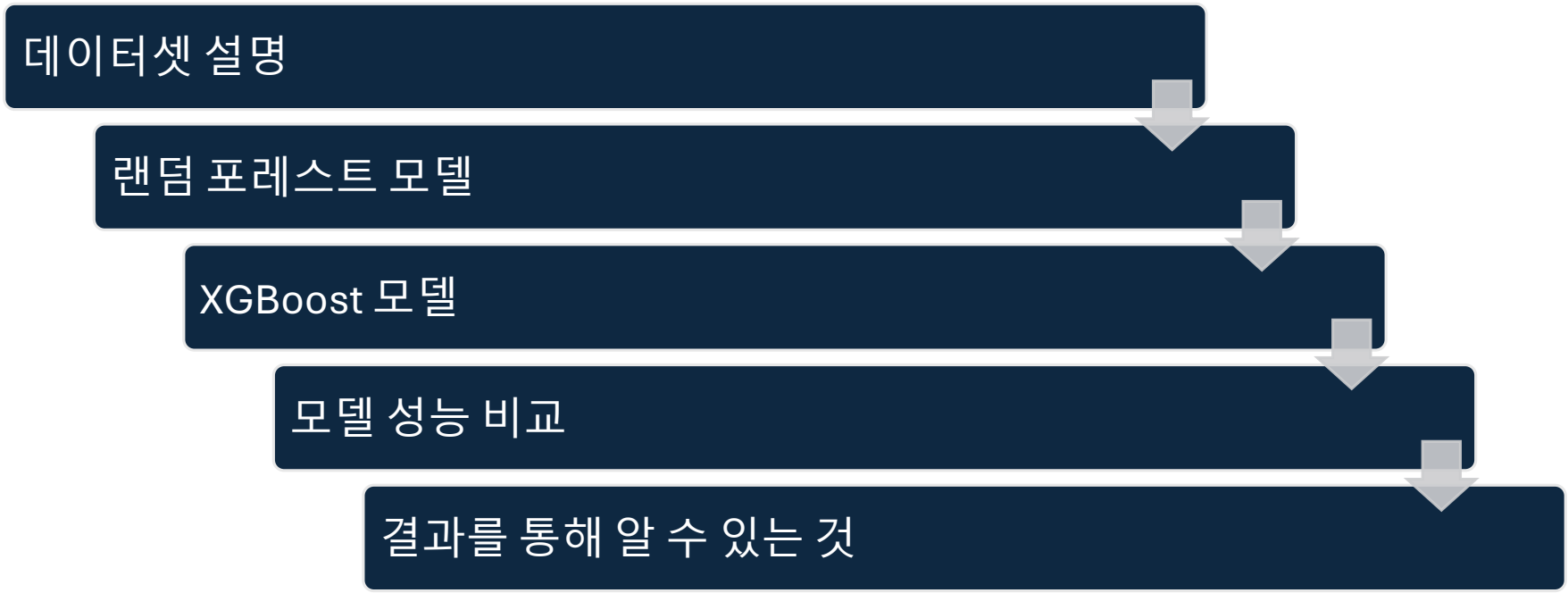
# IoT 네트워크 공격 탐지: 랜덤 포레스트와 XGBoost 모델 비교 분석

2023270112 이현규 인공지능개론 과제

# 목차



# 개요

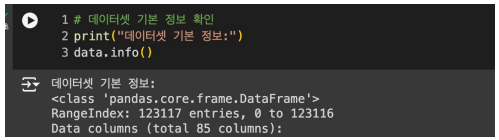
**목적**: IoT 네트워크 공격 유형을 분류하기 위해 랜덤 포레스트와 XGBoost 모델의 성능을 비교 분석

**데이터셋**: IoT 네트워크 트래픽 데이터셋 (RT_IOT2022.csv)

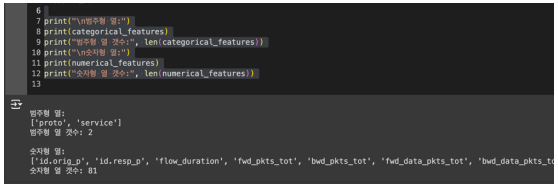
**목표**: 고성능 분류 모델을 활용해 다양한 공격 유형을 정확하게 탐지

# 데이터셋 설명

- **총 데이터 포인트**: 123,117개
- **특성 수**: 85개 (숫자형 82개, 범주형 3개)
- **목표 레이블 (Attack_type)**: 12개의 공격 유형 (예: 'ARP_poisioning', 'DDOS_Slowloris', 'MQTT_Publish' 등)

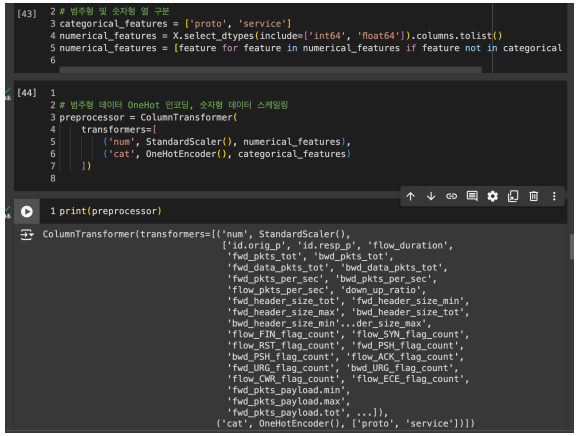


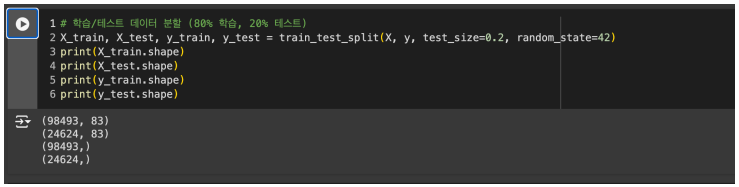




# 데이터셋 설명

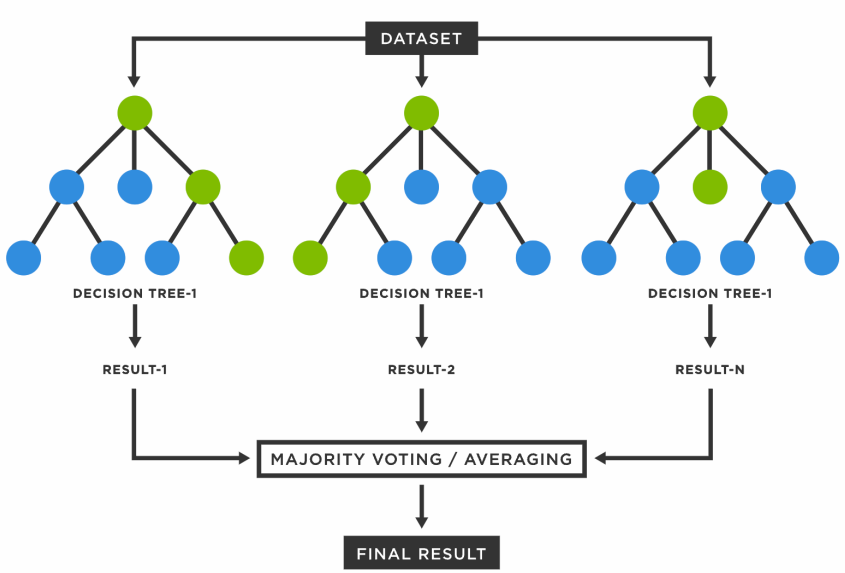
- 전처리
  - **숫자형 데이터**는 StandardScaler()를 사용하여 평균이 0, 분산이 1이 되도록 표준화
    - 이는 다양한 범위를 가진 데이터가 모델에 균형 있게 반영되도록 하기 위함.
  - **범주형 데이터**는 OneHotEncoder()를 사용하여 각 카테고리마다 별도의 열을 생성해 이진형으로 변환.
    - 범주형 변수를 수치형으로 변환해 모델이 이를 효과적으로 학습하도록 하기 위함.



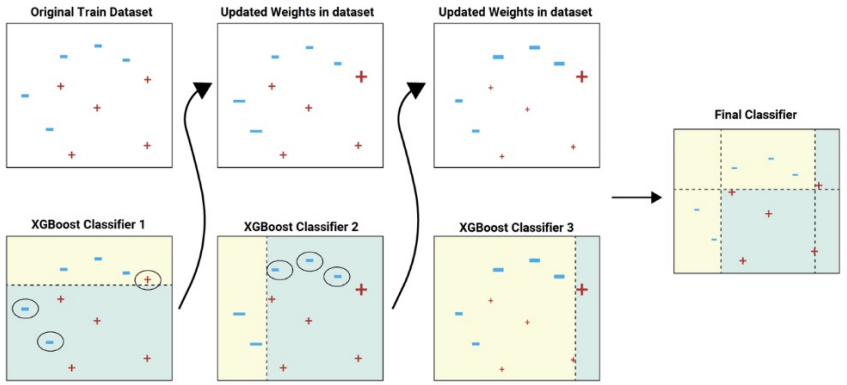
# 랜덤 포레스트 모델

- **개념**: 랜덤 포레스트는 여러 개의 결정 트리(Decision Tree)를 앙상블하여 예측 성능을 높이는 방법입니다. 각 트리는 서로 다른 데이터 샘플과 특성을 이용해 학습하며, 최종 예측값은 다수결 또는 평균값으로 결정됩니다.
- **장점**:
- 과적합 방지: 다수의 트리를 이용하여 과적합(overfitting)을 방지하고 안정적인 예측이 가능함.
- 높은 정확도: 다양한 데이터 패턴을 학습할 수 있어 예측 성능이 높음.
- 사용 용이성: 다양한 데이터 타입과 크기에서도 잘 작동함.
- **단점**:
- 해석성 낮음: 여러 트리가 결합된 결과물이므로 개별 예측의 해석이 어려움.
- 속도: 트리가 많아질수록 학습 및 예측 시간이 오래 걸릴 수 있음.
- **주요 특징**:
- 배깅(Bagging) 기법 사용: 무작위 데이터 샘플로 트리를 생성하여 예측의 다양성과 안정성을 높임.
- 분산 감소: 여러 트리의 예측을 결합해 예측의 변동성을 줄임.



# XGBoost 모델

- **개념**: XGBoost는 그래디언트 부스팅을 개선한 방법으로, 순차적으로 트리를 추가하면서 오류를 줄여 나가는 앙상블 기법입니다. 이전 트리의 오류를 다음 트리가 보완해나가며, 성능과 속도 면에서 뛰어난 최적화가 이루어집니다.
- **장점**:
- 높은 예측 성능: 부스팅 기법을 이용해 오차를 단계적으로 줄여 높은 성능을 보임.
- 효율성: CPU와 메모리 효율이 뛰어나고, 대규모 데이터에서도 빠르게 학습 가능.
- 다양한 기능: 결측값 처리, 조기 종료 기능으로 최적의 성능 달성 가능.
- **단점**:
- 튜닝 필요성: 하이퍼파라미터가 많아 최적의 성능을 위해 세밀한 조정이 필요함.
- 과적합 위험: 트리를 순차적으로 학습하기 때문에 과적합의 위험이 있음.
- **주요 특징**:
- 그래디언트 부스팅 사용: 각 트리가 이전 예측의 오류를 보완해 나가면서 성능을 향상시킴.
- L1, L2 정규화: 모델의 복잡도를 줄이고 과적합을 방지하기 위해 정규화 기법 사용.



# 모델 성능 비교

```python
# 랜덤 포레스트 모델 학습
rf_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])

rf_pipeline.fit(X_train, y_train)
rf_preds = rf_pipeline.predict(X_test)
```

```python
# XGBoost 모델 학습
xgb_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42))
])

xgb_pipeline.fit(X_train, y_train)
xgb_preds = xgb_pipeline.predict(X_test)
```

# 모델 성능 비교

- **정확도 비교**: 랜덤 포레스트(99.79%) vs XGBoost(99.81%)
- **소수 클래스 성능**: XGBoost가 소수 클래스에서 좀 더 높은 성능을 보임
- **추천 모델**: XGBoost - 실시간 탐지 시스템에서 성능이 우수할 것으로 예상

```
1
2 # 평가 함수 정의
3 def evaluate_model(name, y_true, y_pred):
4     print(f"--- {name} Model ---")
5     print("Accuracy:", accuracy_score(y_true, y_pred))
6     print("\nClassification Report:\n", classification_report(y_true, y_pred, target_names=label_encoder.classes_))
7     print("\nConfusion Matrix:\n", confusion_matrix(y_true, y_pred))
8
```

# 모델 성능 비교

```
1
2 # XGBoost 평가
3 evaluate_model("XGBoost", y_test, xgb_preds)
4
```

```
--- XGBoost Model ---
Accuracy: 0.9981319038336582

Classification Report:
                            precision    recall  f1-score   support

              ARP_poisioning       0.98      0.99      0.99      1578
              DDOS_Slowloris       0.96      0.97      0.97       100
               DOS_SYN_Hping       1.00      1.00      1.00     18897
                MQTT_Publish       1.00      1.00      1.00       871
Metasploit_Brute_Force_SSH       0.83      0.83      0.83         6
             NMAP_FIN_SCAN       1.00      0.67      0.80         3
           NMAP_OS_DETECTION       1.00      1.00      1.00       393
             NMAP_TCP_scan       1.00      1.00      1.00       220
             NMAP_UDP_SCAN       0.99      0.98      0.99       489
         NMAP_XMAS_TREE_SCAN       1.00      0.99      1.00       384
               Thing_Speak       0.99      0.99      0.99      1625
                Wipro_bulb       1.00      0.95      0.97        58

                  accuracy                           1.00     24624
                 macro avg       0.98      0.95      0.96     24624
              weighted avg       1.00      1.00      1.00     24624


Confusion Matrix:
 [[ 1568     1     0     0     0     0     0     0     0     9     0]
 [    2    97     0     0     0     0     0     1     0     0     0]
 [    0     0 18897     0     0     0     0     0     0     0     0]
 [    0     0     0   870     0     0     0     0     0     1     0]
 [    0     0     0     0     5     0     0     0     1     0     0]
 [    1     0     0     0     0     2     0     0     0     0     0]
 [    0     0     0     0     0     0   393     0     0     0     0]
 [    0     0     0     0     0     0     0   220     0     0     0]
 [    4     3     0     0     1     0     0     0   479     0     2]
 [    2     0     0     0     0     0     0     1   381     0     0]
 [   14     0     0     0     0     0     0     0     0  1611     0]
 [    1     0     0     1     0     0     0     0     0     1    55]]
```

```
1
2 # 랜덤 포레스트 평가
3 evaluate_model("Random Forest", y_test, rf_preds)
4
```

```
--- Random Forest Model ---
Accuracy: 0.9978882391163093

Classification Report:
                            precision    recall  f1-score   support

              ARP_poisioning       0.98      0.99      0.99      1578
              DDOS_Slowloris       0.99      0.98      0.98       100
               DOS_SYN_Hping       1.00      1.00      1.00     18897
                MQTT_Publish       1.00      1.00      1.00       871
Metasploit_Brute_Force_SSH       0.71      0.83      0.77         6
             NMAP_FIN_SCAN       1.00      0.67      0.80         3
           NMAP_OS_DETECTION       1.00      1.00      1.00       393
             NMAP_TCP_scan       1.00      1.00      1.00       220
             NMAP_UDP_SCAN       0.99      0.99      0.99       489
         NMAP_XMAS_TREE_SCAN       1.00      0.99      1.00       384
               Thing_Speak       0.99      0.99      0.99      1625
                Wipro_bulb       1.00      0.91      0.95        58

                  accuracy                           1.00     24624
                 macro avg       0.97      0.95      0.96     24624
              weighted avg       1.00      1.00      1.00     24624


Confusion Matrix:
 [[ 1567     0     0     0     1     0     0     0     2     0     8     0]
 [    0    98     0     2     0     0     0     0     0     0     0     0]
 [    0     0 18897     0     0     0     0     0     0     0     0     0]
 [    1     0     0   870     0     0     0     0     0     0     0     0]
 [    0     0     0     0     5     0     0     0     1     0     0     0]
 [    1     0     0     0     0     2     0     0     0     0     0     0]
 [    0     0     0     0     0     0   393     0     0     0     0     0]
 [    0     0     0     0     0     0     0   220     0     0     0     0]
 [    3     1     0     0     1     0     0     0   482     0     2     0]
 [    2     0     0     0     0     0     0     1   381     0     0     0]
 [   21     0     0     0     0     0     0     0     0     0  1604     0]
 [    4     0     0     1     0     0     0     0     0     0     0    53]]
```

# 결과를 통해 알 수 있는 것

- **1. 각 모델의 전반적인 성능**
- **랜덤 포레스트**와 **XGBoost** 모델 모두 매우 높은 정확도를 기록했습니다.
    - 랜덤 포레스트: 99.79%
    - XGBoost: 99.81%
- 두 모델 모두 IoT 네트워크의 다양한 공격 유형을 효과적으로 분류할 수 있는 능력을 보여줍니다.
- **2. 주요 공격 유형에 대한 높은 분류 정확도**
- DOS_SYN_Hping, NMAP_TCP_scan, NMAP_OS_DETECTION과 같은 주요 공격 유형에서 두 모델 모두 완벽에 가까운 분류 정확도를 보였습니다.
- 각 주요 공격 유형의 탐지율이 높은 것은 IoT 환경에서 발생할 수 있는 주된 보안 위협을 신속하게 탐지할 수 있음을 의미합니다.
- **3. 소수 클래스에 대한 성능 차이**
- **XGBoost**는 Metasploit_Brute_Force_SSH와 Wipro_bulb 등 소수 클래스에 대해 비교적 안정적인 성능을 보였으며, 랜덤 포레스트보다 다소 높은 정밀도와 재현율을 기록했습니다.
- 이로 인해 XGBoost는 드문 공격이나 특이한 패턴의 공격 탐지에서도 더 강력한 성능을 발휘할 수 있습니다.
- **4. 모델의 적합성 및 최적화 방향**
- **실시간 탐지**: XGBoost는 고성능을 유지하면서도 드문 유형의 공격을 더 잘 탐지할 수 있어, 실시간 IoT 공격 탐지 시스템에 적합합니다.
- **IoT 보안 강화**: 이 결과를 통해 각 공격 유형에 특화된 탐지 시스템을 구축하여 IoT 네트워크의 보안을 강화할 수 있습니다.
- **향후 개선 사항**: 드문 공격 유형에 대한 데이터를 추가 수집하고, 하이퍼파라미터 최적화를 통해 소수 클래스 성능을 더욱 향상시킬 여지가 있습니다.

감사합니다!