A Semi-supervised Learning Application for Hand Posture Classification

Kailiang NAN[1], Shengnan HU[2], Haozhe LUO[3], Patricia WONG[4], and Saeid Pourroostaei Ardakani[5*[0000−0001−8062−6617]]

School of Computer Science, University of Nottingham Ningbo China, 315100
{scykn1[1], scysh1[2], scyhl2[3], smyww1[4], saeid.ardakani[5*]}@nottingham.edu.cn

Abstract. The rapid growth of HCI applications results in increased data size and complexity. For this, advanced machine learning techniques and data analysis solutions are used to prepare and process data patterns. However, the cost of data pre-processing, labelling, and classification can be significantly increased if the dataset is huge, complex, and unlabelled. This paper aims to propose a data pre-processing approach and semi-supervised learning technique to prepare and classify a big Motion Capture Hand Postures dataset. It builds the solutions via Tri-training and Co-Forest techniques and compares them to figure out the best-fitted approach for hand posture classification. According to the results, Co-forest outperforms Tri-training in terms of Accuracy, Precision, recall, and F1-score.

Keywords: Semi-supervised learning · Big data Analysis · Co-forest · Tri-training · hand posture classification.

1 Introduction

Human-computer-Interaction (HCI) data classification is a research field that has attracted computer scientists and researchers during the past decades. For this, Machine Learning (ML) techniques are widely used to learn HCI data patterns and classify the datasets [19]. However, traditional ML approaches are usually unable to offer benefits to modern HCI applications especially if the dataset is huge [4]. This is because the ML models should fit a huge dataset into limited and expensive working memories [7]. It results in increased ML model/memory failures and data processing costs.

Big data analysis applications are increasingly being popular as they have the capacity to automate the processing of huge and complex data [6]. However, ML-enabled Big data solutions usually suffer from the lack of labelled data [8]. This is because manual data labelling is expensive and requires complex data domain knowledge analysis especially if the dataset is big [3]. By this, a combination of unsupervised and supervised learning techniques is used to form Semi-supervised learning (SSL) approaches for data classification and/or prediction. Indeed, SSL refers to supervised learning with additional unlabelled data or unsupervised

learning with labelled data [11]. SSL techniques have the capacity to additionally label the unlabelled data points using the knowledge learned from a small number of labelled data samples. However, the cost of SSL techniques can be sharply increased depending on the dataset size and the ratio of labelled data over unlabelled samples.

This paper aims to propose an SSL-enabled Big data analysis solution to classify and analysis hand postures. A data pre-processing approach is proposed to clean (i.e., missing and noise value removal) and prepare (i.e., feature extraction and data normalisation) a massive Motion Capture Hand Postures dataset [5] with 78095 records. In turn, two advanced SSL techniques inducing *Co-Forest* [20] and *Tri-training* [2] are used on an Apache Spark framework [35] to build a data classification model. The former is a co-training-style Random Forest algorithm, while the latter utilises three classifiers to learn data labels. The results of the two ML algorithms are evaluated and compared to figure out the best-fitted solution. The key contributions of this research are outlined below:

- To deploy a data pre-processing approach for cleaning and preparing a hand posture big dataset.
- To train and test two SSL machine learning approaches for hand posture classification.
- To analyse and evaluate the performance of the SSL solutions and find the best-fitted approach.

This paper is organized as follows: Section 2 reviews self-labelling and SSL techniques and highlights their similarities, differences, and superiorities. Section 3 introduces the research methodology, while Section 4 presents and discusses the experimental results. Section 5 summarises the research's key findings and refers to future works.

2 Literature Review

This section outlines the state-of-the-art literature on self-labelling and SSL techniques. This is not a statistical analysis, however, it surveys the relevant state-of-the-art solutions to highlight their similarities and differences and outline the existing research gaps.

SSL Classification techniques are categorised as transductive and inductive learning classes [10]. The former takes both training and test datasets to train a classification model, while the latter uses only the training dataset to classify the unseen data samples. Self-training is a well-known SSL technique that trains the classifier(s) using the labelled samples to predict the label values for the unlabelled samples [9]. However, limited or low-quality labelled data results in an inaccurate self-training prediction and consequently false labelling and misclassification [22].

SSL approaches can be used in HCI data classification/prediction [16], Computer Vision [18], and Natural Language Processing [17]. They can be deployed on large-scale data analysis frameworks (e.g., Apache Spark) to build parallel

classification models and process big and complex datasets [13], [14], [15]. It offers distributed processing, fault tolerance, and scalability benefits [27]. However, the cost of SSL approaches is still an existing drawback in this field of research especially if the dataset is big and complex. For example, graph-based semi-supervised learning takes cubic time complexity $O(n^3)$ [12].

There are two advanced and widely used SSL algorithms including Tri-training and Co-Forest [20]. They are commonly used in semi-supervised classification due to their performance and accuracy [25]. Tri-Training aims to train three classifiers for labelling the unlabelled samples [2]. The classifiers are refined, and the final label prediction is made via a majority voting technique [24]. Co-Forest is well-known ensemble learning based on a Random Forest technique. It amplifies the power of ensemble modelling and extends the Tri-training technique with additional classifiers to achieve a better result. Ensemble learning is a method that uses multiple machine learning algorithms and builds several classifiers instead of a single classifier [20]. Co-Forest requires neither a dataset with rich attributes/features nor a cross-validation analysis to pick up high-confidence unlabeled samples. Hence, it has the capacity to offer real-time classification applications such as financial anomaly detection or speech pattern recognition.

This literature review summarises the key factors of SSL techniques. However, it is still required to investigate the performance of the SSL techniques once they are used to classify big and complex HCI datasets. For this, SSL approaches need to be adapted to support Big data analysis and parallel data processing, and model training. This paper aims to build, tune and parallelise two SSL approaches on Apache Spark and test and evaluate their performance for a data-driven hand posture classification application.

3 Methodology and Implementation

This section presents the research methodology that focuses on proposing a data pre-processing approach to clean and prepare a big hand posture dataset, and Tri-training and Co-Forest SSL deployment to predict and classify the dataset samples.

3.1 Dataset Selection and Pre-processing

The Motion Capture Hand Postures dataset [5] by Louisiana Tech University is used in this research to build and evaluate the SSL models. It contains 5 static hand postures (78095 records) including 1) Fist with thumb out, 2) Stop (hand flat), 3) Point1 (using pointer finger), 4) Point2 (using both pointer and middle fingers), and 5) Grab (fingers curled to grab). The postures are captured via a Vicon motion capture camera system with markers attached to a left-handed glove. Table 1 outlines the dataset attributes and descriptions, while Figure 1 shows these five postures including fist, stop, pointing with one finger, pointing with two fingers, and grab (fingers curled).

Table 1. Dataset Attributes and Description

Attribute	Description	Datatype
Class	Class value (1 to 5)	integer
User	User ID	integer
Xi	x-coord of the i-th marker position	double
Yi	y-coord of the i-th marker position	double
Zi	z-coord of the i-th marker position	double

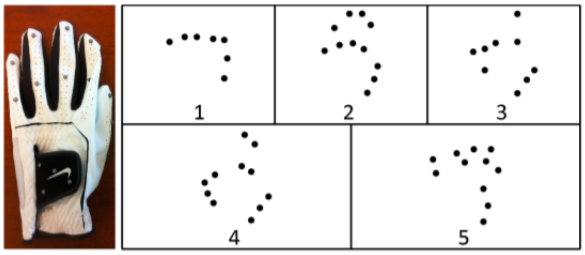

Fig. 1. The data collection glove and hand postures [5].

Data preprocessing is built via Apache Spark DataFrame API (e.g., select, filter, join, and aggregate functions) and MLlib [31] and [30]. Apache Spark provides MLlib (Machine Learning library) which is a scalable library to build machine learning models [28]. It allows large-scale machine learning settings to benefit from model/data parallelism and build a scalable machine learning infrastructure [29]. The Spark-CSV package is used to read the CSV file and convert them into Spark DataFrame. In addition, SparkSQL operations are used to uncover the data characteristics. According to Table 2, the dataset has a balanced class labelling distribution. It is key to include balanced datasets in ML analysis as imbalanced datasets lead to misclassification especially if the dataset is big.

Data normalization and data cleaning techniques are used to prepare the dataset. *StandardScaler* library on Spark is used to normalize data features, while two approaches are used to handle missing data including 1) removing columns with more than 80% of missing values and 2) mean imputation (MI) technique to impute the missing data. According to Bennett [32], missing values can likely result in statistical bias. For this, a data cleaning approach is used to remove columns with 80% of missing values. Then, feature selection and missing value imputation techniques are used to replace missing values with substituted ones.

3.2 Tri-training

Tri-training uses three classifiers which not only allows for easy handling of the labelled confidence estimation problem and the prediction of unlabelled data

Table 2. The Distribution of Dataset Class Labelling

Class	Data Samples
1	16265
2	14978
3	16344
4	14775
5	15733

samples but also improves generalization via ensemble learning. According to the original Tri-training algorithm, an unlabelled sample gives high labelling confidence if it has the same classification results using two classifiers. In turn, the third classifier gives a label value and adds the unlabelled data sample to the labelled set. A voting method is used to integrate the results of the three classifiers and achieve a better and faster classification [2].

Tri-training approach is built via $RandomForestClassifier()$ which forms three Random Forest classifiers. The training dataset is randomly partitioned as labelled (L_train) and unlabelled (U_train) samples using randomSplit() function. Data features are stored as a vector DataFrame (label: Double, features: Vector) using Spark's $VectorAssembler$ transformer, and the columns are assembled into a single vector called 'features'. The Tri-training algorithm takes samples from the L_train dataset using a bootstrap approach. It forms three sub-datasets as h_i,h_j, and h_k to train the Random Forest classifiers. Each classifier calculates the classification error rate of the hypothesis $e[i]$, which is derived from the combination of the other two classifiers, using the $measure\_error()$ function during a Tri-training round.

The classification error rate is used in this approach to find the best-fitted unlabelled samples that should be added to the labelled set. Classifiers h_j and h_k aim to predict U_train and select data samples with the same predictions as the newly labelled data. Let's assume $e[i]$ is the error rate upper bound of h_j and h_k in the classification round of t. Hence, unlabelled sample$i$ is a labelling candidate if $e[i]$ in round t is less than the previous round ($e[i]_{t-1}$). Equation 1 [2] shows the range of accepting an unlabelled sample, where $|L_t|$ and $|L_{t-1}|$ are the sample size during round t and $t-1$.

$$0 < \frac{e[i]_t}{e[i]_{t-1}} < \frac{|L_{t-1}|}{|L_t|} < 1 \tag{1}$$

The unlabelled samples (U) are labelled and added to L_train for further processing and updating the model if their $|L_t|$ is larger than $|L_{t-1}|$ and $e[i] \times |L_t|$ is less than $e[i]_{t-1} \times |L_{t-1}|$. Otherwise, a subsampling method is required to randomly select $|L_t - S|$ samples as L_t where $|L_t|$ stays larger than $|L_{t-1}|$ and Equation 2 works. Equation 3 is used to calculate the value of s and ensure that $e[i] \times |L_t|$ is still less than $e[i]_{t-1} \times |L_{t-1}|$ after subsampling. It is repeated for each of the three classifiers until all the high-confidence unlabelled data samples are

selected and no further update is left. Finally, the results of the three classifiers are combined via a voting approach [2].

$$|L_{t-1}| > \frac{e[i]_t}{e[i]_{t-1} - e[i]_t} \tag{2}$$

$$S = \left[\frac{e[i]_{t-1} |L_{t-1}|}{e[i]_t} - 1 \right] \tag{3}$$

Table 3 shows the initialisation parameters in Tri-training.

Table 3. Tri-training Training: Parameter Initialization

label_ratio	0.05
numTrees	2
maxDepth	12
bootstrap	False
seed	99999
max_iter	10000

3.3 Co-Forest

Co-Forest is an enhanced version of co-training that builds an ensemble model to classify unlabelled samples. It utilises diverse learners aiming to add unlabelled data to the labelled dataset when the voting ratio for a certain label is larger than a predefined threshold. Indeed, Co-Forest selects new unlabelled data for each separated learner based on the ratio of voting in an ensemble model and excludes the learners that use the data to train the new classifiers.

The dataset is automatically partitioned into several parts according to the number of available working nodes/threads on the Apache Spark framework [33]. However, it can be manually partitioned using *repartition* or *partitionBy* methods which are available on Spark DataFrame/RDD [21]. In turn, *VectorAssembler* and *StringIndexer* methods are used to form training and testing datasets. To support inductive learning, the training set is further divided into the labelled set and unlabelled set, while the testing set is used to evaluate the proposed model.

To deploy Co-Forrest, first, the model's parameters including error rates, the weighted sums of confidence, and base classifiers are initialised. Then, the algorithm measures the error rate of a classifier (C_i) for the labelled data. It uses the *sample* method in Apache Spark to subsample some unlabelled data if the error rate is lower than the previous classifier. However, it terminates the model training if it gives a larger error rate than the previously trained classifier. In turn, the classifiers aim to predict the labels based on an ensemble learning fashion. By this, the unlabelled data are labelled using the classifiers'

voting if the labelling confidence is greater than the predefined threshold. The labelling confidence is measured as the number of voter classifiers over the total number of all available classifiers. Finally, the classifier C_i is trained using the new dataset and it is repeated until no classifier needs an update. Table 4 shows the Co-Forest's initialisation parameters.

Table 4. Co-forest Training: Parameter Initialization

num of classifiers	6
seed	99999
threshold	0.75
max_iter	10000

4 Results and Discussions

This section presents and discusses the experimental results to evaluate the performance of the proposed approach for hand posture classification.

4.1 Evaluation Metrics

This research uses Accuracy, F1_score, Precision, and Recall metrics to evaluate the performance of the SSL techniques including Co-Forest and Tri-training [34].

Accuracy is the number of correct predictions according to all the predictions. It is computed using Equation 4, in which True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), are respectively true prediction as positive, false prediction as positive, true prediction as negative, and false prediction as negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision is measured as the proportion of true positive samples over all positive predictions using Equation 5.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall refers to the proportion of correctly predicted positives over all positive labels. It is calculated using Equation 6.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1_score uses Equation 7 to take both precision and recall into account and study the performance of the prediction model.

$$F1\_Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7}$$

Tri-training Results Table 5 shows the performance of a basic Tri-training model which is trained using the initial parameters (Table 3) and according to variously labelled data ratio. The labelled data ratio changes from 0.05% to 95%. The accuracy of the model is increased when the ratio of the labelled data is increased. However, the model's performance is reduced due to the model over-fitting when the labelled data ratio is too high. The Tri-training model is tuned based on tree depth ($maxDepth$) and the number of trees ($numTrees$).

Table 5. Tri-training Performance Based on Labelled Data Ratio

Labelled ratio	**0.05(base)**	**0.10**	**0.20**	**0.30**	**0.50**	**0.60**	**0.95**
Accuracy	0.7269	0.7846	0.8070	0.8254	0.8398	0.8605	0.8578
f1_score	0.7264	0.7849	0.8067	0.8257	0.8394	0.8604	0.8577
precision	0.7270	0.7858	0.8074	0.8265	0.8399	0.8612	0.8585
recall	0.7269	0.7846	0.8070	0.8254	0.8398	0.8605	0.8578

Table 6 highlights the performance of Tri-training according to the tree depth parameter. For this, a base model ($RandomForestClassifier$) is built based on seven different tree depths ($maxDepth$) to find the best-fitted model. According to the results, the performance of Tri-training is increased when tree depth is increased. This stems from the fact that deeper trees build more accurate classifiers. However, the results show over-fitting and accuracy reduction when the tree depth is greater than 12.

Table 6. Tri-training Performance Based on the Depth of Trees

Tree depth	**2**	**4**	**8**	**10**	**12 (base)**	**16**	**20**
Accuracy	0.4237	0.5373	0.7139	0.7408	0.7987	0.7649	0.7423
f1_score	0.3346	0.5289	0.7127	0.7402	0.7982	0.7585	0.7302
precision	0.3403	0.5255	0.7127	0.7408	0.7994	0.7918	0.7878
recall	0.4237	0.5373	0.7139	0.7408	0.7987	0.7649	0.7423

The performance of Tri-training model is studied if the number of trees ($numTrees$) changes. Table 7 shows the model performance according to five scenarios. According to it, a better Tri-training performance is achieved when

the number of trees is increased. Hence, the best Tri-training model works with a labeling ratio of 0.05, and 30 decision trees each of which in the depth of 12.

Table 7. Tri-training Performance Based on the Number of Trees

Number of trees	**2**	**4**	**10**	**20**	**30**
Accuracy	0.7625	0.8144	0.8508	0.8569	0.8604
f1_score	0.7600	0.8145	0.8507	0.8559	0.8601
precision	0.7632	0.8167	0.8515	0.8587	0.8608
recall	0.7627	0.8144	0.8508	0.8564	0.8605

Co-forest Results Table 8 shows the performance of the proposed Co-Forest algorithm according to variously labelled data ratio. According to it, the accuracy of Co-Forest algorithm is enhanced as the ratio of labelled data is increased. SSL uses the labelled dataset to predict new labelled data. Hence, it enhances the model performance if a greater number of labelled data are included in model training. However, the model accuracy drops when the labelled data ratio reaches 95%. It is because of the model over-fitting. According to Tables 8 and 5 Co-Forest outperforms Tri-training especially if the labelled data ratio is low.

Table 8. Co-Forest Performance Based on Labelled Data Ratio

Labelled ratio	**0.05(base)**	**0.10**	**0.20**	**0.30**	**0.50**	**0.60**	**0.95**
Accuracy	0.7987	0.8339	0.8583	0.8757	0.8793	0.8864	0.8761
f1_score	0.7981	0.8332	0.8580	0.8751	0.8789	0.8860	0.8760
precision	0.7993	0.8354	0.8614	0.8790	0.8812	0.8884	0.8787
recall	0.7987	0.8339	0.8583	0.8757	0.8794	0.8865	0.8761

According to Table 9, Co-Forest gives better performance if the tree depth is increased. It is because deeper trees build more accurate base classifiers to accurately classify the data. According to it, the best results are achieved if the tree depth is 16. However, it leads to over-fitting and accuracy reduction if the tree depth is increased to greater than 16.

The performance of Co-Forest algorithm is analysed based on the number of trees/classifiers. As Table 10 shows, the model's accuracy is increased when the classifiers increase until having 6 trees. However, the accuracy drops when the number of base classifiers goes behind 6. Hence, the best-fitted Co-Forest model should be trained with a 5% labelled data ratio, 6 classifiers, and a tree depth of 20.

Table 9. Co-Forest Performance Based on the Depth of Trees

Tree depth	**2**	**4**	**8**	**12 (base)**	**16**	**20**
Accuracy	0.4323	0.5683	0.7060	0.7987	0.8217	0.8092
F1_score	0.3714	0.5680	0.7080	0.7982	0.8216	0.8093
precision	0.3380	0.5791	0.7176	0.7994	0.8232	0.8100
recall	0.4323	0.5683	0.7060	0.7987	0.8217	0.8092

Table 10. Co-Forest Performance Based on the Number of Trees

Number of trees	**2**	**4**	**6 (base)**	**8**	**10**	**12**
Accuracy	0.6684	0.7716	0.7987	0.7975	0.7823	0.7761
F1_score	0.6641	0.7707	0.7982	0.7967	0.7805	0.7761
precision	0.6768	0.7717	0.7994	0.7986	0.7840	0.7786
recall	0.6684	0.7716	0.7987	0.7975	0.7823	0.7761

5 Conclusion and future work

Large-scale hand posture data classification needs to take benefit of Big data-enabled machine learning techniques and Semi-supervised Learning approaches. They offer benefits as compared to classic supervised learning due to the cost of hand posture data labelling in big and complex datasets.

This research aims to propose a data pre-processing approach and build two advanced semi-supervised learning algorithms including Tri-training and Co-Forest for a classification problem. It uses Spark MLlib to support model training parallelism. The semi-supervised learning approaches are evaluated and tuned via an extensive experimental plan to find the best-fitted models according to the given dataset. According to the results, Co-Forest model outperforms Tri-training in terms of Accuracy, F1-score, Precision, and Recall.

The performance of the semi-supervised learning on the Apache Spark framework is improved if a Federated Learning method is used. Federated Learning provides a distributed framework for collaborative model training and has the capacity to offer benefits -mainly model training parallelism with minimised data sharing/leakage. It results in reduced model training delay, especially in Big data applications.

The performance of the SSL approaches can be improved if a multicomputer platform. This allows true parallelism resulting in enhanced performance and more accurate outputs. However, this research utilises a hyper-threading method to build and test the proposed algorithms due to the cost and lack of resources. Hyper-threading is unable to simultaneously run threads due to the restriction of CPU scheduling algorithm and computing resources (e.g., number of CPU cores).

References

1. Chau, V. T. N., and Phung, N. H. (2018, July). Combining self-training and tri-training for course-level student classification. In 2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST) (pp. 1-4). IEEE.
2. Zhou, Z. H., and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transactions on knowledge and Data Engineering, 17(11), 1529-1541.
3. Triguero, I., García, S., and Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowledge and Information systems, 42(2), 245-284.
4. Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. Neurocomputing, 237, 350-361.
5. A. Gardner, C. A. Duncan, J. Kanno, and R. Selmic. '3D hand posture recognition from small unlabeled point sets,' in 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC), Oct 2014, pp. 164-169.
6. Elgendy, N., and Elragal, A. (2014, July). Big data analytics: a literature review paper. In Industrial conference on data mining (pp. 214-227). Springer, cham.
7. L'heureux, A., Grolinger, K., Elyamany, H. F., and Capretz, M. A. (2017). Machine learning with big data: Challenges and approaches. Ieee Access, 5, 7776-7797.
8. Chawla, N. V., and Karakoulas, G. (2005). Learning from labeled and unlabeled data: An empirical study across techniques and domains. Journal of Artificial Intelligence Research, 23, 331-366.
9. Zhu, X., and Goldberg, A. B. (2009). Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 3(1), 1-130.
10. Chen, K., and Wang, S. (2010). Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1), 129-143.
11. Reddy, Y. C. A. P., Viswanath, P., and Reddy, B. E. (2018). Semi-supervised learning: A brief review. Int. J. Eng. Technol, 7(1.8), 81.
12. Sawant, S. S., and Prabukumar, M. (2020). A review on graph-based semi-supervised learning methods for hyperspectral image classification. The Egyptian Journal of Remote Sensing and Space Science, 23(2), 243-248.
13. Kacheria, A. (2021). Semi-Supervised Learning Algorithm for Large Datasets Using Spark Environment (Doctoral dissertation, University of Cincinnati).
14. Balaanand, M., Karthikeyan, N., Karthik, S., Varatharajan, R., Manogaran, G., and Sivaparthipan, C. B. (2019). An enhanced graph-based semi-supervised learning algorithm to detect fake users on Twitter. The Journal of Supercomputing, 75(9), 6085-6105.
15. Melo-Acosta, G. E., Duitama-Munoz, F., and Arias-Londono, J. D. (2017, August). Fraud detection in big data using supervised and semi-supervised learning techniques. In 2017 IEEE Colombian conference on communications and computing (COLCOM) (pp. 1-6). IEEE.
16. Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models.
17. Riloff, E., Wiebe, J., and Phillips, W. (2005, July). Exploiting subjectivity classification to improve information extraction. In AAAI (pp. 1106-1111).
18. Xia, Y., Liu, F., Yang, D., Cai, J., Yu, L., Zhu, Z., ... and Roth, H. (2020). 3d semi-supervised learning with uncertainty-aware multi-view co-training. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3646-3655).

19. Maeireizo, B., Litman, D., and Hwa, R. (2004, July). Co-training for predicting emotions with spoken dialogue data. In Proceedings of the ACL Interactive Poster and Demonstration Sessions (pp. 202-205).
20. Li, M., and Zhou, Z. H. (2007). Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 37(6), 1088-1098.
21. Aziz, K., Zaidouni, D., and Bellafkih, M. (2019). Leveraging resource management for efficient performance of Apache Spark. Journal of Big Data, 6(1), 1-23.
22. Kostopoulos, G., Kotsiantis, S., and Pintelas, P. (2015, October). Estimating student dropout in distance higher education using semi-supervised techniques. In Proceedings of the 19th Panhellenic Conference on Informatics (pp. 38-43).
23. Blum, A., and Mitchell, T. (1998, July). Combining labeled and unlabeled data with co-training. In Proceedings of the eleventh annual conference on Computational learning theory (pp. 92-100).
24. Farouk Abdel Hady, M., and Schwenker, F. (2010). Combining committee-based semi-supervised learning and active learning. Journal of Computer Science and Technology, 25(4), 681-698.
25. Li, K., Zhang, W., Ma, X., Cao, Z., and Zhang, C. (2008, December). A novel semi-supervised SVM based on Tri-training. In 2008 Second International Symposium on Intelligent Information Technology Application (Vol. 3, pp. 47-51). IEEE.
26. Zhang, M., Tang, J., Zhang, X., and Xue, X. (2014, July). Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval (pp. 73-82).
27. Penchikala, S. (2018). Big data processing with apache spark. Lulu. com.
28. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... and Talwalkar, A. (2016). Mllib: Machine learning in apache spark. The Journal of Machine Learning Research, 17(1), 1235-1241.
29. Armbrust, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., ... and Zaharia, M. (2015). Scaling spark in the real world: performance and usability. Proceedings of the VLDB Endowment, 8(12), 1840-1843.
30. López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information sciences, 250, 113-141.
31. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., and Herrera, F. (2016). Big data preprocessing: methods and prospects. Big Data Analytics, 1(1), 1-22.
32. Bennett, D. A. (2001). How can I deal with missing data in my study?. Australian and New Zealand journal of public health, 25(5), 464-469.
33. Zhang, J., Yang, Z., and Benslimane, Y. (2019, July). Exploring and Evaluating the Scalability and Eficinecy of Apache Spark Using Educational Datasets. In 2019 International Conference on Machine Learning and Cybernetics (ICMLC) (pp. 1-6). IEEE.
34. Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.
35. Spark (2022). Apache Spark {https://spark.apache.org/} retrieved August 2022