서론

1. 개요

학습셋의 변수는 총 7개로 모든 변수가 범주형 변수입니다. 이러한 점에서 착안점을 얻어 범주형 변수를 이용하여 분류가 가능한 다양한 머신러닝 모델들을 테스트합니다. 테스트 결과 정확도가 가장 높은 모델인 Cat Boost 모델을 분류 모델로 선정합니다.

모델의 분류 성능을 더욱 높이기 위해 각 변수의 의미를 분석하여 분류 성능에 좋은 영향을 줄 수 있는 파생 변수를 생성합니다. 좋은 feature를 선별하기 위한 방법으로 Cramer's V correlation을 사용하였으며 양의 상관관계를 나타내는 변수를 모델에 사용합니다.

CatBoost 모델 학습 시 K-Fold를 사용하여 5겹 교차검증을 수행하며 모델링을 진행합니다. 그 결과 약 0.9873의 macro-precision을 달성하였습니다.

2. 개발 도구

구분	도구 이름	버전	제조사(출처)	용도
1	Colab	22/09/30	https://colab.research.google.com/	AI 모델 개발

AI 학습 구성도

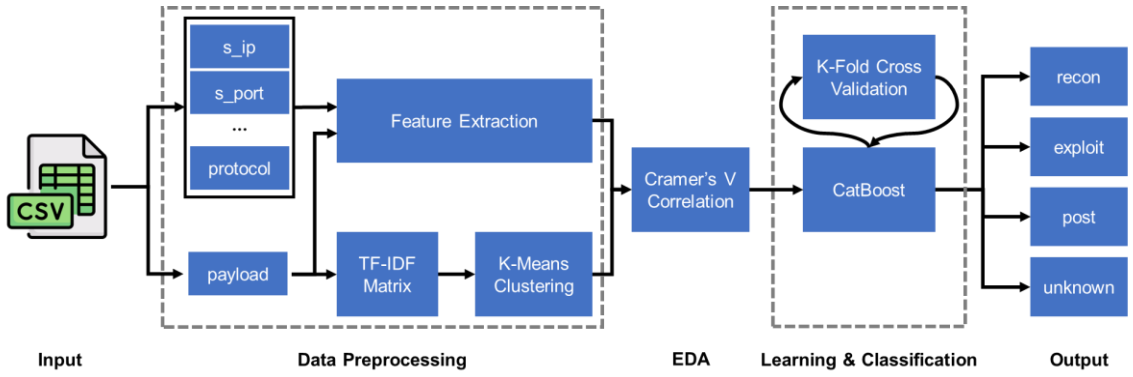


그림 1. 네트워크 공격 분류를 위한 AI 모델 구성도

방법론

1. 데이터 분석

데이터셋은 출발지 IP(s_ip), 출발지 Port(s_port), 목적지 IP(d_ip), 목적지 Port(d_port), 프로토콜(protocol), 페이로드(payload), 리스크(risk)로 구성되어 있습니다. 이 변수들은 모두 범주형 변수입니다. 따라서 공격을 분류하기 위해 범주형 데이터를 잘 처리할 수 있는 머신러닝 모델을 생각하게 되었습니다. 전처리 과정 없이 페이로드를 제외하고 5가지 모델(Random Forest, Lightgbm [1], CatBoost [2], XGBoost[3], Extra trees)을 돌렸을 때 CatBoost 모델이 precision 기준 가장 높은 0.861을 달성하였습니다. 아래는 분류 성능을 높이기 위한 각 feature의 특징 분석과 전처리 관련 내용입니다.

1.1 출발지 IP, 목적지 IP 분석

IP는 네트워크 주소의 시작 비트에 따라 아래와 같이 A~E 클래스로 나눌 수 있습니다 [4].

- A 클래스의 IP 범위: 0.0.0.0 ~ 127.255.255.255
- B 클래스의 IP 범위: 128.0.0.0 ~ 191.255.255.255
- C 클래스의 IP 범위: 192.0.0.0 ~ 223.255.255.255
- D 클래스의 IP 범위: 224.0.0.0 ~ 239.255.255.255
- E 클래스의 IP 범위: 240.0.0.0 ~ 255.255.255.255

또한 IP는 한국인터넷정보센터의 해외 IP 대역별 현황 [5], IP의 위치를 확인하는 사이트 [8, 9]를 이용하면 해당 IP의 국가(Country code)를 알 수 있습니다.

1.2 출발지 Port, 목적지 Port 분석

Port 번호의 범위에 따라 잘 알려진 포트(0~1023), 등록된 포트(1024~49151), 동적 포트(41952~65535)가 있습니다 [6].

1.3 페이로드 분석

MITRE ATT&CK Tatics는 공격 목표에 따른 공격자의 행동을 나타내고 있습니다 [7]. 학습셋은 다양한 공격 목표 중에서 3가지 공격 목표(reconnaissance, exploit, post)와 공격을 수행하지 않은 unknown으로 분류되어 있습니다. 페이로드의 내용이 공격 목표를 식별하는데 영향을 줄 수 있다고 생각하여 페이로드를 분석하였습니다. 그 결과, 페이로드에서 7가지 공격(XML-RPC, Shell shock, SQL injection, passwd 파일 접근, XSS, Directory traversal, Web shell upload)의 패턴을 식별하였습니다.

1.4 프로토콜, 리스크 분석

프로토콜은 TCP, TCP(6), UDP, UDP(17)이 있고, 리스크는 1~3으로 분류가 잘 되어 있어서 추가적인 분석이 불필요하다고 생각했습니다.

1.5 전처리

Purser [10]와 같이 출발지 IP, 목적지 IP를 이용하여 IP 클래스(A~E)를 나타내는 파생 변수(s_ip_class, d_ip_class)를 생성하였습니다. 또한 IPv4 주소(a.b.c.d)에서 각 구역을 표시하는 파생 변수(s_network_id, s_ip2, s_ip3, s_ip4, s_host_ip, d_network_id, d_ip2, d_ip3, d_ip4, d_host_ip)를 생성하였습니다. 예를 들어 s_ip가 210.248.110.200 라면, s_network_id, s_ip2, s_ip3, s_ip4는 각각 210, 248, 110, 200입니다.

출발지 Port, 목적지 Port는 값에 따라 Port의 종류(well-known, registered, dynamic)를 나타내는 파생 변수(s_port_class, d_port_class)를 생성하였습니다.

페이로드에서 특정 공격을 식별할 수 있는 패턴이 존재하면 1, 없으면 0으로 정하고, 공격을 표시하는 변수 8개(has_xml_rpc, has_sql_injection, has_shell_shock, has_malicious_execution, has_xss_a

ttack, has_directory_traversal, has_web_shell, has_shell_file)를 생성하였습니다. 추가로 8개 변수의 값을 연속적으로 이어 붙인 attack_pattern 이라는 변수를 생성하였습니다.

페이로드에는 식별한 8개의 패턴 이외에 식별해내지 못한 패턴이 추가로 존재할 수 있습니다. 또한 학습셋에서 주어진 5만개의 페이로드를 전수조사하기에는 시간이 매우 많이 소요됩니다. 이를 보완하기 위해 페이로드 값에서 텍스트 마이닝 하는 방법을 생각하였습니다. 각 페이로드의 중요 단어를 기준으로 43개로 군집화하였고 각 군집 번호를 가지는 cluster_label 변수를 생성하였습니다.

2. feature

모델에 사용할 feature를 선정하기에 앞서 데이터셋에 있는 변수(출발지 IP, 출발지 Port, 목적지 IP, 목적지 Port, 프로토콜, 페이로드, 리스크)와 전처리 과정에서 생성했던 파생 변수에 대해 feature-label 간 상관 계수(Correlation coefficient)를 구한 결과는 그림 2와 같습니다. 데이터셋에 있는 모든 변수와 전처리 과정에서 생성했던 파생 변수는 모두 범주형 변수이므로 범주형 변수 간 상관 계수를 나타내는 Cramer's V correlation을 이용하였습니다.

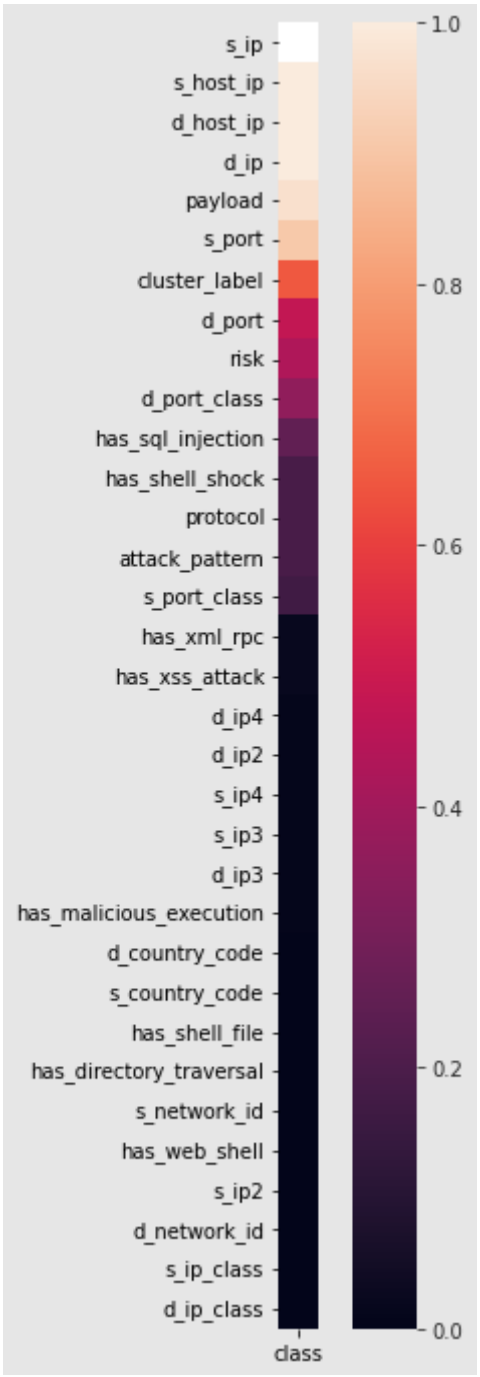


그림 2. 각 feature에 대한 Cramer's V correlation Matrix

그림 2에서 IP 관련 변수(예: s_ip, s_host_ip, d_host_ip 등)가 공격 유형(class)에 대해 강한 양의 상관관계를 가지는 것을 알 수 있었습니다. 위의 히트맵을 참고하여 공격 유형과의 상관관계가 0인 변수(has_xml_rpc ~ d_ip_class)를 제외하고 모델링을 수행하기로 결정하였습니다.

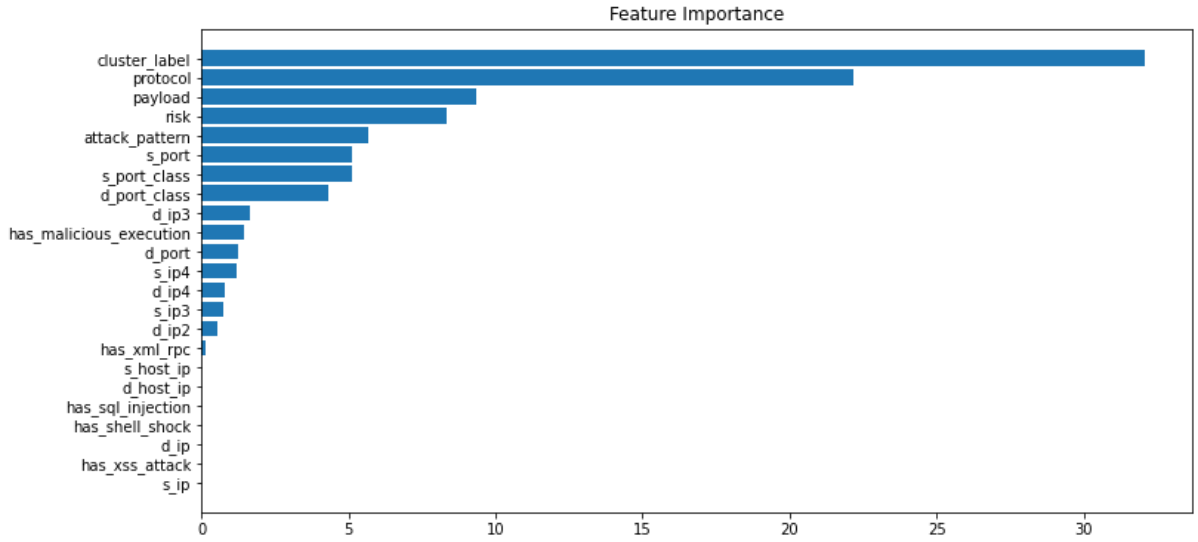


그림 3. CatBoost 모델에서 Feature importance 측정

그림 3은 공격 유형과의 상관관계가 0인 변수를 제외한 변수를 바탕으로 산출된 CatBoost 모델의 Feature importance입니다. 페이로드에서 파생된 cluster_label, protocol 변수는 CatBoost 모델에서 가장 중요한 feature로 확인되었습니다. 그림 2의 Cramer's V correlation Matrix의 내용과 그림 3의 Feature importance의 결과가 다른 것은 Cramer's V correlation으로 설명이 불가한 비선형적인 관계가 존재하기 때문입니다.

표 1. CatBoost 모델의 Classification report

구분	Precision	Recall	F1-Score	Support
1_reconnaissance	0.95	0.96	0.95	627
2_exploit	0.99	0.99	0.99	4841
3_post	1.00	1.00	1.00	2954
4_unknown	1.00	1.00	1.00	4078
accuracy	-	-	0.99	12500
macro avg	0.98	0.99	0.99	12500

weighted avg	0.99	0.99	0.99	12500

표1은 공격 유형과의 상관관계가 0인 변수를 제외한 변수를 바탕으로 모델링한 CatBoost 모델의 성능입니다. 1_reconnaissance에 대한 정밀도가 약간 떨어지지만 좋은 성능을 보이고 있음을 알 수 있습니다.

3. AI 모델링

저희 팀이 구성한 모델은 1) 1장에서 분석한 내용을 바탕으로 새로운 feature를 생성하고 각 feature를 머신러닝 모델에 맞춰 처리하는 Data Preprocessing 단계, 2) 효과적인 분류를 할 수 있도록 feature를 선별하는 EDA 단계, 3) 선별된 feature를 학습하고 분류하는 Learning & Classification 단계로 구성되어 있습니다.

3.1 Data Preprocessing

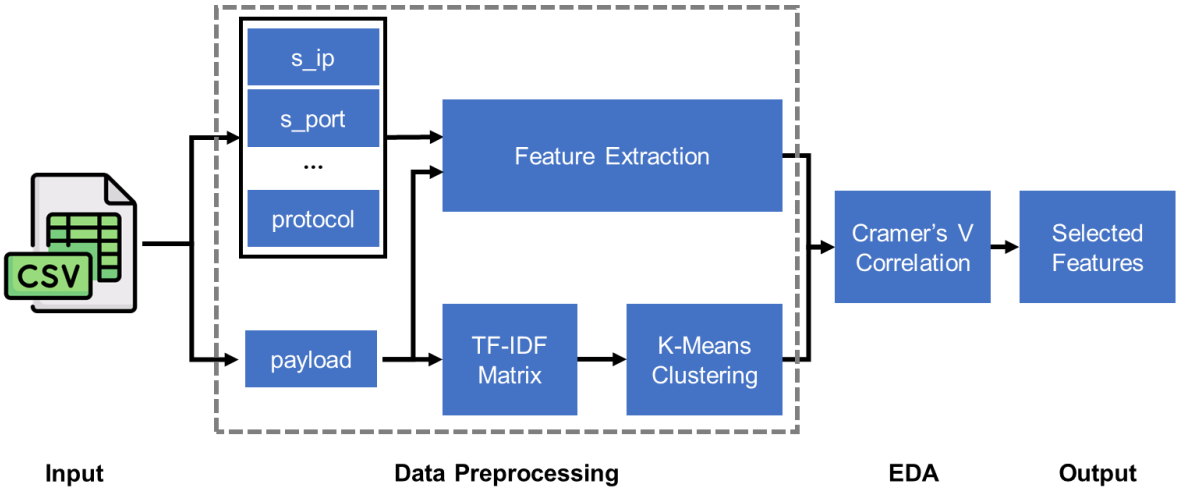


그림 4. Data Preprocessing 단계 흐름도

그림 4와 같이 학습셋에서 주어진 모든 feature(s_ip, s_port, d_ip, d_port, protocol, payload, risk)는 1.5절과 같이 처리합니다. 또한 payload는 아래와 같은 단계로 의미가 있으면서 중요도가 높은 단어를 기준으로 군집화합니다.

- 먼저 불용어를 제거하고 모두 소문자로 변환하여 의미 있는 단위로 토큰화를 합니다.

- 토큰화된 단위에서 단어의 원형을 뽑아내는 lemmatization 어근 추출을 진행합니다.
- 어근으로 정제된 페이로드를 Term Frequency - Inverse Document Frequency(TF-IDF) 행렬로 표현합니다.
- K-means Clustering을 통해 43개의 군집(cluster)으로 군집화를 수행합니다. 군집의 수를 2개부터 60개까지 변화시키면서 모델의 성능을 측정하였고, 그 결과 그림 5와 같이 43개가 가장 높은 성능을 이끌어 냄을 확인했습니다.

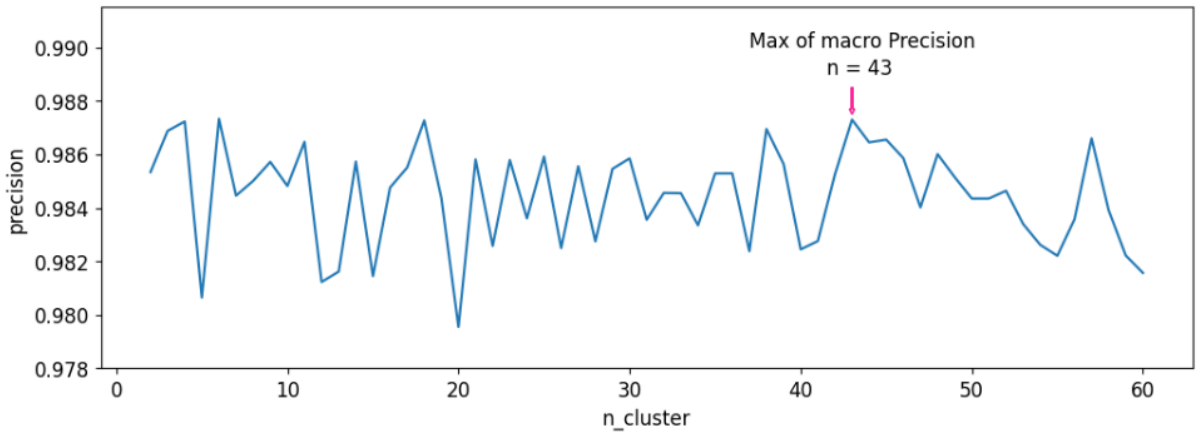


그림 5. 군집 수에 따른 모델 macro precision

3.2 EDA(Exploratory Data Analysis)

데이터셋에 있는 모든 변수와 전처리 과정에서 생성했던 파생 변수는 모두 범주형 변수이므로 범주형 변수 간 상관 계수를 나타내는 Cramer's V correlation을 이용하였습니다.

3.3 Learning & Classification

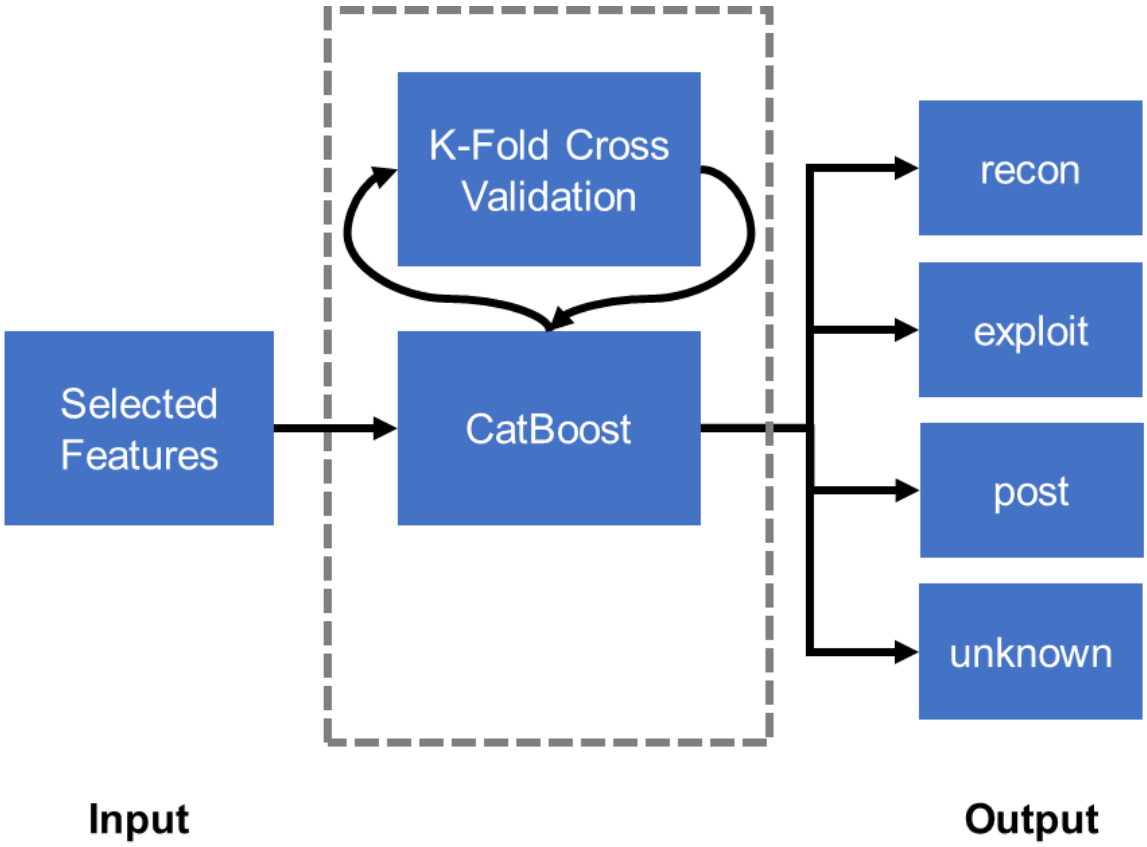


그림 6. Learning & Classification 단계 흐름도

저희 팀은 그림 6과 같이 CatBoost 모델을 이용하여 네트워크 공격을 분류하였으며 CatBoost 모델 학습 시 K-Fold를 사용하여 5겹 교차검증을 수행하며 모델링을 진행하였습니다.

저희 팀이 학습 모델로 CatBoost 모델을 선택한 것은 데이터셋을 관찰한 결과 데이터셋의 feature가 대부분 범주형 변수인 것을 확인했고, CatBoost 모델이 범주형 변수 분류에 적합한 모델이기 때문입니다.

범주형 데이터를 기존의 여러 머신러닝 모델로 돌리기 위해서는 Label Encoding 혹은 One-Hot Encoding의 방식으로, 범주형 변수들을 범주형 값으로 변환하는 과정이 필요합니다. Label Encoding은 문자열 값을 수치형 범주 값으로 변환해주는데, 해당 데이터의 경우 각 변수가 가진 개체 수가 많아서 너무 넓은 범위의 수치형이 생성되어 성능저하를 일으킵니다. One-Hot Encoding은 해당하는 고윳값 칼럼에만 1을 나머지 칼럼에는 0을 부여하는 벡터 표현 방식으로, 각 변수의 개체수가 많은 해당 데이터에 적용하면 칼럼의 수가 지나치게 많아지는 문제가 있습니다. 이러한 문제들을 해결하기 위해 CatBoost 모델을 선정했습니다.

CatBoost는 Categorical Boost의 약자로 이름에서부터 알 수 있듯이 범주형 변수들을 위한 부스팅 기법입니다. 위와 같은 범주형 값으로 변환하는 별도의 과정 없이도 강력한 성능을 자랑하므로, 해당 데이터에 가장 적합하다고 판단하였습니다.

분류결과

표 3. 학습셋의 공격 분류 결과에 대한 Confusion Matrix

pred / real	1_reconnaissance	2_exploit	3_post	4_unknown
1_reconnaissance	2,437	59	6	1
2_exploit	70	19,605	15	0
3_post	11	26	11,673	8
4_unknown	8	11	5	16,065
합계	2,526	19,701	11,699	16,074

전체 50,000개의 데이터 중 1_reconnaissance는 2,526개, 2_exploit는 19,701개, 3_post는 11,699개, 4_unknown은 16,074개로 분류하였으며 이 중에서 정탐은 총 49,780개 입니다.

표 4. 학습셋의 공격 분류 결과에 대한 precision, recall, f1-score (캡쳐화면 필요)

구분	클래스명	Precision	Recall	F1-Score

1	1_reconnaissance	0.95	0.96	0.95
2	2_exploit	0.99	0.99	0.99
3	3_post	1.00	1.00	1.00
4	4_unknown	1.00	1.00	1.00
결과	*Precision with Macro Average = 0.9873*			

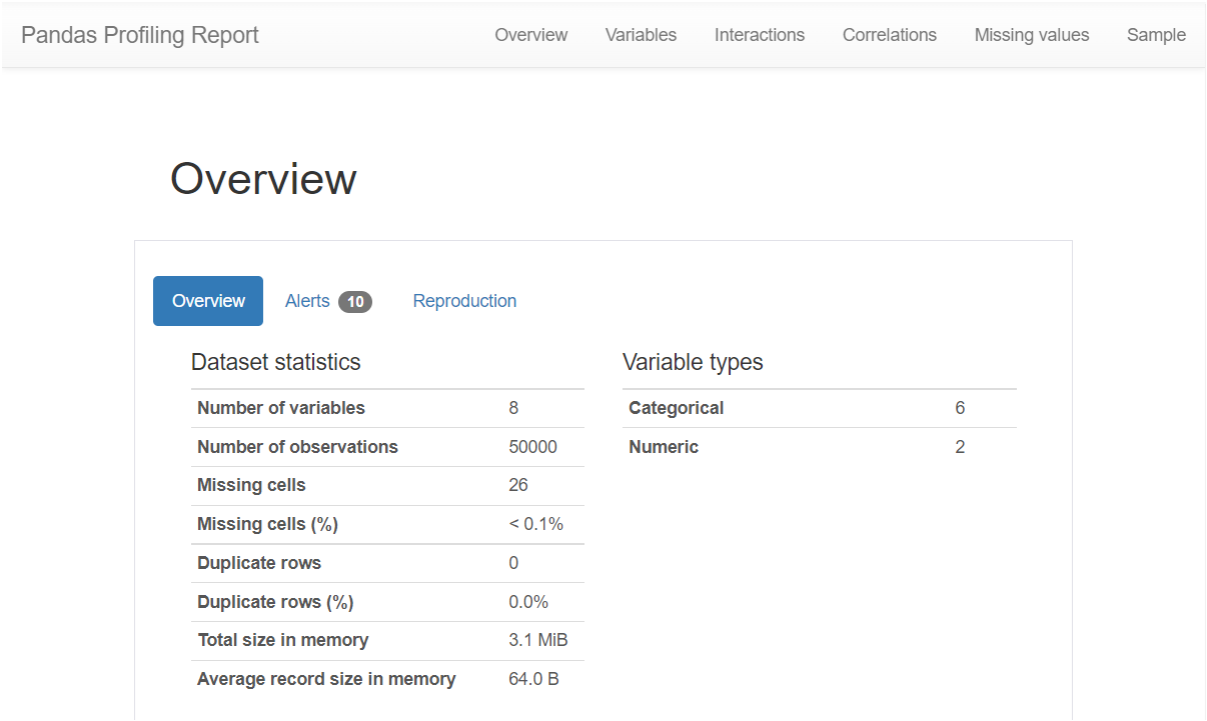
이번 대회의 평가지표는 macro precision으로 저희가 모델링한 모델의 macro precision은 표4와 같이 약 0.9873의 성능을 보였습니다. "3_post", "4_unknown"의 분류는 precision, recall, f1-score 모두 1.00으로 완벽한 수준으로 분류하고 있고, "2_exploit" 역시 모든 평가지표가 0.99로 분류가 잘 되고 있다고 할 수 있습니다. 하지만 "1_reconnaissance"의 평가지표는 약 0.95~0.96으로 다른 class에 비해 약간 떨어지는 성능을 가지고 있습니다. 현재 학습셋으로 주어진 feature 외에 reconnaissance의 특성을 잘 설명할 수 있는 변수를 찾아 모델링 변수로 추가하면 보다 더 나은 성능을 가진 모델을 만들 수 있을 것 같습니다

.

프로그램 설명

1. 데이터 분석

```
x = pd.read_csv("track_b_learn.csv").drop(["idx"],axis=1)
y = pd.read_csv("track_b_learn_label.csv").drop(columns = ["idx"],axis =1)
display(x.head(), y.head())
df = pd.concat([x,y],axis =1)
dftest = df.copy()
profile = ProfileReport(dftest)
profile
```



데이터를 불러오고 pandas_profiling 라이브러리 [11]의 ProfileReport 함수를 사용하여 간단한 데이터 분석을 실행합니다.

2. Data Processing

2.1 Feature Extraction

```
x["s_ip_class"] = np.nan
x["d_ip_class"] = np.nan

x["s_ip_class"] = np.where(x["s_network_id"] <= 127 , "A", x["s_ip_class"])
x["s_ip_class"] = np.where((x["s_network_id"] >= 128) & (x["s_network_id"] <= 191) , "B", x["s_ip_class"])
x["s_ip_class"] = np.where((x["s_network_id"] >= 192) & (x["s_network_id"] <= 223) , "C", x["s_ip_class"])
x["s_ip_class"] = np.where((x["s_network_id"] >= 224) , "D or E", x["s_ip_class"])

x["d_ip_class"] = np.where(x["d_network_id"] <= 127 , "A", x["d_ip_class"])
x["d_ip_class"] = np.where((x["d_network_id"] >= 128) & (x["d_network_id"] <= 191) , "B", x["d_ip_class"])
x["d_ip_class"] = np.where((x["d_network_id"] >= 192) & (x["d_network_id"] <= 223) , "C", x["d_ip_class"])
x["d_ip_class"] = np.where((x["d_network_id"] >= 224) , "D or E", x["d_ip_class"])
```

s_ip, d_ip의 네트워크 주소 값에 따라 네트워크 클래스를 나타내는 s_ip_class, d_ip_class 변수 값을 할당합니다.

```
x["s_port_class"] = np.nan
x["d_port_class"] = np.nan

x["s_port_class"] = np.where(x["s_port"] <= 1023 , "well-known", x["s_port_class"])
x["s_port_class"] = np.where((x["s_port"] >= 1024) & (x["s_port"] <= 49151) , "registered", x["s_port_class"])
x["s_port_class"] = np.where((x["s_port"] >= 49152) & (x["s_port"] <= 65535) , "dynamic", x["s_port_class"])

x["d_port_class"] = np.where(x["d_port"] <= 1023 , "well-known", x["d_port_class"])
x["d_port_class"] = np.where((x["d_port"] >= 1024) & (x["d_port"] <= 49151) , "registered", x["d_port_class"])
x["d_port_class"] = np.where((x["d_port"] >= 49152) & (x["d_port"] <= 65535) , "dynamic", x["d_port_class"])
```

s_port, d_port 변수를 값에 따라 세 가지(well-known, registered, dynamic)로 분류하고 s_port_class, d_port_class 변수에 할당합니다.

```
def is_xml_rpc(s: str):
    return "<methodCall>" in s

def is_shell_shock(s: str):
    return "() { :; }; echo" in s

def is_sql_injection(s: str):
    dmls = ["insert", "update", "delete", "select", "INSERT", "UPDATE", "DELETE", "SELECT"]
    return int(any(list(map(lambda d: d in s, dmls))))

def is_malicious_execution(s: str):
    return "/etc/passwd" in s

def is_xss_attack(s: str):
    return "<script>" in s

def is_directory_traversal(s: str):
    return "apexec.pl" in s

# Post
def is_web_shell(s: str):
    return "aspydrv" in s

def is_shell_file(s: str):
    return "shell" in s
```

payload 변수 값에서 식별한 네트워크 공격 패턴에 따라 변수를 생성합니다.

- XML-RPC: 페이로드에 "<methodCall>" 문자열이 포함되어 있다면 공격이 있는 것으로 판단하였습니다 [12].
- Shell Shock: 페이로드에 "User-Agent" 필드가 있고, 그 값에 "() { :; }; echo" 문자열이 포함되어 있다면 공격이 있는 것으로 판단하였습니다 [13].
- SQL Injection: 페이로드에 "insert", "update", "delete", "select" 가 있다면 공격이 있는 것으로 판단하였습니다.

- Malicious execution: HTTP 메시지 내용에 passwd 파일 경로인 "/etc/passwd" 가 있다면 공격이 있는 것으로 판단하였습니다.
- XSS Attack: HTTP 메시지 내용에 "<script>" 가 있다면 공격이 있는 것으로 판단하였습니다.
- Directory traversal Attack: HTTP 메시지 내용에 "axec.pl" 이 있다면 공격이 있는 것으로 판단하였습니다 [14].
- Web shell: 페이로드에 "Content-Disposition" 필드가 있고, 그 값에 "shell" 이 있다면 공격이 있는 것으로 판단하였습니다 [15, 16].

```
output_csv_path = "ip_location.csv"
output = pd.read_csv(output_csv_path) if pathlib.Path(output_csv_path).exists() else None


s_ip = df1["s_ip"]
d_ip = df1["d_ip"]
len_df1 = len(df1)

columns = ["idx" , "s_country_code", "d_country_code"]
df = pd.DataFrame(list(), columns= columns) if output is None else output
len_df = len(df)

try:
    for i in range(0, 50001): # len_df1
        s_country_code = None if len_df <= i else df["s_country_code"][i]
        d_country_code = None if len_df <= i else df["d_country_code"][i]
        if (s_country_code is not None and not pd.isna(s_country_code)) and (d_country_code is not None and not pd.isna(d_country_code)):
            continue

        print(f"\n{i}")
        s_country_code = find_country_code(api_keys[0], s_ip[i])
        d_country_code = find_country_code(api_keys[1], d_ip[i])
        df.loc[i] = [i, s_country_code, d_country_code]
except KeyboardInterrupt:
    pass
```

s_ip, d_ip 변수 값을 이용하여 국가 코드 변수(s_country_code, d_country_code)를 생성합니다.

2.2 TF-IDF Matrix & K-Means Clustering (페이로드 전처리 및 군집화)

```
from nltk.stem import WordNetLemmatizer
import nltk
import string

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
lemmar = WordNetLemmatizer()

def LemTokens(tokens):
    return [lemmar.lemmatize(token) for token in tokens]

def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from tqdm import tqdm
tfidf_vect = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english' , \
                             ngram_range=(1,2), min_df=0.05, max_df=0.85 )

tfidf = tfidf_vect.fit(x["payload"])
tfidf.get_feature_names()
feature_vect = tfidf_vect.transform(x['payload'])

from sklearn.cluster import KMeans

km_cluster = KMeans(n_clusters = 43, max_iter=10000, random_state=999)
km_cluster.fit(feature_vect)
```

페이로드가 문자열로 되어있기 때문에 이를 모델에 학습시키기 위해 적절하게 데이터형을 바꿔주어야 합니다. TfidfVectorizer를 사용해서 텍스트 벡터화를 수행하고 이를 K-Means clustering을 통해 43개의 군집으로 분류했습니다.

3. CatBoost & K-Fold Cross Validation (모델 학습)

```
from sklearn.model_selection import train_test_split, KFold

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 42)
cv = KFold(n_splits=5, shuffle=True, random_state=999)
```

```
scores = []

models = []
for tri, vai in cv.split(x_train):
    print("="*50)
    preds = []

    model = CatBoostClassifier(iterations=2000,random_state=999,eval_metric="TotalF1",cat_features=cat_features)
    model.fit(x_train.iloc[tri], y_train.iloc[tri],
            eval_set=[(x_train.iloc[vai], y_train.iloc[vai])],
            early_stopping_rounds=50 ,
            verbose = 100
        )

    models.append(model)
    scores.append(model.get_best_score()["validation"]["TotalF1"])
    if False:
        break
```

데이터셋을 학습/검증 데이터셋으로 나누고 CatBoost 모델을 이용하여 학습했습니다.

4. 결과 확인

```
from sklearn.metrics import classification_report

y_pred_tr= model.predict(x_train)
print(classification_report(y_train, y_pred_tr))

y_pred_te= model.predict(x_test)
print(classification_report(y_test, y_pred_te))

from sklearn.metrics import precision_score, f1_score
print(precision_score(y_pred_tr, y_train, average = "macro"))
print(precision_score(y_pred_te, y_test, average = "macro"))
print(f1_score(y_pred_tr, y_train, average = "macro"))
print(f1_score(y_pred_te, y_test, average = "macro"))

crosstab = pd.crosstab(predcon['class'], predt['predclass'], rownames=['real'], colnames=['pred'])
crosstab
```

모델이 데이터를 잘 학습했는지 확인하기 위해 precision, recall, f1-score 지표를 확인하고 교차표를 생성합니다.

결론

전체 50,000개의 데이터 중 1_reconnaissance는 2,526개, 2_exploit는 19,701개, 3_post는 11,699개, 4_unknown은 16,074개로 분류하였으며 macro precision은 약 0.9873, f1-score는 약 0.9860을 달성하였습니다.

저희 팀은 데이터셋이 모두 범주형 변수로 이루어진 것에서 범주형 모델을 사용하는 것을 착안하였습니다. 범주형 변수에 대해 우수한 성능을 자랑하는 CatBoost 모델로 모델링을 수행했습니다.

페이로드 변수는 값 그대로 사용할 수 없기 때문에 텍스트 마이닝하는 방법을 떠올렸습니다. 페이로드 값을 TF-IDF 기반으로 벡터화 한 후 K-means로 군집화를 진행했습니다. 이 과정으로, 처리 및 활용이 다소 어려울 수 있던 페이로드에서 범주형 파생 변수를 생성하여 모델링에 사용했습니다. 실제로 이렇게 만들어진 파생 변수의 feature importance가 사용된 변수들 중 가장 높게 나타나, 모델의 분류 예측에 큰 기여를 한 것을 확인할 수 있었습니다.

저희 팀의 모델은 모든 네트워크 공격유형을 100% 정확하게 분류하기에는 아직 부족함이 존재합니다. 그 이유는 confusion matrix에서 볼 수 있듯이 1_reconnaissance과 2_exploit를 정확하게 분류하는 것이 부족하기 때문입니다. 그러나 오답으로 표시한 데이터들을 분석하여 1_reconnaissance과 2_exploit를

구별할 수 있는 새로운 feature를 연구해 발전시킨다면 모델의 성능을 더욱 향상시킬 수 있을 것으로 기대됩니다
.

참고문헌

[1] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In A dvances in neural information processing systems 30 (2017).

[2] Anna Veronika Dorogush, Vasily Ershov and Andrey Gulin. CatBoost: gradient boo sting with categorical features support. arXiv preprint arXiv:1810.11363 (2018).

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In Pr oceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[4] Wikipedia, "Classful network", https://en.wikipedia.org/wiki/Classful_network, Retri eved October 7, 2022.

[5] 한국인터넷정보센터, "해외 전체현황(IPv4 주소)", https://한국인터넷정보센터.한국/jsp/statboard/ IPAS/ovrse/total/currentV4Addr.jsp, Retrieved October 9, 2022.

[6] Wikipedia, "List of TCP and UDP port numbers", https://en.wikipedia.org/wiki/List _of_TCP_and_UDP_port_numbers, Retrieved October 8, 2022.

[7] MITRE, "MITRE ATT&CK Enterprise tatics", https://attack.mitre.org/tactics/enterpris e Retrieved October 3, 2020.

[8] abstract, "Free IP Geolocation API", https://www.abstractapi.com/api/ip-geolocati on-api, Retrieved October 10, 2022.

[9] ipapi, “IP Address Lookup and Geolocation API”, https://ipapi.co, Retrieved October 10, 2022.

[10] Jessica L Purser. Using Generative Adversarial Networks for Intrusion Detection in Cyber-Physical Systems. Naval Postgraduate School, 2020.

[11] ydataai, “Create HTML profiling reports from pandas DataFrame objects”, https://github.com/ydataai/pandas-profiling, Retrieved October 20, 2022.

[12] Lucian Nitescu Security Blog, “Exploiting the xmlrpc.php on all WordPress versions”, https://nitesculucian.github.io/2019/07/01/exploiting-the-xmlrpc-php-on-all-wordpress-versions/, Retrieved October 11, 2022.

[13] sullo, “nikto_shellshock.plugin”, https://github.com/sullo/nikto/blob/master/program/plugins/nikto_shellshock.plugin, Retrieved October 11, 2022.

[14] Secui, “취약점 상세 설명 보기”, http://update.secui.com/vuln_detail_desc.asp?id=21170&page=534, Retrieved October 11, 2022.

[15] MDN, “Content-Disposition”, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Disposition, Retrieved October 11, 2022.

[16] Yara-Rules, “WShell_THOR_Webshells.yar”, https://github.com/Yara-Rules/rules/blob/master/webshells/WShell_THOR_Webshells.yar, Retrieved October 11, 2022.

===== 사용하지 않는 내용 적는 곳

- 식별한 7개의 공격 이외에 페이로드에는 식별하지 못한 공격이 존재할 수 있습니다. 또한 학습셋에서 주어진 5만개의 페이로드를 전수조사하기에는 시간이 매우 많이 소요되기 때문에 페이로드 값 자체를 가공하는 방법을 생각하였습니다. 따라서 아래와 같은 단계로 페이로드에서 의미가 있으면서 중요도가 높은 단어를 추출하였습니다.
    - 먼저 불용어를 제거하고 모두 소문자로 변환하여 의미 있는 단위로 토큰화를 합니다.

- 토큰화된 단위에서 단어의 원형을 뽑아내는 lemmatization 어근 추출을 진행합니다.
- 어근으로 정제된 페이로드를 Term Frequency - Inverse Document Frequency(TF-IDF) 행렬로 표현합니다.
- K-means clustering을 통해 **43**개의 집합으로 군집화를 수행합니다. 이후 군집 번호를 가지는 cluster_label 변수를 생성합니다.

- 데이터 이산화 (Discretization)

  - 출발지 PORT(s_port), 목적지 PORT(d_port)는 번호 값에 따라 세가지 라벨로 분류 변환합니다. PORT 번호는 16비트로 이루어져 총 65536개가 존재할 수 있으며 해당 PORT들은 번호별로 세 종류로 나뉘져 관리됩니다. 해당 기준을 참고하여 PORT 번호를 분류합니다. PORT 번호가 0부터 1023 이하면 잘 알려진, 유명한 포트로 정의되어 있으므로 "well_known"으로 변환합니다. 1024부터 49151이면 기관이나 사업자들을 위해 관리중인, 등록된 포트이므로 "registered", 49152부터 65535번은 일반 사용자들이 자유롭게 사용할 수 있는 동적인 포트로 "dynamic"으로 분류합니다. 이렇게 65536개의 PORT를 세 분류 값으로 변환합니다.

- 데이터 특징 추출

  - 페이로드(payload)를 데이터의 원래 입력 값 그대로 사용하기에는 어려움이 있습니다. 페이로드 내의 특징을 추출하여 페이로드의 주요 내용을 활용할 수 있는 feature를 생성합니다. 페이로드 내에 특정 공격이 존재하는지 여부를 확인한 후, 존재하면 1을 그렇지 않으면 0을 입력 값으로 가지도록 각각의 공격을 feature로 설정합니다. 포함여부를 확인할 공격 종류는 "xml_rpc", "sql_injection", "shell_shock" ,"malicious_execution", "xss_attack", "directory_traversal", "web_shell", "shell_fire"로 총 8개이며, 따라서 0 또는 1의 값을 가지는 8개의 feature가 생깁니다.

2.1 분류를 위해 선정한 feature

- s_ip: payload의 출발지 IP
- s_port: payload의 출발지 PORT
- d_ip: payload의 도착지 IP
- d_port: payload의 도착지 PORT
- protocol: payload를 전송하는 통신 규약
- payload: 해당 공격 로그에서 전송된 데이터
- risk: payload의 위험도

[s_ip / d_ip 파생변수]

- s_network_id: '.'으로 구분된 출발지 IP의 가장 앞부분 번호
- s_ip2: '.'으로 구분된 출발지 IP의 두 번째 번호
- s_ip3: '.'으로 구분된 출발지 IP의 세 번째 번호
- s_ip4: '.'으로 구분된 출발지 IP의 마지막 번호
- d_network_id: '.'으로 구분된 도착지 IP의 가장 앞부분 번호
- d_ip2: '.'으로 구분된 도착지 IP의 두 번째 번호
- d_ip3: '.'으로 구분된 도착지 IP의 세 번째 번호
- d_ip4: '.'으로 구분된 도착지 IP의 마지막 번호
- s_host_ip: s_ip2, s_ip3, s_ip4를 합친 출발지 host IP 주소
- d_host_ip: d_ip2, d_ip3, d_ip4를 합친 도착지 host IP 주소
- s_ip_class: 클래스 기반의 IP 주소 체계를 참고해, s_network_id로 나눈 클래스
- d_ip_class: 클래스 기반의 IP 주소 체계를 참고해, d_network_id로 나눈클래스
- s_country_code: 국가별 IP 대역을 참고해, 공격 로그를 보내는 IP의 국가 코드
- d_country_code: 국가별 IP 대역을 참고해, 공격 로그의 목적지 IP의 국가 코드

[s_port / d_port 파생변수]

- s_port_class: 출발지 PORT 번호를 세 범위로 나눈 클래스
- d_port_class: 도착지 PORT 번호를 세 범위로 나눈 클래스

[payload 파생변수]

- cluster_label: payload를 기반으로 25개의 집합으로 군집화한 클러스터

  + cluster_label 생성 과정

  payload를 입력값으로 받아 -> 불용어 제거 -> 소문자 변환 -> 단어 토큰화 -> lemmatization 어근 변환 및 추출 -> 추출된 어근으로 feature vectorization 수행해 tf-idf 행렬 생성 -> tf-idf 행렬을 기반으로 25개의 집합으로 군집화 수행 -> 포함되는 군집 번호를 저장하는 cluster_label 생성

- attack_pattern: payload 기반의 공격 여부 변수

  + attack_pattern 생성 과정

  payload를 입력값으로 받아 -> HTTP 요청 메소드 종류 찾기 -> 메소드 내에 8가지의 공격 유형 각각 존재 여부 확인 -> 각 공격 유형을 컬럼으로 생성한 후 -> 존재하면 1, 존재하지 않으면 0 저장 -> 0 또는 1로 채워진 8개의 컬럼 값을 합쳐서 attack_pattern 생성

- 각 공격 유형 컬럼: payload 내 공격 유형 존재 여부 0, 1 값

  : "has_xml_rpc" , "has_sql_injection" , "has_shell_shock" , "has_malicious_execution" , "has_xss_attack" , "has_directory_traversal" , "has_web_shell" , "has_shell_file"

2.2 feature 선정 기준(이유)

- s_ip: payload의 출발지 IP 정보를 담고 있는 feature이므로 중요할 것으로 판단했습니다. 신호를 보내는 곳이 어디인지가 해당 로그의 공격 유형 분류에 큰 영향을 미칠 것으로 예상했습니다.
- s_port: payload의 출발지 PORT이므로 payload를 보내는 곳을 분류할 수 있어, 공격 유형 판정에 중요할 것으로 예상해 선정했습니다.
- d_ip: payload를 보내고자 하는 목적지 IP로, payload가 어디로 보내졌는지가 해당 공격 로그의 유형을 분류하는 데에 도움이 될 것으로 예상했습니다.

- d_port: payload가 도착하는 PORT로, 목적지 식별에 활용되어 큰 영향력을 끼칠 것으로 기대했습니다.
- protocol: 해당 공격 로그가 따르는 통신 규약을 나타내므로, 어떤 규약인지가 공격 유형에 분류에 유의한 영향을 미칠 것으로 예상했습니다.
- payload: 공격 로그에서 전송된 데이터로, 해당 로그의 내용을 담고 있습니다. 각 공격 로그가 보내고자 한 핵심 데이터이므로 공격 유형 분류에 영향도가 클 것으로 예측했습니다.
- risk: 공격 로그의 위험도를 나타내는 feature이므로, 공격 유형과 유의한 상관관계가 있을 것으로 생각되었습니다.
- s_country_code: 공격 로그를 보내는 IP 국가가 공격 유형에 영향을 미칠 것이라고 예상해 해당 feature를 선정했습니다.
- d_country_code: 공격 로드의 신호가 도착하는 목적지 국가로, 목적지 국가가 공격 유형에 영향을 미칠 것으로 기대했습니다.
- s_port_class: 공격 로그를 보내는 PORT를 세 범주로 분류한 feature로, 출발지 PORT의 종류가 공격 로그의 유형을 판별하는 데 영향을 미칠 것으로 기대했습니다.
- d_port_class: 공격 로그가 도착하는 PORT를 세 범주로 분류한 feature로, 해당 공격 로그가 흘러 들어가는 목적지 판별에 도움을 줘 유형 분류에 도움이 될 것으로 예상했습니다.
- cluster_label: 공격 로그의 데이터인 payload를 클러스터링한 값으로, 해당 데이터가 분류되는 클러스터가 공격 유형 분류에도 영향을 미칠 것으로 기대해 선정하였습니다.
- attack_pattern: payload 내에 존재하는 공격 유형을 나타내는 feature로, 해당 공격 로그의 유형을 분류하는 데에 중요도가 있을 것으로 예측했습니다.

2.3 feature 별 중요도

- s_port: 약 15 정도의 중요도를 가지며, 사용한 feature 중 세 번째로 중요하다고 나왔습니다.
- d_port: 약 1.5 정도의 중요도를 가진 것으로 나옵니다. 수치로만 보면 중요도가 크다고 보기는 어려우나 전체 feature 중, 중간 수준의 중요도를 가집니다.
- protocol: 변수 중요도가 약 35 정도로, feature 들 중에서 가장 중요도가 높게 나타납니다.
- risk: 약 10 정도의 중요도를 가지며, 전체 feature 중 네 번째로 높은 중요한 변수입니다.

- s_country_code: 약 0.5의 수준으로 낮은 수치를 보이며 다소 중요도가 낮게 영향을 미친다고 나타났습니다.
- d_country_code: s_country_code와 마찬가지로 약 0.5 정도의 낮은 중요도를 가지는 것을 확인할 수 있습니다.
- s_port_class: 중요도가 전체 feature 중 여섯 번째에 위치하며, 약 3 정도로 나타났습니다.
- d_port_class: s_port_class와 비슷하게 3 정도 수준의 중요도를 보이며, 일곱 번째에 위치합니다.
- cluster_label: 전체 feature 중 두 번째로 높은 중요도로 약 19 정도로 나타납니다.
- attack_pattern: 약 8 정도의 중요도를 가지는 것으로 측정되었으며, 전체 중 다섯 번째로 중요도가 높은 것으로 나타났습니다.
- 각 공격 유형 컬럼: "malicious_execution"과 "directory_traversal", "xml_rpc"는 각각 약 2, 1, 1 정도로 그나마 낮은 중요도로 작은 영향을 미치고 있는 것을 확인할 수 있습니다. "sql_injection", "web_shell", "xss_attack", "shell_file", "shell_shock"은 중요도가 거의 0과 다름이 없는 수치로, 모델링에서 영향력을 끼치지 못한 것으로 나타납니다.

    - 일관된 형식으로 정리된 페이로드를 TF-IDF 행렬로 표현합니다.

      -TF-IDF는 전체의 각 문서(payload)에서 해당 단어가 나온 빈도를 비교하여 단어의 중요도를 고려하는 방식입니다. 단순히 단어의 출현 빈도를 고려하는 DTM보다 더욱 유의미한 정보를 얻을 수 있으므로 채택했습니다.

    - K-means Clustering을 통해 25개의 집합으로 군집화를 수행합니다.

      -주어진 데이터가 대용량이고 payload를 범주화할 기준과 정보가 명확하지 않으므로, 값들 사이의 거리를 측정하여 군집화하는 과정을 반복하여 최적의 군집화 결과를 도출하는 K-means가 적절하다 판단되어 해당 방식을 채택하였습니다.

-군집을 25개로 설정한 것은, 차례로 군집의 수를 늘려가며 정밀도를 측정한 결과, 25개의 군집일 때 정밀도가 우수했고 그 이상으로는 군집을 늘려도 정밀도가 오르지 않았기 때문입니다.

3.1 AI 알고리즘 구성

- 50000행의 데이터셋을 train 데이터와 test 데이터로 75:25의 비율로 나눠줍니다.
- 최적의 모델을 얻기 위해 KFold 기법으로 5겹 교차 검증을 수행합니다. 분류 모델로는 CatBoost를 채택했습니다.
- 5겹 교차 검증으로 총 5개의 fold로 나뉘진 train 데이터로 교차검증을 수행하였고, 모델이 최적값을 찾아가는 eval_metric은 Total F1 Score로 설정하여 모델링을 수행하였습니다.
- 그렇게 선정된 모델에 최종 train 데이터와 test 데이터를 각각 넣어 분류를 수행합니다. 분류 이후 classification report를 통해 각 예측값에 대한 모델의 성능을 확인합니다.
- 마지막으로 해당 대회의 평가지표인 macro precision과 다중분류에서 일반적으로 사용하는 F1 Score를 출력해 전체적인 모델의 성능을 최종적으로 확인합니다.

3.2 AI 알고리즘 선정 기준(이유)

- 데이터셋을 살펴보면 해당 데이터의 Feature가 모두 범주형 변수인 것을 확인할 수 있습니다. 이러한 데이터를 기존의 여러 머신러닝 모델로 돌리기 위해서는 'Label Encoding' 혹은 'One-Hot Encoding'의 방식으로, 범주형 변수들을 범주형 값으로 변환하는 과정이 필요합니다. 'Label Encoding'은 문자열 값을 수치형 범주 값으로 변환해주는데, 해당 데이터의 경우 각 변수가 가진 개체 수가 많아서 너무 넓은 범위의 수치형이 생성되어 성능저하를 일으킵니다. 'One-Hot Encoding'은 해당하는 고유값 컬럼에만 1을 나머지 컬럼에는 0을 부여하는 벡터 표현 방식으로, 각 변수의 개체수가 많은 해당 데이터에 적용하면 컬럼의 수가 지나치게 많아지는 문제가 있습니다. 이러한 문제들을 해결하기 위해 CatBoost 모델을 선정했습니다. CatBoost는 'Categorical Boost'의

약자로 이름에서부터 알 수 있듯이 범주형 변수들을 위한 부스팅 기법입니다. 범주형 값으로 변환하는 별도의 과정 없이도 강력한 성능을 자랑하므로, 해당 데이터에 가장 적합하다고 판단하였습니다.

3.3 모델 하이퍼파라미터 튜닝

- 하이퍼파라미터 튜닝에 민감한 기존의 XGBoost 모델이나 LGBM 모델과 다르게 CatBoost 모델은 하이퍼파라미터 튜닝을 내부적인 알고리즘으로 해결하고 있어 하이퍼파라미터 튜닝이 불필요합니다. 굳이 튜닝을 수행한다면, learning_rate, L2_regulariser 인데 성능에 큰 차이가 없습니다. 결론으로 Catboost 모델은 기본적으로 최적화가 잘 되어있어 하이퍼파라미터 튜닝을 따로 수행하지는 않았고 교차검증을 통해 새로운 데이터가 들어왔을때 얼마나 성능이 균일한가에 대한 검증만을 수행하였습니다.

=====

TODO : 221015

1. ~~recon 분류가 잘 안되고 있다고 하였는데, 개선할 수 있는 방안 논의 + 보고서에 서술 필요. recon 을 분류하는 기술이 우리 팀의 핵심 기술이라고 할 수 있음(~ 월 회의 전)~~
2. ROC 그래프 그리기(해석은 나중에 다같이)(~ 일 23:59 월 본인 출근 전): **맹 -> 원래 이진분류에만 사용하는거라 쓸모없음**
3. ~~방법론에서 IP 를 특정 클래스에 매핑하는 것 보다 위치 정보로 매핑하여 사용하는 것이 공격을 더 잘 판단이라고 서술했는데, 그럼 우리 파쳐 선정 시에서도 사용을 안해야 맞음. 하지만 사용을 하는게 맞을지 vs 위치만 사용할지 결정 필요 (~ 일).~~
4. 피쳐 - 라벨 간의 상관계수(cramer) 구하기(~ 일): **찬성**
    - https://dodonam.tistory.com/217
    - https://github.com/kaveio/phik
5. feature importance 그림(그림 2) 다시 구하기(~ 일): **찬성**
6. 군집 갯수에 따라 성능치 보여주는 그래프 구하기(~ 월): **맹 -> 찬성님 함수 만들어서 진행중**
7. 맨 위의 AI 학습 구성도 그림 그리기(~ 월): **민욱+지수**

8. isolation forest: **맹 -> 이상치가 너무 많아서 피쳐로 넣기에는 의미없음**
9. 프로그램 코드 설명 **맹**
10. ~~1을 맞추는 애들과 앙상블?~~
11. 보고서 고치기: **민욱+지수**
12. 모델링 파트: **지수**
13. 중복페이로드 x 국가코드 상관관계 파악: **지수 -> s_ip국가코드랑 label class 교차분석은 p_value 0.0003정도로 유의하다고 나오고, d_ip국가코드랑 label class 교차분석은 p_value 0.46 정도로 유의하지 않다고 나옵니다.**