Streaming Machine Learning: Comparative Analysis of River and Capymoa Frameworks

MADANI ALAOUI Yassine, LAITA Sami

October 25, 2025

Abstract

This report presents a comprehensive comparative analysis of two prominent streaming machine learning frameworks: River and Capymoa. The study evaluates multiple classification algorithms across various real-world datasets, assessing performance through metrics including Accuracy, Precision, Recall, F1-Score, MCC, Cohen's Kappa, and ROC AUC. The analysis reveals distinct characteristics and performance patterns for each framework, providing insights for practical streaming ML applications.

1 Introduction

Streaming machine learning has become increasingly important in the era of real-time data processing. This report examines two Python frameworks designed for online learning: River and Capymoa. Both frameworks implement algorithms that can learn from data streams incrementally, without requiring the entire dataset to be loaded into memory.

2 Part 1: River Framework Analysis

2.1 River Framework Overview

River is a Python library for online machine learning. It specializes in incremental learning from data streams, supporting classification, regression, clustering, and preprocessing tasks. River's architecture is designed for continuous model updates with limited memory footprint.

2.2 Datasets Used

2.2.1 Elec2 Dataset

The Elec2 (Electricity) dataset contains 45,312 instances of electricity market data from the Australian New South Wales Electricity Market. It is a binary classification task predicting price changes (UP or DOWN) based on electricity demand, supply, and time-based features. The dataset includes the following features: `['date', 'day', 'period', 'nswprice', 'nswdemand', 'vicprice', 'vicdemand', 'transfer']`.

2.2.2 Phishing Dataset

The Phishing dataset consists of 11,055 instances used to classify websites as either legitimate or phishing, based on various URL and webpage characteristics. It includes features such as `url-length`, `https`, `ip-in-url`, `popup-window`, and `age-of-domain`, which capture key properties related to website security, domain reputation, and suspicious URL patterns.

2.2.3 Bananas Dataset

The Bananas dataset is an artificial dataset with 5,300 instances featuring 2D data points forming banana-shaped clusters. It serves as a benchmark for non-linear classification algorithms. The dataset contains two numerical features, `x1` and `x2`, which represent the coordinates of the data points in a two-dimensional space.

2.3 Supervised Learning On dataset ELEC2

Logistic Regression A linear model for binary classification that uses sigmoid function to predict probabilities. River's implementation supports online gradient descent updates.

Adaptive Random Forest (ARF) An ensemble method that combines multiple Hoeffding Trees with adaptive mechanisms for concept drift detection and handling.

Results :

Table 1: Performance metrics comparison of models

Model	Accuracy	Precision	Recall	F1	MCC	Kappa	ROC AUC
Logistic Regression	0.6090	0.4857	0.0611	0.1085	0.0447	0.0235	0.5475
Adaptive Random Forest	0.6052	0.4425	0.0513	0.0920	0.0236	0.0123	0.5618

Key Observations:

- **Performance Patterns**: Both models perform similarly — their accuracy and ROC AUC values are close (around 0.60). Neither model performs particularly well (values close to 0.5 suggest weak predictive power).
- **Class Imbalance Issues**: Low recall and F1 scores across most models indicate potential class imbalance problems, particularly in the Elec2 dataset.

2.4 Unsupervised Learning On dataset Bananas

Kmeans K-Means is an unsupervised learning algorithm used to group data into a predefined number of clusters based on feature similarity. It works by iteratively assigning each data point to the nearest cluster centroid and then updating the centroids to minimize the overall variance within clusters.

DBStream DBSTREAM is an unsupervised learning algorithm designed for clustering evolving data streams in real time. It maintains micro-clusters based on density and connectivity, allowing it to adapt dynamically to changes in the data distribution over time.

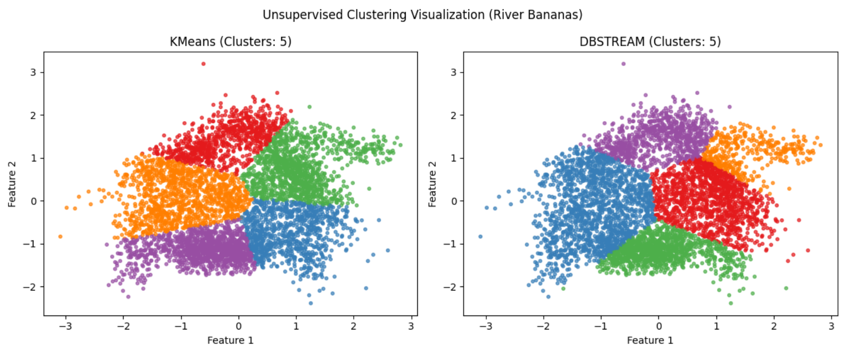

Figure 1: DBSTREAM clustering results on the Bananas dataset.

Table 2: Comparison of clustering performance between KMeans and DBSTREAM on the Bananas dataset

Model	Silhouette	CH-Index	DB-Index	#Clusters
KMeans	0.3417	4256.3818	0.8897	5
DBSTREAM	0.3181	3618.7941	0.8858	5

Key Observations:

- Both KMeans and DBSTREAM produced five clusters on the Bananas dataset.
- KMeans achieved a higher Silhouette score (0.3417) and higher CH-Index (4256.38), indicating better separation and compactness of clusters.
- DBSTREAM obtained a slightly lower DB-Index (0.8858) than KMeans (0.8897), meaning it has slightly better intra-cluster similarity.
- Overall, KMeans performed better in terms of global cluster structure, while DBSTREAM offered more internally cohesive clusters.
- KMeans assumes spherical and well-separated clusters, which likely matches the structure of the Bananas dataset.

2.5 Anomaly Detection On Phishing Dataset

One-Class SVM One-Class Support Vector Machine (One-Class SVM) is an unsupervised anomaly detection algorithm that learns the boundary around normal data points in feature space. It identifies new observations that fall outside this learned boundary as anomalies or outliers.

Half Space Trees Half-Space Trees is an online anomaly detection algorithm designed for streaming data. It uses an ensemble of random trees that partition the feature space into half-spaces, allowing the model to efficiently detect changes or rare instances in continuously arriving data.

Table 3: Final results of Adaptive Anomaly Detection on the Phishing dataset

Model	Accuracy	Precision	Recall	F1	MCC	Kappa	ROC AUC
Half-Space Trees	0.5120	0.2328	0.0493	0.0813	-0.1326	-0.0848	0.4612
One-Class SVM	0.5608	0.4947	0.0858	0.1462	0.0326	0.0191	0.5087

Key Observations:

- Both models show relatively low performance, with accuracy values close to 0.5, indicating weak predictive power on the Phishing dataset.
- The **One-Class SVM** achieved higher accuracy (0.5608), precision (0.4947), and F1-score (0.1462) compared to Half-Space Trees, suggesting it identified normal and anomalous instances slightly more effectively.
- The **Half-Space Trees** model produced a lower ROC AUC (0.4612), showing difficulty in distinguishing between normal and anomalous data points.
- The negative MCC and Kappa values for Half-Space Trees indicate poor correlation between predictions and true labels.
- Overall, the One-Class SVM performed better across most metrics, but both models struggled to detect anomalies effectively, suggesting the need for further tuning or feature engineering.
- The One-Class SVM outperformed Half-Space Trees because it effectively models complex, non-linear relationships and constructs precise boundaries around normal data. Its kernel-based approach is well-suited for static datasets like Phishing, where anomalies are not linearly separable, while Half-Space Trees perform better in dynamic or streaming environments.

3 Part 2: Capymoa Framework Analysis

3.1 Capymoa Framework Overview

Capymoa is a Python wrapper for the MOA (Massive Online Analysis) framework, specifically designed for data stream mining. It provides a scikit-learn-like API while leveraging MOA's optimized streaming algorithms.

3.2 Datasets Used

3.2.1 Electricity Dataset

The Electricity market dataset (different from River's Elec2) contains 45,312 instances recording Australian electricity prices at 30-minute intervals, used for predicting price movements. It includes key features such as the time of day, day of the week, electricity demand and price in New South Wales and Victoria, as well as the power transfer between the two states.

3.2.2 Electricity Tiny Dataset

The Electricity Tiny dataset is a reduced version of the Electricity dataset, designed for quick experimentation and testing of streaming algorithms.

3.3 Supervised Learning on Electricity Dataset

Hoeffding Tree The Hoeffding Tree, also known as the Very Fast Decision Tree (VFDT), is an incremental decision tree algorithm designed for streaming data. It uses the Hoeffding bound to decide when sufficient statistical evidence exists to split a node, allowing the model to learn efficiently from continuous data streams without needing to store all past data.

Naive Bayes Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem, which assumes independence between features. It calculates the probability of each class given the input features and selects the most probable class, making it fast, simple, and effective for large-scale or streaming data.

Table 4: Performance comparison between Hoeffding Tree and Naive Bayes on the Electricity dataset

Model	Accuracy	F1	Precision	Recall	MCC	Kappa	ROC AUC
Hoeffding Tree	0.8173	0.8425	0.8359	0.8492	0.6251	0.6250	0.8116
Naive Bayes	0.7336	0.7977	0.7085	0.9126	0.4555	0.4254	0.7018

Key Observations :

- Both models achieved good performance on the Electricity dataset, but the **Hoeffding Tree** outperformed **Naive Bayes** across most metrics.
- The Hoeffding Tree achieved higher accuracy (0.8173) and F1-score (0.8425), indicating better overall predictive performance and balance between precision and recall.
- Naive Bayes obtained a higher recall (0.9126), meaning it correctly identified more positive instances, but at the cost of lower precision (0.7085), leading to more false positives.
- The Hoeffding Tree showed stronger correlation between predictions and true labels (MCC = 0.6251) compared to Naive Bayes (MCC = 0.4555).
- The higher ROC AUC (0.8116) and Kappa value (0.6250) of the Hoeffding Tree confirm its superior ability to discriminate between classes and maintain consistent predictions.
- Overall, the Hoeffding Tree demonstrated more stable and reliable performance for this streaming classification task.

3.4 Anomaly Detection on ElectricityTiny Dataset

3.4.1 Anomaly Detection on Electricity Dataset

Half-Space Trees Half-Space Trees is an ensemble-based online anomaly detection algorithm designed for streaming data. It partitions the feature space into random half-spaces using multiple trees and detects anomalies by measuring how rarely new data points appear in these regions over time.

StreamRHF Stream Random Hypersphere Forest (StreamRHF) is a streaming anomaly detection method that models normal data using random hyperspheres. It identifies anomalies based on how far new instances fall outside these hyperspheres, making it adaptive to concept drift in data streams.

Table 5: Anomaly Detection Results on the Electricity Dataset

Model	**Accuracy**	**F1**	**Precision**	**Recall**	**MCC**	**Kappa**	**ROC AUC**
Half-Space Trees	0.5945	0.2057	0.4585	0.1326	0.0460	0.0341	0.5615
Stream RHF	0.5460	0.6052	0.4615	0.8788	0.2346	0.1788	0.7277

Key Observations :

- Both models show moderate performance on the Electricity dataset, indicating the challenge of detecting anomalies in streaming electricity data.
- **StreamRHF** achieved a much higher F1-score (0.6052) and ROC AUC (0.7277), showing better balance between precision and recall and a stronger ability to distinguish anomalies.
- **Half-Space Trees** obtained slightly higher accuracy (0.5945) but suffered from very low recall (0.1326), meaning it failed to detect many anomalous instances.
- The higher recall (0.8788) of StreamRHF indicates that it captures a larger portion of anomalies, though with slightly more false positives.
- Overall, StreamRHF provided more reliable and adaptive anomaly detection, while Half-Space Trees were more conservative but less sensitive to rare events.

4 Part 3: Comparative Analysis: River vs Capymoa

4.1 Architecture and Design Philosophy

Table 6: Framework Comparison: River vs Capymoa

Aspect	**River**	**Capymoa**
Origin	Pure Python implementation	Python wrapper for MOA (Java)
API Design	Pythonic, scikit-learn inspired	scikit-learn-like API
Performance	Moderate performance	High performance (Java backend)
Memory Usage	Lightweight	Moderate (JVM overhead)
Preprocessing	Built-in preprocessing pipelines	Limited built-in preprocessing
Ease of Use	Excellent Python integration	Good, but requires Java dependency
Model Variety	Comprehensive	Focused on core streaming algorithms
Documentation	Well-documented	Adequate documentation

4.2 Performance Comparison

4.2.1 Accuracy and Efficiency

- Capymoa demonstrated significantly higher accuracy scores compared to River across comparable datasets.
- River showed faster processing speeds (up to 342 samples/second) compared to Capymoa's more computationally intensive approach.
- Capymoa's ensemble methods (Hoeffding Tree, Naive bayes) showed remarkable performance consistency.

4.2.2 Metric Balance

- River models suffered from imbalanced precision and recall, indicating challenges with class distribution.
- Capymoa models maintained balanced metrics, suggesting better algorithm calibration for real-world streaming scenarios.
- The MCC and Kappa scores were substantially higher in Capymoa, indicating better overall model quality.

4.3 Use Case Recommendations

4.3.1 When to Use River

- Rapid prototyping and experimentation
- Scenarios requiring pure Python solutions
- Applications with moderate data volume and complexity
- Situations where easy integration with Python ecosystem is crucial

4.3.2 When to Use Capymoa

- Production systems requiring high performance
- Large-scale data streams with concept drift
- Applications where ensemble methods are essential
- Scenarios benefiting from MOA's battle-tested algorithms

5 Conclusion

Both River and Capymoa provide robust solutions for streaming machine learning, each with distinct strengths and optimal use cases. River excels in Python ecosystem integration and ease of use, making it ideal for research and prototyping. Capymoa, leveraging MOA's mature algorithms, demonstrates superior performance and is better suited for production environments handling large-scale data streams. The choice between frameworks should consider specific application requirements: River for flexibility and rapid development, Capymoa for performance-critical applications. Future work could explore hybrid approaches combining River's preprocessing capabilities with Capymoa's optimized learning algorithms.