**CO544: Machine Learning and Data Mining**

**Lab 03: Decision Trees and k-Nearest Neighbors Classification**

**E/20/197**

## Task 1:

Build two decision tree classifiers with Gini index and entropy criteria for the given Wine.csv dataset. More information on the dataset is available on UCI Machine Learning Repository (source: https://archive.ics.uci.edu/ml/datasets/Wine).

1. Demonstrate how decision trees deal with missing values.

Decision trees do not explicitly handle missing values, specially in scikit learn. Missing data must usually be handled before training through preprocessing. We can use steps like imputation (replacing missing values using mean, median or mode) or removing (dropping rows or columns with too many missing values).

In some decision tree algorithms like C4.5 handle missing values by using probabilistic splits based on known diss or assigning partial weights to branches during training.

In this case, there are no any missing values in the dataset.

A screenshot of a cell phone

AI-generated content may be incorrect.

1. Evaluate the classifiers with suitable performance metrics.

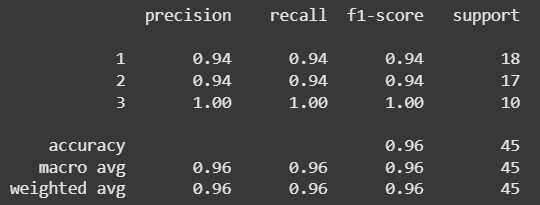
Classification Report 


Figure Classification Report Gini

Figure 2 Classification Report Entropy

**Precision** - Among the predictions made for a given class, how many are correct?

**Recall** - Out of all the actual instances of a class, how many were correctly predicted?

**F1-Score** - Harmonic mean of precision and Recall

**Support** - Actual number of occurrences of the class in your test dataset

When we consider the precision, recall and f1-score, it shows the Gini index is using as the split criterion gives the best decision tree.

1. Demonstrate how pruning can be applied to overcome overfitting of decision tree classifiers.

Pruning reduces overfitting by simplifying the tree. Pre-pruning (early stopping) limits tree growth by setting parameters like max\_depth, min\_samples\_split, min\_samples\_leaf. This prevents the tree from growing too deep and memorizing noise.



Post pruning is not included in scikit learn. It allows to tree to grow completely and then removes branches that do not improve the performance significantly.

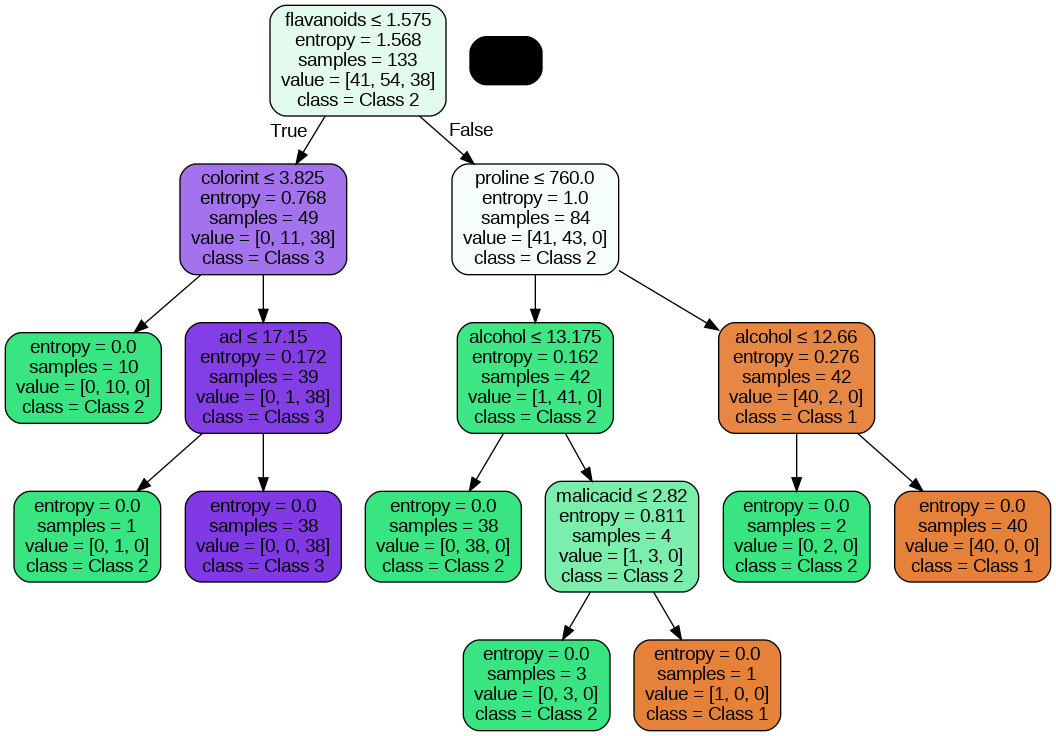
1. Visualize decision trees.

Figure4 Decision Tree using Entropy

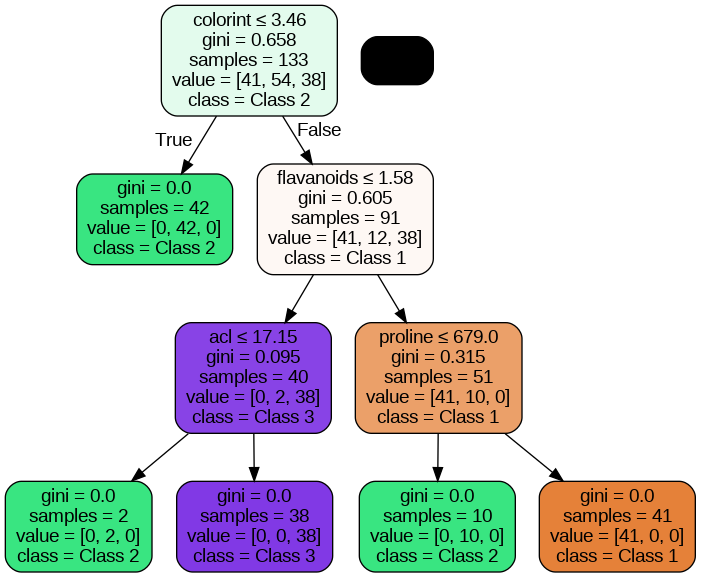


Figure 3 Decision Tree using Gini Index

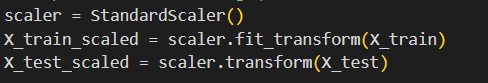
## Task 2

Apply k-Nearest Neighbors to the same Wine.csv dataset.

1. Preprocess with feature scaling.

Preprocessing was done using StandardScaler() from sklearn.

This is essential for kNN, because it uses distance calculations.Without scaling, features with larger ranges would dominate the distance.



1. Tune k (and distance metric) via cross-validation.

A black background with white text

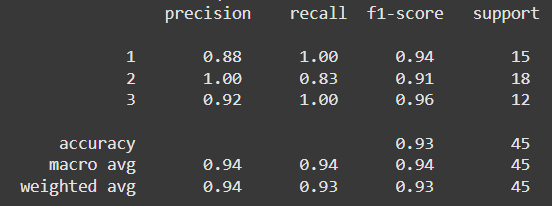
AI-generated content may be incorrect.

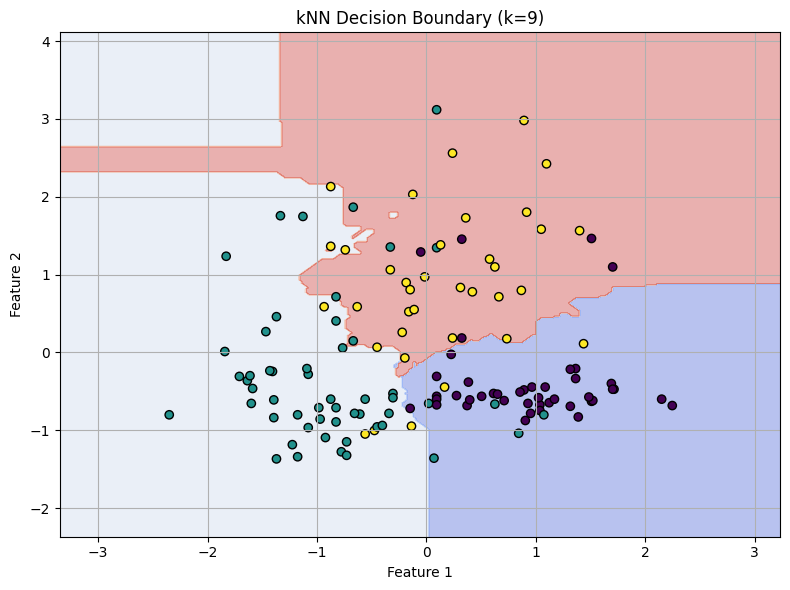
5-fold cross-validation was used to select the best pair based on validation accuracy. In this method, 4 folds are used to train the model and the remaining 1 fold is used to test the model. This process repeated 5 times and each time using a different fold as the test set and the other 4 for training. Finally the model’s performance is averaged over the 5 runs.

1. Compare kNN’s accuracy, precision/recall, and runtime to your decision-tree results.

A screenshot of a computer

AI-generated content may be incorrect.





**Compare the performance**

|  |  |  |
| --- | --- | --- |
| Metric | kNN | Decision Tree |
| Accuracy | 0.9333 | 0.9555 |
| Precision | 0.88, 1.00, 0.92 | 0.95, 1.00, 0.91 |
| Recall | 1.00, 0.83, 1.00 | 1.00, 0.88, 1.00 |
| Model Complexity | No model (Lazy learning) | Tree structure can grow deep |
| Interpretability | Low | High |

If interpretability matters, we can use decision trees.

kNN works well when we have moderate sized data and dataset is well-balanced.