인공지능과 정보보호 E-T (Email_DeTecTion)

정보보호학과 김혜민
정보보호학과 정민희

Malicious-URL-Detection-Website

PlugIn

HyeMinhee/Malicious-URL-Detection-Website-Plugin

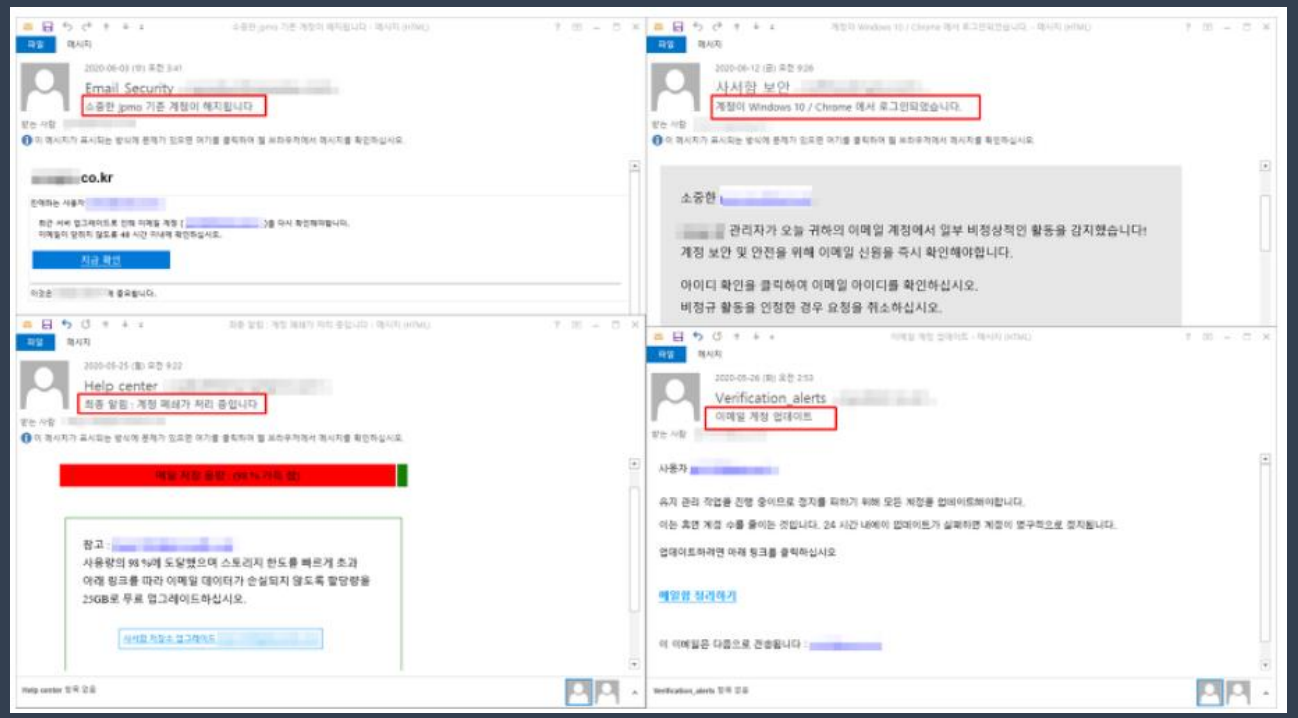
서비스 및 주제 소개 | 데이터 수집 | 데이터 전처리 | 모델 생성 및 학습 | 성능평가 | 서비스 개발

서비스 및 주제 소개

‘악성 이메일에 포함된 악성 URL 탐지(이메일 피싱 방지) 크롬 플러그인 서비스’

– 서비스 주요 기능

: 사용자가 이메일에 포함되어 있는 악성 링크를 클릭하기 전
크롬 플러그인을 통해 조심해야할 URL정보와 신뢰할 수 있는 URL정보를 제공받을 수 있음.

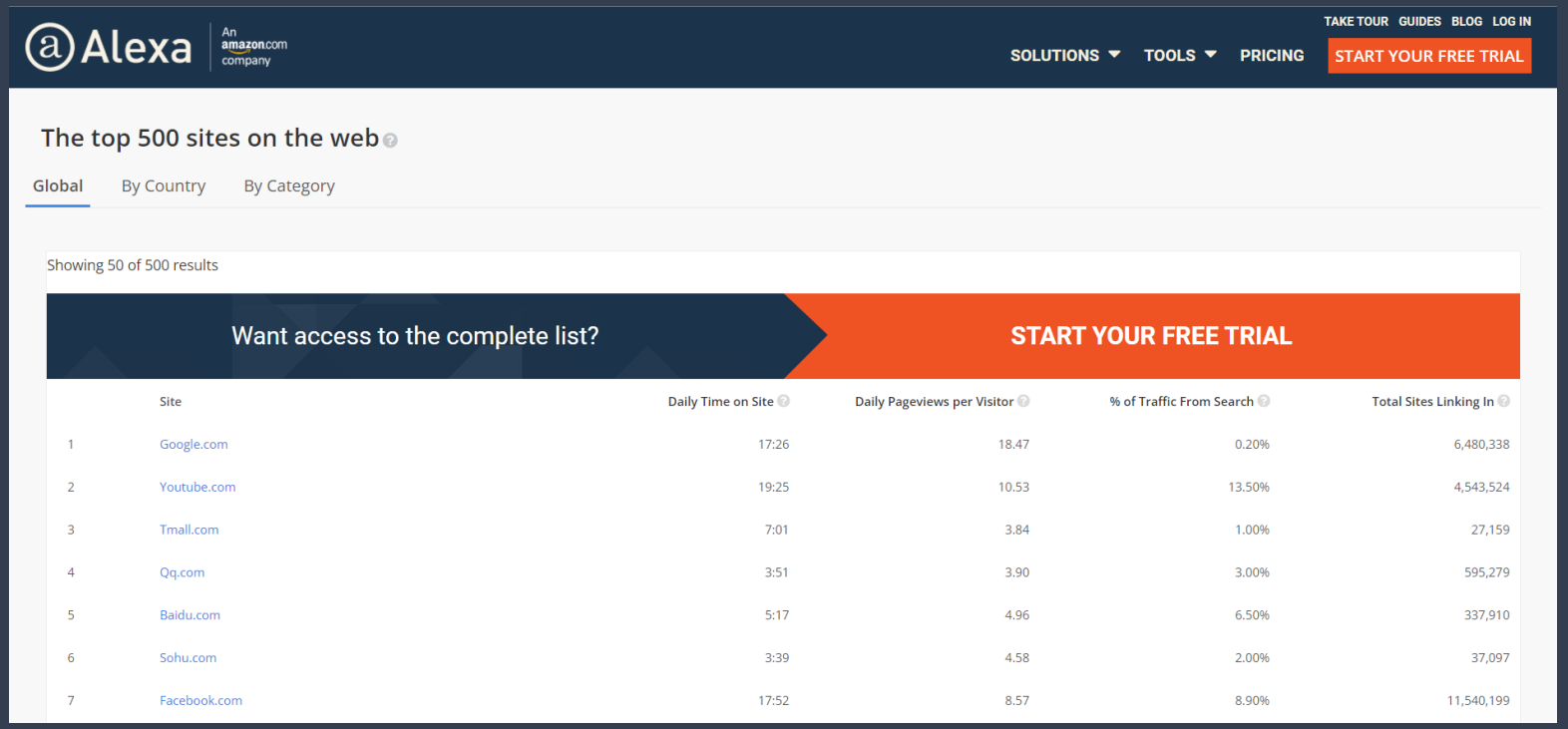


데이터 수집

1. Alexa

Alexa – Benign

정상 URL 50000개 수집



	Site	Daily Time on Site	Daily Pageviews per Visitor	% of Traffic From Search	Total Sites Linking In
1	Google.com	17:26	18.47	0.20%	6,480,338
2	Youtube.com	19:25	10.53	13.50%	4,543,524
3	Tmall.com	7:01	3.84	1.00%	27,159
4	Qq.com	3:51	3.90	3.00%	595,279
5	Baidu.com	5:17	4.96	6.50%	337,910
6	Sohu.com	3:39	4.58	2.00%	37,097
7	Facebook.com	17:52	8.57	8.90%	11,540,199

```
data = pd.read_csv("/content/Alexa.csv")
alexa = data.drop('Unnamed: 0',axis=1)
print(alexa.shape)
alexa.head()
```

(50000, 3)

	url	label	result
0	google.com	benign	0
1	youtube.com	benign	0
2	facebook.com	benign	0
3	baidu.com	benign	0
4	wikipedia.org	benign	0

데이터 수집

2. OpenPhish

OpenPhish – Malicious

악성 URL 500개 데이터 수집

OpenPhish / Phishing Feeds / Phishing Database / Resources / Blog

Timely. Accurate. Relevant Phishing Intelligence.

7-Day Phishing Trends

22,254,570	76,241	268
URLs Processed	Phishing Campaigns	Brands Targeted

Phishing URL	Targeted Brand	Time
https://aichdtv.com/	Citigroup Inc.	02:30:59
https://calgary-flush-answering-patio.trycloudflare.com/	Instagram	02:30:51
http://authenccex.com/shopify/	Shopify	02:30:24
http://termination-trust.info/	Crypto/Wallet	02:30:24
https://clouddoc-authorize.firebaseapp.com/common/oauth2/authorize-client_id...	Outlook	02:26:23
https://co.jp.log6u0o.cn/	Amazon.com Inc.	02:22:15
https://xspin.cawnibos.com/	Tencent	02:21:25
http://a0605443.xsph.ru/appmanager/	French Health Insurance	02:20:21
http://skbcpraha.cz/os.html?email=abc@abc.com	Generic/Spear Phishing	02:16:42
https://634710.selcdn.ru/nwnay127hkireadmekaili65happer000087ha/index.html	Office365	02:11:23

```
[ ] openphish = pd.read_csv("/content/openphish.csv", encoding='cp949')
    print(openphish.shape)
    openphish.head()
```

(500, 3)

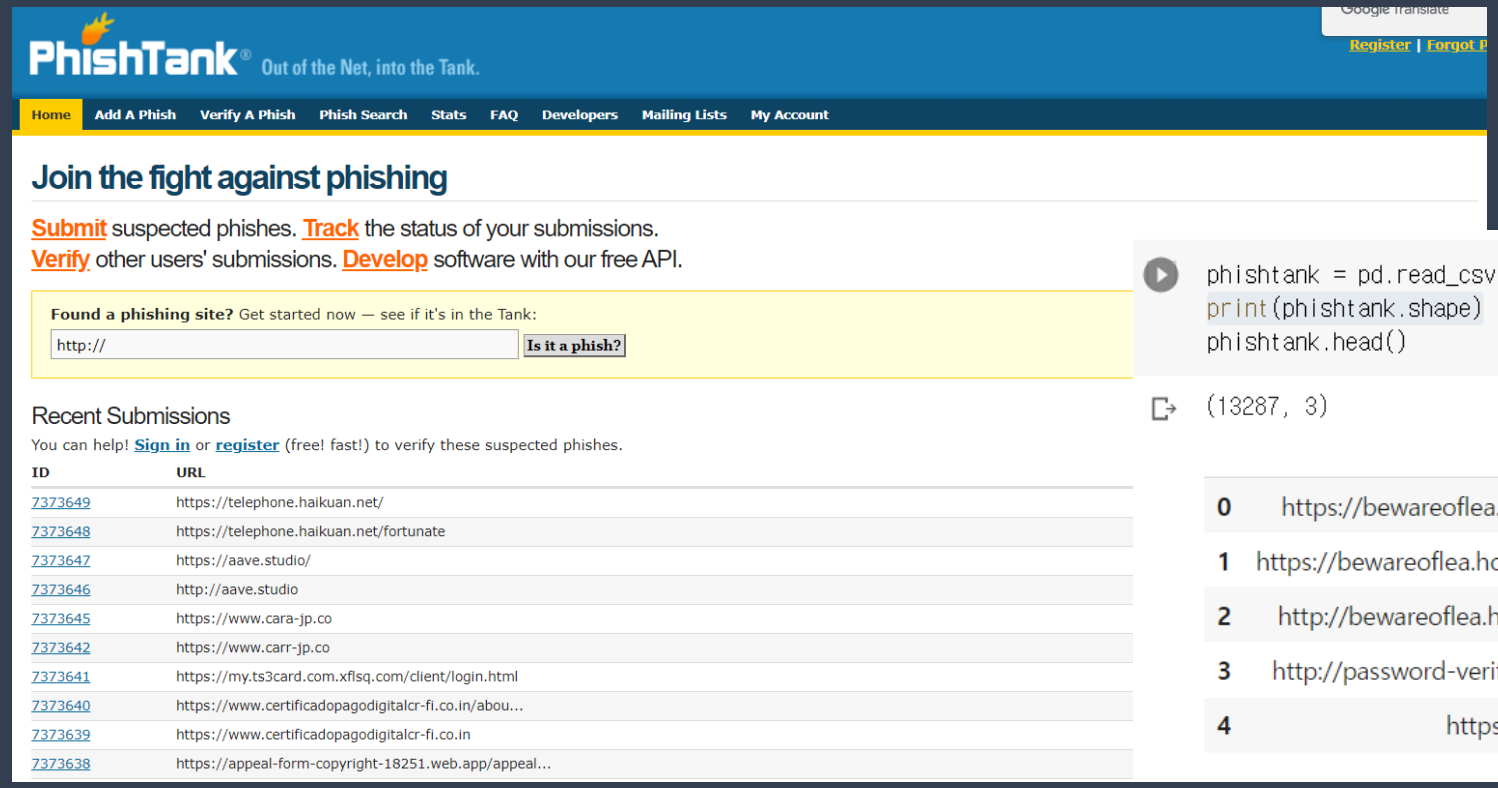
	url	label	result
0	http://pmbonline.unmuha.ac.id/dnd/authorize_cl...	malicious	1
1	http://u1315347ln4.ha004.t.justns.ru/CA/CA/eba...	malicious	1
2	http://u1315347ln4.ha004.t.justns.ru/CA/CA/bcb...	malicious	1
3	http://u1315347ln4.ha004.t.justns.ru/CA/CA/e85...	malicious	1
4	https://www.co.jp.zglwhw.cn/	malicious	1

데이터 수집

3. PhishTank

PhishTank – Malicious

악성 URL 13287개 데이터 수집




```python
phishtank = pd.read_csv("/content/phishtank_verified_online.csv", encoding='cp949')
print(phishtank.shape)
phishtank.head()
```

```
(13287, 3)
```

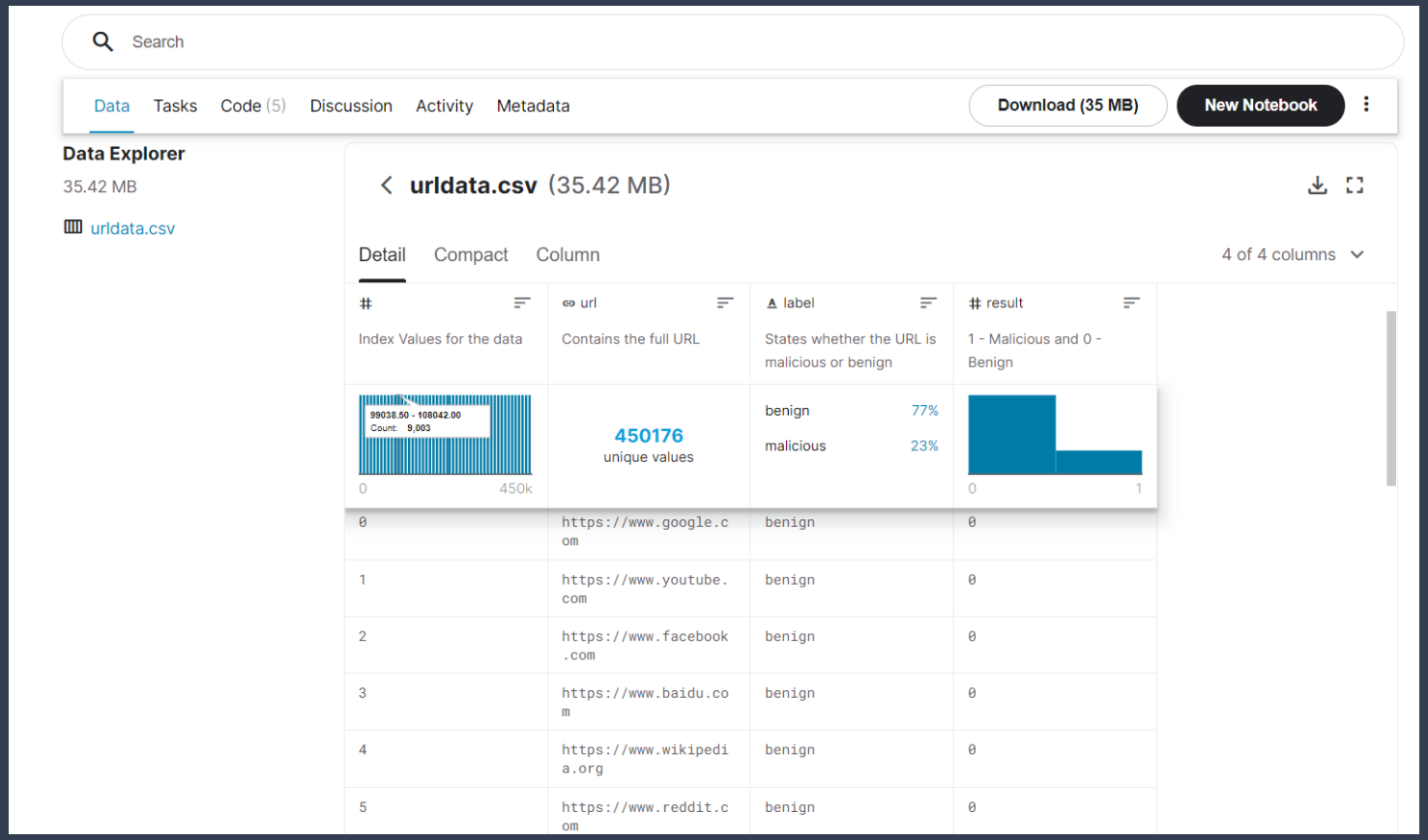
	url	label	result
0	https://bewareoflea.hostingamazonemail.xyz/sig...	malicious	1
1	https://bewareoflea.hostingamazonemail.xyz/signim	malicious	1
2	http://bewareoflea.hostingamazonemail.xyz/sign...	malicious	1
3	http://password-verify.qswedepassrest.cloudns.ph/	malicious	1
4	https://sms.vinted-pl.947519.space	malicious	1

데이터 수집

4. Keras – Beign and Malicious

Keras – Beign and Malicious

악성 및 정상 URL 450176개 데이터 수집




```
url_keras = pd.read_csv("/content/urldata.csv")
url_keras = url_keras.drop('Unnamed: 0',axis=1)
print(url_keras.shape)
url_keras.head()
```

```
(450176, 3)
```

	url	label	result
0	https://www.google.com	benign	0
1	https://www.youtube.com	benign	0
2	https://www.facebook.com	benign	0
3	https://www.baidu.com	benign	0
4	https://www.wikipedia.org	benign	0

03

데이터 전처리

데이터 셋 합치기

Keras – Beign and Malicious
악성 및 정상 URL 450176개 데이터 수집

Alexa – Benign
정상 URL 50000개 수집

PhishTank – Malicious
악성 URL 13287개 데이터 수집

OpenPhish – Malicious
악성 URL 500개 데이터 수집

513963개의 데이터로 이루어진 데이터셋

```
[17] total = pd.DataFrame()
     total = pd.concat([total, url_keras])
     total = pd.concat([total, malicious])
     total = pd.concat([total, alexa])
     print(total.shape)
     total.head()
```

```
(513963, 3)
```

	url	label	result
0	https://www.google.com	benign	0
1	https://www.youtube.com	benign	0
2	https://www.facebook.com	benign	0
3	https://www.baidu.com	benign	0
4	https://www.wikipedia.org	benign	0

```
total.info()
total.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 513963 entries, 0 to 49999
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   url     513963 non-null  object
 1   label   513963 non-null  object
 2   result  513963 non-null  int64
dtypes: int64(1), object(2)
memory usage: 15.7+ MB
url       0
label     0
result    0
dtype: int64
```

데이터 전처리

URL 특징 추출

특징 추출 코드

```
urldata['url_length'] = urldata['url'].apply(lambda i: len(str(i)))
urldata['hostname_length'] = urldata['url'].apply(lambda i: len(urlparse(i).netloc))
urldata['path_length'] = urldata['url'].apply(lambda i: len(urlparse(i).path))
urldata['fd_length'] = urldata['url'].apply(lambda i: fd_length(i))
urldata['count-'] = urldata['url'].apply(lambda i: i.count('-'))
urldata['count@'] = urldata['url'].apply(lambda i: i.count('@'))
urldata['count?'] = urldata['url'].apply(lambda i: i.count('?'))
urldata['count%'] = urldata['url'].apply(lambda i: i.count('%'))
urldata['count.'] = urldata['url'].apply(lambda i: i.count('.'))
urldata['count='] = urldata['url'].apply(lambda i: i.count('='))
urldata['count-http'] = urldata['url'].apply(lambda i : i.count('http'))
urldata['count-https'] = urldata['url'].apply(lambda i : i.count('https'))
urldata['count-www'] = urldata['url'].apply(lambda i: i.count('www'))
urldata['count-digits']= urldata['url'].apply(lambda i: digit_count(i))
urldata['count-letters']= urldata['url'].apply(lambda i: letter_count(i))
urldata['count_dir'] = urldata['url'].apply(lambda i: no_of_dir(i))
urldata['use_of_ip'] = urldata['url'].apply(lambda i: having_ip_address(i))
```

URL 특징 추출

URL 길이, Hostname길이, Path 길이,
특수 문자의 개수, IP 주소 여부, 단축 URL 여부

	url	label	result	url_length	hostname_length	path_length	fd_length	count-	count@	count?	count%	count.	count=	count-http	count-https	count-www	count-digits	count-letters	count_dir	use_of_ip
0	https://www.google.com	benign	0	22	14	0	0	0	0	0	0	2	0	1	1	1	0	17	0	1
1	https://www.youtube.com	benign	0	23	15	0	0	0	0	0	0	2	0	1	1	1	0	18	0	1
2	https://www.facebook.com	benign	0	24	16	0	0	0	0	0	0	2	0	1	1	1	0	19	0	1
3	https://www.baidu.com	benign	0	21	13	0	0	0	0	0	0	2	0	1	1	1	0	16	0	1
4	https://www.wikipedia.org	benign	0	25	17	0	0	0	0	0	0	2	0	1	1	1	0	20	0	1

데이터 전처리

URL 특징 추출

특징 추출 후

```
[ ] #from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split

    from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
    #Independent Variables
    x = urldata[['hostname_length',
           'path_length', 'fd_length', 'count-', 'count@', 'count?',
           'count%', 'count.', 'count=', 'count-http','count-https', 'count-www', 'count-digits',
           'count-letters', 'count_dir', 'use_of_ip']]

    #Dependent Variable
    y = urldata['result']
    print(x.shape)
    print(y.shape)
    print("Percent Of Malicious URLs:{:.2f} %".format(len(urldata[urldata['label']=='malicious'])/len(urldata['label'])*100))
    print("Percent Of Benign URLs:{:.2f} %".format(len(urldata[urldata['label']=='benign'])/len(urldata['label'])*100))

    (513963, 16)
    (513963,)
    Percent Of Malicious URLs:23.00 %
    Percent Of Benign URLs:77.00 %
```

URL 특징 추출

result를 제외한 16개의 칼럼 데이터들은 x변수로,
라벨인 result 칼럼 데이터를 y 변수로 할당

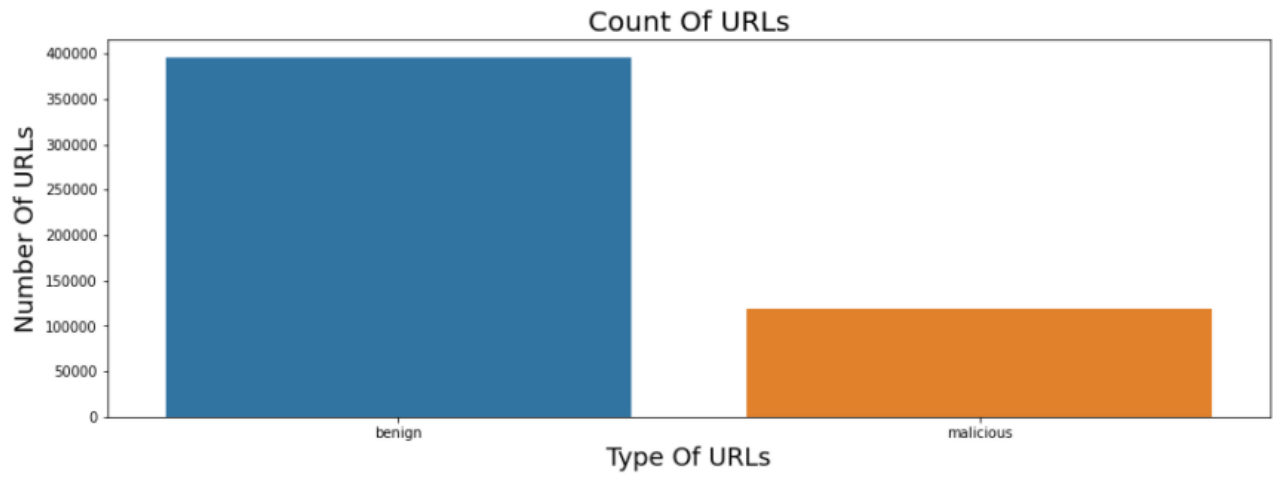
데이터 전처리

URL 비율

URL 비율

```
[ ] plt.figure(figsize=(15,5))
    sns.countplot(x='label',data=urldata)
    plt.title("Count Of URLs",fontsize=20)
    plt.xlabel("Type Of URLs",fontsize=18)
    plt.ylabel("Number Of URLs",fontsize=18)
```

```
Text(0, 0.5, 'Number Of URLs')
```




```
[ ] print("Percent Of Malicious URLs:{:.2f} %".format(len(urldata[urldata['label']=='malicious'])/len(urldata['label'])*100))
    print("Percent Of Benign URLs:{:.2f} %".format(len(urldata[urldata['label']=='benign'])/len(urldata['label'])*100))
```

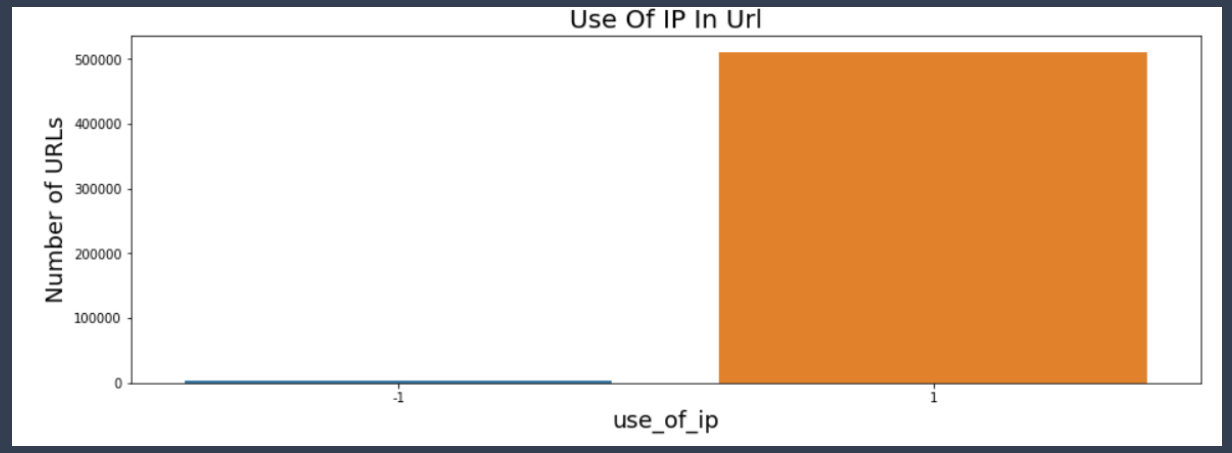
```
Percent Of Malicious URLs:23.00 %
Percent Of Benign URLs:77.00 %
```

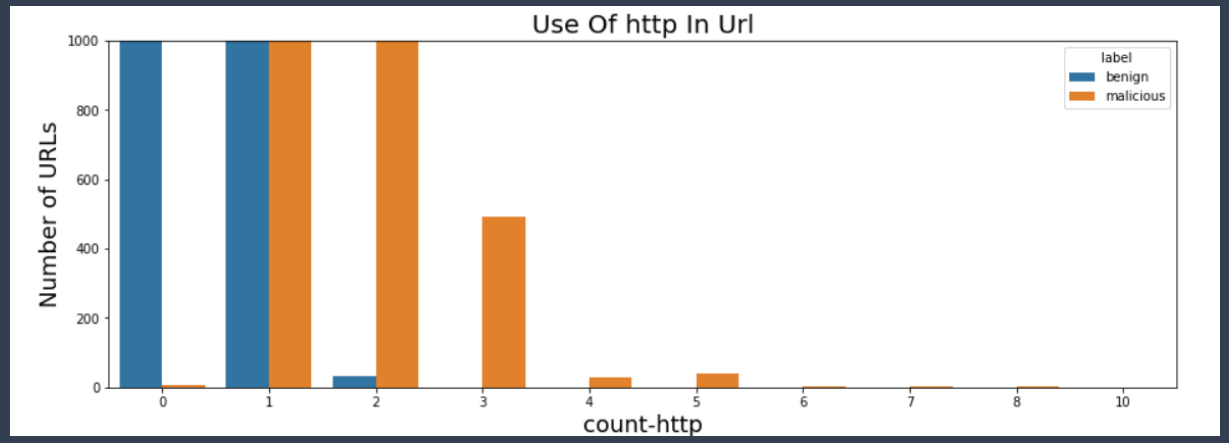
URL 비율

악성 URL : 정상 URL = 23 : 77

데이터 전처리

악성 URL 특징





악성 URL 특징

- URL에 IP 주소 포함할 확률 높음
- http 문자열의 개수가 정상 URL 보다 많음

데이터 전처리

클래스 균형 맞추기

SMOTE

```
[ ] #Oversampling using SMOTE
    from imblearn.over_sampling import SMOTE

    x_sample, y_sample = SMOTE().fit_resample(x, y.values.ravel())

    x_sample = pd.DataFrame(x_sample)
    y_sample = pd.DataFrame(y_sample)

    # checking the sizes of the sample data
    print("Size of x-sample :", x_sample.shape)
    print("Size of y-sample :", y_sample.shape)

    Size of x-sample : (791476, 16)
    Size of y-sample : (791476, 1)
```

클래스 균형 맞추기 : SMOTE()

5 : 5 비율로 만들어 데이터 총 791476개

데이터 전처리

데이터 준비

학습 전 Train,Test, Validation

```
[ ] #Train test split
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
    print("Shape of x_train: ", x_train.shape)
    print("Shape of x_test: ", x_test.shape)
    print("Shape of y_train: ", y_train.shape)
    print("Shape of y_test: ", y_test.shape)

    x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size = 0.2)

    print("Shape of x_train: ", x_train.shape)
    print("Shape of x_test: ", x_test.shape)
    print("Shape of x_valid: ", x_valid.shape)
    print("Shape of y_train: ", y_train.shape)
    print("Shape of y_test: ", y_test.shape)
    print("Shape of y_valid: ", y_valid.shape)
```

```
Shape of x_train:  (359774, 16)
Shape of x_test:  (154189, 16)
Shape of y_train:  (359774,)
Shape of y_test:  (154189,)
Shape of x_train:  (287819, 16)
Shape of x_test:  (154189, 16)
Shape of x_valid:  (71955, 16)
Shape of y_train:  (287819,)
Shape of y_test:  (154189,)
Shape of y_valid:  (71955,)
```

데이터 준비

학습셋 : 시험셋 = 7 : 3
학습셋 : 검증셋 = 8 : 2

모델 생성 및 학습

모델 구축

모델 구축

```
model = Sequential()
model.add(Dense(32, activation = 'relu', input_shape = (16, )))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                544

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 8)                 136

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 1,217
Trainable params: 1,217
Non-trainable params: 0
_________________________________________________________________
```

1 **모델 구축**

Keras의 Sequential ( )
총 4개의 Layer

04

모델 생성 및 학습

학습 방식 설정 및 모델 저장

학습 방식 설정 및 모델 저장

```
from tensorflow import keras
opt = keras.optimizers.Adam(lr=0.0001)
model.compile(optimizer= opt ,loss='binary_crossentropy',metrics=['acc'])

checkpointer = ModelCheckpoint('url.h5', monitor='val_acc', mode='max', verbose=2, save_best_only=True)
```

2

학습 방식 설정

Compile( )
- Optimizer : Adam
- 평가 지표 : Accuracy
- 손실 함수 : binary_crossentropy

3

모델 저장

ModelCheckPoint()
: 모델 저장 경로와 모델 저장 기준 값 설정

모델 생성 및 학습

모델 학습

모델 학습

```
history=model.fit(x_train, y_train, batch_size=256, epochs=5, validation_data=(x_valid, y_valid), callbacks=[checkpointer])
```

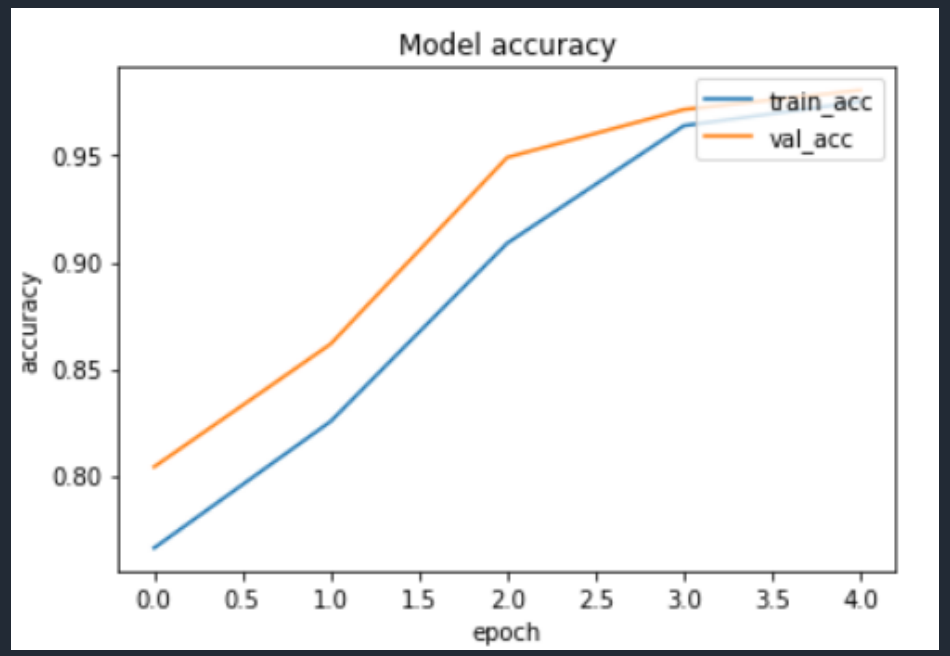
```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
Epoch 1/5
1124/1125 [============================>.] - ETA: 0s - loss: 0.5483 - acc: 0.7660
Epoch 00001: val_acc improved from -inf to 0.80407, saving model to url.h5
1125/1125 [==============================] - 4s 3ms/step - loss: 0.5482 - acc: 0.7660 - val_loss: 0.4618 - val_acc: 0.8041
Epoch 2/5
1107/1125 [============================>.] - ETA: 0s - loss: 0.4052 - acc: 0.8249
Epoch 00002: val_acc improved from 0.80407 to 0.86166, saving model to url.h5
1125/1125 [==============================] - 3s 3ms/step - loss: 0.4044 - acc: 0.8254 - val_loss: 0.3412 - val_acc: 0.8617
Epoch 3/5
1123/1125 [============================>.] - ETA: 0s - loss: 0.2593 - acc: 0.9089
Epoch 00003: val_acc improved from 0.86166 to 0.94915, saving model to url.h5
1125/1125 [==============================] - 3s 2ms/step - loss: 0.2591 - acc: 0.9089 - val_loss: 0.1802 - val_acc: 0.9491
Epoch 4/5
1104/1125 [============================>.] - ETA: 0s - loss: 0.1332 - acc: 0.9638
Epoch 00004: val_acc improved from 0.94915 to 0.97157, saving model to url.h5
1125/1125 [==============================] - 3s 2ms/step - loss: 0.1326 - acc: 0.9640 - val_loss: 0.0976 - val_acc: 0.9716
Epoch 5/5
1116/1125 [============================>.] - ETA: 0s - loss: 0.0828 - acc: 0.9753
Epoch 00005: val_acc improved from 0.97157 to 0.98065, saving model to url.h5
1125/1125 [==============================] - 3s 2ms/step - loss: 0.0827 - acc: 0.9753 - val_loss: 0.0691 - val_acc: 0.9807
```

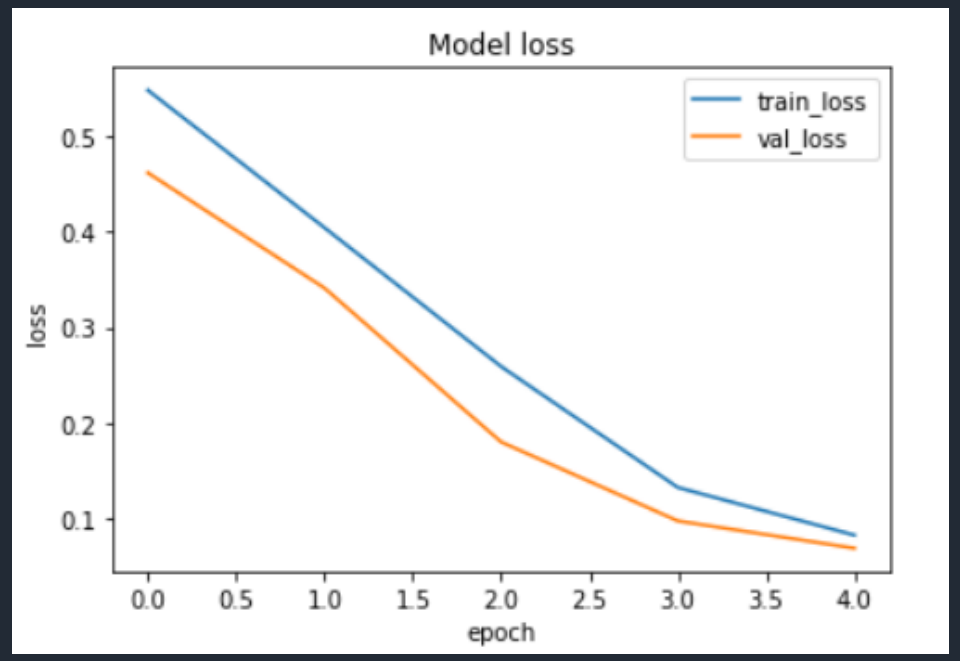
4 **모델 학습** fit ( )

- 입력/결과 데이터 : x_train, y_train
- 검증 셋 : x_valid, y_valid
- epoch : 5

성능 평가

성능 평가





에폭에 따른 정확도와 손실값

성능 평가

성능 평가 & 모델 추출

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

def get_clf_eval(y_test, pred):
    confusion = confusion_matrix(y_test, pred)
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred)
    recall = recall_score(y_test, pred)
    print('Confusion Matrix')
    print(confusion)
    print('정확도:{}, 정밀도:{}, 재현율:{}'.format(accuracy, precision, recall))

get_clf_eval(y_sample, pred_test)
```

```
Confusion Matrix
[[394677   1061]
 [ 23201 372537]]
정확도:0.9693458803551845, 정밀도:0.9971600490366651, 재현율:0.941372827476765
```

```
[ ]  model.save('model.h5')
```

정확도, 정밀도, 재현율

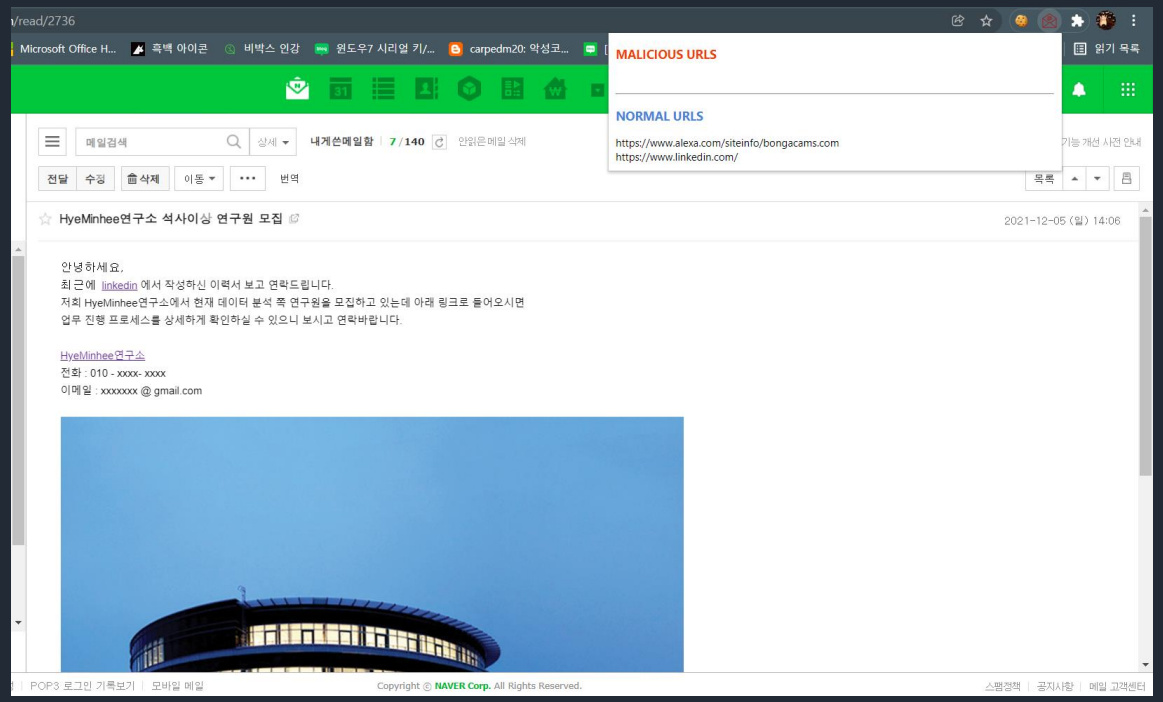
정확도 : 0.96 정밀도 : 0.99 재현율 : 0.94

06

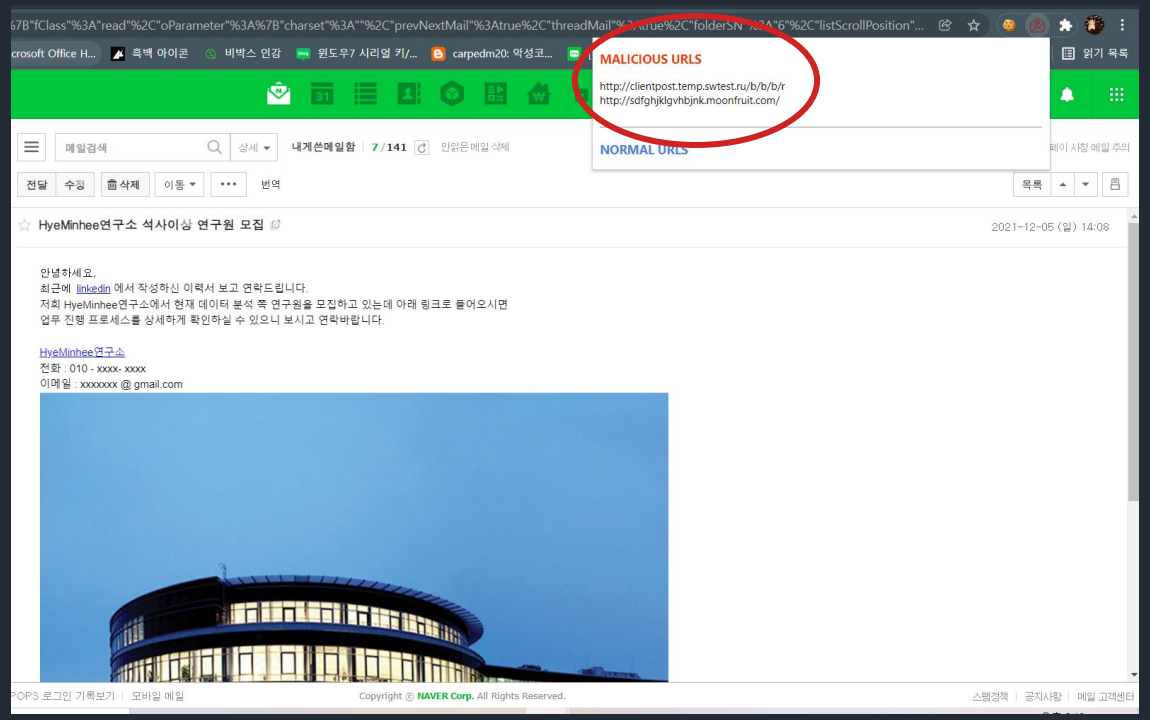
서비스 개발

플러그인

정상 네이버 메일



비정상 네이버 메일

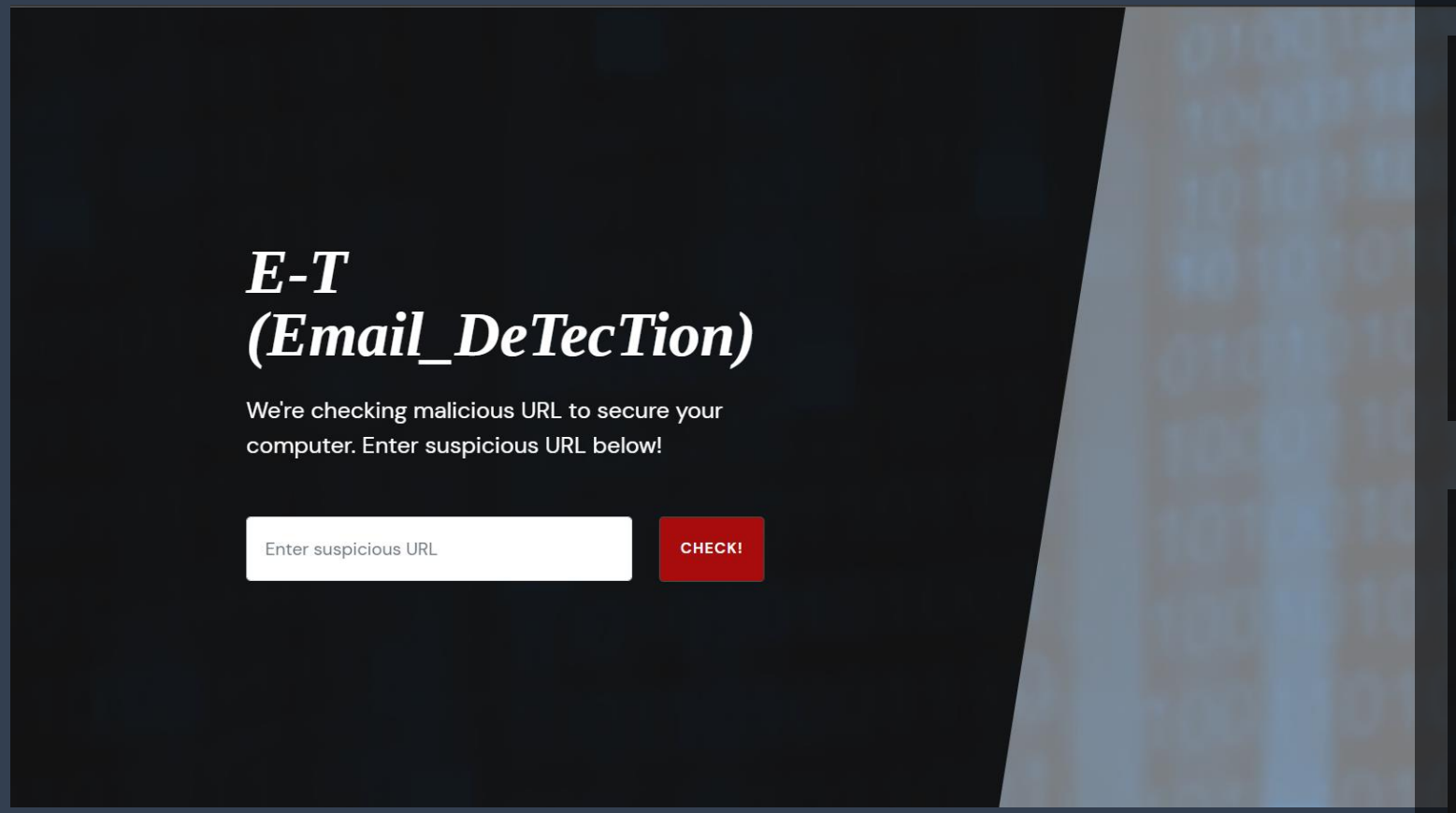


크롬 플러그인 서비스

06

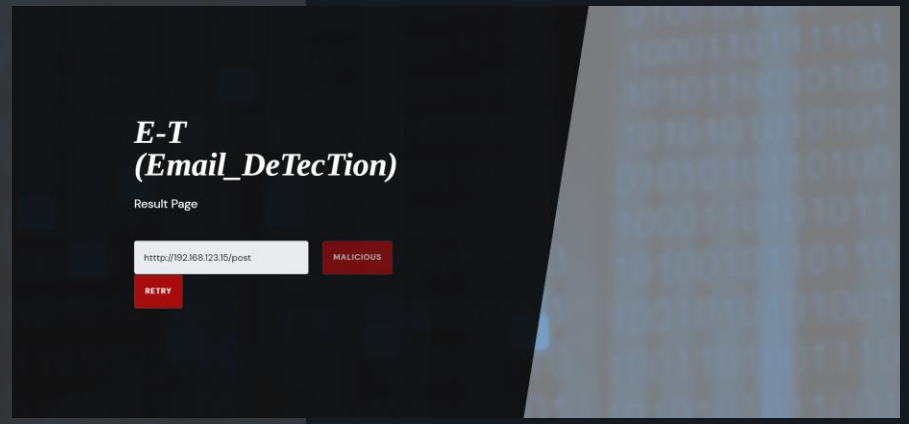
서비스 개발

웹사이트

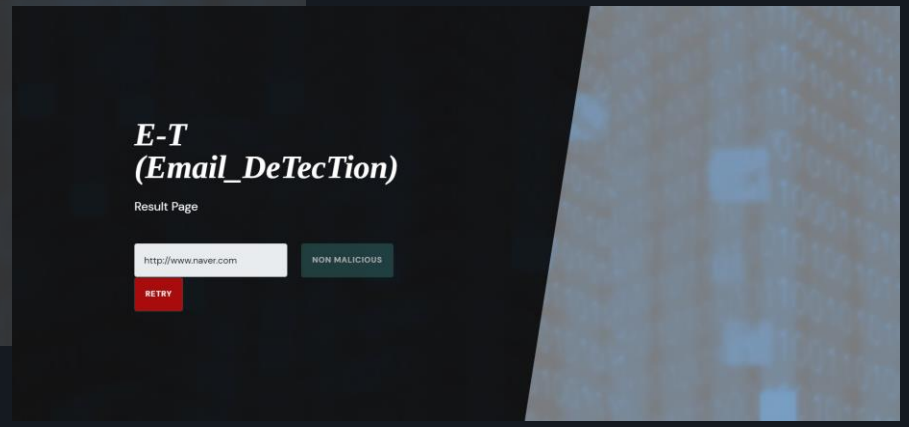


웹 사이트

비정상 URL 입력 시



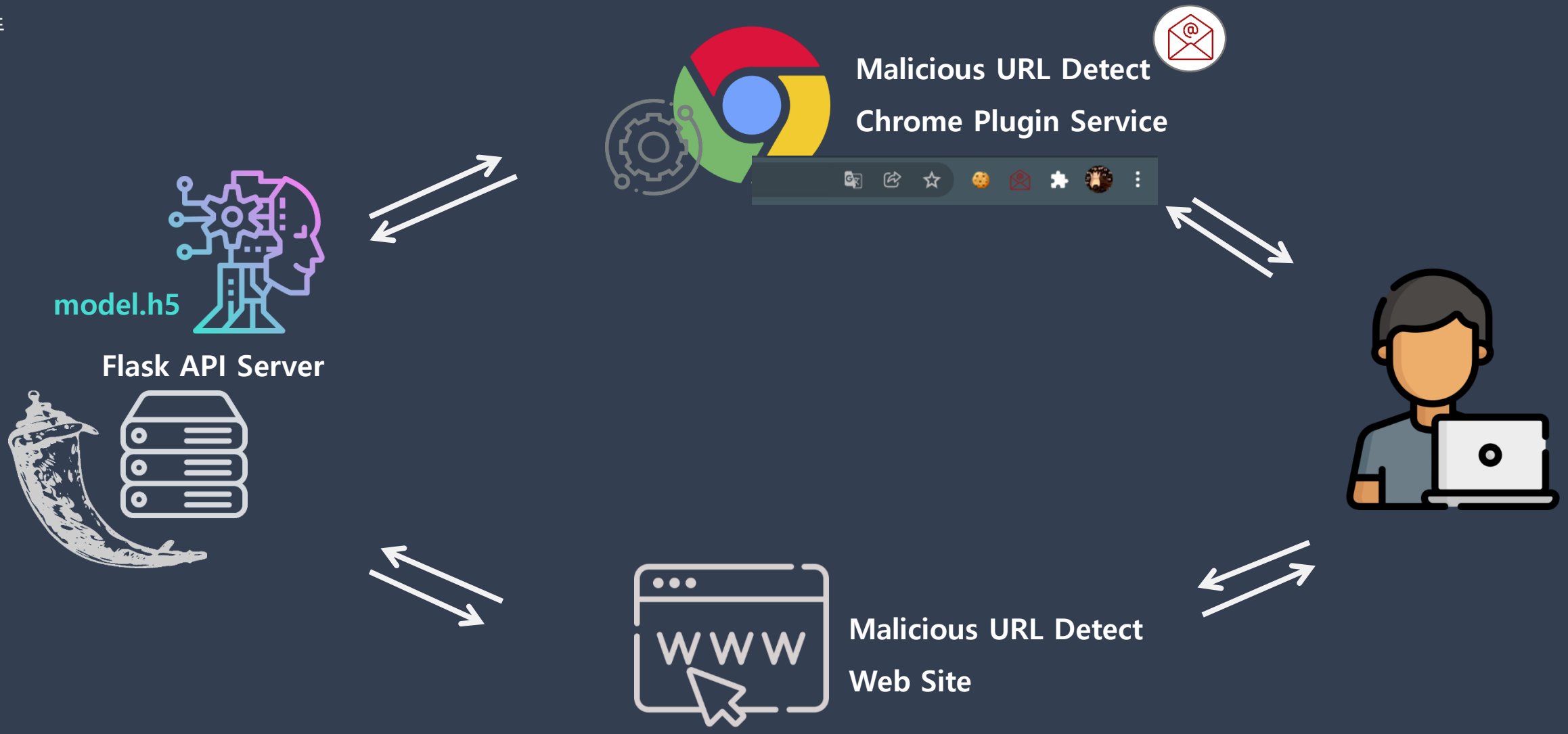
정상 URL 입력 시



06

서비스 개발

구조도


Malicious URL Detect
Chrome Plugin Service
model.h5
Flask API Server
Malicious URL Detect
Web Site

감사합니다

E-T (Email_DeTecTion)