DecissionTree

May 31, 2026

1 Decision Tree with BREAST CANCER dataset as example

1.1 Section 1: Imports

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_text
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

1.2 Section 2: Load Data

[2]:
```
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

print("Shape:", X.shape)
print("Target classes:", np.unique(y))
```

```
Shape: (569, 30)
Target classes: [0 1]
```

1.3 Section 3: Train-Test Split

[3]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42, stratify=y)
```

1.4 Section 4: Baseline Decision Tree

[4]:
```
dt_base = DecisionTreeClassifier(random_state=42)
dt_base.fit(X_train, y_train)

y_pred_base = dt_base.predict(X_test)
print("Baseline Accuracy:", round(accuracy_score(y_test, y_pred_base), 4))
```

```
print("Baseline Confusion Matrix:\n", confusion_matrix(y_test, y_pred_base))
```

```
Baseline Accuracy: 0.9123
Baseline Confusion Matrix:
 [[39  3]
 [ 7 65]]
```

1.5 Section 5: Hyperparameter Tuning

[5]:
```
param_grid = {
    "max_depth": [3, 4, 5, 6, None],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4, 8],
    "criterion": ["gini", "entropy"]
}

grid = GridSearchCV(
    DecisionTreeClassifier(random_state=42),
    param_grid=param_grid,
    scoring="f1",
    cv=5,
    n_jobs=-1
)
grid.fit(X_train, y_train)

best_dt = grid.best_estimator_
y_pred_best = best_dt.predict(X_test)

print("Best Params:", grid.best_params_)
print("Tuned Accuracy:", round(accuracy_score(y_test, y_pred_best), 4))
print("Classification Report:\n", classification_report(y_test, y_pred_best))
```

```
Best Params: {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1,
'min_samples_split': 5}
Tuned Accuracy: 0.9211
Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.93      0.90        42
           1       0.96      0.92      0.94        72

    accuracy                           0.92       114
   macro avg       0.91      0.92      0.92       114
weighted avg       0.92      0.92      0.92       114
```

1.6 Section 6: Cost-Complexity Pruning (ccp_alpha)

[6]:
```
path = best_dt.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

alpha_scores = []
for alpha in ccp_alphas:
    dt_alpha = DecisionTreeClassifier(
        random_state=42,
        ccp_alpha=alpha,
        max_depth=grid.best_params_["max_depth"],
        min_samples_split=grid.best_params_["min_samples_split"],
        min_samples_leaf=grid.best_params_["min_samples_leaf"],
        criterion=grid.best_params_["criterion"]
    )
    dt_alpha.fit(X_train, y_train)
    alpha_scores.append((alpha, dt_alpha.score(X_test, y_test)))

best_alpha, best_alpha_score = max(alpha_scores, key=lambda x: x[1])
print("Best ccp_alpha:", best_alpha)
print("Best pruned accuracy:", round(best_alpha_score, 4))
```

```
Best ccp_alpha: 0.003956043956043957
Best pruned accuracy: 0.9474
```

1.7 Section 7: Feature Importance and Rule Export

[7]:
```
dt_final = DecisionTreeClassifier(
    random_state=42,
    ccp_alpha=best_alpha,
    max_depth=grid.best_params_["max_depth"],
    min_samples_split=grid.best_params_["min_samples_split"],
    min_samples_leaf=grid.best_params_["min_samples_leaf"],
    criterion=grid.best_params_["criterion"]
)
dt_final.fit(X_train, y_train)

importances = pd.Series(dt_final.feature_importances_, index=X.columns)
print(importances.sort_values(ascending=False).head(10))

print("\nTop rules (depth limited):")
print(export_text(dt_final, feature_names=list(X.columns), max_depth=3))
```

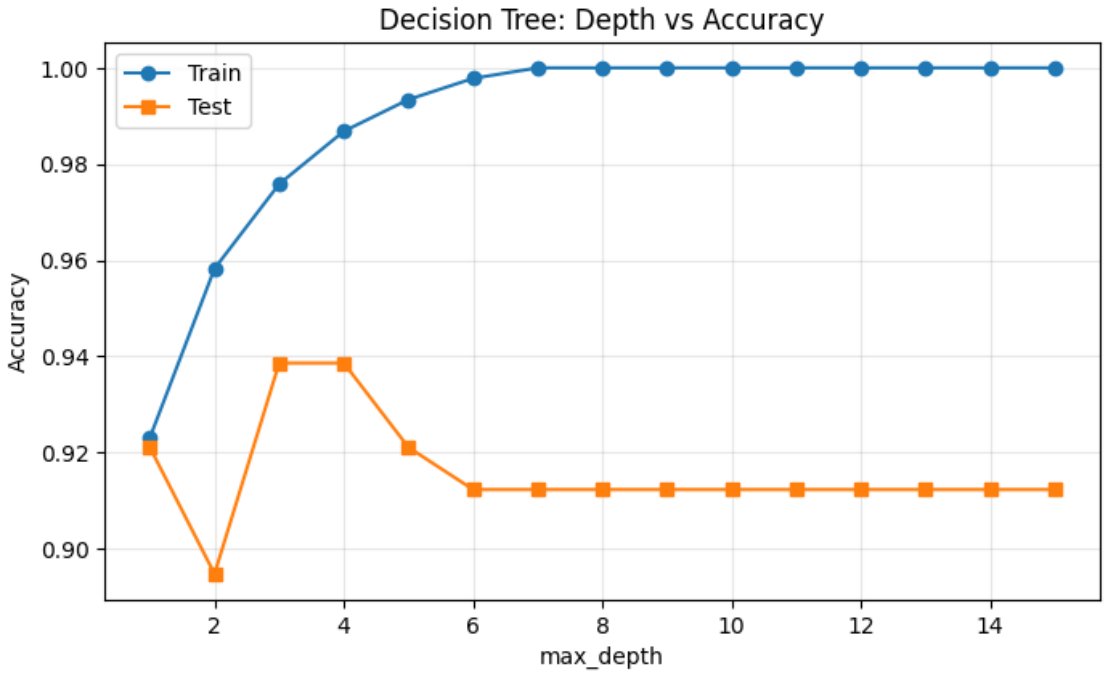
```
worst radius            0.735871
worst concave points    0.122414
texture error           0.045930
worst texture           0.044306
worst concavity         0.017216
```

```
worst smoothness        0.013369
worst symmetry          0.011318
area error              0.009577
mean compactness        0.000000
mean smoothness         0.000000
dtype: float64

Top rules (depth limited):
|--- worst radius <= 16.80
|   |--- worst concave points <= 0.14
|   |   |--- area error <= 91.56
|   |   |   |--- class: 1
|   |   |--- area error >  91.56
|   |   |   |--- class: 0
|   |--- worst concave points >  0.14
|   |   |--- worst texture <= 25.62
|   |   |   |--- worst smoothness <= 0.18
|   |   |   |   |--- class: 1
|   |   |   |--- worst smoothness >  0.18
|   |   |   |   |--- class: 0
|   |   |--- worst texture >  25.62
|   |   |   |--- worst symmetry <= 0.27
|   |   |   |   |--- class: 1
|   |   |   |--- worst symmetry >  0.27
|   |   |   |   |--- class: 0
|--- worst radius >  16.80
|   |--- texture error <= 0.47
|   |   |--- class: 1
|   |--- texture error >  0.47
|   |   |--- worst concavity <= 0.19
|   |   |   |--- worst texture <= 30.98
|   |   |   |   |--- class: 1
|   |   |   |--- worst texture >  30.98
|   |   |   |   |--- class: 0
|   |   |--- worst concavity >  0.19
|   |   |   |--- class: 0
```

1.8 Graph 1: Train vs Test Accuracy by Depth

[8]:
```python
depths = list(range(1, 16))
train_scores = []
test_scores = []

for d in depths:
    model = DecisionTreeClassifier(max_depth=d, random_state=42)
    model.fit(X_train, y_train)
```

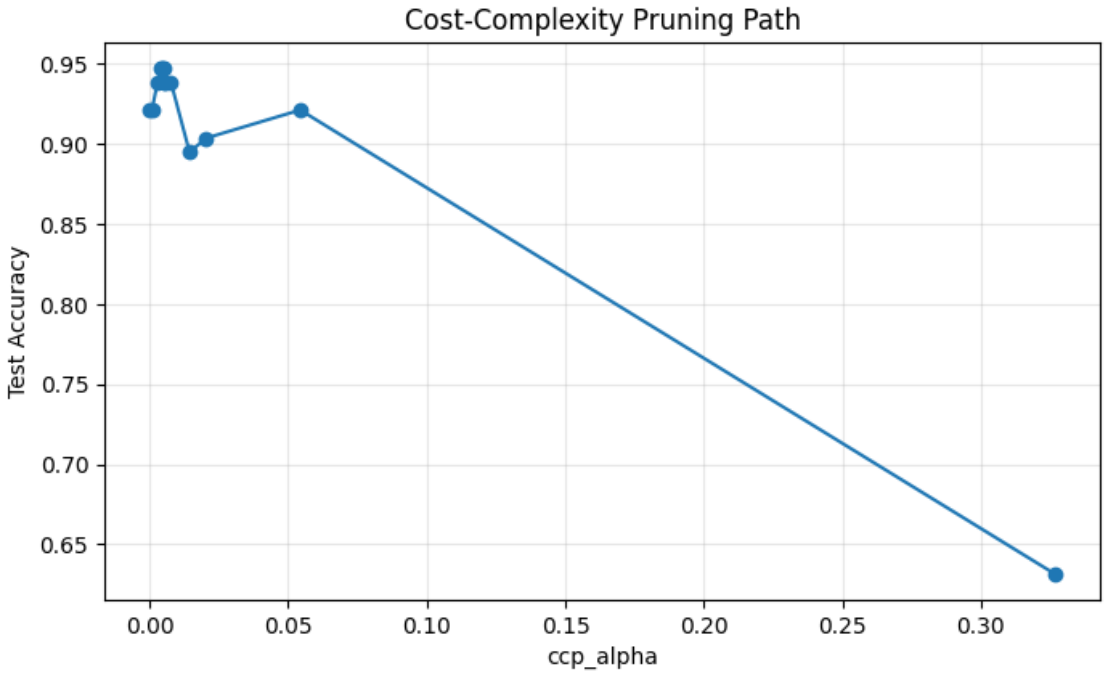
```
    train_scores.append(model.score(X_train, y_train))
    test_scores.append(model.score(X_test, y_test))

plt.figure(figsize=(8, 4.5))
plt.plot(depths, train_scores, marker="o", label="Train")
plt.plot(depths, test_scores, marker="s", label="Test")
plt.xlabel("max_depth")
plt.ylabel("Accuracy")
plt.title("Decision Tree: Depth vs Accuracy")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



1.9 Graph 2: Pruning Curve (ccp_alpha vs Test Accuracy)

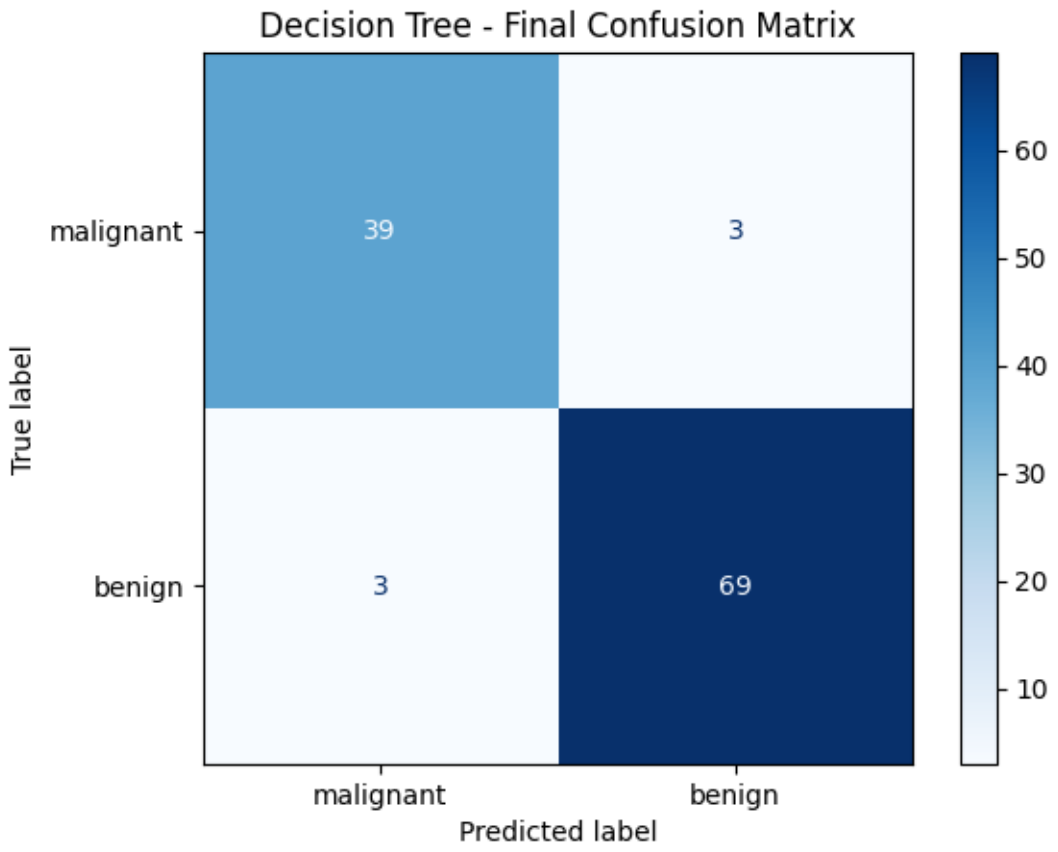
[9]:
```
alphas = [a for a, _ in alpha_scores]
scores = [s for _, s in alpha_scores]

plt.figure(figsize=(8, 4.5))
plt.plot(alphas, scores, marker="o")
plt.xlabel("ccp_alpha")
plt.ylabel("Test Accuracy")
plt.title("Cost-Complexity Pruning Path")
plt.grid(alpha=0.3)
```

```
plt.show()
```



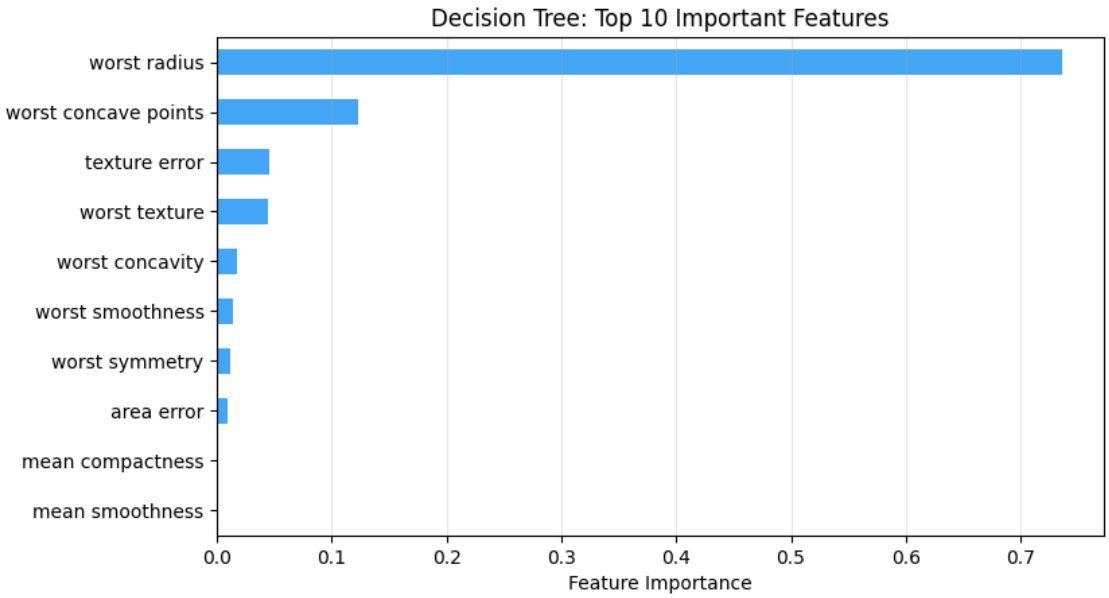
1.10 Graph 3: Confusion Matrix of Final Pruned Tree

[10]:
```
from sklearn.metrics import ConfusionMatrixDisplay

y_pred_final = dt_final.predict(X_test)
ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred_final, display_labels=data.target_names, cmap="Blues",␣
 ↪values_format="d"
)
plt.title("Decision Tree - Final Confusion Matrix")
plt.show()
```



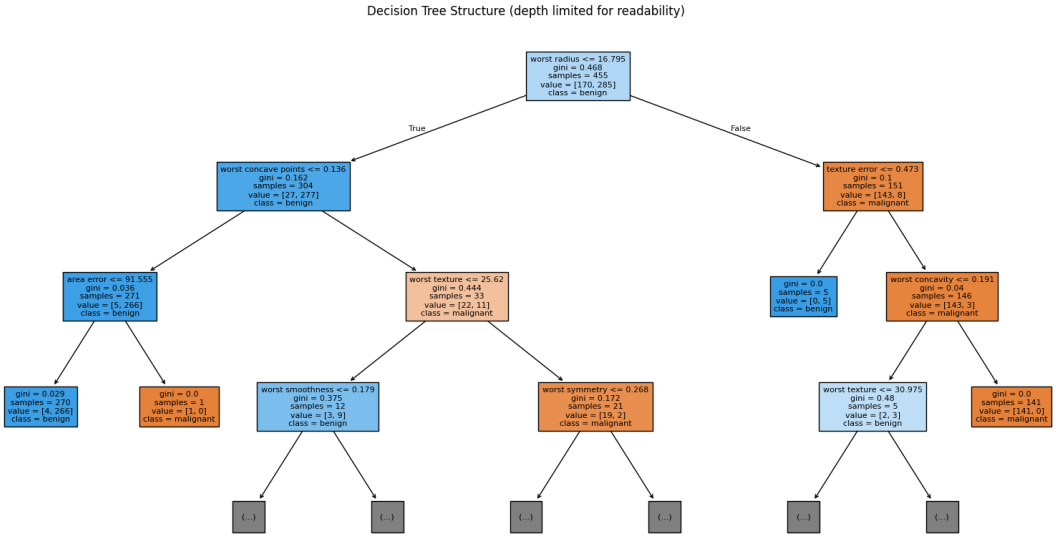
1.11 Graph 4: Top Feature Importances (Same Dataset)

[14]:
```
importances = pd.Series(dt_final.feature_importances_, index=X.columns)
top10 = importances.sort_values(ascending=False).head(10)

plt.figure(figsize=(8.5, 4.8))
top10.sort_values().plot(kind="barh", color="#42a5f5")
plt.xlabel("Feature Importance")
plt.title("Decision Tree: Top 10 Important Features")
plt.grid(axis="x", alpha=0.3)
plt.show()
```



1.12 Graph 5: Tree Structure (Top Levels)

[11]:
```
plt.figure(figsize=(20, 10))
plot_tree(
    dt_final,
    feature_names=X.columns,
    class_names=data.target_names,
    filled=True,
    max_depth=3,
    fontsize=8
)
plt.title("Decision Tree Structure (depth limited for readability)")
plt.show()
```


Decision Tree Structure (depth limited for readability)
worst radius <= 16.795
gini = 0.468
samples = 455
value = [170, 285]
class = benign
True
False
worst concave points <= 0.136
gini = 0.162
samples = 304
value = [27, 277]
class = benign
texture error <= 0.473
gini = 0.1
samples = 151
value = [143, 8]
class = malignant
area error <= 91.555
gini = 0.036
samples = 271
value = [5, 266]
class = benign
worst texture <= 25.62
gini = 0.444
samples = 33
value = [22, 11]
class = malignant
gini = 0.0
samples = 5
value = [0, 5]
class = benign
worst concavity <= 0.191
gini = 0.04
samples = 146
value = [143, 3]
class = malignant
gini = 0.029
samples = 270
value = [4, 266]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
worst smoothness <= 0.179
gini = 0.375
samples = 12
value = [3, 9]
class = benign
worst symmetry <= 0.268
gini = 0.172
samples = 21
value = [19, 2]
class = malignant
worst texture <= 30.975
gini = 0.48
samples = 5
value = [2, 3]
class = benign
gini = 0.0
samples = 141
value = [141, 0]
class = malignant
(...)
(...)
(...)
(...)
(...)
(...)

[ ]: