AI-Powered IDS Model Training - Documentation

# 1. Overview

This project trains a machine learning-based Intrusion Detection System (IDS) using the UNSW-NB15 dataset. It preprocesses the data, encodes categorical features, scales numeric columns, and trains a Random Forest model to detect malicious and benign traffic.

# 2. Import Libraries and Load Dataset

Essential libraries like pandas, scikit-learn, and joblib are imported to manipulate data, preprocess features, train the model, and save it for future use.

import pandas as pd  
df = pd.read\_csv('/content/UNSW\_NB15\_training-set.csv')

# 3. Encode Categorical Features

The 'proto' column (protocol type) is encoded into numeric values using LabelEncoder to make it machine-readable.

from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
df['proto'] = le.fit\_transform(df['proto'])

# 4. Scale Numerical Features

The 'sbytes', 'dbytes', 'sttl', and 'dttl' columns are standardized using StandardScaler for uniform scaling.

from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
numerical\_columns = ['sbytes', 'dbytes', 'sttl', 'dttl']  
df[numerical\_columns] = scaler.fit\_transform(df[numerical\_columns])

# 5. Train-Test Split and Label Encoding

Splits the dataset into training and testing sets. Categorical columns are label-encoded dynamically. Missing values in labels are handled before training.

from sklearn.model\_selection import train\_test\_split  
X = df.drop('label', axis=1)  
y = df['label']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
from sklearn.ensemble import RandomForestClassifier  
  
model = RandomForestClassifier(n\_estimators=100)  
label\_encoders = {}  
for col in X\_train.columns:  
 if X\_train[col].dtype == 'object':  
 le = LabelEncoder()  
 X\_train[col] = le.fit\_transform(X\_train[col].astype(str))  
 label\_encoders[col] = le  
 X\_test[col] = X\_test[col].astype(str)  
 X\_test[col] = X\_test[col].apply(lambda x: le.transform([x])[0] if x in le.classes\_ else -1)  
  
y\_train = y\_train.fillna(0)  
model.fit(X\_train, y\_train)

# 6. Real-time Feature Simulation and Retraining

Another portion of the code simulates real-time features (like src\_port, dst\_port, flags) and retrains the model with a simplified feature set. The model is saved using joblib.

df = pd.read\_csv('/content/UNSW\_NB15\_training-set.csv')  
df = df[['proto', 'sbytes', 'dbytes', 'label']]  
df['pkt\_len'] = df['sbytes'] + df['dbytes']  
df['src\_port'] = 12345  
df['dst\_port'] = 80  
df['flags'] = 2  
  
df = df[['proto', 'pkt\_len', 'src\_port', 'dst\_port', 'flags', 'label']]  
  
le = LabelEncoder()  
df['proto'] = le.fit\_transform(df['proto'])  
  
X = df.drop('label', axis=1)  
y = df['label']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
model = RandomForestClassifier(n\_estimators=100)  
model.fit(X\_train, y\_train)  
  
from joblib import dump  
dump(model, 'ids\_model.pkl')