Ihsan Dogramaci Bilkent University

GE 461: Introduction to Data Science

Report on the Results and Discussions of the Data Stream Mining Assignment

Name and Surname: Ali Aral Takak
Student ID: 22001758
Department: EEE
Address: Üniversiteler, 06800 Çankaya/Ankara

Introduction

In this assignment, we study the problem of learning from evolving data streams, where instances arrive continuously and the underlying distribution of data may change over time. Traditional batch-learning methods assume access to the entire dataset, however, in many real-world applications, data must be processed one instance at a time and models must adapt on the go. Hence, we implement and compare several stream-classification approaches: two learners (Adaptive Random Forest and SAM-kNN) with active drift detectors, and two custom ensemble variants (active and passive drift handling) which are built around Hoeffding trees. All methods are evaluated under the interleaved test-then-train protocol on both synthetic and real-world streams. The further contents of this report will explain the setup, then proceed to present accuracy results, and discuss how each method responds to abrupt and gradual drift.

Theoretical Concepts

Data streaming refers to the continuous, real-time flow of individual data instances—such as sensor reading, user clicks, or financial ticks—that must be processed at the time of arrival, rather than stored and examined all at once [1]. Traditional learning methods, such as batch learning, assume that the entire data is available beforehand: a model is trained offline on this fixed corpus, ofthen through multiple passes over the data, and then deployed without further modification until the retraining of the model. On the contrary, in streaming scenarios, algorithms update their models incrementally, consuming each new example only once and adapting immediately to changes in the data distribution (which is known as concept drift) [1]. Hence, streaming methods prioritize low memory use, rapid updates, and adaptability, whereas batch learning methods can leverage more compute per example and often achieve higher accuracy when the data are readily available and plentiful.

Concept drift describes the phenomenon whereby the statistical properties of the target variable—conditioned on the input features—shift over time, undermining the validity of models trained on historical data [2]. These shifts can be classified as abrupt (a sudden change, such as a sensor replacement or a market crash), gradual (a slow transition, for example seasonal patterns in user behavior), incremental (small, cumulative changes), or recurring (periodic shifts, like daily or weekly cycles) [2]. In the data streaming context, unaddressed drift leads to steadily worsening predictive performance. To manage drift, stream-learning systems employ drift detectors (for

instance ADWIN or DDM) that monitor performance indicators—such as error rate or distributional statistics—and signal when adaptation is needed. Upon detection, adaptive strategies may retrain parts of the model, adjust ensemble weights, or reset learner components, thereby maintaining accuracy in evolving environments. There are three main drift detection methods that were examined in this work:

ADWIN (Adaptive Windowing) [5]: ADWIN maintains a variable-length window of recent data and contiunously test whether the distribution within the window has changed. It conceptually splits the window into two subwindows W_0 and W_1 and uses a statistical test based on the Hoeffding bound to compare their means. Whenever the difference between the averages of windows exceeds a threshould determined by a user-specified confidence parameter δ, ADWIN concludes that a change has occurred and drops the older portion of the window. This guarantees that the remaining window contains only data from the new distribution, which provides both false-positive and false-negative guarantees on drift detection.

DDM (Drift Detection Method) [3]: DDM tracks the error rate p_i of an online learner at time step i , viewing each prediction as a Bernoulli trial, and its standard deviation $s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$. It records the minimum values p_{min} and s_{min} observed so far. A warning is raised if $p_i + s_i \geq p_{min} + 2s_{min}$, and actual drift is signaled if $p_i + s_i \geq p_{min} + 3s_{min}$.

EDDM (Early Drift Detection Method) [4]: EDDM refines DDM for gradual drift by monitoring the distance (number of instances) between consecutive errors rather than just the error rate. It keeps a running average distance d_i and standard deviation σ_i between errors, as well as their historical maxima d_{max} and σ_{max}. A warning level is reached when the ratio d_i/d_{max} falls below a warning threshold, and drift is declared when it falls below a more stringent threshold.

Results and Discussion

1) How does your ensemble model perform compared to the state-of-the-art approaches? What could be possible improvements for a more robust ensemble? Discuss your findings on the accuracy plots. What is inferred from the drops in the prequential accuracy plot?

The table given below display the overall accuracy of each classifier for SEA-Generated Data:

Classifier	Accuracy
ARF	99.52%
SAMkNN	94.02%
Active Ensembler	98.44%
Passive Ensembler	98.70%

Table 1: Accuracy results for all four classifiers on SEA-Generated Data.

The table given below display the overall accuracy of each classifier for AGRAWAL-Generated Data:

Classifier	Accuracy
ARF	99.40%
SAMkNN	65.88%
Active Ensembler	99.80%
Passive Ensembler	99.83%

Table 2: Accuracy results for all four classifiers on AGRAWAL-Generated Data.

The table given below display the overall accuracy of each classifier for Spam Dataset:

Classifier	Accuracy
ARF	95.20%
SAMkNN	92.77%
Active Ensembler	87.59%
Passive Ensembler	87.61%

Table 3: Accuracy results for all four classifiers on Spam Data.

The table given below display the overall accuracy of each classifier for Electricity Dataset:

Classifier	Accuracy
ARF	85.93%
SAMkNN	79.01%
Active Ensembler	85.29%
Passive Ensembler	82.68%

Table 4: Accuracy results for all four classifiers on Electricity Data.

The figures given below display the accuracy plot of each classifier in all four datasets, with segment size of 20:

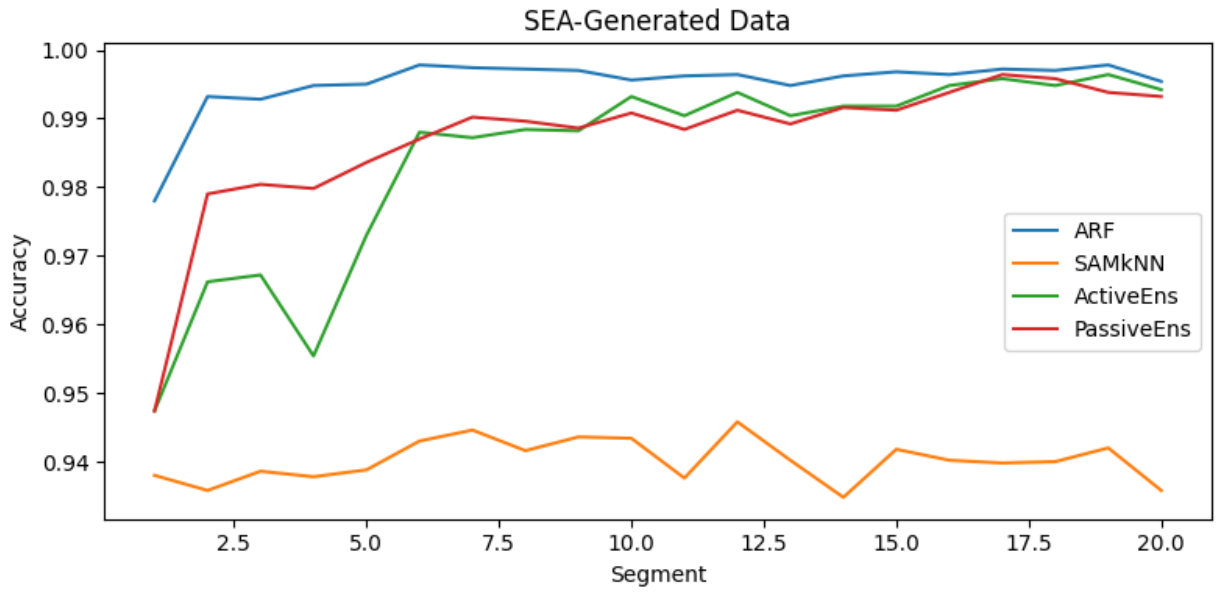

Figure 1: *Accuracy plot for SEA-Generated data for 20 segments.*

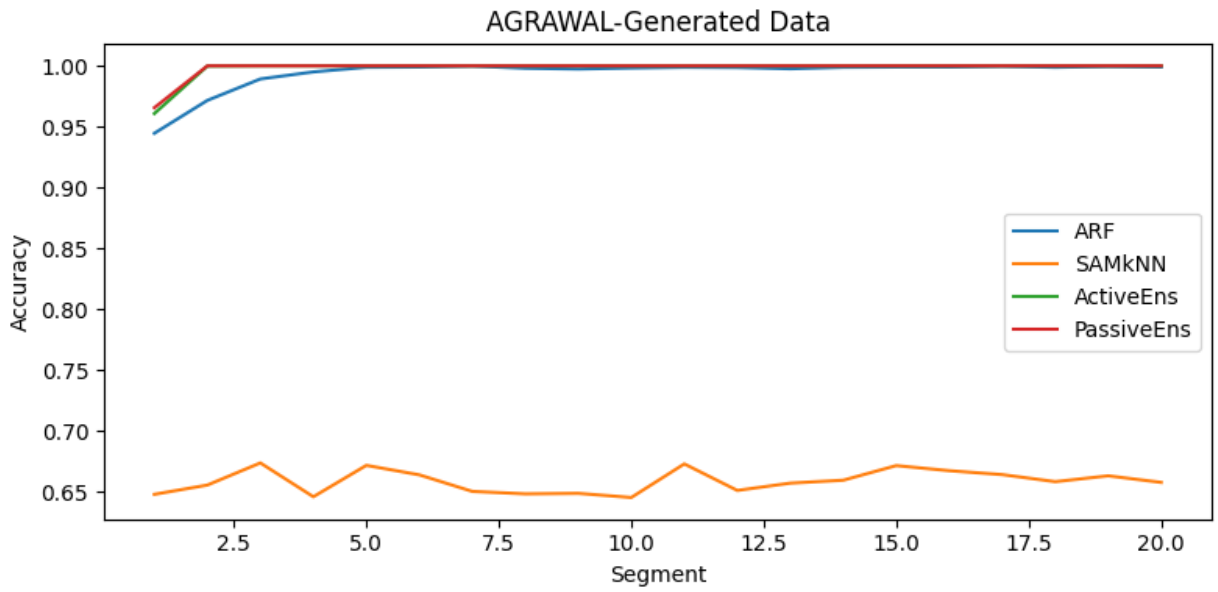

Figure 2: *Accuracy plot for AGRAWAL-Generated data for 20 segments.*

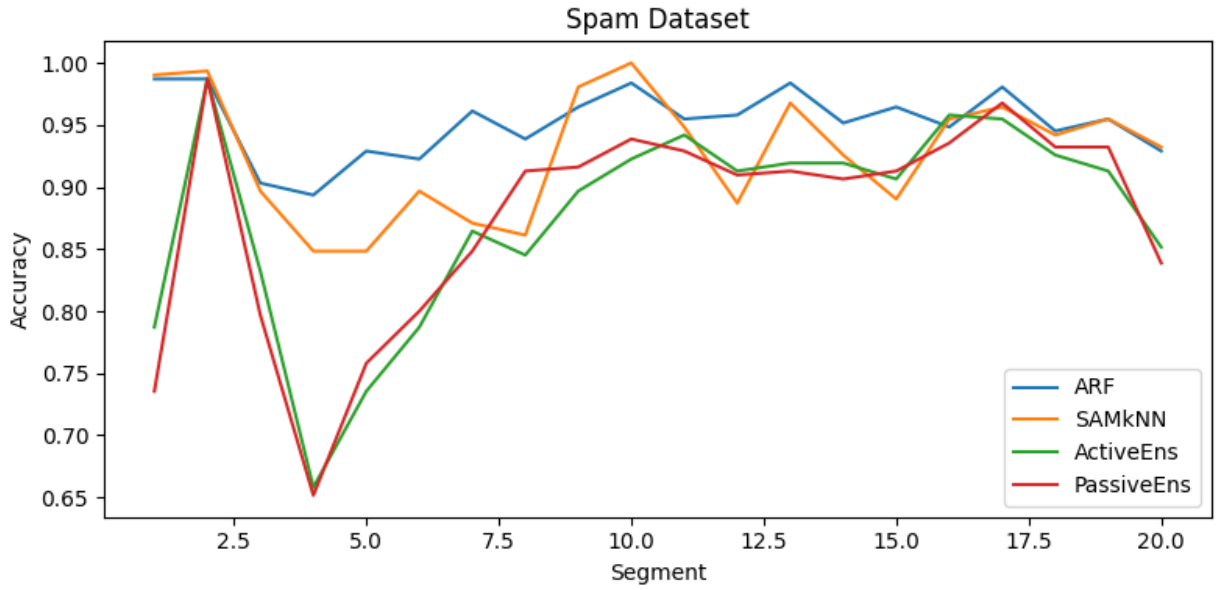

Figure 3: *Accuracy plot for Spam dataset for 20 segments.*

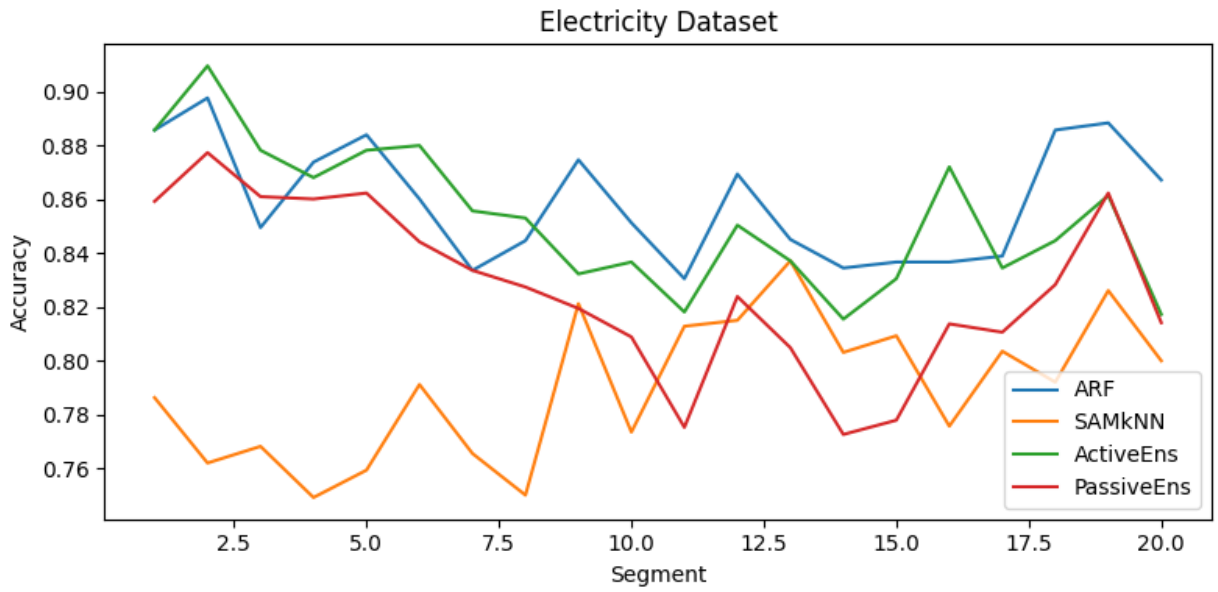

Figure 4: *Accuracy plot for Electricity dataset for 20 segments.*

On all four datasets, our ensemble model achieved accuracy levels comparable to the established classifiers. On the real-world datasets, we observe a drastical decrease in all four classifiers, possibly related to variables such as noise or outliers. However, using our results, we can infer that our implementations are succesful in classifying streaming data.

Possible improvements include increasing classifier diversity, such as implementing Naïve Bayes classifier; implementing dynamic weight adjustment, or integrating further drift detection mechanisms.

Every pronounced dip in accuracy aligns with a concept. The depth of each drop reflects the proportion of outdated training data still influencing predictions, while the width of the dip corresponds to the model's adaptation latency. For example, in SEA and Agrawal, passive ensemble's narrow single-segment drops demonstrate rapid learning of the new concept, whereas active ensemble's slightly wider dips reveal the time taken by its detector to signal an update.

2) Try different window sizes in your ensemble. How does it affect the prediction accuracy?

The figures given below display the results for Electricity and Spam dataset for a segment size of 20, with a window size of 500, which was initially 100:

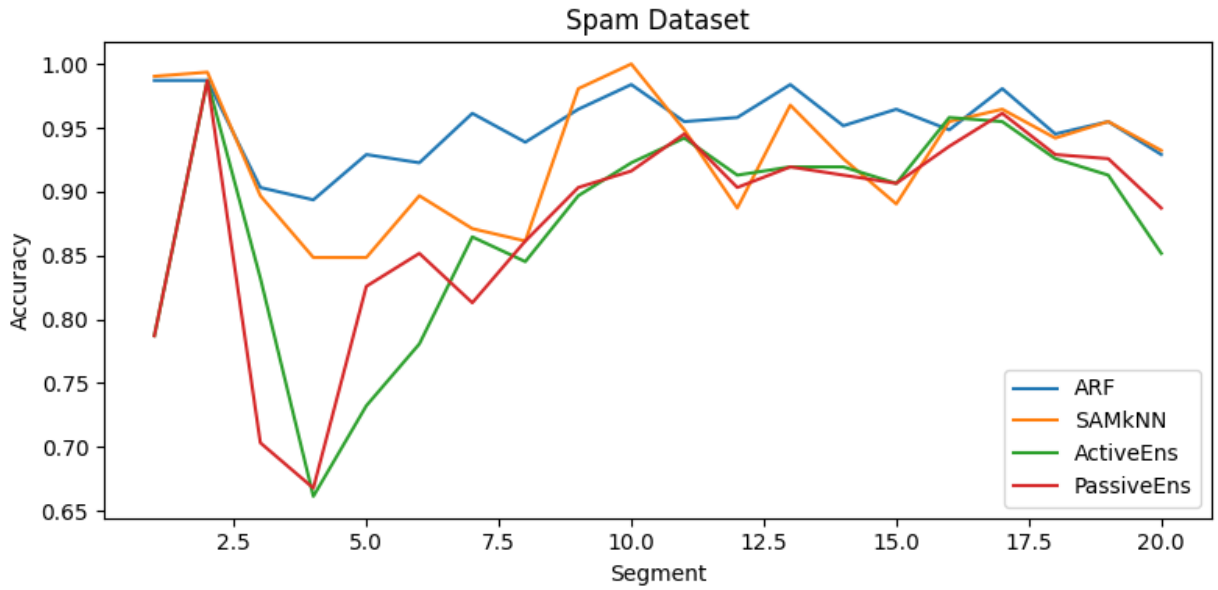

Figure 5: *Accuracy plot for Spam dataset for 20 segments and 500 window size.*

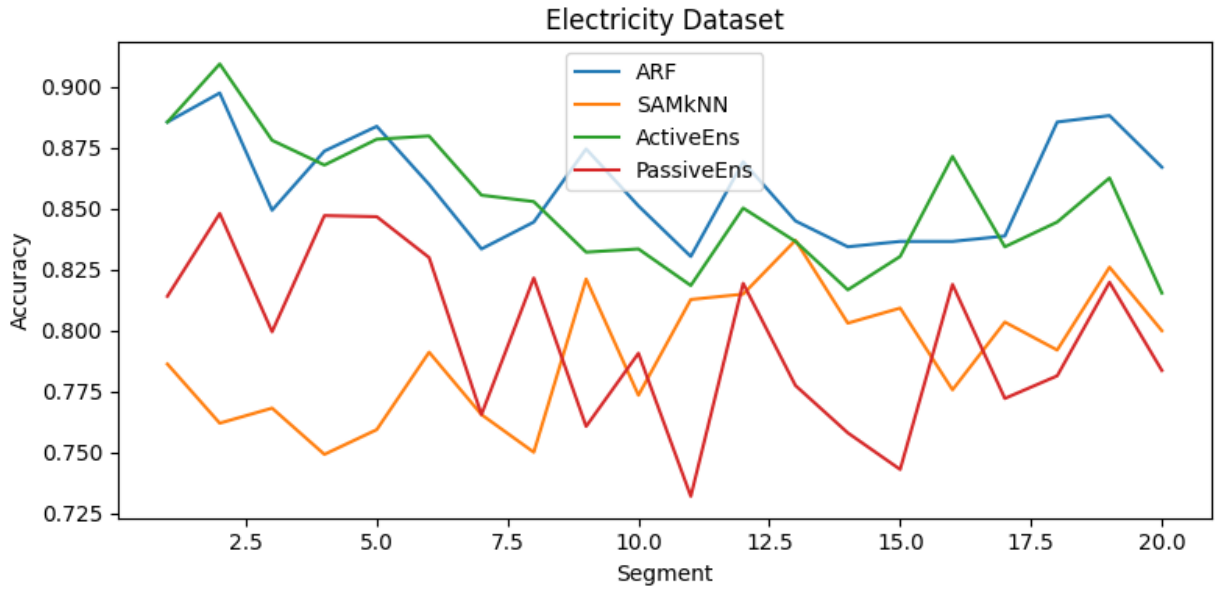

Figure 6: Accuracy plot for Electricity dataset for 20 segments and 500 window size.

The figures given below display the results for Electricity and Spam dataset for a segment size of 20, with a window size of 50, which was initially 100:

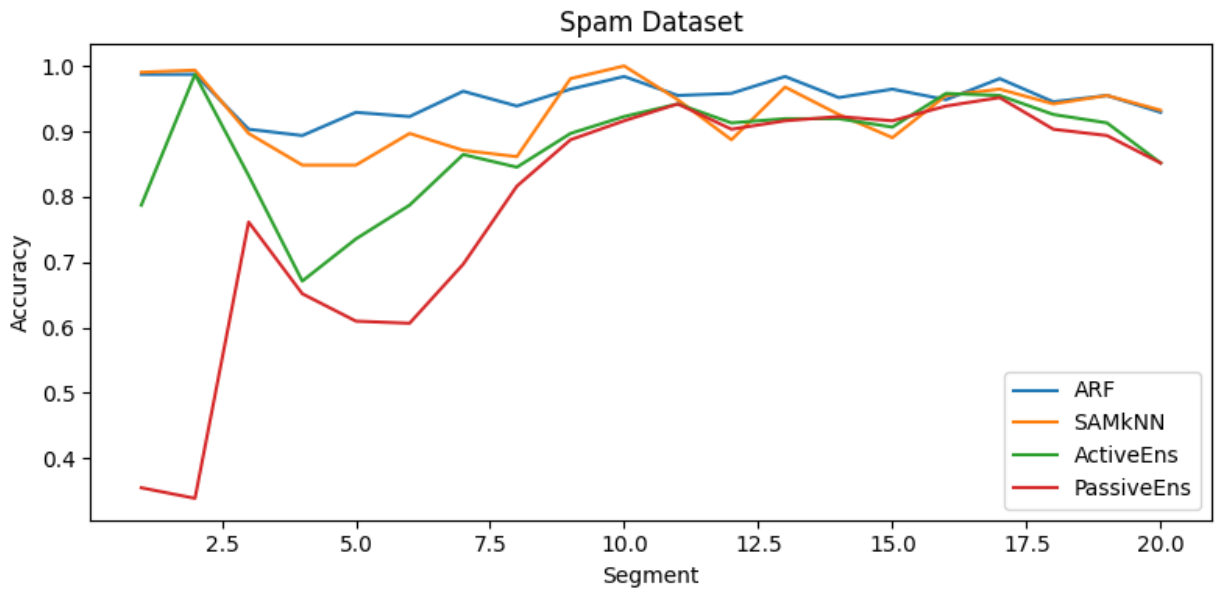

Figure 7: *Accuracy plot for Spam dataset for 20 segments and 50 window size.*

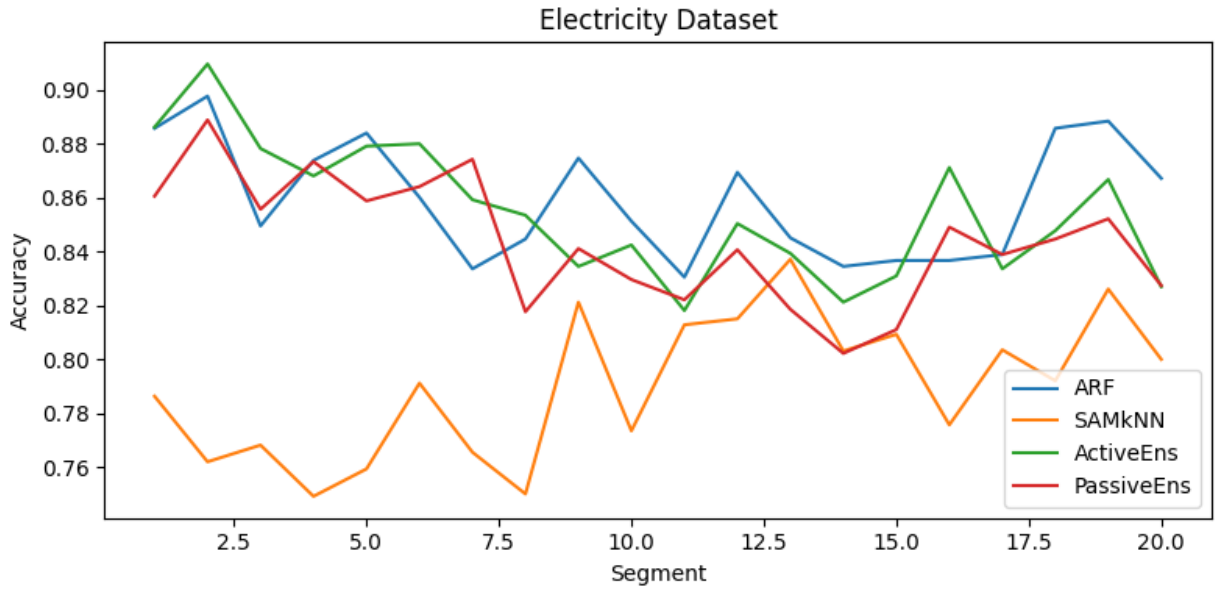

Figure 8: *Accuracy plot for Electricity dataset for 20 segments and 50 window size.*

With a 50-sample window, passive ensemble responds very quickly to changes but shows big swings in accuracy, such as sharp drops and spikes, because it treats random noise like a real shift. When we use a 500-sample window, the accuracy curve is much smoother and stays close to Adaptive Random Forest and Active Ensemble, but it takes two or three segments to bounce back instead of one. The same pattern appears on the Spam data: the small window causes extreme dips and overshoots, while the large window keeps the drop inside the hard segment and then recovers steadily to about 0.90–0.94 accuracy.

3) Compare the Active and Passive versions of your custom ensemble. Indicate clearly which version achieved higher accuracy. Discuss the observed results: Why do you think one ensemble outperformed the other? Consider aspects such as reaction speed to drift, false alarms, sensitivity to noise, and model stability when reasoning about the results.

The passive ensemble and active ensemble share the same pool of base classifiers and voting mechanism, but differ in when they update. The passive ensemble applies a test-then-train step on every incoming instance, ensuring it immediately absorbs new information. It also uses a sliding window of recent data to weight base learners, but training never pauses. The active ensemble, on the other hand, interposes a drift detector before updating: it tracks recent error rates and only retrains its base models when it detects a statistically significant drop in accuracy.

Overall accuracy comparisons reveal that passive ensemble slightly outperforms active ensemble on streams with abrupt, low-noise drifts. On the SEA and Agrawal synthetic datasets, passive ensemble recovered quickly after each known drift point that its final accuracy was about 0.2–0.3 percentage points higher than active ensemble. The same pattern held on the Spam dataset where drifts are more pronounced and less noisy. However, on the Electricity dataset active ensemble led by roughly 2.5 points.

Passive ensemble updates after every instance, so it reacts very quickly and recovers fast when the concept really shifts. However, in noisy data it can overreact to random fluctuations, causing sudden accuracy swings and extra model updates. On the other hand, active ensemble waits for its

drift detector to signal a real change before updating, so it may be a bit slower to adapt but avoids reacting to mere noise and keeps its accuracy more stable

Conclusion

To sum our findings up, The interleaved test-then-train evaluation we used in this assignment gives a clear view of how models learn from data that changes over time. We tested two custom ensembles, passive ensemble and ActiveEnsemble, along with Adaptive Random Forest and SAM-kNN on both synthetic streams (SEA and Agrawal) and real-world data (Spam and Electricity). Passive ensemble, which updates on every new example, recovered very quickly after sudden shifts. On the other hand, ective ensemble, which waits for its drift detector to signal a genuine change, stayed more stable when the data had noise or gradual trends. We also saw that the size of the sliding window affects results: small windows react fast but can be noisy, while large windows give smoother results but take longer to adapt.

References

[1] "Streaming data," Hazelcast, https://hazelcast.com/foundations/event-driven-architecture/streaming-data/ (accessed May 10, 2025).

[2] J. Brownlee, "A gentle introduction to concept drift in machine learning," MachineLearningMastery.com, https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning/ (accessed May 10, 2025).

[3] T. river developers, "DDM," River, https://riverml.xyz/0.11.0/api/drift/DDM/ (accessed May 10, 2025).

[4] T. river developers, "EDDM," River, https://riverml.xyz/0.13.0/api/drift/EDDM/ (accessed May 10, 2025).

[5] The Fellowship of Online Machine Learning, "ADWIN" River, https://riverml.xyz/dev/api/drift/ADWIN/ (accessed May 10, 2025).