TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

Môn học: Trí tuệ nhân tạo

TÌM HIỂU GIẢI THUẬT PHÂN LỚP CÂY QUYẾT ĐỊNH CỦA SCIKIT-LEARN ĐỂ PHÂN LỚP BỘ DỮ LIỆU MẪU CAR

GIẢNG VIÊN HƯỚNG DẪN: PGS.TS. Dương Tuấn Anh
SINH VIÊN THỰC HIỆN: Trần Ngọc Thảo Ngân – 21DH114460
Bùi Tuấn Đạt – 21DH113218
Phạm Gia Huy – 21DH113678

Tp. Hồ chí minh, Ngày 19 tháng 07 năm 2024

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

Môn học: Trí tuệ nhân tạo

TÌM HIỂU GIẢI THUẬT PHÂN LỚP CÂY QUYẾT ĐỊNH CỦA SCIKIT-LEARN ĐỂ PHÂN LỚP BỘ DỮ LIỆU MẪU CAR

GIẢNG VIÊN HƯỚNG DẪN: PGS.TS. Dương Tuấn Anh
SINH VIÊN THỰC HIỆN: Trần Ngọc Thảo Ngân – 21DH114460
Bùi Tuấn Đạt – 21DH113218
Phạm Gia Huy – 21DH113678

Tp. Hồ chí minh, Ngày 19 tháng 07 năm 2024

LỜI CẢM ƠN

Nhóm 18 xin gửi lời cảm ơn chân thành đến Giảng viên Dương Tuấn Anh vì đã tạo điều kiện thuận lợi cho nhóm trong quá trình thực hiện đề tài đồ án này. Thầy đã dành thời gian và tâm huyết để truyền đạt những kiến thức quý báu trong suốt học kì, giúp nhóm chắc chắn về nền tảng kiến thức và hoàn thành đề tài này.

Trong đề tài này, nhóm 18 đã cố gắng hết sức để hoàn thành trong khả năng và phạm vi của mình. Tuy nhiên, chắc chắn vẫn còn thiếu sót. Vì vậy, nhóm mong nhận được sự thông cảm và đóng góp ý kiến tận tình từ quý thầy cô và các bạn.

Một lần nữa, nhóm 18 chân thành cảm ơn Giảng viên Dương Tuấn Anh vì sự nhiệt tình hướng dẫn và đóng góp ý kiến đã giúp cho nhóm hoàn thành đề tài này.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

1. GIỚI THIỆU VỀ CÂY QUYẾT ĐỊNH (DECISION TREE)

1.1. Thuật toán cây quyết định (Decision tree) là gì?

- Cây quyết định (decision tree) là một cây mà trong đó mỗi nút nội biểu diễn một quyết định và mỗi nút lá biểu diễn một kết quả hoặc một nhãn lớp.
- Mỗi nút nội xem xét một số trị của thuộc tính để dẫn đến một vài nhánh. Mỗi nhánh đến lượt nó gắn với một quyết định. Mỗi nhánh khác biệt lẫn nhau và biểu thị đầy đủ mọi khả năng.
- Cây quyết định là công cụ tiện dụng để lựa chọn giữa nhiều tiến trình hành động. Nó cung cấp một cấu trúc hiệu quả để chúng ta đặt ra những lựa chọn và xem xét kết quả của những lựa chọn đó.
- Trong trường hợp của cây quyết định nhị phân (binary decision tree), mỗi nút có hai cạnh đi ra. Một cạnh biểu diễn kết quả "yes" hay "true" và cạnh kia biểu diễn kết qủa "no" hay "false".
- Các mẫu có thể được phân lớp dựa vào cây quyết định mà trong đó mỗi nút trên cây biểu diễn một trạng thái (status) của bài toán sau khi thực hiện một quyết định nào đó.
- Mỗi nút lá ứng với nhãn lớp của một luật phân lớp tương ứng với một lối đi (path) từ nút rễ đến nút lá đó [1] [2] [3]

1.2. Mô tả thuật toán cây quyết định (Decision Tree)

- **Bước 1: Chọn thuộc tính tốt nhất**

Tiêu chí lựa chọn**:** Các tiêu chí phổ biến để chọn thuộc tính tốt nhất bao gồm Gini Index, Entropy và Information Gain (cho bài toán phân loại), hoặc Variance Reduction (cho bài toán hồi quy). Những tiêu chí này đo lường mức độ thuần nhất (homogeneity) của các tập con sau khi phân chia.

- **Bước 2: Phân chia dữ liệu**

Dựa trên thuộc tính đã chọn, dữ liệu được phân chia thành các tập con. Mỗi tập con đại diện cho một nhánh trong cây quyết định.

- **Bước 3: Lặp lại quá trình cho mỗi tập con**

Đối với mỗi tập con mới tạo ra, lặp lại quá trình chọn thuộc tính và phân chia dữ liệu cho đến khi đạt đến các tiêu chí dừng (ví dụ: tất cả các điểm dữ liệu trong tập con thuộc về cùng một lớp, không còn thuộc tính nào để phân chia, hoặc đạt đến độ sâu tối đa của cây). [4]

2. CÁC TIÊU CHÍ ĐỂ PHÂN CHIA

- **Gini Impurity**: Đo lường khả năng phân loại sai một mẫu mới nếu nó được phân loại ngẫu nhiên theo phân bố của các lớp trong tập dữ liệu.

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

Trong đó, p_i là xác suất của một mẫu được phân vào một lớp cụ thể.

- **Độ pha tạp Entropy**: Đo lường mức độ bất định hoặc không thuần nhất trong tập dữ liệu.

$$Entropy = -\sum_{i=1}^{n} p_i log_2(p_i)$$

Trong đó, p_i là xác suất của một mẫu được phân vào một lớp cụ thể.

- **Information Gain**: Đo lường sự giảm đi của entropy hoặc độ Gini sau khi tập dữ liệu được phân chia theo một thuộc tính.

$$Information\ Gain = Entropy_{parent} - \sum_{i=1}^{n} \left(\frac{|D_i|}{|D|} * Entropy(D_i) \right)$$

Trong đó, D_i là tập con của D sau khi phân chia theo một thuộc tính. [4]

3. CÁC ĐỘ ĐO VỀ TÍNH CHÍNH XÁC PHÂN LỚP

3.1. Ma trận đúng sai (Confusion matrix)

Ma trận đúng sai (confusion matrix) là một công cụ quan trọng để đánh giá hiệu suất của mô hình phân loại trong học máy. Nó giúp ghi lại và biểu diễn số lượng các dự đoán đúng và sai lầm của mô hình trên tập dữ liệu kiểm tra hoặc tập dữ liệu đánh giá. Ma trận đúng sai thường được sử dụng để tính toán các độ đo và thống kê quan trọng như độ chính xác, precision, recall, F1-score, và nhiều độ đo khác.

Ma trận đúng sai thường được biểu diễn dưới dạng ma trận 2x2, mỗi một ô tương ứng với một giá trị. Ma trận đúng sai được biểu diễn như sau [5]:

Bảng 1. Bảng mô tả ma trận đúng sai (Connfusion Matrix)

		Điều kiện dự đoán	
		Dương tính (Positive)	Âm tính (Negative)
Điều kiện thực tế	Dương tính (Positive)	**True Positive (TP)**: Số lượng mẫu thuộc lớp dự đoán đúng và cũng thuộc lớp thực tế.	**False Negative (FN):** Số lượng mẫu thuộc lớp dự đoán sai (lớp âm tính) nhưng thuộc lớp thực tế là dương tính.
	Âm tính (Negative)	**False Positive (FP):** Số lượng mẫu thuộc lớp dự đoán sai (lớp dương tính) nhưng thuộc lớp thực tế là âm tính.	**True Negative (TN):** Số lượng mẫu thuộc lớp dự đoán đúng và và cũng thuộc lớp thực tế.

3.2. Độ nhạy (sensitivity) và độ chuyên biệt (specificity)

Độ nhạy còn được gọi là mức độ dương đúng (true positive), tức là tỉ lệ mẫu dương mà được phân lớp đúng. Độ chuyên biệt là tỉ lệ mẫu âm mà được phân lớp đúng.

$$sensitivity = \frac{t\_pos}{pos}$$

$$specificity = \frac{t\_neg}{neg}$$

Với *t_pos* là tổng số mẫu dương đúng (true positive), *pos* là tổng số mẫu lớp dương, *t_neg* là tổng số mẫu âm đúng (true negative) và *neg* là tổng số mẫu lớp âm. [1]

3.3. Độ đo Accuracy, Precision, Recall

Dựa vào các giá trị trong ma trận lỗi, chúng ta có thể tính toán các độ đo quan trọng như sau: [6]

- Độ chính xác (Accuracy): Độ chính xác đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán. Được tính bằng:

$$accuracy = \text{sensitivity}\frac{pos}{(pos+neg)} + specificity\frac{neg}{(pos+neg)} = \frac{t\_pos + t\_neg}{(pos+neg)}$$

- Precision (Độ chuẩn xác): Precision đo lường tỷ lệ số lượng mẫu dự đoán dương đúng (TP) trên tổng số lượng mẫu dự đoán dương (TP + FP). Nó đo lường khả năng của mô hình phân loại mẫu dự đoán dương đúng mà không gây ra sai lầm

$$Precision = \frac{t\_pos}{t\_pos + f\_pos}$$

- Recall (Độ bao phủ): Recall đo lường tỷ lệ số lượng mẫu dự đoán dương đúng (TP) trên tổng số lượng mẫu thực tế dương (TP + FN). Nó đo lường khả năng của mô hình phân loại phát hiện được toàn bộ số lượng mẫu thực tế dương.

$$Recall = \frac{t\_pos}{t\_pos + f\_neg}$$

- F1-Score (F1): F1-Score là một sự kết hợp giữa precision và recall để đo hiệu suất tổng thể của mô hình. Nó được tính bằng:

$$F1\_Score = 2\frac{Precision * Recall}{Precision + Recall}$$

4. ÁP DỤNG DECISION TREE VÀO PHÂN LỚP BỘ DỮ LIỆU CAR EVALUATION

4.1. Tổng quan bộ dữ liệu Car Evaluation

- Tên bộ dữ liệu: Car Evaluation
- Nguồn dữ liệu: UCI Machine Learning Repository
- Cấu trúc gồm: 1728 mẫu, 7 thuộc tính, trong đó thuộc tính "class" là thuộc tính mục tiêu.

Bảng 2. Bảng mô tả bộ dữ liệu Car Evaluation

STT	Thuộc tính	Loại dữ liệu	Ý nghĩa
1	buying	Rời rạc	Giá mua
2	maint	Rời rạc	Giá bảo trì
3	doors	Rời rạc	Số lượng cửa
4	persons	Rời rạc	Sức chứa (số người)
5	lug_boot	Rời rạc	Kích thước cốp xe
6	safety	Rời rạc	Mức độ an toàn ước lượng
7	class	Rời rạc	Mức độ đánh giá (không chấp nhận được, chấp nhận được, tốt, rất tốt)

4.2. Xử lý dữ liệu

- Khai báo các thư viện sử dụng

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

- Thực hiện đọc dữ liệu vào

```
df = pd.read_csv('/content/drive/MyDrive/DoAnAI/car.csv')
df.head()
```

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Hình 1. Năm dòng dầu của tập dữ liệu

- Thông tin của tập dữ liệu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Hình 2. Thông tin của tập dữ liệu

- Xử lý dữ liệu null

```
df.isnull().sum()
buying      0
maint       0
```

```
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

Tập dữ liệu không có dữ liệu null

- Xử lý dữ liệu trùng lắp

```
print(f"Number of duplicate rows: {df.duplicated().sum()}")
Number of duplicate rows: 0
```

Tập dữ liệu không có dòng dữ liệu nào bị trùng

- Kiểm tra tần suất xuất hiện của các biến

```
for col in col_names:
    print(df[col].value_counts())
buying
vhigh    432
high     432
med      432
low      432
Name: count, dtype: int64
maint
vhigh    432
high     432
med      432
low      432
Name: count, dtype: int64
doors
2        432
3        432
4        432
5more    432
Name: count, dtype: int64
persons
2       576
4       576
more    576
Name: count, dtype: int64
lug_boot
small    576
med      576
big      576
Name: count, dtype: int64
safety
low     576
med     576
high    576
Name: count, dtype: int64
class
unacc    1210
acc       384
good       69
vgood      65
```

```
Name: count, dtype: int64
```

Các cột như buying, maint, doors, persons, lug_boot, và safety đều có sự phân phối giá trị đồng đều. Chỉ có cột class (thuộc tính mục tiêu) ta có thể thấy rằng nó có sự chênh lệch giữa các biến trong cột. Phần lớn các mẫu xe được xếp loại "unacc" (không chấp nhận được), trong khi số lượng mẫu xe được xếp loại "good" và "vgood" rất ít.

- Khai báo vector đặc trưng và biến mục tiêu

```
X = df.drop(['class'], axis=1)
y = df['class']
```

X: Biến này lưu trữ một DataFrame mới được tạo ra bằng cách loại bỏ cột class khỏi DataFrame ban đầu df. DataFrame này chứa tất cả các đặc trưng (features) cần thiết cho việc huấn luyện mô hình, ngoại trừ nhãn (labels).

y: Biến này lưu trữ cột class từ DataFrame df, chứa các nhãn (labels) tương ứng với từng hàng. Đây là đầu ra mong muốn mà mô hình sẽ dự đoán.

- Tiến hành mã hóa dữ liệu

Vì các mô hình thống kê và học máy yêu cầu dữ liệu đầu vào dưới dạng số để thực hiện các tính toán và phân tích, nên việc mã hóa các biến phân loại là cần thiết. Chúng tôi sử dụng OrdinalEncoder từ thư viện sklearn.preprocessing để mã hóa các biến phân loại như 'buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', và 'class' thành các giá trị số. Việc sử dụng OrdinalEncoder là phù hợp vì các biến phân loại trong tập dữ liệu của Car Evaluation có thứ tự tự nhiên. Điều này giúp bảo toàn mối quan hệ thứ tự giữa các giá trị phân loại, đảm bảo rằng các mô hình học máy có thể xử lý dữ liệu một cách chính xác và hiệu quả.

```
categories = [
    ['low', 'med', 'high', 'vhigh'],  # buying
    ['low', 'med', 'high', 'vhigh'],  # maint
    ['2', '3', '4', '5more'],         # doors
    ['2', '4', 'more'],               # persons
    ['small', 'med', 'big'],          # lug_boot
    ['low', 'med', 'high'],           # safety
]
from sklearn.preprocessing import OrdinalEncoder
encoder_sklearn = OrdinalEncoder(categories=categories)
X_encoded = encoder_sklearn.fit_transform(X)
X_encoded_df = pd.DataFrame(X_encoded, columns=X.columns)
X_encoded_df.head()
```

	buying	maint	doors	persons	lug_boot	safety
0	3.0	3.0	0.0	0.0	0.0	0.0
1	3.0	3.0	0.0	0.0	0.0	1.0
2	3.0	3.0	0.0	0.0	0.0	2.0
3	3.0	3.0	0.0	0.0	1.0	0.0
4	3.0	3.0	0.0	0.0	1.0	1.0

Hình 3. Kết quả sau khi mã hóa X

4.3. Thực hiện xây dựng mô hình cây quyết định (Decision Tree)

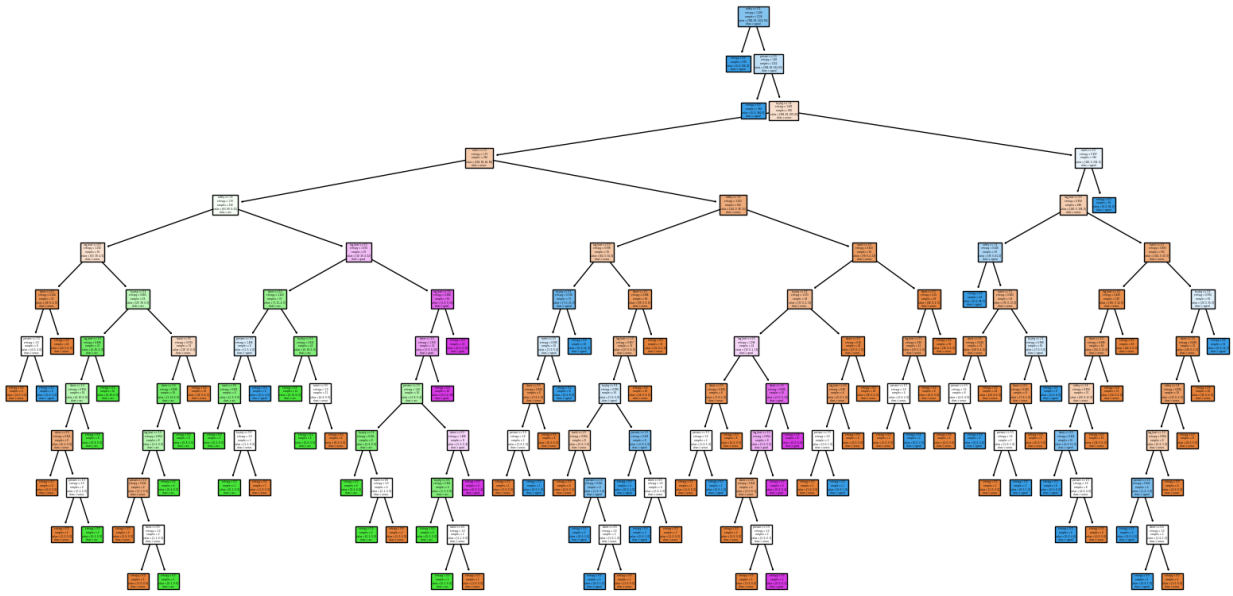
Trong phần này, chúng ta sẽ xây dựng một mô hình cây quyết định để phân loại dữ liệu, sử dụng tiêu chí để phân chia dữ liệu là Entropy nhằm đo lường mức độ thông tin của các thuộc tính. Mô hình được triển khai sử dụng thư viện scikit-learn của Python với random_state được thiết lập là 0 để đảm bảo tính tái lập của kết quả khi chạy lại mô hình.

```
from sklearn.tree import DecisionTreeClassifier
clf_en = DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Sau khi khởi tạo mô hình, chúng ta tiến hành thử nghiệm huấn luyện mô hình trên toàn bộ tập dữ liệu.

```
clf_en.fit(X_encoded_df, y)
```

Sau khi mô hình đã được huấn luyện, cấu trúc của cây quyết định có thể được hiển thị để giúp hiểu rõ hơn về cách mà mô hình đưa ra các quyết định plot_tree từ sklearn.tree được sử dụng để hiển thị cây quyết định.



Hình 4. Hình ảnh cây quyết định khi được huấn luyện trên toàn bộ tập dữ liệu

Độ sâu của cây là 12, số nút lá là 84 lá.

4.4. Đánh giá hiệu suất mô hình

Ở đây chúng tôi sử dụng phương pháp Cross-validation với 15 folds để đánh giá hiệu suất của mô hình. Kỹ thuật này chia dữ liệu thành 15 phần, mỗi phần lần lượt được sử dụng làm tập kiểm tra trong khi các phần còn lại được sử dụng làm tập huấn luyện. Việc sử dụng kỹ thuật này giúp đánh giá hiệu suất mô hình một cách ổn định, có tính tổng quát hơn và giảm thiểu nguy cơ quá khớp (overfitting). Phương pháp này sử dụng hiệu quả dữ liệu, đặc biệt hữu ích khi dữ liệu bị hạn chế hoặc mất cân đối. Với dữ liệu mục tiêu ‘class’ gồm 1210 mẫu unacc, 384 mẫu acc, 69 mẫu good, và 65 mẫu vgood, cross-validation giúp đảm bảo mỗi lớp được đại diện đầy đủ trong quá trình huấn luyện và kiểm tra.

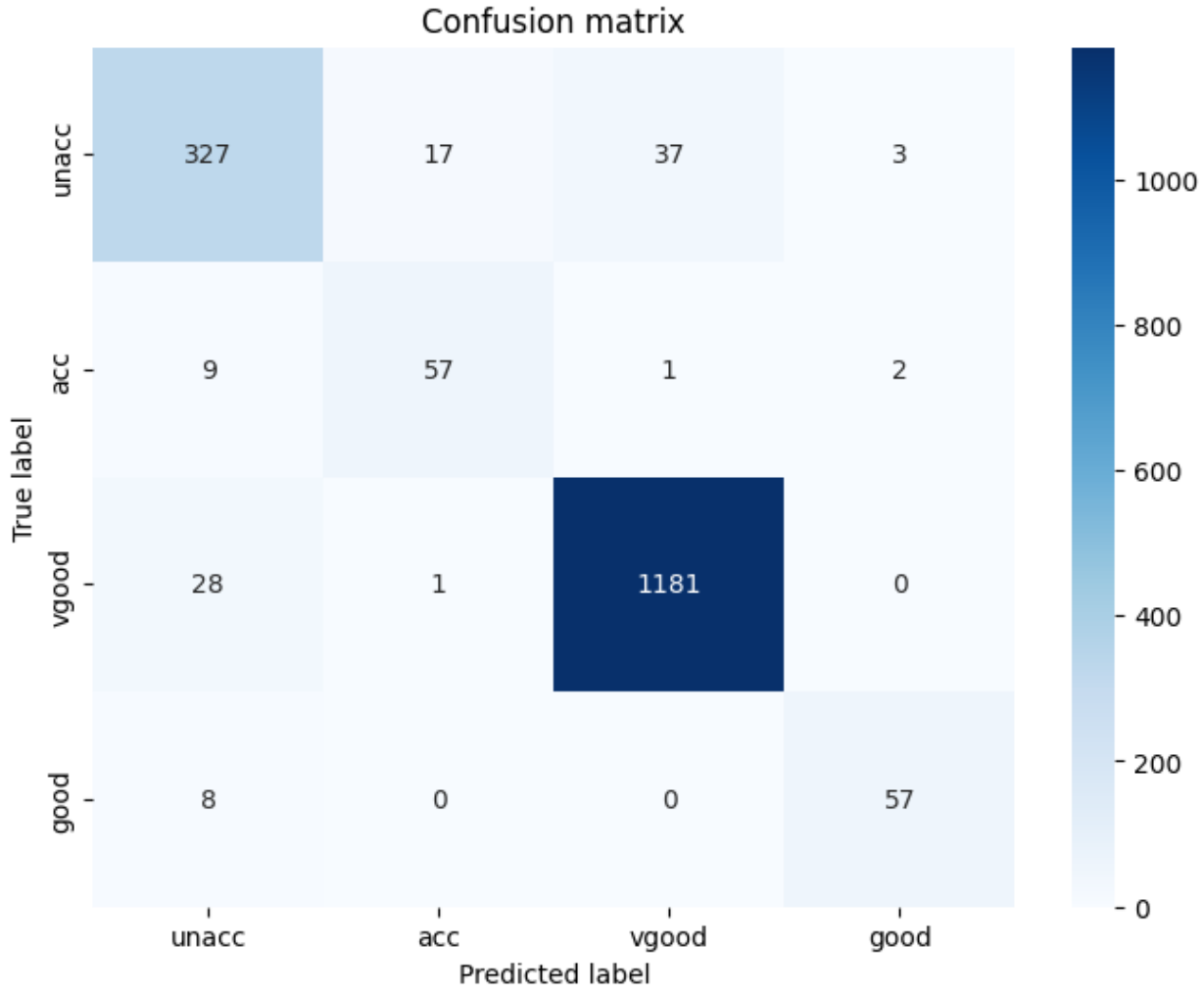
```
from sklearn.model_selection import cross_val_score, cross_val_predict
cv_scores = cross_val_score(clf_en, X_encoded_df, y, cv=15, scoring =
'accuracy')

print(f"Cross-validation scores: {cv_scores}")
print(f"Mean cross-validation score: {np.mean(cv_scores)}")
print(f"Standalization cross-validation score: {cv_scores.std()}")
```

```
Cross-validation scores: [0.95689655 0.96551724 0.85344828 0.94782609 1.
0.97391304
 0.93913043 0.97391304 0.95652174 0.96521739 0.91304348 0.99130435
 0.91304348 0.90434783 0.82608696]
Mean cross-validation score: 0.9386806596701649
Standalization cross-validation score: 0.04733250810579168
```

- Các giá trị điểm số cross-validation dao động từ 0.826 đến 1.0.
- Độ chính xác (accuraccy) trung bình là 0.9387, cho thấy mô hình có độ chính xác khá cao trên các tập con khác nhau của dữ liệu.
- Độ lệch chuẩn là 0.0473, cho thấy sự thay đổi không quá lớn giữa các lần chạy cross-validation, cho thấy mô hình có tính ổn định.

```
y_pred = cross_val_predict(clf_en, X_encoded_df, y, cv=15)
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



Hình 5. Ma trận lỗi (Confusion matrix) của mô hình cây quyết định

Ma trận đúng sai (Confusion Matrix) cho thấy số lượng mẫu mà mô hình phân loại đúng hoặc sai cho từng lớp cụ thể. Đối với lớp 'unacc' (unacceptable), mô hình hoạt động rất tốt với số lượng mẫu phân loại đúng là 327, chỉ có một số nhỏ bị nhầm lẫn với các lớp khác. Tương tự, lớp 'vgood' (very good) cũng có hiệu suất cao với 1181 mẫu được phân loại đúng, mặc dù có một số ít bị nhầm lẫn với các lớp khác. Tuy nhiên, lớp 'good' (good) có độ chính xác thấp hơn, với 8 mẫu bị nhầm lẫn với lớp 'unacc'.

```
print(classification_report(y, y_pred))
              precision    recall  f1-score   support

         acc       0.88      0.85      0.87       384
        good       0.76      0.83      0.79        69
       unacc       0.97      0.98      0.97      1210
       vgood       0.92      0.88      0.90        65

    accuracy                           0.94      1728
   macro avg       0.88      0.88      0.88      1728
weighted avg       0.94      0.94      0.94      1728
```

Báo cáo phân loại (Classification Report) cho thấy các chỉ số đánh giá như độ chuẩn xác (precision), độ bao phủ (recall), F1-score và số lượng mẫu (support) cho từng lớp. Mô hình đạt độ chính xác tổng thể là 0.94, cho thấy hiệu suất cao. Lớp 'unacc' có độ chuẩn xác (precision) và độ bao phủ (recall) rất cao (0.97 và 0.98), cho thấy mô hình phân loại rất tốt đối với lớp này. Tương tự, lớp 'vgood' cũng có hiệu suất tốt với F1-score là 0.90. Tuy nhiên, lớp 'good' có độ chuẩn xác (precision) và độ bao phủ (recall) thấp hơn, cho thấy mô hình gặp khó khăn trong việc phân loại đúng lớp này.

5. KẾT LUẬN

Ưu điểm của thuật toán cây quyết định (Decision Tree):

- Dễ hiểu và giải thích: Cây quyết định là mô hình trực quan, dễ hiểu và giải thích. Các quy tắc phân chia dựa trên các thuộc tính có thể dễ dàng được diễn giải.
- Xử lý dữ liệu phi tuyến: Cây quyết định có thể xử lý các mối quan hệ phi tuyến giữa các thuộc tính và nhãn.
- Không yêu cầu chuẩn hóa dữ liệu: Không cần chuẩn hóa hoặc chuẩn hóa thuộc tính.
- Làm việc với dữ liệu thiếu: Cây quyết định có thể xử lý dữ liệu thiếu bằng cách phân chia và ước lượng giá trị thiếu.

Nhược điểm của thuật toán cây quyết định (Decision Tree):

- Overfitting: Cây quyết định có thể dễ dàng quá khớp với dữ liệu huấn luyện, đặc biệt nếu chúng có nhiều nút và sâu.
- Không ổn định: Một thay đổi nhỏ trong dữ liệu có thể dẫn đến một cây hoàn toàn khác.
- Thiên vị các thuộc tính có nhiều giá trị: Cây quyết định có thể thiên vị các thuộc tính có nhiều giá trị phân loại hơn.

6. TÀI LIỆU THAM KHẢO

[1] P. D. T. Anh, "Chương 6. Phân lớp".

[2] T. t. n. tạo, "Cây Quyết Định (Decision Tree)," [Online]. Available: https://trituenhantao.io/kien-thuc/decision-tree/.

[3] T. Vu, "Decision Tree algorithm," [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html.

[4] GeeksforGeeks, "Decision Tree," [Online]. Available: https://www.geeksforgeeks.org/decision-tree/.

[5] P. L. Minh, "Báo cáo - Tìm hiểu về phân lớp dữ liệu," [Online]. Available: https://www.studocu.com/vn/document/dai-hoc-kinh-te-quoc-dan/cong-nghe-hien-dai-trong-cong-nghe-thong-tin/bao-cao-tim-hieu-ve-phan-lop-du-lieu/85912582.

[6] S. K. Agrawal, "Metrics to Evaluate your Classification Model to take the Right Decisions," [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/.

PHỤ LỤC 1. Bảng mô tả công việc được phân công của từng thành viên trong nhóm

Bảng 3. Bảng mô tả công việc được phân công

STT	MSSV	Họ và tên	Nội dung thực hiện
1	21DH114460	Trần Ngọc Thảo Ngân	4.Áp dụng thuật toán Decision Tree vào bộ dữ liệu Car Evaluation 4.1 Tổng quan bộ dữ liệu 4.2 Xử lý dữ liệu 4.3 Đánh giá hiệu suất mô hình 5.Kết luận
2	21DH113218	Bùi Tuấn Đạt	3.Các độ đo về tính chính xác phân lớp 3.1 Ma trận đúng sai (Confusion matrix) 3.2 Độ nhạy (sensitivity) và độ chuyên biệt (specificity) 3.3 Độ đo Accuracy, Precision, Recall

3	21DH113678	Phạm Gia Huy	1.Giới thiệu về cây quyết định (Decision Tree) 1.1 Thuật toán cây quyết định (Decision Tree) là gì 1.2 Mô tả thuật toán Decision Tree 2.Các tiêu chí phân chia

PHỤ LỤC 2. Code thực thi

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('/content/drive/MyDrive/DoAnAI/car.csv')
df.head()
df.describe()
df.info()
col_names=df.columns
for col in col_names:
    print(df[col].value_counts())
df.isnull().sum()
print(f"Number of duplicate rows: {df.duplicated().sum()}")
X = df.drop(['class'], axis=1)
y = df['class']
column_names = X.columns
categories = [
    ['low', 'med', 'high', 'vhigh'],  # buying
    ['low', 'med', 'high', 'vhigh'],  # maint
    ['2', '3', '4', '5more'],         # doors
    ['2', '4', 'more'],               # persons
    ['small', 'med', 'big'],          # lug_boot
    ['low', 'med', 'high'],           # safety
]
from sklearn.preprocessing import OrdinalEncoder
encoder_sklearn = OrdinalEncoder(categories=categories)
X_encoded = encoder_sklearn.fit_transform(X)
X_encoded_df = pd.DataFrame(X_encoded, columns=X.columns)
X_encoded_df.head()
from sklearn.tree import DecisionTreeClassifier
clf_en = DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
clf_en.fit(X_encoded_df, y)
from sklearn import tree
text_tree = tree.export_text(clf_en,feature_names=list(df.columns[:-1]))
print(text_tree)
class_names = y.unique().astype(str)
class_names
from sklearn.tree import plot_tree
plt.figure(figsize=(20,10))
plot_tree(clf_en, feature_names=X_encoded_df.columns,
class_names=class_names, filled=True)
plt.show()
print("Depth of the tree:", clf_en.get_depth())
print("Number of leaves:", clf_en.get_n_leaves())
from sklearn.model_selection import cross_val_score, cross_val_predict
cv_scores = cross_val_score(clf_en, X_encoded_df, y, cv=15, scoring =
'accuracy')

print(f"Cross-validation scores: {cv_scores}")
print(f"Mean cross-validation score: {np.mean(cv_scores)}")
print(f"Standalization cross-validation score: {cv_scores.std()}")
y_pred = cross_val_predict(clf_en, X_encoded_df, y, cv=15)
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```