

决策树：理论与实践

2025 年 9 月 9 日

1 引言

决策树（Decision Tree）通过递归划分特征空间，形成分段常数的预测模型。其优点是可解释性强、对数据预处理要求低，并能处理非线性边界。

2 原理与公式

以分类树为例，在每个节点选择能够最大化“纯度提升”的划分。设节点数据集为 \mathcal{D} ，类别占比为 p_k 。常见纯度指标有基尼与熵：

$$\text{Gini}(\mathcal{D}) = 1 - \sum_k p_k^2, \quad (1)$$

$$\text{Entropy}(\mathcal{D}) = - \sum_k p_k \log p_k. \quad (2)$$

若划分为左右子节点 L, R ，则划分后的纯度为

$$I_{\text{split}} = \frac{|L|}{|\mathcal{D}|} I(L) + \frac{|R|}{|\mathcal{D}|} I(R), \quad (3)$$

最优划分使得 $\Delta I = I(\mathcal{D}) - I_{\text{split}}$ 最大。停止条件常包括：最大深度、叶子最小样本数、最小纯度提升等。

3 应用与技巧

- **优点：**可解释、能处理非线性、对特征尺度不敏感、可处理类别与数值特征（需编码）。
- **缺点：**容易过拟合、方差较大；可用集成方法缓解。
- **正则化：**调整 `max_depth`、`min_samples_leaf`，或使用复杂度剪枝。
- **基线对比：**与逻辑回归、SVM、随机森林等模型对比评估。

4 Python 实战

在本章节目录运行下述命令，图片将保存到本目录的 figures/:

Listing 1: 生成决策树配图

```
1 python gen_decision_tree_figures.py
```

Listing 2: gen_decision_tree_figures.py 源码

```
1 """
2 Figure generator for the Decision Tree chapter.
3
4 Generates illustrative figures and saves them into the chapter's '
   figures/'
5 folder next to this script, regardless of current working directory.
6
7 Requirements:
8 - Python 3.8+
9 - numpy, matplotlib, scikit-learn
10
11 Install (if needed):
12     pip install numpy matplotlib scikit-learn
13
14 This script avoids newer or experimental APIs for broader compatibility
15 .
16 """
17
18 from __future__ import annotations
19
20 import os
21 import numpy as np
22 import matplotlib.pyplot as plt
23 from matplotlib.colors import ListedColormap
24
25 try:
26     from sklearn.datasets import make_moons, make_classification
27     from sklearn.tree import DecisionTreeClassifier, plot_tree
28     from sklearn.ensemble import RandomForestClassifier
29 except Exception as e:
30     raise SystemExit(
31         "Missing scikit-learn. Please install with: pip install scikit-learn"
32     )
```

```

33 def _ensure_figures_dir(path: str | None = None) -> str:
34     """Create figures directory under this chapter regardless of CWD.
35     """
36     if path is None:
37         base = os.path.dirname(os.path.abspath(__file__))
38         path = os.path.join(base, "figures")
39     os.makedirs(path, exist_ok=True)
40     return path
41
42 def _plot_decision_boundary(ax, clf, X, y, title: str):
43     x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
44     y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
45     xx, yy = np.meshgrid(
46         np.linspace(x_min, x_max, 400), np.linspace(y_min, y_max, 400)
47     )
48     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
49     cmap_light = ListedColormap(["#FFEEEE", "#EEEEFF"])
50     cmap_bold = ListedColormap(["#E74C3C", "#3498DB"])
51     ax.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.8, levels=np.unique(
52         Z).size)
53     ax.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolors="k", s
54               =20)
55     ax.set_title(title)
56     ax.set_xlabel("Feature 1")
57     ax.set_ylabel("Feature 2")
58
59 def fig_dt_decision_boundary_2class(out_dir: str) -> str:
60     np.random.seed(0)
61     X, y = make_moons(n_samples=400, noise=0.25, random_state=0)
62     clf = DecisionTreeClassifier(max_depth=4, random_state=0)
63     clf.fit(X, y)
64
65     fig, ax = plt.subplots(figsize=(5.5, 4.5), dpi=150)
66     _plot_decision_boundary(ax, clf, X, y, "Decision Tree boundary (
67         max_depth=4)")
68     out_path = os.path.join(out_dir, "dt_decision_boundary_2class.png")
69     fig.tight_layout()
70     fig.savefig(out_path)
71     plt.close(fig)
72     return out_path

```

```
72
73 def fig_dt_depth_compare(out_dir: str) -> str:
74     np.random.seed(1)
75     X, y = make_moons(n_samples=500, noise=0.3, random_state=1)
76     models = [
77         (DecisionTreeClassifier(max_depth=3, random_state=1), "
78             max_depth=3"),
79         (DecisionTreeClassifier(random_state=1), "max_depth=None (deep)
80             ")
81     ]
82     fig, axes = plt.subplots(1, 2, figsize=(9.5, 4.2), dpi=150, sharex=
83         True, sharey=True)
84     for ax, (m, title) in zip(axes, models):
85         m.fit(X, y)
86         _plot_decision_boundary(ax, m, X, y, f"Decision Tree: {title}")
87     fig.suptitle("Depth and overfitting")
88     out_path = os.path.join(out_dir, "dt_depth_compare.png")
89     fig.tight_layout(rect=[0, 0.03, 1, 0.95])
90     fig.savefig(out_path)
91     plt.close(fig)
92     return out_path
93
94 def fig_dt_feature_importances(out_dir: str) -> str:
95     X, y = make_classification(
96         n_samples=600,
97         n_features=8,
98         n_informative=3,
99         n_redundant=2,
100         n_repeated=0,
101         random_state=7,
102         shuffle=True,
103     )
104     clf = DecisionTreeClassifier(max_depth=5, random_state=7)
105     clf.fit(X, y)
106     importances = clf.feature_importances_
107
108     fig, ax = plt.subplots(figsize=(6.5, 3.8), dpi=160)
109     idx = np.arange(importances.size)
110     ax.bar(idx, importances, color="#3498DB")
111     ax.set_xticks(idx)
112     ax.set_xticklabels([f"f{i}" for i in idx])
113     ax.set_ylabel("importance")
```

```

112     ax.set_title("Decision Tree feature importances")
113     ax.set_ylim(0, max(0.25, importances.max() + 0.05))
114     for i, v in enumerate(importances):
115         ax.text(i, v + 0.01, f"{v:.2f}", ha="center", va="bottom",
116                 fontsize=8)
117     out_path = os.path.join(out_dir, "dt_feature_importances.png")
118     fig.tight_layout()
119     fig.savefig(out_path)
120     plt.close(fig)
121     return out_path
122
123 def fig_dt_vs_rf_boundary(out_dir: str) -> str:
124     np.random.seed(2)
125     X, y = make_moons(n_samples=500, noise=0.3, random_state=2)
126     dt = DecisionTreeClassifier(max_depth=5, random_state=2).fit(X, y)
127     rf = RandomForestClassifier(n_estimators=100, random_state=2).fit(X
128                                , y)
129
130     fig, axes = plt.subplots(1, 2, figsize=(9.5, 4.2), dpi=150, sharex=
131                             True, sharey=True)
132     _plot_decision_boundary(axes[0], dt, X, y, "Decision Tree")
133     _plot_decision_boundary(axes[1], rf, X, y, "Random Forest")
134     fig.suptitle("Decision Tree vs Random Forest")
135     out_path = os.path.join(out_dir, "dt_vs_rf_boundary.png")
136     fig.tight_layout(rect=[0, 0.03, 1, 0.95])
137     fig.savefig(out_path)
138     plt.close(fig)
139     return out_path
140
141 def fig_dt_tree_plot(out_dir: str) -> str:
142     # Small depth to keep the plot readable
143     X, y = make_moons(n_samples=200, noise=0.25, random_state=3)
144     clf = DecisionTreeClassifier(max_depth=3, random_state=3).fit(X, y)
145
146     fig, ax = plt.subplots(figsize=(10, 6), dpi=150)
147     plot_tree(clf, filled=True, feature_names=["x1", "x2"], class_names
148              =["0", "1"], ax=ax)
149     ax.set_title("Decision Tree (max_depth=3)")
150     out_path = os.path.join(out_dir, "dt_tree_plot.png")
151     fig.tight_layout()
152     fig.savefig(out_path)

```

```
151     plt.close(fig)
152     return out_path
153
154
155 def main():
156     out_dir = _ensure_figures_dir(None)
157     generators = [
158         fig_dt_decision_boundary_2class,
159         fig_dt_depth_compare,
160         fig_dt_feature_importances,
161         fig_dt_vs_rf_boundary,
162         fig_dt_tree_plot,
163     ]
164     print("Generating figures into:", os.path.abspath(out_dir))
165     for gen in generators:
166         try:
167             p = gen(out_dir)
168             print("Saved:", p)
169         except Exception as e:
170             print("Failed generating", gen.__name__, ":", e)
171
172
173 if __name__ == "__main__":
174     main()
```

5 结果

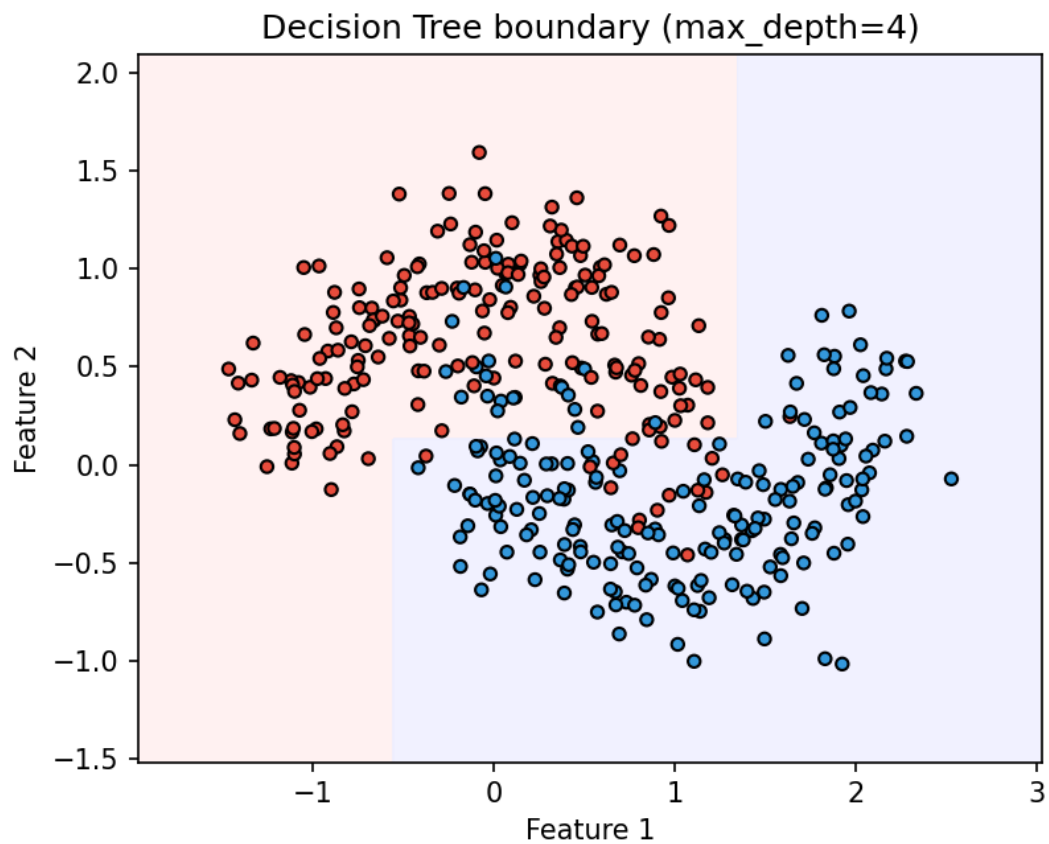


图 1: 决策树在两类数据上的决策边界。

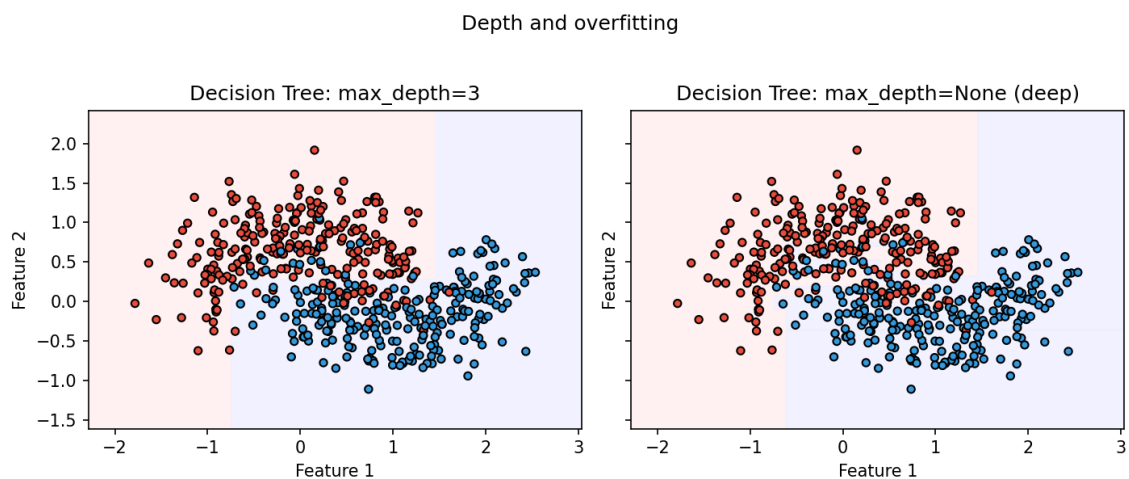


图 2: 树深度影响：浅层与深层（过拟合）对比。

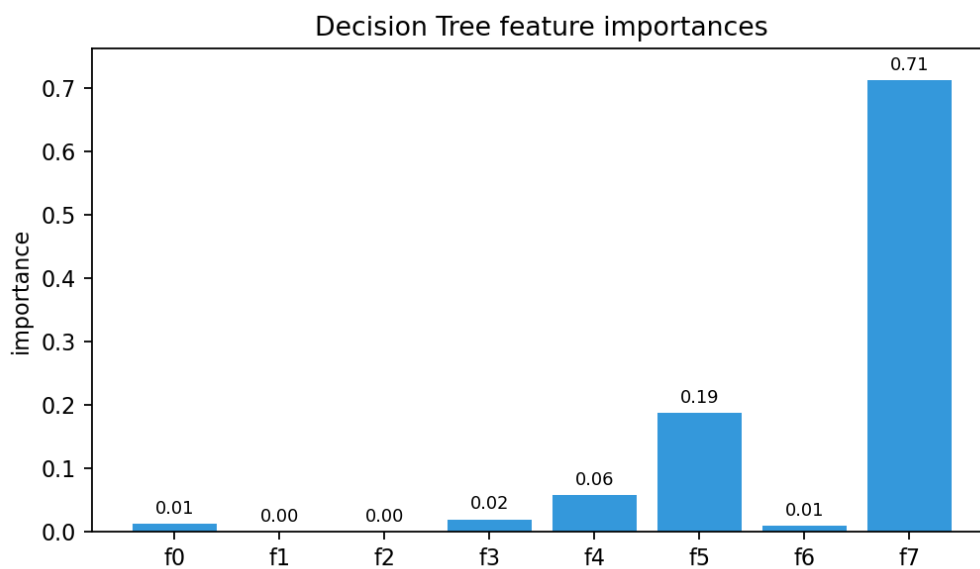


图 3: 决策树的特征重要性可视化。

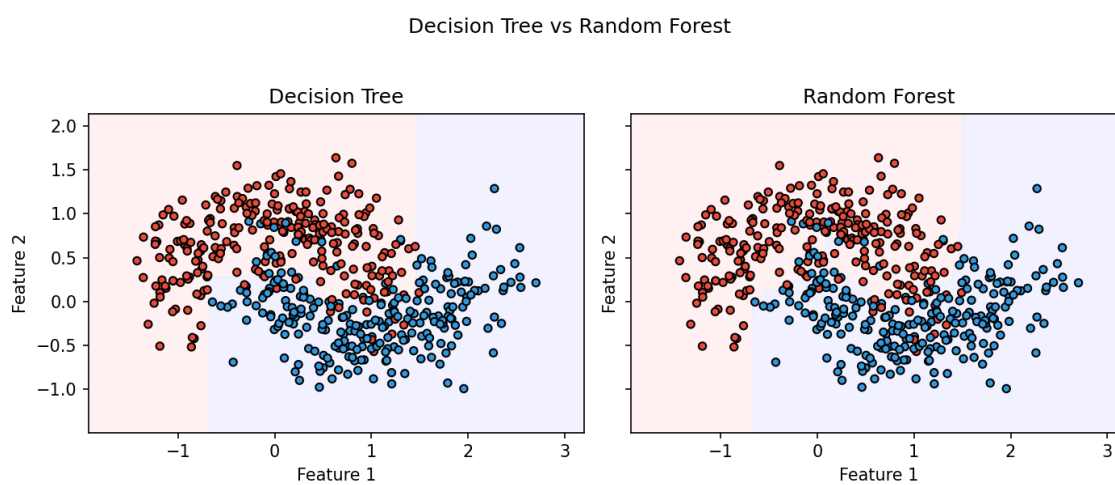


图 4: 单棵决策树与随机森林的决策边界对比。

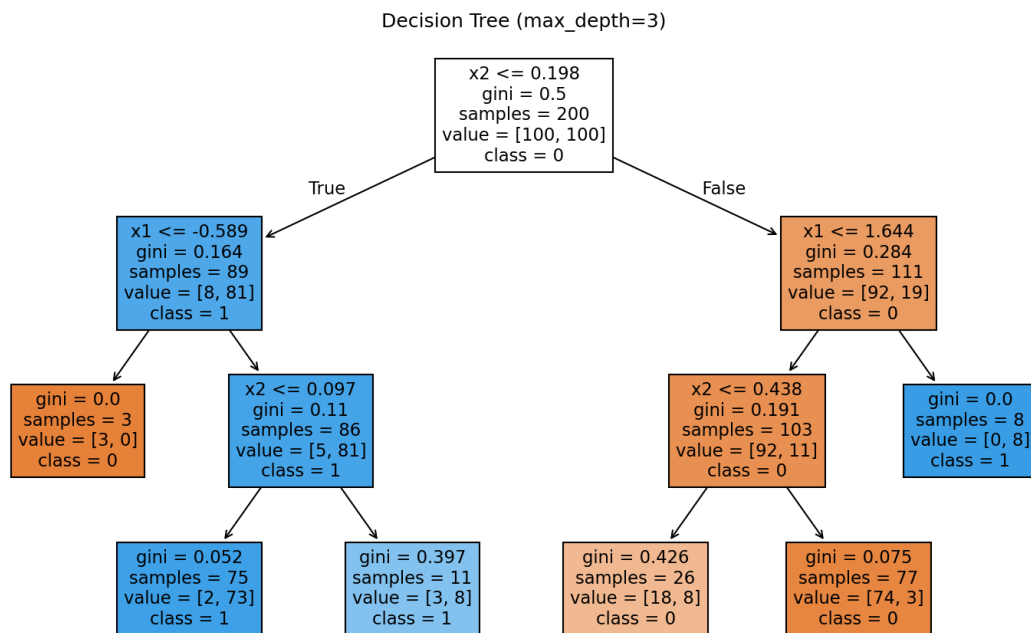


图 5: 树结构可视化 (max_depth=3)。

6 总结

决策树作为可解释且灵活的基线模型，在适当的正则化或与集成方法（随机森林、梯度提升）结合时，能在多种任务上取得具有竞争力的表现。