Pruning decision trees

Decision tree pruning is a critical technique in machine learning used to optimize decision tree models by reducing overfitting and improving generalization to new data. **In this guide, we'll explore the importance of decision tree pruning, its types, implementation, and its significance in machine learning model optimization.**

What is Decision Tree Pruning?

Decision tree pruning is a technique used to prevent decision trees from overfitting the training data. Pruning aims to simplify the decision tree by removing parts of it that do not provide significant predictive power, thus improving its ability to generalize to new data.

Decision Tree Pruning removes unwanted nodes from the overfitted decision tree to make it smaller in size which results in more fast, more accurate and more effective predictions.

Types Of Decision Tree Pruning

There are two main types of decision tree pruning: **Pre-Pruning** and **Post-Pruning**.

Pre-Pruning (Early Stopping)

Sometimes, the growth of the decision tree can be stopped before it gets too complex, this is called pre-pruning. It is important to prevent the overfitting of the training data, which results in a poor performance when exposed to new data.

Some common pre-pruning techniques include:

- **Maximum Depth**: It limits the maximum level of depth in a decision tree.
- **Minimum Samples per Leaf**: Set a minimum threshold for the number of samples in each leaf node.
- **Minimum Samples per Split**: Specify the minimal number of samples needed to break up a node.
- **Maximum Features:** Restrict the quantity of features considered for splitting.

By pruning early, we come to be with a simpler tree that is less likely to overfit the training facts.

Post-Pruning (Reducing Nodes)

After the tree is fully grown, post-pruning involves removing branches or nodes to improve the-model's ability to generalize. Some common post-pruning techniques include:

- **Cost-Complexity Pruning (CCP)**: This method assigns a price to each subtree primarily based on its accuracy and complexity, then selects the subtree with the lowest fee.
- **Reduced Error Pruning**: Removes branches that do not significantly affect the overall accuracy.
- **Minimum Impurity Decrease**: Prunes nodes if the decrease in impurity (Gini impurity or entropy) is beneath a certain threshold.
- **Minimum Leaf Size**: Removes leaf nodes with fewer samples than a specified threshold.

Post-pruning simplifies the tree while preserving its Accuracy. Decision tree pruning helps to improve the performance and interpretability of decision trees by reducing their complexity and avoiding overfitting. Proper pruning can lead to simpler and more robust models that generalize better to unseen data.

Decision Tree Implementation in Python

Here we are going to create a decision tree using preloaded dataset **breast_cancer** in sklearn library.

The Decision Tree model is using pre-pruning technique, specifically, the default approach of scikit-learn's `DecisionTreeClassifier`, which employs the Gini impurity criterion for making splits. This is evident from the parameter `criterion="gini"` passed to the `DecisionTreeClassifier()` constructor. Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the set.

Python3` ``` from sklearn.datasets import load_breast_cancer from sklearn.model_selection import train_test_split from sklearn.tree import DecisionTreeClassifier from sklearn.tree import plot_tree import matplotlib.pyplot as plt

Load breast cancer dataset

X, y = load_breast_cancer(return_X_y=True)

Separating Training and Testing data

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.2, random_state=42)

Train decision tree model

model = DecisionTreeClassifier(criterion="gini") model.fit(X_train, y_train)

Plot original tree

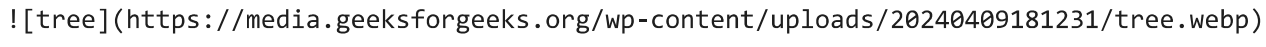
plt.figure(figsize=(15, 10)) plot_tree(model, filled=True) plt.title("Original Decision Tree") plt.show()

Model Accuracy before pruning

accuracy_before_pruning = model.score(X_test, y_test) print("Accuracy before pruning:", accuracy_before_pruning)

`

****Output:****



Accuracy before pruning: 0.8793859649122807

Decision Tree Pre-Pruning Implementation

In the implementation, we pruning technique is hyperparameter tuning through cross-validation

Python3` ```

```
from sklearn.tree import DecisionTreeClassifier
parameter = {
    'criterion' :['entropy','gini','log_loss'],
    'splitter':['best','random'],
    'max_depth':[1,2,3,4,5],
    'max_features':['auto','sqrt','log2']
}
model = DecisionTreeClassifier()
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(model,param_grid = parameter,cv = 5)
cv.fit(X_train,Y_train)
```

`

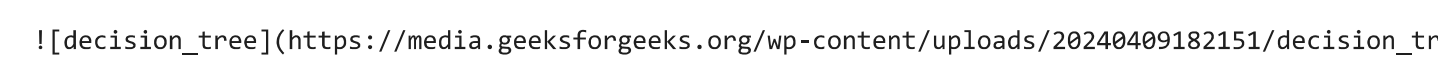
Visualizing

Python3` ``` from sklearn.tree import export_graphviz import graphviz best_estimator = cv.best_estimator_ feature_names = features

dot_data = export_graphviz(best_estimator, out_file=None, filled=True, rounded=True, feature_names=feature_names, class_names=['0', '1', '2']) graph = graphviz.Source(dot_data) graph.render("decision_tree", format='png', cleanup=True) graph

`

****Output:****



Best Parameters

Python3` ```
cv.score(X_test,Y_test)
cv.best_params_

`

Output:

```
0.9736842105263158
{'criterion': 'gini',
 'max_depth': 4,
 'max_features': 'sqrt',
 'splitter': 'best'}
```

Decision Tree Post-Pruning Implementation

Python3` ```

Cost-complexity pruning (Post-pruning)

path = model.cost_complexity_pruning_path(X_train, y_train) ccp_alphas, impurities = path.ccp_alphas, path.impurities

Train a series of decision trees with different alpha values

pruned_models = [] for ccp_alpha in ccp_alphas: pruned_model = DecisionTreeClassifier(criterion="gini", ccp_alpha=ccp_alpha) pruned_model.fit(X_train, y_train) pruned_models.append(pruned_model)

Find the model with the best accuracy on test data

best_accuracy = 0 best_pruned_model = None for pruned_model in pruned_models: accuracy = pruned_model.score(X_test, y_test) if accuracy > best_accuracy: best_accuracy = accuracy best_pruned_model = pruned_model

Model Accuracy after pruning

accuracy_after_pruning = best_pruned_model.score(X_test, y_test) print("Accuracy after pruning:", accuracy_after_pruning)

`

****Output:****

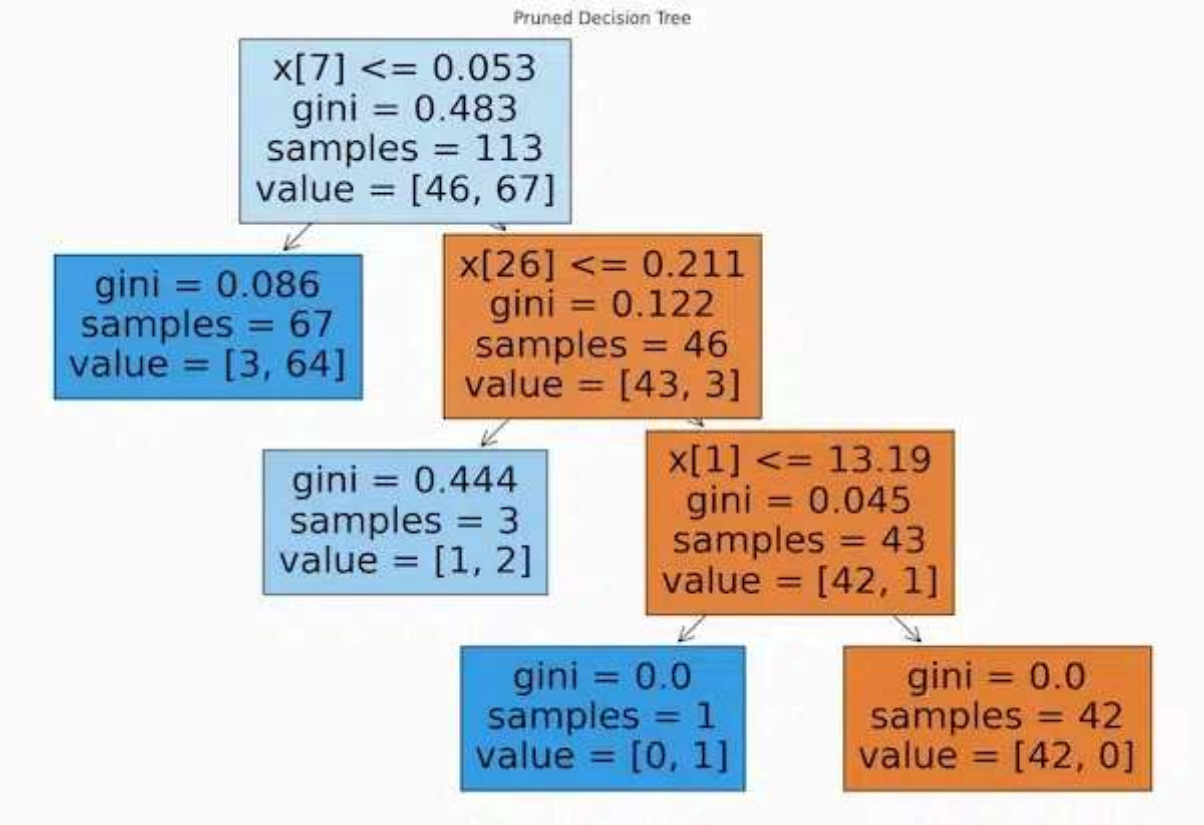
Accuracy after pruning: 0.918859649122807

Python3` ```
# Plot pruned tree
plt.figure(figsize=(15, 10))
plot_tree(best_pruned_model, filled=True)
plt.title("Pruned Decision Tree")

```
plt.show()
```

`

Output:



Why Pruning decision trees is Important?

Decision Tree Pruning has an important role in optimizing the decision tree model. It involves the removal of certain parts of the tree which can potentially reduce its performance. Here is why decision tree pruning is important:

1. **Prevents Overfitting:** Decision trees are prone to overfitting, where the model memorizes the training data rather than learning generalizable patterns. Pruning helps prevent overfitting by simplifying the tree structure, removing branches that capture noise or outliers in the training data.
2. **Improves Generalization:** By reducing the complexity of the decision tree, pruning enhances the model's ability to generalize to unseen data. A pruned decision tree is more likely to capture underlying patterns in the data rather than memorizing specific instances, leading to better performance on new data.

3. **Reduces Model Complexity**: Pruning results in a simpler decision tree with fewer branches and nodes. This simplicity not only makes the model easier to interpret but also reduces computational requirements during both training and inference. A simpler model is also less prone to overfitting and more robust to changes in the data.
4. **Enhances Interpretability**: Pruning produces decision trees with fewer branches and nodes, which are easier to interpret and understand. This is particularly important in applications where human insight into the decision-making process is valuable, such as in medical diagnosis or financial decision-making.
5. **Speeds Up Training and Inference**: Pruned decision trees require less computational resources during both training and inference phases. With fewer branches and nodes, the decision-making process becomes more efficient, resulting in faster predictions without sacrificing accuracy.
6. **Facilitates Model Maintenance**: Pruning helps maintain decision tree models over time by keeping them lean and relevant. As new data becomes available or the problem domain evolves, pruned decision trees are easier to update and adapt compared to overly complex, unpruned trees.

Conclusion

Decision tree pruning plays a crucial role in optimizing decision tree models by preventing overfitting, improving generalization, and enhancing model interpretability. Post-Pruning is used generally for small datasets whereas Pre-Pruning is used for larger ones. Pre-Pruning is considered more efficient and effective as it considered multiple parameters and choose best ones from them.