Understanding and Visualizing Decision Trees:

How Depth Affects Model Performance

---

1. Introduction:

Decision Trees are a foundational tool in machine learning, known for their simplicity, interpretability, and versatility in both classification and regression problems. However, one of the key factors that governs their effectiveness is tree depth—a hyperparameter that directly affects how the model generalizes from training data.

In this tutorial, we use the **Wine Quality dataset** to explore the impact of decision tree depth on model performance. Through visualization, analysis and real-world modelling, we demonstrate how to balance simplicity with accuracy – and how depth can either empower or impair your model.

The full code and visual assets are available on GitHub:

---

2. Dataset Overview

We use the **Wine Quality (Red Wine)** dataset from the UCI Machine Learning Repository. It contains physicochemical properties of red wine samples along with a quality rating (0-10) given by experts.

- **11 numerical features**: Physicochemical properties (e.g., alcohol content, pH, sulphates).
- **Target variable:** Quality score (3–8, integer values).

```
wine_df['quality_class'] = wine_df['quality'].apply(lambda x: 1 if x >=7 else 0)
```

Fig 1: converting to binary classification

To simplify the classification task, we binarize the quality:

- Class 1 (Good wine): Scores ≥ 7 (~14% of data).
- Class 0 (Bad wine): Scores < 7 (86% of data).

This binary transformation addresses class imbalance while maintaining predictive utility for winemaking applications.

---

3. Importing Necessary Libraries

To implement and analyze the decision tree model, we begin by importing essential Python libraries. Each serves a specific role in data handling, model building, evaluation, and visualization. Used Google Colab to execute the python code.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

Fig 2: Importing Libraries

---

4. Initial Exploratory Data Analysis

a. Count of Quality Ratings

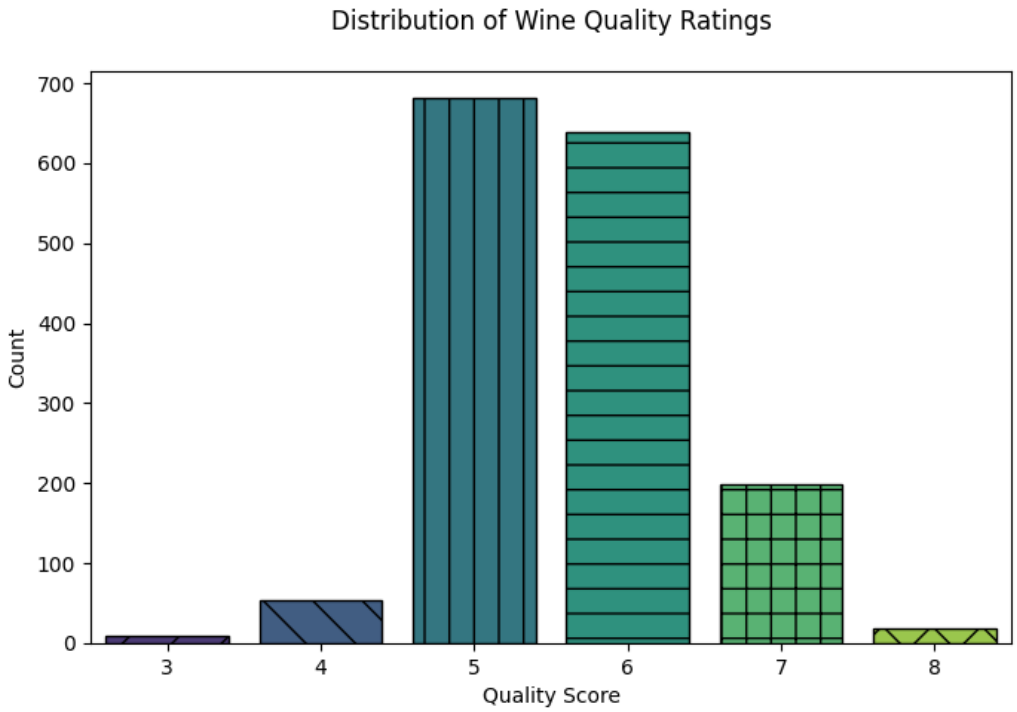

Fig 2: Bar Chart

A bar chart of original quality ratings shows an imbalanced distribution. Most wines fall between scores of 5 and 6.

b. Feature Correlation

A heatmap of feature correlations reveals relationships like:

- Alcohol positively correlates with quality (+0.48).
- Volatile acidity negatively correlates with quality (-0.39).

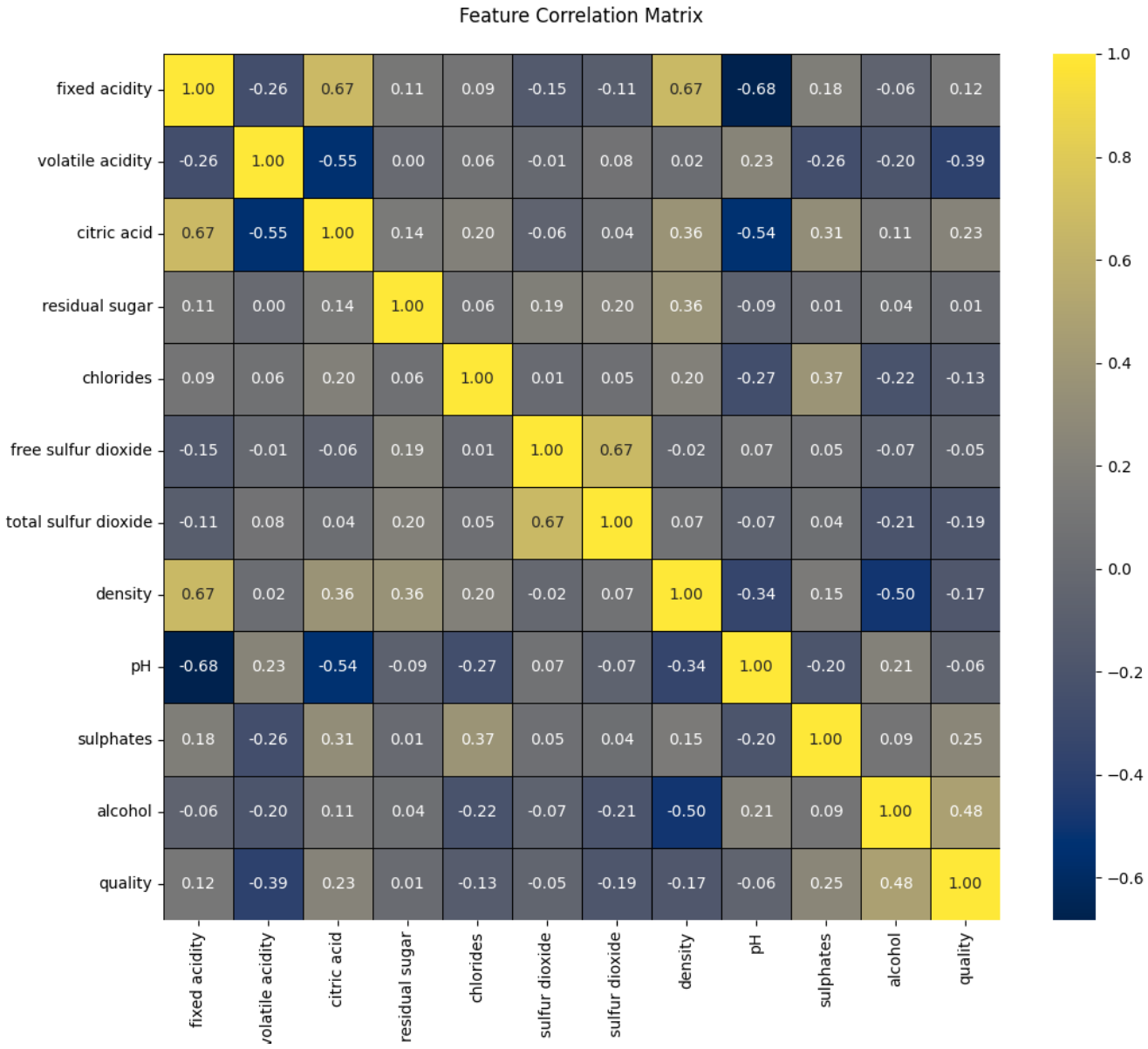

Fig 3: Heatmap

---

5. Preprocessing

Data preparation ensures robust model performance:

- **Scaling:** Features are standardized using StandardScaler (mean=0, variance=1) to ensure equal weighting during splits.
- **Train-test split:** 80% training (1,279 samples), 20% testing (320 samples) with stratified sampling to preserve class ratios.

- **Missing values:** No missing data detected, eliminating the need for imputation.

These steps mitigate biases and enhance reproducibility.

6. Decision Tree Fundamentals

Decision trees make predictions by learning hierarchical rules:

- **Splitting criterion**: Nodes split using metrics like Gini impurity or entropy to maximize class separation.
- **Stopping conditions**: Controlled by parameters like max depth, min samples split, or min_impurity_decrease.
- **Depth impact:**
  - Depth=1: A single split (stump) often underfits (e.g.," If alcohol ≥ 12.5%, predict Good").
  - Depth=10: Complex trees with many splits may overfit by capturing noise (e.g., memorizing specific samples).

---

7. Visualizing Model Performance vs. Tree Depth

We train decision trees with depths 1–10 and evaluate accuracy:

Depth	Training Accuracy	Testing Accuracy
1	0.8686	0.8469
2	0.8937	0.8625
3	0.9030	0.8469
5	0.9382	0.8438
6	0.9586	0.8438
10	0.9891	0.8500

Table 1: Training vs. test accuracy across depths.

Testing accuracy peaks at depth = 2.

Observations:

- Underfitting (Depth 1): Both accuracies are low (~84%), indicating insufficient model complexity.
  Model is too simple; both training and test accuracy are relatively low.
- Optimal depth (2): Testing accuracy peaks at 86.2% (depth=2), Good balance between bias and variance.
- Overfitting (Depth ≥ 3): Training accuracy increases significantly, but test accuracy stagnates or even drops — indicating the model is memorizing the training data rather than generalizing.

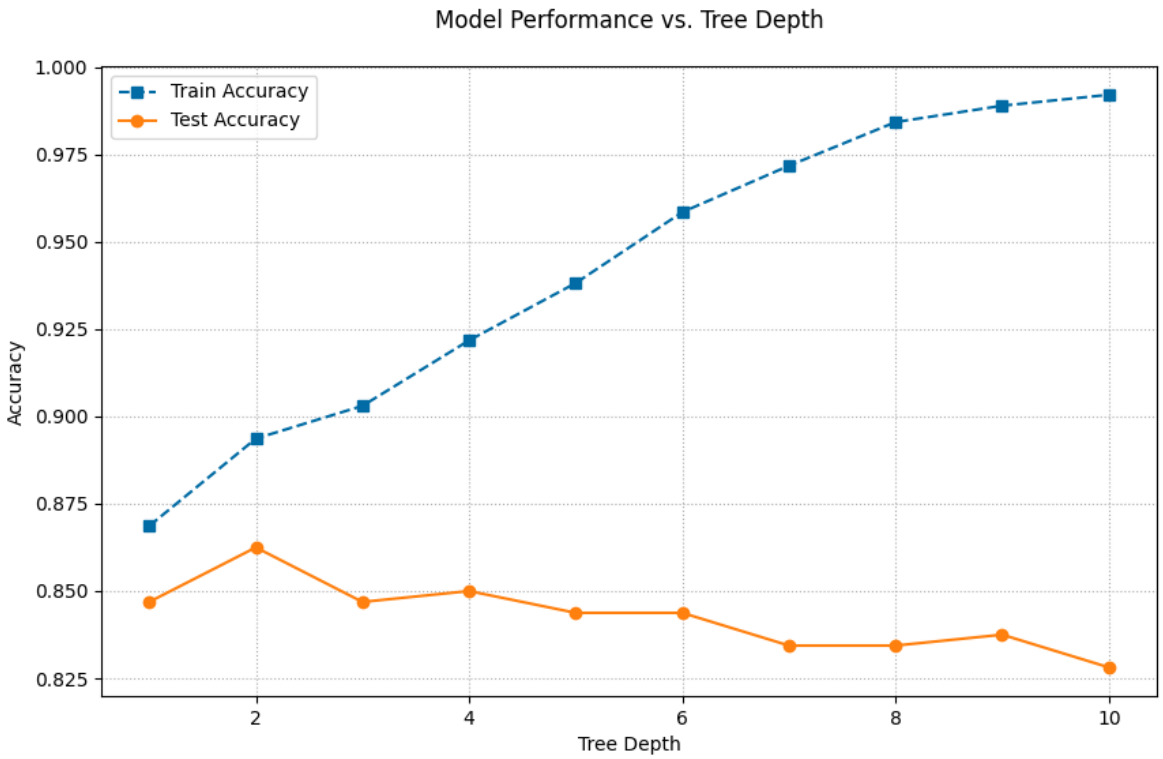

Fig 4: Model Performance vs. Tree Depth

---

8. Hyperparameter Tuning with GridSearchCV

To automate parameter selection, we use GridSearchCV with 5-fold cross-validation:

- **Parameters searched:**
  - – max depth: 1–10
  - – min_samples_split: 2, 5, 10
  - – criterion: Gini impurity or entropy
- **Best parameters:**

{'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2}

The **cross-validation accuracy score** of **0.8881** is the highest, indicating this set of hyperparameters provides the best generalization on unseen data (as measured by 5-fold cross-validation).

- **Why Gini?**
  - **Gini over Entropy:** Although Entropy is theoretically better for handling class imbalance, Gini gave the highest cross-validation score (0.8881) on this dataset. This means it generalized better for this specific data.
  - **Optimal Depth = 2:** Depth 2 strikes the right balance — it avoids underfitting (like depth 1) and overfitting (like depths 6 or 10), while still capturing key patterns.
  - **Model Efficiency:** GridSearchCV shows that a simple, shallow tree with Gini and depth 2 gives the best results — accurate and not overly complex.

9. Visualizing the Optimal Tree

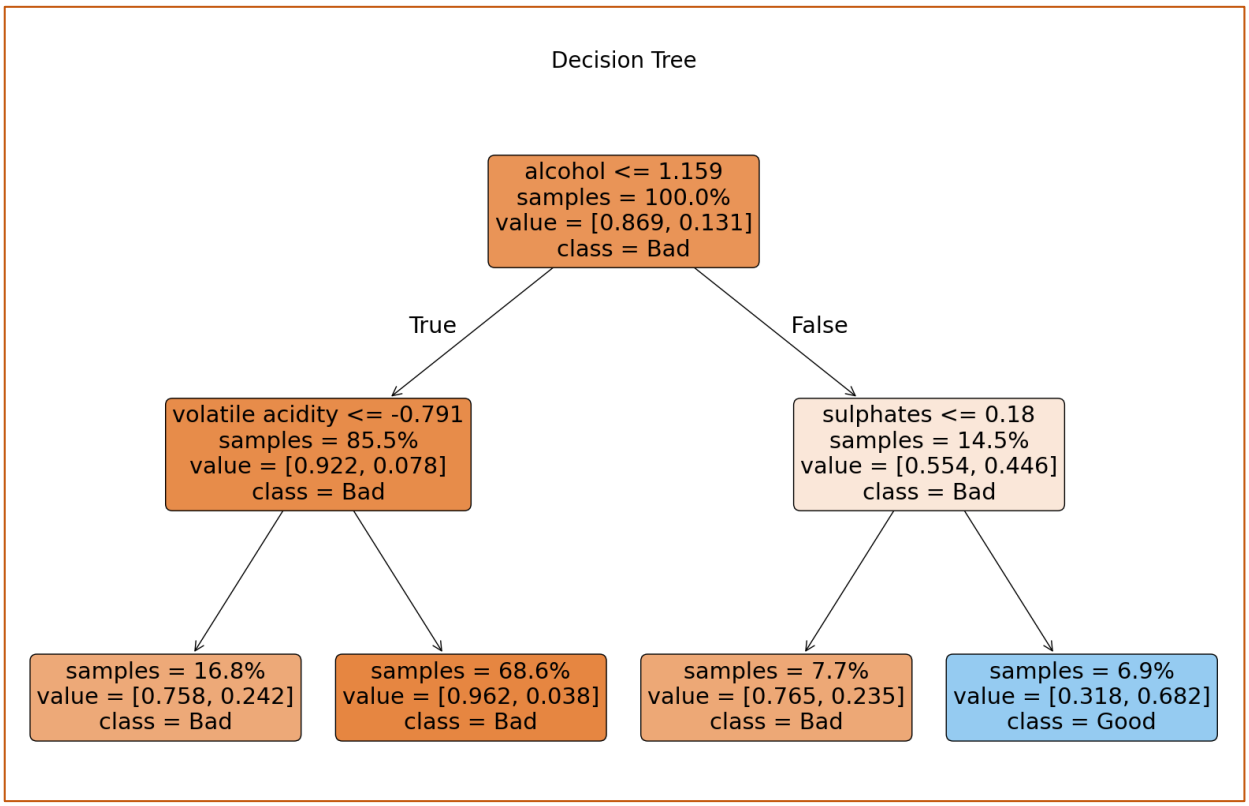

Fig 5: Decision Tree

The pruned decision tree (**max_depth = 2**) reveals:

- **Root Split:**
  The most important feature is **alcohol ≤ 1.159**, which forms the root of the tree. This aligns with earlier EDA, confirming alcohol as a key factor in wine quality.
- **Subsequent Splits:**
  - If alcohol is low (≤ 1.159), the tree further splits on **volatile acidity ≤ -0.791**.
  - If alcohol is high (> 1.159), it splits on **sulphates ≤ 0.18**.
- **Leaf Nodes:**
  - Most branches lead to a **"Bad"** prediction, reflecting class imbalance.
  - Only one leaf node predicts **"Good"** with ~68% confidence, showing a small but clear decision path for high alcohol and higher sulphates.
- **Interpretation:**
  The tree is simple yet effective, offering actionable insights.
  - For example:
    "If alcohol > 1.159 and sulphates > 0.18 → Predict Good quality wine."

Key Takeways

- **Depth matters:** Shallower trees generalize better for this dataset.
- **Tradeoffs:** Higher depth improves training accuracy but harms generalization.
- **Interpretability:** Depth=2 offers a balance—predictive power with explainable rules.
- **Process:** Always validate depth via cross-validation, not just train-test splits.

Limitations

- **Binary simplification:** Converting quality scores into just "Good" and "Bad" removes important distinctions (e.g., between wines rated 6 vs 7).
- **Class imbalance:** The dataset has fewer "Good" samples, making it harder for the model to correctly predict the minority class. This is evident from the leaf nodes in the decision tree.
- **Simple splits:** The optimal tree (depth=2) is easy to interpret but may miss deeper patterns. Complex interactions (like alcohol × sulphates) are not captured.
- **Overfitting risk in deeper trees:** As seen from performance at higher depths (e.g., 6 or 10), deeper trees tend to overfit, reducing test accuracy.

Future Work

- **Advanced models:** Explore Random Forest to improve performance by combining multiple decision trees.
- **Address imbalance:** Apply techniques like **SMOTE** or class weighting to handle underrepresented "Good" wines.
- **Explainability tools:** Use tools like **SHAP** to better understand feature impact at both global and individual sample levels.
- **Feature engineering:** Create interaction or polynomial features (e.g., alcohol × volatile acidity) to help capture non-linear patterns.

Accessibility Considerations

- Colorblind-safe palettes were used
- Markers and hatching aid visual distinction
- Report and notebook are screen-reader friendly

GitHub Repository

All the code, plots, and a copy of this PDF are hosted on the following GitHub repository:

https://github.com/bhuvansai93/ml-tutorial-23098103

References

1. *UCI Machine Learning Repository – Wine Quality Dataset*
2. *Scikit-learn Documentation: Official implementation details for python*
   - ***Decision Trees***
   - ***GridSearchCV***
3. *Hastie, T., Tibshirani, R., & Friedman, J. (2009). – The Elements of Statistical Learning*
   - *See chapter 9 for tree-based methods*
4. *Geeks for Geeks 2025 – Overfitting in Decision Trees Models*