| **ML\_ASSISMENT\_21** |
| --- |

| **1. What is the estimated depth of a Decision Tree trained (unrestricted) on a one million instance training set?**  **Sol:**  The estimated depth of a Decision Tree trained on a one million instance training set is not easily determined without considering other factors. The depth of a Decision Tree depends on various parameters, including the complexity of the data, the number of features, the tree's hyperparameters, and the stopping criteria used during training. In an unrestricted scenario, a tree could potentially grow quite deep to fit the training data perfectly, but it might lead to overfitting. |
| --- |
| **2. 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?**  **Sol:**   * The Gini impurity of a node is usually lower than that of its parent. * The Gini impurity is a measure of the disorder or impurity in a set of instances, and decision trees aim to decrease this impurity as they split nodes. * By dividing the data into subsets, each with a more homogeneous class distribution, the Gini impurity tends to decrease from parent to child nodes. |
| **3. Explain if its a good idea to reduce max depth if a Decision Tree is overfitting the training set?**  **Sol:**   * Reducing the max depth of a Decision Tree can be a good idea if the tree is overfitting the training set. * Overfitting occurs when the tree is too complex and captures noise in the data, leading to poor generalization to unseen data. * By reducing the max depth, the tree becomes less complex, and it's less likely to capture noise. * This can improve the tree's ability to generalize well to new instances. |
| **4. Explain if its a good idea to try scaling the input features if a Decision Tree underfits the training set?**  **Sol:**   * Scaling the input features is generally not necessary for Decision Trees, as they are insensitive to monotonic transformations of the data (like scaling). * Decision Trees make decisions based on feature thresholds, and the order or magnitude of features doesn't affect their splitting decisions. * Scaling might be more relevant for algorithms like k-nearest neighbors or gradient-based methods, where distances between instances matter. |
| **5. 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?**  **Sol:**   * The time it takes to train a Decision Tree is not linear with respect to the number of instances, and it depends on various factors like the complexity of the data and the implementation of the training algorithm. * However, assuming similar conditions, training a Decision Tree on a 10 million instance training set could take more than 10 times longer than training on a 1 million instance set. So, it might take around 10 hours or more, depending on the factors mentioned earlier. |
| **6. Will setting presort=True speed up training if your training set has 100,000 instances?**  **Sol:**   * Setting presort=True in a Decision Tree may not necessarily speed up training, especially if your training set has a large number of instances like 100,000. * The presort option pre-sorts the data to speed up the tree building process, but this comes at a cost, particularly with larger datasets. The sorting operation can be computationally expensive and may outweigh the benefits in terms of training speed. It's often recommended to leave presort as its default (False) for larger datasets |
| **7. Follow these steps to train and fine-tune a Decision Tree for the moons dataset:**  **a. To build a moons dataset, use make moons(n samples=10000, noise=0.4).**  **b. Divide the dataset into a training and a test collection with train test split().**  **c. 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.**  **d. 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.** |
| **8. 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!** |