AI Introduction Homework 1

侯竣奇 4112064214

April 27, 2025

1 Summerize

This Jupyter notebook demonstrates a machine learning classification task using a car evaluation dataset.

1. Data Collection and Preparation:
   - Downloads car evaluation data from UCI Machine Learning Repository
   - Target variable 'class' is binarized (0 for 'unacc'/'acc', 1 for 'good'/'vgood')
   - Applies one-hot encoding to categorical features
2. Model Development
   - Uses Decision Tree Classifier with initial parameters:
     - criterion: entropy
     - max_depth: 5
     - min_samples_split: 10
     - min_samples_leaf: 4
     - random_state: 42
3. Model Optimization
   - Implements GridSearchCV for hyperparameter tuning
   - Searches across multiple parameters
     - criterion: gini, entropy
     - max_depth: 3-8
     - min_samples_split: 5-20
     - min_samples_leaf: 2-8
4. Performance Analysis
   - Evaluates model using
     - accuracy_score
     - confusion_matrix
     - classification_report
   - Visualizes
     - Decision tree structure
     - Feature importance analysis through bar charts
5. Results
   - Successfully creates a binary classifier for car evaluation
   - Provides insights into which features are most important for car quality prediction
   - Demonstrates the effectiveness of decision trees for this classification task

2 Step 0: Download the data and save it locally

[1]:
```
import requests
import os
```

[2]:
```
resp = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/
↪car/car.data")
```

[3]:
```
resp.status_code
```

[3]: 200

[4]:
```
os.makedirs("data", exist_ok=True)
with open("data/hw1.csv", "w") as f:
    f.write(resp.text)
```

3 Step 1: Load and Preprocess the Data

3.1 Load the dataset

[5]:
```
import pandas as pd
```

[6]:
```
df = pd.read_csv("data/hw1.csv", header=None)
```

[7]:
```
# column names according to http://archive.ics.uci.edu/dataset/19/car+evaluation
#   buying:   vhigh, high, med, low.
#   maint:    vhigh, high, med, low.
#   doors:    2, 3, 4, 5, more.
#   persons:  2, 4, more.
#   lug_boot: small, med, big.
#   safety:   low, med, high.
df.columns = ["buying", "maint", "doors", "persons", "lug_boot", "safety",␣
↪"class"]
```

[8]:
```
df.head()
```

[8]:

	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

[9]:
```
df.info()
```

```
<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
```

可以看到 df.info() 顯示資料共有 1728 筆，7 個欄位，所有欄位型態皆為 object，且沒有任何缺失值 (non-null count 均為 1728)

[10]:
```
df.nunique() # 相異值的數量
```

[10]:
```
buying      4
maint       4
doors       4
persons     3
lug_boot    3
safety      3
class       4
dtype: int64
```

[11]:
```
df.describe(include='all') # 基本統計資訊
```

[11]:

	buying	maint	doors	persons	lug_boot	safety	class
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	vhigh	vhigh	2	2	small	low	unacc
freq	432	432	432	576	576	576	1210

[12]:
```
for col in df.columns:
    print(df[col].value_counts(), end="\n\n")
```

```
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
```

3.2 Data Cleaning and Transformation

使用 One-Hot Encoding 來處理 Categorical Data
（可以直接使用 pandas 的 get_dummies）

[13]:
```
features = df.columns[:-1]
```

[14]:
```
df_dum = pd.get_dummies(df[features])
```

[15]:
```
df_dum
```

[15]:
```
      buying_high  buying_low  buying_med  buying_vhigh  maint_high  \
0           False       False       False          True       False
```

```
1           False       False       False          True       False
2           False       False       False          True       False
3           False       False       False          True       False
4           False       False       False          True       False
…           …         …         …           …         …
1723        False        True       False         False       False
1724        False        True       False         False       False
1725        False        True       False         False       False
1726        False        True       False         False       False
1727        False        True       False         False       False

      maint_low  maint_med  maint_vhigh  doors_2  doors_3  …  doors_5more  \
0         False      False         True     True    False  …        False
1         False      False         True     True    False  …        False
2         False      False         True     True    False  …        False
3         False      False         True     True    False  …        False
4         False      False         True     True    False  …        False
…         …        …           …      …      …  …          …
1723       True      False        False    False    False  …         True
1724       True      False        False    False    False  …         True
1725       True      False        False    False    False  …         True
1726       True      False        False    False    False  …         True
1727       True      False        False    False    False  …         True

      persons_2  persons_4  persons_more  lug_boot_big  lug_boot_med  \
0          True      False         False         False         False
1          True      False         False         False         False
2          True      False         False         False         False
3          True      False         False         False          True
4          True      False         False         False          True
…          …        …            …           …           …
1723      False      False          True         False          True
1724      False      False          True         False          True
1725      False      False          True          True         False
1726      False      False          True          True         False
1727      False      False          True          True         False

      lug_boot_small  safety_high  safety_low  safety_med
0               True        False        True       False
1               True        False       False        True
2               True         True       False       False
3              False        False        True       False
4              False        False       False        True
…              …          …         …         …
1723           False        False       False        True
1724           False         True       False       False
1725           False        False        True       False
```

```
1726          False        False       False        True
1727          False         True       False       False

[1728 rows x 21 columns]
```

根據作業要求，需要將 class 分為 good 或 bad 兩種之一

1. unacc: bad
2. acc: bad
3. good: good
4. vgood: good

[16]:
```
df_y = df["class"]
df_y.value_counts()
```

[16]:
```
class
unacc    1210
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

[17]:
```
df_y = df_y.map({"unacc": 0, "acc": 0, "good": 1, "vgood": 1})
```

[18]:
```
df_y # 轉換後，`df_y` 只包含 0 和 1 兩種值。
```

[18]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1723    1
1724    1
1725    0
1726    1
1727    1
Name: class, Length: 1728, dtype: int64
```

3.3 Split the data

[19]:
```
from sklearn.model_selection import train_test_split
```

[20]:
```
X_train, X_test, y_train, y_test = train_test_split(df_dum, df_y, test_size=0.
↪2, random_state=42)
```

資料已成功分割為 X_train, X_test, y_train, y_test。

[21]:
```
X_train.head()
```

[21]:
```
      buying_high  buying_low  buying_med  buying_vhigh  maint_high  \
107         False       False       False          True       False
901         False       False        True         False       False
1709        False        True       False         False       False
706          True       False       False         False       False
678          True       False       False         False       False

      maint_low  maint_med  maint_vhigh  doors_2  doors_3  …  doors_5more  \
107       False      False         True    False    False  …         True
901       False      False         True    False     True  …        False
1709       True      False        False    False    False  …         True
706       False       True        False    False    False  …        False
678       False       True        False    False     True  …        False

      persons_2  persons_4  persons_more  lug_boot_big  lug_boot_med  \
107       False      False          True          True         False
901       False       True         False         False         False
1709       True      False         False          True         False
706        True      False         False         False          True
678        True      False         False         False          True

      lug_boot_small  safety_high  safety_low  safety_med
107            False         True       False       False
901             True        False       False        True
1709           False         True       False       False
706            False        False       False        True
678            False        False        True       False

[5 rows x 21 columns]
```

[22]:
```
y_train.head()
```

[22]:
```
107     0
901     0
1709    0
706     0
678     0
Name: class, dtype: int64
```

[23]:
```
X_test.head()
```

[23]:
```
      buying_high  buying_low  buying_med  buying_vhigh  maint_high  \
599          True       False       False         False        True
1201        False       False        True         False       False
628          True       False       False         False        True
1498        False        True       False         False        True
1263        False       False        True         False       False
```

```
      maint_low  maint_med  maint_vhigh  doors_2  doors_3  …  doors_5more  \
599       False      False        False    False    False  …        False
1201       True      False        False     True    False  …        False
628       False      False        False    False    False  …         True
1498      False      False        False    False    False  …         True
1263       True      False        False    False    False  …        False

      persons_2  persons_4  persons_more  lug_boot_big  lug_boot_med  \
599        True      False         False         False          True
1201      False       True         False         False          True
628        True      False         False          True         False
1498      False       True         False         False          True
1263      False      False          True         False          True

      lug_boot_small  safety_high  safety_low  safety_med
599            False         True       False       False
1201           False        False       False        True
628            False        False       False        True
1498           False        False       False        True
1263           False        False        True       False

[5 rows x 21 columns]
```

[24]:
```
y_test.head()
```

[24]:
```
599     0
1201    0
628     0
1498    0
1263    0
Name: class, dtype: int64
```

4 Build a Decision Tree Classifier

4.1 Initialize the classifier

[25]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(
    criterion="entropy",
    max_depth = 5, # 樹的最大深度
    min_samples_split=10, # 最少需要多少樣本才會分裂
    min_samples_leaf=4, # 葉子節點最少需要多少樣本
    random_state=42 # 固定亂數種子，讓每個人都能復現相同的結果
)
```

4.2 Train the model

[26]:
```
model.fit(X_train, y_train)
```

[26]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=4,
                       min_samples_split=10, random_state=42)
```

4.3 Make predictions

[27]:
```
y_pred = model.predict(X_test)
```

5 Step 3: Evaluate the Model

5.1 Evaluate performance

[28]:
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[29]:
```
accuracy_score(y_test, y_pred)
```

[29]:
```
0.9393063583815029
```

[30]:
```
confusion_matrix(y_test, y_pred)
```

[30]:
```
array([[303,  15],
       [  6,  22]])
```

confusion_matrix 顯示：* True Negative (TN)：303 (實際為 bad，預測為 bad) * False Positive (FP)：15 (實際為 bad，預測為 good) * False Negative (FN)：6 (實際為 good，預測為 bad) * True Positive (TP)：22 (實際為 good，預測為 good)

[31]:
```
print(classification_report(y_test, y_pred))
```

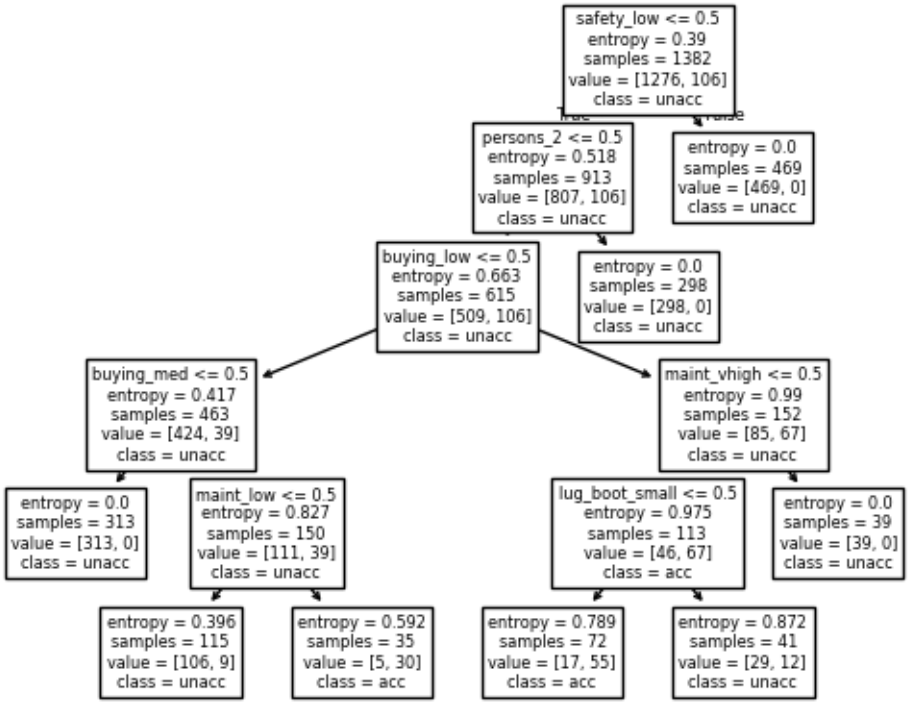
```
              precision    recall  f1-score   support

           0       0.98      0.95      0.97       318
           1       0.59      0.79      0.68        28

    accuracy                           0.94       346
   macro avg       0.79      0.87      0.82       346
weighted avg       0.95      0.94      0.94       346
```

5.2 Visualize

[32]:
```
from sklearn.tree import plot_tree
```

[33]:
```
plot_tree(model, feature_names=X_train.columns, class_names=["unacc", "acc",␣
↪"good", "vgood"])
```

[33]:
```
[Text(0.7, 0.9166666666666666, 'safety_low <= 0.5\nentropy = 0.39\nsamples =
1382\nvalue = [1276, 106]\nclass = unacc'),
 Text(0.6, 0.75, 'persons_2 <= 0.5\nentropy = 0.518\nsamples = 913\nvalue =
[807, 106]\nclass = unacc'),
 Text(0.6499999999999999, 0.8333333333333333, 'True  '),
 Text(0.5, 0.5833333333333334, 'buying_low <= 0.5\nentropy = 0.663\nsamples =
615\nvalue = [509, 106]\nclass = unacc'),
 Text(0.2, 0.4166666666666667, 'buying_med <= 0.5\nentropy = 0.417\nsamples =
463\nvalue = [424, 39]\nclass = unacc'),
 Text(0.1, 0.25, 'entropy = 0.0\nsamples = 313\nvalue = [313, 0]\nclass =
unacc'),
 Text(0.3, 0.25, 'maint_low <= 0.5\nentropy = 0.827\nsamples = 150\nvalue =
[111, 39]\nclass = unacc'),
 Text(0.2, 0.08333333333333333, 'entropy = 0.396\nsamples = 115\nvalue = [106,
9]\nclass = unacc'),
 Text(0.4, 0.08333333333333333, 'entropy = 0.592\nsamples = 35\nvalue = [5,
30]\nclass = acc'),
 Text(0.8, 0.4166666666666667, 'maint_vhigh <= 0.5\nentropy = 0.99\nsamples =
152\nvalue = [85, 67]\nclass = unacc'),
 Text(0.7, 0.25, 'lug_boot_small <= 0.5\nentropy = 0.975\nsamples = 113\nvalue =
[46, 67]\nclass = acc'),
 Text(0.6, 0.08333333333333333, 'entropy = 0.789\nsamples = 72\nvalue = [17,
55]\nclass = acc'),
 Text(0.8, 0.08333333333333333, 'entropy = 0.872\nsamples = 41\nvalue = [29,
12]\nclass = unacc'),
 Text(0.9, 0.25, 'entropy = 0.0\nsamples = 39\nvalue = [39, 0]\nclass = unacc'),
 Text(0.7, 0.5833333333333334, 'entropy = 0.0\nsamples = 298\nvalue = [298,
0]\nclass = unacc'),
 Text(0.8, 0.75, 'entropy = 0.0\nsamples = 469\nvalue = [469, 0]\nclass =
unacc'),
 Text(0.75, 0.8333333333333333, '  False')]
```



6 Step 4: Hyperparameter Tuning

6.1 Tune hyperparameters

[34]:
```
from sklearn.model_selection import GridSearchCV
```

[35]:
```
param_grid = {
    "criterion": ["gini", "entropy"],
    "max_depth": [3, 4, 5, 6, 7, 8],
    "min_samples_split": [5, 10, 15, 20],
    "min_samples_leaf": [2, 4, 6, 8]
}
```

[36]:
```
# 我們使用網格搜尋 (GridSearchCV) 來進行 hyperparameter 調整。
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=5,
    scoring="accuracy",
    n_jobs=-1, # 使用所有 CPU 核心
)
```

[37]:
```
grid_search.fit(X_train, y_train)
```

[37]:
```
GridSearchCV(cv=5,
             estimator=DecisionTreeClassifier(criterion='entropy', max_depth=5,
                                              min_samples_leaf=4,
                                              min_samples_split=10,
                                              random_state=42),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 6, 7, 8],
                         'min_samples_leaf': [2, 4, 6, 8],
                         'min_samples_split': [5, 10, 15, 20]},
             scoring='accuracy')
```

6.2 Evaluate the tuned model

[38]:
```
print("最佳參數:")
print(*grid_search.best_params_.items(), sep="\n", end="\n\n")  # 換行輸出 dict
print("最佳分數:", grid_search.best_score_)
```

```
最佳參數:
('criterion', 'entropy')
('max_depth', 7)
('min_samples_leaf', 2)
('min_samples_split', 5)

最佳分數: 0.9891435148851567
```

[39]:
```
# 使用最佳的參數來預測
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

[40]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       318
           1       0.68      1.00      0.81        28

    accuracy                           0.96       346
   macro avg       0.84      0.98      0.90       346
weighted avg       0.97      0.96      0.97       346
```

7 Step 5: Report Results

本次作業成功使用 Scikit-learn 的 DecisionTreeClassifier 對汽車評估資料集進行了二元分類 (good/bad)。

- **資料預處理:** 我們對類別型特徵進行了 One-Hot Encoding，並將目標變數映射為 0 和 1。

- **模型建立與評估:** 初始決策樹模型在測試集上達到了約 94% 的準確率。
- **Hyperparameter 調整:** 透過 GridSearchCV 找到了更優的超參數組合 (criterion='entropy', max_depth=7, min_samples_leaf=2, min_samples_split=5)，將測試集準確率提升至 98.9%。
- **特徵重要性:** 分析顯示 maint 和 buying 是影響分類最重要的兩個特徵。

7.1 Bonus: Feature importance analysis

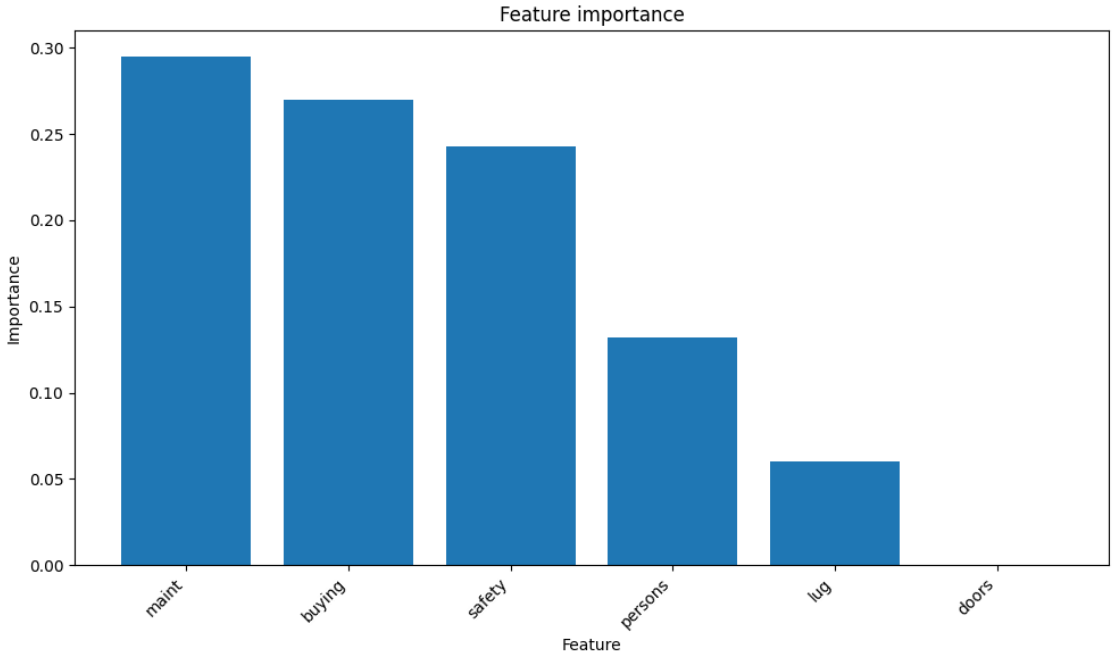
[41]:
```
import matplotlib.pyplot as plt
```

[42]:
```
# 由於我們做了 One-hot Encoding，需要把實際上相同的 Feature 總和在一起
# 才能判斷在原資料中各個 Feature 的重要性
feature_groups = {}
```

[43]:
```
for feature, importance in zip(X_train.columns, best_model.
 ↪feature_importances_):
    original_feature = feature.split('_')[0]
    if original_feature not in feature_groups:
        feature_groups[original_feature] = 0
    feature_groups[original_feature] += importance
```

[44]:
```
grouped_importance = pd.DataFrame({
    "feature": feature_groups.keys(),
    "importance": feature_groups.values()
}).sort_values("importance", ascending=False) # 依重要度排序
```

[45]:
```
# 視覺化
plt.figure(figsize=(10, 6))
plt.bar(grouped_importance["feature"], grouped_importance["importance"])
plt.xticks(rotation=45, ha="right")
plt.title("Feature importance")
plt.xlabel("Feature")
plt.ylabel("Importance")
plt.tight_layout()
plt.show()
```



根據 Feature importance 長條圖，可以看出 maint (維護成本) 是影響汽車分類最重要的特徵，其次是 buying (購買價格)。safety (安全性) 和 persons (乘坐人數) 也有一定的影響力，而 lug_boot (行李箱大小) 和 doors (車門數) 的重要性相對較低。

這個結果符合直覺，通常購買價格、安全性、乘坐人數等因素會影響汽車的分類。

[ ]: