Technical report from Automatic Control at Linköpings universitet

Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting

Marcelo Azevedo Costa, Bernard Wullt, Mikael Norrlöf, Svante Gunnarsson

Division of Automatic Control

E-mail: macosta.est@gmail.com, bernhard.wullt@se.abb.com, mikael.norrlof@liu.se,mikael.norrlof@se.abb.com, svante.gunnarsson@liu.se

30th October 2018

Report no.: LiTH-ISY-R-3107

Address:
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

WWW: http://www.control.isy.liu.se

Technical reports from the Automatic Control group in Linköping are available from http://www.control.isy.liu.se/publications.

Abstract

Modeling and failure prediction is an important task in many engineering systems. For this task, the machine learning literature presents a large variety of models such as classification trees, random forest, artificial neural networks, fuzzy systems, among others. In addition, standard statistical models can be applied such as the logistic regression, linear discriminant analysis, k-nearest neighbors, among others. This work evaluates advantages and limitations of statistical and machine learning methods to predict failures in industrial robots. The work is based on data from more than five thousand robots in industrial use. Furthermore, a new approach combining standard statistical and machine learning models, named *hybrid gradient boosting*, is proposed. Results show that the a priori treatment of the database, i.e., outlier analysis, consistent database analysis and anomaly analysis have shown to be crucial to improve classification performance for statistical, machine learning and hybrid models. Furthermore, local joint information has been identified as the main driver for failure detection whereas failure classification can be improved using additional information from different joints and hybrid models.

Keywords: statistical modeling, machine learning, gradient boosting

Avdelning, Institution
Division, Department

Division of Automatic Control
Department of Electrical Engineering

Datum
Date

2018-10-30

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English

☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
☒ Övrig rapport
☐ _____

ISBN
—

ISRN
—

Serietitel och serienummer
Title of series, numbering

ISSN
1400-3902

LiTH-ISY-R-3107

URL för elektronisk version

`http://www.control.isy.liu.se`

Titel
Title

Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting

Författare
Author

Marcelo Azevedo Costa, Bernard Wullt, Mikael Norrlöf, Svante Gunnarsson

Sammanfattning
Abstract

Modeling and failure prediction is an important task in many engineering systems. For this task, the machine learning literature presents a large variety of models such as classification trees, random forest, artificial neural networks, fuzzy systems, among others. In addition, standard statistical models can be applied such as the logistic regression, linear discriminant analysis, k-nearest neighbors, among others. This work evaluates advantages and limitations of statistical and machine learning methods to predict failures in industrial robots. The work is based on data from more than five thousand robots in industrial use. Furthermore, a new approach combining standard statistical and machine learning models, named *hybrid gradient boosting*, is proposed. Results show that the a priori treatment of the database, i.e., outlier analysis, consistent database analysis and anomaly analysis have shown to be crucial to improve classification performance for statistical, machine learning and hybrid models. Furthermore, local joint information has been identified as the main driver for failure detection whereas failure classification can be improved using additional information from different joints and hybrid models.

Nyckelord
Keywords

statistical modeling, machine learning, gradient boosting

Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting

Marcelo Azevedo Costa[a,*], Bernhard Wullt[c], Mikael Norrlöf[c,b], Svante Gunnarson[b]

[a] *Universidade Federal de Minas Gerais, Brazil*
[b] *Department of Automatic Control, Linköping University, Sweden*
[c] *Robotics and Motion Division, ABB AB, Sweden*

Abstract

Modeling and failure prediction are important tasks in many engineering systems. For these tasks, the machine learning literature presents a large variety of models such as classification trees, random forest, artificial neural networks, fuzzy systems, among others. In addition, standard statistical models such as the logistic regression, linear discriminant analysis, k-nearest neighbors, among others, can be applied . This work evaluates advantages and limitations of statistical and machine learning methods to predict failures in industrial robots. The work is based on data from more than five thousand robots in industrial use. Furthermore, a new approach combining standard statistical and machine learning models, named *hybrid gradient boosting*, is proposed. Results show that the a priori treatment of the database, i.e., outlier analysis, consistent database analysis and anomaly analysis, have been shown to be crucial to improve classification performance for statistical, machine learning and hybrid models. Furthermore, local joint information has been identified as the main driver for failure detection, whereas failure classification can be improved using additional information from different joints and hybrid models.

Keywords: statistical modeling, machine learning, gradient boosting

1. Introduction

In 2001, Leo Breiman published the paper *Statistical modeling: The two cultures*, describing the two different approaches for data modeling: the data

*Corresponding author

modeling culture and the algorithm modeling culture. The data modeling culture first assumes a parametric, statistical model, hereafter named the white box model, and then uses available data to estimate the parameters of the model and provide statistical inference. The parameters of white box models may have physical meanings. Nonetheless, white box models apply, in general, simple mathematical and statistical structures, such as the linear regression equation. Consequently, the parameters of white box models, also known as coefficients, have meaningful interpretations.

The algorithm modeling culture is mostly interested in finding the most predictive model, hereafter named the black box model, which may have a complex structure and use a large number of variables. In general, black box models comprise general function approximators, which rely on non-linear functions and/or transformations. In general, the parameters of black box models do not have a straightforward interpretation as compared to white box models. However, the parameters of black box models can be tweaked in order to adjust the fitness of the function to different databases. Furthermore, in order to adjust the model complexity and/or select variables, cross-validation techniques are applied. Both white and black box approaches have advantages and limitations. Nonetheless, both rely on basic statistical principles (Friedman et al., 2001). For instance, linear regression theory (Seber and Lee, 1977) shows that the larger the number of predictors in a multiple linear regression, the more likely the multicollinearity and the more inconsistent the statistical inference about the parameters of the model. One approach is to select a small subset of predictors, therefore improving statistical properties. Another approach is to include a penalty in the optimization function, with no variable deletion; thus, compromising statistical inference but improving predictive error. Examples of white box models are linear regression models and generalized linear models. Examples of black box models are neural networks, support vector machines, fuzzy systems, among others.

Black box models may produce more reliable information about the structure of the relationship between inputs and outputs than white box models, mostly because they do not assume any prior parametric structure about the underlying structure of the data. On the contrary, white box models produce a simple and understandable picture of the relationship between the input variables and the output variable. For example, logistic regression is frequently used in classification problems because it produces a linear combination of the variables with parameters that indicate the variables' importance.

The basic tool to evaluate prediction performance of both white and

black box models is cross validation. According to Breiman et al. (2001), *cross-validation is a natural route to the indication of quality of any data-deriving quantity*. Briefly, the original data set is divided into two disjoint groups. The first group, named the training set, is used to estimate the parameters of the model. The second group, named the test set, is used to evaluate the error prediction of the previously adjusted model. Models with lower predictive errors are preferable. Finally, it is worth mentioning that Breiman et al. (2001) point out that the best available solution to a data problem may be a data model, or an algorithmic model, or a combination. The data and the problem should guide the solution.

This work evaluates a database comprising failures in industrial robots and production variables. Failure data is available for six joints. The goal is twofold: to find main drivers (features) or main predictors of failures, and to estimate the best predictive model. As previously mentioned, the two objectives are, in general, conflicting. On one side, selecting a few components highly correlated to failure is of utmost importance for good maintenance practices. On the other side, predicting failures as accurately as possible may save production resources and avoid unexpected interruptions in production. Standard statistical and machine learning models are proposed and evaluated.

In addition, a *hybrid gradient boosting* (HGB) method is proposed. HGB first uses a logistic regression model as a base line model. Therefore, it produces a simple statistical model which indicates variable importance. Furthermore, using a statistical framework based on the gradient boosting algorithm (Friedman, 2001), the HGB creates multiple layers of non-linear (black box) models on top of the logistic regression thereby achieving additional accuracy. As a result, the HGB model share both white and black box optimal properties.

It is worth mentioning that, due to natural degradation of the production machines, failures will inevitably happen. Possibly, different failures might happen several times. Thus, up to a certain time, which is unknown, all production machines will fail. Therefore, the available database represents a snapshot of the production robots in a specific time frame. Conclusions with respect to failure drivers are, therefore, constrained to the specific time frame in which the data was collected.

Furthermore, industrial robots are primarily designed to operate under pre-defined conditions, i.e., specific speed range, temperature range, among others. Thus, robots operating under critical or non-expected conditions are more likely to fail. Therefore, outlier detection techniques can be applied to data before using white or black box models. Domingues et al. (2018)

presents a comparative evaluation of outlier detection algorithms. Results show that the Isolation Forest (Liu et al., 2008) is an excellent method to efficiently identify outliers. Briefly, Isolation Forest is a non-parametric method which estimates data density using random partitions and tree based structures. Therefore, given the data, it is expected that observations located in low density regions are more prone to failure.

The following classification models were evaluated: logistic regression, regularized logistic regression (glmnet), random forest, extreme gradient boosting (xgboot) and neural networks with extreme learning machine training. In addition, HGB models were evaluated by combining logistic, xgboost and random forest. Results show that, for lower proportion of failures, simpler methods such as logistic and glmnet achieve best results. Whereas, for larger proportion of failures, xgboost, random forest and HGB achieve best results.

This paper is organized as follows. Proposed classification methods, anomaly score analysis, classification performance and the database description are presented in Section 2. Results are presented in Section 3. Discussion and conclusion are presented in Sections 4 and 5, respectively.

2. Materials and Methods

2.1. Basic concepts and notation

Following Friedman et al. (2001), let $\mathbf{x} = \{x_1, \ldots, x_p\}$ be a vector of p input variables, hereafter named as predictor variables. Given the predictor variables, the objective is to predict an output variable Y, hereafter named as the response variable. This problem is called *supervised learning* since both inputs and output are known in advance. In addition, let $\{y_i, \mathbf{x}_i\}_{i=1}^{N}$ be the observed predictor vector and response variables, where N represents the sample size. If the response variable comprises two-level categorical information, for example, $Y = \{A,\ B\}$, then the problem is also known as a *supervised classification problem* in which the response variable is regularly coded as a binary variable written as $Y = \{0,\ 1\}$.

2.2. The logistic regression model

The logistic regression model is a standard statistical model for classification. It assumes that the response random variable follows a Bernoulli distribution, $Y \sim Bernoulli(\mu)$, where $Y \in \{0,\ 1\}$ and μ is the probability that the random variable Y is equal to one, $P(Y = 1) = \mu$. If predictor variables are available, say $x_1, \ldots, x_p$, then the μ parameter can be written as a function of a linear predictor using the logistic function:

$$P(Y = 1|x_1, \ldots, x_p) = \frac{\exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}{1 + \exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)} \quad (1)$$

where $0 < P(Y = 1|x_1, \ldots, x_p) < 1$ and β's are the parameters of the model which are estimated using maximum likelihood. Further details about the logistic model are found in Dobson and Barnett (2008).

The coefficients associated with each predictor represent the effect of the predictor in increasing or decreasing the probability of $Y = 1$. Alternatively, Equation 1 can be written as:

$$\frac{P(Y = 1|x_1, \ldots, x_p)}{P(Y = 0|x_1, \ldots, x_p)} = \exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) \quad (2)$$

Equation 2 provides a proper interpretation for the parameters of the logistic model. If one unit is added to a specific predictor, say $x_j + 1$, then the ratio between $P(Y = 1)$ and $P(Y = 0)$ is multiplied by e^{β_j}, as shown in Equation 3.

$$\frac{P(Y = 1|x_j + 1)}{P(Y = 0|x_j + 1)} = \frac{P(Y = 1|x_j)}{P(Y = 0|x_j)} \times e^{\beta_j} \quad (3)$$

e^{β_j} is known as the Odds ratio. In practice, positive parameters increase the change of $P(Y = 1)$, whereas negative parameters increase the chance of $P(Y = 0)$. Therefore, a weak predictor has the Odds ratio close to one, or $\beta_j \approx 0$. Statistical inference provides P-values under the null hypothesis of $H_0 : \beta_j = 0$.

2.3. Lasso and Elastic-Net Regularized Generalized Linear Models

As mentioned, the parameters of the logistic regression model are estimated using maximum likelihood function. The Bernoulli log-likelihood function is shown in Equation 4.

$$\log Lik = -\sum_{i=1}^{N} \left[y_i \ln(1 + e^{-\mu_i}) + (1 - y_i) \ln(1 + e^{+\mu_i}) \right] \quad (4)$$

where $\mu_i = \frac{\exp(\mathbf{x}_i^T \boldsymbol{\beta})}{1+\exp(\mathbf{x}_i^T \boldsymbol{\beta})}$, $\mathbf{x} = (1, x_1, \ldots, x_p)$ is the predictor vector, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$ is the vector of parameters and N is the sample size. In general, the larger the number of predictors the more likely the data overfitting. Furthermore, standard statistical inference of the predictors are sensitive to multicollinearity among the predictors. See Nelder and Baker (1972) for further details. Adding a penalty function to Equation 4 helps minimize

the overfit of the data and improves the model prediction. The lasso (*least absolute shrinkage and selection operator*) method proposed by Tibshirani (1996), known as the l_1-norm, and the ridge regression methods(Hoerl and Kennard, 1970), also known as the l_2-norm, are the most widely used penalty functions. The basic difference between l_1-norm and l_2-norm is that when using the l_1-norm, some of the estimated parameters might be equal to zero, whereas when using the l_2-norm, the estimated parameters have reduced absolute values but they are unlikely to be equal to zero. Therefore, the l_1-norm achieves predictor selection. The l_1-norm penalty is written as $l_1 = \sum_{j=1}^{p} |\beta_j|$ and the l_2-norm penalty is written as $l_2 = \sum_{j=1}^{p} \beta_j^2$. See Friedman et al. (2001) for further details.

Zou and Hastie (2005) proposes the elastic net penalty function which combines the l_1- and l_2-norms as written in Equation 5.

$$P_\lambda(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left[\frac{1}{2}(1-\lambda)\beta_j^2 + \lambda|\beta_j| \right] \tag{5}$$

where λ is the regularization parameter, $0 \leq \lambda \leq 1$. From Equations 5 and 4, given a fixed value of λ, the parameters of the logistic model are estimated by minimizing the following objective function,

$$\begin{aligned} \hat{\boldsymbol{\beta}} = & arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left[y_i \ln\left(1+e^{-\mu_i}\right) + (1-y_i)\ln\left(1+e^{+\mu_i}\right) \right] + \\ & \sum_{j=1}^{p} \left[\frac{1}{2}(1-\lambda)\beta_j^2 + \lambda|\beta_j| \right] \end{aligned} \tag{6}$$

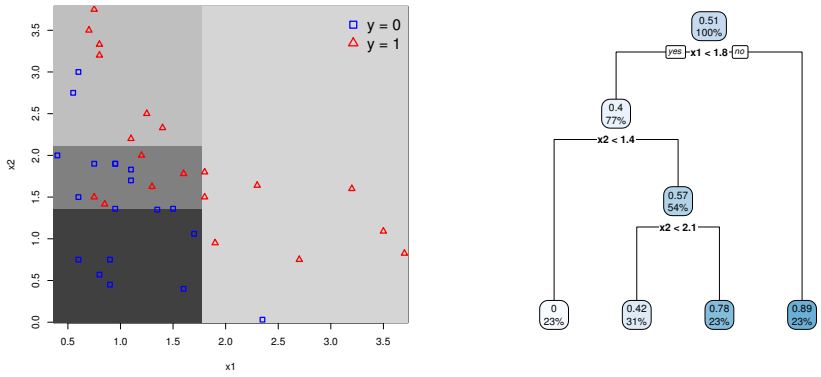
The glmnet package (Friedman et al., 2010) implements the elastic net penalty and estimates the λ parameter using cross-validation.

2.4. Classification Tree and Random Forest

The classification tree model (CART - *Classification And Regression Tree*) is a local mean model which creates partitions of the original data and estimates $P(Y = 1)$ using the local mean, $\bar{y}$ (Murthy, 1998; Breiman, 2017). The CART model can be represented as a binary tree. Each branch of the tree is generated by partitioning the data using one of the available predictors. Each predictor is evaluated separately. The predictor and its decision threshold are chosen based on error minimization, or maximum likelihood, among other criteria. A statistical decision rule, such as the Chi-Squared test (for classification) (Fisher, 1922), or a simple mean t-test (for

regression) (Snedecor and Cochran, 1989), among others, is used to stop the tree growing process. Briefly, if there is no statistical evidence that the average values between two new partitions are different, then the tree growth stops.

Figure 1 illustrates a classification tree model using simulated data with two predictors, x_1 and x_2. Figure 1(a) shows that x_1 and the threshold $\tau_1 = 1.8$ were primarily selected based on error minimization; thereby creating two partitions of the data set: the observations in which $x_1 < 1.8$ and the observations in which $x_1 \geq 1.8$. This procedure is known as the best split. After creating the first two branches of the tree, the partitioning algorithm is applied separately to each partition. In the right partition ($x_1 \geq 1.8$) there was no statistical evidence, using the available predictors, that a new partition creates two statistically different groups. In the left partition ($x_1 < 1.8$), statistical evidence based on minimization criteria identified x_2 and the threshold $\tau_2 = 1.4$ as the best predictor and cutting point, respectively. Therefore, a new partition of the data set was created. A final predictor and cutting point was detected in the subgroup data in which $(x_1 < 1.8) \cap (x_2 \geq 1.4)$. The final predictor space partition is shown in Figure 1(a). Figure 1(b) shows the decision tree. Percentages indicate the proportion of the database in each partition. Local means are indicated above the percentages.



(a) Generated partitions using CART. (b) Classification tree.

Figure 1: Illustration of CART method using simulated data.

Random Forests (Breiman, 2001) are combinations of CART models. Each tree or CART model is adjusted using a random sample of the pre-

dictors. For example, suppose m predictors are available, then k predictors from m ($k << m$) are randomly chosen. These k predictors are used to fit a CART model. In sequence, this procedure is repeated n times. Therefore, a total of n CART models (or trees) are available. The outcome of the Random Forest is a combination of the outcome of each CART model. Either, the mean or the median statistic of the n CART models are the most common aggregation statistics. In addition, the average contribution to the maximization of the likelihood for each predictor is calculated. This statistic, known as the *feature importance*, is used to identify the most important predictors.

2.5. Isolation Tree and Isolation Forest

Although isolation trees (iTree) and CART models are based on binary trees, the iTree aims at detecting data anomalies using an unsupervised approach, i.e, there is no response variable Y. The iTree algorithm creates binary trees by randomly choosing x variables, or features, and randomly choosing thresholds τ. The iTree algorithm requires one parameter which is the maximum height of the tree l. The basic iTree algorithm, adapted from Liu et al. (2008), is shown below.

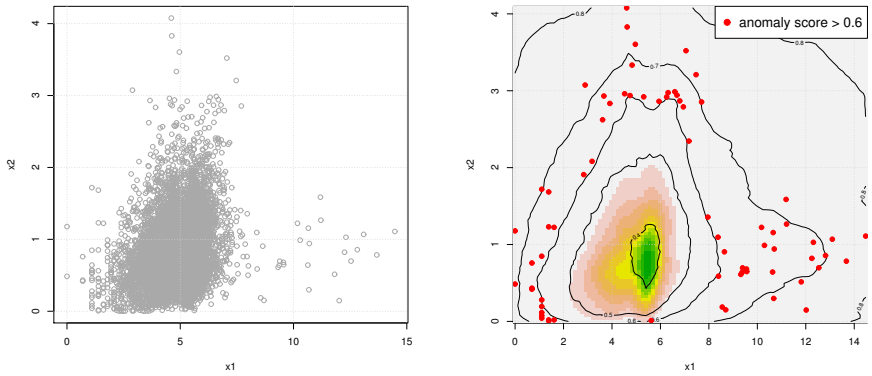
Algorithm 1 iTree(X,e,l)

Input: X input data, e current tree height, l height limit
Output: an iTree
  if $e \geq l$ or $|X| \leq 1$ **then**
    return exNode$Size \leftarrow |X|$
  else
    let Q be a list of variables in X
    randomly select a variable $q \in Q$
    randomly select a split point τ from max and min values of variable q in X
    $X_l \leftarrow \text{filter}(X, q < \tau)$
    $X_l \leftarrow \text{filter}(X, q \geq \tau)$
    inNode{Left $\leftarrow$ iTree($X_l, e+1, l$),
           Right $\leftarrow$ iTree($X_r, e+1, l$)
           SplitAtt $\leftarrow$ q,
           SplitValue $\leftarrow \tau$}
  end if

Briefly, the algorithm randomly creates partitions in the data until the maximum height parameter of the tree is reached, or stops earlier if, in each leaf of the tree, there is only one observation. Similar to random forests, a set of isolation trees is created by randomly selecting subsets of data and

fitting a new iTree to each subset. After creating a large number of iTrees, hereafter known as isolation Forest, or simply iForest, the complete database is presented to each iTree. For each observation, the path length for each iTree is estimated and a final anomaly statistic is calculated by averaging the path length among the iTrees. Anomalies are those observations with short average path lengths on the iTrees. Furthermore, Liu et al. (2008) proposes an anomaly score based on the average path length. The anomaly score lies between zero and one. If the anomaly score is greater than 0.6 then the observation is classified as an anomalous observation. It is worth mentioning that Domingues et al. (2018) claims that iForest is an efficient method to identify outliers.

Figure 2 illustrates the use of the iForest using real data. The scatterplot of the two features (x_1 and x_2) are shown in Figure 2(a). Figure 2(b) shows the kernel density of the data set and the points identified as anomalies. According to Domingues et al. (2018), anomalous points have anomaly scores greater than 0.6. It can be seen that anomalous points are located in a low density region, which is identified using iForest without requiring any parametric assumption about the underlying statistical density distribution of the data.



(a) Scatterplot of the data.

(b) Kernel density and observations identified as anomalies using iForest.

Figure 2: Illustration of the anomaly score using iForest.

The iForest has been applied in anomaly detection applications (Puggini and McLoone, 2018). However, as opposed to detect anomalous data, the present work proposes the use of the anomaly score as a new predictor

variable for failure detection. This is because, in failure detection problems, observations located in low density regions may represent observations under extreme conditions and, therefore, more likely to failure.

2.6. eXtreme Gradient Boosting

The e**X**treme **G**radient **B**oosting, or xgboost, is a fast implementation of the gradient boosting decision tree algorithm (Friedman, 2001). xgboost uses more regularized model formalization to control overfitting, which gives it better performance. The general gradient boosting algorithm is presented below. Let Equation 7 be an additive function,

$$F(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_{m=1}^{M}) = \sum_{m=1}^{M} \beta_m h(\mathbf{x}, \mathbf{a}_m) \tag{7}$$

where $h(\mathbf{x}, \mathbf{a}_m)$ is a simple parameterized function of the input variable $\mathbf{x}$. For example, $h(\mathbf{x}, \mathbf{a}_m)$ may represent a classification tree in which $\mathbf{a}_m$ is the parameter vector of the m-th classification tree. Friedman (2001) proposes the following Gradient Boost algorithm to fit Equation 7 using a training set.

Algorithm 2 Gradient Boost

1: $F_0 = arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2: **for** $m = 1$ to M **do**
3: $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$, $i = 1, \ldots, N$.
4: $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2$
5: $\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L\left(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m)\right)$
6: $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7: **end for**

where $L(y_i, F(\mathbf{x}_i))$ is the loss function which may represent the minimum least square function or the negative likelihood function.

Briefly, the Gradient Boost algorithm applies a greedy stagewise adjustment to function $F(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_{m=1}^{M})$, which can be written as $F_m(\mathbf{x}) = F_{m-1}(x) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$. At each m step, function $F_m(\mathbf{x})$ is updated in order to minimize the loss function $\sum_{i=1}^{N} L(y_i, F_m(\mathbf{x}_i))$.

xgboost (Chen et al., 2018) proposes a general framework for the gradient boost algorithm in order to improve optimization. The basic idea is to take the Taylor expansion of the loss function up to the second order and, therefore, approximate the minimization of the loss function as a minimum

squared error problem. In addition, the objective function includes a regularization term, or a model complexity term, thus creating a structure score. The structure score is used to optimize the size of classification trees, which is the standard classification model used by xgboost. Consequently, using xgboost, feature importance statistics are also estimated for each predictor.

2.7. Neural Networks

The general structure of a two-layer perceptron neural network with a linear output is shown in Equation 8

$$g(\mathbf{x}) = \sum_{h=1}^{H} w2_h \cdot \phi \left(\sum_{j=1}^{N} w1_{jh} \cdot x_j + b1_h \right) + b2 \tag{8}$$

where H is the number of neurons in the hidden layer, ϕ is the activation function, w_2, b_2, w_1 and b_1 are the output and the input weights, respectively; and N is the input ($\mathbf{x}$) dimension. Equation 8 can be represented as a network, shown in Figure 3. Neural networks are universal approximators. It has been shown that feed forward networks with a single hidden layer and with a finite number of neurons can approximate continuous functions (Gybenko, 1989). Nevertheless, neural network training may be time consuming.

Huang et al. (2006) proposes the e**x**treme **l**earning **m**achine (**elm**) algorithm for single-hidden layer feed forward neural networks. The algorithm creates random values for the input weights of the neural networks. For instance, $w1$ and $b1$ weights are randomly generated from a standard normal distribution. Therefore, Equation 8 reduces to a linear model, as shown in Equation 9.

$$g(\mathbf{x}) = \sum_{h=1}^{H} w2_h \cdot \phi_h + b2 \tag{9}$$

where $\phi_h = \phi \left(\sum_{j=1}^{N} w1_{jh} \cdot x_j + b1_h \right)$. The remaining parameters of the neural network, $w2_h$ and $b2$, are estimated using minimum least squares. The **elm** neural network requires more hidden neurons as compared to neural networks using standard gradient based algorithms. In addition, standard regularization techniques for linear models, such as the elasticnet (Zou and Hastie, 2005), described in Section 2.3, can be combined with the **elm** algorithm. Huang et al. (2006) shows that **elm** achieves much faster convergence of the neural network as compared to standard neural network training algorithms.

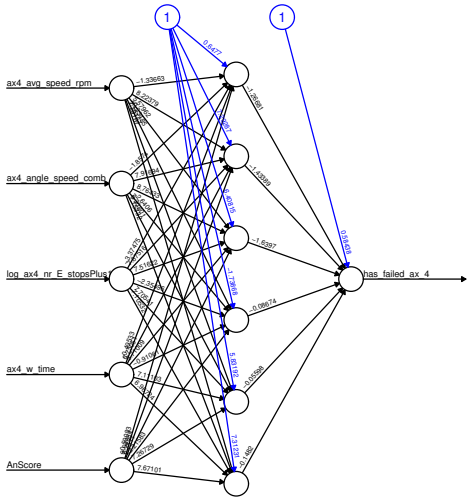


Figure 3: Two layer perceptron neural network with five inputs, six hidden neurons and one output neuron.

2.8. Hybrid gradient boosting

The proposed *hybrid gradient boosting* (HGB) aims at creating a general framework for boosting a base line model, such as the standard logistic regression model. In order to boost the base line model, non-linear models such as random forest, xgboost, extreme learning machines, or combined models, can be applied. For instance, random forests or xgboost models can be applied in order to estimate the feature importance and, therefore, to identify additional variables which are used to improve prediction.

The HGB approach is based on the exponential family distribution used in the generalized linear models (Nelder and Baker, 1972). A random variable Y belongs to the exponential family distribution if its density distribution can be written as

$$\log f_Y(y|\theta,\phi) = \frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi) \tag{10}$$

where θ is the canonical parameter, ϕ is the dispersion parameter, and $b(\theta)$, $c(y,\phi)$ and $a(\phi)$ are functions related to the density distribution of Y. The exponential family distribution has very interesting properties, such as $E(Y) = b'(\theta)$ and $Var(Y) = a(\phi)b''(\theta)$. In general, a linear predictor can be incorporated into the exponential distribution by assuming $\theta(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta}$,

where $\mathbf{x}^T\boldsymbol{\beta} = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$ is the linear predictor. Density distributions such as Normal, Poisson, Binomial, Bernoulli, Multinomial and Gamma can be written using equation 10. Further details about exponential family distribution and generalized linear models are found in Nelder and Baker (1972), Dobson and Barnett (2008), and elsewhere.

Following Friedman (2001), we propose an additive expansion for the canonical parameter $\theta_{(\mathbf{x})}$, shown in Equation 11.

$$\theta_{(\mathbf{x})} = \sum_{m=0}^{M} \theta_m(\mathbf{x}; \boldsymbol{\beta}_m) \tag{11}$$

For instance, $\theta_0(\mathbf{x}; \boldsymbol{\beta}_0)$ represents the standard linear predictor, i.e., the logistic regression solution; whereas $\theta_1(\mathbf{x}; \boldsymbol{\beta}_1), \ldots, \theta_M(\mathbf{x}; \boldsymbol{\beta}_M)$ may represent different models such as classification and regression trees and neural networks, among others. $\boldsymbol{\beta}_m$ represents the vector of parameters associated with each model or *boost*.

Using the exponential family representation and the additive expansion shown in Equation 11, the HGB algorithm is presented below.

Algorithm 3 Hybrid gradient boosting

1: $\theta_0 = arg\max_\theta \log Lik(\mathbf{y}, \theta_{(\mathbf{X})})$
2: **for** $m = 1$ to M **do**
3: $\quad \tilde{y}_i = -\left[\frac{\partial \log Lik(y_i, \theta_{(\mathbf{x}_i)})}{\partial \theta_{(\mathbf{x}_i)}}\right]_{\theta_{(\mathbf{x})} = \theta_{m-1(\mathbf{x})}}, \; i = 1, \ldots, N.$
4: $\quad \boldsymbol{\beta}_m = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left[\tilde{y}_i - h_m(\mathbf{x}_i; \boldsymbol{\beta})\right]^2$
5: $\quad \theta_{m(\mathbf{x})} = \theta_{m-1(\mathbf{x})} + h_m(\mathbf{x}; \boldsymbol{\beta}_m)$
6: **end for**

Line 1 represents the base line model, which is adjusted using maximum likelihood. Using the exponential family representation, Line 3 can be written as $\tilde{y}_i = y_i - E(Y_i|\mathbf{x}_i, \theta_{m-1(\mathbf{x}_i)})$. Line 4 shows that the m-boost model $h_m(\mathbf{x}_i; \boldsymbol{\beta})$ is fitted using a least square minimization problem. Therefore, $h_m(\mathbf{x}; \boldsymbol{\beta})$ comprises a non-linear regression model. It is worth mentioning that the additive expansion shown in Equation 11 can generate positive or negative values, as expected using the exponential family. Nonetheless, the estimated mean is $E(Y_i|\mathbf{x}_i, \theta_{m(\mathbf{x}_i)}) = b'(\theta_{m(\mathbf{x}_i)})$. For instance, if Y follows a Bernoulli distribution, then $b'(\theta_{m(\mathbf{x}_i)})$ is the logistic function.

2.9. k-fold cross validation

According to Breiman et al. (2001), *cross-validation is a natural route to the indication of quality of any data-deriving quantity.* Prediction accuracy

of the proposed statistical and machine learning models was estimated using a 10-fold cross validation procedure, illustrated in Figure 4. Initially, the original database is randomly partitioned into 10 disjoint folds. In sequence, the database and the respective partitions are replicated 10 times. Each replicate represents the original database. For each replicate, one fold is the test set and the remaining 9 folds are the training set. Therefore, 10 models are adjusted and 10 predictions are generated. The resulting prediction for each fold is combined into a new database, which is exactly the same size as the original data. From the predicted database, a final test statistic, or prediction statistic, is calculated. In order to account for the randomness of the 10-fold partition process, the entire procedure is repeated, say 10 times, and the average prediction statistic is evaluated.

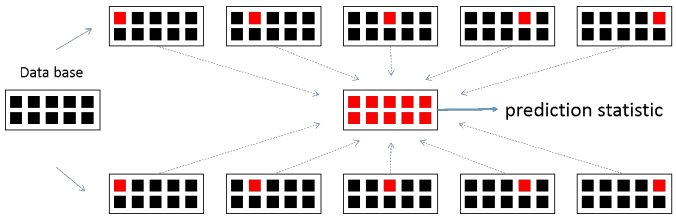


Figure 4: 10-fold cross validation procedure.

The k-fold cross validation procedure, illustrated in Figure 4, has advantages and limitations. The main advantage is that the prediction statistic is estimated using the entire database. The main limitation is that each evaluated model must be adjusted to each replication; therefore, each model is adjusted k-times. If k equals the sample size, then the k-fold cross validation is known as the *leave-one-out* cross validation. The k-fold cross validation procedure is a standard procedure for prediction accuracy estimation.

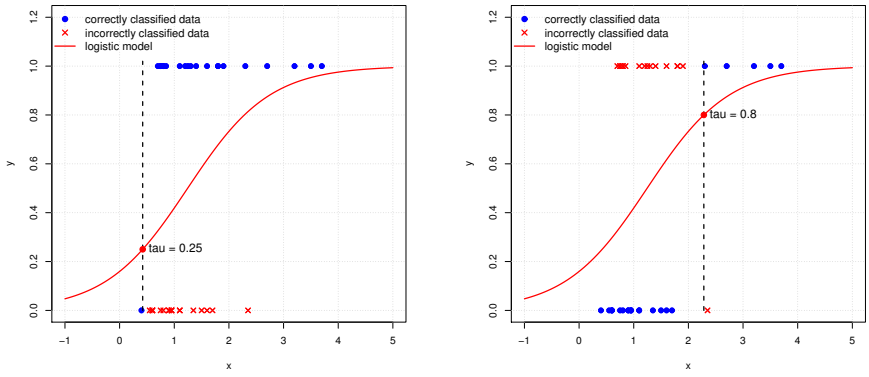
2.10. Sensitivity and Specificity analysis

Sensitivity and specificity statistics are widely applied in medical tests (Altman, 1994). In binary classification tests, these statistics are used as performance measures (Christopher, 2011). Sensitivity, also known as true positive rate, measures the proportion of observed positive classes ($y_i = 1$) which is correctly identified by the classification model. Specificity, also known as true negative rate, measures the proportion of observed negative classes ($y_i = 0$) which is correctly identified by the classification model. Thus, sensitivity and specificity are performance measures related to observed positive and negative classes, respectively.

In general, statistical classification models and machine learning models estimate the probability of a new observation being classified as a positive class given a vector of predictors, $\hat{P}(Y = 1|x_1, \ldots, x_p) < 1$, the analyst must decide whether the event $Y = 1$ will eventually happen. Thus, a threshold τ is selected and the decision rule, shown in Equation 12, is applied.

$$\hat{y}_0 = \begin{cases} 1, & \text{if } \hat{P}(Y = 1|\mathbf{x}_0) \geq \tau \\ 0, & \text{if } \hat{P}(Y = 1|\mathbf{x}_0) < \tau \end{cases} \tag{12}$$

Both sensitivity and specificity statistics are affected by threshold τ. Figure 5 illustrates sensitivity and specificity estimates using simulated data and a logistic model with one predictor (x). Figure 5(a) shows the model outcome if threshold $\tau = 0.25$ and Equation 12 are applied. All observed responses in which $y_i = 1$ are correctly classified. Therefore, the sensitivity statistic is 100%. On the contrary, most of the observed responses in which $y_i = 0$ are incorrectly classified. Therefore, the specificity is close to 0%. If threshold $\tau = 0.80$ is applied, then the sensitivity decreases and the specificity increases as shown in Figure 5(b).

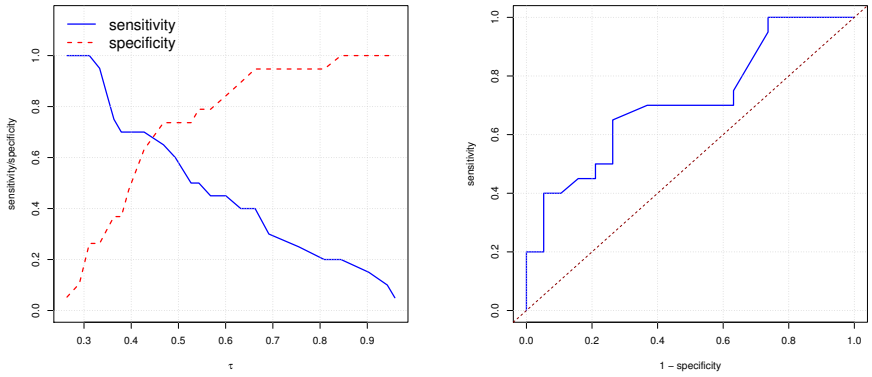


(a) $\tau = 0.25$. (b) $\tau = 0.8$.

Figure 5: Sensitivity and specificity analysis for varying thresholds.

Figure 6(a) illustrates the trade-off between sensitivity and specificity. Smaller values of τ increase sensitivity and decrease specificity; whereas larger values of τ decrease sensitivity and increase specificity. Consequently, there is an optimal value of τ^* in which $sensitivity = specificity$. Figure 6(b) shows the $sensitivity$ on the y-axis against $1 - specificity$ on the x-axis, known as the receiver operating characteristic (ROC) curve (Bradley,

1997; Fawcett, 2004). The area under the ROC curve (AUC area under the curve) is widely used as a classification performance statistic. Optimal classification models have large AUC. In addition to the AUC, this work proposes the use of the intersection point between sensitivity and specificity as a classification performance statistic, hereafter named as classification rate.



(a) Sensitivity and Specificity.

(b) ROC curve.

Figure 6: Sensitivity and specificity analysis for varying thresholds and ROC curve.

2.11. ABB robot data

ABB robotics is a leading supplier of industrial robots. The ABB data available for the analysis comprise approximately 6,000 observations, one observation per unique robot. For each robot, data for the movements of the six joints are available. In total, 275 variables are available in the database for each robot. Variables can be categorized into two major groups: (a) operational features and (b) robot type. Variables in the operational features group can be further divided into joint operational features and robot operational features. For each joint, the following information is available: average speed (avg_speed), average torque (avg_torque), accumulated feature combining angle and speed (angle_speed_comb), accumulated absolute position (moved_distance), number of emergency stops (nr_E_stops), run time in hours (r_time), wait time in hours (w_time) and failed information (has_failed). The failed information comprises the presence ($y = 1$) or absence ($y = 0$) of failure in each joint. It is worth mentioning that the data do not represent random samples from the total fleet of robots. Therefore,

the observed proportion of failures may be different if another set of data is collected from a different group of robots.

Figure 7 illustrates the robotic arm and indicates the six joints. Joint 1 comprises the rotating base of the robot. Joint 2 comprises the horizontal movement of the arm set. Joint 3 is known as the *elbow* and comprises the vertical movement of the arm. Joint 4 comprises the forearm rotation movement. Joint 5 comprises the *wrist* movement of the robotic arm and joint 6 comprises the movement of the hand tool, which may be a welding tool, grip tool, among others.

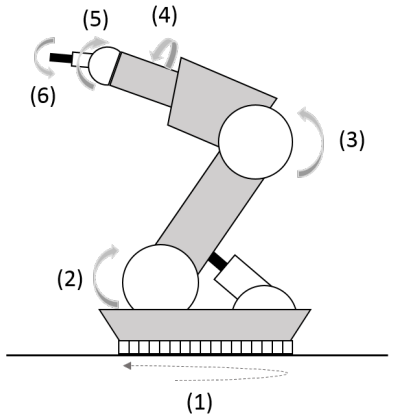


Figure 7: Configuration of the robotic arm.

Figure 8 illustrates the pairwise correlation between joint operational features (or variables). The narrower the ellipse the larger the correlation. Blue ellipses have positive slopes and indicate positive correlations, whereas red ellipses have negative slopes and indicate negative correlations. White circles indicate weak correlations. Figure 8 shows clusters of correlated variables in the center and at the top of the figure.

The robot operational features comprise the following variables: time when the system was first started (date_start), time of the last service (date_service), production time in hours (p_time), robot type (r_type) and accumulated energy consumption (sys_energy). A new variable was created by dividing the accumulated energy consumption by the production time. This variable, named SysDivPtime, represents the energy consumption rate by production hour.

In addition to the joint variables, previously mentioned, histogram information for each joint is also available. This information represents sequen-

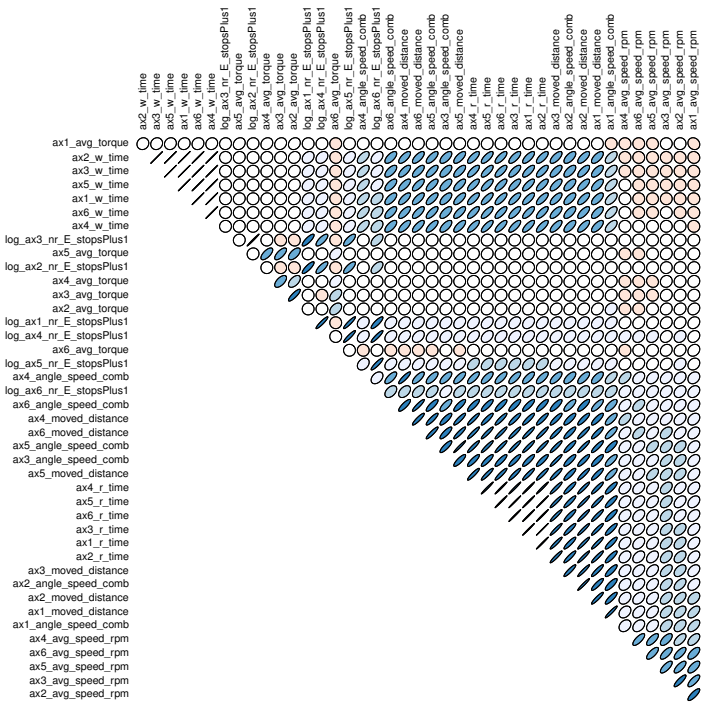


Figure 8: Correlation matrix using average torque, run time, average speed, combined angular position and speed, number of emergency stops, moved distances and wait time for joints 1 to 6.

tial measures for angles, torque and speed; it comprises relative frequencies in pre-specified range intervals, or bins. For instance, suppose speed information is available for joint 1. The variable ax1_avg_speed is the average speed. The entire range of observed values for ax1_speed are divided into a series of adjacent and non-overlapping intervals (bins). Therefore, ax1_hist_speed_bin_0 is the lower bound of the observed relative frequency of speed in joint 1. There are 216 histogram variables in the database.

The proportion of failure for each joint is 0.2% for joint 1, 1.1% for joint 2, 1.6% for joint 3, 6.3% for joint 4, 2.4% for joint 5 and 5.7% for joint 6. Joints 4 and 6 present the largest proportion of failures as compared to the remaining joints in the dataset. Each observation, i.e., each robot, can be classified into one of 5 different robot type categories. 58.9% of the observations comprise robot type 1, 33.1% comprise robot type 2, 7.9% comprise robot type 3 and the few remaining observations comprise robot

types 4 or 5. Empirical analysis, using the proportion of failure for each robot type, indicates the robot type as a potential predictor variable, which is included as a dummy variable.

Data consistency analysis identified a few negative observations for the number of emergency stops. Negative values were replaced by missing data (NA Not Available). In addition, the number of emergency stops presented heavy-tailed empirical distributions with many extreme values. Thus, the logarithm of the number of emergency stops plus one was used as the new predictor variable, as opposed to the raw number of emergency stops.

2.12. Proposed classification models

Six classification models are initially proposed: the standard logistic regression model (logistic); the logistic regression model using elasticnet penalty (glmnet); two neural network models with 75 and 350 hidden nodes, respectively, using a combined elm and elastic net algorithm (glmNNET75 and glmNNET350); the random forest model (RandomForest); and, the extreme gradient boosting model (xgboost).

The anomaly score is estimated using only local joint production variables, i.e., average speed in revolutions per minute (avg_speed_rpm), average torque (avg_torque), accumulated feature combining angle and speed (angle_speed_comb), accumulated absolute position (moved_distance), logarithm of the number of emergency stops plus one (log_nr_E_stopsPlus1), run time in hours (r_time) and wait time in hours (w_time). For instance, to estimate a classification model for joint 4, only local joint production variables are used to calculate the anomaly score. Therefore, separate anomaly scores are calculated for each joint. The anomaly score is sensitive to the number of variables and the number of trees (Liu et al., 2008). Stable anomaly scores are generated using a smaller set of variables and a larger number of trees. The anomaly scores are estimated setting the number of trees equal to 1,000.

The standard logistic regression model is estimated using a white-box modeling approach. The local joint production variables and the anomaly score are evaluated. Empirical evidence using frequency plots shows that the proportion of failures is larger for robot types 2 and 3. Therefore, two dummy variables are also included in the logistic regression model. Variables are selected based on statistical inference, i.e., the P-value. Variables which are not statistically significant at the $\alpha = 0.05$ level are gradually removed from the model. Thus, the final model comprises statistically significant variables.

The glmnet, glmNNET75, glmNNET350, RandomForest and xgboost constitute the black box models. Even though glmnet applies a logistic function, it achieves variable selection, as presented in Section 2.3. These models are estimated and evaluated using the 10-fold cross validation procedure presented in Section 2.9. Initially, all 276 available predictors, including the anomaly score, are used.

3. Results

Table 1 presents the estimated logistic regression coefficients using the total database for failure detection in joint 4. Joint 4 has the largest number of failures (6.3%). The local predictors (joint level), the anomaly score and the robot types 2 and 3 are evaluated. Results show that average speed, combined angle and speed, number of emergency stops and wait time have positive coefficients, as expected. On the contrary, the anomaly score presents a negative coefficient. Thus, the model indicates that the lower the anomaly score, the larger the chance of a failure. As previously described, the anomaly score represents the density of the data. Larger anomaly scores indicate observations in lower density regions. In an industrial setting, large anomaly scores comprise observations under non-normal operating circumstances, as previously illustrated in Figure 2. However, the database comprises a time snapshot of the operating robots. It is expected that, eventually, all robots will present failure. Furthermore, the average speed, combined angle and speed, number of emergency stops and wait time also capture non-normal conditions. The anomaly score coefficient points towards the large density region, i.e., the lower anomaly score. This result suggests that some of the failures are randomly scattered in the predictors space. Consequently, the larger the density, the larger the number of failures, as expected.

One may claim that the anomaly score is highly correlated to average speed, combined angle and speed, number of emergency stops and wait time. Therefore, results presented in Table 1 are subject to multicollinearity. Figure 9 shows the correlation plot between the predictors presented in Table 1. Pairwise correlations are represented as ellipses. The narrower the ellipse, the larger the correlation. Figure 8 shows that the anomaly score variable is positively correlated to combined angle and speed ($\rho = 0.6531$) and wait time ($\rho = 0.5599$). Nevertheless, a Variance Inflation Factor (VIF) analysis (Montgomery et al., 2012) shows that the largest VIF statistics are 2.0141 for the anomaly score and 2.8114 for combined angle and speed. In general, a critical value of 5 (VIF>5) or 10 (VIF>10) is used to indicate strong evidence

Table 1: Logistic regression results for failure detection in joint 4.

Coefficient	Estimate	Std. Error	P-value
Intercept	-2.3134	0.6778	0.0000
ax4_avg_speed_rpm	0.8162	0.1519	0.0000
ax4_angle_speed_comb	0.0587	0.0161	0.0003
log_ax4_nr_E_stopsPlus1	0.3137	0.0563	0.0000
ax4_w_time	0.2240	0.0498	0.0000
anomaly_score	-8.9441	1.9956	0.0000
robot type 2	0.3897	0.1164	0.0008
robot type 3	0.5704	0.1805	0.0016

of multicollinearity among the predictor variables. Therefore, there is no evidence of strong correlation between the anomaly score and the evaluated predictors. As previously described, the anomaly score measures the density of the data rather than a linear combination of the predictors. The logistic regression model, using the total number of observations, achieves an AUC of 0.7083 and a classification rate of 0.6473.

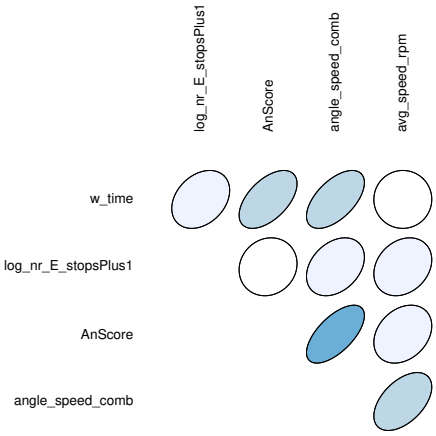


Figure 9: Correlation matrix using average speed, combined angle and speed, number of emergency stops, wait time and anomaly score for joint 4.

Figure 10 shows the statistically significant predictors, using the logistic regression model for each joint. It can be seen that average speed

(avg_speed_rpm) is the common predictor. For joint 1, which has the lowest proportion of failures, only average speed and accumulated absolute position (moved_distance) are statistically significant. Furthermore, for joints 2 to 6 the anomaly score is also a common predictor.

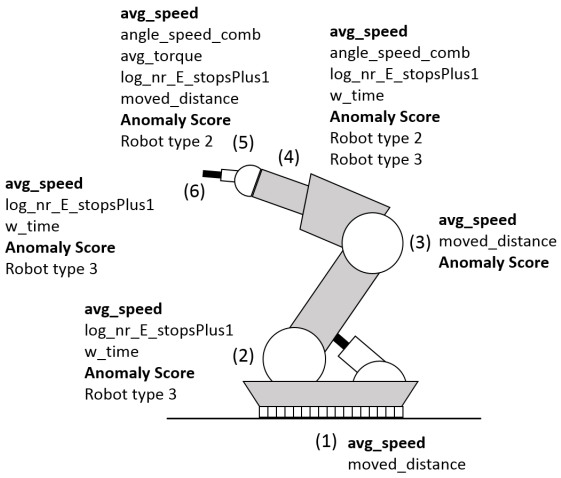
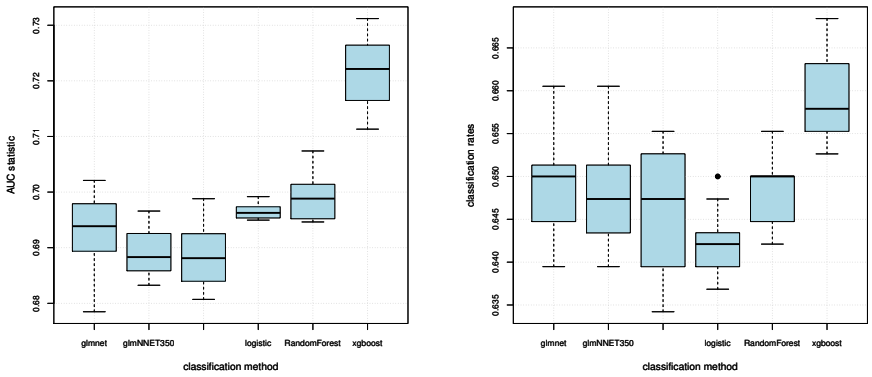


Figure 10: Statistically significant predictors using the Logistic model.

Figure 11(a) shows the boxplot of the AUC statistic for each classification method for joint 4. For comparison purposes, the logistic regression was included in the 10-fold cross validation analysis using the statistically significant predictors. Results show that xgboost achieves the largest AUC, followed by Random Forest. It is worth mentioning that the logistic regression achieved the lowest standard deviation, thereby achieving the greatest precision. Figure 11(b) shows the boxplot of the classification rate statistic for each classification method. Results also show that xgboost achieves the largest classification rate, followed by glmnet and Random Forest.

Table 2 summarizes AUC, classification rates and computing times for all evaluated methods used to predict failure in joint 4. Best results are shown in bold type. It can be concluded that xgboost achieves the best AUC and classification results, whereas the logistic regression achieves lower AUC standard deviation. Regarding computing time, the logistic regression is the fastest method, followed by xgboost and glmNNET75.

It is worth comparing the logistic regression results using the white-box and the grey-box approaches. The white-box approach achieved an AUC of 0.7083 and a classification rate of 0.6473. The grey-box approach, using the white-box predictors and the 10-fold cross-validation, achieved an average



(a) (b)

Figure 11: AUC (Area Under the Curve) statistic (a) and classification rates (b) for failure detection in joint 4 using the selected classification methods in 10 runs.

Table 2: AUC, classification rates and computing times, evaluated in 10 runs of the selected classification methods for failure detection in joint 4. Best results and minimum standard deviations are indicated as bold type.

	AUC		Classification rates		Computing time (min)	
method	mean	sd	mean	sd	mean	sd
glmnet	0.6929	0.0074	0.6486	0.0064	32.2128	4.8953
glmNNET350	0.6891	0.0047	0.6481	0.0067	103.4436	184.2138
glmNNET75	0.6888	0.0062	0.6459	0.0076	13.1653	2.2524
logistic	0.6965	**0.0013**	0.6421	**0.0039**	**0.0138**	**0.0012**
Random Forest	0.6990	0.0043	0.6486	0.0040	65.9301	14.4743
xgboost	**0.7216**	0.0067	**0.6593**	0.0056	0.1468	0.0071

AUC of 0.6965 and an average classification rate of 0.6421, as presented in Table 2. It can be seen that the white-box results are slightly better than using cross-validation. However, as previously discussed, white-box statistics (AUC and classification rates) measure data fitting, while cross-validation statistics measure error prediction.

Table 3 shows the AUC, classification rates and computing times for failure detection in joint 6. Joint 6 has the second largest rate of failures (5.7%). The anomaly score was estimated using the local variables for joint 6. Results show that xgboot achieved the best AUC statistic, followed by Random Forest and glmnet. The xgboost achieved the best classification rate, followed by Random Forest and glmnet. The logistic regression achieved the

best computing time, followed by xgboot. In general, xgboot, Random Forest and glmnet achieved the best AUC and classification rates.

Table 3: AUC, classification rates and computing times evaluated using 10 runs of the selected classification methods for failure detection in joint 6. Best results and minimum standard deviations are indicated in bold type.

	AUC		Classification rates		Computing time (min)	
method	mean	sd	mean	sd	mean	sd
glmnet	0.7432	0.0029	0.6785	0.0031	48.4502	3.9208
glmNNET350	0.7292	0.0044	0.6613	0.0052	48.8511	14.8300
glmNNET75	0.7285	0.0047	0.6581	0.0058	13.0907	2.6724
logistic	0.7218	**0.0014**	0.6648	**0.0024**	**0.0147**	**0.0031**
Random Forest	0.7470	0.0039	0.6846	0.0039	57.5615	1.8703
xgboost	**0.7516**	0.0072	**0.6887**	0.0076	0.1463	0.0081

Table 4 shows the AUC, classification rates and computing times for failure detection in joint 5. Joint 5 has the third largest rate of failures (2.4%). The anomaly score was estimated using the local variables for joint 5. Results show that the glmNNET350 model achieved the best AUC statistic, followed by xgboost and glmNNET75. The xgboost achieved the best classification rate, followed by Random Forest and glmNNET350. The logistic regression achieved the best computing time, followed by xgboost.

Table 4: The AUC, classification rates and computing times, evaluated in 10 runs of the selected classification methods for failure detection in joint 5. Best results and minimum standard deviations are indicated in bold type.

	AUC		Classification rates		Computing time (min)	
method	mean	sd	mean	sd	mean	sd
glmnet	0.7772	0.0034	0.7075	0.0097	46.6676	2.8222
glmNNET350	**0.8134**	0.0044	0.7295	0.0087	79.9149	21.4667
glmNNET75	0.8118	0.0049	0.7260	0.0085	23.3417	5.1136
logistic	0.7118	0.0051	0.6500	**0.0051**	**0.0163**	**0.0024**
Random Forest	0.7991	**0.0032**	0.7397	0.0072	48.5733	12.7230
xgboost	0.8124	0.0086	**0.7404**	0.0075	0.1371	0.0095

Table 5 shows AUC, classification rates and computing time for failure detection in joint 3. Joint 3 has the proportion of failures of 1.6%. The anomaly score was estimated using the local variables for joint 3. Results show that the glmnet model achieved the best AUC statistic, followed by logistic and Random Forest. The glmnet achieved the best classification rate, followed by logistic and Random Forest. The logistic regression achieved the best computing time, followed by xgboost.

Table 5: The AUC, classification rates and computing times, evaluated in 10 runs of the selected classification methods for failure detection in joint 3. Best results and minimum standard deviations are indicated in bold type.

	AUC		Classification rates		Computing time (min)	
method	mean	sd	mean	sd	mean	sd
glmnet	**0.8672**	0.0049	**0.8111**	0.0068	120.5789	10.8479
glmNNET350	0.8138	0.0079	0.7535	0.0136	104.5456	18.0655
glmNNET75	0.8092	0.0071	0.7495	0.0080	36.4443	6.7516
logistic	0.8667	**0.0018**	0.8071	**0.0032**	**0.0142**	**0.0015**
Random Forest	0.8360	0.0111	0.7879	0.0117	47.5031	2.4083
xgboost	0.8074	0.0123	0.7778	0.0158	0.1385	0.0122

Table 6 shows AUC, classification rates and computing time for failure detection in joint 2. Joint 2 has the proportion of failures of 1.1%. The anomaly score was estimated using the local variables for joint 2. Results show that the glmnet model achieved the best AUC statistic, followed by logistic and xgboost. The glmnet achieved the best classification rate, followed by logistic and xgboost. The logistic regression achieved the best computing time, followed by xgboost.

Table 6: The AUC, classification rates and computing times evaluated in 10 runs of the selected classification methods for failure detection in joint 2. Best results and minimum standard deviations are indicated in bold type.

	AUC		Classification rates		Computing time (min)	
method	mean	sd	mean	sd	mean	sd
glmnet	**0.8745**	0.0094	**0.8299**	0.0104	218.7813	23.7175
glmNNET350	0.8229	0.0085	0.7567	0.0123	150.7080	16.2985
glmNNET75	0.8215	0.0124	0.7567	0.0142	47.2849	14.4059
logistic	0.8737	**0.0015**	0.8179	**0.0063**	**0.0168**	**0.0081**
Random Forest	0.8534	0.0119	0.7806	0.0101	37.4156	2.3232
xgboost	0.8605	0.0126	0.8134	0.0106	0.1322	0.0098

Table 7 shows the best three classification methods for each joint and each performance measure. Results do not include joint 1, due to the lower rate of failures. In general, there is no single method that achieved the best AUC or classification rate performance for all joints. In general, the black-box models performed better for larger rate of failure. For lower proportion of failures, the regularized logistic (glmnet) and logistic models performed better. The logistic and xgboost were the fastest methods in all joints. This is because these methods are compiled into lower level programming language, whereas the remaining method uses higher level language. The

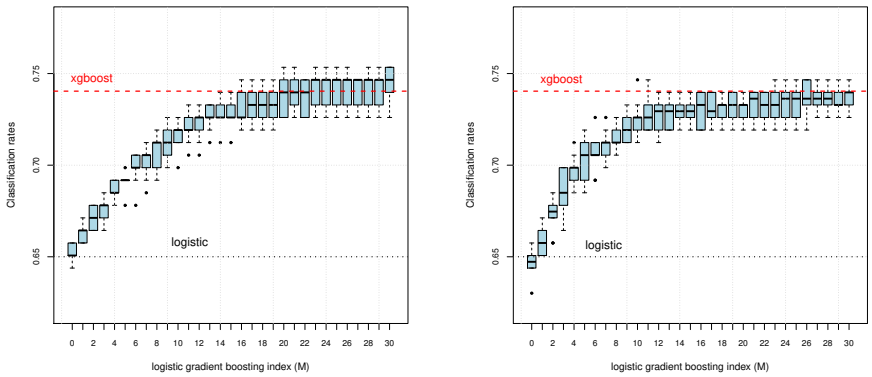
methods were evaluated using the i7 Intel core and the R language (R Core Team, 2017).

Table 7: Best classification models for failure detection in joints 2 to 6. Observed proportion of failures are indicated in parenthesis.

Joint	AUC	Classification rates	Computing time
2 (1.1%)	glmnet, logistic, xgboost	glmnet, logistic, xgboost	logistic, xgboost
3 (1.6%)	glmnet, logistic, Random Forest	glmnet, logistic, Random Forest	logistic, xgboost
4 (6.3%)	xgboost, Random Forest, glmnet	xgboost, Random Forest, glmnet	logistic, xgboost
5 (2.4%)	glmNNET350, xgboost, glmNNET75	xgboost, Random Forest, glmNNET350	logistic, xgboost
6 (5.7%)	xgboost, Random Forest, glmnet	xgboost, Random Forest	logistic, xgboost

Figure 12 shows HGB cross validation results for failure prediction in joint 5, in 10 runs. Joint 5 achieved the largest difference, of 9.04% between the logistic and the best classification rate result, using xgboost. Therefore, the HGB was applied to boost the logistic regression and to identify the additional predictors. Both xgboost and Random Forest were used to boost the logistic regression because they achieved the best classification rate results, as shown in Table 4. Figure 12(a) shows HGB classification rate results using logistic and xgboost. In general, after 20 boosts, the HGB model achieves the same classification rate result obtained using only xgboost. Figure 12(b) shows HGB classification rate results using logistic and Random Forest. In general, after 25 boosts, the HGB model achieves the same classification rate result obtained using only xgboost. Therefore, using xgboost, the HGB requires fewer boosts as compared to Random Forest. Nevertheless, HGB using either xgboost or Random Forest achieves similar final results.

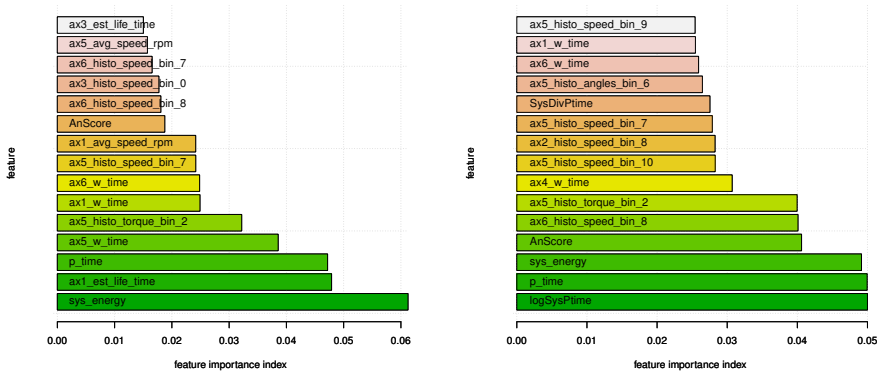
Figure 13 shows the additional predictors selected by xgboost and Random Forest using HBG. Predictors were sorted in decreasing order of feature importance. In general, predictors with larger feature importance are associated with global variables such as production time (p_time), consumed energy (sys_energy) or rate of consumed energy (logSysPtime). Remaining predictors are associate with relative frequency of speed and torque (histogram variables) in different joints. It is worth mentioning that both xgboost and Random Forest rely on random partitions of the predictors. Furthermore, most of the predictors are highly correlated. Therefore, feature importance results may change in different runs.



(a) Hybrid gradient boosting with Logistic and Xgboost.

(b) Hybrid gradient boosting with Logistic and Random Forest.

Figure 12: Classification rates results using hybrid gradient boosting and logistic regression as the base line model. Original logistic and xgboost results are shown as horizontal lines.



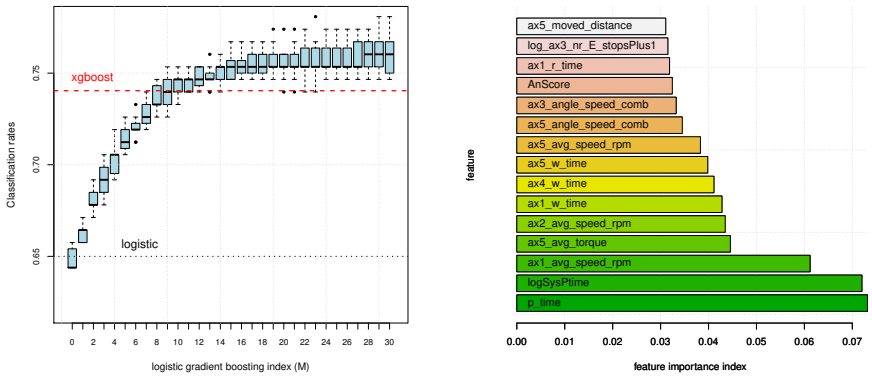
(a) Hybrid gradient boosting with Logistic and Xgboost.

(b) Hybrid gradient boosting with Logistic and Random Forest.

Figure 13: Feature importance analysis using hybrid gradient boosting with logistic regression as the base line model and xgboost.

HGB can also be used to boost a base line model, with a subset of the original data base. This is illustrated in Figure 14. As opposed to using the total of 276 predictors, the HGB was applied using logistic as the base line model and xgboost with a subset of 45 predictors. Local joint predic-

tors and robot operational features were used. Histogram predictors were not included. Results indicate that HGB may improve classification results if a smaller number of predictors is applied. Compared to Figures 12(a) and 12(b), the HGB, using a smaller number of predictors, achieved better classification results. Figure 14(b) shows the xgboost feature importance index. Results show that robot production time (p_time), rate of consumed energy (logSysPtime) and average speed on axis 1 (ax1_avg_speed_rpm) are the main additional predictors for improving failure detection in joint 5, using a non-linear black-box.



(a) Hybrid gradient boosting with Logistic and Xgboost.

(b) Hybrid gradient boosting with Logistic and Random Forest.

Figure 14: Feature importance analysis using hybrid gradient boosting, logistic regression as the base line model, xgboost and a smaller subset of predictors.

4. Discussion

There is a natural scheme for data analysis, which is data consistency analysis followed by data modeling. Data consistency analysis, also known as *data cleaning*, comprises outlier analysis and statistical description of all available variables. Data consistency also includes careful evaluation of the data generation process and checking whether the available information comprises the current beliefs of engineers and analysts with respect to the problem at hand. This is one of the most important steps in data modeling.

Although not reported in this work, the available data set was primarily evaluated using simple descriptive statistics, histograms, correlation matrices, scatter plots, among other exploratory statistical techniques. Further-

more, these results were presented to, and systematically discussed with, engineers and analysists. Future work aims at evaluating additional information in the model. Nevertheless, current findings provided some important insights with respect to failure detection in robotic arms. First, local information is a strong predictor. For instance, the logistic regression achieved a classification cross validation performance of 66.48% for joint 6 using only local variables, while the xgboost achieved an additional improvement performance of 2.39% using all 276 variables. For joint 2, glmnet, logistic and xgboost achieved very similar results, providing evidence that failures in joint 2 are driven mostly by local variables. Second, there is evidence that information from different joints improves prediction for a specific joint. For instance, logistic regression classification rates using local information for joint 5 is 65%, whereas xgboost achieves 74% using information from all joints.

The proposed *hybrid gradient boosting* model presented very interesting results by creating a black-box model on top of a white-box model, thereby improving cross-validation performance and separating predictors into white- and black-box contributors. In the present study, both xgboost and random forest achieved best results for joints having large proportions of failures. An interesting finding is the improvement of HGB results if a smaller data set is applied in the boosting procedure. This might be related to the tree-based models. As indicated by Liu et al. (2008), the isolation forest performs better with a smaller number of predictors. This may be the case for both the xgboost and random forest models.

A critical point, in assessing classification performance, is the selection of the appropriate statistical measure. Although AUC is a common choice for evaluating classification models, in practice, predictive classification error or classification rate is a suitable choice. As previously discussed, most classification models estimate the chance of failure, from which the analyst must decide whether the failure will happen based on a threshold. The AUC statistic is robust for the threshold choice. However, results show major differences between AUC and classification rate. For instance, the xgboost AUC result for joint 6 is 75.16%, whereas for the classification rate it is 68.87%. Therefore, it worth mentioning that AUC should not be used as a proxy for predictive classification rates. Results show that AUC and classification rates are different classification measures, which may or may not achieve similar results.

Regarding the logistic regression results, one may claim that white-box models, such as the logistic regression, do not require cross-validation evaluation because they rely on consistent statistical theory/statistical inference.

In general, this statement is true. Nonetheless, results show that, in general, the white-box model (logistic) resulted in prediction errors. However, for joint 2, both glmnet and logistic achieved the best predictive results, as shown in Table 6. Furthermore, joint 2 has a lower rate of failures (1.1%). This may suggest that, for rare failure events, a simpler white-box model may outperform sophisticated black-box models.

It is worth mentioning that the logistic model can be estimated using a black-box approach. For instance, one may evaluate hundreds or thousands of different logistic models and select the few models with the best error prediction. For example, assuming a fixed number of variables, say 5 variables, and assuming a database with only 50 variables, there are 2,118,760 different models. In this case, a white-box approach saves time and may achieve reasonable results.

5. Conclusion

The present work has proposed classification models for failure detection in industrial robotic arms, using statistical and machine learning methods. The standard logistic regression model was adjusted using the white-box approach, i.e., with the available data set and selecting variables using statistical inference. In addition, four machine learning models were adjusted using the black-box approach, i.e., with all available variables and cross-validation performance. Finally, a *hybrid* approach, using both white- and black-box models, was evaluated. The white-box component indicates the importance of each variable, whereas the black-box component further improves the error prediction.

As expected, the different methods performed differently for the different joints. In general, simple methods achieved better performance for joints with lower rates of failures using local joint information. Furthermore, using anomaly scores as an additional variable helped to improve classifications.

It is worth mentioning that the construction of white-box, black-box and hybrid models for failure detection must rely on data analysis and technical knowledge of the production process. Therefore, decisions, with respect to selection of variables for anomaly score estimation, were based on local joint information previously discussed with engineers and analysts.

In general, the development of failures and wear is dependent on the accumulated usage of the robot. By including several sensors and providing different measurements per robot, a natural step is to correlate these measurements with the failures and, therefore, develop predictive failure classification models. The current database provided some very interesting

insights into the main drivers of failures in robotic arms. However, we believe more data are required in order to continue to improve the classification results.

Acknowledgements

The authors thank CISB Swedish-Brazilian Research and Innovation Center, Saab AB and the VINNOVA sponsored Competence Center LINK-SIC for financial support. We also appreciated fruitful comments and discussions of the anomaly study group coordinated by Professors Eric Frisk and Mattias Krysander.

References

Altman, D. G., 1994. Statistics notes: Diagnostic tests 1: sensitivity and specificity. British Medical Journal 308, 1552.

Bradley, A. P., 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition 30 (7), 1145–1159.

Breiman, L., 2001. Random forests. Machine learning 45 (1), 5–32.

Breiman, L., 2017. Classification and regression trees. Routledge.

Breiman, L., et al., 2001. Statistical modeling: The two cultures (with comments and a rejoinder by the author). Statistical science 16 (3), 199–231.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., 2018. xgboost: Extreme Gradient Boosting. R package version 0.6.4.1.
URL `https://CRAN.R-project.org/package=xgboost`

Christopher, M. B., 2011. Pattern Recognition and Machine Learning. Springer-Verlag New York.

Dobson, A. J., Barnett, A., 2008. An introduction to generalized linear models. CRC press.

Domingues, R., Filippone, M., Michiardi, P., Zouaoui, J., 2018. A comparative evaluation of outlier detection algorithms: Experiments and analyses. Pattern Recognition 74, 406–421.

Fawcett, T., 2004. Roc graphs: Notes and practical considerations for researchers. Machine learning 31 (1), 1–38.

Fisher, R. A., 1922. On the interpretation of χ 2 from contingency tables, and the calculation of p. Journal of the Royal Statistical Society 85 (1), 87–94.

Friedman, J., Hastie, T., Tibshirani, R., 2001. The elements of statistical learning. Vol. 1. Springer series in statistics New York.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of statistical software 33 (1), 1.

Friedman, J. H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189–1232.

Gybenko, G., 1989. Approximation by superposition of sigmoidal functions. Mathematics of Control, Signals and Systems 2 (4), 303–314.

Hoerl, A. E., Kennard, R. W., 1970. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12 (1), 55–67.

Huang, G.-B., Zhu, Q.-Y., Siew, C.-K., 2006. Extreme learning machine: theory and applications. Neurocomputing 70 (1-3), 489–501.

Liu, F. T., Ting, K. M., Zhou, Z.-H., 2008. Isolation forest. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. IEEE, pp. 413–422.

Montgomery, D. C., Peck, E. A., Vining, G. G., 2012. Introduction to linear regression analysis. Vol. 821. John Wiley & Sons.

Murthy, S. K., 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. Data mining and knowledge discovery 2 (4), 345–389.

Nelder, J. A., Baker, R. J., 1972. Generalized linear models. Wiley Online Library.

Puggini, L., McLoone, S., 2018. An enhanced variable selection and isolation forest based methodology for anomaly detection with oes data. Engineering Applications of Artificial Intelligence 67, 126–135.

R Core Team, 2017. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
URL `https://www.R-project.org/`

Seber, G. A., Lee, A. J., 1977. Linear regression analysis, j.

Snedecor, G., Cochran, W., 1989. Statistical methods, eighth edr iowa state univ. Press, Ames, Iowa.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2), 301–320.