

10. 세이지메이커 스튜디오 II

1강. 세이지메이커 스튜디오 시각화 및 디버깅

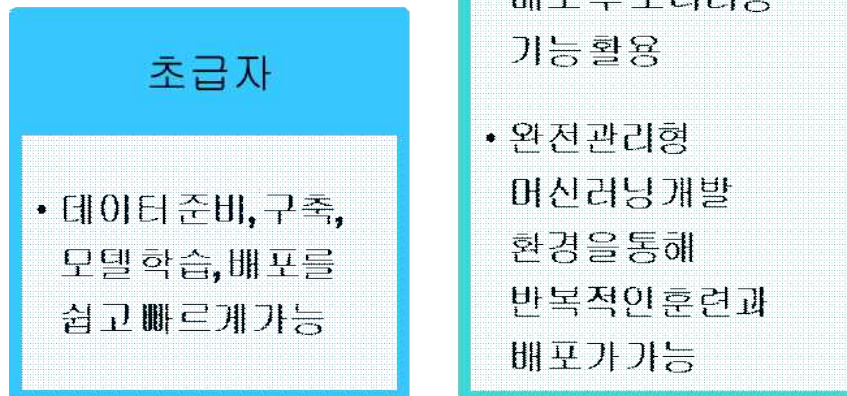
학습목표

- 세이지메이커 스튜디오에서 모델 훈련 결과의 시각화 단계를 설명할 수 있다.
- 세이지메이커 스튜디오에서 모델 훈련 작업의 디버깅 단계를 설명할 수 있다.

학습내용

- 훈련 결과 시각화
- 훈련 작업 디버깅

■ 세상을 잇(IT)다!

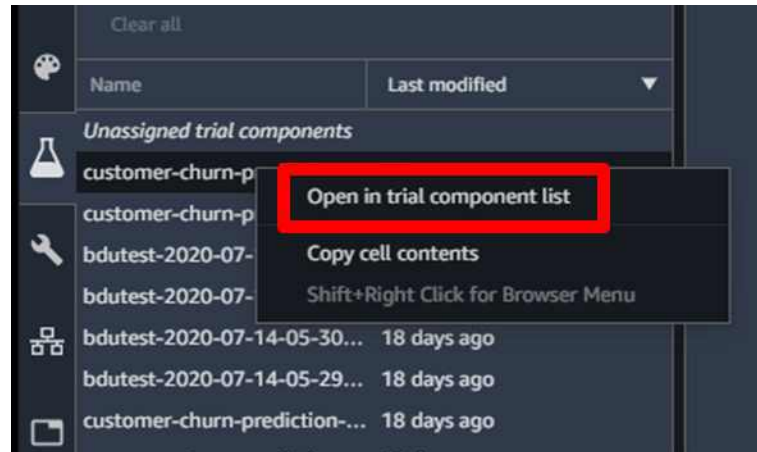


1. 훈련 결과 시각화

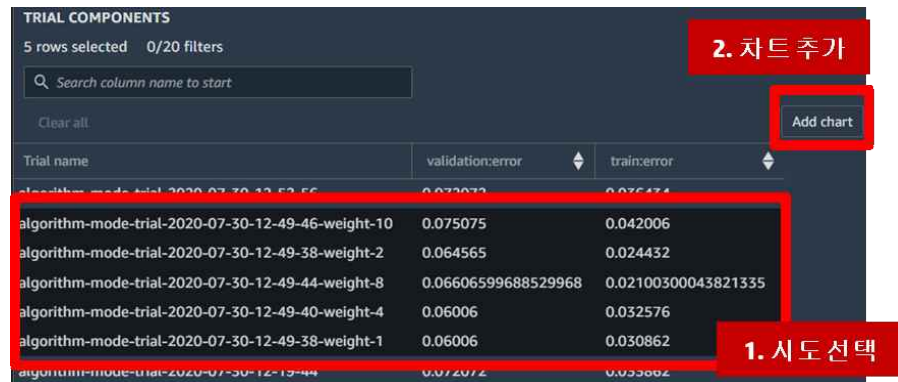
- 훈련 결과 시각화 단계
 - ① 시각화할 시도(Trial) 선택
 - ② 차트 속성 구성
 - ③ 차트 모니터

✓ 시각화할 시도 선택

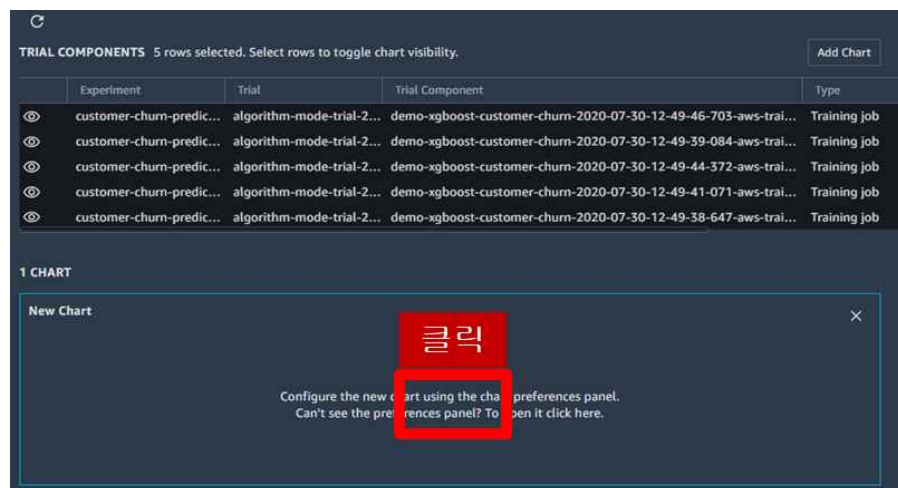
- 실험 아이콘을 클릭하여 실험명 오른쪽 클릭 후 시도 구성요소 목록 클릭



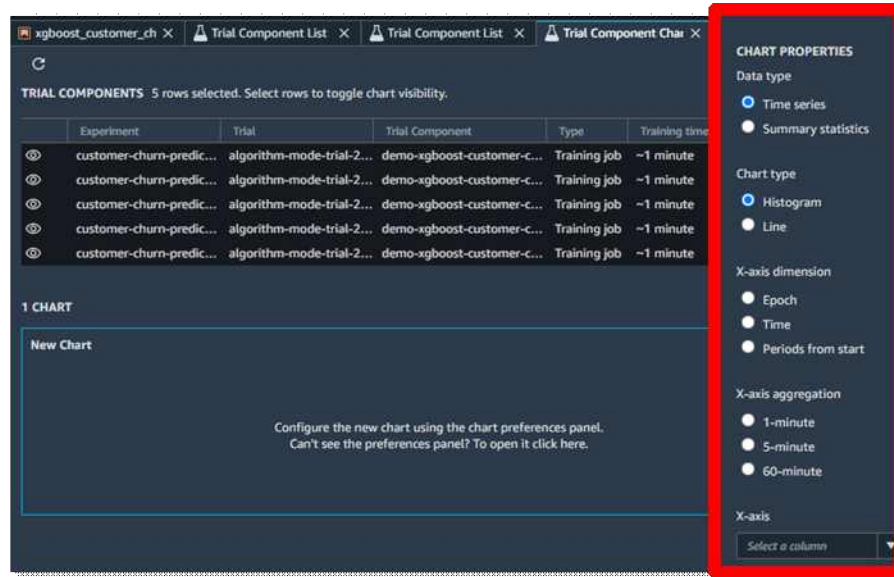
- 시도 구성 요소 목록에서 시각화할 시도를 선택한 다음 차트 추가를 선택



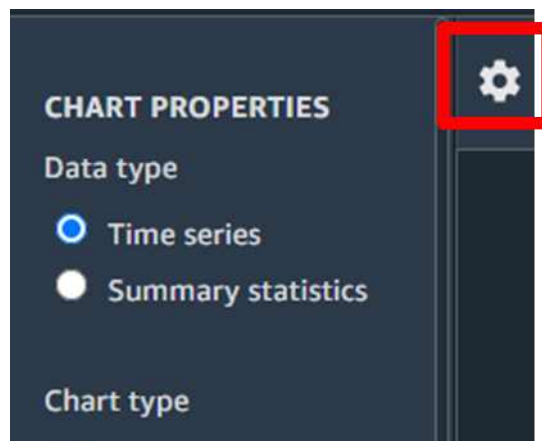
- 하단의 new Chart 클릭



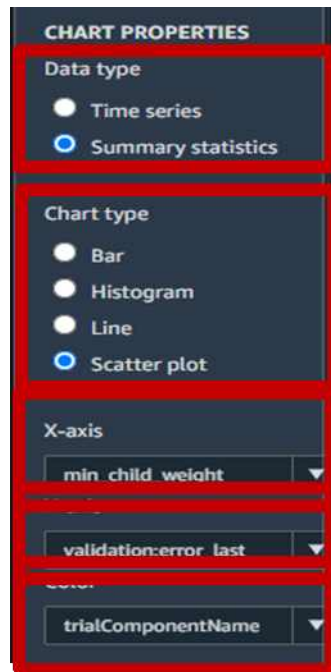
- 클릭 후 오른쪽에 차트 속성 창이 열림



- 차트 속성 창이 열려 있지 않으면 오른쪽 상단 모서리에 있는 설정 아이콘 클릭
- 창을 닫으려면 설정 아이콘을 다시 선택

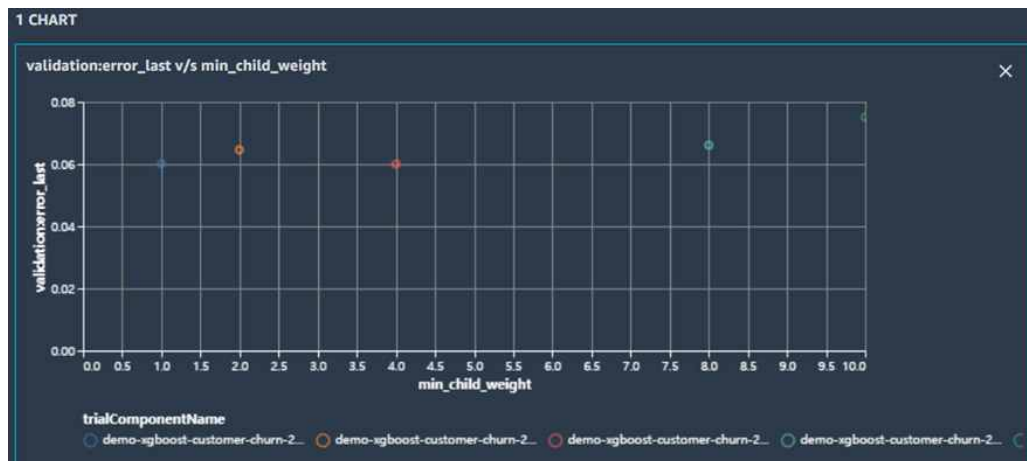


- ✓ 차트 속성 구성
 - 데이터 형식에서 Summary statistics 선택
 - 차트 유형에서 Scatter plot 선택
 - X-axis에서 min_child_weight 선택
 - Y-axis에서 validation:error_last 선택
 - 색상에서 trialComponentName 선택



✓ 차트 모니터

- Studio에 구성된 차트 속성에 따른 차트 생성 완료



1. 훈련 작업 디버깅

- 훈련 작업 디버깅 단계
 - ① 훈련 작업 분석 규칙 지정
 - ② 디버그 규칙을 사용하여 새 시도 생성
 - ③ 디버그 결과 확인

✓ 훈련 작업 분석 규칙 지정

- 디버거를 사용하면 모델 훈련에 필요한 시간, 리소스 및 비용을 획기적으로 줄일 수 있음
- 훈련 작업을 분석하는데 사용할 규칙을 지정

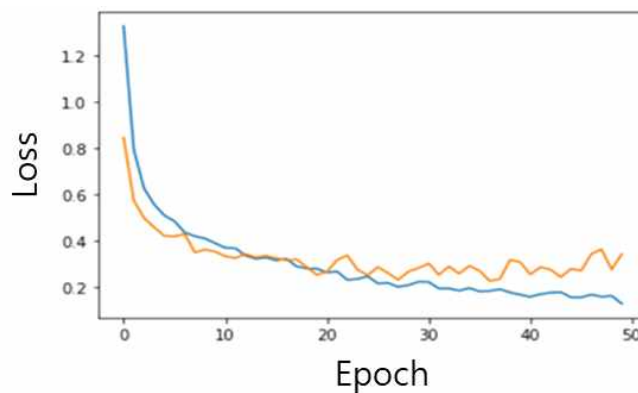
```
debug_rules = [Rule.sagemaker(rule_configs.loss_not_decreasing()),
               Rule.sagemaker(rule_configs.overtraining()),
               Rule.sagemaker(rule_configs.overfit())
               ]
```

✓ 훈련 작업 분석 규칙의 종류

DeadRelu	VanishingGradient	SimilarAcrossRuns
ExplodingTensor	WeightUpdateRatio	UnchangedTensor
Poor WeightInitialization	AllZero	TensorVariance
SaturatedActivation	ClassImbalance	CheckInputImages
Confusion	LossNotDecreasing	NLPSequenceRatio
Overfit	Overtraining	TreeDepth

✓ loss_not_decreasing 규칙

- 손실이 적절한 비율로 감소하지 않는 경우를 감지
- 손실은 스칼라 값 이어야 함



num_steps	•규칙이 손실 감소 여부를 확인한 이후의 최소 단계 수
diff_percent	•손실이 num_steps사이에서 감소해야 하는 최소 백분율 차이

✓ overtraining 규칙

- 모델이 과잉 훈련 중인지 여부를 감지
- 조기 중단으로 과잉 훈련을 피할 수 있음

patience_train	•훈련 손실이 더 이상 개선되지 않는 것으로 간주되기 전에 대기해야 할 단계 수
patience_validation	•유효성 검사 손실이 더 이상 개선되지 않는 것으로 간주되기 전에 대기해야 할 단계 수
delta	•새로운 최적값으로 간주되기 전에 오류가 개선되어야 하는 정도에 따른 최소 임계값

✓ overfit 규칙

- 유효성 검사 및 훈련 손실을 비교하여 모델이 훈련 데이터에 과도하게 적합한지 여부 감지
- 과적합을 방지하기 위한 표준 방법은 모델을 정규화하는 것

start_step	•유효성 검사 및 훈련 손실의 비교를 시작하는 단계
patience	•ratio_threshold가 모델 과적합 한도로 설정된 값을 초과하도록 허용되는 단계 수
ratio_threshold	•평균 유효성 검사 손실 및 평균 훈련 손실 간의 차이와 평균 훈련 손실의 최대 비율

✓ 디버그 규칙을 사용하여 새 시도 생성

- 디버그 규칙을 사용하여 새 시도를 생성 후 재훈련

```
entry_point_script = "xgboost_customer_churn.py"

trial = Trial.create(trial_name="framework-mode-trial-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime()))),
                  experiment_name=customer_churn_experiment.experiment_name,
                  sagemaker_boto_client=boto3.client('sagemaker'))

framework_xgb = sagemaker.xgboost.XGBoost(image_name=docker_image_name,
                                           entry_point=entry_point_script,
                                           role=role,
                                           framework_version="0.90.2",
                                           py_version="py3",
                                           hyperparameters=hyperparams,
                                           train_instance_count=1,
                                           train_instance_type='ml.m4.xlarge',
                                           output_path=s3:///{}//output'.format(bucket, prefix),
                                           base_job_name="demo-xgboost-customer-churn",
                                           sagemaker_session=sess,
                                           rules=debug_rules
                                           )

framework_xgb.fit({'train': s3_input_train,
                  'validation': s3_input_validation},
                  experiment_config={
                      'ExperimentName': customer_churn_experiment.experiment_name,
                      'TrialName': trial.trial_name,
                      'TrialComponentDisplayName': "Training",
                  })
```

- 프레임워크 모드의 XGBoost를 사용
- XGBoost를 프레임워크로 사용하면 내장 알고리즘으로 사용하는 것보다 유연성이 뛰어남
- 사전 처리 및 사후 처리 스크립트를 훈련 스크립트에 통합 기능 가능
- 세이지메이커 디버거가 훈련을 평가할 규칙 목록 지정가능

! pygmentize xgboost_customer_churn.py

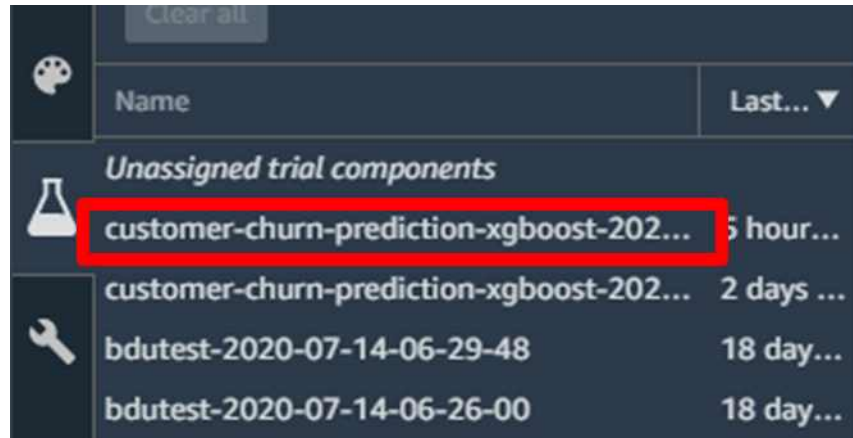
생각해보기



- 내장 XGBoost vs 프레임워크 XGBoost

기본 제공 XGBoost 알고리즘	프레임워크 XGBoost 알고리즘
<ul style="list-style-type: none"> • XGBoost를 내장 알고리즘으로 사용하면 XGBoost 훈련 기간에 대한 알림 • 고유한 XGBoost 훈련 스크립트를 통합하지 않으며 입력 데이터 세트에서 직접 실행 	<ul style="list-style-type: none"> • XGBoost를 프레임워크로 사용하면 고유한 훈련 스크립트를 사용처 지정할 수 있음 • XGBoost를 프레임워크로 사용하여 추가 데이터 처리를 훈련 작업에 통합할 수 있는 사용처 지정 훈련 스크립트를 실행

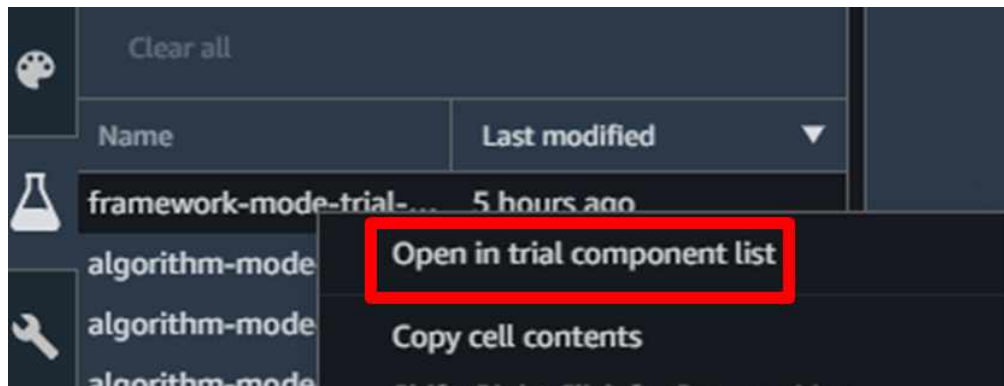
✓ 디버그 결과 확인

- 실험 목록에서 실험 이름을 두 번 클릭하여 시도 목록을 확인

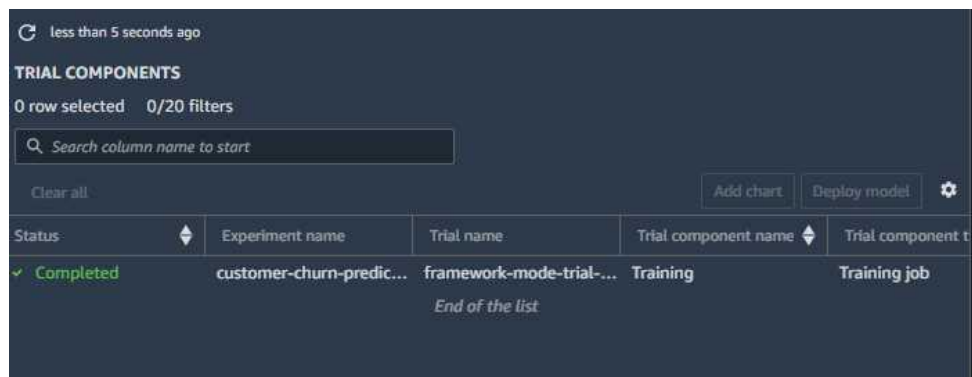


	Name	Last...
Unassigned trial components		
	customer-churn-prediction-xgboost-202...	5 hour...
	customer-churn-prediction-xgboost-202...	2 days ...
	bdutest-2020-07-14-06-29-48	18 day...
	bdutest-2020-07-14-06-26-00	18 day...

- 시도 목록에서 디버그 시도를 마우스 오른쪽 버튼으로 클릭하고 시도 구성 요소 목록에서 열기 선택
- 디버그 시도명 : framework-mode-trial...

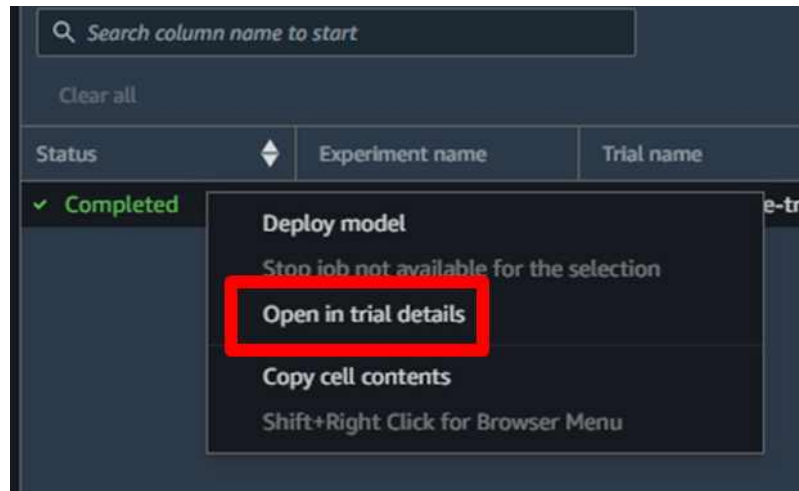


- 실행된 시도 구성요소 목록

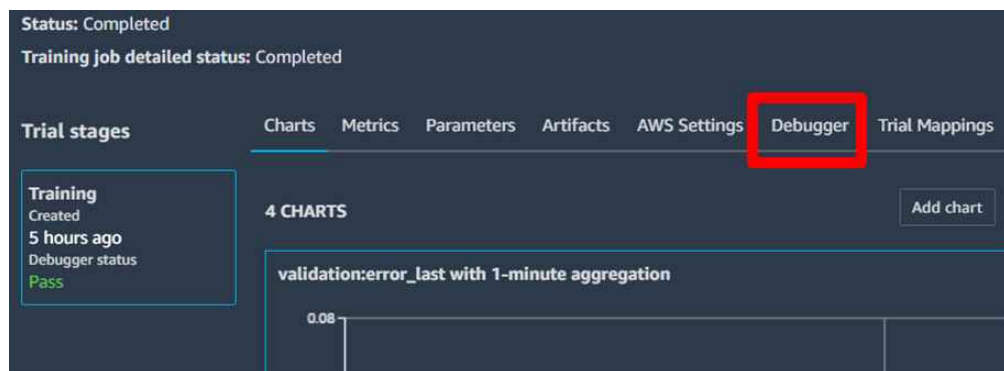


Status	Experiment name	Trial name	Trial component name	Trial component t
✓ Completed	customer-churn-predic...	framework-mode-trial-...	Training	Training job

- 시도 구성 요소 목록에서 시도를 마우스 오른쪽 버튼으로 클릭하고 시도 세부 정보에서 열기 선택



- 시도 세부 정보에서 디버그 클릭



- 디버그를 지정한 각 디버그 규칙에 대한 결과를 확인 가능

Charts	Metrics	Parameters	Artifacts	AWS Settings	Debugger	Trial Mappings
						⚙️
Status	Last modified	Rule name	Job ARN			
No Issues Found	23 minutes ago	LossNotDecreasing	arn:aws:sagemake...			
No Issues Found	23 minutes ago	Overtraining	arn:aws:sagemake...			
No Issues Found	23 minutes ago	Overfit	arn:aws:sagemake...			

평가하기

1. 유효성 검사 및 훈련 손실을 비교하여 모델이 훈련 데이터에 과도하게 적합한지 여부를 감지하는 디버깅 규칙으로 옳은 것을 고르시오.

- ① Overfit 규칙
- ② Overtraing 규칙
- ③ LossNotDecreasing 규칙
- ④ TreeDepth 규칙

- 정답 : ① 번

해설 : Overfit 규칙은 유효성 검사 및 훈련 손실을 비교하여 모델이 훈련데이터에 과도하게 적합한지 여부를 감지하며, 과적합을 방지하기 위한 표준 방법은 모델을 정규화하는 것입니다.

2. 세이지메이커에서 기본 제공된 알고리즘을 사용할 경우 고유한 훈련 스크립트를 사용자가 지정할 수 있다.(O/X)

- 정답 : X

해설 : 세이지메이커에서 고유한 훈련 스크립트를 사용자가 지정하기 위해서는 프레임워크로 제공된 알고리즘을 사용해야 합니다.

학습정리

1. 훈련 결과 시각화

- 시각화할 시도(Trial) 선택
- 차트 속성 구성
- 차트 모니터

2. 훈련 결과 디버깅

- 훈련 작업 분석 규칙 지정
- 디버그 규칙을 사용하여 새시도 생성
- 디버그 결과 확인