

6th Meet

□ 오늘 공부할 것은

04	07.LightGBM	244
	LightGBM 설치	246
	LightGBM 하이퍼 파라미터	247
분류	하이퍼 파라미터 튜닝 방안	248
	파이썬 래퍼 LightGBM과 사이킷런 래퍼 XGBoost,	
	LightGBM 하이퍼 파라미터 비교	249
	LightGBM 적용 – 위스콘신 유방암 예측	250
	08. 베이지안 최적화 기반의 HyperOpt를 이용한	
	하이퍼 파라미터 튜닝	253
	베이지안 최적화 개요	254
	HyperOpt 사용하기	256
	HyperOpt를 이용한 XGBoost 하이퍼 파라미터 최적화	262
	09. 분류 실습 – 캐글 산탄데르 고객 만족 예측	267
	데이터 전처리	268
	XGBoost 모델 학습과 하이퍼 파라미터 튜닝	271
	LightGBM 모델 학습과 하이퍼 파라미터 튜닝	276
	10. 분류 실습 – 캐글 신용카드 사기 검출	279
	언더 샘플링과 오버 샘플링의 이해	279
	데이터 일차 가공 및 모델 학습/예측/평가	281
	데이터 분포도 변환 후 모델 학습/예측/평가	285
	이상치 데이터 제거 후 모델 학습/예측/평가	288
	SMOTE 오버 샘플링 적용 후 모델 학습/예측/평가	292

Now, It's your turn !!!

40min ~ 50min

Chapter 04

분류 (Classification)

[04-07]
LightGBM

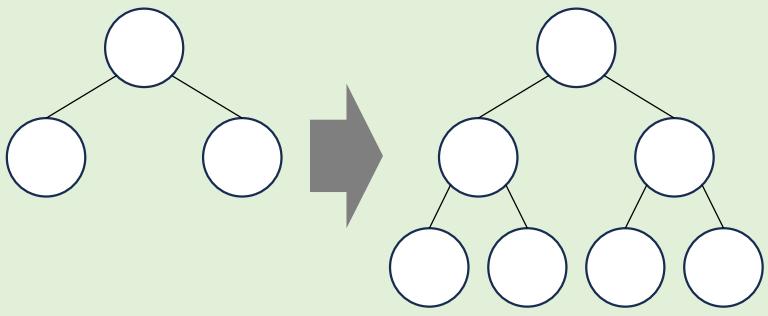
□ LightGBM

- XGBoost와 성능은 유사하지만, 보다 더 빠르고 메모리도 적게 사용
- 카테고리형 피처의 자동 변환 지원
- 하지만, 보통 1만 건 이하의 dataset에서는 과적합 가능성 높음

구분	XGBoost	LightGBM
Initial Release	2014	2016
학습 소요 시간	느림	빠름
메모리 사용량	큼	작음
예측 성능	차이 없음	
Tree Split method	Level Wise	Leaf Wise

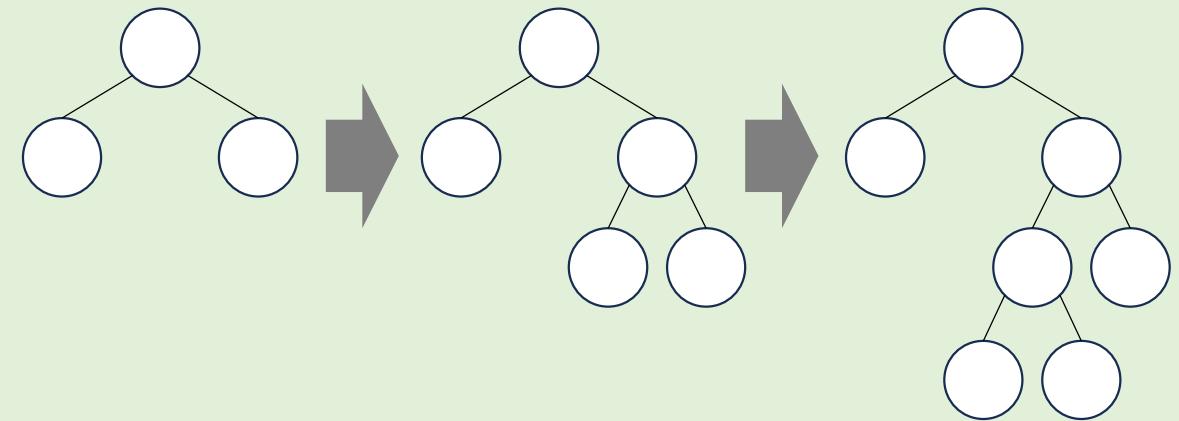
XGBoost

균형 트리 분할 (Level Wise/Depth Wise)



LightGBM

리프 중심 트리 분할 (Leaf Wise)



minimum depth → overfitting에 더 강한 구조

But 균형을 맞추기 위해 시간 소요

'max delta loss'를 갖는 leaf node를 계속 분할

→ deeper depth, more imbalanced tree

→ 학습 반복할수록 균형 트리 방식보다 예측 오류 손실 최소화

□ LightGBM 하이퍼 파라미터 튜닝 방안

- num_leaves 개수를 중심으로 min_child_sample(min_data_in_leaf), max_depth 조정



<https://github.com/microsoft/LightGBM/issues/6244>

A screenshot of a GitHub issue page for the LightGBM repository. The URL is <https://github.com/microsoft/LightGBM/issues/6244>. The issue is titled "Early Stopping does not work in LGBMClassifier #6244". It was opened by ohbtorres on Dec 21, 2023, with 5 comments. A comment from ohbtorres on Dec 21, 2023, describes a bug: "Using `lightgbm.early_stopping` does not work while using Scikit-learn API, and the training still remains after the early stop condition occurs".

<https://github.com/microsoft/LightGBM/pull/4908>

A screenshot of a GitHub pull request page for the LightGBM repository. The URL is <https://github.com/microsoft/LightGBM/pull/4908>. The pull request is titled "[python] remove `early_stopping_rounds` argument of `train()` and `cv()` functions #4908". It was merged by StrikerRUS on Dec 26, 2021, from the `remove_early_stop` branch into the `master` branch. The pull request has 4 conversations, 1 commit, 0 checks, and 6 files changed. A comment from StrikerRUS on Dec 24, 2021, refers to issue #4574.

<https://github.com/microsoft/LightGBM/pull/4832>

A screenshot of a GitHub pull request page for the LightGBM repository. The URL is <https://github.com/microsoft/LightGBM/pull/4832>. The pull request is titled "[python][sklearn] remove `verbose` argument from `fit()` method #4832". It was merged by StrikerRUS on Dec 1, 2021, from the `remove_verbose` branch into the `master` branch. The pull request has 2 conversations, 1 commit, 0 checks, and 5 files changed. A comment from StrikerRUS on Nov 29, 2021, refers to issue #4574.

https://github.com/whatwant-school/python-ml/blob/main/06-week/06-week_01-LightGBM.ipynb

[04-08]

비이지안 최적화 기반의 HyperOpt를 이용한
하이퍼 파라미터 튜닝!

□ Bayesian Rule

- 사전 확률과 사후 확률의 관계를 설명하는 법칙
- . 빈도주의 (Frequentism) : 100번 동전을 던졌을 때, 50번은 앞면이 나온다.
- . 베이지안주의 (Bayesianism) : 동전이 앞면이 나왔다는 주장의 신뢰도가 50%다.
'확률'을 '주장에 대한 신뢰도'로 해석

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

우도 확률 (likelihood) → $P(B|A)$

사전 확률 (prior) → $P(A)$

사후 확률 (posterior) → $P(A|B)$

주변 우도 (evidence, marginal likelihood) → $P(B)$

□ Bayesian Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

우도 확률 (likelihood) → $P(B|A)$

사전 확률 (prior) → $P(A)$

사후 확률 (posterior) → $P(A|B)$

주변 우도 (evidence, marginal likelihood) → $P(B)$

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

기존 신뢰도 기반으로 새로운 정보가 나올 확률 → $P(E|H)$

어떤 사건이 발생했다는 주장에 대한 신뢰도 → $P(H)$

새로운 정보를 받은 후 갱신된 신뢰도 → $P(H|E)$

E (Evidence) : 새로운 정보

H (Hypothesis) : 어떤 사건이 발생했다는 주장

새로운 정보가 나올 확률 → $P(E)$

□ Bayesian Rule

- 질병 A의 빈병률은 0.1%로 알려져 있다.
 - . 이 질병이 실제로 있을 때 질병이 있다고 검진할 확률(민감도)은 99%
 - . 실제로 질병이 없을 때 질병이 없다고 검진할 확률(특이도)은 98%라고 하자.
- 만약 어떤 사람이 질병에 걸렸다고 검진받았을 때, 이 사람이 정말로 질병에 걸렸을 확률은?

H : 실제로 병이 있다 (True)

E : 병이 있다고 진단 (Positive)

$$P(H) = 0.001$$

$$P(E|H) = 0.99$$

$$P(E^c|H^c) = 0.98$$

$$P(H|E) = \frac{P(E|H) P(H)}{P(E)}$$

$$= \frac{P(E|H) P(H)}{P(E|H)P(H) + P(E|H^c)P(H^c)}$$

$$= \frac{0.001 \times 0.99}{0.001 \times 0.99 + 0.999 \times 0.02} = 0.047..$$

	False Negative	True Negative	특이도 (specificity) $P(E^c H^c) = 0.98$
민감도 (sensitivity) $P(E H) = 0.99$	True Positive		
빈병률 (사전확률) $P(H) = 0.001$	False Positive		$P(E H^c) = 0.02$



$$P(E) = P(E|H)P(H) + P(E|H^c)P(H^c)$$

<https://sigopt.com/blog/bayesian-optimization-101/>

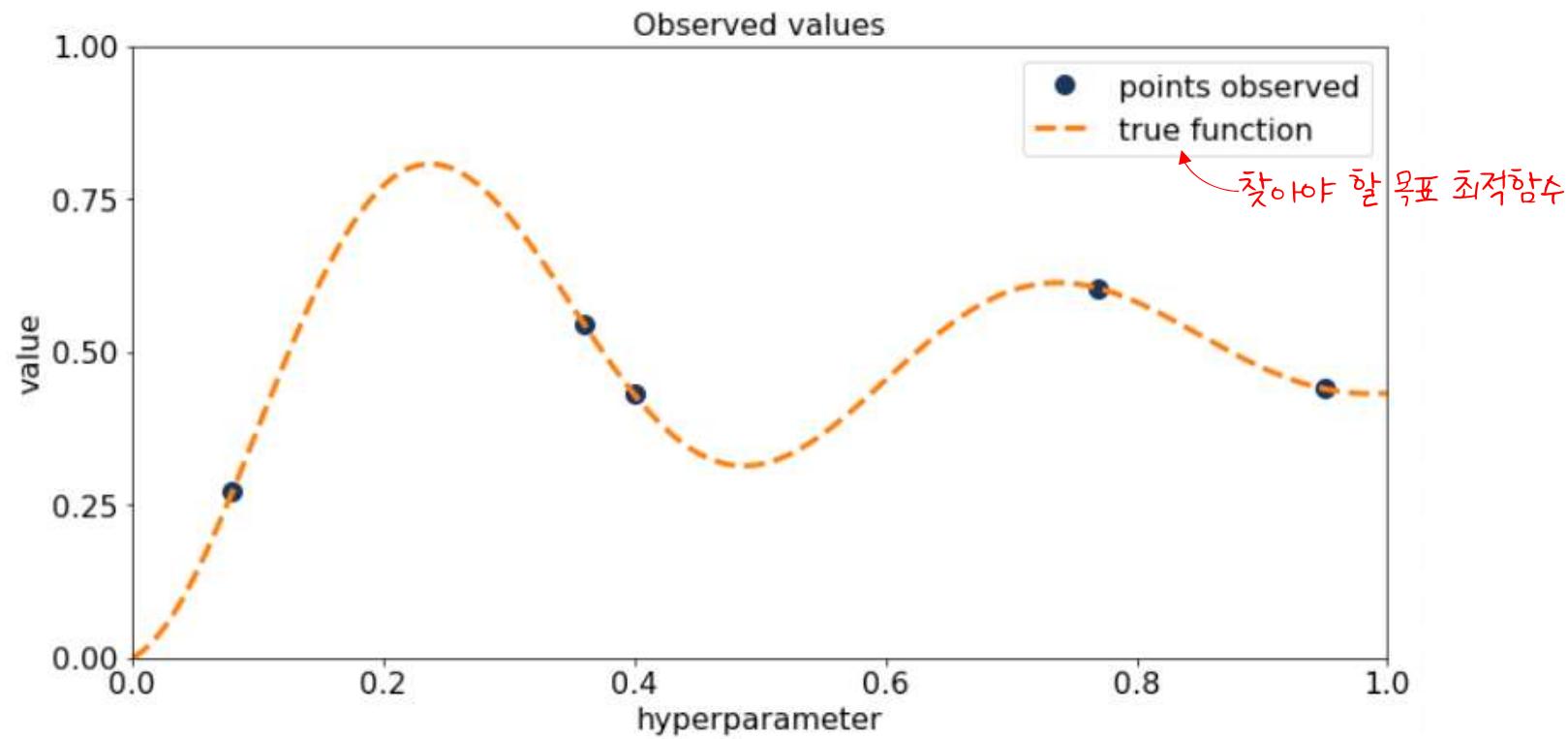
□ Bayesian Optimization # 1/4

Step 1: Sample the parameter space

최초에는 랜덤하게 하이퍼 파라미터 샘플링하고 성능 결과 관측

$$P(H|E) = \frac{P(E|H) P(H)}{P(E)}$$

우도 확률
사전 확률
사후 확률
주변 우도



<https://sigopt.com/blog/bayesian-optimization-101/>

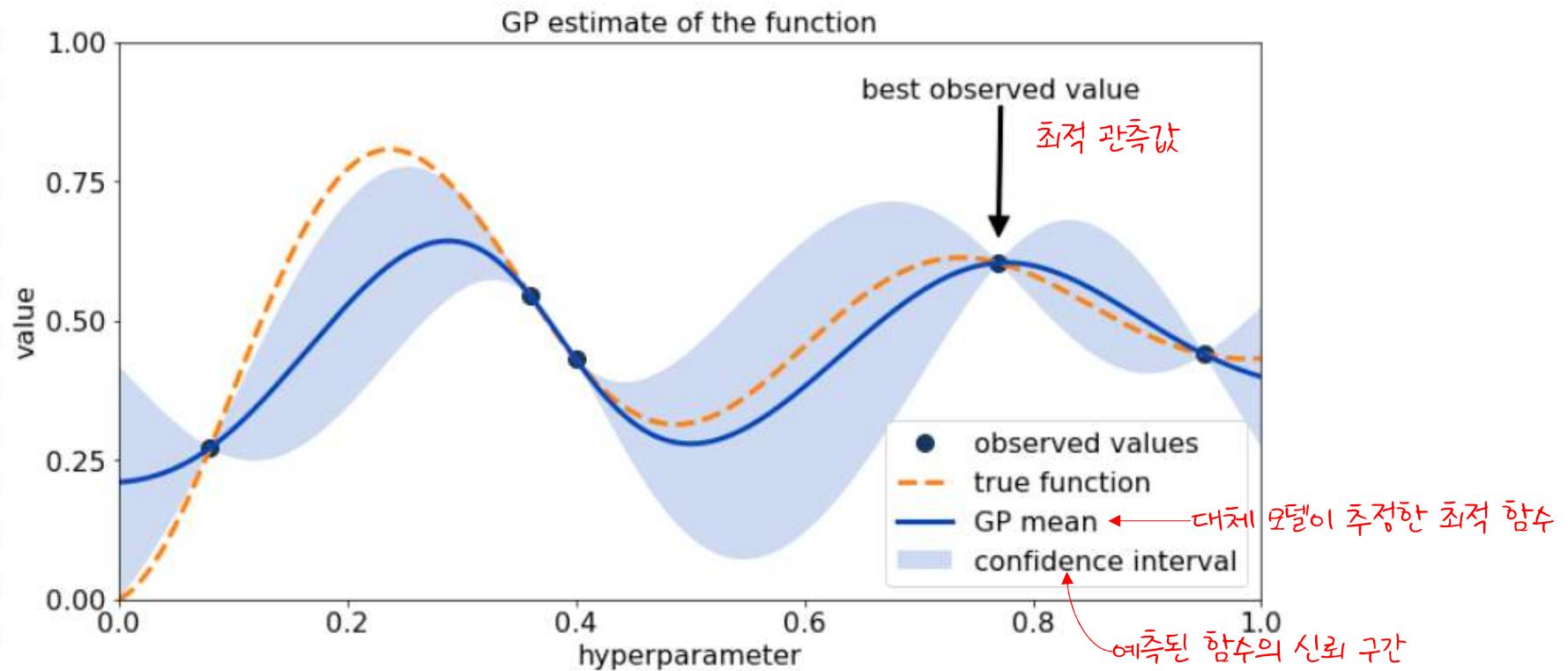
□ Bayesian Optimization # 2/4

Step 2: Build a surrogate model

관측된 값 기반으로 대체 모델은 최적 함수를 추정

$$P(H|E) = \frac{P(E|H) P(H)}{P(E)}$$

우도 확률
사전 확률
사후 확률
주변 우도



<https://sigopt.com/blog/bayesian-optimization-101/>

□ Bayesian Optimization # 3/4

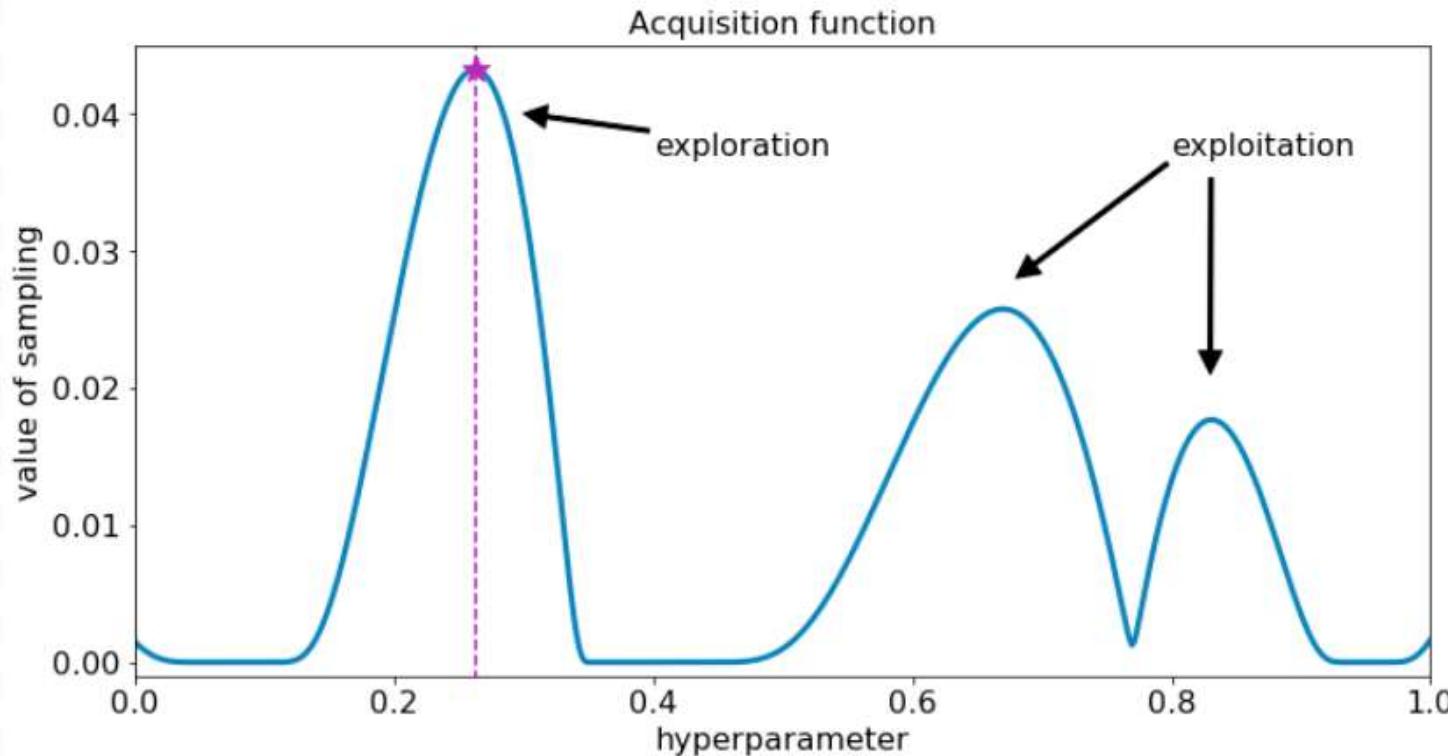
Step 3: Figure out where to sample next

추정된 최적 함수 기반으로 획득 함수(Acquisition Function)는 다음으로 관측할 하이퍼 파라미터 값을 계산

획득 함수는 이전의 최적 관측값 보다 더 큰 최댓값을 가질 가능성이 높은 지점을 찾아서 다음에 관측할 하이퍼 파라미터를 대체 모델에 전달

$$P(H|E) = \frac{P(E|H) P(H)}{P(E)}$$

사후 확률
우도 확률
사전 확률
주변 우도



<https://sigopt.com/blog/bayesian-optimization-101/>

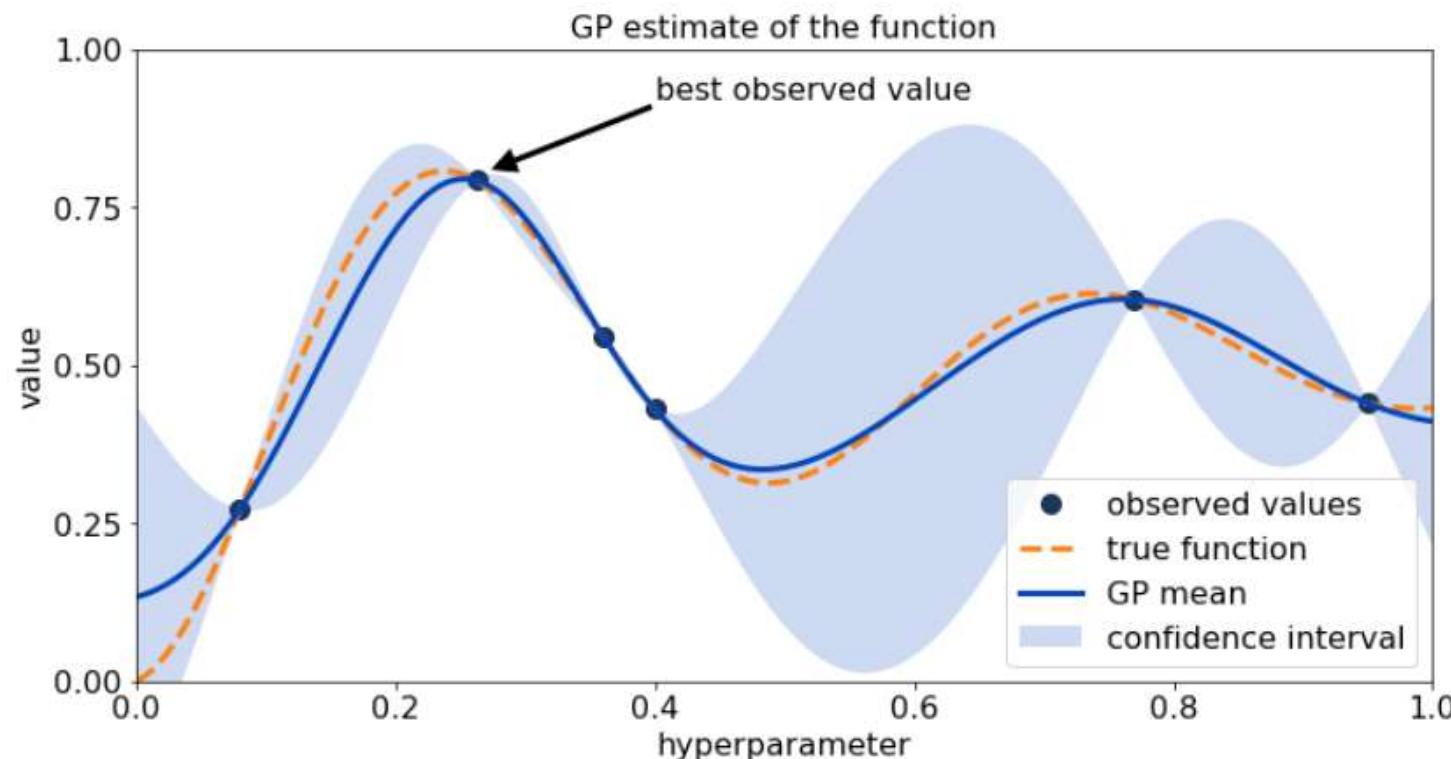
□ Bayesian Optimization # 4/4

Step 4: Sample the parameter space at the points picked on Step 3

획득 함수로부터 전달된 하이퍼 파라미터를 수행하여 관측된 값을 기반으로 대체 모델은 갱신되어 다시 최적 함수를 예측 추정

$$P(H|E) = \frac{P(E|H) P(H)}{P(E)}$$

우도 확률
사전 확률
사후 확률
주변 우도



Bayesian Optimization

The Problem Statement

- Optimise $f(x)$
- Unknown structure
- $f'(x)$ unknown
- Evaluating $f(x)$ is expensive

The Solution?

Bayesian Optimisation

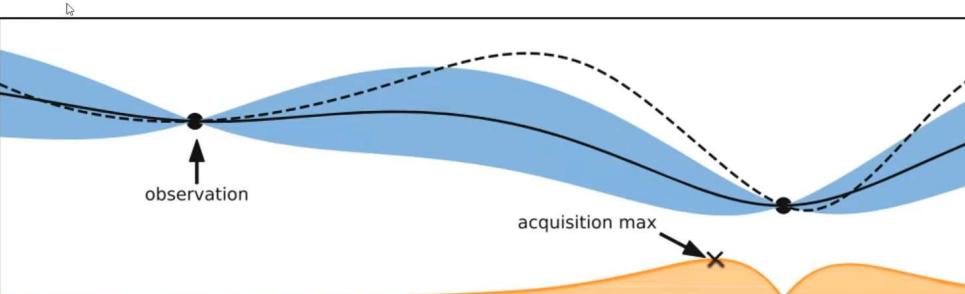
https://www.youtube.com/watch?v=41gKFmKQDIg&ab_channel=MacCormac

□ Bayesian Optimization

Bayesian Optimization

 SUNG KYUN KWAN
UNIVERSITY(SKKU)

Bayesian optimization is a **state-of-the-art** optimization framework for the global optimization of expensive blackbox functions. Bayesian optimization uses a **probabilistic surrogate model** and an **acquisition function**.



Example 1-D hyperparameter optimization process with Bayesian optimization

<https://www.youtube.com/watch?v=w9D8ozS0oC4>

□ HyperOpt – Search Space : hp 모듈

hp.quniform(label, low, high, q)	label로 지정된 입력값 변수 검색 공간을 최솟값 low에서 최댓값 high까지 q의 간격을 가지고 설정
hp.uniform(label, low, high)	최솟값 low에서 최댓값 high까지 정규 분포 형태의 검색 공간 설정
hp.randint(label, upper)	0부터 최댓값 upper까지 random한 정수값으로 검색 공간 설정
hp.loguniform(label, low, high)	$\exp(\text{uniform}(\text{low}, \text{high}))$ 값을 반환하며, 반환값의 log 변환된 값은 정규 분포 형태를 가지는 검색 공간 설정
hp.choice(label, options)	검색값이 문자열 또는 문자열과 숫자값이 섞여 있을 경우 설정. Options는 리스트나 튜플 형태로 제공되며, hp.choice('tree_criterion'.['gini', 'entropy'])과 같이 설정하면 입력 변수 tree_criterion의 값을 'gini'와 'entropy'로 설정하여 입력함

□ HyperOpt – fmin(fn, space, algo, max_evals, trials, rstate)

fn	objective function
space	search_space
algo	Bayesian Optimization 적용 알고리즘 기본적으로 tpe.suggest이며 이는 HyperOpt 기본 최적화 알고리즘인 TPE(Tree of Parzen Estimator)를 의미
max_evals	최적 입력값을 찾기 위한 입력값 시도 횟수
trials	시도한 입력값 및 목적 함수 반환값 결과를 저장하는데 사용 Trials 클래스를 객체로 생성한 변수명을 입력
rstate	random seed 값 HyperOpt 버전별로 입력값 형식이 다름

https://github.com/whatwant-school/python-ml/blob/main/06-week/06-week_02-HyperOpt.ipynb

```
In [14]: from hyperopt import fmin, tpe, Trials

trial_val = Trials()
best = fmin(fn=objective_func,
            space=xgb_search_space,
            algo=tpe.suggest,
            max_evals=50,
            trials=trial_val,
            rstate=np.random.default_rng(seed=42))

print('best:', best)

100%|██████████| 50/50 [00:08<00:00,  6.01trial/s, best loss: -0.9692
111072382943]
best: {'colsample_bytree': 0.5405128265520063, 'learning_rate': 0.09666667836341278, 'max_depth': 17.0, 'min_child_weight': 2.0}
```

[04-09]

분류 실습

캐글 산탄데르 고객 만족 예측



BANCO SANTANDER - FEATURED PREDICTION COMPETITION - 8 YEARS AGO

Santander Customer Satisfaction

Which customers are happy customers?

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#)

Overview

Start

Mar 3, 2016

Close

May 3, 2016



Description

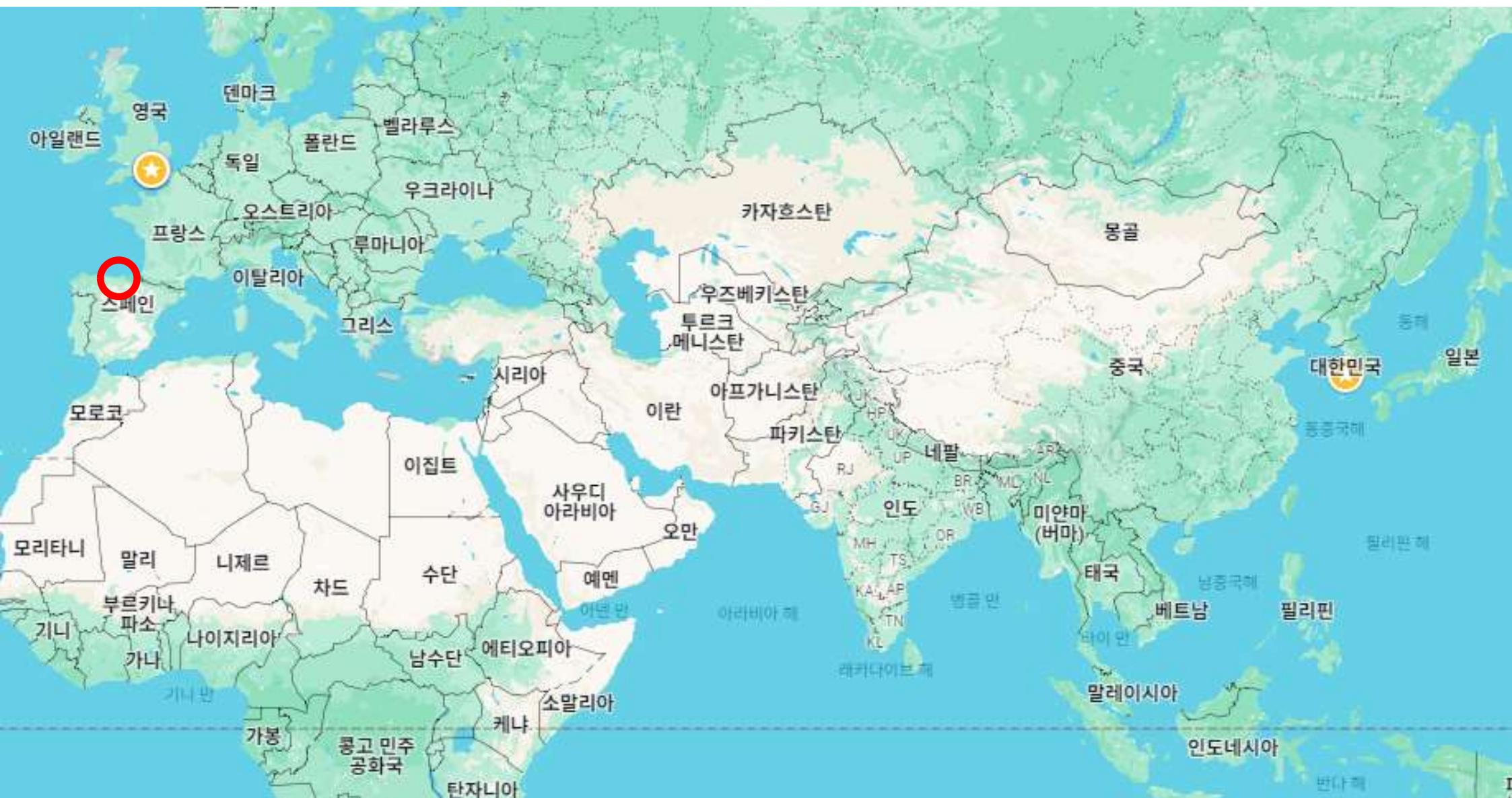


From frontline support teams to C-suites, customer satisfaction is a key measure of success. Unhappy customers don't stick around. What's more, unhappy customers rarely voice their dissatisfaction before leaving.

Santander Bank is asking Kagglers to help them identify dissatisfied customers early in their relationship. Doing so would allow Santander to take proactive steps to improve a customer's happiness before it's too late.

In this competition, you'll work with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their banking experience.

<https://www.kaggle.com/c/santander-customer-satisfaction>



산탄데르 은행

文 A 23개 언어 ▾

문서 토론

읽기 편집 역사 보기 도구 ▾

위키백과, 우리 모두의 백과사전.

산탄데르 은행(Banco Santander, 런던: BNC¹, NYSE: STD²)은 스페인의 은행으로 스페인 칸타브리아 지방 산탄데르의 지방은행으로 시작되었지만 인수합병을 통해 세계 최대 글로벌은행으로 성장하였다. 포트폴리오는 소매금융, 투자금융, 기업금융, 프라이빗뱅킹, 자산운용, 사모펀드 등 모든 분야에서 사업을 하고 있으며 2019년 현재 자산은 대략 2000조이다. 세계 각지에서 사업을 하고 있으며, 유로존 내에서 제일 큰 은행이고, 세계에서는 16 번째로 크다. 최근에는 도쿄 및 서울에도 사무소를 개설하는 등 아시아에서도 사업을 확장해나가는 추세였지만 2016년 RBS, 골드만삭스, 바클레이스, UBS 등을 따라 서울 사무소는 폐쇄하였다.^[1]

역사 [편집]

"산탄데르 은행"(Banco Santander, 1857년 설립)과 센트랄 히스파노 은행(Banco Central Hispano, 1991년 설립)이 1999년 산탄데르 센트랄 히스파노 은행(Banco Santander Central Hispano, 약칭 BSCH)로 합병되어 설립되었다. 2007년 8월 13일, 산탄데르 센트랄 히스파노 은행이 이름을 산탄데르 은행으로 바꾸었다.

각주 [편집]

1. ↑ 파이낸셜 타임즈 글로벌 500, 2011^{PDF}

산탄데르 은행



1857

창립

본사 소재지

산탄데르

웹사이트

www.santander.com/



https://github.com/whatwant-school/python-ml/blob/main/06-week/06-week_03-Santander.ipynb

```
from hyperopt import fmin, tpe, Trials

trials = Trials()

best = fmin(fn=objective_func,
            space=lgbm_search_space,
            algo=tpe.suggest,
            max_evals=50,
            trials=trials,
            rstate=np.random.default_rng(seed=42))

best
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.039291 -> initscore=-3.196685
[LightGBM] [Info] Start training from score -3.196685
[LightGBM] [Info] Number of positive: 1592, number of negative: 38952
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 1.762431 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 12824
[LightGBM] [Info] Number of data points in the train set: 40544, number of used features: 192
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.039266 -> initscore=-3.197339
[LightGBM] [Info] Start training from score -3.197339
[LightGBM] [Info] Number of positive: 1617, number of negative: 38927
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.439664 seconds.
```

[04-10]
분류 실습
캐글 신용카드 사기 검출

<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

MACHINE LEARNING GROUP - ULB AND 1 COLLABORATOR - UPDATED 6 YEARS AGO

▲ 11083 New Notebook Download (69 MB) ⋮

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine



Data Card Code (4605) Discussion (103) Suggestions (0)

About Dataset

Context

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Usability

8.53

License

Database: Open Database, Cont...

Expected update frequency

Not specified

Tags

Finance Crime

https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

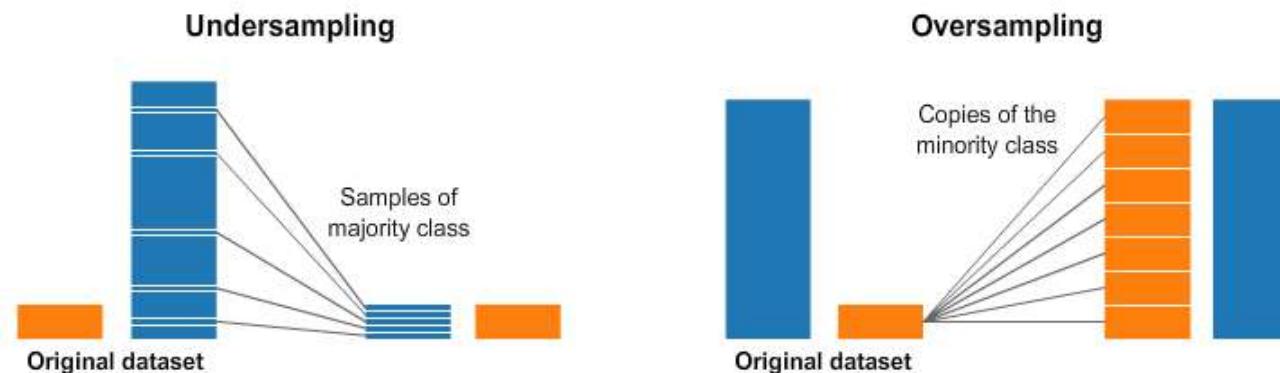
The screenshot shows the 'API reference' section of the imbalanced-learn documentation. The left sidebar has a 'Section Navigation' tree with categories like 'Under-sampling methods', 'Over-sampling methods' (which is expanded), and various sub-methods such as 'SMOTE', 'SMOTENC', etc. The main content area shows the 'SMOTE' class definition:

```
class imblearn.over_sampling.SMOTE(*, sampling_strategy='auto',
random_state=None, k_neighbors=5, n_jobs=None)
```

A purple box highlights the '[source]' link. Below the code, a description states: "Class to perform over-sampling using SMOTE." It notes that this is an implementation of SMOTE - Synthetic Minority Over-sampling Technique as presented in [1]. A link to the 'User Guide' is provided.

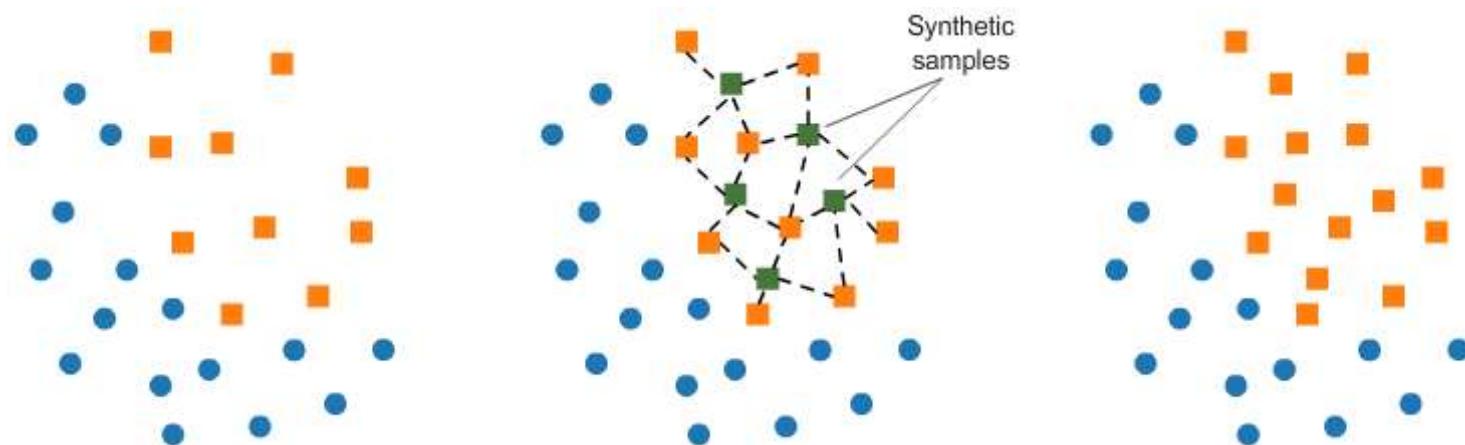
The right sidebar contains links for 'On this page' (SMOTE), 'Examples using imblearn.over_sampling.SMOTE', 'Edit on GitHub', and 'Show Source'.

<https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets?scriptVersionId=1756536&cellId=11>



<https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets?scriptVersionId=1756536&cellId=37>

Over-sampling: SMOTE (Synthetic Minority Oversampling TECnique)



https://github.com/whatwant-school/python-ml/blob/main/06-week/06-week_04-CreditCard.ipynb

```
# !pip install imbalanced-learn

from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=0)
train_X_over, train_y_over = smote.fit_resample(train_X, train_y)
print('SMOTE 적용 전 학습용 피처/레이블 데이터 세트: ', train_X.shape, train_y.shape)
print('SMOTE 적용 후 학습용 피처/레이블 데이터 세트: ', train_X_over.shape, train_y_over.shape)
print('SMOTE 적용 후 레이블 값 분포: \n', pd.Series(train_y_over).value_counts())

SMOTE 적용 전 학습용 피처/레이블 데이터 세트: (199362, 29) (199362,)
SMOTE 적용 후 학습용 피처/레이블 데이터 세트: (398040, 29) (398040,)
SMOTE 적용 후 레이블 값 분포:
   Class
0    199020
1    199020
Name: count, dtype: int64
```

You've really worked hard today

Next Week ~ ?

05**회귀**

11. 스텝킹 양상을	295
기본 스텝킹 모델	297
CV 세트 기반의 스텝킹	300
12. 정리	306
<hr/>	
01. 회귀 소개	308
02. 단순 선형 회귀를 통한 회귀 이해	310
회귀	
03. 비용 최소화하기 – 경사 하강법(Gradient Descent) 소개	312
04. 사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측	321
LinearRegression 클래스 – Ordinary Least Squares	321
회귀 평가 지표	322
LinearRegression을 이용해 보스턴 주택 가격 회귀 구현	324
05. 다항 회귀와 과(대)적합/과소적합 이해	329
다항 회귀 이해	329
다항 회귀를 이용한 과소적합 및 과적합 이해	332
편향–분산 트레이드오프(Bias~Variance Trade off)	336
06. 규제 선형 모델 – 릿지, 라쏘, 엘라스틱넷	337
규제 선형 모델의 개요	337
릿지 회귀	339
라쏘 회귀	342
엘라스틱넷 회귀	345
선형 회귀 모델을 위한 데이터 변환	347
07. 로지스틱 회귀	350

08. 회귀 트리	355
-----------	-----

09. 회귀 실습 – 자전거 대여 수요 예측	362
--------------------------	-----

데이터 클렌징 및 가공과 데이터 시각화	363
-----------------------	-----

로그 변환, 피처 인코딩과 모델 학습/예측/평가	368
----------------------------	-----

10. 회귀 실습 – 캐글 주택 가격: 고급 회귀 기법	375
--------------------------------	-----

데이터 사전 처리(Preprocessing)	375
--------------------------	-----

선형 회귀 모델 학습/예측/평가	380
-------------------	-----

회귀 트리 모델 학습/예측/평가	391
-------------------	-----

회귀 모델의 예측 결과 혼합을 통한 최종 예측	392
---------------------------	-----

스텝킹 양상을 모델을 통한 회귀 예측	394
----------------------	-----

11. 정리	397
--------	-----

01. 차원 축소(Dimension Reduction) 개요	399
-----------------------------------	-----

02. PCA(Principal Component Analysis)	401
---------------------------------------	-----

PCA 개요	401
--------	-----

03. LDA(Linear Discriminant Analysis)	415
---------------------------------------	-----

LDA 개요	415
--------	-----

04. SVD(Singular Value Decomposition)	418
---------------------------------------	-----

SVD 개요	418
--------	-----

사이킷런 TruncatedSVD 클래스를 이용한 변환	424
-------------------------------	-----

05. NMF(Non-Negative Matrix Factorization)	427
--	-----

NMF 개요	427
--------	-----

06. 정리	429
--------	-----

Who ~ ?

See you Next Weekend ~ ?