Online Learning Method for Drift and Imbalance Problem in Client Credit Assessment

Hang Zhang and Qingbao Liu *

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

* Correspondence: liuqingbao@nudt.edu.com; Tel.: +86-138-7499-2460

Received: 18 June 2019; Accepted: 4 July 2019; Published: 8 July 2019

Abstract: Machine learning algorithms have been widely used in the field of client credit assessment. However, few of the algorithms have focused on and solved the problems of concept drift and class imbalance. Due to changes in the macroeconomic environment and markets, the relationship between client characteristics and credit assessment results may change over time, causing concept drift in client credit assessments. Moreover, client credit assessment data are naturally asymmetric and class imbalanced because of the screening of clients. Aiming at solving the joint research issue of concept drift and class imbalance in client credit assessments, in this paper, a novel sample-based online learning ensemble (SOLE) for client credit assessment is proposed. A novel multiple time scale ensemble classifier and a novel sample-based online class imbalance learning procedure are proposed to handle the potential concept drift and class imbalance in the client credit assessment data streams. The experiments are carried out on two real-world client credit assessment cases, which present a comprehensive comparison between the proposed SOLE and other state-of-the-art online learning algorithms. In addition, the base classifier preference and the computing resource consumption of all the comparative algorithms are tested. In general, SOLE achieves a better performance than other methods using fewer computing resources. In addition, the results of the credit scoring model and the Kolmogorov–Smirnov (KS) test also prove that SOLE has good practicality in actual client credit assessment applications.

Keywords: credit assessment; online learning; ensemble learning; sampling method

1. Introduction

Client credit assessment is an important reference for developing bank financial services and loan approval procedures. Its main purpose is to determine the probability of default and to help banks reduce risk. The earliest client credit assessment method was empirical discriminant provided by credit analysts. This qualitative analysis method relies too much on the professional quality and experience of the evaluators and lacks objectivity.

With the digital technology applications available in banking, a large amount of client data and their credit information are collected. Many studies using machine learning algorithms have been conducted [1], such as linear discriminant analysis (LDA) [2], logistic regression (LR) [3], decision trees (DTs) [4], naive Bayes (NB) [5], artificial neural networks (ANNs) [6,7], support vector machines (SVMs) [8], and ensemble approaches [4,9], to achieve automatic credit assessment. The essence of the client credit assessment is a classification problem. According to the default risk, clients are divided into two categories: “good” and “risk” [10].

In actual situations, client data are collected in chronological order as the bank transaction proceeds. The relationship between client characteristics and credit assessment results is not static. With the developing market environment and economic cycles, the relationship between client characteristics...
and credit assessment results may vary [11]. Traditional machine learning algorithms create a learning model based on historical client data and then use the model to predict new client data. In addition, the actual applications also require the evaluation speed of the credit assessment model. Traditional machine learning algorithms cannot update the model in real-time as new client data arrives. Therefore, the learning model learns only the old relationships. A learning model established by traditional machine learning methods will have a good credit assessment ability for new clients if there is no change in the relationship between client characteristics and credit assessment results. Once the relationship has changed, the traditional machine learning models are likely to make incorrect assessments, which could create risks and cause economic losses. To ensure that the learning model can be updated with the latest client data and make an accurate assessment, this paper proposes applying online learning methods in the field of client credit assessment. In contrast to traditional machine learning methods, online learning methods process the training instance once “on arrival” without storing and make predictions using the current model at each time step [12]. Therefore, online learning algorithms can immediately capture potential changes in the relationship between client characteristics and credit assessment. Additionally, online learning has been proven to achieve good performance in practical applications [13,14].

Online learning is often challenged by the joint issue of concept drift and class imbalance in the field of client credit assessment. In contrast to traditional machine learning algorithms that are trained on static datasets, online learning algorithms address the arriving training instances from the data stream one by one. Suppose the data stream has an underlying probability distribution \( P_t(x,y) \) [15], and the instances in the data stream are generated by it. Once the distribution \( P_t(x,y) \) varies, the characteristic of the instances will change and concept drift will occur. In general, concept drift can be classified into sudden drift and gradual drift according to the changing concept speed [16]. As noted, in the field of credit assessment, the relationship between the client characteristics and the credit assessment results is not static. Many client characteristics, such as income, expenditure, asset value, capital gain, and expected income, are susceptible to macroeconomic and market conditions. The income and asset value of a client with good credit in an economic expansion cycle will be significantly different to that of the same client in an economic contraction. This phenomenon constitutes concept drift in the client credit assessment data stream. Although existing research [11] has recognized potential concept drift in client credit assessment, it has used only a simple integrated approach to improve the overall accuracy.

Another important challenge is the imbalanced class distribution in the client credit assessment data [17,18]. In an actual credit business, many possible default clients are directly rejected in the initial screening, leading to a different number of “good” clients and “risk” clients in the data collection period. The number of good clients is larger than the number of risk clients leading to a problem of asymmetry. Moreover, the cost of misclassifying risk clients as good clients, and misclassifying good clients as risk clients is different. Misclassification of risk clients is more expensive and should be avoided as much as possible. Therefore, the learning model must focus on the minority class (risk clients). To address the class imbalance in the field of client credit assessment, many algorithms apply sampling techniques to balance the training dataset. Brown and Mues [2] applied simple random undersampling and oversampling methods to deal with the class imbalance. Zieba et al. [19] used the synthetic minority oversampling technique (SMOTE) to generate risk clients and achieved better performance than the random sampling methods. However, introducing too much synthetic client data will create more subjective factors. In addition, these sampling methods based on static datasets cannot be used in online learning conditions. In summary, the main challenges of credit assessment are the evaluation speed, the concept drift problem and the class imbalance problem.

Many online learning algorithms have been proposed to address concept drift and class imbalance in data streams. Online learning algorithms for concept drift are commonly categorized into active approaches and passive approaches [20]. Active approaches [21] apply a drift detection mechanism to identify the occurrence of concept drift and then take action to address it, which can handle sudden
concept drift better. However, passive approaches [22,23] evolve the classifier continuously without detecting concept drift and are good at overcoming gradual drift. In an actual client credit assessment data stream, the type of concept drift is unknown, which requires the learning model to be capable of handling different types of concept drift. To handle the class imbalance in online learning conditions, Wang et al. [24] proposed a time-decayed indicator to evaluate the real-time class imbalance degree of the data stream. Then the indicator was used to change the sampling times of the ensemble learning algorithm online bagging (OB) [25] to propose oversampling online bagging (OBU) and undersampling online bagging (UOB).

In this paper, a novel sample-based online learning ensemble for client credit assessment is proposed to solve the deficiencies of traditional machine learning methods and handle concept drift and class imbalance in the credit assessment data stream. First, a novel multiple time scale ensemble classifier is proposed that contains a stable classifier and dynamic classifiers. To address gradual concept drift, the stable classifier learns the whole data streams from the moment the learning procedure begins. To handle sudden drift, the dynamic classifier exists for only a period and learns a partial concept. Second, to overcome the class imbalance, a novel sample-based online class imbalance learning procedure is proposed, which combines both oversampling for the minority instances and undersampling for the majority instances. Two bank client credit assessment cases, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) and Gateway Mobile Switching Centre (GMSC), are analyzed. The proposed sample-based online learning ensemble (SOLE) is compared with other state-of-the-art online learning methods by multiple metrics, such as accuracy, recall, F-score, G-mean and prequential area under the curve (PAUC). Two base classifiers, the Hoeffding tree and the naive Bayes, are used to test the preference of the online learning algorithms for the base classifier. The time and memory consumption of the algorithms are also reported. In general, the proposed SOLE achieves better performance using fewer computing resources and shows high adaptability for different base classifiers. In addition, to verify the feasibility of SOLE in practical applications, the credit scoring model and Kolmogorov–Smirnov (KS) test are applied. The results show that SOLE is suitable for practical application and can help services that need fast credit assessments, such as the E-commerce, online loan approval and financial security testing.

The rest of this paper is organized as follows. Section 2 proposes the multiple time scale ensemble classifier. SOLE is proposed in Section 3. Then, the experimental results and analysis reports are presented in Section 4. Section 5 provides the conclusion.

2. Multiple Time Scale Ensemble Classifier

Online learning algorithms should have the ability to handle potential concept drift in the client credit assessment data stream. In this section, a novel multiple time scale ensemble classifier is proposed to solve this issue. The main contribution of the ensemble method is that it is capable of different types of concept drift. As the sudden concept drift varies immediately, a learning algorithm that only learns the latest instances can adapt to the new concept rapidly, while a classifier that learns more instances can help handle gradual concept drift [26]. In addition, if the old concept reappears, the learning model that is trained over a long time and has learned the old concept will perform better. Therefore, the proposed ensemble classifier has base classifiers of different time scales to address different potential concept drifts. Figure 1 shows the structure of the multiple time scale ensemble classifier.
From Figure 1, the multiple time scale ensemble classifier has a long-term stable classifier $C_s$, and a dynamic classifier damped sliding window $C_d (d = 1, 2, \ldots, D)$. Suppose $S$ is an unlimited data stream $x_i, x_{i+1}, x_{i+2}, \ldots$. The real label of the instances $x_i$, which arrives at time $t_i$, is $y_i$. To capture the concept of the data stream at different times, each $l$ instance of the data stream is seen as a time cycle $T_l$. Therefore, as instances continuously come from the data stream, the data stream can be regarded as consequent time cycles $T_1, T_2, \ldots T_n, T_{n+1} \ldots$.

Base classifiers in the multiple time scale ensemble classifier learn different time scale instances. The stable classifier $C_s$ learns all the instances since the learning procedure handles the gradual and cyclic concept drift of the data stream. The dynamic classifier $C_d$ learns only instances in the limited time cycle, which is defined as:

$$
\begin{align*}
S_s &= \sum_{k=1}^{n} T_k \\
S_d &= \sum_{k=d}^{T_n-D+k} T_{n-D+k}, \quad d = 1, 2, \ldots D
\end{align*}
$$

(1)

where the current time cycle is $T_n$. The stable classifier and dynamic classifiers learn instances of different time scales: those learned by the stable classifier are given by $S_s$ and those by the dynamic classifier are given by $S_d$. The dynamic classifier $C_d$ learns only instances of the most recent $D-d+1$ time cycle. Each classifier in the ensemble classifier has its predictive weight, and the actual prediction result is the weighted combination of all classifiers:

$$
f^E_l(x) = w^s f^{C_s}_l(x) + \sum_{d=1}^{D} w^d f^{C_d}_l(x), \quad l = 0, 1
$$

(2)

where $f^E_l(x)$ is the ensemble prediction that the instance $x$ belongs to class $l$, $w^s$ is the weight of the stable classifier and $w^d (d = 1, 2, \ldots, D)$ are the weights of the dynamic classifiers. The weight of the stable classifier $w^s$ is a constant, but the weight of dynamic classifiers $w^d$ decreases over time. The initial weight of the newly created dynamic classifier is $1/D$, and the weight of the old dynamic classifiers decreases repeatedly according to the creation time, which is shown in Equation (3).

$$
\begin{align*}
\omega^s &\leftarrow \frac{1}{2} \\
\omega^d &\leftarrow \omega^d(1-\frac{1}{D}) \quad , d = 1, 2, \ldots D-1 \\
\omega^D &\leftarrow \frac{1}{D}
\end{align*}
$$

(3)
Therefore, old dynamic classifiers have less predictive weight than the new dynamic classifier, which helps the ensemble classifier to focus more on the latest instances of the data stream. Additionally, the stable classifier and dynamic classifiers both have a half predictive weight when the ensemble classifier makes predictions. The SOLE learning procedure is shown in Algorithm 1.

**Algorithm 1. Learning Procedure**

**Inputs:**
1. \( S \): data stream
2. \( I \): number of instances in a time cycle
3. \( D \): number of the dynamic classifiers
4. \( \alpha \): sampling parameter

**Output:** Ensemble classifier \( E \)

**Initiation:**
1. \( p = 0 \): the indicator of processed instances
2. \( k = 0 \): the indicator of dynamic classifier
3. The stable classifier \( C_s \)
4. The candidate classifier \( C_c \)

**Process:**
1. while (\( S \).hasNext()) do
2. \( x_{\text{new}} = S \cdot \text{getNextInstance}() \)
3. \( \text{TrainOnInstance}(x_{\text{new}}, \alpha, \text{DCRI}[l]) \)
4. \( p = p + 1 \)
5. if \( (p \% I == 0) \) then // reach a new time cycle
6. \( K = k + 1 \)
7. if \( (k > D) \) then //number of dynamic classifiers beyond \( D \)
8. \( C_d \leftarrow C_{d+1} \) \( (d = 1, \ldots, D-1) \)
9. \( C_D \leftarrow C_c \)
10. else
11. \( C_k \leftarrow C_c \) //copy \( C_c \) as the new dynamic classifier
12. end if
13. \( C_c \cdot \text{resetLearning}() \) //reset the candidate classifier
14. end if
15. Calculate weight of classifiers according to equation (3)
16. Compute the damped class imbalance ratio by (4)
17. end if
18. end while

At the early stages of the learning procedure, a stable classifier \( C_s \) and a candidate classifier \( C_c \) are created. The stable classifier exists during the whole learning procedure. The candidate classifier learns only instances of a time cycle. Whenever a new time cycle occurs (line 5), the candidate classifier trained by the last time cycle is copied as a new dynamic classifier and included in the dynamic classifier damped sliding window. Then, the candidate classifier is reset. If the number of dynamic classifiers is greater than \( D \) (lines 7–9), the subsequent classifier replaces the former classifiers one by one, \( C_d \leftarrow C_{d+1} \) \( (d = 1, \ldots, D-1) \), to keep the number of dynamic classifiers at \( D \). The candidate classifier is used as the new dynamic classifier \( C_D \). Then, the algorithm updates the weight of the classifiers according to Equation (3) and calculates the damped class imbalance ratio using Equation (4). In the dynamic classifier damped sliding window, older dynamic classifiers are trained by more former arriving instances and have less predictive weight, which helps the ensemble classifier always attach importance to the latest data stream concept. The stable classifier exists for the whole learning process and can address gradual concept drift and cyclic concept drift better.
3. Sample-Based Online Class Imbalance Learning Procedure

If a data stream is class imbalanced, the model will lack minority class training instances. The sampling technique is a commonly used method to address class imbalance learning. However, traditional sampling techniques, such as random sampling or smart sampling [2,19], are not applicable to online learning conditions because that online learning requires the learning model to address one instance for each time step. For the online learning condition, sampling methods can be realized by changing the instance training times. OB [25] is an online ensemble learning algorithm that modifies offline bagging, which contains multiple base classifiers. For each instance, the base classifier is updated \( K \times \) times, following the \( \text{Poisson}(\lambda = 1) \) distribution. For the class imbalance learning condition, the categories are divided into a majority (negative) and a minority (positive) according to their amounts. To balance the class imbalance ratio, the proposed sample-based online class imbalance learning procedure oversamples the positive instances and undersamples the negative instances according to the real-time class imbalance ratio of the data stream. Therefore, the classifier can be trained by a nearly equal number of instances of different classes.

To determine the training times of an instance, the damped class imbalance ratio (DCIR) which calculates the class imbalance ratio in real time, is proposed. The candidate classifier is copied as the dynamic classifier to the ensemble classifier. As the candidate classifier is trained by instances of a time cycle, each dynamic classifier \( C_d \) has its part of initialization instances \( R_d \). The class distribution \( H_d[l] \) of \( R_d \) is also calculated. Therefore, the DCIR is calculated as:

\[
DCIR(l) = \frac{\sum_{d=1}^{D} H_d[l]w^d}{\sum_{l=0}^{1} \sum_{d=1}^{D} H_d[l]w^d}
\]

where \( w^d \) is the weight of the dynamic classifier \( C_d \). First, the number of the instances of each class (0: negative, 1: positive) in the initialization dataset \( R_d \) is summed, and DCIR\((l)\) is the weighted summation of class \( l \) divided by the weighted summation of total classes. According to Equation (3), the weight of dynamic classifier \( w^d \) is time-decayed and the earlier generated dynamic classifier has a lower weight. Therefore, the DCIR is mainly determined by the latest class imbalance ratio of the data stream. The sample-based online class imbalance learning procedure is shown in Algorithm 2.

---

**Algorithm 2. TrainOnInstance**

**Input:**
1. \( x \): instance to be processed
2. \( \alpha \): sampling parameter
3. \( DCIR[l] \): damped class imbalance ratio \((l = \text{negative}, \text{positive})\)

**Output:** The classifier \( C_{\text{new}} \)

**Process:**
1. \( y = \text{getreallabel}(x) \)
2. **if** \( (y \rightarrow \text{negative}) \) **then**
3. \( \text{sampleweight} = DCIR[\text{positive}] / DCIR[\text{negative}] \)
4. **else**
5. \( \text{sampleweight} = DCIR[\text{negative}] / DCIR[\text{positive}] \)
6. **end if**
7. For each classifier in the ensemble, set \( K \sim \text{Poisson}(\text{sampleweight} \times \alpha) \). Then, update the classifier with \( x \) for \( K \) times

---

For a new instance \( x \) from the data stream, the algorithm obtains the real label \( y \) of the instance \( x \). Then, the algorithm determines whether instance \( x \) belongs to the minority class. If \( x \) belongs to the majority class (i.e., \( x \) is negative) (line 2), the algorithm applies undersampling to learn instance \( x \). The sample weight is set to \( DCIR[\text{positive}] \) divided by \( DCIR[\text{negative}] \) (line 3). Otherwise, the algorithm applies the oversampling method to instance \( x \) (line 5). The sampleweight influences the \( \lambda \) of the
Poisson distribution. Finally, the stable classifier, candidate classifier and all the dynamic classifiers train on the instance \(x\) \(K\) times by the Poisson distribution (sampleweight\(*\alpha\)). In general, the classifiers train more times on the positive instance and train fewer times on the minority class, which can balance the training set in the online learning condition. Particularly, this sampling method is based on only the latest instances without using historical information, which can avoid using the instances of an old concept.

### 4. Experiments

In this section, a systematic experiment comparison of online learning algorithms for credit assessment is presented. The proposed SOLE is compared with other state-of-the-art online learning algorithms on two real-world client credit assessment cases by multiple evaluation metrics. Section 4.1 introduces the client credit assessment cases and experimental materials. Section 4.2 presents the evaluation metrics. The comparative algorithms and the contrast base classifier are shown in Section 4.3. The experimental results of SOLE compared with other online learning algorithms are presented in Section 4.4. Section 4.5 applies the credit scoring model and KS test to verify the practicality of SOLE in actual client credit assessment applications. Finally, Section 4.6 presents the resource consumption.

#### 4.1. Research Cases and Materials

Two bank credit assessment cases, PAKDD [11] and GMSC [22], are used in the experimental comparison. GMSC is a classic credit assessment case that comes from the Kaggle competition. GMSC contains the data of bank clients, such as age, monthly income, number of credit cards, and dependents, which are used to determine whether a loan should be granted. GMSC contains 150,000 instances, 10 explanatory attributes and 1 predictive attribute. PAKDD is another commonly used benchmark data stream in online learning. It comes from the PAKDD 2009 competition and mainly tests the impact of market changes in several business years on the performance of the classifier. The class distribution of PAKDD is also imbalanced. PAKDD contains 50,000 instances, 28 explanatory attributes and 1 predictive attribute. Table 1 shows the main characteristics of the two experimental credit assessment cases.

| Case        | No. Instances | No. Attributes | No. Positive | No. Negative | Imbalance Ratio |
|-------------|---------------|----------------|--------------|--------------|-----------------|
| GMSC        | 150 k         | 10             | 10026        | 139974       | 1/14            |
| PAKDD       | 50 k          | 28             | 9874         | 40126        | 1/4             |

Traditional machine learning evaluation methods divide a dataset into a training set and testing set. The model is trained on the static training set and then used to predict the testing set. Online learning methods obtain instances one by one, and the learning models can only learn one instance at each time step, which is a more challenging learning condition. In the comparative experiments, prequential evaluation is used: when a new instance arrives, first test it and then train on it. All the experiments are carried out on the Massive Online Analysis (MOA) [27], which is designed for online learning conditions. The experimental machine has an eight-core CPU (Intel i7-6700) and 32 GB RAM.

#### 4.2. Evaluation Metrics

Traditional overall accuracy and error rate are commonly used indicators to evaluate classification performance. However, when the learning datasets are class imbalanced, they can only reflect the overall performance. Therefore, other metrics have been adopted in the binary class imbalance learning condition [28]. Generally, the minority class is treated as the positive class, and the majority class is usually treated as negative. Table 2 shows the confusion matrix of the binary classification problem, which generates four numbers on the testing data.
Table 2. Confusion matrix of the binary problem.

|                  | Predicted Positive | Predicted Negative |
|------------------|--------------------|--------------------|
| Actual Positive  | True Positive (TP) | False Negative (FN)|
| Actual Negative  | False Positive (FP)| True Negative (TN) |

Then, the recall is defined as:

\[ \text{recall} = \frac{TP}{TP + FN} \] (5)

\[ \text{precision} = \frac{TP}{TP + FP} \] (6)

Recall is the ratio of the True Positive rate with the positive instance percentage. Precision is the proportion of the True Positive rate with respect to the sum of the True Positive rate and the False Positive rate. The best situation of online class imbalance learning is improving recall without decreasing precision. However, it is conflicting to improve recall and precision together. Thus, the F-score is used to show the trade-off between them:

\[ \text{F-score} = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \] (7)

where \( \beta \) is the relative importance factor of recall and precision, and which is usually set to 1. G-mean is also an indicator to replace overall accuracy:

\[ \text{G-mean} = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}} \] (8)

The area under the curve (AUC) computes the area under the receiver operating characteristic (ROC) curve, which is also a good measurement for class imbalance learning condition. However, AUC is only suitable for the offline learning situation. To improve the traditional AUC for online learning conditions, Brzezinski et al. [29] modified the AUC and proposed the prequential area under the curve (PAUC). The value of the indicator is continually updated in the online learning situation. Whenever an instance is processed, the indicator changes. Overall, the experiments in this paper compare the performance of the proposed SOLE with online learning algorithms on traditional accuracy, recall, F-score, G-mean and PAUC.

4.3. Comparative Methods

In this section, the comparative methods in the experiments are introduced. The selected comparative methods are all state-of-the-art online learning methods for concept drift and class imbalance. As the ensemble method tends to be more accurate than the base classifier, all the applied algorithms are ensemble online learning methods. Overall, Leveraging Bagging (LB) [23], learn++NIE [26], OOB, UOB [24], Online Accuracy Update Ensemble (OAUE) [30] and Adaptive Random Forest (ARF) [22] are compared. To control the performance of the base classifier for all algorithms, all the comparative algorithms use the same base classifiers except ARF. The Hoeffding tree and the naive Bayes are used as the base classifier for comparison. The default parameter settings of the base classifier are shown in Table 3.

The Hoeffding tree is an incremental, anytime DT induction algorithm, it is very efficient and suitable for the classification of large amounts of data. The naive Bayes classifier is a relatively simple probability classifier based on Bayes’ theorem and has a low computational resource cost. Both the Hoeffding tree and the naive Bayes are widely used as the base classifiers in studies of the online learning situation. In particular, ARF uses the ARFHoeffding tree, which specifically modifies
the traditional Hoeffding tree as the base classifier. OAUE can only apply the Hoeffding tree as a base classifier.

### Table 3. Default parameter settings of the Hoeffding tree and the naive Bayes.

| Base classifier        | Default Parameter Setting                                                                 | Algorithms |
|------------------------|------------------------------------------------------------------------------------------|------------|
| Hoeffding tree         | memory estimate period: 2, 000, 000, grace period: 100, split criterion: InfoGainSplit, split confidence: 0, tie threshold: 0.05, leaf prediction: NBAdaptive, nbThreshold: 0 | SOLE, LB   |
|                        |                                                                                         | Learn++NIE, OOB, UOB, OAUE             |
| Naive Bayes            | Naive Bayes classifier is a relatively simple probability classifier based on Bayes' theorem. Naive refers to the assumption that all features in the model are highly independent and the correlation between features is not considered. | SOLE, LB   |
|                        |                                                                                         | Learn++NIE, OOB, UOB                   |
| ARFHoeffding tree      | subspace size: 2, memory estimate period: 1, 000, 000, grace period: 100, split criterion: InfoGainSplit, split confidence: 0, tie threshold: 0.05, leaf prediction: NBAdaptive, nbThreshold: 0 | ARF        |

4.4. Experimental Results

4.4.1. Comparison of Online Learning Algorithms

In this section, the experimental results of all the comparison methods are presented using different metrics. All the comparative experiments are carried out 10 times and the average is taken as the result. All the ensemble methods use 10 base classifiers. The default parameters of SOLE are $D = 10$, $T = 500$, and $\epsilon = 1$. $D$ is the number of dynamic classifiers. $T$ is the size of the time cycle, and $\alpha$ is the sampling parameter. Table 4 shows the results using the Hoeffding tree and the ARFHoeffding tree as the base classifiers, including SOLE, LB, learn++NIE, OOB, UOB, OAUE and ARF. And the bold number is the optimal results.

For the GMSC (imbalance ratio: 1/14), LB and ARF achieve the same best accuracy, which is 0.2% higher than the learn++NIE accuracy. However, accuracy reflects only the overall classification performance. For the recall, three sample-based methods, SOLE, OOB and UOB, perform better than other online learning methods, which proves the effectiveness of the sampling technique in dealing with class imbalance. SOLE obtains the highest recall, which is 7.9% higher than UOB. Compared with the accuracy and recall, it can be concluded that while the overall classification performance is good, the actual classification performance on the minority class may be poor. In addition, SOLE achieves first place in all three metrics, F-score, G-mean, and PAUC. SOLE, OOB and UOB perform better than the other methods for these three metrics. Furthermore, learn++NIE and OAUE achieve recall values of only 2.2% and 0.0%, which shows that the algorithm predicts nearly all instances as the majority class. Thus, learn++NIE and OAUE lose their judgment capacity and cannot be used in practical applications.

As for the PAKDD (imbalance ratio: 1/4), learn++NIE and OAUE achieve first place in accuracy, which is 0.1% higher than the LB accuracy. The accuracy of SOLE is lower than that of LB, learn++NIE, OAUE and ARF, but higher than that of the other sampling methods, OOB and UOB. For the recall metric, LB, learn++NIE, OAUE and ARF obtain very low values, meaning that these algorithms regard almost all the instances as the majority class. UOB has the best recall value, of 68.1%, which is 12.4% higher than SOLE. However, the overall accuracy of UOB is only 53.5, which means that UOB misclassifies almost half of the instances. Therefore, it can be concluded that UOB obtains a better classification
performance for the minority class (i.e., recall value) at the expense of misclassifying many majority class instances. In practical client credit assessment applications, treating too many good clients as risk clients will cause adverse effects on the business. For the F-score, G-mean, and PAUC metrics, SOLE achieves first place and is better than the other algorithms.

Table 4. Average results using the Hoeffding tree as the base classifier (%).

| Cases        | Algorithms | Accuracy | Recall | F-score | G-mean | PAUC |
|--------------|------------|----------|--------|---------|--------|------|
| GMSC (IR: 1/14) | SOLE 89.7 68.5 47.0 79.0 88.7 |
|              | LB 93.5 10.2 17.4 31.9 84.1 |
|              | Learn++NIE 93.2 2.2 0.0 15.0 50.0 |
|              | OOB 89.7 55.5 41.9 71.5 84.9 |
|              | UOB 85.7 60.6 36.1 72.8 79.1 |
|              | OAUE 93.3 0.0 - 0.0 50.5 |
|              | ARF 93.5 9.1 15.6 30.0 80.4 |
| PAKDD (IR: 1/4) | SOLE 75.8 55.7 47.9 67.1 69.1 |
|              | LB 80.2 0.6 1.2 7.7 63.8 |
|              | Learn++NIE 80.3 0.1 0.0 0.0 50.1 |
|              | OOB 70.7 38.0 33.9 54.7 63.5 |
|              | UOB 53.5 68.1 36.6 58.3 61.3 |
|              | OAUE 80.3 0.1 0.0 0.0 50.3 |
|              | ARF 80.1 1.7 3.2 12.9 63.2 |

For the experiments using the naive Bayes as the base classifier, only SOLE, LB, learn++NIE, OOB, and UOB are included in the comparison. Table 5 shows the average results using the naive Bayes as the base classifier. For the GMSC (imbalance ratio: 1/14), LB achieves the best classification accuracy, which is 0.2% higher than OOB and UOB. SOLE obtains the highest value on the recall metric, which is 0.4% higher than learn++NIE. LB, OOB and UOB achieve only low recall values, which means they perform poorly for the minority class. In addition, SOLE performs best on the F-score, G-mean, and PAUC metrics and is better than the other algorithms. For the PAKDD (imbalance ratio: 1/4), LB also achieves the best performance for the accuracy metrics but obtains the lowest recall value. OOB achieves the best recall value of 79.7%, and UOB achieves the second place of 79.4%. However, their classification accuracy is lower than 50%, reflecting the fact that they acquire a better performance for the minority class at the expense of poor performance for the majority class. Moreover, SOLE achieves first place for three metrics: F-score, G-mean, and PAUC. SOLE performs better because it proposed a multi-time scale ensemble classifier and a sample-based learning method to address the joint problem of concept drift and class imbalance, while other state-of-the-art methods only focus on solving one problem.

Table 5. The average results using the naive Bayes as the base classifier.

| Cases        | Algorithms | Accuracy | Recall | F-score | G-mean | PAUC |
|--------------|------------|----------|--------|---------|--------|------|
| GMSC (IR: 1/14) | SOLE 87.4 52.3 36.1 68.5 77.8 |
|              | LB 93.2 3.10 5.7 17.6 67.6 |
|              | Learn++NIE 74.5 51.9 21.4 62.9 67.8 |
|              | OOB 93.0 4.4 7.8 21.0 68.2 |
|              | UOB 93.0 10.3 16.4 31.9 71.7 |
| PAKDD (IR: 1/4) | SOLE 52.3 75.6 38.8 59.3 69.7 |
|              | LB 64.9 40.8 31.4 53.8 58.3 |
|              | Learn++NIE 49.6 64.1 33.4 54.3 56.1 |
|              | OOB 43.8 79.7 35.9 52.8 60.2 |
|              | UOB 45.2 79.4 36.4 54.1 61.2 |
4.4.2. Base Classifier Preference

Compared with the performances of the algorithms using different base classifiers, it can be concluded that the algorithms have preferences for the base classifier. The base class is also biased against the dataset. Table 6 shows the bias of the base classifier by using different colors. The values in the table are the results using the Hoeffding tree as the base classifier minus the results using the naive Bayes as the base classifier. A positive value means that an algorithm performs better using the Hoeffding tree as the base classifier. A negative value means that an algorithm performs better using the naive Bayes as the base classifier. Orange cells indicate that the Hoeffding tree performs better, and green cells indicate that the naive Bayes performs better. The depth of the color has three levels according to the size of the value (0–10%, 10–20%, >20%).

| Cases       | Algorithms | Accuracy | Recall | F-score | G-mean | PAUC  |
|-------------|------------|----------|--------|---------|--------|-------|
| GMSC (IR: 1/14) | SOLE       | 2.3      | 16.2   | 10.9    | 10.5   | 10.9  |
|             | LB         | 0.3      | 7.1    | 11.7    | 14.3   | 16.5  |
|             | Learn++NIE | 18.7     | -49.7  | -21.4   | -47.9  | -17.8 |
|             | OOB        | -3.3     | 51.1   | 34.1    | 50.5   | 16.7  |
|             | UOB        | -7.3     | 50.3   | 19.7    | 40.9   | 7.4   |
| PAKD (IR: 1/4) | SOLE       | 23.5     | -19.9  | 9.1     | 7.8    | 0.9   |
|             | LB         | 15.3     | -40.2  | -30.2   | -46.1  | -6.3  |
|             | Learn++NIE | 30.7     | -64.2  | -33.4   | -54.3  | -6.3  |
|             | OOB        | 26.9     | -41.7  | -2      | 1.9    | 3.3   |
|             | UOB        | 8.3      | -11.3  | 0.2     | 4.2    | 0.1   |

Table 6. The performance bias of the base classifier.

First, the performance for different credit assessment cases is compared. It can be concluded that the number of green cells for PAKDD is larger than that for GMSC. This phenomenon may be related to the class imbalance ratio or the characteristics of the dataset. Second, the preferences of different algorithms are compared. Learn++NIE performs better using the naive Bayes as the base classifier. The performance gap between using the two base classifiers is apparent. LB performs better using the Hoeffding tree for GMSC, while it achieves a better performance using the naive Bayes for PAKDD. SOLE, OOB and UOB achieve better results using the Hoeffding tree. In general, SOLE has the best adaptability, as the performance gaps of SOLE between different base classifiers are minimal.

4.4.3. PAUC–Time Curves

In this section, to intuitively show the classification performance of different algorithms, the PAUC–time curves are plotted. Figure 2 shows the PAUC–time curves for GMSC using different base classifiers. SOLE achieves the highest PAUC using both the Hoeffding tree and the naive Bayes as the base classifier. For the PAUC–time curves using the Hoeffding tree as the base classifier, SOLE achieves a low value at the beginning of the learning process. The PAUC value of SOLE continues growing and obtains first place at the end of the process. For the PAUC–time curves using the naive Bayes as the base classifier, the UOB achieves first place at the beginning. However, the PAUC of UOB continues to decrease, and the PAUC of SOLE continues to increase. Thus, SOLE performs better than UOB after approximately the 15 k time step.
Figure 2. PAUC–time curves for GMSC using different base classifiers.

Figure 3 shows the PAUC–time curves for PAKDD using different base classifiers. SOLE achieves the highest PAUC for both using the Hoeffding tree and the naive Bayes as the base classifier. For the PAUC–time curves using the Hoeffding tree as the base classifier, SOLE achieves fourth place at the beginning. Additionally, the SOLE PAUC value continues to increase, but the PAUC of the other algorithms is stable. Therefore, SOLE achieves the highest PAUC at the end. Then, for the PAUC–time curves using the naive Bayes as the base classifier, SOLE is better than the other algorithms throughout the whole learning procedure.

Figure 3. PAUC–time curves for PAKDD using different base classifiers.

4.4.4. Parameter Sensitivity of SOLE

To explore the impact of parameter settings on the classification performance of SOLE, in this section, the parametric comparison experiment is carried out. Experiments compare the main parameters, number of dynamic classifiers $D$, time cycle $T$, and the sampling parameter $\alpha$ whose default settings are $D = 10$, $T = 500$, and $\epsilon = 1$. As for each group of parameter settings, the comparative experiments are carried out in parallel 10 times on both the cases and the averages are calculated. Table 7 shows the average results with different parameter settings by using the Hoeffding tree as the base classifier. Only the Hoeffding tree as the base classifier is presented because of the same phenomena as using the naive Bayes as the base classifier.

From Table 7, all the parameters have an impact on the classification performance of SOLE. First, increasing the number of dynamic classifiers can help improve the performance to some extent. Within
the results of all the parameters, $D = 15$ is the best setting for GMSC, and $D = 20$ is the best setting for PAKDD. Second, SOLE performs better at time cycles of 500 and 750 for PAKDD, but the time cycle shows no apparent influence on GMSC. It shows that the best value of the time cycle is influenced by the features of the data streams. Finally, the classification accuracy improves when the algorithm uses a larger sampling parameter. However, the classification recall shows the opposite situation. Intuitively, the larger sampling parameter will increase the classification performance for the majority class but decrease the classification performance for the minority class.

| Cases               | Parameters | Accuracy | Recall | F-score | G-mean | PAUC |
|---------------------|------------|----------|--------|---------|--------|------|
| GMSC (IR: 1/14)     | $D = 1$    | 84.7     | 66.4   | 36.7    | 75.6   | 82.4 |
|                     | $D = 5$    | 87.8     | 67.6   | 42.5    | 77.7   | 86.9 |
|                     | $D = 10$   | 89.7     | 68.5   | 47.0    | 79.0   | 88.7 |
|                     | $D = 15$   | 90.4     | 68.9   | 48.9    | 79.6   | 89.9 |
|                     | $D = 20$   | 90.1     | 68.8   | 48.1    | 79.4   | 89.4 |
| I = 100             | 89.5       | 68.4     | 46.5   | 78.9    | 88.4 |
| I = 250             | 89.7       | 68.4     | 47.0   | 79.0    | 88.7 |
| I = 500             | 89.7       | 68.5     | 47.0   | 79.0    | 88.7 |
| I = 750             | 89.7       | 68.5     | 47.0   | 79.0    | 88.7 |
| I = 1000            | 89.8       | 68.5     | 47.2   | 79.1    | 88.9 |
| $\varepsilon = 1$   | 89.7       | 68.5     | 47.0   | 79.0    | 88.7 |
| $\varepsilon = 5$   | 91.3       | 66.4     | 50.4   | 78.6    | 87.7 |
| $\varepsilon = 10$  | 91.7       | 65.3     | 51.2   | 78.2    | 86.5 |
| PAKDD (IR: 1/4)     | $D = 1$    | 71.2     | 53.9   | 42.8    | 63.8   | 65.7 |
|                     | $D = 5$    | 73.3     | 54.1   | 44.8    | 65.0   | 67.8 |
|                     | $D = 10$   | 75.8     | 55.7   | 47.9    | 67.1   | 69.1 |
|                     | $D = 15$   | 76.3     | 55.9   | 48.5    | 67.5   | 69.4 |
|                     | $D = 20$   | 76.5     | 55.9   | 48.8    | 67.6   | 69.4 |
| I = 100             | 73.1       | 51.9     | 43.6   | 63.8    | 65.5 |
| I = 250             | 74.7       | 53.4     | 45.8   | 65.4    | 67.4 |
| I = 500             | 75.8       | 55.7     | 47.9   | 67.1    | 69.1 |
| I = 750             | 75.9       | 55.7     | 48.0   | 67.1    | 69.2 |
| I = 1000            | 75.2       | 54.6     | 46.8   | 66.2    | 68.4 |
| $\varepsilon = 1$   | 75.8       | 55.7     | 47.9   | 67.1    | 69.1 |
| $\varepsilon = 5$   | 76.1       | 52.3     | 46.7   | 65.5    | 66.8 |
| $\varepsilon = 10$  | 78.4       | 49.6     | 47.9   | 65.2    | 67.6 |

4.5. Feasibility Analysis in Practical Application

4.5.1. Credit Scoring Model

The traditional credit assessment models provide only the classification probability that a client is a good clients or a risk clients. However, they are not easy to understand and use in practical applications. To present the credit condition of each client, the credit score is calculated based on the predicted probability provided by the proposed SOLE. The credit score is calculated as [31]:

$$score = \ln \frac{P}{1-P} \cdot factor + offset$$

where $P$ is the probability that a client is a good clients, $1-P$ is the probability that a client is a risk clients, $factor$ is the linear transform coefficient, which is usually a logarithmic value, and $offset$ is the adjustment constant that keeps the credit score in the target interval.
4.5.2. Bad Debts and the KS Test

To verify the feasibility of SOLE in practical applications, the credit scores of the clients for GMSC are calculated. The learning model is SOLE using the naive Bayes as the base classifier. factor is set to ln(60), and offset is set to 600. The credit score results are sorted from high to low and then divided into ten intervals. Each score interval has 15 k clients. Table 8 shows the distribution of credit scores.

Table 8. Credit score distribution for GMSC of SOLE using the naive Bayes.

| Score Interval | Num. “Good” | Num. “Risk” | “Good” Percentage | “Risk” Percentage | Bad Debts |
|----------------|-------------|-------------|-------------------|-------------------|-----------|
| 0–598.9        | 10194       | 4806        | 0.07              | 0.48              | 0.32      |
| 598.9–601.9    | 13579       | 1421        | 0.17              | 0.62              | 0.09      |
| 601.9–603.5    | 14181       | 819         | 0.27              | 0.70              | 0.05      |
| 603.5–604.6    | 14339       | 661         | 0.37              | 0.77              | 0.04      |
| 604.6–605.7    | 14464       | 536         | 0.48              | 0.82              | 0.04      |
| 605.7–606.8    | 14552       | 448         | 0.58              | 0.87              | 0.03      |
| 606.8–608.3    | 14577       | 423         | 0.69              | 0.91              | 0.03      |
| 608.3–610.3    | 14674       | 326         | 0.79              | 0.94              | 0.02      |
| 610.3–615.1    | 14686       | 314         | 0.89              | 0.97              | 0.02      |
| 615.1–734.8    | 14730       | 270         | 1.00              | 1.00              | 0.02      |

As shown in Table 8, risk clients are mainly distributed in the score intervals under the credit score of 604.6. As the credit score increases, the bad debt ratio decreases rapidly. Therefore, SOLE effectively reflects the real credit condition of the clients. The credit score can be used as the basis to assess the credit condition of the clients. In the actual financial risk control business, the KS test is usually used to evaluate the performance of a model. The KS test uses the difference between the accumulative percentage of good clients and the accumulative percentage of risk clients to show the distinguishing ability of a model. The larger the value of the KS test, the better the distinguishing ability. Table 9 shows the relationship between the KS test and distinguishing ability [31].

Table 9. The relationship between the Kolmogorov–Smirnov (KS) test and distinguishing ability.

| KS Test Value | Distinguishing Ability |
|---------------|------------------------|
| <0.20         | none                   |
| 0.20–0.30     | low                    |
| 0.30–0.40     | medium                 |
| 0.40–0.50     | high                   |
| 0.50–0.90     | extremely high         |
| >0.90         | anomaly                |

Figure 4 shows the KS curves for the GMSC of SOLE using the naive Bayes as the base classifier. The KS value is 0.47, which represents a high distinguishing ability. Thus, it proves that SOLE can be used in practical credit assessment.
4.6. Resources Consumption

For online learning models, resource consumption is an important feature for verifying whether the algorithms are capable of real-world applications. Table 10 shows the resource consumption of the algorithms for the two cases. In general, SOLE consumes the least time and memory for both GMSC and PAKDD using the naive Bayes as the base classifier. It also consumes the least time and memory for GMSC using the Hoeffding tree as the base classifier. For the experiments on PAKDD using the Hoeffding tree as the base classifier, learn++NIE costs the least time, OAUE costs the least memory, and SOLE also has a good performance on the resource consumption.

First, the resource consumption results by using the Hoeffding tree as the base classifier are compared. LB and ARF consume significantly more resources than other methods, meaning that LB and ARF will have more delays in real-time practical applications. OOB costs more time and memory than UOB because of the different sampling strategy. As the class imbalance ratio increases, OOB will require more resources for minority class instances, but UOB will cost less resources by undersampling the majority class instances. Therefore, the resource consumption of OOB for GMSC is higher than that for PAKDD.

Second, the experimental results are compared by using the naive Bayes as the base classifier. Learn++NIE costs significantly more time and memory. As the naive Bayes is a simpler base classifier than the Hoeffding tree, all the comparative methods except learn++NIE consume fewer resources using the naive Bayes as the base classifier. Therefore, it can be concluded that the learn++NIE has its preference by using the naive Bayes as the base classifier. By using the naive Bayes as the base classifier, learn++NIE will consume more resources and achieve better performance. In summary, the datasets, algorithm characteristics and the base classifiers all affect the source consumption.
Table 10. Resource consumption of different algorithms.

| Algorithm          | SOLE (1/14) | LB (1/14) | Learn++ NIE (1/14) | OOB (1/14) | UOB (1/14) | ARF (1/14) |
|--------------------|-------------|-----------|--------------------|-----------|-----------|------------|
|                    | Time (s)    | RAM (GB)  | Time (s)           | RAM (GB)  | Time (s)  | RAM (GB)   |
| Hoeffding Tree     | 4.2         | 1.6×10^-7 | 5.1                | 2.4×10^-7 | 3.5       | 3.7×10^-8 |
| GMSC (IR: 1/14)    |             |           |                    |           |           |            |
| PAKDD (IR: 1/4)    |             |           |                    |           |           |            |
| Naive Bayes as the Base Classifier |          |            |                    |           |           |            |
| GMSC (IR: 1/14)    |             |           |                    |           |           |            |
| PAKDD (IR: 1/4)    |             |           |                    |           |           |            |

5. Conclusions

Machine learning algorithms have been used in client credit assessment applications. However, there has been no detailed research focusing on solving the joint issues of concept drift and class imbalance in client credit assessment. To handle the potential concept drift and class imbalance in the client credit assessment data stream, the novel SOLE method is proposed for client credit assessment, which also first applies online learning methods in client credit assessment applications. A novel multiple time scale ensemble classifier contains a stable classifier and many dynamic classifiers to address different types of concept drift. A novel sample-based online learning method based on both oversampling the minority instances and undersampling the majority instances is proposed to balance the class distribution. The experiments provide a comprehensive comparison between the proposed SOLE and other state-of-the-art online learning algorithms. Two real-world credit assessment cases, GMSC and PAKDD, were used in the comparison. The results showed that SOLE performs better than other state-of-the-art online learning algorithms and consumes fewer computing resources. To verify that SOLE is capable of practical applications, the credit scoring model and the KS test were also applied to translate the prediction results to the credit score and quantify the distinguishing ability of the model. Compared with the traditional machine learning and modeling methods, SOLE can address the joint problems of concept drift and class imbalance, and shows rapid adaptability, which benefits from the characters of online learning.

Author Contributions: H.Z. conceived the algorithm; H.Z. and Q.L. designed the experiments; H.Z. performed the experiments and wrote the paper. H.Z. and R.W. edited the manuscript. All authors reviewed and approved the final manuscript.

Funding: This research was funded by the China Advance Research Fund (No. 9140C830304150C83352).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Munkhdalai, L.; Munkhdalai, T.; Namsrai, O.-E.; Lee, J.Y.; Ryu, K.H. An empirical comparison of machine-learning methods on bank client credit assessments. *Sustainability* 2019, 11, 699. [CrossRef]
2. Brown, I.; Mues, C. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Syst. Appl.* 2012, 39, 3446–3453. [CrossRef]
3. Arminger, G.; Enache, D.; Bonne, T. Analyzing credit risk data: A comparison of logistic discrimination, classification tree analysis, and feedforward networks. *Soc. Sci. Electron. Publ.* 1997, 12, 293–310.
4. Lessmann, S.; Baesens, B.; Seow, H.V.; Thomas, L.C. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *Eur. J. Oper. Res.* 2015, 247, 124–136. [CrossRef]
5. Kultur, Y.; Caglayan, M.U. Hybrid approaches for detecting credit card fraud. *Expert Syst.* 2017, 34, e12191. [CrossRef]
6. Khemakhem, S.; Said, F.B.; Boujelbene, Y. Credit risk assessment for unbalanced datasets based on data mining, artificial neural network and support vector machines. *J. Mod. Man.* 2018, 13, 932–951. [CrossRef]
7. Li, C.; Peng, H. Credit Risk Assessment for Rural Credit Cooperatives Based on Improved Neural Network. In Proceedings of the International Conference on Smart Grid & Electrical Automation, Changsha, China, 27–28 May 2017.

8. Li, Z.; Ye, T.; Ke, L.; Zhou, F.; Wei, Y. Reject inference in credit scoring using semi-supervised support vector machines. *Expert Syst. Appl.* 2017, 74, 105–114. [CrossRef]

9. Nanni, L.; Lumini, A. An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Syst. Appl.* 2009, 36, 3028–3033. [CrossRef]

10. Huang, C.L.; Chen, M.C.; Wang, C.J. Credit scoring with a data mining approach based on support vector machines. *Expert Syst. Appl.* 2007, 33, 847–856. [CrossRef]

11. Linhart, C.; Harari, G.; Abramovich, S.; Buchris, A. Pakdd Data Mining Competition 2009: New Ways of Using Known Methods. In Proceedings of the Pacific-Asia International Conference on Knowledge Discovery & Data Mining: New Frontiers in Applied Data Mining, Bangkok, Thailand, 27–30 April 2009.

12. Wang, S.; Minku, L.L.; Yao, X. Online class imbalance learning and its applications in fault detection. *Int. J. Comput. Intell. Appl.* 2013, 12, 2340–2355. [CrossRef]

13. Frances-Villora, J.V.; Rosado-Muñoz, A.; Bataller-Mompean, M.; Barrios-Aviles, J.; Guerrero-Martinez, J.F. Moving learning machine towards fast real-time applications: A high-speed FPGA-based implementation of the os-elm training algorithm. *Electronics* 2018, 7, 308. [CrossRef]

14. Sousa, M.R.; Gama, J.; Brandão, E. A new dynamic modeling framework for credit risk assessment. *Expert Syst. Appl.* 2016, 45, 341–351. [CrossRef]

15. Ditzler, G.; Roveri, M.; Alippi, C.; Polikar, R. Learning in nonstationary environments: A survey. *IEEE Comput. Intell. Mag.* 2015, 10, 12–25. [CrossRef]

16. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* 2018, 1. [CrossRef]

17. Zhang, L.; Wang, W.X. A re-sampling method for class Imbalance Learning with credit data. In Proceedings of the International Conference of Information Technology, Las Vegas, NV, USA, 11–13 April 2011.

18. Marqués, A.I.; García, V.; Sánchez, J.S. On the suitability of resampling techniques for the class imbalance problem in credit scoring. *J. Oper. Res. Soc.* 2013, 64, 1060–1070. [CrossRef]

19. Zieba, M.; Härdle, W.K. Beta-boosted ensemble for big credit scoring data. In *Handbook of Big Data Analytics*; Springer: Cham, Switzerland, 2018; pp. 523–538.

20. Wang, S.; Minku, L.L.; Yao, X. A systematic study of online class imbalance learning with concept drift. *IEEE Trans. Neural Netw. Learn. Syst.* 2018, 29, 1–20. [CrossRef] [PubMed]

21. Gama, J.; Medas, P.; Castillo, G.; Rodrigues, P. Learning with Drift detection. In Proceedings of the Brazilian Symposium on Advances in Artificial Intelligence-Sbia, Sao Luis, Brazil, 29 September–1 October 2004.

22. Gomes, H.M.; Bifet, A.; Read, J.; Barddal, J.P.; Enembreck, F.; Pfahringer, B.; Holmes, G.; Abdessalem, T. Adaptive random forests for evolving data stream classification. *Mach. Learn.* 2017, 106, 1469–1495. [CrossRef]

23. Bifet, A.; Holmes, G.; Pfahringer, B. Leveraging bagging for evolving data streams. In Proceedings of the European Conference on Machine Learning & Knowledge Discovery in Databases, Barcelona, Spain, 20–24 September 2010.

24. Wang, S.; Minku, L.L.; Yao, X. Resampling-based ensemble methods for online class imbalance learning. *IEEE Trans. Knowl. Data Eng.* 2015, 27, 1356–1368. [CrossRef]

25. Oza, N.C. Online bagging and boosting. In Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 10–12 October 2005; Volume 2343, pp. 2340–2345.

26. Ryan, E.; Robi, P. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. Neural Netw. Learn. Syst.* 2011, 22, 1517–1531.

27. Bifet, A.; Holmes, G.; Pfahringer, B. Moa: Massive online analysis. *J. Mach. Learn. Res.* 2010, 11, 1601–1604.

28. Luque, A.; Carrasco, A.; Martín, A.; Lama, J.R. Exploring symmetry of binary classification performance metrics. *Symmetry* 2019, 11, 47. [CrossRef]

29. Brzezinski, D.; Stefanowski, J. Prequential auc: Properties of the area under the roc curve for data streams with concept drift. *Knowl. Inf. Syst.* 2017, 52, 531–562. [CrossRef]
30. Brzezinski, D.; Stefanowski, J. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Inf. Sci.* **2014**, *265*, 50–67. [CrossRef]

31. Shan, L.; Mao, X.L. *Modeling and Application of Consumer Credit Score in Internet Finance Times*, 1st ed.; Electronic Industry Press: Beijing, China, 2015.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).