Evolving Data Stream Classification

Evaluation of Different Machine Learning Models in the Presence of Concept Drift

Yağız Özkarahan
Bachelor of Science in Computer Engineering
Bilkent University
Ankara, Türkiye
yozkarahan@gmail.com

ABSTRACT

Traditional machine learning models, despite their power and prevalence today, are not designed to adapt to changing patterns in real-time data streams, a phenomenon known as concept drift. This study evaluates the performance of six machine learning models—Accuracy Weighted Ensemble (AWE), Adaptive Random Forest (ARF), Dynamic Weighted Majority (DWM), Self-Adjusting Memory kNN (SAM-kNN), Leveraging Bagging (LB), and Multi-Layer Perceptron (MLP)—on both artificial and real-world evolving data streams. The models were tested with prequential evaluation, to analyze and plot their accuracy and adaptability over time. Results reveal that model performance varies significantly based on the nature of the data stream and the type of concept drift, with no single model universally superior or inferior. Notably, neural networks (MLP) demonstrated competitive performance despite not being inherently designed for evolving streams, while ensemble methods like ARF and SAM-kNN excelled in specific scenarios. These findings highlight the importance of selecting context-appropriate models for dynamic data environments and highlight the potential for further research in adaptive learning techniques.

KEYWORDS

Data streams, concept drift, prequential evaluation, ensemble models, machine learning, weighted ensembles.

1 Introduction

In the past few years, especially with the rapid developments in neural networks, machine learning algorithms have become significantly more accurate and faster than they ever were before. These algorithms can now capture patterns and mimic human learning at an unprecedented level, and what is even more impressive is how these algorithms are still developing and getting better at what they are doing. The applications and tools that seemed remarkable and revolutionary are already becoming outdated in a year.

However, when people think of machine learning, they often think of traditional batch-learning algorithms. These algorithms, in a nutshell, train on a specific dataset (usually of a very large size) learn its features/patterns, and make predictions based on what they have learned. These algorithms have proven to be highly effective at learning patterns that are not exactly subject to change. An example would be an object detection algorithm used in an autonomous car. These algorithms have been trained on gigantic datasets containing classes such as humans, trees, cars, buildings, stray animals etc. and they can very accurately put objects into either one of these classes. These algorithms work because the classes they have been trained on do not change in appearance over time; a cat always looks like a cat, so does a human or a tree, even cars have universal patterns that haven't changed much in a century (four wheels, doors and windows on side, horizontal shape etc.). The weakness of these algorithms arises when they face data that are subject to change.

Data streams are continuous flows of data that are generated and processed in real time or near real time. Unlike static datasets, the entire data is not available at an instance; it is received and processed as it arrives. By this definition, it can quickly be observed that unlike static datasets, data streams also deal with the dimension of time and are therefore subject to being changed over time. This behaviour of data streams is named concept drift, which means that the arriving data can change over time in terms of its attributes, patterns and features and accordingly, the model processing the data needs to be constantly updated to learn these changes.[1].

This project deals with data streams that exhibit the behaviour of concept drift, which are called evolving data streams. In this project, different machine learning models are evaluated on both artificial and real-life evolving data streams, in terms of how accurately they classify data over time, and see if they can adapt to concept drift. Then, their results are compared and discussed, and their behaviour is interpreted.

2 Related Work

This project was made possible majorly with the help of Python libraries scikit-multiflow and scikit-learn. Scikit-multiflow is a library containing classifier models, estimators, data stream generators and various methods specifically made for the task of learning and classifying data streams. Most of the models tested in this project were created and evaluated directly from scikit-multiflow's tools. Scikit-learn, on the other hand, is also a library

of machine learning models and methods. Scikit-learn consists more of traditional batch-learning tools, instead of tools dealing with evolving data streams.
Part of the data streams used in this project were downloaded directly from a public Git repository created by Ömer Gözüaçık [4].

3 Methodology

In this project, six different classification models were trained and evaluated on four different data streams each. The details of these models, data streams and their evaluation strategies are described in this section.

3.1 Classification Models

3.1.1 Accuracy Weighted Ensemble (AWE)
Accuracy Weighted Ensemble is an ensemble classifier (meaning that the classification is made according to the votes cast by many models) in which the votes cast by each model in the ensemble are weighted according to their individual accuracy. As new data arrives, the individual accuracies of models are also subject to change, therefore so are their voting weights. By assigning more importance to the decisions made by the more accurate models, and dynamically re-evaluating the accuracies of models, this ensemble can accurately respond to the changes and drifts in the data distribution. The default ensemble consists of a maximum of ten estimators. This means that there are ten estimators active at any given time; poorly performing estimators are removed and replaced by a newly trained estimator dynamically. Each estimator in the ensemble is a Naïve Bayes estimator. [5, 6]

3.1.2 Adaptive Random Forest (ARF)
Adaptive Random Forest is an ensemble model that extends the traditional Random Forest algorithm (which is used in traditional batch learning on static datasets) by adding adaptation mechanisms that respond to concept drift. In other words, it is a version of Random Forest(s) designed specifically for evolving data streams. According to scikit-multiflow's definition, Adaptive Random Forests overcome the challenge of concept drift by re-sampling, randomly selecting subsets of features for node splits, and using drift detectors in each tree in the ensemble. The default ARF model consists of ten trees in the ensemble. Each of these trees are defined as Hoeffding Trees. [7, 8]

3.1.3 Dynamic Weighted Majority (DWM)
Dynamic Weighted Majority (DWM) is another ensemble classifier designed for evolving data streams. It employs four strategies to respond to concept drift: training online classifiers, increasing or reducing the weights of the classifiers based on their performance, removing poorly performing classifiers and bringing additional classifiers into the ensemble if the overall accuracy drops below a threshold. With its default parameters, DWM employs five classifiers. Each classifier is defines as a Naïve Bayes classifier, similarly with the AWE ensemble. [9, 10]

3.1.4 k-Nearest Neighbors Classifier with Self Adjusting Memory (SAM-kNN)
Selft Adjusting Memory algorithm is a version of k-Nearest Neighbors algorithm that takes the recent and past data into account while making predictions. SAM-kNN uses two memory buffers: one for short term and one for long term memory (STM and LTM). STM stores patterns (in other words, how inputs relate to outputs) encountered in the recently arrived data, while LTM stores patterns in older data. A cleaning process takes care of maintaining these buffers. While making a decision, the model selects the best of STM or LTM. With this strategy, the model can adapt to both gradual changes, or sudden shifts in the data stream. [11, 12]

3.1.5 Leveraging Bagging (LB)
Leveraging Bagging is an ensemble method which improves upon the online Oza Bagging algorithm. It enhances performance by using a Poisson distribution with a higher λ value (default 6) to increase input diversity through more aggressive re-sampling. Additionally, it introduces output detection codes, where each class label is encoded as an n-bit binary code and n classifiers are trained independently on each bit, enabling partial error correction. To handle concept drift, accuracy is monitored over time; when drift is detected, the worst-performing classifier is reset to adapt to the new pattern. The model uses ten kNN classifiers in its ensemble by default.

3.1.6 Multi-Layer Perceptron (MLP)
Unlike the other models used in this project, Multi-Layer Perceptron is not an ensemble model. It is a type of artificial neural network, consisting of an input layer, hidden layers, and an output layer. Layers are made of neurons/nodes which are trained with back-propogation. The MLP instance used in this project consists of two hidden layers with 16 neurons each.

3.2 Data Streams

The data streams used in this project fall into two categories: artificial and real-world. Artificial data streams were generated and saved with scikit-multiflow's generator methods. While the data streams used here can more accurately be defined as datasets (since all of the data is present before even beginning the implementation), they are transformed into data streams with scikit-multiflow's DataStream method, so the models don't directly access all of the data at once and process them as streams as they are supposed to. For this reason, the terms data stream and dataset are used interchangeably in the results part.

Category	Name	# Features	# Classes	# Samples
Real world	Spam	499	2	6.213
Real world	Rialto	27	10	82.250
Artificial	Waveform	21	3	100,000
Artificial	Hyperplane	10	2	100,000

Table 1: Data streams and their attributes

3.3 Evaluation Strategy

The models were evaluated with prequential evaluation. Prequential evaluation, also known as interleaved test-then-train, is an evaluation strategy where each incoming data instance is first used to test the model (i.e., make a prediction), and then immediately used to train the model (i.e., update its knowledge). This process is repeated for every instance in the stream, therefore continuously assessing model's performance. Since it provides a view of how the model is adapting to the changes in the incoming data, this strategy is well suited for this project. The prequential evaluation method used in this project employs 20 evaluation windows, that is, data is divided into 20 sections of equal size.

4 Experimental Results

Table 2 displays the mean (average over evaluation windows) accuracy of each model, on each data stream.

	AWE	*ARF*	*DWM*	*SAM*	*LB*	*MLP*
Rialto	0.41	0.79	0.33	0.81	0.85	0.50
Spam	0.72	0.95	0.88	0.96	0.94	0.96
Hyperplane	0.92	0.87	0.93	0.88	0.73	0.93
Waveform	0.82	0.83	0.80	0.84	0.75	0.84

Table 2: Mean accuracy of a model over a dataset

4.1 Results on Real World Data Streams

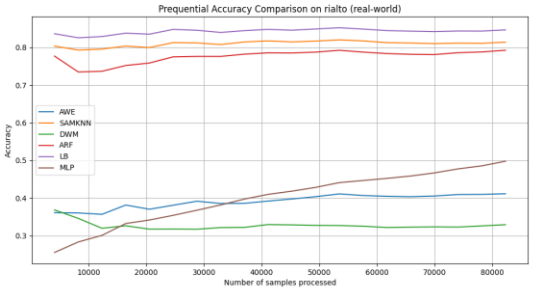

Figure 1: Prequential accuracy graph on rialto dataset

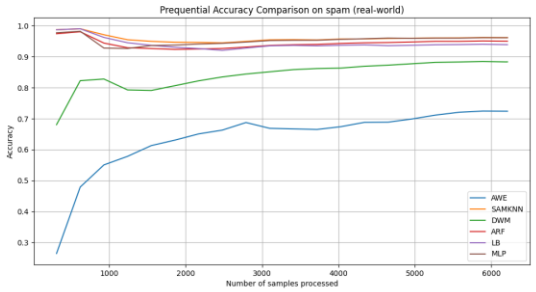

Figure 2: Prequential accuracy graph on spam dataset

4.2 Results on Artificial Data Streams

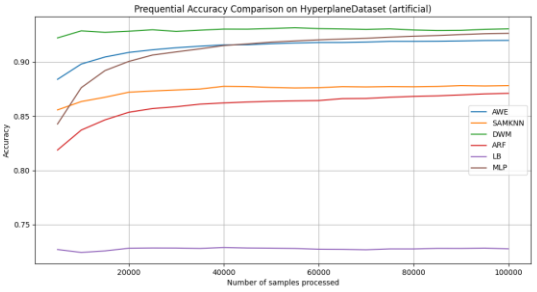

Figure 3: Prequential accuracy graph on hyperplane dataset

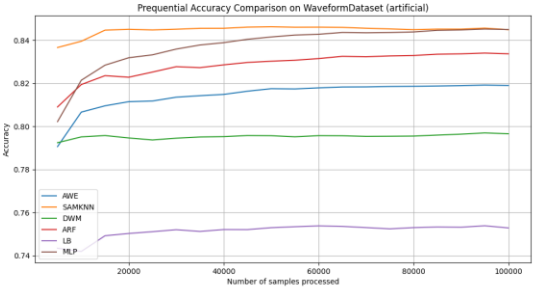

Figure 4: Prequential accuracy graph on waveform dataset

4.3 Discussions on Results

4.3.1 Rialto

Figure 1 shows that all models except MLP behave in a stable manner. Their accuracies do not change significantly as new data arrives. This implies that the rialto dataset is rather stable, the innate distribution of the dataset does not change significantly. Another explanation could be that rialto dataset does have significant concept drift but the models used here are not advanced enough to capture it. However, since the accuracies of best performing models are high enough (around 80%), this does not seem to be the case and the former explanation seems more suitable.

The outlier model in this dataset is the MLP classifier. The other models are stable, but MLP consistently rises in accuracy, although its final accuracy is still in the lower range. This is understandable, as MLP is not designed for concept drift. Since it is a neural network model, it needs more data to truly capture the patterns present in the dataset. Perhaps MLP would catch up to the best-performing models if the dataset had more entries.

Another thing to note here is that Naïve Bayes based models have performed significantly poorer than the random forest or kNN based models. This shows that the rialto dataset is not suitable for Naïve Bayes, and the ensemble results are limited by the performance of the base model.

4.3.2 Spam

As can be observed from Figure 2, most of the models have sudden fluctuations in the first few windows, but they stabilize after that and maintain a very high accuracy around 95%. This shows that the models have successfully adapted to concept drift. The outlier models, again, are the ones based on Naïve Bayes classifiers, which are noticeably less accurate. Another thing to note is, differently from the rialto example, MLP classifier performs consistently very well here. This shows that the underlying patterns in spam emails, despite the concept drift over time, are simple enough for a neural network model to learn.

4.3.3 Hyperplane

One important characteristic of the artificially generated hyperplane dataset is that it employs a gradual and smooth concept drift, instead of sudden changes. Figure 3 reflects this characteristic, as most of the models are consistently accurate after the first windows. One outlier is the LB model, which is always less accurate than the others. This might imply that LB does not perform the best with slow, gradual drift.

4.3.4 Waveform

Results are similar to the other artificially generated data stream, hyperplane. LB compares worse compared to the other models, again implying that it is not best suitable for slow drifting streams. One thing to note in the results of the artificial datasets is that while MLP starts with a slightly lower accuracy, it catches up to the best performers in the end, as it can be seen both in Figure 3 and 4. This is consistent with how neural networks work: they capture finer details and perform better with more data and more training (unless they overfit, but that did not seem to be a problem in these examples).

5 Conclusion

From all the results obtained, it is shown that no single model outperforms or underperforms the others. Instead, the model performances noticeably vary depending on the innate distribution of the data stream and the nature of its concept drift. Naïve Bayes based ensembles (AWE and DWM) perform very poorly in real-world data streams, although they do not seem to struggle in artificial ones. Leveraging Bagging model, despite performing well in real-life data streams, is consistently the worst performer in artificial ones where there is gradual and slow concept drift. MLP model, despite not being innately suitable for the task of evolving data streams, has interestingly performed well in some of the data streams, especially in the artificial ones. This is a good display of the power of neural networks, that with enough data and training, they can handle concept drifts if the underlying patterns are suitable for them. Most importantly, the results show that in each data stream, there is at the very least one model that performs very well, therefore displaying that the challenge of evolving data streams is a challenge that can be overcome. Further research can improve upon this challenge with a higher number of data streams with different characteristics, and by using different model configurations, or perhaps different models altogether.

REFERENCES

[1] Hamed R Bonab and Fazli Can. Goowe: Geometrically optimum and online-weighted ensemble classifier for evolving data streams. ACM *Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):1–33, 2018.

[2] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.github.io/

[3] Scikit-Learn. Retrieved April 13, 2025 https://scikit-learn.org/stable/

[4] Github. Retrieved April 13, 2025 https://github.com/ogozuacik/concept-drift-datasets-scikit-multiflow

[5] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.meta.AccuracyWeightedEnsembleClassifier.html

[6] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 226–235, 2003.

[7] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.meta.AdaptiveRandomForestClassifier.html

[8] Masashi Sugiyama, Tsuyoshi Idé, Shinichi Nakajima, and Jun Sese. Semi-supervised local fisher discriminant analysis for dimensionality reduction. Machine learning, 78:35–61, 2010.

[9] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.meta.DynamicWeightedMajorityClassifier.html

[10] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research, 8:2755–2790, 2007.

[11] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.lazy.SAMKNNClassifier.html

[12] Viktor Losing, Barbara Hammer, and Heiko Wersing. Knn classifier with self adjusting memory for heterogeneous concept drift. In 2016 IEEE 16th international conference on data mining (ICDM), pages 291–300. IEEE, 2016.

[13] Scikit-Multiflow. 2020.(June 2020). Retrieved April 13, 2025 from https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.meta.LeveragingBaggingClassifier.html

[14] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I 21, pages 135–150. Springer, 2010.

[15] Scikit-Learn. Retrieved April 13, 2025 https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html