Cybersecurity Intrusion Prediction

Author: Madhura Atmaram Bhagat
Dataset: Cybersecurity Intrusion Detection Dataset

https://www.kaggle.com/code/madhuraatmarambhagat/cybersecurity-intrusion-prediction

---

1. 개요 (Overview)

이 프로젝트는 보안 트래픽/로그 데이터를 사용하여 **침입 여부를 예측하는 머신러닝 분류 모델**을 구축하는 과정을 다룹니다.
전체 절차는 다음 순서로 구성됩니다.

1) 라이브러리 및 데이터 불러오기
2) 탐색적 데이터 분석 (EDA)
3) 모델 학습(Classifiers)
4) 모델 평가
5) 결론

2. 라이브러리 및 데이터 불러오기

가. 설명

- 기본적인 데이터 분석 및 모델 학습을 위한 라이브러리를 불러옵니다.
- Kaggle dataset을 읽어서 DataFrame 형태로 불러옵니다.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,recall_score,precision_score
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

In [2]:
```
data=pd.read_csv("/kaggle/input/cybersecurity-intrusion-detection-dataset/cybersecurity_intrusion
_data.csv")
data.head(5)
```

Out[2]:

	session_id	network_packet_size	protocol_type	login_attempts	session_duration	encryption_used	ip_reputation_score	failed
0	SID_00001	599	TCP	4	492.983263	DES	0.606818	1
1	SID_00002	472	TCP	3	1557.996461	DES	0.301569	0
2	SID_00003	629	TCP	3	75.044262	DES	0.739164	2
3	SID_00004	804	UDP	4	601.248835	DES	0.123267	0
4	SID_00005	453	TCP	5	532.540888	AES	0.054874	1

3. 데이터 탐색 (EDA)

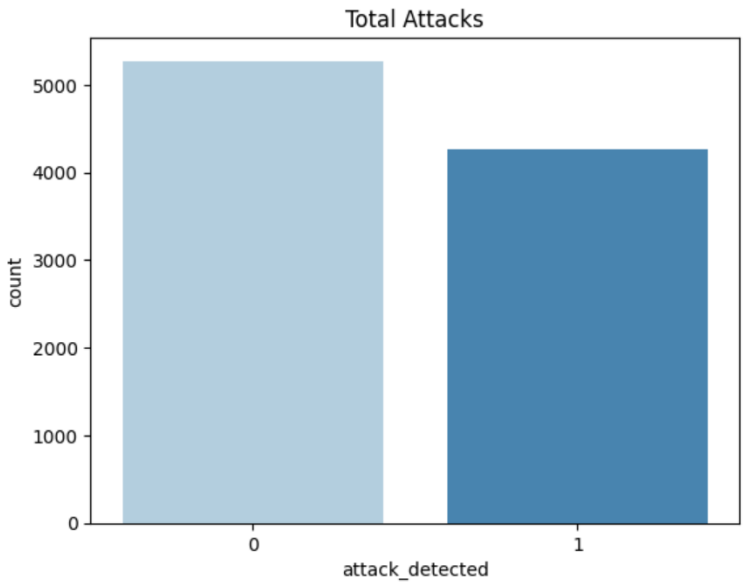
가. 설명

- 데이터 구조 확인
- 클래스 분포(정상 vs 침입 여부) 파악

나. 코드

In [8]:

```
sns.countplot(x="attack_detected",data=data,palette="Blues")
plt.title("Total Attacks")
plt.show()
```

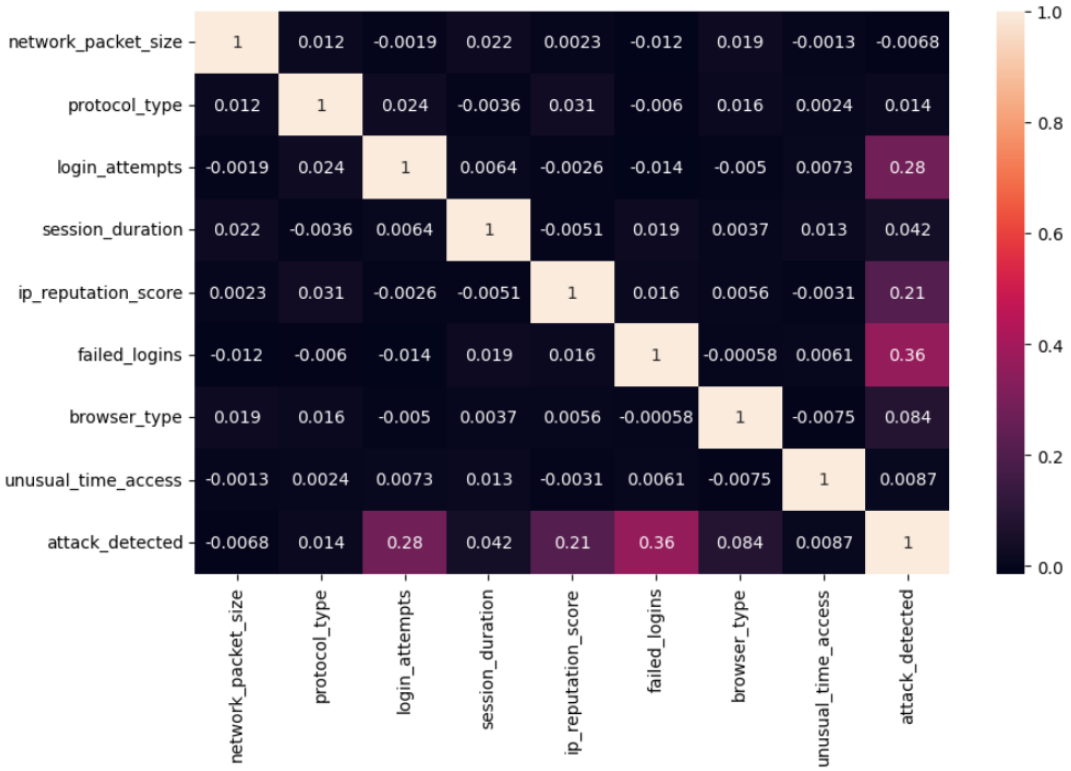


In [9]:

```
le=LabelEncoder()
data["protocol_type"]=le.fit_transform(data["protocol_type"])
data["browser_type"] = le.fit_transform(data["browser_type"])
```

In [10]:

```
plt.figure(figsize=(10,6))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

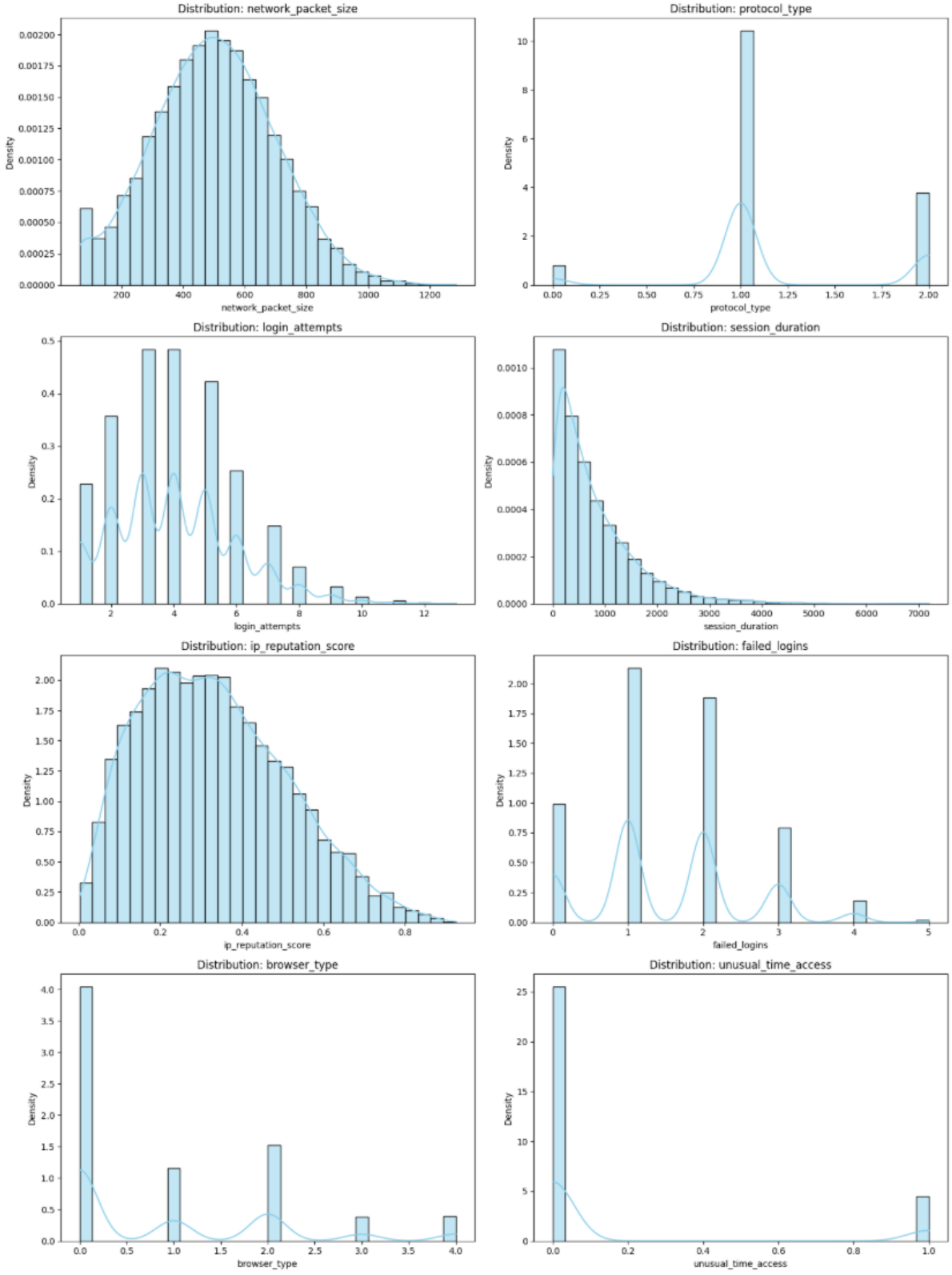


In [11]:

```
numeric_cols = data.select_dtypes(include=[np.number]).columns
num_cols = 2
num_rows = (len(numeric_cols) + num_cols - 1) // num_cols
plt.figure(figsize=(15, num_rows * 5))

for i, column in enumerate(numeric_cols, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.histplot(data[column], kde=True, stat='density', color='skyblue', bins=30)
    plt.title(f"Distribution: {column}")
    plt.xlabel(column)
    plt.ylabel('Density')

plt.tight_layout()
plt.show()
```


Distribution: network_packet_size
Density
network_packet_size
Distribution: protocol_type
Density
protocol_type
Distribution: login_attempts
Density
login_attempts
Distribution: session_duration
Density
session_duration
Distribution: ip_reputation_score
Density
ip_reputation_score
Distribution: failed_logins
Density
failed_logins
Distribution: browser_type
Density
browser_type
Distribution: unusual_time_access
Density
unusual_time_access

4. 모델 학습(Classifier)

가. 설명

- 모델 선택
- 학습용/테스트용 데이터 분리
- 모델 학습

나. 코드

In [13]:
```
models={
    "Random Forest Classifier":RandomForestClassifier(),
    "Decision Tree Classifier":DecisionTreeClassifier(),
    "Ada Boost Classifier":AdaBoostClassifier(),
    "Logistic Regression":LogisticRegression(),
    "Gradiant Boosting Classifier":GradientBoostingClassifier(),
    "Naive Bayes Classifier":GaussianNB()
}
```

In [14]:
```
x=data.drop(["attack_detected"],axis=1)
y=data["attack_detected"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.transform(x_test)
```

5. 모델 평가

가. 설명

노트북에서는 기본적인 지도학습 모델들을 적용하여 성능을 비교합니다.

- Random Forest
- Logistic Regression
- Decision Tree

나. 코드

In [15]:

```
metrics = {
    'Model': [],
    'Accuracy': [],
    'Precision': [],
    'Recall':[]
}
for name,model in models.items():
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    recall=recall_score(y_test,y_pred)
    precision=precision_score(y_test,y_pred)
    metrics['Model'].append(name)
    metrics['Accuracy'].append(acc * 100)
    metrics['Precision'].append(precision * 100)
    metrics['Recall'].append(recall * 100)
    print(f"Model: {name}")
    print(f"Accuracy: {acc*100}")
    print(f"Precision: {precision*100}")
    print(f"F1 Score: {f1*100}")
    print(f"Recall: {recall*100}")
    print("-" * 30)
```

6. 결과

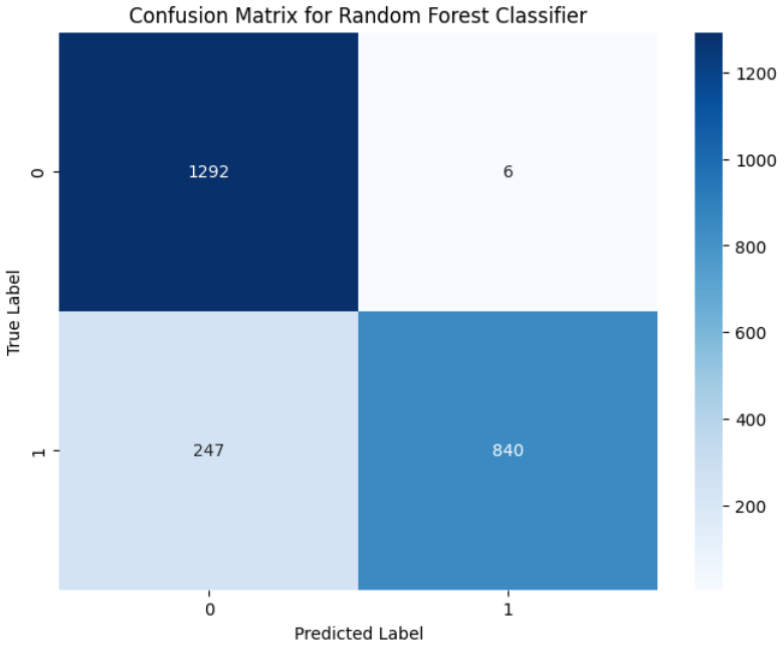
가. 모델 성능 분석

- **Random Forest** 가 가장 높은 정확도와 균형 잡힌 Precision/Recall 성능을 보임
    - Logistic Regression 및 Decision Tree 도 일정 수준의 성능을 보이나 데이터의 복잡도, 피처 간 상호작용 측면에서 Random Forest 가 더 유리함

나. 코드

In [16]:

```
accuracy_results={}
for name,clf in models.items():
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    accuracy=accuracy_score(y_test,y_pred)
    accuracy_results[name]=accuracy
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',xticklabels=np.unique(y), yticklabels=np.un
ique(y))
    plt.title(f'Confusion Matrix for {name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()
```



In [17]:

```
metrics_df = pd.DataFrame(metrics)
plt.figure(figsize=(14, 6))
metrics_df.set_index('Model').plot(kind='bar', width=0.8, figsize=(14, 7), color=['skyblue', 'lig
htgreen', 'lightcoral'])
plt.title('Comparison of Model Metrics (Accuracy, Precision, Recall)', fontsize=16)
plt.ylabel('Score (%)', fontsize=14)
plt.xlabel('Model', fontsize=14)
plt.xticks(rotation=45)
plt.legend(title='Metrics', loc='upper left')
plt.tight_layout()
plt.show()
```