Prediction of Manufacturing Processes Errors: Gradient Boosted Trees Versus Deep Neural Networks

Ionut Anghel, Tudor Cioara, Dorin Moldovan, Ioan Salomie, Madalina Maria Tomus
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Emails: { Ionut.Anghel, Tudor.Cioara, Dorin.Moldovan, Ioan.Salomie} @cs.utcluj.ro,
madalinatomus@yahoo.com

***Abstract*—In this paper we investigate the use of machine learning techniques for optimizing manufacturing processes operation. More precisely we propose, compare and contrast two approaches for predicting errors in manufacturing processes. The first approach is based on machine learning algorithms while the second one uses deep learning techniques. Both approaches are validated using a dataset from literature, the SECOM dataset, which is representative for manufacturing processes. For the machine learning approach features are selected using the Multivariate Adaptive Regression Splines (MARS) algorithm and data is classified using the Gradient Boosted Trees (GBT) algorithm, while for the deep learning approach features are selected using a Support Vector Machine (SVM) algorithm and data is predicted using a Neural Network (NN). The evaluation results show that the best results are obtained using the deep learning approach.**

***Index Terms*—Manufacturing Processes, Machine Learning, Deep Learning, Classification, Features Selection.**

I. Introduction

Manufacturing processes are characterized by feature increased complexity due to their various activities and components that are hard to monitor and due to the large number of steps that must be applied until the final product is obtained. In the last years, the automated error detection techniques gained increased interest due to the performance optimization outcomes for process monitoring and control. Thus, approaches based on data mining, machine learning and deep learning techniques have emerged. Using this kind of techniques it is possible to create a model for the characterization of the process data and based on this model it is possible to realize efficient predictions for the unlabeled data. This paper proposes machine learning and deep learning techniques that can be used to assess manufacturing processes errors. Both approaches use input data and can make predictions regarding the labeling of the new data as valid or invalid.

The domain considered for validation of these techniques is the one of the semiconductors manufacturing. A correctly manufactured semiconductor is suited for industry, without fabrication errors and with a small probability to fail the functioning tests, thus predicting process errors is of great importance for lowering the error rates and generating usable products. A dataset that is representative for semiconductors manufacturing processes and which is used as support for comparing our proposed approaches is the SECOM dataset [1]. This dataset is used as a benchmark dataset for detecting if the products that result after the application of a sequence of manufacturing operations are faulty or not.

The work reported in this paper is done in the context of the OptiPlan project [2], that has as main objective the digitalization and the optimization of manufacturing processes in order to minimize the number of resources from the production line that are used inefficiently and to lower the error rate. This article continues the research from [3] in which several machine learning techniques for sensor-based manufacturing processes are explored.

The main contributions of this article are:

- defining a methodology for detecting errors in manufacturing processes based on the following phases: data cleaning, features selection, splitting in train/test data, model building, classification and model evaluation
- the adaptation, implementation, validation and comparison of two features selection algorithms in the context of manufacturing processes: the MARS algorithm and the SVM algorithm
- the adaptation, implementation and comparison of a machine learning algorithm and a deep learning algorithm: the GBT algorithm and a NN
- the evaluation of the proposed approaches on the SECOM dataset

The rest of the paper is organized as follows: Section II describes similar approaches from the literature, Section III presents the proposed methodology, Section IV compares the MARS and the SVM algorithms for features selection, Section V compares two classification algorithms, the GBT algorithm and the NN algorithm, Section VI presents comparison results using the SECOM dataset, while Section VII presents the conclusions.

978-1-5386-8296-8/18/$31.00 ©2018 IEEE
DOI 10.1109/EUC.2018.00012

II. Related Work

Data generated by manufacturing processes is very large, containing information from many sensors. The traditional approaches are not enough for the analysis of the data streams which are generated by hundreds or thousands of sensors. Machine learning, a subfield of the artificial intelligence, is used for predicting the values that can be taken by new data based on a prediction model which is built using the existing data while deep learning is a subset of the machine learning domain and uses neural networks with more than one hidden layer to classify the data.

In the rest of the section we describe different approaches that are used for the analysis of the manufacturing processes from three perspectives: data cleaning phase, features selection phase and building of the classification model phase.

Data cleaning is the phase in which the data is preprocessed prior to application of more complex algorithms such as features selection algorithms and classification algorithms. It deals with removing or correcting incorrect, irrelevant and ambiguous elements from the dataset. In [4] and [5], the SECOM dataset is used as experimental support and the data cleaning phase eliminates the data that contains constant values and the features that have many missing values. The authors of [6] treat the problem of data cleaning from the perspective of streams under speed constrains and they propose SCREEN, a constraint-based approach that can be used for cleaning stream data. In [7] the data cleaning is approached from the perspective of data imputation. By maximizing the imputation of the missing data, the missing values are filled *more*.

Features selection phase reduces the number of features that characterize the data. In [4] and [5], the features are selected using the Principal Component Analysis (PCA) and the Chi-Square. The Gain Ration (GR), a technique based on theoretical information, is applied further and the results are saved in a different dataset. Using the MeanDiff, two clusters are created. One of the clusters contains the positive samples, while the other one contains the negative samples. The features are then sorted in decreasing order and the first feature is the most relevant feature for the final answer. In [8] it is described a method that uses PCA for the selection of the most relevant features. The analyzed methods are the Multi-way PCA, the Multi-model PCA, the Adaptive Substantial PCA (ASSP) and the Adaptive Substantial PCA with Support Vector Data Description (ASSP-SVDD). The best results are obtained when the approach based on the SVDD is used.

Classification phase is the most complex phase and it represents the phase in which the model that is used for predicting if a product is faulty or not is created. In [4] and [5] the prediction model is created using Decision Trees (DT), Naïve Bayes (NB), k-Nearest Neighbor (k-NN) and Logistic Regression (LR). Another approach that uses the k-Nearest Neighbor is described in [9]. Together with the k-Nearest Neighbor, the Adaptive Mahalanobis Distance is used. This approach is based on the local dependencies between the input variables and gives better results in practice than the approach that uses the Euclidean Distance. In [10], compared to our approach, the authors did not apply Neural Networks (NN) because they consume too much memory. They applied DT, NB and LR.

III. Proposed Methodology

The phases of the proposed methodology are summarized in Figure 1. The dataset used for comparing the machine learning and the deep learning approaches is the SECOM dataset, a dataset that contains 1567 records among which 104 are fails and the rest of them are passes. Then the data is processed in several phases: data cleaning, features selection, splitting in training and testing data, model building, classification and evaluation.

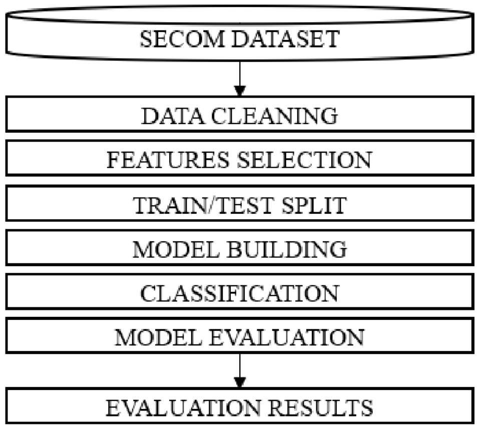

Fig. 1. Proposed Methodology Phases

A. Data Cleaning

In this phase all the features of the dataset are investigated. If a feature contains the same value for all the samples (the standard deviation is 0) then that feature is removed. If for a certain feature more than 55% of the values are missing then that feature is also removed.

B. Feature Selection

After the irrelevant information is removed in the Cleaning Phase, the most relevant features must be selected. The two algorithms that are compared for this phase are MARS and SVM. These two algorithms are detailed in Section IV. The data obtained in this phase is saved in a new dataset and is used in the next phases.

C. Train/Test Split

The dataset is split in a training dataset and a test dataset keeping the proportions 80% and 20%. Considering the fact that in the initial dataset there are more samples which are labeled as pass than samples which are labeled as fail, the proportion being equal with $1 : 14$, an oversampling is performed and thus the number of samples that are classified as fail is increased. After oversampling the ratio between passes and fails is $1 : 1$.

D. Model Building

In this phase, two approaches are used, one that builds the model using the GBT algorithm and one that builds the model using a NN with two hidden layers using Tensorflow. The two algorithms, the GBT and the NN, are described in more details in Section V.

E. Classification

After the model is built, the proposed system is able to classify the input data as fail or as pass. The classification results are evaluated based on a test dataset, thus the system must be able to correctly classify data that is not part of the training dataset.

F. Model Evaluation

The model is evaluated using four metrics: accuracy, precision, recall and the F-measure [11].

IV. Techniques for Features Selection Phase

The features were selected using two algorithms: MARS and SVM. We chose these techniques instead of other methods that are used traditionally such as the Principal Component Analysis (PCA) because these algorithms do not transform the features and because they gave the best results compared to other algorithms used by us in experiments. Since the best results for the deep learning approach are obtained when the number of selected features is higher, we used MARS in combination with a machine learning approach and SVM with a deep learning approach.

A. MARS Algorithm

MARS [12] algorithm selects the most relevant features for the classification model based on the Generalized Cross Validation (GCV) and the Residual Sum of Squares (RSS). The most important steps of the MARS algorithm are presented in Algorithm 1 [13].

Algorithm 1 MARS Algorithm

```
Input: A training dataset $S = \{(x_1, y_1), ..., (x_n, y_n)\}$.
Output: A hierarchy of features $F = \{f_1, ..., f_m\}$.
1: procedure FeaturesSelectionMARS($S$)
2:     Gather data
3:     Calculate candidate functions
4:     Specify constraints
5:     Forward pass
6:     Backward pass
7:     Generalized cross validation
8:     Return $F$
9: end procedure
```

The first step of the MARS algorithm is represented by the data gathering phase. The next step computes a set of candidate functions by combining the observed values and pairs of basis functions. After this step, two constraints are specified: the number of terms that compose the model and the maximum possible value for the interaction degree. The MARS forward pass step is a greedy algorithm that tries new function products in order to decrease the value of the training error. The MARS backward pass step has as objective the fixing of the overfit and it uses Generalized Cross Validation (GCV). Finally, the GCV represents an alternative for having two types of sets: training datasets and datasets for overfit checking. The GCV equation is presented below [14]:

$$GCV(\lambda) = \frac{\sum_{i=1}^{N} \left(y_i - \hat{f}_\lambda(x_i)\right)^2}{\left(1 - \frac{M(\lambda)}{N}\right)^2} \tag{1}$$

The parameters of the equation have the following significance [14]: N is the number of observations, $M(\lambda)$ is the effective number of the parameters from the model, $\hat{f}_\lambda$ is an estimated best model, λ is the number of terms, the x_i's are the observed values and the y_i's are the labels.

B. SVM Algorithm

The Support Vector Machines (SVM) algorithm is used to select the features for the approach based on deep learning. In general, the algorithms that are based on SVM are very useful in the classification problems [15]. The main objective of an algorithm of type SVM is to find a hyperplane that separates two classes of data. In order to achieve this objective, the solution is to maximize the margin around the plane $\left(w^T x + b\right)$, where w is the vector of weights and b is the bias. The SVM algorithm can be applied in the case in which the input is represented by a set of pairs of the form (x_i, y_i) where x_i is a vector of real value attributes and y_i is the label associated to the respective vector. The labels y_i have values from the set $\{-1, 1\}$. The problem associated to the SVM algorithm is the minimization of the following mathematical expression [16]:

$$\min_{w;b} \frac{1}{2} w^T \times w + C \times \sum_{1}^{n} \xi(w, b; x_i, y_i) \tag{2}$$

In this context $\xi(w, b; x_i, y_i)$ is the loss function and C is a regularization parameter from the training phase. For the loss function there are two known alternatives which are called the l_1 loss function and the l_2 loss function. The implementation which is used in the current approach is based on the use of the function l_1. These two functions have the following definitions:

$$l_1 = max(0, 1 - y_i \left(w^T \phi(x_i) + b\right)) \tag{3}$$

and

$$l_2 = max(0, 1 - y_i \left(w^T \phi(x_i) + b\right))^2 \tag{4}$$

The function ϕ represents the mapping of the input data in a highly dimensional space. For the training of a model of type SVM, a similarity measure called kernel [17] is used. A kernel function has two inputs and returns the similarity between the two inputs. Thus, using the kernel, the input data and the labels, it is possible to create a classifier that is based on the

SVM. The decision function or the predictor is expressed using the function:

$$f(x) = sign\left(w^T\phi(x) + b\right) \tag{5}$$

The method of kernel using is more efficient than the use of the vectors of the attributes because it is easier to calculate and it results in a better performance in practice. The kernel function is defined as:

$$K(x_i; x_j) = \phi(x_i)^T\phi(x_j) \tag{6}$$

The kernel can be linear, polynomial, Radial Basis Function (RBF), exponential, laplacian and so on. The most used kernel in practice is the linear kernel, whose expression can be deduced by taking into consideration the fact that for a linear SVM $\phi(x) = x$. The expression of the linear kernel becomes:

$$K(x_i, x_j) = x_i^T \times x_j \tag{7}$$

The RBF kernel is expressed using the formula:

$$K(x_i, x_j) = exp\left(-\gamma \left\|x_i - x_j\right\|^2\right) \tag{8}$$

In this context a linear kernel was used to obtain a hierarchy of features. The current step algorithm is adapted after the one presented in [16] (Algorithm 2 below).

Algorithm 2 SVM Algorithm

Input: A training dataset $S = \{(x_1, y_1), ..., (x_n, y_n)\}$.
Output: A hierarchy of features $F = \{f_1, ..., f_m\}$.

```
1: procedure FEATURESSELECTIONSVM($S$)
2:     Grid search
3:     Model training
4:     Sorting of the features in decreasing order
5:     Return $F$
6: end procedure
```

The steps of the Algorithm 2 are: the Grid search, in which the best value for the value of C is found, the training of a SVM model with the loss function l_1 keeping the best value and the sorting in decreasing order of the features using the absolute value of the weights from the model. After this algorithm, it is possible to apply a selection function whose objective is to select only those features for which the weights are not null. It is considered that the weights that are left after this elimination phase are the most relevant for the ulterior transformations.

V. Techniques for Classification Phase

This section compares two algorithms for the classification phase: the Gradient Boosted Trees (GBT) algorithm for the machine learning approach and the Neural Networks (NN) algorithm for the deep learning approach.

A. *Gradient Boosted Trees Algorithm*

The Gradient Boosted Trees (GBT) algorithm is adapted from [14] and [18]. The algorithm takes as input a training dataset, the differentiable loss function and the number of iterations. The most commonly used loss functions are the following [14]:

$$L(y, F(x)) = \frac{1}{2}\left[y - F(x)\right]^2 \tag{9}$$

$$L(y, F(x)) = \|y - F(x)\| \tag{10}$$

and the Huber loss function:

$$L(y, F(x)) = \begin{cases} \left[y - F(x)\right]^2 & \text{for } \|y - F(x)\| \leq \delta, \\ 2\delta\|y - F(x)\| - \delta^2 & \text{otherwise.} \end{cases} \tag{11}$$

Algorithm 3 GBT Algorithm

Input: A training dataset $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, a loss function $L(y, F(x))$, the number of iterations M.
Output: A classification model F_M.

```
1: procedure ALGORITHMGBT($S$, $L$, $M$)
2:     Initialize the model:
           $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
3:     for $m \in \{1, ..., M\}$ do
4:         for $i \in \{1, ..., N\}$ do
5:             $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$
6:         end for
7:         Get the terminal regions $R_{jm}$ by fitting a regression tree to the $r_{im}$ targets, where $j \in \{1, 2, ..., J_m\}$
8:         for $j \in \{1, ..., J_m\}$ do
9:             $\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$
10:        end for
11:        $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
12:    end for
13:    Return $F_M$
14: end procedure
```

The pseudo code is described in the Algorithm 3. The model is initialized by choosing a value for γ such that the $\sum_{i=1}^{n} L(y_i, \gamma)$ has the minimum value. The next steps of the algorithm are repeated for a number of iterations that is equal with the number of trees. The first step calculates the r_{im} for each sample from the training dataset, the second step gets the terminal regions R_{jm}, the third step calculates the γ_{jm}

values and the fourth step updates the F_m model based on the previous model F_{m-1} and on the results of the computations performed in the previous steps. Finally, the algorithm returns the classification model F_M.

B. Neural Networks

A brief description of the neural network algorithm [19] is presented in the Algorithm 4. At the beginning of the algorithm, the weights and the thresholds required by the network are initialized randomly. The activation function used in the neural network is the rectifier. Then the activation and the weight training are repeated for a number of iterations that is specified as input for the algorithm. In the activation phase the outputs of the neurons from the hidden layers and from the output layers are computed while in the weight training phase the weights are updated. Finally, the algorithm returns a classification model which is used further in the testing part.

Algorithm 4 NN Algorithm

Input: A training dataset $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, the number of iterations M, the number of hidden layers H.
Output: A classification model F_M.

```
1:  procedure ALGORITHMNN($S$, $M$, $H$)
2:      Initialization - set all the weights and the threshold
        values to random numbers
3:      for $m \in \{1, ..., M\}$ do
4:          Activate the network by applying the inputs and
            the desired outputs
5:          Calculate the actual outputs of the neurons in the
            hidden layers and in the output layers
6:          Update the weights
7:          Propagate the backward errors which are associ-
            ated with the output neurons
8:      end for
9:      Return $F_M$
10: end procedure
```

The learning model was built using Tensorflow. The building of a Neural Network (NN) and its training requires the following of a set of steps. In the first place, it is necessary to initialize a model based on the input data. In this context, the X variables will represent the input data and the Y variables will represent the output data or the labels. The neural networks consist of more neurons and each one of them is associated with an input weight based on its importance and a bigger weight represents a higher importance. Thus, the activation function for a neuron has the form $y = w \times x + b$ where x is the input, w is the weight and b is the bias. It is important to specify all the layers of the neural network at the initialization of the model phase and to respect the rule which states that the outputs from the previous layer are inputs to the next layer. A learning algorithm has as objective the determination of the optimal biases and weights for each of the component neurons from the network. This thing is usually achieved using a cost function. The neural network uses the quadratic cost [20] defined below:

$$C(w,b) = \frac{1}{2n} \sum_x \|y(x) - a\|^2 \tag{12}$$

In this context, a is the output given by the network, n is the number of inputs from the training dataset and $y(x)$ is the label associated to an input x. The problem solved by the neural network is the minimization of the cost function. A minimum cost suggests the fact that the prediction made by the network is very close to the desired label. If the total cost is equal with 0 then all the predictions are correct. The problem becomes equivalent to the finding of a point of minimum for the cost function. When the cost function has many variables, the problem becomes more complex and the method of computing the minimum using derivatives becomes inefficient.

If the C function is a function of m variables $x_1, x_2, ..., x_m$, then a change in this function ΔC produced by a small change in $\Delta X = (\Delta x_1, \Delta x_2, ..., \Delta x_n)^T$ can be expressed as:

$$\Delta C = \nabla C \times \Delta X \tag{13}$$

where ∇C is the gradient of the function C and can be defined using the formula:

$$\nabla C = \left(\frac{\partial C}{\partial x_1}, ..., \frac{\partial C}{\partial x_n} \right)^T \tag{14}$$

The formula $\Delta X = -\eta \nabla C$ is used next where η is the learning rate. Thus the cost C can be modified by applying the function $x \to x' = x - \eta \nabla C$ for a repeated number of times. Once the minimum of the cost function is obtained, the weights and the biases of the neural network can be set using the following formulas:

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{15}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l} \tag{16}$$

The input data can be split in smaller datasets called batches with the objective to process them in order in the learning part. The learning can also be realized in more epochs. Each epoch represents the application of the learning algorithm on the entire dataset. The existence of more epochs has as objective the obtaining of the convergence of the algorithm.

VI. Experimental Results

In our approach we consider that the best model is the one that has the smallest value for the false positive rate and the biggest value for the precision. Initially, before applying the machine learning approach or the deep learning approaches, the number of features is 590. In the cleaning phase, 153 features are eliminated as can be seen in Figure 2. The ratio of the selected features is approximately 74%.

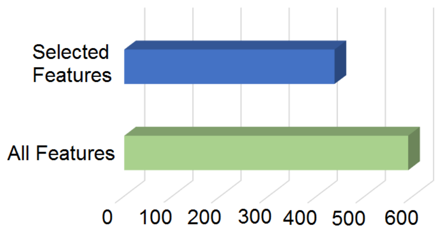

Fig. 2. Initial Number of Selected Features

The approaches used next on this data, either machine learning or deep learning, use the rest of the remaining 437 features.

A. *Machine Learning Technique*

The machine learning approach was developed using R and Apache Spark. The first part presents the results obtained after the features are selected using MARS and the second part presents the classification results when GBT is applied.

1) MARS: The number of features selected by the MARS algorithm is 10. Figure 3 presents the number of features selected by the MARS algorithm [12] compared to the 437 features selected initially. The identified features are: $f55$, $f60$, $f331$, $f113$, $f404$, $f346$, $f92$, $f1$, $f70$ and $f3$.

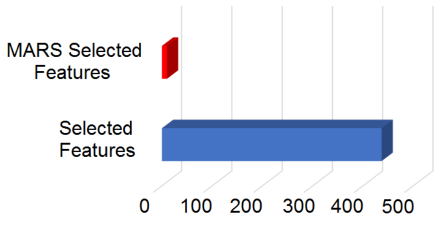

Fig. 3. Number of Selected Features MARS

2) Gradient Boosted Trees: The adjustable parameters that are tuned for the Gradient Boosted Trees algorithm are the number of trees and the maximum depth of the trees. The number of trees is equal with the number of iterations and a big value corresponds to a better performance. In the experiments the number of iterations was set to 100 and the maximum depth was varied between 10 and 15. Figure 4 presents the values for the precision and Figure 5 presents the values for the false positive rate when the maximum depth is varied.

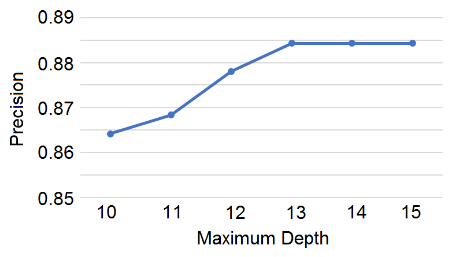

Fig. 4. Precision Values when the Maximum Depth is Varied

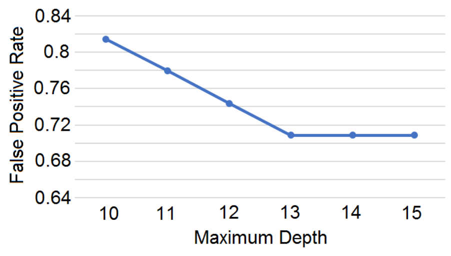

Fig. 5. False Positive Rate Values when the Maximum Depth is Varied

The most significant impact on the performance of the Gradient Boosted Trees algorithm was obtained when the maximum depth was varied rather than when the number of iterations was varied and this is the reason why this work presents only the results obtained when the maximum depth is varied. The best configuration for the input parameters which results in the best values for the false positive rate and for the precision is presented in Table I.

TABLE I
GRADIENT BOOSTED TREES BEST CONFIGURATION PARAMETERS

Parameter	Value
Number of Iterations	100
Maximum Depth	15

B. *Deep Learning Technique*

The deep learning system was developed using Python together with a set of libraries for data preparation such as Numpy, Pandas, Scikit-learn and Tensorflow. Prior to the application of the deep learning approach the data is normalized to take values from the interval $[0, 1]$. In the first part are presented the results when the features are selected using SVM and in the second part are presented the results when the data is classified using NN.

1) SVM: In the features selection phase, the SVM algorithm selects the most relevant 102 features. Figure 6 compares the number of features selected by the SVM algorithm with the 437 features that were selected initially.

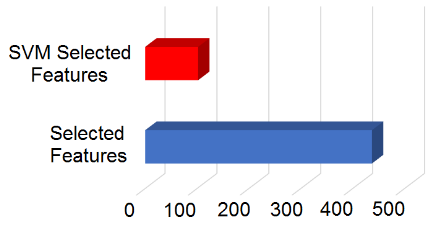

Fig. 6. Number of Selected Features SVM

2) Neural Networks: The values of the configuration parameters of the Neural Network were determined after a series of experiments in which one of the configuration parameters is varied while the others have fixed default values which are used in literature. Figure 7 and Figure 8 show the variation of the false positive rate and of the precision when the learning rate is varied. As can be seen in the figures, the best values are obtained when the learning rate is equal with 0.05.

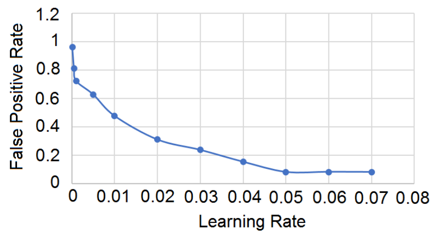

Fig. 7. False Positive Rate Values When the Learning Rate is Varied

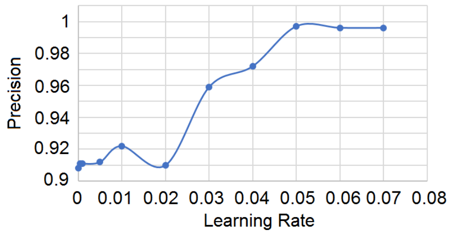

Fig. 8. Precision Values When the Learning Rate is Varied

Figure 9 and Figure 10 show the variation of the false positive rate and of the precision when the batch size is varied. As can be seen in the figures, the best values are obtained when the batch size is equal with 10.

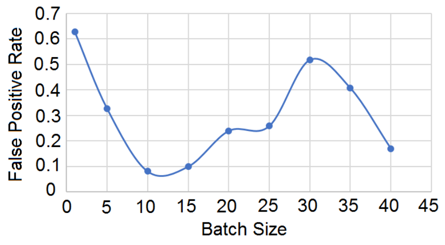

Fig. 9. False Positive Rate Values When the Batch Size is Varied

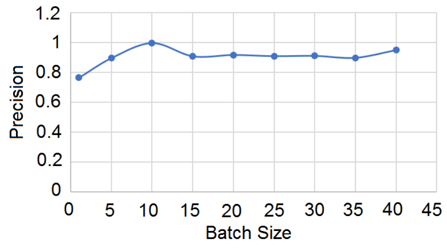

Fig. 10. Precision Values When the Batch Size is Varied

The next two figures, Figure 11 and Figure 12 show the values taken by the false positive rate and by the precision when the number of neurons on the first hidden layer and the number of neurons on the second hidden layer are varied. The best results are obtained when the number of neurons on the first hidden layer is equal with 50 and the number of neurons on the second hidden layer is equal with 25.

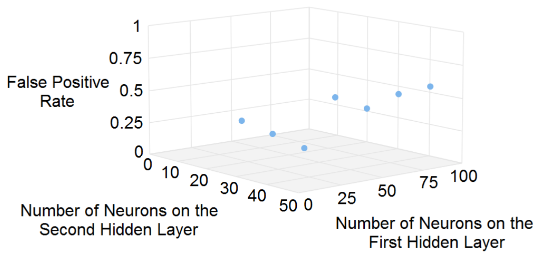

Fig. 11. False Positive Rate Values When the Number of Neurons from the Two Hidden Layers is Varied

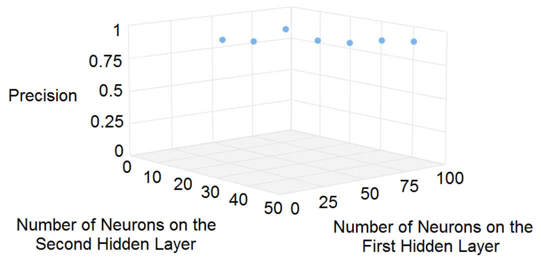

Fig. 12. Precision Values When the Number of Neurons from the Two Hidden Layers is Varied

In the case of the proposed approaches, the best configuration for the input parameters that gives the best values for the false positive rate and for the precision is presented in Table II.

TABLE II
NEURAL NETWORK CONFIGURATION PARAMETERS

Parameter	Value
Learning Rate	0.05
Number of Hidden Layers	2
Number of Neurons on the First Hidden Layer	50
Number of Neurons on the Second Hidden Layer	25
Number of Epochs	8
Batch Size	10

C. Comparison of the Results

The results obtained using the machine learning approach and the deep learning approach and the best configurations for the adjustable input parameters which were determined after performing a series of experiments are presented in Table III.

TABLE III
EXPERIMENTAL RESULTS

	Machine Learning	Deep Learning
accuracy	0.9	0.936
recall	0.9	0.918
precision	0.884	0.997
F-Measure	0.891	0.957

As can be seen in the table, the deep learning approach gives higher values for accuracy, recall, precision and F-Measure. This indicates that in the case of the methodology proposed by us the deep learning approach is the best approach.

D. Comparison with Literature Results

Table IV illustrates results obtained by other researchers [21] that use the SECOM dataset as experimental support.

TABLE IV
LITERATURE RESULTS

classification algorithm	accuracy
Random Forest	0.904
Logistic Regression	0.835
Decision Tree	0.803
Artificial Neural Network	0.889

E. Results Obtained for Other Datasets

Table V presents a selection of the results that were obtained using as experimental support other datasets from the manufacturing domain.

TABLE V
RESULTS OBTAINED FOR OTHER DATASETS

dataset	classification algorithm	accuracy
Regulators Synthetic Dataset [2]	GBT	0.75
SETFI Dataset Preprocessed [22]	NN	0.69

VII. CONCLUSIONS

In this paper we define, compare and contrast two approaches for predicting manufacturing processes errors. The first approach uses machine learning techniques, selects the features using the MARS algorithm and classifies the data using the SVM algorithm, while the second approach uses deep learning techniques, selects the features using the GBT algorithm and classifies the data using a NN. The main goal is to obtain high values for true positive rate, precision and F-measure and a small value for false positive rate. From the perspective of the obtained results, we conclude that the deep learning approach gives better results in practice compared to other existing methods from the literature. As future work we propose (1) the testing and the validation of the proposed methodology on more datasets, (2) the adaptation of the proposed methodology to a system that can analyze manufacturing processes data streams in real time and (3) the identification of the causes that lead to manufacturing processes errors in order to improve the quality of the manufacturing processes.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI UEFISCDI, project number PN-III-P2-2.1-BG-2016-0080, within PNCDI III.

REFERENCES

[1] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[2] "Optiplan," http://coned.utcluj.ro/OptiPlan/index.html.

[3] D. Moldovan, T. Cioara, I. Anghel, and I. Salomie, "Machine learning for sensor-based manufacturing processes," *13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 147–154, 2017.

[4] K. Kerdprasop and N. Kerdprasop, "A data mining approach to automate fault detection model development in the semiconductor manufacturing process," *INTERNATIONAL JOURNAL OF MECHANICS*, vol. 1, 2011.

[5] S. Munirathinam and B. Ramadoss, "Predictive models for equipment fault detection in the semiconductor manufacturing process," *IACSIT International Journal of Engineering and Technology*, vol. 8, 2016.

[6] S. Song, A. Zhang, J. Wang, and P. S. Yu, "Screen: Stream data cleaning under speed constraints," *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 827–841, 2015.

[7] S. Song, A. Zhang, L. Chen, and J. Wang, "Enriching data imputation with extensive similarity neighbors," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1286–1297, 2015.

[8] Z. Ge and Z. Song, "Semiconductor manufacturing processes monitoring based on adaptive substatistical pca," *IEEE Transactions on Semiconductor Manufacturing*, vol. 23, no. 1, 2010.

[9] G. Verdier and A. Ferreira, "Adaptive mahalanobis distance and k-nearest neighbor rule for fault detection in semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 1, 2011.

[10] K. Kerdprasop and N. Kerdprasop, "Feature selection and boosting techniques to improve fault detection accuracy in the semiconductor manufacturing process," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2011.

[11] "Confusion matrix," https://en.wikipedia.org/wiki/Confusion_matrix.

[12] D. S. Kumar and S. Sukanya, "Feature selection using multivariate adaptive regression splines," *International Journal of Research and Reviews in Applied Sciences and Engineering (IJRRASE)*, vol. 8, no. 1, pp. 17–24, 2016.

[13] P. Bansal and J. Salling, "Multivariate adaptive regression splines (mars)," http://www.lans.ece.utexas.edu/courses/ee380l_ese/2013/mars.pdf, The University of Texas at Austin Electrical and Computer Engineering, 2013.

[14] T. Hastie and R. T. an Jerome Friedman, "The elements of statistical learning data mining, inference, and prediction," *Springer Series in Statistics*, 2008.

[15] J. Weston, S. Mukherjee, O. Chapelle, M. Pontiltt, T. Poggiott, and V. Vapni, "Feature selection for svms," *NIPS*, vol. 12, pp. 668–674, 2000.

[16] Y.-W. Chang and C.-J. Lin, "Feature ranking using linear svm," *JMLR: Workshop and Conference Proceedings*, no. 3, pp. 53–64, 2008.

[17] J. Brank, M. Grobelnik, N. Mili-Frayling, and D. Mladenic, "Feature selection using support vector machines," 2002.

[18] "Gradient boosting," https://en.wikipedia.org/wiki/Gradient_boosting#cite_note-hastie-7.

[19] A. Bayesian and C. H. Liu, "On face recognition using gabor filters," *World Academy of Science Engineering and Technology*, no. 28, pp. 51–56, 2007.

[20] M. A. Nielsen, "Neural networks and deep learning," http://neuralnetworksanddeeplearning.com/.

[21] J. Kim, Y. Han, and J. Lee, "Data imbalance problem solving for smote based oversampling: Study on faulty detection prediction model in semiconductor manufacturing process," *Advanced Science and Technology Letters*, vol. 133, pp. 79–84, 2016.

[22] AA&YA and Intel, "Manufacturing data: Semiconductor tool fault isolation," mailto:causality@clopinet.com, 2008.