For all codes, I mostly used my own implementation while utilizing tutorials

Building Decision Tree Classifier from scratch using ID3 algorithm:

The Node Class is a typical component in decision tree implementations. The Node class normally has characteristics such as children, value, isLeaf, and pred or label for storing decision tree node information. For example, VTUPulse's implementation employs a Node class with comparable properties. Entropy and Information Gain are critical computations in the ID3 algorithm. The entropy function computes the entropy of a dataset, while the information gain function computes the gain for a specific property. This idea is a fundamental component of the ID3 algorithm and appears in varied forms among implementations. Building a decision tree requires selecting the optimum characteristic to divide on based on information gain or entropy reduction. The recursive function used to form the tree is a classic strategy in decision tree algorithms, as demonstrated in the VTUPulse tutorial. On managing various types of data, The algorithm takes into account both categorical and numerical data in feature splits. This is a typical technique in decision tree implementations, as various sorts of features necessitate distinct handling methods. Most decision tree implementations also contain a function that predicts labels for additional data points using the created tree. This entails traversing the tree using feature values from the input data until a leaf node (prediction) is reached.

Reference:

Decision Tree ID3 Algorithm in Python - VTUPulse

For the second chunk of the code:

A typical first step in a machine learning workflow is to load data with Pandas (similar to reading an Excel file) and preprocess it (such as addressing missing values or converting categorical characteristics to numerical values). This preprocessing step is critical, especially for datasets containing categorical characteristics, because most machine learning algorithms require numerical input. Several tutorials, including DataCamp and datagy, illustrate similar preparation processes. The process of categorizing a dataset into features (X) and labels (y) is crucial in supervised learning. The feature set is often made up of independent variables, whereas the label is what I'm attempting to predict or categorize. This stage is thoroughly defined in sites like Stack Abuse, which show how to partition data into characteristics and labels. The train_test_split function is commonly used to split data into training and testing sets. This function shuffles and divides the data according to the provided test size, which is critical for testing a model's performance on previously unknown data. This process is a regular practice, as seen in tutorials like datagy and Stack Abuse. Training a machine learning model, such as the ID3 Decision Tree Classifier, and assessing it using metrics such as accuracy score are critical components of model development. The model is trained on the training set first, then its performance is evaluated on the test set. This strategy is a typical procedure in machine learning and is well-documented across several learning platforms and courses.

References:

Python Decision Tree Classification Tutorial: Scikit-Learn DecisionTreeClassifier | DataCamp

Decision Tree Classifier with Sklearn in Python • datagy

Decision Trees in Python with Scikit-Learn (stackabuse.com)

For third chunk of the code:

Setting up a custom classifier and the standard DecisionTreeClassifier from Scikit-Learn for comparison is a fantastic way to grasp the differences and efficacy of different approaches. While individual implementations may differ, the notion of comparing a bespoke model to a standard implementation is used in a variety of machine learning research to assess the performance of alternative models. Assessing classifier performance using a variety of measures such as accuracy, precision, recall, and F1 score offers a comprehensive picture of the model's efficacy. This multi-metric evaluation is a standard strategy in machine learning for obtaining a holistic view of a model's performance, as described in materials from Stack Abuse and Machine Learning Knowledge.

For fourth junk of the code:

Machine learning models are typically evaluated using measures such as accuracy, precision, recall, and F1 score. These metrics give a thorough perspective of the model's performance and are critical for evaluating classification methods. Several publications, like Machine Learning Mastery, frequently address these measures in the context of evaluating machine learning algorithms. Iterating over arrays of max_depths and training_sizes to assess model performance in various contexts is a typical experimental setting in machine learning. This method is used to determine how various hyperparameters and training sizes impact the model's performance.

References:

Metrics To Evaluate Machine Learning Algorithms in Python - MachineLearningMastery.com

Other than above statements, these are the resources I looked up while doing my own implementation:

- https://neptune.ai/blog/ml-model-evaluation-and-selection
- Is it possible to use a custom-defined decision tree classifier in Scikit-learn? - Stack Overflow How to build decision tree from scratch
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html for general overview
- https://datagy.io/sklearn-decision-tree-classifier/ for comparison
- https://www.kdnuggets.com/2021/05/deal-with-categorical-data-machine-learning.html for handling categorical features
- machine learning - Understanding features vs labels in a dataset - Data Science Stack Exchange
- Train Test Split - How to split data into train and test for validating machine learning models? - Machine Learning Plus (for train test-split)
- https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/ How to Calculate Precision, Recall, F1, and More for Deep Learning Models
- Nabi Nabiyev, LU MSc Data Science 2024 student, helped me debugging the code

- https://www.explorium.ai/blog/machine-learning/the-complete-guide-to-decision-trees/
-