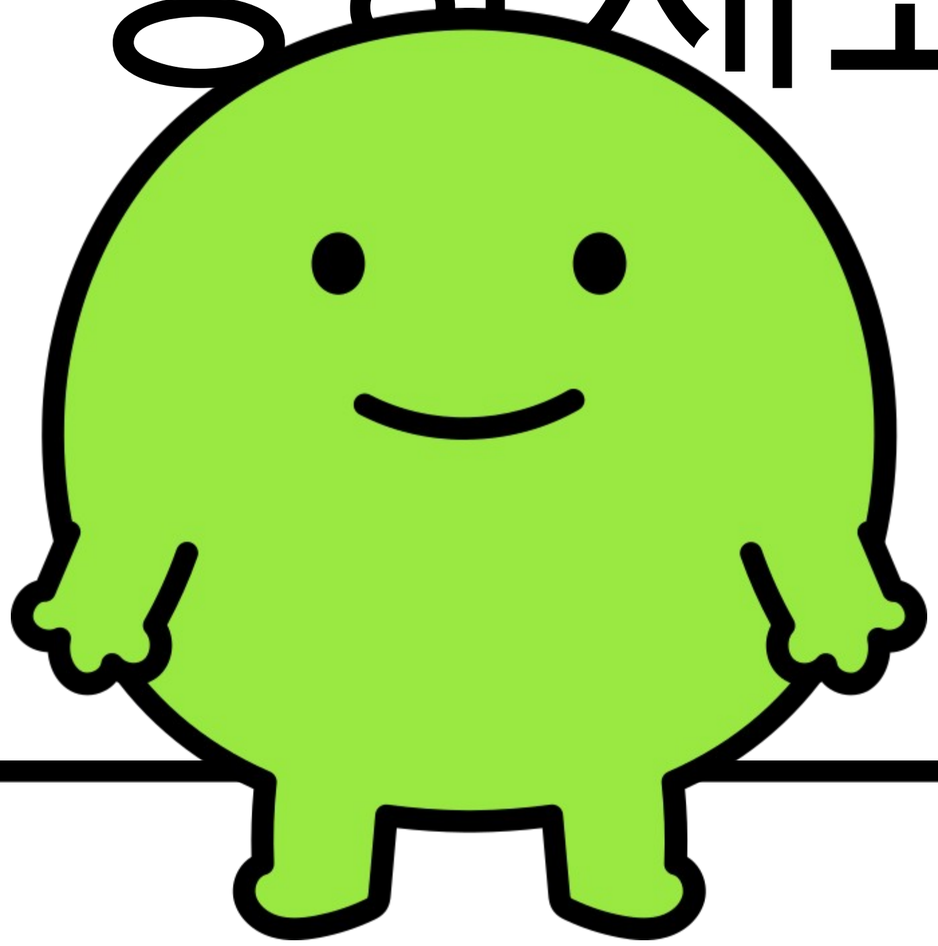
URL 악성 코드 분류

김미영 팀장

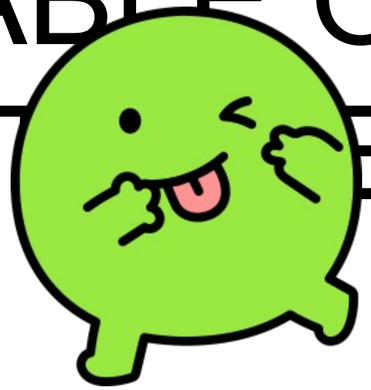
안녕하세요!



조장 황태현
임병남
우병준
박종호

Human-IT
교육센터

TABLE OF CONTENTS



프로젝트 개요

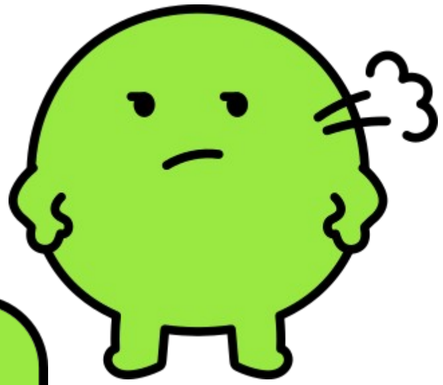
01

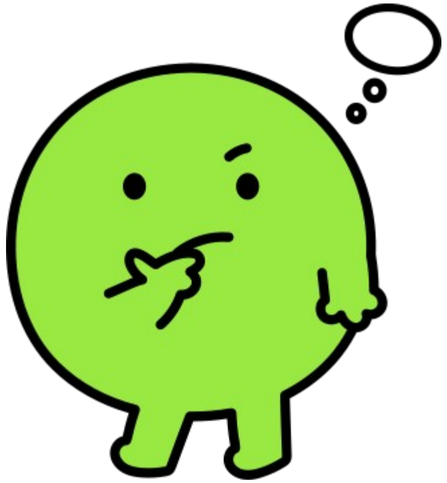
IDEATION

우리 프로젝트는 이렇게 시작됐어요!

택배 배송 확인하세요”, “OO은행 보안 점검” 같은 일상적인 문구로
위장된 링크 하나로 스마트폰이 해킹되거나 개인 정보가 유출되는
사고가 많습니다.

최근 사이버 보안 위협이 급증하면서,
단순한 URL 하나만으로도 치명적인 피해가 발생하고 있습니다.
데이터 분석팀과 백엔드 개발팀의 협업을 통해,
사용자가 입력한 URL이 정상인지 악성인지 실시간 분류하고
그 결과를 API 형태로 전달하는 서비스를 구축한 사례입니다.





악성 URL은 클릭 한 번으로 개인 정보 탈취, 악성코드 설치 등을 유발합니다.
URL만 있어도 분류 가능하여, 다양한 서비스에 연동이 용이합니다 (예: 메신저, 메일 시스템 등).
데이터 분석과 백엔드가 분리되어, 추후 다양한 알고리즘 혹은 데이터셋 적용이 가능합니다.

- 사이버 보안 강화

  악성 URL을 사전에 차단함으로써
  개인정보 유출, 해킹, 랜섬웨어 감염 등 심각한 보안 사고를 미연에 방지할 수 있습니다.

사용자 신뢰도 향상

- 서비스 내에서 안전한 링크만 허용됨으로써

  사용자들이 안심하고 서비스를 이용할 수 있는 환경이 만들어집니다.

빠르고 자동화된 판단 가능

- 사람이 일일이 확인하지 않아도,
  모델이 실시간으로 악성 여부를 자동 판별해 줍니다.
  → 운영 효율성 및 대응 속도 향상

다양한 서비스에 연동 가능

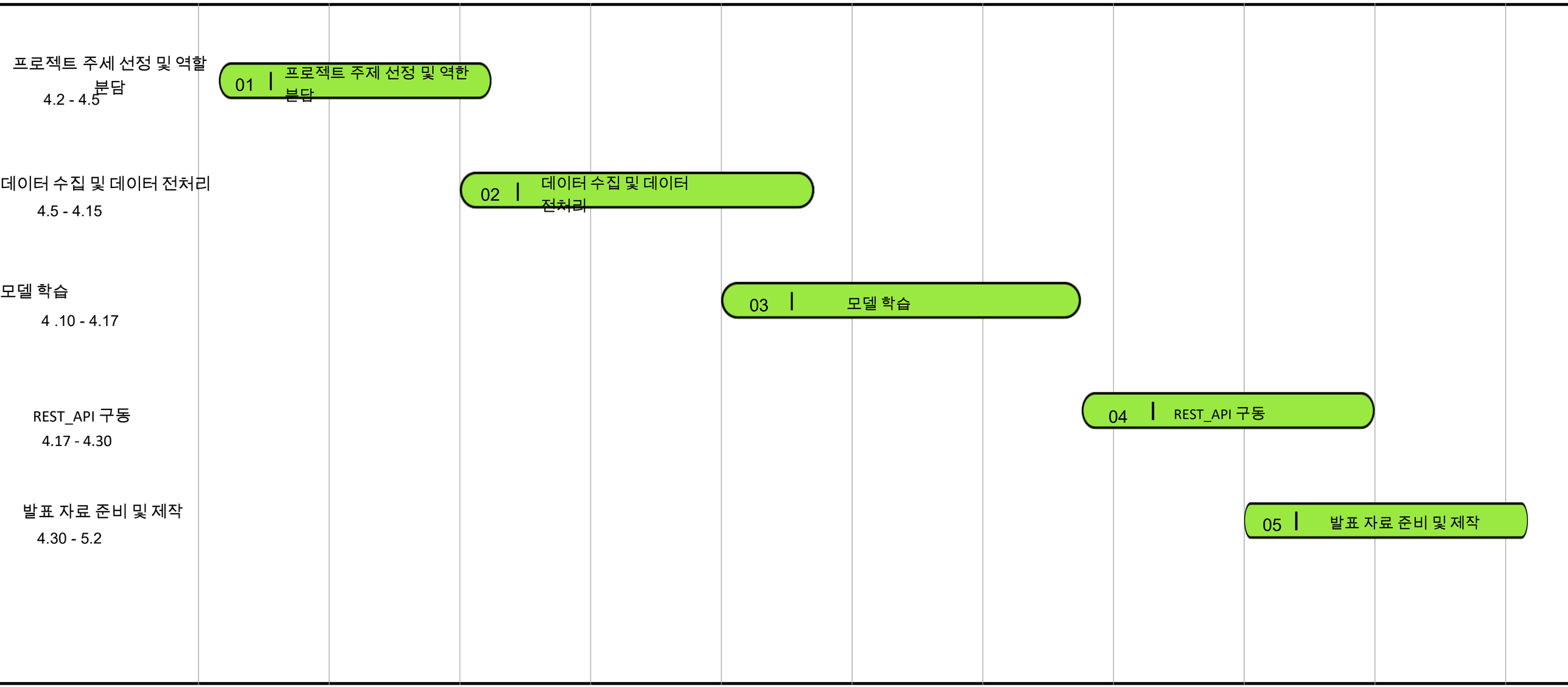
- 모델을 API 형태로 제공하면,
  메일 시스템, 메신저, 웹 브라우저 보안 등 다양한 플랫폼과 연동해 활용할 수 있습니다.

보안 사각지대 감소

- 특히 일반인이 구분하기 어려운 **URL**을 자동 분류함으로써
  소외된 사용자를 보호하고 사회 전반의 보안 수준 향상에 기여합니다.

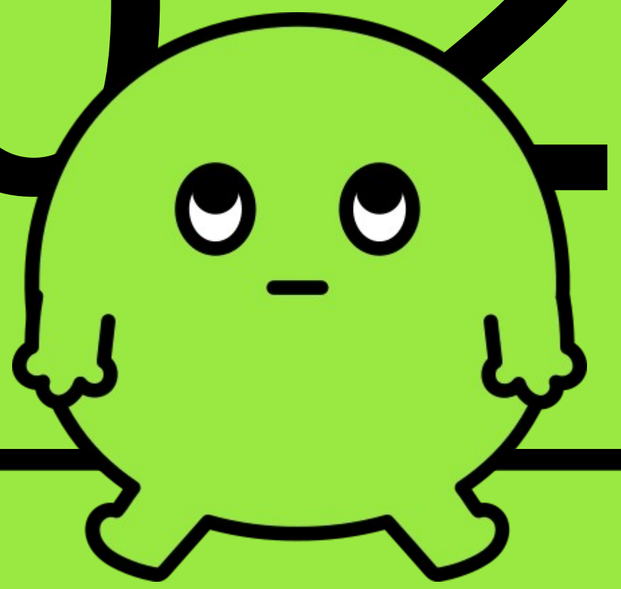
TIME TABLE

타임테이블



02

프로젝트 팀 구성 및 역할



IDEATION

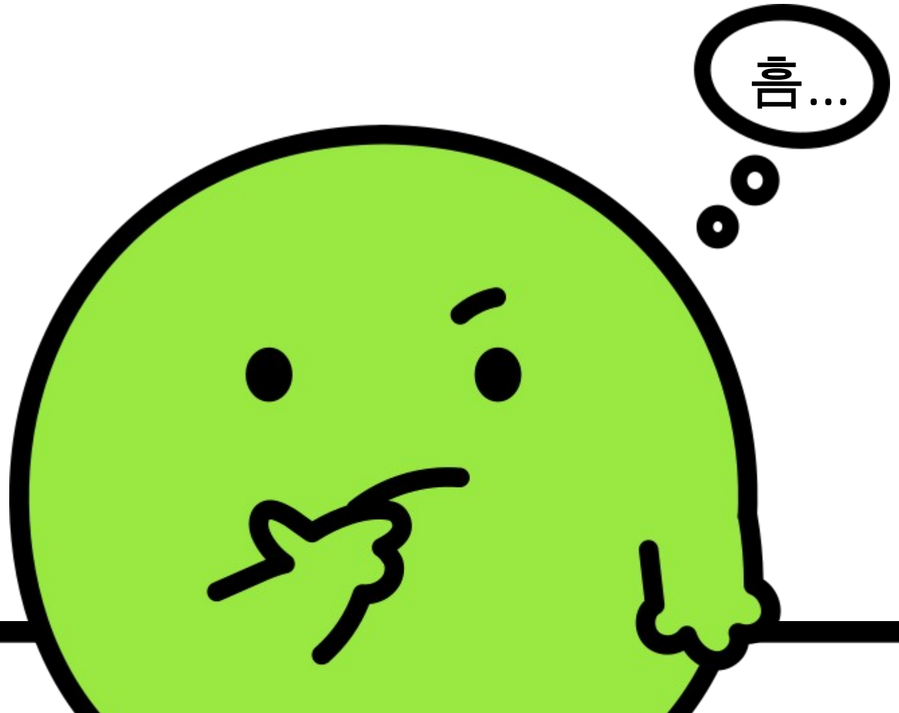
역할 분담

황태현

프론트 구현
모델 연동
백엔드 구현

우병준

데이터 전처리 및 정제
데이터 모델링
모델 성능 평가 및 튜닝



박종호

데이터 전처리 및 정제
데이터 모델링
모델 성능 평가 및 튜닝

임병남

데이터 전처리 및 정제
데이터 모델링
모델 성능 평가 및 튜닝

URL 악성 코드 분류

기술 스택 선정

01

TensorFlow

FastAPI

python™

React

node JS

프로젝트 수행 과정

03

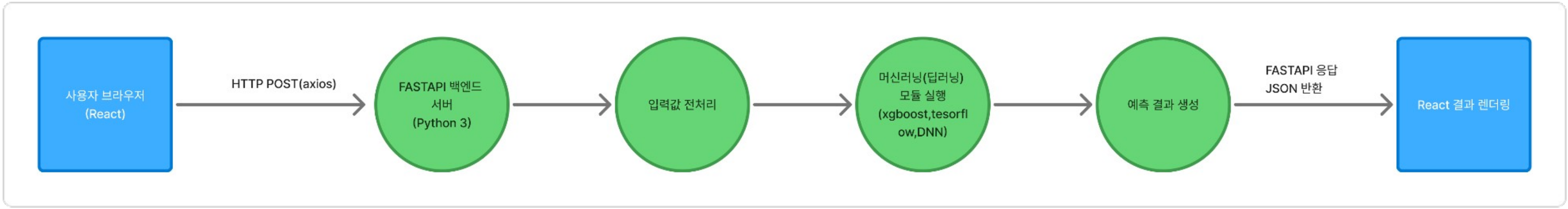
PRROJECT

URL 악성 코드 분류

아키텍처. 데이터 흐름

02

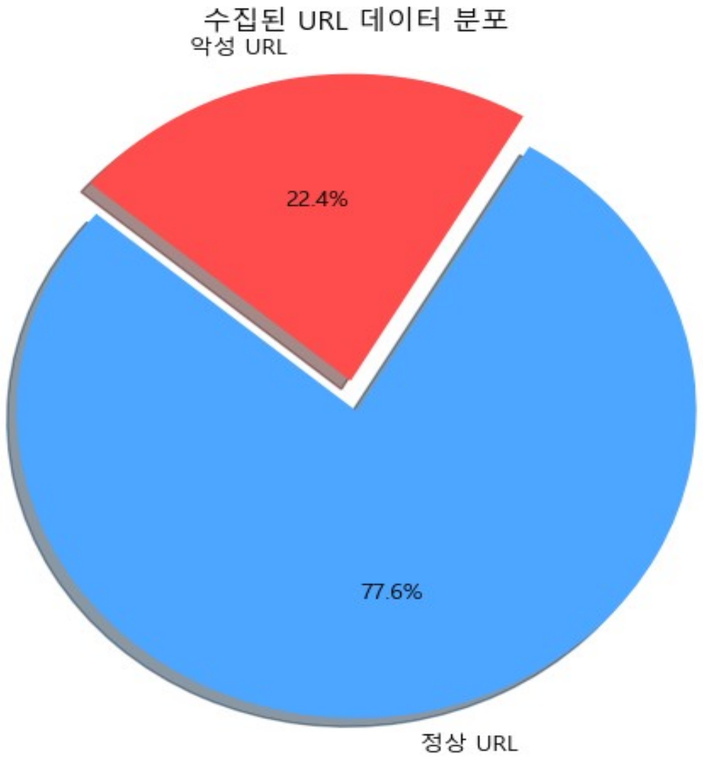
Section 1



데이터 전처리 및 모델링

01

데이터 비율



전체 약700만 건의 데이터 중

-정상 URL : 77.6%

-악성 URL : 22.4%

데이터 전처리 및 모델링

02 URL구조

예시 URL: https://www,example.com/login/reset?user=kim&id=123#reset

프로토콜 : https -> 프로토콜 존재 여부

도메인 : www.example.com -> 도메인 길이, 서브도메인 존재 여부

경로 : /login/reset -> ‘/’ 개수로 경로 깊이 추정

쿼리 파라미터 : ?user=kimid=123 -> 쿼리 존재 여부 , 전체 길이

프래그먼트 : #reset -> 프래그먼트 존재 여부, 전체 길이

데이터 전처리 및 모델링

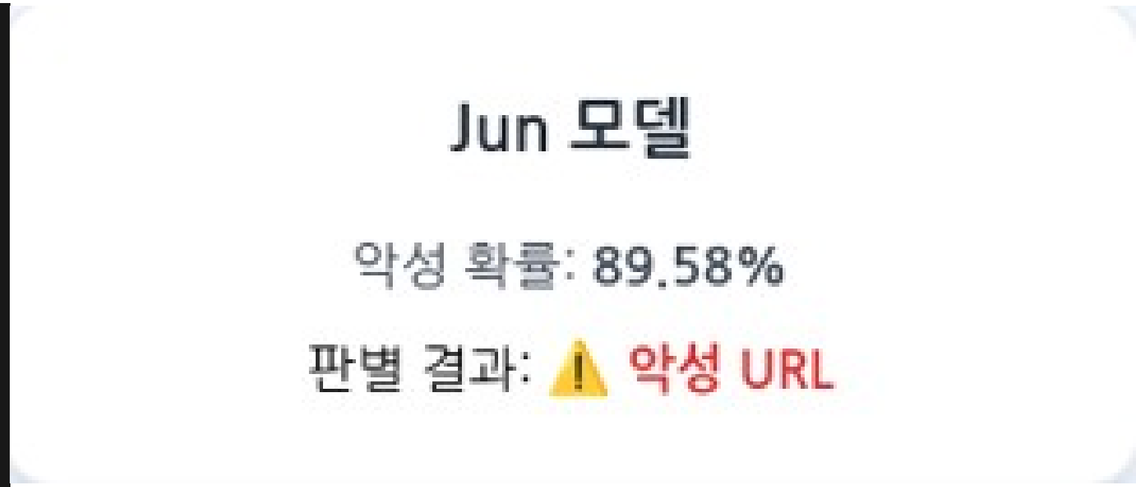
03 데이터전처리

```
suspicious_keywords = [
    'login', 'verify', 'account', 'update', 'secure', 'banking',
    'paypal', 'confirm', 'signin', 'auth', 'redirect', 'free',
    'bonus', 'admin', 'support', 'server', 'password', 'click',
    'urgent', 'immediate', 'alert', 'security', 'prompt'
]

additional_keywords = [
    'wallet', 'cryptocurrency', 'bitcoin', 'ethereum',
    'validation', 'authenticate', 'reset', 'recover', 'access',
    'limited', 'offer', 'prize', 'win', 'winner', 'payment',
    'bank', 'credit', 'debit', 'card', 'expire', 'suspension',
    'unusual', 'activity', 'document', 'invoice'
]

all_keywords = list(set(suspicious_keywords + additional_keywords))

contains_keyword = 0
keyword_count = 0
for keyword in all_keywords:
    if re.search(r'\b' + keyword + r'\b', url, re.IGNORECASE):
        contains_keyword = 1
        keyword_count += 1
```



https://nid.naver.com/nidlogin.login?mode=form&url=https://www.naver.com

login, account 등 키워드 기반 탐지로 인해
정상적인 네이버 로그인 URL도 악성으로
분류되는 사례발생

데이터 전처리 및 모델링

03 데이터전처리

```python
def url_is_whitelisted(url):
    trusted_domains = [
    # 1. 포털 / 검색엔진
    'naver.com', 'daum.net', 'google.com', 'bing.com', 'yahoo.com',

    # 2. 소셜 미디어 / 커뮤니케이션
    'facebook.com', 'instagram.com', 'twitter.com', 'x.com', 'linkedin.com',
    'whatsapp.com', 'kakao.com', 'kakaocorp.com',

    # 3. 동영상 / 스트리밍
    'youtube.com', 'netflix.com', 'twitch.tv', 'tving.com', 'watcha.com',

    # 4. 쇼핑 / 이커머스
    'amazon.com', 'gmarket.co.kr', '11st.co.kr', 'coupang.com', 'ssg.com', 'wemakeprice.com',

    # 5. 금융 / 결제
    'paypal.com', 'kbfg.com', 'shinhan.com', 'hanafn.com', 'wooribank.com',
    'kakaobank.com', 'toss.im',

    # 6. 공공기관 / 교육
    'gov.kr', 'moe.go.kr', 'epeople.go.kr', 'pusan.ac.kr', 'ac.kr',

    # 7. IT / 기술
    'apple.com', 'microsoft.com', 'adobe.com', 'github.com', 'stackoverflow.com'
    ]

    try:
        domain = urlparse(url if '//' in url else '//' + url).netloc.lower()
        for trusted in trusted_domains:
            if domain.endswith(trusted):
                return True
        return False
    except:
        return False
```

신뢰할수 있는 도메인에 대해
화이트리스트를 적용하여 오탐 방지

데이터 전처리 및 모델링

03 데이터전처리

```
for brand in common_brands:
    if brand not in domain:
        similar = False
        # 비슷한 철자 패턴 확인
        patterns = [
            brand.replace('o', '0'),
            brand.replace('i', '1'),
            brand.replace('l', '1'),
            brand.replace('e', '3'),
            brand.replace('a', '4'),
            brand.replace('s', '5'),
            brand + '-',
            brand + '_',
            brand[:-1],  # 마지막 문자 제거
            ''.join(c + c for c in brand),  # 문자 중복
        ]

        for pattern in patterns:
            if pattern in domain:
                similar = True
                break

        if similar:
            return True  # 유사 브랜드가 발견되면 True 반환
```

브랜드명을 교묘하게 변형한 URL 탐지를 위해 숫자 치환 및 철자 유사 패턴을 활용한 전처리 적용

ex) go0gle.com, faceb00k.net

데이터 전처리 및 모델링

03

모델 소개

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(input_dim,)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
], name="Model")
```

데이터가 충분히 많아 복잡한 구조 없이도 효과적인 학습이 가능하다고 판단하여 모델은 비교적 간단하게 작성하였으며,
총 20 에포크 동안 학습한 결과, 훈련 정확도는 **92.48%**, 검증 정확도는 92.44%로 안정적인 성능을 보였습니다.

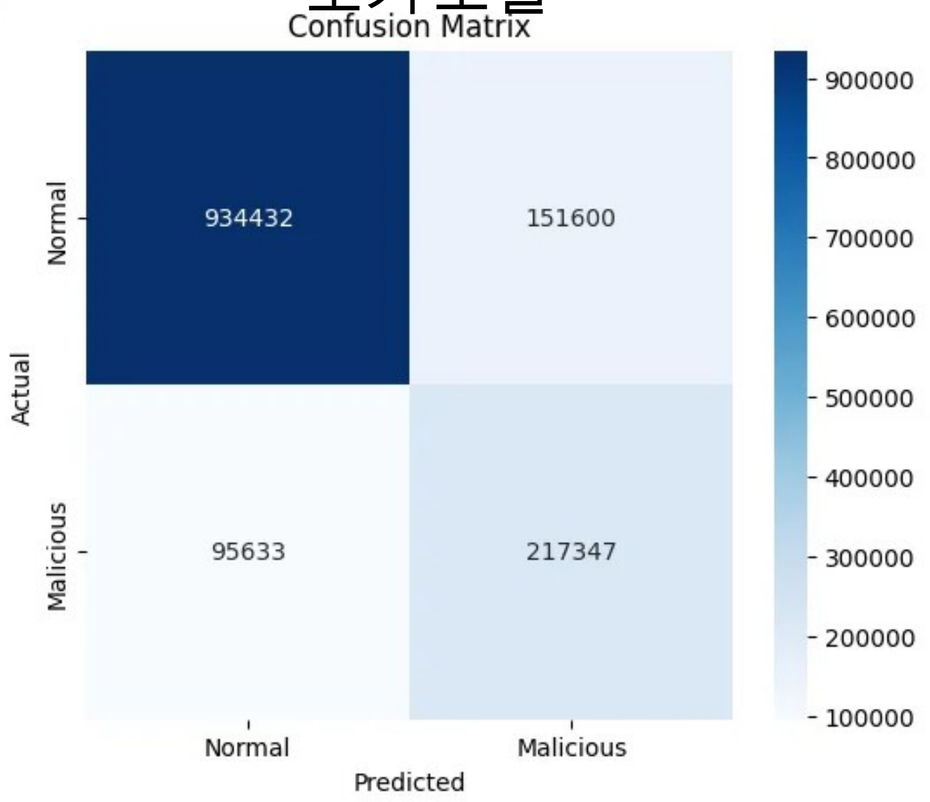
```
Epoch 7/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 68s 2ms/step - accuracy: 0.9221 - loss: 0.2198 - val_accuracy: 0.9231 - val_loss: 0.2184
Epoch 8/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 66s 2ms/step - accuracy: 0.9226 - loss: 0.2188 - val_accuracy: 0.9225 - val_loss: 0.2194
Epoch 9/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 62s 2ms/step - accuracy: 0.9229 - loss: 0.2178 - val_accuracy: 0.9232 - val_loss: 0.2177
Epoch 10/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 63s 2ms/step - accuracy: 0.9230 - loss: 0.2176 - val_accuracy: 0.9235 - val_loss: 0.2170
Epoch 11/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 64s 2ms/step - accuracy: 0.9235 - loss: 0.2170 - val_accuracy: 0.9240 - val_loss: 0.2163
Epoch 12/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 63s 2ms/step - accuracy: 0.9237 - loss: 0.2165 - val_accuracy: 0.9228 - val_loss: 0.2189
Epoch 13/20
...
Epoch 19/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 62s 2ms/step - accuracy: 0.9251 - loss: 0.2135 - val_accuracy: 0.9251 - val_loss: 0.2139
Epoch 20/20
34976/34976 ━━━━━━━━━━━━━━━━━━━━ 61s 2ms/step - accuracy: 0.9248 - loss: 0.2143 - val_accuracy: 0.9244 - val_loss: 0.2148
```

데이터 전처리 및 모델링

04

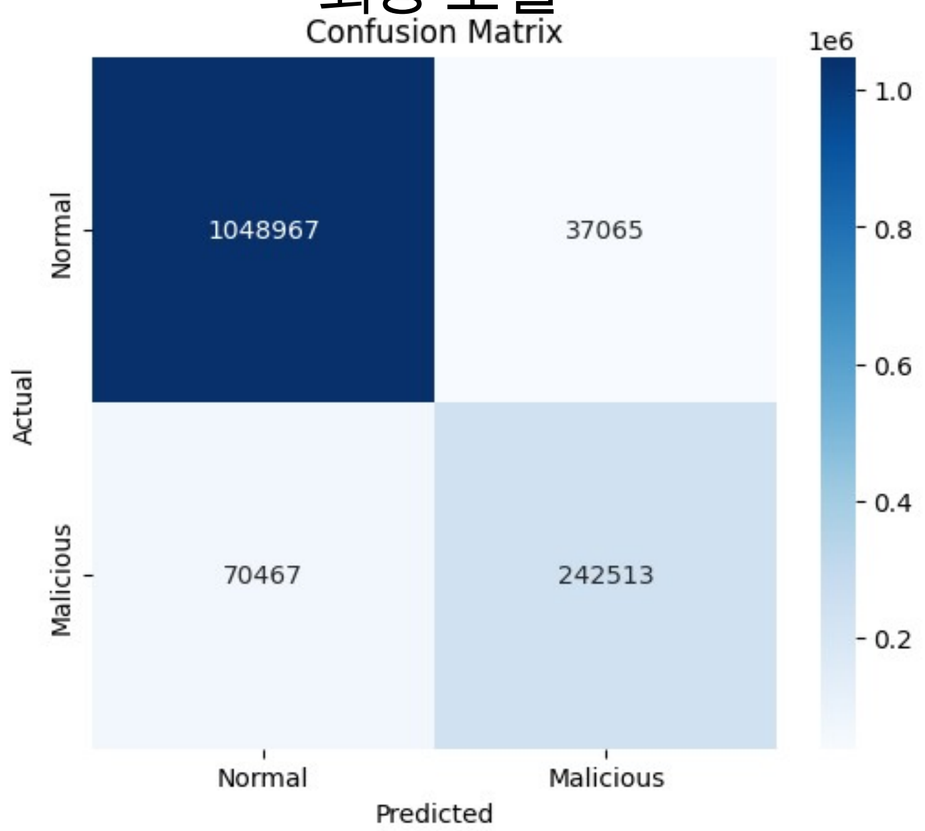
성능평가

초기 모델



Recall = 0.6944

최종 모델



Recall = 0.7748

URL 악성 코드 분류

UI 구현

03

URL 판별기

URL을 입력하세요

검사

결과가 여기에 표시됩니다.

URL 판별기

www.naver.com

검사

Jun 모델

악성 확률: 16.41%

판별 결과: ✅ 정상 URL

Ho 모델

악성 확률: 20.64%

판별 결과: ✅ 정상 URL

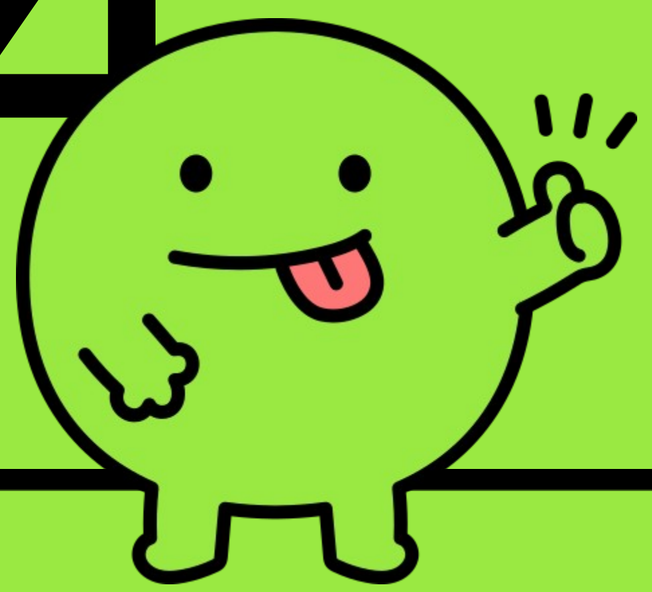
Nam 모델

악성 확률: 48.50%

판별 결과: ✅ 정상 URL

WEB 시현(front backend)

04



THANK YOU FOR WATCHING!

감사합니다