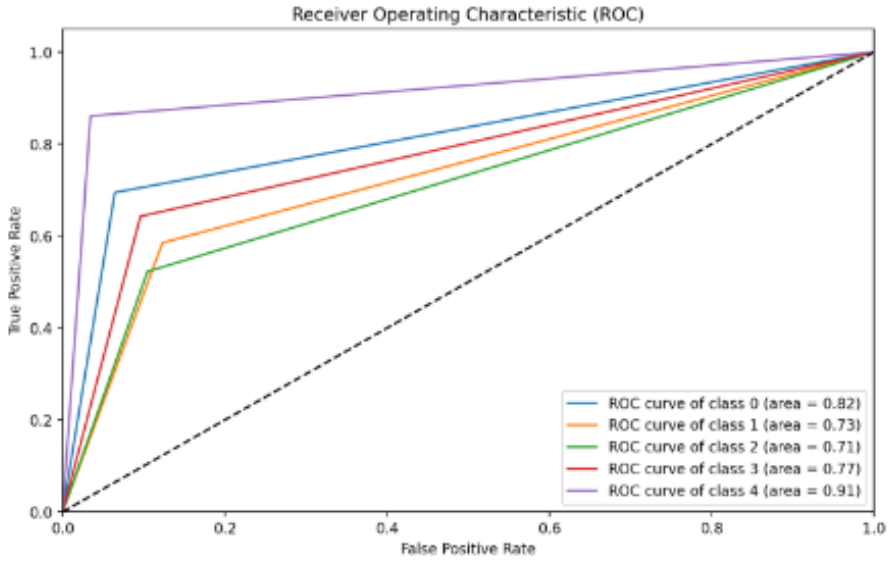
Accuracy: 0.660417

Confusion Matrix

Classification Report

	precision	recall	f1-score	support
0	0.8359	0.6948	0.7589	154
1	0.4948	0.5854	0.5363	82
2	0.5287	0.5227	0.5257	88
3	0.587	0.6429	0.6136	84
4	0.8158	0.8611	0.8378	72
accuracy	0.6604	0.6604	0.6604	0.6604
macro avg	0.6525	0.6614	0.6545	480
weighted av	0.6748	0.6604	0.6645	480

Đường cong ROC



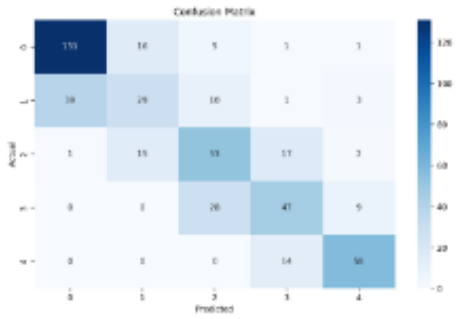
Mô hình 5: Mô hình với giới hạn số lần lặp

Hình 5.5: Giao diện mô hình hồi quy Logistic

Mô hình 5: Mô hình với giới hạn số lần lặp (maximum iterations)

Accuracy: 0.6625

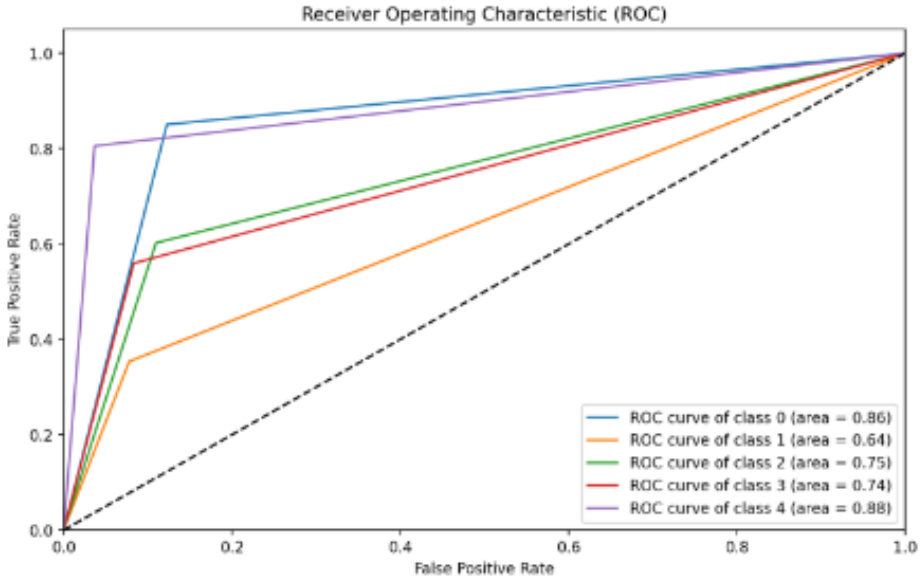
Confusion Matrix



Classification Report

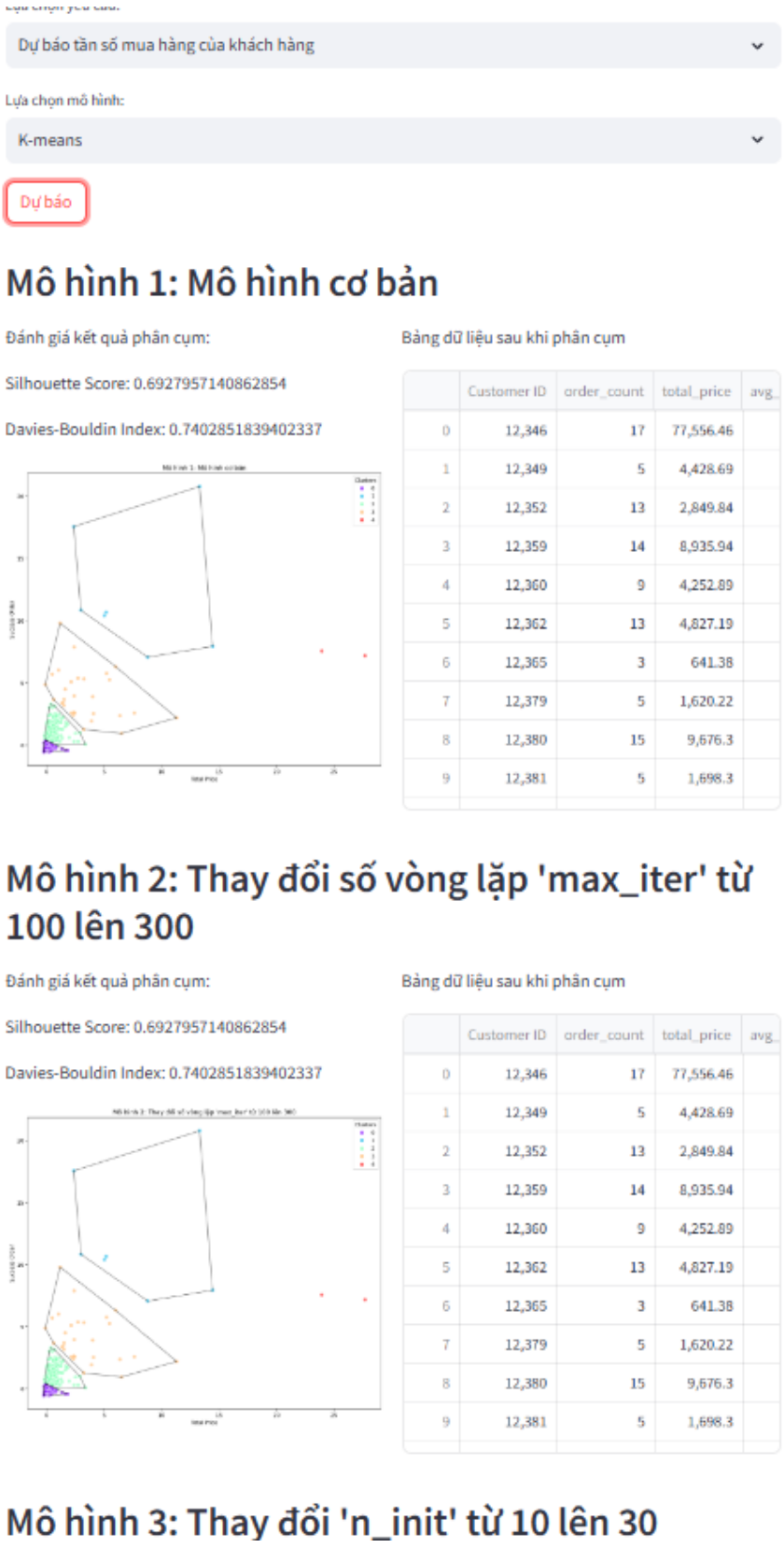
	precision	recall	f1-score	support
0	0.7661	0.8506	0.8062	154
1	0.4833	0.3537	0.4085	82
2	0.5521	0.6023	0.5761	88
3	0.5875	0.5595	0.5732	84
4	0.7945	0.8056	0.8	72
accuracy	0.6625	0.6625	0.6625	0.6625
macro avg	0.6367	0.6343	0.6328	480
weighted av	0.6516	0.6625	0.6543	480

Đường cong ROC



Hình 5.6: Giao diện mô hình hồi quy Logistic

Giao diện mô hình K-Means.



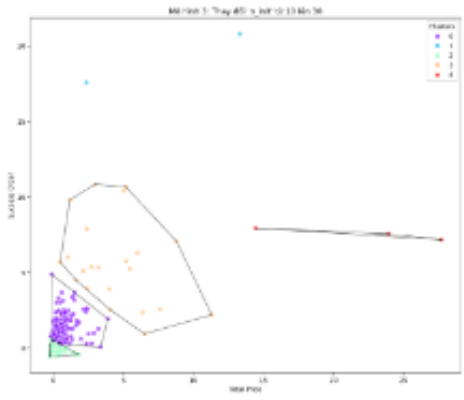
Hình 5.7: Giao diện mô hình K-Means

Mô hình 3: Thay đổi 'n_init' từ 10 lên 30

Đánh giá kết quả phân cụm:

Silhouette Score: 0.7197316180105007

Davies-Bouldin Index: 0.6556311251588427



Bảng dữ liệu sau khi phân cụm

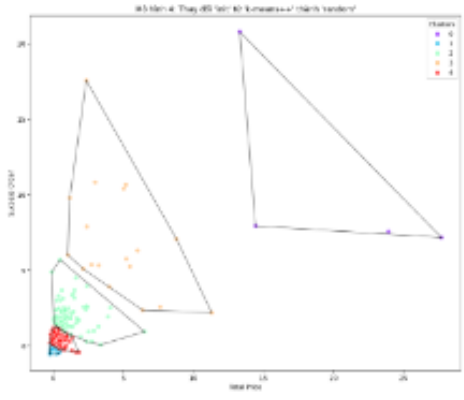
	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

Mô hình 4: Thay đổi 'init' từ 'k-means++' thành 'random'

Đánh giá kết quả phân cụm:

Silhouette Score: 0.5984964239869235

Davies-Bouldin Index: 0.7613655811030795



Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

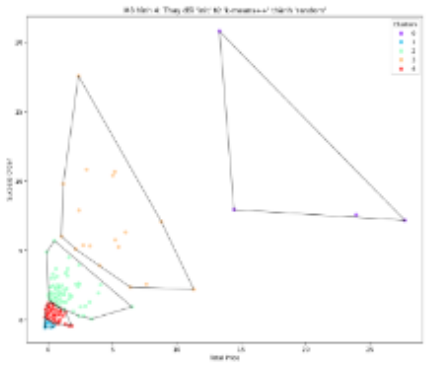
Hình 5.8: Giao diện mô hình K-Means

Mô hình 4: Thay đổi 'init' từ 'k-means++' thành 'random'

Đánh giá kết quả phân cụm:

Silhouette Score: 0.5984964239869235

Davies-Bouldin Index: 0.7613655811030795



Bảng dữ liệu sau khi phân cụm

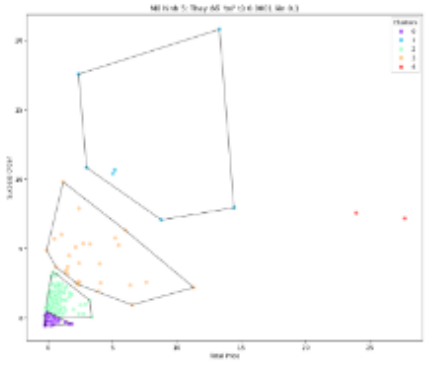
	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

Mô hình 5: Thay đổi 'tol' từ 0.0001 lên 0.1

Đánh giá kết quả phân cụm:

Silhouette Score: 0.6992144759697044

Davies-Bouldin Index: 0.7429826173261699

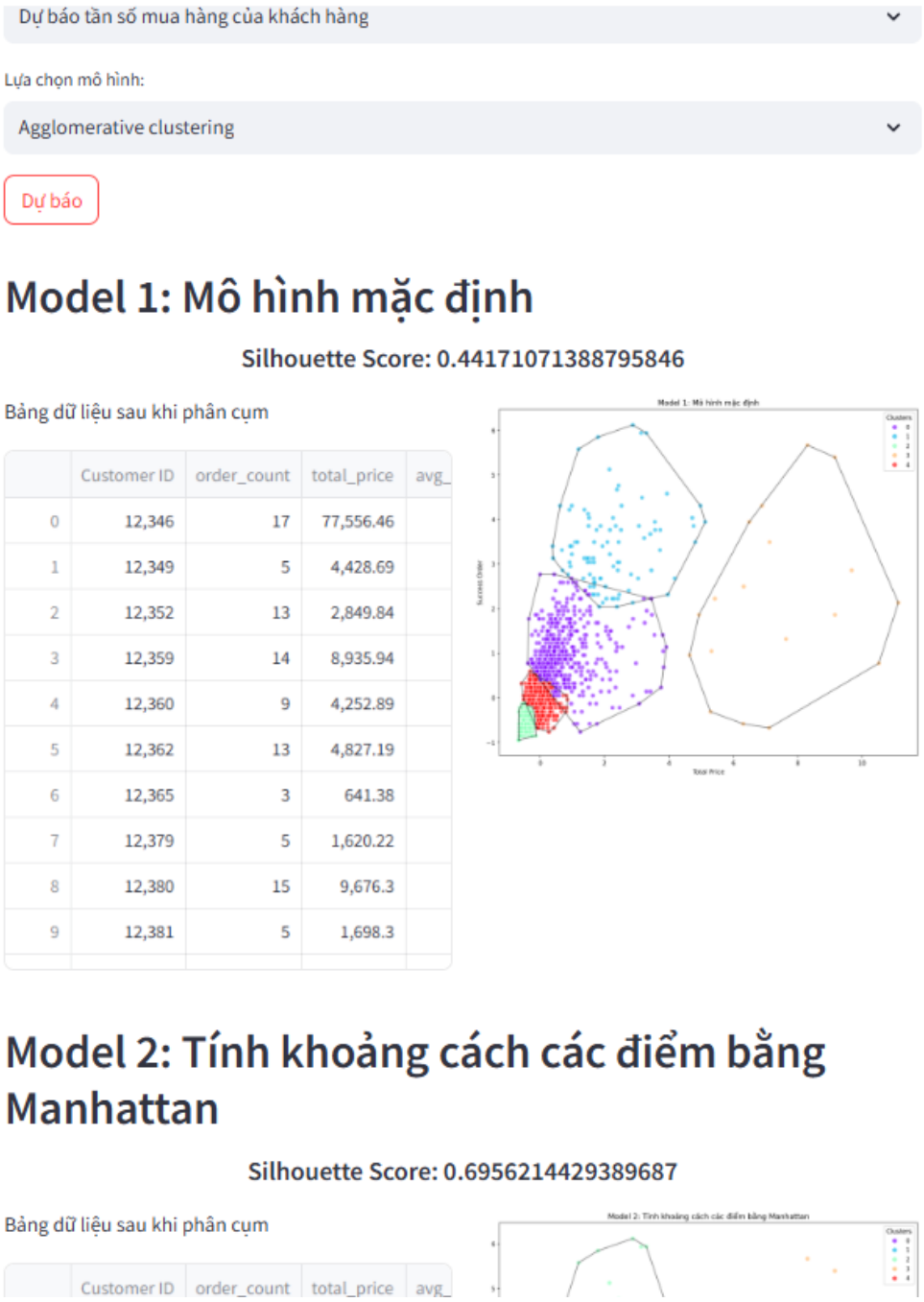


Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

Hình 5.9: Giao diện mô hình K-Means

Giao diện mô hình Agglomerative clustering.



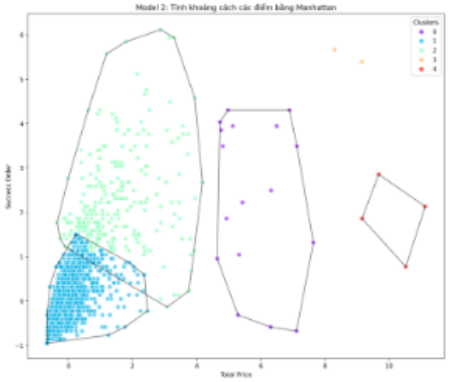
Hình 5.10: Giao diện mô hình Agglomerative clustering

Model 2: Tính khoảng cách các điểm bằng Manhattan

Silhouette Score: 0.6956214429389687

Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

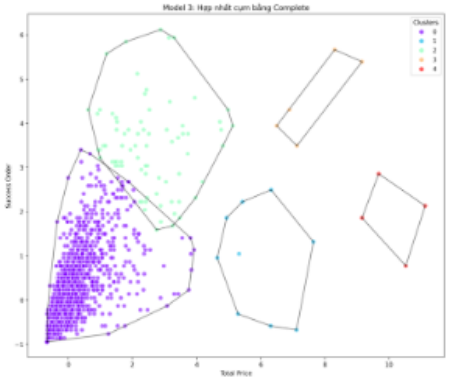


Model 3: Hợp nhất cụm bằng Complete

Silhouette Score: 0.7373135445185246

Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	



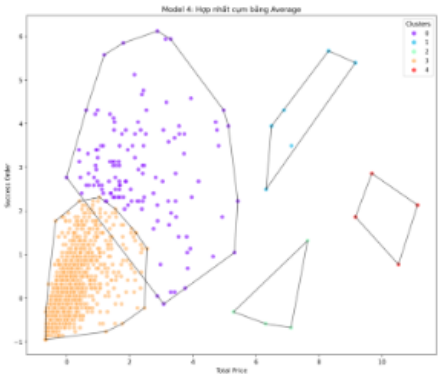
Hình 5.11: Giao diện mô hình Agglomerative clustering

Model 4: Hợp nhất cụm bằng Average

Silhouette Score: 0.7320605176411057

Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	
6	12,365	3	641.38	
7	12,379	5	1,620.22	
8	12,380	15	9,676.3	
9	12,381	5	1,698.3	

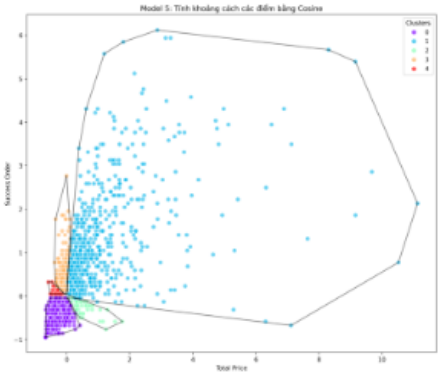


Model 5: Tính khoảng cách các điểm bằng Cosine

Silhouette Score: 0.2054359304404339

Bảng dữ liệu sau khi phân cụm

	Customer ID	order_count	total_price	avg_
0	12,346	17	77,556.46	
1	12,349	5	4,428.69	
2	12,352	13	2,849.84	
3	12,359	14	8,935.94	
4	12,360	9	4,252.89	
5	12,362	13	4,827.19	



Hình 5.12: Giao diện mô hình Agglomerative clustering

CHƯƠNG 6. KẾT LUẬN

Trong bài báo cáo này, nhóm chúng em đã nghiên cứu và áp dụng ba phương pháp phân cụm khách hàng: hồi quy logistic, K-means và Agglomerative Clustering. Mỗi phương pháp đều có những ưu điểm và hạn chế riêng, nhằm mục đích phân loại khách hàng vào các nhóm có tính chất tương đồng.

Đầu tiên, phương pháp hồi quy logistic đã được áp dụng để phân lớp khách hàng dựa trên các đặc trưng quan sát được và những phân lớp được tạo sẵn. Kết quả cho thấy mô hình này có khả năng mô tả rõ ràng mối quan hệ giữa các biến đầu vào và xác suất xảy ra của sự kiện cụ thể.

Tiếp theo, chúng em đã sử dụng phương pháp K-means để phân cụm khách hàng thành các nhóm dựa trên đặc những thông tin về lịch sử giao dịch. K-means giúp xác định được các nhóm khách hàng có hành vi mua hàng tương đồng, từ đó cung cấp thông tin hữu ích cho các chiến lược tiếp thị và phục vụ khách hàng.

Cuối cùng, phương pháp Agglomerative Clustering cho phép tạo ra các nhóm khách hàng dựa trên sự tương tự về mặt dữ liệu, trong đó các cá thể trong cùng một nhóm có đặc tính gần nhau nhất. Kết quả này giúp chúng em hiểu sâu hơn về cách khách hàng có thể được nhóm lại theo các tiêu chí khác nhau.

Tóm lại, việc áp dụng các mô hình phân cụm khách hàng như hồi quy logistic, K-means và Agglomerative Clustering đã mang lại những thông tin quý giá về hành vi của khách hàng và mối quan hệ giữa các nhóm khách hàng trong ngữ cảnh kinh doanh. Các kết quả này có thể hỗ trợ cho các quyết định chiến lược và tối ưu hóa các chiến dịch tiếp thị trong tương lai.

TÀI LIỆU THAM KHẢO

1. Introduction to K-Means Clustering Algorithm (analyticsvidhya.com)
2. [ML] Phân loại với hồi quy Logistic - Greenwich Vietnam
3. AgglomerativeClustering — scikit-learn 1.5.0 documentation
4. Streamlit - A faster way to build and share data apps