**📊 Current Sensor Data Generation Logic Explained**

Here's exactly how each sensor value is being calculated in your system:

**🔊 1. Spectral Frequency (S1)**

spectral\_freq = 700 + 50 \* np.sin(timestep / 5000) + random.uniform(-5, 5)

**Logic Breakdown:**

* **Base Frequency**: 700 Hz (typical acoustic range)
* **Sinusoidal Wave**: 50 \* sin(timestep/5000) creates a ±50 Hz oscillation
* **Time-based**: timestep/5000 makes it change slowly over time
* **Random Noise**: ±5 Hz for realistic sensor variability
* **Result**: Values typically range from ~645-755 Hz

**Example of the pattern**: 660.23 = 700 + 50\*sin(t/5000) + noise

**📈 2. Kurtosis (K1)**

signal = np.random.normal(0, 1, 1000)

kurtosis = np.mean((signal - np.mean(signal))\*\*4) / (np.var(signal)\*\*2)

**Logic Breakdown:**

* **Generates 1000 samples** from normal distribution (mean=0, std=1)
* **Calculates 4th moment**: (signal - mean)⁴
* **Normalizes by variance²**: Creates standardized kurtosis measure
* **Typical range**: 2.5-3.5 (normal distribution = 3.0)
* **Measures "tailedness"**: Higher = more outliers/extremes

**📉 3. Skewness (SK1)**

skewness = np.mean((signal - np.mean(signal))\*\*3) / (np.std(signal)\*\*3)

**Logic Breakdown:**

* **Uses same 1000 samples** as kurtosis calculation
* **Calculates 3rd moment**: (signal - mean)³
* **Normalizes by std³**: Creates standardized skewness measure
* **Typical range**: -0.5 to +0.5 (normal distribution = 0.0)
* **Measures asymmetry**: Positive = right tail, Negative = left tail

**⚡ 4. RMS Power (RMS1)**

rms\_power = np.sqrt(np.mean(signal\*\*2))

**Logic Breakdown:**

* **Root Mean Square** of the same 1000-sample signal
* **Formula**: √(mean(signal²))
* **Measures signal magnitude/energy**
* **Typical range**: 0.8-1.2 (for unit variance normal signal)
* **Physical meaning**: Effective voltage/current equivalent

**Example**: 1.014 = √(mean of squared values from normal distribution)

**🎯 5. ML Performance Scores**

**Accuracy**: 0.90 + random.uniform(-0.02, 0.02)

* **Base**: 90% accuracy
* **Range**: 88-92%
* **Simulates**: Overall model correctness

**Precision**: 0.85 + random.uniform(-0.03, 0.03)

* **Base**: 85% precision
* **Range**: 82-88%
* **Simulates**: True positives / (True positives + False positives)

**Recall**: 0.88 + random.uniform(-0.02, 0.02)

* **Base**: 88% recall
* **Range**: 86-90%
* **Simulates**: True positives / (True positives + False negatives)

**AUC**: 0.91 + random.uniform(-0.03, 0.02)

* **Base**: 91% AUC
* **Range**: 88-93%
* **Simulates**: Area under ROC curve (discrimination ability)

**🔄 Key Design Patterns:**

1. **Time-based Variation**: Spectral frequency uses timestep for consistent wave patterns
2. **Statistical Foundation**: Kurtosis/skewness use proper statistical formulas
3. **Realistic Ranges**: All values stay within physically meaningful bounds
4. **Controlled Randomness**: Small random components for realism without chaos
5. **Signal Correlation**: Kurtosis, skewness, and RMS power all derive from the same base signal

**🎛️ Leak Impact Simulation:**

When you trigger a leak, these sensors respond differently:

* **Spectral Frequency**: May show frequency shifts due to acoustic changes
* **Kurtosis/Skewness**: Distribution changes as signal characteristics alter
* **RMS Power**: Energy levels change with leak severity
* **ML Scores**: Could eventually connect to actual model performance

This synthetic data generation provides realistic sensor behavior that you can later replace with actual sensor hardware readings or live ML model outputs! 🚀