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

problem 1

[12]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_score, recall_score, f1_score

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train decision tree models and evaluate performance at different depths
results = []
for depth in range(1, 6):
    clf = DecisionTreeClassifier(max_depth=depth, min_samples_split=5, min_samples_leaf=2, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    results.append((depth, precision, recall, f1))

# Print results
print("Depth | Precision | Recall | F1 Score")
for depth, precision, recall, f1 in results:
    print(f"{depth}     | {precision:.4f}  | {recall:.4f} | {f1:.4f}")
```

```
Depth | Precision | Recall | F1 Score
1     | 0.5667  | 0.7111 | 0.6148
2     | 0.9794  | 0.9778 | 0.9777
3     | 1.0000  | 1.0000 | 1.0000
4     | 1.0000  | 1.0000 | 1.0000
5     | 1.0000  | 1.0000 | 1.0000
```

Based on the results, when the maximum depth is 3, 4, and 5, the Recall, Precision, and F1 score all reach 1.0000, indicating that the

E.N.D

model can perfectly predict the class of all samples at these depths. However, when the depth is 1, the Precision is the lowest (0.5667), which is because the tree depth is too small, making the model overly simple and unable to capture the complex features of the data, resulting in inaccurate predictions. When the depth is 2, the F1 score is 0.9777, which is slightly lower than the models with a depth of 3 or above but still performs well. Regarding the scoring methods, the micro method calculates the global metrics for all samples and is suitable for imbalanced class scenarios; the macro method averages the metrics for each class, treating all classes equally; and the weighted method calculates the metrics by weighting them according to the sample size of each class, making it suitable for imbalanced class scenarios.

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?

problem2

[13]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
columns = ['id', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
           'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei',
           'Bland Chromatin', 'Normal Nucleoli', 'Mitoses', 'Class']
df = pd.read_csv(url, names=columns)
df.replace('?', pd.NA, inplace=True)
df.dropna(inplace=True)

# Features and target variables
X = df.iloc[:, 1:-1]
y = df['Class'].apply(lambda x: 0 if x == 2 else 1)  # Convert classes to 0 and 1

# Train the decision tree model
clf = DecisionTreeClassifier(max_depth=2, min_samples_split=5, min_samples_leaf=2, random_state=42)
clf.fit(X, y)

# Calculate feature importance
print("Feature Importances:", clf.feature_importances_)
```

```
Feature Importances: [0.         0.87326695 0.08084315 0.         0.         0.0458899
 0.         0.         0.        ]
```

E.N.D

In Problem 2, the feature importance is [0. 0.87326695 0.08084315 0. 0. 0.0458899 0. 0. 0.], indicating that the second feature plays a dominant role in the decision tree split, with the highest importance (0.87326695), while the importance of other features is low or zero, meaning these features were not used in the splitting process. The Information Gain can be obtained by calculating the difference in entropy or Gini index before and after the split, and the feature with the highest Information Gain is typically selected for the split.

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?"

problem3

[14]:
```
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# PCA for dimensionality reduction
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Train the decision tree model on PCA-transformed data
clf_pca = DecisionTreeClassifier(max_depth=2, min_samples_split=5, min_samples_leaf=2, random_state=42)
clf_pca.fit(X_pca, y)

# Evaluate the model
y_pred_pca = clf_pca.predict(X_pca)
cm = confusion_matrix(y, y_pred_pca)
precision = precision_score(y, y_pred_pca)
recall = recall_score(y, y_pred_pca)
f1 = f1_score(y, y_pred_pca)

print("Confusion Matrix:\n", cm)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Confusion Matrix:
 [[429  15]
 [  1 238]]
Precision: 0.9407114624505929
Recall: 0.99581589958159
F1 Score: 0.967479674796748
```

E.N.D

The confusion matrix shows that the False Positives (FP) are 15, the True Positives (TP) are 238, the False Positive Rate (FPR) is FP / (FP + TN) = 15 / (15 + 429) ≈ 0.0337, and the True Positive Rate (TPR) is TP / (TP + FN) = 238 / (238 + 1) ≈ 0.9958. After performing PCA dimensionality reduction on the continuous data, the performance of the model using only the first principal component (F1 score of 0.9675, Precision of 0.9407, Recall of 0.9958) slightly decreases compared to the original data but remains at a high level. When using the first and second principal components, the model performance may further improve. Overall, using continuous data helps capture more detailed information, but in high-dimensional data, PCA dimensionality reduction can effectively reduce computational complexity and avoid overfitting.

E.N.D