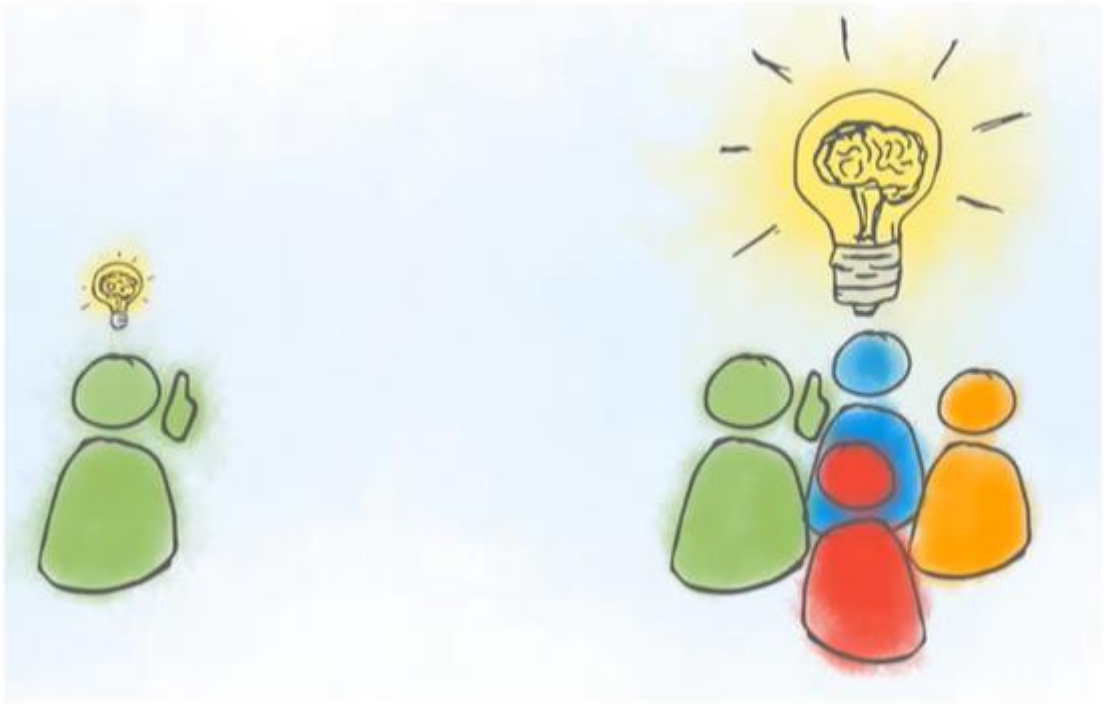
Yoo_Kye0m

RandomForest

91714167 유재겸

Contents

앙상블 학습이란??

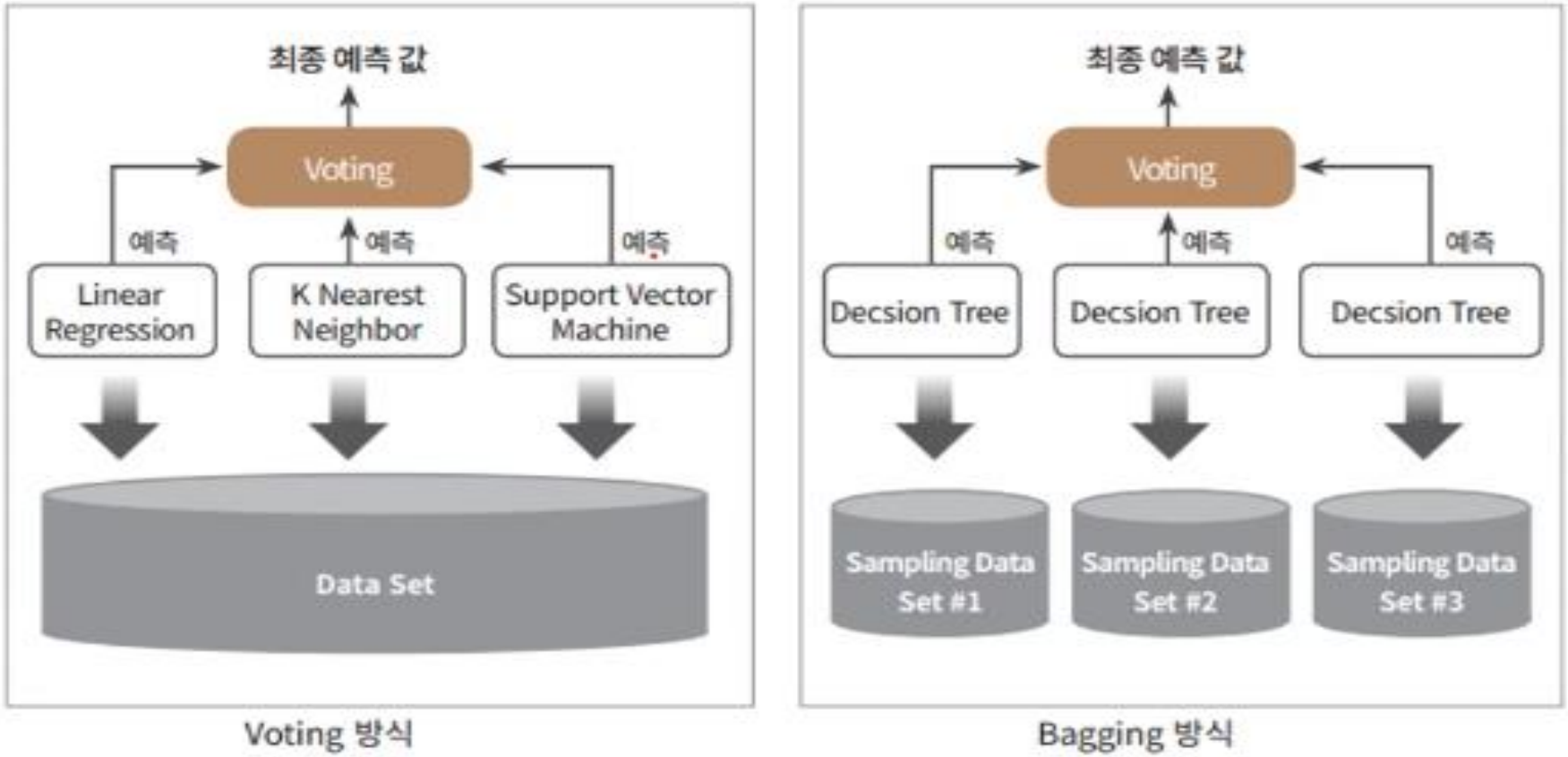


앙상블 학습은 여러 개의 분류기를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법을 말합니다.

앙상블 유형

보팅(Voting)　　배깅(Bagging)　　부스팅(Boosting)

보팅과 배깅

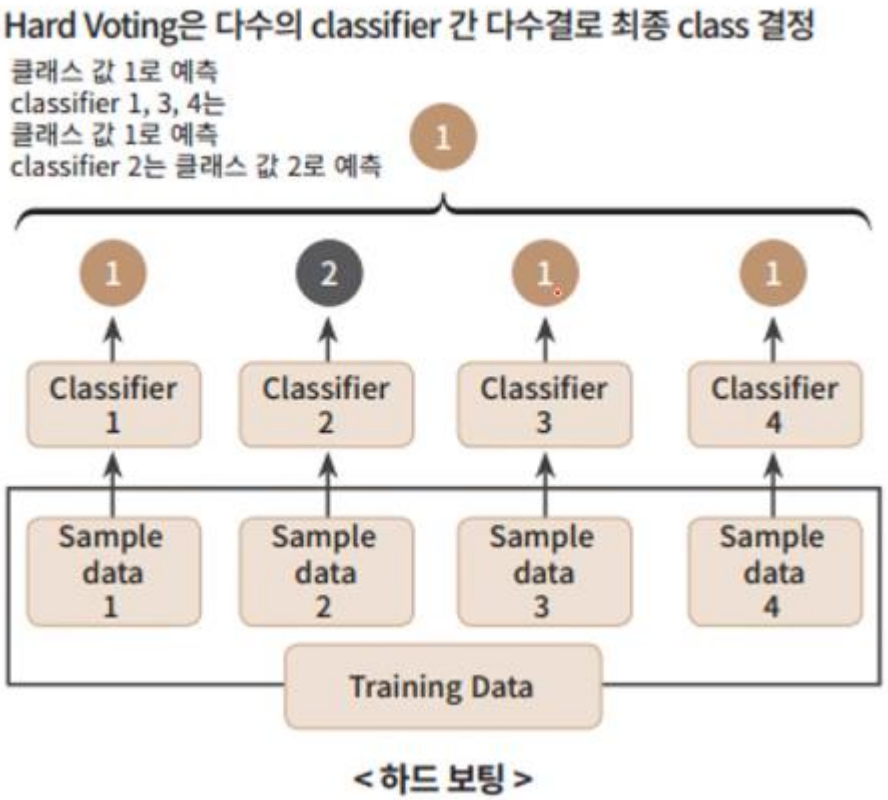


Voting 방식

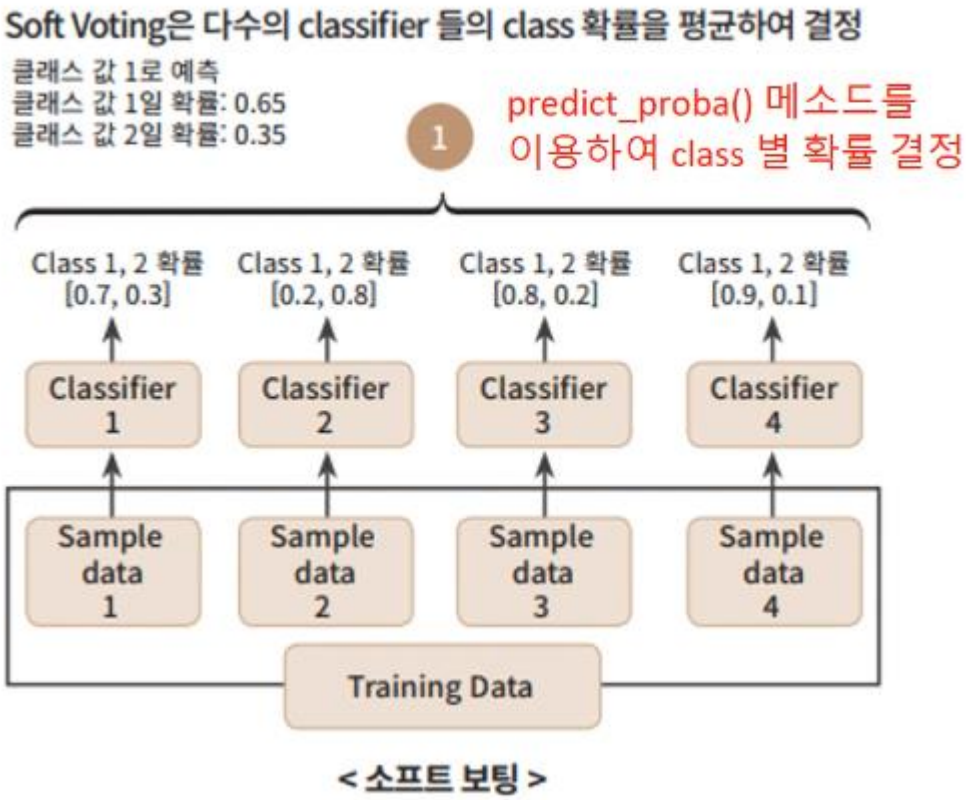
Bagging 방식

보팅과 배깅

First Second

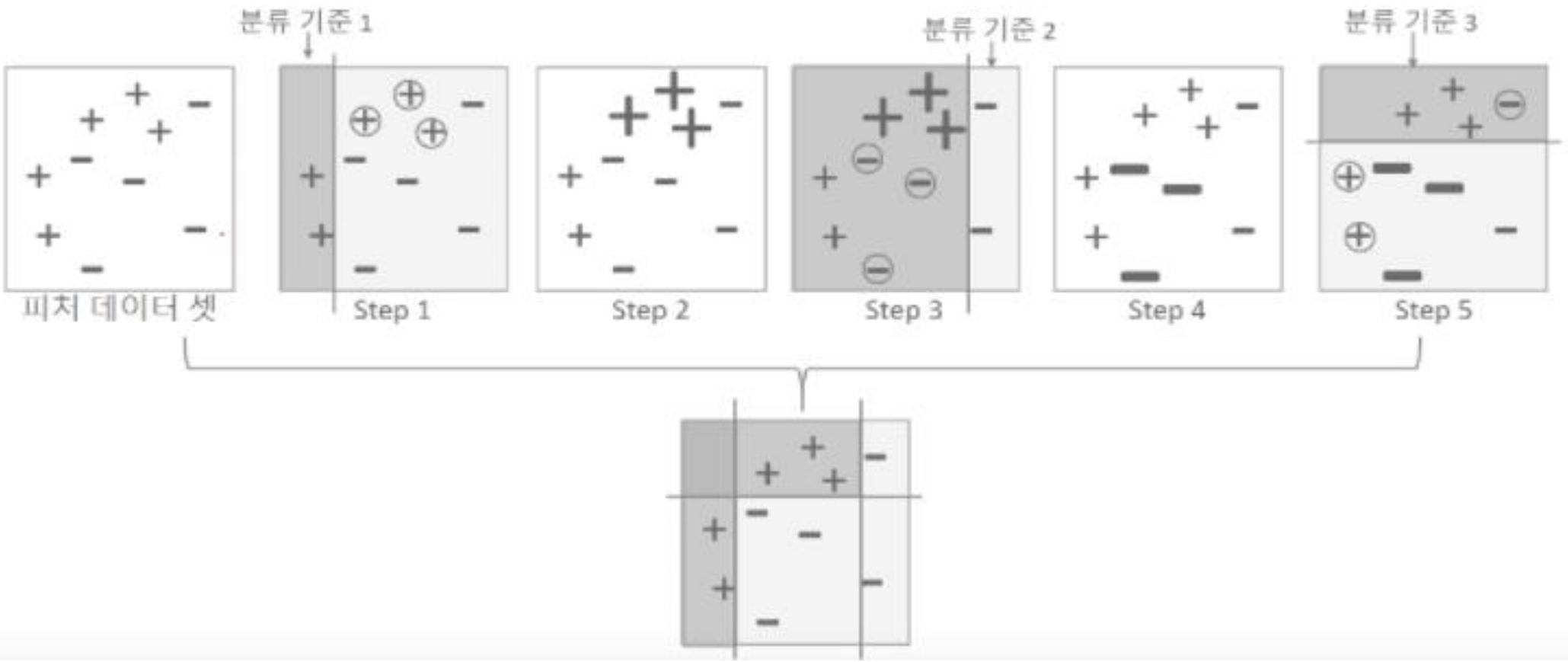


<하드 보팅>



<소프트 보팅>

부스팅



랜덤포레스트


서브세트 #1
1 2 3 3 3 5 6 8 8 9
부트스트래핑 분할
서브세트 #2
1 2 3 4 5 6 7 8 9 10
1 3 4 5 6 8 8 9 9 10
서브세트 #3
1 1 3 4 4 5 6 6 9 9

랜덤포레스트 알고리즘 구현

데이터 불러오기

일반적인 결정트리로 정확도 확인

랜덤포레스트 방식 정확도 확인

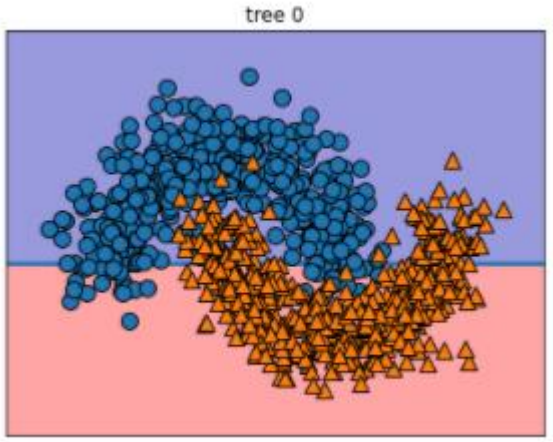
GridSearch로 하이퍼 파라미터 튜닝

최종 정확도 확인

데이터 불러오기

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
import mglearn
from sklearn.tree import DecisionTreeClassifier
```

```
X, y = make_moons(n_samples=1000, noise=0.2, random_state=3)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
 random_state=1)
```

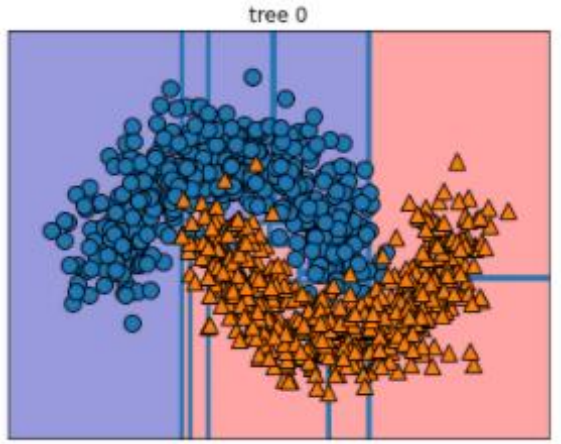


일반적인 결정트리로 정확도 확인

```
dt_clf = DecisionTreeClassifier(random_state=1)
dt_clf.fit(X_train, y_train)

pred = dt_clf.predict(X_test)
pred
```

```
array([1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 1], dtype=int64)
```




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

```
예측 정확도: 0.9600
```

랜덤포레스트 방식 정확도 확인

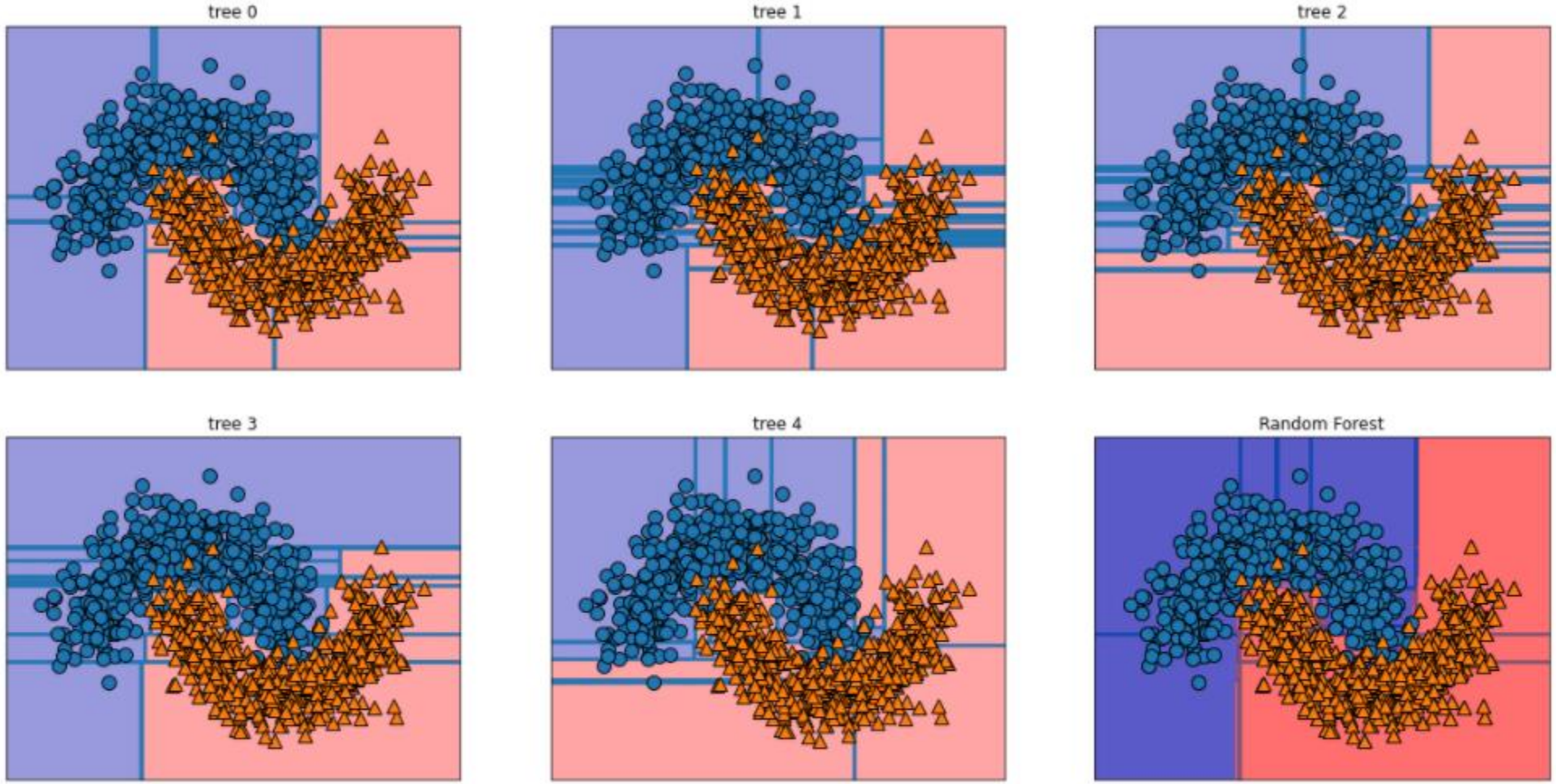
In [256]:
```
forest = RandomForestClassifier()
forest.fit(X_train, y_train)
```

Out[256]:
```
RandomForestClassifier()
```

In [257]:
```
fig, axes = plt.subplots(2, 3, figsize=(20, 10))
for i, (ax, tree) in enumerate(zip(axes.ravel(), forest.estimators_)):
    ax.set_title("tree {}".format(i))
    mglearn.plots.plot_tree_partition(X, y, tree, ax=ax)

mglearn.plots.plot_2d_separator(forest, X, fill=True, ax=axes[-1, -1], alpha=.4)
axes[-1, -1].set_title("Random Forest")
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
```

랜덤포레스트 방식 정확도 확인



랜덤포레스트 방식 정확도 확인

```
1 from sklearn.metrics import accuracy_score
2 print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test,pred)))
```

예측 정확도: 0.9600

일반 결정트리
-> RandomForest

```
1 print("테스트 세트 정확도: {:.4f}".format(forest.score(X_test, y_test)))
```

테스트 세트 정확도: 0.9760

GridSearch로 하이퍼 파라미터 튜닝

```
from sklearn.model_selection import GridSearchCV

params = { 'n_estimators' : [3,5,8,10,20,30,40,50],
           'max_depth' : [1,2,3,4,5,6,7,8],
           'min_samples_leaf' : [3,4,5,6,7,8],
           'min_samples_split' : [2,3,4,5,6,7,8]
            }

grid_cv = GridSearchCV(forest, param_grid = params, cv = 2, n_jobs = -1)
grid_cv.fit(X_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
```

최적 하이퍼 파라미터: {'max_depth': 7, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 5}

최종 정확도 확인

```python
from sklearn.metrics import accuracy_score
print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test,pred)))
```

예측 정확도: 0.9680

일반 결정트리
-> 랜덤포레스트

```python
print("테스트 세트 정확도: {:.4f}".format(forest.score(X_test, y_test)))
```

테스트 세트 정확도: 0.9760

하이퍼파라미터 최적화

```python
forest_2 = RandomForestClassifier(n_estimators = 5,
                                  max_depth = 7,
                                  min_samples_leaf = 4,
                                  min_samples_split = 5,
                                  )
forest_2.fit(X_train, y_train)
print("테스트 세트 정확도: {:.4f}".format(forest_2.score(X_test, y_test)))
```

테스트 세트 정확도: 0.9800

…

Thank you :-)