Introduction to Data Stream using River

PyConZA 2023: Durban 5-6 October

Tajudeen Akinosho
(M.Sc Candidate at University of South Africa)

Data Streams

Streaming data is unlimited, potentially unbounded, and continuously evolving with time (Abid et al., 2019)

Data streams are everywhere and around us these days.

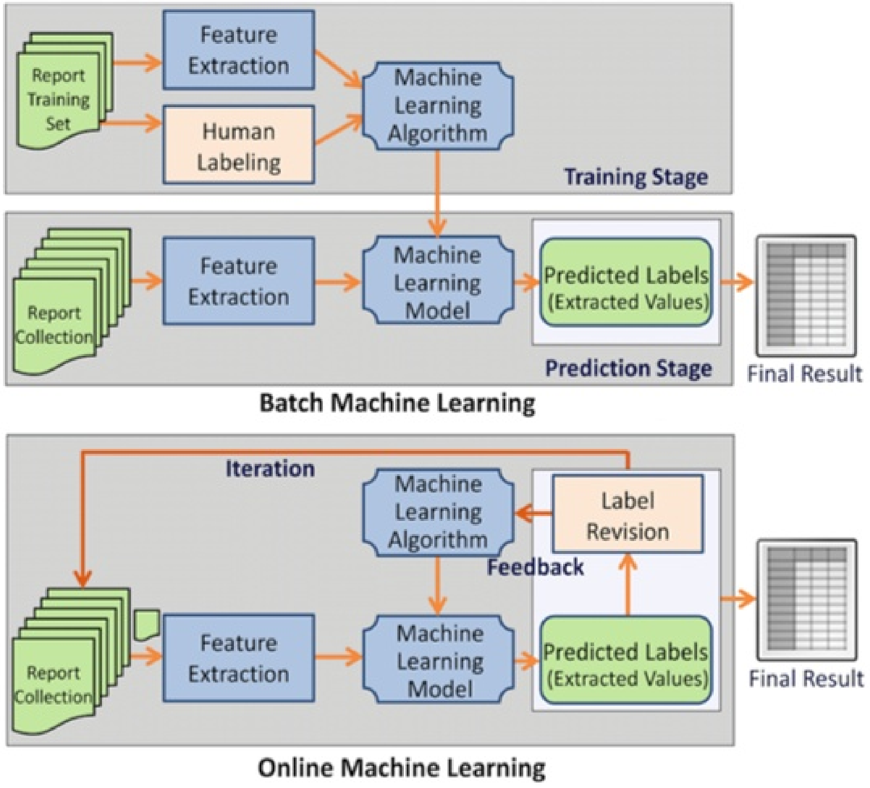
Our everyday life is now getting stuffed with devices that are emanating many data streams.

Cell-phones, cars, security sensors, radio frequency identification (RFID), health monitoring system, and televisions are just some examples.

Smart houses of the future are likely to have many different types of sensors.

Batch/Offline Learning vs O[illegible]e Learning

- Batch/offline learning approaches are incapable of gradual learning
- They usually build models from the entire training set
- Are computationally expensive in terms of computer resources
- In online learning, the training is done in small groups
- Each learning phase is quick and inexpensive
- Use limited number of computational resources.



Batch Learning Vs Online Learning (https://www.researchgate.net/figure/Online-machine-learning-versus-batch-learning-a-Batch-machine-learning-workflow-b_fig1_316818527)

Python Libraries for Online Machine Learning

To execute online machine learning, many frameworks are available

- ✓Scikit-Multiflow (also known as skmultflow)
- ✓Jubatus: Open-source online machine learning and distributed computing system.
- ✓Crème framework: This framework learn a stream of data continually.
- ✓River: Combines the scikit-multiflow and crème libraries for executing online machine learning data.

River Python Library

- River is a Python package for online/streaming machine learning.
- River is a library for incremental learning.
- Incremental learning is a machine learning regime where the observations are made available one by one.
- It’s the combination of two of Python’s most popular stream learning packages: Crème and Scikit-Multiflow
- River can perform learning task like classification, regression, clustering, and conce

River Python Library (c

- All predictive models perform two core functions: learn and predict
- Learning takes place in learn_one method
- Depending on the learning task, models provide predictions via:
  - ❖predict_one (classification, regression, and clustering)
  - ❖predict_proba_one (classification), and
  - ❖score_one (anomaly detection)
- River also contains transformers via transform_one method

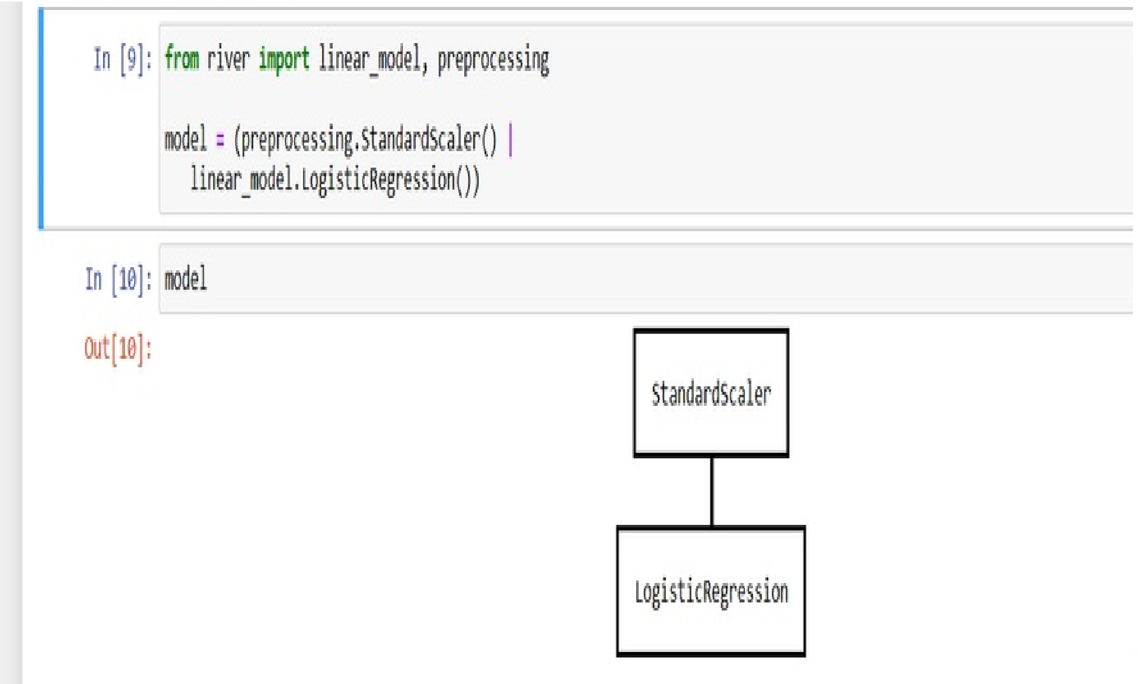
Simple Machine Learning code

- A complete machine learning task (learning, prediction, and performance measurement) easily implemented in a couple lines of code:

```
In [1]: import river
        from river import evaluate, metrics, tree
        from river import datasets

In [2]: stream = datasets.synth.Waveform(seed=42).take(1000)
        model = tree.HoeffdingTreeClassifier()
        metric = metrics.Accuracy()

In [3]: evaluate.progressive_val_score(stream, model, metric)

Out[3]: Accuracy: 79.28%

In [ ]:
```

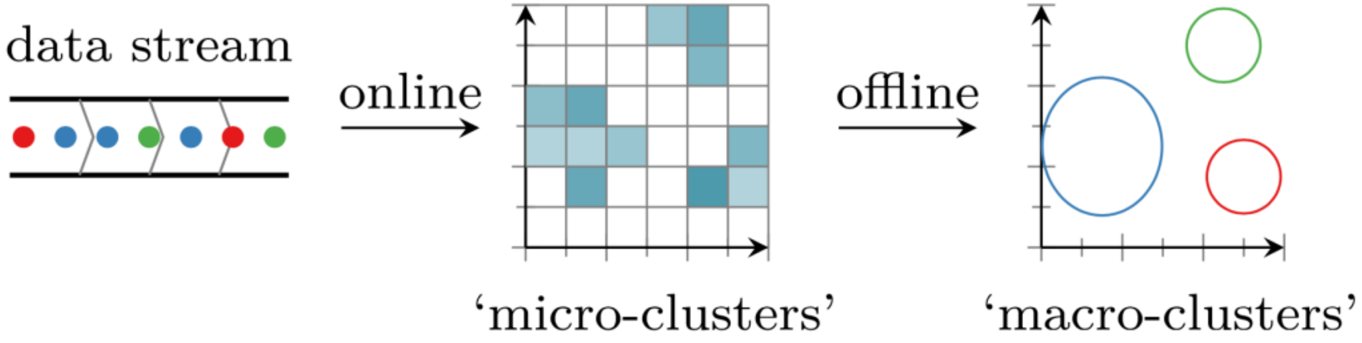
Pipelines

- Pipelines are an integral part of River.
- They are a convenient and elegant way to “chain” a sequence of operations and warrant reproducibility.
- A pipeline is essentially a list of estimators that are applied in sequence.



River Clustering

- Currently, River offers 6 clustering algorithms, including CluStream, DBSTREAM, DenStream, KMeans, STREAMKMeans, and TextClust.
- River includes most number of clustering algorithms apart the Massive Online Analysis (MOA).
- Clustering process in stream clustering algorithms are OFFLINE



Two-phase stream clustering with grid-based approach
(Source: Matthias Carnein et al. 2017. An empirical comparison of stream clustering algorithms.)

CluStream Algorithm

Aggarwal, C.C., Jianyong Wang, J.H. & Yu, P.S. 2003. A Framework for Clustering Evolving Data Streams. in Proceedings of the 29th International Conference on Very Large Data Bases (VLDB ’03), Vol. 29. VLDB Endowment Berlin, Germany. 81–92

- The algorithm utilizes a two-phase model: Online micro-clustering component and offline macro-clustering.
- It is a partitioning-based clustering and has a spherical shaped cluster.
- However, it cannot detect arbitrary-shaped clusters and sensitive to outliers.

CluStream with River

```
In [1]: from river import cluster
        from river import stream

        X = [[1, 2],[1, 4],[1, 0],[-4, 2],
            [-4, 4],[-4, 0],[5, 0],[5, 2],
            [5, 4]
        ]

        clustream = cluster.CluStream(
            n_macro_clusters=3,
            max_micro_clusters=5,
            time_gap=3,
            seed=0,
            halflife=0.4
        )

In [2]: for x, _ in stream.iter_array(X):
            clustream = clustream.learn_one(x)

In [3]: clustream.predict_one({0: 1, 1: 1})
Out[3]: 1

In [4]: clustream.predict_one({0: -4, 1: 3})
Out[4]: 2

In [5]: clustream.predict_one({0: 4, 1: 3.5})
Out[5]: 0
```

CluStream with River

In [6]:
```python
import pandas as pd
df = pd.read_csv("t4.8k.csv", header=None)
df.head()
```

Out[6]:

	0	1
0	68.601997	102.491997
1	454.665985	264.808990
2	101.283997	169.285995
3	372.614990	263.140991
4	300.989014	46.555000

In [7]:
```python
import numpy as np
```

In [8]:
```python
Z = np.array(df)
Z
```

Out[8]:
```
array([[ 68.601997, 102.491997],
       [454.665985, 264.80899 ],
       [101.283997, 169.285995],
       ...,
       [267.605011, 141.725006],
       [238.358002, 252.729996],
       [159.242004, 177.431    ]])
```

In [9]:
```python
for x, _ in stream.iter_array(Z):
    clustream = clustream.learn_one(x)
```

In [13]:
```python
clustream.predict_one({0: 50, 1: 200})
```

Out[13]:
```
1
```

DenStream Algorithm

Cao, F., Ester, M., Qian, W. & Zhou, A. 2006. Density-Based Clustering over an Evolving Data Stream with Noise. in SIAM Conference on Data Mining. 328–339 https://doi.org/10.1137/1.9781611972764.29

- DenStream is a Density-based algorithm with ability to discover arbitrary-shaped clusters in evolving data stream.
- It can handle outliers but risky when there is noise.
- DenStream has three micro-cluster features which are:

✓core-micro-cluster which summarize clusters with arbitrary-shapes,

✓potential core-micro-cluster to identify potential clusters, and

✓outlier micro-cluster for outliers detect

DenStream with River

```
In [21]: from river import cluster
         from river import stream

         X = [
             [-1, -0.5], [-1, -0.625], [-1, -0.75], [-1, -1], [-1, -1.125],
             [-1, -1.25], [-1.5, -0.5], [-1.5, -0.625], [-1.5, -0.75], [-1.5, -1],
             [-1.5, -1.125], [-1.5, -1.25], [1, 1.5], [1, 1.75], [1, 2],
             [4, 1.25], [4, 1.5], [4, 2.25], [4, 2.5], [4, 3],
             [4, 3.25], [4, 3.5], [4, 3.75], [4, 4],
         ]

         denstream = cluster.DenStream(decaying_factor=0.01,
                                       beta=0.5,
                                       mu=2.5,
                                       epsilon=0.05,
                                       n_samples_init=10)

         for x, _ in stream.iter_array(X):
             denstream = denstream.learn_one(x)

         denstream.predict_one({0: -1, 1: -2})

Out[21]: 0

In [22]: denstream.predict_one({0: 5, 1: 4})

Out[22]: 1

In [23]: denstream.predict_one({0: 1, 1: 1})

Out[23]: 0
```

DenStream with River

```
In [25]: import pandas as pd
         df = pd.read_csv("t4.8k.csv", header=None)
         df.head()
```

Out[25]:

	0	1
0	68.601997	102.491997
1	454.665985	264.808990
2	101.283997	169.285995
3	372.614990	263.140991
4	300.989014	46.555000

```
In [26]: import numpy as np

In [27]: Z = np.array(df)
         Z

Out[27]: array([[ 68.601997, 102.491997],
                [454.665985, 264.80899 ],
                [101.283997, 169.285995],
                ...,
                [267.605011, 141.725006],
                [238.358002, 252.729996],
                [159.242004, 177.431   ]])

In [28]: for x, _ in stream.iter_array(Z):
             denstream = denstream.learn_one(x)

In [29]: denstream.predict_one({0: 150, 1: 200})

Out[29]: 6
```

River

Thank you for your attention!

Presenter Email: tajuakins1@gmail.com

LinkedIn:
https://www.linkedin.com/in/tajudeen-akinosho-78114010

References

Abid, A., Jamoussi, S., Hamadou, A. Ben, 2019. AIS-Clus: A Bio-Inspired Method for Textual Data Stream Clustering. Vietnam Journal of Computer Science 06, 223–256. https://doi.org/10.1142/s2196888819500143

Aggarwal, C.C., Jianyong Wang, J.H. & Yu, P.S. 2003. A Framework for Clustering Evolving Data Streams. in Proceedings of the 29th International Conference on Very Large Data Bases (VLDB ’03), Vol. 29. VLDB Endowment Berlin, Germany. 81–92

Cao, F., Estert, M., Qian, W., Zhou, A., 2006b. Density-Based Clustering over an Evolving Data Stream with Noise, In Proceedings of the 2006 SIAM International Conference on Data Mining. Society for industrial and applied mathematics., pp. 328–339. https://doi.org/10.1137/1.9781611972764.29

Carnein, M., Assenmacher, D., Trautmann, H., 2017. An empirical comparison of stream clustering algorithms, In: ACM International Conference on Computing Frontiers 2017, CF 2017. Association for Computing Machinery, Inc, pp. 361–366. https://doi.org/10.1145/3075564.3078887