On solving fault detection problem and risk estimation monitoring with deep neural networks and postprocessing

Ivan Ryzhikov[1] Mika Liukkonen[2] Ari Kettunen[2] Yrjö Hiltunen[1]

[1]Department of environmental and biological sciences, University of Eastern Finland, Finland, {ivan.ryzhikov,yrjo.hiltunen}@uef.fi

[2]Sumitomo SHI FW Energia OY, Relanderinkatu 2, 78200, Varkaus, Finland {mika.liukkonen, ari.kettunen}@shi-g.com

Abstract

In this study, we consider fault prediction problem and production process risk monitoring based on observational data. We consider case, when there are no variables, by which one could classify the situation preceding to the fault. We propose an approach that is based on a specific auxiliary risk variable and modifications of the modeling accuracy estimation criterion, so the fault detection problem is reduced to supervised learning problem. We use deep learning and examine different model architectures. Trained model produces the risk estimations for new observations, then we use postprocessing to interpret the estimations to decision-maker. This work confirms that data-driven risk estimation can be integrated into digital services to successfully manage plant operational changes and support plant prescriptive maintenance. This was demonstrated with data from a commercial circulating fluidized bed firing various biomass and residues but is generally applicable to other production plants.

Keywords: *deep learning, fault detection, risk estimation, postprocessing*

1 Introduction

In this paper we consider a real-world problem concentrating on boiler fault prediction in biomass-fired circulating fluidized bed (CFB) power plants. These plants are extremely important and have not only the financial benefits, but also benefits for the environment as they can be used to replace fossil-fuel -based power generation. Plants of this type can utilize challenging fuels such as biomass or waste residues efficiently, but the drawback is that these types of fuel may often cause different problems such as blockages in the material flow. Especially this concerns biomass fractions that include large amounts of alkali metals. Although the consequences of the blockages are serious, we still cannot measure the quality of the fuel accurately and need to control the process using the observational data coming from different other sensors. At the same time rapidly evolving energy market sets challenges to traditional combustion-based power plants as it demands efficiency and flexibility in terms of fuel and load range. For example, the share of biomass as an energy source has increased significantly during recent years and it is expected to keep on increasing. In this study we propose and apply an approach to find patterns in a system state that takes place priorly to the fault.

The fault prediction problem and state monitoring problem appear in many different industries. This problem is serious because faults bring damage to production process and causes loss of profit. Faults can cause production blockage or disfunction and companies require resources to stabilize the process. In (Paltrinieri and Khan, 2016) the importance of risk assessment is considered for chemical industries. In energy sector faults consequences are serious too: any unexpected load limitation or shutdown of a power unit can cause considerable economical losses. Usually, the cost of undoing the damage is much higher than the cost of preventing the fault and that is why it is important to monitor and analyze the system state. The production system state analysis can predict if the process is in risky state and we need to act to lessen the risk.

Production processes are complex so many of those does not have adequate mathematical models based on physics or chemistry. But if even we had a mathematical model, there is still high level of uncertainty: we cannot measure all the inputs and all the system states. Once we met uncertainty, we use data to fight it. This leads us to hypothesis of using the data and data-driven modeling to solve fault prediction problem.

But how do we know that some of the system states causes faults? In general, there are no state variables indicating that situation is getting risky. Even process experts cannot name the conditions by which we could determine the pre-fault state. Anyway, if there are such variables, the approach we consider in this study can be applied too. Commonly, these conditions could be complex, and first we need to recognize those. When we can identify the system state with some value representing how close it is to pre-fault condition, and this is where we use mathematical model. In this case mathematical model is a mapping that reflects the observations to indicative values that shows if the system is risky and earlier it led to one of the fault cases. The next step would be transforming the indicative value to decision. This next step is based on another mathematical model, which we call a post-processing unit, that helps production expert to categorize the situation.

To sum up, we propose ap approach that is based on recognition of patterns in data that led to failure in the past and help production experts in decision-making by mapping state evaluation to clear and succinct labels. Of course, similar patterns can be met also in common functioning, so it is important part of the approach to deal with interpretation contradictions. But adjustment of proposed decision-making support system needs to involve economical effect calculation. We can make system more sensitive and increase the number of times, when the system indicates, that the current situation is risky, so the production expert needs to act. Or we can make it less sensitive and focus only on patterns that proved their statistical relation to fault. System sensitivity is the question that can be solved with business only and by measuring the economic effect.

Proposed approach helps analyzing the patterns leading to the faults and revealing if similar conditions caused the fault in different cases.

In this study we reduced the fault prediction problem to regression problem, where we use observational data to train the model utilizing machine learning techniques. We adjusted the modeling criterion to fit the problem and applied a specific mapping to the modeling results to interpret the model predictions so the model and its postprocessing unit can work online to solve the risk monitoring problem.

2 Risk Estimation and Fault Prediction

Statistical modeling is applicable to solve various application problems (Kuhn and Johnsson, 2016) and computational resources today allow solving complex modeling problems. We can apply deep neural networks, train them on large datasets and produce a value for the production. Deep learning algorithms proved their efficiency in solving complex modeling problems (Chollet and Allaire, 2018) and (Goodfellow et al., 2016). Digital transformation or Industry 4.0 has high demand in statistical models and modeling methods (Brink et al., 2016) since many of models are data-driven or learning from the data.

In many different studies machine learning algorithms were applied to solve the fault prediction problem, but considered approaches are applicable to specific domain or when there is known variable, by which one can measure how safe is the process. For example, in (Paltrinieri et al., 2019) the machine learning based approach is considered as a promising tool of solving risk estimation problems, but in their study was a variable that represents the risk level. But the process we study does not have such variables. If we could label the system states data, we could apply the approach considered in study (Bondyra et al., 2018). But we have thousands of observations and no information on how we can estimate the degree of risk for each system state.

Approach based on labeled data is also presented in study (Rackshani et al., 2009), where authors consider the fault prediction problem for a power plant boiler and solved it by means of deep neural network. In their study the risk variable was constructed based on the fact of immediate faults and 8-hours operating cycle. This approach is difficult to be applied if there are only a few fault cases in dataset. It also makes it difficult to use this model to make just in time decisions, since the model is trained on aggregated data. It is also hard to detect if the reason for the fault was observed earlier than the working cycle interval. Approach without data aggregation was considered in the study (Hujanen, 2019), where the problem was reduced to the classification problem with 3 classes and deep neural networks were trained.

In this study we propose an approach that is based on recognizing of specific patterns in data, that caused the system fault in the past. We construct the auxiliary risk variable that indicates how dangerous is the current state. We assume that risk starts to grow some time before the fault and all the other time it is low. This risk interpretation is a simplification of the risk definition done by (Kaplan and Garrick, 1981), and we are not estimating the consequences and probabilities.

Having risk variable makes it possible to reduce the fault detection problem to supervised learning problem. But there is uncertainty of the actual risk value for the observations that do not belong to the prior to the fault interval. In following paragraphs, we consider the risk variable construction, the adjustment of criterion and the postprocessing of the modeling results.

2.1 Problem Reduction

The considered process state can be characterized by different inputs that correspond to the sensor data from the different parts of the boiler plant. Each of these inputs can be described as time series with fixed step size: $X = \{x_1, x_2, \dots, x_s\}, T = \{t_1, t_2, \dots, t_s\}$, where s is a sample size. We also know m times at which the fault happened: $t_i^f, i = \overline{1, m}$, so we assume that there had been some time before that, at which the risk began to grow.

2.1.1 Risk Variable Construction

This time before the fault is a parameter Δ of the proposed approach. We put forward a hypothesis, that there is no risk in any other timestamp, than timesteps before the fault limited by the parameter. We also assume that risk increases monotonically starting from zero, and it reaches its maximum value of one by the fault time, so the risk variable can be evaluated by the following function

$$r(t,t^f) = \begin{cases} \frac{t-t^f}{\Delta} + 1, t^f - \Delta \le t \le t^f, \\ 0, otherwise, \end{cases} \quad (1)$$

where t_f is the fault time and Δ is the parameter. Since there could be m different faults, the risk function for whole observation time can be evaluated as a sum of single fault functions (1):

$$r(t) = \sum_{i=1}^{m} r(t, t_i^f). \quad (2)$$

We assume that there is always a normal system state between the different faults, so it is possible to find such Δ that $\nexists i,j,\ t_i^f < t_j^f : t_j^f - t_i^f < \Delta$, so non-zero intervals of the risk functions are not overlapping. According to this approach, we need to find a relation between the system state variables and the risk feature. In this study we assume that the risk is increasing identically before any of the faults.

2.1.2 Criterion Adjustment

We need to split the data on train and test sets to estimate the adequacy of model and its generalization. Since we work with time series, which consists of several intervals corresponding to several faults, we consider two splitting schemes. First option is to leave the data for one of the faults for the test and to keep other faults data for the train. This would help us to understand which faults have similar (or different) patterns corresponding to the risk increase. Second option is to split the data on two subsets, one before some date as train and validation and second after that date as test. In that case we can see, how good is historical data in predicting the future faults. To provide validation we used stratification, so train and validation contain observations from a common process and observations from the interval before the fault.

As a modeling criterion we used the root mean square error

$$I(\tilde{r}) = \sqrt{\sum_{i=1}^{n} \left(r(t_i) - \tilde{r}(x_i)\right)^2}, \quad (3)$$

where n is a test or validation subset size, $r(t_i), i = \overline{1,n}$ are risks (2) at t_i timestamps and $\tilde{r}(x_i), i = \overline{1,n}$ are risk estimations at the same time points by the model. Since we cannot properly estimate the risk for the time, when no fault was detected and we cannot estimate the risk for time intervals right after the fault, we suggested to use specific weights for these errors in the sum (3):

$$I_w(\tilde{r}) = \sqrt{\sum_{i=1}^{n} w(t_i) \cdot \left(r(t_i) - \tilde{r}(x_i)\right)^2}, \quad (4)$$

where $w(t)$ is a weighting function,

$$w(t) = \begin{cases} w_{after}, t \in T_{after}, \\ w_{normal}, t \in T_{normal}, \\ w_{risk}, t \in T_{risk}, \end{cases} \quad (5)$$

and T_{after} are the time intervals corresponding to states after the faults, T_{risk} are the time intervals before the faults and T_{normal} are the other intervals. Here w_{after}, w_{normal} and w_{risk} are weighing coefficients. These coefficients are used for increasing the influence of errors caused at the points, when the risk was growing and decrease the influence of errors of risk estimation for the time intervals for which the risk value is uncertain.

2.1.3 Supervised Learning Problem

The goal of our risk modelling approach is to estimate the risk of the current system state and to observe its dynamics for decision making. It means that we need to have model with optimal parameters α^*, which is adequate in risk estimation and thus minimizing the criterion (4):

$$a^* = \operatorname*{argmin}_{\alpha} I_w\left(\tilde{r}(x|\alpha)\right), \quad (6)$$

where $\tilde{r}(x|\alpha)$ is the model prediction in case of its parameters α, and

$$r^*(x) = \tilde{r}(x|a^*), \quad (7)$$

is the best model by criterion (5) for the data we have. The fault prediction problem is reduced to minimization problem (4), where we use specific weight coefficients (5). The solution of reduced problem is optimal model parameters (6), that we use to estimate a risk by system state variables. Now the risk estimation (7) can be used for fault prediction and decision making, but in this study, we consider interpreting risk estimations for decision making in production control.

2.2 Postprocessing

As a result of learning process, we have a model (7), which takes the system state as an input and returns risk prediction as an output. But we cannot use the risk prediction value to make decisions, because the single number cannot be interpreted. To solve the interpretation problem, we need another computational module, which takes the risk predictions and classifies the current situation.

Let $\tilde{r}_t, \tilde{r}_{t-1}, \dots, \tilde{r}_{t-m}$ be the latest m predictions of the model (7) and the postprocessing function is

$$P(\cdot): R^m \to R, \\ P(\tilde{r}_t, \tilde{r}_{t-1}, \dots, \tilde{r}_{t-m}) = l_t, \quad (8)$$

where l_t is the postprocessed value or label that classifies the production process state.

In this study we used three different classes: “good”, “warning” and “dangerous”, so $\forall i, l_i \in \{\text{good}, \text{warning}, \text{dangerous}\}$. First label indicates that process is running regularly, second label requires attention to the production process and the third one indicates that the situation can lead to a fault. In general, one can use any other classes and labels.

We used postprocessing function, which is based on filtering the low values and summation of all the risk values:

$$v_t = \begin{cases} 0 \text{ if } \exists i < m: \tilde{r}_{t-i} > b, \\ \sum_{i=0}^{m} \tilde{r}_{t-i}, \text{otherwise}, \end{cases} \quad (9)$$

where v_t is an intermediate numeric value and b is filter parameter.

Now we use intermediate values (9) to classify the process state:

$$l_t = \begin{cases} \text{"good" if } v_t < q_1, \\ \text{"warning" if } q_1 \le v_t < q_2, \\ \text{"dangerous" if } v_t > q_2, \end{cases} \quad (10)$$

and $q_1 < q_2$ are classification parameters.

2.3 Approach Parameters

Proposed approach has parameters, which need to be tuned. The first group of parameters is related to time intervals: Δ and time after the fault. In general, each interval can be characterized by its own parameters, but it this study we assume that Δ and time after the fault are similar for all fault cases. Production expert opinion is useful in determining time after the fault.

Weights (5) that we use in criterion have strong influence on results. General recommendations are to make $w_{risk} > 10 \cdot w_{normal} > w_{after}$, so recognition of risk increase before the fault is more important than small risk value in case of regular production process. It is also useful to resolve the contradiction if the same pattern led to fault in one case and did not in the other.

Prost-processing parameters are window size m, filter value b, borders q_1 and q_2. These parameters need to be tuned with production and business experts, because of their relation to the decision-making process. Window size depends on dynamical character of the process and rate of observations. Filter value among with the labeling borders can be tuned on the basis of the training data by adjusting the sensitiveness of the postprocessing system.

3 Data-driven Risk Estimation and Monitoring

The production process we want to estimate risk for has many observation variables. Each variable is measured every 15 minutes. We explicitly selected the variables with help of production experts to avoid overfitting and to focus on the factors of the main interest. The dataset contains 50879 observations and has a gap in observations. In given observation time there were 8 faults, each could be caused by own reasons and we do not know that in advance.

In this study we manually tried different time delta parameters and finally used $\Delta = 2$ hours. For that parameter and according to observation step size, we have only 192 observations that can be labelled as leading to the fault. One can see that the dataset is unbalanced: 50687 of “good” observations versus 192 of “leading to the fault” observations. For some production processes, it is typical that the faults occur uniquely, so there is imbalance between classes.

We also consider modeling when we include lags from the previous observations. In this paper we check the last 5 observations, which equals to 1.5-hour lookback. We will label these modes specifically.

The weighs (5) for criterion (4) are set as following: $w_{after} = w_{normal} = 1$, $w_{risk} = 10$. The weights were tuned manually and based on the modeling results feedback from the production experts. Weights (5) are important when adjusting the balance between fault sensitivity and the number of fail detections. In general, these characteristics should be tuned as a part of decision-making support system. It depends on resources company loses with any missed fail and resources company loses when act in case of alarming signal produced by the fault detection system.

We used the Keras framework (Allaire and Chollet, 2018) for modeling, and the application were implemented in R (R Core Team, 2018). We made a web application with R Shiny framework (Chang et al., 2021), that can be deployed to the company server. Since this application has access to the data needed it gives the results in visual form directly to the decision maker.

In this study we tried different deep neural network (DNN) architectures. Previously we tested that the proposed approach works for failure prediction problem solving (Ryzhikov et al., 2020) and now test if the efficiency changes with different DNN models. Each model layer has a dropout with 0.5 probability. When train model we use root mean square back propagation algorithm with a batch size of 5000 and 100 iterations.

For each model given in Table 1, we calculated the root mean square error (RMSE) on train and test data. As one can see, in Table 1 there are models, which include lagged variables. For those models we used lags for the previous 5 observations to check if including the historical data will improve the modeling results.

In this study we use specific learning data splitting: for each fault we produce training dataset, which contains all observations except ones that belong to an interval containing the fault, and test dataset, which is this interval. This splitting helps us to understand if one fault case can be predicted with model, which was trained on another fault cases.

When comparing model, we are interested in how these models predict on train data on intervals before the fault cases and all other intervals, and the same for test dataset. Intervals before fault and all other intervals for train data and for test data are given in Figures 1-4. Since dropout and initial coefficients are random, we provide boxplots of the RMSE and run each problem for 10 times.

Table 1. DNN architectures.

DNN	*Neurons by layer*	*With lag?*
1	64, 64, 64	no
2	64, 64, 64, 64	no
3	128, 64, 64	no
4	256, 64, 64	no
5	128, 64, 8, 64	no
6	64, 64, 64	yes
7	64, 64, 64, 64	yes
8	128, 64, 64	yes
9	256, 64, 64	yes
10	128, 64, 8, 64	yes

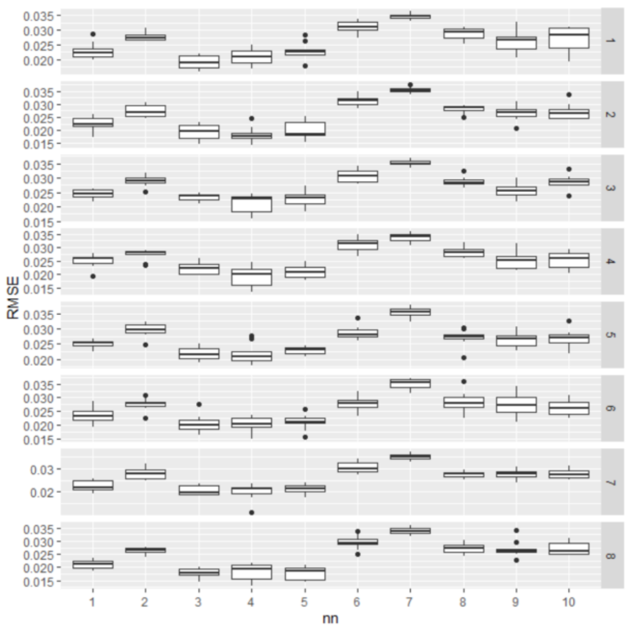

Figure 1. RMSE on train data, fault intervals, by model architecture on x axis and problem on y axis.

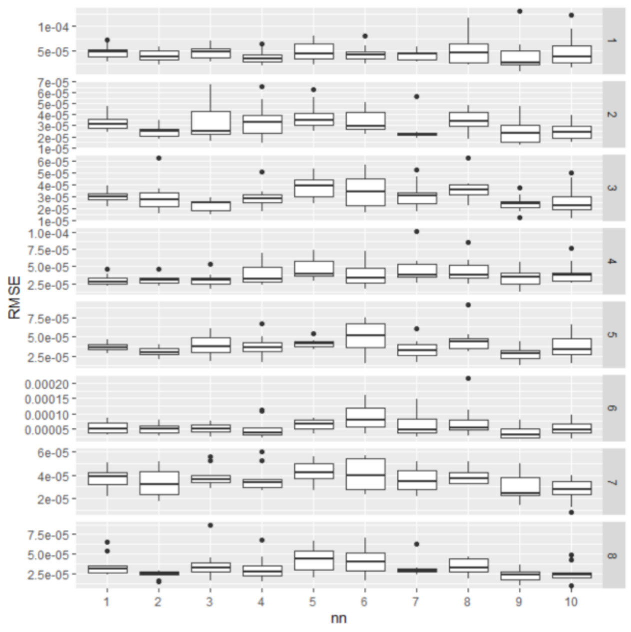

Figure 2. RMSE on train data, regular process intervals, by model architecture on x axis and problem on y axis.

As one can see, some of architectures outperform other architectures on pre-fault intervals risk estimation. Based on pre-fault intervals RMSE statistics we could assume that some of the models are preferable than others. When we look at RMSE statistics on regular process intervals, we can see that the variation of average results is not as big, as it is for fault intervals. Nevertheless, the most important part is prediction for data in test dataset.

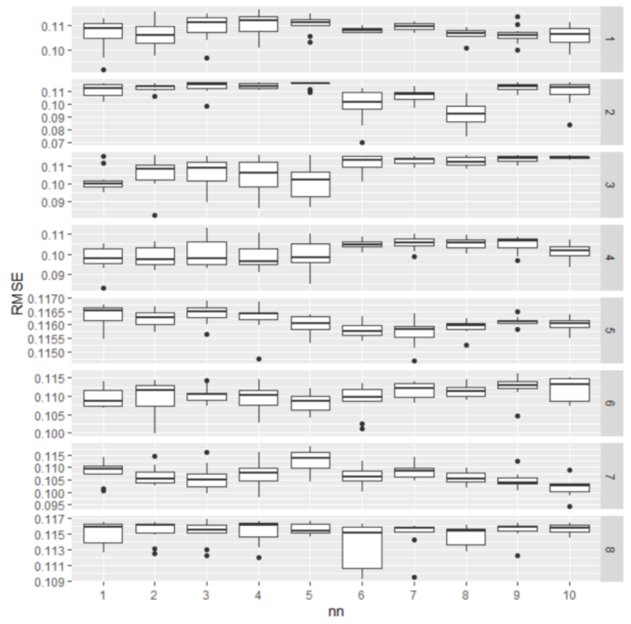

Figure 3. RMSE on test data, fault intervals, by model architecture on x axis and problem on y axis.

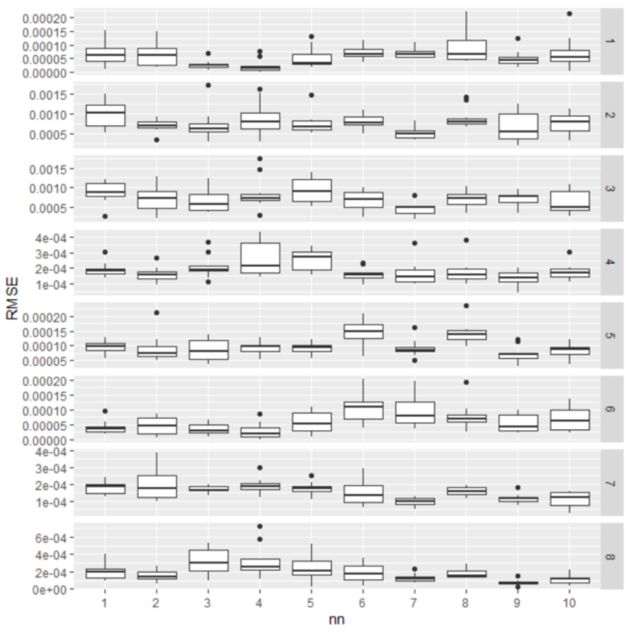

Figure 4. RMSE on test data, regular process intervals, by model architecture on x axis and problem on y axis.

When we compare model performance on test data, we can see some surprising results. For example, architecture 6, which was worse than 1-5 architectures, shows nearly the same or better mean values in 6 problems.

We need to have a closer look at model predictions on train in test data to understand the reason for that effects and why we cannot use RMSE to select the best model. First, our modeling approach assumes (1) and (2), that risk can be explained by the following function. But when we train the model, we are interested in having values greater than 0 in pre-fault intervals and values close to 0 in all other intervals. And models deliver that, but with different magnitude of values. Second, reasons for the faults and selected parameter for pre-fault interval could be different from one fault to another.

We demonstrate this effect in Figures 5-8. Figures 5 and 6 refer to DNN architecture 7, which outperforming statistic we observed above. We randomly selected one of the models from 10 runs for each architecture.

The subject of further research is another metric for comparing modeling results, that is based on ability to predict the fault in advance and

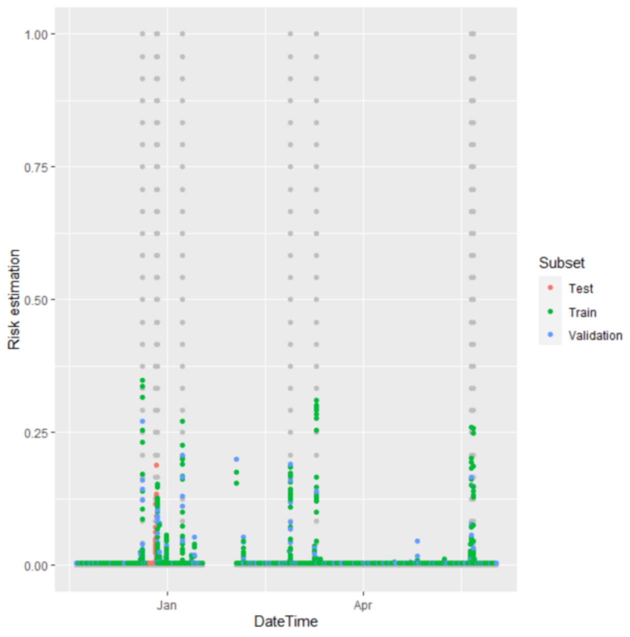

Figure 5. Risk estimation for all dataset, DNN architecture 7.

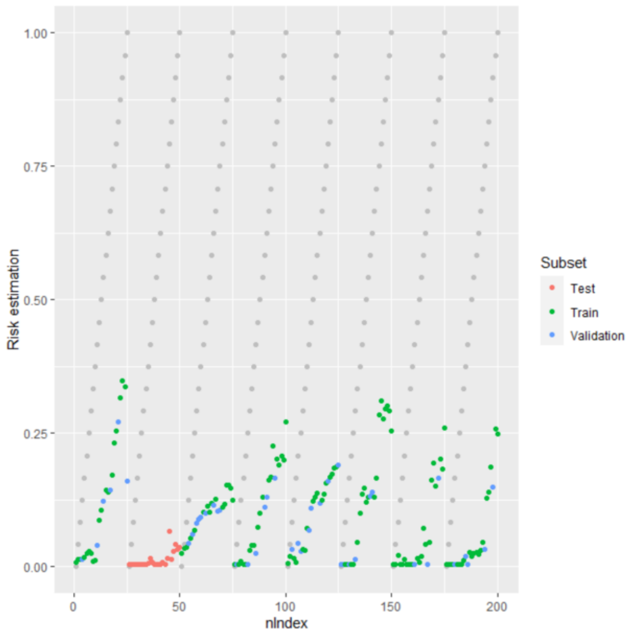

Figure 6. Risk estimation for pre-fault intervals only, DNN architecture 7.

According to plots on Figures 5 and 6, we can assume that model gives a good prediction on a training data, all other observations are near 0 and target interval shows some positive estimations of risk value. The maximum risk value is near 35. Let us compare these two plots with similar ones done for DNN with architecture 2.

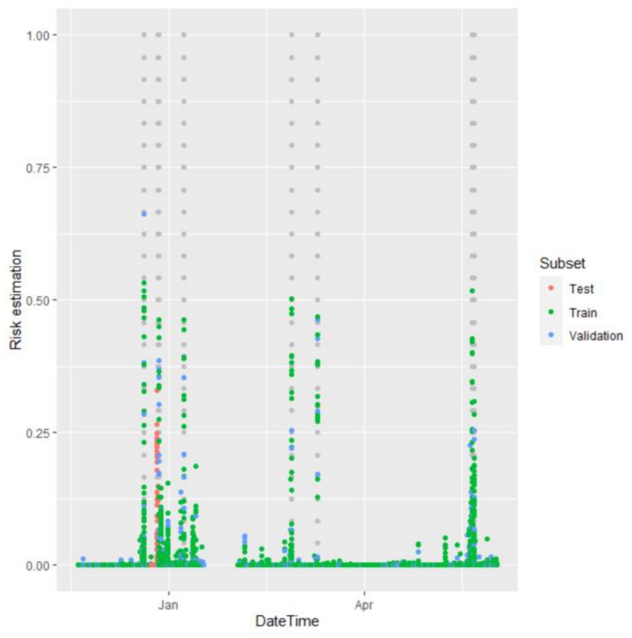

Figure 7. Risk estimation for all dataset, DNN architecture 2.

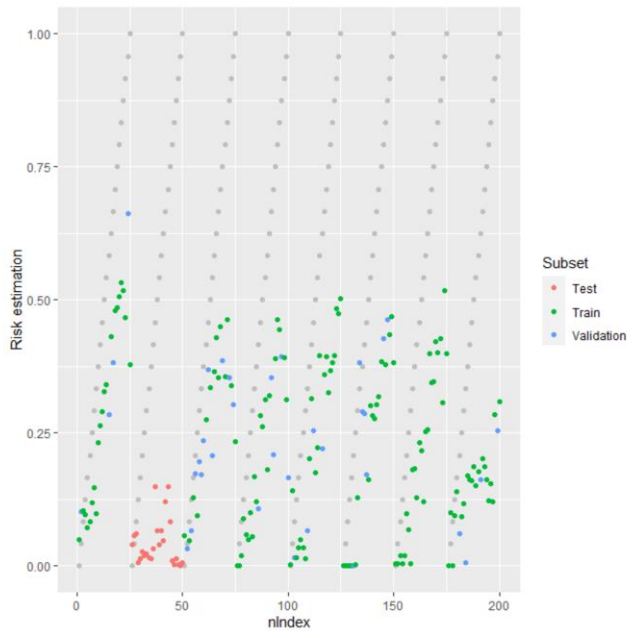

Figure 8. Risk estimation for all dataset, DNN architecture 2.

The prediction on all dataset looks like Figure 5, but fuzzier. The prediction of risk before the target fault is better, according to values, but all the other fault predictions have greater magnitude than ones in Figure 6. This proves our assumption that we need to design and use another criterion co compare different models and architectures, so we could produce them automatically.

In this study and solving real-world fault prediction and risk monitoring problem, we observed all the models manually, running tens and hundreds of model trainings to then choose several. Chosen model was taken as the basis for risk monitoring system. We used model predictions and (8)-(10) postprocessing approach. We adjusted the parameters of postprocessing manually, so these parameters detect the faults in advance.

Data loading, data preprocessing, modeling and postprocessing were implemented in web-application, implemented in R and R Shiny framework. In Figures 9-11 one can see the simulation where model receives new data, estimates risk and then postprocessor signals to application user interface that process is running fine, there is warning, and situation is dangerous. Interface shows also preselected number of previous risk estimations, so the decision-maker can see the situation dynamics.

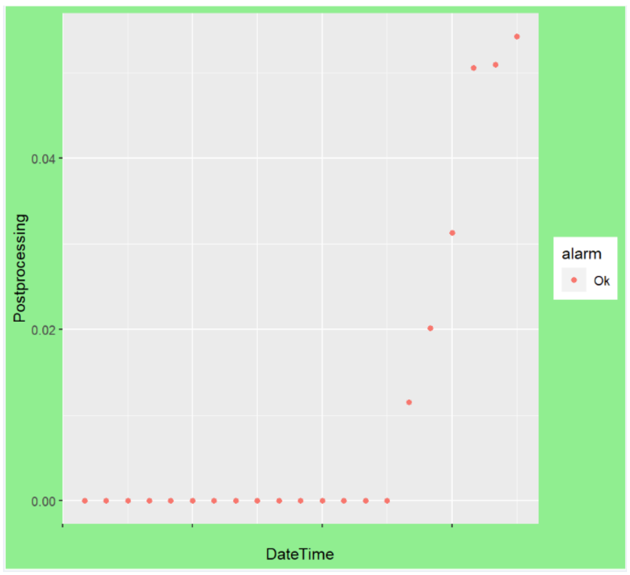

Figure 9. Risk monitoring, postprocessor receiving “Ok” state.

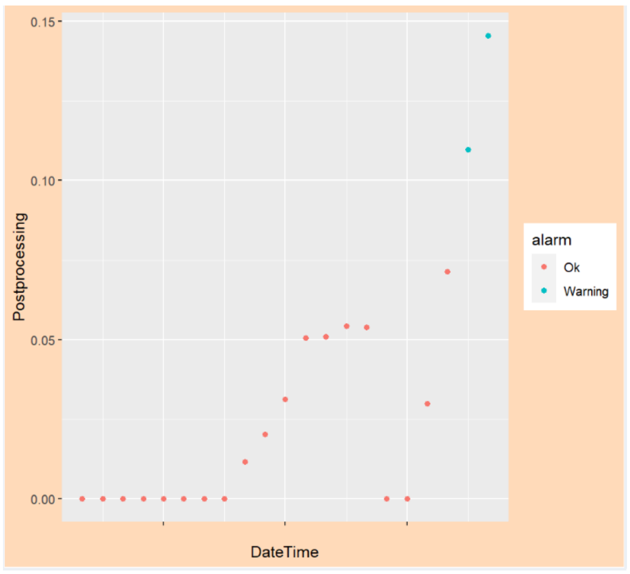

Figure 10. Risk monitoring, postprocessor receiving “Warning” state.

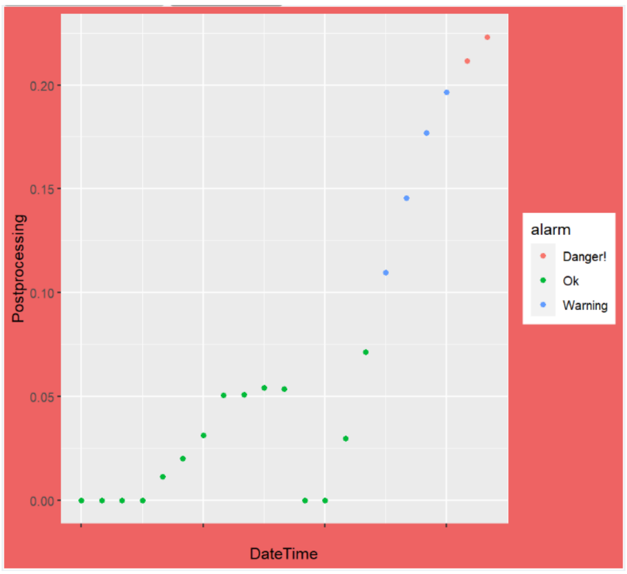

Figure 11. Risk monitoring, postprocessor receiving "Danger!" state.

4 Conclusion

In this study we examined proposed approach of risk estimation and fault detection. We applied postprocessing scheme that is based on filtering and risk summation, which makes possible to interpret the risk estimation model outputs and use this interpretation in decision-making.

Risk is constructed as auxiliary variable that monotonically increases in time interval prior to the fault. This variable makes possible revealing the patterns that possibly caused the faults even if these patterns were observed in different time before the fault and/or during the regular process. Once we apply leave one out testing and validation, we can estimate if the faults caused by similar system states. It is important to mention, that auxiliary risk variable helps us solving the fault detection problem, when there are no variables by which one could detect that there is something wrong with the production process and there is a fault risk.

Postprocessing of the model estimations makes it possible to interpret the results by once adjusting the mapping algorithm. As one can see, different models give risk estimations that differ in magnitude and sensitivity. By adjusting postprocessing we could suggest what is indication fits the production experts most: is one sensitive enough or accurate enough.

Further work is related with automatic model selection. We need to design criterion and searching algorithm that will compare models with different magnitudes of risk estimations and deal with uncertainty of different risk intervals.

We implemented the fault detection and risk estimation as web application with R, keras and R Shiny framework. This application can be deployed to the company network and work online, demonstrating the decision-maker the current estimation of risk.

Acknowledgements

This research is a part of the *AI-DA* projects, which is funded by Business Finland, European Regional Development Fund (ERDF), and seven companies.

References

J.J. Allaire and François Chollet. keras: R Interface to 'Keras'. R package. version 2.4.0., 2021. https://CRAN.R-project.org/package=keras

Henrik Brink, Joseph W. Richards, Mark Fetherolf. *Real-World Machine Learning*, Manning, 2016.

Adam Bondyra, Przemysław Gąsior, Stanislaw Gardecki and Andrej Kasiński. Development of the Sensory Network for the Vibration-based Fault Detection and Isolation in the Multirotor UAV Propulsion System. *In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics*, 2: 112-119, 2018. DOI: 10.5220/0006846801120119

Winston Chang, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert and Barbara Borges, 2021. shiny: Web Application Framework for R. R package version 1.6.0. https://CRAN.R-project.org/package=shiny.

François Chollet, J.J. Allaire. *Deep Learning with R*. Manning. 2018.

Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press. 2016

Jussi Hujanen. Machine Learning Methods for Early Process Deviation Detection in Circulating Fluidized Bed Boilers. *In Proceedings of Nordic Flame Days*, 2019.

Stanley Kaplan, B. John Garrick. On the quantitative definition of risk. Risk Analysis, 1(1): 11-27, 1981.

Max Kuhn, Kjell Johnson. *Applied predictive modeling*. Springer, 2016.

Nicola Paltrinieri, Louise Comfort, Genserik Reniers. Learning about risk: Machine learning for risk assessment. Safety science, *Elseiver*, 118: 475-486, 2019

Nicola Paltrinieri, Faisal I. Khan. *Dynamic Risk Analysis in the Chemical and Petroleum Industry, Dynamic Risk Analysis in the Chemical and Petroleum Industry: Evolution and Interaction with Parallel Disciplines in the Perspective of Industrial Application*. Butterworth-Heinemann. 2016. DOI:10.1016/B978-0-12-803765-2.01001-5.

R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2018. URL https://www.R-project.org

Elyas Rakhshani, Iman Sariri, Kumars Rouzbehi. Application of data mining on fault detection and prediction in Boiler of power plant using artificial neural network. *In proceedings of 2009 International Conference on Power Engineering, Energy and Electrical Drives*, pages 473-478, 2009. DOI: 10.1109/POWERENG.2009.4915186.

Ivan Ryzhikov, Mika Liukkonen, Ari Kettunen, Yrjö Hiltunen. Risk Estimation in Data-driven Fault Prediction for a Biomass-fired Power Plant. *In Proceedings of the 12th International Joint Conference on Computational Intelligence*, pages 423-429, 2020. DOI: 10.5220/0010113104230429