



Prioritized Binary Transformation Method for Efficient Multi-label Classification of Data Streams with Many Labels

Onur Yildirim
o.yildirim@bilkent.edu.tr
Bilkent University
Ankara, Turkey

Sepehr Bakhshi
sepehr.bakhshi@bilkent.edu.tr
Bilkent University
Ankara, Turkey

Fazli Can
canf@cs.bilkent.edu.tr
Bilkent University
Ankara, Turkey

Abstract

Real-time data processing systems generate huge amounts of data that need to be classified. The volume, variety, velocity, and veracity (uncertainty) of this data necessitate new approaches and the adaptation of existing classification methods. Moreover, the arriving data can belong to more than one class at the same time. As the number of labels grows larger, a significant portion of the multi-label data stream classification methods become computationally inefficient. We propose a novel online approach: the Prioritized Binary Transformation (PBT) method, which can classify data with large numbers of labels by ordering the labels using Principal Component Analysis (PCA) within a fixed-size window. This order is then used to transform the label vectors for classification. We perform an empirical analysis on 12 datasets and compare PBT to four prominent baselines using four evaluation metrics. PBT achieves the best average ranking in three of the four evaluation metrics. Moreover, we investigate efficiency under average execution time per data item and memory consumption where PBT achieves second and first average rankings, respectively.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Information systems** → **Data stream mining**.

Keywords

Data stream; multi-label classification; problem transformation

ACM Reference Format:

Onur Yildirim, Sepehr Bakhshi, and Fazli Can. 2024. Prioritized Binary Transformation Method for Efficient Multi-label Classification of Data Streams with Many Labels. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*, October 21–25, 2024, Boise, ID, USA. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3627673.3679980>

1 Introduction and Related Work

The vast majority of data mining tasks focus on static data; however, social media platforms, networks, and IoT devices generate very large online data [13, 24]. Data streams carry an enormous amount of information, and instances arrive at high speed. In such an environment, every data instance has to be classified using limited

resources. Therefore, the need for new algorithms and adaptation of existing algorithms arises to keep up with the speed, memory requirements, and changes in the data distribution [9, 14].

A great portion of the data stream classification research focuses on single-labeled data where each data instance is associated with only one label in its label set [1, 21]. On the other hand, multi-label classification methods aim to classify data instances that are associated with one or more labels [2, 19]. For instance, an image can contain a person, a sea, and the sun in the background. However, considering more labels leads to inefficiency which is a major problem in multi-label classification in both static [3, 20] and online settings [15], especially when a large number of labels is considered.

Multi-label classification methods are divided into two categories: *Problem Transformation (PT)*, and *Algorithm Adaptation* [5, 21]. *PT* methods transform a multi-label classification problem into single-label problems [5, 21]. The widely used *PT* methods are Binary Relevance (BR), Classifier Chains (CC), and Label Powerset (LP).

BR is an intuitive approach that trains one classifier per label [23]. However, it is criticized for missing the correlation between labels as it assumes the labels are independent [5, 11, 16, 23]. CC is a variation of BR that aims to capture associations between class labels. To achieve this, it manipulates the feature space by adding the ground truth of the previous class during training [16]. LP turns a multi-label classification problem into a multi-class classification problem by assigning a unique class to every label combination in the training data [18].

A recently proposed *PT* method named Binary Transformation (BT) [10], employs a linear regressor to transform the labels. As stated in [10], the main problem of this method is the loss of information as the number of labels (L) grows larger due to the compression method employed.

BR and CC exhibit linear complexity with respect to the number of labels, whereas LP demonstrates exponential complexity [17]. BT is superior to other *PT* methods (BR, CC, and LP) in terms of efficiency.

In this study, we propose the Prioritized Binary Transformation (PBT) method that prioritizes the labels according to their principal components using Principal Component Analysis (PCA). We store n label vectors where n is a user-defined parameter and perform PCA on them to obtain the principal components. The importance of a variable (label in our case) in a principal component model is indicated by the size of its residual variance [22]. Therefore, we order labels according to their importance and mitigate the main issue of BT which is the loss of information on important classes in large numbers of labels while maintaining efficiency. Table 1 shows the notations we use throughout this paper.



This work is licensed under a Creative Commons Attribution International 4.0 License.

CIKM '24, October 21–25, 2024, Boise, ID, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0436-9/24/10
<https://doi.org/10.1145/3627673.3679980>

Our contributions are (i) PBT, an online multi-label classifier that can handle large numbers of labels while preserving efficiency, which is a major problem in multi-label classification, (ii) a comparative analysis on 12 datasets using four baseline methods, (iii) the source code of PBT and our experimental setup.

Table 1: Symbols and notations for PBT and multi-label classification.

Symbol	Meaning
D	Data stream
M	Number of attributes in a data instance
L	Number of possible labels in a data instance
N	Number of data instances in the data stream
$LC(D)$	Label cardinality is the average number of true labels in D
$LD(D)$	The label density in D calculated as: $LD(D) = \frac{LC(D)}{L}$
Y	$n \times L$ label matrix used in PCA
PC	Principal component indices. $PC = \langle PC_1, PC_2, \dots, PC_L \rangle$
C	The integer value of a label vector
\hat{C}	The predicted integer value from the regressor for a feature vector
x	A data instance. $x = \langle x_1, x_2, \dots, x_M \rangle$
y	Ground truth vector. $y = \langle y_1, y_2, \dots, y_L \rangle = \{0, 1\}^L$
\hat{y}	Predicted vector. $\hat{y} = \langle \hat{y}_1, \hat{y}_2, \dots, \hat{y}_L \rangle = \{0, 1\}^L$
y_o	Ordered y according to PC . $y_o = \langle y_{o_1}, y_{o_2}, \dots, y_{o_L} \rangle = \{0, 1\}^L$
\hat{y}_o	Ordered \hat{y} according to PC . $\hat{y}_o = \langle \hat{y}_{o_1}, \hat{y}_{o_2}, \dots, \hat{y}_{o_L} \rangle = \{0, 1\}^L$
n	User-defined parameter for the size of Y , the label matrix used in PCA
t	Current time stamp

2 Prioritized Binary Transformation

Binary Transformation (BT) [10] is an efficient PT method that employs a linear regressor. It converts each y into an integer C by the Equation 1 which turns the problem into a multi-class problem. Then it trains the linear regressor with (x, C) .

$$C = \sum_{i=1}^L y_i \times 2^{L-i} \quad (1)$$

In the prediction phase for an x , the \hat{C} is obtained from the regressor and the \hat{y} is obtained by solving the equation:

$$\hat{C} = \hat{y}_1 \times 2^0 + \hat{y}_2 \times 2^1 + \dots + \hat{y}_L \times 2^{L-1} \quad (2)$$

BT can preserve common label information, e.g., $\langle 1, 0, 1 \rangle = 5$ and $\langle 1, 1, 0 \rangle = 6$, assigning both a close integer (C). However, as L grows, gaps between C values increase, losing information for labels near the least significant bit of y . To mitigate this, we order labels based on principal components, keeping those with higher variance close to each other and the most significant bit.

We propose a method that mitigates the problem of BT: the information loss while working with large values of L . In BT, each y is cumulatively converted into an integer from the least to the most significant bit. Therefore, if a label that creates a high variance is close to the least significant bit, its effect vanishes throughout the conversion process. To address this, We use PCA to find the labels that create the highest variances and order them accordingly. When PBT is initialized, the Y matrix is empty and the PC vector is used as the default order from 1 to L .

The training and prediction steps are given in Algorithm 1. First, the indices in the label vector (y) are ordered according to PC and stored in y_o (line 2). For instance, for the following example where the numbers in parentheses are the label indices for y and

y_o : If $y = \langle 1(1), 0(2), 1(3), 0(4) \rangle$ and $PC = \langle 2, 1, 4, 3 \rangle$ then $y_o = \langle 0(2), 1(1), 0(4), 1(3) \rangle$. By doing so, we keep the labels that create the highest variances close to each other and to the most significant bit. In the next step, y_o is converted into integer C , (line 3), and the linear regressor is trained on the (x, C) pair (line 4). After the linear regressor is trained, the Y matrix is appended with y (line 5) and if the number of instances in Y is equal to the user-defined parameter n , a PCA is performed (line 7). The principal components of the matrix Y are sorted in descending order and the corresponding indices are stored in PC (line 8). This is the new order of labels to be used for the linear regression. For every update on PC , the Y matrix is reset (line 9).

Algorithm 1 Training and prediction processes where $|x| = M$ and $|y| = |y_o| = |\hat{y}| = L$.

```

1: function LEARNONE( $x, y$ )
2:    $y_o \leftarrow$  Swap the values in  $y$  according to  $PC$  and store.
3:    $C \leftarrow \sum_{i=1}^L y_{o_i} \times 2^{L-i}$ 
4:   Train linear regressor with  $(x, C)$ 
5:   Append  $y$  to the matrix  $Y$ .
6:   if Row count of  $Y$  mod  $n = 0$  then
7:     Compute PCA on  $Y$ 
8:      $PC \leftarrow$  The principal components indices
9:     Reset matrix  $Y$ 
10:  end if
11: end function
12:
13: function PREDICTONE( $x$ )
14:   $\hat{C} \leftarrow$  Predict from the linear regressor using  $x$  and round
15:   $\hat{y}_o \leftarrow$  Solve  $\hat{C} = \hat{y}_{o_1} \times 2^0 + \hat{y}_{o_2} \times 2^1 + \dots + \hat{y}_{o_L} \times 2^{L-1}$ 
16:   $\hat{y} \leftarrow$  Swap the values in  $\hat{y}_o$  according to  $PC$  and store
17:  return  $\hat{y}$ 
18: end function

```

To obtain \hat{y} of a data item x , first, the \hat{C} value for x is predicted from the regressor, and rounded (line 14). Then the ordered prediction vector (\hat{y}_o) for the \hat{C} value is obtained by Equation 2 (line 15). However, the label order in this vector is the same as the indices in PC , therefore, we reverse the y_o according to PC and obtain the final prediction vector (line 16). An example of the transformation process in both learning and prediction steps is given in Figure 1.

As stated before, the conversion of a vector to an integer is conducted cumulatively, assigning more weights to the labels close to the most significant bit. Thus, we group the labels that create the highest variances closer to the left side of the vector. Hence, their impact on the final C value is greater in datasets with a large number of labels. The contribution of the ground truth values that are important does not vanish as they have the highest multiplier by being closer to the most significant bit.

3 Experimental Setup

We use 12 datasets¹ that are given in Table 2. To show the performance of PBT on datasets with large numbers of labels, we choose 10 datasets with $L > 100$ and two with smaller L values to show that PBT is also capable of working with small label sets.

The method and experiments are implemented using the River [12] framework and the experiments are conducted on a machine with an Intel Core i7-9750H CPU, 16 GB of RAM, and Windows 11

¹Datasets are available at <http://www.uco.es/kdis/mlresources/>

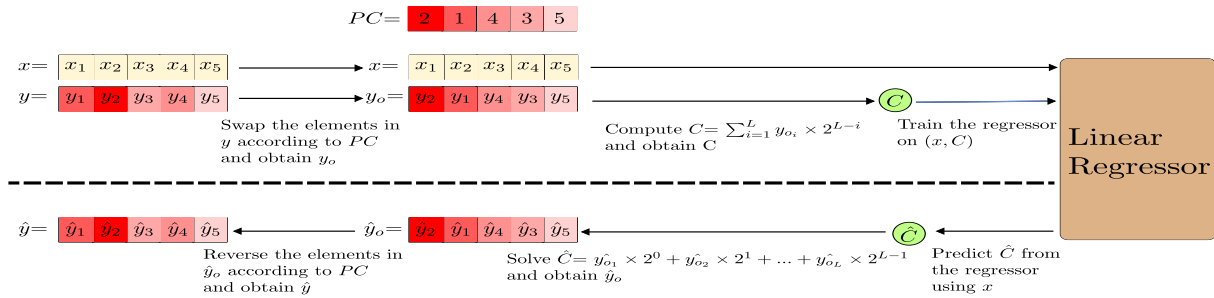


Figure 1: An example of the process of learning using one data pair (x, y) and making prediction from a feature vector x . The color intensities represent the importance of the corresponding labels with the darkest being the most important. The learning and prediction processes are divided by a dashed line, where the upper part is training and the lower part is prediction.

Table 2: Datasets used in the experiments.

Dataset	Type	N	M	L	$LC(D)$	$LD(D)$
20NG	Text	19,300	1,006	20	1.426	0.071
Bibtex	Text	7,395	1,836	159	2.402	0.015
Corel5k	Image	5,000	499	374	3.522	0.009
Corel16k	Image	13,770	500	153	2.859	0.019
Delicious	Text	16,110	500	983	19.020	0.019
Enron	Text	1,702	1,001	53	4.275	0.081
Mediamill	Video	43,907	120	101	4.376	0.043
Reuters	Text	6,000	500	103	1.462	0.014
Stek-chem.	Text	6,971	540	175	2.109	0.012
Stek-cook.	Text	10,490	577	400	2.225	0.006
Stek-cs	Text	9,270	635	274	2.556	0.009
Stek-phil.	Text	3,971	842	233	2.272	0.010

as the operating system. Our experimental setup and PBT implementation are publicly available on GitHub².

We employ the Hoeffding Tree (HT) [7] as the base classifier for all methods. The classification process is done with the interleaved test-then-train approach where the data instance is tested first and then fed into the models for training [8]. For PBT, we conducted an empirical analysis and set the default n value as 1000 based on its average ranking in effectiveness metrics.

4 Results and Discussion

In this section, we first present the effectiveness results, followed by the efficiency results. The effectiveness and efficiency differences of the ranks between the methods are statistically supported with the *Friedman Test with Nemenyi post-hoc analysis* [6]. This test computes a Critical Distance (CD), and classifiers further apart than this value are considered statistically different.

Effectiveness Analysis • The experimental results are given in Table 3. PBT achieves the first average ranking in all evaluation metrics except for the Hamming score. The reason for the poor performance on the Hamming score can be the process of mapping the label vectors into integers which creates a great gap between each class. We also observe a relatively poor ranking on the Hamming score for the BT approach. Although LP performs a similar transformation, the gap between created class IDs is not as significant compared to BT and PBT.

However, the Hamming score is a disputable metric for datasets with large numbers of labels. In [4], it is claimed that in datasets

²Our implementation is available at <https://github.com/o-yildirim/PBT>

with large label sets and low label densities, *high recall* models may have considerably low Hamming Scores due to the nature of the metric. PBT, achieving the first average ranking in Micro-Averaged F1 Score and Example-Based F1 Score but achieving the lowest in Hamming Score supports that claim as high rankings in these two metrics point to a *high recall* model.

The statistical significances of the differences between the methods on all evaluation metrics are given in Figure 2. We observe that PBT is statistically significantly better than CC and BR on every evaluation metric except for the Hamming score. However, BT is not statistically significantly better than any of the methods in any evaluation metric.

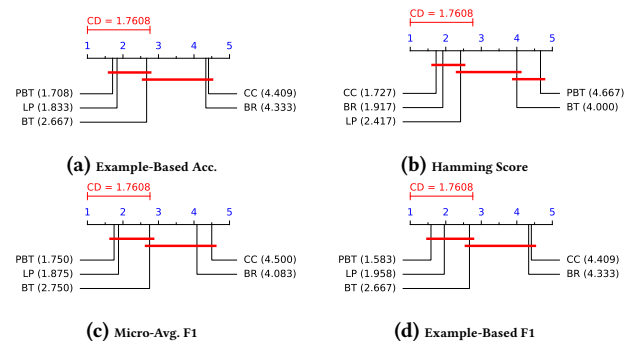


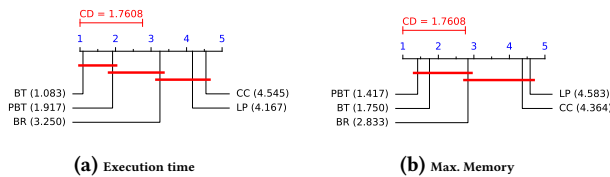
Figure 2: Critical distance diagrams of effectiveness metrics.

Efficiency Analysis • The first efficiency metric we use is the average execution times per data item for datasets given in Table 4. BT achieves the first ranking in execution times and PBT achieves the second as PBT performs additional operations. It is important to emphasize the inefficiency of BR and especially CC and LP for an online environment. As the label count grows, BT and PBT are not affected drastically as BR, CC, and LP are. The inefficiency of LP can be due to the inability of HT to perform with large numbers of labels in multi-class classification. Figure 3.a shows that PBT is statistically faster than CC and LP, and not slower than BT.

Our second efficiency metric is memory consumption. The maximum memory usages of the methods are given in Table 5. PBT achieves the best average rank in terms of memory efficiency. Figure 3.b shows that PBT is statistically significantly more efficient

Table 3: Effectiveness results. Executions that exceed 10,000,000 centiseconds are marked as infeasible (-).

Example-Based Accuracy	PBT	BR	BT	CC	LP	Hamming Score	PBT	BR	BT	CC	LP
20NG	0.201	0.001	0.185	0.048	0.247	20NG	0.905	0.926	0.901	0.931	0.934
Bibtex	0.063	0.000	0.041	0.000	0.001	Bibtex	0.874	0.944	0.926	0.944	0.944
Corel5k	0.025	0.000	0.014	0.000	0.032	Corel5k	0.981	0.991	0.989	0.991	0.982
Corel16k	0.011	0.000	0.011	0.000	0.000	Corel16k	0.965	0.983	0.972	0.984	0.984
Delicious	0.009	0.000	0.001	-	0.040	Delicious	0.934	0.981	0.977	-	0.968
Enron	0.123	0.029	0.098	0.000	0.000	Enron	0.912	0.938	0.914	0.919	0.919
Mediamill	0.209	0.000	0.053	0.000	0.322	Mediamill	0.910	0.955	0.949	0.957	0.958
Reuters	0.034	0.000	0.027	0.000	0.067	Reuters	0.968	0.982	0.954	0.986	0.977
Stck-chem.	0.023	0.000	0.008	0.000	0.106	Stck-chem.	0.961	0.988	0.970	0.988	0.984
Stck-cook.	0.006	0.000	0.001	0.000	0.001	Stck-cook.	0.987	0.994	0.989	0.994	0.994
Stck-cs	0.017	0.000	0.005	0.000	0.078	Stck-cs.	0.978	0.991	0.976	0.991	0.988
Stck-phil.	0.021	0.000	0.003	0.000	0.113	Stck-phil.	0.943	0.990	0.983	0.990	0.988
Average Rank	1.708	4.333	2.667	4.409	1.833	Average Rank	4.667	1.917	4.000	1.727	2.417
Micro-Averaged F1 Score						Example-Based F1 Score					
20NG	0.334	0.014	0.308	0.098	0.401	20NG	0.283	0.001	0.261	0.059	0.309
Bibtex	0.118	0.000	0.091	0.000	0.002	Bibtex	0.178	0.000	0.100	0.000	0.001
Corel5k	0.035	0.000	0.026	0.000	0.043	Corel5k	0.047	0.000	0.023	0.000	0.044
Corel16k	0.027	0.000	0.025	0.000	0.000	Corel16k	0.027	0.000	0.022	0.000	0.000
Delicious	0.032	0.000	0.008	-	0.089	Delicious	0.071	0.000	0.008	-	0.073
Enron	0.411	0.482	0.397	0.000	0.000	Enron	0.178	0.033	0.146	0.000	0.000
Mediamill	0.224	0.003	0.127	0.000	0.429	Mediamill	0.360	0.000	0.095	0.000	0.453
Reuters	0.038	0.006	0.033	0.000	0.059	Reuters	0.044	0.000	0.041	0.000	0.070
Stck-chem.	0.029	0.000	0.022	0.000	0.125	Stck-chem.	0.045	0.000	0.019	0.000	0.138
Stck-cook.	0.011	0.000	0.007	0.000	0.003	Stck-cook.	0.012	0.000	0.003	0.000	0.001
Stck-cs	0.025	0.000	0.015	0.000	0.113	Stck-cs.	0.034	0.000	0.015	0.000	0.113
Stck-phil.	0.020	0.000	0.012	0.000	0.124	Stck-phil.	0.049	0.000	0.009	0.000	0.145
Average Rank	1.750	4.083	2.750	4.500	1.875	Average Rank	1.583	4.333	2.667	4.409	1.958

**Figure 3: Critical distance diagrams of efficiency metrics.****Table 4: Average processing times per data item in centiseconds (cs). The most efficient method is marked in bold. Execution times over 10,000,000 cs are marked as infeasible (-).**

Dataset	PBT	BR	BT	CC	LP
20NG	0.616	10.087	0.572	12.793	71.651
Bibtex	1.168	113.258	1.105	188.260	438.930
Corel5k	0.356	69.167	0.276	162.326	60.346
Corel16k	0.317	30.353	0.293	57.815	182.527
Delicious	0.603	71.006	0.354	-	556.209
Enron	0.587	26.119	0.558	34.888	27.959
Mediamill	1.194	6.219	0.643	14.722	48.088
Reuters	0.335	21.365	0.339	33.329	20.359
Stck-chem.	0.434	37.379	0.368	72.046	77.540
Stck-cook.	0.503	53.648	0.345	221.495	150.072
Stck-cs	0.567	47.882	0.513	140.384	131.953
Stck-phil.	0.697	79.985	0.606	134.541	73.670
Average Rank	1.917	3.250	1.083	4.545	4.167

than LP and CC. As we previously observed, PBT is also statistically faster while being more effective than CC (Figure 3) in three metrics. Thus, our empirical analysis shows that we can consider PBT as superior to CC in an online environment in our experimental setup.

In the effectiveness section, we observed that LP achieves close average rankings to PBT. However, in terms of efficiency, PBT is statistically significantly faster and more memory-efficient than LP. Despite its effectiveness, we consider LP as infeasible for data

streams as it achieves the worst and second worst rankings in memory consumption and average execution time respectively.

Table 5: Maximum memory usage in megabytes (MB). The most efficient method is marked in bold. Execution times over 10,000,000 centiseconds, are marked as infeasible (-).

Dataset	PBT	BR	BT	CC	LP
20NG	60.043	285.765	74.875	406.699	676.060
Bibtex	69.591	2,156.438	93.693	2,360.468	7,932.402
Corel5k	87.184	557.726	85.748	1,246.556	1,154.159
Corel16k	107.300	555.963	111.851	1,261.094	2,911.567
Delicious	719.385	804.454	710.317	-	5,637.395
Enron	8.476	64.428	8.809	122.983	470.116
Mediamill	473.692	340.093	602.011	861.705	716.398
Reuters	40.229	220.750	43.972	304.246	291.160
Stck-chem.	67.798	632.339	63.971	643.224	1,168.019
Stck-cook.	198.540	566.102	199.244	2,714.279	2,659.094
Stck-cs	135.270	655.122	129.356	1,845.530	1,966.291
Stck-phil.	49.852	493.846	53.061	939.771	1,200.911
Average Rank	1.417	2.833	1.750	4.364	4.583

5 Conclusion

In this study, we propose PBT, an efficient online multi-label classifier that can handle data with large numbers of labels. To do this, it performs PCA and arranges the order of the labels based on the variance differences they create. We perform an empirical analysis with five problem transformation methods and 12 datasets using four evaluation metrics. We observe that PBT outperforms the other methods in average ranking on all metrics except for the Hamming score. PBT achieves the best ranking in memory and the second ranking in runtime efficiency. Our future work involves developing a mechanism for PBT to adapt to the changes in the data which is a common scenario in data streams.

References

- [1] ABADIFARD, S., BAKHSHI, S., GHEIBUNI, S., AND CAN, F. DynED: Dynamic ensemble diversification in data stream classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (2023), CIKM '23, Association for Computing Machinery, p. 3707–3711.
- [2] BAKHSHI, S., AND CAN, F. Balancing efficiency vs. effectiveness and providing missing label robustness in multi-label stream classification. *Knowledge-Based Systems* 289 (2024), 111489.
- [3] BI, W., AND KWOK, J. Efficient multi-label classification with many labels. In *International Conference on Machine Learning* (2013), PMLR, pp. 405–413.
- [4] BÜYÜKÇAKIR, A., BONAB, H., AND CAN, F. A novel online stacked ensemble for multi-label stream classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1063–1072.
- [5] CHERMAN, E. A., MONARD, M. C., AND METZ, J. Multi-label problem transformation methods: a case study. *CLEI Electronic Journal* 14, 1 (2011), 4–4.
- [6] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.
- [7] DOMINGOS, P., AND HULTEN, G. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2000), pp. 71–80.
- [8] GAMA, J., SEBASTIAO, R., AND RODRIGUES, P. P. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), pp. 329–338.
- [9] GAMA, J. A., ŽLIOBAITUNDEFINED, I., BIFET, A., PECHENIZKIY, M., AND BOUCHACHIA, A. A survey on concept drift adaptation. *ACM Computing Surveys* 46, 4 (2014).
- [10] GULCAN, E. B., ECEVIT, I. S., AND CAN, F. Binary transformation method for multi-label stream classification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (2022), pp. 3968–3972.
- [11] LUACES, O., DÍEZ, J., BARRANQUERO, J., DEL COZ, J. J., AND BAHAMONDE, A. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence* 1 (2012), 303–313.
- [12] MONTIEL, J., HALFORD, M., MASTELINI, S. M., BOLMIER, G., SOURTY, R., VAYSSE, R., ZOUTINE, A., GOMES, H. M., READ, J., ABDESSALEM, T., ET AL. River: machine learning for streaming data in python. *The Journal of Machine Learning Research* 22, 1 (2021), 4945–4952.
- [13] NGUYEN, H.-L., WOON, Y.-K., AND NG, W.-K. A survey on data stream clustering and classification. *Knowledge and Information Systems* 45 (2015), 535–569.
- [14] RAMÍREZ-GALLEGO, S., KRAWCZYK, B., GARCÍA, S., WOŹNIAK, M., AND HERRERA, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* 239 (2017), 39–57.
- [15] READ, J., BIFET, A., HOLMES, G., AND PFAHRINGER, B. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning* 88 (2012), 243–272.
- [16] READ, J., PFAHRINGER, B., HOLMES, G., AND FRANK, E. Classifier chains for multi-label classification. *Machine Learning* 85 (2011), 333–359.
- [17] SZYMAŃSKI, P., KAJDANOWICZ, T., AND KERSTING, K. How is a data-driven approach better than random choice in label space division for multi-label classification? *Entropy* 18, 8 (2016), 282.
- [18] TIDAKE, V. S., AND SANE, S. S. Multi-label classification: a survey. *International Journal of Engineering and Technology* 7, 4.19 (2018), 1045–1054.
- [19] TSOUMAKAS, G., AND KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)* 3, 3 (2007), 1–13.
- [20] TSOUMAKAS, G., KATAKIS, I., AND VLAHAVAS, I. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)* (2008), vol. 21, pp. 53–59.
- [21] TSOUMAKAS, G., KATAKIS, I., AND VLAHAVAS, I. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook* (2010), 667–685.
- [22] WOLD, S., ESBENSEN, K., AND GELADI, P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1 (1987), 37–52.
- [23] ZHANG, M.-L., LI, Y.-K., LIU, X.-Y., AND GENG, X. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science* 12 (2018), 191–202.
- [24] ZHENG, X., LI, P., CHU, Z., AND HU, X. A survey on multi-label data stream classification. *IEEE Access* 8 (2020), 1249–1275.