랜덤 포레스트 (Random Forest)

1. 랜덤 포레스트란?

결정 트리를 기반으로 하는 알고리즘
배깅(같은 알고리즘으로 여러 개의 분류기를 만드는 알고리즘)을 기반으로 각자의 데이터를 샘플링 하여 학습을 수행한 후 최종적으로 보팅을 통해 예측 결정
부트스트래핑 방식으로 분할(복원추출)

각각의 tree는 비교적 예측을 잘 할 수 있지만 데이터 일부에 overfitting하는 경향을 가진다
잘 작동하되 서로 다른 방향으로 overfitting 된 tree를 많이 만들면 그 결과를 평균냄으로써 overfitting 된 양을 줄일 수 있다
random forest에서 tree를 random하게 만드는 방법

1. tree 만들 때 사용하는 데이터를 무작위로 선택
2. 분할 테스트에서 특성을 무작위로 선택

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

x_train, x_test, y_train, y_test = human_dataset()
rf = RandomForestClassifier(random_state=0)
rf.fit(x_train, y_train)
pred = rf.predict(x_test)
accuracy = accuracy_score(y_test,pred)
```

2. 랜덤 포레스트 하이퍼 파라미터

트리 기반의 알고리즘의 단점 : 하이퍼 파라미터가 많다, 시간을 많이 투자함에도 불구하고 예측 성능이 크게 좋아지는 경우가 없다

- n_estimators : 결정 트리의 개수, 디폴트 값 10개
- max_features : 디폴트 값 auto
  - 크게 하면 : tree들은 같은 특성을 고려하므로 tree들이 매우 비슷해지고 가장 두드러진 특성을 이용해 데이터에 잘 맞춰진다
  - 작게 하면 : tree들은 많이 달라지고 각 tree는 데이터에 맞추기 위해 tree의 깊이가 깊어진다
- max_depth : 얼마나 깊게 트리를 만드냐
  - None : 최대한 깊이 (불순도 혹은 복잡도가 0일 때까지)
  - 클수록 : 정확하게 & 과대적합
  - 작을수록 : 가지치기 & 과대적합 방지

- max_leaf_nodes : 최대 몇 개 잎 노드 만들어 질 때까지 split 할 것이냐
  - 클수록 : 정확하게 & 과대적합
  - 작을수록 : 가지치기 & 과대적합 방지
- min_samples_split : 샘플이 최소한 몇개 이상이어야 split 할 것인지
  - 클수록 : 가지치기 & 과대적합 방지
  - 작을수록 : 정확하게 & 과대적합
- min_samples_leaf : 노드가 되려면 가지고 있어야 하는 최소 샘플 수
  - 클수록 : 가지치기 & 과대적합 방지
  - 작을수록 : 정확하게 & 과대적합

```
from sklearn.model_selection import GridSearchCV
params ={
    'n_estimators':[100],
    'max_depth':[6,8,10,12],
    'min_samples_leaf':[8,12,18],
    'min_samples_split':[8,16,20]
}

# GridSearch 통해 최적의 params 찾기
rf = RandomForestClassifier(random_state=0, n_jobs=-1)
grid_cv = GridSearchCV(rf, param_grid=params, cv=3, n_jobs=-1)
grid_cv.fit(x_train,y_train)
print('final params', grid_cv.best_params_)


rf_1=RandomForestClassifier(n_estimators=300, max_depth=10, min_samples_leaf=8, min_samples_split=8,
random_state=0)
rf_1.fit(x_train,y_train)
pred=rf_1.predict(x_test)

print('예측 정확도:{0:.4f}'.format(accuracy_score(y_test,pred)))
```

3. 랜덤 포레스트 피처 중요도

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

importances_values = rf.feature_importances_
importances = pd.Series(importances_values, index=x_train.columns)
top20 = importances.sort_values(ascending=False)[:20]
plt.figure(figsize=(8, 6))
plt.title('Feature importances Top 20')
sns.barplot(x = top20, y = top20.index)
plt.show()
```