**Understanding Data Set and Building The Model**

The synth.FriedmanDrift dataset from the river library generates a synthetic dataset with concept drifts designed for evaluating machine learning models and drift detection algorithms. Here's an in-depth look at its parameters, how to generate the data, and how to use it for training and detecting concept drift:

**Description**

**Features**

* **Number of features:** 10
* **Feature values:** Sampled uniformly in [0, 1]
* **Relevant features:** Only the first 5 features are relevant to the target.

**Target**

* **Function definition:** The target is defined by different functions depending on the type of the drift.

**Parameters**

**drift\_type**

* **Type:** str
* **Default:** 'lea'
* **Options:**
  + 'lea': Local Expanding Abrupt drift
    - Concept drift appears in two distinct regions of the instance space.
    - Three points of abrupt change in the training dataset.
    - Regions of drift expand at every consecutive change.
  + 'gra': Global Recurring Abrupt drift
    - Concept drift appears over the whole instance space.
    - Two points of concept drift, with the old concept reoccurring at the second point.
  + 'gsg': Global and Slow Gradual drift
    - Concept drift affects the entire instance space gradually.
    - Two change points with a transition window where old and new concepts are generated with equal probability.

**position**

* **Type:** tuple[int, ...]
* **Default:** (50000, 100000, 150000)
* **Description:** The number of monitored instances after which each concept drift occurs. Must have at least two elements, with each number greater than the preceding one. If drift\_type='lea', the tuple must have three elements.

**transition\_window**

* **Type:** int
* **Default:** 10000
* **Description:** Length of the transition window between two concepts, applicable only when drift\_type='gsg'. If set to zero, drifts will be abrupt. When transition\_window > 0, it defines a window in which instances of the new concept are gradually introduced among examples from the old concept.

**seed**

* **Type:** int | None
* **Default:** None
* **Description:** Random seed number used for reproducibility.

**Attributes**

**desc**

* Returns the description from the docstring.

**Drift Detection Method (DDM) Overview**

DDM monitors the error rate of the model and uses statistical methods to detect significant deviations. Here’s a simplified version of the process:

1. **Initial Phase:**
   * The model is assumed to be in a stable phase initially.
   * The error rate and its standard deviation are monitored and updated continuously as new data points are processed.
2. **Monitoring Phase:**
   * For each new data point, the prediction is checked for correctness.
   * The error rate (proportion of incorrect predictions) is updated.
3. **Detection Phase:**
   * DDM calculates the mean error rate (p\_t) and the standard deviation of the error rate (s\_t) over time.
   * It uses these statistics to determine if the error rate is within acceptable bounds or if a drift is occurring.
4. **Drift Detection:**
   * DDM defines two thresholds:
     + **Warning Level:** A less strict threshold indicating a potential drift.
     + **Drift Level:** A stricter threshold indicating a confirmed drift.
   * If the monitored error rate exceeds the warning level, DDM raises a warning.
   * If the monitored error rate exceeds the drift level, DDM signals that a drift has occurred.

### **Problem:** **Simulate Data Stream and Check for Drift**[**¶**](http://localhost:8888/notebooks/Untitled.ipynb#Simulate-Data-Stream-and-Check-for-Drift)

# Iterate through the dataset and update the DDM detector

for i, (x, y) in enumerate(dataset):

sample = pd.DataFrame([x.values()]) # Convert sample to DataFrame

true\_label = y

predicted\_label = model.predict(sample)[0]

# Update DDM with the model's prediction

dd.update(value=int(predicted\_label != true\_label)) # Update with 1 if incorrect, 0 if correct

# Check for drift

if dd.drift:

print(f'Change detected at index {i}')

break # Stop if a drift is detected

elif dd.warning:

print(f'Warning detected at index {i}')

**Took so long to run:**

Optimizing the iteration through the dataset and updating the DDM detector can be approached in several ways. Here are a few optimization strategies:

1. **Batch Processing:** Instead of processing one sample at a time, process samples in batches.
2. **Avoid Redundant Conversions:** Reduce the number of times you convert data to DataFrame.
3. **Vectorized Operations:** Use vectorized operations where possible to leverage the efficiency of libraries like NumPy and pandas.
4. **Efficient Sampling:** If applicable, sample a subset of data for drift detection instead of using the entire dataset.

### Visualizing Concept Drift

To visualize concept drift, we can plot the target variable over time and mark the positions where concept drift occurs. For simplicity, let's assume we are dealing with Local Expanding Abrupt drift (LEA) with three drift points.

### Step-by-Step Visualization

#### Step 1: Generate and Prepare the Dataset

python

Copy code

from river.datasets import synth

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Create the FriedmanDrift dataset

dataset = synth.FriedmanDrift(drift\_type='lea', position=(50000, 100000, 150000), seed=42)

# Convert the dataset to a DataFrame for easier handling

data = [dict(x, \*\*{'y': y}) for x, y in dataset.take(200000)] # Taking 200,000 samples as an example

df = pd.DataFrame(data)

# Show the first few rows of the DataFrame

print(df.head())

#### Step 2: Visualize the Target Variable Over Time

python

Copy code

# Plot the target variable over time to visualize concept drift

plt.figure(figsize=(14, 7))

plt.plot(df.index, df['y'], label='Target Variable')

plt.axvline(x=50000, color='r', linestyle='--', label='First Drift Point')

plt.axvline(x=100000, color='g', linestyle='--', label='Second Drift Point')

plt.axvline(x=150000, color='b', linestyle='--', label='Third Drift Point')

plt.xlabel('Sample Index')

plt.ylabel('Target Value')

plt.title('Concept Drift in FriedmanDrift Dataset')

plt.legend()

plt.show()

### Explanation:

* **Sample Index**: Each row or record in the dataset represents a sample at a given point in time. With 200,000 samples, each sample index (0 to 199,999) represents a specific time window.
* **Drift Points**: The red, green, and blue vertical lines mark the positions where concept drift occurs (at sample indices 50,000, 100,000, and 150,000).

#### Step 3: Detailed Feature Visualization (Optional)

You can also visualize how individual features change over time, especially the relevant ones (first 5 features):

python

Copy code

# Plot the first 5 features over time to visualize their behavior

plt.figure(figsize=(14, 12))

for i, column in enumerate(df.columns[:5], 1):

plt.subplot(3, 2, i)

plt.plot(df.index, df[column])

plt.axvline(x=50000, color='r', linestyle='--', label='Drift Point' if i == 1 else "")

plt.axvline(x=100000, color='g', linestyle='--', label='Drift Point' if i == 1 else "")

plt.axvline(x=150000, color='b', linestyle='--', label='Drift Point' if i == 1 else "")

plt.title(f'Feature {column} Over Time')

plt.xlabel('Sample Index')

plt.ylabel(f'{column} Value')

plt.legend()

plt.tight\_layout()

plt.show()

### Explanation:

* This set of plots shows how each of the first 5 features changes over time, with vertical lines marking the drift points. This helps in understanding how the features themselves might be impacted by the concept drift.