트리 기반 분류 모델 (Tree-Based Classification)

- 데이터의 특성(feature)을 기반으로 연속적인 질문(분기 규칙)을 만들어 데이터를 분류하는 알고리즘
- 나무(Tree) 구조를 사용하여 데이터 공간을 재귀적으로 분할하고, 각 데이터가 어떤 클래스에 속하는지 예측

1. **결정 트리 분류 (Decision Tree Classifier)**:
    - 단일 트리를 사용하여 예측을 수행합니다.
    - 각 분기점(노드)에서는 모든 특성에 대해 분할 조건을 탐색하며, **정보 이득(Information Gain)**이 최대가 되거나 **지니 불순도(Gini Impurity)**가 최소가 되는 최적의 분할을 찾습니다. 즉, 분할을 통해 데이터가 특정 클래스로 가장 잘 나뉘는 조건을 찾습니다.
    - 이 과정을 반복하여 트리를 성장시키고, 최종 리프 노드(leaf node)에서는 가장 많은 데이터가 속한 클래스를 해당 노드의 예측 클래스로 결정합니다.
    - **장점**: 모델이 직관적이고 시각화하여 해석하기 매우 쉽습니다.
    - **단점**: 훈련 데이터에 과적합(Overfitting)되기 매우 쉬워 일반화 성능이 떨어질 수 있습니다.
2. **랜덤 포레스트 분류 (Random Forest Classifier)**:
    - 결정 트리의 과적합 문제를 해결하기 위해 등장한 **앙상블(Ensemble)** 기법입니다.
    - **배깅(Bagging)**: 원본 훈련 데이터에서 중복을 허용하여 여러 개의 샘플(부트스트랩 샘플)을 만듭니다.
    - **무작위 특성 선택**: 각 샘플로 결정 트리를 훈련시키되, 각 노드에서 분할을 위한 최적의 특성을 찾을 때 모든 특성을 고려하는 것이 아니라, 무작위로 선택된 일부 특성 중에서만 탐색합니다. (이를 통해 각 트리가 서로 다른 특성을 학습하게 되어 다양성이 증가합니다.)
    - **결과 취합 (투표)**: 이렇게 만들어진 수많은(e.g., 100개) 결정 트리들이 각각 예측한 클래스들 중, 가장 많이 예측된 클래스(다수결 투표, Hard Voting)를 최종 예측 클래스로 결정합니다.
    - **장점**: 개별 결정 트리보다 훨씬 안정적이고 높은 예측 성능을 보이며, 과적합에 강합니다.
    - **단점**: 단일 트리보다 모델이 복잡해져 해석이 어렵고, 훈련 속도가 느립니다.

적용 가능한 상황

- 데이터의 비선형 관계나 특성 간의 복잡한 상호작용을 모델링해야 할 때 매우 효과적입니다.
- 특성의 스케일에 영향을 받지 않으므로, 별도의 스케일링 전처리가 필수는 아닙니다.
- 모델의 예측 과정을 설명하고 해석하는 것이 중요할 때 (특히 결정 트리).

구현 방법

`scikit-learn`의 `tree` 모듈에 있는 `DecisionTreeClassifier`와 `ensemble` 모듈의 `RandomForestClassifier`를 사용합니다.

용도

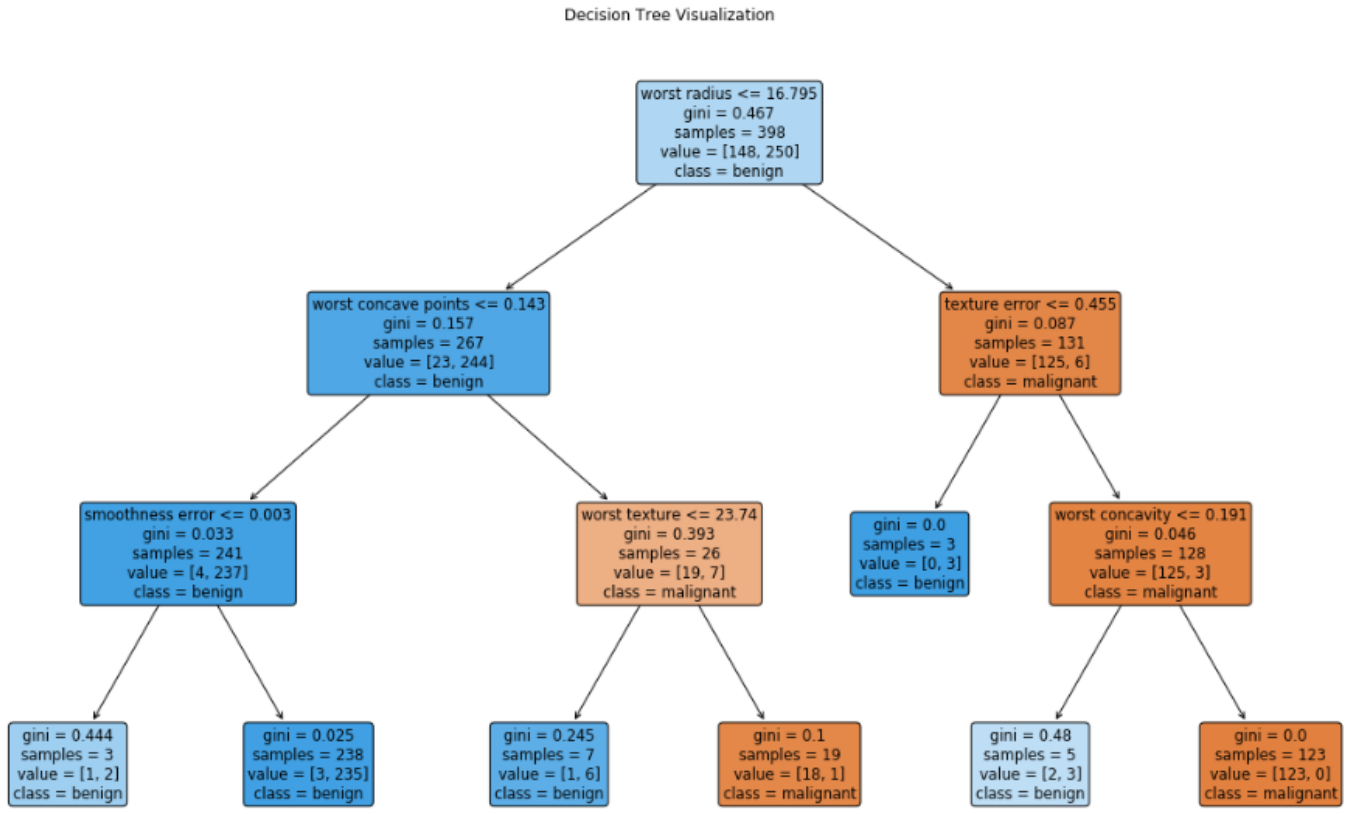
- 비선형 결정 경계를 가지는 분류 문제를 해결합니다.

주의사항

- **하이퍼파라미터 튜닝**: 트리 기반 모델은 과적합을 방지하고 최적의 성능을 내기 위해 하이퍼파라미터 튜닝이 매우 중요합니다.
  - `criterion`: 분할 품질을 측정하는 기능 ('gini', 'entropy'). (기본값='gini')
  - `max_depth`: 트리의 최대 깊이. 너무 깊으면 과적합됩니다.
  - `min_samples_split`: 노드를 분할하기 위한 최소 샘플 수.
  - `min_samples_leaf`: 리프 노드가 되기 위한 최소 샘플 수.
  - `max_features` (랜덤 포레스트): 각 분할에서 고려할 최대 특성의 수.
- **`random_state`**: `DecisionTreeClassifier`와 `RandomForestClassifier` 모두 무작위성을 포함하므로, 결과 재현을 위해 `random_state`를 고정하는 것이 좋습니다.

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt

# 1. 데이터 준비
cancer = load_breast_cancer()
X, y = cancer.data, cancer.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)

# 2. 결정 트리 분류
# 주요 하이퍼파라미터
# criterion: 분할 품질 측정 기준 ('gini' 또는 'entropy')
# max_depth: 트리의 최대 깊이.
dt_clf = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_clf.fit(X_train, y_train)
dt_pred = dt_clf.predict(X_test)
print(f"Decision Tree 정확도: {accuracy_score(y_test, dt_pred):.3f}") # 0.924

# 3. 랜덤 포레스트 분류
# 주요 하이퍼파라미터
# n_estimators: 생성할 트리의 개수.
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42,
n_jobs=-1)
rf_clf.fit(X_train, y_train)
rf_pred = rf_clf.predict(X_test)
print(f"Random Forest 정확도: {accuracy_score(y_test, rf_pred):.3f}") # 0.942

# 4. 특성 중요도 (Feature Importance) 확인
importances = pd.Series(rf_clf.feature_importances_, index=cancer.feature_names)
print("\nFeature Importances:\n",
importances.sort_values(ascending=False).head(10))
'''
worst concave points    0.163779
worst area              0.152536
worst perimeter         0.087027
mean radius             0.078079
```

```
worst radius            0.075766
mean perimeter          0.075618
mean concave points     0.063133
mean concavity          0.055289
mean area               0.042579
worst concavity         0.031887
'''

# 5. 모델 시각화 (결정 트리)
plt.figure(figsize=(20, 12))
plot_tree(dt_clf, feature_names=cancer.feature_names,
class_names=cancer.target_names, filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```



결과 해석 방법

- **모델 시각화 (`plot_tree`)**: 결정 트리의 분기 규칙을 시각적으로 확인할 수 있습니다.
  - 각 노드에는 분기 조건, `gini`(지니 불순도), `samples`(해당 노드에 속한 샘플 수), `value`(클래스별 샘플 수), `class`(예측 클래스)가 표시됩니다.
  - 이 시각화를 통해 모델이 어떤 특성을 기준으로 어떻게 분류를 수행하는지 명확하게 이해할 수 있습니다.
- **특성 중요도 (`feature_importances_`)**: 랜덤 포레스트나 결정 트리에서 제공하는 이 속성은 모델이 예측을 만들 때 각 특성이 불순도를 줄이는 데 얼마나 기여했는지를 나타냅니다. 값이 높을수록 중요한 특성입니다.

장단점 및 대안

- **장점**:
  - 비선형 관계와 변수 간 상호작용을 잘 모델링합니다.
  - 특성 스케일링이 필수가 아니며, 이상치에 비교적 강건합니다.
  - 결정 트리는 시각화를 통해 해석이 매우 용이하고, 랜덤 포레스트는 높은 성능과 특성 중요도 정보를 제공합니다.
- **단점**:
  - 결정 트리는 데이터의 작은 변화에도 구조가 크게 바뀔 수 있으며, 과적합되기 매우 쉽습니다.
  - 랜덤 포레스트는 훈련 데이터 범위 밖의 패턴을 예측하기 어렵고, 모델이 복잡하여 해석이 어렵습니다.
- **대안**:
  - **부스팅 계열 모델 (Gradient Boosting, XGBoost, LightGBM)**: 랜덤 포레스트와 유사한 앙상블 기법이지만, 일반적으로 더 높은 성능을 보입니다.
  - **서포트 벡터 머신 (SVC)**: 고차원 데이터나 복잡한 결정 경계를 가진 문제에서 좋은 성능을 낼 수 있습니다.
  - **로지스틱 회귀**: 모델의 해석이 매우 중요하고, 데이터가 선형적으로 잘 분리될 때 좋은 선택입니다.