

27기 여름방학세미나

1팀
이수정
이은서
이정우
이진모
조혜현

INDEX

1. Intro

2. EDA

3. Process

4. 최종 선정 모델

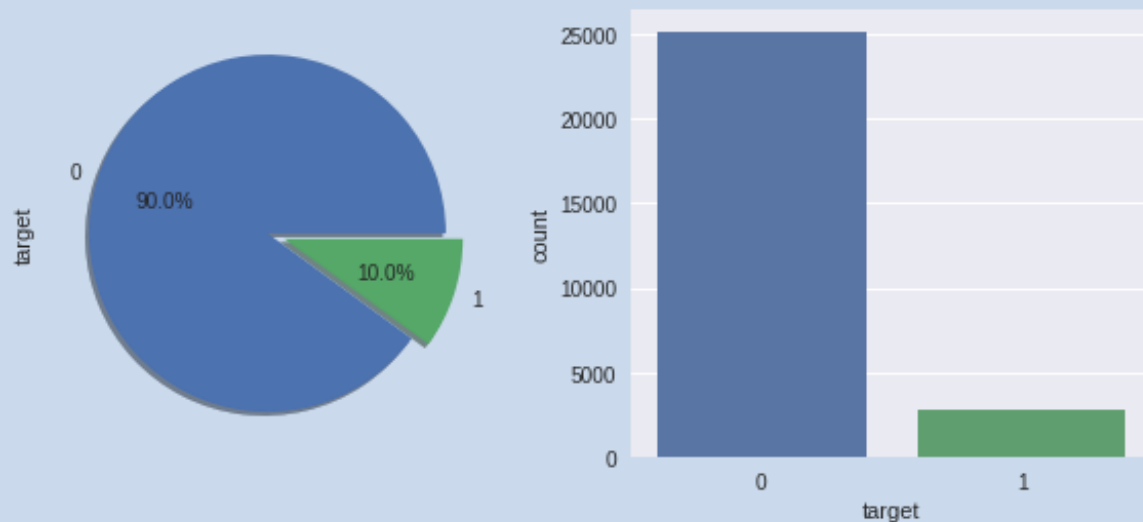
5. 세미나를 마치며

1

Intro

문제 상황

1. 종속 변수
클래스 불균형

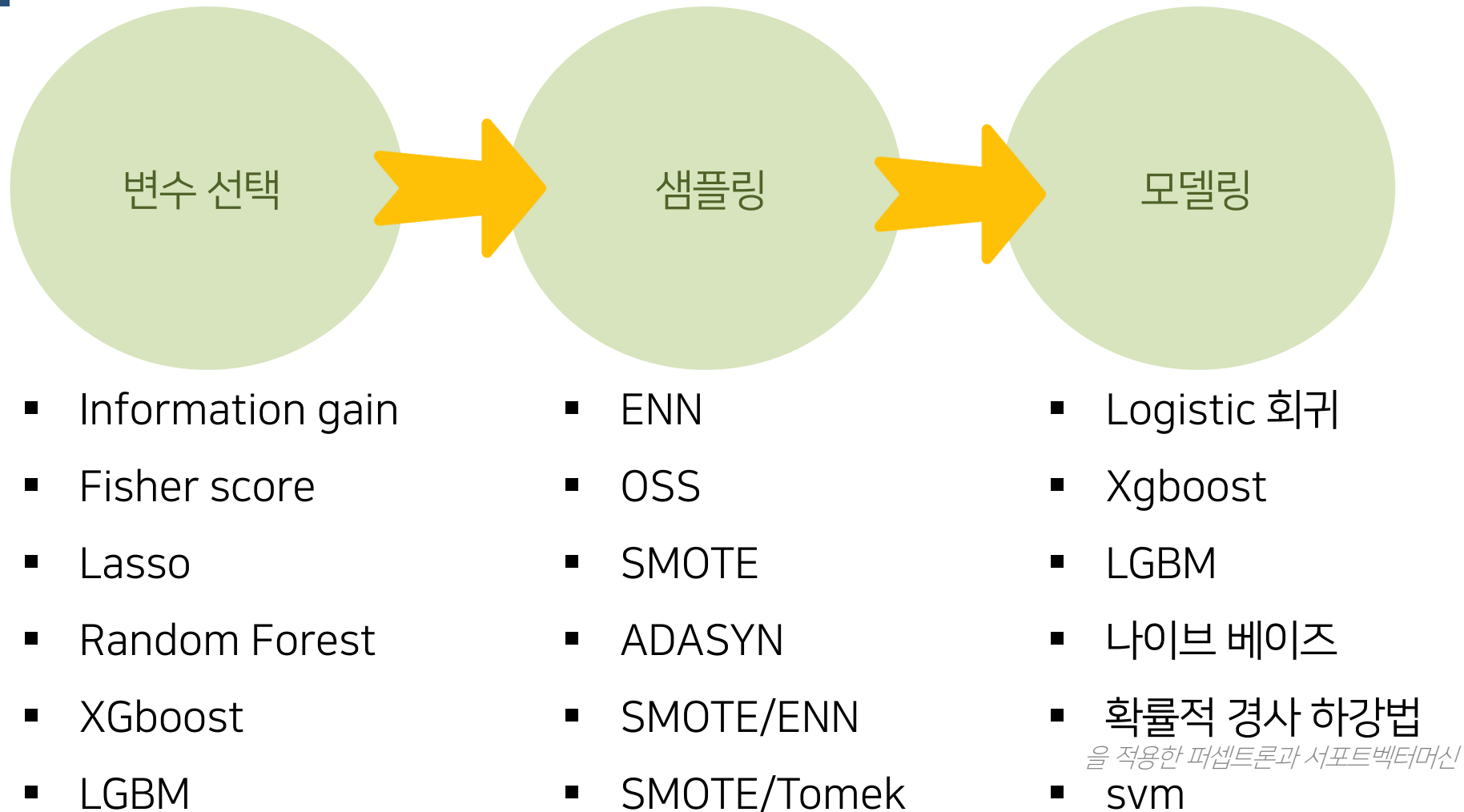


2. 독립 변수 多

| | target | var_0 | var_1 | var_2 | var_3 | var_4 | var_5 | var_6 | var_7 | var_8 | var_9 | var_10 |
|---|--------|---------|---------|---------|--------|---------|----------|--------|---------|---------|--------|---------|
| 0 | 0 | 5.0702 | -0.5447 | 9.5900 | 4.2987 | 12.3910 | -18.8687 | 6.0382 | 14.3797 | -0.4711 | 7.3198 | 4.6603 |
| 1 | 1 | 16.3699 | 1.5934 | 16.7395 | 7.3330 | 12.1450 | 5.9004 | 4.8222 | 20.9729 | 1.1064 | 8.6978 | 2.3287 |
| 2 | 0 | 5.0615 | 0.2689 | 15.1325 | 3.6587 | 13.5276 | -6.5477 | 5.2757 | 9.8710 | 2.5569 | 9.4701 | -7.4401 |
| 3 | 0 | 8.4199 | -1.8128 | 8.1202 | 5.3955 | 9.7184 | -17.8390 | 4.0959 | 15.2860 | 1.9016 | 7.0967 | -9.0265 |
| 4 | 0 | 8.2703 | -5.6854 | 12.6862 | 7.2755 | 12.3713 | -7.7521 | 6.7252 | 18.4270 | -2.7730 | 7.7828 | 1.5081 |

5 rows × 201 columns

예측 분류 흐름



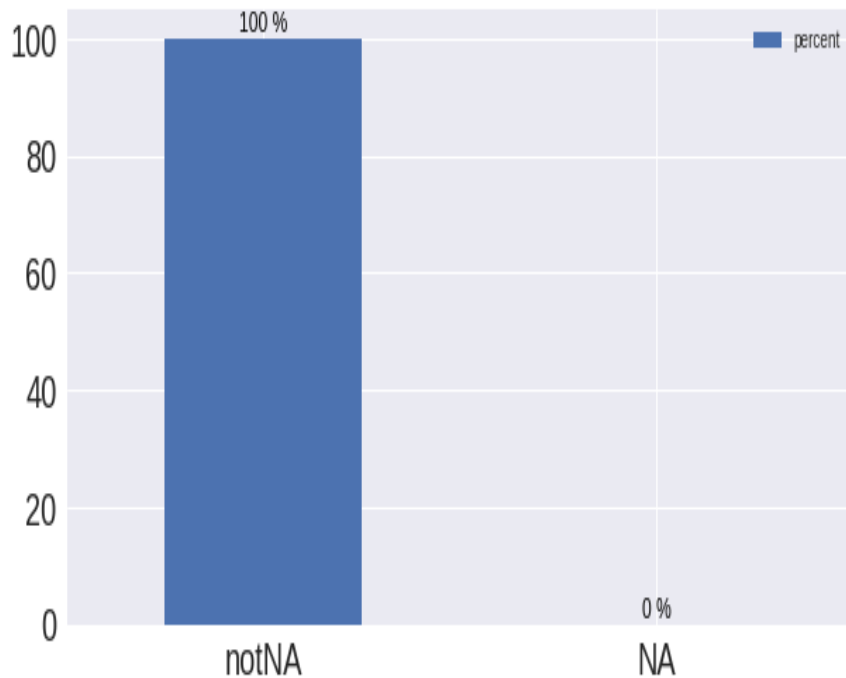
2

EDA

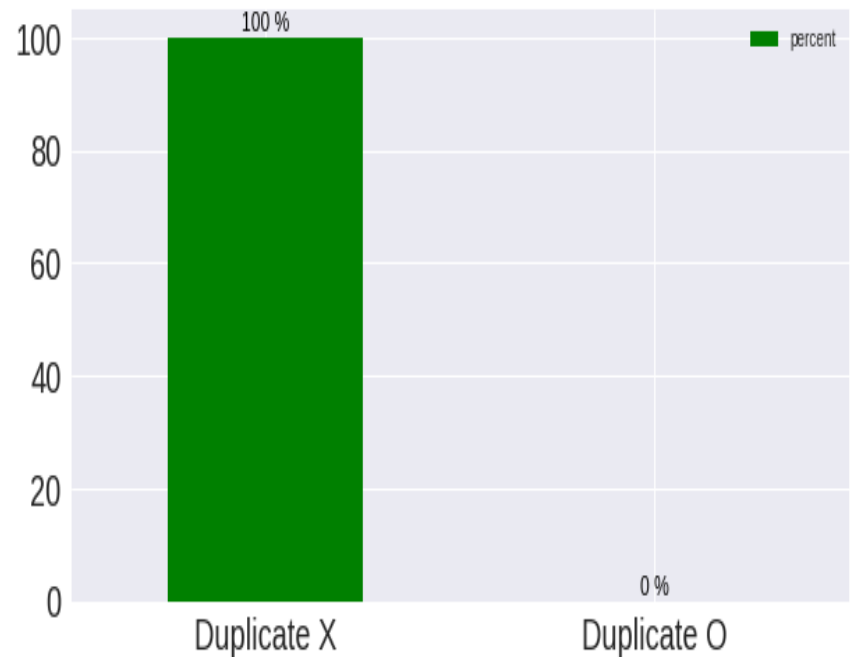
데이터 시각화

데이터: 익명의 200개 변수로 이루어진 거래고객 예측에 대한 정보

1. 결측값 확인

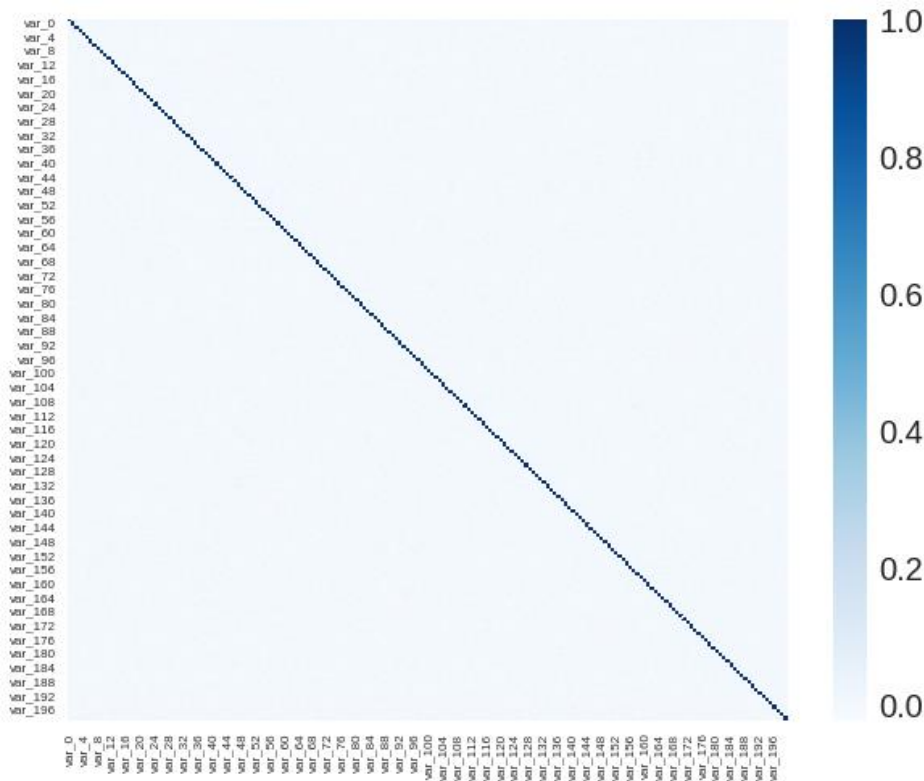


2. 중복데이터 확인

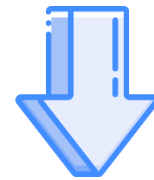


데이터 시각화

3. 변수 간 상관계수



독립변수 간 상관계수가
모두 0.1을 넘지 않음



유의미한 상관관계를 가지는
변수들이 없으므로

PCA를 통한 차원 축소 **부적절**

데이터 시각화

여기서 잠깐!!

3. 변수 간 상관관계



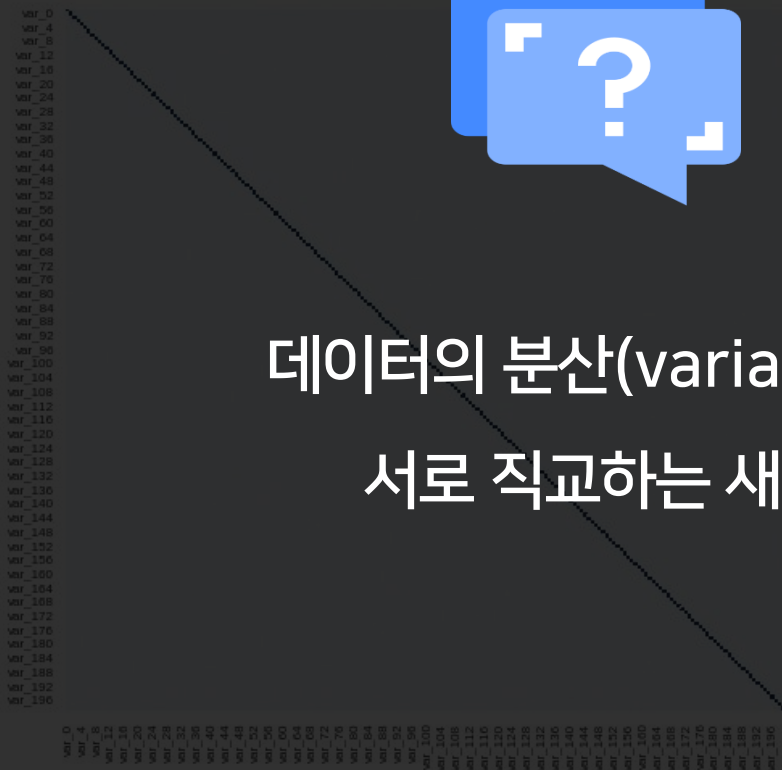
PCA란?

독립변수 간 상관관계수가
모두 0.1을 넘지 않음

데이터의 분산(variance)을 최대한 보존하면서
서로 직교하는 새 기저를 찾아 차원 축소

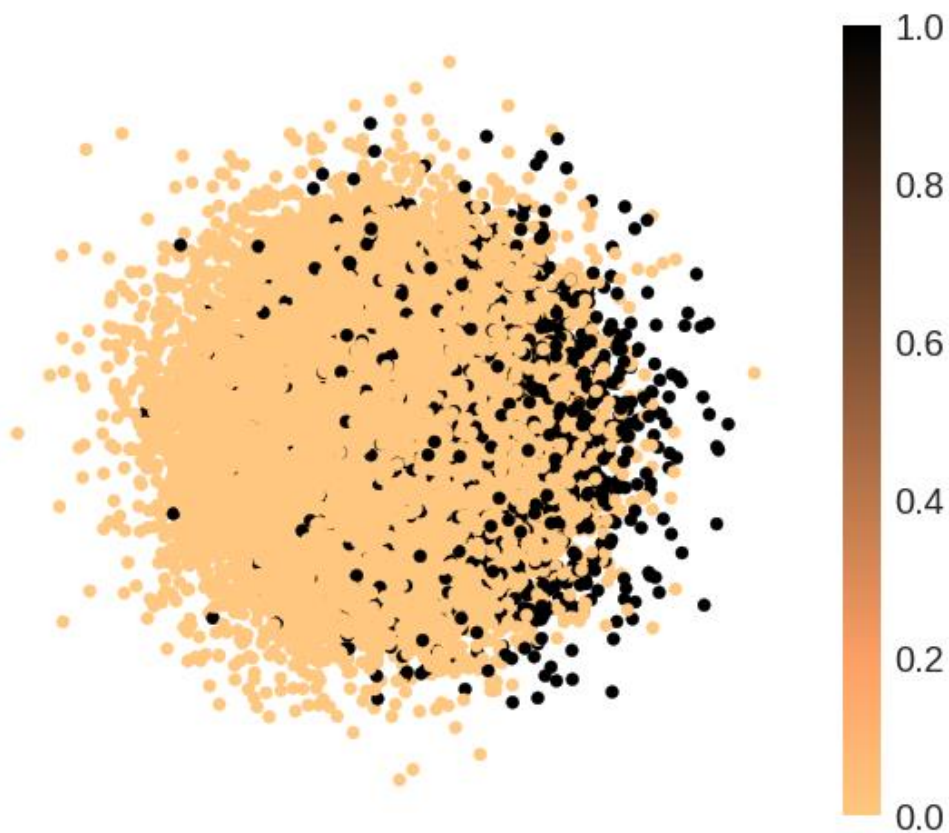
의미한 상관관계를 가지는
변수들이 없으므로

PCA를 통한 차원 축소 불가능

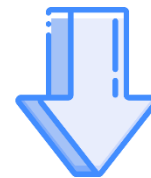


1.0
0.8
0.6
0.4
0.2
0.0

데이터 시각화



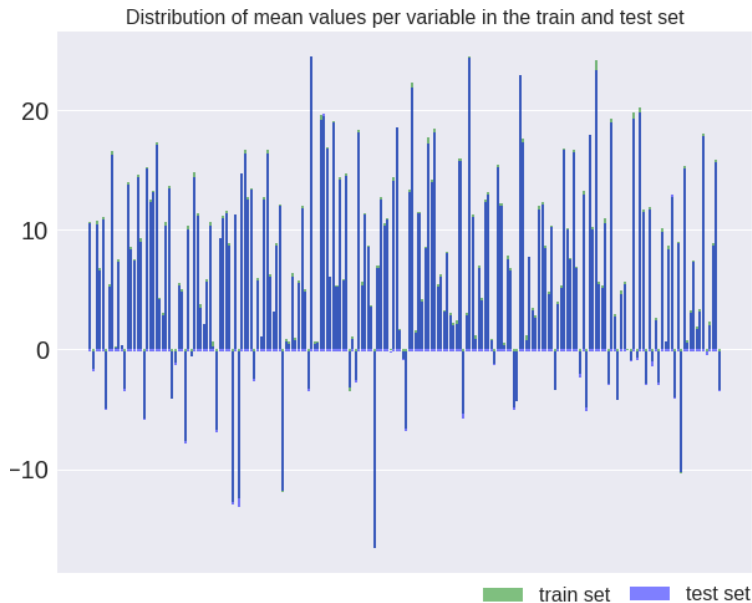
주성분 2개를 선택한 PCA 산포도



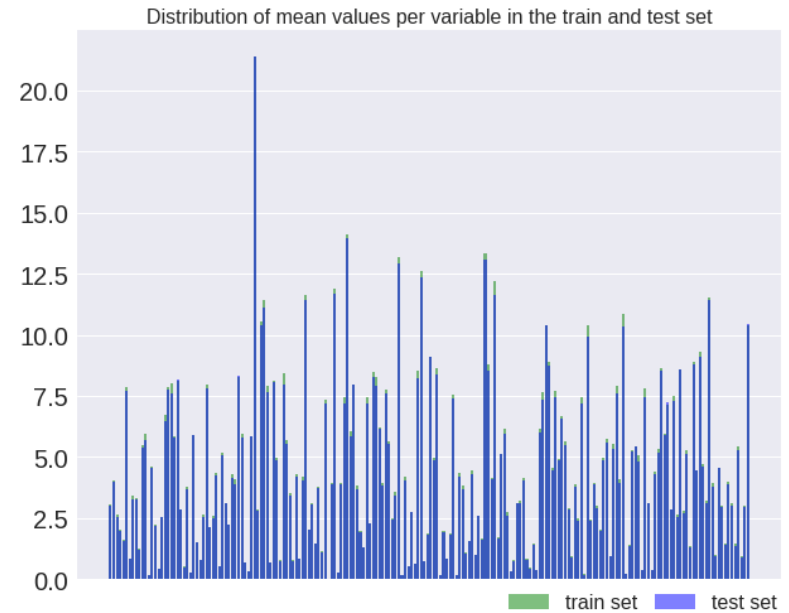
패턴이 보이지 않아 차원축소
방법으로 사용하기 **어려움**

데이터 시각화

4. train set과 test set의 분포의 동일성 확인



변수별 평균 분포



변수별 표준편차 분포

3

Process

변수선택 필요성

| var_0 | var_1 | var_2 | ... | var_197 | var_198 | var_199 |
|---------|---------|---------|-----|---------|---------|----------|
| 5.0702 | -0.5447 | 9.59 | | 9.2553 | 14.2914 | -7.6652 |
| 16.3699 | 1.5934 | 16.7395 | | 9.0419 | 15.6064 | -10.8529 |
| 5.0615 | 0.2689 | 15.1325 | | 8.6367 | 20.2548 | 11.1524 |

Train data: row 약 3만개, column 200개 (고차원 데이터)

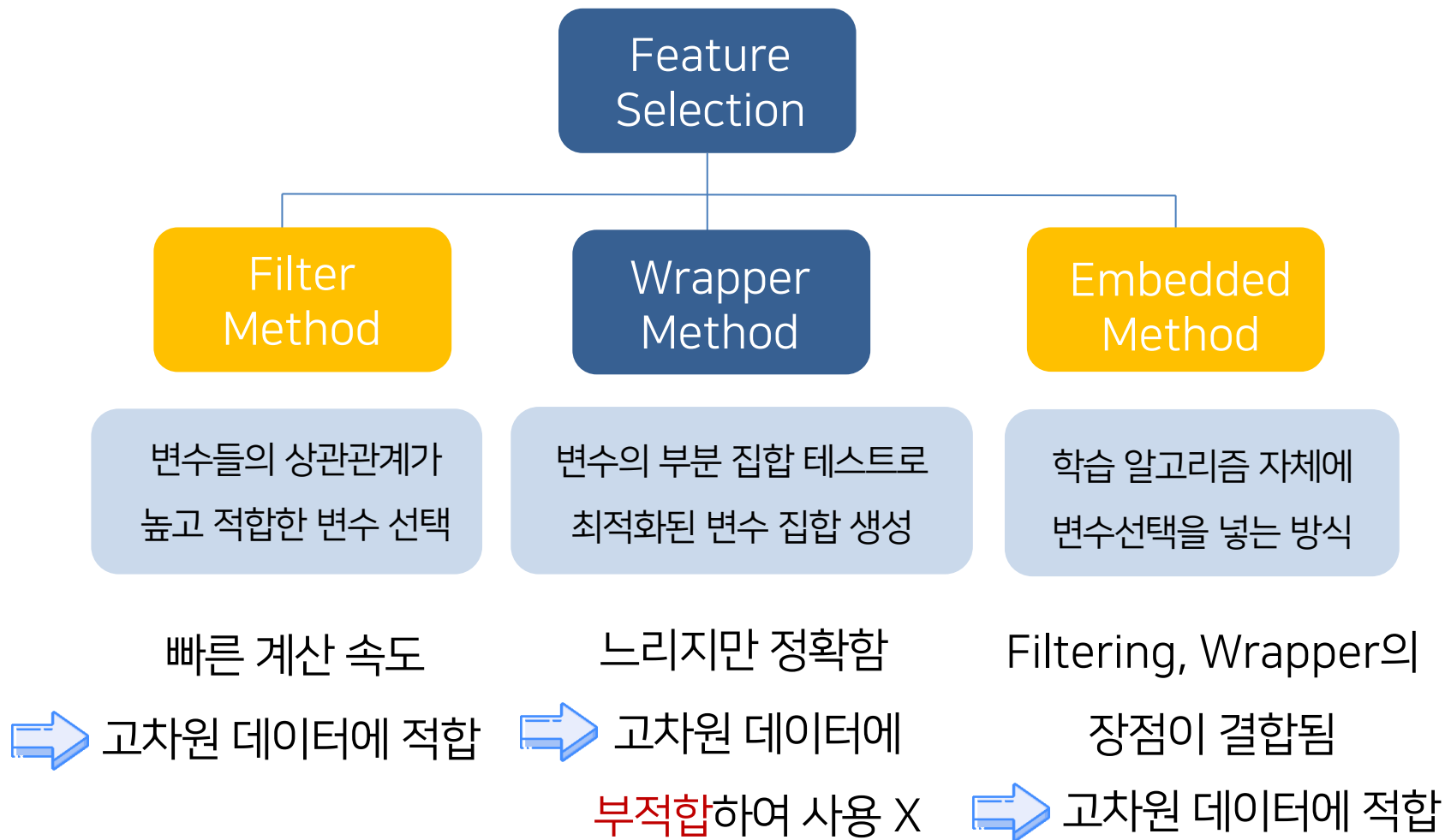
변수 ↑, 계산 복잡도 ↑, 시간 ↑

불필요한 변수 포함하여 계산하는 것은 비효율적임



Feature Selection (중요한 일부 변수만 사용)

변수선택 종류



Information Gain: Filter Method

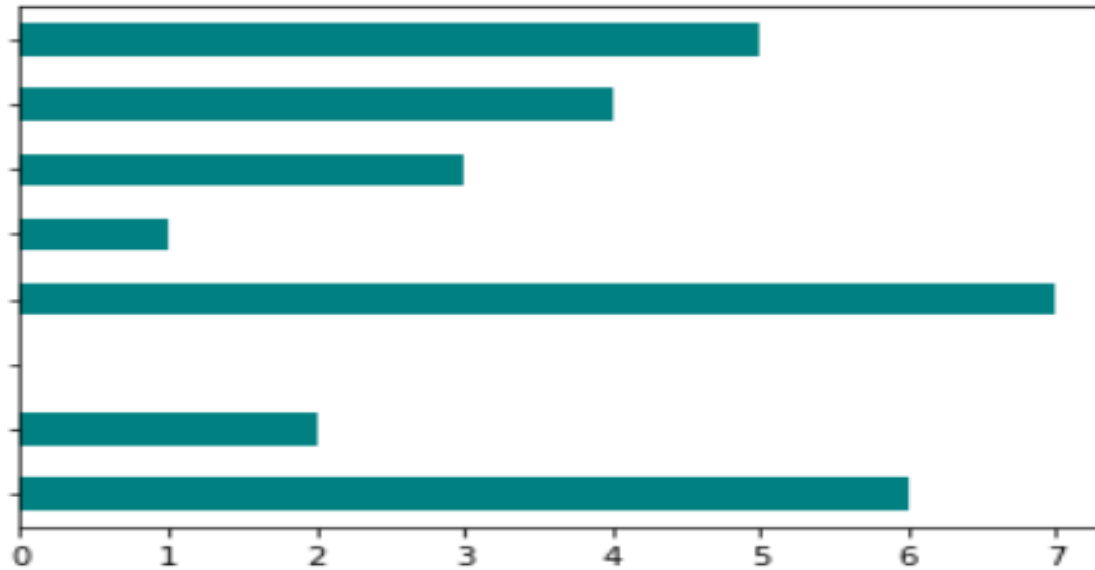
| target | var_81 | var_76 | var_139 | var_190 | var_164 |
|--------|---------|---------|---------|---------|---------|
| 0 | 15.1476 | -1.6632 | 15.7759 | 7.2780 | -9.9638 |
| 1 | 12.1677 | 18.8226 | 1.4873 | 7.4002 | -7.0936 |
| 0 | 9.6918 | 10.9165 | 16.8975 | 0.2717 | 0.0044 |

* IG를 통한 선택한 변수 5개

- 어떤 분류에 대해 얼마나 정보에 대한 이득이 생겼나 판단 (IG ↑ 변별력 ↑)

* 같은 모델에서 IG 변수 선택한 것보다 Full data 쓴 게 성능이 좋은 경우 많았음

Fisher score: Filter Method

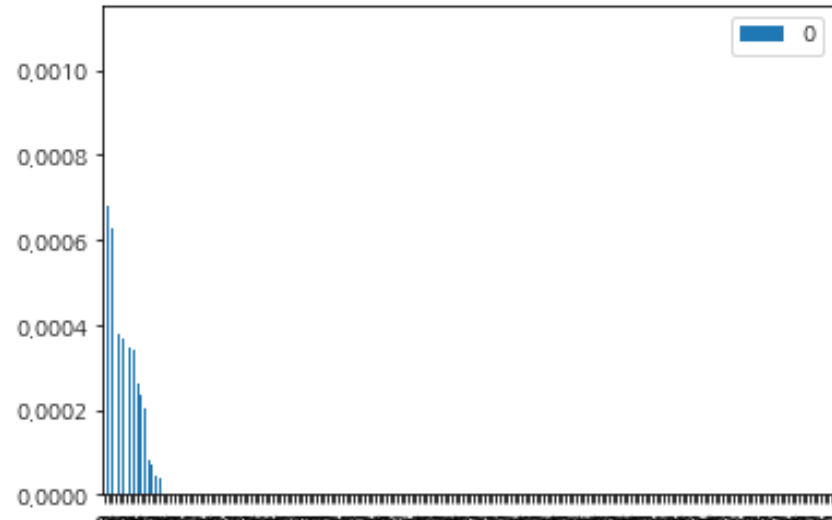
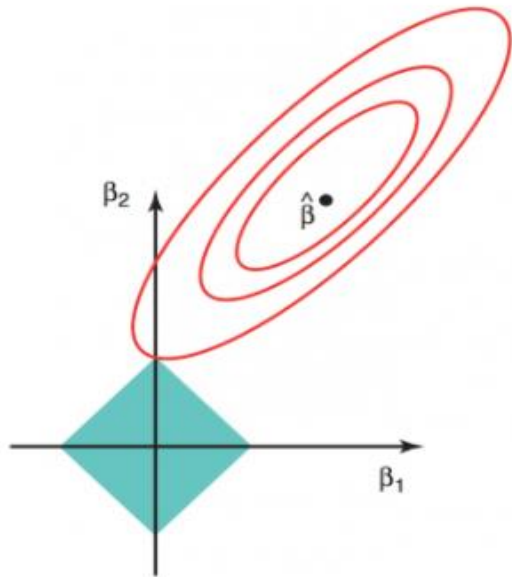


- 가장 흔하게 사용되는 변수 선택법
 - 로그 우도 함수 미분의 gradient
- Fisher score를 기준으로 변수 순위를 반환하여 변수 선택



램 터짐..

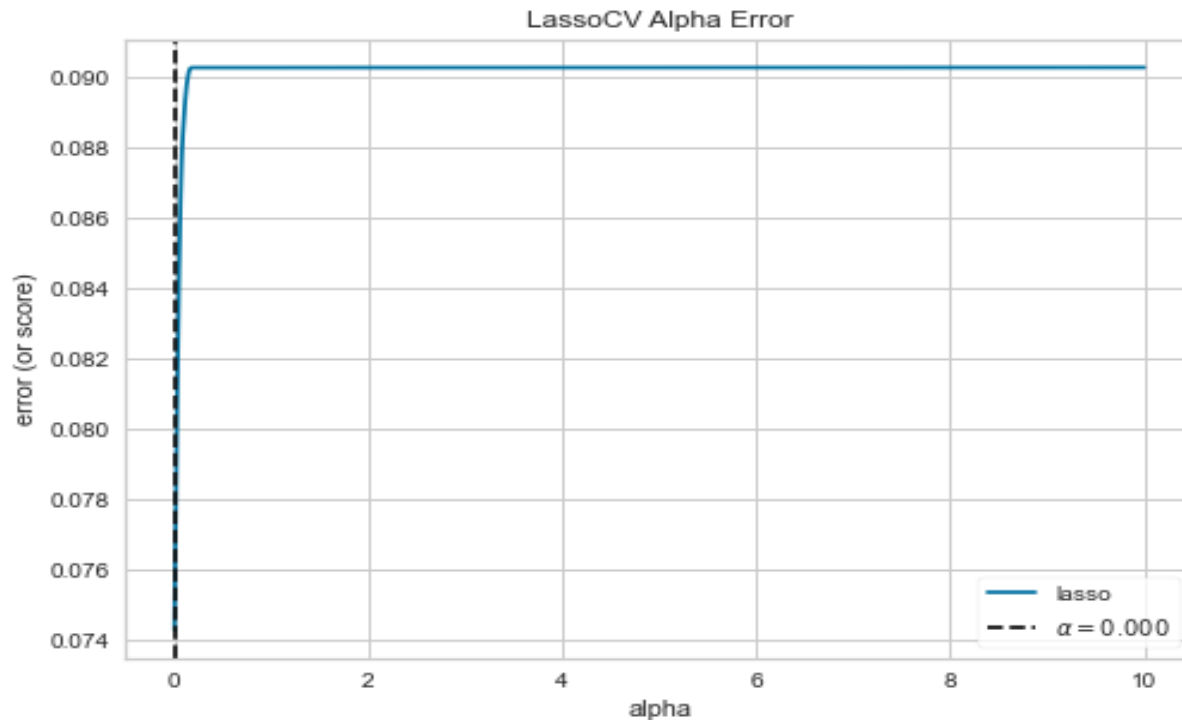
Lasso: Embedded Method



* Lasso로 선택한 변수 16개

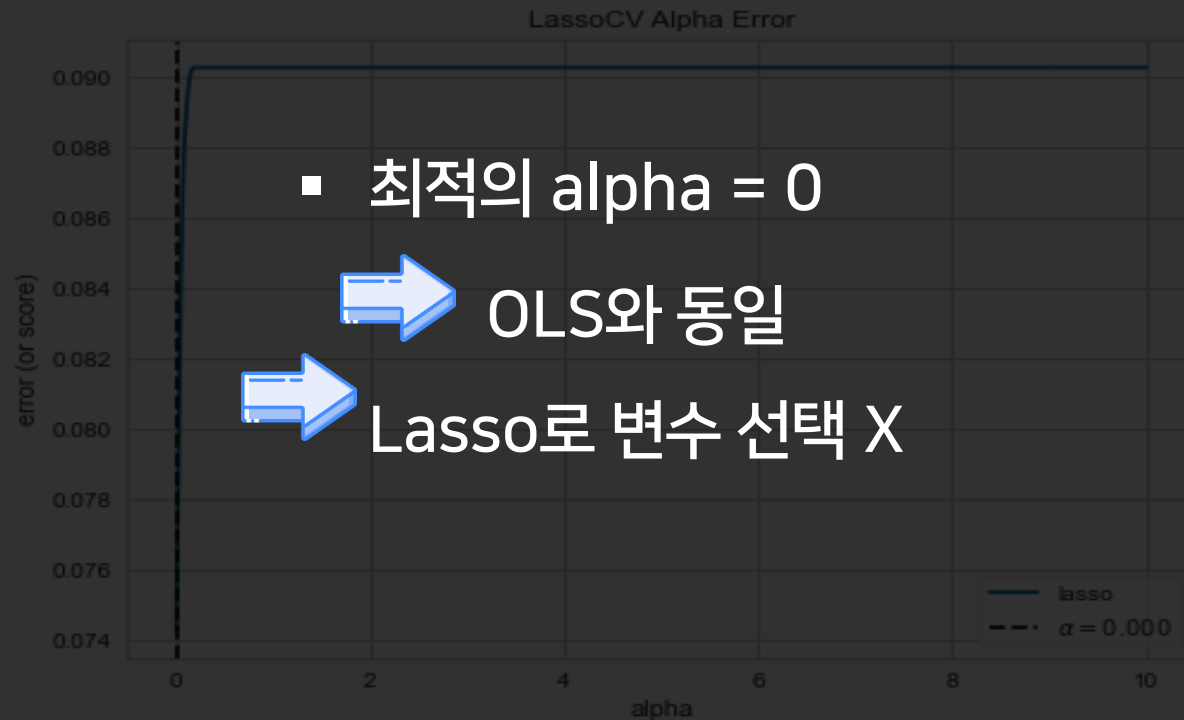
- 회귀 모형 최적화할 때 오차 & 회귀계수의 절대값 최소화
- 회귀계수가 0으로 수렴하면 일부 변수 제외되어 변수선택 가능

Lasso: Embedded Method



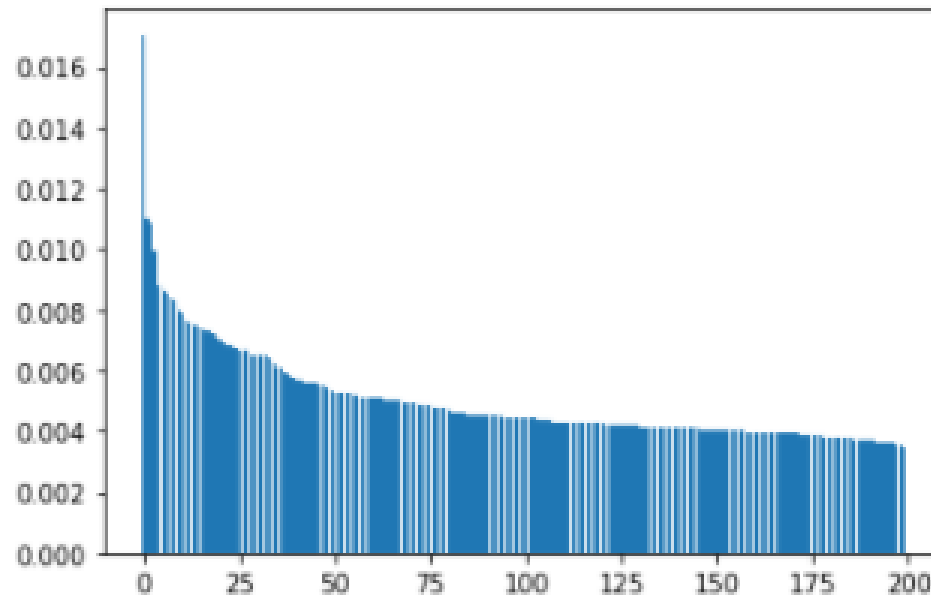
- Yellow Brick: alpha 변화에 따른 모델을 보여주어 최적의 alpha를 제시하는 모듈
- alpha: 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터

Lasso: Embedded Method



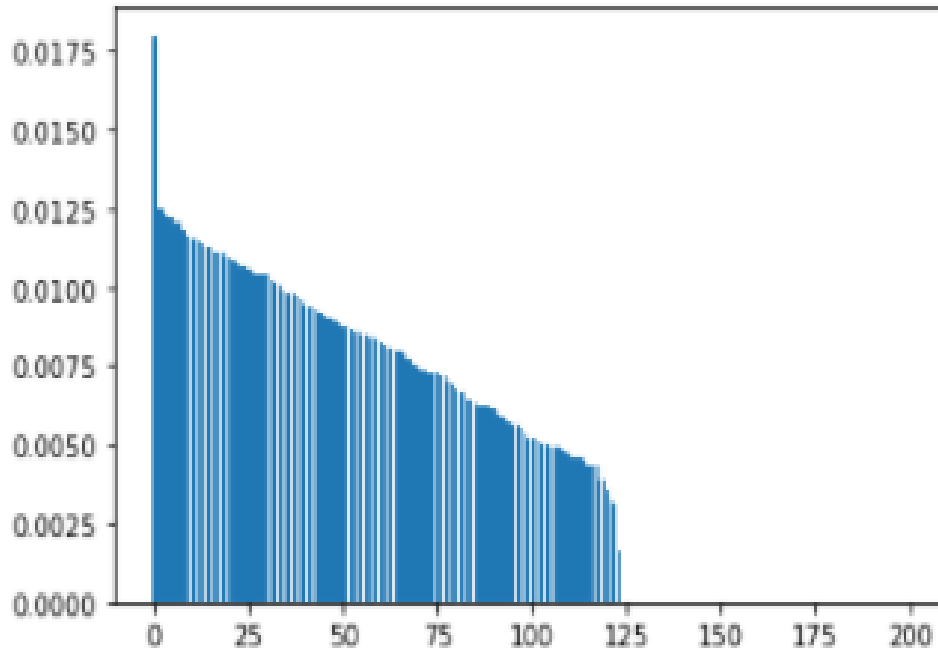
- Yellow Brick: α 변화에 따른 모델을 보여주어 최적의 α 를 제시하는 모듈
- α : 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터

Random Forest: Embedded Method



- 불순도 지표로 Gini 계수 사용하여 불순도를 가장 크게 감소시키는 변수 선택
- 특성 중요도가 임계값 0.004 이상인 변수 157개 선택됨

Xgboost: Embedded Method



다양한 방법의 **importance** 측정

① Gain/total gain (default)

해당 변수가 모델 예측에 미친 영향 측정



변수 124개 선택됨

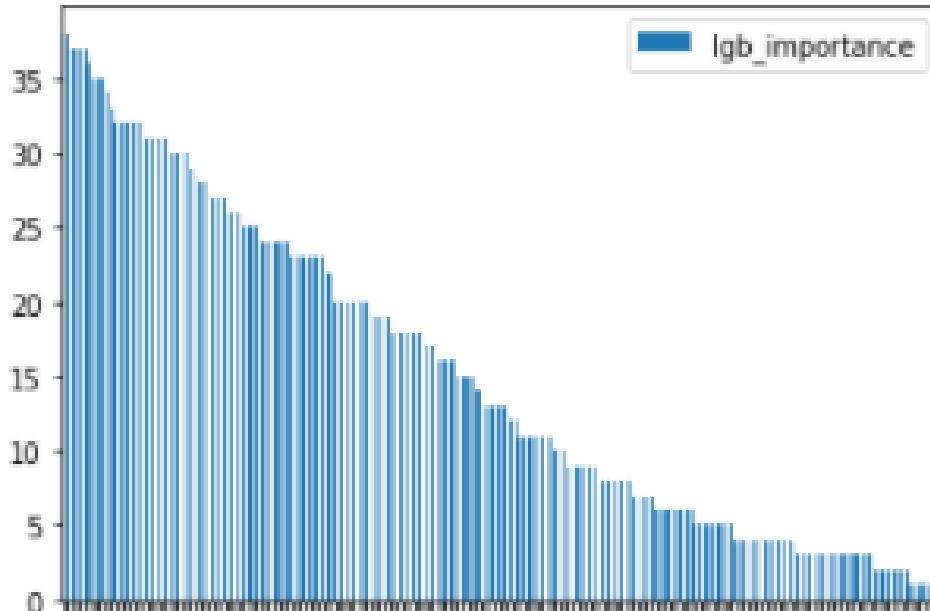
② Cover/ total cover

해당 변수와 관련 있는 샘플의 상대적 개수

③ Weight

해당 변수가 노드 분기에 사용된 횟수

LGBM: Embedded Method



다양한 방법의 **importance** 측정

① Gain

변수가 트리에서 사용될 때

평균 이득 계산



변수 185개 선택됨

② Split

모든 트리에서 데이터를 분할하는데

변수가 사용되는 횟수

변수선택에 대한 총평

- 변수의 개수가 지나치게 적으면 성능이 나쁨
ex) information gain 5개 변수, lasso 16개 변수
- 성능이 낮은 Information gain, Lasso 선택 데이터는
세미나 기간 후반에는 거의 사용하지 않음

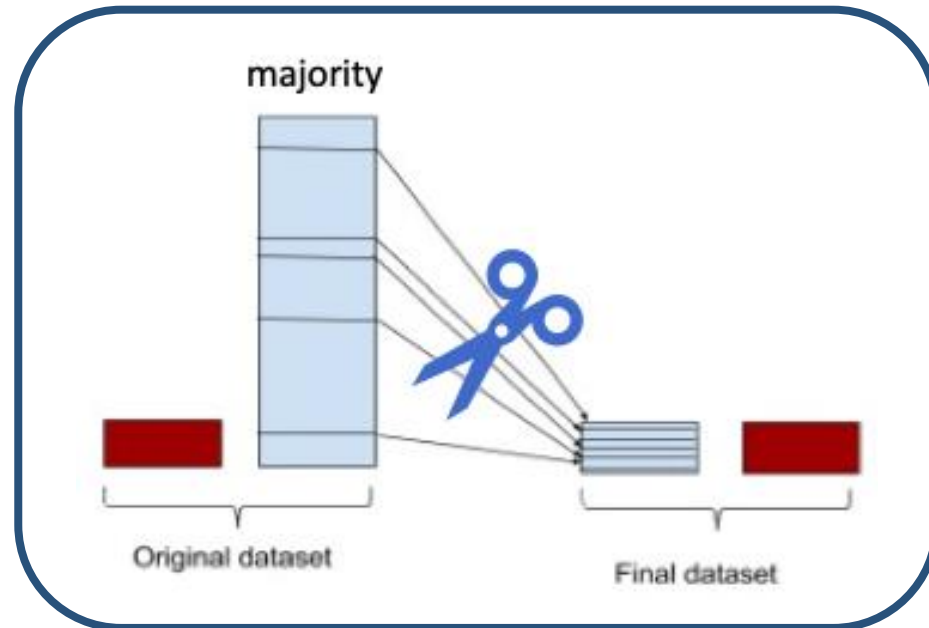
- XGBoost로 변수 선택한 데이터, RF로 변수 선택한 데이터,
LGBM으로 변수 선택한 데이터, 모든 변수 데이터를 사용하게 됨



그러나 학습하는 모델과 파라미터, 적용한 샘플링 방법에 따라
데이터셋마다 성능이 들쭉날쭉 해 다양한 조합을 시도했음

Sampling

Under Sampling

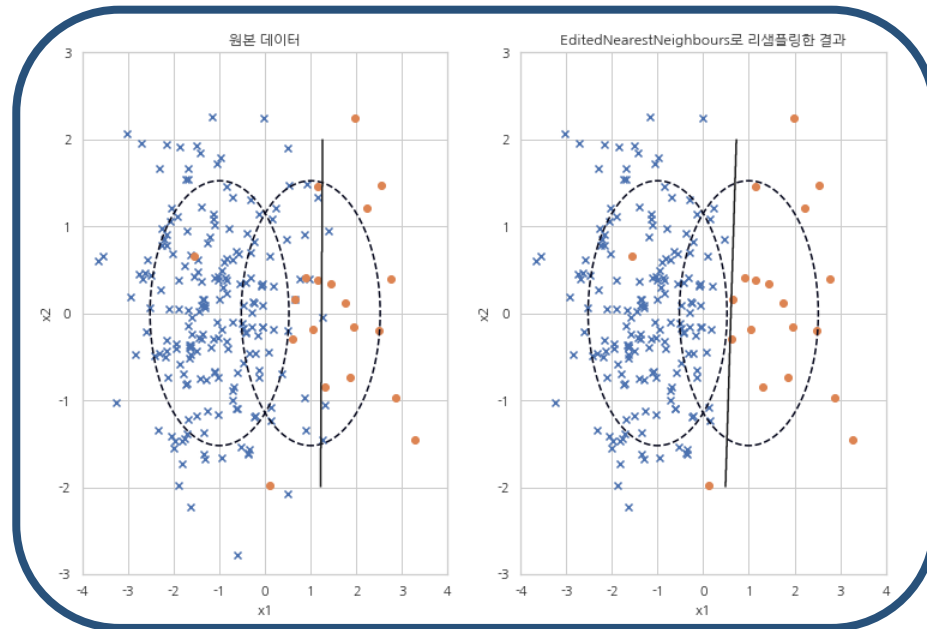


- 분포가 큰 클래스를 낮은 분포의 클래스 크기에 맞춰주는 샘플링 방법
 - 장점: 유의미한 데이터만을 남길 수 있음
 - 단점: 정보가 유실되는 문제가 생길 수 있음
ex) ENN, OSS, Random Under Sampling, etc.

Sampling

Under Sampling

ENN (Edited Nearest Neighbors)

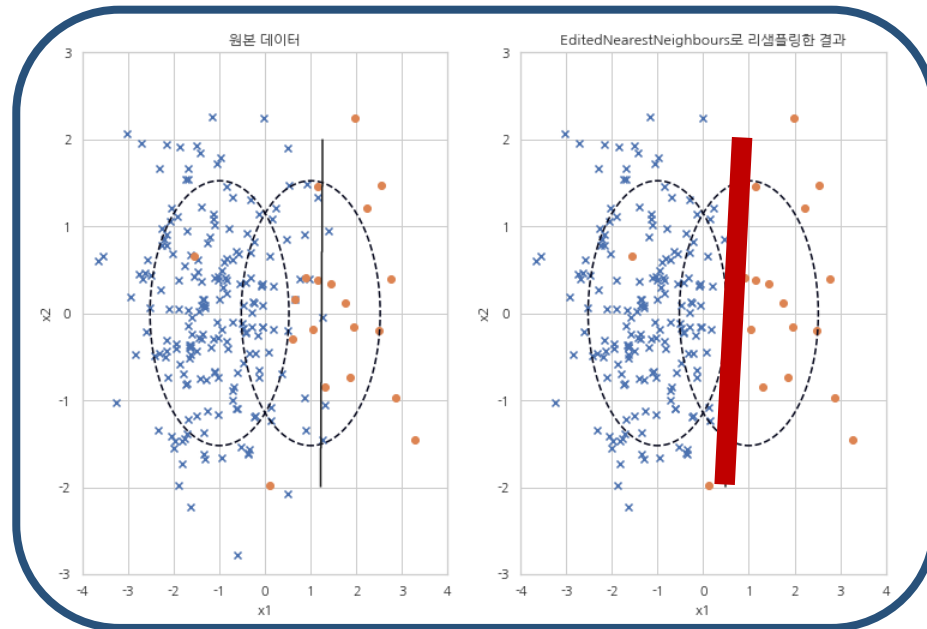


- 소수 클래스 주변의 다중 클래스 값을 제거하는 방법
- 클래스를 구분하는 임계점을 다중 클래스 쪽으로 밀어낼 수 있지만 제거 효과가 크지 않음

Sampling

Under Sampling

ENN (Edited Nearest Neighbors)

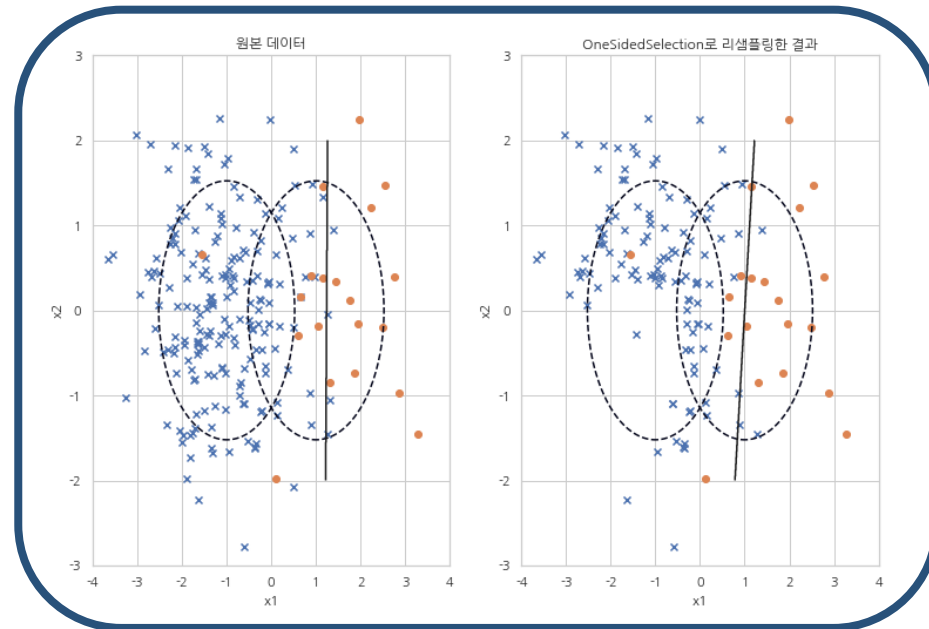


- 소수 클래스 주변의 다중 클래스 값을 제거하는 방법
- 클래스를 구분하는 임계점을 다중 클래스 쪽으로 밀어낼 수 있지만 제거 효과가 크지 않음

Sampling

Under Sampling

OSS (One Sided Selection)

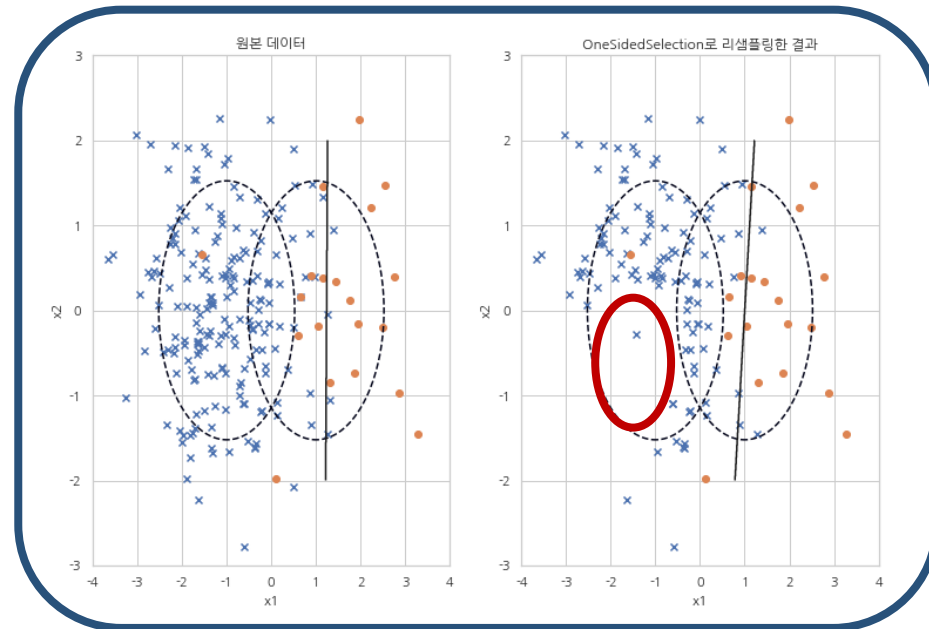


- 데이터가 많은 클래스에서 서로 다른 클래스에 속하며 가까운 한 쌍의 데이터(x_+ , x_-)인 Tomek's link에 해당하는 데이터를 제거함
 - 나머지 데이터에 대해서는 Condensed Nearest Neighbor를 이용하여 밀집된 데이터를 제거함

Sampling

Under Sampling

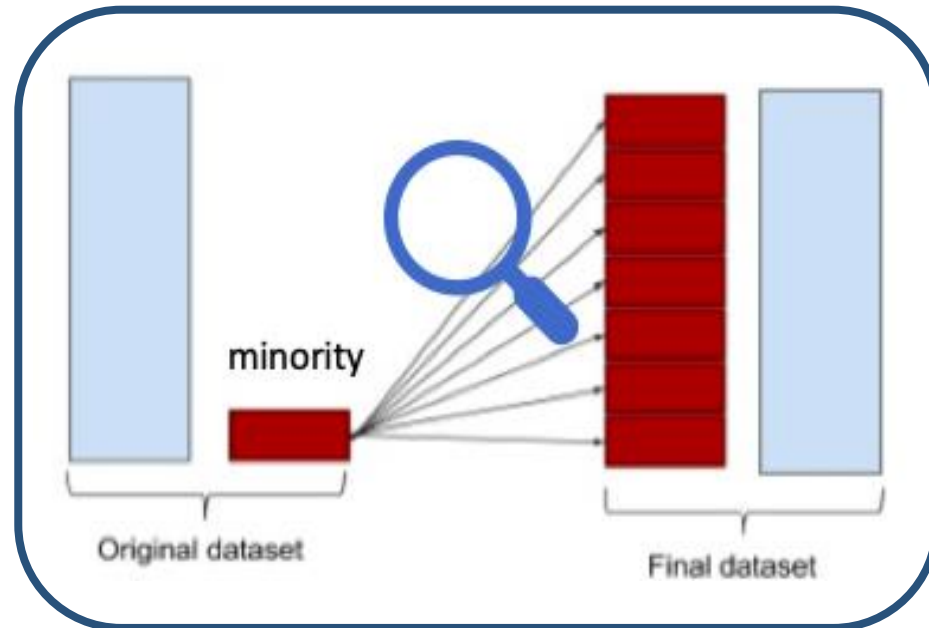
OSS (One Sided Selection)



- 데이터가 많은 클래스에서 서로 다른 클래스에 속하며 가까운 한 쌍의 데이터(x_+ , x_-)인 Tomek's link에 해당하는 데이터를 제거함
 - 나머지 데이터에 대해서는 Condensed Nearest Neighbor를 이용하여 밀집된 데이터를 제거함

Sampling

Over Sampling

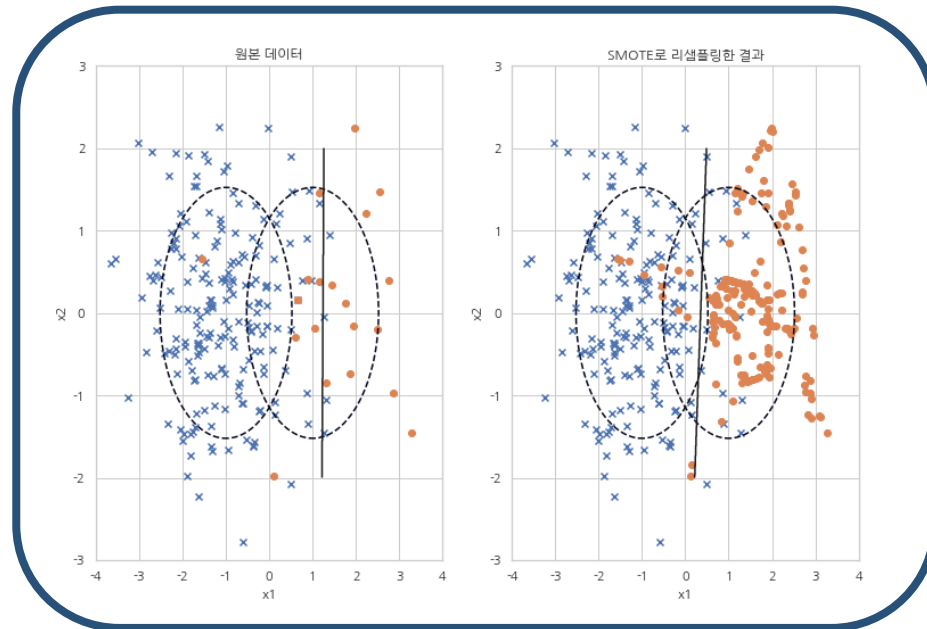


- 분포가 작은 클래스를 **큰 분포의 클래스 크기로 맞춰주는** 샘플링 방법
 - 장점: 정보의 손실을 막을 수 있음
- 단점: 여러 유형의 관측치를 다수 추가하기 때문에 Over-fitting 야기함
ex) SMOTE, ADASYN, Random Over Sampling, etc.

Sampling

Over Sampling

SMOTE

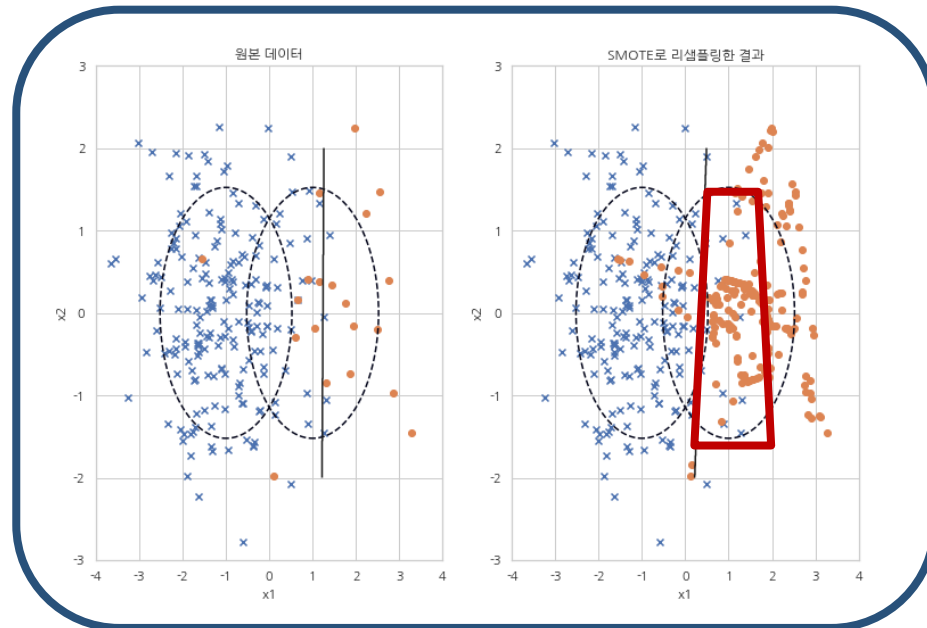


- 임의의 소수 클래스 데이터로부터 인근 소수 클래스 사이에 새로운 데이터를 생성하는 방법
 - 생성된 데이터를 소수 클래스로 분류함

Sampling

Over Sampling

SMOTE

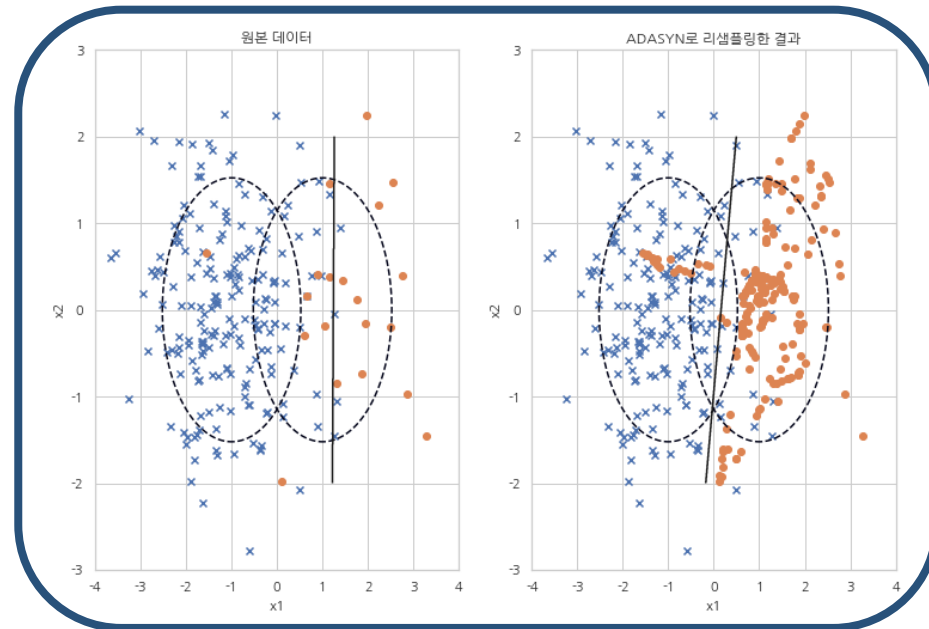


- 임의의 소수 클래스 데이터로부터 인근 소수 클래스 사이에 새로운 데이터를 생성하는 방법
 - 생성된 데이터를 소수 클래스로 분류함

Sampling

Over Sampling

ADASYN

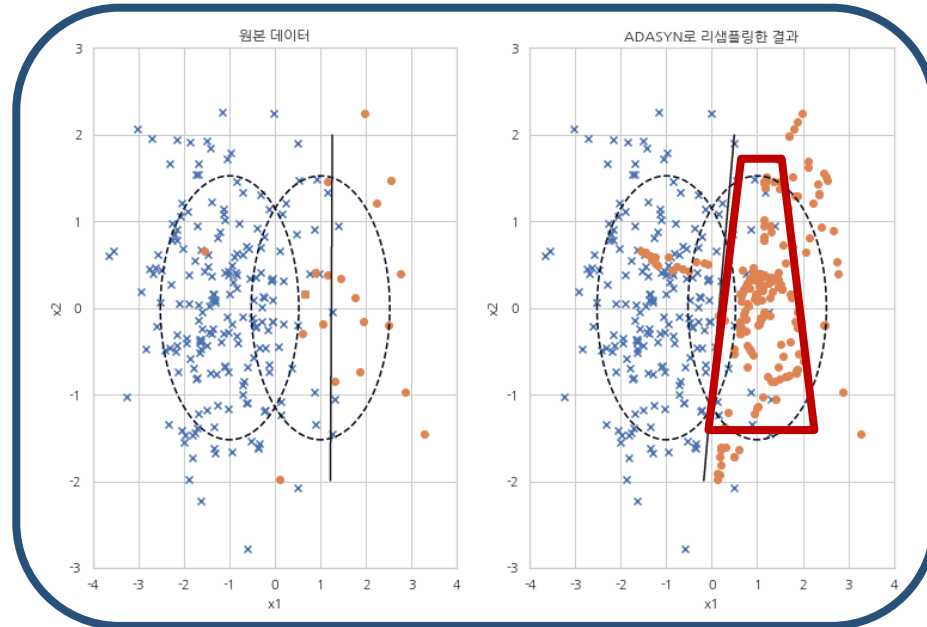


- SMOTE를 발전시켜, 주위 데이터의 분포에 따라 발생시킬 데이터의 수를 조절한 방법
 - 주로 경계값 근처에서 소수 클래스 데이터를 만듦

Sampling

Over Sampling

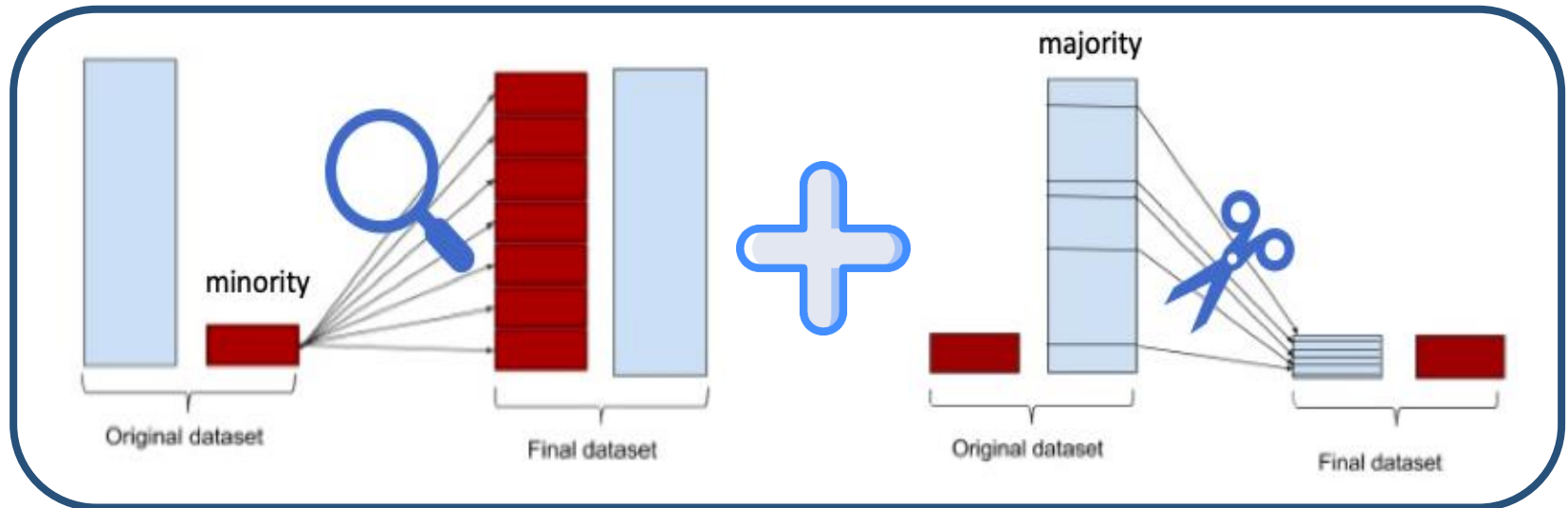
ADASYN



- SMOTE를 발전시켜, 주위 데이터의 분포에 따라 발생시킬 데이터의 수를 조절한 방법
 - 주로 경계값 근처에서 소수 클래스 데이터를 만듦

Sampling

Combine Sampling



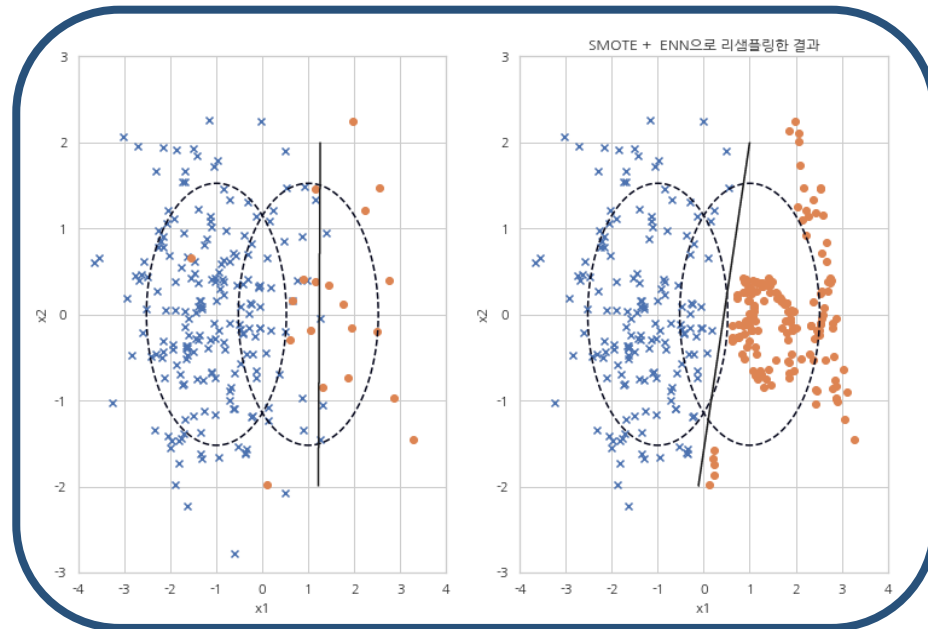
Over Sampling과 Under Sampling을 결합한 샘플링 방법

ex) SMOTE + ENN, SMOTE + Tomek

Sampling

Combine Sampling

SMOTE + ENN

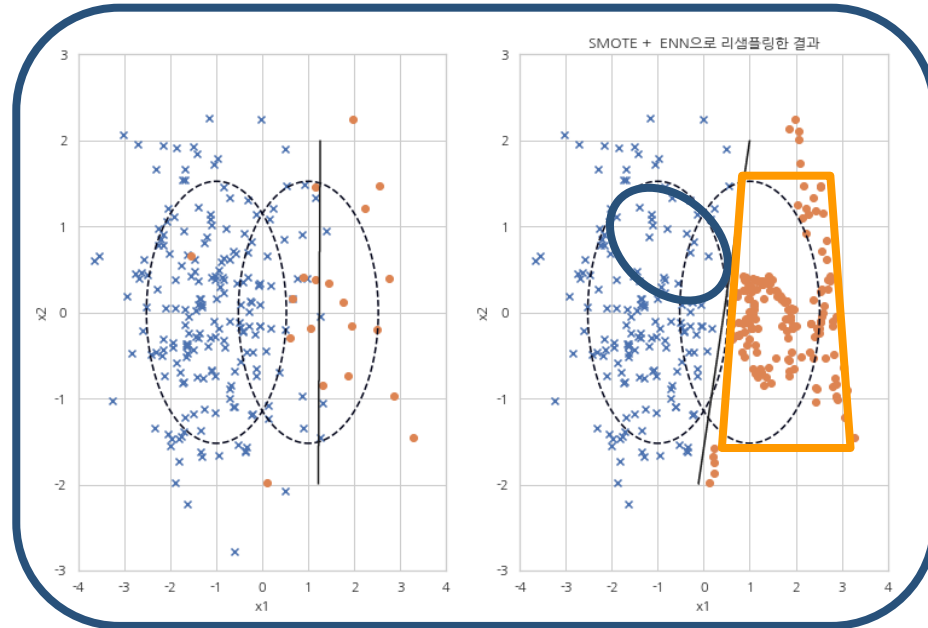


- SMOTE를 적용하여 Over Sampling 한 이후,
ENN을 사용하여 Under Sampling 하는 방법
- SMOTE sampling된 데이터에서 소수 클래스 주변의
다중 클래스 값을 제거함

Sampling

Combine Sampling

SMOTE + ENN

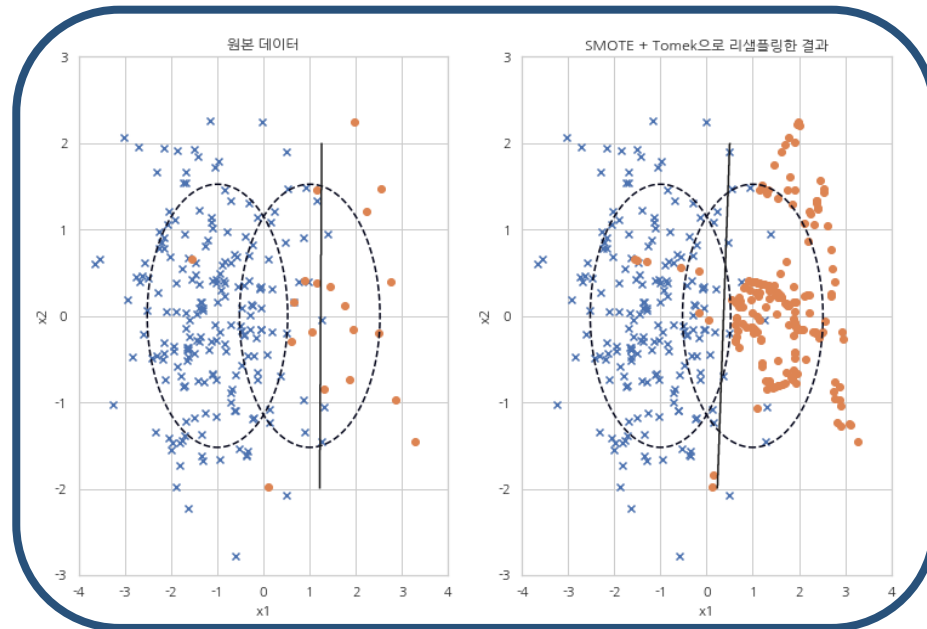


- SMOTE를 적용하여 Over Sampling 한 이후,
ENN을 사용하여 Under Sampling 하는 방법
- SMOTE sampling된 데이터에서 소수 클래스 주변의
다중 클래스 값을 제거함

Sampling

Combine Sampling

SMOTE + Tomek

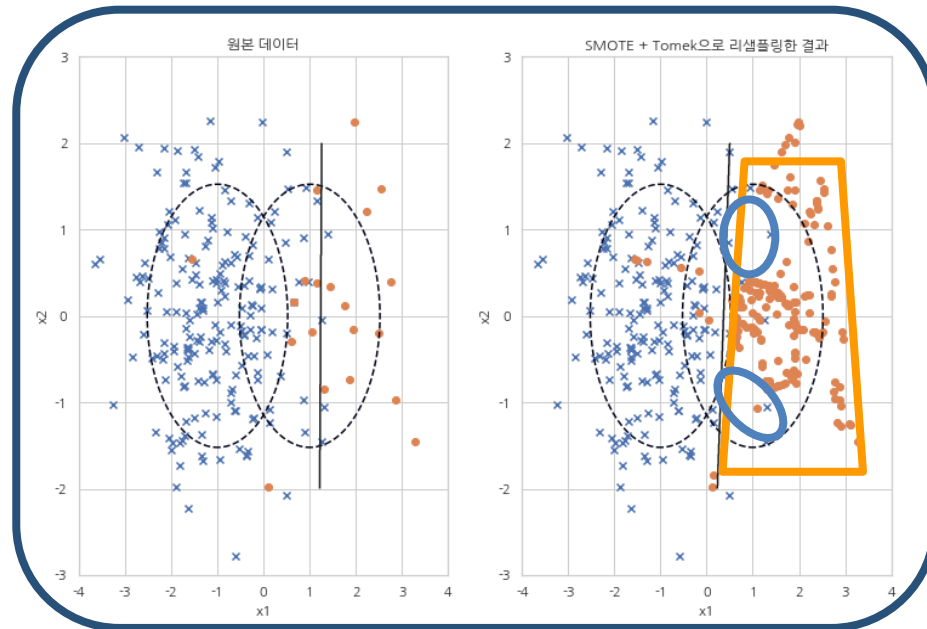


- SMOTE를 적용하여 Over Sampling 한 이후,
Tomek을 사용하여 Under Sampling 하는 방법
- SMOTE sampling된 데이터에서 소수 클래스와 가까운
다수 클래스 데이터를 제거함

Sampling

Combine Sampling

SMOTE + Tomek

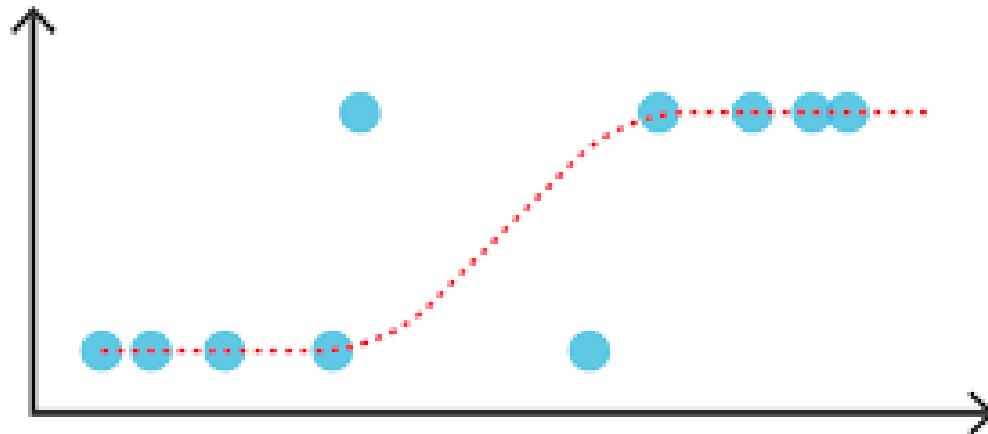


- SMOTE를 적용하여 Over Sampling 한 이후, Tomek을 사용하여 Under Sampling 하는 방법
- SMOTE sampling된 데이터에서 소수 클래스와 가까운 다수 클래스 데이터를 제거함

모델링

1. 로지스틱 회귀 모델

- 이진 분류를 수행하는 데 사용 됨
- 로그 오즈를 구한 후 시그모이드 함수 적용
- 데이터가 해당 클래스에 속할 확률을 계산



모델링

2. 나이브 베이즈

- 조건부 확률에 기반해 데이터가 클래스에 속할 특징 확률을 계산
 - 노이즈나 결측치가 많은 데이터 처리에 특화
 - 예측한 특징이 상호 독립이라 가정하고 계산을 단순화

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

Diagram illustrating the Naive Bayes formula with labels:

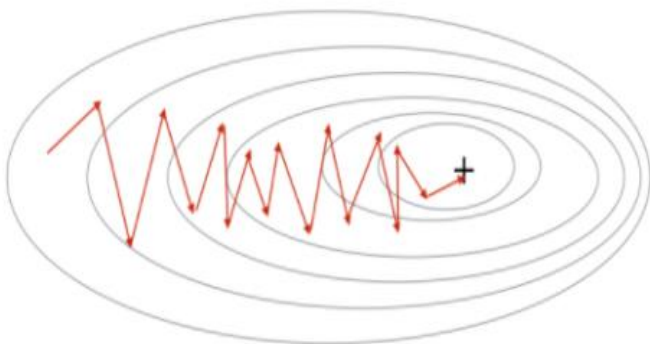
- $P(c | x)$ is labeled **Posterior Probability**.
- $P(x | c)$ is labeled **Likelihood**.
- $P(c)$ is labeled **Class Prior Probability**.
- $P(x)$ is labeled **Predictor Prior Probability**.

모델링

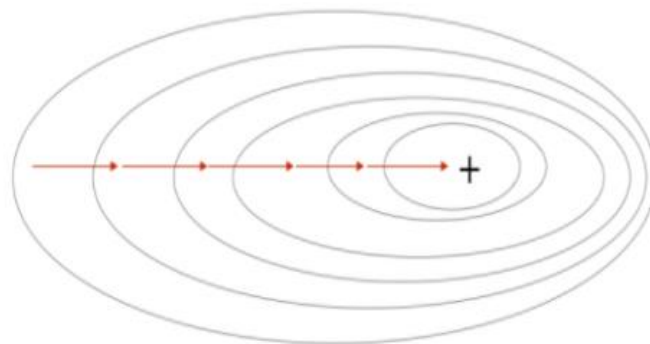
3. 확률적 경사 하강법 (SGD)

- 점진적 학습 모델 중 하나로, 하나의 샘플씩 경사를 조금씩 내려감
- 전체 샘플을 소진하고도 최적점에 도달하지 못한 경우 과정을 반복
 - 퍼셉트론을 손실함수로 사용해 분류 문제에 활용

Stochastic Gradient Descent



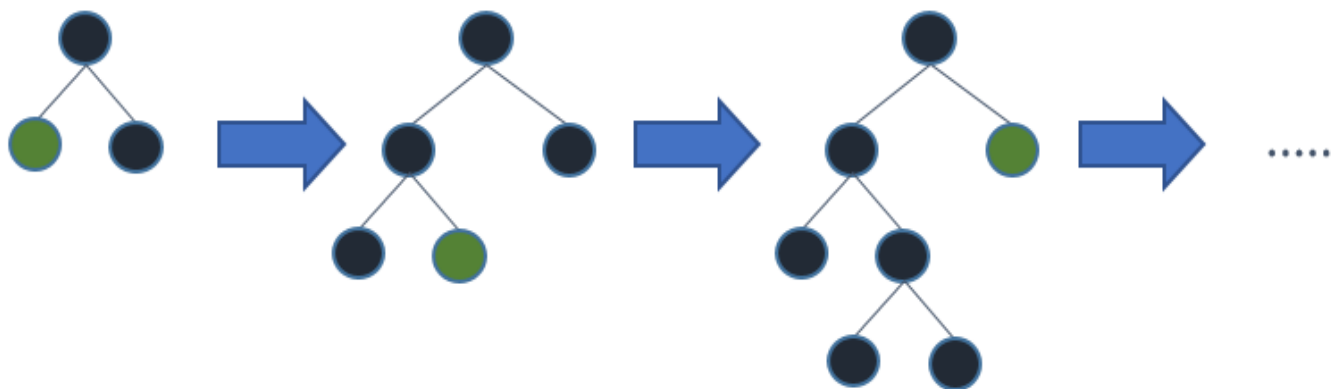
Gradient Descent



모델링

4. LGBM 분류기

- 트리기반의 부스팅 모델
- 다른 트리모델과 다르게 leaf-wise 확장을 하는 특징을 가짐
 - 예측한 특징이 상호 독립이라 가정하고 계산을 단순화

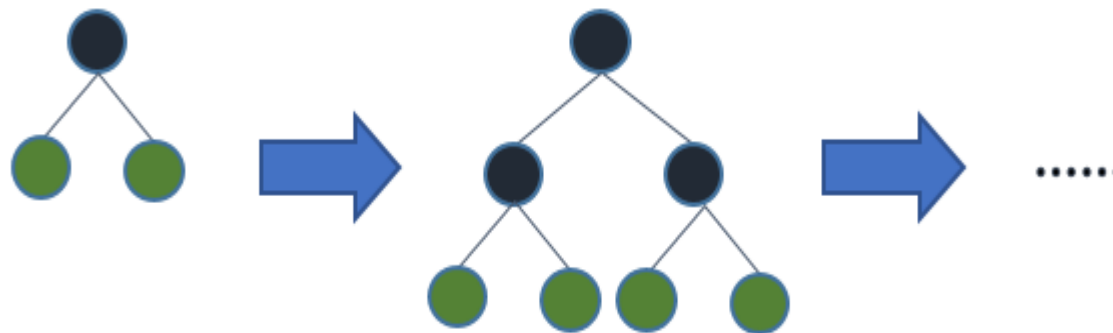


Leaf-wise tree growth

모델링

5. XGBoost 분류기

- 트리기반의 부스팅 모델
- Pruning을 통해 분할 수를 줄임으로써 오버피팅을 방지
 - Level-wise 확장으로 대칭적 트리를 만듦



Level-wise tree growth

조합 이모저모

Information Gain + SMOTE + Logistic Regression

```
[ ] feat_importances_nonzero = feat_importances[feat_importances>0].index  
  
[ ] x_selected = x[feat_importances_nonzero]  
  
[ ] x_selected_train, x_selected_test, y_selected_train, y_selected_test = train_test_split(x_selected, y, test_size=0.3)  
  
[ ] x_selected_train_smote, y_selected_train_smote = smote.fit_resample(x_selected_train, y_selected_train)  
  
[ ] log_reg = LogisticRegression()  
    log_reg.fit(x_selected_train_smote, y_selected_train_smote)  
    pred = log_reg.predict(x_selected_test)
```

```
▶ get_clf_eval(y_selected_test, pred)
```

```
Confusion_matrix:  
[[5820 1746]  
 [ 226  608]]  
Accuracy: 0.7652  
Precision: 0.2583  
Recall: 0.7290  
F1: 0.3814  
AUC: 0.7491
```

조합 이모저모

Information Gain + SMOTE + Logistic Regression

```
[ ] feat_importances_nonzero = feat_importances[feat_importances>0].index  
  
[ ] x_selected = x[feat_importances_nonzero]  
  
[ ] x_selected_train, x_selected_test, y_selected_train, y_selected_test = train_test_split(x_selected, y, test_size=0.3)  
  
[ ] x_selected_train_smote, y_selected_train_smote = smote.fit_resample(x_selected_train, y_selected_train)  
  
[ ] log_reg = LogisticRegression()  
    log_reg.fit(x_selected_train_smote, y_selected_train_smote)  
    pred = log_reg.predict(x_selected_test)
```

▶ get_clf_eval(y_selected_test, pred)

```
Confusion_matrix:  
[[5820 1746]  
 [ 226  608]]  
Accuracy: 0.7652  
Precision: 0.2583  
Recall: 0.7290  
F1: 0.3814  
AUC: 0.7491
```

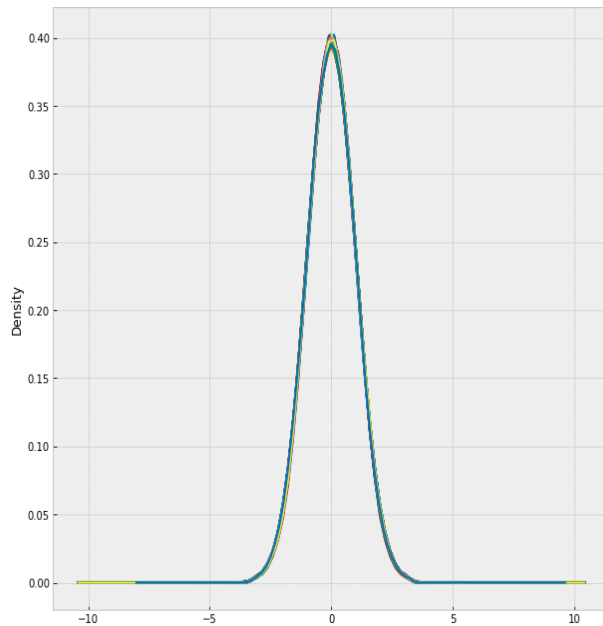


부스팅 계열 모델보다 f1 score가 낮음

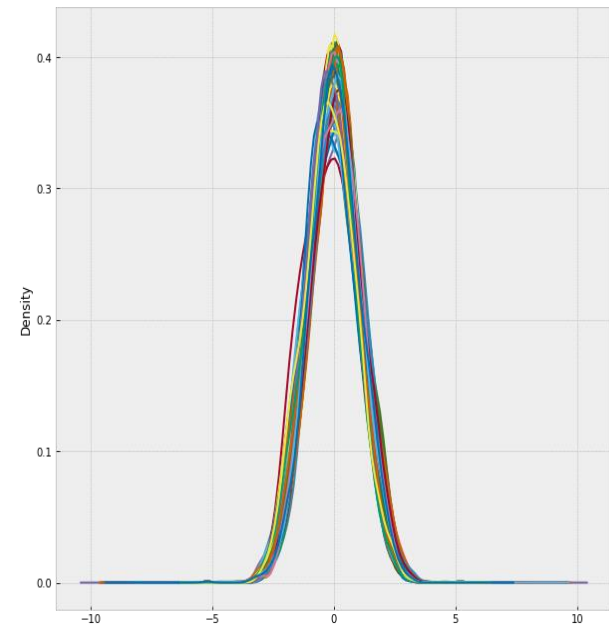
조합 이모저모

Full Data + Gaussian Naïve Bayes

Likelihood KDE Plots for the Negative Class after Quantile Transformation



Likelihood KDE Plots for the Positive Class after Quantile Transformation



각 클래스의 연속적인 값들이 **가우스 분포**를 따름을 확인

조합 이모저모

```
[ ] from sklearn.metrics import f1_score
    from sklearn.model_selection import train_test_split

    train_x, val_x, train_y, val_y = train_test_split(X_train, y_train, test_size = 0.3, random_state = 2021)

    pipeline = make_pipeline(QuantileTransformer(output_distribution='normal'), GaussianNB())
    pipeline.fit(train_x, train_y)

    y_pred = pipeline.predict_proba(val_x)[:,-1]
    y_pred = (y_pred>0.5).astype(int)
```

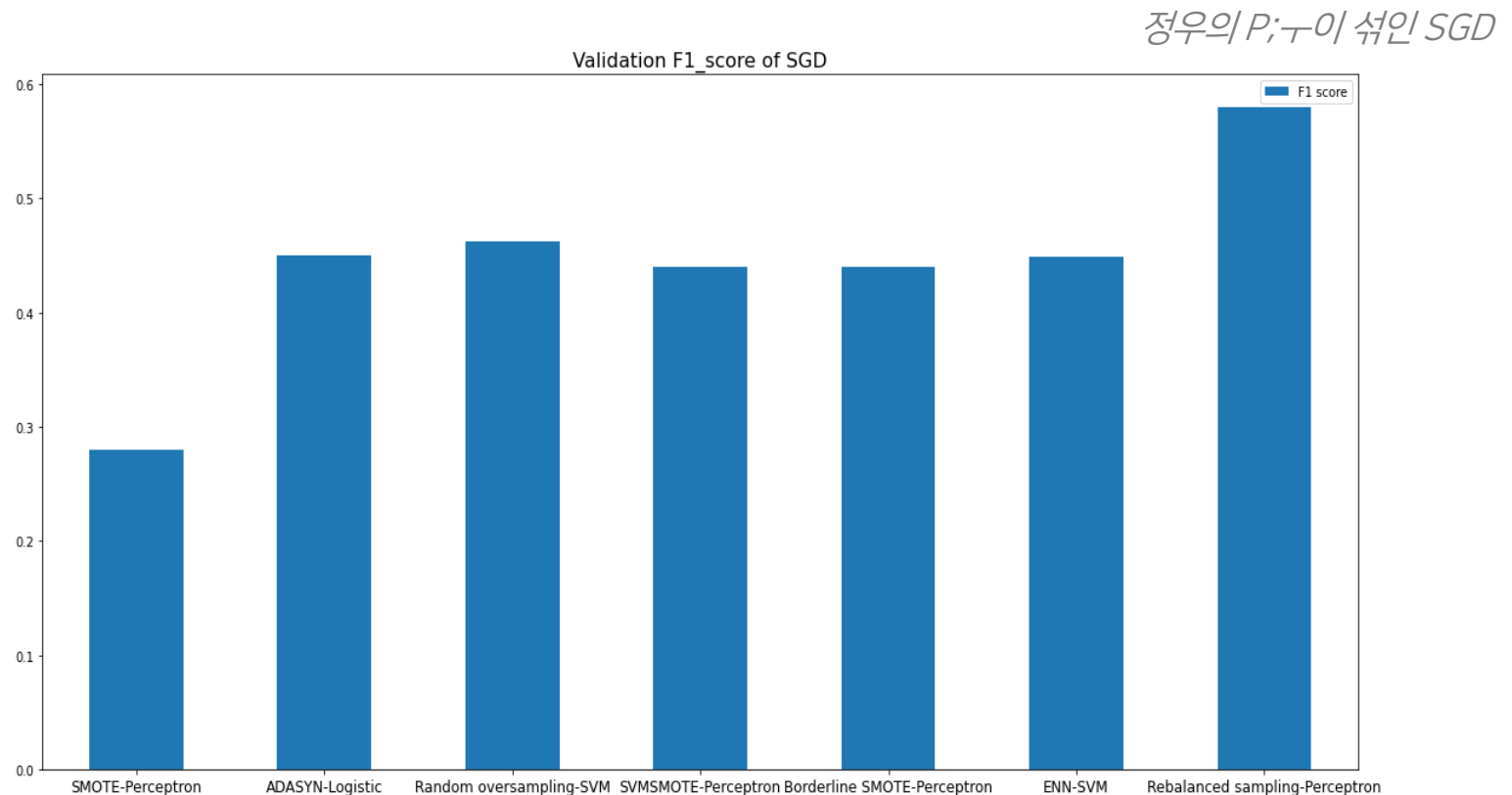
```
[ ] f1_score(val_y, y_pred, pos_label = 1)
```

```
0.4576547231270358
```



부스팅 계열 모델보다 f1 score가 낮음

조합 이모저모



확률적 경사하강법을 로지스틱 회귀, 퍼셉트론, SVM에 적용한 모델로 샘플링 방법과 조합

➡ Validation F1 score는 최대 0.58까지 나왔으나, test F1 score에서 0.40 이하를 기록..

조합 이모저모

최악의 조합

full + SMOTE/Tomeklinks 샘플링 + LGBM 분류

| | | |
|---------------|----------------------|------|
| Boosting | 부스팅 방법 | gdbt |
| bagging_freq | bagging의 비율 | 1 |
| Learning_rate | 학습률 | 0.05 |
| Num_leaves | 노드 분할 시 필요한 instance | 40 |
| Max_depth | 트리의 최대 깊이 | 5 |

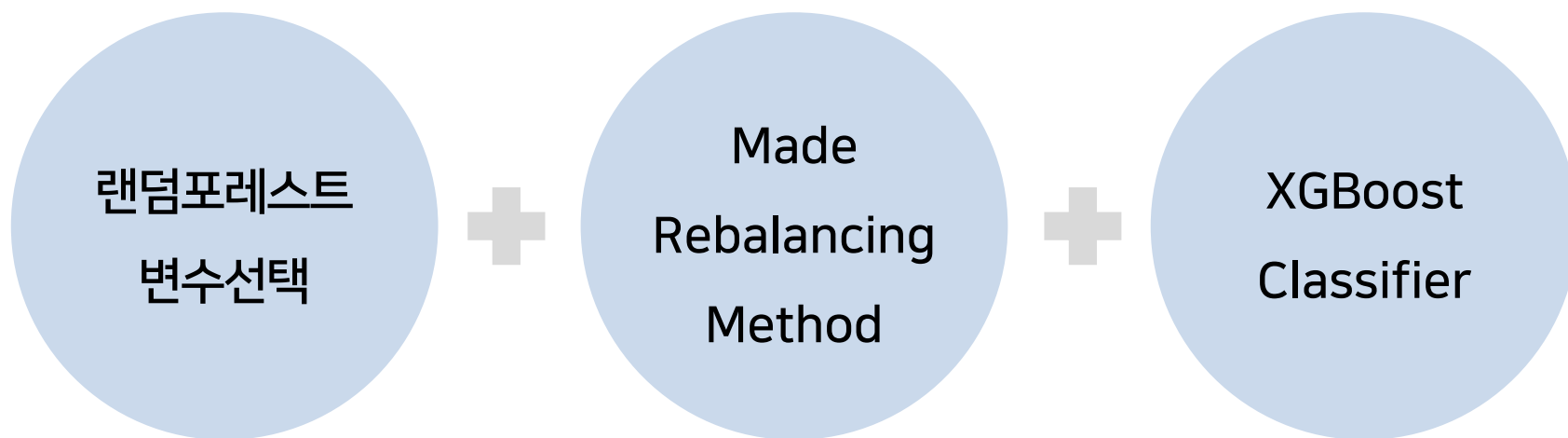


F1_score: 0.13

4

최종 모델

최종 모델



리더보드 기준 f1 score: 0.50717

최종 모델



베개를 부비적대는 라이언

오 피피티 만들다가 lgbm 변수 선택 vs full 데이터
비교하면서 full+random search+ 최종병기+lgbm
모델 썼는데 이게 1위가 됐

됐네요..?

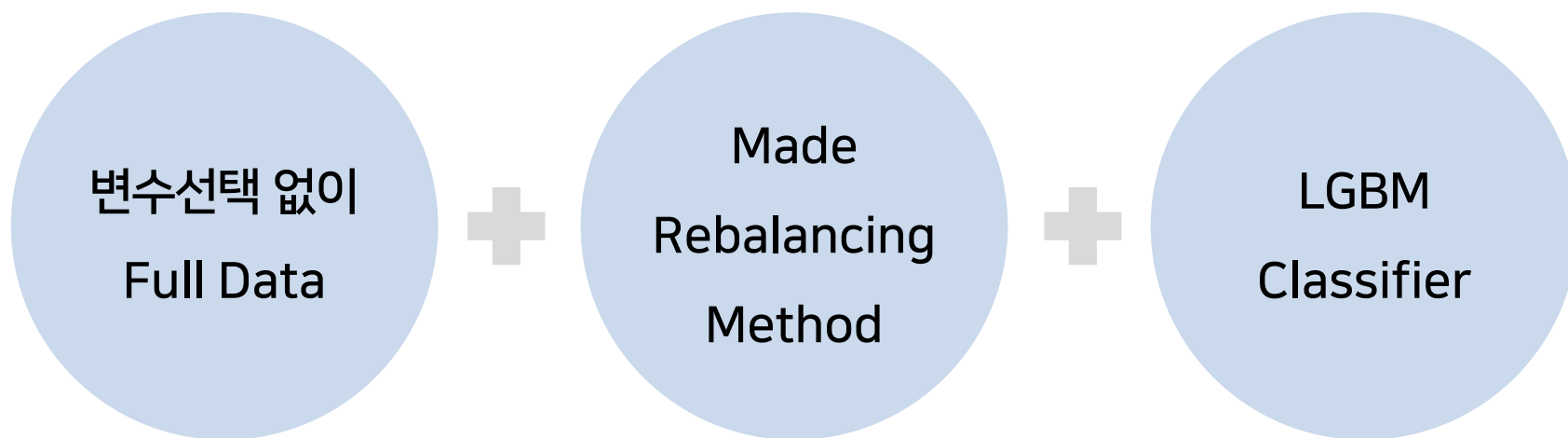
오후 12:40

제출 당일 성능이 더 좋은 조합이 발견 됨!



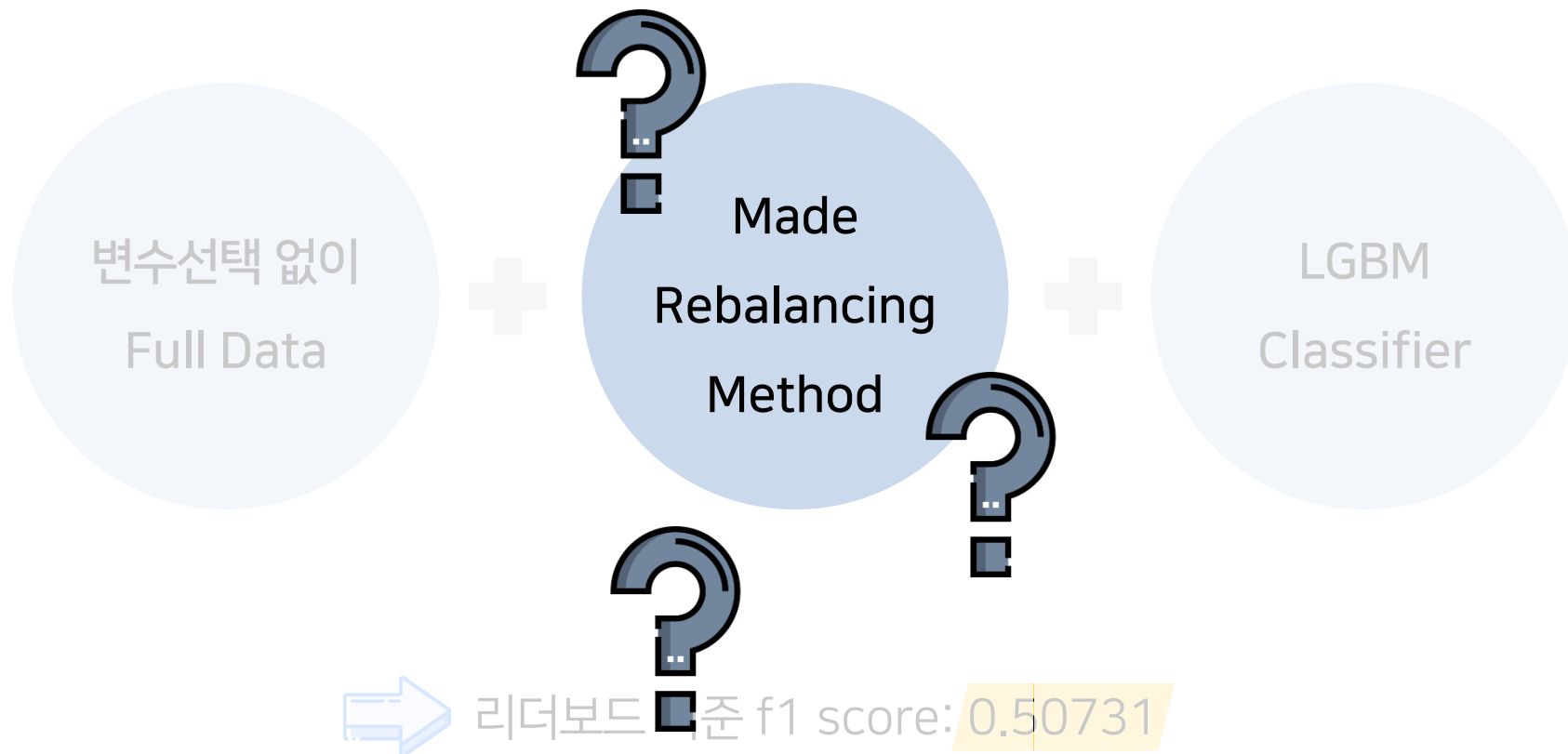
리더보드 기준 f1 score: 0.50717

최종 모델



➡ 리더보드 기준 f1 score: 0.50731

최종 모델




직접 만든 샘플링

Made Rebalancing Method

최종병기 진모쿵야...
샘플링 '끝'

```
def balanced_sampling(input_df, factor):  
  
    # 앞서 만든 함수로 수치형 변수만 갖는 데이터프레임 불러옴  
  
    train = numeric_cols(input_df)  
    y= train['target']  
  
    # Target이 1인 row와 Target이 0인 row를 나누고 각각의 row 개수를 셈  
  
    X_one = train[train.target==1]  
    X_zero= train[train.target==0]  
    total_target = X_one.shape  
    print("Target Size : ",total_target[1],total_target[0])  
  
    # 여기서 factor는 함수 호출시 입력받는 숫자  
  
    scale_factor = factor  
  
    # Target이 0인 row 개수가 더 많으므로 factor 곱하기 Target이 1인 row 개수만큼만 Target이 0인 row에서 샘플링  
  
    X_zero1=X_zero.sample(scale_factor*total_target[0], random_state = 1)
```



직접 만든 샘플링

Made Rebalancing Method

```
def balanced_sampling(input_df, factor):
```

```
    # 앞서 만든 함수로 수치형 변수만 갖는 데이터프레임 불러옴
```

```
    train = numeric_cols(input_df)  
    y= train['target']
```

```
    # Target이 1인 row와 Target이 0인 row를 나누고 각각의 row 개수를 셈
```

```
    X_one = train[train.target==1]  
    X_zero= train[train.target==0]  
    total_target = X_one.shape
```

```
    print(' Target Size : ',total_target[1],total_target[0])
```

```
    # 여기서 factor는 함수 호출시 입력받는 숫자
```

```
    scale_factor = factor
```

```
    # Target이 0인 row 개수가 더 많으므로 factor 곱하기 Target이 1인 row 개수만큼만 Target이 0인 row에서 샘플링
```

```
    X_zero1=X_zero.sample(scale_factor*total_target[0], random_state = 1)
```

Target이 1인 데이터의
row 개수 계산

입력 값 \times 위에서 구한 row 개수만큼
Target이 0인 row에서 Sampling

직접 만든 샘플링

함수 입력 값을 3으로 설정



Target이 1인 row 개수 × 3 만큼을

Target이 0인 row에서 추출

```
def balanced_sampling(input_df, factor):
```

```
# 앞서 만든 함수로 수치형
```

```
train = numeric_cols(input_df)
y = train['target']
```

```
# Target이 1인 row와 Target이 0인 row를 나누고, 각각의 row 개수를 셈
```

Target 0 25191 : 2809

Target이 1인 데이터의

row 개수 계산

Target 1

```
X_one = train[train.target==1]
X_zero = train[train.target==0]
total_target = X_one.shape
```

```
print('Target Size : ', total_target[1], total_target[0])
```

입력 값 × 위에서 구한 row 개수만큼

```
# 여기서 factor는 함수 호출시 입력받는 숫자
```

Target 0 8427 : 2809

Target이 0인 row에서 Sampling

Target 1

```
scale_factor = factor * total_target[1]
```

```
# Target이 0인 row 개수가 더 많으므로 factor 곱하기 Target이 1인 row 개수만큼만 Target이 0인 row에서 샘플링
```

```
X_zero1 = X_zero.sample(scale_factor * total_target[0], random_state = 1)
```

최종 모델

LGBM Classifier

```
{ 'learning_rate': [0.04,0.06,0.08,0.1], 'n_estimators': [200,500,4000],  
  'num_leaves': [4,6,31], 'boost': ['gbdt'],  
  'objective': ['binary'], 'seed': [500],  
  'reg_alpha': [0.1, 2, 2.1], 'reg_lambda': [0,0.2, 2.54],  
  'num_threads': [8], 'max_bin': [165,256],  
  'min_child_samples': [153], 'feature_fraction': [1],  
  'bagging_freq': [1], 'bagging_fraction': [0.85], 'max_depth' : [1], 'gamma' : [1], 'min_child_weight' : [0.1],  
  'subsample' : [0.8], 'colsample_bytree' : [1],
```

```
  'colsample_byrow' : [0.8], 'colsample_bytree' : [1]
```

최종 모델

LGBM Classifier

```
learning_rate : [0.04, 0.05, 0.08, 0.1], n_estimators : [200, 500, 4000],  
'num_leaves': [4, 6, 31], 'boost': ['gbdt'],  
'objective': ['binary'], 'seed': [500],  
'reg_alpha': [0.1, 2, 2.1], 'reg_lambda': [0, 0.2, 2.54],  
'num_threads': [8], 'max_bin': [165, 256],  
'min_child_samples': [153], 'feature_fraction': [1],  
'bagging_freq': [1], 'bagging_fraction': [0.85], 'max_depth' : [1], 'gamma' : [1], 'min_child_weight' : [0.1],  
'subsample' : [0.8], 'colsample_bytree' : [1],
```

LGBM으로 이진분류를 수행한 여러 사례들을 찾아 파라미터 기준을 설정!









Random Search CV 진행

최종 모델

LGBM Classifier

| Parameter | Value |
|-------------------|-------|
| n_estimators | 4000 |
| learning_rate | 0.1 |
| max_depth | 1 |
| Max_bin | 256 |
| min_child_samples | 153 |
| subsample | 0.8 |
| colsample_bytree | 1 |
| Bagging_fraction | 0.85 |
| Bagging_freq | 1 |
| Gamma | 1 |
| Reg_alpha | 1 |
| Reg_lambda | 0 |
| Subsample_freq | 0 |

리더보드 결과

| # | Team Name | Notebook | Team Members | Score ? |
|--|-----------------|----------|---|---------|
| 1 | LeEEunSEO | |  | 0.50731 |
| 2 | Jinmo Lee | |  | 0.50731 |
| Your Best Entry ↑ Your submission scored 0.50731, which is an improvement of your previous score of 0.49760. Great job!  Tweet th | | | | |
| 3 | Hye Hyun jo | |  | 0.50717 |
| 4 | SKKU_JungWooLee | |  | 0.50246 |
| 5 | Claire | |  | 0.50000 |



진모콩야의 샘플링으로
리더보드 Top5를 모든 팀원이 점령!

5

세미나를 마치며

의의와 한계



의의

- 여러가지 이진분류 모형과 불균형 데이터 샘플링 방법에 익숙해질 수 있었음
- 변수선택 + 샘플링 + 모델링 방법을 여러가지 적용해보며 수십 개 조합의 성능 결과를 얻을 수 있었음
- 베이지안 최적화를 적용해보고 성능이 좋음을 확인함



한계

- 그리드 서치가 시간이 많이 걸려서 랜덤 서치로만 파라미터 튜닝을 진행했다는 점이 아쉬움
- 어떤 모델에 어떤 샘플링이 잘 맞고 그걸 해석해서 다음에 어떻게 돌려볼지 결정해보는 과정을 경험하고 싶었는데, 어떤 샘플링과 모델을 조합했을 때 잘 나오는지에 대한 해석을 할 수 없었음

방학 세미나 마친 소감

열정맨/우먼들 모인 팀이었네요.. 팀원분들이 다들 순하고 성격이 너무 좋으신 것 같아요! 그리고 사실 회의때마다 재밌었어요ㅋㅋㅋㅋ회의 더 하고 싶었는데 아쉽네요.. 다음에 기회가 되면 또다른 활동에서 팀으로 만났으면 좋겠어요! 진모님 혜현님 은서님 수정님 모두 너무 고생하셨습니다~~

대면으로 한번도 만나지 못하고 화면 너머로만 얘기를 나누게 가장 아쉽네요. 1학기때 활동하며 알지 못한 분들과 친분을 쌓을 수 있어서 좋은 기회였습니다. 다들 실력도 넘 출중하시고 열정도 가득하셔서 리더보드 1-5등 모두 차지할 수 있었는데 이 기세 몰아서 방세 1등 해봅시다!
Team 1 수고하셨습니다 :-)

방학세미나 시작할 때부터 리더보드 2등할 생각은 아예 없었고 결국 1등 할 수 있어서 너무 기쁘고 함께 노력한 팀원들에게 고맙습니다♡

이번 1주 동안 너무 수고 많으셨고 이끌어주셔서 감사합니다!! 팀원분들 덕분에 좋은 결과를 낼 수 있었습니다!!

다들 열정과 능력이 뛰어나셔서 일주일동안 많이 배울 수 있었습니다. 다음에도 팀으로 만나면 좋을거 같아요 모두 수고하셨습니다!!



THANK YOU

