LightGBM

부스팅 계열 알고리즘에서 끝판왕.

사실상 윈도우 전용이라고 생각해야 함(다른 환경에서는 설치가 매우 어렵고 성공률도 낮다.)

XGBoost가 하이퍼파라미터를 튜닝하면 학습시간이 매우 오래 걸리는데 이를 획기적으로 단축한 모델

전체 성능(GUI사용)을 내기 위해서는 `Install Visual Studio (2015 or newer)` 환경에서 소스코드를 직접 내려받아 빌드하는 과정이 필요함.

#01 패키지 참조

> Microsoft Visual C++ 재배포 가능도구가 설치되어 있어야 한다.

CPU기반에서 동작하는 기본 버전임

```
$ pip install lightgbm
```

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
from lightgbm import LGBMClassifier
from lightgbm import plot_importance

from matplotlib import pyplot as plt
import seaborn as sb

from pandas import read_excel
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, roc_auc_score
```

#02. 데이터 가져오기 및 전처리

```
origin = read_excel("https://data.hossam.kr/G02/breast_cancer.xlsx")

# 독립/종속 변수 분리
x = origin.drop('target', axis=1)
y = origin['target']

# 데이터 스케일링
std_x = StandardScaler().fit_transform(x)
```

```
# 훈련/검증 데이터 분리
x_train, x_test, y_train, y_test = train_test_split(
    std_x, y, test_size=0.3, random_state=2021)

# SMOTE 적용
x_sm, y_sm = SMOTE(random_state=2021).fit_resample(x_train, y_train)

x_sm.shape, y_sm.shape, x_test.shape, y_test.shape
```

```
((500, 30), (500,), (171, 30), (171,))
```

#03. 단일 학습 모델

학습모델 적합

학습 성능 확인

혼동 행렬

```
confusion = confusion_matrix(y_test, y_pred)
confusion
```

```
array([[ 59,   5],
       [  1, 106]], dtype=int64)
```

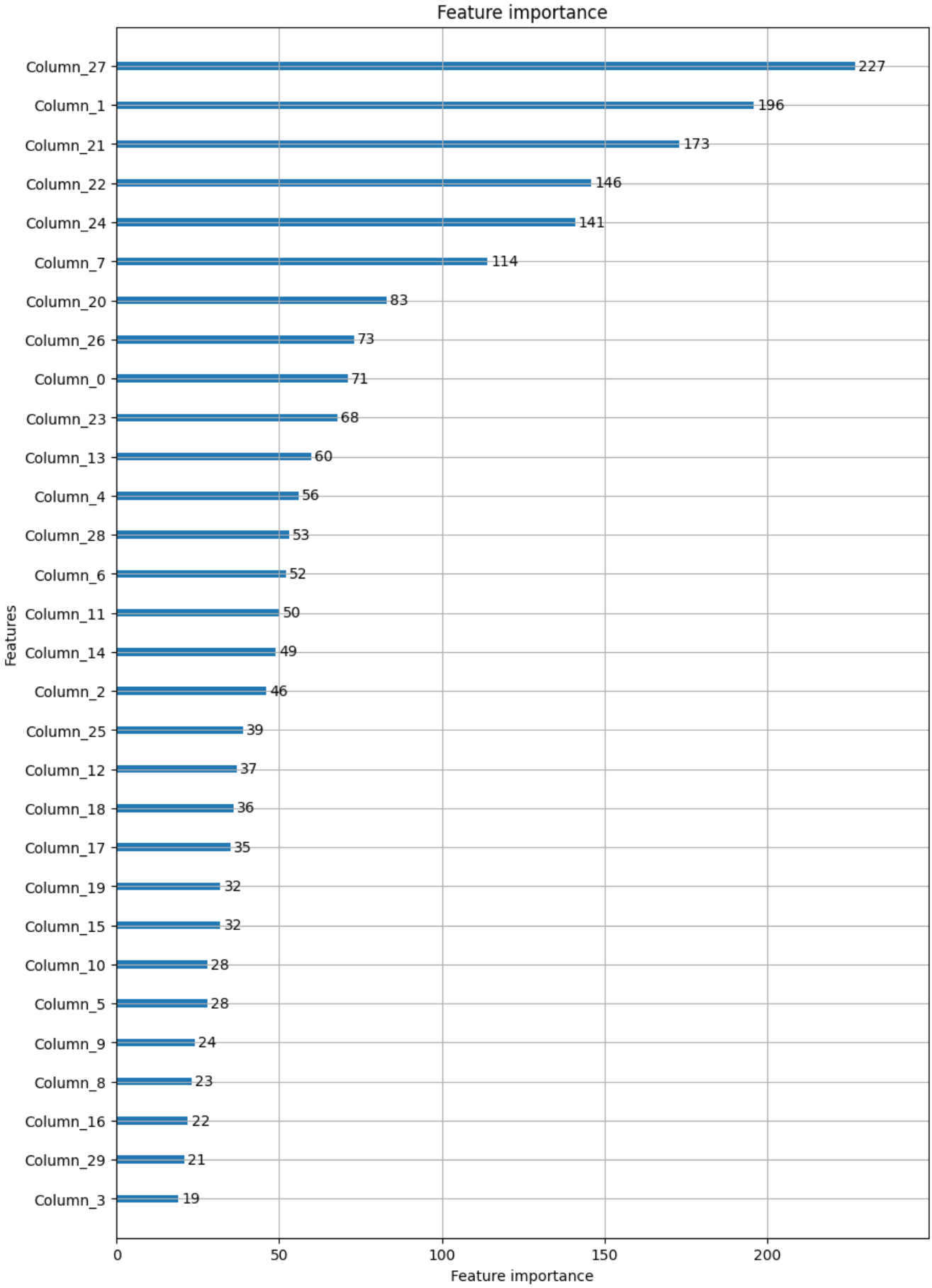
각종 성능지표 확인

```
accuracy = accuracy_score(y_test , y_pred)
precision = precision_score(y_test , y_pred)
recall = recall_score(y_test , y_pred)
f1 = f1_score(y_test, y_pred)
print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f}, F1:
{3:.4f}'.format(accuracy, precision, recall, f1))
```

```
정확도: 0.9649, 정밀도: 0.9550, 재현율: 0.9907, F1: 0.9725
```

변수별 중요도 확인

```
plt.figure(figsize=(10,15))
plot_importance(lgbm, ax=plt.gca())
plt.show()
plt.close()
```



#03. 하이퍼 파라미터 튜닝

주요 파라미터

파라미터 명[기본값]	설명
num_iterations[default=100]	반복수행하려는 트리의 개수 지정(사이킷런에서 XGB의
	n_estimators 과 같은 파라미터)
learning_rate[dafault = 0.1]	0~1사이의 값을 지정하며, 부스팅 스텝을 반복적으로 수행할 때 업데이트 되는 학습률 값
	(n_estimators를 크게하고, learning_rate를 작게해서 예측성능을 주로할 수있음)
max_depth[default =-1]	트리 기반 알고리즘의 max_depth와 같다.
	0보다 작은값으로 지정시, 깊이에 대한 제한이 없음
min_data_in_leaf[default =20]	DecisionTree의 min_samples_leaf와 같음
	* 사이킷런 LGBM에서는 min_child_samples로 쓴다
	"최종 결정 클래스인 리프노드가 되기 위해서 최소한으로 필요한 레코드의 수"
num_leaves	하나의 트리가 가질 수 있는 최대 리프 개수
boosting[default =gbdt]	부스팅의 트리를 생성하는 알고리즘을 기술
	gbdt : 일반적 그래디언트 부스팅 결정트리
	rf : 랜덤 포레스트
bagging_fraction[default = 1.0]	트리가 커져서 과적합 되는 것을 제어하기 위해, 데이터를 샘플링 하는 비율을 지정
	*사이킷런LightGBMClassifier에서는 subsample로 변경
feature_fraction[default = 1.0]	개별 트리를 학습할 때마다 무작위로 선택하는 피처의 비율
	과적합을 막기위해 사용
	(LightGBMClassifier : colsample_bytree 로 변경)
lambda_l2[default = 0.0]	L2 regulation 제어용
	reg_lambda로 변환
lambda_l1[default =0.0]	L1 regulation 제어용
	reg_alpha로 변환

Learning Task Parameter

objective : 최솟값을 가져야 할 손실함수를 정의 (XGBoost의 objective 파라미터와동일) 회귀,다중 클래스 분류, 이진분류에 따라 objective인 손실함수가 지정

```
lgbm = LGBMClassifier(random_state=777, n_jobs=-1)

params = {
    "n_estimators": [300, 400],
    "learning_rate": [0.05, 0.1, 0.15],
    "max_depth": [5, 7]
}

grid = GridSearchCV(lgbm, param_grid=params, scoring='accuracy', cv=5)

grid.fit(x_sm, y_sm)
```

```
print(grid.best_params_)
print(grid.best_score_)
best_model = grid.best_estimator_
best_model
```

```
LGBMClassifier(learning_rate=0.05, max_depth=5, n_estimators=400, n_jobs=-1,
               random_state=777)
```

성능 지표 확인

```
y_pred = best_model.predict(x_test)

accuracy = accuracy_score(y_test , y_pred)
precision = precision_score(y_test , y_pred)
recall = recall_score(y_test , y_pred)
f1 = f1_score(y_test, y_pred)
print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f}, F1:
{3:.4f}'.format(accuracy, precision, recall, f1))
```

```
정확도: 0.9649, 정밀도: 0.9633, 재현율: 0.9813, F1: 0.9722
```