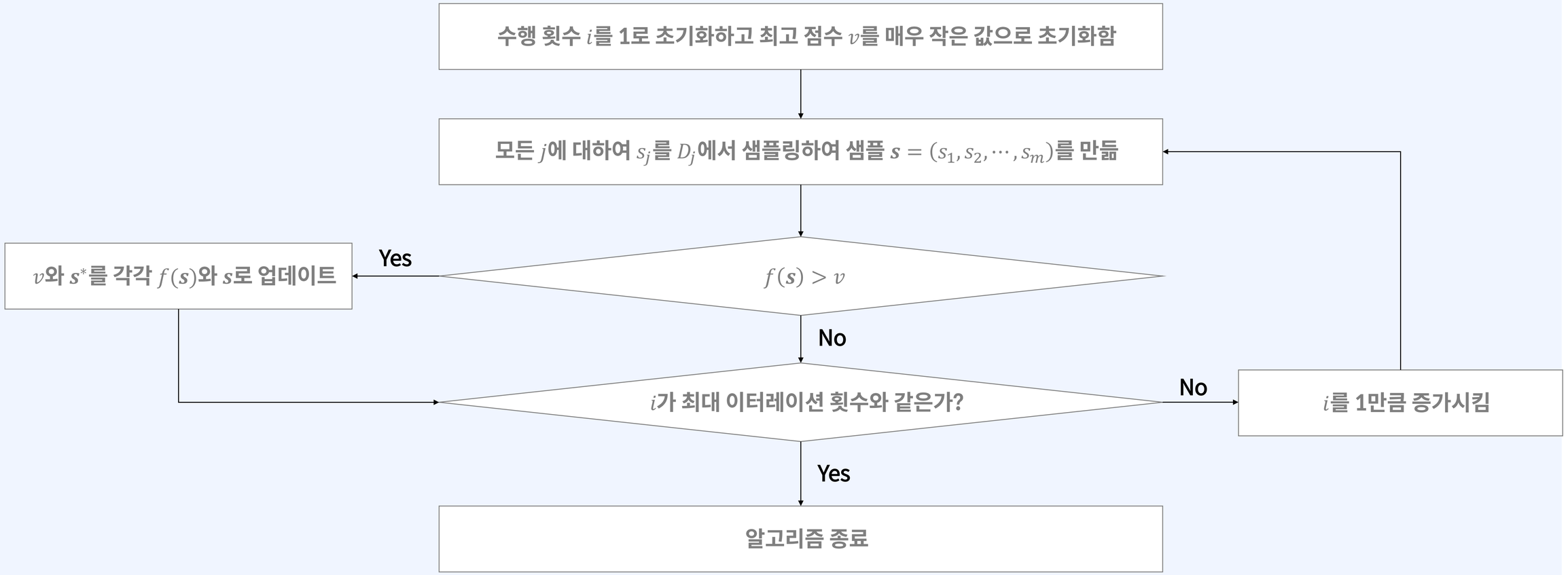
2. 랜덤 서치

1 랜덤 서치 개요

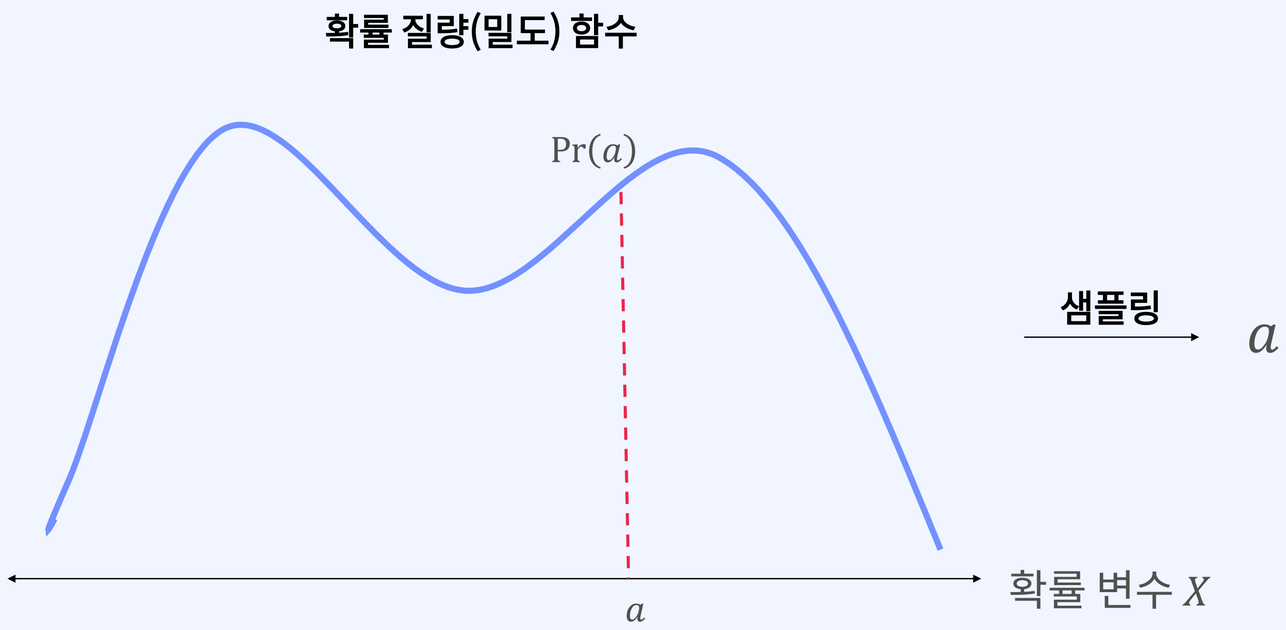
랜덤 서치란?

랜덤 서치는 이름 그대로 탐색 공간에 있는 해를 임의로 선택해서 탐색하는 방법입니다. 어느 문제에나 손쉽게 적용할 수 있다는 장점 덕분에, 하이퍼파라미터 튜닝뿐만 아니라 많은 최적화 문제를 해결하는 데 자주 사용합니다.



확률적 샘플링

확률적 샘플링이란 특정한 확률 분포를 따르는 확률 변수로부터 값을 추출하는 것을 의미합니다.



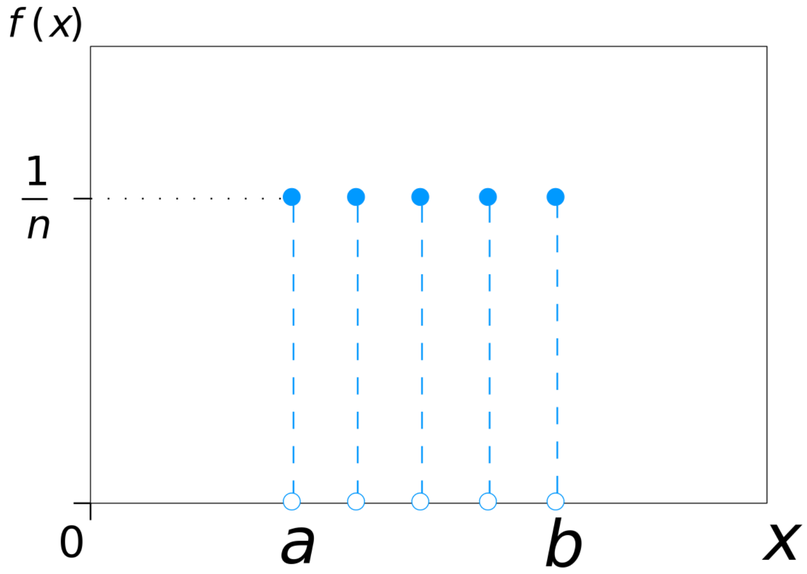
이산형 확률 분포

(1) 유니폼 분포

이산 유니폼 분포란 주사위를 던졌을 때 윗면에 나오는 숫자의 분포처럼 상태 공간의 크기가 유한하고 상태 공간의 각 값의 출현 확률이 같은 분포를 의미합니다.

확률 질량 함수: $\Pr(X = v_i) = \frac{1}{n}$

- 상태 공간: $\mathrm{V} = \{v_1, v_2, \cdots, v_n\}$

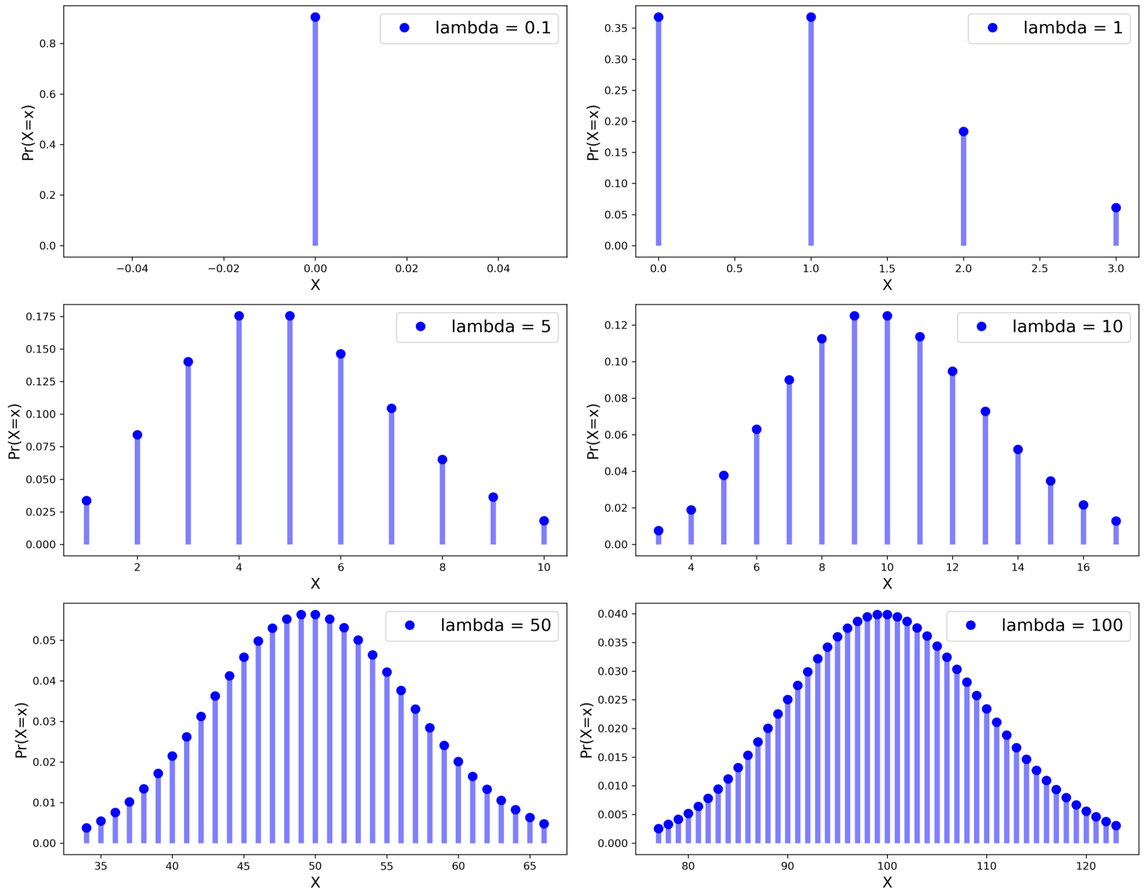


이산형 확률 분포
(2) 푸아송 분포

푸아송 분포(Poisson distribution)는 단위 시간 안에 어떤 사건이 몇 번 발생할지를 나타내는 이산 확률 분포로, 범주형 변수가 특정 구간의 값을 주로 갖는 상황에서 자주 사용합니다.

확률 질량 함수: $\Pr(x = n; \lambda) = \frac{\lambda^n \exp(-\lambda)}{n!}$

- λ: 단위 시간 안에 어떤 사건의 발생 횟수에 대한 기댓값
- λ가 클수록 확률 변수가 큰 값을 취할 가능성이 커짐

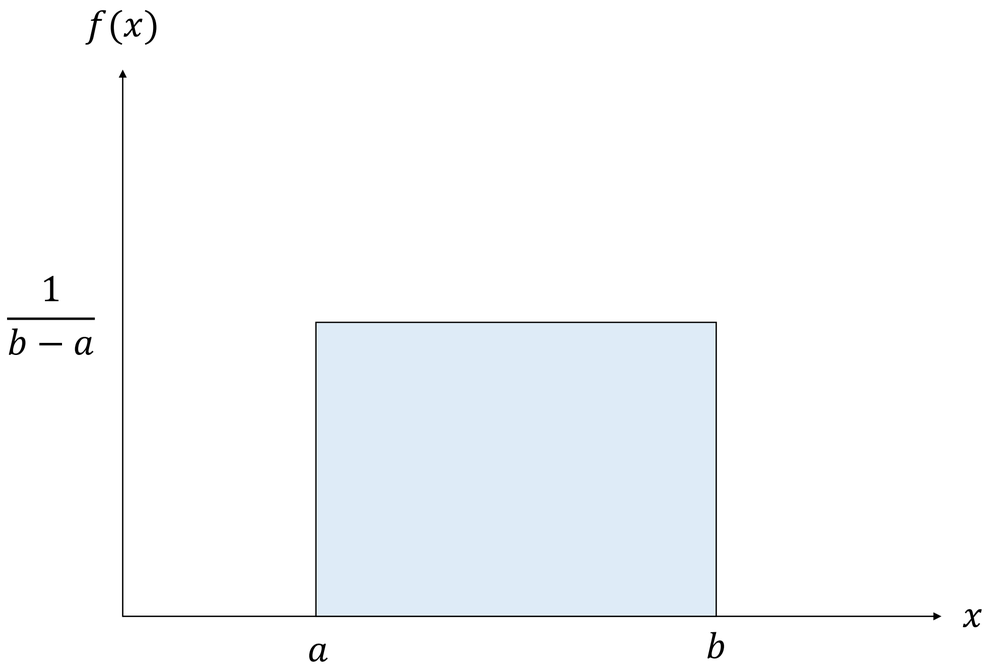


연속형 확률 분포
(1) 유니폼 분포

연속 유니폼 분포란 주사위를 던졌을 때 특정 구간에 속하는 값의 출현 확률이 같은 분포를 의미합니다.

확률 밀도 함수: $f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$

- a: 분포의 하한
- b: 분포의 상한

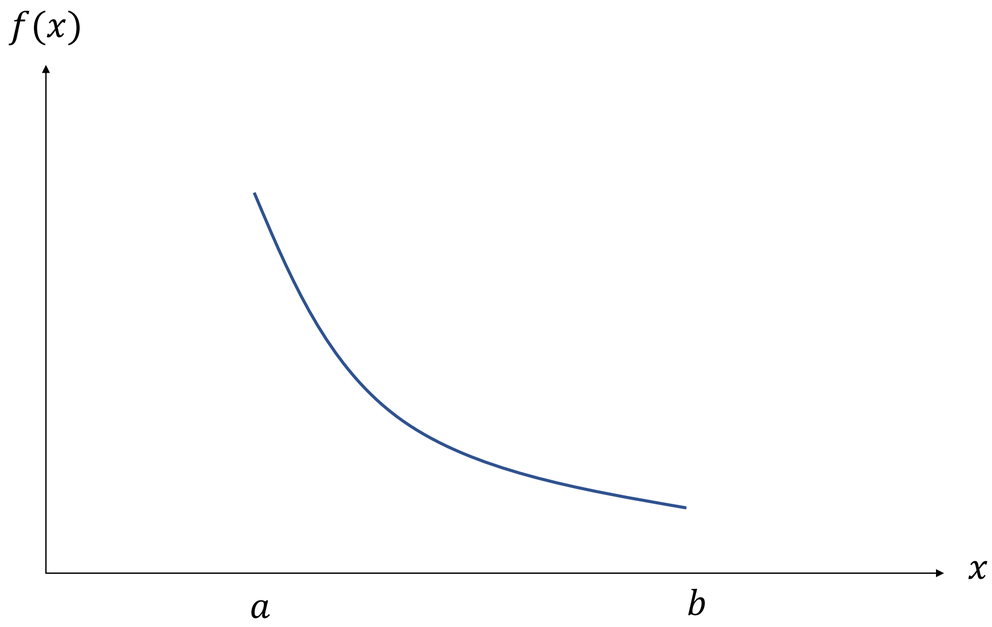


연속형 확률 분포
(2) 로그 유니폼 분포

로그 유니폼 분포는 계수 페널티의 가중치 처럼 범위가 매우 넓고 특정 값보다 크면 값 간 차이가 무의미한 변수에 대해 가정하기에 적절한 분포입니다.

확률 밀도 함수: $f(x) = \begin{cases} \frac{1}{x \ln b/a}, & \text{if } a \leq x \leq b, a \geq 0 \\ 0, & \text{otherwise} \end{cases}$

- a: 분포의 하한
- b: 분포의 상한



샘플링 함수
(1) np.choice

넘파이의 random.choice 함수를 사용하면 배열에서 하나의 값을 임의로 선택할 수 있으며, 이는 이산 유니폼 분포로부터 샘플링하는 것과 같습니다.

np.choice 예제

```
1  import numpy as np
2  X = [1, 2, 3, 4, 5]
3  s = np.random.choice(X)
4  print(s)
```

2 • 임의로 선택한 것이므로 실행 때마다 결과가 달라질 수 있음

샘플링 함수
(2) scipy.stats 모듈

scipy.stats 모듈에 속한 클래스를 사용하면 특정한 확률 분포를 따르는 확률 변수 인스턴스를 생성할 수 있습니다.

주요 클래스

구분	분포	클래스
범주형	베르누이 분포	bernoulli
	이항 분포	binom
	푸아송 분포	poisson
연속형	지수 분포	expon
	유니폼 분포	uniform
	로그 유니폼 분포	loguniform
	정규 분포	normal

이들 클래스는 rvs라는 메서드를 이용해 샘플링을 할 수 있으며, 이 메서드는 생성할 샘플 수를 입력으로 받습니다.

샘플링 함수
(2) scipy.stats 모듈 (계속)

scipy.stats 모듈에 속한 클래스를 사용하면 특정한 확률 분포를 따르는 확률 변수 인스턴스를 생성할 수 있습니다.

푸아송 분포 샘플링 예제

```
1  from scipy.stats import poisson
2  poisson_rv = poisson(10) # 파라미터를 10으로 설정
3  X = poisson_rv.rvs(5) # 샘플 5개 샘플링
4  print(X)
```

```
[8  5  11  11  14]
```

유니폼 및 로그 유니폼 분포 샘플링 예제

```
1  from scipy.stats import uniform, loguniform
2  uni_rv = uniform(10, 10) # 하한 (첫 인자) = 10, 상한 (첫 인자 + 두 번째 인자) = 10 + 10
3  log_uni_rv = loguniform(10, 10000) # 하한 = 10, 상한 = 10000
4  X1 = uni_rv.rvs(10)
5  X2 = log_uni_rv.rvs(10)
6
7  print(X1)
8  print(X2)
```

```
[19.15535225 18.19430921 12.92362357 12.29357986 13.80995471 16.35638269
 17.48338166 20.90384396 11.13755342 17.0021346 ]

[1065.18632007 1278.81581239   18.30310808   46.27101763 1110.97646689
  260.63172014   38.55509436 2752.13811504   21.36694434   73.83282293]
```

2. 랜덤 서치

2 RandomSearchCV를 이용한 랜덤 서치

예제 데이터 불러오기

랜덤 서치를 실습할 데이터를 불러옵니다. 이때, k-겹 교차 검증을 사용해 해를 평가할 예정이므로 train_test_split으로 학습 데이터와 평가 데이터로 분리하지는 않았습니다.

예제 데이터 불러오기

```
1  import pandas as pd
2  df = pd.read_csv("../../data/classification/movement_libras.csv")
3  X = df.drop('y', axis = 1)
4  y = df['y']
```

RandomSearchCV 클래스

RandomSearchCV 클래스는 랜덤 서치 방식으로 하이퍼파라미터를 튜닝합니다.

주요 인자

인자	설명
estimator	분류 및 회귀 모델 인스턴스
param_distribution	하이퍼파라미터의 분포 (자료형: 사전)
cv	폴드 개수
scoring	평가 척도
n_iter	평가 횟수
refit	가장 좋은 성능의 하이퍼파라미터를 갖는 모델을 전체 데이터로 재학습할지 여부

- param_distribution의 키는 estimator의 인자이고 값은 해당 인자가 따르는 확률 분포임

(예시)

```
dist = {"max_features":loguniform(0.5, 1),
        "max_depth": range(3, 8),
        "criterion":["gini", "entropy"]}
```

랜덤 서치

2.
RandomSearchCV
를 이용한 랜덤 서치

RandomSearchCV를 이용해 랜덤 포레스트의 하이퍼파라미터를 튜닝해보겠습니다.

RandomSearchCV를 이용한 하이퍼파라미터 튜닝 예제

```
1  from sklearn.ensemble import RandomForestClassifier as RFC
2  from sklearn.model_selection import RandomizedSearchCV
3  clf = RandomizedSearchCV(RFC(random_state = 2022),
4                           dist,
5                           n_iter = 10,
6                           random_state=2022).fit(X, y)
7  result = pd.DataFrame(clf.cv_results_)
8  display(result[['params', 'mean_test_score', 'mean_fit_time']])
```

	params	mean_test_score	mean_fit_time
0	{'criterion': 'entropy', 'max_depth': 7, 'max_...	0.675000	4.630423
1	{'criterion': 'entropy', 'max_depth': 4, 'max_...	0.569444	2.971182
2	{'criterion': 'gini', 'max_depth': 5, 'max_fea...	0.586111	1.276401
3	{'criterion': 'entropy', 'max_depth': 4, 'max_...	0.575000	4.135532
4	{'criterion': 'entropy', 'max_depth': 6, 'max_...	0.663889	4.743011
5	{'criterion': 'gini', 'max_depth': 6, 'max_fea...	0.625000	1.444848
6	{'criterion': 'gini', 'max_depth': 3, 'max_fea...	0.519444	0.864892
7	{'criterion': 'gini', 'max_depth': 5, 'max_fea...	0.586111	1.218038
8	{'criterion': 'gini', 'max_depth': 4, 'max_fea...	0.563889	1.134817
9	{'criterion': 'entropy', 'max_depth': 4, 'max_...	0.583333	4.431382

랜덤 서치 (계속)

RandomSearchCV를 이용해 랜덤 포레스트의 하이퍼파라미터를 튜닝해보겠습니다.

RandomSearchCV를 이용한 하이퍼파라미터 튜닝 예제

```
1  print(clf.best_estimator_)
2  print(clf.best_score_)
3  print(clf.best_params_)
```

```
RandomForestClassifier(criterion='entropy', max_depth=7, max_features=0.7066451376545405, random_state=2022)
0.675
{'criterion': 'entropy', 'max_depth': 7, 'max_features': 0.7066451376545405}
```

2. 랜덤 서치

3 랜덤 서치 직접 구현하기

랜덤 서치 직접 구현하기

랜덤 서치를 이용한 하이퍼파라미터 튜닝을 직접 구현해보겠습니다. 그리드 서치와 마찬가지로 직접 구현하는 것이 실무에서 사용하기에 적절합니다.

랜덤 서치 직접 구현하기 예제

```
1  from sklearn.metrics import *
2  from sklearn.model_selection import KFold
3  kf = KFold(n_splits = 5)
4  best_score = -1
5  num_iter = 10
```

랜덤 서치 직접 구현하기 (계속)

랜덤 서치를 이용한 하이퍼파라미터 튜닝을 직접 구현해보겠습니다. 그리드 서치와 마찬가지로 직접 구현하는 것이 실무에서 사용하기에 적절합니다.

랜덤 서치 직접 구현하기 예제

```
6   for _ in range(num_iter):
7       total_score = 0
8       for train_index, test_index in kf.split(X):
9           X_train = X.loc[train_index]
10          X_test = X.loc[test_index]
11          y_train = y.loc[train_index]
12          y_test = y.loc[test_index]
13          _max_features = loguniform(0.5, 1).rvs(1)[0]
14          _max_depth = np.random.choice(range(3, 8))
15          _criterion = np.random.choice(["gini", "entropy"])
16
17          model = RFC(max_features = _max_features,
18                      max_depth = _max_depth,
19                      criterion = _criterion).fit(X_train, y_train)
20          y_pred = model.predict(X_test)
21          score = accuracy_score(y_test, y_pred)
22          total_score += score / 5
23
24      if total_score > best_score:
25          best_score = total_score
26          best_parameter = [_max_features, _max_depth, _criterion]
```

- **라인 5~6:** 사용자가 지정한 num_iter만큼 샘플링과 평가를 반복합니다.
- **라인 8~12:** KFold를 이용해 k-겹 교차 검증을 수행합니다.
- **라인 13~15:** max_features, max_depth, criterion을 주어진 분포 내에서 샘플링한 결과를 각각 _max_features, _max_depth, _criterion에 저장합니다. max_features는 유니폼 분포를 따르는 확률 변수로부터 하나의 값을 샘플링했습니다. 이때 rvs(1)을 하더라도 배열로 반환되므로 [0]을 이용해 값만 가져왔습니다.
- **라인 17~19:** 샘플링한 값 _max_features, _max_depth, _criterion을 하이퍼파라미터로 하는 모델을 학습합니다.
- **라인 24~26:** 현재 탐색하는 하이퍼파라미터를 사용했을 때의 평가 점수가 지금까지의 최고 점수보다 크다면 best_score와 best_parameter를 업데이트합니다.

랜덤 서치 직접 구현하기 (계속)

랜덤 서치를 이용한 하이퍼파라미터 튜닝을 직접 구현해보겠습니다. 그리드 서치와 마찬가지로 직접 구현하는 것이 실무에서 사용하기에 적절합니다.

랜덤 서치 직접 구현하기 예제

```
1    print(best_parameter, best_score)
```

```
[0.6994322394713137, 7, 'gini']    0.75
```