AutoML 실습해보기

학습 목표

- 실습을 통해 AutoML에 대해 이해해봅니다.

학습 내용

- pycaret는 무엇일까?
- pycaret를 이용한 실습

01. pycaret는 무엇일까?

- 여러 모델을 한번에 학습, 비교해 주는 오픈 소스 머신러닝 라이브러리이다.
- Web : https://pycaret.gitbook.io/docs/ (https://pycaret.gitbook.io/docs/)
- scikit-learn 패키지를 기반으로 하고 있다.
- 설치
  - conda create -n pycaret python=3.8
  - conda activate pycaret
  - pip install jupyter
  - pip install pycaret
- 문서 개발 환경 : pycaret 2.3.10

In [1]:

```
import pycaret
from IPython.display import Image, display
import pandas as pd
```

In [2]:

```
print(pycaret.__version__)
```

2.3.10

02. pycaret를 이용한 실습

데이터 불러오기

- get_data() 함수로 데이터를 불러올 수 있음.

In [3]:

```
from pycaret.datasets import get_data
dataset = get_data('credit')
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...
0	20000	2	2	1	24	2	2	-1	-1	-2	...
1	90000	2	2	2	34	0	0	0	0	0	...
2	50000	2	2	1	37	0	0	0	0	0	...
3	50000	1	2	1	57	-1	0	-1	0	0	...
4	50000	1	1	2	37	0	0	0	0	0	...

5 rows × 24 columns

In [4]:

```
print(dataset.shape)
dataset.head()
## target : default
```

(24000, 24)

Out[4]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...
0	20000	2	2	1	24	2	2	-1	-1	-2	...
1	90000	2	2	2	34	0	0	0	0	0	...
2	50000	2	2	1	37	0	0	0	0	0	...
3	50000	1	2	1	57	-1	0	-1	0	0	...
4	50000	1	1	2	37	0	0	0	0	0	...

5 rows × 24 columns

dataset를 train, test로 나누어주기

In [5]:

```
train = dataset.sample(frac=0.9, random_state=77)
test = dataset.drop(train.index)

train.shape, test.shape
```

Out[5]:

((21600, 24), (2400, 24))

In [6]:

```
### index 재설정
train.reset_index(inplace=True, drop=True)
test.reset_index(inplace=True, drop=True)

display(train), display(test)
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
0	80000	2	3	1	39	-1	-1	-1	0	-1
1	300000	2	1	1	45	-1	-1	-1	0	0
2	30000	2	3	3	48	0	0	2	0	0
3	180000	1	1	1	37	1	-2	-2	-2	-1
4	160000	2	2	1	31	3	2	2	0	0
...	...	...	...	...	...	...	...	...	...	...
21595	80000	1	1	2	34	2	2	2	2	2
21596	80000	1	1	2	33	0	0	0	0	-2
21597	20000	1	2	2	22	0	0	2	2	2
21598	80000	1	2	1	34	1	2	2	0	0
21599	170000	1	1	1	38	1	-2	-2	-2	-2

21600 rows × 24 columns

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
0	100000	2	2	2	23	0	-1	-1	0	0
1	50000	2	3	2	30	0	0	0	0	0
2	360000	1	1	2	33	0	0	0	0	0
3	400000	2	2	1	29	0	0	0	0	0
4	200000	1	1	1	57	-2	-2	-2	-1	2
...	...	...	...	...	...	...	...	...	...	...
2395	50000	1	3	1	46	3	3	2	0	0
2396	360000	1	1	2	31	-1	-1	-1	0	0
2397	50000	1	2	1	37	1	2	2	2	0
2398	50000	1	2	1	44	1	2	2	2	0
2399	80000	1	2	2	34	2	2	2	2	2

2400 rows × 24 columns

Out[6]:

(None, None)

설정

- set_up() 함수를 이용하여 사용할 데이터와 출력(target)를 설정
  - session_id : random_state와 같은 개념
  - data : 사용할 데이터 셋
  - target : 예측할 target 특징을 선택

In [7]:

```
from pycaret.classification import *

model_set = setup(data=train, target='default', session_id=123)
```

	Description	Value
0	session_id	123
1	Target	default
2	Target Type	Binary
3	Label Encoded	None
4	Original Data	(21600, 24)
5	Missing Values	False
6	Numeric Features	14
7	Categorical Features	9
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(15119, 89)
12	Transformed Test Set	(6481, 89)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	7092
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	False
30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None
33	PCA	False

	Description	Value
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Remove Perfect Collinearity	True
45	Clustering	False
46	Clustering Iteration	None
47	Polynomial Features	False
48	Polynomial Degree	None
49	Trignometry Features	False
50	Polynomial Threshold	None
51	Group Features	False
52	Feature Selection	False
53	Feature Selection Method	classic
54	Features Selection Threshold	None
55	Feature Interaction	False
56	Feature Ratio	False
57	Interaction Threshold	None
58	Fix Imbalance	False
59	Fix Imbalance Method	SMOTE

모델 비교

- 모델을 비교한 이후에 결과가 출력된다. 각 metric별 가장 성능이 좋은 위치에 노락색으로 하이라이트 되어 출력이 된다.
- fold : cross_validation의 fold를 지정(default=10)
- sort : 정렬기준 지표 설정
- n_select : 상위 n개의 모델 결과만 출력.

In [8]:

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.8207	0.7703	0.3770	0.6856	0.4864	0.3887	0.4143	0.2230
ridge	Ridge Classifier	0.8205	0.0000	0.3644	0.6934	0.4776	0.3818	0.4106	0.0260
gbc	Gradient Boosting Classifier	0.8195	0.7803	0.3632	0.6889	0.4756	0.3790	0.4073	0.9040
ada	Ada Boost Classifier	0.8180	0.7738	0.3539	0.6866	0.4668	0.3705	0.4002	0.2290
lightgbm	Light Gradient Boosting Machine	0.8171	0.7768	0.3709	0.6697	0.4772	0.3773	0.4016	0.0840
rf	Random Forest Classifier	0.8127	0.7655	0.3682	0.6493	0.4699	0.3665	0.3883	0.4800
et	Extra Trees Classifier	0.8032	0.7426	0.3768	0.6018	0.4633	0.3505	0.3650	0.5470
dummy	Dummy Classifier	0.7746	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0180
lr	Logistic Regression	0.7745	0.6471	0.0000	0.0000	0.0000	-0.0001	-0.0014	1.1100
knn	K Neighbors Classifier	0.7496	0.6053	0.1907	0.3866	0.2552	0.1253	0.1366	0.5210
dt	Decision Tree Classifier	0.7218	0.6132	0.4155	0.3902	0.4023	0.2213	0.2216	0.0880
svm	SVM - Linear Kernel	0.7112	0.0000	0.1856	0.2105	0.1491	0.0429	0.0487	0.1550
qda	Quadratic Discriminant Analysis	0.6754	0.5502	0.3225	0.2944	0.3037	0.0954	0.0965	0.1650
nb	Naive Bayes	0.3574	0.6415	0.9058	0.2473	0.3886	0.0533	0.1141	0.0300

하나의 모델 지정 후, 최적화

- create_model() : 하나의 모델을 최적화 시킨다.

모델의 종류

- 'lr' : Logistic Regression
- 'knn', 'nb'(Naives Bayes), 'dt'(의사결정트리)
- 'svm', 'rbfsvm', 'gpc'(Gaussian Process Classifier)
- 'mlp', 'ridge'(Ridge Classifier), 'rf'
- 'qda', 'ada'(Ada Boost Classifier), 'lda', 'et'(Extra Trees Classifier), 'xgboost', 'lightdbm', 'catboost'

In [9]:

```
model_rf = create_model('rf')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8003	0.7623	0.3412	0.5979	0.4345	0.3240	0.3428
1	0.8148	0.7630	0.3724	0.6580	0.4757	0.3735	0.3959
2	0.8128	0.7760	0.3578	0.6559	0.4630	0.3613	0.3857
3	0.8128	0.7738	0.3842	0.6422	0.4807	0.3753	0.3937
4	0.8122	0.7507	0.3607	0.6508	0.4642	0.3614	0.3846
5	0.8155	0.7754	0.3783	0.6582	0.4804	0.3781	0.3995
6	0.8115	0.7574	0.3666	0.6443	0.4673	0.3631	0.3845
7	0.8108	0.7766	0.3695	0.6396	0.4684	0.3632	0.3835
8	0.8254	0.7639	0.3842	0.7081	0.4981	0.4035	0.4311
9	0.8107	0.7563	0.3676	0.6378	0.4664	0.3613	0.3816
Mean	0.8127	0.7655	0.3682	0.6493	0.4699	0.3665	0.3883
Std	0.0058	0.0089	0.0124	0.0257	0.0155	0.0188	0.0205

모델의 하이퍼 파리미터 튜닝을 진행

- tune_model([model], optimize=[]) 함수를 사용하여 모델의 하이퍼 파라미터 튜닝 진행
  - optimize : 평가 metric 지정

In [10]:

```
tunning_model = tune_model(model_rf)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7976	0.7389	0.2853	0.6062	0.3880	0.2851	0.3143
1	0.8194	0.7485	0.3372	0.7099	0.4573	0.3650	0.4015
2	0.8194	0.7644	0.3607	0.6910	0.4740	0.3777	0.4068
3	0.8168	0.7647	0.3754	0.6667	0.4803	0.3795	0.4026
4	0.8208	0.7423	0.3343	0.7215	0.4569	0.3664	0.4054
5	0.8261	0.7671	0.3636	0.7294	0.4853	0.3945	0.4291
6	0.8155	0.7523	0.3519	0.6742	0.4624	0.3640	0.3921
7	0.8234	0.7415	0.3578	0.7176	0.4775	0.3852	0.4191
8	0.8168	0.7550	0.3314	0.6975	0.4493	0.3557	0.3912
9	0.8154	0.7296	0.3294	0.6871	0.4453	0.3506	0.3848
Mean	0.8171	0.7504	0.3427	0.6901	0.4576	0.3624	0.3947
Std	0.0073	0.0119	0.0242	0.0340	0.0265	0.0287	0.0296

In [14]:

```
tunning_model
```

Out[14]:

```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight={},
                       criterion='entropy', max_depth=5, max_features=1.0,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0002, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=10,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=-1, oob_score=False, random_state=123, verbose=0,
                       warm_start=False)
```

여러 모델을 사용하여 새로운 모델을 생성(blending)

- blend_models() 함수를 사용하여 여러 모델 혼합이 가능

In [15]:

```
dt = create_model('dt')  # 의사결정트리
rf = create_model('rf')  # RandomForest

blender_model_2 = blend_models(estimator_list = [dt, rf])
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7275	0.7452	0.4235	0.4000	0.4114	0.2343	0.2345
1	0.7189	0.7300	0.3783	0.3772	0.3777	0.1962	0.1962
2	0.7176	0.7441	0.4106	0.3825	0.3960	0.2121	0.2123
3	0.7242	0.7428	0.4018	0.3914	0.3965	0.2178	0.2179
4	0.7249	0.7294	0.4252	0.3973	0.4108	0.2316	0.2318
5	0.7123	0.7443	0.4370	0.3801	0.4065	0.2179	0.2188
6	0.7348	0.7390	0.4340	0.4157	0.4247	0.2525	0.2526
7	0.7123	0.7436	0.4135	0.3750	0.3933	0.2053	0.2058
8	0.7321	0.7398	0.4311	0.4106	0.4206	0.2465	0.2467
9	0.7121	0.7212	0.4000	0.3706	0.3847	0.1972	0.1974
Mean	0.7217	0.7379	0.4155	0.3900	0.4022	0.2211	0.2214
Std	0.0079	0.0078	0.0175	0.0147	0.0144	0.0186	0.0186

5가지 모델 혼합

In [16]:

```
best_model_5 = compare_models(n_select=5)
blender_model_5 = blend_models(best_model_5)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8016	0.0000	0.3176	0.6136	0.4186	0.3133	0.3380
1	0.8221	0.0000	0.3724	0.6978	0.4857	0.3899	0.4180
2	0.8181	0.0000	0.3607	0.6833	0.4722	0.3747	0.4027
3	0.8261	0.0000	0.3930	0.7053	0.5047	0.4094	0.4352
4	0.8221	0.0000	0.3724	0.6978	0.4857	0.3899	0.4180
5	0.8188	0.0000	0.3607	0.6872	0.4731	0.3762	0.4048
6	0.8188	0.0000	0.3724	0.6791	0.4811	0.3824	0.4078
7	0.8287	0.0000	0.3959	0.7181	0.5104	0.4169	0.4441
8	0.8214	0.0000	0.3636	0.7006	0.4788	0.3838	0.4139
9	0.8173	0.0000	0.3588	0.6778	0.4692	0.3713	0.3987
Mean	0.8195	0.0000	0.3668	0.6861	0.4779	0.3808	0.4081
Std	0.0069	0.0000	0.0205	0.0269	0.0236	0.0265	0.0270

예측 수행

- finalize_model() : 최종 모델로 설정 후, 마지막 학습 진행
- predict_model() : 예측 결과를 'Label'변수에 저장

In [17]:

```
last_model = finalize_model(blender_model_5)
```

In [18]:

```
pred = predict_model(last_model, data=test)
pred[0:10]
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.8333	0.6538	0.3526	0.6654	0.4609	0.3732	0.3995

Out[18]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...
0	100000	2	2	2	23	0	-1	-1	0	0	...
1	50000	2	3	2	30	0	0	0	0	0	...
2	360000	1	1	2	33	0	0	0	0	0	...
3	400000	2	2	1	29	0	0	0	0	0	...
4	200000	1	1	1	57	-2	-2	-2	-1	2	...
5	10000	1	2	1	56	2	2	2	0	0	...
6	130000	2	3	2	29	1	-2	-2	-1	2	...
7	280000	1	2	1	41	2	2	2	2	2	...
8	240000	1	1	2	28	-1	-1	-1	-1	-1	...
9	180000	1	1	1	36	0	0	0	0	0	...

10 rows × 25 columns

- pred는 출력입니다. Label이라는 column이 생기며, 여기에 모델이 예측한 결과가 추가된다.

평가

In [19]:

```
from pycaret.utils import check_metric
check_metric(test['default'], pred['Label'], metric='Accuracy')
```

Out[19]:

0.8333

REFERENCE

- https://pycaret.gitbook.io/docs/ (https://pycaret.gitbook.io/docs/)