7.3

랜덤 패치/ 랜덤 서브스페이스

7.3, BaggingClassifier

특성을 대상으로 하는 두 개의 하이퍼파라미터

1 max_features

- 학습에 사용할 특성 수를 지정
- 특성 선택은 무작위.
  - 정수 : 지정된 수만큼 특성 선택
  - 부동소수점 : 지정된 비율만큼 특성 선택
- max_samples 와 유사 기능 수행.

2 bootstrap_features

- 학습에 사용할 특성을 선택할 때 중복 허용 여부 지정.
- 기본값은 False
- bootstrap 과 유사 기능 수행.

☑ 이미지 등과 같이 매우 높은 차원의 데이터셋을 다룰 때 유용

☑ 특성 샘플링은 더 다양한 예측기를 만들며, 일반적으로 편향이 커지지만 분산은 낮아짐.

WHY?
보다 적은 수의 특성을 사용하면 아무래도 예측이 부정확해질 수밖에 없는 반면에,
데이터 샘플에 대해 보다 덜 민감하게 반응하게 된다.

7.3, 랜덤 패치/랜덤 서브스페이스 기법

훈련 세트와 특성에 대한 샘플링 방식에 따라 나뉨.

랜덤 패치

훈련 샘플과 훈련 특성 모두를 대상으로 중복을 허용하며 임의의 샘플 수와 임의의 특성 수만큼을 샘플링 해서 학습하는 기법

- bootstrap=True / max_samples < 1.0

- bootstrap_features=True / max_features < 1.0

랜덤 서브스페이스

전체 훈련 세트를 학습 대상으로 삼지만 훈련 특성은 임의의 특성 수만큼 샘플링 해서 학습하는 기법

- bootstrap=False & max_samples=1.0

- bootstrap_features=True / max_features < 1.0

7.4

랜덤 포레스트

7.4, 랜덤 포레스트

배깅/페이스팅 방법을 적용한 의사결정나무의 앙상블을 최적화한 모델

RandomForestClassifier 를 이용하여 랜덤 포레스트 구현
(회귀용 예측기: RandomForestRegressor)

```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(
    n_estimators=500,
    max_leaf_nodes=16,
    n_jobs=-1,
    random_state=42
)

rnd_clf.fit(X_train, y_train)
y_pred_rf = rnd_clf.predict(X_test)
```

- n_estimators=500: 500개의 의사결정나무 학습
- max_leaf_nodes=16: 사용되는 의사결정나무의 잎의 수를 16개로 제한
- n_jobs=-1: 가용 가능한 모든 CPU 사용

7.4, 랜덤 포레스트와 앙상블 학습

DecisionTreeClassifier 를 BaggingClassifier 와 함께 사용한 앙상블 모델

```
bag_clf_auto = BaggingClassifier(
    DecisionTreeClassifier(max_features="auto", max_leaf_nodes=16, rando
m_state=42),
    n_estimators=500,
    max_samples=1.0,
    bootstrap=True,
    n_jobs=-1,
    random_state=42
)

bag_clf_auto.fit(X_train, y_train)
y_pred_auto = bag_clf_auto.predict(X_test)
```

- max_leaf_nodes=16: 잎의 수를 16개로 제한함.
- max_samples=1.0
- max_features="auto": 학습에 사용되는 특성의 수를 전체 특성 수의 제곱근 값으로 제한한다. 즉, max_features=sqrt(n_features) 와 동일하게 작동함

랜덤 포레스트 모델과 배깅 모델이 동일한 예측을 한다❗

7.4, 엑스트라트리

극단적으로 무작위한 트리의 랜덤 포레스트

편향은 늘고, 분산은 줄어든다❗

```
bag_clf_randAuto = BaggingClassifier(
    DecisionTreeClassifier(splitter="random",
                           max_features="auto",
                           max_leaf_nodes=16,
                           random_state=42),
    n_estimators=500,
    max_samples=1.0,
    bootstrap=True,
    n_jobs=-1,
    random_state=42
)

bag_clf_randAuto.fit(X_train, y_train)
y_pred_randAuto = bag_clf_randAuto.predict(X_test)
```

7.4, 엑스트라트리

극단적으로 무작위한 트리의 랜덤 포레스트

```
from sklearn.ensemble import ExtraTreesClassifier

extra_clf = ExtraTreesClassifier(
    n_estimators=500,
    max_leaf_nodes=16,
    n_jobs=-1,
    random_state=42
)

extra_clf.fit(X_train, y_train)
y_pred_extra = extra_clf.predict(X_test)
```

RandomForestClassifier와 ExtraTreesClassifier 중에 누가 더 좋을지는 미리 알 수 없으며 교차 검증으로 확인하는 수 밖에 없다. 또한 그리드 탐색을 이용하여 하이퍼파라미터를 튜닝할 수도 있다.

7.4, 특성 중요도 - 붓꽃 데이터셋

해당 특성을 사용한 마디가 평균적으로 불순도를 얼마나 감소시키는지를 측정

불순도를 많이 줄이면 그만큼 중요도가 커진다❗

➰ RandomForestClassifier는 훈련이 끝난 뒤 특성별 중요도의 전체 합이 1이 되도록 하는 방식으로 특성별 상대적 중요도를 측정한 후 feature_importances_ 속성에 저장한다.

```
from sklearn.datasets import load_iris

iris = load_iris()
rnd_clf = RandomForestClassifier(n_estimators=500, random_state=42)
rnd_clf.fit(iris["data"], iris["target"])

for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
  print(name, score)
```

- 꽃받침 길이(sepal length) (cm) 0.11249225099876375
- 꽃받침 너비(sepal width) (cm) 0.02311928828251033
- 꽃잎 길이(petal length) (cm) 0.4410304643639577
- 꽃잎 너비(petal width) (cm) 0.4233579963547682

7.4, 특성 중요도 - MNIST 손글씨 데이터셋

해당 특성을 사용한 마디가 평균적으로 불순도를 얼마나 감소시키는지를 측정

```
from sklearn.datasets import fetch_openml

mnist = fetch_openml('mnist_784', version=1)
mnist.target = mnist.target.astype(np.uint8)

rnd_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rnd_clf.fit(mnist["data"], mnist["target"])

def plot_digit(data):
  image = data.reshape(28, 28)
  plt.imshow(image, cmap=mpl.cm.hot, interpolation="nearest")
  plt.axis("off")

plot_digit(rnd_clf.feature_importances_)

cbar = plt.colorbar(ticks=[rnd_clf.feature_importances_.min(), rnd_clf.feature_importances_.max()])
cbar.ax.set_yticklabels(['Not important', 'Very important'])

save_fig("mnist_feature_importance_plot")
plt.show()
```

7.4, 특성 중요도 – MNIST 손글씨 데이터셋

해당 특성을 사용한 마디가 평균적으로 불순도를 얼마나 감소시키는지를 측정