CS 422 Due:23h00

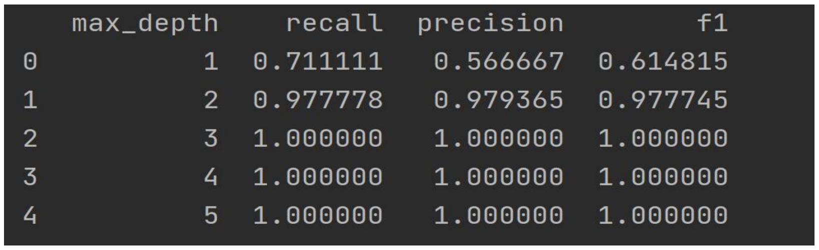
Data Mining Homework 2 March 23rd, 2025

Practicum Problems

These problems will primarily reference the lecture materials and examples provided in class using Python. It is recommended that a Jupyter/IPython notebook be used for the programmatic components. Students are expected to refer to the prescribed textbook or credible online resources to answer the questions accurately.

Problem 1

Load the Iris sample dataset from sklearn (using load_iris()) into Python with a Pandas DataFrame. Induce a set of binary decision trees with a minimum of 2 instances in the leaves (min_samples_leaf=2), no splits of subsets below 5 (min_samples_split=5), and a maximum tree depth ranging from 1 to 5 (max_depth=1 to 5). You can leave other parameters at their default values. Which depth values result in the highest Recall? Why? Which value resulted in the lowest Precision? Why? Which value results in the best F1 score? Also, explain the difference between the micro, macro, and weighted methods of score calculation.



```
   max_depth    recall  precision        f1
0          1  0.711111   0.566667  0.614815
1          2  0.977778   0.979365  0.977745
2          3  1.000000   1.000000  1.000000
3          4  1.000000   1.000000  1.000000
4          5  1.000000   1.000000  1.000000
```

The running result of the program shows that:

Max_depth 3, 4, 5 result in the highest recall. Recall measures the model's ability to correctly identify positive samples. Higher max_depth values make the model more complex, allowing it to capture more details and thus improve recall. However, if the tree is too deep, it may lead to overfitting.

Max_depth 1 results in the lowest precision. Precision measures the proportion of predicted positive samples that are actually positive. Lower max_depth values make the model too simple, potentially leading to more false positives (FP), thereby reducing precision.

Max_depth 3, 4, 5 result in the best F1 score. The F1 score is the harmonic mean of precision and recall, making it suitable for imbalanced datasets. The best F1 score typically corresponds to a max_depth value that balances precision and recall.

The **micro-average** method calculates global precision, recall, and F1 score by aggregating all true positives (TP), false positives (FP), and false negatives (FN) across all classes, treating the dataset as a whole. It is suitable for imbalanced datasets because each sample contributes equally to the final metric.

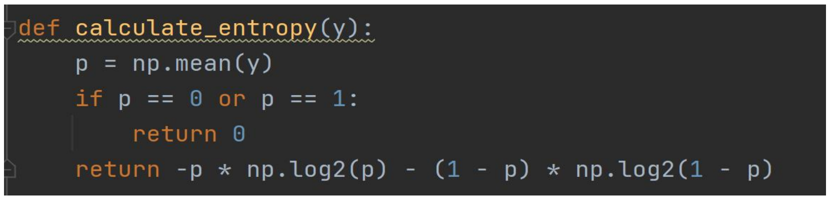
The **macro-average** method computes precision, recall, and F1 score for each class individually and then takes the unweighted mean, giving equal importance to each class regardless of its sample size. This is ideal for balanced datasets.

The **weighted-average** method also calculates metrics for each class individually but takes a weighted mean based on the number of samples in each class, making it suitable for imbalanced datasets where class distribution varies significantly.

Problem 2

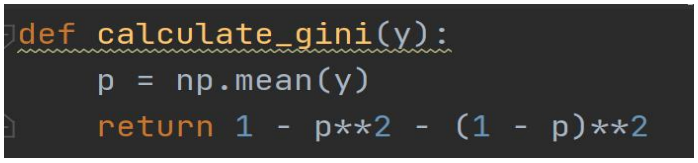
Load the Breast Cancer Wisconsin (Diagnostic) sample dataset from the UCI Machine Learning Repository (the discrete version at: breast-cancer-wisconsin.data) into Python using a Pandas DataFrame. Induce a binary Decision Tree with a minimum of 2 instances in the leaves, no splits of subsets below 5, and a maximum tree depth of 2 (using the default Gini criterion). Calculate the Entropy, Gini, and Misclassification Error of the first split. What is the Information Gain? Which feature is selected for the first split, and what value determines the decision boundary?

calculate the Entropy



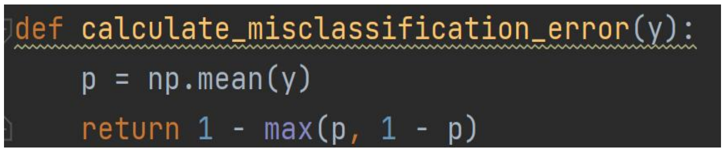
```
def calculate_entropy(y):
    p = np.mean(y)
    if p == 0 or p == 1:
        return 0
    return -p * np.log2(p) - (1 - p) * np.log2(1 - p)
```

calculate the Gini



```
def calculate_gini(y):
    p = np.mean(y)
    return 1 - p**2 - (1 - p)**2
```

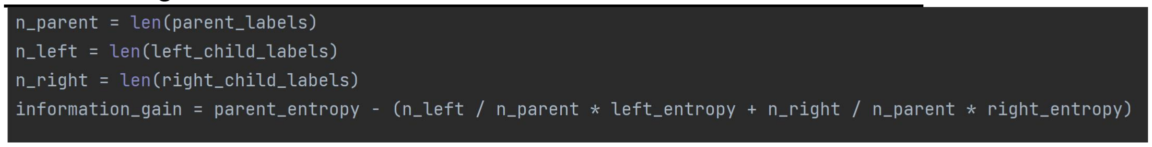
calculate the Misclassification Error



```
def calculate_misclassification_error(y):
    p = np.mean(y)
    return 1 - max(p, 1 - p)
```

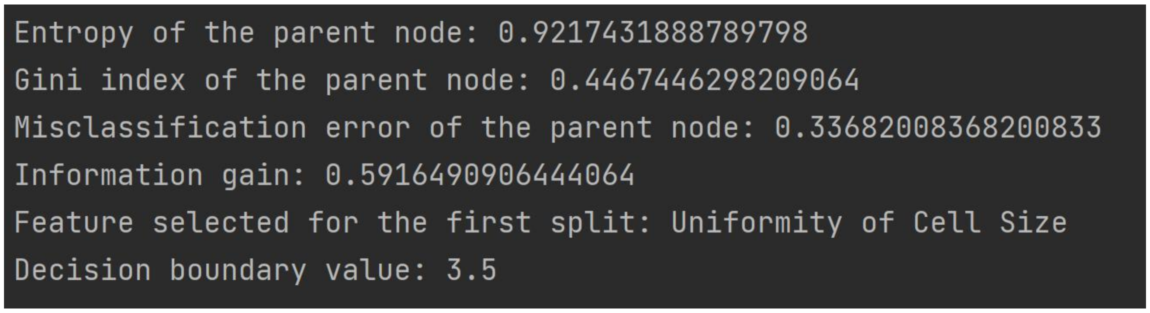
calculate the Information Gain

E.N.D



```
n_parent = len(parent_labels)
n_left = len(left_child_labels)
n_right = len(right_child_labels)
information_gain = parent_entropy - (n_left / n_parent * left_entropy + n_right / n_parent * right_entropy)
```

Here are the results:



```
Entropy of the parent node: 0.9217431888789798
Gini index of the parent node: 0.4467446298209064
Misclassification error of the parent node: 0.33682008368200833
Information gain: 0.5916490906444064
Feature selected for the first split: Uniformity of Cell Size
Decision boundary value: 3.5
```

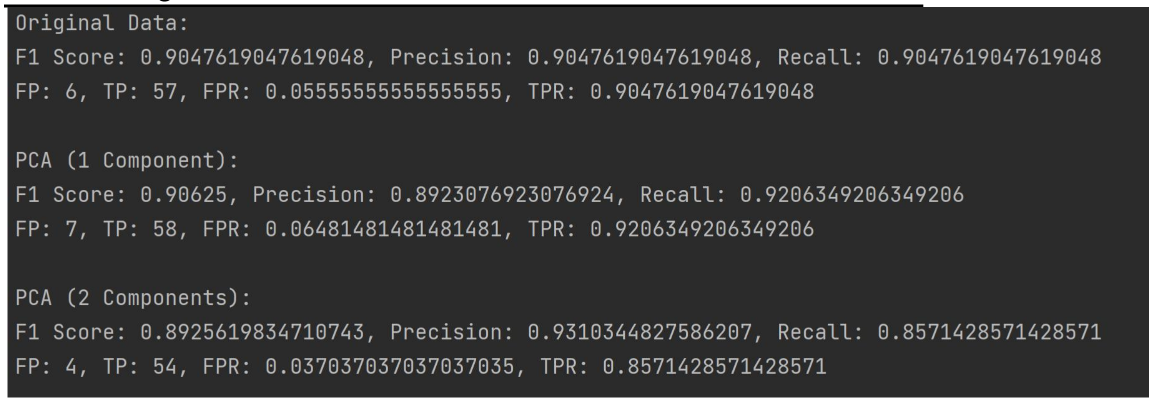
The results show that the entropy of the parent node is 0.9217, the Gini index is 0.4467, and the misclassification error is 0.3368, indicating that the initial dataset has high uncertainty and poor classification performance. Through the first split, the information gain reaches 0.5916, significantly improving the purity of the dataset. The decision tree selects "Uniformity of Cell Size" as the feature for the first split, with a decision boundary of 3.5, demonstrating that this feature plays a key role in classification and can effectively distinguish samples of different categories.

Problem 3

Load the Breast Cancer Wisconsin (Diagnostic) sample dataset from the UCI Machine Learning Repository (the continuous version at: wdbc.data) into Python using a Pandas DataFrame. Induce the same binary Decision Tree as above (now using the continuous data), but perform PCA dimensionality reduction beforehand. Using only the first principal component of the data for model fitting, what are the F1 score, Precision, and Recall of the PCA-based single factor model compared to the original (continuous) data? Repeat the process using the first and second principal components. Using the Confusion Matrix, what are the values for False Positives (FP) and True Positives (TP), as well as the False Positive Rate (FPR) and True Positive Rate (TPR)? Is using continuous data beneficial for the model in this case? How?"

Here the results:

E.N.D



```
Original Data:
F1 Score: 0.9047619047619048, Precision: 0.9047619047619048, Recall: 0.9047619047619048
FP: 6, TP: 57, FPR: 0.05555555555555555, TPR: 0.9047619047619048

PCA (1 Component):
F1 Score: 0.90625, Precision: 0.8923076923076924, Recall: 0.9206349206349206
FP: 7, TP: 58, FPR: 0.06481481481481481, TPR: 0.9206349206349206

PCA (2 Components):
F1 Score: 0.8925619834710743, Precision: 0.9310344827586207, Recall: 0.8571428571428571
FP: 4, TP: 54, FPR: 0.037037037037037035, TPR: 0.8571428571428571
```

Based on the running results, PCA dimensionality reduction with the first principal component achieves a slightly higher F1 score (0.9063) compared to the original data (0.9048), along with a higher recall (0.9206), indicating that it reduces the number of features with almost no loss in performance, or even a slight improvement. When using two principal components, the F1 score (0.8926) slightly decreases, but the precision (0.9310) significantly improves, and the false positive rate (0.0370) is the lowest, demonstrating better performance in reducing false positives. Overall, the original continuous data generally performs better due to containing more information, but PCA dimensionality reduction (especially with the first principal component) significantly reduces computational complexity with almost no performance loss, making it an effective trade-off. Therefore, if computational resources are limited, using the first principal component is an efficient choice; if higher precision and lower false positive rates are desired, using two principal components can be considered; and when resources are sufficient, directly using the original continuous data yields the best performance.

E.N.D