Classification of the drifting data streams using heterogeneous diversified dynamic class-weighted ensemble

Martin Sarnovsky [Corresp., 1], **Michal Kolarik** [1]

[1] Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University in Kosice, Kosice, Slovakia

Corresponding Author: Martin Sarnovsky
Email address: martin.sarnovsky@tuke.sk

Data streams can be defined as the continuous stream of data coming from different sources and in different forms. Streams are often very dynamic, and its underlying structure usually changes over time, which may result to a phenomenon called concept drift. When solving predictive problems using the streaming data, traditional machine learning models trained on historical data may become invalid when such changes occur. Adaptive models equipped with mechanisms to reflect the changes in the data proved to be suitable to handle drifting streams. Adaptive ensemble models represent a popular group of these methods used in classification of drifting data streams. In this paper, we present the heterogeneous adaptive ensemble model for the data streams classification, which utilizes the dynamic class weighting scheme and a mechanism to maintain the diversity of the ensemble members. Our main objective was to design a model consisting of a heterogeneous group of base learners (Naive Bayes, k-NN, Decision trees), with adaptive mechanism which besides the performance of the members also takes into an account the diversity of the ensemble. The model was experimentally evaluated on both real-world and synthetic datasets. We compared the presented model with other existing adaptive ensemble methods, both from the perspective of predictive performance and computational resource requirements.

Classification of the drifting data streams using a heterogeneous diversified dynamic class-weighted ensemble

Martin Sarnovsky[1] **and Michal Kolarik**[1]

[1]**Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University in Kosice, Kosice, Slovakia**

Corresponding author:
Martin Sarnovsky[1]

Email address: martin.sarnovsky@tuke.sk

ABSTRACT

Data streams are continuous sequences of data coming from different sources and in various forms. Streams are often very dynamic, and their underlying structures usually change over time, which may result in a phenomenon called concept drift. When solving predictive problems using streaming data, traditional machine learning models trained on historical data may become invalid when such changes occur. Adaptive models equipped with mechanisms to reflect the changes in the data proved to be suitable for handling drifting streams. Adaptive ensemble models represent a popular group of these methods used in the classification of drifting data streams. In this paper, we present the heterogeneous adaptive ensemble model for data stream classification, which utilizes the dynamic class weighting scheme and a mechanism to maintain the diversity of the ensemble members. Our main objective was to design a model consisting of a heterogeneous group of base learners (Naive Bayes and Decision Trees), with an adaptive mechanism that, besides the performance of the members, also takes into account the diversity of the ensemble. The model was experimentally evaluated on both real-world and synthetic datasets. We compared the presented model with other existing adaptive ensemble methods, both from the perspective of predictive performance and computational resource requirements.

INTRODUCTION

Nowadays, the size of data is growing in a much faster fashion than in the past. Information is being collected from household appliances, tools, mobile devices, vehicles, sensors, websites, social networks, and many other devices. An increasingly large number of organizations are starting to analyze large volumes of data, as the information obtained from these data can provide a competitive advantage over other businesses. Data collection from devices is often continuous, and data come in the form of data streams.

Data stream classification is an active field of research, as more data sources can be considered as streaming data. When solving classification tasks using streaming data, the data generation process is not strictly stationary, and its underlying structure may change over time. The changes in the underlying data distribution within the streams may result in dynamic, non-stationary target concepts (Gama et al. (2014b); Žliobaitė (2010)). This phenomenon is called concept drift, and from the perspective of the training of classification models on the drifting data, the most crucial requirement is the ability of the model to adapt and incorporate new data into the model in order to react to potential changes (Barddal et al. (2017)). In concept drift, the adaptive learning algorithms are advanced machine learning methods that can reflect the changing concepts in data streams in real time. Multiple approaches were proposed to extend the standard machine learning models with the ability to adapt to the changes in streams, including drift detectors (Gonçalves et al. (2014); Baena-García et al. (2006)) and various sliding window techniques (Bifet and Gavaldà (2007)).

Ensemble models are a popular classification method, often providing better performance when compared to the standard machine learning models (Breiman (1996, 2001); Freund and Schapire (1996)).

When processing dynamic data streams, where the concepts change over time, dynamic adaptive ensembles present a suitable method that retains long-present historical concepts and covers newly appearing ones. Ensemble methods proved to be capable of handling streaming data by updating its base learners (Kolter and Maloof (2007); Bifet et al. (2015); Brzeziński and Stefanowski (2011); Gomes et al. (2017)) in either block-based, batch mode or instance-based, incremental mode. One of the crucial aspects of the ensemble classifiers for static data is to ensure the diversity of the base classifiers within the ensemble. Opposed to the diversity of the ensembles for the static data (Carney and Cunningham (2000); Kuncheva and Whitaker (2003)), which is fairly well studied in the literature, there are fewer studies dealing with the ensemble's diversity in the presence of concept drift (Brzezinski and Stefanowski (2016); Abassi (2019)).

The main objective of the work presented in this paper is to present the design and implementation of a novel adaptive ensemble classification algorithm. The primary motivation of this study is to design a model capable of handling various types of drifts. The proposed method can be characterized as a heterogeneous, chunk-based approach that utilizes different types of base classifiers within the ensemble. The model also uses Q statistic as a metric to measure the diversity between the ensemble members and dynamic weighting scheme. We assume that the creation of a heterogeneous ensemble consisting of different base learners with an adaptation mechanism that ensures the diversity of its members can lead to a robust ensemble that is able to handle the drifting data efficiently. To confirm these assumptions, we conducted an extensive experimental study, where we evaluated the proposed model performance on the selection of both real-world and synthetic, generated datasets with different types of concept drift. We also compared the proposed method to 12 adaptive ensemble methods, including state-of-the-art adaptive ensembles. The main objective was to compare the performance of the methods using standard evaluation metrics, as well as training times and resource consumption.

The paper is organized as follows. The background section provides basic definitions of the terms related to the data streams and the concept drift therein. The following section is dedicated to the state of the art in the area of the adaptive ensemble models used to handle the concept drift and defines the motivation for the presented approach. The following section describes the designed and implemented adaptive ensemble method. We then describe the datasets used in the experimental evaluation. The experimental results section summarizes the conducted experiments and achieved results. Concluding remarks and possibilities for future work in this area are then summarized.

BACKGROUND

A data stream is defined as an ordered sequence of data items, which appear over time. Data streams are usually unbounded and ordered sequences of the data elements $\langle x_1, x_2, \ldots, x_j, \ldots, \rangle$ sequentially appearing from the stream source item by item (Gama et al. (2008)). Each stream element is generated using a probability distribution P_j. The time interval between the particular elements on the stream may vary. Particular data elements in the stream are usually of standard size, and most of the data streams are generated at high speed, e.g., the elements in the data streams appear rapidly.

Compared with static data, which is usually analyzed offline, data streams need to be analyzed in real time. The differences in the processing of the data streams compared to the processing of the static data can be summarized according to (Gama (2010)):

- The data elements in the stream arrive online or in batches. Elements appearing online are processed one by one, and batches usually have the same size and are processed at once.

- The processing system has no control over the order in which data elements appear, either within a data stream or across multiple data streams.

- Data streams are potentially unbound in size. The size of the particular elements is usually small. In contrast, the entire data size may be huge, and it is generally impossible to store the whole stream in memory.

- Once an element from a data stream has been processed, it is usually discarded. It cannot be retrieved unless it is explicitly stored in memory.

The data streams can be divided into two groups:

- stationary data streams—the data distribution does not change over time, e.g., the stream elements are generated from a fixed probability distribution;

- non-stationary data streams—data are evolving, and the data distribution may change over time. Usually, these changes may also affect the target concepts (classes).

Concept drift

When solving predictive data analytical tasks on the static data, the data distribution usually does not change, and data used for the training and testing of the model have the same distribution. When processing the data streams, we often observe the changing nature of the data. In predictive data stream analytical tasks, we experience a phenomenon called concept drift.

Concept drift is related to the data distribution $P_t(x,y)$, where $x = (x^1, x^2 \ldots x^n)$ is a data sample represented by an n-dimensional feature vector appearing at time t, and y represents the target class. The concepts in the data are stable (or stationary) if all the data samples are generated with the same distribution. If there is an x in the interval between t and $t + \Delta$, which holds the expression $P_t(x,y) \neq P_{t+\Delta}(x,y)$, then concept drift is present (there is a change in the underlying data distribution) (Žliobaitė (2010)). Concept drift usually occurs in a non-stationary and dynamically changing environment, where the data distribution or relation between the input data and the target variable changes over time.

The concept drift phenomenon may occur in various real-world data and corresponding applications (Žliobaitė et al. (2016)):

- computer systems or networks, through network intrusion detection, where new techniques and methods may appear (Liu et al. (2017); Mukkavilli and Shetty (2012));

- industry, when dynamic data streams are produced by sensors in production equipment and machines (Lin et al. (2019); Zenisek et al. (2019));

- marketing and management, when users change their buying behaviour and their preferences (Black and Hickey (2003); Chiang et al. (2013); Lo et al. (2018));

- medical data, e.g., in the case of antibiotic resistance (Stiglic and Kokol (2011); Tsymbal et al. (2006));

- social networks, when users change their behavior and generated content (Lifna and Vijayalakshmi (2015); Li et al. (2016));

- spam categorization, where spam keywords can change over time (Delany et al. (2005); Ruano-Ordás et al. (2018)).

The authors in (Tsymbal (2004); Žliobaitė (2010); Khamassi et al. (2019)) describe a complex taxonomy of existing drift types. In general, there are several different types of concept drift, based on how the phenomenon occurs within the data stream:

- Sudden/Abrupt—In this case, the concept change occurs suddenly. A concept (e.g., a target class) is suddenly replaced by another one. For example, in the topic modelling domain, the main topic of interest may unexpectedly switch to a different one.

- Incremental—Changes in the data distribution are slower and proceed over time. Changes are not as visible as in a sudden drift, but they gradually emerge. The changes are usually relatively slow and can be observed when comparing the data over more extended time periods.

- Gradual—In this drift type, both concepts are present, but over time, one of them decreases, while the other one increases. For example, such a change may reflect the evolution of points of interest, e.g., when a point of interest is gradually being replaced by a newer one.

- Re-Occurring—A previously active concept reappears after some time. Re-occurrence may appear in cycles or not (e.g., reappearing fashion trends).

139   Besides the mentioned drift types, some publications (Gama et al. (2014a)) distinguish between two
140 kinds of concept drift: real drift and virtual drift. Virtual concept drift is defined by the changes in data
141 distribution but does not affect the target concept. Real concept drift (also called concept shift) represents
142 a change in the target concept, which may modify the decision boundaries.
143   Fig. 1 visualizes the drift types. In concept drift detection, it is also necessary to distinguish between
144 the drift and outlier occurrence. Outliers may produce false alarms when detecting the concept drift.
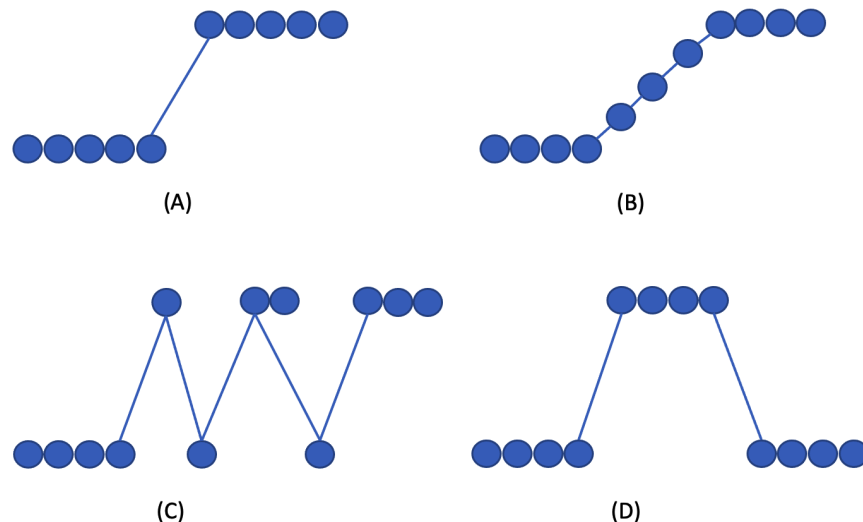
Figure 1. Concept drift types according to (Gama et al. (2014a)). (A) Sudden/Abrupt, (B) Incremental, (C) Gradual, (D) Re-occuring.

145   When processing the non-stationary drifting streams, the necessary feature of the predictive algorithms
146 is their ability to adapt. Some of the algorithms are naturally incremental (e.g. Naive Bayes), while
147 others require significant changes in the algorithm structure to enable incremental processing. Therefore,
148 the learning algorithms applied on the drifting streams are usually extended with a set of mechanisms,
149 which enhance the models with the ability of continuously forgetting the obsolete learned concepts and
150 updating the model with the newly arrived data in the stream. There are several types of models used to
151 handle concept drift. To detect the concept drift in data streams, we can use drift detectors. These can
152 detect possible concept drift by analyzing the incoming data or by monitoring the classifier performance.
153 Detectors process the signal from the data about changes in data stream distribution. Drift detectors
154 usually signalize drift occurrence and trigger the updating/replacement of the classifier. There are several
155 drift detection methods available (Gonçalves et al. (2014)), and the Drift Detection Method (DDM) (Gama
156 et al. (2004)), the Early Drift Detection Method (EDDM) (Baena-García et al. (2006)), and ADWIN (Bifet
157 and Gavaldà (2007)) are the most popular.
158   For predictive data modelling applied on the drifting streams, advanced adaptive supervised machine
159 learning methods are used. Supervised learning methods used for drifting stream classification could
160 be categorized from several perspectives, depending on how they approach the adaptation (Ditzler et al.
161 (2015); Krawczyk et al. (2017)):

162   • Active/Passive—Active methods usually utilize drift detection methods to detect the drift and to
163   trigger the model update. Passive methods periodically update the model, without any knowledge
164   of the drift occurrence.

165   • Chunk-Based/Online—Chunk-based methods process the streaming data in batches (each batch
166   consists of a specified fixed number of stream elements). Online methods process the stream
167   elements separately when they appear.

168   Ensemble models represent a popular solution for the classification of drifting data streams. An
169 ensemble classification model is composed of a collection of classifiers (also called base learners,

170 ensemble members, or experts) whose individual decisions are combined (most often by voting) to
171 classify the new samples (Sagi and Rokach (2018)). The main idea of the ensemble model is based on
172 the assumption that a set of classifiers together can achieve better performance than individual classifiers
173 (Kuncheva and Whitaker (2003)). The selection of the ensemble experts is a crucial factor, as the ideal
174 ensemble consists of a set of diverse base learners. Ensemble models are also suitable for data stream
175 classification, where target concepts change over time. The following section summarizes the use of
176 ensembles in the classification of drifting streams.

RELATED WORK

178 There are several approaches to the design of adaptive ensemble models. Some of them use the same
179 technique as approaches to static data processing, such as Online Boosting (Wang and Pineau (2013)),
180 which is based on the classic Boosting method, extended with online processing capabilities. For
181 adaptation to concept drift, it uses concept drift detection. If a concept drift occurs, the entire model is
182 discarded and replaced by a new model. Another well-known model is the OzaBagging (Oza (2005))
183 ensemble. Unlike Bagging for static data, OzaBagging does not use random sampling from the training
184 data, but each of the samples is trained k times, which leads to a Poisson distribution.

185 Further studies have focused on the design of the ensemble models that would be simple (in terms
186 of their run time) and able to adapt to the concept drift dynamically. For example, the AWE (Accuracy
187 Weighted Ensemble) (Brzeziński and Stefanowski (2011)) uses the assignment of weight to the base
188 classifiers based on a prediction error. Old and weak members are gradually being replaced by the new
189 ones, with a lower error rate. The update mechanism is based on the assumption that the latest training
190 chunk will better represent the current test chunk. Another model, DWM (Dynamic Weighted Majority)
191 (Kolter and Maloof (2007)), dynamically changes the weights of the base classifiers in the case of incorrect
192 classification. A new classifier is added if the model incorrectly classifies the training example, and
193 old classifiers are discarded if their weights fall below a threshold value. Online Bagging and Boosting
194 algorithms were recently used as a basis for more advanced streaming ensembles, such as Adaptive
195 Ensemble Size (AES) (Olorunnimbe et al. (2018)), which dynamically adapts the ensemble size, or an
196 approach (Junior and Nicoletti (2019)), where boosting is applied to the new batches of data and maintains
197 the ensemble by adding the base learners according to the ensemble accuracy rate. Learn++ (inspired by
198 AdaBoost) is an incremental learning ensemble approach consisting of base learners trained on a subset
199 of training data and able to learn the new classes (Polikar et al. (2001)). Several modifications of this
200 approach exist, focused on improvement of the number of generated ensemble members (Muhlbaier et al.
201 (2004)). The Random Forests method is probably the most popular ensemble method on static data at
202 present. Its adaptive version for stream classification, Adaptive Random Forests (ARF), was introduced
203 in (Gomes et al. (2017)) and has shown a high learning performance on streaming data. More recently,
204 multiple adaptive versions of popular ensemble methods gaining improved performance or achieving
205 speedup in execution have been introduced, e.g., the adaptive eXtreme Gradient Boosting method (Montiel
206 et al. (2020)), the streaming Active Deep Forest method (Luong et al. (2020)), or Random Forests with an
207 implemented resource-aware elastic swap mechanism (Marrón et al. (2019)).

208 All ensemble models work with the assumption of the diversity of the individual classifiers in the
209 ensemble, while the diversity is achieved in different ways. Diversity can help in evolving data streams, as
210 the most suitable method may also change as a result of the stream evolution Pesaranghader et al. (2018).
211 Diverse ensembles by themselves cannot guarantee faster recovery from drifts, but can help to reduce the
212 initial increase in error caused by a drift Minku et al. (2010). There are several ways to achieve diversity
213 in the ensemble. Either the classifiers are trained on different data samples, or the model is composed of a
214 set of heterogeneous classifiers. Recently, Khamassi et al. (Khamassi et al. (2019)) studied the influence
215 of diversity techniques (block-based, weighting-data, and filtering-data) on adaptive ensemble models
216 and designed a new ensemble approach that combines the three diversity techniques. The authors in
217 (Sidhu and Bhatia (2018)) experimented with a diversified, dynamic weighted majority voting approach
218 consisting of two ensembles (with low and high diversity, achieved by replacing the Poisson (1) with
219 Poisson (κ) distribution in online bagging (Oza and Russell (2001)). The Kappa Updated Ensemble
220 (KUE) Cano and Krawczyk (2020) trains its base learners using different subsets of features and updates
221 them with new instances with a given probability following a Poisson distribution. Such an approach
222 results in a higher ensemble diversity and outperforms most of the current adaptive ensembles. However,
223 there are not many studies where the model uses the model diversity score as a criterion for the base

224 classifiers in the ensemble (Krawczyk et al. (2017)) as opposed to static data processing, where such a
225 complex model exists (Lysiak et al. (2014)). According to (Yang (2011)), diversity correlates with model
226 accuracy. A suitable diversity metric used in ensembles is a paired diversity Q statistic (Kuncheva (2006)),
227 which provides information about differences between two base classifiers in the ensemble.

228   Another aspect of the ensemble classifiers is the composition of the base classifiers in the model. The
229 most common are homogeneous ensemble methods, which use the same algorithm to train the ensemble
230 members (Fernandez-Aleman et al. (2019)). On the other hand, heterogeneous approaches are based on
231 the utilization of multiple algorithms to generate ensemble members. Such an approach could lead to the
232 creation of more diverse ensembles. For the data stream classification, a HEFT-Stream (Heterogeneous
233 Ensemble with Feature drift for Data Streams) Nguyen et al. (2012) builds a heterogeneous ensemble
234 composed of different online classifiers (e.g., Online Naive Bayes). Adaptive modifications of the
235 heterogeneous ensembles were also successfully applied on the drifting data streams (Van Rijn et al.
236 (2016); Frías-Blanco et al. (2016); van Rijn et al. (2018); Idrees et al. (2020)), and many of them proved
237 suitable to address issues such as class imbalance (Large et al. (2017); Fernández et al. (2018); Ren et al.
238 (2018); Wang et al. (2018); Ghaderi Zefrehi and Altınçay (2020)). The approach described in this paper
239 aims to combine the construction of the adaptive heterogeneous ensemble with a diversity-based update
240 of the ensemble members. This approach could result in a robust model, with the adaptation mechanism
241 ensuring that newly added members are as diverse as possible during the ensemble updates. Maintaining
242 the diversity of the overall model can also lead to a reduction of model updates and therefore faster
243 execution during the run time.

DDCW ENSEMBLE METHOD

245 In the following section, we introduce the design of the Diversified Dynamic Class Weighted (DDCW)
246 ensemble model. The design of the model is based on the assumption that a robust model consists of
247 a collection of heterogeneous base classifiers that are very diverse. When applied to static data, the
248 diversity is used within the ensemble models to tune the combination rule for voting and the aggregation
249 of component classifier predictions. We propose the design of a heterogeneous ensemble model, which
250 combines the dynamic weighting of the ensemble members with the mutual diversity score criterion. The
251 diversity measures are used to rank the members within the ensemble and update their weights according
252 to the diversity value, so the model prefers experts with higher mutual diversity, thereby creating a more
253 robust ensemble. When ranking the base classifiers, the diversity measurement is combined with the
254 lifetime of individual base classifiers in the model. The criterion is expected to cause the importance
255 of the long-lasting base classifiers to gradually fade, which should ensure the relevance of the whole
256 ensemble to evolving and changing data streams.

257   The model is composed of m ensemble members $e_1, \ldots, e_m$, trained using each chunk of incoming
258 samples in the stream, as depicted in Fig. 2. Each of those experts $e_1, \ldots, e_m$, have assigned weights
259 for each target class. The weights are tuned after each period (after each chunk is processed) based on
260 individual base classifier performance. First, for each class that a base classifier predicts correctly, the
261 weight is increased. Second, after each chunk is processed, the model calculates Q pairwise diversity
262 between each of the ensemble members and uses this value to modify model weights.

263   Pairwise Q diversity metric is calculated as follows (Kuncheva and Whitaker (2003)): let $Z = z_1, ..., z_N$
264 be a labeled data set, $z_j \in \mathbb{R}^n$ coming from the classification problem. The output of a classifier D_i is an
265 N-dimensional binary vector $y_i = [y_{1,i}, ..., y_{N,i}]^T$, such that $y_{j,i} = 1$, if D_i recognizes correctly z_j, and 0
266 otherwise, $i = 1, ..., L$. Q statistic for two classifiers, D_i and D_k, is then computed as: $Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$
267 where N^{ab} is the number of elements z_j of Z for which $y_{j,i} = a$ and $y_{j,k} = b$. Q varies between -1 and 1;
268 classifiers that tend to recognize the same samples correctly will have positive values of Q, and those that
269 commit errors on different objects will render Q negative. For statistically independent classifiers, the
270 value of $Q_{i,k}$ is 0.

271   The value for each member div_e_i in DDCW model is obtained as the average of contributions of
272 individual pair diversities and is calculated as follows: $div\_e_i = \frac{1}{m-1} \sum_{k=1, k \neq i}^{m} Q_{i,k}$. Then, after each period,
273 the lifetime coefficient T_i of each ensemble member is increased. Afterwards, the weights of each of the
274 base classifiers are modified using the lifetime coefficient. After this step, the weights are normalized for
275 each class, and a score is calculated for each target class by the classifier predictions. In the last step, the
276 global ensemble model prediction is selected as a target class with the highest score.
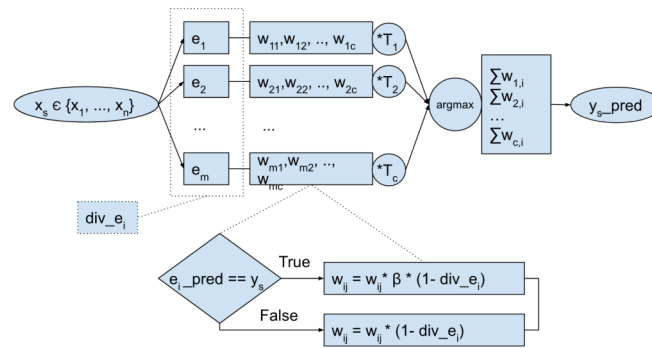
Figure 2. Overall scheme of the proposed ensemble model.

Model weights can be represented as a matrix $W_{m \times c}$, where m is a number of classifiers in the ensemble, and c is a number of target classes in the data. The weights $w_{i,j}$ directly determine the weight given to classifier i for class j as seen in Table 1. In the beginning, the weights are initialized equally, based on the number of classes in the data. During the process, the individual weights for each base classifier and corresponding target class are modified using the parameter β. The weight matrix allows the calculation of the score of the base classifiers, as well as the score of predicted target classes. The score of the classifier is calculated as $\sum_{j=1}^{c} w_{i,j}$ for each classifier. This score allows the identification of poorly performing base classifiers in the ensemble. However, the score of the target class is calculated as the contribution of weight $w_{i,j}$ of classifier i and its predicted class j.

The weight matrix enables the efficient contribution of each of the ensemble members in building the model. We proceeded from the assumption of having a diverse ensemble. Such a model could consist of members, which perform differently in specific target class values, e.g. some of the members are more precise when predicting a particular class than others and vice versa. In that case, we can use a set of classifiers; each of them focused on a different target class (while the distribution of these classes may change over time). The class weighting alone may lead to an ensemble consisting of similar, well-performing models. But the combination of class weighting with diversity measure can lead to a set of balanced members, more focused on specific classes and complementing each other.

	C_1	C_2	...	C_c	Classifier score
e_1	$w_{1,1}$	$w_{1,2}$	...	$w_{1,c}$	$\sum_{j=1}^{c} w_{1,j}$
e_2	$w_{2,1}$	$w_{2,2}$	...	$w_{2,c}$	$\sum_{j=1}^{c} w_{2,j}$
...	...	...	...	...	...
e_m	$w_{m,1}$	$w_{m,2}$	...	$w_{m,c}$	$\sum_{j=1}^{c} w_{m,j}$

Table 1. Weight matrix.

The proposed ensemble model belongs to the category of passive chunk-based adaptive models. In the presented approach, the size of the model dynamically changes according to the global model performance. The minimum size of the model is set by the parameter k, and the maximum size is set by the parameter l.

Each base classifier in the ensemble is assigned with a weight vector for each target class. If a new target class appears in the chunk used in training, the new target will be added to the weight vector for each base classifier. Initially, the ensemble consists of a randomly initialized set from a list of defined base algorithms (Hoeffding Tree or Naive Bayes). Other experts can be added in the following periods (interval representing the chunk of arriving data, where base learners and their weights are modified) until the minimum size of the model is reached, i.e. either the model size is smaller than the defined minimum size or the particular member weight falls under the defined threshold.

In each iteration, the experts are used to predict the target class of incoming samples in the processed chunk (Lines 3-5). If a prediction of an expert is correct, the weights of the particular expert and target class are multiplied by a coefficient β (Line 7). In the case period p has occurred, Q statistic diversity is

307 calculated for each pair of experts in the ensemble, and the weights of each expert is modified using the
308 particular expert's diversity (Line 12 and 16). This mechanism enables the construction of more robust
309 ensembles, by preferring the diverse base models. The weights of base classifiers are also reduced by
310 the exponential value of their lifetime in the ensemble (Line 15). In this case, the lifetime of the expert
311 represents the number of periods since its addition to the ensemble. The exponential function is used,
312 so the experts are influenced minimally during their initial periods in the ensemble but become more
313 significant for the long-lasting members. This implementation works as a gradual forgetting mechanism
314 of the ensemble model, as the weakest experts are gradually removed from the model and replaced by the
315 new ones.

316 After the update, the weights are normalized for each target class (Line 24). Afterwards, if the
317 maximum size of the model is reached and the global prediction is incorrect, the weakest expert is
318 removed from the ensemble (Line 27). A new random expert can then be added to the ensemble (Lines
319 30-31). In each period, all experts where the sum of weights is lower than defined threshold θ are
320 removed from the ensemble. In the end, each sample is weighted by a random uniform value m times,
321 where m represents the actual size of ensemble (Line 41). Each expert is than trained with a new set
322 of incoming samples, with individual weights from the last chunk of data (Line 42). After training, the
323 global predictions of actual samples are retrieved, and the algorithm then continues back to Line 3.

Algorithm 1. Diversified Dynamic Class Weighted ensemble

Procedure: DDCW($\{X, Y\}, p, k, l, \alpha, \beta, \theta$)
Input: Data and labels $\{x,y\}$, chunk size p, min_experts k, max_experts l, fading factor α, multiplier β, threshold θ
Output: Global predictions G

```
1:  Experts ← create_random_experts(k);
2:  initialize class weights w_i,j
3:  for s = 0,...,n do
4:      for i = 1,..., num_experts(Experts) do
5:          Local_predictions = classify(Experts_i, x_s);
6:          if Local_predictions = y_s then
7:              w_{i,L} = β * w_{i,L}; ← Multiply weight of particular expert and target class from local prediction by β
8:          end if
9:      end for
10:     if all samples in a chunk are processed then
11:         Local_predictions = classify(Experts, x_s);
12:         Diversity = calculate_diversity(Local_predictions, y_s);
13:         for i = 1,..., num_experts(Experts) do
14:             expert_lifetime ← Increase expert lifetime in each period;
15:             w_i = w_i - (exp(α * expert_lifetime) - 1)/10;
16:             w_i = w_i * (1 - Diversity_i);
17:         end for
18:     end if
19:     for j = 0,...,Class_labels do
20:         Global_predictions_j ← sum(w_j);
21:     end for
22:     Global_predictions ← argmax(Global_predictions_j);
23:     if all samples in chunk are processed then
24:         w ← normalize_weights(w);
25:         if Global_predictions_s != y_s then
26:             if num_experts(Experts) == l then
27:                 {Experts, w, expert_lifetime} ← Remove weakest expert e_i based on experts score
28:             end if
29:             if num_experts(Experts) < l then
30:                 Experts_new ← create_random_expert();
31:                 w_new ← 1/num_experts(Experts);
32:             end if
33:         end if
34:         {Experts, w, expert_lifetime} ← Remove experts which score is below threshold θ
35:         if num_experts(Experts) < k then
36:             Experts_new ← create_random_expert();
37:             w_new ← 1/num_experts(Experts);
38:         end if
39:     end if
40:     for i = 1,..., num_experts(Experts) do
41:         Sample_weights_s ← random_uniform_weight();
42:         Experts_i ← learn_expert(Experts_i, x_s, y_s, Sample_weights_s);
43:     end for
44:     return Global_predictions;
45: end for
```

DATASETS DESCRIPTION

To experimentally evaluate the performance of the presented approach, we decided to use both real-world and synthetically generated datasets. We tried to include multiple drifting datasets that contain different types of concept drift. Datasets used in the experiments are summarized in Table 2.

Dataset	Drift type	Dataset type	Samples	Features	Classes
ELEC	?	R	45,312	6	2
KDD99	?	R	494,021	41	23
AIRL	?	R	539,383	7	2
COVT	?	R	581,012	54	7
SHUTTLE	?	R	58,000	9	7
POWERSUPPLY	?	R	29,928	3	24
CONNECT-4	?	R	65,557	43	3
BNG_BRIDGES	?	R	1,000,000	13	6
BNG_BRIDGES1vsAll	?	R	1,000,000	13	6
BNG_HEPATITIS	?	R	1,000,000	20	2
BNG_ZOO	?	R	1,000,000	17	7
BNG_LYMPH	?	R	1,000,000	19	4
AGR_a	A	S	1,000,000	9	2
AGR_g	G	S	1,000,000	9	2
SEA_a	A	S	1,000,000	3	2
SEA_g	G	S	1,000,000	3	2
STAGGER	A	S	100,000	3	2
LED	G	S	100,000	24	10
MIXED_BALANCED	A	S	1,000,000	5	2
MIXED_IMBALANCED	A	S	1,000,000	5	2
RBF	-	S	1,000,000	50	4
RBF_DRIFT	G	S	1,000,000	50	4
WAVEFORM	-	S	1,000,000	40	3
WAVEFORM_DRIFT	G	S	1,000,000	40	3

Table 2. Datasets used in the experiments. [Dataset type] R: real, S: synthetic. [Drift type] A: abrupt, G: gradual, -: none, ?: unknown.

Real datasets

In our study, we used 12 real datasets, including the frequently used ELEC dataset (Harries et al. (1999)), the KDD 99 challenge dataset (Tavallaee et al. (2009)), Covtype (Blackard (1998)), the Airlines dataset introduced by Ikonomovska [1], and data streams from the OpenML platform Bischl et al. (2019) generated from a real-world dataset using a Bayesian Network Generator (BNG) van Rijn et al. (2014). We included a wide range of datasets to evaluate the performance of the DDCW model on datasets with both binary and multi-class targets or with balanced and imbalanced classes, especially some of them, such as KDD99 and Shuttle are heavily imbalanced. To examine the imbalance degree of the datasets, we included the class ratios in the documentation on the GitHub repository [2]. As it is difficult to determine the type of a real drift contained in such data, we tried to estimate and visualize possible drift occurrences. Multiple techniques for concept drift visualization exist (Pratt and Tschapek (2003)). We used visualization based on feature importance (using the Gini impurity) and the respective changes within the datasets, as they may signalize changes in concepts in the data. Based on the work described in (Cassidy and Deviney (2015)), we used feature importance scores derived from the Online Random Forest model trained on the datasets. Such an approach can help to visualize the so-called feature drift, which occurs when certain features stop (or start) being relevant to the learning task. Fig. 3 shows such visualizations for the real-world datasets used in the experiments. The visualization depicts how the importance of the features

---

[1] http://kt.ijs.si/elena_ikonomovska/data.html

[2] https://github.com/Miso-K/DDCW/blob/master/class_ratios.txt

changes over time in the data. *x*-axis represents number of samples, *y*-axis feature indices and the size of the dots correspond to a feature importance in the given chunk.
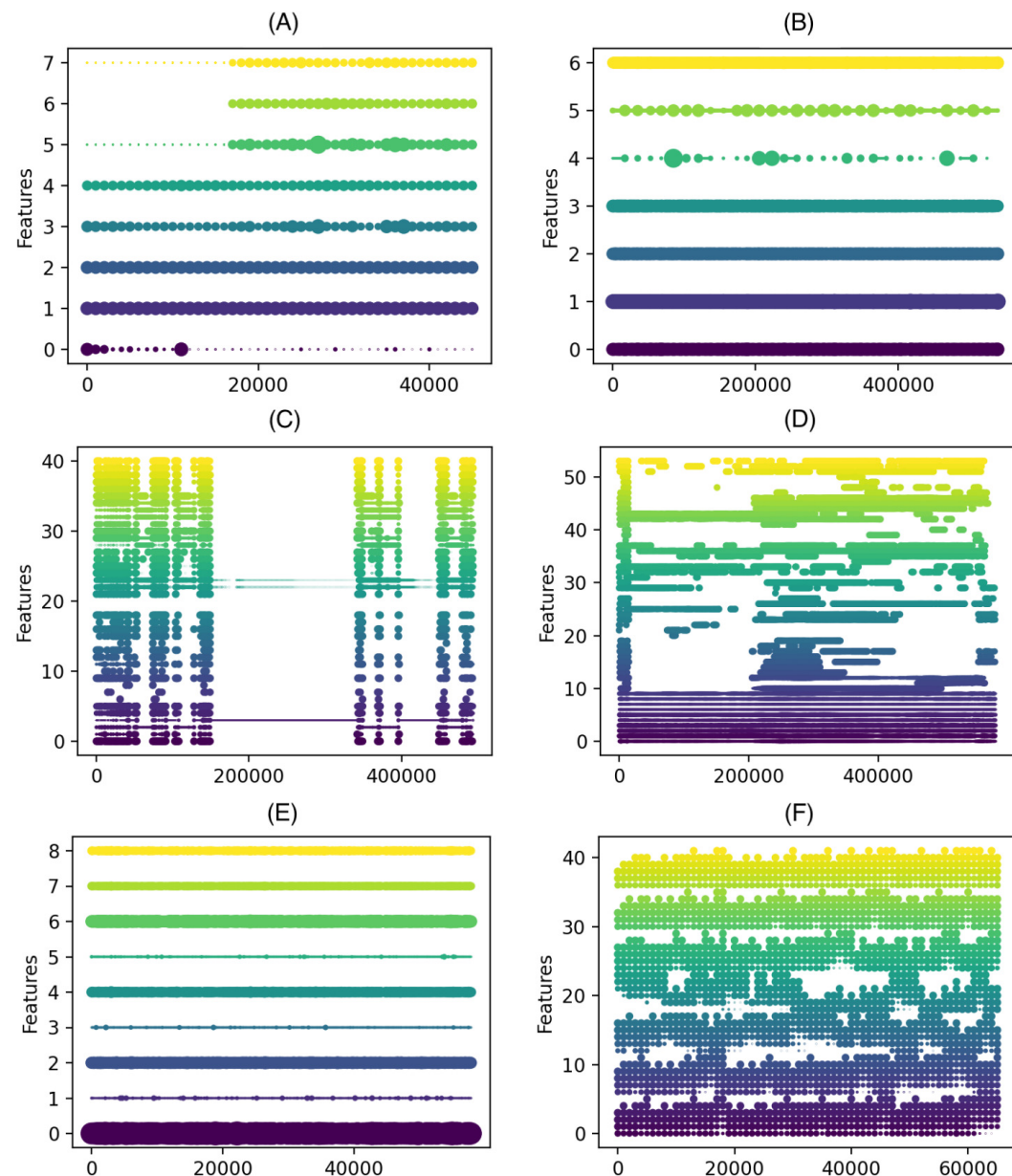
Figure 3. Feature importance progress in the real-world datasets. (A) ELEC, (B) Airlines, (C) KDD 99, (D) Covtype, (E) Shuttle, (F) Connect-4. [*x* axis] number of samples, [*y* axis] feature indices, the size of the dots correspond to a feature importance in the given chunk.

Synthetic datasets

Besides the real-world datasets, we used synthetic data streams containing generators with various types of drifts. In most cases (except LED and STAGGER data), we used streams of 1,000,000 samples, with three simulated drifts. We used the Agrawal generator (Agrawal et al. (1993)) and SEA (Nick Street and Kim (2001)) generators with abrupt and gradual drifts, RBF and Waveform streams without any drift and with simulated gradual drift, a Stagger Concepts Generator (Schlimmer and Granger (1986)) with abrupt drift, an LED (Gordon et al. (1984)) stream with gradual drift, and a Mixed stream with an abrupt drift with balanced and imbalanced target attributes.

EXPERIMENTAL RESULTS

The purpose of the experiments was to determine the performance of the DDCW model in a series of different tests. We used the python implementation of all models, and the experiments were performed using the Scikit-multiflow framework (Montiel et al. (2018)). All experiments were performed on a virtual server equipped with 6 CPU cores and 8 GB RAM.

During the first series of experiments, we aimed to examine the impact of the different setting of the chunk-size parameter in which the model is updated and the diversity is calculated. The model was tested with varying sizes of the chunk, with values set to 100, 200, 300, 400, 500, and 1000 samples on all considered datasets. The primary motivation of the experiments was to find the most suitable chunk size parameter for different datasets, which will be used to compare the DDCW with other ensemble models. To evaluate the models, we used prequential evaluation (or interleaved test-then-train evaluation), in which the testing is performed on the new data before they are used to train the model.

In the second set of the experiments, the main goal was to compare the performance of the DDCW model with the selected other streaming ensemble-based classifiers. We considered multiple ensemble models: DWM, AWE, Online Boosting, OzaBagging, and the currently best performing streaming ensembles such as ARF and KUE. To analyze the performance of these models, standard classification metrics were used (accuracy, precision, recall, and F1). Besides the comparison of the model performance, we measured the metrics related to resource consumption, such as total time required for training, scoring time of an instance, and memory requirements of the models. During these experiments, we set the chunk size to 1000 samples (although we included DDCW with a chunk size of 100 samples for comparison) and set the model's hyper-parameters to create similar-sized ensemble models (min. 5 and max. 20 members in the ensemble).

Performance with the different chunk sizes

In this experiment, we explored the influence of the chunk window on the classifier's performance on different datasets. The main goal was to find the optimal chunk window size for a particular dataset. We set different chunk sizes and measured the model performance using selected metrics in defined periods (e.g., every 100 samples). We computed the average model performance on the entire dataset using the above-mentioned classification metrics. A comparison of the DDCW classifier accuracy and F1 with different sizes of the chunks is shown in Table 3.

Besides setting the chunk size, we fine-tuned the model hyper-parameters. Our main objective was to estimate the suitable combinations of the parameters α, β, and θ. As the experiments were computationally intensive, most of the models were trained using the default hyper-parameter settings, with particular values set to $\alpha = 0.002$, $\beta = 3$, and $\theta = 0.02$. Regarding the model behavior with different hyper-parameter settings, α influences the lifetime of an expert in the ensemble and the speed of degradation of the expert score with increasing lifetime. Increasing values led usually to a more dynamic model, able to adapt rapidly, while lower values led to a more stable composition of the model. β influences the preference of the experts, which classified the samples correctly. Higher values of this parameter can suppress the poor-performing experts and raise the probability of updating them in the following iteration. θ serves as a threshold for the expert update. Lower values usually lead to more weak experts in the ensemble, a marginal contribution to the performance , but raise the model complexity significantly. Higher values force the updating of weak experts. However, some of them may be missing later on, after drift occurrence and the reappearance of previous concepts.

The results proved, that the chunk size does have an impact on the model performance, and there are mostly minor differences in the performance metrics with different chunk size parameter setting. Although accuracy is not affected much, F1 metric improves significantly with larger chunk sizes, especially on the BNG_Zoo and BNG_Lymph datasets. In general, we can observe, that on the larger data (with more than 100,000 samples in the stream), larger windows resulted in slightly better performance. On the other hand, smaller chunk sizes enable the model to react more quickly to concept drift. In some cases, the accuracy metric proved to be not very useful, as the target class is strongly unbalanced or multi-class. It is evident mostly on the KDD99 or BNG_Lypmh datasets, where high accuracy values are caused mainly by the classification into the majority class, while other minor classes do not influence this metric very much. A much better perspective on the actual model performance could be given by F1 measure.

The experiments summarized averaged results that the models achieved on the entire stream, but it is also important to explore how the performance progressed during the stream processing and observe their

Chunk size	100	200	300	400	500	1000
	accuracy	accuracy	accuracy	accuracy	accuracy	accuracy
ELEC	**0.849**	0.842	0.831	0.847	0.832	0.810
KDD99	0.995	0.995	0.995	**0.996**	0.995	0.991
AIRL	0.636	0.641	0.644	0.645	0.645	**0.649**
COVT	**0.849**	0.842	0.831	0.847	0.832	0.810
SHUTTLE	0.953	0.955	0.965	0.975	0.953	**0.992**
POWERSUPLY	0.156	0.155	0.156	0.153	0.155	**0.157**
CONNECT4	0.686	0.693	0.687	0.669	0.674	**0.705**
BNG_BRIDGES	0.689	0.703	0.711	0.715	0.718	**0.725**
BNG_BRIDGES1vsAll	0.962	0.963	0.964	0.965	0.965	**0.966**
BNG_HEPATITIS	0.858	0.867	0.870	0.873	0.874	**0.897**
BNG_ZOO	0.876	0.892	0.900	0.905	0.908	**0.921**
BNG_LYMPH	0.796	0.811	0.819	0.824	0.830	**0.879**
AGR_a	0.857	0.871	0.874	0.876	0.878	**0.879**
AGR_g	0.827	0.847	0.852	0.856	0.859	**0.874**
SEA_a	0.873	0.876	0.877	0.878	0.881	**0.888**
SEA_g	0.868	0.874	0.873	0.874	0.877	**0.884**
STAGGER	**0.946**	0.946	0.933	0.938	0.912	0.923
LED	**0.892**	0.888	0.882	0.884	0.884	0.860
MIXED_BALANCED	0.927	0.934	0.935	0.939	0.943	**0.964**
MIXED_IMBALANCED	0.924	0.930	0.932	0.936	0.939	**0.964**
RBF	0.855	0.872	0.877	0.879	0.881	**0.882**
RBF_DRIFT	0.546	0.562	0.573	0.585	0.592	**0.601**
WAVEFORM	0.819	0.826	0.830	0.832	0.835	**0.837**
WAVEFORM_DRIFT	0.820	0.826	0.829	0.834	0.835	**0.836**
	F1	F1	F1	F1	F1	F1
ELEC	**0.815**	0.807	0.791	0.811	0.791	0.760
KDD99	0.570	0.591	**0.602**	0.581	0.582	0.596
AIRL	0.531	**0.538**	0.533	0.537	0.529	0.535
COVT	**0.815**	0.807	0.791	0.811	0.791	0.760
SHUTTLE	0.506	0.510	0.536	0.606	0.672	**0.702**
POWERSUPLY	0.109	0.110	0.109	0.105	0.108	**0.110**
CONNECT4	0.483	**0.486**	0.470	0.466	0.481	0.464
BNG_BRIDGES	0.588	0.609	0.620	0.627	0.632	**0.635**
BNG_BRIDGES1vsAll	0.866	0.872	0.873	0.877	0.877	**0.883**
BNG_HEPATITIS	0.907	0.913	0.915	0.918	0.918	**0.935**
BNG_ZOO	0.784	0.806	0.818	0.825	0.831	**0.853**
BNG_LYMPH	0.548	0.577	0.595	0.607	0.618	**0.752**
AGR_a	0.842	0.858	0.864	0.868	0.868	**0.869**
AGR_g	0.808	0.830	0.835	0.840	0.845	**0.874**
SEA_a	0.895	0.898	0.899	0.900	0.902	**0.909**
SEA_g	0.892	0.896	0.896	0.897	0.899	**0.906**
STAGGER	**0.949**	0.949	0.937	0.941	0.919	0.928
LED	**0.892**	0.888	0.882	0.884	0.884	0.860
MIXED_BALANCED	0.928	0.934	0.935	0.939	0.943	**0.964**
MIXED_IMBALANCED	0.930	0.935	0.936	0.940	0.943	**0.966**
RBF	0.850	0.868	0.873	0.875	0.878	**0.881**
RBF_DRIFT	0.528	0.550	0.562	0.570	0.576	**0.618**
WAVEFORM	0.814	0.822	0.826	0.828	0.832	**0.834**
WAVEFORM_DRIFT	0.814	0.822	0.824	0.830	0.832	**0.834**

Table 3. Performance of the DDCW model with different chunk sizes.

reactions to concept drift.

Fig. 4 visualizes the accuracy achieved by the DDCW models on the real datasets with both chunk sizes. The performance of the method with both settings is overall similar; however, we can see a difference in cases when a change (possible drift) occurs. On the KDD99 dataset, there is a significant decrease in the accuracy of the model after around 52,000 samples. Shorter chunk windows resulted in a much earlier reaction to the change, without any significant decrease in performance. On the Elec and Covtype datasets, the earlier reactions are also present and visible, resulting in higher performance metrics.

Fig. 5 depicts the DDCW model performance on the synthetic datasets with both chunk sizes. In the
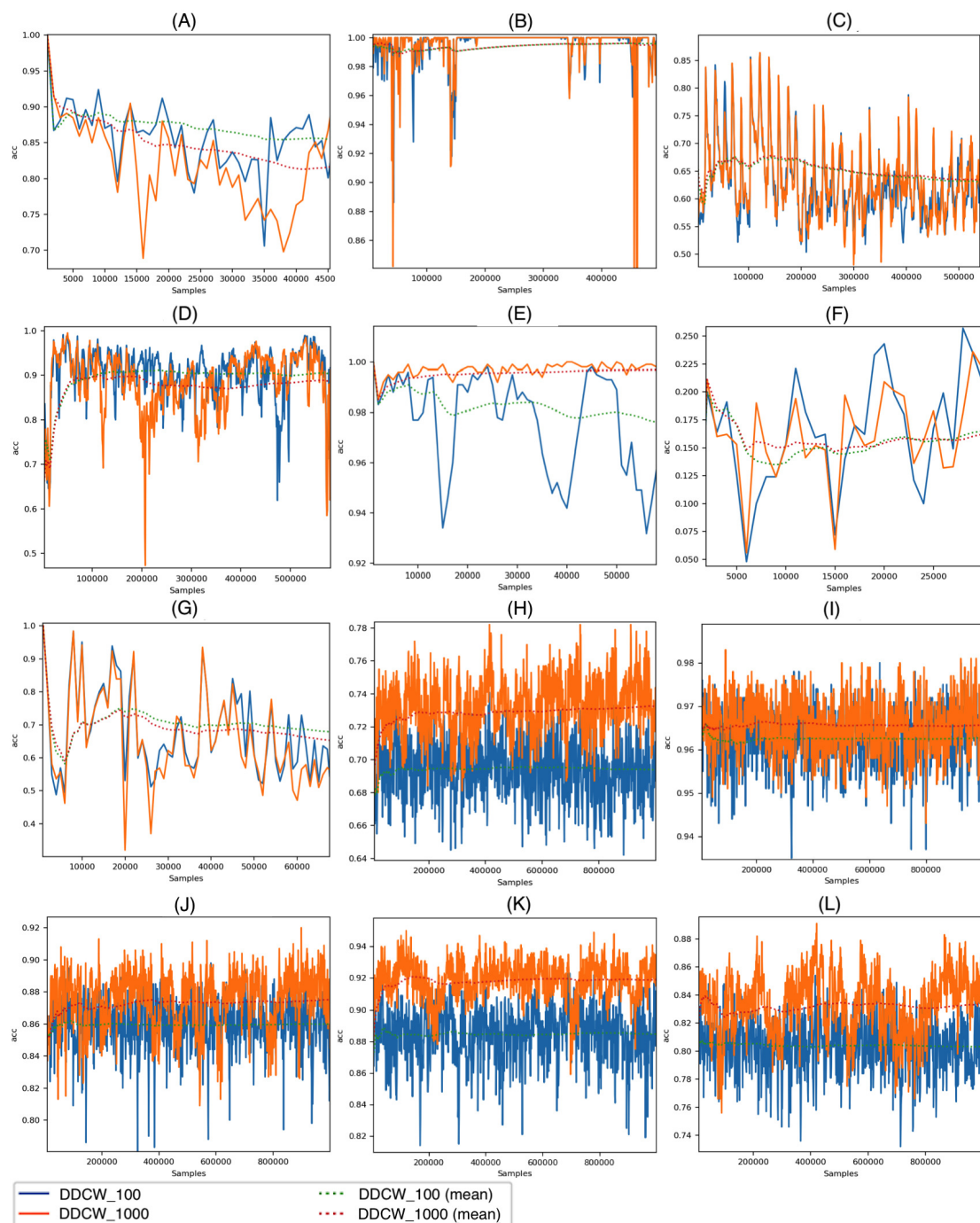
Figure 4. Performance of the DDCW model on the real datasets. (A) ELEC, (B) KDD 99, (C) Airlines, (D) Covtype, (E) Shuttle, (F) Powersupply, (G) Connect-4, (H) BNG_Bridges, (I) BNG_Bridges1vsAll, (J) BNG_Hepatitis, (K) BNG_Zoo, (L) BNG_Lymph. [*y* axis] accuracy, [*x* axis] number of samples.

467 case of Stagger and LED datasets, the effects of different chunk sizes are comparable with the impact
468 on the real datasets. Larger chunk sizes lead to later reactions to drift and a more significant decrease in
469 accuracy. In contrast, performance evaluation on larger streams, such as AGR or Mixed streams, showed
470 that the chunk size effect on stream processing is different. Contrary to previous datasets, larger chunk
471 sizes resulted in a more robust model, still covering some of the previous concepts after drift occurrence.
472 After the drift, the model performance dropped significantly. When using larger chunk sizes, the model

473 was able to use more data to update the ensemble, which led to improved performance.
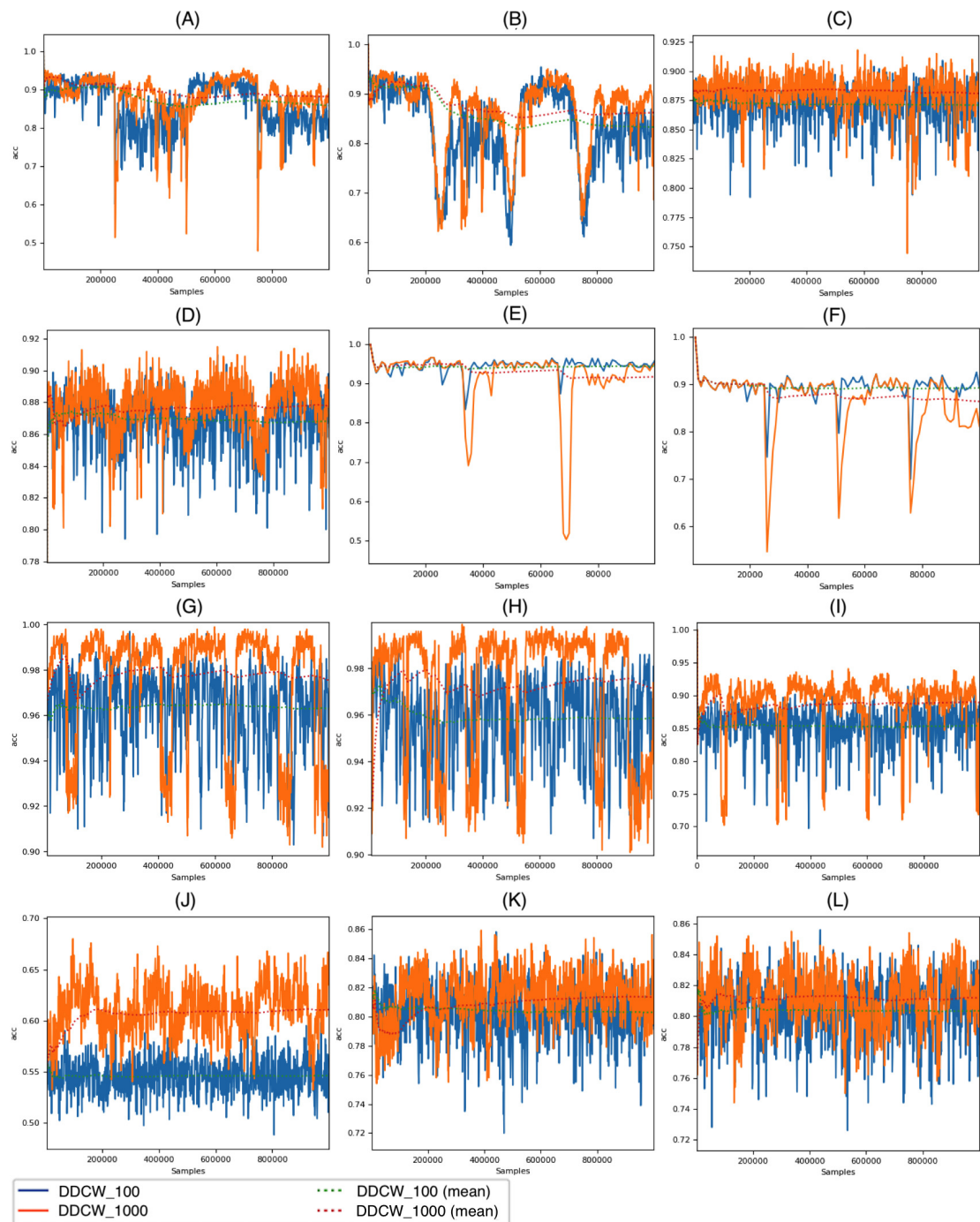
Figure 5. Performance of the DDCW model on the synthetic datasets. (A) AGR_a, (B) AGR_g, (C) SEA_a, (D) SEA_g, (E) Stagged, (F) LED, (G) Mixed-balanced, (H) Mixed-imbalanced, (I) RBF, (J) RBF_Drift, (K) Waveform, (L) Waveform_drift. [y axis] accuracy, [x axis] number of samples.

474 Fig. 6 depicts the size of the ensemble during stream processing on selected datasets with the diversity
475 of the ensemble members disabled and enabled. During the experiments, we set the minimum size of the
476 DDCW ensemble to 5 and the maximum size to 20. Initially, we performed a set of tests to estimate the
477 optimal ensemble size. Larger ensembles did not perform significantly better, while the computational
478 complexity of the model was considerably higher. The experiments proved that, during the run time, the

algorithm preferred a smaller pool of ensemble members. The algorithm added more ensemble members
when a concept drift occurred. Enabling the diversity-based selection of experts resulted in a more stable
composition of the ensemble and required fewer member updates.
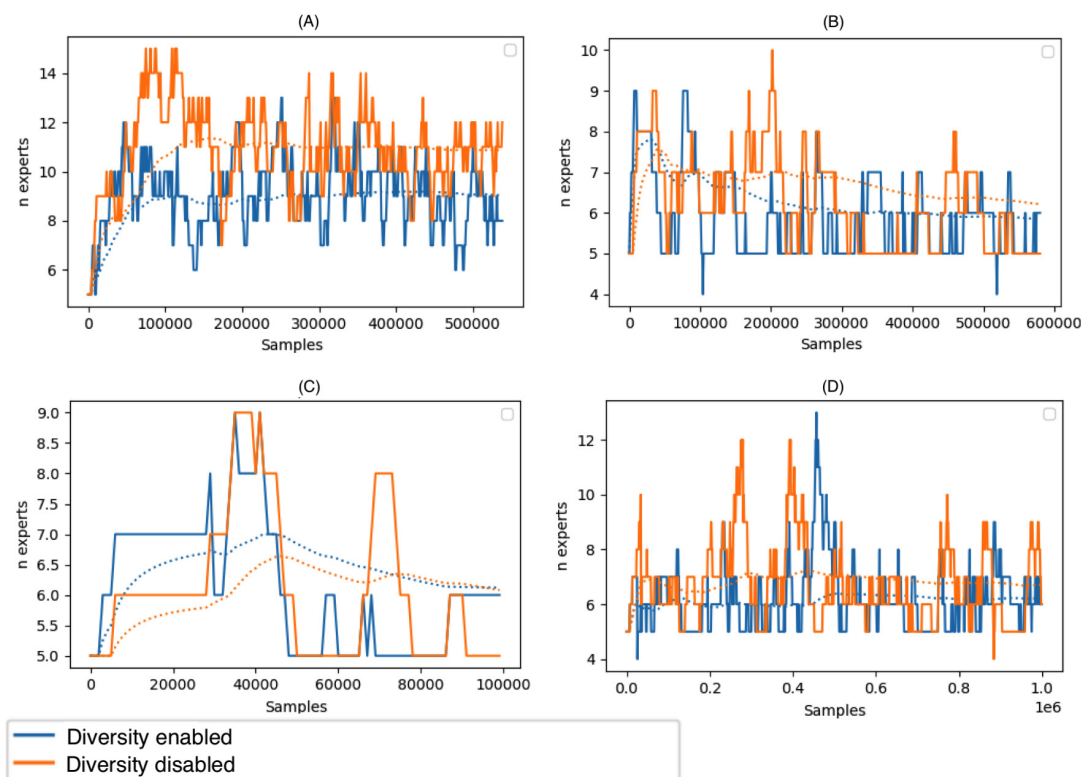
Figure 6. Size of the DDCW ensemble during the run-time on the selected datasets. (A) Airlines, (B) Covtype, (C) Stagger, (D) AGR_a. [y axis] Number of ensemble members, [x axis] number of samples.

Comparison with other ensemble models

	DDCW$_{HT}$	DDCW$_{HTNB}$	DWM$_{NB}$	AWE$_{NB}$	DWM$_{HT}$	AWE$_{HT}$	OB$_{kNN}$	Oza$_{kNN}$	OB$_{HT}$	Oza$_{HT}$	OB$_{NB}$	Oza$_{NB}$	ARF$_{HT}$	KUE$_{HT}$
Accuracy														
ELEC	0.853	0.810	0.800	0.756	0.869	0.788	0.765	0.780	0.858	0.793	0.792	0.734	0.857	0.668
KDD99	0.995	0.991	0.983	0.420	0.989	0.103	0.999	0.998	0.998	0.995	0.995	0.946	0.999	0.999
AIRL	0.662	0.649	0.640	0.618	0.620	0.575	0.587	0.639	0.634	0.653	0.619	0.644	0.666	0.663
COVT	0.853	0.810	0.823	0.592	0.812	0.215	0.927	0.918	0.876	0.871	0.783	0.871	0.941	0.904
SHUTTLE	0.996	0.992	0.896	0.949	0.946	0.949	0.990	0.991	0.982	0.978	0.950	0.922	0.998	0.997
POWERSUPLY	0.158	0.158	0.074	0.186	0.074	0.187	0.029	0.167	0.013	0.162	0.007	0.161	0.156	0.218
CONNECT4	0.719	0.705	0.671	0.286	0.693	0.613	0.702	0.728	0.675	0.696	0.636	0.572	0.739	0.650
BNG_BRIDGES	0.737	0.725	0.611	0.698	0.612	0.698	0.621	0.670	0.699	0.750	0.687	0.684	0.756	0.738
BNG_BRIDGES1vsAll	0.970	0.966	0.962	0.967	0.962	0.967	0.936	0.958	0.970	0.973	0.957	0.962	0.973	0.958
BNG_HEPATITIS	0.909	0.897	0.854	0.877	0.856	0.913	0.839	0.884	0.913	0.920	0.868	0.853	0.922	0.923
BNG_ZOO	0.928	0.921	0.812	0.913	0.806	0.913	0.909	0.927	0.931	0.915	0.903	0.889	0.942	0.939
BNG_LYMPH	0.883	0.878	0.801	0.825	0.802	0.825	0.787	0.828	0.846	0.809	0.818	0.806	0.871	0.903
F1														
ELEC	0.826	0.760	0.749	0.694	0.852	0.757	0.721	0.730	0.828	0.744	0.722	0.595	0.825	0.668
KDD99	0.587	0.549	0.502	0.045	0.397	0.029	0.712	0.642	0.593	0.645	0.577	0.530	0.649	NaN
AIRL	0.556	0.535	0.320	0.293	0.534	0.425	0.436	0.430	0.563	0.522	0.534	0.259	0.577	NaN
COVT	0.826	0.760	0.594	0.156	0.569	0.085	0.740	0.718	0.721	0.675	0.626	0.679	0.779	NaN
SHUTTLE	0.611	0.702	0.397	0.398	0.446	0.399	0.517	0.435	0.623	0.452	0.530	0.621	0.677	NaN
POWERSUPLY	0.113	0.110	0.066	0.134	0.066	0.134	0.034	0.161	0.106	0.014	0.009	0.107	0.149	NaN
CONNECT4	0.470	0.464	0.466	0.272	0.437	0.382	0.587	0.517	0.431	0.406	0.420	0.381	0.496	0.506
BNG_BRIDGES	0.649	0.635	0.515	0.601	0.515	0.601	0.527	0.567	0.611	0.669	0.599	0.596	0.673	0.656
BNG_BRIDGES1vsAll	0.895	0.883	0.871	0.886	0.869	0.886	0.790	0.857	0.899	0.906	0.852	0.872	0.906	0.915
BNG_HEPATITIS	0.943	0.935	0.904	0.921	0.907	0.922	0.898	0.929	0.945	0.950	0.914	0.902	0.952	0.887
BNG_ZOO	0.860	0.853	0.714	0.832	0.704	0.832	0.824	0.855	0.865	0.844	0.821	0.821	0.887	0.882
BNG_LYMPH	0.740	0.752	0.610	0.603	0.609	0.603	0.602	0.584	0.735	0.658	0.679	0.659	0.636	0.827

Table 4. Comparison of accuracy and F1 metrics of evaluated ensemble models on the real data streams.

	DDCW$_{HT}$	DDCW$_{HTNB}$	DWM$_{NB}$	AWE$_{NB}$	DWM$_{HT}$	AWE$_{HT}$	OB$_{kNN}$	Oza$_{kNN}$	OB$_{HT}$	Oza$_{HT}$	OB$_{NB}$	Oza$_{NB}$	ARF$_{HT}$	KUE$_{HT}$
Accuracy														
AGR$_a$	0.918	0.879	0.770	0.821	0.810	0.822	0.611	0.662	0.818	0.913	0.816	0.658	0.926	0.943
AGR$_g$	0.896	0.874	0.750	0.800	0.792	0.800	0.600	0.649	0.794	0.845	0.786	0.658	0.897	0.943
SEA$_a$	0.890	0.888	0.876	0.879	0.874	0.878	0.753	0.863	0.871	0.868	0.846	0.854	0.894	0.902
SEA$_g$	0.887	0.884	0.874	0.876	0.872	0.875	0.750	0.860	0.867	0.868	0.845	0.854	0.891	0.910
STGR	0.937	0.923	0.901	0.948	0.947	0.947	0.929	0.945	0.941	0.872	0.924	0.602	0.949	0.962
LED	0.873	0.860	0.831	0.893	0.831	0.893	0.682	0.770	0.768	0.848	0.770	0.698	0.892	0.893
MIXED_BALANCED	0.984	0.964	0.916	0.919	0.973	0.919	0.967	0.976	0.922	0.988	0.912	0.920	0.995	0.998
MIXED_IMBALANCED	0.985	0.964	0.916	0.920	0.922	0.920	0.968	0.976	0.926	0.988	0.909	0.920	0.995	0.998
RBF	0.924	0.882	0.712	0.733	0.712	0.733	0.915	0.940	0.816	0.940	0.719	0.720	0.941	0.954
RBF_DRIFT	0.618	0.601	0.520	0.516	0.516	0.512	0.554	0.572	0.580	0.705	0.544	0.546	0.655	0.788
WAVEFORM	0.840	0.837	0.797	0.830	0.797	0.830	0.762	0.815	0.844	0.855	0.807	0.805	0.843	0.844
WAVEFORM_DRIFT	0.840	0.836	0.797	0.830	0.797	0.830	0.763	0.815	0.843	0.855	0.807	0.805	0.843	0.852
F1														
AGR$_a$	0.912	0.869	0.750	0.812	0.810	0.812	0.592	0.616	0.808	0.802	0.773	0.542	0.922	0.944
AGR$_g$	0.888	0.864	0.728	0.787	0.788	0.786	0.580	0.602	0.782	0.835	0.773	0.543	0.890	0.939
SEA$_a$	0.910	0.909	0.900	0.901	0.898	0.901	0.791	0.889	0.895	0.894	0.875	0.885	0.914	0.909
SEA$_g$	0.908	0.906	0.898	0.899	0.896	0.899	0.799	0.886	0.892	0.893	0.874	0.885	0.911	0.905
STGR	0.941	0.928	0.947	0.951	0.949	0.949	0.934	0.947	0.945	0.887	0.930	0.633	0.952	0.963
LED	0.873	0.860	0.832	0.893	0.832	0.893	0.682	0.768	0.768	0.848	0.770	0.693	0.892	0.893
MIXED_BALANCED	0.984	0.964	0.917	0.919	0.973	0.919	0.967	0.976	0.923	0.988	0.912	0.920	0.995	0.998
MIXED_IMBALANCED	0.985	0.966	0.923	0.925	0.928	0.926	0.969	0.977	0.928	0.989	0.914	0.927	0.996	0.998
RBF	0.923	0.881	0.716	0.738	0.716	0.738	0.916	0.941	0.816	0.939	0.721	0.720	0.941	0.954
RBF_DRIFT	0.620	0.619	0.545	0.533	0.521	0.517	0.559	0.585	0.578	0.703	0.577	0.595	0.644	0.788
WAVEFORM	0.839	0.834	0.784	0.824	0.784	0.824	0.762	0.814	0.843	0.854	0.796	0.792	0.842	0.844
WAVEFORM_DRIFT	0.840	0.834	0.784	0.824	0.784	0.824	0.763	0.814	0.842	0.854	0.795	0.792	0.842	0.852

Table 5. Comparison of accuracy and F1 metrics of evaluated ensemble models on the synthetic data streams.

In these experiments, we compared the DDCW model performance with the selected ensemble models. In the comparison, we included AWE, DWM, Online Boosting, OzaBagging, ARF, and KUE models. Each of the ensemble models was tested with different base learners. We evaluated Naive Bayes, Hoeffding Tree, and k-NN as base learners. Similar to the previous set of experiments, we used the accuracy, precision, recall, and F1 metrics for comparison purposes computed in the prequential fashion. When using the DDCW model, we included the DDCW model with a combination of Naive Bayes and Hoeffding trees as base learners, as well as a tree-based homogeneous ensemble alone.

Performance comparison

To summarize the experiments, Table 4 compares the performance of all evaluated models on the real datasets and Table 5 provides the similar comparison on the synthetic data streams. As in the previous experiments, the tables consists of overall averaged results that the models achieved on the entire stream. While most of the studies focus only on a comparison of accuracy, we decided to analyze other classification metrics as well. Especially in the case of multi-class or heavily imbalanced data (e.g., KDD99), accuracy might not be the best choice to truly evaluate the performance of the model, therefore we choose also F1 metric as well. Please note, that we were unable to properly obtain F1 values from the KUE model on some of the datasets.

The DDCW model proved to be suitable for data streams with different concept drifts and either binary or multi-class classification tasks. When considering the composition of the DDCW ensemble, the fact that the model relies on different base learners enables it to utilize the strengths of particular learners. Dynamic composition of the ensemble enables it to adapt to the particular stream by preferring a base learner that is more suitable for the given data. In general, the DDCW performs very well on the generated streams, gaining at least competitive results compared to the other models on the real-world datasets. The method appears to struggle more with some of the imbalanced datasets, as is apparent from the F1 results achieved on the KDD 99 or Airlines dataset. During this experiment, we also used two different DDCW method setups to compare the effect of the base learner selection. We used DDCW with only Hoeffding trees as a base learner and DDCW with a combination of Naive Bayes and Hoeffding trees. Although the homogeneous ensemble mostly performed slightly better, the heterogeneous one was usually faster to train and score and maintained a similar performance, which was a result of the inclusion of fast Naive Bayes ensemble members. In a similar fashion, we experimented with an integration of k-NN into the DDCW model, but as expected, k-NN base learners raised the resource requirements of the model and failed to provide a sufficient performance boost, so we did not include k-NN base learners in further experiments.

Performance comparison showed that the DDCW method can produce results that are comparable to

the related ensemble models on both, real and synthetic streams. In many cases, it is able to outperform existing algorithms (e.g., AWE, DWM, OZA, and OB) in both of the explored metrics. Current state-of-the-art methods such as KUE and ARF usually produce slightly better results, but the DDCW method showed a fairly competitive performance, surpassing on several datasets one of those methods in both, accuracy and F1 scores. However, the evaluation metrics represent only one aspect of the adaptive models' performance. During the experiments, we tried to evaluate another aspect of the studied models that may influence the run time of the models during deployment in real-world scenarios. We focused mostly on monitoring the model performance in terms of their demand on resources and resource consumption during the process. During the experiments, we collected data about the overall run-time aspects of the model. The following section compares the models from the perspective of training/scoring times and memory requirements.

	$DDCW_{HT}$	$DDCW_{HTNB}$	DWM_{NB}	AWE_{NB}	DWM_{HT}	AWE_{HT}	OB_{kNN}	Oza_{kNN}	OB_{HT}	Oza_{HT}	OB_{NB}	Oza_{NB}	ARF_{HT}
Train													
ELEC	143.26	111.69	45.11	129.48	13.57	20.79	1604.23	509.71	1147.40	84.04	1039.32	15.83	315.83
KDD99	2101.06	2558.51	669.26	678.63	2522.47	3040.64	10423.15	10829.81	23207.21	2062.85	15924.35	1650.80	3107.94
AIRL	2742.45	1958.96	103.02	303.11	966.34	1462.39	3278.04	1214.47	4808.00	2728.98	3626.52	132.82	9427.38
COVT	143.26	111.69	486.37	4316.73	8679.77	19836.23	6433.28	5748.82	52179.02	15154.78	25203.72	9532.05	8798.81
SHUTTLE	122.82	139.38	133.04	431.61	211.05	644.96	1733.08	914.14	3279.41	223.35	2931.21	45.55	582.35
POWERSUPLY	554.17	452.09	70.71	305.48	225.03	757.13	2984.78	644.79	19965.02	408.03	10153.15	28.47	773.20
CONNECT4	887.58	810.98	306.71	1189.89	486.08	1189.16	7480.17	3040.63	6988.81	678.24	9876.03	184.61	1267.93
BNG_BRIDGES	20336.45	15750.75	2623.52	11661.23	7238.69	23103.62	105021.95	42943.55	23113.54	6226.40	19416.33	987.16	18992.29
BNG_BRIDGES1vsAll	4244.05	3223.40	1388.32	4423.00	3459.20	11226.27	95129.22	42928.10	13133.03	3429.59	6847.12	766.14	12352.68
BNG_HEPATITIS	7234.04	5903.23	1909.77	6160.89	5921.37	17505.05	142511.73	61844.19	17275.70	5807.02	12077.91	1333.14	22191.77
BNG_ZOO	8833.11	7587.96	831.53	3786.11	3085.02	9840.18	23267.58	11843.88	14623.41	2335.37	17983.05	983.95	24609.51
BNG_LYMPH	52199	60795	27.21	95.20	121.78	312.60	1666.61	633.64	7795.36	147.28	13468.46	1049.58	16545.83
AGR$_a$	1836.59	1729.20	1255.87	4079.91	1909.92	6579.19	33991.85	14658.03	7781.19	1774.19	5960.05	627.15	13707.31
AGR$_g$	2867.83	2731.58	1228.79	3910.93	1350.43	4336.84	33765.55	14634.41	7825.25	2044.13	5961.76	627.01	14255.32
SEA$_a$	1390.82	1144.45	537.41	1355.54	903.92	2407.77	25996.58	11367.54	4050.33	945.55	2600.97	243.14	14138.98
SEA$_g$	1938.19	1562.57	535.09	1358.02	898.35	2375.66	25845.26	11367.24	4052.31	924.03	2535.35	237.78	10720.81
STGR	99.96	95.56	119.05	189.38	191.96	339.70	3181.51	1129.84	2262.76	205.48	1778.44	86.18	738.52
LED	1710.85	1164.57	1539	3415	3622.99	5589.79	7682	3097	34339.45	3447.84	28733.92	478.55	6099.67
MIXED_BALANCED	2580.27	1531.19	970.63	2488.19	946.31	7679.21	22126.14	15055.02	9481.48	2000.08	6858.02	437.57	8358.76
MIXED_IMBALANCED	2712.32	1540.93	982.09	2551.43	2733.50	7749.73	22211.42	14447.70	7448.70	1907.75	6907.52	651.22	8540.42
RBF	8090.26	5870.58	1517.56	4729.81	3503.11	10058.28	18172.70	16260.66	13485.50	3528.25	9509.59	986.65	19776.25
RBF_DRIFT	14255.57	7468.35	1467.39	4411.77	2433.44	6011.22	38684.40	16204.23	8952.23	2793.67	9863.56	955.75	20931.48
WAVEFORM	18050.07	10330.24	3292.84	16965.00	10035.01	26579.84	59396.41	25562.21	25414.68	10446.38	17079.50	1394.92	37111.64
WAVEFORM_DRIFT	19060.56	9819.87	3217.57	12837.63	5691.92	17527.29	59255.53	25508.46	25871.33	10373.11	16140.68	1368.95	36135.95
Scoring													
ELEC	44.18	44.59	30.08	60.34	8.76	5.92	383.55	399.24	46.97	54.61	39.86	35.49	30.33
KDD99	826.72	1789.61	582.33	261.48	1108.80	938.46	5220.47	10181.99	1653.69	6070.47	29248.61	454.54	
AIRL	682.19	642.45	69.00	137.29	296.88	659.61	964.92	972.81	882.57	1436.29	765.43	175.72	709.79
COVT	44.18	44.59	348.98	952.85	3345.07	6734.88	2674.34	5429.85	11320	7799.45	8935.04	4950.78	1190.09
SHUTTLE	43.83	74.66	106.36	190.00	101.52	237.67	546.34	756.35	200.45	188.33	254.86	263.51	83.27
POWERSUPLY	206.63		186.37	124.85	143.21	263.09	547.29	468.14	678.93	680.06	398.07	298.00	58.48
CONNECT4	119.21	180.59	225.28	448.38	119.71	264.27	2555.47	2831.78	452.03	204.62	621.94	633.43	125.87
BNG_BRIDGES	3658.96	4447.50	2192.79	4485.01	3452.78	7233.22	40374.27	40626.88	6059.87	3820.15	6090.57	5606.51	1623.26
BNG_BRIDGES1vsAll	1228.14	1337.69	932.74	1872.88	1336.98	3714.29	3985.74	40591.08	2963.44	2264.16	1954.08	1759.19	1253.93
BNG_HEPATITIS	1432.58	1755.10	1289.06	2554.61	2349.87	5490.33	58901.34	59433.80	4161.07	2695.03	3287.22	3064.09	1353.66
BNG_ZOO	1655.25	2796.35	711.68	1419.12	1406.56	3003.11	10138.36	11355.23	1877.31	1299.39	7308.81	6687.30	294.30
BNG_LYMPH	2265.33	3324.49	20.80	40.57	52.41	104.56	486.95	584.83	92.46	101.46	4520.05	4209.24	12.97
AGR$_a$	589.06	681.66	859.27	1695.15	825.03	2156.90	12148.39	12004.23	1676.38	1247.63	1569.22	1465.04	1375.23
AGR$_g$	842.44	1030.88	842.34	165.97	557.85	1466.99	11938.29	12016.64	1684.98	1328.85	1576.46	1516.61	1375.41
SEA$_a$	491.00	463.57	366.80	721.22	475.68	929.90	9368.47	9501.99	899.43	753.90	616.91	565.17	980.09
SEA$_g$	690.24	660.24	363.96	717.64	471.14	927.79	9309.85	9463.46	871.64	735.32	602.78	553.37	802.78
STGR	43.36	46.87	70.61	144.67	86.07	166.89	935.82	891.94	173.56	155.34	140.56	122.58	96.58
LED	544.33	603.01	1305.11	2665.37	1610.95	3274.27	2777.76	2834.50	3019.31	2594.30	2753.80	2781.52	528.32
MIXED_BALANCED	1118.60	737.17	655.21	1280.62	533.98	2711.89	8627.57	12252.98	2219.60	1939.11	1701.65	940.76	1094.69
MIXED_IMBALANCED	1189.05	748.56	667.03	1308.05	1459.17	2723.83	8646.08	11612.31	2214.76	175.07	1746.56	1432.22	1119.65
RBF	1778.62	2104.27	1053.82	2084.06	1626.98	3302.48	6634.78	13305.02	2861.86	2219.82	2500.18	2238.74	1765.03
RBF_DRIFT	2592.40	1755.61	1013.02	1981.98	1028.25	2006.14	13536.52	13383.73	2039.09	1956.97	2457.97	2199.96	2596.57
WAVEFORM	2738.06	2474.19	2501.26	4900.75	4298.73	7585.06	22613.52	22503.25	5233.95	5764.22	4922.47	4636.79	3266.90
WAVEFORM_DRIFT	2934.28	2234.43	2460.86	4947.01	1028.25	2006.14	22610.44	22508.14	2039.09	1956.97	4672.90	4375.81	3250.45
Memory													
ELEC	4509	3166	52	383	169	186	4116	4468	1994	2180	181	101	1502
KDD99	18891	18825	976	1872	1703	1888	33283	41683	1610	20538	2150	10128	1335
AIRL	21508	19926	56	436	64	451	4828	4868	3776	23113	224	115	34904
COVT	4509	3166	489	5419	863	5432	6821	23218	3355	201265	1838	195008	9484
SHUTTLE	4664	2952	136	779	371	797	3986	5640	310	1114	412	330	3376
POWERSUPLY	2653	2664	129	899	133	914	7478	7610	981	642	926	627	21425
CONNECT4	53333	46010	372	2724	376	2742	18579	18527	1465	12246	878	782	6267
BNG_BRIDGES	42056	37951	201	1375	206	1392	6970	6877	2119	27417	561	399	101039
BNG_BRIDGES1vsAll	43681	44147	98	713	430	730	6941	6855	156252	53302	308	192	279918
BNG_HEPATITIS	47294	47956	150	1084	155	1102	10010	9857	102517	108163	401	296	234600
BNG_ZOO	38211	40955	308	2079	313	2097	8550	8800	21231	9192	718	614	24674
BNG_LYMPH	52199	60795	199	1384	204	1401	9188	9134	1092	408	556	401	12007
AGR$_a$	37091	35128	74	535	133	562	5739	5647	632	37433	278	145	40581
AGR$_g$	33633	31176	74	545	75	562	5726	5647	594	56528	288	145	29348
SEA$_a$	33881	33449	29	220	34	239	3435	3341	11497	21888	157	55	206872
SEA$_g$	32350	31976	28	220	33	238	3430	3340	9279	22056	156	55	121890
STGR	3794	3661	29	220	59	138	3391	3318	599	627	163	55	479
LED	4712	2235	553	3680	558	3694	11580	11505	1068	9890	1182	1103	2674
MIXED_BALANCED	34658	34182	36	276	789	292	2767	3794	204	9199	180	69	15355
MIXED_IMBALANCED	35603	34243	37	276	231	293	1738	3791	410	9369	179	69	16163
RBF	45067	38146	82	598	86	616	2835	6323	572	65277	267	160	169534
RBF_DRIFT	44461	43113	82	598	86	615	6222	6130	1283	59310	311	160	45777
WAVEFORM	54302	49396	200	1419	204	1432	10797	10639	49221	68083	557	395	371158
WAVEFORM_DRIFT	56276	43703	200	1419	204	1433	10799	10640	53793	69302	557	395	380476

Table 6. Comparison of average total training and scoring times (in seconds) and average model size in memory (in kB).

527 ***Training time and memory usage***
528 We analyzed the training and scoring times and the memory consumption during the training process
529 to provide a different view of the models' performance, comparing performance metrics with resource
530 consumption requirements. We measured the overall training and scoring times on the entire data by
531 summing up all partial re-training and scoring times over the stream. Table 6 summarizes the results of all
532 evaluated models. The table compares the total training time consumed in the training and re-training
533 of the models, the total scoring time of all processed instances, and the average size of the model in
534 memory during the entire stream processing. The results represent an averaged value of the total of five
535 separate runs of each experiment. It is important to note that the KUE model was not included in this
536 comparison. We used Python implementations of all evaluated models and the scikit-multiflow library
537 during the experiments. The KUE model was available only in its Java implementation using the MOA
538 library, therefore using different underlying technologies could influence the objective comparison of
539 resource consumption. At the same time, it is essential to note that the Java implementation of the KUE
540 model was significantly effective than all Python-based models, mostly in training times, which were
541 remarkably shorter.

542 The choice of a base classifier heavily influences the overall run-time requirements of all models.
543 Most apparently, the choice of k-NNADWIN as a base learner for OnlineBagging and OzaBagging
544 methods leads to a massive increase of memory consumption and data processing times. Nearest
545 neighbour classifiers require to store the training data in memory, which could lead to increased memory
546 consumption and therefore increased training times. On the other hand, ensembles which consist of Naive
547 Bayes and Hoeffding Tree classifiers as the base learners are much faster to train and require significantly
548 lower memory during the run-time. However, Online Boosting and OzaBagging methods are much more
549 sensitive to the number of features of the processed data. It can be observed on KDD99 and Covtype
550 datasets, where these relatively faster models, required significantly longer times to either train or score
551 the instances. DDCW ensemble training time and memory consumption requirements reflect the fact that
552 the model consists of a mixture of Hoeffding Tree and Naive Bayes classifiers. When experimenting with
553 the homogeneous DDCW ensemble, the performance results were better on many of the datasets. On the
554 other hand, heterogeneous DDCW model provided a small decrease of the performance, but in most of
555 the cases, inclusion of a Naive Bayes base learner led to a shorter training times and reduced memory
556 usage (most significantly on e.g., Waveform data, where the training time was reduced to a half of the
557 total training time of the homogeneous DDCW ensemble). When taking into consideration both aspects,
558 DDCW model can, in some cases present a compromise between performance and resource requirements.
559 Although Online Boosting or OzaBagging model performs with higher degrees of accuracy on some of
560 the datasets, their computational intensiveness and more extended training and scoring times may be a
561 factor to consider a simpler model. Similarly, ARF and KUE models provide superior performance on
562 the majority of the datasets. When compared to those state of the art methods, DDCW method produced
563 mostly comparable results, but needed less training time with lesser memory requirements (especially
564 on the larger synthetic data streams) than ARF method. DDCW ensemble, in this case, may offer a
565 reasonable alternative by providing a well-performing model, while maintaining reasonable requirements
566 on run-time.

CONCLUSIONS

568 In the presented paper, we propose a heterogeneous adaptive ensemble classifier with a dynamic weighting
569 scheme based on the diversity of its base classifiers. The algorithm was evaluated on a wide range of
570 datasets, including real and synthetic ones, with different types of concept drift. During the experiments,
571 we compared the proposed method with several other adaptive ensemble methods. The results proved
572 that the proposed model is able to adapt to drift occurrence relatively fast and is able to achieve at least
573 comparable performance to the existing approaches, on both real and synthetically generated datasets.
574 While still performing well, the model also manages to maintain reasonable resource requirements in
575 terms of memory consumption and time needed to score the unknown samples.The proposed approach
576 is also dependent on chunk size parameter setting, as the performance of the model on certain datasets
577 change significantly with different chunk sizes. Further research with adaptive heterogeneous ensemble
578 models may lead to an exploration of modifications to weighting schemes that improve performance
579 in multi-class classication problems or classifications of heavy imbalanced data. Another interesting
580 field for future work is the integration of adaptation mechanisms with semantic models of the application

581 domain. A domain knowledge model could provide a description of the data, the essential domain
582 concepts, and their relationships. Such a model could also be used to improve classification performance
583 by capturing expert domain knowledge and utilizing it in the process of classification of unknown samples.
584 A knowledge model could be used to extract new expert features not previously contained in the data or
585 to extract interesting trends present in the data stream. Such extensions could represent expert knowledge
586 and could thus be leveraged to detect frequent patterns leading to concept drift while reducing the time
587 normally needed to adapt the models with that knowledge.

ACKNOWLEDGMENTS

589 This work was supported by Slovak Research and Development Agency under the contract No. APVV-
590 16-0213.

REFERENCES

592 Abassi, M. S. (2019). Diversity of ensembles for data stream classification.
593 Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining Association Rules Between Sets of Items in
594   Large Databases. *ACM SIGMOD Record.*
595 Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., and Morales-Bueno, R. (2006).
596   Early Drift Detection Method. *4th ECML PKDD International Workshop on Knowledge Discovery*
597   *from Data Streams.*
598 Barddal, J. P., Gomes, H. M., Enembreck, F., and Pfahringer, B. (2017). A survey on feature drift
599   adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*,
600   127:278 – 294.
601 Bifet, A., De Francisci Morales, G., Read, J., Holmes, G., and Pfahringer, B. (2015). Efficient online
602   evaluation of big data stream classifiers. In *Proceedings of the ACM SIGKDD International Conference*
603   *on Knowledge Discovery and Data Mining.*
604 Bifet, A. and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceed-*
605   *ings of the 7th SIAM International Conference on Data Mining*, pages 443–448.
606 Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and
607   Vanschoren, J. (2019). Openml benchmarking suites.
608 Black, M. and Hickey, R. (2003). Learning classification rules for telecom customer call data under
609   concept drift. *Soft Computing.*
610 Blackard, J. A. (1998). *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest*
611   *Cover Types*. PhD thesis, USA. AAI9921979.
612 Breiman, L. (1996). Bagging predictors. *Machine Learning.*
613 Breiman, L. (2001). Random forests. *Machine Learning.*
614 Brzeziński, D. and Stefanowski, J. (2011). Accuracy updated ensemble for data streams with concept
615   drift. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence*
616   *and Lecture Notes in Bioinformatics)*, 6679 LNAI(PART 2):155–163.
617 Brzezinski, D. and Stefanowski, J. (2016). Ensemble diversity in evolving data streams. In *Lecture Notes*
618   *in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*
619   *Bioinformatics).*
620 Cano, A. and Krawczyk, B. (2020). Kappa Updated Ensemble for drifting data stream mining. *Machine*
621   *Learning.*
622 Carney, J. G. and Cunningham, P. (2000). Tuning diversity in bagged ensembles. *International journal of*
623   *neural systems.*
624 Cassidy, A. P. and Deviney, F. A. (2015). Calculating feature importance in data streams with concept
625   drift using Online Random Forest. In *Proceedings - 2014 IEEE International Conference on Big Data,*
626   *IEEE Big Data 2014.*
627 Chiang, R. D., Wang, Y. H., and Chu, H. C. (2013). Prediction of members' return visit rates using a time
628   factor. *Electronic Commerce Research and Applications.*
629 Delany, S. J., Cunningham, P., Tsymbal, A., and Coyle, L. (2005). A case-based technique for tracking
630   concept drift in spam filtering. In *Knowledge-Based Systems.*
631 Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in Nonstationary Environments: A
632   Survey.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., and Herrera, F. (2018). *Learning from Imbalanced Data Sets*.

Fernandez-Aleman, J. L., Carrillo-De-Gea, J. M., Hosni, M., Idri, A., and Garcia-Mateos, G. (2019). Homogeneous and heterogeneous ensemble classification methods in diabetes disease: A review. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*.

Freund, Y. and Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. *Proceedings of the 13th International Conference on Machine Learning*.

Frías-Blanco, I., Verdecia-Cabrera, A., Ortiz-Díaz, A., and Carvalho, A. (2016). Fast adaptive stacking of ensembles. In *Proceedings of the ACM Symposium on Applied Computing*.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman & Hall/CRC, 1st edition.

Gama, J., Aguilar-Ruiz, J., and Klinkenberg, R. (2008). Knowledge discovery from data streams. *Intelligent Data Analysis*, 12(3):251–252.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014a). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4).

Gama, J. a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014b). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4).

Ghaderi Zefrehi, H. and Altınçay, H. (2020). Imbalance learning using heterogeneous ensembles. *Expert Systems with Applications*.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495.

Gonçalves, P. M., De Carvalho Santos, S. G., Barros, R. S., and Vieira, D. C. (2014). A comparative study on concept drift detectors.

Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and Regression Trees. *Biometrics*.

Harries, M., cse tr, U. N., and Wales, N. S. (1999). Splice-2 comparative evaluation: Electricity pricing. Technical report.

Idrees, M. M., Minku, L. L., Stahl, F., and Badii, A. (2020). A heterogeneous online learning ensemble for non-stationary environments. *Knowledge-Based Systems*, 188:104983.

Junior, B. and Nicoletti, M. d. C. (2019). An iterative boosting-based ensemble for streaming data classification. *Information Fusion*.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., and Ghédira, K. (2019). A New Combination of Diversity Techniques in Ensemble Classifiers for Handling Complex Concept Drift. (August 2018):39–61.

Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156.

Kuncheva, L. (2006). Ten measures of diversity in classifier ensembles: limits for two classifiers.

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*.

Large, J., Lines, J., and Bagnall, A. J. (2017). The heterogeneous ensembles of standard classification algorithms (HESCA): the whole is greater than the sum of its parts. *CoRR*, abs/1710.09220.

Li, C. T., Shan, M. K., Jheng, S. H., and Chou, K. C. (2016). Exploiting concept drift to predict popularity of social multimedia in microblogs. *Information Sciences*.

Lifna, C. S. and Vijayalakshmi, M. (2015). Identifying concept-drift in Twitter streams. In *Procedia Computer Science*.

Lin, C. C., Deng, D. J., Kuo, C. H., and Chen, L. (2019). Concept drift detection and adaption in big imbalance industrial IoT data using an ensemble learning method of offline classifiers. *IEEE Access*.

Liu, S., Feng, L., Wu, J., Hou, G., and Han, G. (2017). Concept drift detection for data stream learning based on angle optimized global embedding and principal component analysis in sensor networks.

688   *Computers and Electrical Engineering*.

689   Lo, Y. Y., Liao, W., Chang, C. S., and Lee, Y. C. (2018). Temporal Matrix Factorization for Tracking
690   Concept Drift in Individual User Preferences. *IEEE Transactions on Computational Social Systems*.

691   Luong, A. V., Nguyen, T. T., and Liew, A. W.-C. (2020). Streaming active deep forest for evolving data
692   stream classification.

693   Lysiak, R., Kurzynski, M., and Woloszynski, T. (2014). Optimal selection of ensemble classifiers using
694   measures of competence and diversity of base classifiers. *Neurocomputing*.

695   Marrón, D., Ayguadé, E., Herrero, J. R., and Bifet, A. (2019). Resource-aware elastic swap random forest
696   for evolving data streams.

697   Minku, L. L., White, A. P., and Yao, X. (2010). The impact of diversity on online ensemble learning in
698   the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*.

699   Montiel, J., Mitchell, R., Frank, E., Pfahringer, B., Abdessalem, T., and Bifet, A. (2020). Adaptive
700   XGBoost for Evolving Data Streams.

701   Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming
702   framework. *Journal of Machine Learning Research*, 19(72):1–5.

703   Muhlbaier, M., Topalis, A., and Polikar, R. (2004). Learn++.MT: A New Approach to Incremental
704   Learning. In Roli, F., Kittler, J., and Windeatt, T., editors, *Multiple Classifier Systems*, pages 52–61,
705   Berlin, Heidelberg. Springer Berlin Heidelberg.

706   Mukkavilli, S. K. and Shetty, S. (2012). Mining concept drifting network traffic in cloud computing
707   environments. In *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid
708   Computing, CCGrid 2012*.

709   Nguyen, H. L., Woon, Y. K., Ng, W. K., and Wan, L. (2012). Heterogeneous ensemble for feature drifts
710   in data streams. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial
711   Intelligence and Lecture Notes in Bioinformatics)*.

712   Nick Street, W. and Kim, Y. S. (2001). A streaming ensemble algorithm (SEA) for large-scale classification.
713   In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and
714   Data Mining*.

715   Olorunnimbe, M. K., Viktor, H. L., and Paquet, E. (2018). Dynamic adaptation of online ensembles for
716   drifting data streams. *Journal of Intelligent Information Systems*.

717   Oza, N. C. (2005). Online bagging and boosting. In *Conference Proceedings - IEEE International
718   Conference on Systems, Man and Cybernetics*.

719   Oza, N. C. and Russell, S. (2001). Experimental comparisons of online and batch versions of bagging
720   and boosting. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge
721   Discovery and Data Mining*.

722   Pesaranghader, A., Viktor, H., and Paquet, E. (2018). Reservoir of diverse adaptive learners and stacking
723   fast hoeffding drift detection methods for evolving data streams. *Machine Learning*.

724   Polikar, R., Upda, L., Upda, S. S., and Honavar, V. (2001). Learn++: an incremental learning algorithm for
725   supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications
726   and Reviews)*, 31(4):497–508.

727   Pratt, K. B. and Tschapek, G. (2003). Visualizing concept drift. In *Proceedings of the ACM SIGKDD
728   International Conference on Knowledge Discovery and Data Mining*.

729   Ren, S., Liao, B., Zhu, W., Li, Z., Liu, W., and Li, K. (2018). The Gradual Resampling Ensemble for
730   mining imbalanced data streams with concept drift. *Neurocomputing*.

731   Ruano-Ordás, D., Fdez-Riverola, F., and Méndez, J. R. (2018). Concept drift in e-mail datasets: An
732   empirical study with practical implications. *Information Sciences*.

733   Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey.

734   Schlimmer, J. C. and Granger, R. H. (1986). Incremental Learning from Noisy Data. *Machine Learning*.

735   Sidhu, P. and Bhatia, M. P. S. (2018). A novel online ensemble approach to handle concept drifting
736   data streams: diversified dynamic weighted majority. *International Journal of Machine Learning and
737   Cybernetics*, 9(1):37–61.

738   Stiglic, G. and Kokol, P. (2011). Interpretability of sudden concept drift in medical informatics domain.
739   In *Proceedings - IEEE International Conference on Data Mining, ICDM*.

740   Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99
741   data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications,
742   CISDA 2009*.

743   Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science*
744   *Department, Trinity College Dublin*.
745   Tsymbal, A., Pechenizkiy, M., Cunningham, P., and Puuronen, S. (2006). Handling local concept drift
746   with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In
747   *Proceedings - IEEE Symposium on Computer-Based Medical Systems*.
748   van Rijn, J. N., Holmes, G., Pfahringer, B., and Vanschoren, J. (2014). Algorithm Selection on Data
749   Streams. In Džeroski, S., Panov, P., Kocev, D., and Todorovski, L., editors, *Discovery Science*, pages
750   325–336, Cham. Springer International Publishing.
751   Van Rijn, J. N., Holmes, G., Pfahringer, B., and Vanschoren, J. (2016). Having a blast: Meta-learning and
752   heterogeneous ensembles for data streams. In *Proceedings - IEEE International Conference on Data*
753   *Mining, ICDM*.
754   van Rijn, J. N., Holmes, G., Pfahringer, B., and Vanschoren, J. (2018). The online performance estimation
755   framework: heterogeneous ensemble learning for data streams. *Machine Learning*.
756   Wang, B. and Pineau, J. (2013). Online Ensemble Learning for Imbalanced Data Streams. pages 1–15.
757   Wang, S., Minku, L. L., and Yao, X. (2018). A Systematic Study of Online Class Imbalance Learning
758   with Concept Drift. *IEEE Transactions on Neural Networks and Learning Systems*.
759   Yang, L. (2011). Classifiers selection for ensemble learning based on accuracy and diversity. *Procedia*
760   *Engineering*, 15:4266–4270.
761   Zenisek, J., Holzinger, F., and Affenzeller, M. (2019). Machine learning based concept drift detection for
762   predictive maintenance. *Computers and Industrial Engineering*.
763   Žliobaitė, I. (2010). Learning under Concept Drift: an Overview. pages 1–36.
764   Žliobaitė, I., Pechenizkiy, M., and Gama, J. (2016). An Overview of Concept Drift Applications.