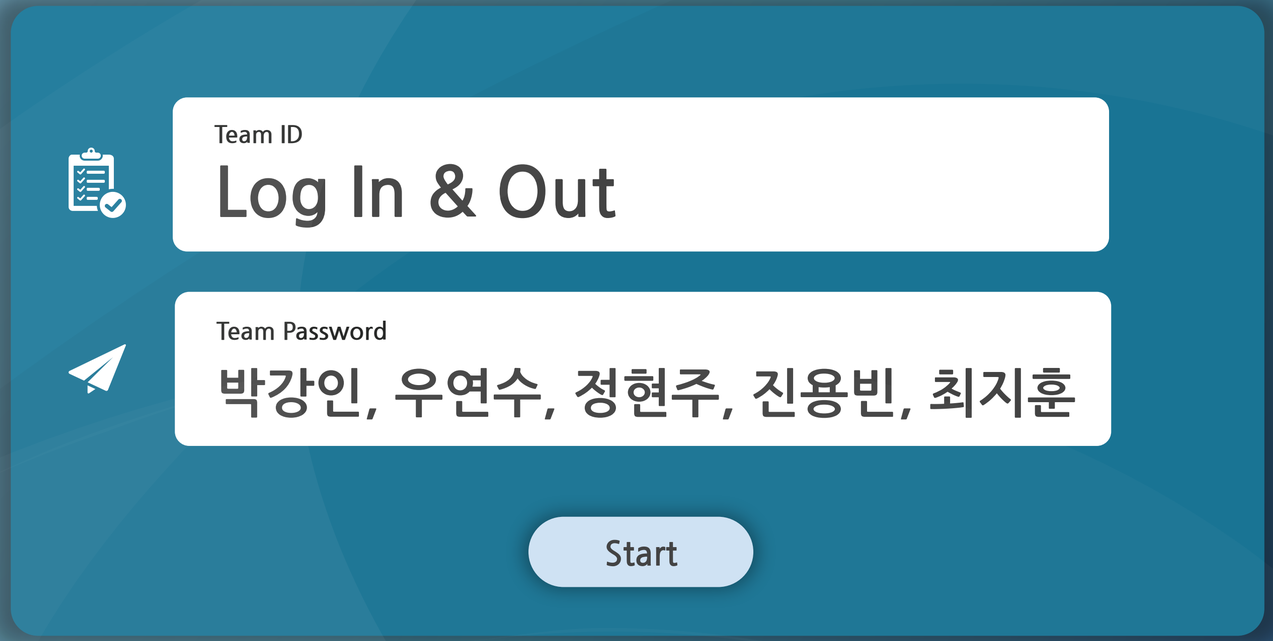
로그 분석을 통한 보안 위험도 예측



Dacon 로그 분석을 통한 보안 위험도 예측 AI 경진대회

목차

01. 프로젝트 개요

01 프로젝트 개요

TEAM
Log In & Out

박강인(팀장), DA
프로젝트 총괄
프로젝트 산출물 관리
개발환경 구축
데이터 전처리 및 모델링

우연수, AA
기술 탐색 및 분석 방향 설계
웹 서비스 기획
데이터 전처리 및 모델링

진용빈, DA
로그 데이터 분석 및 정의
도메인 자료 분석
데이터 전처리 및 모델링

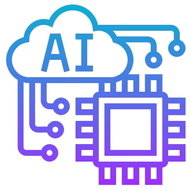
정현주, TA
데이터 전처리 및 모델링
로그 데이터 시각화
웹 서비스 구현

최지훈, TA
모델 탐구
데이터 전처리 및 모델링
로그 데이터 분석 및 정의

01 프로젝트 개요

파이썬 기반 응용 AI 개발자 양성과정

- 실무 기반의 Python 교육 과정 이수
- Machine Learning, Deep Learning 이론 및 실습
- 자연어처리(NLP) 실무 프로젝트 진행

아시아경제 교육센터

2020.12.21 ~ 2021.06.04

Dacon AI 경진대회 2021.04.19 ~ 2021.05.14

- 로그데이터를 통한 시스템 보안 위험도 등급 예측
- 기존에 없던 패턴의 공격을 탐지
- 분야 : 보안 | 로그 | 자연어 | 이상치 탐지

DACON
DATA TO VALUE

KAERI
한국원자력연구원
Korea Atomic Energy Research Institute

INNOPOLIS
연구개발특구진흥재단

01 프로젝트 개요

로그의 위험도를 예측하고
기존에 없던 패턴의 공격 탐지

AS IS

패턴 기반 기존 모니터링의 한계

규칙이 너무 많거나 잘못 설정되면
보안 효과 감소

다양한 사이버 테러 위험 증가

기존에 없던 패턴의 로그에 대한
규칙이 존재하지 않음

비효율적인 업무 프로세스

관리자의 지속적인 모니터링 요구

AI 기술을 활용한
로그 분석 시스템

프로젝트

TO BE

AI를 통해 패턴 기술의 한계 극복

AI기술을 적용하여
정교한 자동 모니터링 시스템 구축

사이버 테러 위험 감소

새로운 패턴의 위협도
신속하게 인지하여 대응 준비

효율적인 업무 프로세스

자동 모니터링 시스템을 도입하여
관리자는 통찰력이 필요한 로그에만 집중

01 프로젝트 개요

프로젝트 아키텍처

응용

로그 분석 및 위험도 예측
발생한 로그를 분석하여 위험도 예측
위험도에 따른 전략 수립으로 비용 절감

최적화
예측 모델을 지속적으로 개선
장기적인 고객 만족 추구

서비스 구현
분석 결과를 시각화하고 위험도를 예측하는
서비스 제공

python
jupyter
ANACONDA
scikit learn
TensorFlow
django
KiSTi
국가슈퍼컴퓨팅센터

기술

자연어처리(NLP) 기술
정규표현식을 활용한 효율적인 마스킹
단어를 수치화하여 연산

머신러닝/딥러닝 모델
기존 패턴 알고리즘의 한계를
극복하는 AI 기술 적용
정확하고 신속한 대응 가능

이상치 탐지
위험도 판단에 Threshold 도입
새로운 유형의 위험도 탐지 기능

01 프로젝트 개요

개발PC 대비 슈퍼컴퓨팅 시스템의 처리 속도가 15배 빠름

비교	국가 슈퍼컴퓨팅 시스템	개발PC(로컬)
작업 모델	훈련 데이터 : 472,972개 분류 모델 : ExtraTreesClassifier 파인튜닝 모델 : RandomizedSearchCV	
하드웨어 SPEC	CPU : Intel Xeon Ivy Bridge (E5-2670) / 2.50GHz (10-core) / 2 socket RAM : 노드 당 128GB DDR3 Memory GPU : Lenovo nx360-m4	CPU : Intel Core i7 5700HQ RAM : 8GB GPU : GeForce GTX 960M
하이퍼 튜닝 소요시간	**1,496 초 (약 20분 소요)**	**17,397 초 (약 5시간 소요)**

KiSTi 국가슈퍼컴퓨팅센터 제공

01 프로젝트 개요

서비스 워크플로우

1.데이터 분석

보안 도메인 지식을 쌓고,
Dacon에서 제공한
훈련 데이터를 분석했습니다.

2.데이터 전처리

숫자가 가지는 의미에 따라
마스킹 처리하고,
이상치와 중복 데이터를
제거했습니다.

3.모델링

AI모델 예측 결과에 임계치
(Threshold)를 적용해
테스트 데이터의 위험도 0~7를
분류했습니다.

4.서비스 구현

웹프레임워크 django로
데이터 분석 결과를 시각화하고
예측 서비스를 구현했습니다.

02. 데이터 분석 및 전처리

02 데이터 분석 및 전처리

로그 데이터 소개

과제

1. 위험도 0~6 분류
2. 기존에 없던 위험도 7 탐지

컬럼 명	컬럼 내용
id	데이터 식별자
level	보안 위험 등급(0~6)
full_log	데이터 전체 로그

기존 패턴 및 새로운 위험요소 존재

테스트 데이터

1,418,916개

	id	full_log
0	1000000	Feb 8 15:47:26 localhost kibana: {"type":"err...
1	1000001	Sep 24 03:46:39 localhost kibana: {"type":"err...
2	1000002	type=SYSCALL msg=audit(1611888200.428:210563):...

예측

기존에 확인된 위험도 0~6 포함

훈련 데이터

472,972개

	id	level	full_log
0	0	0	Sep 24 10:02:22 localhost kibana: {"type":"err...
1	1	0	Feb 8 16:21:00 localhost logstash: [2021-02-0...
2	2	0	Jan 13 01:50:40 localhost kibana: {"type":"err...

검증

새로운 위험도 검증용

검증 데이터

3개

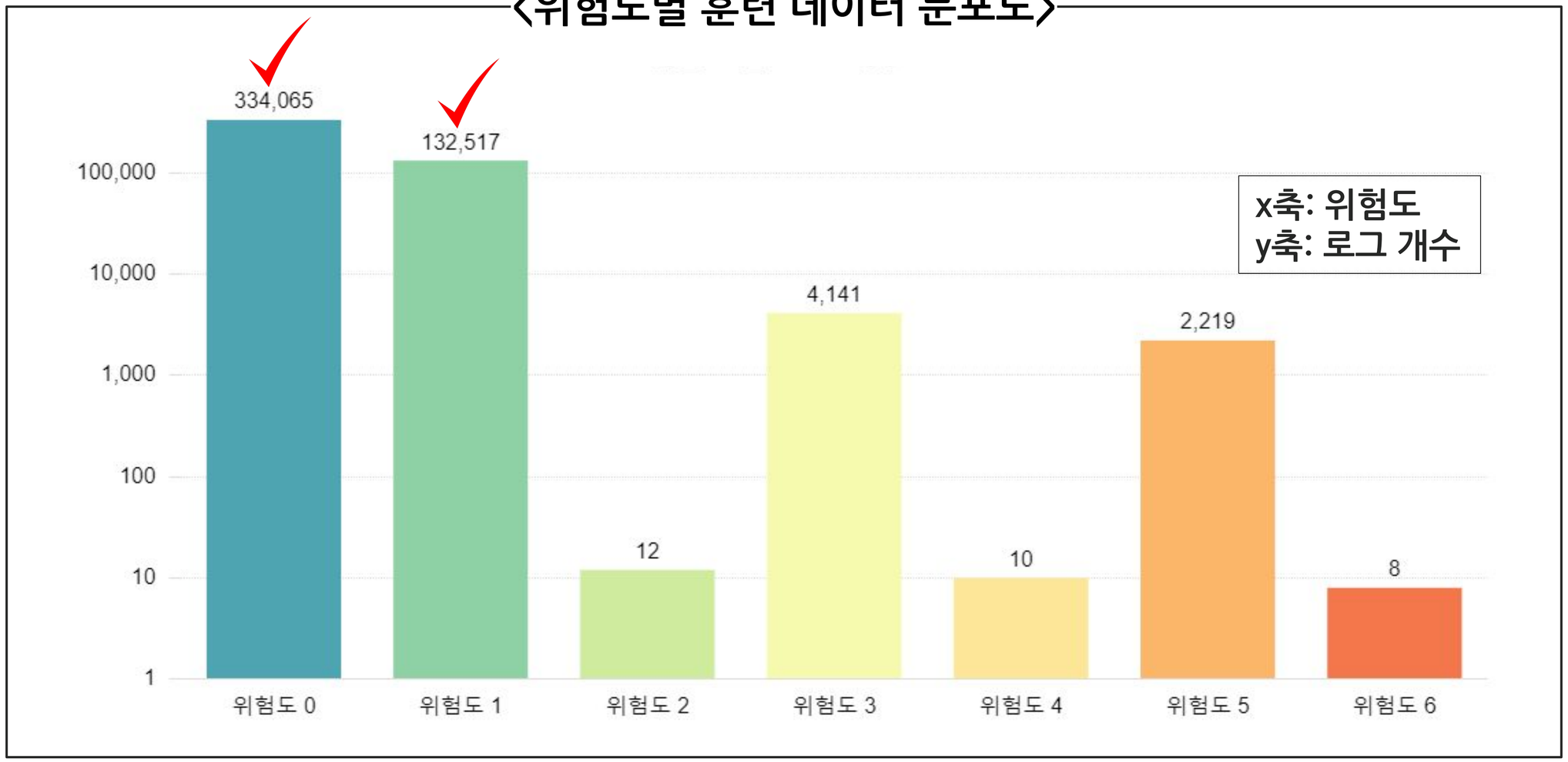
	full_log
0	type=ANOM_PROMISCUOUS msg=audit(1600402733.466...
1	oscap: msg: "xccdf-result", scan-id: "00016007...
2	Sep 22 10:56:19 localhost kernel: Out of memor...

데이터 제공 : Dacon, 한국원자력연구원

02 데이터 분석 및 전처리

전체 훈련 데이터의 98%가 위험도 0과 위험도 1에 분포하는 불균형 데이터

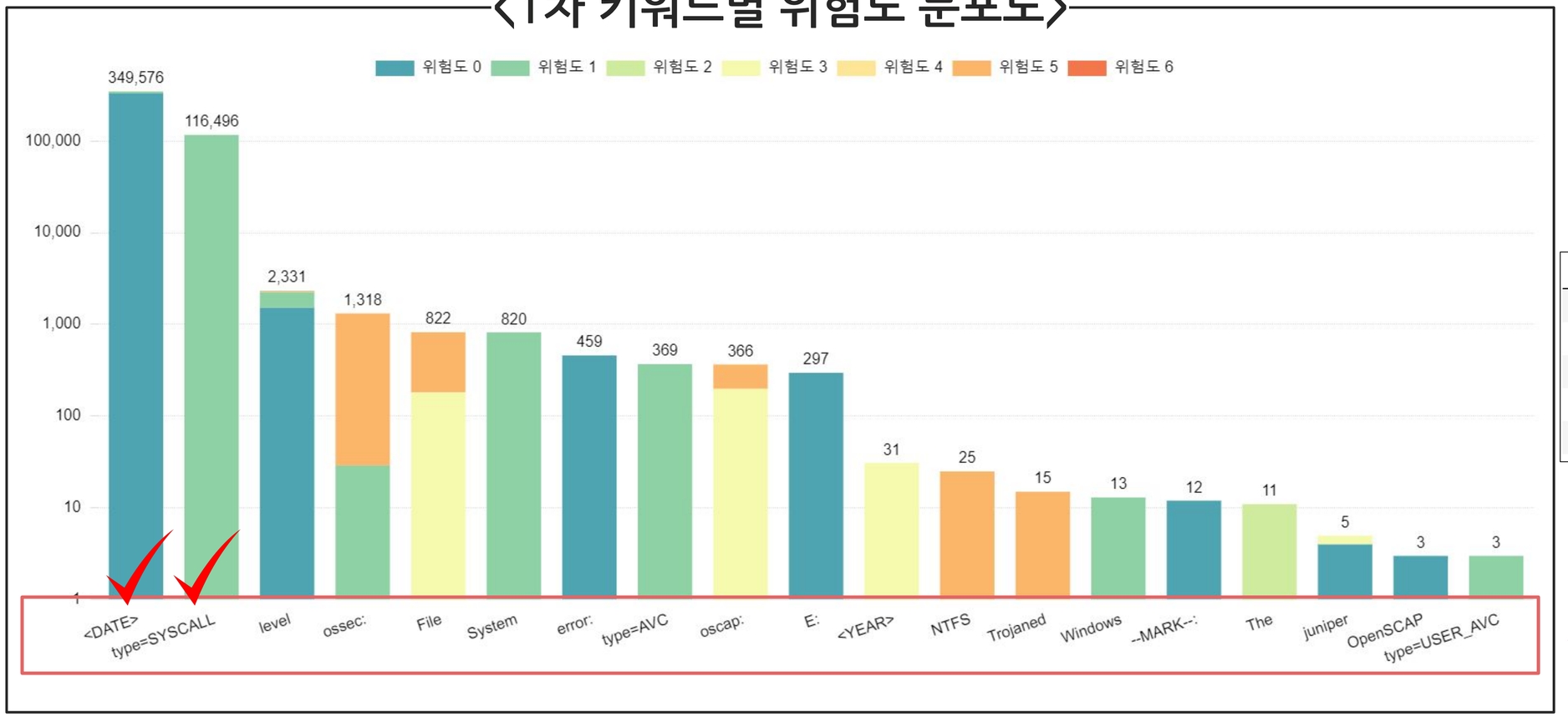
<위험도별 훈련 데이터 분포도>



02 데이터 분석 및 전처리

로그의 첫번째 키워드를 추출하여 1차 키워드별 위험도 분포를 확인

〈1차 키워드별 위험도 분포도〉



로그의 첫 키워드

id		
0	0	Sep 24 10:02:22 localhos
1	0	Feb 8 16:21:00 localhos
2	0	Jan 13 01:50:40 localhos
3	0	Jan 4 10:18:31 localhos
4	1	type=SYSCALL msg=audit(1

02 데이터 분석 및 전처리

1, 2차 키워드를 기준으로 데이터를 정렬, 특정 키워드에서 일관되게 나타나는 위험도를 확인

〈1, 2차 키워드별 위험도 분류표〉

full_log			위험도 Level						
first_word	패턴		0	1	2	3	4	5	6
oscap:	oscap: ERROR: Timeout	expired.	0	0	0	0	0	1	0
oscap:	msg: "xccdf-overview"	profile-title: "Common Profile for General-Purpose Systems"	0	0	0	0	0	2	0
oscap:	msg: "xccdf-overview"	profile-title: "PCI-DSS v3 Control Baseline for Red Hat Enterpri	0	0	0	2	0	0	0
oscap:	msg: "xccdf-result"	Configure Period	severity: "medium"	0	0	0	0	0	5	0
oscap:	msg: "xccdf-result"	Configure auditd	severity: "medium"	0	0	0	0	0	5	0
oscap:	msg: "xccdf-result"	Configure auditd	severity: "medium"	0	0	0	0	0	4	0
oscap:	msg: "xccdf-result"	Configure auditd	severity: "medium"	0	0	0	0	0	4	0
oscap:	msg: "xccdf-result"	Disable At Servic	severity: "low"	0	0	0	1	0	0	0
oscap:	msg: "xccdf-result"	Disable Automat	severity: "low"	0	0	0	5	0	0	0
oscap:	msg: "xccdf-result"	Enable GNOME3 S	severity: "medium"	0	0	0	0	0	5	0
oscap:	msg: "xccdf-result"	Enable GNOME3 S	severity: "medium"	0	0	0	0	0	6	0
oscap:	msg: "xccdf-result"	Enable Smart Car	severity: "medium"	0	0	0	0	0	6	0
oscap:	msg: "xccdf-result"	Ensure /var/log L	severity: "low"	0	0	0	3	0	0	0
oscap:	msg: "xccdf-result"	Ensure /var/log/a	severity: "low"	0	0	0	3	0	0	0
oscap:	msg: "xccdf-result"	Ensure Logrotate	severity: "low"	0	0	0	4	0	0	0

1, 2차 키워드

02 데이터 분석 및 전처리

서로 다른 위험도에서 내용이 중복되는 데이터를 확인

〈로그별 위험도 분류표〉

full_log	0	1	3	5
level : 3, log : Unable to find OID for PIC: i2c-id	2	1	0	0
level : 3, log : Unable to fork task-name: error-message	3	4	0	0
level : 3, log : Unable to fork: error-message	3	1	0	0
level : 3, log : Unable to fork: too many child processes	1	0	0	0
level : 3, log : Unable to generate OID for identifier (error-message)	1	1	0	0
level : 3, log : Unable to generate OID: oid (error-message)	1	0	0	0
level : 3, log : Unable to initialize event library: error-message	1	0	0	0
level : 3, log : Unable to listen on pathname	0	1	0	0
level : 3, log : Unable to listen on socket file-descriptor: error-message (errno error-code)	1	0	0	0
level : 3, log : Unable to locate table table-name for identifier (return-value)	1	0	0	0
level : 3, log : Unable to lock PID file pathname: error-message	0	1	0	0
level : 3, log : Unable to lock PID file: error-message	1	0	0	0
level : 3, log : Unable to lock PID file; another program-name was running	1	0	0	0

02 데이터 분석 및 전처리

	전처리 내용	코드

마스킹	숫자가 가진 의미를 살려 마스킹
	
	`<NUM>, <HEX>` : 숫자, 16진수
	`<TIME>, <YEAR>, <DAY>, <DATE>` : 시간, 연도, 날짜
	`<IP>` : IP 주소
		`PATTERNS = [('\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z?', '<TS>'),`
		`('\d{4}(?= (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec))', '<YEAR>'),`
		`('\d+\.\d+\.\d+\.\d+(?:\.\d+)?', '<IP>'),`
		`('(?<![0-9a-fA-F])0x[0-9a-fA-F]+(?=\W|$)', '<HEX>')]`
		
		`def apply_masking(df):`
		`for pat, repl in PATTERNS:`
		`df['full_log'] = df['full_log'].str.replace(pat, repl, regex=True)`

특수문자	단어 간 경계를 명확히 하기 위해
	특수문자 사이와 앞뒤에 공백을 추가
	
	전 `"pid":<NUM>,"level":"error"`
	후 `"pid":<NUM> , "level":"error"`
		`PATTERNS = [('\]\[', '] ['),`
		`(',', ' , '),`
		`('\s+', ' ')]`
		
		`def apply_masking(df):`
		`for pat, repl in PATTERNS:`
		`df['full_log'] = df['full_log'].str.replace(pat, repl, regex=True)`

중복 데이터	빈도수가 높은 위험도의 로그만 남기고
	나머지를 제거
		`for index in dupl.index:`
		`targets = train[(train['full_log']==index) & (train['level']!=dupl['level'][index])].index`
		`train.drop(index=targets, inplace=True)`

03. 모델링

03 모델링

모델명	특징	코드
랜덤 포레스트 (Random Forest)	과적합이 적게 발생하고 결측치의 비율이 높아도 높은 정확도를 보임	`clf = RandomForestClassifier(n_estimators=100, n_jobs=-1)`

익스트림 부스트
(XGboost)
	GBM에 비해 학습 속도가 빠름
	과적합이 적게 발생
		`xgb_clf = XGBClassifier(n_estimators = 1000, learning_rate = 0.1 , max_depth = 3,`
		`objective = 'multi:softmax', num_class = 7)`
		`evals = [(eval_x, eval_y)]`
		`xgb_clf.fit(train_x, train_y, early_stopping_rounds = 200,`
		`eval_metric="merror", eval_set = evals, verbose=True)`

라이트 지비엠 (LightGBM)	학습 속도가 빠르며 메모리 사용량이 적음	`LightGBM(boosting_type='gbdt', max_depth=30, learning_rate=0.1)`

캣 부스트
(Catboost)
	높은 예측 성능과 모델 튜닝의 간소화
	및 텍스트 처리 기능 제공
		`clf = CatBoostClassifier(**params)`
		
		`params = {`
		`'task_type': 'GPU',`
		`'loss_function': 'MultiClass',`
		`"iterations" : 50000,`
		`"early_stopping_rounds" : 100}`
		
		`clf.fit(X_train, y_train, cat_features=cat_features, eval_set=(X_eval, y_eval), verbose=100, use_best_model=True)`

03 모델링

모델명	특징	코드

엑스트라 트리
(Extra Tree)
	결정 트리에 비해 편향과 분산을 감소하며 학습	`est = ExtraTreesClassifier(n_jobs=-1, n_estimators=500)`
		`params = {'criterion':['gini','entropy'],`
		`'max_features':randint(1,6),`
		`'max_depth':[3,10,None],`
		`'bootstrap':[True,False],`
		`'min_samples_leaf':randint(1,10)}`

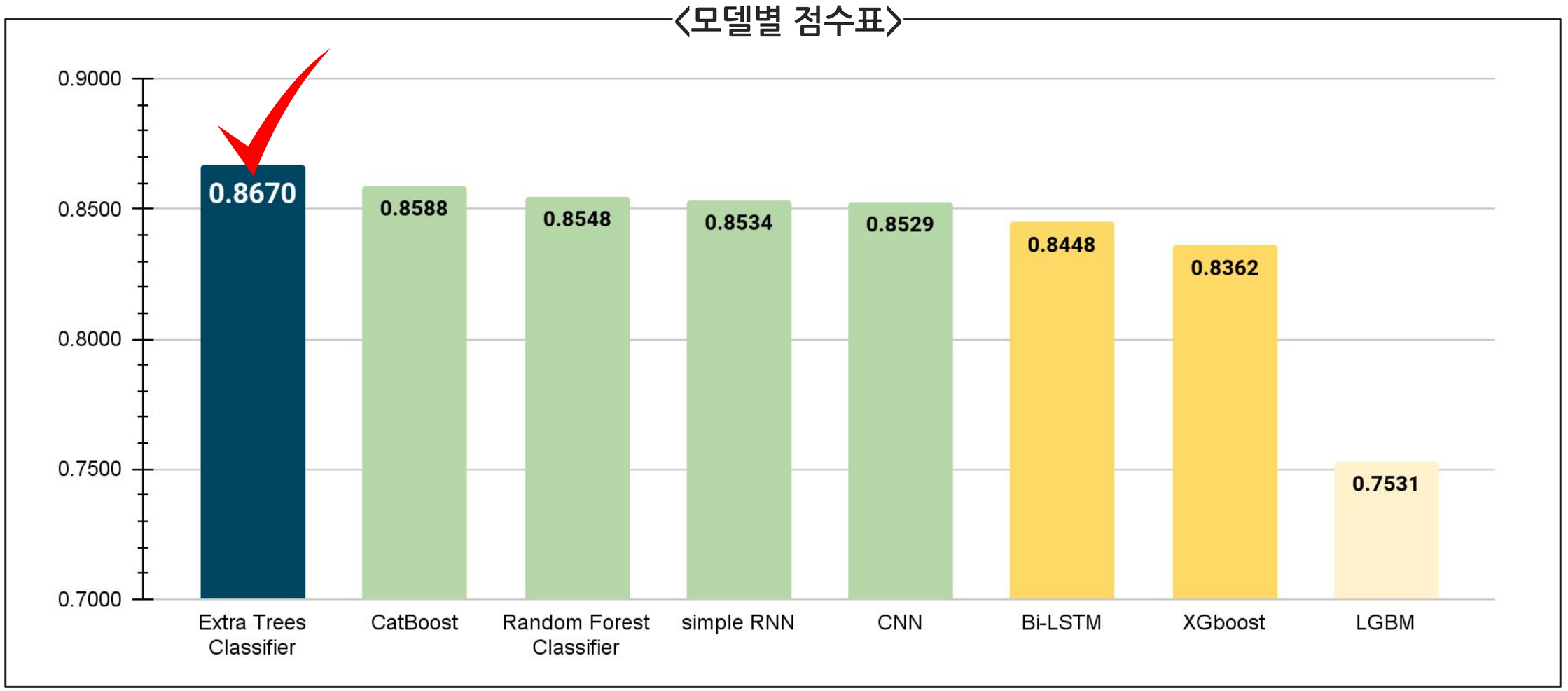
심플 RNN
(simple RNN)
	음성, 문자 등 순차적으로 등장하는
	데이터 처리에 적합한 모델
		`model = Sequential()`
		`model.add(Embedding(vocab_size, 200)) # 임베딩 벡터의 차원`
		`model.add(SimpleRNN(32)) # RNN 셀의 hidden_size는 32`
		`model.add(Dense(7, activation='softmax'))`
		`model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', metrics=['acc'])`
		`history = model.fit(X_train, Y_train, epochs=4, batch_size=64, validation_split=0.2)`

양방향 LSTM
(Bi-LSTM)
	뒤의 문맥까지 고려하기 위해 문장을 반대로 읽는
	역방향의 LSTM 셀을 함께 사용하는 모델
		`model = Sequential([Embedding(VOCAB_SIZE, EMBEDDING_DIM),`
		`Bidirectional(LSTM(32, recurrent_initializer='random_uniform',`
		`bias_initializer='he_uniform')),`
		`Dense(32, activation='relu'),`
		`Dense(7),`
		`Activation('softmax')`
		`], name="BI_LSTM_32")`

합성곱 신경망
(CNN)
	문장에서 n그램을 추출하고,
	텍스트의 맥락 정보를 특징으로 학습
		`model = Sequential()`
		`model.add(Embedding(vocab_size , EMBEDDING_DIM, input_length=X.shape[1]))`
		`model.add(Conv1D(32, 8, padding='valid', activation='relu'))`
		`model.add(MaxPooling1D(pool_size=2))`
		`model.add(Flatten())`
		`model.add(Dense(7, activation='softmax'))`
		`model.compile(optimizer='adam', loss = 'categorical_crossentropy', metrics = [F1_MACRO])`
		`history = model.fit(X_train, Y_train, epochs=20, batch_size=128,`
		`validation_split=0.2, callbacks=[cb_early_stopping, cb_checkpoint, cb_reduceLR])`

03 모델링

정밀도와 재현율의 평균 점수가 가장 높은 Extra Trees를 최종 모델 선정



최종 모델 상세 정보

- **TfidfVectorizer:** 문서별 단어 빈도에 가중치를 적용, 단어 수치화
- **max_features:** 빈도 순으로 포함시킬 단어의 최대 개수 지정
- **stratify:** 타겟 데이터 분포 비율을 유지하면서 평가데이터 샘플링
- **n_etimators:** 트리의 개수 지정

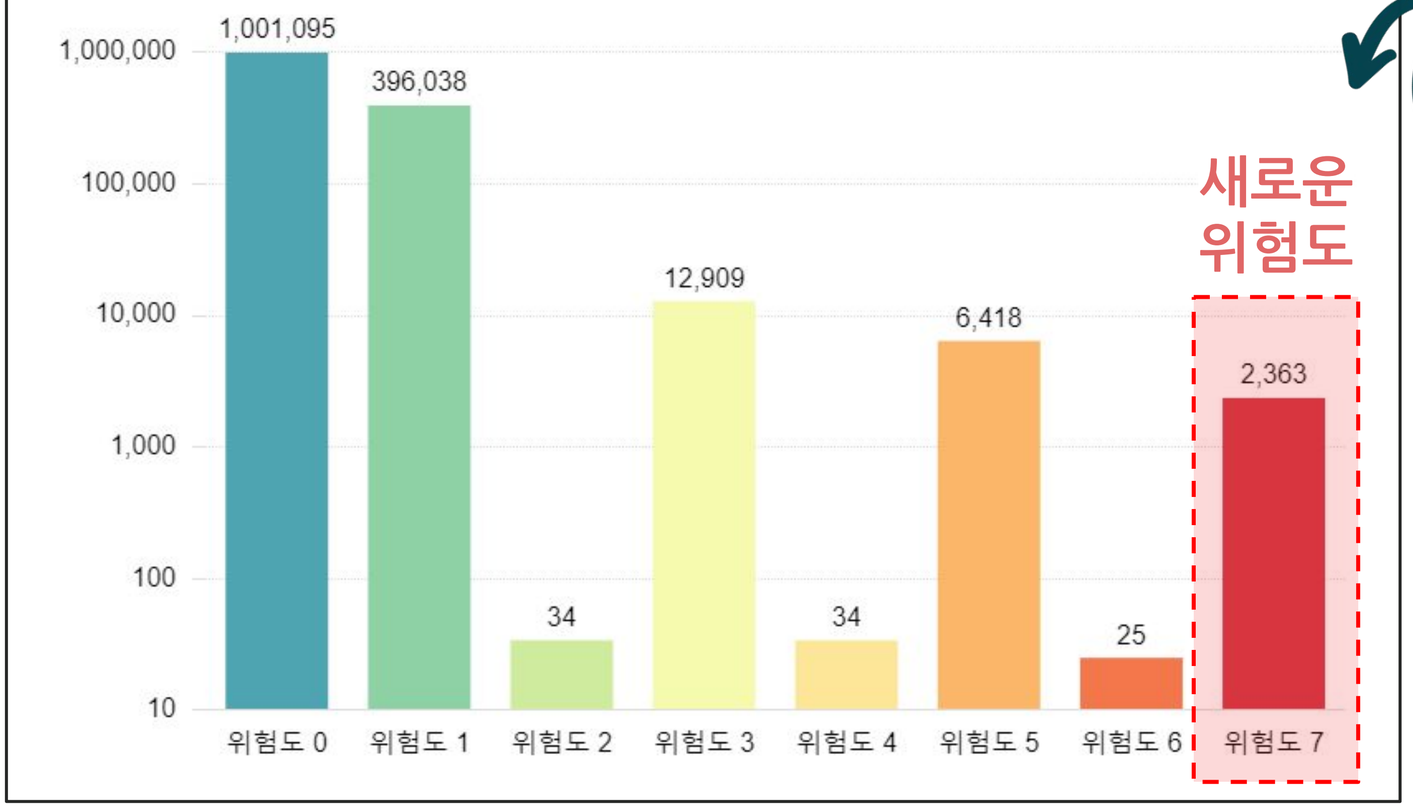
```
VOCAB_SIZE = 10000
vectorizer = TfidfVectorizer(analyzer='word', max_features=VOCAB_SIZE)
X_features = vectorizer.fit_transform(X_data)

X_train, X_eval, y_train, y_eval = train_test_split(
    X_features, y_data, test_size=0.08, random_state=100, stratify=y_data
)

clf = ExtraTreesClassifier(n_estimators=100, n_jobs=-1)
clf.fit(X_train, y_train)
```

04. 예측 및 검증 결과

04 예측 및 검증 결과

테스트 데이터의 예측 결과

〈임계치를 적용한 최종 결과〉



임계치 적용

〈기존 위험도로 예측한 결과〉


```
X_test = test['full_log']
X_test = vectorizer.transform(X_test)
results_proba = clf.predict_proba(X_test)
results = np.argmax(results_proba, axis=-1)
results[np.where(np.max(results_proba, axis=1) < THRESHOLD)] = 7
```

04 예측 및 검증 결과+

새로운 위험 패턴 탐지를 위해 임계치 도입

최고 예측 확률이 임계치 미만일 때 새로운 위험도 7로 분류

```
X_valid = validation['full_log']
X_valid = vectorizer.transform(X_valid)
valid_proba = clf.predict_proba(X_valid)

valid_proba
```

```
[[0.   0.99 0.   0.01 0.   0.   0.  ]  데이터 1
 [0.03 0.01 0.   0.19 0.   0.77 0.  ]  데이터 2
 [0.82 0.07 0.01 0.03 0.   0.05 0.02]] 데이터 3
```

〈데이터의 위험도별 예측 확률〉

예측 확률	위험도 0	위험도 1	위험도 2	위험도 3	위험도 4	위험도 5	위험도 6
데이터 1	0.	**0.99**	0.	0.01	0.	0.	0.
데이터 2	0.03	0.01	0.	0.19	0.	**0.77**	0.
데이터 3	**0.82**	0.07	0.01	0.03	0.	0.05	0.02

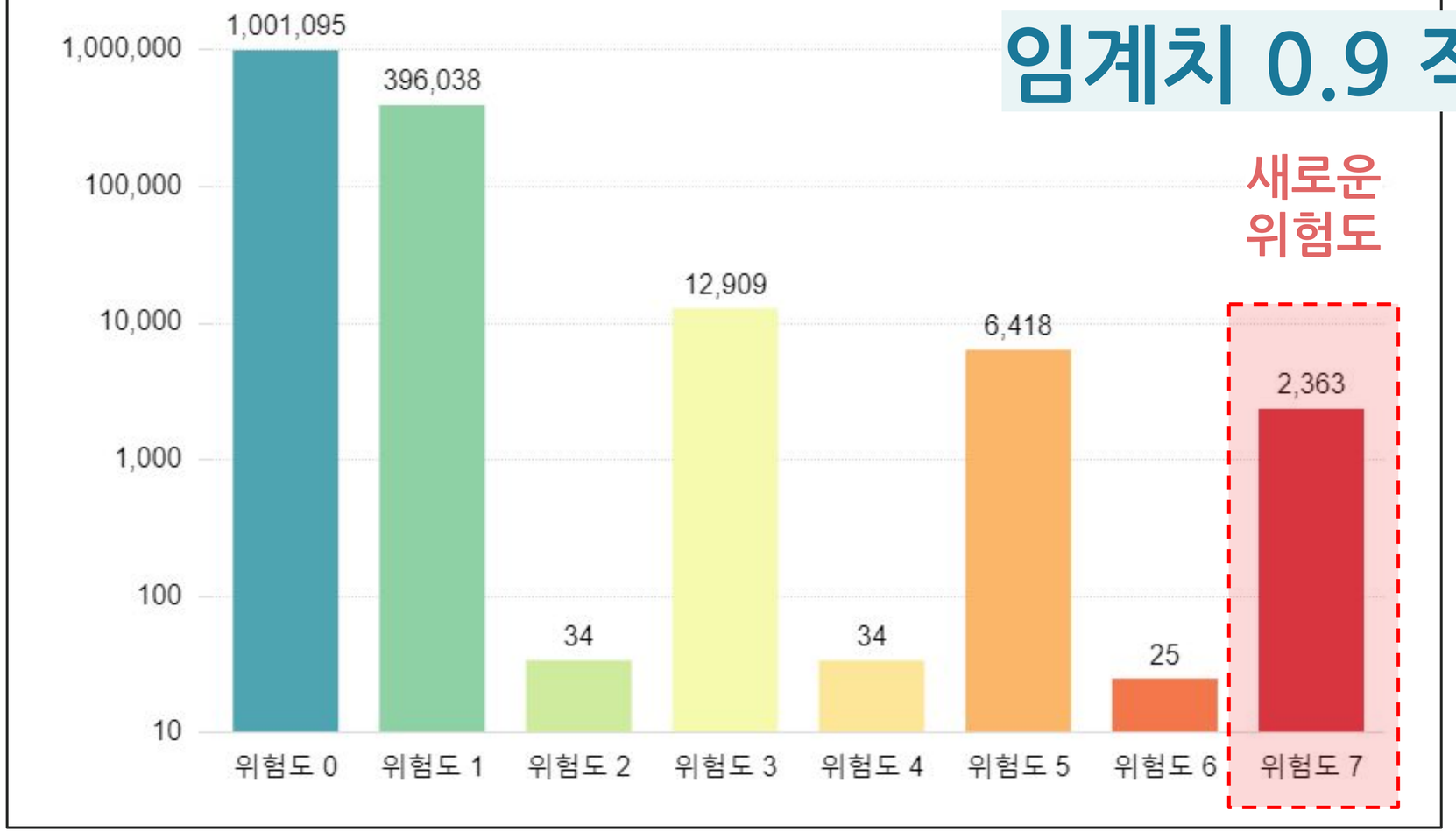
- **임계치**는 검증 데이터를 활용하여 적절하게 조정

최적 임계치 0.9

04 예측 및 검증 결과

테스트 데이터의 예측 결과

<임계치를 적용한 최종 결과>



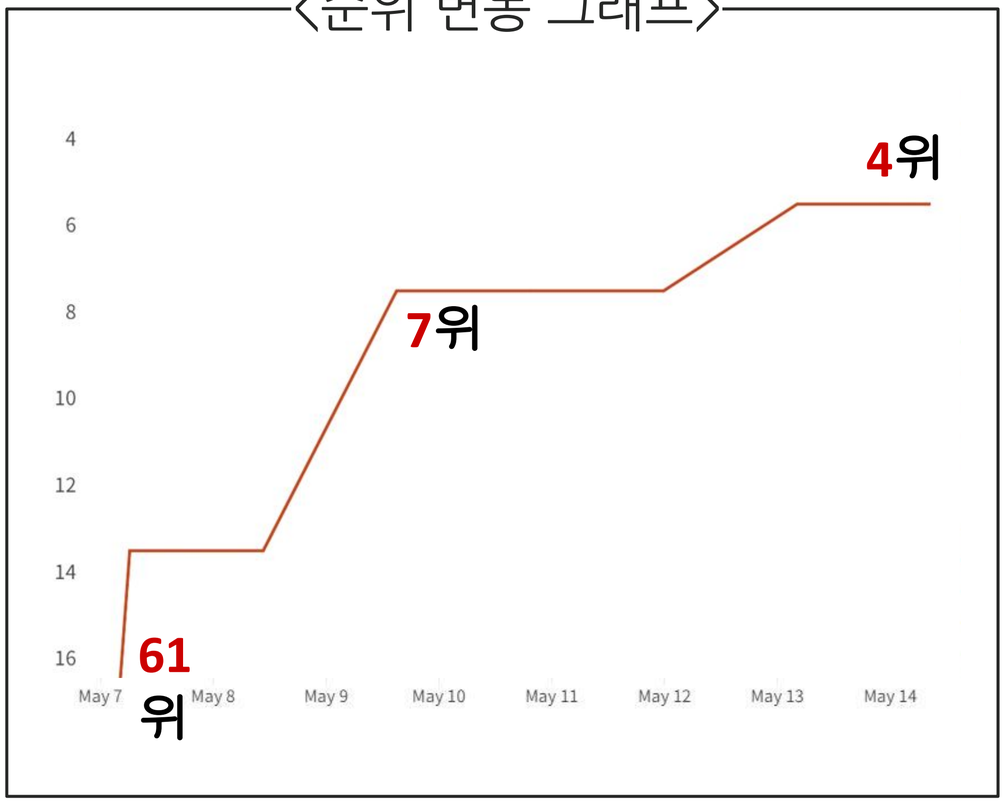
임계치 0.9 적용

새로운 위험도

04 예측 및 검증 결과

Dacon AI경진대회, 참가 445팀 중 최종 4위

〈순위 변동 그래프〉



〈최종 순위〉

#	팀	팀 멤버	최종점수
1	Team SsulleBal		0.93199
2	심		0.92294
3	최정명		0.89427
4	Log In&Out		0.88938
5	lumi		0.88179

05. 결론

05 결론

보안 위험도 등급을 예측 및 기존에 없던 패턴의 공격 탐지

NLP 기술 → 정확성

인공지능 → 신속성

이상 탐지 → 탐지력

효과적인 데이터 분석 ⟷ AI 활용 ⟷ 효율적인 위험도 분류

- 규칙과 패턴 기반 알고리즘의 한계 극복
- 자연어처리 기반 입력된 로그의 위험도 예측 및 새로운 유형의 로그를 감지할 수 있는 서비스 구축
- AI의 정확성과 신속성을 바탕으로 사이버 공격 대응시간을 현저히 줄이는 즉각적인 인사이트 제공

06. 서비스 시연

Q & A

감사합니다