Q1. What is the estimated depth of a Decision Tree trained (unrestricted) on a one million instance training set?

The estimated depth of a Decision Tree trained (unrestricted) on a one million instance training set can be quite deep. In theory, a Decision Tree can have a depth of up to the number of instances in the training set, resulting in a tree with a depth of one million. However, in practice, the depth of the tree may be controlled by hyperparameters such as max\_depth, min\_samples\_split, and others.

Q2. Is the Gini impurity of a node usually lower or higher than that of its parent? Is it always lower/greater, or is it usually lower/greater?

The Gini impurity of a node is usually lower than that of its parent. This is because the decision tree algorithm aims to split nodes in a way that reduces impurity and creates more homogeneous subsets. As the tree grows, each split is designed to minimize the Gini impurity, leading to lower values as we move down the tree.

Q3. Explain if its a good idea to reduce max depth if a Decision Tree is overfitting the training set?

Yes, it is a good idea to reduce the max depth if a Decision Tree is overfitting the training set. Overfitting occurs when the tree becomes too deep, capturing noise and specific details of the training data. By reducing the max depth, the tree becomes simpler and less likely to overfit, resulting in improved generalization to new data.

Q4. Explain if its a good idea to try scaling the input features if a Decision Tree underfits the training set?

No, it is not a good idea to try scaling the input features if a Decision Tree underfits the training set. Decision Trees are insensitive to the scale of the input features, as they make decisions based on thresholds and feature comparisons. Scaling won't have a significant impact on the performance of Decision Trees.

Q5. How much time will it take to train another Decision Tree on a training set of 10 million instances if it takes an hour to train a Decision Tree on a training set with 1 million instances?

Training another Decision Tree on a training set of 10 million instances would likely take more than 10 hours if it takes an hour to train a Decision Tree on a training set with 1 million instances. The time required for training scales roughly with the number of instances, especially if the dataset size is significantly increased.

Q6. Will setting presort=True speed up training if your training set has 100,000 instances?

No, setting presort=True will not speed up training if your training set has 100,000 instances. The presort option in Scikit-Learn's DecisionTreeClassifier is useful for smaller datasets, where the presorting process might help speed up the initial tree building process. However, for larger datasets like the one you mentioned, the overhead of presorting can outweigh the potential benefits.

Q7. Follow these steps to train and fine-tune a Decision Tree for the moons dataset:

1. To build a moons dataset, use make moons(n samples=10000, noise=0.4).
2. Divide the dataset into a training and a test collection with train test split().
3. To find good hyperparameters values for a DecisionTreeClassifier, use grid search with cross-validation (with the GridSearchCV class). Try different values for max leaf nodes.
4. Use these hyperparameters to train the model on the entire training set, and then assess its output on the test set. You can achieve an accuracy of 85 to 87 percent.

Answer File name : fine-tune a Decision Tree for the moons dataset.ipynb

Q8. Follow these steps to grow a forest:

a. Using the same method as before, create 1,000 subsets of the training set, each containing 100 instances chosen at random. You can do this with Scikit-ShuffleSplit Learn's class.

b. Using the best hyperparameter values found in the previous exercise, train one Decision Tree on each subset. On the test collection, evaluate these 1,000 Decision Trees. These Decision Trees would likely perform worse than the first Decision Tree, achieving only around 80% accuracy, since they were trained on smaller sets.

c. Now the magic begins. Create 1,000 Decision Tree predictions for each test set case, and keep only the most common prediction (you can do this with SciPy's mode() function). Over the test collection, this method gives you majority-vote predictions.

d. On the test range, evaluate these predictions: you should achieve a slightly higher accuracy than the first model (approx 0.5 to 1.5 percent higher). You've successfully learned a Random Forest classifier!

Answer File name: Random Forest classifier.ipynb