Application of Decision Trees to Detect Process Disruptions in Aluminum Production[*]

Nina Lugovaya[1][0000-0002-2939-0298], Anton Mikhalev[1][0000-0002-8986-5953] and Tatiana Penkova[2][0000-0002-0057-0535]

[1] Siberian Federal University, 26, Kirenskogo str., Krasnoyarsk, 660074, Russia
[2] Institute of computational modelling of the Siberian Branch of the Russian Academy of Sciences, 50/44 Akademgorodok, Krasnoyarsk, 660036, Russia
asmikhalev@yandex.ru

Abstract. This paper considers the task of elaborating the tools that enable early detection of process disruptions in aluminum production using the technology of decision trees. The suggested method to forecast the process disruptions are based on the data on daily average process indicators. The method includes a necessary stage of preliminary processing of inputs and consequent construction of a math model. The study defined the most informative properties, solved the problem of unbalanced data, and compared approaches based on decision trees. The quality metrics revealed the most effective method to solve the set task.

Keywords: Decision Trees, Random Forest, Gradient Boosting, Process Disruptions.

1 Introduction

Aluminum production is a strategically important industry of economy. Competitive performance of this industry is primarily determined by reaching high technical and economic indexes, which, in their own turn, depend on the technology quality control and timely estimation of the technical condition of both separate units and the entire complex of aluminum production as a whole. Process disruptions that occur in the cycle of aluminum production are the main impediment on the way towards reaching the highest technical and economic indexes, regarding both the potline and the whole enterprise. Unfortunately, the causes of ineffective operation of cells are investigated after the event. However, extensive introduction of hardware/software packages to control the production flow allows considerable volumes of monitoring data to be accumulated and used in decision-making. Therefore, it is becoming particularly relevant to develop tools for early detection of process disruptions based on the monitoring data and troubleshooting the causes of reductions in productivity of cells.

[*] Copyright c 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The most common process disruptions in aluminum production include: occurrence of anode effect, formation of coal froth, and distortion of the anode surface relief. The latter is considered to be the gravest disruption [1]: a “spike” is a buildup at the anode bottom of a regular cylindrical or conical shape; “lagging” is a bulge of the anode face of a rectangular shape, or an irregularity that covers up to 50-60% of the anode area; “overglow” is a buildup at the bottom of the anode of an irregular shape (sphere, mushroom, etc.) that is formed around any side of the anode unit. Such process disruptions manifest themselves only in advanced stages when the buildups at the bottom develop a bulge that is embedded in the cathodic metal, which is always accompanied by changes in the process parameters. As a rule, these problems trigger changes in the values of such process parameters. Methods of machine learning as they are applied to the monitoring data will reveal specific interdependencies across the data and, on this basis, allow for the identification of process disruptions in the cell operation.

The identification of process disruptions can be viewed as a binary classification task. The accuracy of its solution depends on the volume and quality of inputs collected during the monitoring stage, selected methods of classification, diagnostic quality criteria, and criticality of the controlled function indicators. The body of the article is structured as follows. Section 2 spells out the classification objective. Section 3 describes the inputs. Section 4 gives a description of the applied input data preprocessing methods. Section 5 elaborates on the applied classification algorithms. Section 6 presents the results of how the classification models operate.

2 Research Objective

The classification task is set as follows. Let us assume that there is a set of objects $X = \{X^{(1)}, \dots, X^{(n)}\}$, each characterized by an m-dimensional vector of attributes $X^{(i)} = (x_1^{(i)}, \dots, x_m^{(i)}), i = \overline{1,n}$. Every object under study is attributed to a certain class $C_j \in Y = \{C_1, \dots, C_k\}, j = \overline{1,k}$. In this case, classification is aimed at the following. It requires a rule (algorithm) to be formulated $a: X \to Y$, so that based on a setpoint value of attributes new unknown objects could be attributed to one of the classes.

As it relates to the problem of early detection of process disruptions based on the monitoring data, the task of classification boils down to dividing the states of a process facility into two classes: operative $Y = 0$ and faulty $Y = 1$ (functioning with errors). The input data samples are used as a basis for an algorithm that must be able to use the set operative indicators of a given facility to diagnose its state with sufficiently high accuracy.

Binary classification tasks normally use the following indicators as their metrics:

– *accuracy* is a relation of all correctly classified objects to the total number of all classified objects:

$$accuracy = \frac{TP+TN}{TP+TN+FP+F} \tag{3}$$

Here, *TP* stands for true-positive results (objects classified as “positive” and that are actually positive, i.e. belong to class $Y = 1$), *TN* stands for true-negative results (objects classified as “negative” and that are actually negative, i.e. belong to class $Y = 0$), *FP* stands for false-positive results (objects classified as “positive” but that are actually negative, i.e. belong to class $Y = 0$), *FN* stands for false-negative results (objects classified as “negative” but that are actually positive, i.e. belong to class $Y = 1$).

In case of imbalance between the classes, regular accuracy is replaced with balanced accuracy:

$$accuracy = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{2}$$

– *precision* is a relation of all objects classified as “positive” and that are actually positive to the total number of objects classified as “positive”:

$$precision = \frac{TP}{TP+FP} \tag{3}$$

Precision characterizes the ability of a given prediction model to correctly classify positive objects in relation to the number of all objects classified as “positive”.

– *recall* is a relation of all objects classified as “positive” and that are actually positive to the total number of actually positive objects:

$$recall = \frac{TP}{TP+FN} \tag{4}$$

Recall characterizes the ability of a given prediction model to correctly classify positive objects from the set of all positive objects combined.

– *F1 score* is a harmonic mean between the values of *precision* and *recall*:

$$F1 = 2 * \frac{precision * recall}{precision + rec} \tag{5}$$

F-score demonstrates how many cases are classified by the model correctly, and how many true items can be classified by the model correctly.

3 Data Description

The software/hardware package aimed at forecasting process disruptions is based on the daily average data collected through monitoring the operation of cell series in potrooms No. 9 and 10 of the Khakas aluminum smelter for the period from 2017 to 2019.

The set of controlled cell operation indicators consists of 40 process parameters, including the following: duration of metal tapping (sec), metal level (cm), electrolyte level (cm), electrolyte temperature (°C), alumina dose (kg), bath chemistry parameters, parameters of the point feeding system for alumina and aluminum fluoride, parameters of adjustment the anode-to-cathode distance, amperage (kA), voltage parameters, back EMF (V), state and service life of cells (month), as well as registered process disruptions: number of anode effects, number of “spikes” and “lagging”.

In this study, the prediction model is developed for one of the process disruptions, namely the anode effect. The input array contains about 300,000 entries.

4 Data Analysis and Processing

The inputs are processed in several stages that include filling missing values of object attributes, identifying and deleting errors, selecting informative attributes, and normalizing their values.

The first stage entails processing incomplete data. At first, attributes with the number of gaps that exceeded the set threshold were deleted (over 50% of entries), then the rest of the data underwent reconstruction of missing values by the EM-algorithm [2]. The biggest number of serially interpolated data is 5. The rest of the entries with missing values were deleted. In addition, the data also underwent a correlation analysis, with collinear attributes removed from the set (one of the attributes was removed when the correlation coefficient exceeded 0.8).

Next, the method of quartiles was used to identify the outliers – the values of removed outliers were replaced with upper or lower quartile values.

The inputs in the samples are unbalanced. The number of entries “with no disruptions” is significantly higher than those “with disruptions”. One way to tackle the issue is to use various sampling strategies [3]: undersampling, oversampling, and the hybrid method that uses both strategies simultaneously. The undersampling technique balances the data by removing samples in majority class. The oversampling technique adds the synthetic data samples to minority class. The hybrid method combines both approaches: one – to create additional samples in minority class, the other – to remove those samples that may lead to overfitting. The study presented in this paper used the hybrid method [3].

The next stage featured selection of the most informative attributes. Presence of uninformative attributes among the samples causes overfitting. Attributes were sorted out by the method of recursive feature elimination (RFE) [4] combined with the random forest algorithm. The RFE method relies on consecutive construction of models, when a new model is built in every cycle, with the least informative features being eliminated. As a result, there was a set of the most significant features that contained 15 parameters ready to be used for detecting process disruptions.

5 Description of Algorithms

Some of the most common methods of machine learning used for building diagnostic models are decision trees [5], ensembles of algorithms [6], artificial neural networks [7], fuzzy logic algorithms [8], etc. in this study, the diagnostic model for detection of process disruptions was built on the basis of decision trees.

A decision tree is a model that presents rules for making decisions in a hierarchical sequence structure [9]. Decision trees consist of roots that carry the conditions to be tested (attributes), and leaves that contain outcomes (one of two classes).

The decision tree model employs the principle of recursive decomposition of object space into subsets. Every node, starting at the root, has an attribute that is selected as a division basis for decomposition of all data into 2 classes. The process runs until the stopping criterion gets activated.

The objects are classified using a decision tree by moving top-down from root to leaf in accordance with the conditions set at each top [5].

A random forest is an algorithm used in machine learning that incorporates an ensemble of decision trees [10]. An ensemble of decision trees is a set of decision trees in which each of the trees is built as per the samples that result from the original ones using the bootstrap technique [11]. The qualification result based on the random forest algorithm is determined through voting (the class that has been forecasted by the biggest number of trees is selected as true).

Boosting in decision trees is an algorithm applied in machine learning that uses an ensemble of decision trees [10]. Unlike the technique employed in the random forest algorithm, boosting combines an ensemble of weak classifiers to convert it into a stronger classifier. The point is that every sequential decision tree is trained based on the data about errors in the previous decision tree (i.e. every sequential decision tree corrects the errors of preceding ones and boosts the quality of the entire ensemble). As of now, the boosting algorithm with decision trees has a number of modifications. One of the most effective of them is gradient boosting [10].

The math models and algorithms to monitor the state of process facilities were developed using Python tools. Python has a large number of libraries for machine learning that employ various classification algorithms, including approaches that feature decision trees.

The current study uses the following methods of classification on the basis of decision trees:

- decision tree (scikit-learn library) [12];
- gradient boosting XGBClassifier (XGboost library) [13];
- gradient boosting CatBoostClassifier (Catboost library) [14];
- boosting in unbalanced data RUSBoostClassifier (imbalanced-learn scikit-learn library) [15];
- random forest in unbalanced data BalancedRandomForestClassifier (imbalanced-learn scikit-learn library) [15].

Hyperparameters in the models were set up with the help of the random search method with cross-validation. Parameters for the models were selected based on the principle of maximum accuracy. Tables 1-5 show optimal values of the main hyperparameters for each model.

Table 1. Hyperparameter setup for the decision tree model.

Parameter name	Description	Value
max_depth	Nodes are extended until they reach the maximum depth of a tree.	8
min_samples_split	The minimum number of samples required for splitting the internal node.	4
min_samples_leaf	The minimum number of samples that must be present in the leaf node. The splitting point at any depth will only count if it accounts for at least the min_samples_leaf value of the learning samples in each of the branches	3

Table 2. Hyperparameter setup for the XGBClassifier model.

Parameter name	Description	Value
n_estimators	Number of gradient boosted trees	100
max_depth	Maximum tree depth for base learners	8
learning_rate	Learning rate	0.1

Table 3. Hyperparameter setup for the CatBoostClassifier model.

Parameter name	Description	Value
iterations	Number of iterations	10
depth	Tree depth	4
loss_function	The loos function is a parameter that measures the model accuracy during training	MultiClass
learning_rate	Learning rate	0.1

Table 4. Hyperparameter setup for the RUSBoostClassifier model.

Parameter name	Description	Value
n_estimators	The maximum number of estimators at which boosting is terminated	100
sampling_strategy	Sampling information to sample the data set	‘majority’
learning_rate	Learning rate shrinks the contribution of each classifier by learning_rate	0.1

Table 5. Hyperparameter setup for the BalancedRandomForestClassifier model.

Parameter name	Description	Value
n_estimators	The number of trees in the forest	100
max_depth	The maximum depth of the tree	8
min_samples_leaf	The minimum number of samples required to be at a leaf node	4
min_samples_split	The minimum number of samples required to split an internal node	4
criterion	The function to measure the quality of a split	'gini'

The method of gradient boosting enables the setup of weighting factors when it comes to dealing with unbalanced data. The scale_pos_weight hyperparameter makes it possible to control the balance among classes by setting up the weights for every class. Class weights were selected on the basis of the metaheuristic algorithm of global optimization of orb-weaving spiders (Araneidae algorithm, AA) [16].

6 Analysis and Comparison of Results

To train the model, the samples were split into the training and testing sets in the ratio 2:1. What came out after the abovementioned metrics had been calculated is quoted for each of the algorithms in Table 6. The best results in the considered metrics were demonstrated by the Catboost algorithm.

Table 6. Results of the comparison between the considered metrics.

	accuracy	precision	recall	F-score
Decision tree	0.89723	0.51379	0.38462	0.43478
XGBClassifier	0.73782	0.39620	0.54193	0.46537
Catboost	0.71852	0.88724	0.87532	0.65116
RUSBoostClassifier	0.86166	0.41861	0.64286	0.50704
BalancedRandomForestClassifier	0.71542	0.22472	0.86957	0.35714

7 Conclusion

The paper presents the results of the study aimed at the development of tools for early detection of process disruptions in aluminum production using the decision tree technology. The suggested model predicts the process disruptions in aluminum production based on the information about the daily average process indicators. The method includes the compulsory stage of preprocessing of inputs and further construction of a math model. The study revealed the most informative attributes, solved the problem of unbalanced data, and compared a number of approaches. The math models and diagnostic algorithms were performed for one of the process disruptions, namely, the anode effect. The best results among the analyzed approaches were shown by the Catboost algorithm. In the future, for more accurate forecasting, process disruptions are planned to be predicted using the ensemble method with multiple learning algorithms.

References

1. Puzanov, I.I., Zavadyak, A.V., Klykov, V.A., Makeev, A.V., Plotnikov, V.N.: Continuous monitoring of information on anode current distribution as means of improving the process of controlling and forecasting process disturbances. J. Sib. Fed. Univ. Eng. technol. **9(6)**. 788–801 (2016). doi: 10.17516/1999-494X-2016-9-6-788-801
2. Pigott, T.D.: A review of methods for missing data. Educational research and evaluation. **7(4)**. 353–383 (2010). doi: 10.1076/edre.7.4.353.8937
3. He, H., Ma, Y. (Eds.): Imbalanced Learning. Foundations, Algorithms, and Applications. Wiley-IEEE Press (2013)
4. Batista, G.E., Bazzan, A.L.C., Monard, M.C.: Balancing Training Data for Automated Annotation of Keywords: a Case Study. In: WOB. pp. 10–18 (2003)
5. Rokach, L., Maimon O.: Data Miningwith Decision Trees. Theory and Applications. London: World Scientific Publishing Co (2008)
6. Zhang, C., Ma, Y.: Ensemble machine learning: methods and applications. USA: Springer (2012)
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. Cambridge: MIT press (2016)
8. Zadeh, L., Aliev, R.: Fuzzy logic Theory and applications. London: World Scientific Publishing Co (2019)
9. Smith, C.: Decision trees and random forests. Blue Windmill Media (2017)
10. Natekin, A., Knoll, A.: Gradient Boosting Machines, A Tutorial. Frontiers in neurorobotics **7(21)**. (2013). doi: 10.3389/fnbot.2013.00021
11. Buhlmann, P. Bagging, boosting and ensemble methods. Handbook of computational statistics: Concepts and methods. Berlin: Springer. pp. 877–907 (2004)
12. Decision Tree Classifier, https://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.htm/l
13. XGBoost Documentation, https://xgboost.readthedocs.io/en/latest/index.html
14. CatBoost, https://catboost.ai/
15. Imbalanced-learn documentation, https://imbalancedlearn.readthedocs.io/en/stable/index.html
16. Baranov, V.A., Lugovaya, N.M., Mikhalev, A.S., Kudymov, V.I., Strekaleva, T.V.: The algorithm of overall optimization based on the principles of intraspecific competition of orb-web spiders. In: Proceedings of the II Conference on Advanced Technologies in Aerospace, Mechanical and Automation Engineering - MIST: Aerospace-2019, vol. 734 (2019)