Final Project Data Stream Processing

M2 Data Science IP Paris 2025-2026

Defense date : 05/01/2026

Submit date : 03/01/2026 before 23h59

>> Update the groups and choose your project in the Excel available HERE 2526_StudentsChoices << **before 29/11/2025 23h59**

You will work in a team of **2 or 3 people during this project**. We propose 3 different projects. You will classify them according to your wish. **We will assign a project to each group and let you know.**

General instructions

- **Define your team member and the list of the 3 projects you would like to work on directly in this file** (2526_StudentsChoices) We will assign you 1 project.
- **Delivrables for defense**
    - Documented source code and Readme (.py, HTML, etc.) in Zip file and available on a **Github**
    - Report describing the project, your contributions, experimental results, etc. (at least 5 pages)
    - PowerPoint slides for oral presentation
- **How to submit the deliverables**
    - The deliverables need to be submitted on Moodle + email.
    - The object of the mail will follow this structure : M2 IPP-Project Delivrables-Group[xx]-[TheProject]-LastName1+LastName2+LastName3
    - Each group should submit only once in the moodle deposit box.
- **Defense (20mn / team)**
    - Presentation (13 mn)
        - Problematic, Method or Models implemented / research papers analysis or benchmark if applicable
        - Experiments results if applicable
        - Conclusion
        - Demo (Proof that the code compile and everything run)
    - Questions and Answers (7 mn)
- For any question please contact mariam.sa.barry@gmail.com and maurras.togbe@isep.fr, sathiyapkr@gmail.com, yanlei.diao@polytechnique.edu

Project 1 : Online Machine Learning with Python using CapyMOA or River (Dr. Mariam BARRY)

In this project, you will implement a streaming algorithm following the guidelines provided by CapyMOA (**(**https://capymoa.org/**)**. A list of research papers is provided below as the basis for your implementation.
Teams that produce high-quality implementations may be invited to contribute their code to the CapyMOA or River project on GitHub. Please note that this contribution is entirely voluntary and will not affect the grading of your project.

Specifics instructions : Report & requirements

Your implementation should be accompanied by a report that reproduces some of the experiments from the original paper, and some tests. While it may not be possible to achieve identical results due to unavailable implementation or experimental details from the original publication, it is crucial to attempt to replicate key experiments and clearly describe your approach.

In addition, you should complement your experiments by comparing the implemented algorithm against those already available in the CapyMOA library. Depending on the task, the number of relevant algorithms may be limited (e.g., semi-supervised learning). However, alternative comparisons are possible, such as evaluating a semi-supervised learning algorithm against a supervised algorithm trained only on labeled data.
The report should also include implementation details, particularly if any adaptations were necessary or algorithmic choices were made that were not explicitly outlined in the original paper.

List of papers (Students can propose their own paper and need to request our approval).

AutoML in River

1. Evolution-Based Online Automated Machine Learning. It is available here.
   Automated Machine Learning (AutoML) deals with finding well-performing machine learning models and their corresponding configurations without the need of machine learning experts. However, if one assumes an online learning scenario, where an AutoML instance executes on evolving data streams, the question for the best model and its configuration with respect to occurring changes in the data distribution remains open. Algorithms developed for online learning settings rely on few and homogeneous models and do not consider data mining pipelines or the adaption of their configuration. We, therefore, introduce EvoAutoML, an evolution-based online learning framework consisting of heterogeneous and connectable models that supports large and diverse configuration spaces and adapts to the online learning scenario. We present experiments with an implementation of EvoAutoML on a diverse set of synthetic and real datasets, and show that our

proposed approach outperforms state-of-the-art online algorithms as well as strong ensemble baselines in a traditional test-then-train evaluation.

2. **Cybersecurity** **CyberCScope: Mining Skewed Tensor Streams and Online Anomaly Detection in Cybersecurity Systems**
   Cybersecurity systems are continuously producing a huge number of time-stamped events in the form of high-order tensors, such as {count; time, port, flow duration, packet size, . . .,} and so how can we detect anomalies/intrusions in real time? How can we identify multiple types of intrusions and capture their characteristic behaviors? The tensor data consists of categorical and continuous attributes and the data distributions of continuous attributes typically exhibit skew. These data properties require handling skewed infinite and finite dimensional spaces simultaneously. In this paper, we propose a novel streaming method, namely CyberCScope. The method effectively decomposes incoming tensors into major trends while explicitly distinguishing between categorical and skewed continuous attributes. To our knowledge, it is the first to compute hybrid skewed infinite and finite dimensional decomposition. Based on this decomposition, it streamingly finds distinct time-evolving patterns, enabling the detection of multiple types of anomalies. Extensive experiments on large-scale real datasets demonstrate that CyberCScope detects various intrusions with higher accuracy than state-of-the-art baselines while providing meaningful summaries for the intrusions that occur in practice.

3. **Adaptive Forest for Classification** https://arxiv.org/abs/2510.22991

   Random Forests (RF) and Extreme Gradient Boosting (XGBoost) are two of the most widely used and highly performing classification and regression models. They aggregate equally weighted CART trees, generated randomly in RF or sequentially in XGBoost. In this paper, we propose Adaptive Forests (AF), a novel approach that adaptively selects the weights of the underlying CART models. AF combines (a) the Optimal Predictive-Policy Trees (OP2T) framework to prescribe tailored, input-dependent unequal weights to trees and (b) Mixed Integer Optimization (MIO) to refine weight candidates dynamically, enhancing overall performance. We demonstrate that AF consistently outperforms RF, XGBoost, and other weighted RF in binary and multi-class classification problems over 20+ real-world datasets.

4. **REGRESSION FINANCE** **An Online Kernel Adaptive Filtering-Based Approach for Mid-Price Prediction**

   The idea of multivariate and online stock price prediction via the kernel adaptive filtering (KAF) paradigm is proposed in this article. The prediction of *stock prices is traditionally done with regression and classification, thereby requiring a large set of batch-oriented and independent training samples. This is problematic considering the nonstationary nature of a financial time series. In this research, we propose an online kernel adaptive filtering-based approach for stock price prediction to overcome this challenge. To examine a stock's performance and demonstrate the work's superiority, we use ten different KAF family of algorithms. In this paper, we take on this challenge and propose an approach for predicting stock prices. To analyze a stock's performance and demonstrate the work's superiority, we use ten distinct KAF algorithms. Besides, the results are analyzed on nine-time windows such as one day, sixty minutes, thirty minutes, twenty five minutes, twenty minutes, fifteen minutes, ten minutes, five minutes, and one minute. We are the first to experiment with several time windows for all fifty stocks on the Indian National Stock Exchange, to the best of our knowledge. It should be noted here that the experiments are performed on stocks making up the main index: Nifty-50. In terms of performance and compared to existing methods, we have a 66% probability of correctly predicting a stock's next upward or downward movement. This number clearly shows the edge that the proposed method has in actual deployment. Furthermore, the experimental findings show that KAF is not only a better option for predicting stock prices but that it may also be used as an alternative in high-frequency trading due to its low latency.*

5. **MemStream: Memory-Based Streaming Anomaly Detection**

Given a stream of entries over time in a multi-dimensional data setting where concept drift is present, how can we detect anomalous activities? Most of the existing unsupervised anomaly detection approaches seek to detect anomalous events in an offline fashion and require a large amount of data for training. This is not practical in real-life scenarios where we receive the data in a streaming manner and do not know the size of the stream beforehand. Thus, we need a data-efficient method that can detect and adapt to changing data trends, or concept drift, in an online manner. In this work, we propose MemStream, a streaming anomaly detection framework, allowing us to detect unusual events as they occur while being resilient to concept drift. We leverage the power of a denoising autoencoder to learn representations and a memory module to learn the dynamically changing trend in data without the need for labels. We prove the optimum memory size required for effective drift handling. Furthermore, MemStream makes use of two architectural design choices to be robust to memory poisoning. Experimental results show the effectiveness of our approach compared to state-of-the-art streaming baselines using 2 synthetic datasets and 11real-world datasets.

Project 2 : Real-time RAG (Retrieval Augmented Generation) Application with Apache Kafka (Dr. Sathiya KUMAR)

A **Real-time Retrieval-Augmented Generation (RAG) system** leverages continuously updated streaming data (not static documents) to generate responses based on the freshest context. This requires a pipeline that constantly ingests, chunks, embeds, and indexes the streaming data into a vector store. Here are a few project ideas using real-time public streaming APIs.

Key Architectural Considerations for these Projects:

To handle the "real-time" aspect, you would need:

- **Streaming Ingestion:** Tools like **Apache Kafka** to handle the high volume of incoming data.
- **Vector Database:** A database optimized for high-speed writes and low-latency retrieval, such as **Chroma Db, Qdrant**, **Weaviate**, or **Pinecone**, which can update the index continuously.
- **Temporal Awareness:** The RAG retrieval step must be heavily weighted by **recency** (timestamp metadata) to ensure the retrieved context is seconds old, not minutes or hours old.

Note : The student is free to choose one of the themes listed below or propose a new theme with their own finding on available public streaming api. The theme proposed by the student is subject to validation.

Theme 1. Real-Time Financial Market News Analyst

This RAG system would provide instant, up-to-the-second analysis and summarization of market-moving events.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Financial News APIs** (e.g., NewsAPI, Twelve Data, Alpha Vantage), filtering for specific stock tickers or sectors.	"What caused the drop in TSLA stock price in the last 10 minutes?"
Knowledge Base	Streaming news articles, SEC filings data, or social media mentions (processed for sentiment).	**Goal:** Provide **just-in-time commentary** and context on why a stock or commodity price is currently moving, retrieving news that is only seconds old.

RAG Function	Retrieve the 3 most recent, highly-relevant news headlines and their snippets; generate a **brief summary of the current market sentiment** for the queried entity.	

Theme 2. Public Transit Incident Responder

This RAG system provides real-time information and alternative routing advice based on live incident data.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Public Transit APIs** (e.g., GTFS Realtime feeds for major cities, MTA, or specific national rail data).	"Is the Northern Line running smoothly, and what's the fastest alternative route from point A to B right now?"
Knowledge Base	Live incident reports, service delays/closures, estimated arrival times (ETAs), and historical alternative routes.	**Goal:** Augment a general route-finding query with **live, unannounced disruption information** to give a contextually accurate answer.
RAG Function	When a query mentions a line or station, the system retrieves any live incident data within the last 5 minutes and uses that context to **re-calculate or suggest** an unaffected route.	

Theme 3. Global Disaster/Event Monitor (Streaming Social Data)

This RAG system tracks and synthesizes verified data from public, streaming sources during rapidly unfolding events.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Social Media Streams** (e.g., filtering a public X/Twitter stream for keywords like "earthquake," "wildfire," "power outage"), **Open Weather APIs** for real-time alerts.	"Summarize the latest confirmed information about the wildfire near region X."

Knowledge Base	Streaming, geo-tagged posts that have been filtered for high information value (e.g., from verified news accounts or official sources), current weather alerts, and official agency updates.	**Goal:** Combat misinformation by providing **LLM-generated summaries** that are *only*augmented by context derived from recently verified, real-time data streams.
RAG Function	Use retrieved streaming posts/alerts to generate an answer, including a **timestamp of the latest event** used for retrieval (e.g., "As of 17:05 UTC, authorities confirm...").	

Theme 4. Real-Time Severe Weather Advisory System (Weather API)

This RAG system would provide dynamic, localized weather summaries and actionable advice based on rapidly changing meteorological conditions.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Weather Alerts/Feeds** (e.g., NOAA/NWS API, OpenWeatherMap One Call API). Data includes severe weather warnings, radar updates, and minute-by-minute forecasts.	"What is the likelihood of a flash flood in my neighborhood in the next 30 minutes, and what road closures are reported?"
Knowledge Base	Geospatial data on current warnings, precipitation rates, wind speeds, and real-time reports of localized impact (e.g., power outages, minor flooding).	**Goal:** Augment a general weather query with **specific, time-sensitive risks** tailored to the user's location, moving beyond a simple forecast.
RAG Function	Retrieve any weather warnings or radar data posted within the last 5 minutes for the user's geocode, and use that context to generate a summary of the immediate hazard and required preparation.	

Theme 5. Live Sports Play-by-Play Analyst (Sports Commentary API)

This RAG system offers instant, contextual commentary on sporting events, answering questions about *what just happened* and why it matters to the game state.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Live Sports Data Feeds** (e.g., ESPN live updates, dedicated sports API services). Data includes play-by-play text, score changes, player stats, and foul/penalty events.	"Why did the referee stop play just now? What's the impact on team A's possession?"
Knowledge Base	Event logs (the last 10 plays), real-time game statistics (shot clocks, time remaining), and commentary text, indexed by game time.	**Goal:** Provide detailed, narrative answers using the most recent commentary and data, simulating an expert analyst.
RAG Function	Retrieve the text commentary and game log entries from the exact moment of the query (e.g., the last 30 seconds of play). The LLM generates a **contextual explanation** of the event, its rule basis, and the current score/time situation.	

Theme 6. Cyber Threat Intelligence Reporter (Cyber Threat API)

This RAG system delivers immediate, synthesized intelligence on emerging digital threats and vulnerabilities as they are detected globally.

Component	**Streaming API / Data Source**	**Query Example**
Ingestion	**Threat Intelligence Feeds** (e.g., MISP feeds, VirusTotal streams, or open-source threat feeds like AlienVault OTX). Data includes new malware hashes, identified C2 servers, and critical vulnerability announcements.	"Give me a summary of the new critical vulnerability being exploited via phishing this morning, including IoCs."

Knowledge Base	Freshly identified Indicators of Compromise (IoCs), exploit details, mitigation steps, and real-time reports of active attacks, all indexed by severity and time.	**Goal:** Immediately synthesize raw threat data into **actionable summaries**for security analysts, speeding up incident response.
RAG Function	Retrieve IoCs and description snippets that were published in the last 60 minutes matching the query keywords. The LLM generates a threat brief, including a recommended **patch/mitigation priority**.	

Theme 7. Local Government Activity Tracker (Government/Civil Activity API)

This RAG system provides citizen-focused summaries of local government decisions, planning changes, and public feedback as they occur.

Component	Streaming API / Data Source	Query Example
Ingestion	**Local Government Open Data Portals** (e.g., city council meeting live feeds, planning application status updates, public comment streams).	"What were the key changes approved in last night's City Council meeting regarding the downtown development project?"
Knowledge Base	Transcript snippets from live meetings, official document metadata (as soon as a document is published), and public sentiment data from streamed civil activity.	**Goal:** Simplify complex governmental procedures and **provide accessible summaries** of the most recent decisions affecting local citizens.
RAG Function	Retrieve the transcript segments or decision announcements from the last 24 hours related to the queried topic. The LLM generates a **plain-language summary** of the outcome and its expected timeline.	

Theme 8. Movie Review (Reddit/Youtube/IMDB API)

This RAG system provides real-time analysis and insightful summaries of public opinion regarding recently released movies and ongoing films.

Component	Streaming API / Data Source	Query Example
Ingestion	Public Streaming API (e.g., Twitter/Reddit feed filtered by movie keywords, or a simulated stream of reviews from a dynamic source).	"What is the consensus on the new action movie 'Stellar Siege' based on reviews from the last 2 hours, and what did people say about the lead actor's performance?"
Knowledge Base	A continuously updated vector database of review embeddings, indexed primarily by timestamp and movie title, capturing the very latest public sentiment and critical details.	**Goal:** Augment a general inquiry about a movie with **live, un-indexed opinion trends** to provide a contextually accurate and time-sensitive summary of public reception.
RAG Function	When a query mentions a movie title, the system retrieves the most relevant and *most recent* 50 reviews (e.g., within the last 24 hours) from the vector database and uses that context to **generate an updated summary, pinpoint key themes (e.g., cinematography, plot), and extract specific answers.**	

-

Project 3 : Dynamic Graph Anomaly Detection Implementation Project: SAD Method (Prof. Yanlei DIAO)

Context

Dynamic graphs are ubiquitous in real-world applications, from social networks and financial transactions to cybersecurity and communication systems. Unlike static graphs, dynamic graphs evolve over time with nodes and edges appearing, disappearing, or changing their properties. Anomaly detection in such temporal graph structures presents unique challenges, requiring methods that can capture both structural patterns and temporal dynamics while dealing with limited labeled data.

You will implement the SAD (Semi-Supervised Anomaly Detection on Dynamic Graphs) method and integrate it into our existing dynamic graph anomaly detection framework. This framework provides standardized interfaces for data loading, model training, and evaluation across multiple benchmark datasets. The SAD method is particularly interesting as it addresses the challenge of limited labeled data in dynamic graph anomaly detection through a novel combination of temporal graph neural networks, memory-based reference distribution modeling, and pseudo-label contrastive learning.

Implementation Requirements

Core Implementation

You must implement the complete SAD algorithm as described in the paper, including:

- Temporal Graph Encoder: TGAT-based encoder for processing dynamic graphs with continuous-time message passing
- Anomaly Detector Network: MLP that maps node embeddings to scalar anomaly scores
- Time-Equipped Memory Bank: FIFO queue storing past anomaly scores with time-decay weighting for reference distribution
- Deviation Loss: Semi-supervised loss that pulls normal nodes toward reference distribution and pushes anomalous nodes away
- Pseudo-Label Contrastive Learning: Supervised contrastive learning using deviation scores to create soft groups
- Framework Integration: Your method must integrate with the framework that you used in your previous lab work

Dataset Requirements

You must evaluate your implementation on three specific datasets:

- email-dnc: Email communication network from the DNC
- bitcoin-otc: Bitcoin OTC trust network with temporal trust relationships
- wiki: Wikipedia temporal graph dataset with article linking patterns

Technical Specifications

- Programming Language: Python 3.11+
- Deep Learning Framework: PyTorch with PyTorch Geometric
- Dependencies: All requirements are specified in the provided pyproject.toml file
- Platform Compatibility: The code only works on macOS and Linux. Windows users must use WSL (Windows Subsystem for Linux)
- Code Quality: Well-documented, modular code following Python best practices
- Framework Integration: Your implementation must use the same integration patterns from your previous lab work

Report Requirements

Your implementation must be accompanied by a comprehensive report that includes the following sections:

1. SAD Method Presentation

- Problem Formulation: Clear explanation of semi-supervised anomaly detection on dynamic graphs
- Method Overview: Detailed description of the SAD algorithm and its key components
- Technical Details: Mathematical formulation of the deviation loss, memory bank mechanism, and contrastive learning approach
- Innovation Analysis: What makes SAD innovative compared to existing dynamic graph anomaly detection methods? Discuss the key contributions and novel aspects

2. Experimental Setup

- Dataset Preparation: How you processed and prepared the three required datasets (email-dnc, bitcoin-otc, wiki)
- Hyperparameter Configuration: Your hyperparameter choices and tuning strategy
- Training Procedure: Detailed description of your training setup, including batch sizes, learning rates, and training duration
- Evaluation Protocol: How you evaluated the method and what metrics you used

3. Results and Discussion

- Performance Results: Present your experimental results clearly with appropriate visualizations
- Baseline Comparisons: Compare your SAD implementation against existing methods from your previous lab or other baselines available in the framework
- Cross-Dataset Analysis: Discuss how SAD performs across the three different datasets and what this reveals about the method
- Limitations and Challenges: Honest discussion of limitations you encountered, implementation challenges, and areas where the method could be improved
- Insights: What did you learn about semi-supervised learning on dynamic graphs? What worked well and what didn't?

The report should demonstrate both your understanding of the SAD method and your ability to critically analyze its performance and limitations in practice.

References:

[**Sad paper**]: Tian, S., Dong, J., Li, J., Zhao, W., Xu, X., Song, B., ... & Chen, L. (2023). Sad: Semi-supervised anomaly detection on dynamic graphs. arXiv preprint arXiv:2305.13573.
https://arxiv.org/pdf/2305.13573