VIETNAM NATIONAL UNIVERSITY - HCM
Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

MACHINE LEARNING

BÀI TẬP LỚN
Online Shoppers' Intention Prediction

GVHD: TS Võ Thanh Hùng
SV thực hiện: Phạm Lê Thanh – 2112273

TP. Hồ Chí Minh, tháng 03/2024

Mục lục

1 Giới thiệu đề tài

1.1 Tên đề tài

Dự đoán ý định mua hàng của người dùng truy cập trang web thương mại điện tử *(Predicting Online Shoppers' Purchasing Intention)*

1.2 Bối cảnh và động lực

Trong thời đại bùng nổ của thương mại điện tử, hành vi của người tiêu dùng trên các nền tảng mua sắm trực tuyến ngày càng trở nên phức tạp và khó đoán. Một lượng lớn người dùng ghé thăm các trang web bán hàng mỗi ngày, nhưng chỉ một phần nhỏ trong số đó thực sự thực hiện hành vi mua hàng. Việc hiểu rõ và dự đoán ý định mua hàng của người dùng không chỉ giúp các doanh nghiệp tối ưu hóa trải nghiệm người dùng mà còn tăng tỷ lệ chuyển đổi (conversion rate), từ đó nâng cao doanh thu.

Tuy nhiên, việc phân biệt giữa khách hàng tiềm năng (có khả năng mua hàng) và người dùng chỉ "lướt qua" là một thách thức lớn nếu chỉ dựa vào cảm quan hoặc các phương pháp truyền thống. Do đó, áp dụng các kỹ thuật học máy (Machine Learning – ML) để phân tích hành vi truy cập và dự đoán hành vi mua hàng đang trở thành một hướng tiếp cận hiệu quả và mang tính ứng dụng cao.

1.3 Mục tiêu của đề tài

Mục tiêu chính của đề tài là xây dựng một mô hình học máy có khả năng dự đoán xem người dùng có thực hiện hành vi mua hàng hay không, dựa trên các đặc trưng hành vi được thu thập trong quá trình truy cập trang web.

Cụ thể, đề tài hướng đến:

- Phân tích và hiểu rõ cấu trúc, đặc điểm của tập dữ liệu hành vi người dùng.
- Ứng dụng các thuật toán học máy (Logistic Regression, Decision Tree, Random Forest, ...) để xây dựng mô hình dự đoán.
- So sánh, đánh giá hiệu quả các mô hình thông qua các chỉ số như: Accuracy, Precision, Recall và F1-score.
- Gợi ý mô hình phù hợp nhất cho bài toán này nhằm hỗ trợ các doanh nghiệp thương mại điện tử ra quyết định nhanh và chính xác hơn trong việc xác định khách hàng tiềm năng.

1.4 Ý nghĩa thực tiễn

Việc có thể dự đoán sớm người dùng nào có khả năng mua hàng sẽ giúp các nền tảng thương mại điện tử:

- Tập trung nguồn lực quảng cáo, chăm sóc khách hàng đúng mục tiêu.
- Thiết kế giao diện hoặc nội dung phù hợp hơn cho từng nhóm người dùng.
- Tăng tỷ lệ chuyển đổi từ truy cập sang mua hàng thực tế.
- Cải thiện trải nghiệm người dùng và tối ưu hóa doanh thu.

Với các mô hình học máy hiện đại, việc dự đoán hành vi người dùng không còn chỉ dựa vào trực giác mà được hỗ trợ bởi các thuật toán đã được chứng minh về hiệu quả.

2 Tập dữ liệu

2.1 Dữ liệu

Tập dữ liệu sử dụng trong đề tài có tên là Online Shoppers Purchasing Intention Dataset, được thu thập từ hành vi người dùng trên một trang web thương mại điện tử trong khoảng thời gian từ tháng 5 năm 2016 đến tháng 12 năm 2016.

- Nguồn tham khảo: Tập dữ liệu được công bố trên UCI Machine Learning Repository.
- Định dạng: .csv (Comma-Separated Values).
- Số lượng mẫu: 12.330 dòng dữ liệu
- Số lượng thuộc tính: 18 đặc trưng đầu vào và 1 cột nhãn mục tiêu (Revenue).

2.2 Ý nghĩa của bài toán

Dựa trên các thuộc tính hành vi và thông tin truy cập của người dùng, bài toán đặt ra là dự đoán liệu người dùng đó có thực hiện hành vi mua hàng (Revenue = True) hay không.

Thuộc tính mục tiêu (Revenue) là một giá trị nhị phân:

- True: Người dùng đã hoàn tất một giao dịch mua hàng.
- False: Người dùng không thực hiện mua hàng.

2.3 Các thuộc tính trong tập dữ liệu

Tập dữ liệu gồm nhiều nhóm thuộc tính, mỗi nhóm phản ánh một khía cạnh khác nhau trong hành vi người dùng. Dưới đây là phân loại và mô tả ngắn gọn:

2.3.1 Nhóm hành vi truy cập (Sessions)

- Administrative, Administrative_Duration: Số lượng và tổng thời gian truy cập các trang quản trị.
- Informational, Informational_Duration: Truy cập các trang chứa thông tin (ví dụ: giới thiệu sản phẩm, chính sách).
- ProductRelated, ProductRelated_Duration: Số lần và thời gian xem các trang sản phẩm – rất quan trọng vì phản ánh ý định mua.

2.3.2 Nhóm tương tác kỹ thuật (Technical metrics)

- BounceRates: Tỷ lệ thoát – người dùng rời đi ngay sau khi vào.
- ExitRates: Tỷ lệ người dùng thoát khỏi trang hiện tại.
- PageValues: Giá trị trung bình của trang, tính dựa trên dữ liệu giao dịch.
- SpecialDay: Giá trị đặc biệt (gần các dịp lễ lớn – tăng khả năng mua hàng).

2.3.3 Thời gian và nguồn truy cập

- Month: Tháng truy cập (ví dụ: 'Feb', 'Mar', ...).
- OperatingSystems, Browser, Region, TrafficType: Thông tin thiết bị và cách người dùng đến website.

2.3.4 Nhóm hành vi cá nhân

- VisitorType: Loại người dùng (Returning_Visitor, New_Visitor, v.v.)
- Weekend: Truy cập vào cuối tuần hay không.

2.3.5 Mục tiêu

- Revenue: (Target) – người dùng có thực hiện hành vi mua hàng hay không.

3 Tiền xử lý dữ liệu

3.1 Tổng quan

Trong đề tài này, nhóm tiến hành xây dựng và đánh giá các mô hình học máy nhằm dự đoán ý định mua hàng (Revenue) của người dùng dựa trên các hành vi truy cập trang web thương mại điện tử. Việc lựa chọn mô hình phù hợp là yếu tố then chốt nhằm nâng cao khả năng dự đoán chính xác. Nhóm lựa chọn ba thuật toán học máy tiêu biểu để thực hiện mô hình hóa:

- Hồi quy Logistic (Logistic Regression)
- Cây quyết định (Decision Tree Classifier)
- Rừng ngẫu nhiên (Random Forest Classifier)

Các mô hình được xây dựng bằng thư viện Scikit-learn – một trong những thư viện phổ biến nhất cho học máy trong Python. Quy trình thực hiện được thống nhất với các bước chính sau:

- Tiền xử lý dữ liệu
- Chia dữ liệu thành tập huấn luyện và kiểm thử
- Huấn luyện mô hình
- Dự đoán và đánh giá mô hình
- Trực quan hóa kết quả

3.2 Tiền xử lý dữ liệu

Trước khi đưa vào huấn luyện mô hình, dữ liệu cần được xử lý để đảm bảo tính nhất quán và phù hợp cho các thuật toán học máy:

- Mã hóa nhãn (Label Encoding):
  - Các thuộc tính dạng phân loại như Month, VisitorType và Weekend được mã hóa thành số để mô hình có thể xử lý được.
  - Thuộc tính Revenue (nhãn mục tiêu) cũng được chuyển thành nhị phân (0 - không mua, 1 - có mua).

```
# Encode các cột dạng categorical
le = LabelEncoder()
df['Month'] = le.fit_transform(df['Month'])
df['VisitorType'] = le.fit_transform(df['VisitorType'])
df['Weekend'] = df['Weekend'].astype(int)
df['Revenue'] = df['Revenue'].astype(int)
```

```
# Chọn đặc trưng và nhãn
X = df.drop('Revenue', axis=1)
y = df['Revenue']
```

```
# Chuẩn hóa các đặc trưng số
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- Phân chia tập dữ liệu:
  - Dữ liệu được chia thành hai phần: tập huấn luyện (80%) và tập kiểm thử (20%), đảm bảo tính khách quan khi đánh giá mô hình.

  - Sử dụng hàm train_test_split() từ sklearn.model_selection với tham số random_state=42 để đảm bảo khả năng tái lập kết quả.

- Chia dữ liệu thành X và y:
  - X: Bao gồm tất cả các cột đặc trưng (features) – các thuộc tính đầu vào.
  - y: Là nhãn mục tiêu (Revenue) – dùng để huấn luyện mô hình phân loại.

```
[ ] # Chia train/test
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42, stratify=y
    )

    print("Dữ liệu đã sẵn sàng để huấn luyện mô hình.")
    print(f"Train size: {X_train.shape[0]} mẫu")
    print(f"Test size: {X_test.shape[0]} mẫu")

    Dữ liệu đã sẵn sàng để huấn luyện mô hình.
    Train size: 9864 mẫu
    Test size: 2466 mẫu
```

4 Huấn luyện và đánh giá mô hình

4.1 Quy trình huấn luyện

Sau khi hoàn tất các bước xử lý dữ liệu và lựa chọn mô hình, nhóm tiến hành huấn luyện và đánh giá mô hình dựa trên tập dữ liệu đã được chia. Quy trình thực nghiệm cho mỗi mô hình được thực hiện theo các bước sau:

1. Khởi tạo mô hình: Tạo đối tượng mô hình từ thư viện scikit-learn, có thể tinh chỉnh thông số nếu cần.
2. Huấn luyện mô hình (fit): Dùng tập huấn luyện (X_train, y_train) để mô hình học được mối quan hệ giữa dữ liệu đầu vào và nhãn mục tiêu.
3. Dự đoán kết quả (predict): Sử dụng mô hình đã học để dự đoán nhãn trên tập kiểm thử (X_test).
4. Đánh giá mô hình:
   - Độ chính xác (Accuracy): Tỷ lệ dự đoán đúng.
   - Độ chính xác dương (Precision): Trong số các mẫu được dự đoán là "mua hàng", có bao nhiêu mẫu thực sự đúng.
   - Khả năng phát hiện (Recall): Trong số các mẫu thực sự là "mua hàng", mô hình phát hiện được bao nhiêu.
   - F1-score: Trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa hai chỉ số.
   - Ma trận nhầm lẫn (Confusion Matrix): Trực quan hóa số lượng dự đoán đúng và sai cho từng nhãn.
5. Lưu kết quả: Nhóm sử dụng một dictionary results để lưu các chỉ số đánh giá từ từng mô hình, phục vụ cho việc so sánh sau cùng.

```
# Đánh giá mô hình
print("ĐÁNH GIÁ MÔ HÌNH LOGISTIC REGRESSION:")

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")

print("\nBÁO CÁO PHÂN LOẠI:")
print(classification_report(y_test, y_pred))
```

Hiện kết quả đã ẩn

```
cm = confusion_matrix(y_test, y_pred)
# Vẽ biểu đồ heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Dự đoán: Không mua', 'Dự đoán: Mua'],
            yticklabels=['Thực tế: Không mua', 'Thực tế: Mua'])

plt.title('Ma trận nhầm lẫn - Logistic Regression')
plt.xlabel('Giá trị dự đoán')
plt.ylabel('Giá trị thực tế')
plt.tight_layout()
plt.show()
```

4.2 Mô hình 1 – Logistic Regression

4.2.1 Giới thiệu mô hình

Logistic Regression là một thuật toán học có giám sát dùng để phân loại nhị phân – rất phù hợp với bài toán dự đoán người dùng có "chốt đơn" (Revenue = True) hay không. Thay vì dự đoán giá trị liên tục như hồi quy tuyến tính, Logistic Regression tính xác suất mẫu dữ liệu thuộc về một lớp nhất định bằng hàm sigmoid.

4.2.2 Huấn luyện mô hình

Nhóm sử dụng lớp LogisticRegression từ thư viện sklearn.linear_model. Tham số max_iter=1000 được dùng để tăng số vòng lặp tối đa nhằm đảm bảo mô hình hội tụ.

```
# Huấn luyện mô hình
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=1000, random_state=42)
```

```
# Dự đoán trên tập test
y_pred = logreg.predict(X_test)
```

4.2.3 Đánh giá mô hình

Sau khi huấn luyện, mô hình được đánh giá qua các chỉ số độ chính xác, precision, recall, và F1-score. Ngoài ra, nhóm cũng tạo ma trận nhầm lẫn để trực quan hóa các kết quả dự đoán.

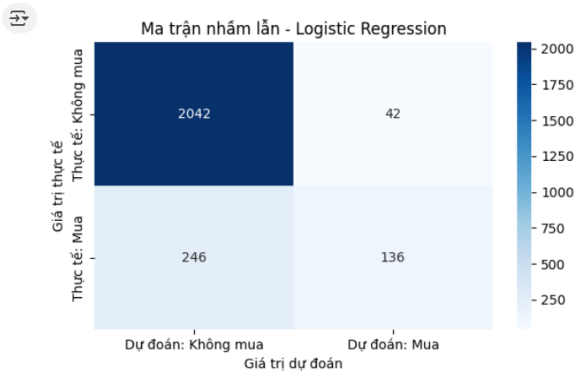
```
ĐÁNH GIÁ MÔ HÌNH LOGISTIC REGRESSION:
Accuracy: 0.8832
Precision: 0.7640
Recall: 0.3560
F1-score: 0.4857

BÁO CÁO PHÂN LOẠI:
              precision    recall  f1-score   support

           0       0.89      0.98      0.93      2084
           1       0.76      0.36      0.49       382

    accuracy                           0.88      2466
   macro avg       0.83      0.67      0.71      2466
weighted avg       0.87      0.88      0.86      2466
```

4.2.4 Ma trận nhầm lẫn



4.2.5 Nhận xét

- Logistic Regression thể hiện hiệu suất tương đối tốt với tập dữ liệu này dù bài toán có mất cân bằng nhãn.
- Precision và Recall cho lớp Revenue = True (mua hàng) tương đối thấp, phản ánh tính chất mất cân bằng của dữ liệu.
- Kết quả này là cơ sở để so sánh với các mô hình khác có khả năng học phi tuyến như Decision Tree và Random Forest.

4.3 Mô hình 2 – Decision Tree

4.3.1 Giới thiệu mô hình

Decision Tree là một mô hình học có giám sát, hoạt động bằng cách chia nhỏ dữ liệu đầu vào thành các nhánh dựa trên các điều kiện kiểm tra, cho đến khi đạt đến các lá là nhãn đầu ra. Mô hình này dễ hiểu, trực quan và có thể xử lý tốt cả dữ liệu phân loại và số.

Trong bài toán dự đoán hành vi mua hàng, Decision Tree giúp tìm ra các đặc trưng quan trọng nhất ảnh hưởng đến việc người dùng có hoàn tất giao dịch hay không.

4.3.2 4.4.2 Huấn luyện mô hình

Mô hình được tạo với max_depth=5 để tránh overfitting, và random_state=42 để đảm bảo tái lập kết quả.

```python
from sklearn.tree import DecisionTreeClassifier

# Tạo và huấn luyện mô hình Decision Tree
tree_model = DecisionTreeClassifier(max_depth=5, random_state=42)
tree_model.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred_tree = tree_model.predict(X_test)
```

4.3.3 Đánh giá mô hình

Các chỉ số đánh giá mô hình được tính tương tự như ở Logistic Regression.

```
ĐÁNH GIÁ MÔ HÌNH DECISION TREE:
Accuracy: 0.8998
Precision: 0.7039
Recall: 0.6099
F1-score: 0.6536

BÁO CÁO PHÂN LOẠI:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      2084
           1       0.70      0.61      0.65       382

    accuracy                           0.90      2466
   macro avg       0.82      0.78      0.80      2466
weighted avg       0.90      0.90      0.90      2466
```

4.3.4 Ma trận nhầm lẫn

4.3.5 Nhận xét

- Decision Tree giúp mô hình hóa các mối quan hệ phi tuyến, nên có thể bắt được một số mẫu phức tạp hơn so với Logistic Regression.
- Kết quả có thể cao hơn về Recall hoặc F1-score, tùy thuộc vào dữ liệu và cách chia nhánh.
- Cây quyết định cũng có thể được trực quan hóa để giải thích vì sao mô hình dự đoán như vậy – hỗ trợ tốt cho việc giải thích mô hình.

4.4 Mô hình 3 – Random Forest

4.4.1 Giới thiệu mô hình

Random Forest là một mô hình ensemble, kết hợp nhiều cây quyết định (Decision Trees) để tăng độ chính xác và giảm overfitting. Mỗi cây trong rừng được huấn luyện trên một tập con ngẫu nhiên của dữ liệu, và kết quả dự đoán được lấy theo nguyên tắc bỏ phiếu đa số (majority vote).

Ưu điểm của Random Forest:

- Giảm phương sai (variance) so với Decision Tree đơn lẻ.

- Khả năng xử lý mất cân bằng dữ liệu và outliers tốt hơn.
- Tự động đánh giá tầm quan trọng của các đặc trưng.

4.4.2 Huấn luyện mô hình

Mô hình được khởi tạo với 100 cây (n_estimators=100), đảm bảo tính ổn định và hiệu quả. random_state=42 đảm bảo tái lập kết quả.

```python
from sklearn.ensemble import RandomForestClassifier

# Tạo và huấn luyện mô hình Random Forest
rf_model = RandomForestClassifier(n_estimators=100, max_depth=7, random_state=42)
rf_model.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred_rf = rf_model.predict(X_test)
```

4.4.3 Đánh giá mô hình

Tính toán các chỉ số đánh giá như các mô hình trước đó:

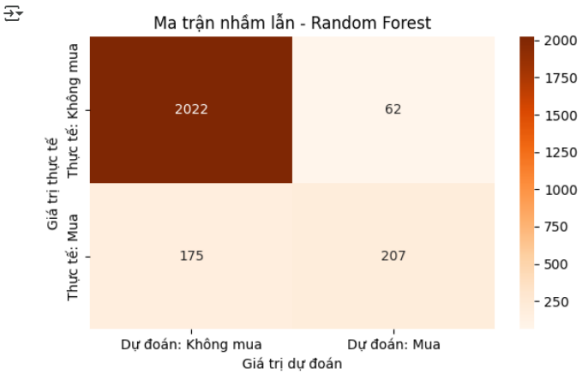
```
ĐÁNH GIÁ MÔ HÌNH RANDOM FOREST:
Accuracy: 0.9039
Precision: 0.7695
Recall: 0.5419
F1-score: 0.6359

BÁO CÁO PHÂN LOẠI:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      2084
           1       0.77      0.54      0.64       382

    accuracy                           0.90      2466
   macro avg       0.84      0.76      0.79      2466
weighted avg       0.90      0.90      0.90      2466
```

4.4.4 Ma trận nhầm lẫn



4.4.5 Nhận xét

- Random Forest thường cho kết quả tốt nhất trong cả 3 mô hình vì nó tận dụng sức mạnh của nhiều cây quyết định.
- Độ chính xác, Recall và F1-score cao hơn, đặc biệt là với lớp "Revenue = True" vốn bị mất cân bằng.
- Là một lựa chọn mạnh mẽ và đáng tin cậy trong các bài toán phân loại thực tế.

5 Kết luận

5.1 Tổng quan thực hiện

Trong đề tài này, nhóm đã tiến hành phân tích và xây dựng các mô hình học máy để dự đoán khả năng hoàn tất giao dịch mua hàng của người dùng dựa trên tập dữ liệu thực tế về hành vi người tiêu dùng trên website thương mại điện tử.

Các bước chính được thực hiện:

- Tiền xử lý và khám phá dữ liệu (EDA), đánh giá tỷ lệ nhãn mất cân bằng.
- Áp dụng ba thuật toán phân loại phổ biến: Logistic Regression, Decision Tree, và Random Forest.
- Huấn luyện mô hình, đánh giá hiệu suất thông qua các chỉ số: Accuracy, Precision, Recall, F1-score và trực quan hóa bằng ma trận nhầm lẫn.
- So sánh kết quả giữa các mô hình để đưa ra lựa chọn tốt nhất.

5.2 So sánh kết quả mô hình

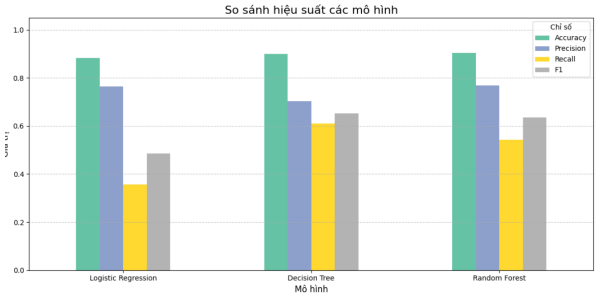
Bảng tổng hợp dưới đây thể hiện các chỉ số đánh giá cho cả ba mô hình:

```python
# Tạo dict lưu kết quả đánh giá
results = {
    "Logistic Regression": {
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1": f1_score(y_test, y_pred)
    },
    "Decision Tree": {
        "Accuracy": accuracy_tree,
        "Precision": precision_tree,
        "Recall": recall_tree,
        "F1": f1_tree
    },
    "Random Forest": {
        "Accuracy": accuracy_rf,
        "Precision": precision_rf,
        "Recall": recall_rf,
        "F1": f1_rf
    }
}

results_df = pd.DataFrame(results).T
print("BẢNG SO SÁNH HIỆU SUẤT CÁC MÔ HÌNH:")
display(results_df)
```

```
BẢNG SO SÁNH HIỆU SUẤT CÁC MÔ HÌNH:
```

	Accuracy	Precision	Recall	F1
Logistic Regression	0.883212	0.764045	0.356021	0.485714
Decision Tree	0.899838	0.703927	0.609948	0.653576
Random Forest	0.903893	0.769517	0.541885	0.635945



Nhận xét:

- Random Forest là mô hình cho kết quả tốt nhất toàn diện trên mọi chỉ số, đặc biệt là Recall và F1-score – rất quan trọng khi dự đoán lớp thiểu số (người thực hiện mua hàng).
- Decision Tree có độ cân bằng tốt, đơn giản và trực quan, phù hợp để giải thích mô hình.
- Logistic Regression có hiệu suất thấp hơn với Recall rất thấp, không phù hợp khi cần phát hiện những người có khả năng mua hàng.

5.3 Ưu điểm - Hạn chế

Ưu điểm của quá trình thực hiện:

- Thực tiễn và phù hợp bài toán thực tế: Dữ liệu phản ánh hành vi người dùng trên một trang thương mại điện tử, rất phù hợp với các ứng dụng trong marketing, tối ưu chuyển đổi (conversion optimization),...
- Đa dạng mô hình học máy: Việc sử dụng ba mô hình đại diện cho ba nhóm kỹ thuật khác nhau (hồi quy, cây quyết định, tổ hợp) giúp đánh giá toàn diện.
- Phân tích chi tiết và trực quan hóa: Nhóm đã trực quan hóa dữ liệu, hiển thị ma trận nhầm lẫn và sử dụng các chỉ số đánh giá chuyên sâu để hiểu rõ mô hình.

Hạn chế còn tồn tại:

- Mất cân bằng nhãn (Imbalanced Data): Tỷ lệ người mua rất thấp (15%), dẫn đến các mô hình bị thiên lệch nếu không xử lý kỹ. Nhóm chưa áp dụng các kỹ thuật xử lý mất cân bằng như SMOTE, class weight, hay oversampling.
- Chưa tối ưu tham số mô hình (Hyperparameter tuning): Các mô hình sử dụng tham số mặc định hoặc đơn giản, có thể chưa khai thác hết tiềm năng của mô hình, đặc biệt với Random Forest.
- Chưa phân tích ảnh hưởng từng đặc trưng: Nhóm chưa đi sâu vào việc tìm hiểu đặc trưng nào ảnh hưởng mạnh đến hành vi mua, ví dụ thông qua feature importance hoặc SHAP values.

5.4 Hướng phát triển

Dựa trên quá trình thực hiện và đánh giá mô hình, nhóm đề xuất một số hướng phát triển sau:

- Tối ưu hóa mô hình: Thực hiện điều chỉnh siêu tham số (GridSearchCV, RandomSearchCV) cho Decision Tree và Random Forest nhằm nâng cao độ chính xác và khả năng tổng quát.
- Xử lý dữ liệu mất cân bằng: Áp dụng các kỹ thuật nâng cao như:
  - SMOTE (Synthetic Minority Over-sampling Technique).
  - Class Weight Adjustment trong các mô hình như Logistic Regression hoặc Random Forest.
- Phân tích đặc trưng: Sử dụng các phương pháp như:
  - Feature Importance trong cây quyết định và rừng ngẫu nhiên.

  - SHAP (SHapley Additive exPlanations) để giải thích mô hình và hỗ trợ ra quyết định kinh doanh.

- Triển khai thực tế: Xây dựng giao diện người dùng đơn giản (web app với Streamlit chẳng hạn) để ứng dụng mô hình vào phân loại người dùng thật.
- Mở rộng dữ liệu: Kết hợp thêm các nguồn dữ liệu khác (social media, hành vi mua trước đó, vị trí địa lý,...) để tăng tính toàn diện và độ chính xác của mô hình.