Static Data Sources (for Training & Testing)

Dataset Name	Source / Access	Type of Data	Usage in Project
CIC-IDS2017	Canadian Institute for Cybersecurity (available on Kaggle)	Realistic labeled network traffic — DoS, DDoS, brute-force, botnet, normal	Used to train and test ML model for attack classification
UNSW-NB15	University of New South Wales (available on Kaggle / UNSW site)	Modern synthetic + real network data — includes 9 attack types	Used to evaluate model accuracy and generalization on new unseen attack patterns

➡️ **Interface Usage:**

- User uploads dataset files (`.csv`) from local storage.
- System preprocesses (normalization, encoding, splitting).
- ML model is trained using this static data.

🌐 Live Data Sources (for Real-Time Prediction)

Feed / API Name	Access Method	Type of Data	Usage in Project
AlienVault OTX API	Free REST API (requires OTX API key)	Real-time Indicators of Compromise (IOCs): IPs, domains, malware hashes, attack campaigns	Interface fetches recent threats and passes them through the trained model for **real-time prediction**
AbuseIPDB API	Free REST API (requires API key)	Live reports of **malicious or abusive IPs** (spamming, brute force, DoS)	Interface checks live IPs against model to **predict threat score or category**

➡️ **Interface Usage:**

- "Fetch Live Data" button retrieves current threat intel from APIs.
- Live data is converted into the same format as training data.
- Trained ML model predicts whether the new data indicates a **malicious threat** or **normal activity**.

🔄 Data Flow Explanation

1. **Static Data Flow**
    - User uploads datasets (CSV) → stored in backend → preprocessing → model training → saves trained model file (`model.pkl`).
2. **Live Data Flow**
    - APIs fetch live feeds (using API keys) → data parsed and normalized → passed to trained model → predicted threat class (Normal / Suspicious / Malicious) → displayed on dashboard.

Classical Machine Learning Models

Model	Description	Advantages	Limitations
Logistic Regression	Linear model used for binary classification tasks such as predicting whether a network connection is normal or malicious.	Simple, interpretable, and fast to train.	Ineffective for complex nonlinear data.
Random Forest	Ensemble of multiple decision trees for improved classification accuracy.	Handles noisy and imbalanced data well; robust to overfitting.	Slower for large-scale data; less interpretable.
XGBoost	Gradient boosting model that builds trees sequentially to improve accuracy.	High predictive performance and scalability; handles imbalance efficiently.	Requires fine-tuning of parameters.
Support Vector Machine (SVM)	Finds optimal hyperplane to separate classes in multidimensional space.	Effective in high-dimensional datasets.	Computationally expensive on large datasets.
Naïve Bayes	Probabilistic model based on Bayes' theorem.	Efficient for spam/phishing detection; easy to implement.	Assumes feature independence, which may not hold in network data.

Deep Learning Models

Model	Description	Advantages	Limitations
LSTM (Long Short-Term Memory)	Recurrent Neural Network variant that learns temporal dependencies in sequential data such as packet flow logs.	Excellent at modeling time-based attack patterns.	High computational cost; requires large datasets.
CNN (Convolutional Neural Network)	Learns spatial patterns; effective in detecting localized features in structured or visualized traffic data.	Detects correlations in network behavior; high accuracy.	Needs careful preprocessing and reshaping.
Autoencoder	Unsupervised neural network that learns normal patterns for anomaly detection.	Effective in identifying unknown or rare threats.	Prone to false positives without tuning.
Deep Neural Network (DNN)	Fully connected network for general classification tasks.	Captures complex feature relationships.	Risk of overfitting without regularization.

Hybrid / Ensemble Models

To enhance performance, **hybrid** and **ensemble** models are considered:

- **Stacked Ensemble:** Combines Random Forest and XGBoost for higher accuracy.
- **Autoencoder + Classifier:** Autoencoder detects anomalies, and the classifier labels them.
- **CNN + LSTM:** Captures both spatial and temporal threat features in network traffic.

Model Architecture: CNN–BiLSTM Hybrid

This hybrid architecture combines **Convolutional Neural Networks (CNNs)** and **Bidirectional Long Short-Term Memory (BiLSTM)** networks.
The goal is to leverage both **spatial feature extraction** (by CNN) and **temporal/sequential learning** (by BiLSTM) for effective attack classification.

1️⃣ Input Layer

- **Input:** Normalized feature vector from preprocessed dataset (e.g., 40–80 network traffic features).
- **Purpose:** Ensure all numerical values are on a similar scale (typically between 0 and 1) to help the model converge faster and prevent dominance by large-scale features.

2️⃣ Convolutional Layers (CNN)

- **Purpose:** Detect local feature patterns and correlations within the input traffic data.
  For example, it can capture how combinations of network features (like packet rate, flow duration, etc.) relate to malicious activity.
- **Operation:**
  - 1D Convolutions slide filters across the feature dimension.
    Learn spatial dependencies between neighboring features.
- **Typical setup:**
  - Filters: 64 or 128
  - Kernel size: 3–5
  - Activation: ReLU
  - Pooling: MaxPooling1D to reduce dimensionality and noise.

3️⃣ BiLSTM Layers

- **Purpose:** Capture **temporal dependencies** — how features interact over time or sequential patterns in network traffic flows.

- **Bidirectional LSTM** means it processes the sequence **forward and backward**, learning context from both past and future data points.
- **Benefit:** Ideal for intrusion detection where attacks may have a time-based signature or evolving behavior pattern.
- **Typical setup:**
  - Units: 64 or 128
  - Dropout: 0.4 (to prevent overfitting)

- Activation: tanh or sigmoid (internally handled by LSTM)

4️⃣ Dense (Fully Connected) Layer

- **Purpose:** Combines extracted features from CNN–BiLSTM layers to form a high-level, nonlinear representation
- **Operation:**
  - Applies learned weights to integrate all features.
  - Uses **ReLU** activation for nonlinearity.
- **Example:**

```
Dense(64, activation='relu')
```

5️⃣ Output Layer

- **Purpose:** Classify the network traffic into multiple attack categories (e.g., DoS, DDoS, PortScan, Normal, etc.).
- **Activation: Softmax**, which outputs probability distribution across all classes.
- **Example:**

```
Dense(num_classes, activation='softmax').
```