1. How does a decision tree work?

A decision tree works by recursively splitting the data into subsets based on the most significant features. It's like a flowchart, where each internal node represents a test on a feature (e.g., "Is chol > 250?"), each branch represents the outcome of the test, and each leaf node represents a class label (e.g., "No Disease" or "Disease"). The tree continues to split the data until it can no longer find a useful split or reaches a predefined depth.

1. **What is entropy and information gain?**
   * **Entropy** is a measure of the impurity or randomness in a dataset. In the context of a classification problem, it tells you how mixed the classes are in a given set of data. A set with all items belonging to the same class has zero entropy (perfectly pure), while a set with an equal mix of classes has maximum entropy (perfectly impure).
   * **Information Gain** is the reduction in entropy that results from a data split. When building a decision tree, the algorithm chooses the feature that provides the highest information gain, meaning it creates the purest possible child nodes from a given parent node.
2. How is a random forest better than a single tree?

A single decision tree can be prone to overfitting, meaning it learns the training data too well and performs poorly on new, unseen data. A random forest is an "ensemble" of many decision trees. It builds multiple trees on random subsets of the data and a random subset of the features. The final prediction is determined by a majority vote of all the trees. This averaging process reduces the model's variance and helps it generalize better to new data, making it more robust and accurate than a single tree.

1. What is overfitting and how do you prevent it?

Overfitting occurs when a machine learning model learns the training data too specifically, including its noise and random fluctuations, rather than the underlying patterns. This results in high accuracy on the training set but low accuracy on the test set.

You can prevent it by:

* + **Limiting tree depth:** As shown in the Canvas, setting a max\_depth (e.g., max\_depth=5) prevents the tree from becoming too complex.
  + **Pruning:** Removing branches from a tree that have little predictive power.
  + **Using ensemble methods:** Random Forests, as discussed above, are a great way to combat overfitting.
  + **Cross-validation:** Evaluating the model on different subsets of the data helps you get a more reliable estimate of its performance on unseen data.

1. What is bagging?

Bagging, short for Bootstrap Aggregating, is an ensemble meta-algorithm used to improve the stability and accuracy of machine learning algorithms. It works by training multiple versions of a model on different, random subsets of the training data (drawn with replacement). The final prediction is then made by combining the results of all the models, typically through voting (for classification) or averaging (for regression). Random Forest is a prime example of a model that uses bagging.

1. How do you visualize a decision tree?

In Python, you can visualize a decision tree using the plot\_tree function from Scikit-learn. The script uses this to show a pruned version of the tree, which is much easier to read than a full, deep tree. This visualization helps you understand the decision-making process, showing which features are most important and at what thresholds the splits are being made.

1. How do you interpret feature importance?

Feature importance scores indicate how much a feature contributed to the model's predictive power. In a random forest, this is typically measured by how much the accuracy decreases if you remove that feature. The higher the score, the more important the feature is to the model's decision-making process. The bar chart in the script visually represents this, showing you which features (like thal and cp) were the most influential in predicting the target outcome.

1. **What are the pros and cons of random forests?**
   * **Pros:** High accuracy, robust against overfitting, can handle a large number of features, less sensitive to outliers, and provides a useful measure of feature importance.
   * **Cons:** Can be slow to train, especially with a large number of trees, and the model itself is less interpretable than a single decision tree due to the large number of individual trees.