Robust Ensemble Learning for Concept Drift and Adversarial Attack Mitigation

Bahadır Yüzlü
Bilkent University
Ankara, Turkey
bahadir.yuzlu@ug.bilkent.edu.tr

ABSTRACT

This paper investigates the application of ensemble models in data stream classification, with a focus on addressing challenges such as data drifting and adversarial attacks. Key concepts explored include data stream dynamics, built-in models, ensemble methodologies, and modifications to enhance model resilience. Through experimental evaluation, the study compares the performance of basic and modified ensemble models across diverse datasets, highlighting the effectiveness of enhancements in mitigating the impact of adversarial conditions. The findings underscore the importance of robust classification models capable of adapting to evolving data environments.

Keywords

Ensemble Learning; Concept Drift; Adversarial Attacks; Data Streams; Classification

1. Concepts

Data Stream: refers to a scenario where data flows continuously over time, requiring us to handle incoming data incrementally rather than having access to all data upfront. Unlike traditional batch learning, where all data is available at once for model training, data stream processing involves updating the model continuously as new data arrives in real-time.

Concept Drift: denotes a shift in the fundamental distribution of data over time. For instance, in a spam email detection model, the attributes of spam emails may evolve, necessitating the system to adjust and learn new spam characteristics accurately.

Adversarial attack: is presumed to occur with the intention of sabotaging the underlying classification system. This attack aims to induce the system to adapt to false or misleading data, ultimately compromising the accuracy and reliability of the classification process.

Prequential Evaluation: is also recognized as the interleaved test-then-train evaluation technique, emphasizing the real-time evaluation of model performance in dynamic data stream environments. Each incoming data instance is used to test the model before it is utilized to update the model itself.

Ensemble model: combines multiple individual models to boost accuracy, reduce overfitting, and improve overall robustness in machine learning tasks. They aggregate predictions from diverse models to make more reliable decisions compared to using a single model alone.

Built-in classifiers: are pre-implemented machine learning algorithms available in libraries like scikit-multiflow, enabling easy application of complex classification techniques without the need to develop them from scratch.

Concept Drift Detectors: are algorithms specifically designed to recognize changes in the statistical characteristics of data over time. By detecting concept drift, these algorithms enable timely adaptation and maintenance of model accuracy in dynamic environments.

Adversarial Attack Detectors: are tools that identify and counter attempts to deceive machine learning models. They use various techniques like statistical analysis, adversarial training, and ensemble methods to monitor model behavior and detect anomalies indicative of adversarial attacks.

2. Requirements

NumPy: used for efficient numerical operations and array manipulations essential for preprocessing data and performing mathematical computations in machine learning.

Pandas: provided powerful tool for working with structured data, including loading data from various file formats, cleaning and preprocessing datasets.

Scikit-multiflow: employed for handling streaming data, including generating synthetic streams, training classifiers on evolving data, and detecting concept drift to adapt models over time.

Scikit-learn: utilized for calculating accuracy scores for performance assessment.

Matplotlib: used to create diverse plots and visualizations for exploring diagnosing model behavior, and presenting insights effectively in machine learning models.

3. Dataset Preparation and Classifier Implementation Details

3.1 Dataset Preparation

For dataset preparation, the AGRAWALGenerator() and SEAGenerator() classes from scikit-multiflow were employed to generate synthetic data streams, each comprising 100,000 instances. These streams were intentionally subjected to drift points at 25,000, 50,000, and 75,000 instances to simulate changing data distributions. Additionally, real-world datasets such as Spam and Electricity datasets will be used for further experimentation.

To ensure future accessibility and ease of manipulation, the four datasets (two synthetic and two imported) were stored in Data Frames.

3.2 Implementations for Handling Concept Drift

3.2.1 Common Approach among the Models

The initial step involved creating the datasets. A loop algorithm was then utilized to iterate through the rows of these datasets,

effectively imitating a data stream. Within this loop, the interleaved test-then-train approach was applied: the feature part of each row was used to predict the row's label, and then testing was conducted. Following this, the model was updated using the partial_fit function.

For evaluation, a window size of 1000 was chosen. Accuracy was calculated for every 1000 rows and subsequently plotted. Additionally, a final overall accuracy was computed for the entire dataset. These plots facilitated an easy understanding of partial performance changes, while the overall accuracy provided a clear picture of the model's performance across the complete dataset.

3.2.2 Classification Model Training

Classification models such as Adaptive Random Forest, Streaming Agnostic Model with k-Nearest Neighbors (SAM-kNN), and Dynamic Weighted Majority were implemented using built-in classifiers on each dataset. The models' responses to drifts were observed using prequential accuracy mapping.

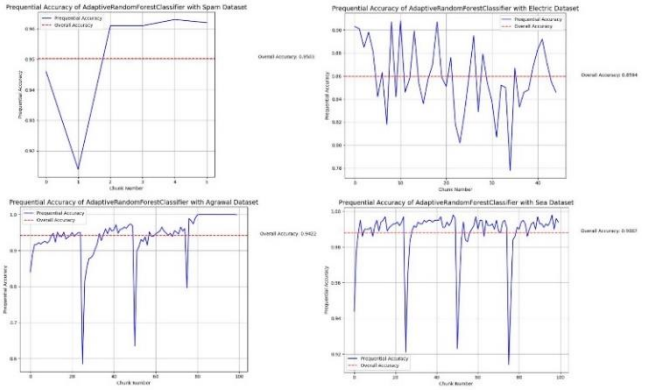

Figure 1. Adaptive Random Forest Classifier's Prequential Accuracy Plots

Adaptive Random Forest (ARF) is an ensemble model implemented in the Scikit-multiflow library, designed for data stream classification tasks. As an ensemble model, it combines multiple decision trees to enhance prediction accuracy, reduce overfitting, and improve robustness. The ARF classifier is particularly adept at handling evolving data streams and concept drift, making it a powerful tool for real-time data analysis.

On the Spam dataset, the ARF model performed exceptionally well. The overall accuracy and prequential accuracy graph were highly consistent, showing parallel trends. After processing the first 1000 samples, the model experienced a slight accuracy drop of approximately 3%. However, beyond this initial decrease, the model maintained an accuracy of around 96%, demonstrating its effectiveness and stability in this context.

For the Electricity dataset, the accuracy graph showed significant fluctuations, indicating potential adversarial attacks or inherent volatility in the dataset. Despite these fluctuations, the model achieved an overall accuracy of 85%, reflecting its resilience and ability to perform well under challenging conditions.

When applied to the synthetic Agrawal and SEA datasets, the ARF model demonstrated strong responsiveness to data drifts. In the Agrawal dataset, the model's prequential accuracy showed significant drops, approximately 25%, in response to each drift. However, these declines were instantaneous, indicating the model's rapid adaptation to changing data distributions. The SEA dataset also reflected the model's capability to detect and adjust to drifts effectively. Final accuracies was %94 and %98 for SEA and AGRAWAL datasets respectively. Overall, the ARF model exhibited excellent performance in sensing and adapting to data drifts, validating its utility in dynamic environments.

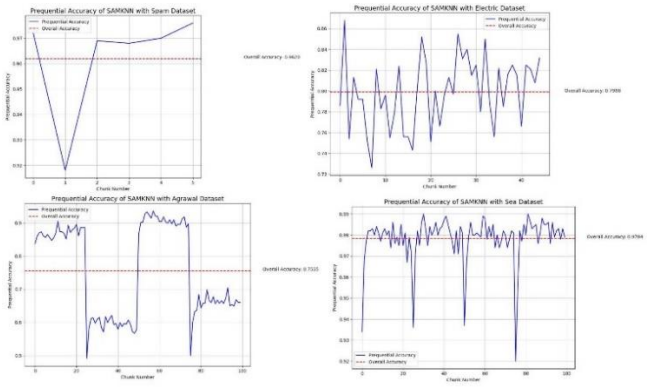

Figure 2. SAM-kNN Classifier's Prequential Accuracy Plots

The Streaming Agnostic Model with k-Nearest Neighbors (SAM-kNN) is a lazy learning algorithm available in the Scikit-multiflow library. Unlike eager learning algorithms, which build a model based on the entire training dataset before making predictions, lazy learning algorithms like SAM-kNN store the training instances and defer the computation until a query is received. This approach allows SAM-kNN to be highly adaptable to new data and evolving patterns, making it well-suited for data stream classification tasks.

On the Spam dataset, the SAM-kNN model exhibited a drop in accuracy after processing the first 1000 samples, similar to the ARF model. Despite this initial decline, the model maintained an overall accuracy of 96%, indicating its strong performance in recognizing and adapting to the characteristics of the dataset.

For the Electricity dataset, the accuracy graph displayed noticeable fluctuations, suggesting potential adversarial attacks or inherent volatility. Despite these fluctuations, the SAM-kNN model achieved an overall accuracy of 79%, showcasing its ability to handle challenging and dynamic data conditions.

However, the SAM-kNN model struggled with the synthetic Agrawal dataset. It lacked the ability to understand and adapt to drifts promptly, resulting in prolonged accuracy falls of around 25% during some drift periods. The model could not update itself effectively between the first and subsequent drifts, highlighting its limitations in dealing with abrupt changes in data distribution.

SAM-kNN model performed better on the SEA dataset. It successfully detected and adapted to the drifts, achieving an overall accuracy of approximately 94%.

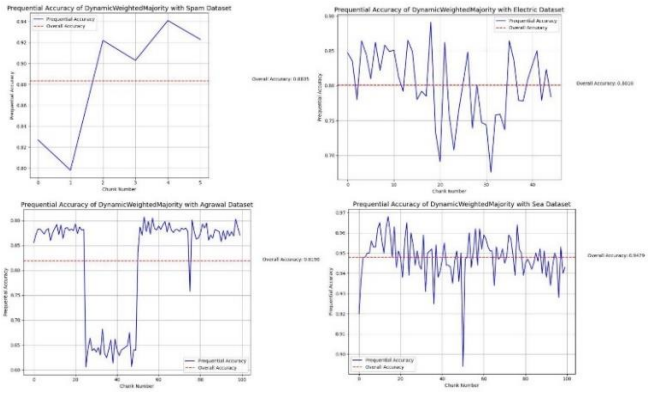

Figure 3. Dynamic Weighted Majority Classifier's Prequential Accuracy Plots

Dynamic Weighted Majority (DWM) is an ensemble model implemented in the Scikit-multiflow library. This algorithm combines multiple classifiers to enhance prediction accuracy and robustness, dynamically adjusting the weights of each classifier based on their performance over time. DWM is particularly effective for handling evolving data streams, making it a suitable choice for environments where data distributions change frequently.

On the Spam dataset, the DWM model experienced an accuracy drop after the first 1000 samples. After the second chunk of 1000 samples, the model became rather unstable, ultimately resulting in an overall accuracy of 88%. This indicates some difficulty in maintaining stability in this dataset.

For the Electricity dataset, the accuracy graph showed significant fluctuations, reflecting the volatility of the data. Despite these fluctuations, the model achieved a final accuracy of 80%, demonstrating its resilience but also highlighting the challenges posed by the dataset's dynamic nature.

In the Agrawal dataset, the model did not update itself effectively after the first drift but adapted well after the second drift. This adaptation resulted in an overall accuracy of approximately 81%, showcasing the model's ability to eventually learn from changes in the data distribution.

Interestingly, on the SEA dataset, the DWM model did not detect any drifts other than the second one. Despite this, it achieved an overall accuracy of 94%, indicating strong performance in this synthetic dataset and its ability to handle certain types of data drifts effectively.

The prequential performance of the models across different datasets revealed varying strengths and weaknesses in their ability to detect and adjust to drifts. The Adaptive Random Forest (ARF) consistently demonstrated strong adaptability to data drifts, particularly excelling in the synthetic datasets. In the Agrawal dataset, ARF showed significant drops in accuracy in response to drifts but quickly recovered, indicating its robustness. The SEA dataset also highlighted ARF's ability to maintain high accuracy despite the presence of drifts.

The SAM-kNN model exhibited good performance on the Spam and SEA datasets, maintaining high accuracy and effectively handling drifts. However, it struggled with the Agrawal dataset, showing prolonged accuracy falls during drift periods, indicating a limitation in dealing with abrupt changes.

Dynamic Weighted Majority (DWM) showed mixed results. It handled the SEA dataset well, achieving high accuracy without sensing most drifts, which might indicate stability in some contexts. However, its performance on the Spam and Electricity datasets was less stable, with noticeable fluctuations and a lower overall accuracy compared to the other models. In the Agrawal dataset, DWM took longer to adapt to the drifts, showing improved performance only after the second drift.

3.2.3 Ensemble Model Training

Besides using built-in models, ensemble models can also be developed from scratch. In this part, an implementation of a custom ensemble model is presented. This ensemble model comprises multiple HoeffdingTreeClassifiers and a single ADWIN drift detector. The principle behind this approach is to reset the classifiers whenever the ADWIN detector identifies a drift. This reset is applied to each base classifier model, minimizing the impact of the drift on the prediction mechanism. The final prediction is determined through a majority voting system among the base classifiers. This method ensures that the ensemble model remains robust and accurate, even in the presence of data drifts.

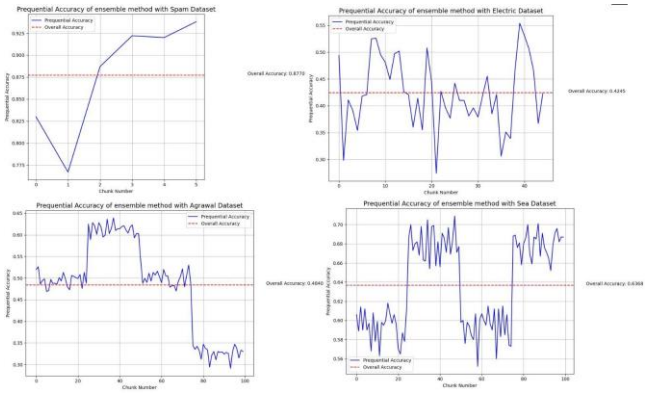

Figure 4. Drift Detection with Ensemble Classifier's Prequential Accuracy Plots

A custom ensemble model was implemented using five HoeffdingTreeClassifiers and a single ADWIN drift detector. The aim was to reset the classifiers whenever the ADWIN detector identified a drift, minimizing the impact on the prediction mechanism. Predictions were made using a majority voting system among the base classifiers.

On the Spam dataset, this custom ensemble model produced results similar to the built-in models, indicating its effectiveness in this context. However, performance significantly declined on the other three datasets. The ensemble model exhibited numerous jumps after drifts, resulting in overall low accuracies. This suggests a limitation in the base learners' ability to handle these datasets. Even when the number of base predictors was increased to ten, no significant improvement was observed, and computation time increased substantially.

Additionally, the model failed to detect the second drift in the Agrawal dataset and the second and third drifts in the SEA dataset. This indicates potential issues with the ADWIN algorithms drift detection capabilities or the reset handling method.

Experiments with different threshold values for ADWIN were conducted to investigate their impact on performance. Lowering the threshold made the model more sensitive to changes, resulting in more frequent resets and slightly improved drift detection.

However, this also led to increased instability and computational overhead. Raising the threshold reduced the number of resets, leading to smoother performance but at the cost of failing to detect some drifts, similar to the original results. These experiments suggest that the ADWIN algorithm's performance and the reset handling method need to be carefully tuned to balance sensitivity and stability for optimal results.

3.3 Implementations for Handling Adversarial Attack

3.3.1 Data Poisoning

Adversarial poisoning attacks were applied to the dataset to evaluate the robustness of the custom ensemble model. Specifically, two attacks were introduced: between the 40,000th and 40,500th data instances, 10% of the labels were flipped randomly, and between the 60,000th and 60,500th data instances, 20% of the labels were flipped randomly. For example, if a label was 1, it was transformed into 0, or vice versa.

3.3.2 Ensemble Model Modifying

To handle adversarial attacks, the ensemble model was modified to include detection mechanisms based on recent label patterns. The model keeps a log of the latest 50 labels, updating this log with each new label. By calculating the variance of these labels, the model can identify unusual patterns that might indicate an attack. Additionally, the model finds the most common label (mode) and tracks its frequency. If the mode count is less than 15 or the variance exceeds 0.1, the sample is flagged as potentially adversarial. When a potential adversarial sample is detected during the partial_fit function, it is dismissed, and the model is cleansed of the corrupted data. This ensures that predictions are made using a clean dataset, while training continues with the non-poisoned dataset, preventing the model from being influenced by poisoned data.

In addition predictions are made using the entire dataset, including both clean and potentially adversarial instances, but accuracy is calculated based solely on the non-corrupted labels.

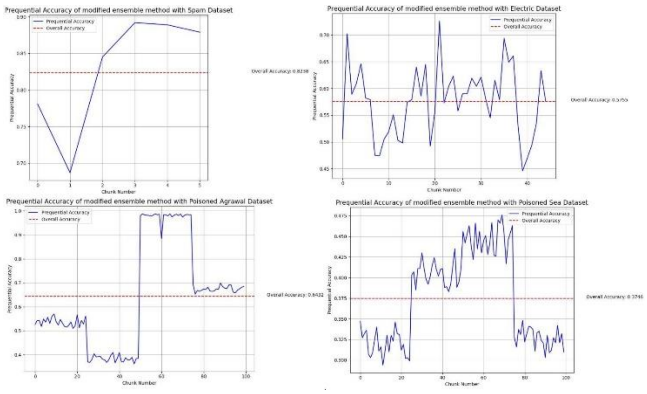

Figure 5. Modified Ensemble Classifier Prequential Accuracy Plots

Both the modified ensemble and the basic ensemble exhibited similar performance on non-attacked datasets. However, the modified ensemble showcased a greater overall accuracy of 57% compared to 42% for the basic ensemble. This difference can be attributed to the characteristics of the Electricity dataset, which displayed a volatile prequential accuracy throughout, indicating the likelihood of adversarial instances within the dataset. The modifications implemented in the modified ensemble were successful in identifying and mitigating these adversarial rows.

In the Agrawal dataset, the overall accuracy also improved from 48% to 56% with the modified ensemble. Interestingly, there was no significant difference in performance between the 10% and 20% flip attacks, suggesting that the modifications had a positive effect on diagnosing attacks and maintaining accuracy.

However, in the SEA dataset, the modification resulted in an overall accuracy decrease from 63% to 33%. It's important to note that this decrease was not related to adversarial attacks, as the attacked points were distant from the areas of accuracy decrease. Instead, the decline in accuracy can be attributed to the inherent characteristics of the dataset, indicating potential challenges unrelated to adversarial attacks.

3.4 Conclusion

In this assignment, the exploration of data stream classification and its associated challenges, including data drifting, formed the foundational concepts. The utilization of built-in models and ensemble models, with a focus on their resilience to adversarial attacks, served as the core of the investigation. The comparison between basic and modified ensemble models shed light on the effectiveness of modifications in enhancing model performance.

Overall, the assignment provided a comprehensive analysis of ensemble models in the context of data stream classification, emphasizing the significance of addressing adversarial conditions through modifications. By leveraging built-in classifiers and implementing ensemble techniques, the study contributed valuable insights into the development of robust classification models capable of adapting to dynamic data environments.

4. REFERENCES

[1] "API reference - pandas 2.2.2 documentation." pandas. Available: https://pandas.pydata.org/docs/reference/index.html. Accessed: May 2024.

[2] "scikit-multiflow." scikit. Available: https://scikit-multiflow.github.io/. Accessed: 2020.

[3] "NumPy reference - NumPy v1.26 Manual." NumPy. Available: https://numpy.org/doc/stable/reference/index.html. Accessed: May 2024.

[4] "Matplotlib." Visualization with python. Available: https://matplotlib.org/. Accessed: May 2024.

[5] Korycki, Ł., & Krawczyk, B. (2023). "Adversarial concept drift detection under poisoning attacks for robust data stream mining." Machine Learning, 112(10), 4013-4048.