Decision Tree Classifier Tutorial

Module: Machine Learning and Neural Networks
Assignment Type: Individual Report
Student Name: NIKHIL KUMAR PANAKANTI
Student ID: 23109231
Submission Date: 27 March 2025
Tutor: Peter Scicluna
GitHub link: https://github.com/nikhilkumar-panakanti/Machine-learning-individual-assignment

Decision Tree Classifier Tutorial

Table of Contents

Decision Tree Classifier Tutorial

Introduction:

Unlike black-box models, Decision Trees mirror how we naturally make decisions step by step, question by question.
Imagine evaluating a job offer you consider salary, commute, perks. A Decision Tree does the same, splitting data based on features to reach clear outcomes.
In this tutorial, we explore how Decision Trees work, including Gini and Entropy for splitting. Using the Titanic dataset, we'll build, visualize, and evaluate a model highlighting its strengths, best practices, and limitations.

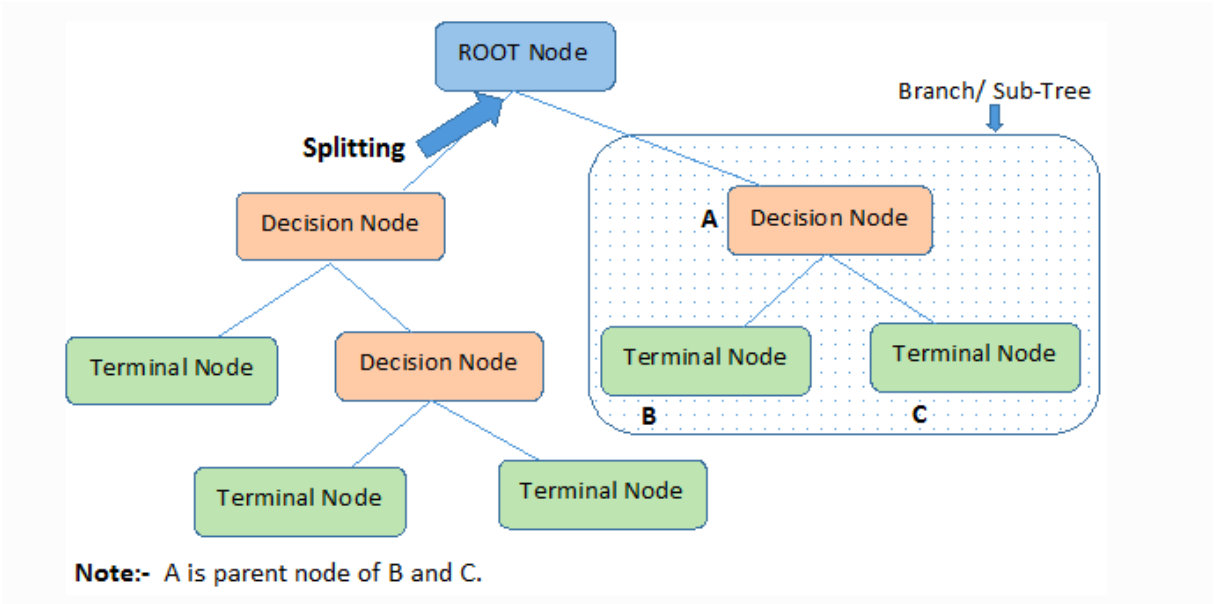


Figure 1: Structure of a Decision Tree showing the root, decision, and terminal nodes. A visual breakdown of parent-child relationships and subtrees.

Where:

- Root Node: The first question that starts the decision process.
- Splitting: Breaking a node into branches based on feature values.
- Branch/Subtree: A segment of the tree that grows from a decision point.
- Decision Node: A node that splits into more branches depending on outcomes.
- Leaf Node: The final point in a path where the prediction is made.

The diagram shows how a Decision Tree flows from the Root to Terminal Nodes, mapping decisions, outcomes, and the hierarchy between branches.

Decision Tree Classifier Tutorial

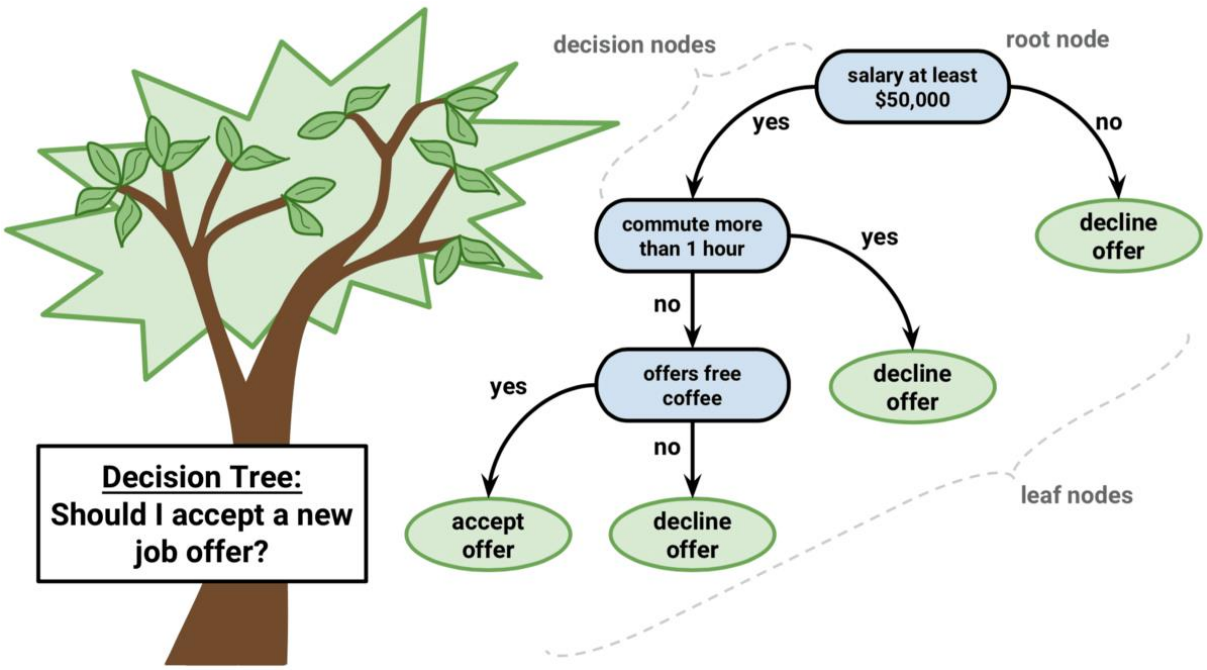


Figure 2: Decision Tree simulating a job offer choice using real-world conditions and outcomes.

Let's simplify this with a real-life example. Suppose you're weighing a new job offer. The tree starts with: "Is the salary at least $50,000?" If yes, it moves on to factors like commute time and perks. Step by step, it leads to a clear decision accept or decline. This mirrors how Decision Trees break down complex choices into simple, logical steps, just like we do in everyday life.

2.Theoretical Concepts

Before building a model, it's important to understand how Decision Trees work. They ask structured, data-driven questions to split data into smaller groups—each one becoming purer and more consistent in outcome.

2.1 Splitting Criteria (Gini Impurity and Entropy)

Each time a tree splits, it evaluates how well a feature divides the data. To measure this, we use impurity or uncertainty metrics. Two commonly used ones are Gini Impurity and Entropy.

Gini Impurity

Gini Impurity calculates the likelihood of misclassifying a random sample if it were labelled according to the distribution of labels in that node. It is defined as:

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

Where p_i is the probability of a data point belonging to class i.

- A Gini value of 0 means the node is perfectly pure (only one class exists there).
- It's fast to compute and often used as the default in libraries like Scikit-learn

Decision Tree Classifier Tutorial

Entropy

Entropy measures the amount of disorder in the data. It originates from information theory and is defined as:

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

- Entropy = 0 → perfect purity (one class)
- Entropy = high → classes are evenly split
  Both Gini and Entropy assess split quality by measuring group purity.

2.2 Information Gain

Once the impurity is calculated, the next step is to measure how much improvement a split provides. This is done using Information Gain.

$$Information\ Gain = Impurity_{parent} - \sum_{children} \left(\frac{N_{child}}{N_{parent}} * Impurity_{child} \right)$$

- The feature with the highest information gain is chosen for splitting, helping the tree improve classification at each step.
- Information Gain = impurity before split − impurity after split.

2.3 Overfitting and Pruning

One challenge with Decision Trees is their tendency to overfit, especially when allowed to grow too deep.

Overfitting

Decision Trees can be overfit by learning noise in the training data, hurting performance on new data.
Pruning helps prevent this:

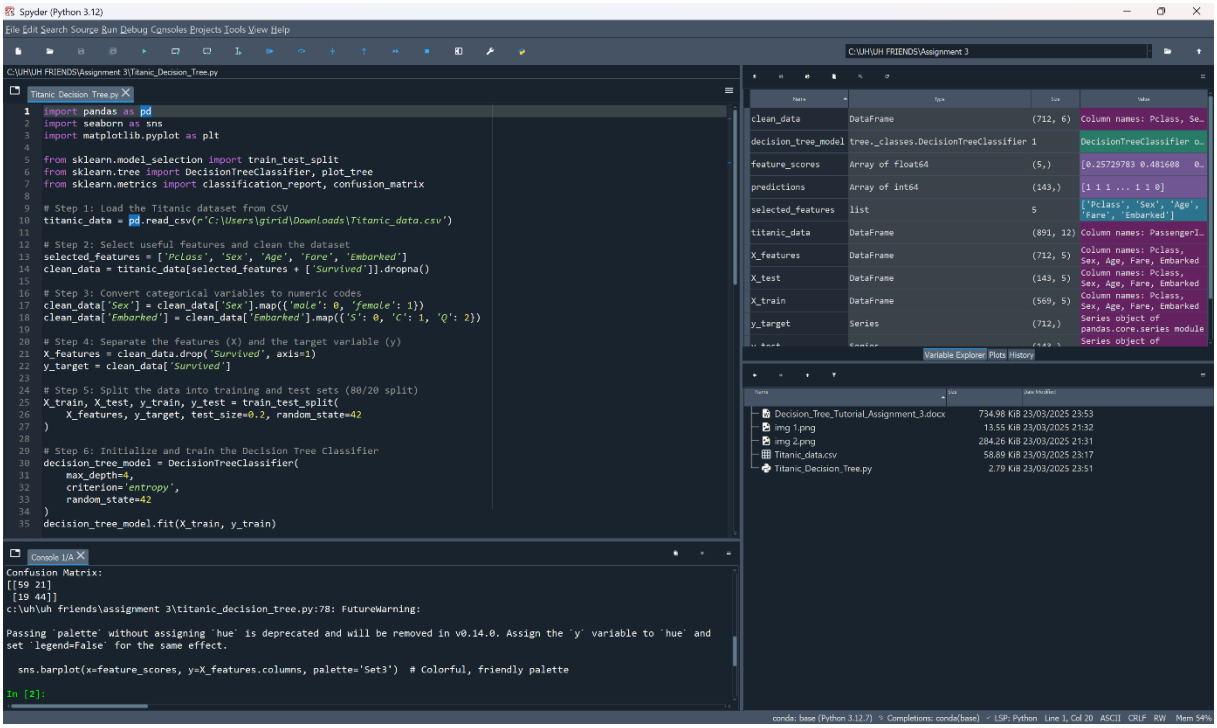
- **Pre-Pruning:** Set limits like max_depth or min_samples_leaf during training.
- **Post-Pruning:** Trim the fully grown tree after evaluating performance.

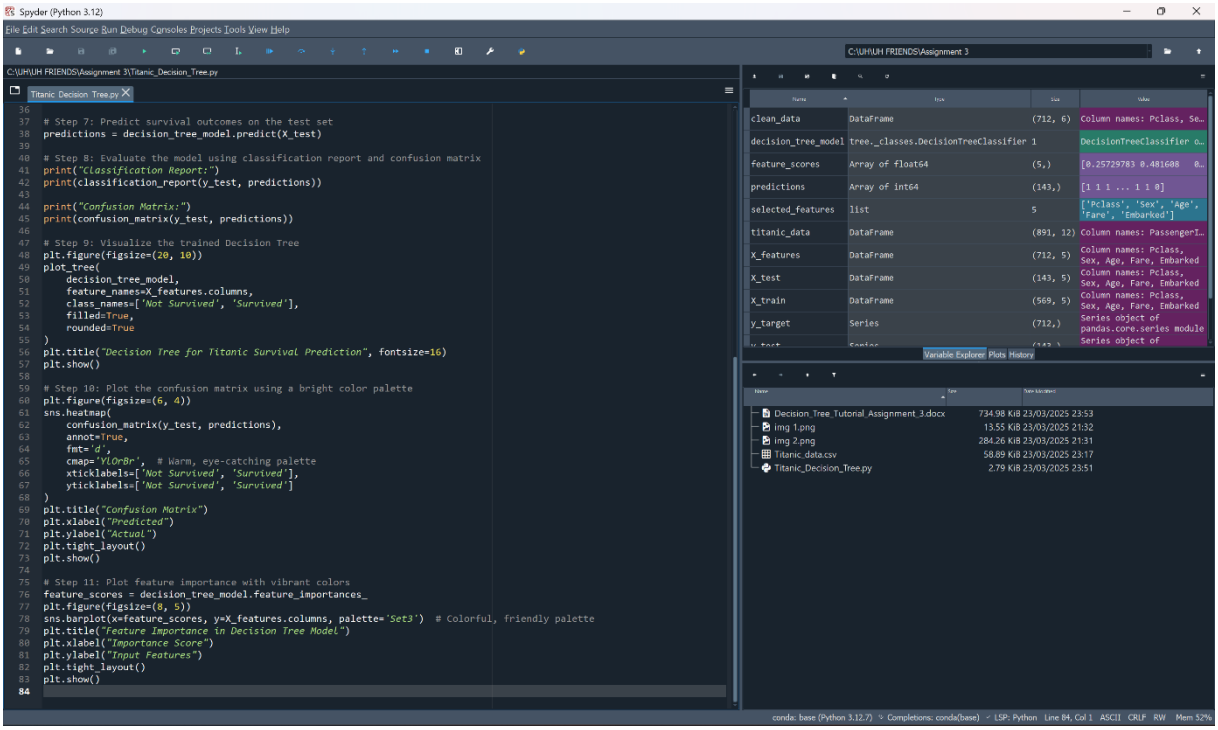
Pruning makes models simpler, faster, and more generalizable.

Decision Tree Classifier Tutorial

3. Model Training: Titanic Dataset

This tutorial's Python code trains a Decision Tree on the Titanic dataset selecting features, cleaning data, and encoding categories. Using the entropy criterion, it evaluates the model and visualizes results with a tree plot, confusion matrix, and feature importance chart.





Decision Tree Classifier Tutorial

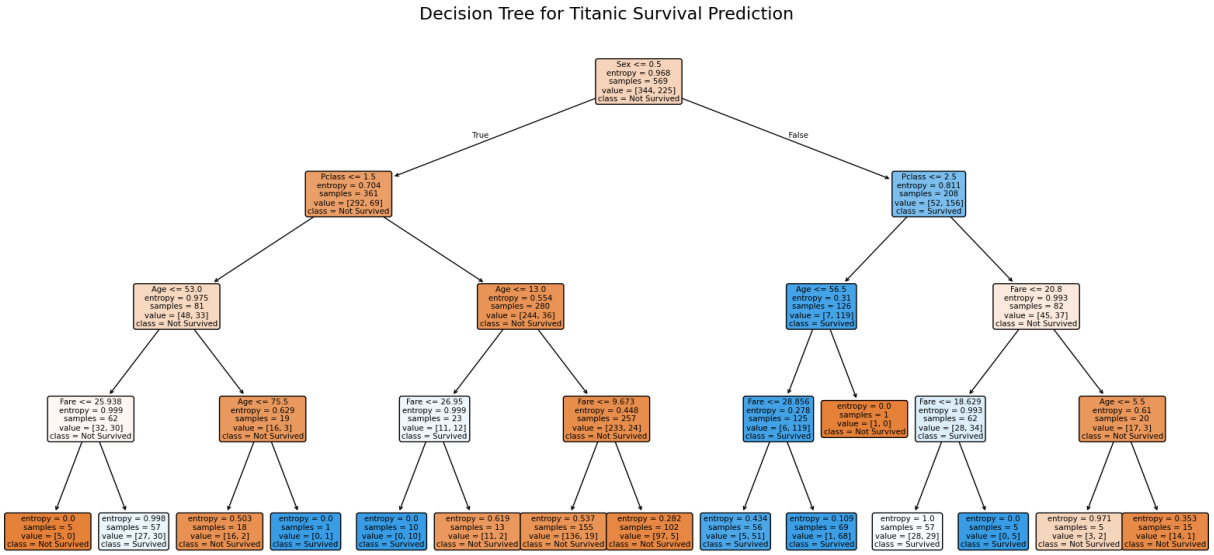


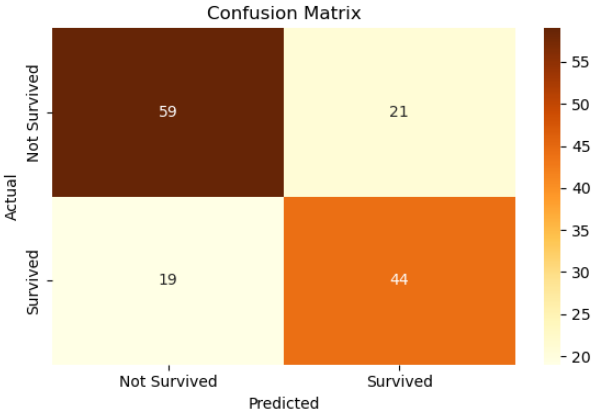
Figure 1: Decision Tree Structure for Titanic Survival Prediction

This plot shows how the Decision Tree makes predictions on the Titanic dataset. Starting from the root, it splits data step by step using key features like Sex, Pclass, Age, and Fare.

Each node tests a condition, branching toward clearer outcomes. Color-coded boxes show predicted classes and how confident the split is. As you move deeper, the decisions get more specific, with leaf nodes displaying the final results.

It's a clear visual of how complex decisions are broken into simple, logical steps.

Figure 2: Confusion Matrix – Model Performance on Titanic Dataset

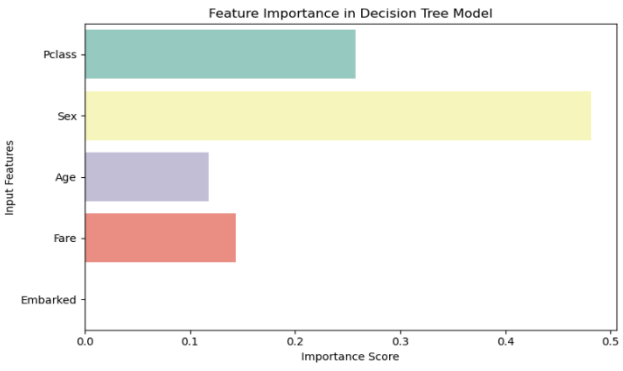


This heatmap illustrates the Decision Tree's performance on the test set. Each cell shows prediction counts:

- 59 true negatives
- 44 true positives
- 21 false positives
- 19 false negatives

Darker colours indicate higher values.
Ideally, correct predictions appear on the diagonal. This matrix helps assess accuracy and spot areas needing improvement.

Figure 3: Feature Importance – Decision Tree Model Insights



The bar chart shows which features most influenced the model. Sex had the highest impact, followed by Pclass and Fare, while Embarked had minimal effect. This helps make the model's decision process clear and explainable.

In this section, we built a Decision Tree from scratch using the Titanic dataset handling feature selection, data prep, and visualizations. These steps gave us a solid understanding of how the model works and set the stage for deeper evaluation and real-world use.

Decision Tree Classifier Tutorial

4. Model Evaluation Metrics

Evaluating a Decision Tree isn't just about getting predictions right—it's about knowing where it succeeds, where it slips, and how well it handles new data.

Key metrics like accuracy, precision, recall, and F1 score help assess its true performance.

- **Accuracy**

  What it tells you: How often the model was correct overall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  Where:

  - TP = True Positives (correctly predicted "survived")
  - TN = True Negatives (correctly predicted "not survived")
  - FP = False Positives (predicted "survived" but didn't)
  - FN = False Negatives (predicted "not survived" but did)

  NOTE: Accuracy is useful when classes are balanced, but it can be misleading if one class dominates the dataset.

- **Precision**

  What it tells you: Of all the passengers the model predicted as "survived", how many actually did survive?

$$Precision = \frac{TP}{TP + FP}$$

  Precision is critical when the cost of false positives is high (e.g., falsely predicting someone survived when they didn't).

- **Recall (Sensitivity)**

  What it tells you: Of all the passengers who actually survived, how many did the model correctly identify

$$Recall = \frac{TP}{TP + FN}$$

  This metric is vital in cases where missing a positive case is risky, such as in healthcare or safety-critical applications.

- **F1 Score**

  What it tells you: The balance between precision and recall. It's the harmonic mean of both metrics.

$$F1\ Score\ = 2 \times \frac{Precision\ \times Recall}{Precision\ +\ Recall}$$

F1 Score is especially useful when you need to balance the consequences of both false positives and false negatives.

- **Confusion Matrix: Making Metrics Visual**

  A confusion matrix is a visual breakdown of how the model's predictions compare to actual outcomes. It looks like this:

	Predicted: NO	**Predicted: YES**
Actual: NO	True Negative	False Positive
Actual: YES	False Negative	True Positive

Confusion Matrix Recap

- **TN:** Correctly predicted not survived
- **TP:** Correctly predicted survived
- **FP:** Incorrectly predicted survived
- **FN:** Incorrectly predicted not survived
  (See Figure 2 for visual reference)

Why It Matters

Accuracy alone isn't enough.

- **Low precision:** Many false "survived" predictions
- **Low recall:** Misses actual survivors
- **F1 Score:** Balances both crucial for imbalanced datasets and real-world reliability.

5. Real-World Use Cases of Decision Trees

Decision Trees are widely used in industries where clarity, speed, and explainability matter. Their simple logic makes them ideal for making accurate, transparent decisions.

Key Applications:

- **Healthcare:** Help diagnose illnesses based on symptoms and test results.
- **Banking:** Assess loan approvals and detect fraud using credit and income data.
- **Retail:** Recommend products and target promotions based on customer behaviour.
- **Telecom:** Predict customer churn through usage and complaint patterns.
- **Education:** Identify students needing support using performance data.
- **Government:** Evaluate risk and eligibility for services with clear, fair rules.

What sets Decision Trees apart is their **transparency** every decision can be explained, which is vital when outcomes affect lives or finances.

Decision Tree Classifier Tutorial

6. Best Practices and Interpretability

Decision Trees are powerful yet simple tools but using them wisely is key. Follow these tips to build models that are both accurate and interpretable:

- **Limit Tree Depth:** Deep trees overfit. Set max_depth or a minimum sample size to keep the model generalizable.
- **Clean Your Data:** Handle missing values and convert categorical data properly for better splits.
- **Choose Relevant Features:** Focus on impactful variables—more isn't always better.
- **Visualize the Tree:** Use tools like plot_tree() to make decisions traceable and easy to explain.
- **Ensure Explainability:** Clearly communicate why a prediction was made—essential in sensitive domains.
- **Look Beyond Accuracy:** Evaluate precision, recall, and F1 score, especially for imbalanced data.
- **Prune When Needed:** Use pre- or post-pruning to reduce complexity and prevent overfitting.
- **Why It Matters:** Interpretability builds trust. Decision Trees are favoured because their logic is transparent and easy to follow.

In short, smart use of Decision Trees leads to models that perform well and make sense.

7. Challenges and Limitations

While Decision Trees are simple, powerful, and easy to interpret, they're not without challenges. Knowing these limitations helps avoid pitfalls and ensures you choose the right tool for the task.
The table below outlines key issues, their impact, and how to address them.

Challenge	Why It Matters	How to Handle It
Overfitting	The model learns the noise in the training data, reducing performance on new data.	Limit tree depth, use pruning, or apply regularization.
Sensitive to Small Data Changes	Even slight changes in data can restructure the entire tree.	Use ensemble methods like Random Forest for better stability.
Bias Toward Multi-Level Features	Features with many unique values dominate splits, even if not helpful.	Preprocess or encode categorical data carefully; remove irrelevant variables.
Poor at Capturing Complex Patterns	Can't model highly non-linear or intricate relationships.	Use more advanced models like Gradient Boosting or Neural Networks if needed.
Sharp Jumps in Regression Tasks	For regression, predictions change abruptly instead of smoothly.	Use smoothing techniques or ensemble regressors (like Random Forest Regressor).

Decision Tree Classifier Tutorial

8.Future Scope and Enhancements

This tutorial sets a strong foundation, but there's plenty of room to grow. Here are keyways to level up the project:

- Hyperparameter Tuning: Use tools like GridSearchCV to optimize settings like max_depth and criterion.
- Compare with Ensembles: Test models like Random Forests or XGBoost for better accuracy.
- Cross-Validation: Replace single train-test split with k-fold cross-validation for more reliable results.
- New Datasets: Apply the same pipeline to different datasets for broader insights.
- Interactive App: Turn the model into a web app using Streamlit or Flask for real-time predictions.
- Document & Share: Publish your project on GitHub with a clear README, visuals, and usage instructions.

With these steps, this basic Decision Tree model can grow into a polished, real-world data science solution.

9.Conclusion

This tutorial explored Decision Tree Classifiers one of the most intuitive and beginner-friendly machine learning techniques. We showed how they simplify complex problems into clear, logical steps.

Using the Titanic dataset, we built a complete model from scratch, covering data prep, training, evaluation, and visualization. Visual tools like tree diagrams and confusion matrices enhanced clarity and accessibility.

What makes Decision Trees stand out is their transparency. Unlike black-box models, they clearly show how decisions are made essential in fields where trust matters.

We also looked beyond accuracy, focusing on precision, recall, and F1-score, while addressing real-world use cases, common challenges, best practices, and accessibility.

In the end, this was more than just writing code it was about building a strong, ethical, and explainable foundation in machine learning. Mastering Decision Trees is a powerful first step toward real-world impact.

Keep exploring, keep building and let your curiosity grow as deep as the trees you now know how to create.

Decision Tree Classifier Tutorial

10.References

- Kaggle. *Titanic: Machine Learning from Disaster*. Retrieved from https://www.kaggle.com/competitions/titanic
- Scikit-learn. *Machine Learning in Python.* Retrieved from https://scikit-learn.org
- Pandas. *Python Data Analysis Library*. Retrieved from https://pandas.pydata.org
- Matplotlib. *2D Plotting Library.* Retrieved from https://matplotlib.org
- Seaborn. *Statistical Data Visualization.* Retrieved from https://seaborn.pydata.org
- R2D3. *A Visual Introduction to Machine Learning.* Retrieved from https://www.r2d3.us
- Towards Data Science. *Machine Learning Tutorials & Insights.* Retrieved from https://towardsdatascience.com
- Nuggets. *Decision Tree Algorithm, Explained.* (Source for Figure 1) https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
- DataCamp. *Decision Trees in R: A Step-by-Step Tutorial.* (Source for Figure 2) https://www.datacamp.com/tutorial/decision-trees-R