Iterative Subset Selection for Feature Drifting Data Streams

Lanqin Yuan
University of Waikato
Hamilton, New Zealand
fyempathy@gmail.com

Bernhard Pfahringer
Deparment of Computer Science
University of Auckland
Auckland, New Zealand
b.pfahringer@auckland.ac.nz

Jean Paul Barddal
Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
jean.barddal@ppgia.pucpr.br

ABSTRACT

Feature selection has been studied and shown to improve classifier performance in standard batch data mining but is mostly unexplored in data stream mining. Feature selection becomes even more important when the relevant subset of features changes over time, as the underlying concept of a data stream drifts. This specific kind of drift is known as *feature drift* and requires specific techniques not only to determine which features are the most important but also to take advantage of them. This paper presents a novel method of feature subset selection specialized for dealing with the occurrence of feature drifts called Iterative Subset Selection (ISS), which splits the feature selection process into two stages by first ranking the features, and then iteratively selecting features from the ranking. Applying our feature selection method together with Naive Bayes or k-Nearest Neighbour as a classifier, results in compelling accuracy improvements, compared to prior work.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Online learning settings**; **Feature selection**;

KEYWORDS

Data Stream Mining; Feature Selection; Concept Drift; Embedded Feature Selection; Iterative Subset Selection;

ACM Reference Format:
Lanqin Yuan, Bernhard Pfahringer, and Jean Paul Barddal. 2018. Iterative Subset Selection for Feature Drifting Data Streams. In *Proceedings of ACM SAC Conference (SAC'18).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/https://doi.org/10.1145/3167132.3167188

1 INTRODUCTION

Nowadays, many information systems are fed with continuously generated sequential data, so-called data streams, which are potentially infinite. Examples of streaming data include posts on social media, wearable gadgets, and stock market trades. Motivated by these and several other applications, the data mining community has shifted its attention to streaming scenarios, where new techniques are proposed every year. Many of the current developments tackle the transient characteristics of data streams, i.e., often the underlying function that maps instances to classes change over time, thus giving rise to a phenomenon called *concept drift*.

In this paper, we focus on a specific kind of concept drift: feature drifts. A feature drift occurs whenever a feature becomes, or ceases to be, relevant to the class determination. Recent studies on this topic [5, 6] have shown that the number of techniques that can dynamically determine which features are the most important over time, and that is also able to take advantage of this information, is relatively small.

We introduce a novel method for feature selection called Iterative Subset Selection (ISS), which seeks to deal with feature drifting scenarios. Our proposal is an embedded feature selection method, where the feature selection process is part of the classifier's model construction process. We evaluate ISS' effectiveness at addressing the effects of the occurrence of drifts with two well known and widely used classifiers: k-Nearest Neighbours and Naïve Bayes. Additionally, we also evaluate our proposed algorithm against existing works on the topic, highlighting its effectiveness and efficiency.

This paper is structured as follows: Section 2 introduces the task of data stream classification, while Section 3 details the type of drift we wish to tackle: feature drifts. Section 4 surveys related works on feature drift adaptation, which is later used during the empirical analysis. Section 5 introduces our proposed method, which is assessed and compared with prior approaches in Section 6. Finally, Section 7 provides conclusions and directions for future work.

2 DATA STREAM CLASSIFICATION

Learning from ephemeral sequences of data comprises all the problems of conventional batch learning, e.g., missing values, noise, class imbalance, or sparsity. On top of that, it introduces further difficulties, such as single-pass processing, limited computational resources, and concept drifts [15].

We focus on the classification task, which regards learning and updating predictive models over time. Let S denote a data stream providing instances in the $i^t = (\vec{x}^t, y^t)$ form, where t is its arrival time stamp, $\vec{x}^t$ is a vector of features from the entire feature set X and $y^t \in Y$ its class. To denote the i^{th} feature of an instance $\vec{x}^t$, we will use the X_i notation, while $\vec{x}_i^t$ denotes the value of this feature for an instance $\vec{x}^t$, where t will be dropped if unnecessary. Our ultimate goal in classification is to learn and update a model $h : X \rightarrow Y$ over time, upon the arrival of instances from S.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SAC'18, April 9-13, 2018, Pau, France
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/https://doi.org/10.1145/3167132.3167188

In streaming scenarios, classifiers process instances sequentially as they arrive and discard them right after. Although there is no restriction against buffering and processing instances periodically, this must not jeopardize computational resources. This is important since main memory is finite and its usage must be optimised so that classifiers and their buffers fit into this limited space. Processing time per instance must also be limited. If not, arriving instances will be enqueued until the system crashes due to the lack of memory. Lastly, since data streams are inherently temporal, they are also expected to be ephemeral: the underlying function mapping instances to classes is likely to change, thus giving rise to a phenomenon named **concept drift** [21].

3 FEATURE DRIFT

As described in the seminal work of Widmer [21], data streams are susceptible to different types of drifts, including: (i) changes in the characteristic feature values, (ii) evolution of the value domains of features over time, (iii) *features that were once important and that now may become meaningless or the other way around*, and so on. In this paper, we tackle the above-emphasized type of drift, called **feature drift**. A feature drift occurs when a subset of features becomes, or ceases to be, relevant to the learning task [5].

To formalize feature drifts, we must first be able to discern between relevant and irrelevant features [26]. Given the entire feature set X, such that $S_i = X \setminus \{X_i\}$, then a feature X_i is **relevant** *iff* Definition 1 holds, and otherwise, it is said to be **irrelevant**.

Definition 1. A feature X_i is relevant *iff* $\exists S'_i \subset S_i$, such that $P[Y|X_i, S'_i] > P[Y|S'_i]$ holds.

If an arbitrary relevant feature is removed, then it will result in a reduction of overall prediction power, because (i) it alone is strongly correlated with the class, or (ii) it forms a feature subset with other features that together are correlated with the class (this concept is commonly referred as feature interaction [16]).

A feature drift occurs whenever a subset of features becomes, or ceases to be, relevant to class prediction. Given X at a timestamp t, we are able to select the ground-truth relevant features $X^* \subseteq X$ such that $\forall X_i \in X^*$ Definition 1 holds and $\forall X_j \in X \setminus X^*$ the same condition does not. A feature drift then occurs if, at a timespan between t_i and $t_j = t_i + \Delta$, X^* at t_i differs from X^* at t_j.

Like other types of drift, changes in the relevant subset of features X^* affect the ground-truth decision boundary. Ideally, we expect classification models to detect and adapt to changes in X^*, while changes may be (i) steep, where the classifier would forget its entire model and learn from scratch; or (ii) progressive, where its model would be gradually updated given drifts [19].

4 RELATED WORK

Determining which features are relevant has been widely discussed in batch learning. However, the same cannot be said about streaming environments. Recently, the surveys of [4, 5] highlighted that there are few works that explicitly tackle the problem of drifting features, and these basically include decision trees [11], rule learning [3] and ensembles [2, 19]. Furthermore, [5] provides a comprehensive evaluation of some stream classifiers, concluding that a single adaptive decision tree, namely the Hoeffding Adaptive Tree (HAT) [8], was the best performing classifier over all w.r.t. the trade-off between classification rate, processing time and memory usage.

The HAT is an extension to the incremental Very Fast Decision Tree classifier [11], which uses the ADWIN drift detector [7] inside decision nodes to monitor the internal error rates of the tree. If a feature that is being used by a decision node becomes irrelevant, one would expect the error rate to increase, and ADWIN will flag a change. Consequently, the affected decision node will reset the entire sub-tree, replace the irrelevant split with a better one, potentially identifying and selecting a newly relevant attribute, and restart growing a new sub-tree. This allows the HAT classifier to detect and overcome feature drifts quickly and effectively.

More recently, the authors of [6] proposed a dynamic weighting scheme for the problem of classification over data streams. In that work, the estimation of the discriminative power of each feature is updated using a sliding window approach. Feature evaluation was performed using the Symmetrical Uncertainty scoring operator. The discriminative power of each feature was used as a weighting factor in the prediction process of both k-Nearest Neighbours and Naïve Bayes classifiers, which resulted in useful accuracy gains, at the expense of a reasonable amount of additional processing time and memory usage.

A similar feature selection algorithm called INTERACT has been proposed in [25]. The main differences between ISS compared to INTERACT are that ISS wraps around and embeds the feature selection process into the classifier to select feature subsets and that ISS is incremental.

Finally, it is worth mentioning that feature drifts have also been recently investigated in the context of regression, such as in the work of [12]. However, regression is outside the scope of the work presented here.

5 METHOD

A naïve brute force approach to the problem of identifying irrelevant features in a stream would, for every new example arriving in the stream, exhaustively search through every possible subset of features, evaluate each of these subsets using some goodness of fit criterion, and select the features of the best performing subset. Such an algorithm implicitly deals with changes in feature relevance as the classification for each example is independent. On the other hand, it is extremely inefficient and prohibitively expensive, regardless of working in a batch or streaming settings, and therefore unfeasible in practice.

Our proposed method improves upon this naïve approach: features are first ranked using a scoring function, based on the existing batch feature selection method called ESFS [13, 23]. Then only a limited linear number of subsets is evaluated. Sections 5.1 and 5.2 detail the ranking and subset selection steps, while Sections 5.3 and 5.4 show how these steps are embedded within kNN and Naive Bayes, respectively.

5.1 Ranking

Initially, the relevance of each feature is determined by evaluating it individually and independently w.r.t. class prediction, using some scoring function. The method sorts all features by their discriminative power, also allowing for filtering out irrelevant features. Only

the top f number of features (where f is a parameter) are kept for use in the next stage of subset selection, features below the top f ranks are ignored. In general, it is recommended to set f to cover most, if not all features in the stream, so the method is able to adapt to drastic drifts. Smaller f values reduce the search space and can be used if there exists knowledge of the nature of features in the data stream before evaluation. Rankings are recomputed periodically, using a sliding window of instances, therefore each ranking is independent of any previous ranking. Thus the method can implicitly deal with drift as only new instances in the window are utilized. Re-ranking is performed only every r instances to reduce computational load.

The three ranking functions used here are the Average Euclidean Distance, Information Gain, and Symmetric Uncertainty [22].

Average Euclidean Distance (AED) is a simple ranking function based on the idea that after normalization, features which have the most influence on the decision of the class value will be on average further apart when plotted in Euclidean space. The calculation of AED is different depending on whether an attribute is nominal or numeric, while not requiring discretization as the other ranking functions. The numeric formula of AED is given in Equation 1, where X_i is a numeric attribute in a stream with L classes, $N_c(X_i)$ is the number of times that the class value c appeared in the window with a valid value for X, and X_c^n is the value of X for the n-th instance with a class value of c.

$$\mathrm{AED}_{\mathrm{num}}(X_i) = \sqrt{\sum_{0 \le c < j < L} \left(\mathrm{MV}(X_c) - \mathrm{MV}(X_j)\right)^2} \tag{1}$$

where MV is the mean value, calculated as follows:

$$\mathrm{MV}(X_c) = \frac{1}{N_c(X)} \sum_{n=0}^{N_c(X)} X_c^n \tag{2}$$

The AED formula for a nominal attribute is defined as follows:

$$\mathrm{AED}_{\mathrm{nom}}(X_i) = \sum_{0 \le c < j < L} \left[\frac{1}{V} \sum_{v=0}^{V} \sqrt{\left(\frac{X_{cv}}{C_c(X_i)} - \frac{X_{jv}}{C_j(X_i)} \right)^2} \right] \tag{3}$$

where X_i is a nominal attribute with V categories in a stream with L number of known classes, X_{cv} is the number of instances with a c-th class label and variable value v, and $C_c(X_i)$ is the total number of instances with a label c with a valid (non-missing) value for X_i.

Information Gain (IG), a synonym for the Kullback-Leibler divergence [17], is a widely used measure in machine learning and data mining. It is a measure of the reduction of entropy by the introduction of a variable. As it is only possible to calculate Information Gain for nominal attributes, discretization is required for numeric attributes. The discretization algorithm used in this work was PiD [14], a single pass discretization algorithm specialized for data streams. The formula used to calculate IG for an attribute X_i and a class attribute Y is defined as follows:

$$\mathrm{IG}(Y|X_i) = H(Y) - H(Y|X_i) \tag{4}$$

where $H(X_i)$ is the entropy for an attribute X_i with N distinct values is given by the following formula:

$$H(X_i) = \sum_{j=0}^{N} -P(X_i = x_j) \log P(X_i = x_j) \tag{5}$$

and $H(Y|X_i)$ is the conditional entropy, given by:

$$H(Y|X_i) = \sum_{j=0}^{N} p(x_j) H(Y|X_i = x_j) \tag{6}$$

Symmetric Uncertainty (SU) can be seen as a normalized version of Information Gain, and has been shown to give good performance for feature selection [24]. Its main advantage over Information Gain is that it overcomes the bias in the data (when there is significantly more or less of some class/classes than others). Again, Symmetric Uncertainty can only be calculated for nominal attributes, and thus, discretization is required for numeric attributes. The formula for Symmetric Uncertainty for an attribute X with the class attribute Y is is defined as follows:

$$\mathrm{SU}(X, Y) = \frac{IG(X|Y)}{H(X) + H(Y)} \tag{7}$$

5.2 Subset Selection

As streaming environments require algorithms to be fast, to keep up with the stream, therefore an exhaustive search of all combinations is out of the question. We utilize the ranking to reduce the search space and Backward Feature Elimination [22] to select the best feature subset. As a consequence, we limit the search to a maximum of f iterations, meaning that the growth of the number of evaluations is linear in f rather than exponential, thus also avoiding a potential combinatorial explosion. Backward Elimination is selected over Forward Selection due to its potential for a slight optimisation for the k-Nearest Neighbour classifier while giving the same results. This is discussed in more detail in Section 5.3.

Initially, a subset S that contains all f ranked features is evaluated and its prediction result recorded. The lowest ranked features are iteratively removed from S, and the new smaller S is re-evaluated until S is empty. The best subset is selected based on an estimate of the accuracy achieved by each of the subsets by maintaining counts of the number of times a subset correctly predicts the class when the classifier uses only features in the given subset. These counts are fuzzy counts as they are subject to a decay factor d to allow for implicit adaptation to change. Counts are tied to the size of the subset rather than the composition of the subset, similar to how counts are kept in the Space Saving Algorithm [18], thus reducing memory usage and complexity. The accuracy estimate for a subset S is computed using the simple formula of $\frac{\text{\# of correct predictions with S}}{\text{total \# of predictions with S}}$ and the final prediction of the classifier selected based on the subset with the current best estimate. While only the final prediction is output, counts are updated for all subsets. The decay factor of the counts is a user-given parameter d and is applied at fix intervals i.

5.3 k-Nearest Neighbours Classifier

k-Nearest Neighbours (kNN) was one of the classifiers we explored. The implementation of the ISS subset selection for kNN is shown in Algorithm 1. kNN classifies the target instance based on a majority vote by its k nearest neighbors' class values. Euclidean distance is

often selected as the distance measure due to its simplicity. The cumulative nature of Squared Euclidean distance also allows for easy maintenance of distances for each instance in the window. The kNN classifier itself can utilize a slight optimisation from backward elimination using the knowledge that the Euclidean distance between the target instance and instances in the sliding window are normalized. Due to normalisation, the absolute difference in squared Euclidean distance for each new feature added or taken away is bounded between 0 and 1, meaning we can separate instances into 3 subsets: (i) set A where instances' distances are updated and the instances used in kNN evaluation, (ii) set B where instances' distances are kept up to date but the instances are ignored in kNN evaluation; and (iii) set C where instances are ignored entirely and their distances not kept up to date. Instances are able to move between A and B, and from B to C, with instances unable to leave C once they enter. Initially all instances begin in A. For any instance i with a current distance D_i we define $n_i = \lceil D_i - D_k - 1 \rceil$, where D_k is the distance of the current k-th nearest neighbour. If $n_i \geq 0$, then we can safely ignore i for the next n_i iterations. In other words i moves from A to B and is unable to return to A the next n_i iterations of subset selection. Instances in B still need their distances kept up to date as after n_i iterations, i might rejoin A and n_i is recomputed again, with the algorithm repeating until all f subsets have been explored.

After $f/2$ total evaluations, we can potentially eliminate instances from the search as it becomes impossible for i to return to A from B if n_i is greater than the number of iterations left l. In this case, i would move B to C if $n_i \geq l$, eliminating i from any further distance updating as it will no longer be evaluated. Forward selection would not allow for this optimisation to the same degree as all distances would initially start between 0 and 1 for the initial subset of only the top-ranked feature, meaning it is not possible to determine which instances can be placed into B initially.

Despite this optimisation, it was found that searching through every subset size for data streams is still generally computationally prohibitive, as streams are prone to dramatic drifts and may have a large number of features, which subsequently requires a large f parameter to be able to adapt to change. While the ranking is iterative and computationally inexpensive, A and B are still potentially very large even after several iterations of subset selection. In the worst case, subset selection runs the kNN search algorithm f times for every new instance, which results in a complexity of $O(kfw)$ where w is the size of the window containing the instances, and k is the number of nearest neighbors.

To alleviate this problem, the simple algorithm of Hill Climbing is utilized in a novel way to select the subset size and limit the number of subsets evaluated. Hill Climbing is a naïve greedy local search algorithm which incrementally and greedily selects the best solution from the search frontier. While Hill Climbing is known to be prone to becoming stuck in local optima, the occurrence of local optima is found to be mitigated by first initially sorting the features in the ranking stage, so long as the ranking function can sort relevant features above irrelevant features. The search frontier is limited by only evaluating subsets with a difference in a number of features of at most $\pm h$. This hill climb window h controls the number of subset sizes searched around the best subset size selected from the previous iteration. Complexity is therefore bounded by $O(kw(2h+1))$, which allows experiments to complete in reasonable time. It was found that Hill Climbing caused a very negligible decrease (less than 1%) in overall accuracy compared to a static f comprising all features in the stream due to slightly worse initial accuracy, as the algorithm needs time to "climb".

Algorithm 1 k-Nearest Neighbours Classifier Subset Selection

Input:
a: Array of accuracy estimates for each subset size
h: Hill climbing window size
b: Size of previous best subset
f: Upper bound of subset size
r: List of ranked features sorted descending from most relevant to least relevant
Output:
q: Final classification result after feature selection
b: Size of new best subset

```
1:  function SelectSubsetKNN(a,h,b,f,r)
2:      l ← b − h; l ≥ 1                          ▷ Lower bound of subset size
3:      u ← b + h; u ≤ f                          ▷ Upper bound of subset size
4:      for i from 1 to u do
5:          S ← r[i]                              ▷ fill S with ranked features
6:      end for
7:      for i from u to l do
8:          p ← result from kNN classifier with only features in S
9:          Update a[size(S)]
10:         S ← S − r[i]                          ▷ Remove bottom ranked feature from S
11:         if i = b then
12:             q ← p
13:         end if
14:     end for
15:     b ← index of max(a)                       ▷ Set new best subset size
16:     return q, b        ▷ b is used for the next iteration of subset selection
17: end function
```

5.4 Naïve Bayes Classifier

The Naïve Bayes classifier is a probabilistic classifier that determines its classification result using Bayes' theorem while making a strong (naïve) independence assumption between each of the features in a stream, given the class value. ISS works well with the Naïve Bayes classifier as bayesian probability is cumulative when features are assumed to be independent. This means that the next smaller feature subset is easily and cheaply computed by removing the probabilities of the least desirable feature from the current total probabilities. Thus, the Naïve Bayes classifier does not face the same performance limitations that the kNN classifier does. Therefore, heuristic search, such as hill climbing, is not required to ensure reasonable run-times, even for large f. The implementation of ISS subset selection for the Naïve Bayes classifier is shown in Algorithm 2. A point to note is the necessity for the algorithm to maintain a separate sliding window for ranking as unlike the kNN classifier, the NB classifier does not use sliding windows by default.

Algorithm 2 Naïve Bayes Classifier Subset Selection

Input:
a: Array of accuracy estimates for each subset size
w: Ranking window
b: Size of previous best subset
f: Upper bound of subset size
r: List of ranked features sorted descending from most relevant to least relevant
Output:
q: Final classification result after feature selection
b: Size of new best subset

```
function SelectSubsetNB(a,b,w,f,r)
    for i from 1 to f do
        S ← S + r[i]                    ▷ Fill S with ranked features
    end for
    for i from f to 1 do
        p ← result from NB classifier with only features in S
        Update a[size(S)]
        S ← S − r[i]                    ▷ Remove bottom ranked feature from S
        if i = b then
            q ← p
        end if
    end for
    add instance to w
    while w's size ≥ w's max capacity do
        Remove oldest instance in w
    end while
    b ← index of max(a)                 ▷ Set new best subset size
    return q, b                         ▷ b is used for the next iteration of subset selection
end function
```

6 EVALUATION

In this section, we compare the average accuracy obtained by classifiers, their processing time, and the memory (RAM-Hours) required to process each of the tested data streams.

Experiments were conducted using the Massive Online Analysis framework [9]. A large combination of ISS parameter configurations was explored for both the kNN and NB classifiers. The parameters explored were: the interval at which decay occurs for the accuracy counts i (500, 1000, and 1500 were tested), the decay factor of the accuracy counts d (0.05, 0.1, 0.15, 0.2, and 0.25 were tested), the window size used by ranking functions w (500, 1000, and 1500 were tested), and the interval at which ranking occurs r (500, 1000, and 1500 were tested). The number of ranked features f was set to encompass all features in the stream. The kNN with ISS specific hill climb window size h parameter, which controls the number of subsets searched around the current best size, was also explored with sizes of 2, 4 and 6. The source files of the experiments conducted can be found online at https://github.com/EMPaThy789/ISS-MOA.

As accessibility to real-world data streams was limited and surrounded by ethical and privacy issues, artificially generated data streams were used in their steed. All of the generators feature some number of irrelevant features. Stream generator which featured gradual feature drift tested were: AGRAWAL [1], Assets [6], BG [4], LED [9] and SEA [5, 20]. Sudden feature drift was tested using a custom made generator for this work called the Conditional Generator, also available in the project repository.

The Conditional generator generates classes with certain attributes associated with each class. Each attribute associated with a class has some valid ranges or values, depending on whether the attribute is numeric or nominal. Each instance is generated by first generating a class, then randomly re-setting the associated attributes to valid ranges for the class. Drift are introduced by re-associating attributes and their valid ranges. The streams generated using this generator were: CONDITIONAL, FY A, FY B, FY C, and FY D, with each stream increasing in complexity. FY A and FY B both feature no drift and were repeated as separate streams to encompass feature drifts, and are referred to as FY A drift and FY B drift.

Each experiment consisted of running the classifier on 500,000 instances of generated stream data. Points for feature drift were set at every 200,000 instances for all feature drifting streams, except FY A drift, FY B drift, and FY C, where only a single drift occurs after 250,000 instances.

Finally, statistical significance across methods is verified using Wilcoxon's test, or a combination of Friedman and Nemenyi's tests depending on the number of methods compared [10]. All tests assume a confidence level of 95%.

6.1 k Nearest Neighbour Classifier

Due to time and memory constraints, only experiments with the SU ranking function were run for the kNN classifier in this work. The SU ranking function was selected as it was found to give better performance than the other two ranking functions overall in almost all cases for the kNN classifier.

While a large number of classifier parameters were tested, we focus on a default parameter configuration featuring a window size w of 1000, a ranking interval r of 500, a decay rate d of 0.1, a decay interval i of 1000, and a hill climbing window h of 4. This particular configuration was selected based on the trade-off between accuracy and computational resources. We compare this default configuration's results to the baseline kNN classifier's results, which used a k of 10 and a window size of 1000. Overall results are analyzed more broadly. Full results of experiments conducted for this work can be found on GitHub.

A comparison between this default configuration and the kNN standalone classifier is shown in Table 1. The kNN classifier benefits from ISS feature selection in classification accuracy in all experiments for the default configuration, and these improvements are statistically significant according to the Wilcoxon's test. Runtime was generally slower, except for the streams generated by the Conditional generator. There no significant differences were found. Memory usage was also lower for ISS for the BG, LED, and SEA data streams: an improvement in RAM-Hours was observed despite slower runtime results. The difference again is significant according to a Wilcoxon's paired test.

Overall, all configurations of kNN experiments conducted in this work produced a higher average accuracy than the kNN baseline classifier. The impact of parameters on overall average classification accuracy is varying depending on the streams. Most experiments

Stream	Avg Accuracy (%)	CPU Time (s)	RAM-Hours (kb-hour)
AGRAWAL	88.97 (+25.34)	353.31 (+153.63)	63.58 (+28.04)
ASSETS	85.53 (+1.41)	304.95 (+153.00)	41.13 (+20.79)
BG	89.79 (+5.45)	456.48 (+236.69)	86.43 (-53.03)
BG 2	89.76 (+9.72)	517.89 (+53.49)	152.55 (+16.24)
BG 3	89.59 (+12.62)	565.14 (+91.24)	166.47 (+125.05)
CONDITIONAL	91.86 (+14.61)	735.15 (-559.53)	341.12 (-252.37)
FY A	72.87 (+10.69)	770.34 (-788.02)	443.87 (-440.08)
FY B	85.33 (+11.11)	1621.89 (-4640.54)	2713.85 (-7602.82)
FY C	44.99 (+26.96)	745.03 (-4758.63)	1149.98 (-7077.83)
FY D	67.90 (+28.35)	1384.43 (-4978.44)	2356.87 (-8125.98)
FY A Drift	81.27 (+7.40)	1000.18 (-408.01)	580.64 (-224.66)
FY B Drift	80.53 (+17.21)	1576.40 (-3964.14)	2644.58 (-6506.91)
LED	73.62 (+9.29)	824.93 (+69.26)	326.19 (-29.44)
SEA	88.29 (+8.75)	426.26 (+151.83)	98.32 (-35.83)

Table 1: ISS experiment results for default configuration for the kNN classifier. Difference kNN with ISS and between kNN without ISS shown in brackets, with green indicating better results for kNN with ISS and red indicating worse.

Stream	Avg Accuracy (%)	CPU Time (s)	RAM-Hours (kb-hour)
AGRAWAL	81.78 (+8.46)	3.00 (+1.60)	0.54 (+0.54)
ASSETS	84.85 (+3.86)	17.66 (+1.04)	2.39 (+2.38)
BG	88.55 (+8.76)	2.32 (+1.48)	0.44 (+0.44)
BG 2	75.89 (+8.94)	3.64 (+2.07)	1.08 (+1.08)
BG 3	68.06 (+11.68)	3.55 (+2.37)	1.05 (+1.05)
CONDITIONAL	89.10 (+11.13)	8.43 (+5.66)	3.93 (+3.92)
FY A	87.13 (0.00)	12.08 (+6.82)	7.03 (+6.99)
FY B	91.84 (-0.04)	41.75 (+25.49)	70.57 (+70.22)
FY C	53.01 (+12.30)	58.21 (+22.91)	92.59 (+90.76)
FY D	77.92 (+15.88)	66.83 (+26.86)	117.44 (+115.09)
FY A Drift	84.47 (+4.97)	12.07 (+6.60)	7.01 (+6.98)
FY B Drift	84.62 (+15.61)	41.60 (+25.98)	70.25 (+69.92)
LED	72.79 (+16.35)	5.50 (+3.18)	2.19 (+2.18)
SEA	83.90 (+0.14)	4.18 (+2.51)	0.97 (+0.97)

Table 2: ISS experiment results for default configuration for the NB classifier. Difference NB with ISS and between NB without ISS shown in brackets, with green indicating better results for NB with ISS and red indicating worse.

had only slight differences (< 2%) in overall classifier accuracies. In 4 streams (AGRAWAL, BG2, SEA, and LED), parameters had little impact on the overall mean classifier accuracy, with only around a 1% difference in overall mean accuracy between the best performing ISS experiment and the worst performing ISS experiment tested in this work.

While it was found that accuracy tended to be largely similar across all configurations, runtime and RAM-hours performance was found to be much more dependent on the parameters set for ISS. It was found that the size of the w had the most impact on computation time and RAM-Hours, as a larger w meant more instances to search through for each kNN evaluation. The size of the hill climb window h also had a noticeable impact on computation time, and RAM-Hours as a larger window hill climbing window means that more subsets were searched at every iteration; however, this was found to be secondary to the size of w.

6.2 Naïve Bayes Classifier

As with the kNN classifier, we also focus on a default configuration of ISS for comparison. The default configuration used a ranking window w of 1000, a ranking interval r of 500, a decay rate d of 0.1, a decay interval i of 1000, with the ranking function being SU to be in line with the kNN experiment. Overall results are again analyzed more broadly in this section.

We more closely examined results for all three ranking functions (AED, IG, and SU) for the NB classifier. All three ranking functions were very similar in their performance. Difference between accuracy was less than 1%, run time less than 1 second, and RAM hours less than 0.8 kb-hour for all ranking functions on all streams. A point of interest is that the IG and AED ranking functions were more competitive with the SU ranking function, often giving slightly better accuracies for the NB classifier, which is in contrast to the kNN classifier where the SU ranking function was always slightly better than the other two.

The accuracy results for the default configuration are presented in Table 2. Concerning accuracy, the default configuration of NB with ISS outperformed NB standalone by a noticeable amount in most of the streams tested. The exceptions to this were the FY A, and FY B scenarios, where NB performed within 0.1% accuracy to NB with ISS. Despite these minor accuracy losses, Wilcoxon's test showed significant improvements when ISS is used with NB classification.

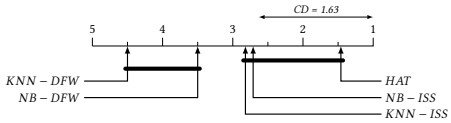

Figure 1: Critical difference chart for accuracy rates.

Regarding computational resources, we notice that NB with ISS was always slower than NB by itself. This can be attributed to the high speed of the NB classifier which means that the reduction in computation time gained from feature selection is still not as large as the overhead cost incurred from ISS. Despite these increases being statistically significant, the runtime for NB with ISS is still fast, especially compared to the kNN classifier.

6.3 ISS against DFW and HAT

We compared ISS against kNN/NB with DFW [6], an algorithm which attempts to address feature drifts via feature weighing, and HAT, a classifier which has been shown to give good performance in feature drifting streams [5]. Comparison results between ISS, DFW, and HAT are shown in Tables 3, 4, and 5.

Overall, it was observed that HAT produced the highest mean accuracy for most experiments, and was generally competitive with NB-ISS/kNN-ISS for the stream scenarios where it was not the highest. DFW was observed to be competitive with ISS for the kNN classifier for mean accuracy; however, a much more pronounced difference in mean accuracy was observed for NB where ISS achieved better results. The results obtained by the statistical tests are shown in Figure 1: they corroborate that HAT is still the best performing classifier, yet, the results obtained are not statistically superior when compared to both NB-ISS and KNN-ISS.

Computation time (Table 4) and RAM-hours (Table 5) were where ISS excelled, with NB-ISS being the fastest classifier for all of the streams tested by a large margin. Both the ISS versions of kNN and

Stream	NB-ISS	NB-DFW	kNN-ISS	kNN-DFW	HAT
AGRAWAL	81.78	68.97	88.97	88.99	**89.46**
ASSETS	84.85	74.00	85.53	91.56	**94.13**
BG	88.55	83.94	89.79	89.78	**89.86**
BG 2	75.89	75.67	89.76	89.50	**89.77**
BG 3	68.06	56.11	**89.59**	89.44	**89.59**
CONDITIONAL	89.10	78.12	91.86	92.56	**96.17**
FY A	**87.13**	62.98	72.87	66.99	86.56
FY B	**91.84**	72.88	85.33	80.69	91.81
FY C	**53.01**	46.70	44.98	27.98	46.01
FY D	**77.92**	68.06	67.90	50.43	75.82
FY A Drift	84.47	68.58	81.27	77.92	**89.14**
FY B Drift	84.62	73.74	80.53	73.77	**89.18**
LED	**83.90**	72.28	73.62	71.76	73.69
SEA	72.79	73.61	88.29	84.80	**88.87**

Table 3: Accuracy rates (%) obtained by ISS and DFW for both NB and kNN classifiers. Best result highlighted.

Stream	NB-ISS	NB-DFW	kNN-ISS	kNN-DFW	HAT
AGRAWAL	**3.00**	14.48	353.31	1077.56	7.42
ASSETS	**17.66**	33.63	304.95	500.09	35.10
BG	**2.32**	8.81	456.48	450.95	6.14
BG 2	**3.64**	16.60	517.89	691.29	7.11
BG 3	**3.55**	20.07	565.14	778.33	7.88
CONDITIONAL	**8.43**	106.61	735.15	2980.35	14.32
FY A	**12.08**	262.93	770.34	4064.95	40.81
FY B	**41.75**	2284.51	1621.89	14990.08	105.56
FY C	**58.21**	1719.44	745.03	12047.13	233.72
FY D	**66.83**	2816.18	1384.43	19907.43	281.96
FY A Drift	**12.07**	305.55	1000.18	4262.95	38.05
FY B Drift	**41.60**	2317.48	1576.40	15277.31	117.59
LED	**4.18**	59.67	824.93	931.52	14.81
SEA	**5.50**	30.54	426.26	1813.26	11.43

Table 4: CPU Time (s) obtained by ISS and DFW for both NB and kNN classifiers. Best result highlighted.

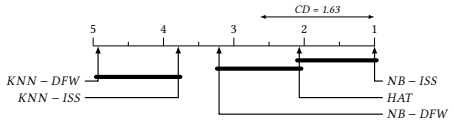

Figure 2: Critical difference chart for CPU time rates.

NB having much better computation time and RAM hour results than their DFW counterparts for all streams. kNN specifically had significantly better computation time and RAM hour performance for ISS than DFW. NB-ISS generally used half as much CPU time compared to the HAT, and achieved better RAM hour results for all but the BG2, BG3, and LED scenarios. In Figures 2 and 3 we report the results of the statistical tests performed for CPU time and RAM-Hours measurements. We see that indeed $NB-ISS$ is the fastest algorithm, followed by the HAT and $NB-DFW$, and that these three are statistically faster when compared to kNN methods. Regarding RAM hours, $NB-ISS$ is again highlighted as the most light-weighted approach, followed by HAT.

Stream	NB-ISS	NB-DFW	kNN-ISS	kNN-DFW	HAT
AGRAWAL	**0.54**	2.65	63.58	49454.93	1.28
ASSETS	**2.39**	4.67	41.13	190.83	6.61
BG	**0.44**	1.70	86.43	222.97	0.47
BG 2	1.08	4.96	152.55	626.55	**0.69**
BG 3	1.05	5.99	166.47	547.08	**0.82**
CONDITIONAL	**3.93**	49.64	341.12	352504.03	7.69
FY A	**7.03**	151.83	443.87	625934.49	54.21
FY B	**70.57**	3811.55	2713.85	539595.71	261.76
FY C	**92.59**	2659.90	1149.98	1980181.04	5495.08
FY D	**117.44**	4803.79	2356.87	1327763.88	3916.08
FY A Drift	**7.01**	176.45	580.64	657448.93	38.27
FY B Drift	**70.25**	3866.58	2644.58	562537.78	386.37
LED	0.97	23.99	326.19	1056.71	**0.93**
SEA	**2.19**	7.12	98.32	165520.00	2.35

Table 5: RAM-Hours (kb-hour) obtained by ISS and DFW for both NB and kNN classifiers. Best result highlighted.

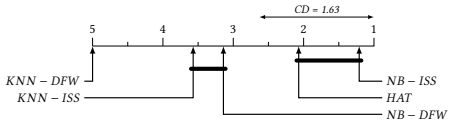

Figure 3: Critical difference chart for RAM-Hours rates.

7 CONCLUSION

In this paper, we introduced a novel feature selection method aimed at addressing feature drift in fast-flowing data streams. Our method divided feature selection into two stages, by first ranking features and creating a subset, then iteratively evaluating and removing the lowest ranked feature from the subset until the subset is empty. Experimental results indicate that our method is successful at improving mean classification accuracy in feature drifting streams for both k-Nearest Neighbours and Naïve Bayes classifiers. In addition to the accuracy improvements, noticeable gains in runtime and RAM hours were achieved for the kNN classifier in scenarios with a large number of irrelevant features.

Future work includes the exploration of different ranking functions or methods in addition to the ones mentioned in this work, as well as exploring different classifiers' interaction with ISS. The accuracy difference between each subset after every iteration of subset selection potentially has value to ranking which was not explored in this work. A potentially more relevant feature or set of features may cause a larger drop in accuracy when it is removed from the subset in comparison to potentially less relevant features.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, 1993. Special issue on Learning and Discovery in Knowledge-Based Databases.

[2] G. H. Albert Bifet, Eibe Frank and B. Pfahringer, editors. *Accurate Ensembles for Data Streams: Combining Restricted Hoeffding Trees using Stacking*, volume 13 of *JMLR Proceedings*. JMLR.org, 2010.

[3] E. Almeida, C. A. Ferreira, and J. Gama. Adaptive model rules from data streams. In *ECML/PKDD (1)*, volume 8188 of *Lecture Notes in Computer Science*, pages 480–492. Springer, 2013.

[4] J. P. Barddal, H. M. Gomes, and F. Enembreck. A survey on feature drift adaptation. In *Proceedings of the International Conference on Tools with Artificial Intelligence*. IEEE, November 2015.

[5] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, 127(Supplement C):278–294, 2017.

[6] J. P. Barddal, H. M. Gomes, F. Enembreck, B. Pfahringer, and A. Bifet. On dynamic feature weighting for feature drifting data streams. In *ECML/PKDD'16*, Lecture Notes in Computer Science. Springer, 2016.

[7] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007.

[8] A. Bifet and R. Gavaldà. *Adaptive Learning from Evolving Data Streams*, pages 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[9] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, Aug. 2010.

[10] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, Dec. 2006.

[11] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[12] J. Duarte and J. Gama. Feature ranking in hoeffding algorithms for regression. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, pages 836–841, 2017.

[13] H. Fu, Z. Xiao, E. Dellandréa, W. Dou, and L. Chen. *Image Categorization Using ESFS: A New Embedded Feature Selection Method Based on SFS*, pages 288–299. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[14] J. Gama and C. Pinto. Discretization from data streams: Applications to histograms and data mining. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 662–667, New York, NY, USA, 2006. ACM.

[15] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, Mar. 2014.

[16] I. Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[17] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[18] A. Metwally, D. Agrawal, and A. El Abbadi. *Efficient Computation of Frequent and Top-k Elements in Data Streams*, pages 398–412. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[19] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, and L. Wan. Heterogeneous ensemble for feature drifts in data streams. In *Advances in Knowledge Discovery and Data Mining*, volume 7302 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2012.

[20] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[21] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, Apr. 1996.

[22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[23] Z. Xiao, E. Dellandrea, W. Dou, and L. Chen. ESFS: A new embedded feature selection method based on SFS. Technical Report RR-LIRIS-2008-018, LIRIS UMR 5205 CNRS/INSA de Lyon/Universite Claude Bernard Lyon 1/Universite Lumiere Lyon 2/Ecole Centrale de Lyon, Sept. 2008.

[24] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 856–863, 2003.

[25] Z. Zhao and H. Liu. Searching for interacting features. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1156–1161, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[26] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu. Advancing feature selection research. *ASU feature selection repository*, pages 1–28, 2010.