

# 방학세미나 후기



불꽃 학회장팀

박이현 홍지우

# INDEX

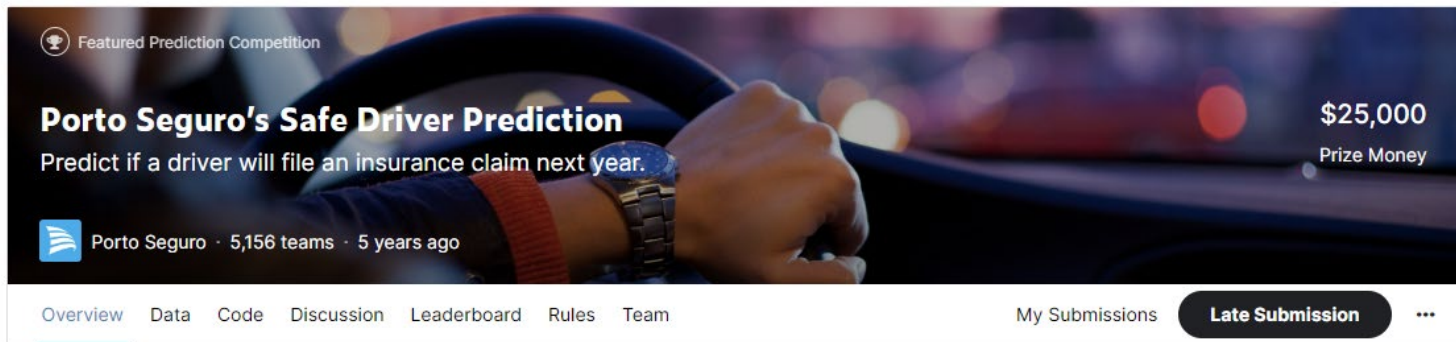
---

1. 출제 의도
2. 방법론 참고
3. 공통 피드백
4. 1등팀 발표

1

출제 의도

## DATA



<https://www.kaggle.com/competitions/porto-seguro-safe-driver-prediction/data>

Kaggle에서 진행했던 차량회사의 보험 예측 Competition

운전자가 내년에 보험 청구를 할 것인지 여부 예측

0 : 보험 미청구 / 1 : 보험 청구

## 목적

불균형 클래스에 대한 분류모델 생성 연습

## 평가지표

### F1 Score

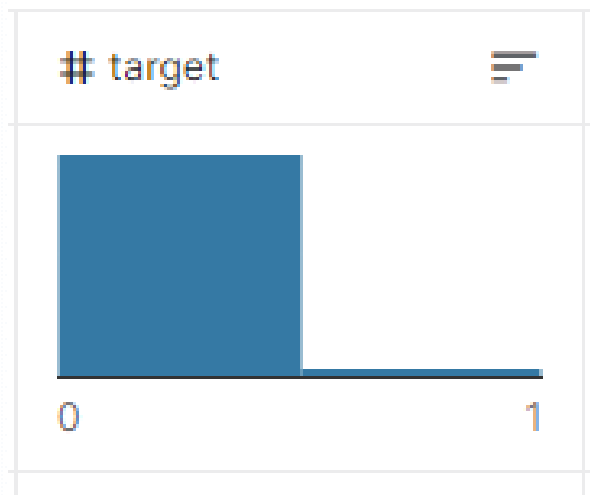
: 정밀도와 재현율의 조화평균

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

분류 모델의 성능을 평가하는 지표로 사용

## 평가지표

### F1 Score



클래스의 분포가 약 30:1인

**Imbalanced Data**의 분류 문제

(보험 가입자 중 사고가 발생하는 경우는 희소)

불균형한 데이터에 대한 성능 평가에는

일반적으로 Accuracy보다

**F1 Score**나 **Cross-Entropy**를 사용

불균형한 분류 모델의 성능을 평가하는 지표로 사용

## 평가지표

### F1 Score

	미청구	청구
미청구	TP	FP
청구	FN	TN

Accuracy :  $(TP+TN)/(TP+FP+FN+TN)$

보험회사의 이익을 위해서는 보험료를

청구할 사람을 예측하는 것이 중요

하지만, Accuracy를 사용하면 Major Class인

보험 미청구자에 의해 결과가 왜곡될 수 있음

따라서, **F1 Score**를 이용

(기존 대회에서는 유사한 Normalized Gini Index 이용)

**불균형한 분류 모델의 성능**을 평가하는 지표로 사용

## 분석 과제

### 불균형 클래스

0과 1이 30:1의 비율로 분포  
Minor Class에 대한 예측

### 수치/범주/순서형 혼합

각 변수들의 영향력 조정 필요  
변수의 상관성 분석

### 변수 분포에 따른 모델 선택

변수의 분포가 굉장히 다양  
모수/비모수/준모수 모델

### NA값 처리

NA값을 대체 또는 제외 필요  
MICE 등 다양한 방법론

### 차원 축소 문제

59개의 변수가 존재  
PCA, 변수선택 등을 이용

### 하이퍼파라미터 튜닝 및 시각화

파라미터 튜닝 시 평가지표  
시각화의 효율성



## 분석 과제

Main!

### 불균형 클래스

극단적인 클래스 불균형 문제는  
어떻게 접근을 해야하나?

수치/범주/순서형 혼합

각 변수들의 영향력 조정 필요  
변수의 상관성 분석

변수 분포에 따른 모델 선택

변수의 분포가 굉장히 다양  
모수/비모수/준모수 모델

NA값 처리

NA값을 대체 또는 제외 필요  
MICE 등 다양한 방법론

차원 축소 문제

59개의 변수가 존재  
PCA, 변수선택 등을 이용

하이퍼파라미터 튜닝  
및 시각화

파라미터 튜닝 시 평가지표  
시각화의 효율성

## 불균형 클래스

### 분석 과제

클래스의 크기가 약 **30:1**로 매우 불균형한 형태를 보임

다양한 샘플링 기법을 통한 분류 문제 해결  
(Upsampling, Downsampling, SMOTE...)



범주형 3주차 클린업 참고

하지만, 불균형 정도가 심해 샘플링 기법만으로는 성능 향상에 한계



단순한 이진분류의 문제의 관점에서 벗어나, **이상치 탐지 문제**로 접근

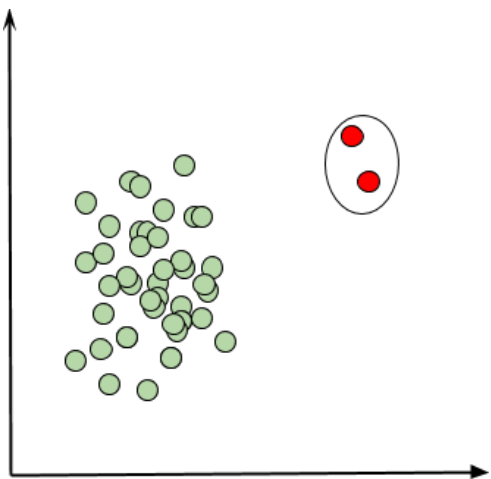
# 2

방법론 참고

## 이상치 탐지

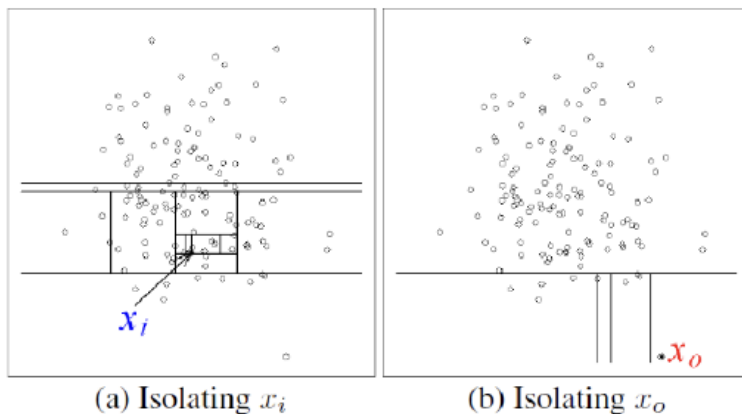
### Anomaly Detection

클래스 비율이 극단적으로 차이가 나는 상황에서는 이상치 탐지로 접근 가능



비지도학습의 일종 (Unsupervised)  
대부분의 Sample이 정상이라고 가정  
Iforest, KNN, SVDD, Gaussian 등  
다양한 이상치 탐지 방법이 존재  
신용, 사기 탐지 등 다방면에서 사용

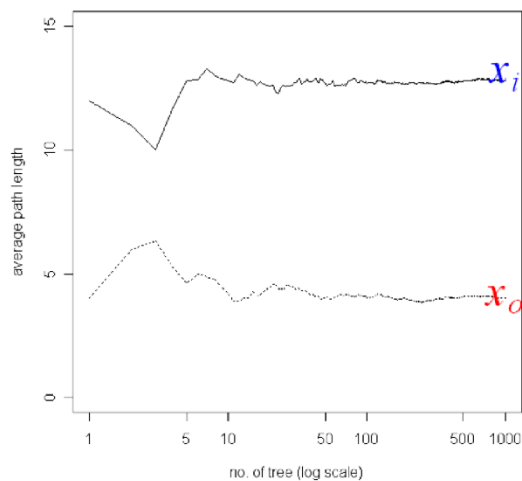
## Isolation Forest



기본적으로 트리 모델과 유사하게 분할을 진행해 나감

정상치는 한 지역에 몰려있고, 이상치는 떨어져 있을 것이라고 가정  
이상치는 적은 횟수의 분할만으로도 구분이 되고, 정상치는 구분이 안됨  
즉, 이상치일수록 시작점인 Root Node와 가까움 (분할 횟수가 적다)

## Isolation Forest



(c) Average path lengths converge

평균적으로 이상치는 분할 횟수가 적음

정상치는 분할 횟수가 많음

평균 분할 거리를 **Anomaly Score**로

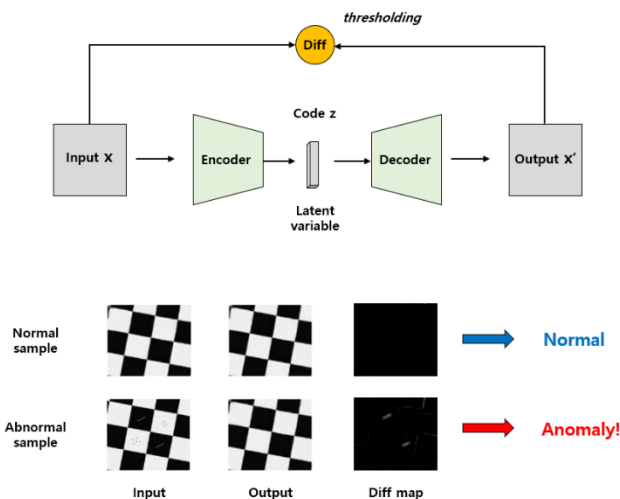
적용하여 이상치를 구분해 냄

자세한 내용은 해당 논문 참고...

("Isolation Forest", F. T. Liu, K. M. Ting and Z. Zhou)

유사하지만 더 발전된 알고리즘인 **Extended Isolation Forest**도 존재!

## Auto-Encoder



출처 : <https://hoya012.github.io/blog/anomaly-detection-overview-1/>

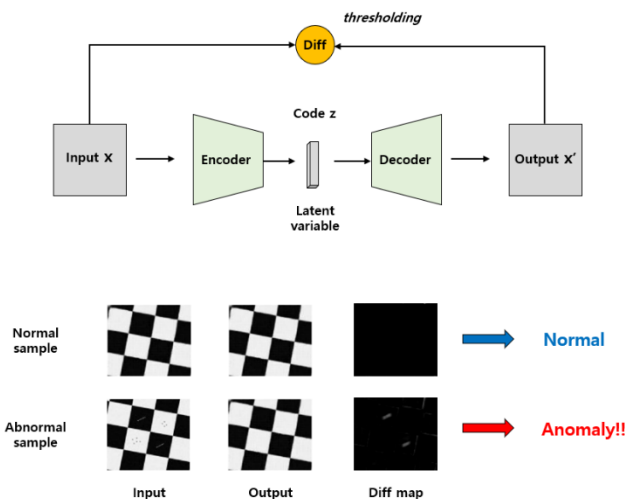
### PCA와 유사한 형태의 알고리즘

데이터를 Encoder에 투입하여 잠재변수로 변환 후

Decoder로 다시 원본에 가깝게 복원하는 과정

비정상적인 Sample은 해당 과정에서 차이가 발생

## Auto-Encoder



출처 : <https://hoya012.github.io/blog/anomaly-detection-overview-1/>

하이퍼파라미터에 따라 전반적인 복원 성능이 좌우되므로  
Supervised Learning에 비해 다소 불안정한 단점 존재  
별도의 라벨링 과정 없이 성능을 낼 수 있다는 장점 보유



## Auto-Encoder

nr	feat	Norm- alization	unsupervised		model		5-fold CV		kaggle	
			type	time[h]	type	time[h]	gini	logloss	public	private
1	f0	-	-	0	lightgbm objective=binary, 1400 rounds, boosting_type=gbdt, learning_rate=0.01, max_bin=255, num_leaves=31, min_data_in_leaf=1500, feature_fraction=0.7, bagging_freq=1, bagging_fraction=0.7, lambda_l1=1, lambda_l2=1	0.13	0.28843	0.15163	0.28368	0.29097
2	f0	Rank Gauss	denoising autoencoder, deep stack topology=221-1500r-1500r-1500r-2211, lRate=3e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.15, nEpochs=1000	5	neural net topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=150	3.45	0.29036	0.15184	0.28970	0.29298
3	f0	Rank Gauss	denoising autoencoder, deep stack topology=221-1500r-1500r-1500r-2211, lRate=3e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.07, nEpochs=1000	5	neural net topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=200	4.6	0.28942	0.15185	0.28846	0.29377
4	f0	Rank Gauss	denoising autoencoder, deep stack topology=221-1500r-1500r-1500r-2211, lRate=2.9e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.15, nEpochs=1000 colGroups=1	5	neural net topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=76	1.8	0.29062	0.15174	0.28778	0.29265
5	f0	Rank Gauss	denoising autoencoder, bottleneck topology=221-15000r-15000r-3000l-15000r- 15000r-2211, lRate=1e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.1, nEpochs=300	75.2	neural net Topology=3000-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=200	3.2	0.28904	0.15202	0.28745	0.29373
6	f0	Rank Gauss	denoising autoencoder, deep stack topology=221-1500r-1500r-1500r-2211, lRate=2.8e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.2, nEpochs=1000	5	neural net topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=150	3.4	0.29091	0.15180	0.28856	0.29303
					linear blend, all w=1 of 1,2,3,4,5,6	0.01	0.29442	0.15159	0.29136	0.29653

기존 대회에서 성능이 가장 좋았던 모델은

AutoEncoder를 깊게 쌓은 **딥러닝 모델**이었음!

<https://www.kaggle.com/competitions/porto-seguro-safe-driver-prediction/discussion/44629>

## 사용 시 참고

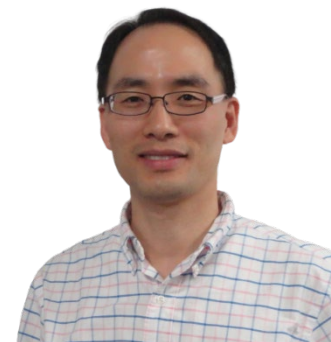
### 대회 목적

해당 보험사의 고객이 **내년에 보험을 청구할지**에 대한 예측이 목적

오토 인코더는 예측 성능이 높았으나, 왜 이런 결과가 도출되었는지 분석 어려움

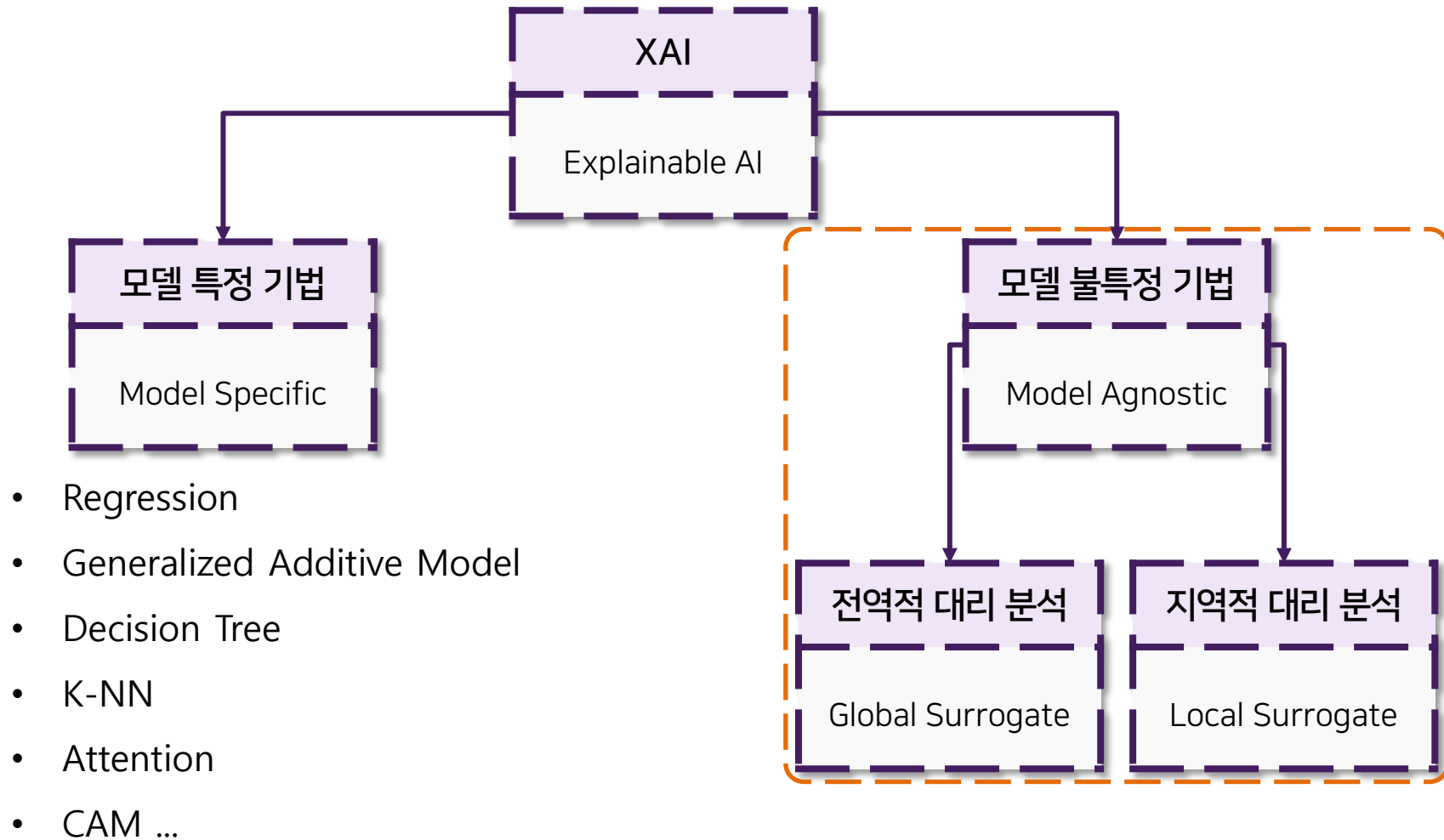
데이터 실무 및 공모전에서는 성능도 중요하지만, **해석이 더 중요한 경우** 많음

신용평가사 등 데이터를 분석하는 직무에서 일을 할 때는  
모델의 성능 자체도 필요하지만, 성능이 다소 낮더라도  
해석이 가능하여 향후 경영 전략에 도움이 되는 데이터 분석  
공모전에서도 성능만 좋은 모델은 좋은 평가를 못받음

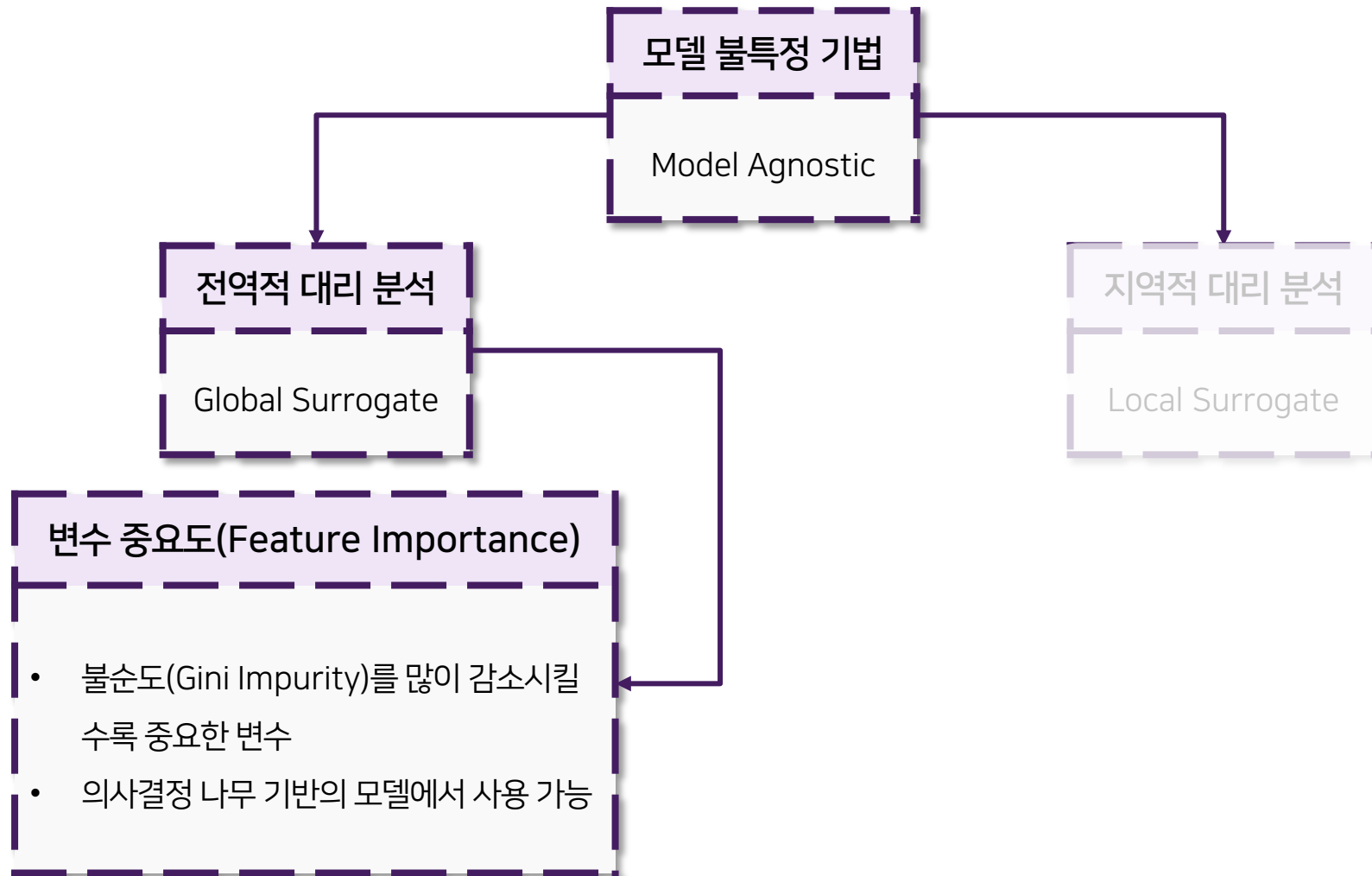


근백 리 교수님 + 신용평가사에 계신  
선배님들 조언...

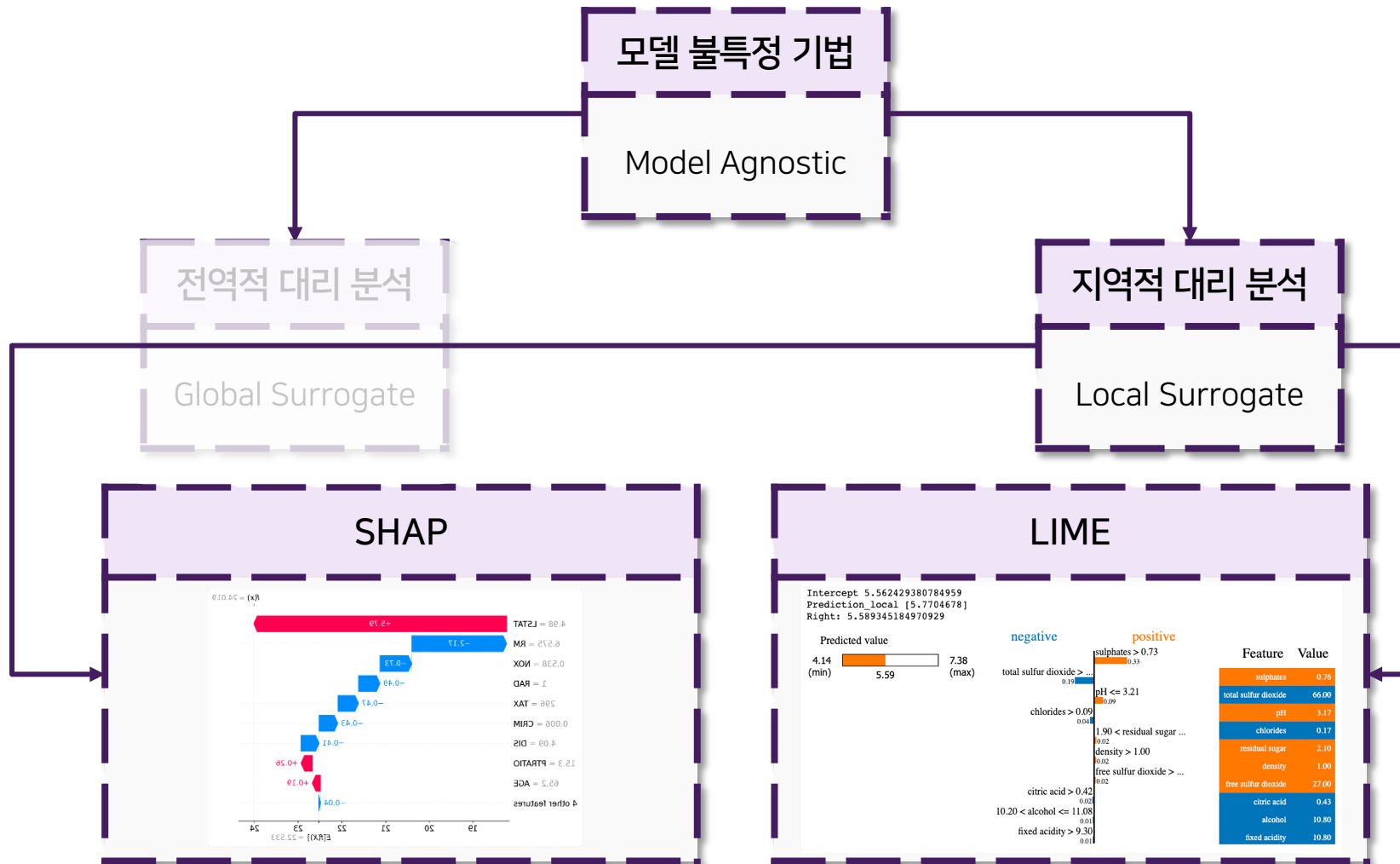
## 학습 결과 해석



## 학습 결과 해석



## 학습 결과 해석



# 3

공통 피드백

## 공통 피드백

클래스가 매우 불균형한 상태이기 때문에 Oversampling 기법을 많이 사용  
Oversampling을 통해 Major와 Minor 클래스를 1:1까지 증강



클래스 불균형이 매우 심한 상태에서는 과도한 Oversampling을 통해 증강을 하면  
데이터가 과적합되고 분석에 문제가 발생할 가능성이 높음!  
과도한 불균형 데이터에 대해서는 이상치 탐지로 접근해보는 것도 좋을 것 같음

## 공통 피드백

트리기반 모델을 사용할 때, 범주형 자료에 대해서 One-Hot Encoding 수행



Python에서는 Factor형 자료 구조가 존재하지 않으므로,  
One-Hot Encoding에서 결과값이 달라지는 문제가 발생할 수 있음 !  
따라서 트리모델을 분석할 때는 R을 이용하는 것이 더 편리할 것 같음



## 공통 피드백

MICE를 통하여 결측치를 대체할 경우, Target 값을 제외한 X 변수들로만  
결측치를 대체해야 함

각 변수들에 대해서 MICE를 이용하여 대체한 값들의 분포와  
기준에 존재하는 관측치들의 분포가 유사한지 확인해야 함!



From. 통모머  
김재직 교수님



MICE는 효과적인 대체방법이며 성능이 좋다고 알려져 있지만  
위와 같은 선결 조건이 갖추어 진다면 더 좋은 성능을 낼 수 있음

## 공통 피드백

Test 데이터에 대해 결측값 대체와 같은 전처리를 진행할 경우  
Test 데이터를 이용해 결측치를 대체하면 Data Leakage 문제 발생

만약 Test 데이터에 대한 결측치를 대체할 경우, Train 데이터로 학습을 시킨 후  
모델을 대체하거나, 해당 변수들에 대해 평균값 등은 Train의 값을 사용하자!




Data Leakage 문제가 발생할 경우 Train 데이터에 대해서 모델의 성능이 높게  
나오나 예측의 성능은 낮은 Overfitting 문제가 발생!


4

1등팀 발표





다들 한주동안 너무  
고생 많으셨습니다!



다들 개강까지 노트북을 켜지 맙시다...