A Random Forest Based Classifier for Error Prediction of Highly Individualized Products

Gerd Gröner

Carl Zeiss Vision International GmbH
http://www.zeiss.com
gerd.groener@zeiss.com

Abstract. This paper presents an application of a random forest based classifier that aims at recognizing flawed products in a highly automated production environment. Within the course of this paper, some data set and application features are highlighted that make the underlying classification problem rather complex and hinders the usage of machine learning algorithms straight out-of-the-box. The findings regarding these features and how to treat the concluded challenges are highlighted in a abstracted and generalized manner.

Keywords: random forest classifier, imbalanced data, complex tree-based models, high peculiarity of data

1 Introduction

In a manufacturing process with highly individual products like ophthalmic lenses, which are produced according to personalized prescriptions, it is difficult to identify orders that are likely to fail within the production process already in advance. These products might fail due to their difficult and diverse parameter combinations. The parameters cover raw material characteristics, lens design, geometry and manufacturing parameters (i.e., machine setting values). Even such individual, prescribed products are not excluded from hard market competitions. Accordingly, avoiding waste of material and working time is an emerging problem. Obviously, since such customer-specific, individual products are not interchangeable or replaceable by other products (like in case of on-stock products), it is highly valuable to avoid any kind of scrap / failure already beforehand the production. Summing up, it is becoming more and more useful to analyze product (order) parameters and find features and feature correlations in order to predict (potential) failures already prior to the start of any manufacturing process.

In our case, we are confronted with a rather hard problem since the products can not be perfectly discriminated into good or bad ones solely based on their product characteristics (which are given by individual prescription and design in our case) and their corresponding target processing parameters. Therefore, it is a challenging machine learning (ML) task to remedy this problem within an advance distinction between good and potential faulty products, while, at the

J. Beyerer et al. (Eds.), *Machine Learning for Cyber Physical Systems*, Technologien für die intelligente Automation 9, https://doi.org/10.1007/978-3-662-58485-9_4

same time, avoiding ML pitfalls like over-fitting. Furthermore, the pure number of features is high and the data set is quite imbalanced, hampering the straight forward exploitation of ML models.

Until now, ML is used for error detection in different manufacturing areas (e.g., [1–3]), but due to the domain-specific data (highly individualized) and fully-automated and very standardized manufacturing processes, the gap between different parameter combinations and the resulting processing steps is an open challenge for applying ML technologies and assessing their benefits accordingly.

We present a *random forest classifier* for error prediction that resulted from a deep analysis of different ML algorithms, which has been used to train various models. These models are evaluated in terms of their classification quality. The best model is presented in detail. Interestingly, doubts (like difficult distinction) and findings (like important features) of the domain experts form the manufacturing division were confirmed by the model. Finally, we give an argumentation why the random forest model outperforms other (rather complex) models like Neural Networks and Support Vector Machines (SVM) within this particular use case.

2 Background

This section shortly outlines background information on a particular studied use case, followed by some principles on machine learning.

I. Use Case: Error Recognition and Prediction. For an ordered product, we focus on the relevant product features and the according machine setting parameters. Summing up to 130 features that describe the product, i.e., lens in our case by data on geometry, shape, target prescriptions, coatings and tinting values. We removed identifiers like order number and dates. In the used data set, we have about 560000 entries in total (i.e., products), covering those products without errors and such cases, where the first production was erroneous and a further (second) production cycle was necessary.

As we train, test and evaluate our model with historical data, for each product there is the corresponding characteristic whether it is an error or a non-error (binary classification). Since we are interested in an advance classification of products (and their corresponding to-be processing parameters), we neglect in the historic data those errors that were cased by operators, by unexpected machine failures or by other arbitrary circumstances. The remaining proportion of (final) errors is about 5.4 %.

II. Machine Learning (in Practice). Based on the use case, we are faced with a binary classification problem (i.e., we distinguish – at least in a first step – between good and potential bad products). This problem (*classification*) constitutes one group of algorithms in the realm of supervised machine learning, while the second group of algorithms of supervised learning is referred to as a *regression* problem, where instead of discrete categories (as in our case) a continuous value is the target output of a model. Among *classifications*, there are

a variety of algorithms (cf. [4–6]), ranging from rather basic ones like regression and Naive Bayes, to more difficult algorithms (in terms of setting-up and computation) like artificial neural networks (ANN), support vector machines (SVM), decision trees and extensions of them like random forests classifiers (RFCs) and boosted decision trees. Boosted decision trees and random forests belong to the so-called ensemble algorithms, i.e., a set of trees or a forest is built by an ensemble of decision trees. Ensemble algorithms implement methods to generate multiple classifiers and then aggregate their results (cf. [16]). Boosted decision tree algorithms apply a strategy of state-wise optimization of trees (measured in terms of loss functions) [14, 15]. Trees within the ensemble of random forests are built by randomly selecting the input features. Each tree in the ensemble is obtained by randomly selecting the input features. Within each tree, each node is still split by using the best feature (measured in terms of cost functions). The final result of the forest is obtained by unit votes from the trees for the most popular class.

3 Characteristics of the Data Set and the Application Scope

The data set is obtained from a rather dedicated domain, following a production process for highly individualized products, there are some essential key characteristics that are comparable and transferable to different problems in completely other domains. Therefore, we have to tackle challenges to cope with the following data and application characteristics.

The data set is highly *imbalanced*, which is actually in the nature of error and non-error classification problems. As already mentioned, we have a relationship of roughly 5.4 % belonging to the minority class (error case), while slightly more than the remaining 94.6 % of the data samples belong to the majority class (non-error case). It is well known that the best classification results can be achieved on balanced data sets (cf. [11–13]). Furthermore, in our case, we are not only interested in the correct classification, we also want to know which are the most influential features for ending-up in one of these two classes. Thus, a sound prediction model that is able to do a proper classification (i.e., a non-guessing solution!) is needed.

A further property is the *complexity* of the model. The pure number of samples (roughly 560000 entries in the data set) is a decent size, but the compared amount of features (about 130) is rather high. In particular, not only the number itself is an issue, it is rather the feature characteristic that counts for complexity, as we will see later. There are no dominating single features and the number of influential features is high, ending up with models that need a deep consideration of feature manifestation and combinations, as demonstrated in the next section.

Finally, the third characteristic is the vague *discriminability*, which is the most difficult one to handle in our case. Given all the features of a particularly ordered product of an error case, the manufacturing process at the first time has failed, while the second run with quite similar or even the same features (in-

cluding machine setting parameters) ended-up with a good quality. Accordingly, such a concrete characteristic of product attributes is not able to determine in advance whether an error or a non-error case is given.

4 A Random Forest Model for Error Prediction

This section presents the set-up of the model training, starting with the necessary data preparation steps, the part of algorithm set-up and result comparison, followed by the evaluation and an discussion of the design decisions and the achieved results.

4.1 Data Preparation and Preprocessing

After the basic step of creating a data model within a database and cleaning tasks like dealing with outliers and missing values, we applied several feature engineering steps. We have to deal with various categorical values. Even if some algorithms are able to directly handle them, we applied a general encoding of all categorical features. We use the established `one-hot-encoding` method for this step. Furthermore, for some parameters with different values within the production steps (steps in the production process), the results improved by adding aggregations of these parameters like average values to the data set.

4.2 Features and Feature Distribution

Among the features (independent variables) there is a clear ordering regarding feature importance, but there is no clear dominance of a single feature or of a small group of features. For instance, the relative importance of the most important feature is about 0.0383, the 10th important feature still reaches a relative importance of roughly 0.0302.

Figure 3 shows the distribution of the first and the tenth important feature. The features are renamed here, param. 1 refers to the first / most important feature (Figure 1) and param. 2 to the tenth important feature (Figure 2). We added suffixes in the plots to show the distribution of the error and non-error case separately. The plots depict the distribution of the whole data set (i.e, including data of the train and test part). The left box (i.e., the suffix "majority") refers to the values of the majority class (i.e., non-error case), while the suffix "minority" refers to the values of the minority class (i.e., error case)).

4.3 Algorithm Comparison and Selection

We built all models by training with several algorithms, using the Python programming language and libraries like the Scikit-learn library[1] in Python.

The data set is split up into training (0.7) and test (0.3) data. The results show that the data contains rather complex interactions among the most relevant

[1] Scikit-learn: `http://scikit-learn.org/stable/`

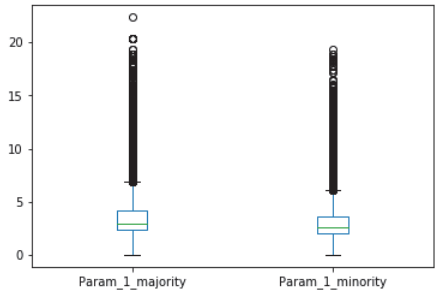

Fig. 1. Most important feature.

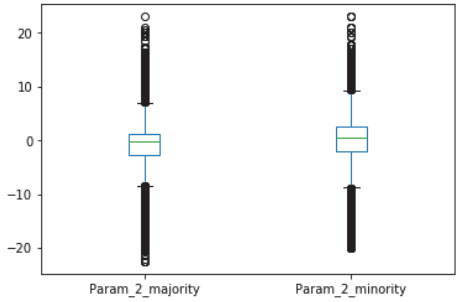

Fig. 2. 10th most important feature.

Fig. 3. Box plots for the distribution of two features.

features. Moreover, the discrimination between error and non-error (if possible at all) requires the comprehensive consideration of various features and their relations, which has been outlined in our comparison. For instance, less-complex algorithms like Naive Bayes and regressions are not able to do a decent classification. Algorithms known as complex and partially hard to initialize like support vector machines (SVM) and artificial neural networks (ANN) are able to make proper binary classifications, but with a low F1 score. Tree-based algorithms outperform all others. The best results are obtained by boosted trees and, slightly better, by random forest classifiers.

Table 1 shows an excerpt of an algorithm comparison. The first column describes the used algorithm to train the model. Column two gives the setting parameters of the algorithm. If no parameter is given, the default values are taken (from Scikit learn). The presented setting parameters are those which ended up in the best results, mainly received by several trials and applying cross-validation strategies (We used a 5-fold cross validation on the training data set).

The third column describes the performance in terms of *precision*, followed by the *recall* in column four and the summarized *F1 score* in column five, concluded by the ROC-AUC value (area under the ROC curve). All models where trained with these algorithms from the Scikit learn package in Python.

For the *random forest classifier (RFC)*, we explicitly parametrized the algorithm with the minimum number of samples for a split to 3, and no limit of the maximum depth of the branches in a tree. The quality of a split is measured by the Gini impurity. This measure judges the quality of a selected target variable, which is used to split a node, i.e., reflecting the *importance* or *"best split criteria"* in a tree. The Gini impurity measures how often an element is wrongly classified (i.e., assigned to a subset (bin)), if the "correct" label reflects the random label assignment of the distribution of labels within the subset.

The *boosted decision tree* (implemented by AdaBoost in Scikit learn) has been constituted within a rather similar setting. The tree properties are set to the minimum number of samples for a split to three, no limitation on the depth

and also the Gini impurity is used to assess the split quality. The learning rate shrinks the contribution of a single classifier within the ensemble. We use the default boosting algorithm (SAMME.R), which aims at converging faster than the other options.

The *artificial neural network (ANN)* (also referred to as multi-layer percetron - MLP - classifier) uses an adaptive learning rate, which means that the learning rate is reduced (divided by five) as far as in two successive runs the training loss does not decrease. The parameter alpha represents the regulation of the L2 penalty (i.e., Ridge penalty). The value is higher than the default, implying smaller coefficients (weights). The parameter on the hidden layers defines the number of hidden layers (five in our case) and also the number of nodes (neurons) in each layer.

For the *support vector machines (SVM)* (or support vector classifier), we use the rbf (radial basis function) kernel. (The rbf kernel uses a squared Euclidean distance as measurement for data (point) separation. The gamma coefficient is set to auto, which meas that the quotient from one and the number (n) of features. The penalty parameter for errors (C) is five. This parameter is balancing between errors in training compared to errors in testing, i.e., it influences the generalization of a classifier to unseen data.

Table 1. Comparison of Model Performance.

Algorithm	Parameter	Performance			
		Precision	Recall	F1 Score	ROC-AUC
RFC	criterion: Gini min-sample-split: 3 max-depth (tree): none	0.74	0.4	0.52	0.72
Boosted Tree (AdaBoost)	criterion: Gini min-sample-split: 3 max-depth (tree): none learning rate: 0.4	0.72	0.39	0.51	0.71
ANN (MLP)	learning rate: adaptive alpha (L2 penalty): 0.1 hidden layer sizes: (70,70,50,40,40)	0.59	0.24	0.34	0.55
SVM	kernel: rbf gamma (coef.): auto (=1/n) C (penalty for error): 5%	0.55	0.19	0.28	0.52

The random forest classifier was set up by using a 5-fold cross validation (grid search with parameter alternatives) in order to find the best parameter combinations (e.g., the minimum samples within a leaf). We need very deep trees (setting no depth limitation) and a very low splitting rate in the nodes (best results are achieved with three sample splits). The average tree depth is 51. A further interesting finding is the distance between precision and recall. While the precision is about 0.74, recall ended up with 0.4 (F1 score is 0.52).

Fig. 4 depicts the ROC curve (Receiver-Operating-Characteristic curve) for the random forest classifier. The true positive rate (i.e., the recall rate or also referred to as sensitivity) is depicted on the y-axis, the x-axis shows the false positive rate.

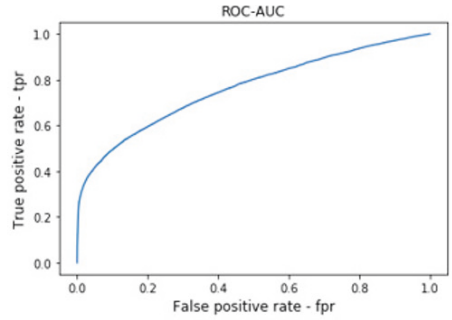

Fig. 4. The ROC curve of the random forest classifier.

4.4 Algorithm Comparison and Selection

While it is often argued that both described tree algorithms (i.e., boosted decision trees and random forests) tend to perfectly adapt their feature values and thus suffer often from overfitting, Breimann [5] showed that random forests are robust against overfitting, providing (among others) possibilities to set regularization parameters.

4.5 Evaluation, Results and Design Decision Revisited

It is worth to notice that due to the rather low ratio of the error samples (so-called minority class), we applied re-sampling methods [7, 8] to obtain a more balanced data set. The best results were achieved by down-sampling (i.e., reducing the data set size) in combination with a slight up-sampling, such that the error ratio raises up to nearly 18 %. There is no dominating feature among the most important features.

While several practical comparisons (e.g., [19]) show that the complex ANN outperforms random forests, the variety of important (but not dominating features) combined with their different results of interactions and the threat of overfitting might cause the predominance of random forests in our case.

Nevertheless, we stress that the best results of the random forests is based on the underlying data set and application use case with no indication as a general superiority of random forest classifiers to other classification algorithms, which was for instance argued in [18], but later contradicted (in terms of generalizability) in [17].

It is definitely hard (or even impossible) to explain why a certain algorithm (like random forests in our case) provide the best results compared to other algorithm. We will follow some discussions like on KDnuggets[2], on blocks like Towards Data Science[3] as well as in a work on energy consumption analysis [19].

The models built by random forests are known as rather robust models, i.e., they are able to better handle outliers, missing data or just weird values. We realize a slight overfitting, which is a well-known problem of random forests (especially with deep trees), but it is minor and negligible in our case.

Neural networks (and also SVM) are more difficult to parametrize. Although we applied various training iterations with different parameter settings (always including default parameters), it is still imaginable that a better parameter combination for the algorithms exists and the resulting model would outperform our current best random forest solution. Furthermore, our model covers very complex interactions among features, which is shown by the very deep trees (compared to the total number of features). However, all features are numerical values or categorical values, there are no images and we are not in the realm of image or speech processing, which are known areas where neural networks and SVM (especially for text data) mostly outperform other algorithms.

5 Summary and Outlook

In this paper, we presented a study for in-advance error classification in a highly individualized production environment. The best predictions are achieved by tree-based algorithm, in particular by a random forest classifier that achieves a rather decent precision rate to forecast whether a particular ordered product is likely to fail or not. However, the recall is comparable low. As the data set is highly imbalanced, we used sampling strategies to slightly improve the ratio between errors and non-errors in our data set.

As future work, we train our models with an updated (newer) data set, containing more data in both dimensions, i.e., for data entities samples, but also slightly more features. The expectation is that this will increase the algorithm performance.

References

1. Henmi, T., Deng, M., Yoshinaga, S.: Early Detection of Plant Faults by Using Machine Learning. *Int. Conf. on Advanced Mechatronic Systems (ICAMechS)*, 2016
2. Zidek, K., Maxim, V.: Diagnostics of Product Defects by Clustering and Machine Learning Classification Algorithm. *Journal of Automation and Control*, vol.3, 2015

[2] Post on KDnuggets: "When Does Deep Learning Work Better Than SVMs or Random Forests?", `https://www.kdnuggets.com/2016/04/deep-learning-vs-svm-random-forest.html`

[3] `https://towardsdatascience.com`

3. Meshram, A., Haas, C.: Anomaly Detection in Industrial Networks using Machine Learning: A Roadmap. *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2016. Ed.: J. Beyerer*, Springer, pp. 65–72, 2017
4. Géron, A.: Hands-On Machine Learning with Scikit-Learn & TensorFLow. O'Reilly, 2017
5. Breiman, L.: Random Forests. *Machine Learning*, pp. 5–32, vol. 7, Kluwer Academic Publishers, 2001
6. Rashid, T., Neuronale Netze selbst programmieren. O'Reilly, 2017
7. 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*, vol. 18, pp. 1-5, 2017
8. 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. *ACM SIGKDD*, vol. 6 (1), pp. 20–29, 2004
9. Ahmad, M.W., Mourshed, M., Rezgui, Y.: Trees vs Neurons: Comparison between Random Forest and ANN for High-Resolution Prediction of Building Energy Consumption. *Energy and Buildings, Elsevier*, vol. 147, pp. 77–89, 2017
10. Zhang, Y., Guo, W., Ray, S.: On the Consistency of Feature Selection With Lasso for Non-linear Targets. *Proc. of the 33rd Int. Conference on Machine Learning*, vol. 48, pp. 183–191, 2016
11. Eitrich, T., Kless, A., Druska, C., Meyer, W., Grotendorst, J.: Classification of Highly Unbalanced CYP450 Data of Drugs Using Cost Sensitive Machine Learning Techniques. *Journal of Chemical Information and Modeling*, vol. 47 (1), pp. 92–103, 2007
12. Wang, S., Yao, X.: Multiclass Imbalance Problems: Analysis and Potential Solutions. *Systems Man Cybernetics Part B - Journal IEEE Transactions on Cybernetics*, vol. 42, pp. 1119–1130, 2012
13. Kubat, M., Matwin, S.: Addressing the Course of Imbalanced Training Sets: One-Sided Selection. *Proc. of the 14th Int. Conference on Machine Learning*, pp. 217–225, 1997
14. Wyner, A.J., Olson, M., Bleich, J., Mease, D.: Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers. *Journal of Machine Learning Research*, vol. 18, pp. 48:1–48:33, 2017
15. Friedman, J.: Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, pp. 1189–1232, 2001
16. Liaw, A., Wiener, M.: Classification and Regression by Randomforest. *R news*, vol. 2 (3), pp. 18–22, 2002
17. Wainberg, M., Alipanahi, B., Frey, B.,J.: Are Random Forests Truly the Best Classifiers?. *Journal of Machine Learning Research*, vol. 17, pp. 110:1–110:5, 2016
18. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014
19. Ahmad, W. M., Mourshed, M., Rezgui, Y.: Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption *Journal on Energy and Buildings*, vol. 147, pp. 77–89, 2017