**Assignment No. 3**

**Problem Statement:** Implement a Decision Tree model

**Objective:**

1. Understand the mechanics of Decision Trees for both classification and regression.
2. Learn how to control overfitting through pre‑pruning (limiting tree growth) and post‑pruning (pruning the fully grown tree).
3. Implement and evaluate Decision Tree models with performance metrics.
4. Use cross‑validation to assess model generalization.

**Prerequisite :**

1. A Python environment with essential libraries like pandas, numpy, matplotlib, seaborn, and scikit-learn.
2. Basic knowledge of Python, statistics, and machine learning principles.

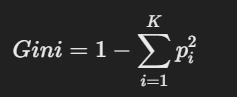
**Theory :**

**1. Decision Trees**

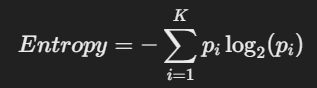
A **Decision Tree** is a supervised learning algorithm used for both classification and regression. It models decisions by recursively partitioning the input space into distinct regions, each represented by a node in a tree. At each decision node, the algorithm selects a feature and a threshold that best splits the data into two (or more) subsets that are more homogeneous regarding the target variable.

**For Classification:**

* **Impurity Measures:**
  + **Gini Impurity:**



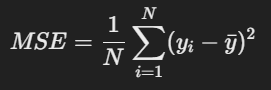
* + where pi is the proportion of samples of class i in the node. Gini impurity quantifies the likelihood of an incorrect classification of a randomly chosen element if it were randomly labeled according to the distribution in the node.
  + **Entropy (Information Gain):**



* + Lower entropy indicates a purer node. When a split significantly reduces entropy, it is considered a good split.

**For Regression:**

* **Variance Reduction (Mean Squared Error):**  
  The goal is to split the data such that the variance (or the Mean Squared Error, MSE) within each subset is minimized.



* Here, yˉ​ is the mean target value of the samples in the node. The split is chosen to maximize the reduction in MSE.

Decision Trees are highly interpretable since each decision can be traced back through the tree, providing a clear understanding of how input features influence the final prediction.

**2. Pre-Pruning (Early Stopping)**

**Pre-pruning** (or early stopping) is the practice of halting the growth of the tree before it fully learns the training data. This is done by specifying constraints during the tree-building process to prevent overfitting. Overfitting happens when the tree captures noise along with the underlying pattern, resulting in poor performance on unseen data.

**Key Parameters:**

1. **max\_depth:**  
   Limits the maximum depth of the tree. A shallow tree may underfit, while a very deep tree might overfit.
2. **min\_samples\_split:**  
   The minimum number of samples required to split an internal node. If a node has fewer samples than this threshold, no further split is attempted.
3. **min\_samples\_leaf:**  
   The minimum number of samples that must be present in a leaf node. This ensures that leaves have enough data to make reliable predictions.
4. **max\_leaf\_nodes:**  
   Limits the total number of leaf nodes in the tree, effectively reducing complexity.

By controlling these parameters, pre-pruning reduces the risk of developing overly complex trees and encourages better generalization on unseen data.

**3. Post-Pruning (Cost Complexity Pruning)**

**Post-pruning** is a method applied after a decision tree has been grown to its full depth. The idea is to remove parts of the tree that do not provide substantial predictive power, thereby simplifying the model.

**Cost Complexity Pruning:**

* **Concept:**  
  Once the tree is fully grown, some branches may be capturing noise rather than signal. Cost complexity pruning introduces a penalty for complexity, quantified by the parameter α(often called ccp\_alpha in scikit-learn). Higher values of α lead to simpler trees.
* **Process:**
  1. **Grow the full tree:**  
     The tree is grown without constraints.
  2. **Compute the cost complexity measure:**  
     For each subtree, calculate a cost that includes both the error and a penalty for the number of terminal nodes.
  3. **Prune subtrees:**  
     Remove branches that result in a minimal increase in error relative to the reduction in complexity.
  4. **Select the optimal α\alphaα:**  
     Use cross-validation to determine the value of α\alphaα that results in the best generalization performance.

Post-pruning helps in striking a balance between bias and variance by simplifying the model while retaining its predictive capability.

**4. Decision Tree Regression**

When applying decision trees to regression tasks, the goal is to predict a continuous output value. Instead of classifying samples into discrete categories, the decision tree regression model partitions the feature space into regions where the response variable is approximately constant.

**Key Aspects:**

1. **Splitting Criterion:**  
   At each node, the model chooses splits that minimize the MSE. The optimal split is the one that results in the greatest reduction in variance.
2. **Prediction:**  
   For a given leaf, the prediction is typically the mean of the target values of the training samples that fall into that leaf.
3. **Handling Overfitting:**  
   Just like in classification, overfitting is a concern. Pre-pruning parameters (e.g., max\_depth, min\_samples\_split) and post-pruning techniques can also be applied to regression trees to improve generalization.

Decision Tree Regression models are particularly useful for problems where the relationship between input features and the target is non-linear and complex, yet interpretable.

**5. Cross-Validation**

**Cross-validation** is a robust technique to evaluate the generalizability of a model. Instead of relying on a single train/test split, cross-validation divides the dataset into multiple folds and iteratively uses different folds for training and validation.

**Key Methods:**

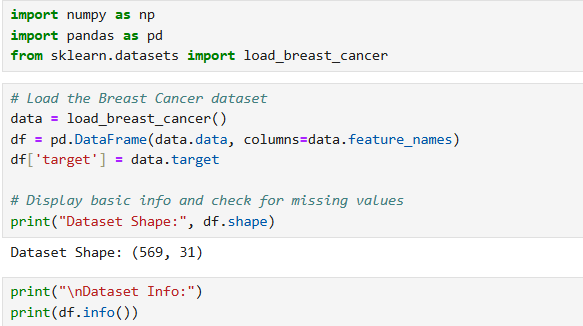
1. **k-fold Cross-Validation:**  
   The dataset is partitioned into k folds. For each iteration, one fold is used as the test set and the remaining k−1 folds are used for training. The performance metric is averaged over all k iterations.
2. **GridSearchCV:**  
   This technique combines exhaustive hyperparameter search with k-fold cross-validation. By specifying a grid of hyperparameter values ( different values for max\_depth, min\_samples\_split, or ccp\_alpha ), GridSearchCV trains and evaluates the model for every combination and selects the one with the best average performance.

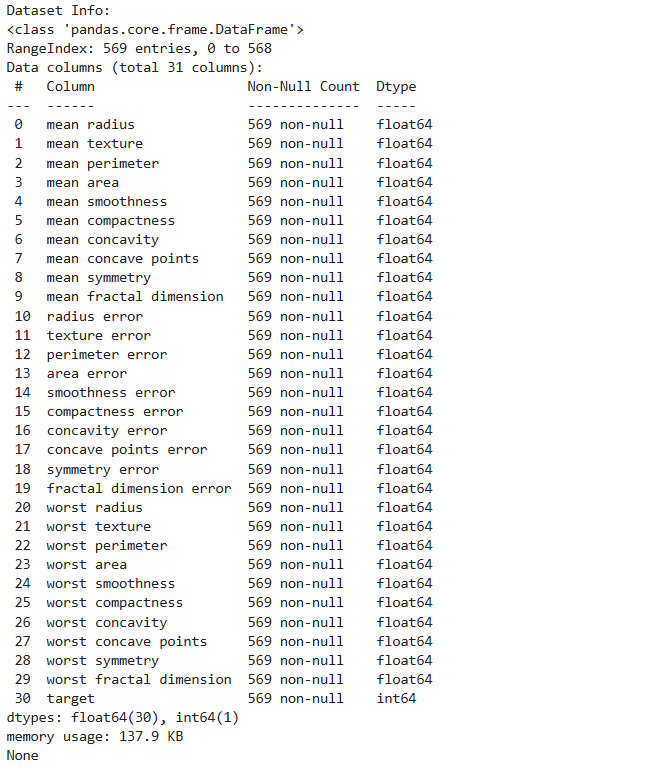
**Benefits:**

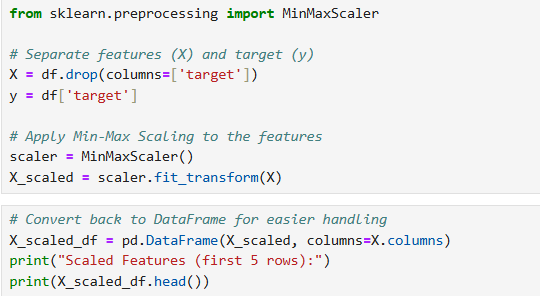
1. **Robust Performance Estimation:**  
   Reduces the risk that the model’s performance is an artifact of a particular split.
2. **Hyperparameter Tuning:**  
   Helps in systematically finding the optimal parameters for the model, balancing bias and variance.
3. **Generalization:**  
   Provides a better estimate of how the model will perform on unseen data.

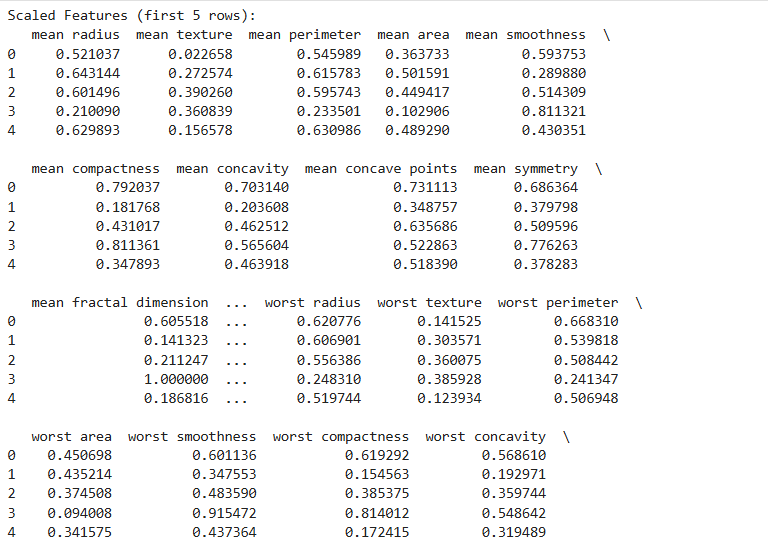
Cross-validation is crucial when working with limited data or when tuning complex models like decision trees, as it ensures that the model’s performance is not overestimated.

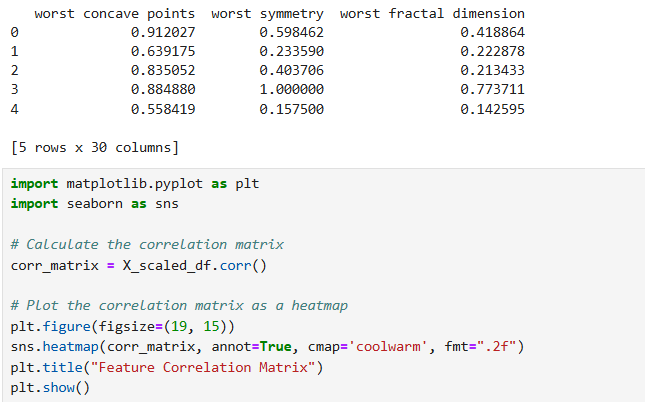
1. **Code & Output**

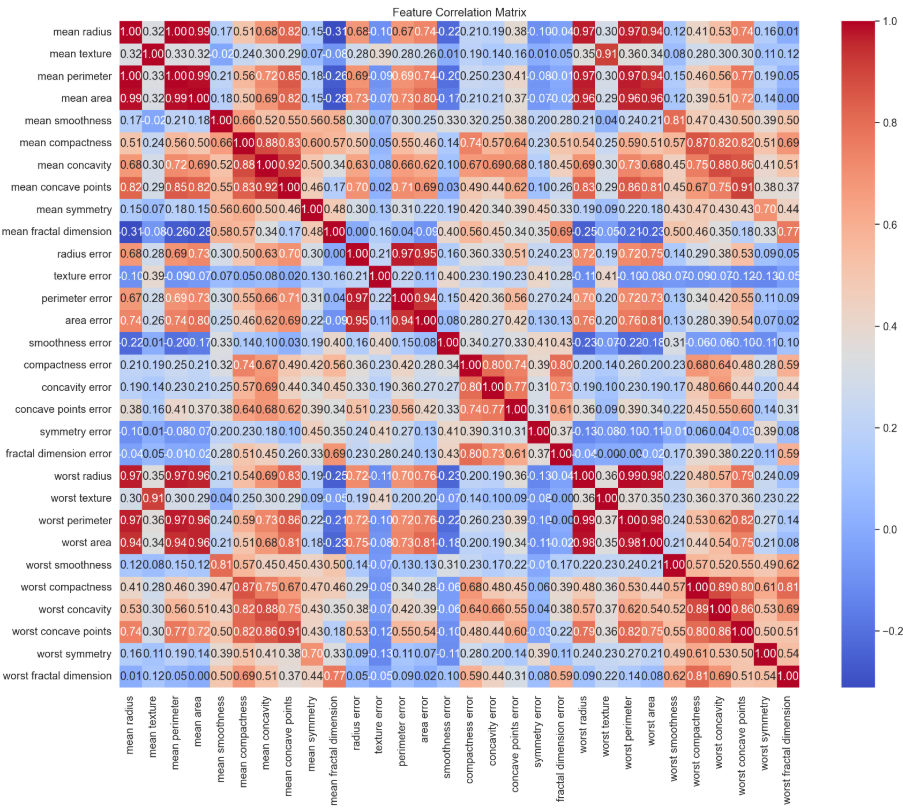
****

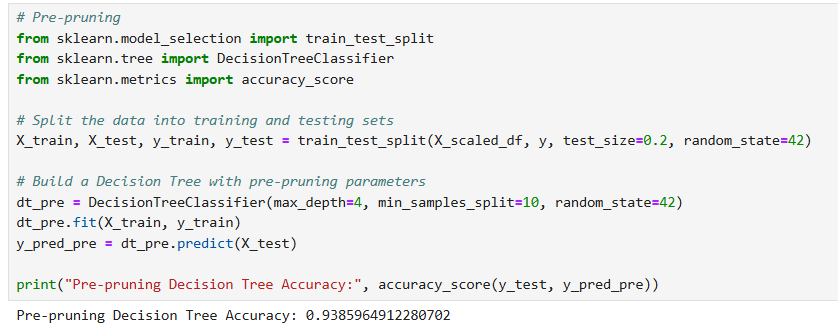
****



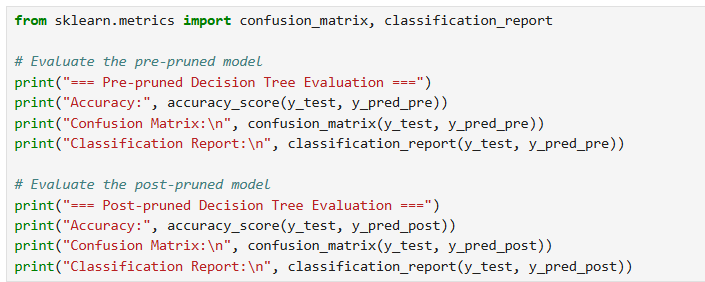


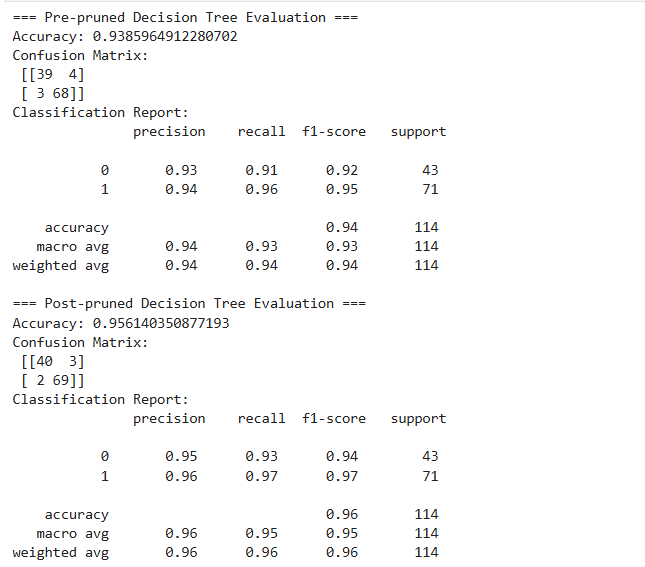


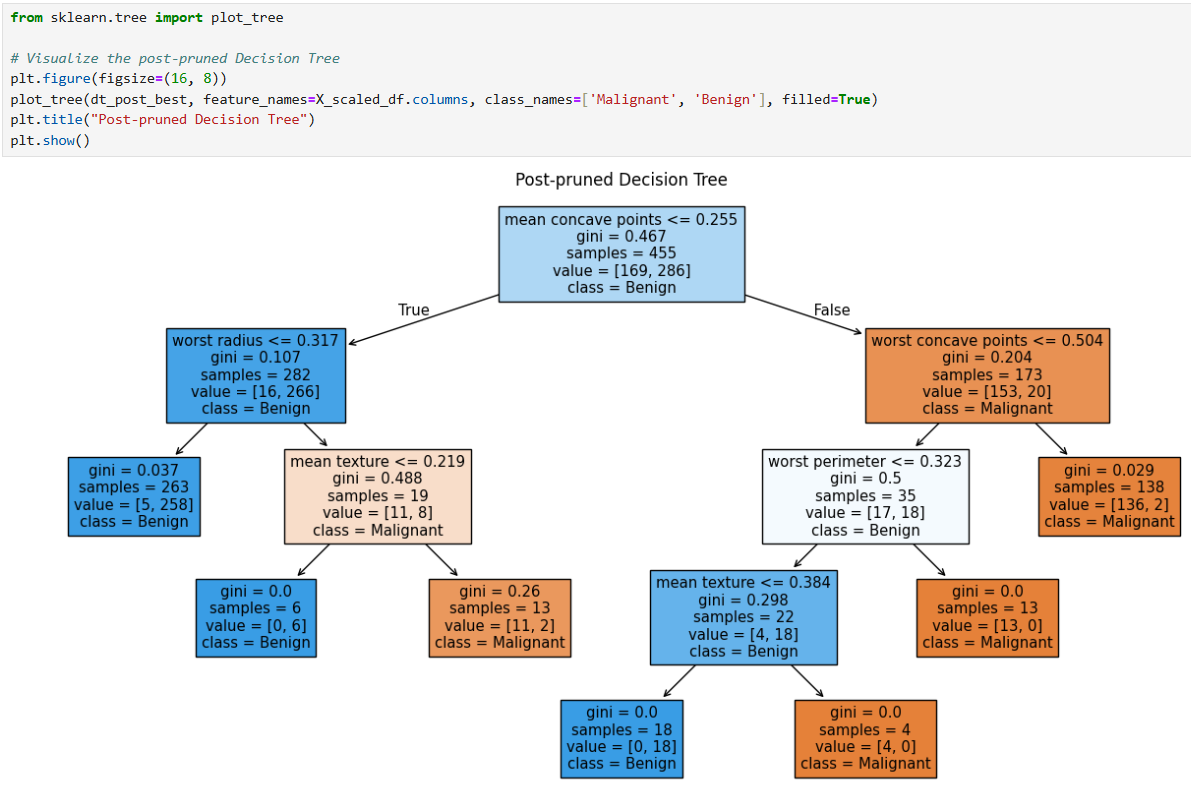


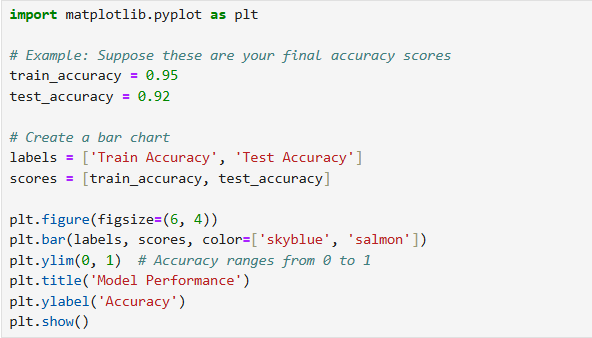










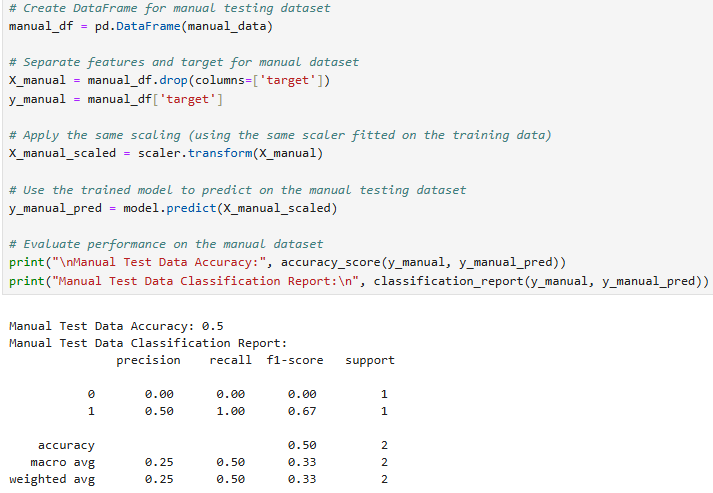












Github: <https://github.com/dnyaneshwardhere/ML>

## **Conclusion**

In this assignment, we explored Decision Tree models for both classification and regression. We implemented pre‑pruning techniques to control tree growth and reduce overfitting, and we applied post‑pruning using cost complexity pruning to further refine the model. Cross‑validation was used to assess model performance and ensure generalizability. This comprehensive approach provides valuable insights into model tuning and evaluation, laying a strong foundation for further work with Decision Trees.