Concept drifts and their impact on ensemble classifiers: an analysis

Kacper Pieniażek

Wrocław University of Technology

Keywords: data streams · classifiers · ensemble learning · concept drift

1 Introduction

As amount of data being sent increases, stream data processing is becoming more and more important. Computer systems and devices are gathering and sharing useful data at any time to the point that it becomes impossible to store all of it. That creates a need for online data processing and extracting useful knowledge without storing all underlying information. Looking into classifiers, those systems need to be able process vast amounts of data quickly and immediately, as they won't be able to look back

Processing of data streams comes with its' share of problems [4] that need to be solved for proper data classification, which will be discussed further. Compared to traditional machine learning for static data, streams are constant and ever-changing. Classification models need to be able to adapt to changes in data distribution in order to do accurate predictions and they need to classify on the fly. Such models can use different methods to process data either one-by-one, or in batches.

For classification problem, there has been an increasing interest in ensemble learning [20]. This field of research is expanding rapidly along with data stream classification and concept drift detection.

The goal of this literature review is to provide an analysis of ensemble learning methods for data streams classification. We aim to present a detailed overview of the various methods used in ensemble learning, and examine their suitability in context of data streams. Regarding data streams, this review will focus on problem of concept drifts, which will be also shown to be vital in this area of classification. First we will look into underlying theory to understand problems that will be tackled in later stages, when different methods will be reviewed.

2 Concept drift in data streams

Distribution generating data can change over time, and as statistical properties of input data change, it can lead to degradation in model accuracy. Such process is called concept drift [19] [18], which can happen for a number of reasons. One example is spam detection; what is considered spam or not is very specific and up to the user. User preferences on what is considered spam can change overtime without any possibility to predict it.

As in the example above, it is assumed concept drifts are unpredictable; if some change in distribution can be forseen, model can be prepared and trained for it ahead of time. Concept drifts also take place whenever data stream is not certain in terms of its' distribution. This leads into two separate research areas:

– Stationary/static data streams, where it is assumed that concept drifts can be ignored,
– Non-stationary/evolving data streams, where concept drift is a key issue to be solved.

This paper will focus on non-stationary streams, as one of our goals is to review methods of handling concept drifts. It's also more realistic approach to data streams, and quite important in terms of building accurate models; data streams are expected to have unstable data distribution over time [14]. It's worth noting that concept drift can cause a major drop in classification accuracy, as with each change in underlying data distribution, pre-trained model's knowledge of a source can be deprecated [3]. Because it all happens in data streams, re-building whole model whenever it's not accurate is not viable.

3 Concept drift detection

Concept drift detection methods are measured [7] within given criteria:

– Amount of real detections,
– Amount of false detections,
– Time between drift occurence and detection

These metrics result from the underlying nature of data streams. Model accuracy can not get worse then given threshold, so drift detection must happen as soon as possible, before it leads to meaningful drop in accuracy. Similarly, false detection can lead to unnecessary re-training processes and loss of resources. There are methods that allow to measure balance between those metrics in order to find best solution for given problem [1].

Dedicated concept drift detection methods can be used as a separate algorithm that informs classifier at the time of drift detection of upcoming changes [2]. Model can then gather new data needed for re-training process to adapt to the new state of data source. It's important such method informs classifier to prepare ahead of time, so it has time to gather data before its' accuracy goes below allowed threshold. Some classifiers can detect concept drift on their own, or even rebuild when it's needed without explicitly detecting such drifts.

4 Data stream classifiers

Data stream classification techniques can differ based on how they parse data, how they handle concept drifts or even based on their recommended use cases.

Concept drift can heavily affect classifier's accuracy, so models need to adapt to ever-changing environment. We will specifically look into ensemble learning methods, which often use single-classifier methods in their base learners. Ensemble learning uses multiple classifier models and pipes their output into combining function, resulting generally in more certain label. Ensemble learning techniques have efficient ways of handling concept drifts, or changes in data source in general. This paper will review chosen classifier methods alongside their expected performance within non-stationary data streams.

Data stream ensemble classification methods can be divided into different categories, specifically we can distinguish such categorizations in scope of non-stationary data streams [7]:

- Approach to data processing:
  - On-line learning models,
  - Chunk-based learning models.
- Concept drift detection:
  - Active,
  - Passive.
- Different aggregation and base component updating methods.

5 Ensemble learning for data streams

Ensemble learning methods can be grouped based on four basic strategies [8] of reconstruction. Those strategies are needed, because our models need to be adapting to changes in data stream. They lead to high performance and accuracy of the ensemble, as they do not require re-building whole structure and they allow to adapt model to new environment by making changes in base components. We can define those strategies as:

- Adapting combination phase,
- Updating training data,
- Updating base classifiers,
- Updating structure of the ensemble

Adapting combination phase can be used to assign weights to each base classifier, for example based on their performance on most recent data. That way classifiers that statistically have the best performance will have the most influence on the output. Similarly, combination function itself can be modified to account for changes in data distribution.

Strategy of updating training data can be used to construct base classifiers based on newest changes in the data source. This way new or re-constructed classifiers learn on new data instead of deprecated one, as we assume classifiers to work within evolving data stream.

Base classifiers can be updated over time in order to adapt to changes in data stream. It's worth noting that it does not require the general model to detect or be informed of concept drifts; Whenever accuracy is getting worse, it can be re-trained based on new data to consider newest changes in its' distribution.

Structure of the ensemble can be updated by adding or removing base classifiers. As data stream changes, performance of a single classifier can drop drastically. Whenever it's not performing above certain threshold, it can be removed. Similarly, new classifiers can be added to boost general performance of the ensemble.

These four strategies can be used interchangeably to maintain high performance; some of them will be discussed further as we dive into specific methods. Some ensemble learning methods do not handle concept drifts on their own, but they can be used alongside dedicated detection methods to keep up with their performance. Generally ensemble methods have often been used to handle concept drifts in online data sources [9].

6 Some ensemble learning methods

Ensemble learning methods were chosen based on different strategies and techniques they use in order to handle their inner structure of base classifiers, drift detection methods and based on how they work with streamed data.

- SEA (Streaming Ensemble Algorithm) [13]
- AWE (Accuracy Weighted Ensemble) [16]
- KBS (Knowledge-Based Sampling) [12]
- DWM (Dynamic Weighted Majority) [6]
- ACE (Adaptive Classifiers-Ensemble) [10]

This research is going to utilize some of these ensembles and compare them against each other.

7 Experiments aims

The aims of the experiments are to investigate and compare ensemble classifiers under concept drifts. The goal is to evaluate the effectiveness of ensemble techniques and how concept drifts influence given ensemble compared to their base classifiers. We will also try to utilize ACE-like [10] caching mechanisms to improve general ensemble efficiency.

8 Research questions

This research will dive into ensemble classification and drift-related performance. It aims to provide insights into the effectiveness and potential limitations of ensemble methods. Formulated questions provide insight on how well various combinations of those perform.

- **Effectiveness** - How does the ensemble classification technique perform compared to a baseline model?
- **Resilience** - Which method shows best resilience to concept drifts?

– **Usefulness** - How does the dedicated drift detection method contribute to the ensemble's ability to detect and adapt to concept drifts?
– **Drift-awareness** - What impact do different concept drift patterns and frequencies have on tested models?
– **Performance** - What are the trade-offs between the ensemble's adaptability to concept drifts and its computational requirements?

9 Datasets

Considering this research relies heavily on concept drifts, provided real-world datasets should contain such drifts, or at least data should be manipulated so that drifts are synthetic. The other approach is to utilize synthetic data sets such as SEA generator [13], that provide unique data generation scheme.

In this experiment a combination of real-world and synthetic data sets will be used. All of them are publicly available.

– **SEA generator** [13] - This dataset consists of three-dimensional data with all three attributes in range from 0 to 10. It has two decision classes and besides data, it provides sudden concept drifts,
– **Hyperplane** - Another synthetic dataset generator used in other research papers [17]. It provides gradual concept drifts and each instance is described by 10 features and 2 decision class values,
– **Electricity** - real-life dataset, contains 45312 instances with 8 input attributes, recorded every half an hour for two years from Australian New South Wales Electricity. Classification predicts either a rise or fall in the electricity price. Concept drift may happen because of changes in consumption habits, seasonality, unexpected events; hence we assume that both gradual and suddent concept drifts can happen,
– **Forest CoverType** - real-life dataset with 581012 instances classifying 7 forest cover types. It contains 54 attributes.

Each of synthetic data streams will also be piped through a noise generator to randomly differentiate the instances.

10 Scope of the research

The scope of this research is to investigate ensemble classification with concept drifts and assess the effectiveness of ensemble techniques combined with dedicated drift detection methods. The research aims to provide insights into the performance, adaptability, and limitations of ensemble models in dynamic environments.

This research will focus on the following aspects:

– Ensemble classification - This research will explore different ensembles that combine base classifiers to improve accuracy. The effectiveness of these ensemble methods in handling concept drifts will be evaluated,

- Concept drift detection - Dedicated drift detection methods will be integrated into the ensemble models to detect the occurrence of concept drifts. This research will investigate different methods that identify changes in the data distribution. It will be also evaluated how do such methods influence underlying ensemble classifiers.
- Performance evaluation - Performance of ensemble models will be evaluated in therms of different measures, such as accuracy, sensitivity or G-Mean.

Classifiers will be measured using most popular metrics [7], such as:

- **Accuracy** - the proportion of all correct predictions to the total number of examples,
- **Sensitivity** of the class of interest,
- **G-Mean** - the geometric mean of sensitivity, often applied on class-imbalanced data streams. Even though this research does not focus on imbalanced streams, some datasets are expected to be imbalanced,
- **Update time** - Amount of time that a classifier requires to update its structure and accommodate new data from the stream. Ideally, update time should be lower than the time of next chunk/object arrival,
- **Decision time** - Time it takes to make a decision on new instances coming from the stream.

11 Experiment loop

To investigate the influence of concept drifts on data stream classifiers, the following experiment loop will be employed:

- Firstly, data sets used in an experiment will be thoroughly analysed to understand underlaying stream structure and how and when concept drifts occur,
- Single classifiers such as Naive Bayes and Multi-layer Perceptron will be analysed and treated as baseline models, where every other result will be compared against them. The goal is to present benefits of using ensembles over single classifiers in such environment,
- Some of the ensemble classifiers will be tested to measure differences between implementations of different methods,
- Custom implementation of an ensemble will also be tested to analyse how specific features influence experiment results.

12 Environment of experiments

The experiments will be conducted using Python 3 [15] with SciKit [11] libraries for building, training and running simulations. Matplotlib [5] will allow for data visualization and further analysis of experiments' results. Additionally, NumPy and Pandas are widely used for data manipulation and processing. All dependencies needed for those libraries to work properly are also provided in the environment.

13 Results

This section presents some of the results of an experiment that was led to analyse the influence of concept drifts on data classification. First, we're gonna look at performance of single classifiers and then dive into ensemble environment.

13.1 Data analysis

Each data source was thoroughly analysed in terms of it's classes balance, frequency, correlations and concept drifts. The goal was to understand the underlying structure and flow of each stream, so it's easy to understand what each classifier deals with in terms of predictions and calculations. Measurements and plots can be found in the source code for this experiment.

For classification tasks it's especially important to note that both SEA and Hyperplane streams provide balanced data, so each label should be equally distributed throughout experiment. This is not true for CoverType stream, which has a heavy imbalance of classes.

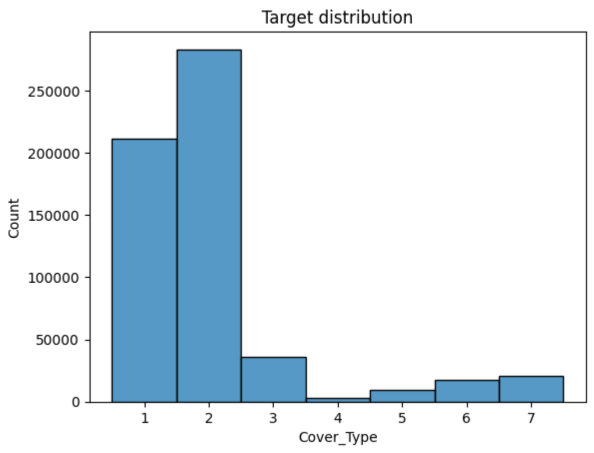


Fig. 1. Covertype stream target distribution

Considering this imbalance, it's important to look into "timeline" of said stream, so we understand what to expect for example from fully fitted models, that are only trained on first parts of the stream. As can be seen on a provided plot, there's quite a big gap at the start with most of the target labels, which means that there will be most likely very weak metrics for said labels if a classifier is trained and tested on the first half of provided stream.

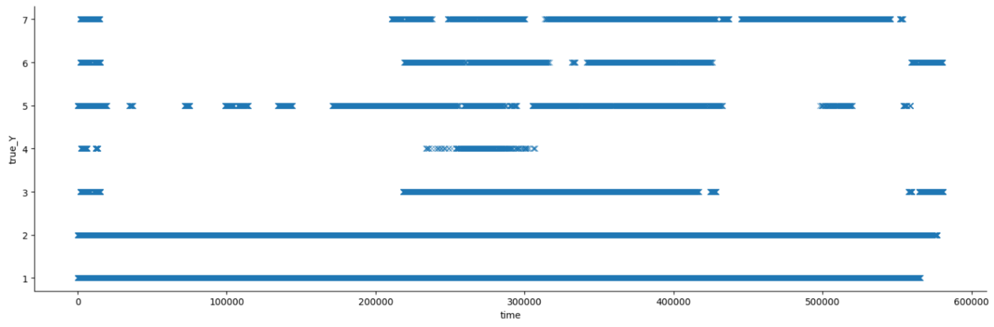


Fig. 2. Covertype target timeline

13.2 Single classifiers

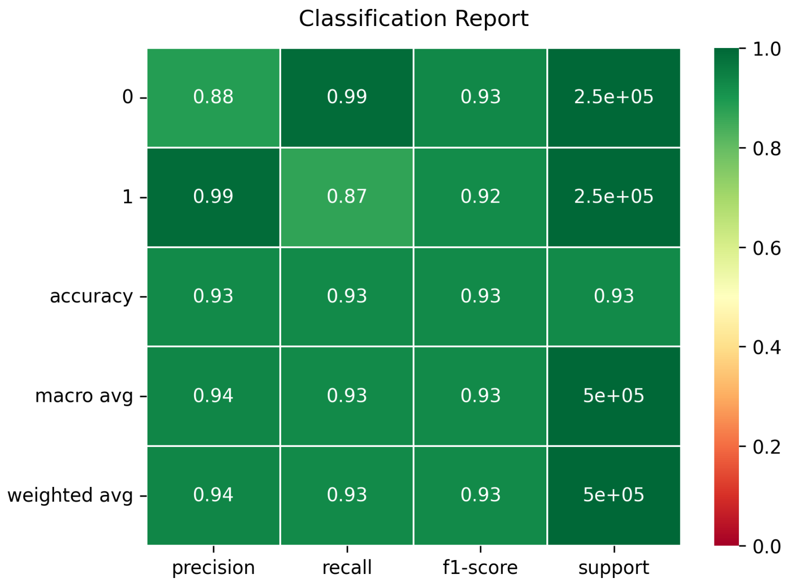
Single classifiers were tested in three ways, each time yielding slightly different results:

- Classifier was fully trained on the starting batch of data, tested on the rest,
- Classifier was being partially fit at each step of stream data,
- Classifier was being partially fit at each step and completely rebuilt if drift occured.

The metrics of single classifiers are unstable, meaning that starting parameters and random values at classifier initialization heavily influenced later progress, so the results were heavily varied, depending on the iteration. This means that even if the results could be considered good in terms of performance, they were not at all reliable, as next run would turn out to have completely different outputs.

Naive Bayes was used as a first classifier and compared against MLP to see if and how classifier metrics change. It was the fastest of all presented methods and the results were also not quite as bad as originally expected by the author. When looking at the results, one can easily notice when this classifier was at its' best.

Classification yielded best results for SEA stream (with generated drifts) with almost every metric above 90%, which means that Naive Bayes is a quite dependable classifier for data provided by this stream. If we compare it to metrics from an experiment led with no concept drifts on SEA stream, we can see that even though specific labels' metrics change, overall efficiency does not change. This means that Naive Bayes is resistant to concept drifts in SEA stream.

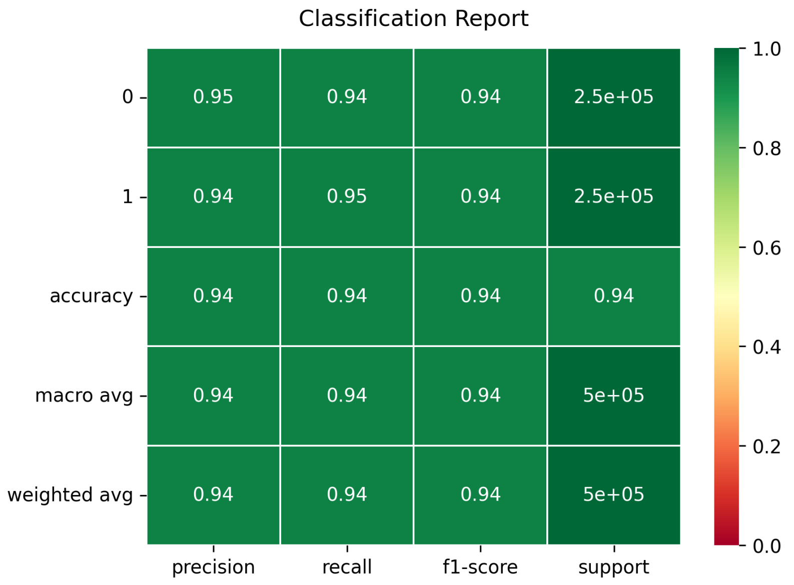


Classification Report

	precision	recall	f1-score	support
0	0.88	0.99	0.93	2.5e+05
1	0.99	0.87	0.92	2.5e+05
accuracy	0.93	0.93	0.93	0.93
macro avg	0.94	0.93	0.93	5e+05
weighted avg	0.94	0.93	0.93	5e+05

1.0
0.8
0.6
0.4
0.2
0.0

Fig. 3. Naive Bayes performance on SEA stream w/ drifts



Classification Report

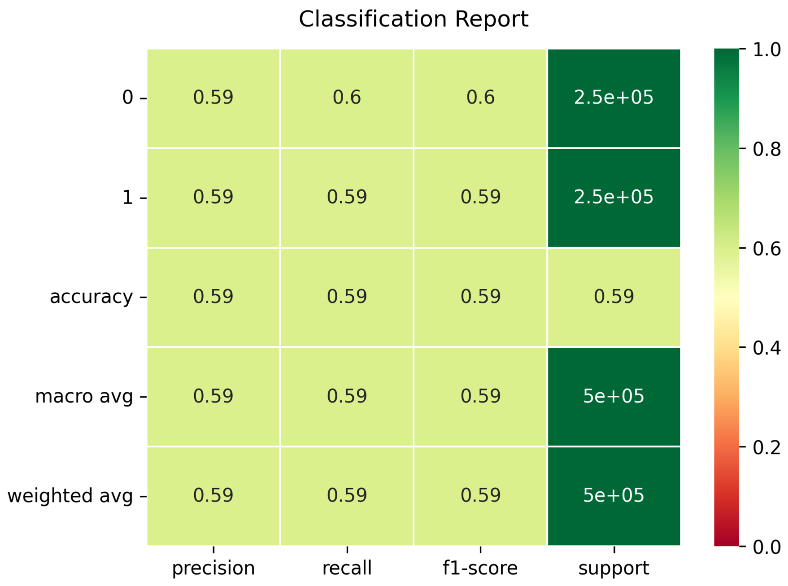
	precision	recall	f1-score	support
0	0.95	0.94	0.94	2.5e+05
1	0.94	0.95	0.94	2.5e+05
accuracy	0.94	0.94	0.94	0.94
macro avg	0.94	0.94	0.94	5e+05
weighted avg	0.94	0.94	0.94	5e+05

1.0
0.8
0.6
0.4
0.2
0.0

Fig. 4. Naive Bayes performance on SEA stream w/out drifts

This result seems to make further research on SEA redundant, as now it's expected to see high efficiency on every other classifier. This is especially true for ensemble classifiers that utilize Naive Bayes as a base classifier; if one NB can yield such results, multiple NB will be only more stable.

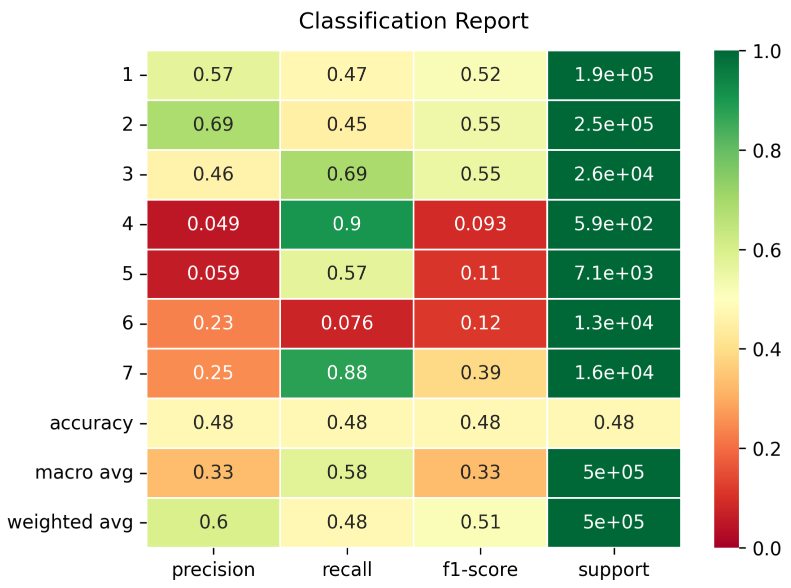
To consider why Naive Bayes has such good results on SEA stream, we need to look at the provided data. SEA stream has only three features (two that are actually relevant to the classification task) and two labels. Its' classification functions do not heavily change the results as well, so when concept drifts is generated for SEA stream, even though it's a sudden drift change, the results are often not that different. As we can see, Naive Bayes can handle such changes well if the classification task is small enough, so it's expected to see the metrics become much worse for both Hyperplane and Covertype streams, as they introduce more features, labels and "heavier" concept drifts.



	precision	recall	f1-score	support
0	0.59	0.6	0.6	2.5e+05
1	0.59	0.59	0.59	2.5e+05
accuracy	0.59	0.59	0.59	0.59
macro avg	0.59	0.59	0.59	5e+05
weighted avg	0.59	0.59	0.59	5e+05

Fig. 5. Naive Bayes performance on Hyperplane stream

As can be seen above, Naive Bayes does much worse for Hyperplane, which introduces more features. The classification task becomes more difficult, so when concept drifts occur, it's becoming much harder for a classifier to guess a correct answer based on previous, deprecated data it was trained on.

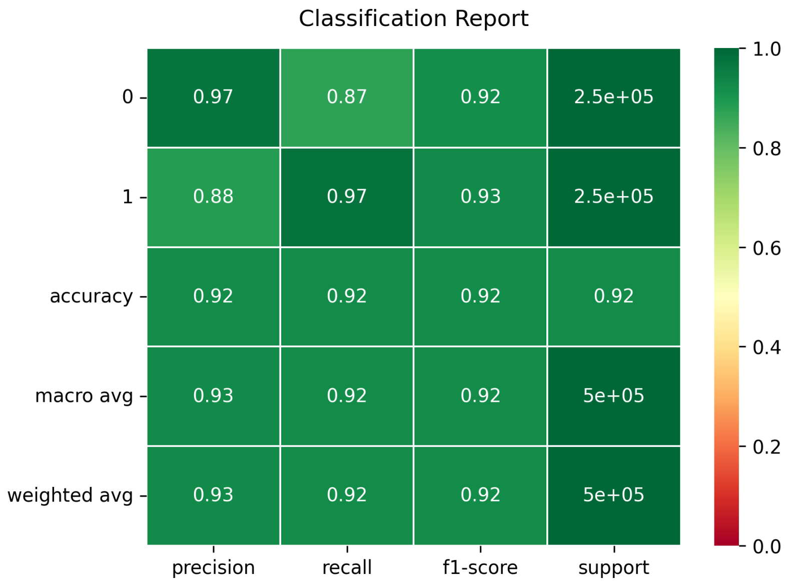


	precision	recall	f1-score	support
1	0.57	0.47	0.52	1.9e+05
2	0.69	0.45	0.55	2.5e+05
3	0.46	0.69	0.55	2.6e+04
4	0.049	0.9	0.093	5.9e+02
5	0.059	0.57	0.11	7.1e+03
6	0.23	0.076	0.12	1.3e+04
7	0.25	0.88	0.39	1.6e+04
accuracy	0.48	0.48	0.48	0.48
macro avg	0.33	0.58	0.33	5e+05
weighted avg	0.6	0.48	0.51	5e+05

Fig. 6. Naive Bayes performance on CoverType stream

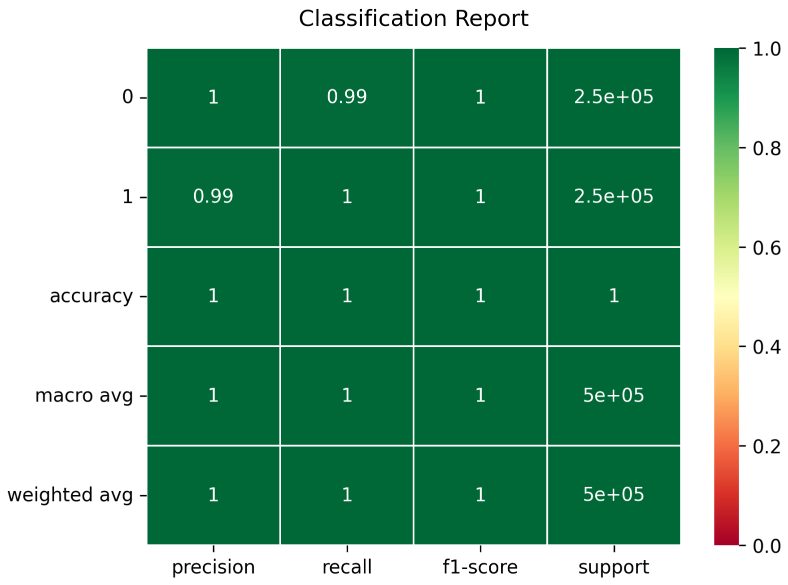
Finally, classification on CoverType stream yields extremely unreliable results. This data contains some imbalanced classes, but generally classes 1 and 2 are the most common and appear throughout the stream. So when NB was trained on old data, concept drifts that occur change classification task enough, so that classifier measures 50% accuracy at best.

Multi-layer Perceptron was trained using default hyperparameters as an alternative to Naive Bayes for classification tasks. As was previously shown, NB classifier has good results for SEA stream, and as expected, so does MLP. With no concept drifts, it had almost perfect score and the results under concept drifts were similar to what was measured in NB. This makes further research on SEA stream redundant, because it is expected for ensemble classifiers to do at least as well as its' base classifiers. With no well-visible variances in received results, SEA stream will be ignored in further research.



	precision	recall	f1-score	support
0	0.97	0.87	0.92	2.5e+05
1	0.88	0.97	0.93	2.5e+05
accuracy	0.92	0.92	0.92	0.92
macro avg	0.93	0.92	0.92	5e+05
weighted avg	0.93	0.92	0.92	5e+05

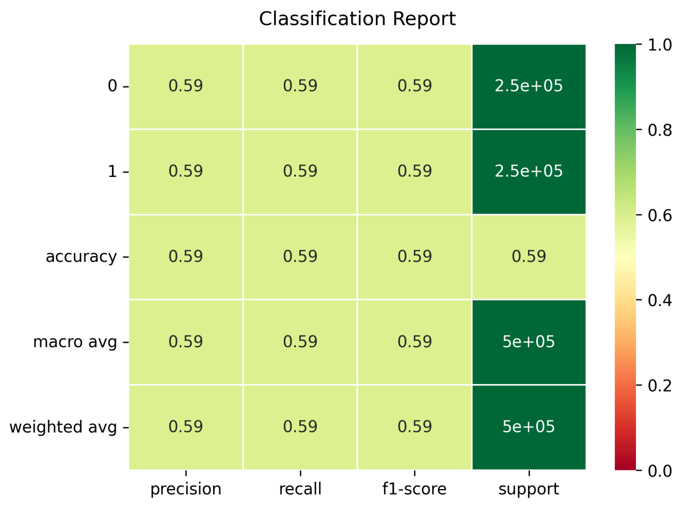
Fig. 7. MLP performance on SEA stream w/ drifts



	precision	recall	f1-score	support
0	1	0.99	1	2.5e+05
1	0.99	1	1	2.5e+05
accuracy	1	1	1	1
macro avg	1	1	1	5e+05
weighted avg	1	1	1	5e+05

Fig. 8. MLP performance on SEA stream w/out drifts

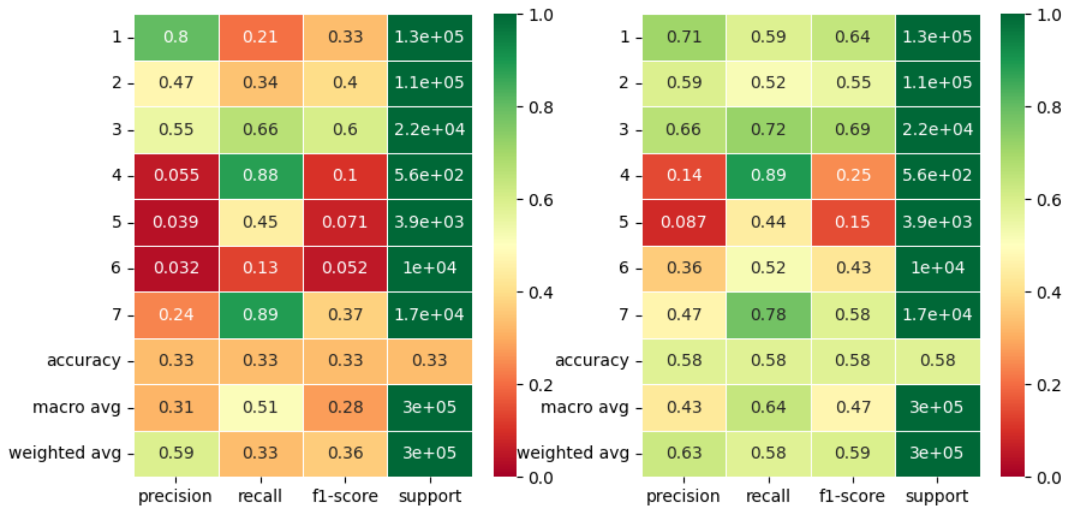
For Hyperplane stream, MLP has similar results to the previous classifier. This seems quite interesting, because as can be seen below, results for Covertype stream are much more different. This leads to an assumption that MLP handles such drifts similarly as NB.



	precision	recall	f1-score	support
0	0.59	0.59	0.59	2.5e+05
1	0.59	0.59	0.59	2.5e+05
accuracy	0.59	0.59	0.59	0.59
macro avg	0.59	0.59	0.59	5e+05
weighted avg	0.59	0.59	0.59	5e+05

Fig. 9. MLP performance on Hyperplane stream

Additionally, both NB and MLP were trained and directly compared on an experiment loop. MLP in general provided more stable and more accurate results, so even though it was trained only on first batches of stream data, it was able to figure out some relations that were at least partially maintained throughout concept drifts and data imbalance. Considering previous metrics for Hyperplane, data imbalance and amount of features on possible correlations seem to be what influences this difference in the most part.



	precision	recall	f1-score	support
1	0.8	0.21	0.33	1.3e+05
2	0.47	0.34	0.4	1.1e+05
3	0.55	0.66	0.6	2.2e+04
4	0.055	0.88	0.1	5.6e+02
5	0.039	0.45	0.071	3.9e+03
6	0.032	0.13	0.052	1e+04
7	0.24	0.89	0.37	1.7e+04
accuracy	0.33	0.33	0.33	0.33
macro avg	0.31	0.51	0.28	3e+05
weighted avg	0.59	0.33	0.36	3e+05

	precision	recall	f1-score	support
1	0.71	0.59	0.64	1.3e+05
2	0.59	0.52	0.55	1.1e+05
3	0.66	0.72	0.69	2.2e+04
4	0.14	0.89	0.25	5.6e+02
5	0.087	0.44	0.15	3.9e+03
6	0.36	0.52	0.43	1e+04
7	0.47	0.78	0.58	1.7e+04
accuracy	0.58	0.58	0.58	0.58
macro avg	0.43	0.64	0.47	3e+05
weighted avg	0.63	0.58	0.59	3e+05

Fig. 10. NB and MLP comparison

13.3 Interleaved evaluation

Both NB and MLP classifiers were evaluated on Covertype stream using interleaved test-than-train method. For the first half of an experiment, both classifiers seem to be equally efficient. Later results start to deteriorate for both of them, recall and F1 seems to especially drop for MLP classifier. This might be due to overfitting, as around 200,000 objects were used for fitting at this point. Rebuilding the classifier once its' results drop this heavily could improve overall metrics. In fact, that's what some ensembles are doing. If we were to stop training model and replace it with one trained only on new batch of data, it is expected to notice a general improvement in measurements and no drops sudden drops in mean values, such as ones shown below.

13.4 Ensemble classifiers

As expected, ensemble classifiers took much longer time to evaluate. They're doing much more than single classifiers; not only do they predict and fit using multiple base classifiers, but they also handle some ensemble-related logic, such as weighting base classifiers or calculating voting mechanisms.

For base classifiers, Naive Bayes classifiers were chosen due to their simplicity and efficiency. Ensembles are going to utilize up to 10 base members, so it's important to consider classifiers that will be fairly quickly trained and rebuilt if needed.

The metrics of such classifiers are expected to be better and the variance of results is expected to be much lower; ensembles utilize multiple base classifiers, so the end result is usually more stable.

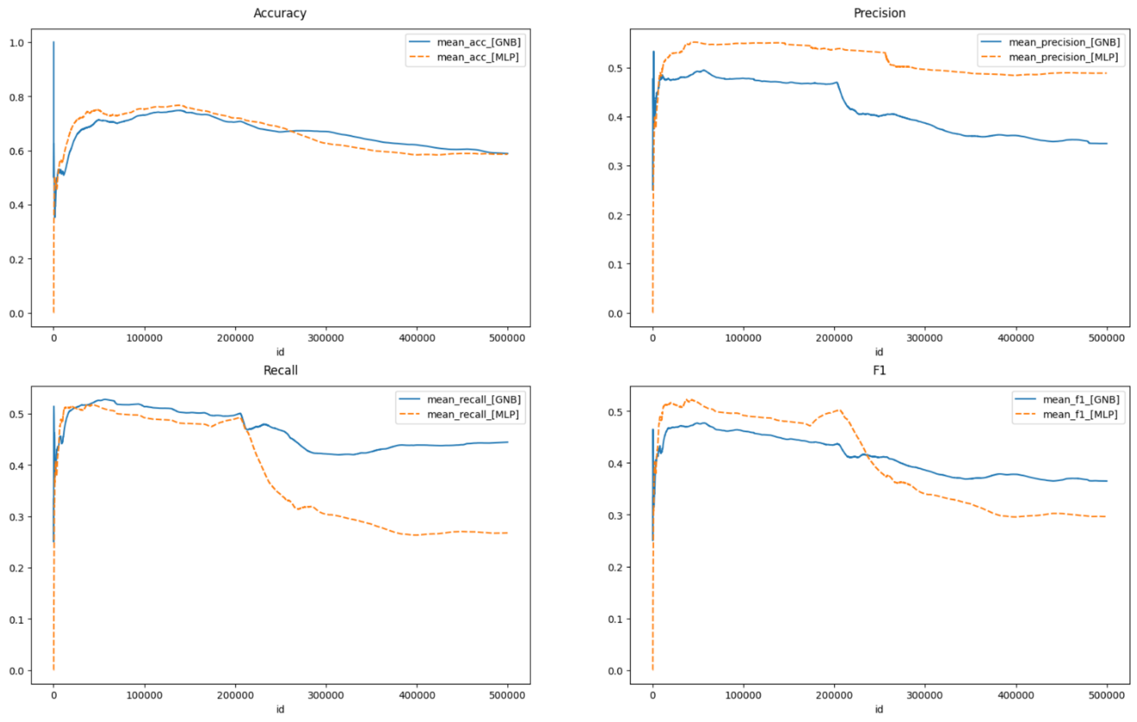


Fig. 11. NB and MLP comparison on Covertype stream

Accuracy Weighted Ensemble does, as expected, much better for Hyperplane stream. Where measured single classifiers had efficiency around 60% at most, which is almost as good as just blindly guessing, AWE has all metrics above 90%. As described previously, AWE utilizes Naive Bayes base classifiers and utilizes some memory cache to store, update and replace classifiers depending on what action seems the best to the algorithm. We are able to get this accuracy even though concept drifts occur throughout whole stream, because classifiers are replaced with ones trained on new data if their accuracy becomes undependable.

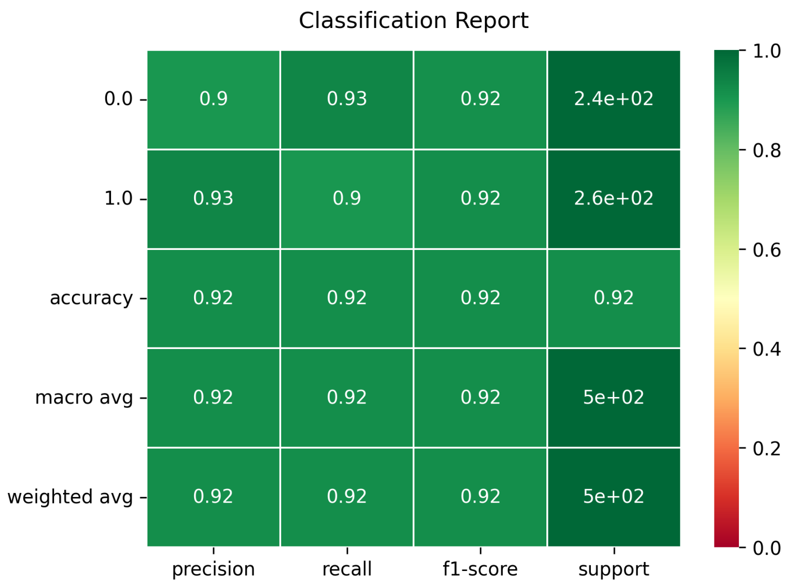
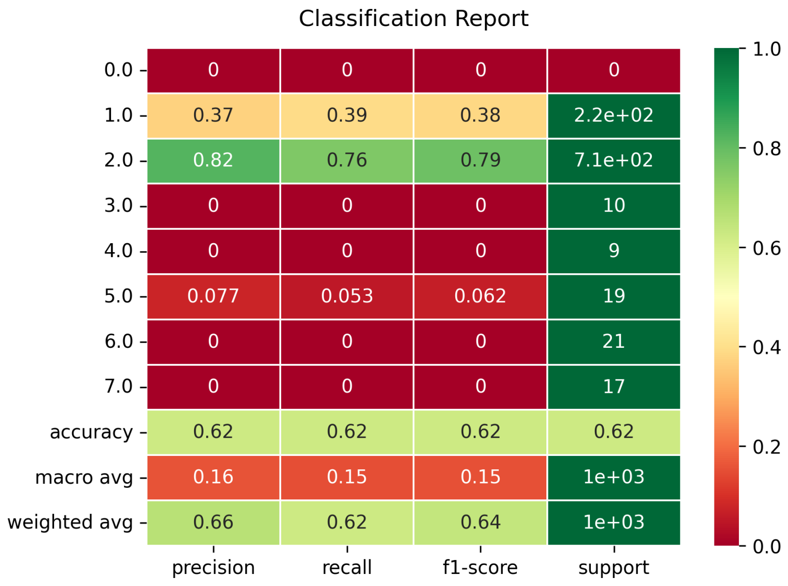


Fig. 12. AWE performance on Hyperplane stream

If we look at AWE classification report for Covertype, it looks much worse than previous plot. It's worth noting that all labels that have few representative objects did not get labeled properly at all. Big issue here is data imbalance, as it can't be measured as a reliable result if we only have around 10 objects tested for these labels. Labels 1 and 2 have most of the objects labeled and while label 2 has fairly good metrics, label 1 is not doing so good. Before drawing any conclusions from that, let's look at how other ensemble algorithm was evaluated on the same data.


Classification Report

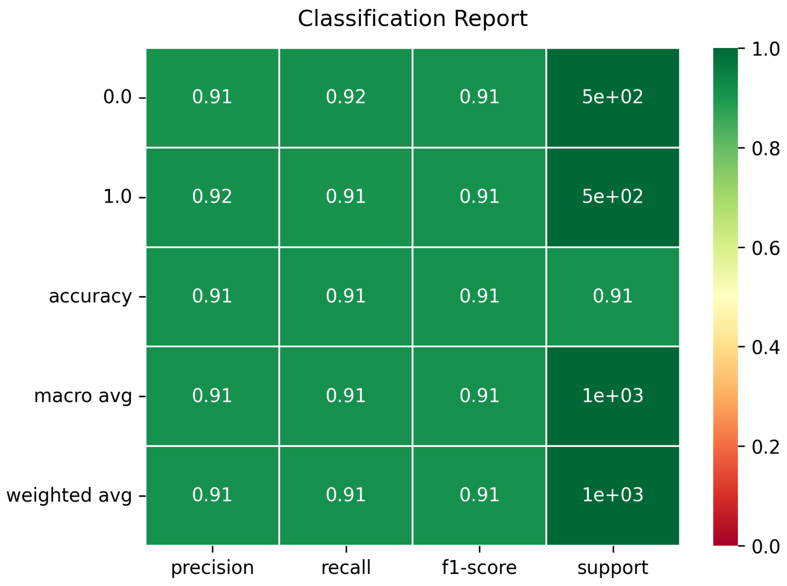
	precision	recall	f1-score	support
0.0	0	0	0	0
1.0	0.37	0.39	0.38	2.2e+02
2.0	0.82	0.76	0.79	7.1e+02
3.0	0	0	0	10
4.0	0	0	0	9
5.0	0.077	0.053	0.062	19
6.0	0	0	0	21
7.0	0	0	0	17
accuracy	0.62	0.62	0.62	0.62
macro avg	0.16	0.15	0.15	1e+03
weighted avg	0.66	0.62	0.64	1e+03

1.0
0.8
0.6
0.4
0.2
0.0

Fig. 13. AWE performance on Covertype stream

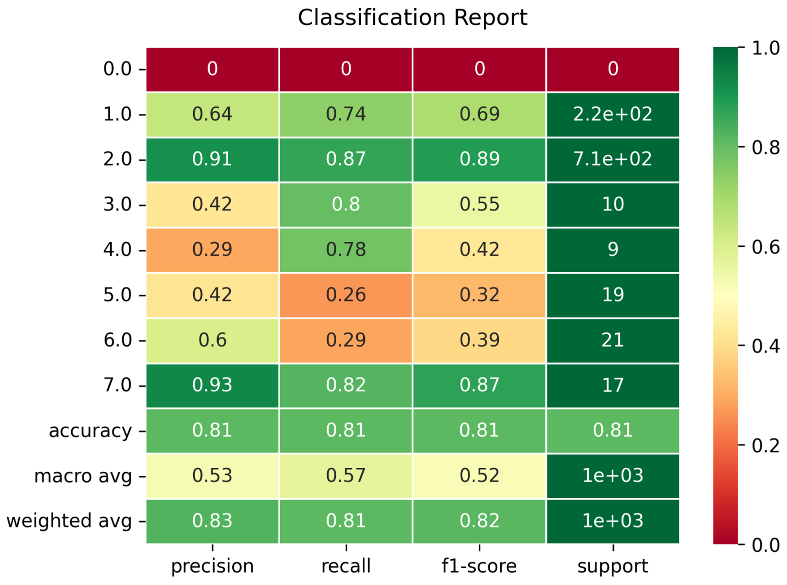
Dynamic Weighted Majority overall provided better results than previously measured AWE. For Hyperplane stream, all metrics look just as good, which is not suprising considering that Hyperplane is fairly simple classification task, having only two labels. It's expected to see any sensible ensemble methods to do well on this data. Now that it's checked that DWM is evaluated properly, let's look into classification report for Covertype stream.

It's worth noting that results are already much better than with AWE ensemble. Imbalanced targets that have only a handful of objects assigned to them were at least partially labelled correctly and most populated target 1 and 2 also have much better metrics for DWM. This goes to show that just using ensemble isn't enough; depending on environment, data and used algorithm we might get vastly different results.


Classification Report
1.0
0.8
0.6
0.4
0.2
0.0

	precision	recall	f1-score	support
0.0	0.91	0.92	0.91	5e+02
1.0	0.92	0.91	0.91	5e+02
accuracy	0.91	0.91	0.91	0.91
macro avg	0.91	0.91	0.91	1e+03
weighted avg	0.91	0.91	0.91	1e+03

Fig. 14. DWM performance on Hyperplane stream


Classification Report
1.0
0.8
0.6
0.4
0.2
0.0

	precision	recall	f1-score	support
0.0	0	0	0	0
1.0	0.64	0.74	0.69	2.2e+02
2.0	0.91	0.87	0.89	7.1e+02
3.0	0.42	0.8	0.55	10
4.0	0.29	0.78	0.42	9
5.0	0.42	0.26	0.32	19
6.0	0.6	0.29	0.39	21
7.0	0.93	0.82	0.87	17
accuracy	0.81	0.81	0.81	0.81
macro avg	0.53	0.57	0.52	1e+03
weighted avg	0.83	0.81	0.82	1e+03

Fig. 15. DWM performance on Covertype stream

14 Summary

It was shown that streams have underlying concept drifts and the author looked into how they influence data changes and their probabilistic distribution. Only part of these measurements were shown here, as investigation of concept drifts in data sources is not the topic discussed in this paper, but was presented in the source code. The author looked into classifiers and how they are influenced by these changes in data, making their predictions worse with time. Differences between single and ensemble classifiers were noted, not only in their overall metrics for chosen data sources, but also in overall improvement in result variance.

SEA data stream was shown to not be as reliable as originally expected. It was expected to be a source of sudden drifts, which would drastically change probabilistic distributions of data, making classifiers struggle when being evaluated, but as it was discussed, SEA stream does not seem to be complex enough for those changes to heavily impact classification task.

Some of the ensemble methods were measured to provide quite good, but definitely different, results. Process of creating and maintaining an ensemble was vital part of writing and understanding this experiment.

15 Future work

Another goal was also to measure custom iterative implementation of an ensemble, but the author did not manage to do that in the appointed time. Future works would cover creating such an ensemble and measuring what additions and changes improve classifier efficiency under concept drifts.

Base classifiers can also be measured with dedicated drift detection methods to detect when accuracy drops to a certain threshold; when it happens, model can be completely re-trained if needed, possibly making single-classifier measurements more reliable. The author would like to measure how well such classifiers handle evaluation and what is the upper bound on their metrics to better understand how much does an ensemble method really improve performance.

References

1. Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. Hierarchical change-detection tests. *IEEE Transactions on Neural Networks and Learning Systems*, 28:246–258, 2017.
2. Manuel Baena-García, José Campo-Ávila, Raúl Fidalgo-Merino, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno. Early drift detection method. 01 2006.
3. Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowl. Based Syst.*, 245:108632, 2022.
4. Mohamed Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: A review. *SIGMOD Record*, 34:18–26, 06 2005.
5. J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

6. J.Z. Kolter and M.A. Maloof. Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Third IEEE International Conference on Data Mining*, pages 123–130, 2003.
7. Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.
8. Ludmila I. Kuncheva. Classifier ensembles for changing environments. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
9. Leandro L. Minku, Allan P. White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.
10. Kyosuke Nishida and Koichiro Yamauchi. Adaptive classifiers-ensemble system for tracking concept drift. In *2007 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3607–3612, 2007.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. Martin Scholz and Ralf Klinkenberg. An ensemble classifier for drifting concepts. 01 2005.
13. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. pages 377–382, 07 2001.
14. Bai Su, Yi-Dong Shen, and Wei Xu. Modeling concept drift from the perspective of classifiers. In *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1055–1060, 2008.
15. Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
16. Haixun Wang, Wei Fan, Philip Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. 07 2003.
17. Haixun Wang, Wei Fan, Philip Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. 07 2003.
18. Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai-Long Nguyen, and François Petitjean. Characterizing concept drift. *CoRR*, abs/1511.03816, 2015.
19. Gerhard Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23, 11 1994.
20. Yongquan Yang and Haijun Lv. Discussion of ensemble learning under the era of deep learning. *CoRR*, abs/2101.08387, 2021.