Question 1:

1. How are trees constructed?

Trees are constructed using a recursive splitting process, where a dataset is divided into smaller subsets based on feature values. The process starts at the root node and continues down, forming branches and nodes until a stopping criterion (like the maximum depth or minimum samples per leaf) is met.

2. How do trees handle non-linear relationships between variables? Compare this to linear models.

Trees handle non-linear relationships well because they recursively partition the feature space into smaller, more homogeneous regions without assuming a specific form (like linearity). Unlike linear models, which assume a linear relationship between predictors and the target variable, trees can capture the more complex interactions between variables.

3. Why is the Gini a good loss function for categorical target variables?

Gini impurity measures the likelihood of incorrectly classifying a randomly chosen instance from a subset of data. It is a good loss function for categorical targets because it provides a computationally efficient way to determine how "pure" a node is, meaning how well-separated the classes are.

4. Why do trees tend to overfit, and how can this tendency be constrained?

Trees tend to overfit because they can grow too deep and learn patterns that are specific to the training data rather than generalizable to unseen data. To constrain overfitting, you can limit the tree depth, set a minimum number of samples required for a split and use pruning techniques.

5. True or false, and explain: Trees only really perform well in situations with lots of categorical variables as features/covariates.

False. While decision trees handle categorical variables well (because they can split on specific categories), they also perform well with numerical variables. Trees can capture interactions between numerical and categorical features without needing to convert categorical variables into numerical representations explicitly.

6. Why don't most versions of classification/regression tree concept allow for more than two branches after a split?

Most decision tree implementations use binary splits (i.e., two branches per node) because binary splitting simplifies optimization and leads to more balanced trees. It also reduces computational complexity, as choosing the best split among many possible multi-way splits can be inefficient.

7. What are some heuristic ways you can examine a tree and decide whether it is probably over- or under-fitting?

Signs of overfitting:

- The tree is too deep with many nodes, indicating it has memorized the training data rather than learning general patterns.
- There is a very high accuracy on training data but much lower performance on validation/test data.

Signs of underfitting:

- The tree is too shallow, meaning it has too few splits and fails to capture important patterns.
- The training and validation errors are both high, suggesting the model is too simple to fit the data well.