Designing Context-aware Data Plausibility Automation Using Machine Learning

1st Mohaddeseh Basiri
KTH Royal Institute of Technology
Stockholm, Sweden
mbasiri@kth.se

2nd Johannes Himmelbauer
Software Competence Center Hagenberg GmbH
Hagenberg, Austria
johannes.himmelbauer@scch.at

3rd Lisa Ehrlinger
Software Competence Center Hagenberg GmbH
Hagenberg, Austria
lisa.ehrlinger@scch.at

4th Mihhail Matskin
KTH Royal Institute of Technology
Stockholm, Sweden
misha@kth.se

***Abstract*—In the last two decades, computing and storage technologies have experienced enormous advances. Leveraging these recent advances, Artificial Intelligence (AI) is making the leap from traditional classification use cases to automation of complex systems through advanced machine learning and reasoning algorithms. While the literature on AI algorithms and applications of these algorithms in automation is mature, there is a lack of research on trustworthy AI, i.e., how different industries can trust the developed AI modules. AI algorithms are data-driven, i.e., they learn based on the received data, and also act based on the received status data. Then, an initial step in addressing trustworthy AI is investigating the plausibility of the data that is fed to the system. In this work, we study the state-of-the-art data plausibility check approaches. Then, we propose a novel approach that leverages machine learning for an automated data plausibility check. This novel approach is context-aware, i.e., it leverages potential contextual data related to the dataset under investigation for a plausibility check. We investigate three machine learning solutions that leverage auto-correlation in each feature of dataset, correlation between features, and hidden statistics of each feature for generating the checkpoints. Performance evaluation results indicated the outstanding performance of the proposed scheme in the detection of noisy data in order to do the data plausibility check.**

***Index Terms*—Artificial intelligence; Machine learning; Automation; Plausibility check; Anomaly detection; Ontology; Context-aware.**

I. Introduction

Due to the rapid development of information technology and manufacturing process, traditional manufacturing enterprises have been transformed to the digital and smart factories [1], [2]. This improvement leads to the emerging complex systems with thousands of components and sub-systems, in which continuous monitoring of these systems is of crucial importance. From the data analytic point of view, this means surveillance of large amounts of time series data in order to ensure the correctness of the data and run data plausibility checks. So, regarding the huge amounts of data, human monitoring of data is not feasible, which conducts us to the automated plausibility check using Machine Learning (ML) and data mining approaches [3].

Data plausibility describes the state when data seems reasonable. Conversely, an anomaly or outlier is a data point that is remarkably different from the remaining data. A possible approach for implementing outlier detection is to run plausibility checks [4]. Rapid and efficient outlier detection is critical for many applications including intrusion detection systems, credit card fraud, sensor events, medical recognition, law enforcement, etc. [5]. Although outlier detection is an intensively researched topic in the machine learning and statistics community [6], there are still many open challenges in practice. The first challenge is context dependence. For example, a very high fluctuation rate in a company dataset might be reasonable for a catering service, but not for a construction company. Thus, the decision of whether a data sample seems reasonable (i.e., it is not an outlier) often depends on the context within it appears. Second, the high dimensionality of the dataset creates difficulties for data plausibility check [7]. Since the number of features increases in a high-dimensional dataset, the amount of data for accurate generalization also raises, which results in data sparsity and scattering. This data sparsity is because of inessential features or irrelevant attributes that hide the correct anomalies. So, anomaly detection is becoming a challenging task by increasing the number of features and attributes in large datasets. In addition to these challenges, there are some inherent issues such as difficulties in the design of threshold between normal and anomalous data, and much noise existence due to incorrect measurements or sensor malfunctioning that may cause the false notifications. On the other hand, data imbalance as the common problem in anomaly detection approaches affects the robustness of models, as very few outlier samples are available.

In order to address the aforementioned challenges, we present a novel context-aware approach for an automated data plausibility check, where there is a lack of research in the literature. In this approach, machine learning techniques are leveraged on top of semantic models, e.g., ontology, and benefited from side information in the datasets. Semantic data models like ontology [8] facilitate the incorporation of semantic information into the data. This work is the extended version of [1]. The focus of this paper is on multivariate outlier detection on the level of records (i.e., samples, rows) instead of single values. In this regard, the main contributions of this work include:

1) Presenting a data plausibility check framework; including test ontology, test data generator, checkpoint, and their

message exchanges.

2) Disclosing three types of tests, to be deployed in the test ontology, executed in the test generator, and used in decision making in the checkpoint module. These tests include:
   a) Inter-feature check, checking features based on their relations leveraging an machine learning module for prediction of a feature from some related features (list of neighbors is given by the test ontology from training)
   b) Intra-feature check (1), checking a feature based on its lags (previous values) using an ML module for prediction based on the lags (number of lags is given by the test ontology from training),
   c) Intra-feature check (2), checking a feature leveraging metadata and its long-term statistics (the type of needed metadata and action on them are given by the test ontology)
3) Presenting a comprehensive analysis of the performance of the proposed solution on a propriety dataset and drawing insights and conclusions from the analyses.

The rest of this paper is organized as follows: Section II presents state-of-the-art anomaly detection techniques. Section III describes the needed background for the work in more details. Section IV presents the data and models used to solve the problem. Section V describes our solution for solving the problem. Simulation results and discussion are presented in Section VI. In Section VII, the findings of this work are presented in a brief but succinct manner.

II. Related Work

Anomaly detection, as the concept of identifying patterns or data points that are significantly different from the expected behavior, has been widely studied. State-of-the-art using anomaly detection algorithms can be categorized as following [9]:

Classification Based: This algorithm strives to discern normal data instances from the abnormal ones in the given dataset space by using a trained model. It is categorized into one-class and multi-class models. In one-class models, a distinguished threshold is learned to label data points outside of this threshold as anomalies instances [10]. In multi-class models, multiple classifiers are trained. A data point is recognized as an anomaly if none of the classifiers can label it as the normal instance [11]. Various anomaly detection techniques such as neural networks, Bayesian networks, support vector machines, and rule-based utilize different classification algorithms to build their classifiers.

Nearest Neighbor Based: In this technique, normal data points are in compact neighborhoods, while anomalous data points are far from their nearest neighbors. This technique needs a distance or similarity measurement between two data points in order to recognize which data points are far from or different from other points. For continuous features, Euclidean distance is used, and for categorical features, a simple matching coefficient is a common option. In multivariate data points, the combination of computed distance for each feature is usually leveraged. The nearest neighbor technique is categorized into two groups regarding how they compute the anomaly score: 1) The distance of a data point to its k^{th} nearest neighbor is used as the anomaly score, e.g., k-nearest neighbor approach [12]. 2) The relative density of each data point is computed as the anomaly score, e.g., Local Outlier Factor (LOF) [13].

Clustering Based: In this algorithm, similar data instances are grouped into clusters. There are three categories of clustering-based anomaly detection techniques. First, techniques that suppose normal data instances belong to a cluster, while abnormal data points do not belong to any cluster, e.g., SNN clustering [14]. Second, algorithms that consider normal data instances are near to the closest cluster centroid, while outliers are far from their closest cluster centroid, e.g., Self-Organizing Maps [15]. Third, those assume normal data instances create large and dense clusters, while anomalous data points create small or scattered clusters, e.g., Cluster-Based Local Outlier Factor (CBLOF) [16].

Statistical: Regarding the basic assumption of statistical anomaly detection techniques, a data point is anomaly if it is not generated by the stochastic model. In other words, normal data points happen in high probability areas of a stochastic model, while outliers happen in the low probability areas of the stochastic model. In these approaches, a statistical model (usually for normal patterns) is applied to the dataset and then a statistical inference test is utilized to identify whether a data point fits well to this model or not. Regarding the applied test statistic, data instances that there are low probability to be created from the learn model are considered as anomalous data. Parametric and non-parametric techniques are two approaches that can be leveraged to fit a statistical model. Parametric techniques benefit the distribution knowledge and compute parameters from the given data, while non-parametric techniques do not. Gaussian model based algorithms like Maximum Likelihood Estimation (MLE) [17], regression model based like Auto-regressive Integrated Moving Average (ARIMA) [18], and combination of parametric distribution based algorithms like Expectation Maximization (EM) [19] are instances of parametric techniques. Histogram based such as Intrusion-Detection Expert System (IDES) [20], and kernel function based like parzen windows estimation [21] are samples of non-parametric techniques.

Information Theoretic: In this approach, the information content of the dataset is analyzed. The purpose of this technique is to solve a double optimization problem in order to determine the minimized subset that maximizes the complexity reduction of the dataset, and finally label that subset as the outlier. Entropy and Kolmogorov Complexity [22] are two examples of this category.

Spectral: This technique tries to find a lower-dimensional subspace in such a way that outliers and normal data points are remarkably different. Hence, anomalies can be easily distinguished. Principal Component Analysis (PCA) is used in many techniques in order to project data points into a lower dimensional space [23].

In order to have better overview of different techniques and their algorithms, advantages and disadvantages of each techniques are summarized in Figure 1.

III. Background

In this section, ARIMA, decision tree, and random forest as machine learning algorithms and ontology are described in more details.

A. ARIMA

ARIMA (Auto-Regressive Integrated Moving Average) is an extension of an auto-regressive moving average (ARMA) model. Both of these models are utilized in order to have better understanding of time series data or predict future values of an attribute [18]. The AR part of ARIMA shows that the attribute of interest is regressed on its own lagged (i.e., on its prior values). The MA part is representation of the regression error, which is the linear combination of contemporaneous error values and errors at various times in the past. The I (for ”integrated”) shows that the data values have been substituted with the discrepancy between their values and the previous values. ARIMA model is denoted by $ARIMA(P, I, Q)$, where P is the order of auto-regressive model (number of time lags), I is the degree of differencing, and Q is the order of moving-average model. The aim of each of these features is to make the model fit well with the data.

B. Decision Tree and Random Forest

Decision tree as a rule-based classifier corresponds each internal node of the tree to an attribute. Each branch of the tree represents a condition (rule) on the related attribute. The result of the condition on the related attribute can be binary, categorical, or real-valued. Depending on the result of the condition, a test example pursues the related branches starting from the root node and moves down to a leaf node. Leaf nodes represent the labels, which are the results of classification. The basic idea of a single decision tree is leveraged for random forests (RF)s and ensemble learning. Regarding the main principle, utilizing an ensemble of several naive weak classifiers can cause to a much more powerful classifier, such that each of this unique weak classifier can perform rather more powerful than random estimation and independent of all other classifiers [24].

As shown in Figure 2, random forest works based on the bagging algorithm and uses ensemble learning technique. It builds as several trees as possible on the subset of data and merges the results of all the trees together. In this way, it decreases overfitting problem and also reduces the variance and hence improves the accuracy. This classifier can handle missing values and does not need feature scaling. Random forest is usually stable to outliers. Even if a new data instance is inserted in the dataset, the entire algorithm is not affected much. Since only one tree might be impacted by the new data, it is difficult to impact all the trees. Moreover, random forest is comparatively less impacted by noise.

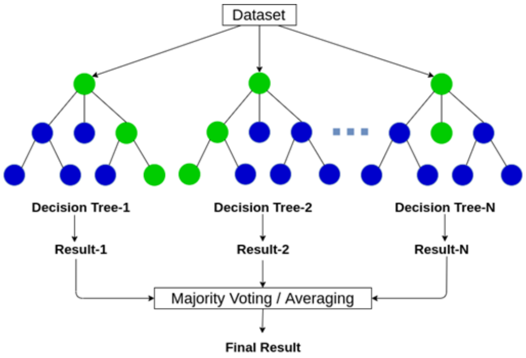


Fig. 2. How random forest algorithm works. (Source: [25])

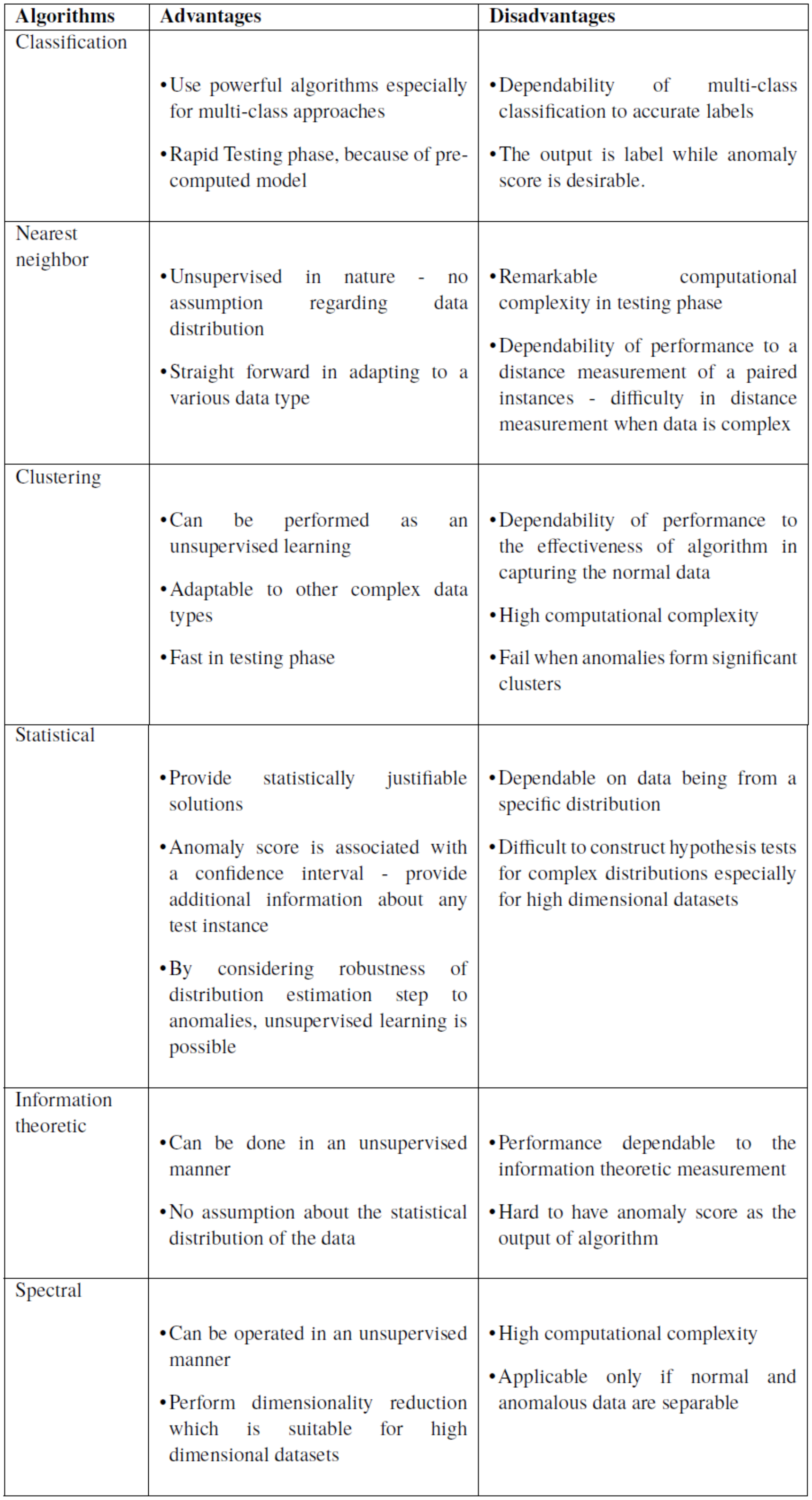
C. Ontology

Ontology is utilized to obtain knowledge about some domain of interest. An ontology defines the concepts in the domain and also the relationships that exist between these concepts, i.e., an ontology defines common words in order to share common understanding of the structure of information in a domain [26]. Various ontology languages provide different possibilities. Our focus is on introducing the components of OWL ontology as the most recent development in standard ontology languages [27]. An OWL ontology consists of Individuals, Properties, and Classes as the components. In the following, each of these components is introduced.

Individuals: Individuals expose objects in the domain of interest. OWL does not use the Unique Name Assumption (UNA). This implies that two different names could refer to the same individual. For instance, ’Queen Elizabeth’, ’The Queen’, and ’Elizabeth Windsor’, all of them might refer to the identical individual. In OWL, individuals must be explicitly declared that they refer to the same object or they are different. Figure 3 depicts a demonstration of some individuals in various domain.

Properties: Properties are relations that connect two individuals together. As shown in Figure 4, the property *livesIn* connects the individual *Matthew* to the individual *England*, or the property *hasSibling* links the individual *Matthew* to the individual *Gemma*. Properties could be inverted. For instance, the inverse of *hasOwener* property is *isOwnedBy* property. Also, properties could be either *transitive* or *symmetric*.

Classes: OWL classes behave like *sets* that contain individuals. They precisely declare the needs of the class memberships. For example, the class *Person* would contain all the individuals that are persons in the domain of interest. Classes might have superclass-subclass taxonomy. For instance, assume the classes *Animal* and *Dog* - *Dog* is the subclass of *Animal*. So, *Animal* is the superclass of *Dog*. This means that all dogs are animals



Algorithms	Advantages	Disadvantages
Classification	• Use powerful algorithms especially for multi-class approaches • Rapid Testing phase, because of pre-computed model	• Dependability of multi-class classification to accurate labels • The output is label while anomaly score is desirable.
Nearest neighbor	• Unsupervised in nature - no assumption regarding data distribution • Straight forward in adapting to a various data type	• Remarkable computational complexity in testing phase • Dependability of performance to a distance measurement of a paired instances - difficulty in distance measurement when data is complex
Clustering	• Can be performed as an unsupervised learning • Adaptable to other complex data types • Fast in testing phase	• Dependability of performance to the effectiveness of algorithm in capturing the normal data • High computational complexity • Fail when anomalies form significant clusters
Statistical	• Provide statistically justifiable solutions • Anomaly score is associated with a confidence interval - provide additional information about any test instance • By considering robustness of distribution estimation step to anomalies, unsupervised learning is possible	• Dependable on data being from a specific distribution • Difficult to construct hypothesis tests for complex distributions especially for high dimensional datasets
Information theoretic	• Can be done in an unsupervised manner • No assumption about the statistical distribution of the data	• Performance dependable to the information theoretic measurement • Hard to have anomaly score as the output of algorithm
Spectral	• Can be operated in an unsupervised manner • Perform dimensionality reduction which is suitable for high dimensional datasets	• High computational complexity • Applicable only if normal and anomalous data are separable

Fig. 1. Advantage and disadvantages of various anomaly detection techniques [9]

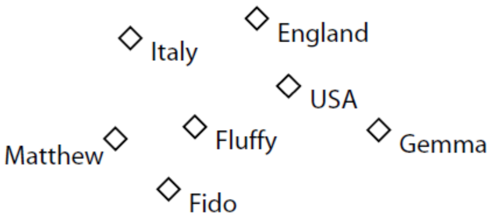


Fig. 3. Demonstration of individuals. (Source: [27])

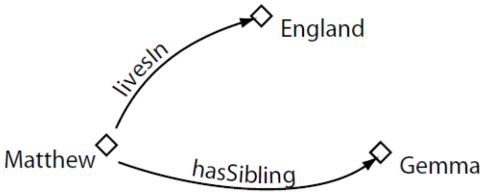


Fig. 4. Demonstration of properties. (Source: [27])

and all members of class *Dog* are members of class *Animal*. Figure 5 depicts a demonstration of some classes, which are containing some individuals. Classes are shown as circles or ovals and individuals are as the instances of classes.

IV. Data And Models For Experiments

This section sheds light on the data under investigation. Furthermore, it provides details on the pre-processing performed on the received data, and the planned data analytics and verification procedures.

A. Data Collection

The relation of data to AI is as food to the human being. In other words, there is no artificial intelligence in isolation, and any AI approach needs corresponding data for learning. For this project, we receive the dataset through our industrial partner, from a third-party company. While the data itself is confidential and could not be shared open access on the web, in this section we try to provide insights into the data, in order to make the reader familiar with the approaches that will be presented in the next section.

1) A deep Look into the Dataset: Our dataset contains 18 unique test runs for produced machine parts. Each of these tests has been run for a different period of time, i.e., there are different reported cycles per test.

2) Features available per test: The first dataset (testoverview.csv) provides a comprehensive list of features available per test (out of 18 tests). These features include the type of material used in the experiments, e.g., the oil, and the setting that has been applied in the experiment, e.g., distance between disks. This metadata has been collected to be used for verification of dataset and its reproducibility, as we will see in the next section (Section V-C).

3) Features available per test cycle: For each of the tests mentioned above, measurements have been done for different periods of time, and a number of features have been recorded per time cycle in the second dataset (tests.csv). In other words, this dataset presents a comprehensive list of features available per time cycle for each test. In contrast to the first dataset, most of the features of the second dataset are unknown to the reader and have not been revealed by the third company to us.

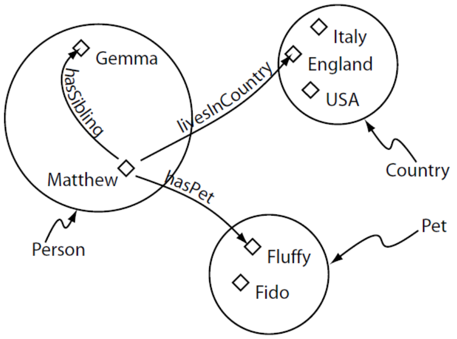


Fig. 5. Demonstration of classes, containing individuals and properties between them. (Source: [27])

B. Pre-processing of Data

For pre-processing of data, we investigate NaN values and missing entries in the dataset. Then, we start plotting the data to see trends in the results from each test. Figure 6 represents two features of a specific test across time. It is interesting to see that the features represent 3 trends in 3 different phases, including (a) an increasing trend at the start phase (up to 600 cycles, with a return to 50 periodically for the second feature), (b) a semi-constant trend from 600 cycles until the end cycle -600 cycles, and (c) an increasing trend in the last 600 cycles (with a return to 50 periodically for the second feature). In order to see if it is a recurring trend, we investigate the same thing for other tests. For example, Figure 7 represents the same phenomena for another test. Here, the start and end phases show a decreasing trend, while the middle phase is semi-constant with low variations. The increasing/decreasing trend at the start/end phases and the semi-constant trend in the middle phase are observed in all tests unless one test (depicted in Figure 8), and this test is excluded from our analysis based on the human expert information, as it does not show the standard behavior.

C. Planned Data Analysis

Figure 9 represents the plausibility check problem and the planned analysis for dealing with this problem. Based on this figure, we receive the data per test per time cycle (as the data pipeline from the bottom of the blue box), and also some metadata per test (as the left data pipeline), and aim at investigating if each test data is plausible or not. The focus of this work is on the design of the plausibility check module and the design of an ontology for the generation of the check data to be used in the plausibility checker module.

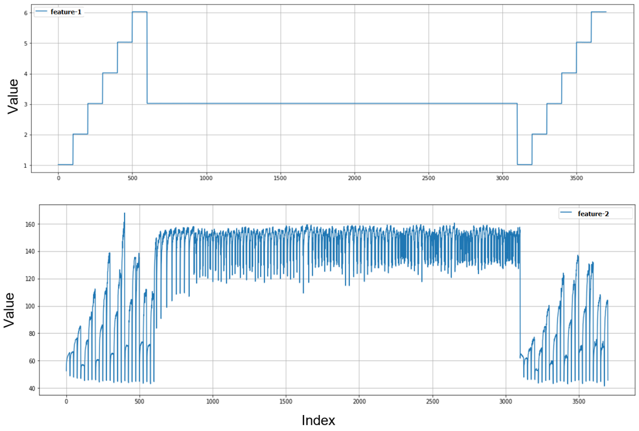


Fig. 6. Description of subset-1 of data versus cycle index

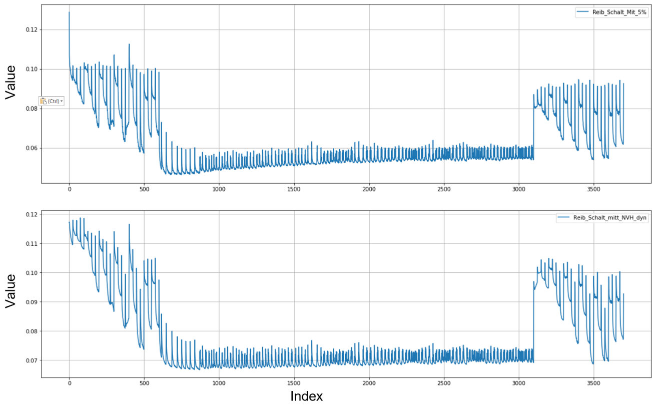


Fig. 7. Description of subset-2 of data versus cycle index

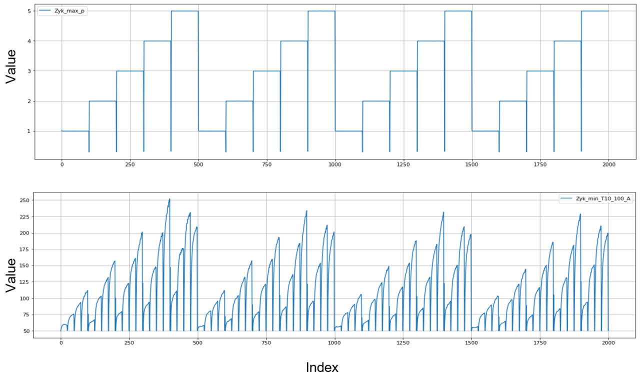


Fig. 8. Description of subset-3 of data versus cycle index

1) Evaluation Metric: In this work, we focus on predicting the test values and comparing them with the real values for detection of a potential anomaly, i.e., performing regression analysis. Regression refers to predictive modeling, and involves predicting a numeric value, and is different from the classification that involves predicting the label of a class of data. In regression analysis, we use Mean Squared Error (MSE), as an error metric designed for evaluating predictions made on regression problems. The MSE metric is derived as the mean or average of the squared differences between real and predicted values, i.e., $MSE = \frac{1}{N}\sum_{i=1}^{N}(X[i] - \tilde{X}[i])^2$, in which, $X[i]$ is the i'th real value in the dataset and $\tilde{X}[i]$ is the i'th predicted value. The difference is squared, which has the effect of resulting in a positive error value and inflating or magnifying the large errors.

2) Evaluation Framework: Figure 9 represents the evaluation framework for performance assessment of the proposed plausibility check solution. Based on this figure, we will add two types of error, including constant bias noise and random noise, to the test data per cycle, and will check if the plausibility check module is capable of finding inconsistency in the data.

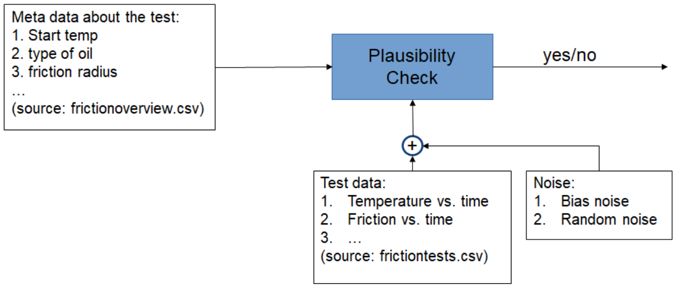


Fig. 9. Planned evaluation framework

V. The Proposed Solution

This section aims at presenting contributions of the work. Our contributions include the design of a data analytics unit for plausibility check of data. The schema of the proposed solution has been depicted in Figure 10. This proposed unit includes two novel functions: (a) the test data generator function and (b) the plausibility check function. The former one collects further information about the test and generates checkpoints (contextual data) to be evaluated by the checker function. The checker function compares the checkpoints with the threshold values and makes the plausibility decision. Then, before storing data in the database or actuating based on the received data, the customer can pass the data through the data analytics unit and check whether this data is plausible or not. As we will see in detail of the proposed approaches, the test data generator function includes an intelligent agent for generating the test data.

Implementation of the proposed solution requires contextual data to be collected. Contextual data is test data, which is

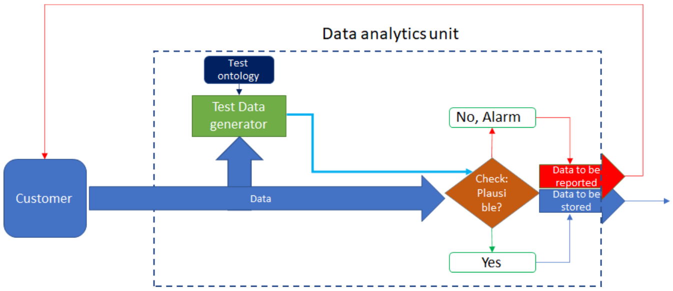


Fig. 10. The proposed solution

related to the dataset to be checked at the plausibility check function. Also, the contextual data should be contributory in the plausibility check of the dataset. In the following, three ideas are presented for generating contextual data:

1) Cross-correlation between columns of the dataset is used for prediction of the column of interest. The performance of prediction (MSE) is reported as a property of column of interest for a plausibility check.
2) Prediction of future values of each column based on the previous values of that column and comparison with the received data (Auto-regression). The performance in terms of MSE is used for a plausibility check.
3) Finding rules and statistics for each column based on metadata and configuration available for the test, e.g., type of oil used at the machine part.

A. Design of contextual information for plausibility check: The first solution

In tests.csv dataset, there are 18 unique tests with 29 data columns, unique hash codes, and different cycles. The columns of the dataset could be correlated together. Then, one can use some columns to check the plausibility of other columns.

For testing the hypothesis of mutual correlation between different columns, we consider one unique test and find the correlation between each column with itself and with 28 other columns, by using the built-in correlation function of Python. As shown in Figure 11, the correlation results of each test are stored in a matrix of $29 * 29$. The correlation number in each cell $c_{i,j}$ of this matrix is an amount between -1 and 1 and this number states that how much the column i is correlated to the column j. The higher the absolute value of each cell $c_{i,j}$, the more correlated the column i to the column j.

Since the correlations between columns in one test might randomly be high or low, the correlation matrix is calculated for each 18 unique tests, and 18 correlation matrices of $29 * 29$ are obtained. Then, each cell of correlation matrices is averaged over all 18 tests. Figure 12 refers to the result of averaged correlation matrices over 18 tests. This correlation matrix is for the starting phase. Since the behavior of features in the various phases is different, the correlation matrix for the steady-state and ending phase are calculated separately.

As the absolute value of the correlation matrix is of importance, the features with the hottest and coldest colors are more correlated together. As shown in Figure 12, the results confirm the existence of strongly related features for plausibility check of each feature.

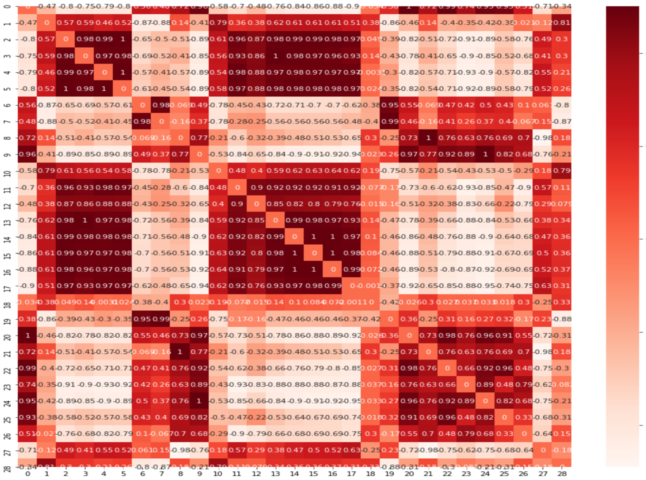


Fig. 11. The correlation matrix for one test before averaging

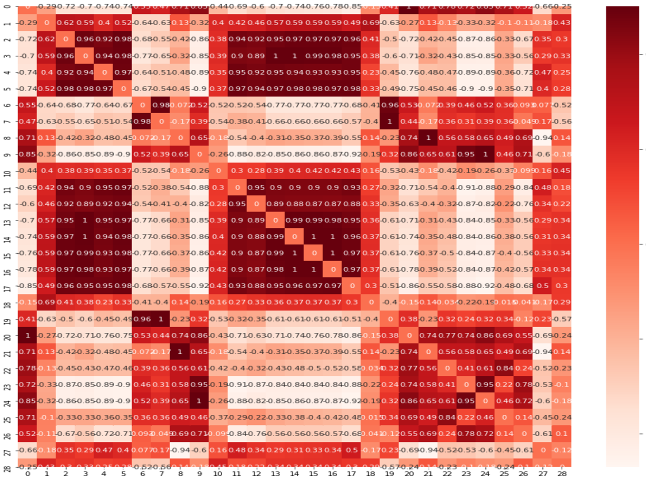


Fig. 12. The correlation matrix after averaging over all available tests

Having access to the m most related columns for each column, we can train a machine-learning algorithm to predict the value of feature of interest (FoI) based on the selected features. Here, we select the three most related features for prediction. If the prediction based on the selected features matches the recorded data, there is a low probability of implausibility. If the predicted and recorded values do not match, an alarm could be raised. For deploying this idea, we need an ML agent. Figure 13 depicts the check data generation and decision-making procedures in more detail. In this figure, the FoI is X_1, and the subset of features related to it is X_2. Then, X_2 is fed to the test generator node, and a prediction of X_1 based on X_2 is generated (call it $\tilde{X}_1$). The predicted value, $\tilde{X}_1$ along with X_1 are fed to the comparator

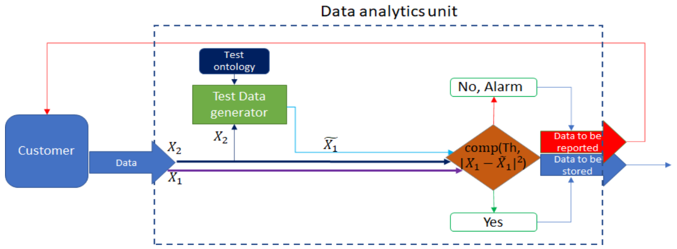


Fig. 13. Feature selection and decision making procedure in more details for the first solution.

node, and from the comparison, the system can carry out the validation process. Finally, the X_1 data will be accepted or an alarm will be triggered. One must note that the test ontology can trigger generating any kind of test data for X_1 based on X_2. For example, after setting the ontology by a human expert, the ML agent in the test generator node takes ontology and customer dataset as inputs. Ontology determines what contextual information should be collected. In the above example, ML agent understands from the ontology that MSE is required to be collected for the FoI. So, the ML agent, by applying an appropriate algorithm, generates the MSE in the prediction of feature-1 using the three most related features to it. In the decision-making step, this MSE is compared with the ground-truth value. If the value of MSE is less than or equal to the ground-truth value, then the customer data is plausible and can be stored in the database, otherwise, the data is implausible and an alarm is raised.

Towards deploying the ML agent, we need to select an ML algorithm, prepare a train and test dataset, train it over train dataset, and test it over test dataset. To select an ML algorithm, we need to consider some points such as simplicity in usage, scalability, being model-free, explainability, resistance against overfitting and noise, resistance against non-available values in measurements, and working with categorical and continuous values. Regarding these tips, a random forest (RF) algorithm for regression is selected to be implemented in the ML agent. Investigation of the RF algorithm on our dataset for configuration of its parameter, i.e., number of estimator trees, showed us that the best performance, in terms of speed and overfitting, is achieved by 50 trees. Performance of the RF algorithms for plausibility check is investigated in subsection VI-A of the next section. Towards using RF algorithm, we train an RF agent based on several tests (out of 18 as described in the previous section), and then test this agent on a test dataset (excluding the training datasets).

B. Design of contextual information for plausibility check: The second solution

Not only does cross-correlation exists between columns of the dataset, but also auto-correlation among values of one column could be considered. It means that one can utilize the previous values of a column to check the plausibility of a specific value in this column.

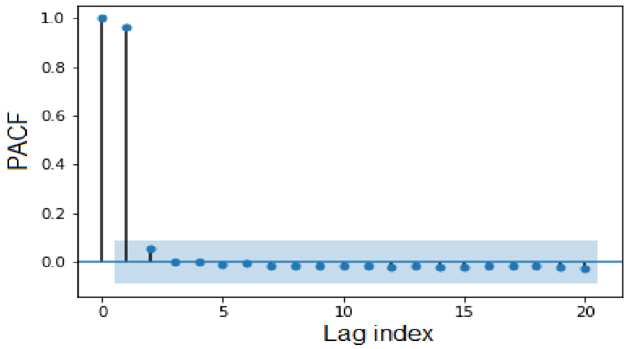


Fig. 14. Auto-correlation of feature 1 in the starting phase of test A

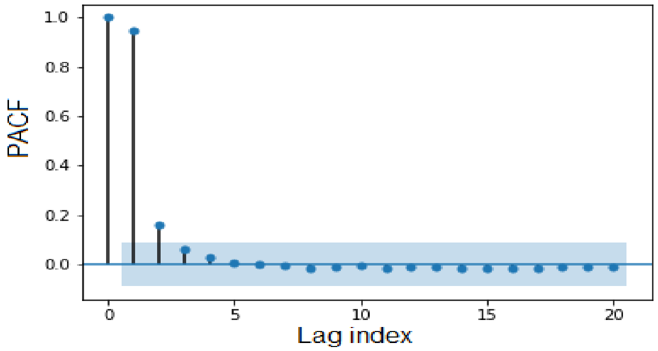


Fig. 15. Auto-correlation of feature 1 in the starting phase of test B

To see if auto-correlation could be used for the prediction of a feature from its lags, we consider one unique test and find the auto-correlation for each feature of this test. By using auto-correlation, we can find how a value of a feature in time t is related to the previous values of this feature at time $t-1$, $t-2$, $t-3$, ..., $t-n$. Figure 14 and Figure 15 depict the auto-correlation of E-spec in starting phase of test A and B respectively.

Since the values of auto-correlation for a specific feature of one test could randomly be high or low, we repeat auto-correlation for this feature over the 18 tests and average the

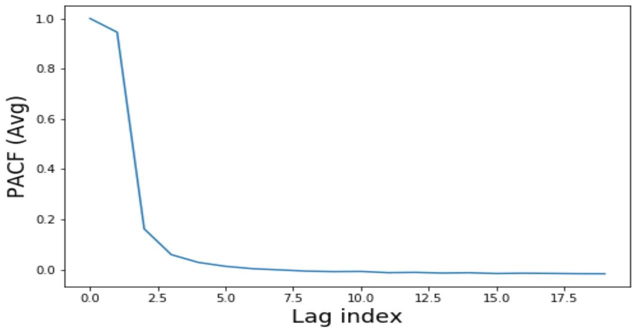


Fig. 16. Average of auto-correlation for feature 1 over different tests in the starting phase

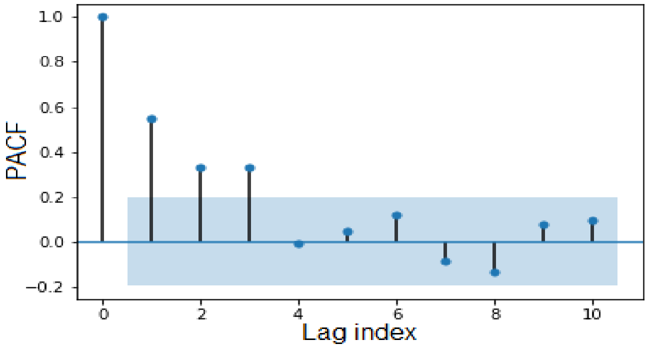


Fig. 17. Auto-correlation of feature 2 in the steady phase of test C

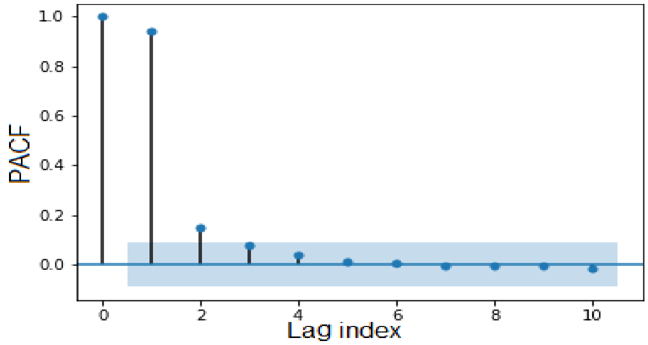


Fig. 18. Auto-correlation of feature 1 in the end phase of test C

values of these tests. So, Figure 16 is resulted. Then, among these averaged values, previous m recent values are selected for use in the ML agent. Since the behavior of features in the various phases follows different models, the auto-correlation function is calculated for each phase of a feature separately. Figure 17 and Figure 18 refer to the auto-correlation functions in the steady phase and end phase of feature 1 and 2 for the same test. Figure 19 and Figure 20 show the average of auto-correlation function over different tests in the steady phase of feature 2 and ending phase of feature 1.

Having access to the previous m recent values of a feature, we can train a machine-learning algorithm to predict the value of FoI at time t based on the previous values of the feature at time $t-1$, $t-2$, ..., and $t-n$. If the prediction value at time t based on the previous m recent values match the recorded data, there is low probability of implausibility, otherwise because of mismatch of prediction data and recorded one, an alarm could be raised. Figure 21 depicts the overall architecture of the second solution in more detail. In this figure, part of $X_1[0:N_2]$, e.g., $X_1[0:N_1]$ in which $N_1 < N_2$, is fed to the test data generator (Note: X_1 is the FoI.). Then, based on the test ontology, e.g., time series forecasting of X_1 using ARIMA, test data for the validity of $X_1[0:N_2]$ will be generated, e.g., $\tilde{X}_1$. Finally, at the comparator node, the real value of X_1 will be compared against $\tilde{X}_1$. Based on this comparison, X_1 data will be accepted or an alarm will be triggered.

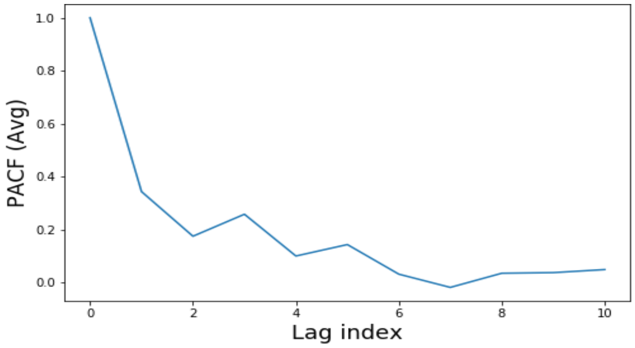


Fig. 19. Average of auto-correlation for feature 2 over different tests in the steady phase

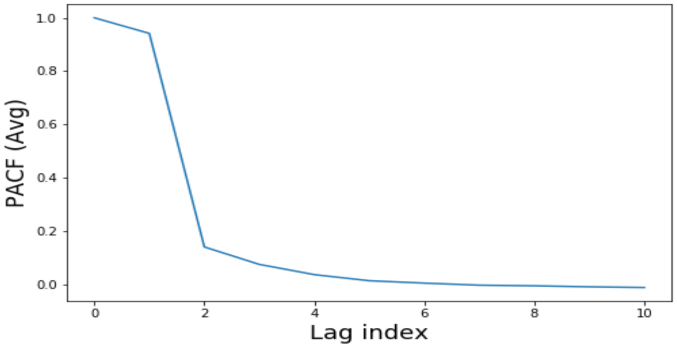


Fig. 20. Average of auto-correlation for feature 1 over different tests in the end phase

Toward deploying an ML agent for the second hypothesis, Random Forest (RF) and ARIMA algorithms are implemented. As mentioned in subsection V-A, we use the RF algorithm with 50 estimators for our test purpose. For the RF algorithm, the plausibility of each data point is checked based on the 10 lags of the data, i.e., $x[n]$ is checked based on $x[n\text{-}10]$:$x[n\text{-}1]$. For ease of notation, we call this RF algorithm as RF(50,10). For the ARIMA approach, the investigation of parameters on our dataset showed that P=3, Q=I=0, i.e., ARIMA(3,0,0) matches our dataset. Performance of ARIMA and RF algorithms for

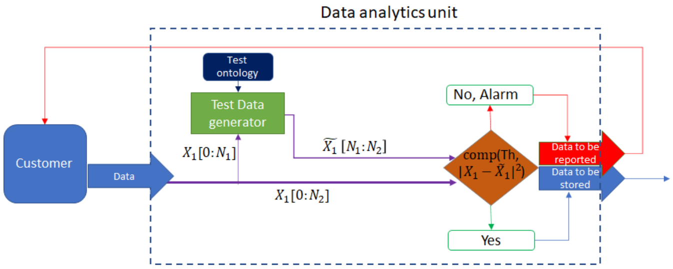


Fig. 21. Feature selection and decision making procedure in more details for the second solution.

plausibility check is investigated in subsection VI-B of the next section. Towards using ARIMA and RF algorithms, we train the ML agent based on several datasets (out of 18 tests), and then test these agents on a test dataset (excluding training tests).

C. *Design of contextual information for plausibility check: The third solution*

In the previous sections, we have leveraged the information in the features, either in the FoI or a combination of features, for plausibility check. In other words, the other contextual data gathered by the test maker related to the overall test have not been considered. In this section, we aim at investigating the impact of such contextual data on the statistics of FoI, and the potential application of such connection in plausibility check for the dataset. Figure 22 represents the overall structure of the proposed solution. In this figure, the metadata about X_1, which is the FoI, is fed to the test data generator along with X_1. Then, based on the test ontology, e.g., partitioning Cumulative Distribution Function (CDF) of X_1 based on states of the metadata, test data for the validity of X_1 will be generated, e.g., $\tilde{S}_{X_1}$. Finally, at the comparator node, the real value of S_{X_1} from received X_1, e.g., the average value of X_1 will be compared against the $\tilde{S}_{X_1}$. Based on this comparison, X_1 data will be accepted or an alarm will be triggered.

In our dataset, there are several contextual information corresponding to each unique test that potentially have impacts on the statistics of features. Examples of such contextual data include type of the *oil* and *separator metal* used in the experiment. Let us focus on oil. The initial hypothesis is that there is a connection between the type of oil used in a test and the statistics of measurements in this test. For example, the min, max, variance, median, mean values of distribution for Oil-A have considerable differences from the ones of Oil-B. Figure 23 and Figure 24 show the statistics for *feature-2*. One can observe that the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) of this feature are different for various oil types. Furthermore, the min and max values of this feature for *type-A* oil differ from *type-B* oil. So, using these explored statistics, we can add some rules to the ontology to discover the implausibility of the data. If the data would be implausible, the related statistics will change in comparison with the normal ones.

We train the metrics of decision-making using statistics of feature-2. If statistics of the test dataset comply with the statistics of the trained dataset, i.e., metrics like min, median, and variance are within the accepted bound found in the training, the decision-maker accepts the test data as plausible. Performance of plausibility check by using statistics of the data is investigated in subsection VI-C of the next section.

VI. Results And Discussion

A. *Performance test for the first solution*

Recall the first proposed solution in Figure 13. In this solution, the test ontology mandates predicting FoI for validity check based on the three most-related features. It also proposes MSE as the prediction analysis metric. Then, the three most related features to the FoI are fed to the test data generator and are used for predicting the FoI. Figure 25 shows the performance test results such that the prediction values are fitted well with the real values of FoI (here, feature-1).

In this figure, along with the test data and predicted data, the three most related features to feature-1 can be seen as well. One can observe that these three features have almost either direct or inverse (because of negative values of correlation) relationship with the feature of interest (feature-1). The above tests have been repeated for the steady phase and ending phase, and the same behavior has been almost observed for tests in these phases. We aim at leveraging the proposed ML agent

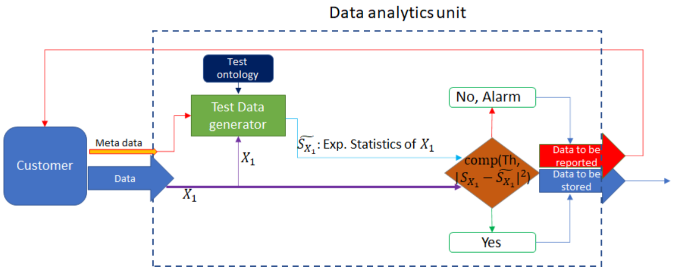


Fig. 22. Feature selection and decision making procedure in more details for the third solution.

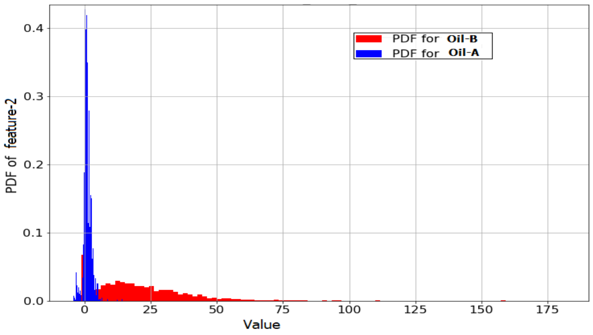


Fig. 23. PDF of *feature-2*

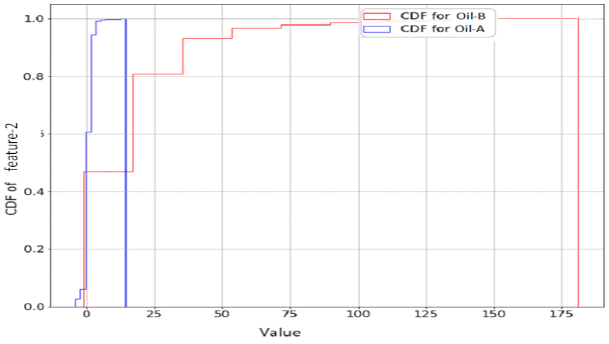


Fig. 24. CDF of feature-2

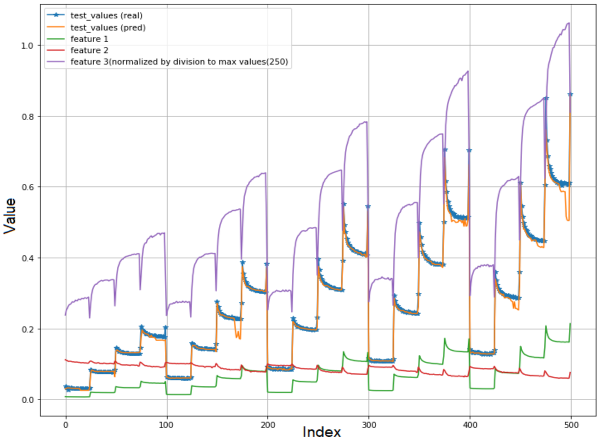


Fig. 25. Testing agent for predicting feature-1 with more details of 3 most related features

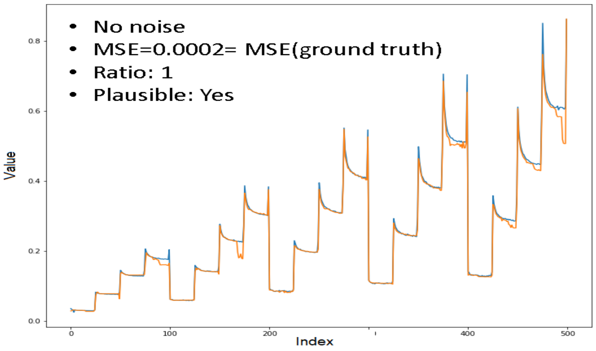


Fig. 26. Testing agent for predicting feature-1 based on 3 most related feature. No noise has been applied, MSE=0.0002= MSE (ground truth). (Blue: real data, Orange: predicted data)

for carrying plausibility checks out. So, seven test cases are presented based on applying bias measurement errors, and random measurement errors to the column of interest, and most related columns. The first plausibility check is related to the state that there is no noise in the data. As shown in Figure 26, plausibility of data has been confirmed. In second plausibility test, bias noise is added on the feature of interest (E-spec). From results of Figure 27, it can be observed that the predicted values are not the same as real values. So, the ML agent can detect the error on the data and conclude the implausibility of data. The third plausibility test, as shown in Figure 28, is related to the adding bias noise to the least related feature. In forth plausibility check, bias noise is added to the most related feature. The result is depicted in Figure 29. The same plausibility tests are done by adding random noise on the feature of interest, least related feature, and most related feature. The results of these tests are shown in Figure 30, Figure 31, and Figure 32, respectively.

B. Performance test for the second solution

Recall the second proposed solution in Figure 21. In this solution, the test ontology mandates predicting FoI for validity check based on its lags. It also proposes MSE as the prediction

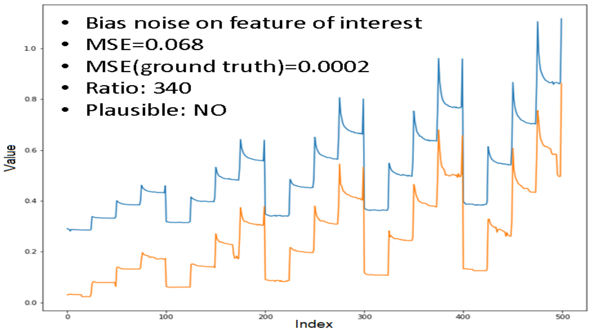


Fig. 27. Testing agent for predicting feature-1 based on 3 most related features. Bias noise on the feature of interest, MSE=0.068 (340 times more than ground truth). (Blue: real data, Orange: predicted data)

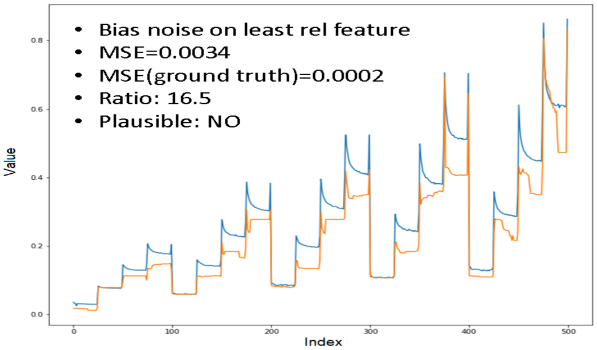


Fig. 28. Testing agent for predicting feature-1 based on 3 most related feature. Bias noise on the least related feature, MSE=0.0034 (16.5 times more than the ground truth). (Blue: real data, Orange: predicted data)

analysis metric. Then, the lags of FoI are fed to the test data generator, and are used for predicting the FoI. Figure 33 shows the performance test result using random forest for predicting feature-1 (FoI). In the random forest algorithm, we used 10 recent values of feature-1 for prediction. Figure 34 depicts the performance test result for predicting feature-1 using auto-correlation and ARIMA. In our implementation, ARIMA works with three recent values of feature-1. Both Figure 33 and Figure 34 confirm that the prediction values fit well with the real values of feature-1. We do the performance

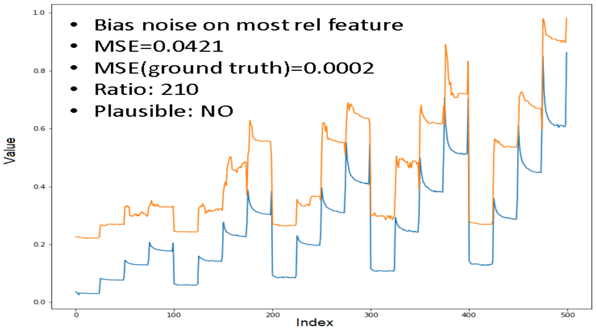


Fig. 29. Testing agent for predicting feature-1 based on 3 most related feature. Bias noise on the most related feature, MSE=0.0421 (210 times more than the ground truth). (Blue: real data, Orange: predicted data)

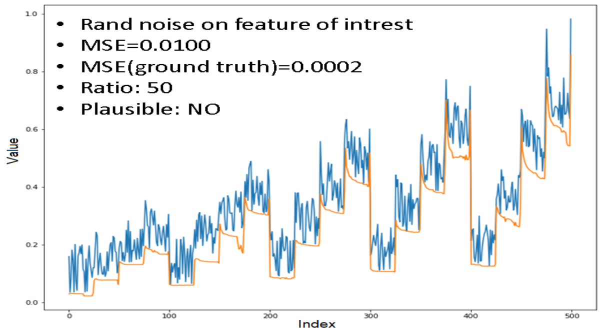


Fig. 30. Testing agent for predicting feature-1 based on 3 most related feature. Random noise on the feature of interest, MSE =0.01 (50 times higher than the ground truth). (Blue: real data, Orange: predicted data)

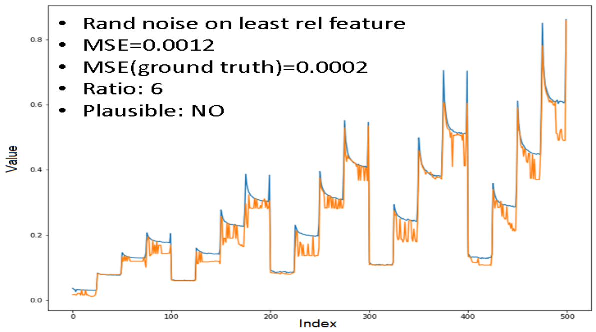


Fig. 31. Testing agent for predicting E-spec based on 3 most related feature. Random noise on the least related feature, MSE=0.0012 (6 times more than the ground truth).

test for the steady phase and ending phase of feature-1 using random forest and ARIMA algorithms and the results for these phases also follow the same trend.

For plausibility check using the auto-correlation contextual data, we apply bias measurement errors and random measurement errors to the FoI, and examine if the proposed solution can assess the incorrectness of data. Towards this end, we leverage the FoI's forecasting results using ARIMA and random forest methods. From Figure 35, Figure 36, Figure 37, and Figure 38 with random and bias noises, one can observe that the predicted values are not the same as real values of FoI. So, the ML agent can detect the error on the data and conclude the implausibility of the data. Table II summarizes the results of plausibility tests for the second solution.

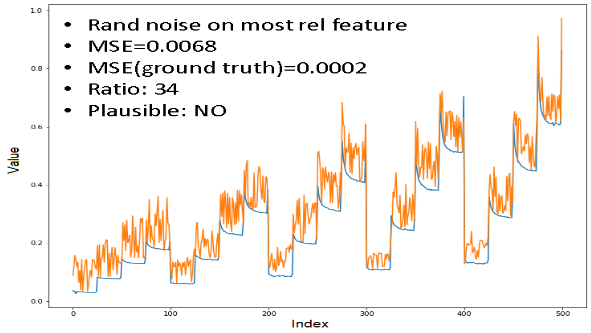


Fig. 32. Testing agent for predicting feature-1 based on 3 most related feature. Random noise on the most related feature, MSE= 0.00068 (34 times more than the ground truth). (Blue: real data, Orange: predicted data)

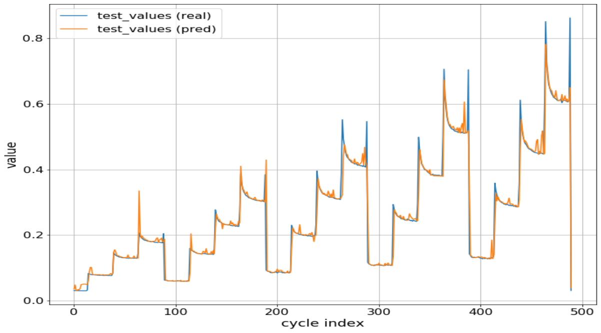


Fig. 33. Testing agent for predicting feature-1 using auto-correlation and random forest.

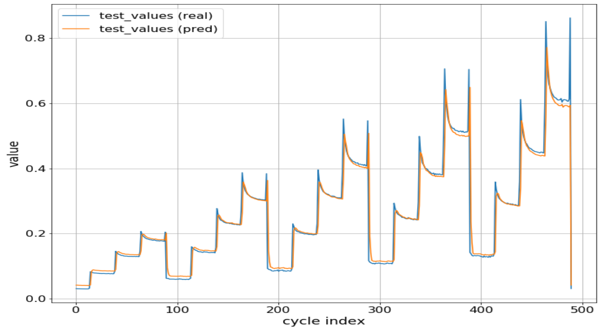


Fig. 34. Testing agent for predicting feature-1 using auto-correlation and ARIMA.

C. Performance test for the third solution

Recall the third proposed solution in Figure 22. In this solution, the test ontology collects metadata about FoI for validity check. Then, the past values of this feature are fed to the test data generator, and are used for extraction of statistics of this feature, and predicting the validity of the feature based on the extracted statistics. Here, we focus on the oil data and try to partition the PDF of FoI based on the type of oil used

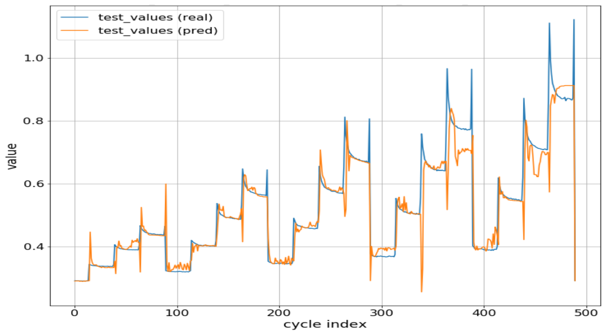


Fig. 35. Plausibility check for predicting feature-1 using auto-correlation, Random forest, and bias noise.

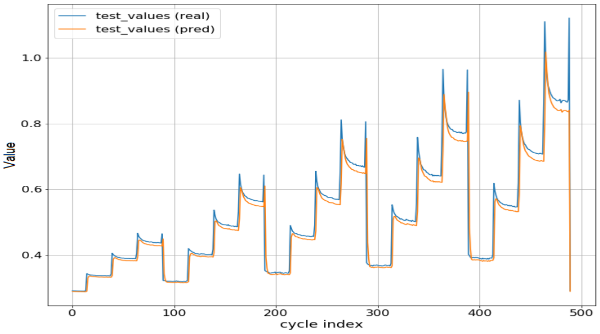


Fig. 36. Plausibility check for predicting feature-1 using auto-correlation, ARIMA, and bias noise.

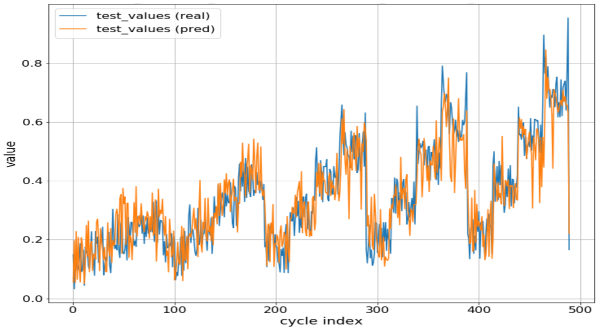


Fig. 37. Plausibility check for predicting feature-1 using auto-correlation, random forest, and random noise.

in the experiment. Figure 39 represents the partitioned PDF of the feature-2 (FoI) based on the type of oil used in the experiment. One observes the same trend from the test data and train data when there is no noise added to data (plausible test dataset).

In this section, we apply bias noise and random noise on the test data to check if our designed solution can detect the implausible data. Figure 40 and Figure 41 show the results of performance analysis for bias and random noise respectively. One observes in Figure 40 that adding the noise to the test data (red one) clearly shifts the plot to the right. Figure 41 represents the dataset with random noise. One can observe that the noise added to the data has changed the shape of PDF in both cases of bias and random noise, e.g., the mean and median have changed in Figure 40 and Figure 41 in comparison with the original data without noise shown in Figure 39.

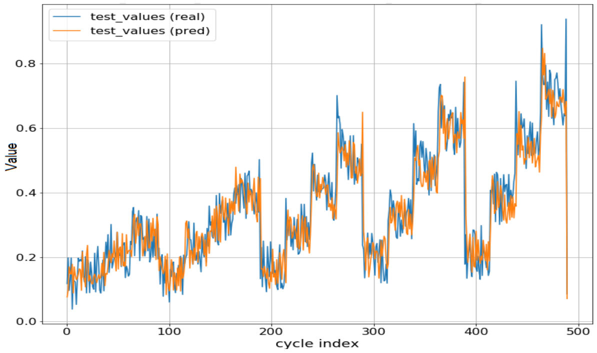


Fig. 38. Plausibility check for predicting feature-1 using auto-correlation, ARIMA, and random noise.

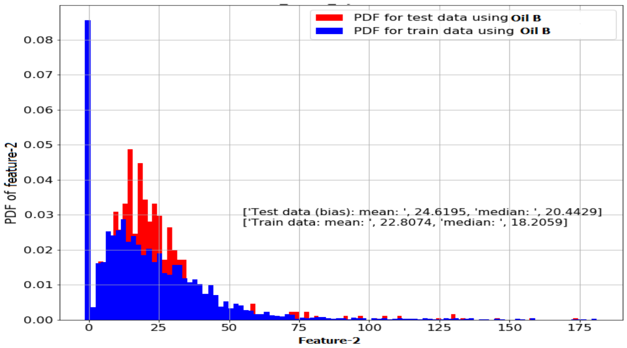


Fig. 39. Comparison of PDF of FoI in two tests

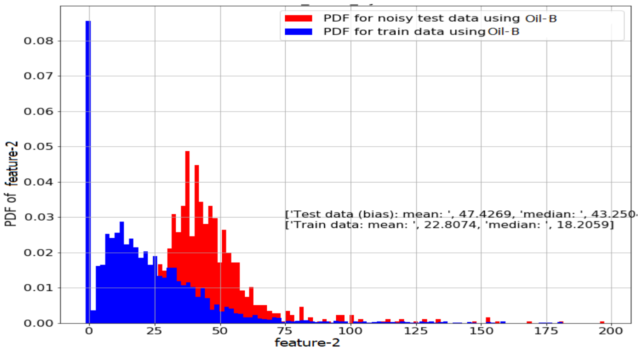


Fig. 40. Comparison of PDF of FoI with and w/o bias noise

D. Discussion

In Table I, the results of plausibility test for the first solution (subsection VI-B) have been summarized in more details. Table II summarizes the performance results for the second solution (subsection VI-B). Table III summarizes the results of Figures 39, 40, and 41 in subsection VI-B.

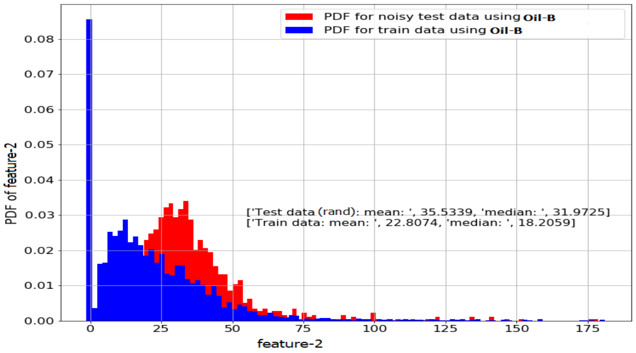


Fig. 41. Comparison of PDF of FoI with and w/o random noise

TABLE I
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 1

Test description	**MSE**	**MSE**$_{\text{ratio}}$: $\frac{\text{MSE}}{\text{MSE}_{\text{true}}}$	**Check**: $\text{MSE}_{\text{ratio}} < \text{Ratio}_{\text{th}}$; $\text{Ratio}_{\text{th}} = 1.5$
True data	0.0002	1	Y
Bias error on column of interest (feature-1)	0.068	360	N
Bias error on least related feature	0033	16.5	N
Bias error on most related feature	0.0420	210	N
Random error on feature of interest (feature-1)	0.010	50	N
Random error on least related feature	0.0012	6	N
Random error on most related feature	0.0068	34	N

TABLE III
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 3

Test description	**Mean**	**Mean-ratio**	**Median**	**Median -ratio**	**Plaus. check**
Train data (base measurement)	22.8	1	18.2	1	-
Test data w/o noise	24.6	1.08	20.4	1.12	Y
Test data with bias error	47.42	2.08	43.4	2.38	N
Test data with random error	35.8	1.57	31.7	1.74	N

From Table I, it is clear that the plausibility check solution, which is powered by the prediction of FoI based on the most related features, performs well against the bias noise. In other words, when a constant value, i.e., a measurement error, is added to the reading of a sensor, the plausibility check module can easily detect that data is inconsistent with the past learning (from 16.5 to 360 times more MSE has been reported). For the random noise, when the amount of the added noise to the data could vary, the performance is lower than the bias noise, but still completely acceptable (from 6 to 50 times more MSE has been reported). For example, one observes that the plausibility test has been shown 6 times more MSE in the prediction of FoI when random noise on the least relevant feature to the FoI has been added. Furthermore, Table II showed that the second solution (using RF) is not vulnerable to the random noise, and it performs equivalently for the bias and random noises (7 times more MSE in prediction of FoI). In the same time, we observe that the ARIMA has a poor performance as an ML agent for this solution, and it misses the alarm for the test-case with bias noise on the the FoI (the corresponding MSE-ratio is 1.125, which is lower than the threshold value, i.e., 1.5). Finally, the third approach shows a weaker performance than the previous ones (around two times more MSE has been reported). One must note that the stronger performance of the first approach and relatively the second approach is achieved at the cost of further computing required for them. In other words, there is a hidden reliability-complexity trade-off here, where going from solution 1 to 3, complexity is reduced and the probability of error in plausibility check is increased.

VII. Conclusions

In this work, we investigated data plausibility automation for a given dataset from a smart factory. Towards this end, a data analytics framework, consisting of a contextual data generation function (which generates checkpoints based on a given ontology) and a plausibility check function (which works based on the designed checkpoints), was proposed. For the implementation of the first function, we have investigated three machine learning approaches that leverage auto-correlation in each feature, correlation between features, and hidden statistics of each feature for generating the checkpoints. Performance evaluation results indicated the outstanding performance of the proposed schemes in the detection of noisy data. The main concluding remarks of this work include: (i) This study indicated that each feature of the dataset, or a collection of features, could be used without any other data for plausibility check leveraging machine learning. (ii) Metadata about the test, including conditions in which the test has been carried out, could be an important part of the design of the plausibility check. (iii) Checking of plausibility for a dataset that may contain random noise on some features (or some cycles) is much harder than checking the presence of static noise on the data. (iv) Performances of different checkpoint generation functions (using different ML approaches) are not the same. The ones based on the investigation of each cycle of the test, solutions 1 and 2, are more complex and provide a better distinction between noisy and healthy data. While the third solution is a lightweight solution with a lower reliability performance.

Acknowledgement

The research reported in this paper has been partially funded by BMK, BMDW, the State of Upper Austria in the frame of the COMET Programme managed by FFG and by the EC H2020 project *"DataCloud: Enabling the Big Data Pipeline Lifecycle on the Computing Continuum"* (Grant nr. 101016835).

References

[1] M. Basiri, J. Himmelbauer, L. Ehrlinger, and M. Matskin, "Context-aware data plausibility check using machine learning," in *The Fourteenth International Conference on Advances in Databases, Knowledge, and Data Applications*. DBKDA, 2022.

[2] V. Q. Nguyen, L. Van Ma, and J. Kim, "Lstm-based anomaly detection on big data for smart factory monitoring," *Journal of Digital Contents Society*, vol. 19, no. 4, pp. 789–799, 2018.

[3] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1939–1947.

[4] S. So, J. Petit, and D. Starobinski, "Physical layer plausibility checks for misbehavior detection in v2x networks," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 84–93.

[5] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2015, pp. 237–263.

TABLE II
SUMMARY OF PLAUSIBILITY CHECK USING SOLUTION 2

Test description	**MSE in prediction of FoI using itself by ARIMA**	**MSE$_{\text{ratio}}$ for ARIMA:** $\frac{\text{MSE}}{\text{MSE}_{\text{true-AR}}}$	**Plausibility for ARIMA:** MSE$_{\text{ratio}}$ $<$ Ratio$_{\text{th}}$; Ratio$_{\text{th}} = 1.5$	**MSE in prediction of FoI using itself by RF**	**MSE$_{\text{ratio}}$ for RF:** $\frac{\text{MSE}}{\text{MSE}_{\text{true-RF}}}$	**Plausibility for RF: MSE$_{\text{ratio}}$** $<$ **Ratio$_{\text{th}}$; Ratio$_{\text{th}}$** $= 1.5$
True data (feature-1)	0.0024 $=$ MSE$_{\text{true-AR}}$	1	Y	0.0012 $=$ MSE$_{\text{true-RF}}$	1	Y
Bias error on feature of interest (feature-1)	0.0027	1.125	Y	0.0046	7.8	N
Random error on feature of interest (feature-1)	0.0063	2.8	N	0.0088	7.3	N

[6] C. C. Aggarwal and S. Sathe, *Outlier ensembles: An introduction*. Springer, 2017.

[7] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, “A comprehensive survey of anomaly detection techniques for high dimensional big data,” *Journal of Big Data*, vol. 7, no. 1, pp. 1–30, 2020.

[8] L. Ehrlinger and W. Wöß, “Towards a definition of knowledge graphs.” *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, pp. 1–4, 2016.

[9] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[10] V. Roth, “Kernel fisher discriminants for outlier detection,” *Neural computation*, vol. 18, no. 4, pp. 942–960, 2006.

[11] D. Barbara, N. Wu, and S. Jajodia, “Detecting novel network intrusions using bayes estimators,” in *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 2001, pp. 1–17.

[12] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, “Iknn: Informative k-nearest neighbor pattern classification,” in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 248–264.

[13] A. H. Abuzaid, “Identifying density-based local outliers in medical multivariate circular data,” *Statistics in Medicine*, vol. 39, no. 21, pp. 2793–2798, 2020.

[14] G. Moreira, M. Y. Santos, J. M. Pires, and J. Galvão, “Understanding the snn input parameters and how they affect the clustering results,” *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 11, no. 3, pp. 26–48, 2015.

[15] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, “Clustering approaches for anomaly based intrusion detection,” *Proceedings of intelligent engineering systems through artificial neural networks*, vol. 9, 2002.

[16] S. Ali, G. Wang, R. L. Cottrell, and T. Anwar, “Detecting anomalies from end-to-end internet performance measurements (pinger) using cluster based local outlier factor,” in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 982–989.

[17] F. W. Scholz, “Maximum likelihood estimation,” *Wiley StatsRef: Statistics Reference Online*, 2014.

[18] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “A comparison of arima and lstm in forecasting time series,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[19] G. E. Box and G. C. Tiao, “A bayesian approach to some outlier problems,” *Biometrika*, vol. 55, no. 1, pp. 119–129, 1968.

[20] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, “An intrusion detection model based on feature reduction and convolutional neural networks,” *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[21] E. Parzen, “On estimation of a probability density function and mode,” *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[22] P. Vitányi, “How incomputable is kolmogorov complexity?” *Entropy*, vol. 22, no. 4, p. 408, 2020.

[23] L. Parra, G. Deco, and S. Miesbach, “Statistical independence and novelty detection with information preserving nonlinear maps,” *Neural Computation*, vol. 8, no. 2, pp. 260–269, 1996.

[24] A. Azari, P. Papapetrou, S. Denic, and G. Peters, “User traffic prediction for proactive resource management: learning-powered approaches,” in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[25] A. Sharma. (2020) Decision tree vs. random forest – which algorithm should you use? Accessed: 2022-12-15. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm

[26] N. F. Noy, D. L. McGuinness *et al.*, “Ontology development 101: A guide to creating your first ontology,” 2001.

[27] M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens, and C. Wroe, “A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 2,” *The university of Manchester*, vol. 107, 2009.