Data Stream Processing Lab2

WANG Mingxuan & MIGNOT Soline

10 November 2025

0. Datasets

Disclaimer: The datasets we are using to compare River and CapyMOA are from CapyMOA. Initially, we were supposed to take the datasets from the River library and use them on the CapyMOA methods. However, we were unable to successfully convert the River datasets to use them in CapyMOA. Therefore, we have decided to use the ones from CapyMOA instead. These datasets will be used in both River and CapyMOA, and then the libraries are going to be compared.

Table 1: Summary of Available Datasets in CapyMOA

Dataset	Instances	Attributes	Classes / Targets
Bike	17,379	12	1 (target, regression)
CovtFD	581,011	104	7
Electricity	45,312	8	2 (UP, DOWN)
ElectricityTiny	1,000	8	2 (UP, DOWN)
Fried	40,768	10	1 (target, regression)
Hyper100k	100,000	10	2
RTG_2abrupt	100,000	30	5

1. Binary Classification

1.1. Datasets and Methods

The classifiers I found on both River and CapyMOA were: ADWIN Bagging, Adaptive Random Forest, Bagging, Hoeffding Adaptive Tree, Hoeffding Tree, Leveraging Bagging, Naive Bayes, k-Nearest Neighbors, sklearn SGDClassifier. We will compare both libraries on these classifiers. The datasets we will use are from CapyMOA and are the following: Electricity (8 attributes, 2 classes (UP and DOWN), 45 312 instances), Electricity Tiny (8 attributes, 2 classes (UP and DOWN), 1 000 instances), Hyper100K (10 attributes, 2 classes, 100 000 instances).

1.2. Results

The following table compares the results from CapyMOA and River on each model and each dataset. In each line, the bold results are the best ones out of CapyMOA and River.

First of all, let's analyze the lengths of time it took for the algorithms to run. CapyMOA tends to be faster for lightweight or less complex models. Hoeffding Adaptive Tree, Hoeffding Tree, Naive Bayes, and k-Nearest Neighbors consistently execute faster in CapyMOA, often with a significant margin. For example, Hoeffding Tree on Hyper100k runs in 0.62 seconds in CapyMOA compared to 5.62 seconds in River.

This suggests that CapyMOA is better optimized for simpler models, particularly those based on decision trees or basic classification algorithms. River outperforms CapyMOA in execution time for heavier or ensemble-based models. ADWIN Bagging, Adaptive Random Forest, Bagging, Leveraging Bagging, and SGDClassifier are faster in River. For instance, Adaptive Random Forest on Electricity takes 29.01 seconds in River vs 46.53 seconds in CapyMOA. River probably has optimizations tailored for complex structures like random forests or bagging methods.

In terms of performance of the models: CapyMOA dominates in terms of F1 and Accuracy. Out of 30 result rows (10 models $\times$ 3 datasets), 24 rows show superior performance for CapyMOA in either F1 or Accuracy. Only 6 rows are better for River: Adaptive Random Forest (F1 on Electricity), Hoeffding Adaptive Tree (F1 on Electricity and Electricity Tiny), Naive Bayes (F1 and Accuracy on Electricity).

In conclusion, it seems that for the binary classification methods, CapyMOA has much better results than River. Although the running time is sometimes slower, on average the performances are much better.

2. Multi-Class Classification

1.1. Datasets and Methods

The classifiers I found on both River and CapyMOA were: ADWIN Bagging, Adaptive Random Forest, Bagging, Hoeffding Adaptive Tree, Hoeffding Tree, Leveraging Bagging, Naive Bayes, k-Nearest Neighbors, sklearn SGDClassifier. We will compare both libraries on these classifiers. The datasets are usually very big and thus the algorithms take very long to run. Therefore, we only have the results for one dataset : CovtFD. It has 581 011 instances, 104 attributes (10 continuous, 44 categorical, 50 dummy) and 7 classes.

2.2. Results

The graph below shares the performances of the multi-class classifiers for River and CapyMOA for the CovtFD dataset.

The computational times vary significantly between CapyMOA and River. CapyMOA generally requires much longer execution times for several models, including ADWIN Bagging, Adaptive Random Forest, Bagging, Leveraging Bagging, k-Nearest Neighbors, and SGDClassifier. For instance, ADWIN Bagging in CapyMOA takes 1156.58 seconds, compared to just 237.92 seconds in River, making River nearly five times faster for this model. Similarly, Adaptive Random Forest in CapyMOA requires 3117.22 seconds, whereas River completes the task in 643.93 seconds. However, River does not always outperform CapyMOA in terms of speed. For models such as Hoeffding Adaptive Tree, Hoeffding Tree, and Naive Bayes, River takes longer to compute.

Table 2: Comparison of models across CapyMOA and River frameworks (bold indicates the higher value per row).

Model	Dataset	CapyMOA			River		
		Time (s)	F1	Accuracy	Time (s)	F1	Accuracy
ADWIN Bagging	Electricity	7.64	**84.51**	**84.94**	**4.02**	73.21	57.89
ADWIN Bagging	Electricity Tiny	0.34	**84.82**	**85.25**	**0.16**	56.65	39.52
ADWIN Bagging	Hyper100k	20.96	**89.91**	**89.92**	**10.37**	66.91	50.48
Adaptive Random Forest	Electricity	46.53	89.81	**90.07**	**29.01**	**90.12**	88.48
Adaptive Random Forest	Electricity Tiny	1.32	**88.64**	89.00	**1.24**	86.29	**89.14**
Adaptive Random Forest	Hyper100k	128.52	**88.03**	**88.03**	**85.25**	84.32	84.30
Bagging	Electricity	5.04	**82.35**	**82.83**	**2.76**	73.21	57.89
Bagging	Electricity Tiny	0.23	**84.57**	**85.00**	**0.11**	56.65	39.52
Bagging	Hyper100k	14.80	**89.84**	**89.84**	**7.11**	66.91	50.48
Hoeffding Adaptive Tree	Electricity	**0.25**	83.46	**83.91**	3.98	**85.11**	82.62
Hoeffding Adaptive Tree	Electricity Tiny	**0.01**	**83.32**	84.15	0.19	79.31	**84.44**
Hoeffding Adaptive Tree	Hyper100k	**1.02**	87.11	87.11	15.27	**87.80**	**87.75**
Hoeffding Tree	Electricity	**0.12**	81.25	**81.73**	1.81	**83.11**	79.77
Hoeffding Tree	Electricity Tiny	**0.01**	**82.10**	**82.65**	0.08	59.28	69.83
Hoeffding Tree	Hyper100k	**0.62**	**87.91**	**87.92**	5.62	83.99	84.36
Leveraging Bagging	Electricity	16.68	**88.87**	**89.16**	**13.28**	73.21	57.88
Leveraging Bagging	Electricity Tiny	0.54	**86.05**	**86.70**	**0.53**	56.65	39.52
Leveraging Bagging	Hyper100k	46.42	**89.15**	**89.15**	**33.81**	66.79	50.18
Naive Bayes	Electricity	**0.09**	72.84	73.36	3.14	**81.36**	**76.12**
Naive Bayes	Electricity Tiny	**0.01**	**83.28**	**84.00**	0.13	9.58	53.73
Naive Bayes	Hyper100k	**0.55**	**93.67**	**93.67**	7.49	66.80	69.95
k-Nearest Neighbors	Electricity	**2.30**	**83.68**	**84.08**	3.21	73.68	58.90
k-Nearest Neighbors	Electricity Tiny	**0.08**	**81.04**	**81.60**	0.13	56.70	39.57
k-Nearest Neighbors	Hyper100k	8.95	**79.97**	**79.97**	**8.26**	66.75	50.10
SGDClassifier	Electricity	8.13	**83.80**	**84.18**	**0.95**	73.26	58.00
SGDClassifier	Electricity Tiny	0.35	**84.00**	**84.70**	**0.04**	56.65	39.52
SGDClassifier	Hyper100k	18.79	**88.83**	**88.83**	**2.59**	66.97	50.61

For example, Hoeffding Adaptive Tree runs in 39.34 seconds in CapyMOA but requires 598.72 seconds in River, making CapyMOA significantly faster for these specific models. This indicates that while River is generally faster for most models, CapyMOA can be more efficient for certain classifiers, particularly those based on simpler or more traditional algorithms.

In terms of performance, both frameworks deliver competitive results, with some notable exceptions. CapyMOA tends to achieve higher F1 and Accuracy scores across most models. For example, Adaptive Random Forest in CapyMOA achieves an F1 score of 0.87 and an Accuracy of 0.93, compared to 0.85 and 0.81, respectively, in River. Similarly, Leveraging Bagging in CapyMOA achieves an F1 score of 0.85 and an Accuracy of 0.92, outperforming River's 0.79 and 0.76. These results suggest that CapyMOA may offer superior predictive performance for many models, making it a preferable choice when accuracy is a priority. The Naive Bayes model is an outlier, performing poorly in both frameworks, with particularly low Accuracy and F1 scores. This suggests that Naive Bayes may not be well-suited for the CovtFD dataset, regardless of the library used.

Overall, the choice between CapyMOA and River depends on the specific requirements of the task. If computational efficiency is a priority, River is generally the better choice for most models, except for Hoeffding Adaptive Tree, Hoeffding Tree, and Naive Bayes, where CapyMOA is faster. On the other hand, if predictive performance is the primary concern, CapyMOA tends to deliver better results across most models.

Table 3: Comparison of models across CapyMOA and River for the multi-class classifiers.

Model	Dataset	**CapyMOA**			**River**		
		Time(s)	F1	Accuracy	Time(s)	F1	Accuracy
ADWIN Bagging	CovtFD	1156.58	**0.77**	**0.84**	**237.92**	0.77	0.73
Adaptive Random Forest	CovtFD	3117.22	**0.87**	**0.93**	**643.93**	0.85	0.81
Bagging	CovtFD	911.12	0.65	0.76	**197.98**	**0.76**	**0.72**
Hoeffding Adaptive Tree	CovtFD	**39.34**	0.77	**0.85**	598.72	**0.78**	0.72
Hoeffding Tree	CovtFD	**26.60**	0.65	**0.76**	307.38	**0.73**	0.66
Leveraging Bagging	CovtFD	1665.93	**0.85**	**0.92**	**956.86**	0.79	0.76
Naive Bayes	CovtFD	**23.22**	**0.48**	**0.52**	305.57	0.14	0.25
k-Nearest Neighbors	CovtFD	484.67	**0.78**	**0.86**	**240.14**	0.76	0.72
SGDClassifier	CovtFD	684.32	**0.84**	**0.93**	**77.69**	0.76	0.73

3. Regression

3.1. Datasets and Methods

The regressors I found on both River and CapyMOA were: Passive-Aggressive Regressor, Stochastic Gradient Tree, [baseline] Mean predictor, k-Nearest Neighbors, Adaptive Random Forest. We will compare both libraries on these classifiers. The datasets we will use are from CapyMOA and are the following : Bike (12 attributes and 17 379 instances), Fried (10 attributes and 40 768 instances).

3.2. Results

For the Passive-Aggressive Regressor, the River library has two modes. Because mode 1 had overall the best results, it is the one we will use to compare to CapyMOA.

The following table compares the results from CapyMOA and River on each regressor on both datasets. In each line, the bold results are the best ones out of CapyMOA and River.

When it comes to execution time, River demonstrates a clear advantage with the Passive-Aggressive Regressor, executing approximately ten times faster than CapyMOA on both datasets. However, for the remaining models (Mean Predictor, k-Nearest Neighbors, and Adaptive Random Forest) CapyMOA consistently outperforms River. We think it is important to note that two methods were about 10x faster than the other two : Passive-Aggresive Regressor and Mean Predictor vs k-Nearest Random Forest and Adaptive Random Forest. So if we need a fast method, these would be the top choices. However, the performance of the mean predictor is not very good (R2 = 0).

In terms of performance metrics, CapyMOA is generally better. For the Passive-Aggressive Regressor, CapyMOA achieves superior RMSE, MAE, and R^2 values on both datasets, despite its slower execution time. The Mean Predictor also performs better in CapyMOA for the Fried dataset, while River is only slightly better in MAE for the Bike dataset. The k-Nearest Neighbors model is the only exception where River performs better than CapyMOA. For the Adaptive Random Forest, CapyMOA is better across all metrics, delivering significantly better RMSE, MAE, and R^2 values for both datasets.

In conclusion, for regressors, CapyMOA shows much better results, in terms of time and performance.

Table 4: Comparison of the regressors for CapyMOA and River frameworks.

Model	Dataset	CapyMOA				River			
		Time (s)	**RMSE**	**MAE**	**R2**	**Time (s)**	**RMSE**	**MAE**	**R2**
Passive-Aggressive Regressor	Fried	9.18	**3.56**	**2.79**	**0.49**	**0.91**	7.53	5.97	-1.27
Passive-Aggressive Regressor	Bike	3.85	**103.98**	**67.37**	**0.67**	**0.42**	137.12	91.74	0.43
Mean predictor	Fried	**0.08**	**4.99**	4.07	**0.00**	0.55	5.00	**4.06**	-0.00
Mean predictor	Bike	**0.06**	**181.39**	**131.80**	**0.00**	0.26	233.54	162.79	-0.66
k-Nearest Neighbors	Fried	**3.30**	2.88	2.28	0.67	73.50	**2.78**	**2.18**	**0.69**
k-Nearest Neighbors	Bike	**1.53**	127.94	83.79	0.50	21.63	**122.58**	**81.18**	**0.54**
Adaptive Random Forest	Fried	**48.64**	**2.20**	1.72	**0.81**	84.30	**2.20**	**1.67**	**0.81**
Adaptive Random Forest	Bike	**6.72**	**97.43**	**65.93**	**0.71**	16.02	115.02	78.37	0.60

4. Anomaly Detection

4.1. Datasets and Methods

For this last part, we were supposed to compare the anomaly detection methods between CapyMOA and River. However, we unfortunately only found one anomaly detector in common: Half Space Trees. Therefore, that is the only one we will be testing. We will be testing it on two datasets for CapyMOA : Electricity Tiny (8 attributes, 2 classes and 45 312 attributes) and Hyper100K (10 attributes, 2 classes, 100 000 instances)..

4.2. Results

The table below shows the performances of the Half Space Tree for River and CapyMOA for the two datasets.

The computational efficiency of the Half Space Trees model does not vary two significantly between River and CapyMOA. For the Electricity Tiny dataset, River demonstrates an advantage in speed, completing the algorithm in just 0.17 seconds, compared to 0.36 seconds for CapyMOA. This makes River nearly two times faster for this dataset. However, for the Hyper100K dataset, the bigger one, CapyMOA is faster than River, taking 4.9 seconds compared to 6.9 seconds for River. This indicates that CapyMOA may be more efficient for certain datasets.

River is the better choice for AUC, especially given its clear lead.

Therefore, for anomaly detection, it seems that River is the better framework.

As for the other anomaly detectors, it seems that the others take way longer to run,

especially Local Outlier Factor for River and Streaming Isolation Forest in CapyMOA.

Table 5: Comparison of anomaly detection models between CapyMOA and River.

Model	Dataset	**CapyMOA**		**River**	
		Time(s)	Accuracy	Time(s)	AUC
Half Space Trees	Electricity Tiny	0.36	53.58	**0.17**	**87.49**
Half Space Trees	Hyper100k	**4.89**	30.89	6.91	**1.0**
Autoencoder	Electricity Tiny	1.41	57.52		
Autoencoder	Hyper100k	22.19	50.03		
Streaming Isolation Forest	Electricity Tiny	18.69	60.71		
Streaming Isolation Forest	Hyper100k	100.67	49.52		
OneClassSVM	Electricity Tiny			0.03	78.04
OneClassSVM	Hyper100k			2.68	52.27
Local Outlier Factor	Electricity Tiny			82.49	0.99
Local Outlier Factor	Hyper100k			> 10 mins	NA

5. Conclusion

In the end, both CapyMOA and River perform well, but each in their own domain. It seem that for binary classification and regressors, CapyMOA has stronger results. For multi-class and anomaly detection, River is a stronger choice. This said, this depends on which model is being used and most importantly, it depends on the priority of the user : compromising accuracy for a speedier algorithm or having a better prediction even though it takes longer.