RandomForest

Randomforest 이용해 반려견 안구질환 데이터를 분류하는 모델 구현을 위한 코드

```
# 1. 기본 모델 적용
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# 기본 RandomForestClassifier 모델 초기화
rf_model = RandomForestClassifier(random_state=42)

# 모델 학습 (훈련 데이터 전체 사용)
rf_model.fit(x_train, y_train)

# Train predictions
y_train_pred = rf_model.predict(x_train)

# Test predictions
y_pred = rf_model.predict(x_test)

# Train Accuracy 계산
train_accuracy = accuracy_score(y_train, y_train_pred)
print("\nTrain Accuracy:")
print(f"{train_accuracy:.4f}")

# Test Accuracy 계산
test_accuracy = accuracy_score(y_test, y_pred)
print("\nTest Accuracy:")
print(f"{test_accuracy:.4f}")
```

#1 기본 모델

- 훈련 데이터 x_train, y_train으로 기본 모델을 학습합니다.
- 학습된 모델로 훈련 데이터와 테스트 데이터에 대한 예측을 수행합니다.
- accuracy_score를 사용해 훈련 정확도와 테스트 정확도를 계산하고 출력합니다

```python
#2 Randomized Search 이용해 최적의 하이퍼파라미터 조합 찾기

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

# RandomForestClassifier 초기화
rf = RandomForestClassifier(random_state=42)

param_dist = {
    'n_estimators': [50, 100, 150],
    'max_depth': [10, 30 ,50,100 ],
    'min_samples_split': [ 5 ,10],
    'min_samples_leaf': [3, 5 , 10]

}

# RandomizedSearchCV 설정
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=25,  # 탐색 횟수
    cv=5,       # 교차 검증 폴드 수
    verbose=2,
    random_state=42
)

# 하이퍼파라미터 탐색 수행
random_search.fit(x_train, y_train)

# 최적 하이퍼파라미터 출력
print("Best Parameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)

# 최적 모델 추출
best_rf = random_search.best_estimator_
```

#2 RadommizedSearch 이용해 hyperparmeter tuning

- 랜덤 포레스트 모델을 초기화하고, 탐색할 하이퍼파라미터의 범위를 param_dist 딕셔너리로 정의합니다.
- RandomizedSearchCV를 설정하여, 지정된 하이퍼파라미터 조합(72개) 중 25개의 무작위 샘플에 대해 5-폴드 교차 검증을 수행합니다.
- random_search.fit()를 통해 x_train, y_train 데이터를 사용해 모델을 훈련하며 최적의 하이퍼파라미터를 찾습니다.
- best_params_는 최적의 하이퍼파라미터 조합을, best_score_는 그 조합의 최고 교차 검증 점수를 반환합니다.

```python
#3 최적의 하이퍼파라미터 조합을 이용해 5-Folfd CV 진행 그리고 정확도 평가

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay


# Stratified K-Fold 설정
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# 이미 학습된 best_rf_model을 사용하여 교차 검증만 수행
cv_scores = cross_val_score(best_rf, x_train, y_train, cv=cv, scoring='accuracy')
print(f"Cross-Validation Accuracy Scores: {cv_scores}")
print(f"Mean CV Accuracy: {cv_scores.mean():.4f}")

# 모델 학습 (훈련 데이터 전체 사용)
best_rf.fit(x_train, y_train)

# Test predictions
y_pred = best_rf.predict(x_test)

# 예측 확률 계산
y_proba = best_rf.predict_proba(x_test)

# 평가 지표 출력
print("\nConfusion Matrix:")
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
ConfusionMatrixDisplay(conf_matrix).plot()

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nTest Accuracy:")
test_accuracy = accuracy_score(y_test, y_pred)
print(f"{test_accuracy:.4f}")
```

#3 Best hyperparmeter 조합으로 RandomForest 5-fold CV 진행

- 최적 하이퍼파라미터를 기반으로 RandomForestClassifier 모델을 초기화하고, Stratified 5-Fold 교차 검증을 수행합니다.
- cross_val_score를 사용하여 교차 검증을 수행하고, 각 폴드의 정확도와 평균 정확도를 출력합니다.
- x_train 데이터를 사용해 모델을 학습하고, x_test를 기반으로 예측 값과 예측 확률을 생성합니다.
- 혼동 행렬(Confusion Matrix)을 계산하고 시각화하며, 이를 통해 모델의 분류 성능을 확인합니다.
- 테스트 데이터에 대한 분류 리포트와 정확도, confusion matrix 를 출력하여 모델의 전반적인 성능을 평가합니다.
-

#4 Best hyperparmeter 조합에서 n_estimator = 300으로 변경

➔ #3 코드에서 n_estimator = 300으로 늘렸을 때 성능이 아주 소폭 상승함을 확인할 수 있었습니다.