



# Anomaly Detection

Detection Malware Using Machine Learning Models 

# What is Anomaly Detection

*“Anomaly (outlier, novelty) detection is identification of unusual observations that are unlike most of the data”*



CCTV

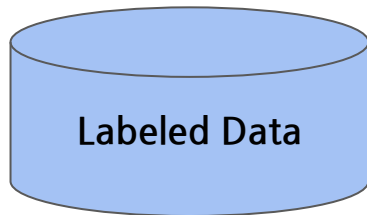


medical imaging

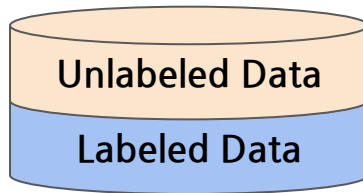


social network

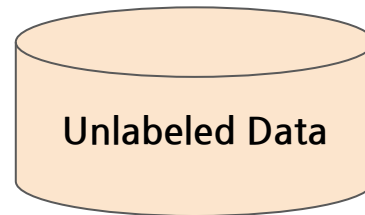
# Data Characteristic for anomaly detection



Supervised



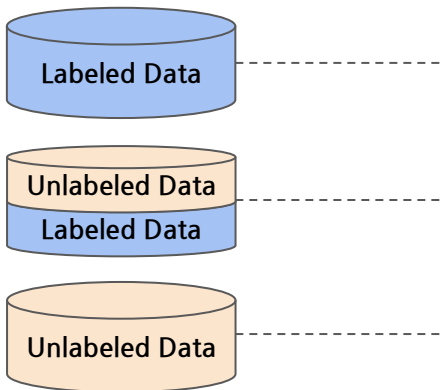
Semi-Supervised



Unsupervised

- Different Anomaly Detection approaches
  - Supervised : Inputs and associated Good and Bad labels
  - Unsupervised : At training time all you have is a completely unlabeled set of data, but anomalies are relatively uncommon.
  - Semi-Supervised : At training time all you have is representation of good. You want to learn bad in terms of deviation from good.

# Data Characteristic for anomaly detection



The diagram on the left shows three data sources represented as cylinders. The top cylinder is blue and labeled 'Labeled Data'. The middle cylinder is orange with a blue band at the bottom labeled 'Labeled Data'. The bottom cylinder is orange and labeled 'Unlabeled Data'. Dashed lines connect these cylinders to the rows of the table: the top cylinder to the 'Supervised' row, the middle cylinder to the 'Semi-Supervised' row, and the bottom cylinder to the 'Unsupervised' row.

Learning Method	Normal Sample	Abnormal Sample
Supervised	Use for training	Use for training
Semi-Supervised	Use for training	Does not use for training
Unsupervised	Don't Know (no label). Assume that major samples are normal.	

# Applications of Anomaly Detection

## “ *Malware Executable Detection(악성코드실행탐지)* ”

Cases of detecting malware (malware). **Classification** and **clustering** are mainly used, and Malware tabular data is used as it is, or it is converted into a gray scale image and used.

- Classification : Supervised learning As a kind of supervised learning, it is a process of identifying the category relationship of existing data and determining the category of newly observed data by itself.  
supervised learning 지도학습의 일종으로 기존에 존재하는 데이터의 category 관계를 파악하고 새롭게 관측된 데이터의 category를 스스로 판별하는 과정
- Clustering : Clustering is a method of classifying data in the absence of any information.  
군집화는 아무런 정보가 없는 상태에서 데이터를 분류하는 방법

# Importing Libraries & Load Dataset

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
```

```
data = pd.read_csv('../input/malware-executable-detection/uci_malware_detection.csv')
data.head()
```

	Label	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	...	F_522	F_523	F_524	F_525	F_526	F_527	F_528	F_529	F_530	F_531
0	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
1	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
2	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
3	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
4	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0

5 rows × 532 columns

The data file consists of a total of 373 samples, of which 301 are malicious files and the remaining 72 are non-malicious data, with 531 functions from F1 to F531, and a label indicating whether the file is malicious or non-malignant. I have a fever.

데이터파일은 총 373개의 샘플로 구성  
그 중 301개는 악성 파일이고 나머지 72개는  
악성이 아닌 데이터로 구성  
F1부터 F531까지 531개의 기능이 있으며  
파일이 악성인지 비악성인지를 나타내는  
레이블 열이 있다.

# Checking for NaN values / EDA & Visualization

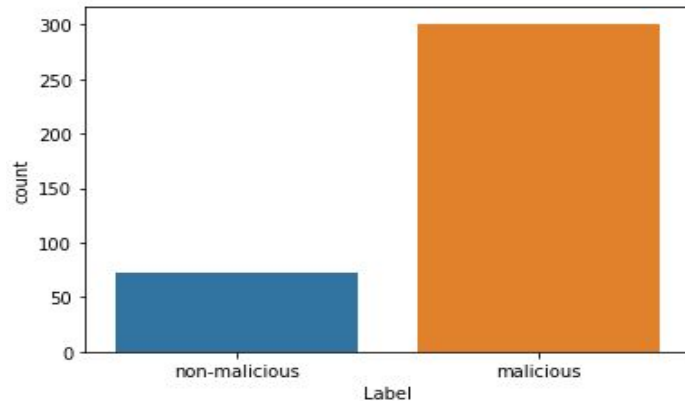
## Checking for NaN values

```
data.isnull().sum()
```

```
Label      0  
F_1        0  
F_2        0  
F_3        0  
F_4        0  
..  
F_527      0  
F_528      0  
F_529      0  
F_530      0  
F_531      0  
Length: 532, dtype: int64
```

## EDA & Visualization

```
sns.countplot(x='Label', data=data);
```



# Data Splitting

```
X = data.drop(["Label"],axis=1)
y = data['Label'].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=10)
```



# Training models

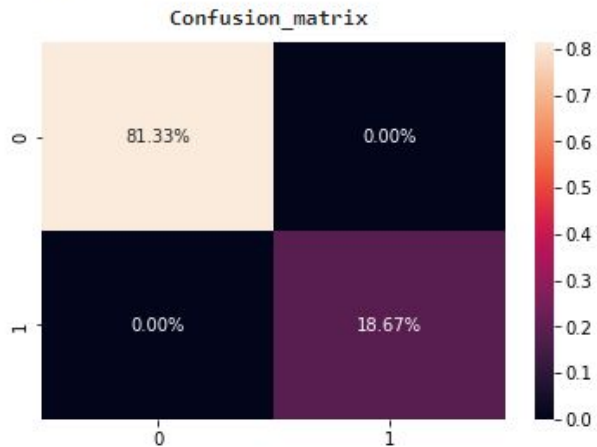
```
models = [RandomForestClassifier, DecisionTreeClassifier, KNeighborsClassifier, AdaBoostClassifier, SGDClassifier,
           ExtraTreesClassifier, GaussianNB]
accuracy_test=[]
model = []
for m in models:
    model_name = type(m).__name__
    print('#####-Model =>\033[07m {} \033[0m'.format(type(m).__name__))
    model_ = m()
    model_.fit(X_train, y_train)
    pred = model_.predict(X_test)
    acc = accuracy_score(pred, y_test)
    accuracy_test.append(acc)
    model.append(model_name)
    print('Test Accuracy :\033[32m \033[01m {:.5f}% \033[30m \033[0m'.format(acc*100))
    print('\033[01m          Classification_report \033[0m')
    print(classification_report(y_test, pred))
    print('\033[01m          Confusion_matrix \033[0m')
    cf_matrix = confusion_matrix(y_test, pred)
    plot_ = sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt= '0.2%')
    plt.show()
    print('\033[31m#####- End -#####\033[0m')
```

- RandomForestClassifier
- DecisionTreeClassifier
- KNeighborsClassifier
- AdaBoostClassifier
- SGDClassifier
- ExtraTreesClassifier
- GaussianNB

# Evaluation

Test Accuracy : 100.00000%

Classification_report				
	precision	recall	f1-score	support
malicious	1.00	1.00	1.00	61
non-malicious	1.00	1.00	1.00	14
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75
weighted avg	1.00	1.00	1.00	75



- RandomForestClassifier

Algorithms based on decision trees

결정 트리를 기반으로 하는 알고리즘

Several decision tree classifiers sample their data based on bagging, perform training, and finally decide a prediction through voting. 여러 개의 결정 트리 분류기가 Bagging을 기반으로 각자의 데이터를 샘플링 하여 학습을 수행한 후에 최종적으로 보팅을 통해 예측을 결정하게 된다.

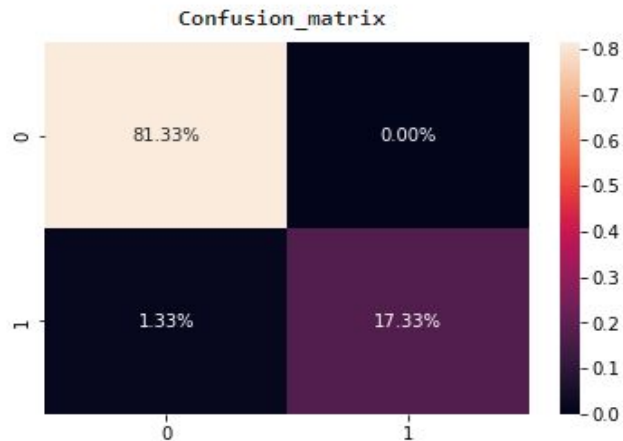
- Bagging : 데이터 샘플링을 통해 모델을 학습시키고 결과를 집계하는 방법
- Voting : 즉 단어 뜻 그대로 투표를 통해 최종 예측 결과를 결정하는 방식

precision	precision : 정밀도(모델이 true라고 분류한 것 중 에서 실제 true 인 것의 비율)
recall	recall : 재현율(실제 true 인 것 중에서 모델이 true 라고 예측한 것의 비율)
f1 score	f1 score : precision 과 recall 사이의 조화 평균(주로 성능 비교를 할 때 사용)
support	샘플개수
heatmap()	혼동행렬, 정확도를 가지고 시각화
macro avg	전체의 값들의 평균
weighted avg	각 클래스에 해당하는 데이터의 개수에 가중치를 주어 평균을 구한것

# Evaluation

Test Accuracy : 98.66667%

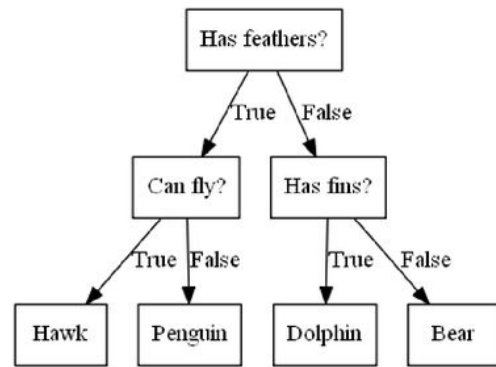
Classification_report				
	precision	recall	f1-score	support
malicious	0.98	1.00	0.99	61
non-malicious	1.00	0.93	0.96	14
accuracy			0.99	75
macro avg	0.99	0.96	0.98	75
weighted avg	0.99	0.99	0.99	75



## - DecisionTreeClassifier

Algorithm that automatically finds rules in data through learning and creates tree-based classification rules

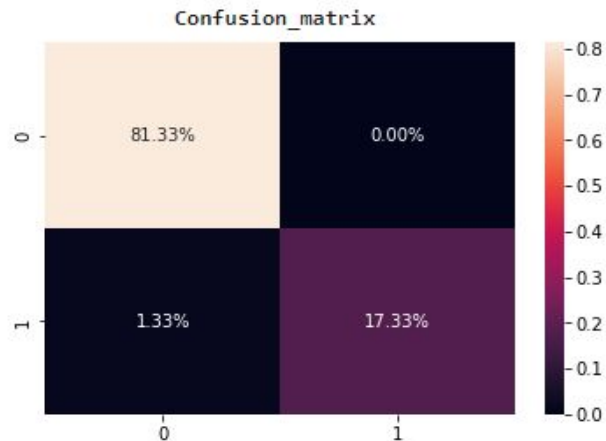
데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리 기반의 분류 규칙을 만드는 알고리즘



# Evaluation

Test Accuracy : 98.66667%

Classification_report				
	precision	recall	f1-score	support
malicious	0.98	1.00	0.99	61
non-malicious	1.00	0.93	0.96	14
accuracy			0.99	75
macro avg	0.99	0.96	0.98	75
weighted avg	0.99	0.99	0.99	75



## - KNeighborsClassifier

Simple Machine Learning Classification Using Nearest Neighbor Classification Algorithm

Sorting by the closest member

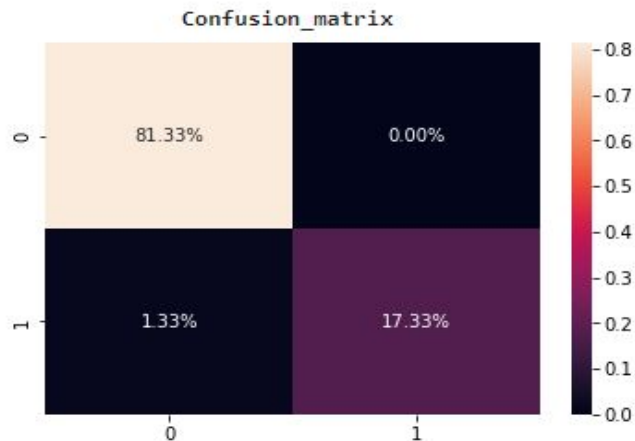
최근접 이웃 분류 알고리즘을 활용한 간단한 머신러닝 분류

가장 가깝게 위치하는 멤버로 분류하는 방식

# Evaluation

Test Accuracy : 98.66667%

Classification_report				
	precision	recall	f1-score	support
malicious	0.98	1.00	0.99	61
non-malicious	1.00	0.93	0.96	14
accuracy			0.99	75
macro avg	0.99	0.96	0.98	75
weighted avg	0.99	0.99	0.99	75



## - AdaBoostClassifier

It is to modify the weights for the samples at each iteration.

Each trained classifier is weighted according to its performance, and the label with the highest value is selected by applying weights based on the label predicted by each classifier in the prediction.

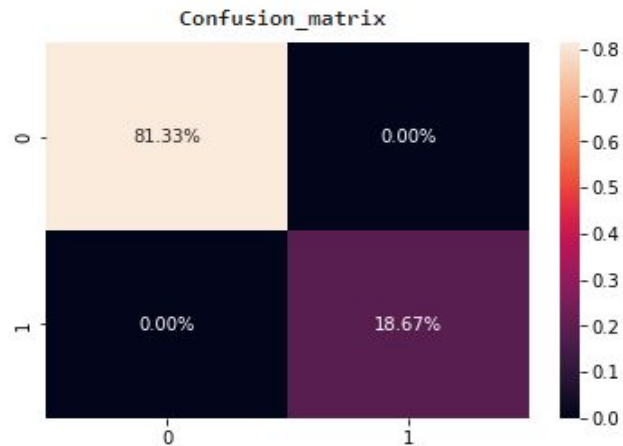
반복마다 샘플에 대한 가중치를 수정하는 것이다.

훈련된 각 분류기는 성능에 따라 가중치가 부여되고, 예측에서 각 분류기가 예측한 레이블을 기준으로 가중치까지 적용해 가장 높은 값을 가진 레이블을 선택한다.

# Evaluation

Test Accuracy : 100.00000%

Classification_report				
	precision	recall	f1-score	support
malicious	1.00	1.00	1.00	61
non-malicious	1.00	1.00	1.00	14
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75
weighted avg	1.00	1.00	1.00	75



## - SGDClassifier

Implementing a linear model using Stochastic Gradient Descent (SGD)

Based on the calculated value, if the calculated value is less than 0, it is classified as -1, if it is greater than 0, it is classified as 1.

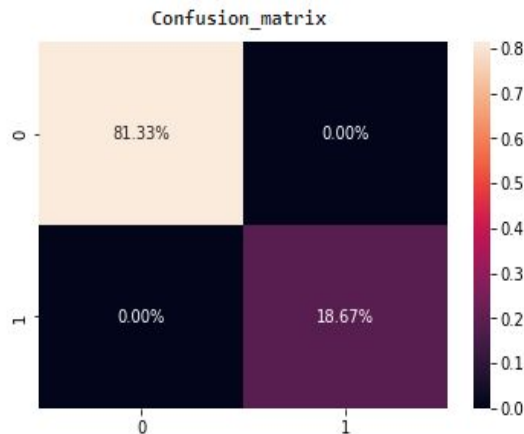
확률적 경사하강법(SGD, Stochastic Gradient Descent)을 이용하여 선형 모델을 구현

계산값을 기반으로 계산값이 0보다 작으면 -1, 0보다 크면 1로 분류

# Evaluation

Test Accuracy : 100.00000%

Classification_report				
	precision	recall	f1-score	support
malicious	1.00	1.00	1.00	61
non-malicious	1.00	1.00	1.00	14
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75
weighted avg	1.00	1.00	1.00	75



## - ExtraTreesClassifier

The randomness is increased by randomly partitioning each candidate feature of the forest tree.

To make the tree more random, instead of finding the optimal threshold, we split it randomly using candidate features and then choose the best split among them.

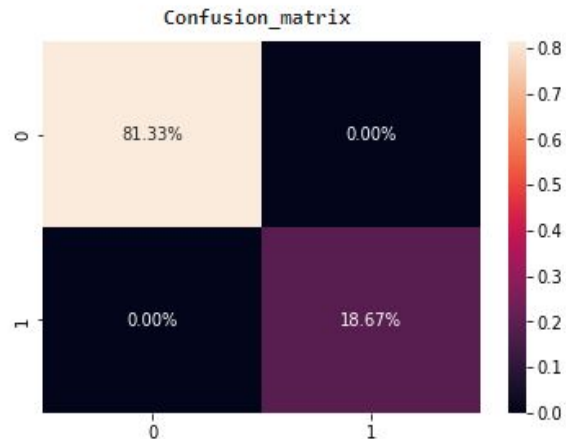
포레스트 트리의 각 후보 특성을 무작위로 분할하는식으로 무작위성을 증가시킨다.

트리를 더 무작위하게 만들기 위해 최적의 임계값을 찾는 대신 후보 특성을 사용해 무작위로 분할한 다음 그중에서 최상의 분할을 선택한다.

# Evaluation

Test Accuracy : 100.00000%

	Classification_report			
	precision	recall	f1-score	support
malicious	1.00	1.00	1.00	61
non-malicious	1.00	1.00	1.00	14
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75
weighted avg	1.00	1.00	1.00	75



## - GaussianNB

As a simple technique to create a classifier, it is not trained through a single algorithm, but is trained using several algorithms based on general principles.

분류기를 만들 수 있는 간단한 기술로써 단일 알고리즘을 통한 훈련이 아닌 일반적인 원칙에 근거한 여러 알고리즘들을 이용하여 훈련된다.



# Final Report

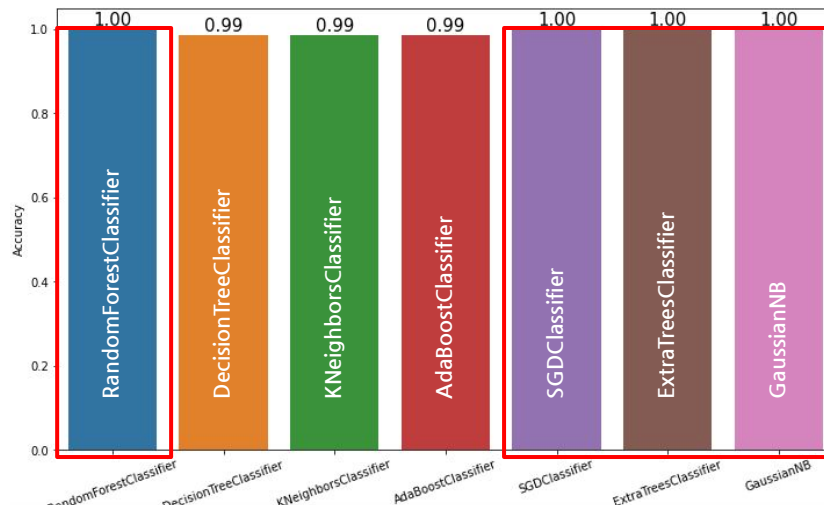
```
model = pd.Series(model, name='Model').astype(str)
accuracy = pd.Series(accuracy_test, name='Accuracy')
output = pd.concat([model, accuracy], axis=1)
```

output

	Model	Accuracy
0	RandomForestClassifier	1.000000
1	DecisionTreeClassifier	0.986667
2	KNeighborsClassifier	0.986667
3	AdaBoostClassifier	0.986667
4	SGDClassifier	1.000000
5	ExtraTreesClassifier	1.000000
6	GaussianNB	1.000000

```
plt.figure(figsize=(12, 7))
plots = sns.barplot(x='Model', y='Accuracy', data=output)
for bar in plots.patches:
    plots.annotate(format(bar.get_height(), '.2f'),
                  (bar.get_x() + bar.get_width() / 2,
                   bar.get_height()), ha='center', va='center',
                  size=15, xytext=(0, 8),
                  textcoords='offset points')

plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.xticks(rotation=20);
```



# References

- <https://www.kaggle.com/code/hamzamanssor/detection-malware-using-machine-learning-models>
- <https://hoya012.github.io/blog/anomaly-detection-overview-1/>
- <https://www.youtube.com/watch?v=-h9yK4FUyk>