Comprehensive Notes on Decision Trees and Weighted K-Nearest Neighbors

These notes cover the core concepts taught in the class about Decision Trees (DT) and Weighted K-Nearest Neighbors (KNN) using the audio transcript from the session.

Decision Trees (DT)

Introduction to Decision Trees

- Decision Trees are a popular method for various classification and regression tasks. They function by splitting the data into subsets based on the value of input features and use a tree-like model of decisions.

How Decision Trees Work

- **Root Node**: The topmost node of a decision tree which starts the decision-making process. It includes all samples and is the most heterogeneous.
- **Splitting**: Splitting is typically performed on the feature providing the highest Information Gain, meaning it reduces the entropy or impurity the most.
- **Entropy**: Entropy measures the impurity or disorder. Lower entropy means higher purity and is desired after a split. Entropy is given by the formula, $Entropy(S) = -\sum(p_i \cdot log_2(p_i))$ where p is the proportion of class i in the set S【6:6†source】.

Splitting Criteria

- **Information Gain**: Calculated using the initial entropy minus the weighted sum of the entropies of each split. This gain is highest when the split produces the purest child nodes .
- **Entropy Calculation Example**: Given a node with 100 points split into two nodes with entropies calculated as shown, the weighted average gives the net entropy .

- **Goal**: To create leaf nodes (terminal nodes) with minimal entropy (high purity).
- **Pure Node**: A pure node has a singular class, resulting in zero entropy .

Decision Trees for Regression

- Though primarily used for classification, decision trees can also perform regression by predicting the means of continuous output values.

Real-world Analogies and Examples

- Decision Trees can be thought of as if-else statements that classify data points by progressing from the root node to the leaf nodes according to feature tests at each node .

Weighted K-Nearest Neighbors (KNN)

Introduction to Weighted KNN

- Weighted KNN is a variant of the KNN algorithm where each of the k nearest neighbors is given different importance (weights) based on their distance to the query point.

How Weighted KNN Works

- **Weight Calculation**: The weight assigned to each neighbor is typically inversely proportional to the distance from the query point. For example, weight can be defined as $weight_i = \frac{1}{distance_i}$.
- **Classification Decision**: After weighing each neighbor based on proximity, classification is based on voting with these weights. The class with the highest summed weights wins .

Examples and Edge Cases

- The algorithm benefits scenarios with varying densities or where certain classes need to be more influential due to their proximity, despite being fewer in number .

- Weighted KNN is particularly useful in datasets where the distribution of classes is uneven or where data points are not uniformly distributed over the feature space .

Distance Metrics

- Commonly used distance metrics include Euclidean (L2 norm), Manhattan, and Minkowski distance. The choice depends on the distribution and nature of the data .

These notes summarize key insights from the class and provide a detailed breakdown of the concepts with examples and formulas for a deeper understanding. Make sure to practice implementing these algorithms to solidify your learning.