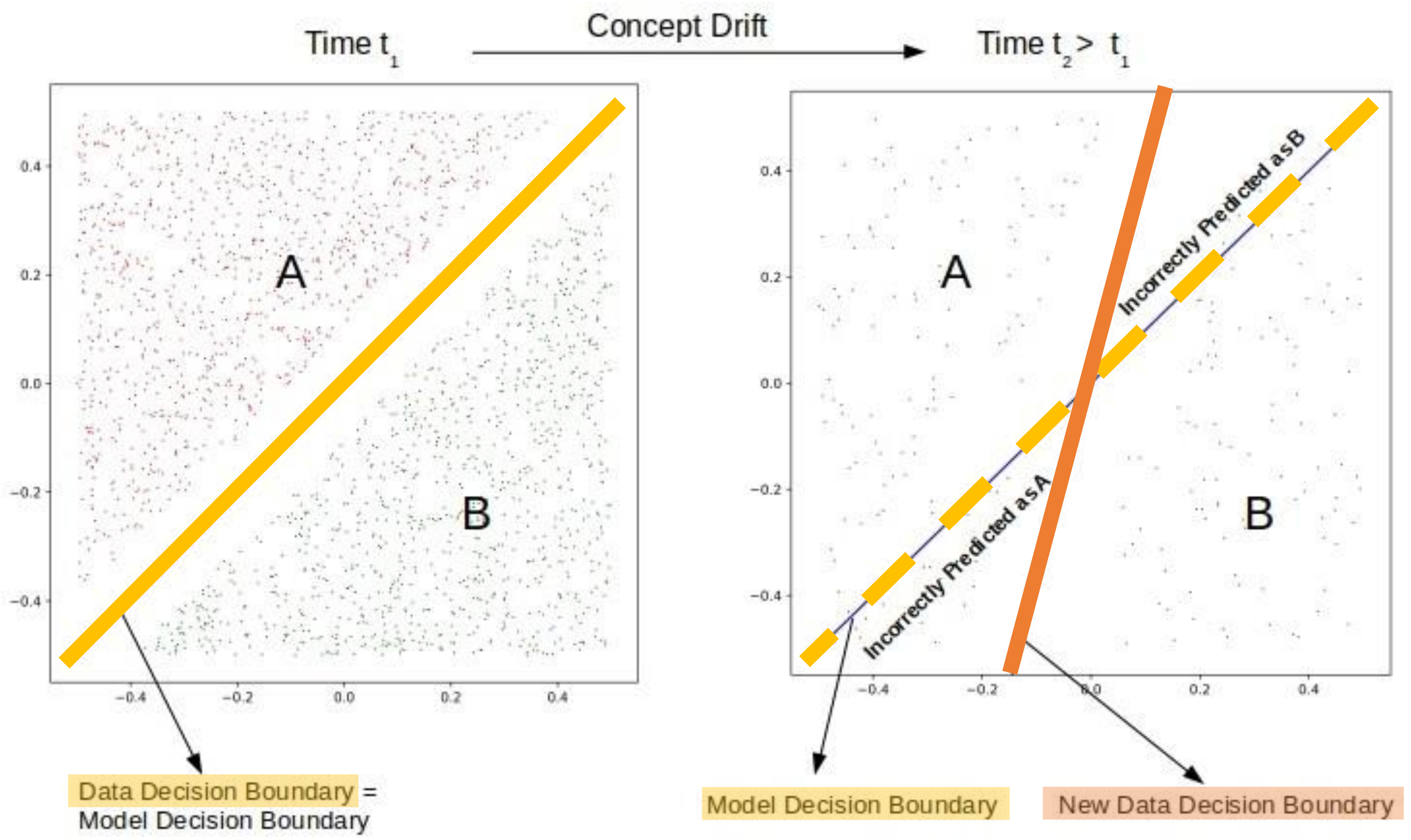
A Decision-Based Dynamic Ensemble Selection Method for Concept Drift

Tyng-Jiun Kuo, Yaser Alagele

Outline

- Problem Statement and Solution
- Method Process
- DDM
- ADWIN
- Result
- Conclusion

Concept Drift



when data are continuously generated in streams, data and target concepts may change over time.

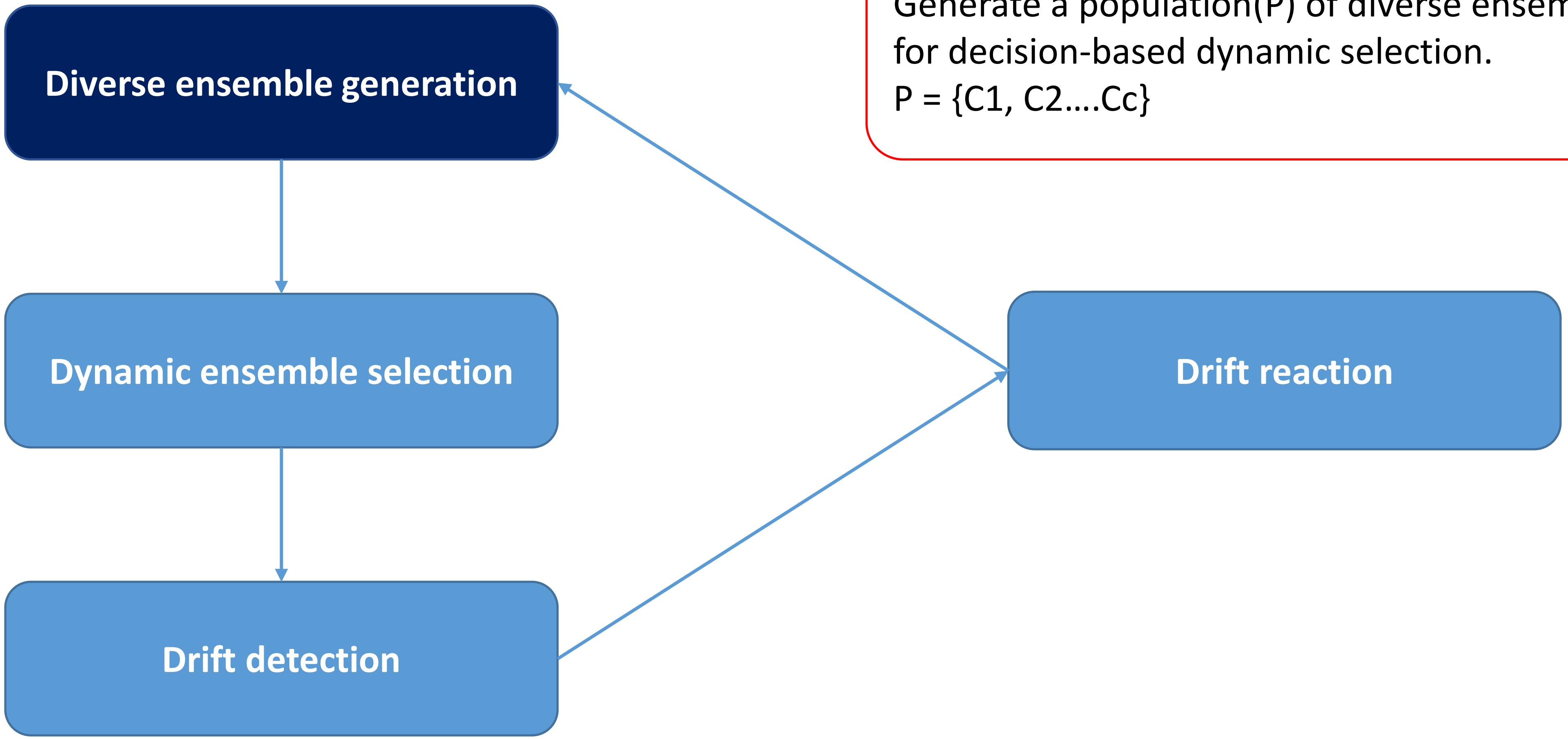
Solution

- Drift detector is a common solution
- focus on monitoring whether the class distribution is stable over time.

- Online learning ensembles
- Tackle the drift problem
- Works in blin manner.

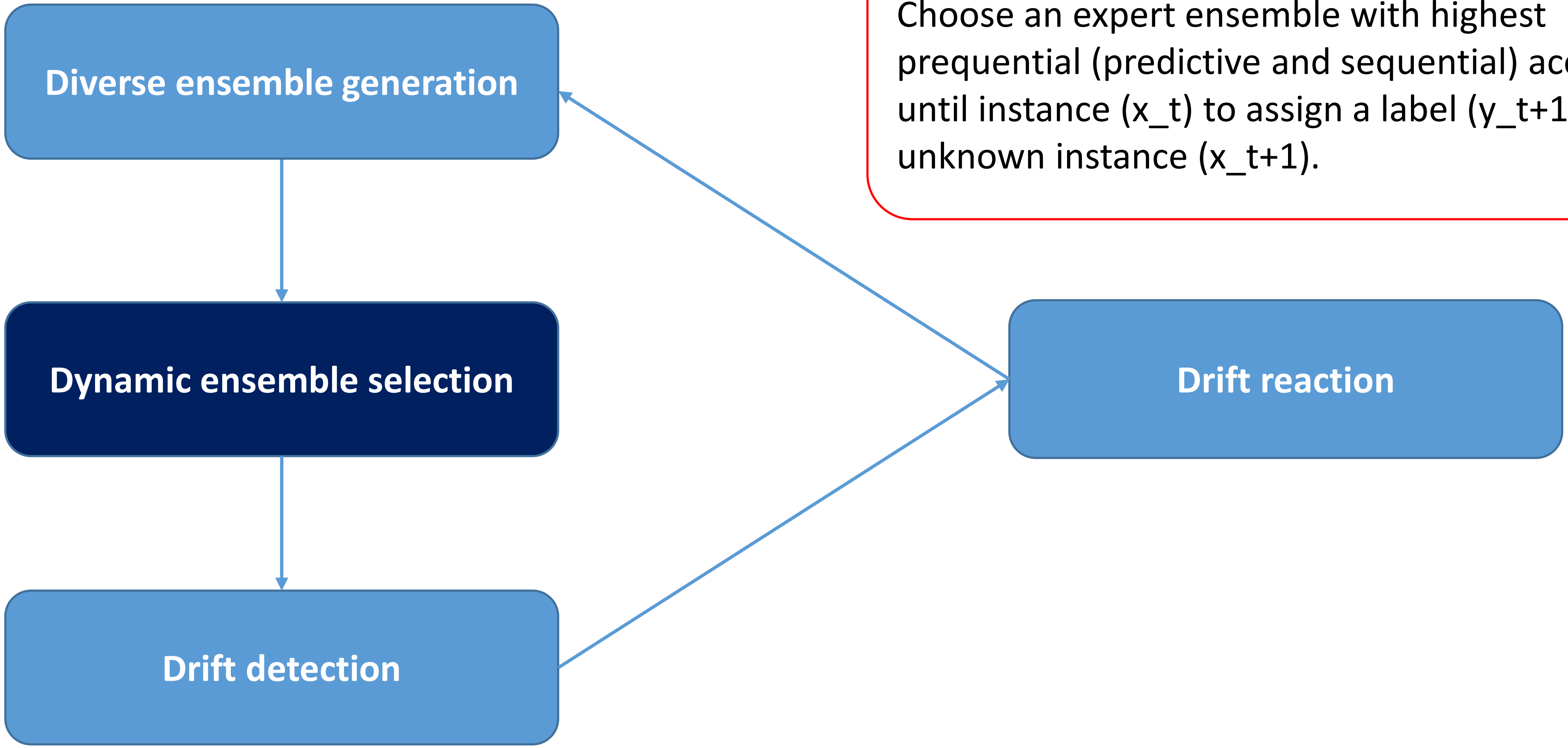
- Auxiliary drift detector: DDM, ADWIN
- Dynamic Ensemble Selection for Drift Detection (DESDD)

Method Process



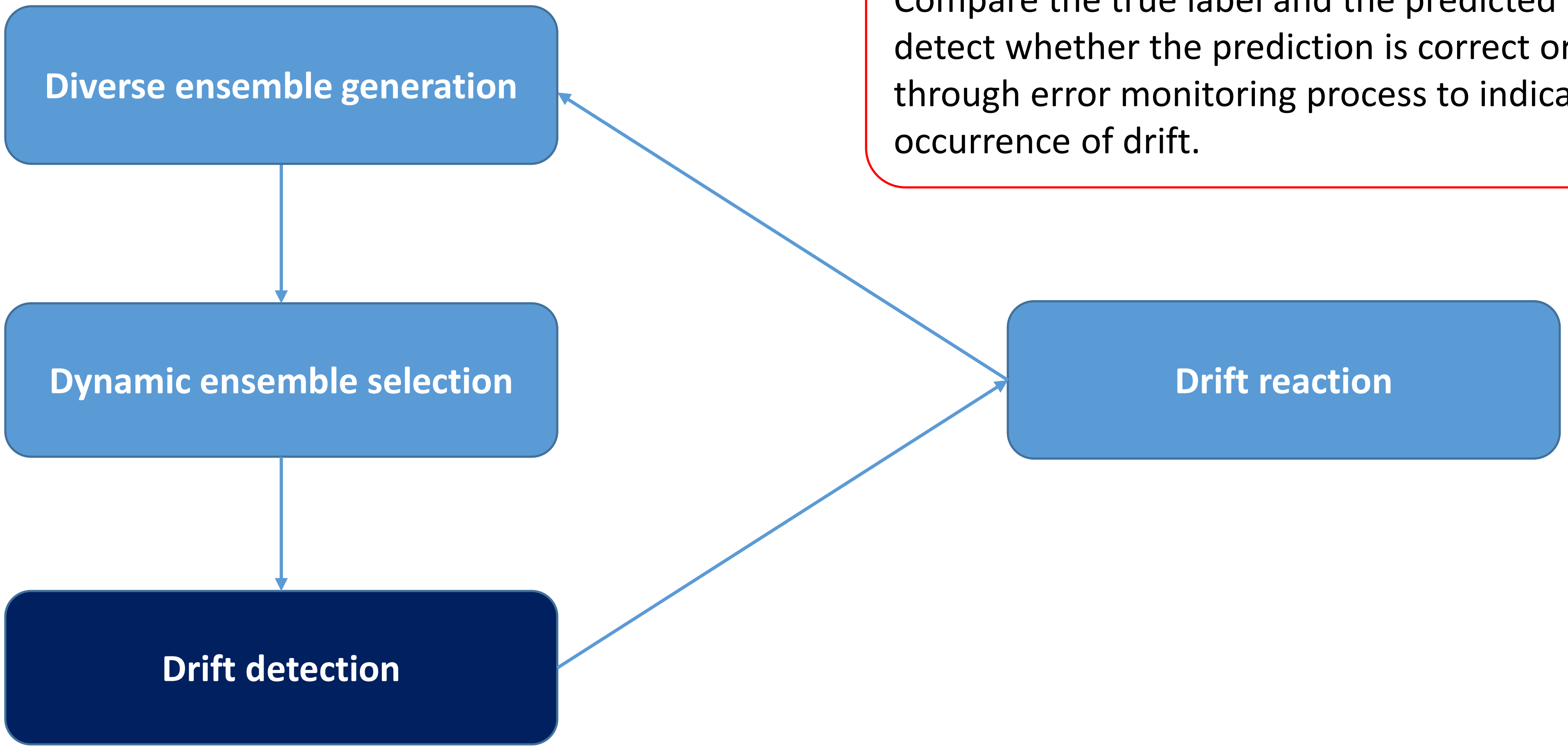
Generate a population(P) of diverse ensemble(C) for decision-based dynamic selection.
P = {C1, C2….Cc}

Method Process



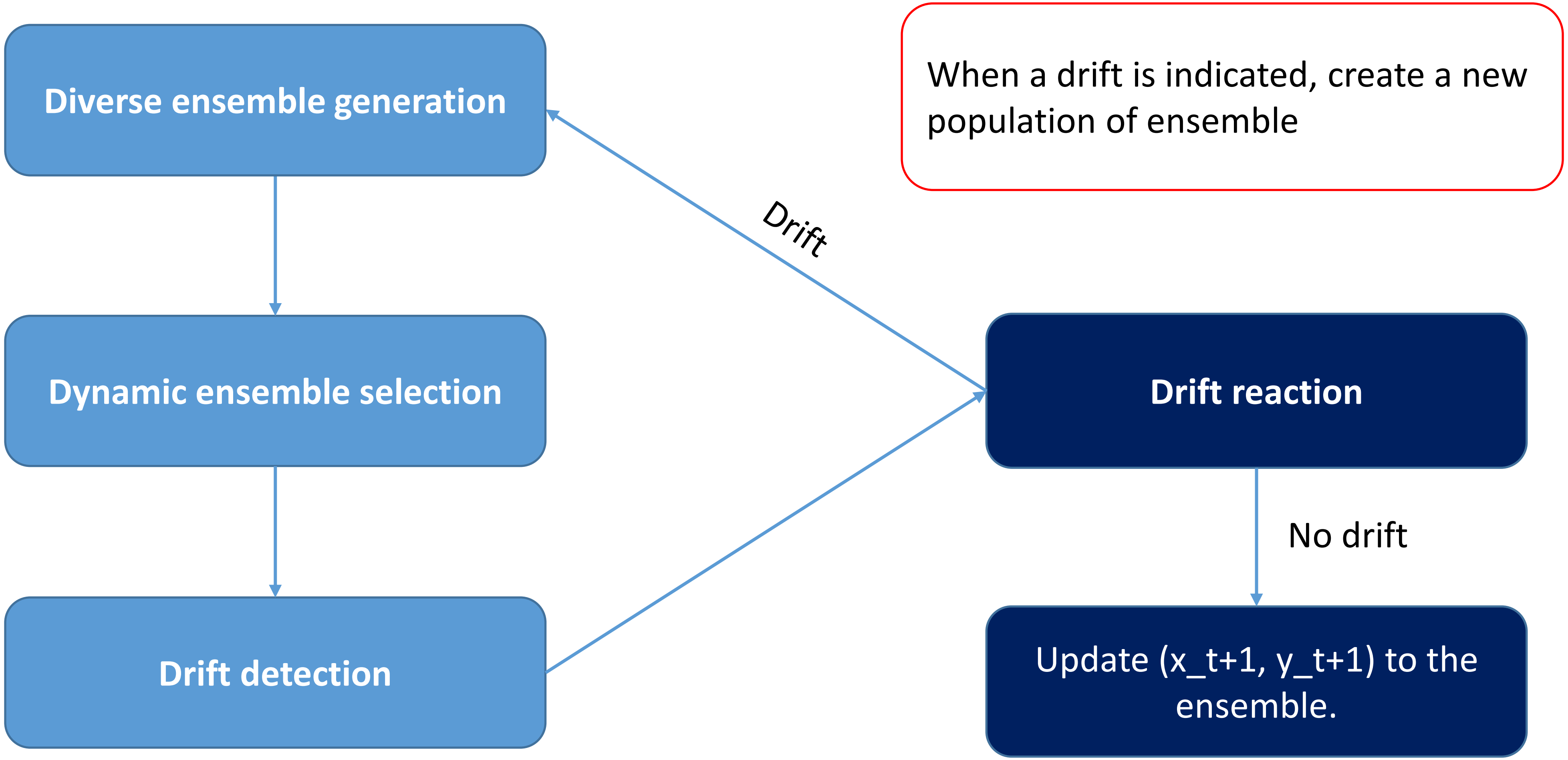
Choose an expert ensemble with highest prequential (predictive and sequential) accuracy until instance (x_t) to assign a label (y_t+1) for unknown instance (x_t+1).

Method Process



Compare the true label and the predicted label to detect whether the prediction is correct or not, then through error monitoring process to indicate the occurrence of drift.

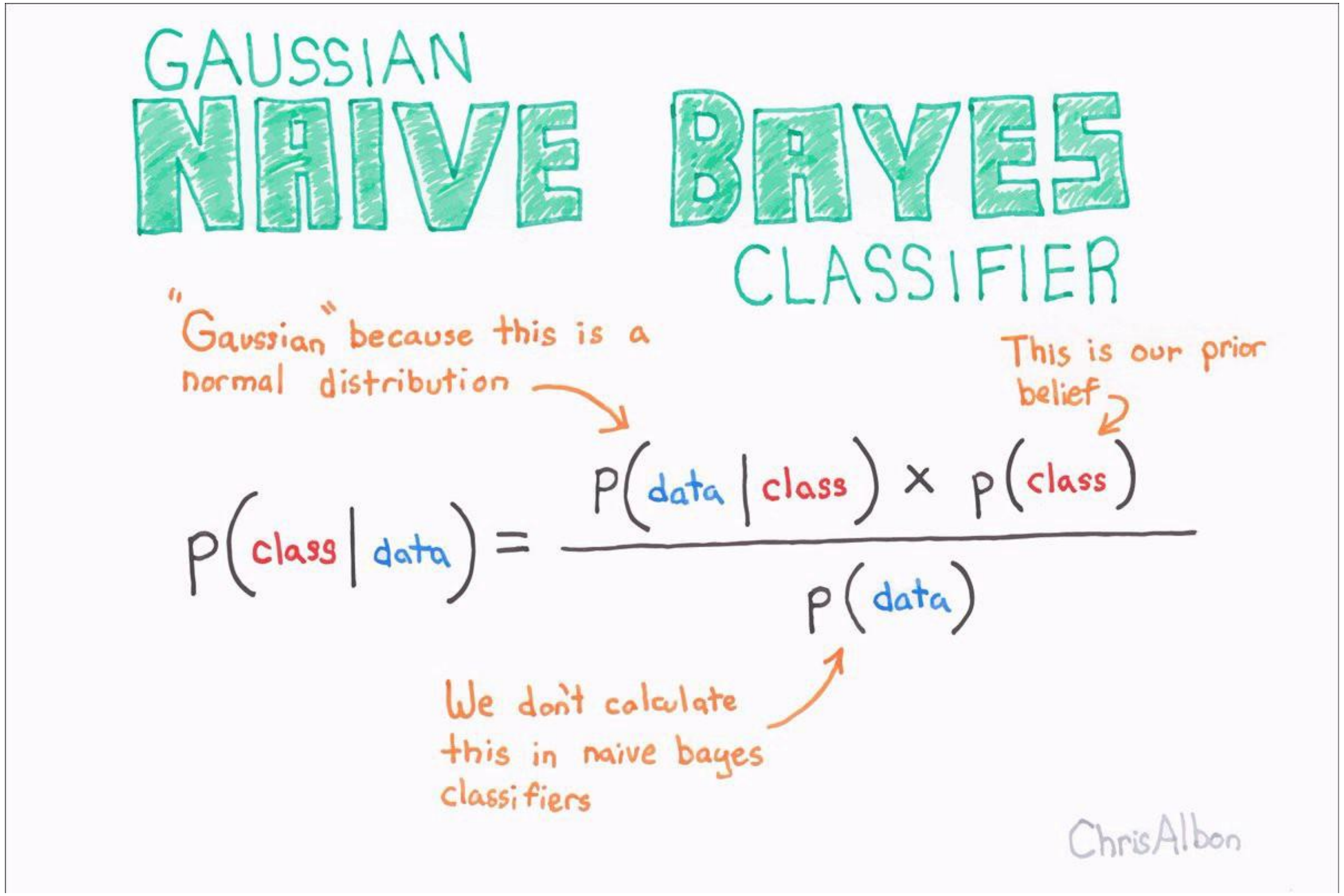
Method Process



Drift Detection Method(DDM)

- Models the number of classification errors with a Binomial distribution.
- Each iteration an online classifier predicts the decision class of an example

Gaussian Naive Bayes



- Used in a wide variety of classification tasks. Such as filtering spam, classifying documents, sentiment prediction etc.
- assumes the features that go into the model are independent of each other.

Drift Detection Method(DDM)

Pi: probability of false prediction
Si: Standard deviation

Registers for tracking error rate

Check if the prediction is correct

$$s_i = \sqrt{\frac{p_i(1-pi)}{i}} \quad (1)$$

Input:

- S: a data stream of examples
- C: classifier

Output: W : a window with examples selected to train classifier C

DDM: Drift Detection Method

Initialize $(i, \boxed{p_i, s_i}, ps_{min}, \boxed{p_{min}, s_{min}})$;
$newDrift \leftarrow false$;
$W \leftarrow \emptyset$;
$W' \leftarrow \emptyset$;
for all examples $x_i \in S$ **do**
  if prediction $C(x_i)$ is incorrect **then**
    $p_i \leftarrow p_i + (1.0 - p_i)/i$;
  else
    $p_i \leftarrow p_i - (p_i)/i$;
  compute s_i using (1);
  $i \leftarrow i + 1$;
  if $i > 30$ (approximated normal distribution) **then**
    if $p_i + p_s \leq ps_{min}$ **then**
      $p_{min} \leftarrow p_i$;
      $s_{min} \leftarrow s_i$;
      $ps_{min} \leftarrow p_i + s_i$;
    if drift detected (3) **then**

Drift Detection Method(DDM)

Warning Level Condition

$$p_i + s_i \geq p_{min} + \alpha s_{min} \quad (2)$$

Alarm Level Condition

$$p_i + s_i \geq p_{min} + \beta s_{min} \quad (3)$$

Update values for register is needed

Input:

- S: a data stream of examples
- C: classifier

Output: W : a window with examples selected to train classifier C

DDM: Drift Detection Method

Initialize $(i, p_i, s_i, ps_{min}, p_{min}, s_{min})$;
$newDrift \leftarrow false$;
$W \leftarrow \emptyset$;
$W' \leftarrow \emptyset$;
for all examples $x_i \in S$ **do**
  if prediction $C(x_i)$ is incorrect **then**
    $p_i \leftarrow p_i + (1.0 - p_i)/i$;
  else
    $p_i \leftarrow p_i - (p_i)/i$;
  compute s_i using (1);
  $i \leftarrow i + 1$;
  if $i > 30$ (approximated normal distribution) **then**
    if $p_i + p_s \leq ps_{min}$ **then**
      $p_{min} \leftarrow p_i$;
      $s_{min} \leftarrow s_i$;
      $ps_{min} \leftarrow p_i + s_i$;
    if drift detected (3) **then**

Drift Detection Method(DDM)

Alarm level reached

Warning level reached

Warning Level Condition

$$p_i + s_i \geq p_{min} + \alpha s_{min} \quad (2)$$

Alarm Level Condition

$$p_i + s_i \geq p_{min} + \beta s_{min} \quad (3)$$

```
    Initalize (i, p_i, s_i, ps_min, p_min, s_min;
    W ← W';
    W' ← Ø;
  else if warning level reached (2) then
    if newDrift = true then
      W' ← Ø;
      newDrift ← false;
    W' ← W' ∪ x_i;
  else
    newDrift ← true;
W ← W ∪ x_i;
```

Adaptive Sliding Window (ADWIN)

- suitable for data streams with sudden drift.
- whenever two “large enough” sub windows of W exhibit “distinct enough” averages, one can conclude that the corresponding expected values are different, and the older portion of the window is dropped.

Hoeffding Tree

- Incremental decision tree learner.
- Assumes that the data distribution is not changing over time.
- It grows incrementally a decision tree based on the theoretical guarantees of the Hoeffding bound.

Adaptive Sliding Window (ADWIN)

n: size of W
n0, n1: size of W0, W1
µw0, µw1: average of W0 and W1

checks if the observed average in both sub windows differs by more than threshold

- threshold is calculated using the Hoeffding bound

check all pairs of sub windows W0 and W1 created by splitting

ADWIN0: ADAPTIVE WINDOWING ALGORITHM

1: Initialize Window W
2: **for** each $t > 0$
3: **do** $\{x_t\} \bigcup W \rightarrow W$ (i.e., add x_t to the head of W)
4: **repeat** Drop elements from the tail of W
5: **until** $\boxed{|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \epsilon_{cut}}$ holds
6: $\boxed{\text{for every spilt of } W \text{ into } W = W_0 W_1}$
7: output $\hat{\mu}_W$

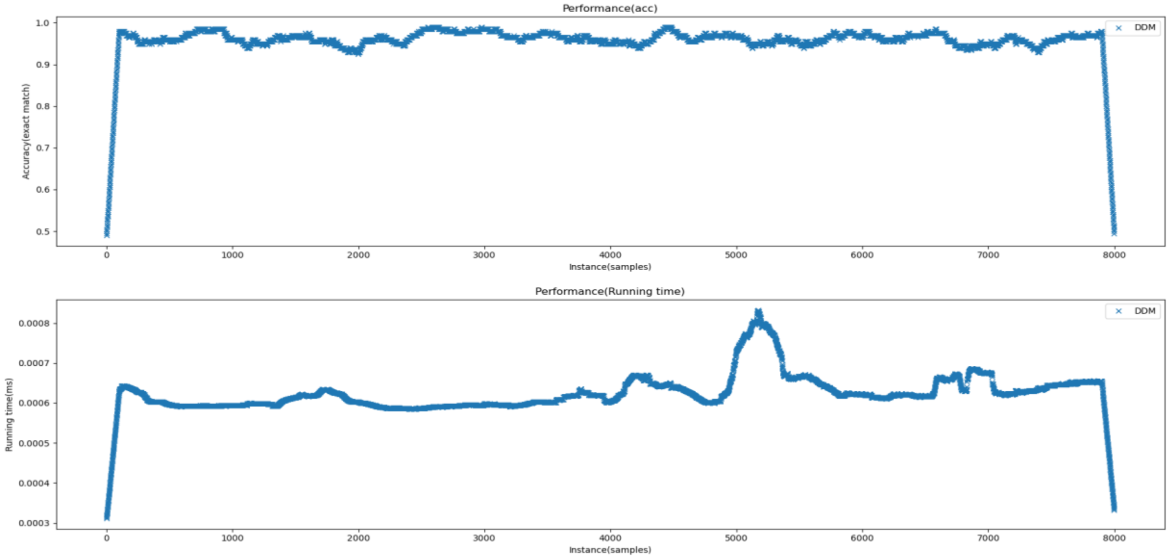
Dataset

- Four artificial datasets
- Balanced, contain 10,000 instances, contain 10% examples with class noise.
- Allow us to evaluate the methods in terms of detection rate, detection delay and miss detection, besides prequential accuracy.
- The datasets are:Agrawal Gradual, Agrawal Abrupt, Sea Gradual, Sea Abrupt. The datasets are represented by nine features and divided into two classes.

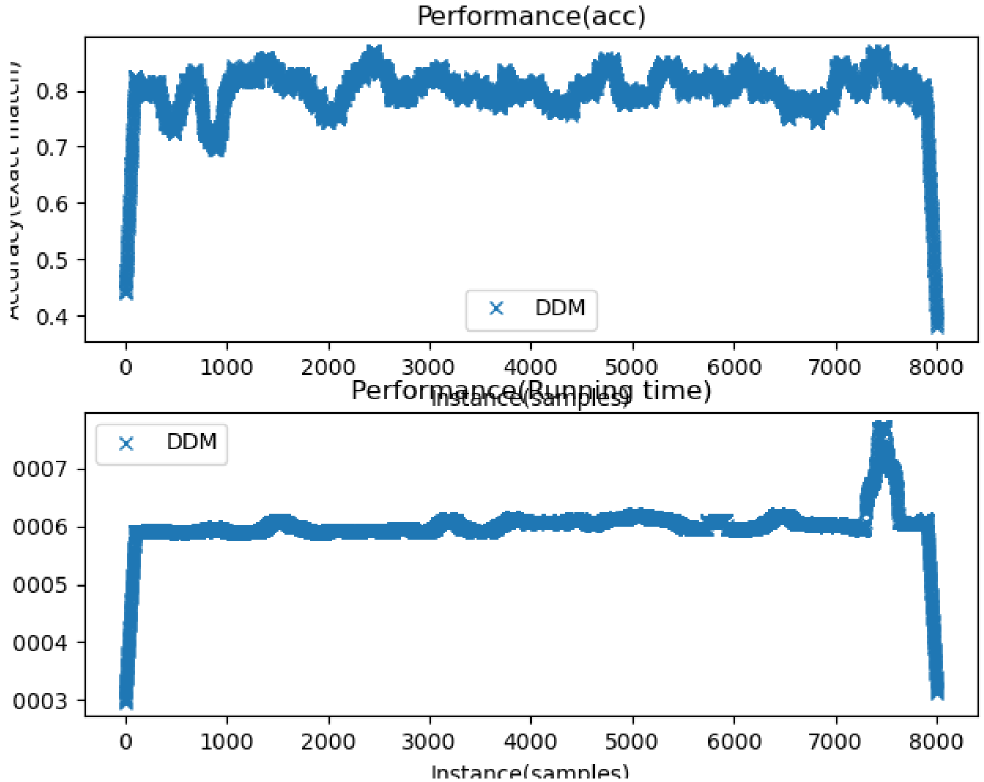
Dataset

- Four real-world dataset
- Real datasets only allow the analysis of the investigated methods regarding accuracy
- Forest Covertype: 581,012 instances divided into 7 classes representing forest cover types
- KDD Cup 1999: 489,844 instances divided into two classes.
- Poker-Hand: 829,201 instances, 10 classes
- SPAM: 9,324 instances divided into spam and ham classes

DESDD-DDM Agarwal

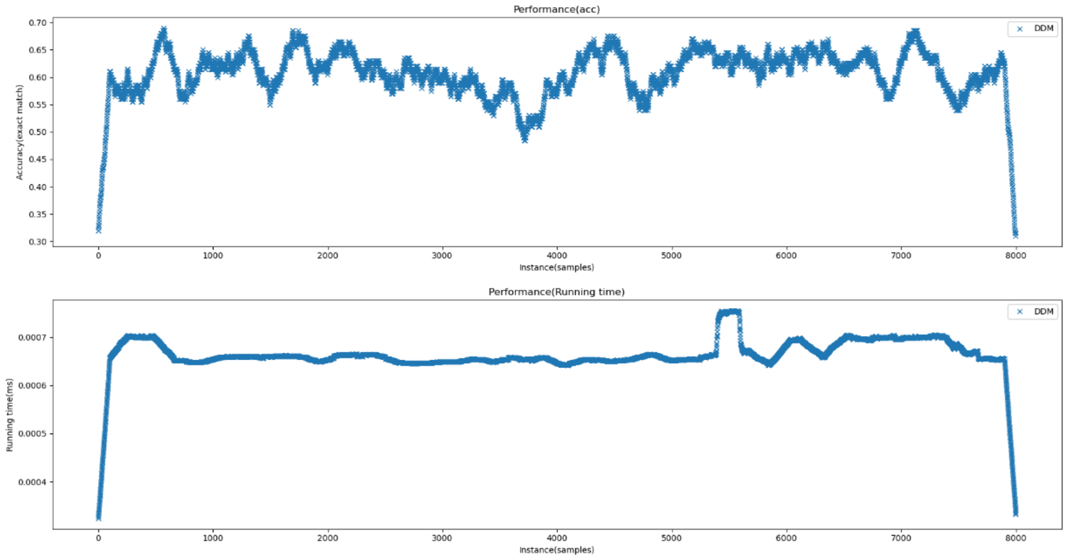


Abrupt

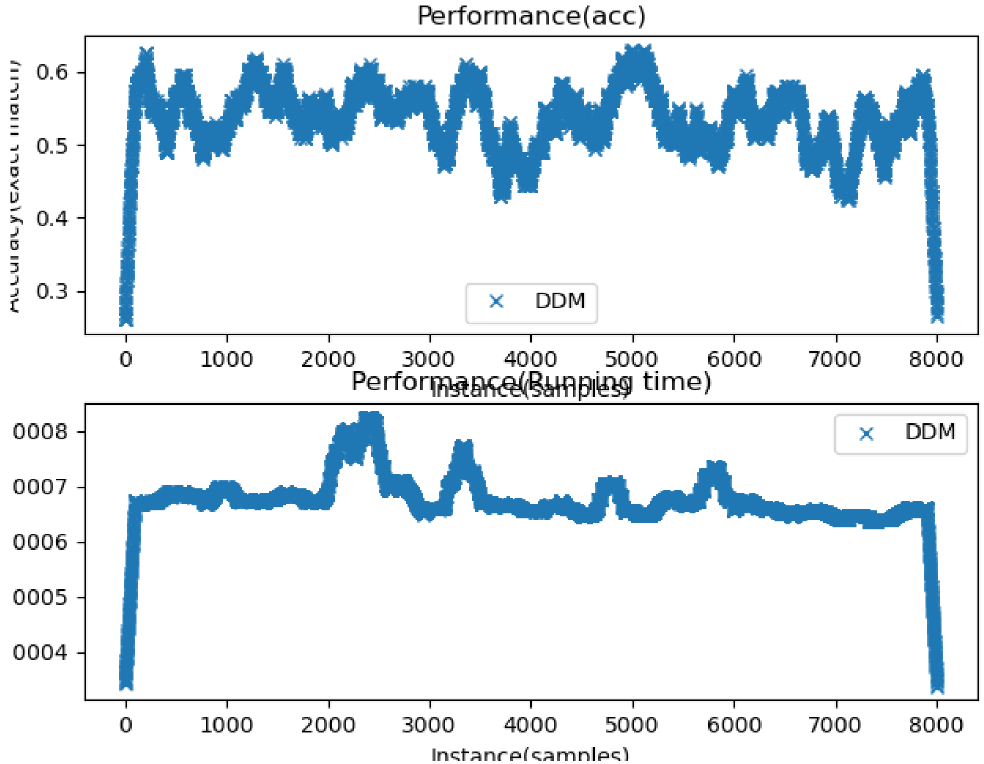


Gradual

DESDD-DDM SEA

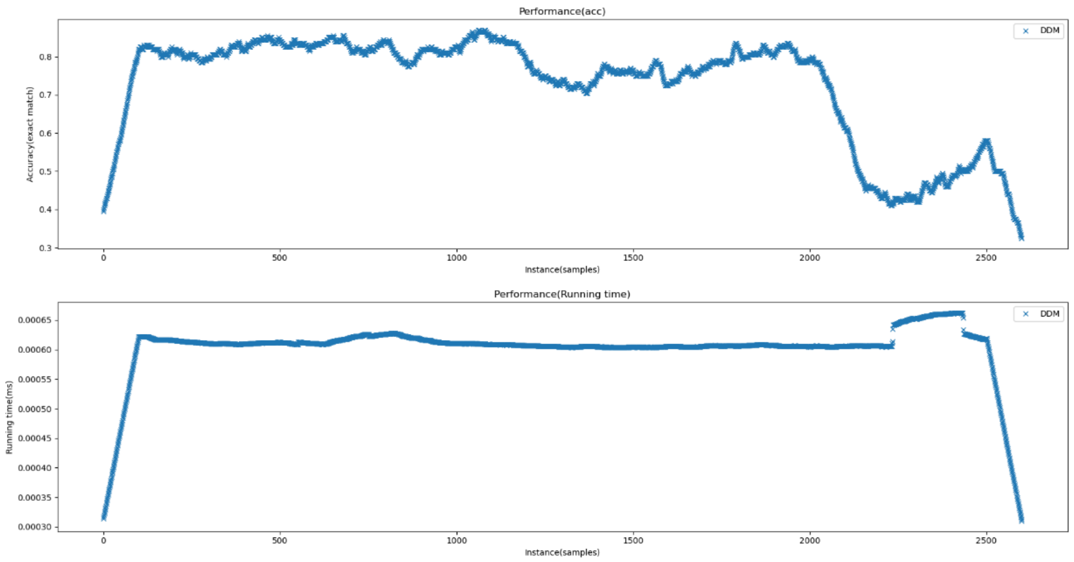


Abrupt

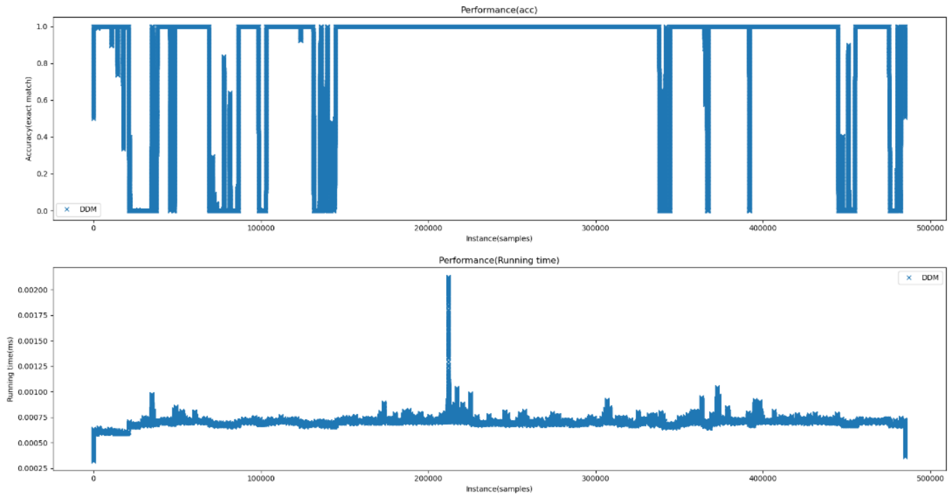


Gradual

DESDD-DDM

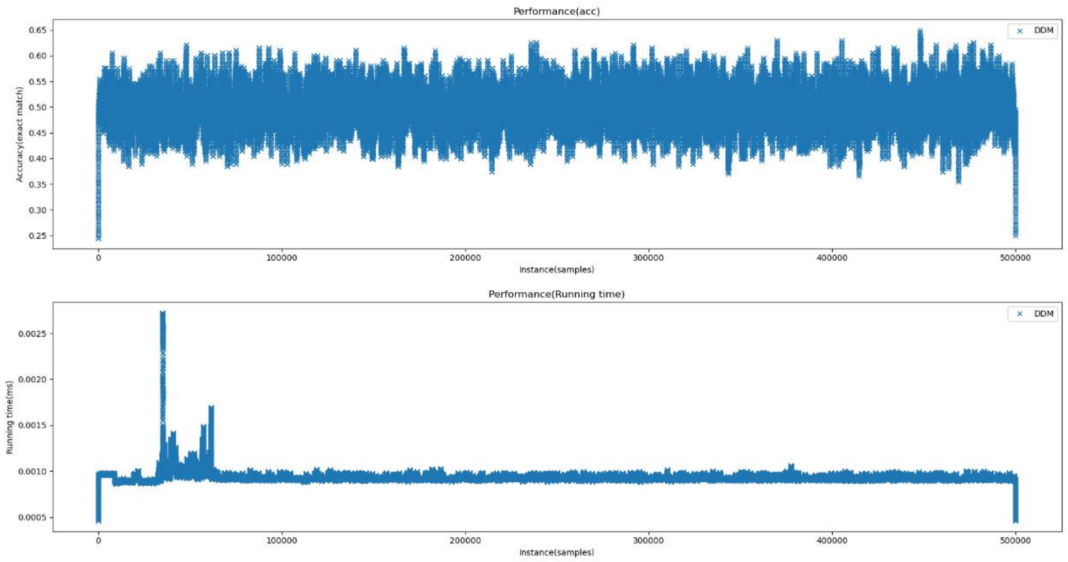


Spam

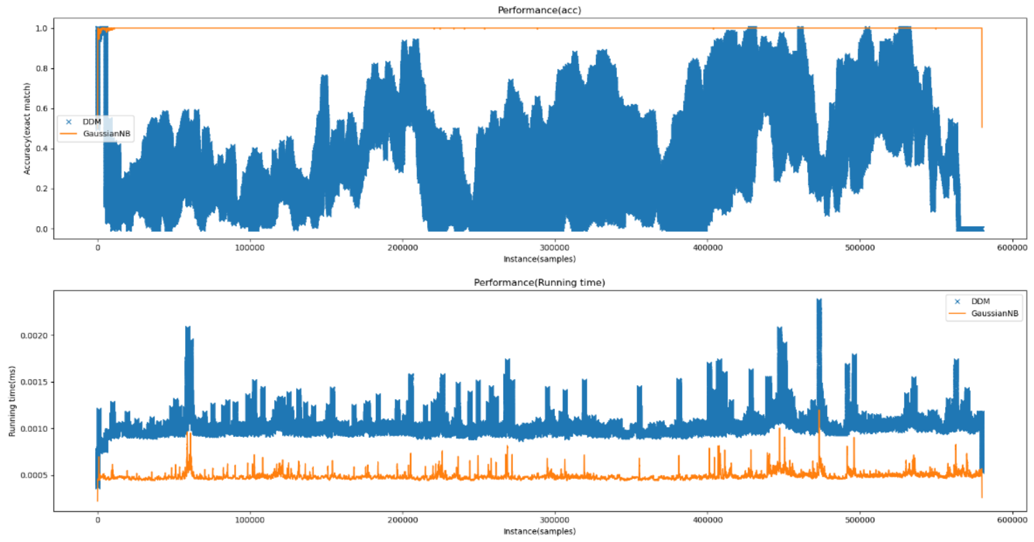


KDD cup99

DESDD-DDM



Pocker-hand



Forest Covertype

DESDD-ADWIN

```
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
 #################### [100%] [94.62s]
Processed samples: 100000
Mean performance:
HAT - Accuracy     : 0.5005
HAT - Kappa        : 0.0021
HAT - Precision: 0.1021
HAT - Recall: 0.1002
HAT - F1 score: 0.0699
HAT - Size (kB)          : 320.2549
2000 samples analyzed.
OzaBaggingClassifier Accuracy: 0.494
********************************
Results for pocker_hand.csv
********************************
```

Pocker-hand

```
##------------------ [10%] [1.04s]<Figure size 432x288 with 0 Axes>
#####--------------- [25%] [1.53s]<Figure size 432x288 with 0 Axes>
#######------------- [35%] [1.86s]<Figure size 432x288 with 0 Axes>
##########---------- [50%] [2.50s]<Figure size 432x288 with 0 Axes>
#############------- [65%] [3.18s]<Figure size 432x288 with 0 Axes>
###############----- [75%] [3.66s]<Figure size 432x288 with 0 Axes>
##################-- [90%] [4.34s]<Figure size 432x288 with 0 Axes>
#################### [100%] [4.76s]
Processed samples: 4601
Mean performance:
HAT - Accuracy     : 0.9868
HAT - Kappa        : 0.9716
HAT - Precision: 0.9814
HAT - Recall: 0.9826
HAT - F1 score: 0.9820
HAT - Size (kB)          : 173.8330
2000 samples analyzed.
OzaBaggingClassifier Accuracy: 0.8815
********************************
Results for spam.csv
********************************
```

Spam

```
##################-- [90%] [7.69s]<Figure size 432x288 w
###################- [95%] [8.22s]<Figure size 432x288 w
#################### [100%] [8.75s]
Processed samples: 10000
Mean performance:
HAT - Accuracy     : 0.9962
HAT - Kappa        : 0.9924
HAT - Precision: 1.0000
HAT - Recall: 0.9926
HAT - F1 score: 0.9963
HAT - Size (kB)          : 336.2822
2000 samples analyzed.
OzaBaggingClassifier Accuracy: 1.0
********************************
```

SEA Abrupt

https://github.com/scikit-multiflow/scikit-multiflow/issues/116

Result Comparison DESDD-DDM

Dataset	Accuracy	Number of Detected Drifts	From paper
Agarwal Abrupt	96%	3	79.22%
Agarwal Gradual	80%	11	76.85%
Sea Abrupt	61%	17	83.87%
Sea Gradual	54%	21	83.17%
Spam	37%	2521	96.26%
KDD cup 99	85%	16	99.97%
Pocker-hand	50%	46	91.06%
Forest covertype	74%	15	93.28%

Result Comparison DESDD-ADWIN

Dataset	Accuracy ADWIN	From paper
Sea Abrupt	99.6%	83.02%
Spam	98.6%	-
Pocker-hand	50%	-

The author didn’t implement the real-world dataset

Conclusion

From the experimental analysis, the following decisions can be made.

- When the number of detected drift is high, the accuracy is lower.
- If the number of classes is high, the accuracy is relatively lower.
- The choice of hyperparameters changes the accuracy and drift drastically, so it’s hard to replicate the results without knowing all the hyperparameters.

Github page

https://github.com/TyngJiunKuo/Data-Mining-Labor

Code	Add files via upload	now
dataset	Update and rename Agrawal to Readme.md	17 hours ago
README.md	Update README.md	18 hours ago

README.md

A Dicision-Based Dynamic Ensemble Selection Method for Concept Drift

cited paper: https://ieeexplore.ieee.org/document/8995320
R. A. S. Albuquerque, A. F. J. Costa, E. Miranda dos Santos, R. Sabourin and R. Giusti, "A Decision-Based Dynamic Ensemble Selection Method for Concept Drift," 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 2019, pp. 1132-1139, doi: 10.1109/ICTAI.2019.00158.

Abstract

The main task of this paper is concept drift. Concept drift will occur when data are continuously generated in streams, data and target concepts may change over time. For this problem `drift detector` is a commom solution, so the author proposed an online method which monitoring the stabilization of class distribution over time which named `Dynamic Ensemble Selection for Drift Detection(DESDD)`.

According to the idea of the author, the model should be able to estimate the class for each unknown instance, in order to raise the possibility of making a correct classification, the author then proposed an `ensemble-based method` which include **diverse population** of ensambles with different **member's diversity**, called `dynamic ensemble selection(DES)`, which elect a single ensemble that is probably the best qualified to predict the class for given sample.

Proposed Method

DESDD is divided into four step:

Thank you for your attention