건설기계 작동 오일 상태판단 모델 개발

2019380602 홍서연
2019390810 윤혜준

통계학과 머신러닝 PBL 10조

목차

분석 주제 및 목적

건설기계 작동오일 상태판단 모델 개발

주 제

건설장비에서 작동오일의 상태를 실시간으로 모니터링하기 위한
오일 상태 판단 모델 개발 (정상, 이상의 이진분류)

목 적

건설 장비 내부 기계 부품의 마모 상태 및 윤활 성능을 오일 데이터 분석을 통해 확인하고,
다양한 분류 모델을 적용시키고 모델을 개발하여 적절한 교체 주기 파악하기

데이터 설명 - EDA

분석에 사용할 데이터

- 데이터 출처 : 현대제뉴인 / 건설기계 오일 상태 분류 경진대회 https://dacon.io/competitions/official/236013/data

데이터 형태	범주형 + 수치형	- 범주형 자료의 인코딩 필요
데이터 크기	14,095 (개)	- train, test 6:4으로 분할하여 분석
변수의 개수	52 (개)	- COMPONENT_ARBITRARY를 제외한 51개의 수치형 변수 - Y_LABEL : 0(정상) / 1(이상)

	ID	COMPONENT_ARBITRARY	ANONYMOUS_1	YEAR	SAMPLE_TRANSFER_DAY	ANONYMOUS_2	AG	AL	B	BA	...	U25	U20	U14	U6	U4	V	V100	V40	ZN	Y_LABEL
0	TRAIN_00000	COMPONENT3	1486	2011	7	200	0	3	93	0	...	NaN	NaN	NaN	NaN	NaN	0	NaN	154.0	75	0
1	TRAIN_00001	COMPONENT2	1350	2021	51	375	0	2	19	0	...	2.0	4.0	6.0	216.0	1454.0	0	NaN	44.0	652	0
2	TRAIN_00002	COMPONENT2	2415	2015	2	200	0	110	1	1	...	0.0	3.0	39.0	11261.0	41081.0	0	NaN	72.6	412	1
3	TRAIN_00003	COMPONENT3	7389	2010	2	200	0	8	3	0	...	NaN	NaN	NaN	NaN	NaN	0	NaN	133.3	7	0
4	TRAIN_00004	COMPONENT3	3954	2015	4	200	0	1	157	0	...	NaN	NaN	NaN	NaN	NaN	0	NaN	133.1	128	0

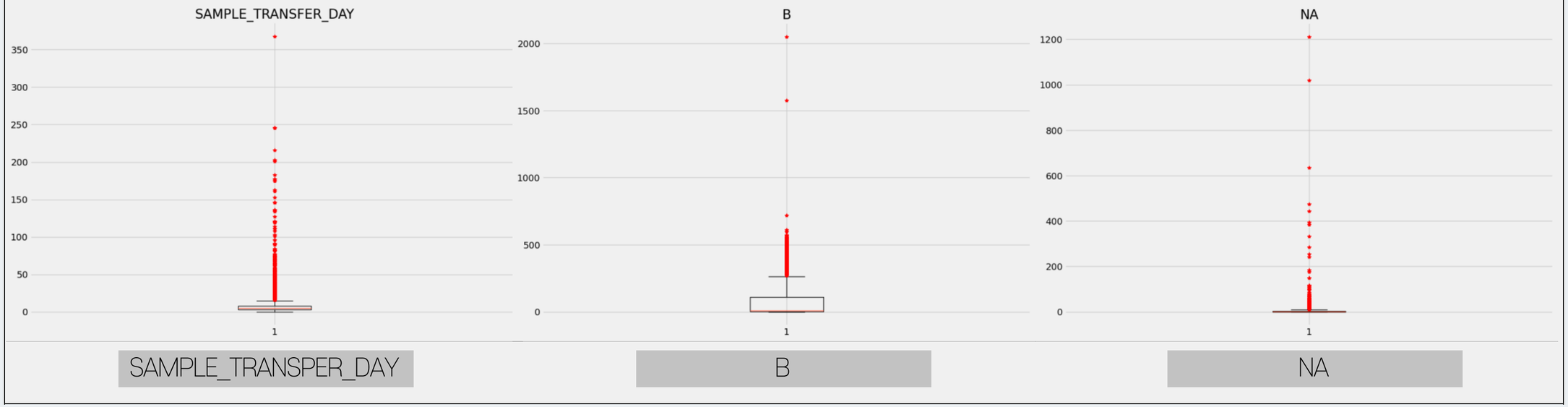
5 rows × 54 columns

< head( ) 결과 >

데이터 설명 - EDA

변수 분포 확인

< Boxplot 결과 >



▶ 대부분의 변수들이 위와 같은 분포, 자료 내의 이상치가 매우 많음을 확인

▶ 이상치를 제거하게 되면 데이터의 대부분이 소실되어 정확한 데이터 정보 파악이 불가능

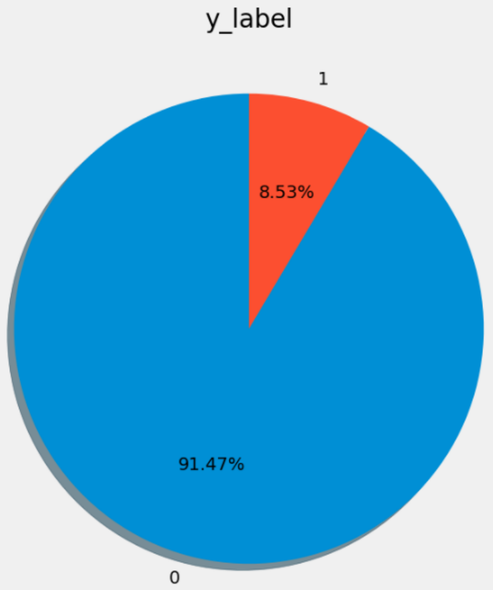
→ 이상치 포함해 데이터 시각화 진행

데이터 설명 - EDA

변수 분포 확인

< Y_LABEL 분포 >

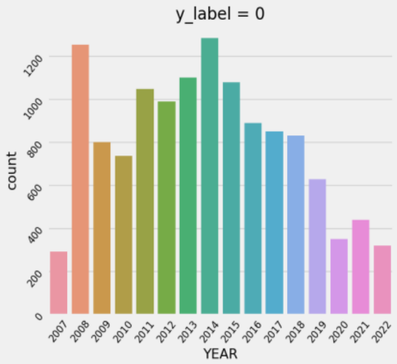
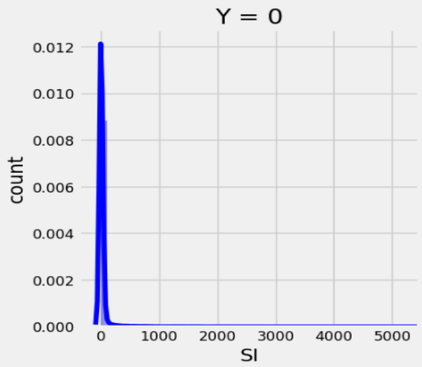
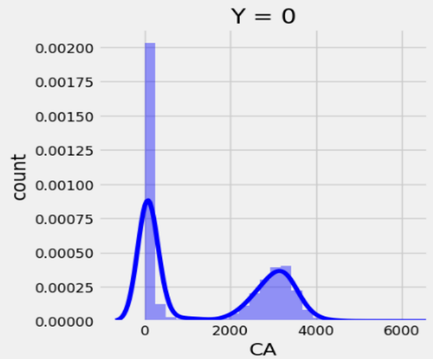
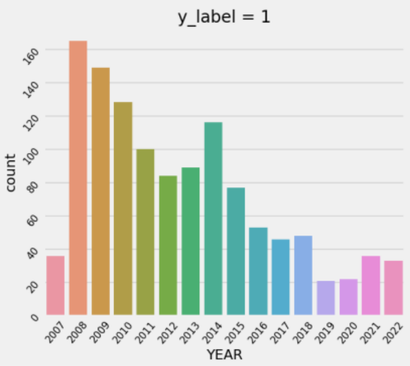
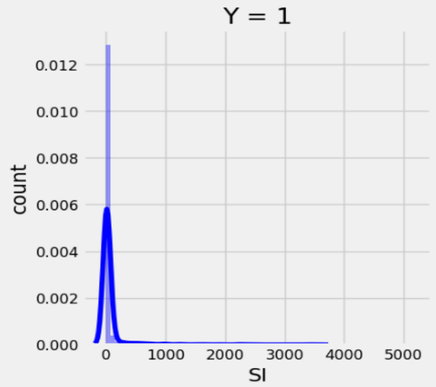
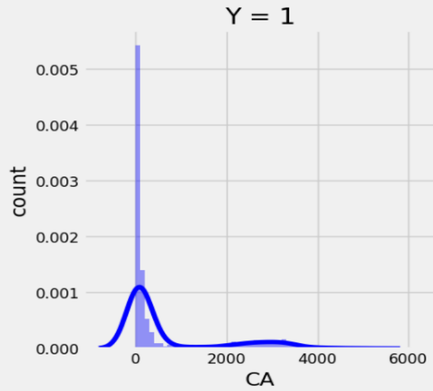
```
plt.subplots(figsize = (8,8))
plt.pie(df['Y_LABEL'].value_counts(), labels = df['Y_LABEL'].value_counts().index,
        autopct="%.2f%%", shadow = True, startangle = 90)
plt.title('Y Ratio', size=20)
plt.show()
```



▶ Y_LABEL 분포의 불균형

▶ Y_LABEL 값을 반영한 그래프 해석 필요

< Y_LABEL에 따른 변수 분포 >

	YEAR	SI	CA
Y = 0			
Y = 1			

데이터 설명 - EDA

변수 분포 확인

< Y_LABEL ↔ 변수 간 상관관계 분석 >

	Y_LABEL
Y_LABEL	1.000000
AL	0.370512
BA	0.104840
FOPTIMETHGLY	0.055908
FNOX	0.053337
FE	0.047992
NI	0.046806
ANONYMOUS_1	0.044197
SI	0.036731
FOXID	0.032596
PQINDEX	0.028966
S	0.027923
TI	0.025637
CU	0.024975
MN	0.024274
K	0.023963
V40	0.023195
FSO4	0.020865
V	0.020862
FTBN	0.016417
AG	0.014671
CR	0.014233
FH2O	0.013476
SOOTPERCENTAGE	0.012351
V100	0.011434
BE	0.010685
CO	0.008175
P	0.007602
CD	0.005241
LI	0.002921
SN	0.002359
MO	0.001206
U6	-0.001242
SB	-0.002028
PB	-0.003549
U14	-0.004104
U20	-0.004116
H2O	-0.004262
SAMPLE_TRANSFER_DAY	-0.004315
U4	-0.005535
FUEL	-0.007714
MG	-0.008807
U25	-0.009844
U50	-0.009986
NA	-0.010820
U100	-0.020294
U75	-0.023249
ZN	-0.027551
B	-0.029787
ANONYMOUS_2	-0.033641
CA	-0.150379

```
correlation = corr.unstack()
df_temp = pd.DataFrame(correlation['Y_LABEL'].sort_values(ascending=False), columns=['Y_LABEL'])
df_temp.style.background_gradient(cmap='viridis')
```

▶ Y_LABEL과 가장 연관성이 큰 변수는 AL,
알루미늄 함유량이 Y_LABEL 값에 가장 많은 영향

▶ 하지만 관계계수의 값이 작아 큰 영향은 X

▶ 상관관계 분석 결과 Y_LABEL에 큰 영향을 주는 변수가
거의 없음을 확인

데이터 분석 - 전처리

① One-Hot Encoding

< One-Hot Encoding 전 >

ID	
TRAIN_00000	COMPONENT3
TRAIN_00001	COMPONENT2
TRAIN_00002	COMPONENT2
TRAIN_00003	COMPONENT3
TRAIN_00004	COMPONENT3
	...
TRAIN_14090	COMPONENT3
TRAIN_14091	COMPONENT1
TRAIN_14092	COMPONENT3
TRAIN_14093	COMPONENT2
TRAIN_14094	COMPONENT2

< One-Hot Encoding 후 >

	COMPONENT1	COMPONENT2	COMPONENT3	COMPONENT4
ID				
TRAIN_00000	0	0	1	0
TRAIN_00001	0	1	0	0
TRAIN_00002	0	1	0	0
TRAIN_00003	0	0	1	0
TRAIN_00004	0	0	1	0
...	...	...	...	...
TRAIN_14090	0	0	1	0
TRAIN_14091	1	0	0	0
TRAIN_14092	0	0	1	0
TRAIN_14093	0	1	0	0
TRAIN_14094	0	1	0	0

```
# COMPONENT_ARBITRARY one-hot encoding
one_hot_encoded = pd.get_dummies(df['COMPONENT_ARBITRARY'])
print(one_hot_encoded)
```

```
# 원래 열 제거
df = df.drop('COMPONENT_ARBITRARY', axis =1)
```

```
# 새로운 열들 추가
df.insert(0, 'COMPONENT1', one_hot_encoded['COMPONENT1'])
df.insert(1, 'COMPONENT2', one_hot_encoded['COMPONENT2'])
df.insert(2, 'COMPONENT3', one_hot_encoded['COMPONENT3'])
df.insert(3, 'COMPONENT4', one_hot_encoded['COMPONENT4'])
print(df)
```

▶ COMPONENT_ARBITRARY 제거 후 COMPONENT1, COMPONENT2, COMPONENT3, COMPONENT4 추가

데이터 분석 - 전처리

② 결측치 처리

< 결측치 제거 >

CD	9.890032	SOOTPERCENTAGE	72.401561
CO	0.000000	TI	0.000000
CR	0.000000	U100	83.568641
CU	0.000000	U75	83.568641
FH2O	72.401561	U50	83.568641
FNOX	72.401561	U25	83.568641
FOPTIMETHGLY	72.401561	U20	83.568641
FOXID	72.401561	U14	84.973395
FSO4	72.401561	U6	84.973395
FTBN	72.401561	U4	84.973395
FE	0.000000	V	0.000000
FUEL	72.401561	V100	73.579283
H2O	0.000000	V40	0.000000
K	16.310748		

```
df.isnull().sum()/df.shape[0]*100
```

```
percent30UP = list(df_info[df_info['nullPCT'] >= 30.0]['dataFeatures'])
df = df.drop(percent30UP,axis=1)

print(df.shape)
```

```
(14095, 39)
```

*COMPONENT 전처리로 인한 열 추가

▶ 결측치 비율이 30% 이상인 변수 제거 (변수 *57개 → 39개)

< 결측치 처리 >

```
# 결측치 비율이 30 이하인 컬럼들에 대해서만 결측치 처리
for column in df.columns:
    df[column].fillna(df[column].mean(), inplace=True)
```

▶ 결측치 비율이 30% 이하인 변수들에 대해 평균값 처리

데이터 분석 - 전처리

분석에 사용하는 변수

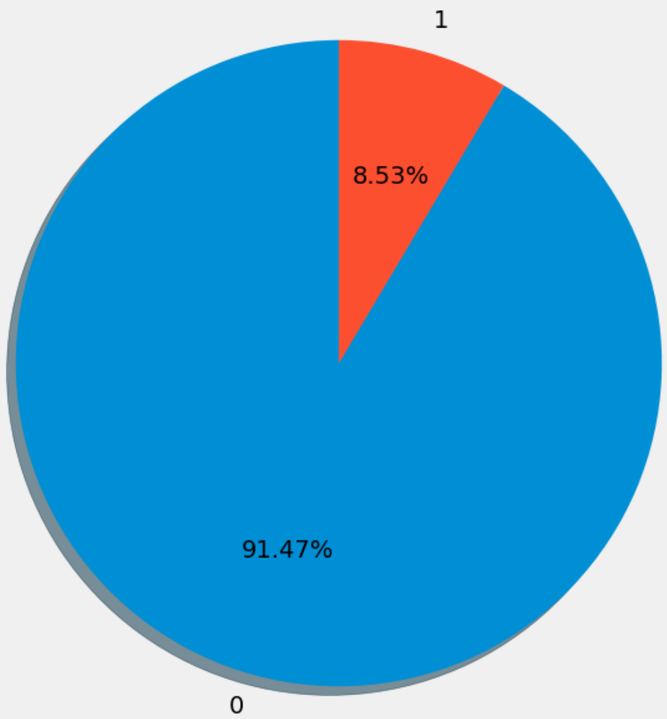
< 분석에 사용할 변수 >	
df.columns	▶ COMPONENT1, COMPONENT2, COMPONENT3, COMPONENT4(샘플 오일 관련 부품), ANONYMOUS_1, YEAR(오일 진단 해), SAMPLE_TRANSFER_DAY, ANONYMOUS_2, AG, AL, B, BA, BE, CA, CD, CO, CR, CU, FE, H2O, K, LI, MG, MN, MO, NA, NI, P, PB, PQINDEX(Particle Quantifier Index), S, SB, SI, SN, TI, V(점도), V40, ZN, Y_LABEL ▶ 39개의 feature

< 제거된 변수 >	
percent20UP	▶ FH2O(물 수치), FNOX(NOx 수치), FOPTIMETHGLY, FOXID(Oxidation 수치), FSO4(SO4 수치), FTBN(염기성 첨가제물질 수치), FUEL(연료 함유량), SOOTPERCENTAGE(Soot 함유량), U100(Particle Count), U75, U50, U25, U20, U14, U6, U4, V100

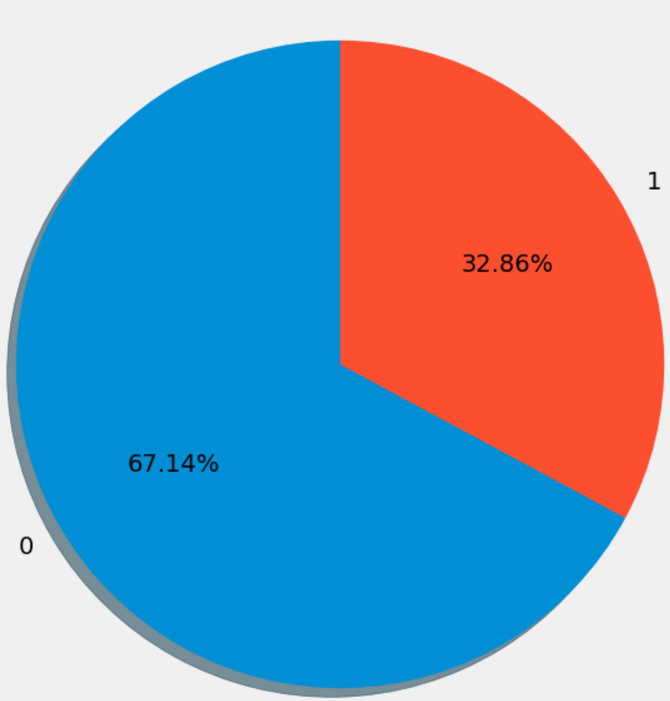
데이터 분석

Oversampling

< ADASYN 전 >



< ADASYN 후 >



▶ ADASYN 사용으로 y=1 라벨의 데이터 증가. 원래 비율 9 : 1 -> 적용후 비율 2:1

데이터 분석

모델 적용 결과

ADASYN + GridSearch+ KNN

	precision	recall	f1-score	support
0	0.98	0.92	0.95	2632
1	0.85	0.95	0.90	1209
accuracy			0.93	3841
macro avg	0.91	0.94	0.92	3841
weighted avg	0.94	0.93	0.93	3841

ADASYN + GridSearch+ LDA

	precision	recall	f1-score	support
0	0.79	1.00	0.88	2632
1	0.98	0.41	0.58	1209
accuracy			0.81	3841
macro avg	0.88	0.70	0.73	3841
weighted avg	0.85	0.81	0.78	3841

ADASYN + GridSearch+ QDA

	precision	recall	f1-score	support
0	0.86	0.99	0.92	2632
1	0.98	0.65	0.78	1209
accuracy			0.89	3841
macro avg	0.92	0.82	0.85	3841
weighted avg	0.90	0.89	0.88	3841

ADASYN + GridSearch+ Logistic

	precision	recall	f1-score	support
0	0.89	0.98	0.93	2632
1	0.95	0.73	0.83	1209
accuracy			0.90	3841
macro avg	0.92	0.86	0.88	3841
weighted avg	0.91	0.90	0.90	3841

SMOTE + GridSearch+ DecisionTree

	precision	recall	f1-score	support
0	0.92	0.91	0.92	2732
1	0.86	0.88	0.87	1780
accuracy			0.90	4512
macro avg	0.89	0.90	0.89	4512
weighted avg	0.90	0.90	0.90	4512

SMOTE + GridSearch+ SVM

	precision	recall	f1-score	support
0	0.84	0.98	0.91	2732
1	0.96	0.72	0.82	1780
accuracy			0.88	4512
macro avg	0.90	0.85	0.87	4512
weighted avg	0.89	0.88	0.87	4512

SMOTE + GridSearch+ Kernel SVM

	precision	recall	f1-score	support
0	0.89	0.98	0.93	2732
1	0.96	0.82	0.88	1780
accuracy			0.91	4512
macro avg	0.92	0.90	0.91	4512
weighted avg	0.92	0.91	0.91	4512

데이터 분석

모델 적용 결과

F-1(Y=0) : 0.97

F-1(Y=1) : 0.40

	precision	recall	f1-score	support
0	0.94	1.00	0.97	2590
1	1.00	0.25	0.40	229
accuracy			0.94	2819
macro avg	0.97	0.62	0.68	2819
weighted avg	0.94	0.94	0.92	2819

<Original>

F-1(Y=0) : 0.93

F-1(Y=1) : 0.88

	precision	recall	f1-score	support
0	0.89	0.98	0.93	2732
1	0.96	0.82	0.88	1780
accuracy			0.91	4512
macro avg	0.92	0.90	0.91	4512
weighted avg	0.92	0.91	0.91	4512

<F1 score 가장 높은 모델>

Smote + Grid Search +Kernel SVM

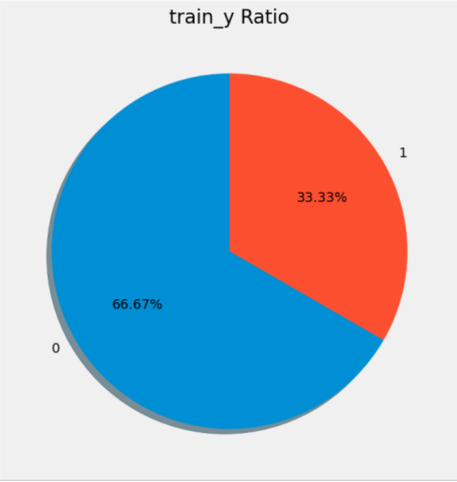
데이터 분석

훈련 데이터만 증대

< train data 데이터 증대 >

```
# ADASYN 적용
adasyn = ADASYN(sampling_strategy=0.5, n_neighbors=4, random_state =888)
X_train_adasyn, y_train_adasyn = adasyn.fit_resample(X_train, y_train)

# SMOTE 적용
smote = SMOTE(sampling_strategy=0.5, random_state = 888)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```




```
scaler = StandardScaler()

# 트레인 데이터에 대해 표준화를 적용하고, 테스트 데이터에 대해 표준화를 적용
X_train_std_smote = scaler.fit_transform(X_train_smote)
X_test_std = scaler.transform(X_test)
```

▶ 데이터에 StandardScaler 적용

▶ train data만 ADASYN과 SMOTE을 통한 y=0 오버샘플링 후 데이터 표준화 적용

데이터 분석

ADASYN, SMOTE 적용 및 Gridsearch 결과

	Logistic Regression	LDA	QDA
ADASYN	precision recall f1-score support 0 0.96 0.98 0.97 5157 1 0.74 0.51 0.61 481 accuracy 0.94 5638 macro avg 0.85 0.75 0.79 5638 weighted avg 0.94 0.94 0.94 5638 ▶ f1-score : 0.79 (y=0 : 0.97, y=1 : 0.61)	precision recall f1-score support 0 0.93 1.00 0.96 5157 1 0.93 0.23 0.37 481 accuracy 0.93 5638 macro avg 0.93 0.62 0.67 5638 weighted avg 0.93 0.93 0.91 5638 ▶ f1-score : 0.37 (y=0 : 0.96, y=1 : 0.37)	precision recall f1-score support 0 0.95 0.99 0.97 5157 1 0.89 0.46 0.61 481 accuracy 0.95 5638 macro avg 0.92 0.73 0.79 5638 weighted avg 0.95 0.95 0.94 5638 ▶ f1-score : 0.79 (y=0 : 0.97, y=1 : 0.61)
SMOTE	precision recall f1-score support 0 0.96 0.98 0.97 5157 1 0.75 0.51 0.61 481 accuracy 0.94 5638 macro avg 0.85 0.75 0.79 5638 weighted avg 0.94 0.94 0.94 5638 ▶ f1-score : 0.79 (y=0 : 0.97, y=1 : 0.61)	precision recall f1-score support 0 0.96 0.93 0.94 5157 1 0.44 0.59 0.50 481 accuracy 0.90 5638 macro avg 0.70 0.76 0.72 5638 weighted avg 0.92 0.90 0.91 5638 ▶ f1-score : 0.72 (y=0 : 0.94, y=1 : 0.50)	precision recall f1-score support 0 0.95 1.00 0.97 5157 1 0.91 0.46 0.61 481 accuracy 0.95 5638 macro avg 0.93 0.73 0.79 5638 weighted avg 0.95 0.95 0.94 5638 ▶ f1-score : 0.79 (y=0 : 0.97, y=1 : 0.61)

※ class1의 f1-score 0.5 미만인 모델 제외 결과

데이터 분석

ADASYN, SMOTE 적용 및 Gridsearch 결과

ADASYN

SVM

	precision	recall	f1-score	support
0	0.95	0.99	0.97	5157
1	0.80	0.50	0.61	481
accuracy			0.95	5638
macro avg	0.88	0.74	0.79	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.79

(y=0 : 0.97, y=1 : 0.61)

Kernel SVM

	precision	recall	f1-score	support
0	0.95	0.99	0.97	5157
1	0.77	0.49	0.60	481
accuracy			0.94	5638
macro avg	0.86	0.74	0.78	5638
weighted avg	0.94	0.94	0.94	5638

▶ f1-score : 0.78

(y=0 : 0.97, y=1 : 0.60)

Random Forest

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.80	0.52	0.63	481
accuracy			0.95	5638
macro avg	0.88	0.75	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.63)

SMOTE

SVM

	precision	recall	f1-score	support
0	0.95	0.99	0.97	5157
1	0.81	0.50	0.62	481
accuracy			0.95	5638
macro avg	0.88	0.74	0.79	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.79

(y=0 : 0.97, y=1 : 0.62)

Kernel SVM

	precision	recall	f1-score	support
0	0.95	0.98	0.97	5157
1	0.75	0.49	0.59	481
accuracy			0.94	5638
macro avg	0.85	0.74	0.78	5638
weighted avg	0.94	0.94	0.94	5638

▶ f1-score : 0.78

(y=0 : 0.97, y=1 : 0.59)

Random Forest

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.79	0.52	0.63	481
accuracy			0.95	5638
macro avg	0.88	0.75	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.63)

※ y=1의 f1-score 0.5 미만인 모델 제외 결과

데이터 분석

ADASYN, SMOTE 적용 및 Gridsearch 결과

XGBoost

ADASYN

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.82	0.52	0.64	481
accuracy			0.95	5638
macro avg	0.89	0.75	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.64)

SMOTE

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.81	0.52	0.63	481
accuracy			0.95	5638
macro avg	0.89	0.75	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.63)

Light GBM

ADASYN

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.81	0.52	0.63	481
accuracy			0.95	5638
macro avg	0.88	0.76	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.63)

SMOTE

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.85	0.53	0.65	481
accuracy			0.95	5638
macro avg	0.90	0.76	0.81	5638
weighted avg	0.95	0.95	0.95	5638

▶ f1-score : 0.81

(y=0 : 0.97, y=1 : 0.65)

CatBoost

ADASYN

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.82	0.52	0.64	481
accuracy			0.95	5638
macro avg	0.89	0.75	0.80	5638
weighted avg	0.94	0.95	0.94	5638

▶ f1-score : 0.80

(y=0 : 0.97, y=1 : 0.64)

SMOTE

	precision	recall	f1-score	support
0	0.96	0.99	0.97	5157
1	0.85	0.52	0.65	481
accuracy			0.95	5638
macro avg	0.90	0.76	0.81	5638
weighted avg	0.95	0.95	0.95	5638

▶ f1-score : 0.81

(y=0 : 0.97, y=1 : 0.65)

※ y=1의 f1-score 0.5 미만인 모델 제외 결과

데이터 분석

변수 중요도

< 변수 중요도 >

Feature 9 : 69.340652465820312
Feature 4 : 29.956439971923828
Feature 20 : 13.387453079223633
Feature 1 : 9.878078460693359
Feature 0 : 7.156118869781494

XGBoost

Feature 13: 491
Feature 4: 425
Feature 9: 395
Feature 10: 304
Feature 27: 282

Light GBM

Feature 9: 16.143389929241720
Feature 13: 8.578571584879950
Feature 20: 7.968099604905092
Feature 4: 7.775759399631011
Feature 5: 6.441781492097342

CatBoost

▶ Feature 9 : AL

▶ Feature 4 : ANONYMOUS_1

▶ Feature 13 : CA

▶ Feature 20 : K

최종 모델

SMOTE + CatBoost

< 최종 모델 및 교차검증 >

```
from catboost import CatBoostClassifier, Pool, cv
import matplotlib.pyplot as plt

train_pool = Pool(X_train_std, y_train)

model = CatBoostClassifier(iterations=300, l2_leaf_reg=1, learning_rate=0.1, depth=10, loss_function='Logloss')

# 교차 검증 수행
cv_params = model.get_params()
cv_data = cv(train_pool, cv_params, fold_count=5, plot=True)

# 교차 검증 결과 추출
cv_iterations = range(len(cv_data['test-Logloss-mean']))
train_scores = cv_data['train-Logloss-mean']
test_scores = cv_data['test-Logloss-mean']
```

- ▶ SMOTE를 이용한 oversampling + CatBoost
- ▶ iterations=300, l2_leaf_reg=1, learning_rate=0.1, depth=10, loss_function='Logloss'
- ▶ f1-score : 0.81 (y=0 : 0.97, y=1 : 0.65)

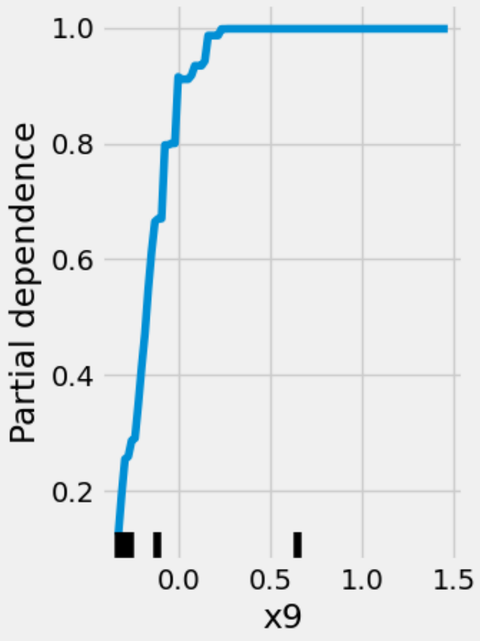
최종 모델 - CatBoost

Partial Dependence

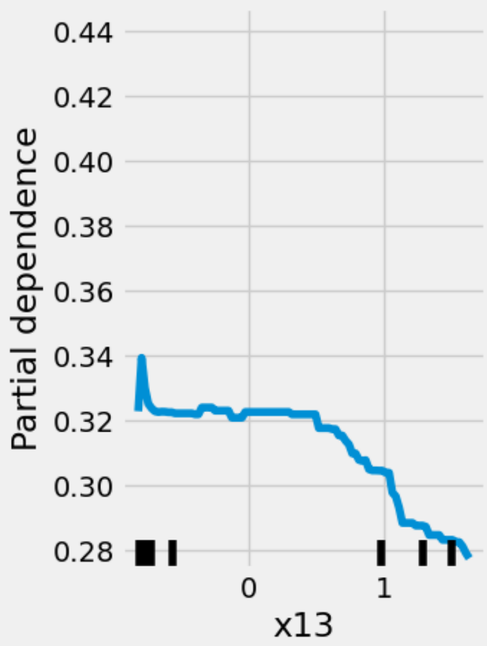
```
Feature 9: 16.14338992924172
Feature 13: 8.57857158487995
Feature 20: 7.9680996049050 92
Feature 4: 7.7775939963101 16
```

< 부분 의존성 >

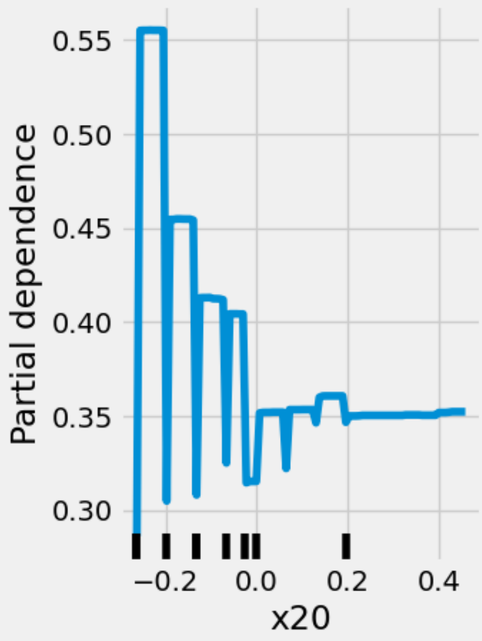
AL



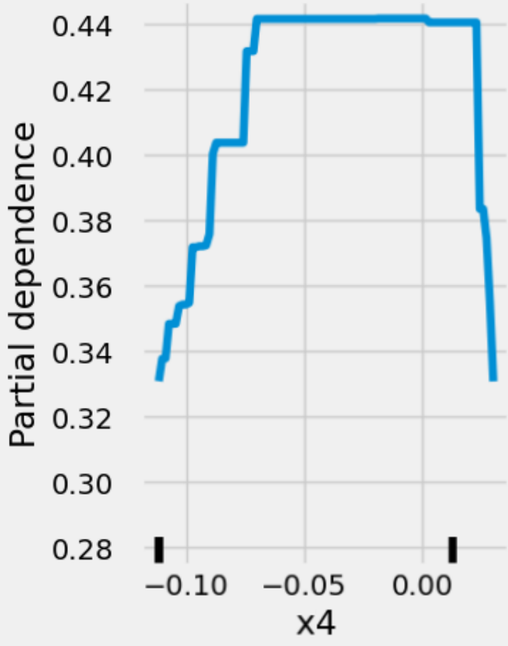
Ca



K



ANONYMOUS_1



4. 결론

인사이트

결론

1. f1-score 향상 : 0.78 → 0.81 / y=1 : 0.59 → 0.65 (6% 증가)
2. 다양한 모델 적합, 오버샘플링 이용
3. GridSearch를 통한 하이퍼파라미터 탐색

한계

1. feature가 너무 많고 유의미한 변수가 적어서 모델 적용 어려움
2. 불균형데이터 → 클래스 별 성능 차이로 인한 모델 편향성 개선 시도

감사합니다.

건설기계 작동오일 상태판단 모델 개발

윤혜준, 홍서연