A Comparison of Machine Learning Methods for Extremely Unbalanced Industrial Quality Data

Pedro José Pereira[1], Adriana Pereira[2], Paulo Cortez[1], and André Pilastri[3]

[1] ALGORITMI Centre, Dep. Information Systems, University of Minho, Guimarães, Portugal
id6927@alunos.uminho.pt, a67662@alunos.uminho.pt, pcortez@dsi.uminho.pt
[2] Bosch Car Multimedia, Braga, Portugal
adriana.pereira@pt.bosch.com
[3] EPMQ - IT Engineering Maturity and Quality Lab, CCG ZGDV Institute, Guimarães, Portugal
andre.pilastri@ccg.pt

Abstract. The Industry 4.0 revolution is impacting manufacturing companies, which need to adopt more data intelligence processes in order to compete in the markets they operate. In particular, quality control is a key manufacturing process that has been addressed by Machine Learning (ML), aiming to improve productivity (e.g., reduce costs). However, modern industries produce a tiny portion of defective products, which results in extremely unbalanced datasets. In this paper, we analyze recent big data collected from a major automotive assembly manufacturer and related with the quality of eight products. The eight datasets include millions of records but only a tiny percentage of failures (less than 0.07%). To handle such datasets, we perform a two-stage ML comparison study. Firstly, we consider two products and explore four ML algorithms, Random Forest (RF), two Automated ML (AutoML) methods and a deep Autoencoder (AE), and three balancing training strategies, namely None, Synthetic Minority Oversampling Technique (SMOTE) and Gaussian Copula (GC). When considering both classification performance and computational effort, interesting results were obtained by RF. Then, the selected RF was further explored by considering all eight datasets and five balancing methods: None, SMOTE, GC, Random Undersampling (RU) and Tomek Links (TL). Overall, competitive results were achieved by the combination of GC with RF.

Keywords: Anomaly Detection· Industrial Data · Random Forest.

1 Introduction

The Industry 4.0 concept is increasing the pressure of companies to adopt data intelligence processes in order to remain competitive in the markets they operate [12]. In particular, quality control is a crucial manufacturing process that can directly impact on productivity by reducing costs, defective products and complaints, among others [16]. In the past, several studies have explored Machine Learning (ML) algorithms to model quality control [2, 6]. For instance, in

2016 there was a Kaggle challenge that addressed an industrial manufacturing quality prediction by using ML approaches [9, 12, 16].

Usually industrial quality ML prediction is addressed as a binary classification task, which is often a nontrivial task for two main reasons. Firstly, there is typically a lack of failures in modern manufacturing processes, thus the classification task is highly unbalanced [6]. For instance, there can be more than 99% of normal cases. Under such extreme unbalanced distribution, ML algorithms might produce misleading results due to the usage of standard loss functions (e.g., classification accuracy), which do not correctly measure the detection of faulty products. Secondly, industrial quality often involves big data, due to the volume and velocity of the produced data records, which increases the computational effort required by the ML algorithms.

In this paper, we address a relevant industrial manufacturing quality prediction task from a major automotive assembly company. The goal is to reduce the quantity of performed tests while maintaining the product quality, thus reducing inspection times and costs. The analyzed data includes millions of records but is extremely unbalanced, containing less than 0.1% of faulty products. This contrasts with related works, which handled a substantially higher number of failures (from 0.58% to 7%, as shown in Section 2). In particular, we handle eight extremely unbalanced datasets by exploring different ML algorithms and balancing training methods. Using a reduced set of two products, we first compare three supervised learning methods, Random Forest (RF) and two Automated ML (AutoML) approaches [8], and an unsupervised deep learning AutoEncoder (AE). Each ML is tested using three balancing strategies: no balancing (None), Synthetic Minority Oversampling Technique (SMOTE) [3] and Gaussian Copula (GC) [13]. Since RF provided interesting results in terms of both classification performance and computational effort, the RF algorithm was further selected as the base model for the remainder experiments, which considered all eight product datasets and five balancing training strategies: None, SMOTE, GC, Random Undersampling (RU) and Tomek Links (TL) [10].

This paper is organized as follows. The related work is presented in Section 2. Then, Section 3 describes the industrial data, ML methods and evaluation procedure. Next, Section 4 details the obtained results. Finally, Section 5 discusses the main conclusions and the future work.

2 Related Work

Several ML approaches have been proposed for industrial quality prediction tasks, which tends to produce unbalanced datasets. For instance, in [2] a semiconductor manufacturing test was modeled as a binary classification task that contained 7% of failures. In 2016, the "Bosch Production Line Performance" dataset, which included only 0.58% of failures, was made publicly available via a Kaggle competition [9]. Several studies explored this dataset by using the XGBoost algorithm [12, 16]. However, none of the previous works explored training data balancing techniques, such as oversampling, undersampling, SMOTE or

TL [3]. More recently, Fathy et al. [6] also addressed manufacturing quality prediction as binary classification task, exploring a dataset that contained 1.7% of faults. The authors used data augmentation techniques to balance the training data, namely SMOTE and Generative Adversarial Networks (GANs). In terms of ML algorithms, several supervised methods were compared, including Logistic Regression (LR), RF and XGBoost. While interesting results were achieved, no undersampling technique was explored in the comparison. Moreover, only a single dataset was used.

Regarding evaluation metrics, the related works used mostly measures based on class labels, such as: Matthews Correlation Coefficient (MCC) [12,16]; a combination of the True Positive Rate (TPR) and True Negative Rate (TNR) [2]; and F1-Score [6]. However, when class decision scores or probabilities are available, is it possible to compute the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve [7]. The AUC measure provides several advantages over class label metrics [15]: it does not consider a single TPR to TNR trade-off; quality values are not affected if the classification data is unbalanced; and AUC values have an easy human interpretation (e.g., 50% is the performance of a random classifier, while 100% corresponds to a perfect discrimination). However, only one of industrial quality detection study has considered the AUC metric [12].

In this work, we analyse a manufacturing quality prediction task from a major automotive assembly company and that involves a tiny percentage of failure cases (less than 0.1%) that is much smaller than what has been handled in related works. Moreover, in contrast with [6], we handle eight different datasets and compare a larger set of balancing methods (including GC and two undersampling methods, RU and TL). Finally, since we handle big data, we consider both the classification performance (using the AUC metric) and the computational effort (in terms of time elapsed) when evaluating the ML methods, allowing to assess if they are feasible for a real industrial environment deployment.

3 Materials and Methods

3.1 Data

This work was developed within a larger R&D project set within the Industry 4.0 concept and that aims to design an Artificial Intelligence (AI) technological infrastructure to improve the manufacturing processes of a major automotive assembly company. The company provided a total of 8 datasets, each related with a distinct type of steering wheel angle sensor. Due to business privacy issues, the products are here denoted as P01, P02, ..., P08. Each product is assembled during the production line, either by robots, humans or a combination of both. Then, the products are subject to two different types of tests: functional, executed immediately after assembly, and torque, performed after the functional tests in order to measure the amount of torque being applied to an object.

The functional tests return a numeric value that measures a particular physical property. The measurements are compared with an acceptance interval set

(composed of lower and upper bounds). In total, there are 10 functional tests, termed here as F01, F02, ..., F10. If a given product fails any of the functional tests, it is immediately considered as a faulty product and thus it is not evaluated by the final torque testing. Otherwise, the product is subject to a sequence of 4 torque tests, each returning also a numerical output that is compared with an acceptance interval. If any of the torque tests fails, the global quality status of the product is "fail", else it is labelled as "pass" (normal product).

Table 1 summarizes the analyzed data attributes. The ML goal is to predict the overall torque class label ($y \in \{$"fail","pass"$\}$) based on the functional test values, which are used as the inputs of the ML algorithms. Our datasets only include the more challenging records, the products that passed all individual functional tests and have a final torque inspection value (y). A high performing ML method can potentially provide value to the company by reducing the amount of executed torque tests, which results in energy, time and other savings (e.g., torque instrumentation maintenance costs).

Table 1. Description of the industrial quality data attributes.

Attribute	**Description**	**Range [min, max]**
F01	Sensitivity	[0.999, 1.001]
F02	Hysteresis	[0.035, 2.500]
F03	Maximum nonlinearity (clockwise)	[0.100, 1.764]
F04	Minimum nonlinearity (clockwise)	[-1.778, -0.099]
F05	Maximum nonlinearity (anti-clockwise)	[0.090, 1.799]
F06	Minimum nonlinearity (anti-clockwise)	[-1.790, -0.090]
F07	Maximum K (clockwise)	[0.000, 0.188]
F08	Minimum K (clockwise)	[-0.180, 0.000]
F09	Maximum K (anti-clockwise)	[0.000, 0.184]
F10	Minimum K (anti-clockwise)	[-0.188, 0.000]
y	If a product passes a torque test	"pass" or "fail"

Nowadays, modern manufacturing lines produce high volumes of quality products, which results in a tiny fraction of failures. Indeed, our 8 datasets are extremely unbalanced, with the percentage of failures being below 0.1%. Table 2 presents the total number of records and percentage of failures for each product. The data records were collected in the years of 2019 and 2020. Excepting P07, all products have more than 100,000 records, with P06 containing almost 2 million examples. While we handle big data, there is a clear lack of minority class examples, with the percentage of failures ranging from 0.006% to 0.074%.

3.2 Balancing Methods

Data unbalancement can be quite harmful during the learning phase of classification algorithms. A common practice to solve this issue is apply data balancing

Table 2. Number of records and percentage of failures for each product.

Product	No. of Records	Failures (%)
P01	610,380	0.013
P02	142,100	0.049
P03	714,816	0.006
P04	287,496	0.014
P05	219,860	0.022
P06	1,823,845	0.011
P07	33,897	0.074
P08	592,124	0.015

techniques to the training data, which can be classified into two main approaches: undersampling and oversampling. The former consists on reducing the number of examples from the majority class, while the latter generates synthetic records from the minority class. Previous studies in smart manufacturing only considered oversampling techniques [6], namely SMOTE and GANs. In this work, we compare both undersampling and oversampling approaches. Given that we work with big data (Table 2), we do not explore the GAN method, since it requires a high computational effort during its training phase. Thus, we adopt faster balancing methods, namely two oversampling methods (SMOTE and GC) and two undersampling techniques (RU and TL). These methods are compared with the simpler no balancing method (None).

Concerning undersampling techniques, RU is quite easy to implement, it consists on randomly selecting only a few examples from the majority class, aiming to achieve balanced data. Given that our datasets have several hundred thousands records but only a few hundreds of failures, we did not to completely balance the data, since this would result in very small training set sizes. Instead, for RU we selected a more reasonable 25% random selection of the majority class records (resulting in a 75% reduction of the normal cases). As for the TL method, it performs a more sophisticated selection of positive examples. TL are pairs of examples from opposite classes that have high proximity, i.e., that are more similar. Such examples are noisy and make it difficult for the classifier to draw a borderline between classes. The TL technique tries to identify and remove the majority class records contained in these pairs, leaving the minority ones untouchable, aiming to create a consistent subset of data, thus, smoothing the modelling phase [10].

In terms of oversampling, SMOTE [4] is a popular data augmentation technique for the minority class. In the past, SMOTE has obtained interesting results for quality prediction data [6]. The synthetic data generation process starts by randomly selecting a minority class sample s_1 and searching for its k nearest neighbours, also belonging to minority class. In this work, we assumed $k = 5$, which is the default SMOTE implementation value, thus the neighbourhood

samples are $s_2, s_3, ..., s_6$. Then, for each pair $(s_1, s_2), (s_1, s_3), ..., (s_1, s_6)$, a synthetic example is generated, considering the line segment that unites them [4]. This technique does not guarantee that generated data is realistic [6], which may be problematic for our datasets context. Considering that a torque test is only performed if all functional tests are within the acceptance intervals, it must be guaranteed that the synthetic data are also set within the same intervals. Therefore, after applying SMOTE for data augmentation, we replace values that are outside these intervals by the interval limit (lower or upper). For instance, considering the acceptance interval $[-1, 1]$ and a synthetic value $v = 1.5$ (>1), after applying our synthetic data treatment, we get $v = 1$. Lastly, GC is a model based on mathematical copula functions that converts all data columns distributions to a standard normal, aiming to remove any bias that might be induced [13]. The GC implementation used on this work allows to define a set of restrictions that must be fulfilled by new generated data to ensure its validity. Thus, unlike SMOTE, it is not necessary to perform any verification after the generation of the data. Instead, we define acceptance intervals for each column and GC guarantees that new data are within these intervals and valid. Both SMOTE and CG were set to generate balanced datasets with 50% of instances for each class.

In terms of implementation, all code was developed using the Python programming language. For SMOTE, RU and TL techniques, we used the *imbalanced-learn* library [11], while *sdv* [13] was used for GC. All methods were implemented with their default parameter values.

3.3 Machine Learning Algorithms

Product quality prediction is often modeled as a supervised learning binary classification task, where the purpose is to know in advance if a given product has enough quality to pass the next production step (e.g., "pass" or "fail"). When the number of failures is low, a popular ML approach is to assume an unsupervised Anomaly Detection (AD), which only uses normal records (thus one-class) during the training phase. In this paper, both one-class and binary classification strategies are compared. It should be noted that balancing methods (such as described in Section 3.2) can only be applied to binary classification, since they required labeled training data (with two or more classes).

Concerning the binary classification algorithms, we consider the RF algorithm and two AutoML implementations, namely H2O (`https://www.h2o.ai/products/h2o-automl/`)and AutoGluon (`https://auto.gluon.ai/`). In 2014, the RF tree ensemble algorithm was ranked favorably when compared with hundreds of classifiers for a large set of classification tasks [5]. As for the H2O and AutoGluon tools, they provided good results in a recent AutoML benchmark study [8]. AutoML automatically compares several algorithms with different parameter combinations, returning the best ML model for a given task. For both AutoML tools, the best ML model is set by randomly splitting the training data into fit (2/3) and validation (1/3) sets. Then, the AUC metric computed on the validation set is used as the selection criterion. While automating the ML algorithm and parameter tuning, the AutoML approach tends to require more

computational resources, since it requires the training of a larger number of ML algorithms. The two AutoML tools were set used with their default configurations, which assumes a a search of the best within the following ML algorithms: H2O – Generalized Linear Model (GLM), RF, Extremely Randomized Trees (XRT), Gradient Boosting Machine (GBM), XGBoost, Deep Learning Neural Network (DLNN) and two Stacked Ensembles; AutoGluon – GBM, CatBoost Boosted Trees, RF, Extra Trees, k-Nearest Neighbors (k-NN), a DLNN and a Stacked Ensemble. All supervised ML methods (RF, H2O and AutoGluon) return a failure class probability ($p_i \in [0, 1]$ for the i-th example) and that is used to compute the ROC curves [7]. When needed, class labels can be defined by using a decision threshold K, where it is considered a failure if $p_i > K$.

As for the one-class learning, we adopted an AE, which is a popular deep Learning architecture for AD [17]. An AE is composed by an encoder, a bottleneck layer (defining the latent space) and then a decoder. The model is trained only with normal data, aiming to generate outputs similar to the inputs. A well trained AE reconstructs normal examples with smaller errors, tending to produce larger reconstruction errors when faced with anomalous situations. After some preliminary experiments, conducted using product P01, the AE was set as fully connected feedforward deep neural network with: 4 hidden layers (each with 8 nodes) that defines the encoder; a bottleneck layer of 4 nodes; and a decoder component that is similar to the encoder. All nodes use the ReLu activation function and each transforming layer is coupled with a batch normalization layer. The AE is trained to minimize the reconstruction error, which was set as the Mean Squared Error (MSE). In each training iteration (epoch), 10% of the training data is randomly used as a validation set, allowing to monitor the reconstruction error and perform an early stopping. The Adam optimizer was used to adjust the AE weights, being stopped if there is no improvement after 25 epochs (early stopping) or after a maximum of 100 epochs. After the model is trained, we use the reconstruction error (MSE_i for the i-th example) to compute the failure probability, where the higher the error, the higher is the anomaly class probability (p_i is computed as the normalized MSE_i values, such that $p_i \in [0, 1]$). Similarly to the supervised learning methods, a threshold K is used to assign class labels.

All ML algorithms were implemented using the Python programming language. For H2O and AutoGluon, we used the `h2o` and `autogluon` libraries, both of them with default parameter values that includes an execution time limit of 1 hour. The RF assumes the `scikit-learn` [14] implementation, which uses a default of 100 trees. Finally, AE was implemented using `tensorflow` [1].

3.4 Evaluation

To evaluate methods, we use the AUC measure of the ROC curve [7]. The ROC represents the discrimination performance of a binary classifier when considering all possible K threshold values, plotting one minus the specificity (x-axis) versus the sensitivity (y-axis). The AUC is computed as $\int_0^1 ROC dK$. We also stored the computational effort, measured in terms of the time elapsed for training (in

s) and predicting one example (in ms). Furthermore, to produce more robust results, for each product we apply five runs of a holdout training and test split, using 67% of the data records (random stratified selection) for training and the remaining 33% examples for testing. The data balancing techniques are applied only to the training data, thus, both validation and test subsets are kept unbalanced. We particularly note that validation sets are only used by the AutoML and AE algorithms. For the AutoML, it is used to set the leaderboard, which contains the best set of models and their hyperparameters. As for the AE, the validation set is used by the early stopping procedure and it only includes normal examples (one-class). All created subsets of data, either by splits or balancing techniques, were stored locally in order to ensure all models were evaluated using the same datasets (e.g., same test sets). All evaluation measures (AUC and computational effort) are aggregated by considering the average of the five runs.

4 Results

The experiments were executed in an Intel Xeon 1.70GHz server. When using oversampling, the amount of records almost duplicates, which increases the execution time. Since five runs are applied for each dataset, it is computationally costly to apply all balancing techniques and ML algorithms to all products. Thus, we conducted an initial comparison study by considering two datasets (P01 and P02) and both SMOTE and GC oversampling techniques, aiming to select a reasonable performing ML algorithm for the remainder comparison scenarios.

Table 3 presents the average results for the first comparison study. For product P01, all models achieved a poor performance, with most AUC values being close to 50% (random classifier). In particular, AutoGluon performed worst on both synthetic data generators, H2O only had a slight AUC improvement when using SMOTE and AE obtained the second worst AUC value. As for the RF, it achieved the highest AUC value on P01 data when using the GC oversampling technique. Regarding P02 data, the AUC results are considerably better for all ML algorithms. Specifically, AutoGluon achieved the best AUC value (83.52%), followed by H2O (82.70%) and RF (81.51%), all using GC as the balancing data technique. AE presented the worst predictive performance on product P02 (65.72%). Overall, when considering both products, RF and AutoGluon obtained similar predictive performances. However, the RF training is much faster than AutoGluon (around ten/sixty times faster). For this reason, we selected RF for the remainder quality prediction experiments.

Table 4 presents the second quality prediction comparison results, which uses RF as the base ML model and explores five different balancing methods over all 8 datasets. An analysis to the table shows that GC is clearly the best data balancing technique, achieving the highest AUC values for 6 of the analyzed 8 products (P01, P02, P03, P04, P05 and P06). On the remainder datasets (P07 and P08), RF obtained the best predictive performance when using RU and SMOTE techniques, respectively. The last five rows of Table 4 show the average performance of each approach when considering all eight products. The average results also

Table 3. First quality prediction comparison results (**bold** denotes best average AUC).

Product	ML method	Balancing Technique	AUC	Train Time (s)	Prediction Time (ms)
P01	AutoGluon	None	57.81	2028	0.022
		GC	54.98	2197	0.069
		SMOTE	50.90	2137	0.016
	H2O	None	56.60	2988	0.007
		GC	53.00	3224	0.070
		SMOTE	56.75	3215	0.010
	RF	None	51.15	45	0.008
		GC	**59.23**	226	0.017
		SMOTE	50.98	242	0.012
	AE	None	53.72	78	0.036
P02	AutoGluon	None	83.32	1931	0.047
		GC	**83.52**	1930	0.084
		SMOTE	82.51	1922	0.033
	H2O	None	78.85	3210	0.021
		GC	82.70	3226	0.085
		SMOTE	80.82	3219	0.039
	RF	None	78.62	5	0.007
		GC	81.51	32	0.016
		SMOTE	79.41	34	0.009
	AE	None	65.72	27	0.034

favor the GC oversampling technique, which produces a positive impact on the AUC values, presenting a difference of 7.19 and 10.08 percentage points when compared with the RU (second best overall balancing method) and no balancing methods (None, the worst overall approach). In terms of the final quality prediction quality, the obtained AUC GC RF results reflect the difficulty of modeling extremely unbalanced datasets. For some products, a very good discrimination was achieved (e.g., 82% for P02 and P07, 73% for P04), but there are products that obtained a much lower AUC values (e.g., 50% for P08, 59% for P01). On average, the GC RF class discrimination performance is reasonable (around 67%). Regarding the training times, and as expected, both oversampling techniques (GC and SMOTE) require a larger computational effort. Nevertheless, the obtained GC RF models can still be achieved within a reasonable computational effort. In effect, it requires around 18 minutes of training effort for the largest dataset, which originally contains 1,8 million records (before the application of the GC method). As for the inference times, the GC RF predictions require 0.017 ms, which means that a trained model can be used to produce real-time industrial product quality predictions.

Table 4. Second quality prediction comparison results (**bold** denotes best average AUC).

Product	Balancing Technique	AUC	Train Time (s)	Prediction Time (ms)
P01	None	51.15	44.79	0.008
	GC	**59.23**	226.19	0.017
	SMOTE	50.98	242.24	0.012
	RU	53.08	9.91	0.009
	TL	50.35	45.73	0.008
P02	None	78.62	5.06	0.007
	GC	**81.51**	31.73	0.016
	SMOTE	79.41	34.08	0.009
	RU	81.22	1.25	0.007
	TL	78.18	5.24	0.007
P03	None	53.13	43.08	0.007
	GC	**62.18**	278.16	0.018
	SMOTE	53.82	341.49	0.010
	RU	54.57	8.78	0.007
	TL	52.47	43.57	0.008
P04	None	50.25	7.48	0.006
	GC	**73.07**	66.96	0.014
	SMOTE	59.45	96.52	0.008
	RU	53.63	1.97	0.008
	TL	51.01	7.65	0.006
P05	None	51.11	9.38	0.006
	GC	**68.53**	69.69	0.016
	SMOTE	55.63	87.34	0.009
	RU	57.97	1.99	0.006
	TL	53.00	9.31	0.006
P06	None	51.66	339.33	0.011
	GC	**61.87**	1104.61	0.029
	SMOTE	53.84	1164.85	0.014
	RU	54.85	52.74	0.011
	TL	51.51	354.12	0.011
P07	None	72.60	0.82	0.006
	GC	81.85	5.92	0.012
	SMOTE	76.53	6.02	0.008
	RU	**82.18**	0.27	0.006
	TL	72.59	0.87	0.006
P08	None	49.43	40.22	0.008
	GC	50.21	248.32	0.018
	SMOTE	**51.31**	225.06	0.011
	RU	50.37	9.64	0.009
	TL	49.41	43.78	0.009
Average	None	57.24	61.27	0.008
	GC	**67.31**	253.95	0.017
	SMOTE	60.12	274.70	0.010
	RU	60.98	10.82	0.008
	TL	57.32	63.78	0.008

The obtained results were shown to the manufacturing company experts, who considered them very positive. In particular, the experts highlighted the GC RF discrimination results that were obtained for three of the analysed products (P02, P04 and P07). Moreover, they confirmed that required computational effort is adequate for a real industrial deployment of the ML models.

5 Conclusions

The Industry 4.0 revolution is transforming manufacturing companies, which are increasingly adopting data intelligence processes in order to remain competitive in the market. In the last years, several works used Machine Learning (ML) to enhance product quality control, which is a key manufacturing element. Currently, modern manufacturing companies tend to have a high quality production, which results in a tiny percentage of failures, thus originating extremely unbalanced data that is challenging for common ML algorithms.

In this paper, we analyze millions of records related with eight products assembled by a major automotive company. Only a tiny fraction (less than 0.07%) correspond to failures. To handle such extremely unbalanced data, we compared four ML algorithms and five balancing techniques. Overall, the best results were achieved by a Gaussian Copula (GC) oversampling technique when adopting a supervised Random Forest (RF) base learner. In particular, a very good class discrimination was achieved for three of the eight analyzed products. Moreover, the GC RF combination requires a computational effort (in terms of training and prediction times) that is feasible for the analyzed domain (e.g., it requires around 18 minutes to process 3.6 million records).

In future work, we intend to explore more datasets by testing the proposed GC RF model over a larger range of products. Also, we plan to deploy the ML algorithms and balancing methods in a real industrial setting, which would allow us to monitor the capability of the ML models through time and assess if they can provide productivity gains (e.g., by reducing the number of torque tests).

Acknowledgments

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n 39479; Funding Reference: POCI-01-0247-FEDER-39479].

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J.,

Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/
2. Adam, A., Chew, L.C., Shapiai, M.I., Lee, W.J., Ibrahim, Z., Khalid, M.: A hybrid artificial neural network-naive bayes for solving imbalanced dataset problems in semiconductor manufacturing test process. In: HIS. pp. 133–138. IEEE (2011)
3. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. **6**(1), 20–29 (2004). https://doi.org/10.1145/1007730.1007735
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
5. Delgado, M.F., Cernadas, E., Barro, S., Amorim, D.G.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014), http://dl.acm.org/citation.cfm?id=2697065
6. Fathy, Y., Jaber, M., Brintrup, A.: Learning with imbalanced data in smart manufacturing: A comparative analysis. IEEE Access **9**, 2734–2757 (2021)
7. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**, 861–874 (2006)
8. Ferreira, L., Pilastri, A., Martins, C.M., Pires, P.M., Cortez, P.: A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. In: Int. Joint Conference on Neural Networks, IJCNN 2021, July. IEEE (2021)
9. Kaggle: Bosch production line performance. https://www.kaggle.com/c/bosch-production-line-performance, accessed: 2021-04-27
10. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: One-sided selection. In: ICML. pp. 179–186. Morgan Kaufmann (1997)
11. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research **18**(17), 1–5 (2017), http://jmlr.org/papers/v18/16-365
12. Mangal, A., Kumar, N.: Using big data to enhance the bosch production line performance: A kaggle challenge. In: IEEE BigData. pp. 2029–2035. IEEE Computer Society (2016)
13. Patki, N., Wedge, R., Veeramachaneni, K.: The synthetic data vault. In: 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada, October 17-19, 2016. pp. 399–410. IEEE (2016). https://doi.org/10.1109/DSAA.2016.49
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
15. Pereira, P.J., Cortez, P., Mendes, R.: Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction. Expert Syst. Appl. **168**, 114287 (2021). https://doi.org/10.1016/j.eswa.2020.114287
16. Zhang, D., Xu, B., Wood, J.: Predict failures in production lines: A two-stage approach with clustering and supervised learning. In: IEEE BigData. pp. 2070–2074. IEEE Computer Society (2016)
17. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017. pp. 665–674. ACM (2017). https://doi.org/10.1145/3097983.3098052