```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# --- PAGE CONFIGURATION ---
st.set_page_config(page_title="AI NIDS Dashboard", layout="wide")

# Custom Title and Description
st.title("AI-Powered Network Intrusion Detection System")
st.markdown("""
### Project Overview
This system uses Machine Learning (**Random Forest Algorithm**) to analyze network traffic in real-time.
It classifies traffic into two categories:
* **Benign:** Safe, normal traffic.
* **Malicious:** Potential cyberattacks (DDoS, Port Scan, etc.).
""")

# --- 1. DATA LOADING (Simulation or Real) ---
@st.cache_data
def load_data():
```

```
    """
    Generates a synthetic dataset that mimics network traffic logs (CIC-IDS2017 structure).
    In a real deployment, this would be replaced by pd.read_csv('network_logs.csv').
    """
    np.random.seed(42)
    n_samples = 5000

    # Simulating features common in Network Logs
    data = {
        'Destination_Port': np.random.randint(1, 65535, n_samples),
        'Flow_Duration': np.random.randint(100, 100000, n_samples),
        'Total_Fwd_Packets': np.random.randint(1, 100, n_samples),
        'Packet_Length_Mean': np.random.uniform(10, 1500, n_samples),
        'Active_Mean': np.random.uniform(0, 1000, n_samples),
        'Label': np.random.choice([0, 1], size=n_samples, p=[0.7, 0.3]) # 0=Safe, 1=Attack
    }

    df = pd.DataFrame(data)

    # Introduce patterns for the AI to learn
    # E.g., Attacks (Label 1) usually have very high packet counts or very short duration
    df.loc[df['Label'] == 1, 'Total_Fwd_Packets'] += np.random.randint(50, 200,
size=df[df['Label']==1].shape[0])
    df.loc[df['Label'] == 1, 'Flow_Duration'] = np.random.randint(1, 1000,
size=df[df['Label']==1].shape[0])

    return df
```

```
# Load Data
df = load_data()

# Sidebar Controls
st.sidebar.header("Control Panel")
st.sidebar.info("Adjust model parameters here.")
split_size = st.sidebar.slider("Training Data Size (%)", 50, 90, 80)
n_estimators = st.sidebar.slider("Number of Trees (Random Forest)", 10, 200, 100)

# --- 2. PREPROCESSING & 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=(100-split_size)/100,
random_state=42)

# --- 3. MODEL TRAINING ---
st.divider()
col_train, col_metrics = st.columns([1, 2])

with col_train:
    st.subheader("1. Model Training")
    if st.button("Train Model Now"):
        with st.spinner("Training Random Forest Classifier..."):
            # Initialize and train the model
```

```
            model = RandomForestClassifier(n_estimators=n_estimators)
            model.fit(X_train, y_train)
            st.session_state['model'] = model # Save model to session
            st.success("Training Complete!")

    if 'model' in st.session_state:
        st.success("Model is Ready for Testing")

# --- 4. EVALUATION METRICS ---
with col_metrics:
    st.subheader("2. Performance Metrics")
    if 'model' in st.session_state:
        model = st.session_state['model']
        y_pred = model.predict(X_test)
        acc = accuracy_score(y_test, y_pred)

        m1, m2, m3 = st.columns(3)
        m1.metric("Accuracy", f"{acc*100:.2f}%")
        m2.metric("Total Samples", len(df))
        m3.metric("Detected Threats", np.sum(y_pred))

        # Visualization: Confusion Matrix
        st.write("### Confusion Matrix")
        cm = confusion_matrix(y_test, y_pred)
        fig, ax = plt.subplots(figsize=(4, 2))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', ax=ax)
```

```
        st.pyplot(fig)
    else:
        st.warning("Please train the model first.")

# --- 5. LIVE ATTACK SIMULATOR ---
st.divider()
st.subheader("3. Live Traffic Simulator (Test the AI)")

st.write("Enter network packet details below to see if the AI flags it as an attack.")

c1, c2, c3, c4 = st.columns(4)
p_dur = c1.number_input("Flow Duration (ms)", 0, 100000, 500)
p_pkts = c2.number_input("Total Packets", 0, 500, 100)
p_len = c3.number_input("Packet Length Mean", 0, 1500, 500)
p_active = c4.number_input("Active Mean Time", 0, 1000, 50)

if st.button("Analyze Packet"):
    if 'model' in st.session_state:
        model = st.session_state['model']
        # Create input array matching the training features
        # [Destination_Port (Random), Flow_Duration, Total_Fwd_Packets, Packet_Length_Mean, Active_Mean]
        input_data = np.array([[80, p_dur, p_pkts, p_len, p_active]])

        pred = model.predict(input_data)
```

```
    if pred[0] == 1:
        st.error("ALERT: MALICIOUS TRAFFIC DETECTED!")
        st.write("**Reason:** High packet count with low duration is suspicious.")
    else:
        st.success("Traffic Status: BENIGN (Safe)")
else:
    st.error("Please train the model first!")
```