CO544 Lab03 E20280

Task 1: Decision Tree Classifiers

(a) Handling Missing Values in Decision Trees

Decision trees handle missing values through several strategies:

1. **Imputation (used here):**
    - Missing values are replaced by the **mean** of the feature.

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
```

2. **Native Handling (scikit-learn >= 1.0):**
    - Decision trees can route samples with missing values during splitting.
3. **Indicator Variables:**
    - Introduce new binary features indicating if a value was missing.
4. **Model-Based Imputation:**
    - Estimate missing values using predictive models.

In this lab, **mean imputation** was applied before model training.

(b) Evaluate the Classifiers with Suitable Performance Metrics

Two decision trees were trained using:

- **Gini Index**
- **Entropy**

Metrics used for evaluation:

- Accuracy
- Precision
- Recall
- F1-Score

Results:

Gini Report:

```
              precision    recall  f1-score   support

           1       0.95      1.00      0.97        18
           2       1.00      0.88      0.94        17
           3       0.91      1.00      0.95        10

    accuracy                           0.96        45
   macro avg       0.95      0.96      0.95        45
weighted avg       0.96      0.96      0.95        45
```

Entropy Report:

```
              precision    recall  f1-score   support

           1       0.94      0.94      0.94        18
           2       0.94      0.94      0.94        17
           3       1.00      1.00      1.00        10

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

Both models show excellent classification performance with **96% accuracy**.

(c) Demonstrate Pruning to Overcome Overfitting

Pre-Pruning (max_depth):

- Reduces tree size by limiting the maximum depth before training.
- Prevents overfitting by restricting tree complexity.

```
clf_gini_pruned = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)
```

Post-Pruning (max_depth):

- Prunes the tree after fitting by limiting the depth.
- Alternatively, **ccp_alpha** parameter can be used for **cost-complexity pruning**.

```
clf_gini_postpruned = DecisionTreeClassifier(criterion='gini', max_depth=2, random_state=0)
```

Other Pruning Techniques:

- `min_samples_split`
- `min_samples_leaf`
- `max_leaf_nodes`
- `ccp_alpha`

Both pre- and post-pruned decision trees were visualized to demonstrate reduced complexity.

(d) Visualisations done during the lab

Gini tree (pre-pruned) (Gini tree (pre-pruned))

Gini tree (post-pruned) (Gini tree (post-pruned))

Entropy tree (pre-pruned) (Entropy tree (pre-pruned))

Entropy tree (pre-pruned) (Entropy tree (post-pruned))

Task 2: k-Nearest Neighbors (kNN)

(a) Preprocess with Feature Scaling

kNN relies on distance calculations, so **feature scaling** is critical. Used **StandardScaler**:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

(b) Tune k (and Distance Metric) via Cross-Validation

Used **GridSearchCV** to search for the optimal hyperparameters:

- Number of neighbors (`n_neighbors` from 1 to 20)
- Distance metric: `'euclidean'` or `'manhattan'`

```
param_grid = {
    'n_neighbors': list(range(1, 21)),
    'metric': ['euclidean', 'manhattan']
}
```

Best Parameters Found:

```
Best k: 7
Best metric: euclidean
```

The runtime for grid search and fitting: **~0.67 seconds**.

(c) Compare kNN's Accuracy, Precision/Recall, and Runtime to Decision Tree Results

kNN Results:

```
Accuracy: 0.9556
Classification Report:
              precision    recall  f1-score   support
```

```
           1       0.90      1.00      0.95        18
           2       1.00      0.88      0.94        17
           3       1.00      1.00      1.00        10

    accuracy                           0.96        45
   macro avg       0.97      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

Decision Tree (Gini) Results:

```
Accuracy: 0.9556
Classification Report:
              precision    recall  f1-score   support

           1       0.95      1.00      0.97        18
           2       1.00      0.88      0.94        17
           3       0.91      1.00      0.95        10
```

Decision Tree (Entropy) Results:

```
Accuracy: 0.9556
Classification Report:
              precision    recall  f1-score   support

           1       0.94      0.94      0.94        18
           2       0.94      0.94      0.94        17
           3       1.00      1.00      1.00        10
```

Summary Table:

Model	Accuracy	Macro Avg F1	Runtime (s)
kNN (k=7, euclidean)	0.9556	0.96	~0.67
Decision Tree (Gini)	0.9556	0.95	~0.01
Decision Tree (Entropy)	0.9556	0.96	~0.01

- **Accuracy:** All models achieved ~95.56% accuracy.
- **Macro Avg F1:** Slight advantage for Entropy Decision Tree and kNN.
- **Runtime:** Decision Trees were significantly faster to train than kNN with GridSearch.

Conclusion

- **Decision Trees:** Fast, interpretable, and effective. Can overfit but can be controlled by pruning.
- **kNN:** Slightly better balanced precision/recall. Needs careful scaling and parameter tuning.

Both models are highly competitive for this dataset. Runtime efficiency leans toward Decision Trees for smaller datasets; kNN may be beneficial for its slightly better balanced performance on certain classes.