Certainly! Here's a clear explanation of how your **supervised labeled data** is created from sample.csv, the library involved, which fields are used, and how the process works — plus a documentation-ready summary.

**How Supervised Data is Created from sample.csv**

**1. Library Used for Labeling:**

**sklearn.ensemble.IsolationForest**

* The **Isolation Forest** algorithm is an **unsupervised anomaly detection** method.
* It is used here to **generate labels** (normal vs anomaly) for the originally unlabeled sample.csv.
* This is a common approach when labeled data is not available: use an unsupervised method to generate initial labels, then train a supervised model.

**2. How Isolation Forest Works (Briefly):**

* It isolates observations by randomly selecting a feature and then randomly selecting a split value between the max and min of the selected feature.
* Anomalies are “few and different,” so they tend to have shorter path lengths in these random trees.
* By setting a **contamination** rate (percentage expected to be anomalies), the algorithm predicts which points are anomalies (-1) and which are normal (1).

**3. Fields Measured and Used for Labeling:**

You use the following fields (features) extracted from sample.csv as input to Isolation Forest:

| **Feature** | **Description** |
| --- | --- |
| Flow\_Duration | Time duration of a network flow (ms or sec) |
| Total\_Fwd\_Packets | Number of forward packets in the flow |
| Total\_Backward\_Packets | Number of backward packets |
| Fwd\_Packet\_Length\_Max | Maximum size of forward packets |
| Bwd\_Packet\_Length\_Max | Maximum size of backward packets |
| Flow\_Bytes\_s | Number of bytes per second |
| Flow\_Packets\_s | Number of packets per second |
| Fwd\_IAT\_Mean | Mean inter-arrival time of forward packets |
| Bwd\_IAT\_Mean | Mean inter-arrival time of backward packets |
| Packet\_Length\_Variance | Variance of packet lengths |
| Average\_Packet\_Size | Average packet size |

**4. Example of How Labeling Works:**

Suppose you have a flow with these feature values:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Feature** | **Value** | | --- | --- | | Flow\_Duration | 1000 | | Total\_Fwd\_Packets | 50 | | Total\_Backward\_Packets | 48 | | Fwd\_Packet\_Length\_Max | 1500 | | Bwd\_Packet\_Length\_Max | 1400 | | Flow\_Bytes\_s | 2000 | | Flow\_Packets\_s | 100 | | Fwd\_IAT\_Mean | 15 | | Bwd\_IAT\_Mean | 15 | | Packet\_Length\_Variance | 250 | | Average\_Packet\_Size | 500 | |

* The Isolation Forest algorithm looks at this flow in comparison to others.
* If these values fit **normal traffic patterns**, it will label the flow as 1 (normal).
* If any feature is an extreme outlier (e.g., extremely high bytes per second or unusually short duration with many packets), it may label it as -1 (anomaly).

**5. How It Is Implemented in Code:**

from sklearn.ensemble import IsolationForest

import pandas as pd

# Load data

df = pd.read\_csv('sample.csv')

# Select features and clean data

features = ['Flow\_Duration', 'Total\_Fwd\_Packets', 'Total\_Backward\_Packets', 'Fwd\_Packet\_Length\_Max',

'Bwd\_Packet\_Length\_Max', 'Flow\_Bytes\_s', 'Flow\_Packets\_s', 'Fwd\_IAT\_Mean', 'Bwd\_IAT\_Mean',

'Packet\_Length\_Variance', 'Average\_Packet\_Size']

X = df[features].replace([np.inf, -np.inf], np.nan).dropna()

# Initialize Isolation Forest

iso\_forest = IsolationForest(contamination=0.05, random\_state=42)

# Fit and predict labels (-1 anomaly, 1 normal)

labels = iso\_forest.fit\_predict(X)

# Map labels to strings

label\_map = {1: 'normal', -1: 'anomaly'}

df['class'] = pd.Series(labels, index=X.index).map(label\_map)

# Save labeled data

df.to\_csv('classified\_traffic\_full.csv', index=False)

**Documentation Summary**

**Creating Supervised Labeled Data for Anomaly Detection**

In the absence of pre-labeled network traffic data, this project utilizes the **Isolation Forest** algorithm from the **scikit-learn** library to label sample.csv data automatically.

**Why Isolation Forest?**

* An efficient unsupervised anomaly detection method that isolates outliers without needing labeled examples.
* It detects anomalous traffic based on unusual patterns in statistical network features.

**Features Used for Labeling**

* Time-based, packet count, packet size, inter-arrival times, and statistical variance of packet sizes.

**Labeling Process**

1. **Data Cleaning:** Remove rows with infinite or missing values.
2. **Isolation Forest Training:** Fit the model on the cleaned features.
3. **Prediction:** Assign 'normal' or 'anomaly' labels based on the Isolation Forest output.
4. **Saving Results:** Labeled data is saved as classified\_traffic\_full.csv.

**Example**

An observed flow with an unusually high number of bytes per second and very low duration is more likely to be labeled **anomaly** compared to stable, long-duration flows.

Sure! Here's a brief explanation of **how Isolation Forest works for unsupervised labeling**, along with a simple visual idea you can imagine or plot.

**How Isolation Forest Works for Unsupervised Labeling**

**Core Idea:**

* **Isolation Forest (IF)** isolates anomalies instead of profiling normal data.
* It builds many random decision trees by randomly selecting features and split values.
* **Anomalies are isolated faster** because they differ significantly from normal data points.
* Normal points need more splits to isolate because they lie in dense regions.

**Process:**

1. **Random Splitting**  
   The algorithm randomly picks a feature and splits the data by a random split value between min and max of that feature.
2. **Tree Construction**  
   This splitting repeats recursively, building many trees (forest).
3. **Path Length**  
   The number of splits (path length) required to isolate a data point is recorded.
   * **Short path length = anomaly** (few splits needed to isolate it)
   * **Long path length = normal** (needs many splits)
4. **Anomaly Score**  
   Average path length over all trees determines anomaly score.  
   Points with scores above a threshold are labeled **normal**, below are **anomalies**.

**Visual Explanation (Conceptual):**

Imagine a 2D scatterplot of network traffic data points on two features, say:

* **Feature X:** Packet Size
* **Feature Y:** Flow Duration

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| **Plot** |  |  |

* The **black dots** are dense, normal traffic points.
* The **white dots** are sparse, anomalous points far away.

**How IF isolates:**

* Random splits quickly separate red points (fewer splits → short paths).
* Many splits needed to separate blue cluster points (long paths).

**Simple Code to Visualize 2D Example (Optional)**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import IsolationForest

# Generate synthetic data

rng = np.random.RandomState(42)

X\_normal = 0.3 \* rng.randn(100, 2)

X\_normal = np.r\_[X\_normal + 2, X\_normal - 2] # Two clusters

X\_outliers = rng.uniform(low=-4, high=4, size=(20, 2))

X = np.r\_[X\_normal, X\_outliers]

# Fit Isolation Forest

clf = IsolationForest(contamination=0.1, random\_state=42)

clf.fit(X)

y\_pred = clf.predict(X)

# Plot

plt.scatter(X[y\_pred == 1][:, 0], X[y\_pred == 1][:, 1], c='blue', label='Normal')

plt.scatter(X[y\_pred == -1][:, 0], X[y\_pred == -1][:, 1], c='red', label='Anomaly')

plt.legend()

plt.title("Isolation Forest Detection")

plt.xlabel("Feature X (Packet Size)")

plt.ylabel("Feature Y (Flow Duration)")

plt.show()

**Summary**

| **Step** | **What happens** |
| --- | --- |
| Build trees with random splits | Separate points quickly or slowly based on data density |
| Measure path length | Shorter path = more likely anomaly |
| Score and label points | Threshold splits data into normal/anomaly |

Let me know if you'd like me to create this visual plot with your real data features or embed this explanation in a document!