

DIGITAL OIL REFINERY: UTILIZING REAL-TIME ANALYTICS AND CLOUD COMPUTING OVER INDUSTRIAL SENSOR DATA



A Dissertation

by

Athar Khodabakhsh

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

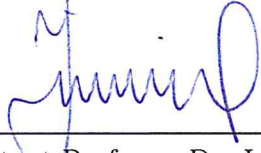
in the
Department of Computer Science

Özyeğin University
February 2019

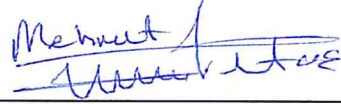
Copyright © 2019 by Athar Khodabakhsh

DIGITAL OIL REFINERY: UTILIZING REAL-TIME
ANALYTICS AND CLOUD COMPUTING
OVER INDUSTRIAL SENSOR DATA

Approved by:



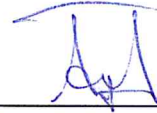
Assistant Professor Dr. Ismail Arı,
Advisor
Department of Computer Science
Özyeğin University



Associate Professor Dr. Mehmet
Aktaş
Department of Computer Science
Yıldız Technical University

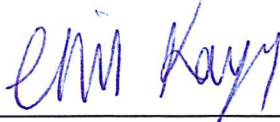


Associate Professor Dr. Murat Şensoy
Department of Computer Science
Özyeğin University



Associate Professor Dr. Ali Fuat
Alkaya
Department of Computer Science
Marmara University

Date Approved: 14 Dec 2018



Assistant Professor Dr. Enis Kayış
Department of Industrial Engineering
Özyeğin University



To My Beloved Parents

ABSTRACT

This thesis addresses big data challenges seen in large-scale, mission-critical industrial plants such as oil refineries. These plants are equipped with heavy machinery (boilers, engines, turbines, *etc.*) that are continuously monitored by thousands and various types of sensors for process efficiency, environmental safety, and predictive maintenance purposes. However, sensors themselves are also prone to errors and failure. The quality of data received from them should be verified before being used in system modeling or prediction. There is a need for reliable methods and systems that can provide data validation and reconciliation in real-time with high accuracy. Furthermore, it is necessary to develop accurate, yet simple and efficient analytical models that can be used with high-speed industrial data streams.

In this thesis, design and implementation of a novel method called DREDGE, is proposed and presented first by developing methods for real-time data validation, gross error detection (GED), and gross error classification (GEC) over multivariate sensor data streams. The validated and high quality data obtained from these processes is later used for pattern analysis and modeling of industrial plants. We obtained sensor data from the power and petrochemical plants of an oil refinery and analyzed them using various time-series modeling and data mining techniques that are integrated into a complex event processing (CEP) engine. Next, the computational performance implications of the proposed methods are studied and regimes that are sustainable over fast streams of sensor data are uncovered.

Distributed Control Systems (DCS) continuously monitor hundreds of sensors in industrial systems, and relationships between variables of the system can change over time. Operational mode (or state) identification methods are developed and presented

for these large-scale industrial systems using stream analytics, which are shown to be more effective than batch processing models, especially for time-varying systems. To detect drifts among modes, predictive modeling techniques such as regression analysis, K -means and DBSCAN clustering are used over sensor data streams from an oil refinery and models are updated in real-time using window-based analysis. In addition, the shifts among steady states of data are detected, which represent systems' multiple operating modes. Also, the time when a model reconstruction is required is identified using DBSCAN algorithm. An adaptive window size tuning approach based on the TCP congestion control algorithm is proposed, which reduces model update costs as well as prediction errors.

Finally, we proposed a new Lambda architecture for Oil & Gas industry for unified data and analytical processing over DCS. We discussed cloud integration issues and share our experiences with the implementation of sensor fault detection and classification modules inside the proposed architecture.

ÖZETÇE

Bu tezde büyük-ölçekli ve görev-kritik işlerin yürütüldüğü endüstriyel tesislerin büyük veri problemleri ele alınmaktadır. Bu tesislerde çalışan ağır sanayi makineleri (kazanlar, motorlar, türbinler, vb.) binlerce sayıda ve çeşitli tipte duyargalar ile sürekli olarak ölçümlenmekte ve üretimde verimlilik artışı, iş-çevre güvenliği ve sezgisel bakım planlaması gibi konularda kararlar alınmaktadır. Ancak duyargalar da bozulabilmekte veya hatalı ölçümler yapabilmektedirler. Bunlar tarafından ölçülen verinin kalitesi, sistem modellemesi veya tahminlemede kullanılmadan önce doğrulanmalıdır. Bu sebeple, gerçek-zamanlı ve yüksek doğrulukla veri düzeltmesi yapabilecek güvenilir metodlara ihtiyaç duyulmaktadır. Geliştirilen metodların çok-hızlı endüstriyel veri akışlarını takip edebilmeleri için, doğrulukları kadar, basit ve etkin olmaları da gerekmektedir.

Bu tezde DREDGE isimli, öncelikle gerçek-zamanlı veri doğrulama, kaba hata tespiti ve kata hata sınıflandırması yapabilen yenilikçi metodun tasarım ve gerçekleştirme bilgileri sunulmaktadır. Bu süreçler sonrasında elden edilen doğrulanmış ve yüksek kaliteli veri, desen analizi ve endüstri tesislerinin bütünsel modellemesi için kullanılmaktadır. Çalışmada bir petrol rafinerisinin güç üretim ve petrokimya tesislerinden elde edilmiş gerçek duyarga verileri zaman-serisi ve veri madenciliği modellerinin eğitiminde kullanılmış ve elde edilen modeller Karmaşık Olay İşleme motoruna entegre edilmişlerdir. Daha sonra, bu modellerin motor içerisindeki performansları çalışılarak, hızlı veri akışları üzerindeki sürdürülebilir rejimlerin tespiti sağlanmıştır.

Sonrasında, büyük-ölçekli endüstriyel sistemlerin operasyonel durum tespiti için akış analitiğine dayalı metodlar geliştirilmekte ve özellikle zamana-bağlı değişen sistemlerde, toplu işleme yapan modellere göre daha etkin çalıştıkları gösterilmektedir.

Dağıtık kontrol sistemleri sürekli olarak üzlerce duyargayı takip etmekte ise de, değışkenler arası ilişkiler zaman içerisinde değışebilmektedir. Petrol rafinesindeki cihazların modlar arası geçişlerini farkedebilmek için veri akışları üzerinde gerçek-zamanlı ve zaman-penceresi temelli regresyon analizi, K -means ve DBSCAN kümeleme sezgisel modelleme teknikleri kullanılmıştır. Ayrıca, durağan-durumlar arası kayışlar tespit edilerek, operasyonel mod geçişleri tespit edilmiştir. Gerçek-zamanlı DBSCAN kullanarak model değışikliği veya düzenlemesinin gerektiği anlar tespit edilmiştir. Bu bölümde son olarak, zaman-pencere boyutlarının TCP algoritmaları ile adaptif olarak değıştirilmeleri önerilmiş ve model güncelleme maliyeti ile tahminsel hatalarına olumlu etkileri gözlemlenmiştir.

Son olarak, rafineri endüstrisi için dağıtık kontrol sistemleri (DCS) verilerine yönelik, birleşik (çevrimiçi-çevrimdışı) analitik işleme içeren yeni bir Lambda mimarisi önerilmektedir. Önerilen mimarinin buluta entegrasyonu ile içerisindeki sensör hata tespiti ve sınıflandırmasına yönelik modül geliştirilmesine yönelik tecrübe paylaşımı yapılmaktadır.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Professor Ismail Ari for his continuous support of my Ph.D. study and research, for mentoring me through the learning process, for his patience, motivation, and immense knowledge. His guidance helped me at all times during my research and writing of this thesis. Besides my advisor, I would like to thank the rest of my thesis committee: Professor Murat Şensoy, Professor Enis Kayış, Professor Mehmet Aktaş and Professor Ali Fuat Alkaya for their time and valuable comments. I would also like to thank Professor Ali Özer Ercan and Professor Murat Şensoy for their insightful comments and encouragement during my progress report sessions, but also for the questions which incited me to widen my research from various perspectives.

And special thanks to my family. Words cannot express how grateful I am to my father and mother for all of the sacrifices that they have made on my behalf. Special gratitude to my father who is my moral role model through my life and my mother who taught me how to be ambitious. I thank my brother Ali who is a great encouragement and not just a brother but my best friend and my sister for her kind support through all these years.

I would like to thank all my fellow labmates from past, Buse Yilmaz, Ugur Kocak, Erdi Olmezogullari, Erkan Kemal and current labmates, Gokce Guven, Kaan Meneksedag, Can Altun, Faizaan Siddiqui and all my friends for stimulating discussions, working together, and for all the fun we have had in the last few years. I am also thankful to all the staff at Özyegin University for giving us an opportunity to work with wonderful people in a warm and friendly environment. This research was sponsored by a grant from TÜPRAŞ (Turkish Petroleum Refineries Inc.) R&D group.

TABLE OF CONTENTS

DEDICATION	iii
ABSTRACT	iv
ÖZETÇE	vi
ACKNOWLEDGEMENTS	viii
LIST OF TABLES	xii
LIST OF FIGURES	xiii
I INTRODUCTION AND PROPOSED TASKS	1
1.1 Problem Definition	2
1.2 Proposed Tasks	3
II BACKGROUND AND RELATED WORK	8
2.1 Data Validation and Reconciliation	11
2.1.1 Steady-State System Model	12
2.1.2 Time-Varying Kalman Filter	14
2.2 Time-Series Analysis	16
2.2.1 Exponential Smoothing	17
2.2.2 Auto Regressive Moving Average (ARMA)	18
2.3 Modeling System Using Data Mining Techniques	19
2.3.1 Regression Analysis Method	20
2.3.2 Classification Techniques	20
2.3.3 Clustering Techniques	21
2.4 Related Work	21
III MULTIVARIATE SENSOR DATA ANALYSIS FOR OIL REFINERIES	28
3.1 Description of CPS for Oil Refineries	28
3.2 Methodology in DREDGE	32
3.2.1 Steady-State and Instantaneous Mass Balance (IMB)	33

3.2.2	Capturing System Memory Using ARMA Model	35
3.2.3	Time-Varying Kalman Filter (KF)	37
3.2.4	DREDGE: Novel Approach	37
3.2.5	Gross Error Classification (GEC): Novel Approach	40
3.2.6	Rule-Based Algorithm for Labeling Gross Error Types	41
3.3	Experimental Results	43
3.3.1	Results for Synthetic Data	44
3.3.2	Results for Real Refinery Data	47
3.3.3	Computational Performance Evaluation	53
3.4	Holt-Winters vs. ARMA	58
3.5	Complexity of Algorithms	60
IV	SYSTEM OPERATIONAL MODE IDENTIFICATION AND MODEL UPDATES	61
4.1	Sub-model Analysis and Parameter Estimation	61
4.2	Operational Mode Identification Methodology	62
4.2.1	Regression Analysis Method	63
4.2.2	<i>K</i> -means Clustering	64
4.2.3	DBSCAN Clustering	64
4.2.4	System Operational State Analysis in Real-time	65
4.3	Experiments and Results	67
4.3.1	Regression Model Evaluation	68
4.3.2	<i>K</i> -means Clustering vs. RM Mode Identification	70
4.3.3	Density-Based Clustering	71
4.3.4	Using DBSCAN for Mode Change Detection	72
4.3.5	Using Two-Term Gaussian Function for Mode Change Detection	78
4.3.6	Discussion: Sensor Error or System Anomaly	80
4.4	Adaptive Window Size Tuning	80
4.5	Summary and Conclusions	83

V	A CLOUD-BASED BIG DATA PROCESSING ARCHITECTURE FOR INDUSTRY 4.0	84
5.1	Cloud Computing	85
5.2	Private Cloud Implementation	87
5.2.1	Spark Cluster	90
5.2.2	Machine Learning Library: MLlib	90
5.2.3	Kafka Streaming Platform	91
5.3	Fault Detection and Classification Modules	92
5.4	Public Cloud Implementation	94
5.4.1	Spark on Amazon EMR	95
5.4.2	AWS-Kinesis and AWS-S3	96
5.5	Summary and Conclusion	96
VI	CONCLUSIONS AND FUTURE WORK	99
	REFERENCES	101
	VITA	112

LIST OF TABLES

1	Sample data for y_t , input/output flow values (<i>ton/hour</i>) of one of the boilers. The complete data spans 5 months worth of measurements from 12/2014 to 4/2015.	34
2	Sample flow measurements for 17 sensors y_t (<i>ton/hour</i>) of the petrochemical system of Figure 6. The complete data spans 2 months worth of measurements from 08/2014 to 10/2014.	34
3	A result of data reconciliation using IMB method.	35
4	GED results of IMB, KF, and DREDGE for Bias-type gross error over synthetic data.	44
5	GED results for IMB, KF, and DREDGE over different gross error types on synthetic data.	46
6	F-measure for GEC results of CDT, NN, and KNN over different gross error types on Real and Synthetic (Synt) data.	47
7	GEC results for CDT, NN, and KNN over different gross error types on real data.	47
8	GEC results of IMB, KF, and DREDGE over real dataset.	50
9	Values are average of RMSE, comparing two model over real data, using 1-4 days data for modeling.	58
10	R^2 values of fixed-window size for regression model prediction performance.	68
11	Gaussian parameter analysis in fixed-window size for sub-model detection.	78
12	R^2 , RMSE values of adaptive-window size for regression model prediction performance.	83

LIST OF FIGURES

1	Overview of the organization and contribution for GED and GEC. . .	5
2	Data quality dimension chart.	10
3	Common types of gross error.	13
4	Illustration of the cyber-physical systems inside the oil refinery; high-level operation of the power plant, petrochemical plant for crude oil processing, and data stream processing services.	29
5	Illustration of a boiler power plant. Various types of sensors are implanted for real-time data collection. The boiler can change states among the desired stream pressure levels (low, high, very-high) or go automatically into heating, cooling, recycling, and condensing modes. KBS: cold water input; VHP: Very-high power steam output; DSH: De-Super Heater.	30
6	Petrochemical process showing crude oil input, preflash, debutanizer, atmospheric columns, and sensors (1-17) measuring input and output flows.	36
7	Different types of gross errors are inserted on top of refinery power plant real data. Blue dots show that gross errors can correctly detected using KF. These error are then classified by mean-variance tracking.	41
8	Categorization of gross error types with respect to changes in mean and variance of sensor values.	43
9	Snippet of the CEP + DVR development.	45
10	(a) GED for the water/vapor lines of the Power Plant data, (b) GED for debutanizer lines 3=5+6+7 of the Petrochemical process data. . .	48
11	(a) Water temperature/pressure lines of power plant used for GED, (b) Vapor temperature/pressure lines of power plant used for GED, by DREDGE.	49
12	(a) GEC for errors detected by IMB, (b) GEC for errors detected by KF.	52
13	Power Plant: computational loads of reference queries and GED algorithms with respect to total memory usage (MB) for different window sizes over streaming data.	55
14	Power Plant: total processing time (<i>sec</i>) for different window sizes over streaming data.	55

15	Petrochemical Plant: computational loads of reference queries and GED algorithms with respect to total memory usage (MB) for different window sizes over streaming data.	56
16	Petrochemical Plant: total processing time (<i>min</i>) for different window sizes over streaming data.	56
17	ARMA model: Real Data Results.	59
18	Holt-Winters Smoothing: Real Data Results.	59
19	Model evaluation of pressure/temperature data.	66
20	Regression model on streaming sensor data, (a) in windows #1→#2 RM models are close and system works in the same operational mode, but in (b) window #3 data requires a new RM sub-model since the distribution is changed and shows a transition among system's operational modes, (c) system clearly drifted to a new operational mode with updated RM in window #4, and (d) window #5 stays in the new operational mode.	69
21	(a) Comparison of RMSE values of regression sub-model from previous window and current streaming window, (b) window-based stream data used for operational modes identification analysis depicted for windows #1→#10.	70
22	<i>K</i> -means clustering is also used for operational mode identification, in (a) window #3, observations are clustered into two partitions in a transient mode, (b) window #4 observations consist of two clusters where one centroid has a close distance to previous centroid from window #3.	71
23	DBSCAN clustering, (a) in window #1 two clusters are identified in steady-state mode outliers and inliers, in (b) window #2 the system is under same operational state as window #1, but in (c) window #3 data points are partitioned into 3 separate clusters since the distribution is changed and a transient in system operational mode has occurred, (d) system clearly drifted to a new operational mode.	73
24	DBSCAN model on streaming sensor data water/vapor flow rate: (a) in window #8 (refer to Figure 19), DBSCAN model shows formation of one cluster of inliers indicating one operational state, (b) as new data samples are recieved from window #9, data gets split into 2 main clusters and some outliers that indicates a drift in data context and that the system is in a transient mode. (c)-(d) Windows #10→#11 show that the system is operating under a new steady-state mode.	74

25	DBSCAN model on streaming sensor data water/vapor, pressure/ temperature: similar to flow rate, (a) in window #8 DBSCAN model shows formation of one cluster of inliers indicating one operational state, (b) in window #9, data split into 2 main cluster and some outliers that is a distinctive indication of drift in data context and the system goes into transient mode, requiring a model update. (c)-(d) Windows #10→#11 show that the system is operating under a new steady-state mode. . .	75
26	Comparison of regression model result on (a) batch analysis, (b) window-based analysis. Although a single regression model describes the data, operational modes of the system are not distinguished by batch analysis.	76
27	Histogram of input and output flow rates (<i>ton/hour</i>).	77
28	Gaussian distribution model on streaming sensor data, (a)-(b) in windows #1→#2 Gaussian curves are close and system works in the same operational mode, but in (c) window #3 data requires a new Gaussian sub-model since the distribution is changed and shows a transition among system's operational modes, (d) system clearly drifted to a new operational mode with updated curve in window #4.	79
29	Adaptive window size tuning over data.	82
30	Three cloud service layers.	86
31	Proposed Lambda architecture for Oil & Gas data analysis.	89
32	The different components of Spark.	91
33	Kafka cluster's core API.	92
34	Running fault detection application on Spark cluster.	93
35	AWS architecture for stream processing.	96
36	Running application on AWS-EMR.	97

CHAPTER I

INTRODUCTION AND PROPOSED TASKS

In modern industrial plants hundreds of sensor data streams measuring flow rates, temperature, pressure, *etc.* are recorded automatically for purpose of process control and system evaluation. The quality of data, whether processed online or stored, affects the performance of successively used monitoring and system control software. In industrial systems measurements always contain errors that affect and sometimes invalidate the process model. Measurement errors can occur during data acquisition, data processing, and data transmission. The difference between measured and true value is caused by two types of errors: *random error* and *gross error* which in turn reflect to the quality of streaming data.

While managing high “volume” and large “variety” of data have always been a challenge for IT, its increasing “velocity” is the new issue of the 21st century due to increased use of sensors and “things” in industrial plants. Attempting to store these data first to analyze them later creates additional costs, unwanted delays to actionable information, and mishandling of threats or opportunities. Fortunately, there are now tools to process data *on-the-fly* as they move from DCS to selected destinations. Yet, adding the challenges of “veracity”, *i.e data correctness factor*, creates the ultimate Big Data problem (with “4Vs”) for industrial plants. Neither relational databases nor distributed batch processing systems alone are designed to cope with this problem. Accordingly, three concerns of utmost importance for mission-critical industrial systems are:

- process efficiency,
- long-term reliability,

- and continuous safety.

Toward achieving all of these goals, it is necessary to continuously monitor and verify the accuracy of measurements streaming in, from numerous and various types of sensors placed around the plants.

1.1 Problem Definition

In an oil refinery, everything happens in big proportions: liquids flow in *ton/hour* rate, temperatures are measured in hundreds to thousands °C degree, and electricity is produced in megawatts. Thousands of people and millions of dollars are at stake every moment as one tiny malfunction or mistake in the system can cause serious damage to the entire plant and the workers, or generate losses in revenue. Thus, achieving continuous safety, process efficiency, long-term durability and planned (vs. unplanned) downtimes are among the main goals for industrial plant management.

Due to mission-criticality of industrial processes, Oil & Gas businesses have already implanted thousands of sensors inside and around their physical systems [1]. Raw sensor data continuously streams via distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems measuring temperature, pressure, flow rate, vibration, level, *etc.* of drills, turbines, boilers, pumps, compressors, and injectors.

Another aspect in real-time system identification, is updating the models according to currently received pattern from stream data [2]. Since the systems are dynamic and the quality of models are dependent on both quality of data and system model, it is crucial to assess the current context of the system as the relations among model variables can change over time.

Achieving all of these goals, necessitate to continuously monitor and verify the accuracy of measurements streaming in from numerous and various types of sensors

placed all around the refinery. However, sensors are also prone to failures and measurement errors. At normal operation, sensor measurements are assumed to be noisy (*i.e.* to have random errors). Electro-mechanical effects such as malfunctioning, uncalibrated or broken sensors and human factors also introduce systematic errors (*a.k.a.* “gross errors”) to these measurements. Ability to differentiate sensor errors from real system abnormalities is crucial for the safety of the plants. However, there is lack of real-time industrial Data Reconciliation (DR) and Gross Error Detection (GED) software and data services that can be used by oil refineries.

The underlying platform for such data processing and efficient management in real-time becomes a challenge in large-scale industrial plants, thus design and implementation of a cloud platform service is required to provide performance and elastic scalability for the digital plants [3].

In summary, the scientific problems faced in modeling and analysis of industrial systems are as follows:

1. multivariate industrial data validation and reconciliation (DVR) problem,
2. data analysis and system modeling in real-time,
3. model accuracy problem for dynamic systems,
4. elastic scale for multivariate, high dimensional and fast sensor data processing.

1.2 Proposed Tasks

This thesis presents big data challenges seen in large-scale mission-critical industrial plants equipped with thousands of sensors of various types. In this regard, I propose two main approaches to acquire veracity aspect of sensor data. In the first approach, data validation and reconciliation (DVR) technique [4] is proposed for sensors that are prone to errors and failure to satisfy plants models such as Mass Balance equilibrium for improving the accuracy of data. For this purpose, the process model equations such as equilibrium relation and conservation law is used to perform data

reconciliation and gross error detection. Regimes (e.g. windows sizes) are found to be sustainable over fast streams of sensor data to improve the performance [5]. In the second approach, gross error detection and error classification solution is proposed, which combines techniques for time-series analysis such as the ARMA model with signal processing tools such as Kalman Filter, integrated into a commercial Complex Event Processing (CEP) engine. In addition, the shifts among different states in datasets are detected to represent systems' multiple operating modes, and for identifying the time when a model reconstruction is required. On the other hand, design and implementation of a cloud platform service [6] is proposed to provide scaling characteristics for fast streaming data in industrial systems.

To motivate and demonstrate the use of real-time multivariate data analysis, we obtained time-series data from the power and petrochemical plants of a real refinery with approximately 11.5 million *ton/year* processing capacity [7]. Contributions of this thesis can be summarized as follows:

- First, multivariate data including flow, temperature, pressure is analyzed using ARMA time-series modeling and synthetic datasets are generated for ground truth tests.
- Second, accuracy of three GED models are compared: an optimizer using Instantaneous Mass Balance constraint (IMB), Kalman Filter (KF), and a new Kalman Filter with Unity Gain constraint (KF-UG counterpart) over real and synthetic sensor data.
- Third, for the detected errors, gross error classification (GEC) is applied using a Complex Decision Tree (CDT), neural network (NN), and K-Nearest Neighbor (KNN) algorithms, and sensor errors are classified into four types called Bias, Drift, Precision Degradation and Failure [8], [9] illustrated in Figure 1. Applying GEC on top of GED is crucial for oil refineries, and it is demonstrated that

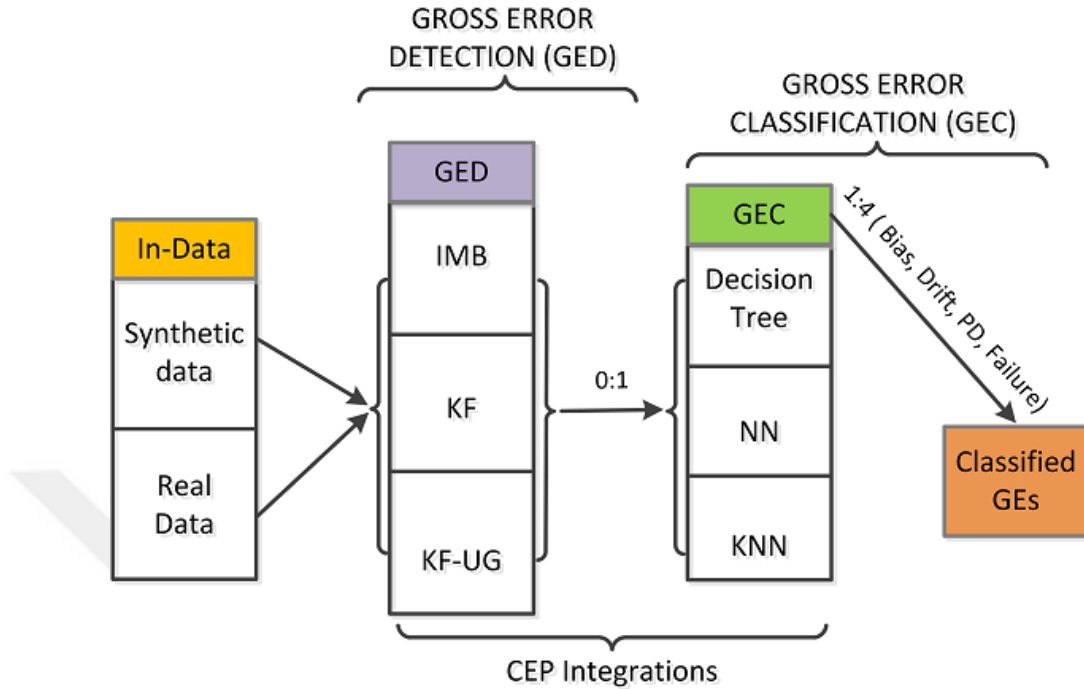


Figure 1: Overview of the organization and contribution for GED and GEC.

accuracy of GED techniques can vary per error type. Each error type requires a relevant action on sensors: some error types are fixable whereas others aren't and need sensor replacement.

- Fourth, the trained DR-GED, GEC models are integrated into a complex event processing (CEP) engine and the accuracies and performances of the proposed techniques are verified over real refinery datasets.
- Finally, validated data is used for analysis of changing system modes. The aim of operational mode analysis is to identify the time where operation shifts from one steady-state to another and a model reconstruction is required. The context shift is analyzed based on fluctuations in streaming sensor data using DBSCAN clustering.

I propose to solve the described scientific problems using the following approaches:

1. Select and integrate a set of analytical tools from different domains such as

statistics, signal processing, and distributed systems, as a solution for the challenges faced in multivariate data reconciliation problem.

2. Evaluate gross error detection (GED) accuracy for three different tools that are applied for DVR: Instantaneous Mass Balance using Mass Balance constraint (IMB), Kalman Filter (KF), and Kalman Filter with unity-gain (DREDGE) over controlled datasets.
3. Real-time data validation (*i.e.* detect gross errors) and reconciliation, integrated into a complex event processing (CEP) engine, for classifying gross errors into four different types including bias, drift, precision degradation and total failure [5, 6].
4. Using regression model for analysis of changing modes in the system and validate results using clustering techniques such as K -means and DBSCAN methods.
5. Integrate these models into a real-time CEP engine for studying sustainability of the proposed techniques at scale as a cloud platform service, (Amazon AWS [10]) and implementation on the Private Cloud services.
6. Using modern open-source distributed cluster-computing framework like Apache Spark in real-time, scalable (multi-core machines) for implementation of Big Data engines [11].

Overall, selection and synergetic use of a set of analytical tools from different domains including data mining, statistics, and distributed systems are discussed to address challenges faced in petrochemical industry. The aim of this thesis is not only detecting outliers and improving the accuracy of data but also recognizing the state of physical sensors as well as industrial systems and identifying the time of model reconstruction.

I hope the contributions in this thesis for oil refineries will also contribute to smarter industrial models, Industry 4.0 and digital transformation [12] efforts of other future industries.



CHAPTER II

BACKGROUND AND RELATED WORK

Data-intensive computation have been recently considered as an important paradigm for science. Two factors in this trend are: (1) vast amount of data obtained from applications and; (2) the infrastructures that provide storing these vast amounts of data, sharing data and data processing [13]. Every day 2.5 quintillion bytes of data is created by users and about 90% of available data is generated and collected in the last two years. These data are collected from sensors, Internet of Things (IoT), Industrial IoT (IIoT), social media, digital media, business transactions to name a few. This data is called Big Data. We face new waves and innovations for analyzing large datasets. Big Data is not simply defined with large size of data but allowing new content available for scientific and business requirements which were beyond our reach in past [14].

The current needs for better control, monitoring, and management in many areas by development of IoT, have originated the appearance and creation of multiple systems like smart-home, smart-city, smart-grid, and smart-factories. In general smart-cities rely on sensors and a vast number of sensing devices for integrating received data into underlying platform for relevant data processing [15]. IoT devices are limited in storage and computational power and for complex analysis, they require technology of Cloud Computing. While IoT generates high amount of varied data, Cloud technologies are available to gather and process data with almost infinite computational and storage power [14].

On the other hand, IoT is enabling numerous and various types of sensors to be placed around critical industrial facilities, thus allowing distributed data collection

in real-time. Specifically, IIoT devices collect data from complex physical devices and instruments with time-varying and possibly nonlinear behavior [16]. Estimated cyber models for collected data are expected to help industries with process efficiency tracking, load forecasting [17], predictive maintenance [18] decisions, anomaly detection [19], reducing factory downtimes and increasing safety as well as profits. Thus, it is necessary to develop accurate, yet simple and efficient models that can be used with high-speed industrial data streams [20], [21].

In this regard, artificial intelligence is becoming popular in smart-factories and industry 4.0 revolution for communication between machines and components for production processes. Intelligent systems are used in digital control of facilities for quality insurance and the sensors are implanted in smart-factories and machine learning uses the data received from them for analysis, maintenance, and future predictions. Reducing costs and improving the safety of pipelines, machine failure prediction before it actually happens, improving equipment reliability for preventing potential downtimes, and identifying significant changes in data to give alert before damage occurs are few use cases of real-time onsite intelligence in Oil & Gas smart-industry [22].

Data is the main resource in industrial and business processes that should be utilized to support decision making (*i.e.* decision support systems). The quality of data has a high value when it can provide timely and relevant information to organizations. Therefore, data quality is a challenging problem when data quantity is increasing with poor quality. “Fitness for use” is a comprehensive definition for data quality requirements beyond traditional concerns on data accuracy. We need to develop cost-effective solutions for data quality assurance that can sustain over high volumes [23], [24]. This is a multidimensional concept assuring suitability according to the nature of a problem [25].

Six dimensions of data quality depicted in Figure 2 can be described as follows:

- **Accuracy:** Data conforms to its correct value for the object or event being



Figure 2: Data quality dimension chart.

described.

- **Completeness:** Information is not missing any items and/or is sufficient for the task.
- **Validity:** Data complies with its defined syntax (format, type, range).
- **Consistency:** Representation of data values remain the same in multiple locations and forms.
- **Timeliness:** Data is up to date and it is available on time.
- **Integrity:** The relation between entities and attributes is consistent.

The problem of data quality and anomaly detection we attempt to solve in Oil & Gas industry, will involve machine learning techniques in order to model the system and take a relevant action based on the knowledge acquired from analysis of industrial sensor data. According to data quality dimensions described here and importance of this issue in mission-critical industrial systems, techniques for assuring some of these dimensions are applied on the data obtained from Turkish Petroleum & Refineries Inc. in this thesis.

2.1 *Data Validation and Reconciliation*

In modern chemical plants or refineries, hundreds of data streams such as flow rates, temperature, pressure, *etc.* are measured and recorded automatically for online process control and evaluation purpose. In the past 40 years, data validation and reconciliation techniques have been developed to satisfy plant models and improve accuracies. For this purpose, the process model equations such as equilibrium relation and conservation law, are used to perform data reconciliation and gross error detection. Data reconciliation is a technique developed to improve the accuracy of measurements by reducing the effect of random errors on data, which is a mathematical model [26] to reduce inconsistency between measured data and industrial process model [27]. Major difference between data reconciliation and other filtering techniques is that the former explicitly uses process model constraints by adjusting measurements to satisfy the constraints. Reconciled estimates are expected to be more accurate, and in order for data reconciliation to be accurate there should be no gross errors in measurements. GED is a companion technique to data reconciliation. This technique is developed to detect and eliminate gross error from data. In summary, data reconciliation and GED are interrelated and can be applied to improve the accuracy of measured data. For applying them on data, the plant should be modeled according to its constraints. The natural laws of mass or energy conservation are typically used for constraints in data reconciliation, which can be expressed in the following form [28]:

$$\textit{Input} - \textit{output} + \textit{generation} - \textit{consumption} - \textit{accumulation} = 0 \quad (1)$$

In this thesis, two main models are used for data reconciliation, mass balance constraints for systems in steady-state and Kalman Filter [29] for time-varying processes.

2.1.1 Steady-State System Model

The quality of process data in mission-critical industrial systems affects the performance of process monitoring and system control software. In industrial systems in which the data is collected from several resources, the measurements always contain error which affects and sometimes invalidate the process model. These errors can occur during physical measurement, data processing, and transmission of the measured signal. The difference between a measured value and a true value is represented as contribution of two types of errors: random error and gross error [8]. The magnitude and sign of random error may not be predicted, which means even if the same process is repeated with the same instruments, it may result in a different value. The reasons for these errors can be power supply fluctuation, network transmission and signal conversion noise, *etc.* This kind of errors usually have small magnitudes and can be completely eliminated. Gross errors, on the other hand, are caused by nonrandom events like: instrument malfunctioning, miscalibration or corrosion of sensors. The nonrandom nature of these errors usually identifies itself by having certain magnitude and sign in any given time. In other words, if an experiment is repeated in the identical circumstance, the systematic measurement error will be the same [8]. In Figure 3, common types of gross error are illustrated which includes:

- *Bias*: Due to uncalibrated sensors, the received values contain constant shift with respect to correct value.
- *Drift*: Due to uncalibrated sensors, the data contains an increasing or decreasing amount of error with time.
- *Precision Degradation (PD)*: The sensor plates may wear out or get dirty over time, resulting in received data from sensors containing errors resembling random noise but with higher variance.
- *Failure*: Due to sensor failure, the measured data is almost constant.

Another important issue while performing data reconciliation is model selection:

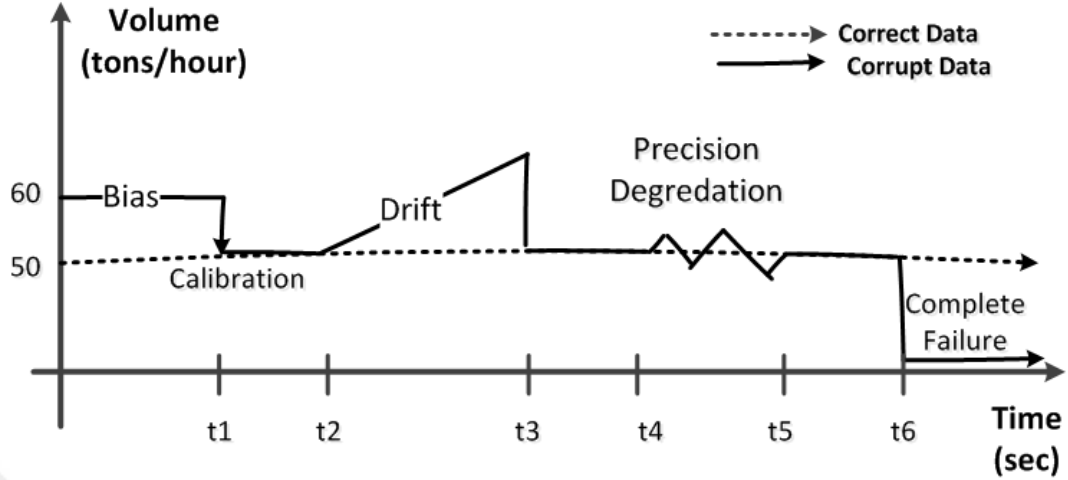


Figure 3: Common types of gross error.

steady-state or dynamic model. In theory, a system or process is in steady-state if the parameters that define the system's behavior, are not changing over time. In practice, a process is never truly at steady-state. However, a plant normally operates for hours or days in a region around steady-state. For applications that have a low frequency of change, steady-state reconciliation can be performed. In transient periods of changing states a dynamic model can be applied. The general formulation of linear models is described in equations 2-5 can be explained as follows:

$$y = x + \epsilon \quad (2)$$

where y is vector of n measurements, x is corresponding true values and ϵ is vector of unknown random errors. The constraint explained in Equation 1 can be represented in general by:

$$Ax = 0 \quad (3)$$

where A is $m \times n$ matrix and 0 is $m \times l$ vector whose elements are zero. Each row of equation 3 corresponds to related constraints such as mass balance. The elements of matrix A are either +1, -1 or 0, depending on corresponding stream flow is input, output or respectively not associated in process. The objective function can

be represented in general by equation 4:

$$\min_x (y - x)^T W (y - x) \quad (4)$$

where W is usually diagonal matrix representing weights that contain statistical properties of errors. The analytical solution to the above problem can be obtained using mathematical optimization strategy like Lagrange multiplier method as follows:

$$\hat{x} = y - W^{-1} A^T (A W^{-1} A^T)^{-1} A y \quad (5)$$

This equation denotes the solution for estimated values by \hat{x} . The estimation satisfies the corresponding constraints of the system. Data reconciliation is built on the assumption of normal distribution of random errors (white noise). Accordingly, the reconciled values may be inaccurate if measurements contain a significant process leak. To identify and remove such errors is a critical problem to be solved which is known as gross error detection process.

2.1.2 Time-Varying Kalman Filter

If we consider systems that require regular control and accurate estimates of process variables, data reconciliation has to be applied frequently as well. In this case, the whole process can no longer be assumed in steady-state. The time-varying Kalman filter is a generalization of the steady-state for time-varying systems with nonstationary noise covariance [30].

Kalman filter is a state space model used for tracking and parameter estimation formulated with the system Equation 6 where x_{n+1} is the state vector at time n that is transformed to y_n the measurement vector. A, B, C are state transition and measurement matrices and w_n and v_n are white Gaussian noise with zero mean.

$$\text{System state model : } x_{n+1} = A x_n + B u_n + w_n \quad (6)$$

$$\text{Measurement model : } y_n = C x_n + v_n$$

KF has two phases: prediction for projecting current state obtaining *a priori* estimate as shown in Equation 7, and correction step for obtaining *posteriori* estimate by incorporating actual measurement into the a priori estimate as $\hat{x}_{n|n}$, P is the estimate error covariance, Q and R are covariance matrices and, M_n is the KF gain [8]. The prediction and correction steps are executed recursively as follows:

Prediction step

$$\hat{x}_{n+1|n} = A\hat{x}_{n|n} + Bu_n \quad (7)$$

$$P_{n+1|n} = AP_{n|n}A^T + Q_n$$

Correction step

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + M_n(y_n - C\hat{x}_{n|n-1})$$

$$M_n = P_{n|n}C^T(CP_{n|n-1}C^T + R_n)^{-1} \quad (8)$$

$$P_{n|n} = (I - M_nC)P_{n|n-1}$$

Given the observed input data, the system state is tracked in a KF. The “innovation” is computed in the correction step of the KF, which is subjected to a Chi-Squared test. If the test fails, a gross error is detected. KF is used for random error detection and data reconciliation in sensor data and an adapter is required to turn its output prediction error (called *innovations*) into a GED tool, similar to the statistical global test used in IMB. Constructing a statistical test in dynamic linear systems is possible by utilizing the properties of KF innovations (*i.e.* output prediction error), which is computed in the correction step of KF. Innovations have normal distributions with expected values and a covariance matrix V_k given by Equation 9, where $P_{k|k-1}$ is *a priori* estimate covariance, C is observation model and Q_k is noise covariance matrix [8].

$$V_k = CP_{k|k-1}C^T + Q_k \quad (9)$$

Equation 10 is used to obtain a γ value, where V_k is innovation covariance and v_k is innovation residual at time k . The γ value follows a Chi-squared distribution with

1 degree of freedom and if it exceeds the criterion 95% corresponding probability for the desired confidence interval, the test fails and the gross error existence, as well as its location, are detected [8].

$$\gamma = v_k^T V_k^{-1} v_k \quad (10)$$

2.2 Time-Series Analysis

For understanding the underlying mechanism in a time-varying system, time-series analysis method is required to achieve system control. There are two main types for time-series model: time domain and frequency domain. In time domain models, data is studied using mathematical models with respect to time. Time-series variation modeling include three main classes, autoregressive, integrated and moving average models for the purpose of system control and future forecast. In general, time-series data show seasonal behavior and analysis of seasonality and seasonal forecasting is considered as an important issue. Seasonality analysis requires model selection for forecasting when the data from time-series is captured as data sequences [31]. After selecting a model, the variables and equational relationships and parameters are involved for model specification. Once the model is specified, the characteristics should be validated by comparison of forecasted values with real data (historical data) [32]. Error measures such as Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), and Root Mean Square Error (RMSE) are used for model validation. For validation, time-series forecasting methods assume there is a combination of pattern and random error in data and understanding the pattern's trend and its seasonality is its main goal. Methods like Moving Average, Linear Regression, and Exceptional Smoothing are commonly used for time-series forecasting.

2.2.1 Exponential Smoothing

Smoothing for time-series analysis is a technique for reducing random fluctuations and adjustment of seasonal components [31]. Seasonality is defined by tendency of time-series behavior that repeats itself every L periods. Seasonality trends types contain Additive, Multiplicative, Linear and Exponential growth in time-series. Exponential Smoothing though, is a procedure for revising a forecast in light of recent experience and assigns exponentially decreasing weights as the observations gets older, a simple exponential smoothing is described in the Equation 11 as below:

$$S_t = \alpha X_t + (1 - \alpha)S_{t-1} \quad (11)$$

Applying this recursively to each successive observations, each new smoothed forecast is computed as weighted average of current observation and previous smoothed observations. The initial value of S_t plays an important role in computing all the subsequent values. Setting it to y_1 is one method of initialization. Another possibility would be to average the first four or five observations. The smaller the value of α , the more important is the selection of the initial value of S_t [33].

For example, Holt-Winters is a Triple Exponential Smoothing technique used for data that shows trend and seasonality. In addition, overall smoothing, trend b_t , and seasonality S_t has to be updated as described below:

Overall Smoothing (level)

$$R_t = \alpha * y_t / S_{t-L} + (1 - \alpha)(R_{t-1} + G_{t-1}) \quad (12)$$

Smoothing Trend Factor

$$G_t = \beta * (R_t - R_{t-1}) + (1 - \beta)G_{t-1} \quad (13)$$

Smoothing Seasonal Index

$$S_t = \gamma * y_t/R_t + (1 - \gamma)S_{t-L} \quad (14)$$

The value of forecast for next period is given by:

$$F_t = R_{t-1} + G_{t-1} + S_t \quad (15)$$

where $0 < \alpha, \beta, \gamma < 1$ are smoothing constants, and values of these parameters are initialized according to length of data [34], [33].

2.2.2 Auto Regressive Moving Average (ARMA)

ARMA is a time domain model used for time-series analysis and includes autoregressive (AR) and moving average (MA) methods for modeling variations of a process [32]. This model can capture the dynamic nature or “memory effect” of the systems with respect to sudden changes in the input. Mathematically AR model is defined as:

$$X_t = c + \sum_{i=0}^p \phi_i X_{t-i} + \epsilon_t \quad (16)$$

where c is constant ϵ_t is white noise and ϕ_i are model’s parameters, and MA can be expressed as:

$$X_t = \mu + \epsilon_t + \sum_{i=0}^q \theta_i \epsilon_{t-i} \quad (17)$$

where μ is expectation of X_t , θ_i parameters of the model, and ϵ_t white noise error term. Combining AR and MA models, the ARMA can be described by the recursive equation:

$$y_k + \alpha_1 y_{k-1} + \dots + \alpha_n y_{k-n} = \beta_0 x_k + \dots + \beta_m x_{k-m} \quad (18)$$

where y_k are the output, and x_k are the input of the system at time k . Error terms in this model are assumed to be independent and randomly distributed with zero mean. This model can be used for future forecasting of time-series as well [31].

Exponential smoothing and ARMA models are the two most widely-used approaches to time-series forecasting, and provide complementary approaches to the

problem. While exponential smoothing models are based on a description of trend and seasonality in the data, ARMA models aim to describe the autocorrelations in the data. Autocorrelation is a measure to find correlation between values of time-series in different points. This measure can be used for discovering patterns or identification of missing frequency in a signal. Mathematically autocorrelation, when mean and variance of time-series are time-independent, can be defined as:

$$R(\tau) = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2} \quad (19)$$

where time-lag is $\tau = t - s$ at any time t and s .

2.3 Modeling System Using Data Mining Techniques

Data mining is a process of learning, improving and extracting useful information using algorithms to learn automatically by machines, from data repositories and past experiences [35]. Applications using large datasets for learning methods are called data mining [36]. This process helps us explain the observed data, detect certain patterns and make predictions, build models and detect anomalies. Machine learning techniques, build mathematical model of processes using the theory of statistics and consist of two steps: training a model and use the learned model as algorithmic solution [35]. There are two approaches of supervised and unsupervised learning, where in supervised learning the output of training dataset is provided but in unsupervised learning, the output is not available and the problem is how to cluster data into different classes. Machine learning is being used in a wide variety of applications in computer science and currently is attracting attention from organizations for stock market analysis, optimization problems, control, customer behavior for data analysis in large scale. Industrial applications are also deploying services for complicated processing to solve their problems efficiently and accurately using machine learning approaches.

The problem of data quality and anomaly detection we attempt to solve in Oil &

Gas industry will involve these techniques for classification of detected errors in order to take relevant action on data and sensors. Supervised learning algorithms such as Rule-Based, Decision Tree, Neural Networks are used in this work as a descriptive task [36] for deriving patterns of data points received from sensor stream-data and techniques such as K -means and DBSCAN clustering for operational mode detection of industrial systems over fast streams of data.

2.3.1 Regression Analysis Method

Regression is a widely used technique that is applicable to time-varying systems. Sensor measurements in industrial plants may be interdependent and fluctuations on one measurement may have an affect on other measurements (*i.e.* input and output of a system). This information can be extracted from the estimated parameters.

Linear regression model (RM) is a statistical learning method used for analysis of relationships between system variables [37]. It is a low degree polynomial regression model that usually does a satisfactory job [38]. Observed sensor data collected via IIoT from physical devices are fed into stream mining process for cyber model extraction. In general, a statistical relation of observed data is explained by $y_i = b_0 + b_1x_i$, where b_0, b_1 are linear model parameters, variable y_i is the observed response, variable x_i is the predictor for experimental unit i [39].

The linear model is explained by Equation 20 where \hat{y}_i is the predicted response. It is clear that the predicted values will contain “prediction error” or “residual error”.

$$\hat{y}_i = b_0 + b_1x_i \quad (20)$$

2.3.2 Classification Techniques

In machine learning classification is a supervised learning approach that learns from past data to classify new observations. Data-driven techniques such as Complex Decision Trees (CDT), Neural Networks (NN) and K -Nearest Neighbors (KNN) classifiers are used in this thesis. CDT is a regression tree for modeling relationship

between variables [40] and the dataset breaks down to smaller subsets and constructs a tree with decision and leaf nodes. Artificial NN is used for pattern recognition and classification of data which consists of neurons in several layers and applies nonlinear functions to pass outputs to the next layer. KNN is a density-based clustering method that classifies data based on top- k nearest neighbor by labeling a new data point according to the label of most of its neighbors [41].

2.3.3 Clustering Techniques

Clustering methods can be used for modeling a system's behavior. Data points collected from different processes can be analyzed for knowledge extraction. In this thesis sensor data is analyzed for system's operating states. Data points would be grouped in one cluster denoting one operating state and formation of new clusters is interpreted as drift to new conditions or emerging patterns. As such, a locality-based clustering approach, without specifying cluster numbers in advance is beneficial in modeling a system's behavior. Density-based clustering methods are suitable for this evaluation and can be compared to other clustering techniques such as K -means.

DBSCAN algorithm can be used for discovering clusters in arbitrary shapes based on density of data points. It requires two input parameters ($Eps, minPts$) where Eps -neighborhood for a point p is all neighbors within range Eps defined in Equation 21 that has more than $minPts$ data points [42].

$$N_{Eps} = \{q \in D \mid dist(p, q) \leq Eps\} \quad (21)$$

2.4 Related Work

Data validation and reconciliation (DVR) techniques were developed to improve data quality and satisfy plant models within the last few decades. The process model equations such as mass equilibrium and conservation laws were used to perform DRGED. Data reconciliation is a mathematical model [26] that reduces inconsistency

between measured data and physical model by reducing the effect of random errors on data. Reconciled estimates are expected to be more accurate and without (or at least smoothed) outliers. To accomplish accuracy improvement of data, outlier detection methods were developed and applied together as a companion technique to DR. Tang et al. [43] studied outlier detection and anomaly detection [44], [45] as data-driven approaches developed for identification of unexpected patterns in data and can be categorized into four types: distribution-based, distance-based, clustering-based and density-based [46] that are applicable to data directly. But, DR-GED techniques use the underlying physical system model to satisfy constraints in addition to outlier detection. Approaches like statistical outlier detection are applicable where data follows a certain distribution. For model construction of a system fed with stream data, time-series analysis such as autoregression, moving average and exponential smoothing are required to fit model, monitor and understand underlying forces of process model [47], [48] in order to make future predictions and replace possible missing values [49]. Time-series models are applicable in a wide variety of sectors such as health care [50], transportation [47], forecasting [51], and stock market. Error detection and classification such as ours provide a data quality improvement for better system modeling and isolation of faulty processes.

DR-GED applications have been used in chemical or petrochemical processes [52], [53] since their analysis quality has been known to directly improve the process performance and increase profits as well as safety [54]. Most prior studies focused on performing DR-GED offline using static samples of data collected from relatively old system logs. Over the past decades, the number of data resources such as sensors used in industrial facilities and Internet of Things (IoT) has risen dramatically. Yet, online data processing remains a challenge. Attempting to store these data first to analyze them later creates additional IT costs, unwanted delays to actionable information, and mishandling of threats or opportunities. Fortunately, there are now tools to

process data *on-the-fly* as they move from DCS and SCADA systems to selected destinations. Neither relational databases nor distributed batch processing systems [55] alone are designed to cope with industrial data analytics. In sectors such as finance and mobile telecommunications, enterprises started employing CEP engines [56] for tracking Key Performance Indicators (KPI) in real-time or for carrying out rule-based alarm management. Nowadays, heavy industries also want to complete their “digital transformation” or Industry 4.0 journey [57] by extending their data architectures with real-time complex analytical [58], [59]) capabilities. However, data needs to be validated first in real-time, for the rest of the online analytical models to work accurately.

Do Valle, et al. [60] collected benchmarks for DR-GED issues introduced in the literature for mass and energy balance preservation. Zhang, et al. [61] studied DR and parameter estimation (DR-PE) for systems with multi-operating conditions and suggested a PCA-based steady-state detection technique extended with clustering to partition the data into different modes of operation, so that accurate reconciliation can be applied for each mode. They also addressed the DR-GED problem for dynamic systems using particle filters [27]. Guo, et al. [62] proposed a systematic approach for DR of a thermal system using mass balance and improved data accuracy. Ruan, et al. [63] used a symbolic representation of time-series data for reducing the volume while also extracting patterns.

Rafiee et al. [53] studied DR-GED using material and energy conservation laws in natural gas processing. Jiang, et al. [64] studied GED for data obtained from a coal-fired power plant. They modeled the system at steady-state and reconciled the data using the mass and energy balance equations. They employed statistical global test to detect gross errors and serial elimination technique to identify the error sources. Their steady-state technique compares to the basic Instantaneous Mass Balance (IMB) method described in this thesis. We find that IMB and similar “stateless” techniques

are less effective compared to modified Kalman filters that track system behavior when the system is not at steady-state. They also did not discuss applying GED on top of streaming data.

The approach used in this thesis integrates the prior information from the system in steady-state to model plants and uses statistical tests for validation of data in real-time relates to system identification work in the literature. For example, Alenany, et al. [65], applied a subspace identification approach using constrained least squares with prior information of the system, which leads to a computationally efficient approach. Larimore [66] also applied maximum likelihood methods using subspace system identification to obtain the most appropriate statistical model of a system. Buchholz, et al. [67] compared noiseless and noisy data models and the effect of noise on a system model. Robust parameter estimations of systems in a linear subspace model were applied to reflect confidence in an *apriori* model by McWhorter, et al. [68]. For all these cases, the quality of models depends on the quality of data. The proposed real-time GED and GEC techniques are therefore complementary to all and aim to fill this gap in the literature.

In data stream analysis evolution of states should be taken into consideration during adaptive modeling. When the measurements collected from IIoT devices consist of multi-operating conditions the model's parameters change state and identifying these states is useful for understanding normal behavior, anomaly detection, and data reconciliation. Researchers have employed different techniques and algorithms in the past to address similar problems [69]. Zhang, et al. [70] proposed a quality-directed adaptive analysis framework AQUA, for incremental model tracking. Lughofer, et al. [71] proposed an incremental rule splitting concept to autonomously deal with gradual drifts for local distributions. Zhu, et al. [72] proposed a "multi-scenario" parameter estimation for dynamic systems. Zheng, et al. [61] proposed parameter estimation

with multi-operating conditions for data reconciliation in steady-state. For handling drifts in data streams Shaker, et al. [73] proposed an adaptive forgetting factor depending on the current intensity of drift in stream data. In another approach Lughofer, et al. [2] proposed a dynamic clustering method based on evolving vector quantization for cluster extraction of online data. A related work for building Oil & Gas ontologies [74] and addressing data integration issues includes POSC Caesar Association (PCA)'s ISO 15926 standardization efforts. Algorithmic designs also need to change to support incremental updates over streams. Within the last decade, special "stream mining" versions [75] of rule-mining, pattern recognition, classification and clustering algorithms have been developed and embedded into emerging data architectures. The aim is to identify the time where the state is shifting from one steady-state to another and a model reconstruction is triggered. The context shift of the system is detected according to the error fluctuations of the trained DBSCAN clusters. This study involves using and comparing stream-based regression and clustering methods over oil and gas sensor data for differentiating operational models of industrial systems and is complementary to these related works.

The proposed approaches combine steady-state modeling with real-time model updates, operational mode identification and data cleaning via error detection all at once.

For using data processing tools next to industrial requirement in data management there are several architectures to implement Cyber-Physical System (CPS) of industrial plants. CPS is used to develop required environment by: (1) a methodology to model and analyze CPS's behavior; and (2) CPS architecture for monitoring the system based on cloud technologies for sensor network data analysis [76].

The cloud platform provides the opportunity for collecting real-time information coming from real-world resources to improve decision making processes and pattern extraction from data. IoT (Internet of things) can generate this kind of real-time,

large amounts of varied data when there are millions of things feeding data to cloud computing [77]. Knowledge discovery and decision making from such data are challenging tasks and an emerging trend known as Big Data. This paradigm combines large scale computation with data intensive techniques and mathematical modeling for analyzing data. Big data computation needs huge storage, repositories for applications that have high volume in exabytes and associated applications are encountered with challenges in software development [78]. This is a clear example of Big Data that demands to take cloud computing into account [15]. This large volume of data is collected from IoT are fast with high velocity and consequently require frequent decision making using machine learning and Big Data framework to divide problems to subproblems and solve in parallel [79]. Miller, et al. [80] developed a framework called SCALATION for sensor data analytics using traffic data. Jian, et al. [81] analyzed Spark's framework performance, running machine learning instance (linear regression) on a local cluster deployment. Richter, et al.[82] compared tools used for machine learning in big data frameworks with regard to scalability and speed of algorithms which can be used as a guide to select an applicable tool for a problem. Similarly, features of distributed stream computing architectures are introduced and discussed by Cia, et al. [83] that is used for component selection of proposed Big Data architecture in this thesis. Also, another architecture is proposed to perform data analysis in real-time by Mauro [84] for network traffic data.

In [85] a Big Data architecture is proposed in which DQ is pervasive throughout the platform. Their architecture consists of a markup language SDQ-ML for describing sensor services developed for applications for declaration of sensor requirements. To create the ability to query over sensor data, Wang et al. [86] used tags as data source creation time, this gives user ability to filter out data with undesirable specification and data quality for their requirements. Testing data quality in Big Data is another aspect in DQ assurance and tools like Zoho [87] is proposed for this purpose

in the cloud. Technologies for real-time analytics like Apache Storm [88], Spark [89] Streaming and Complex Event Processing(CEP) are getting attention for Big Data applications [90]. In addition to Big Data solutions in these technologies, the focus is also moved to online stream processing. Guanitz et al. [91] used Apache Storm with CEP in applications for dynamic and event-driven data processing in real-time. Santos et al. [92] proposed two new technologies: JStreams a CEP engine for local analytics and DiAIM for distributed analytics taking benefit of cloud computing technologies. Ophidia is also a research effort for mining scientific data using MySQL array-based data analysis [93].

The rest of the thesis is organized as follows. Chapter 3 describes the cyber-physical systems and proposed data architecture for oil refineries and compares different GED methods including IMB, KF, and DREDGE and experimental results on multivariate sensor data, also compares the accuracy and computational performance of approaches, and explains different GEC methods including CDT, NN, and KNN for gross error classification and discusses the accuracies. Chapter 4 describes the identification of operational modes of industrial systems using sub-model analysis and details the proposed algorithm for adaptive window size tuning and related results. Chapter 5 describes the proposed data architecture for industrial plant in private and public clouds. Finally, Chapter 6 concludes this thesis.

CHAPTER III

MULTIVARIATE SENSOR DATA ANALYSIS FOR OIL REFINERIES

For knowledge extraction from physical data resources a systematical transformation technology is defined for interconnecting system's physical components to cyber computational space called Cyber-Physical System (CPS) [94]. CPS are described by Rajkumar, et al. [95] as “physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core”. Industry 4.0 will be realized by connecting CPS with Cloud via Internet of Things (IoT) and providing distributed, secure, and intelligent analytical data services at the Edge or the Cloud [96], [97].

3.1 Description of CPS for Oil Refineries

Oil & Gas businesses have already implanted thousands of sensors inside and around their physical systems. Sensor data continuously streams in via DCS and SCADA systems measuring temperature, pressure, flow rate, *etc.* of drills, turbines, boilers, pumps, compressors, and injectors. Figure 4 illustrates the the CPS and software architecture of our oil refinery. The major “physical” components are the power systems depicted in Figure 5 and the crude oil processing columns depicted in Figure 6; their “cyber” counterparts are composed of the sensors, servers, and the data-based services that store and process the digital models. The power plant provides electricity to the rest of the refinery and has about 80 megawatts of generation capacity. There are 8 boilers with maximum flow capacities of 100 *ton/hour*, which turn hot water into super-heated and highly-pressurized vapor. The vapor output from every boiler

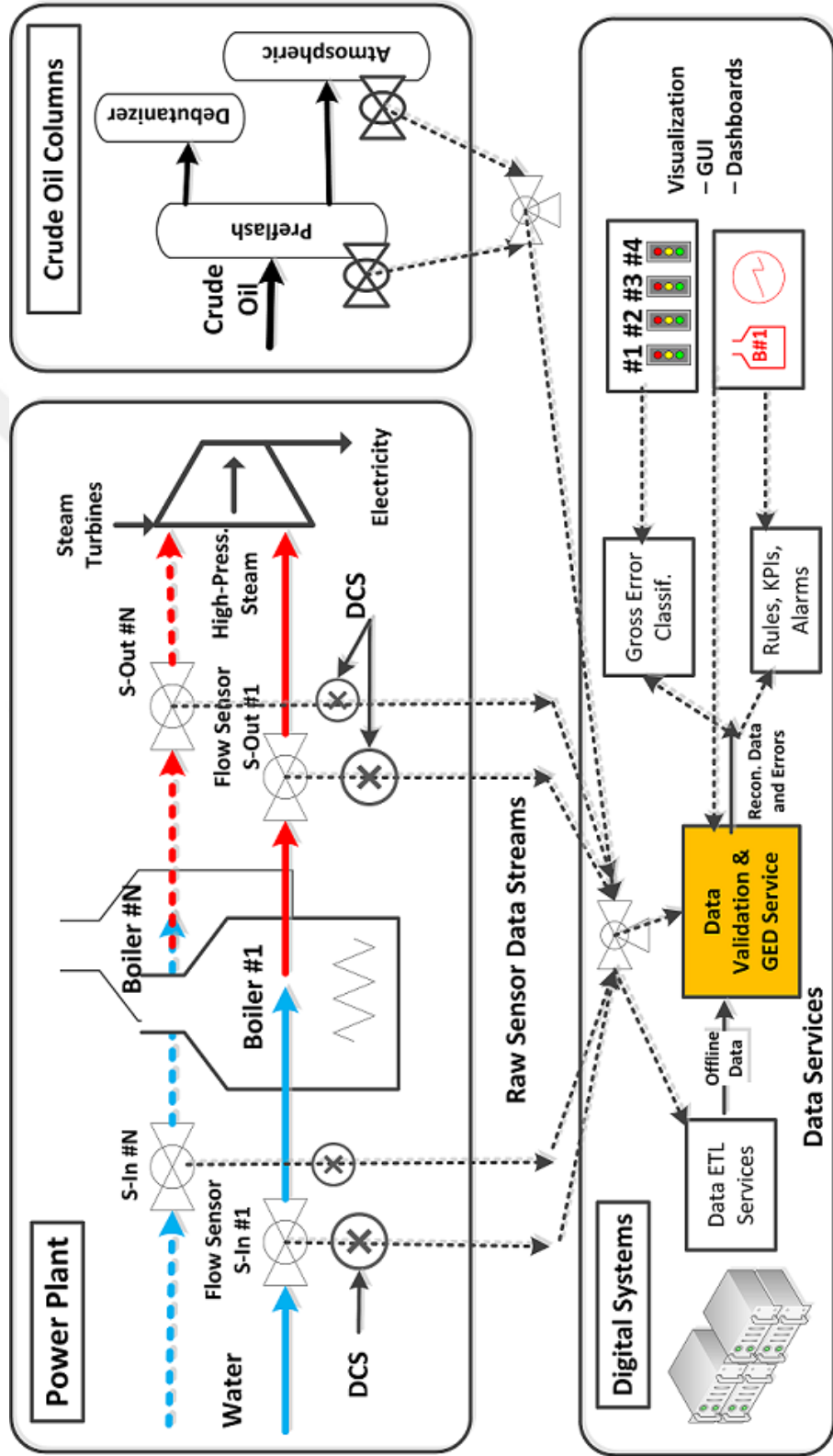


Figure 4: Illustration of the cyber-physical systems inside the oil refinery; high-level operation of the power plant, petrochemical plant for crude oil processing, and data stream processing services.

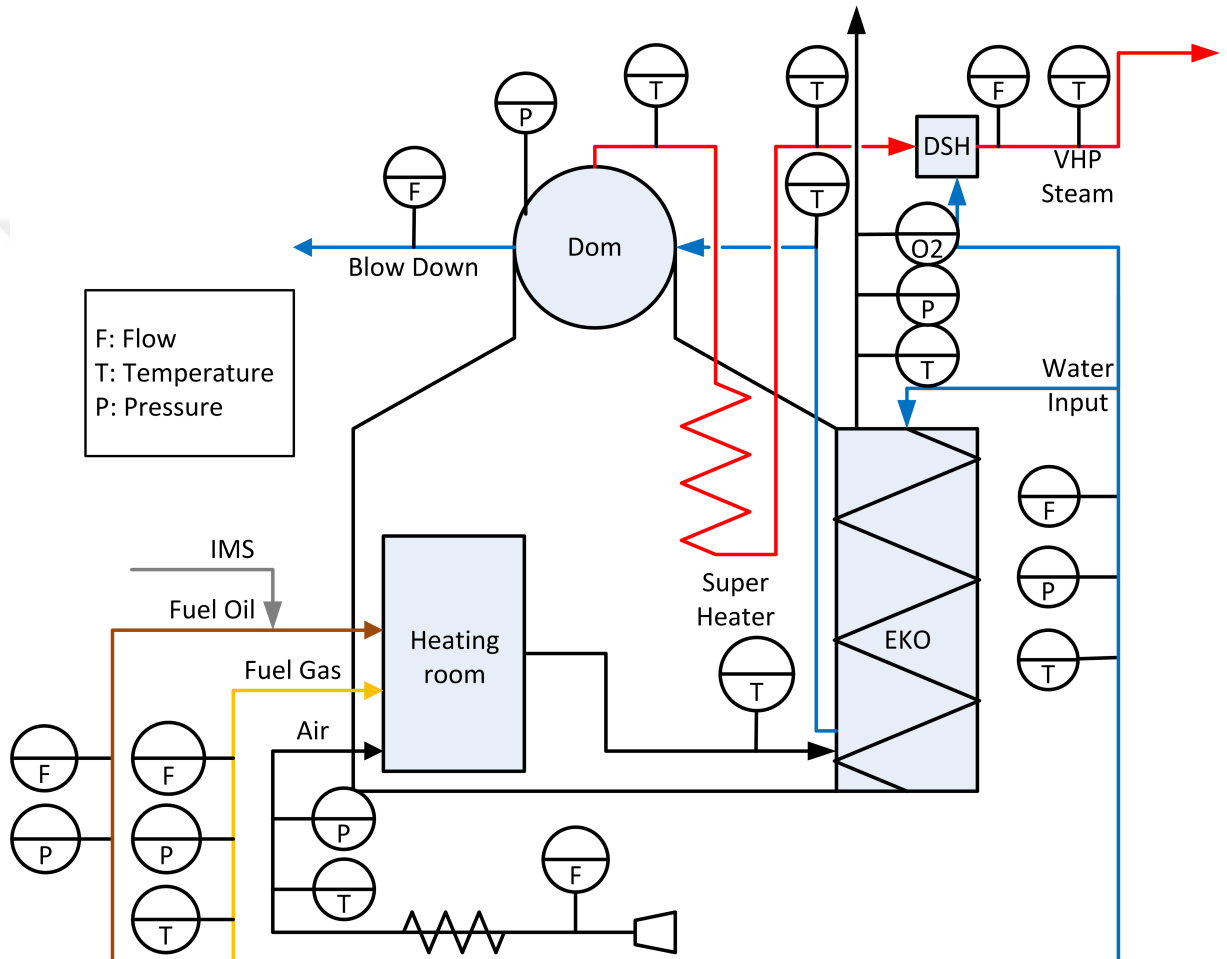


Figure 5: Illustration of a boiler power plant. Various types of sensors are implanted for real-time data collection. The boiler can change states among the desired stream pressure levels (low, high, very-high) or go automatically into heating, cooling, recycling, and condensing modes. KBS: cold water input; VHP: Very-high power steam output; DSH: De-Super Heater.

is directly fed into a corresponding steam turbine with an alternator, which turns the thermo-kinetic energy into electrical energy. The flow rates of the inputs (hot water) and the outputs (vapor) of the boilers are measured by flow rate sensors (S-In1, S-Out1, *etc.*) and these measurements are fed into the models. Placing redundant (*i.e.* extra) sensors around the physical systems increases reliability and allows replacement of missing values that helps to locate and classify errors or detect sensor failures.

Crude oil columns take the oil as input and deliver several by-products such as liquid propane gas, fuel oil, kerosene, diesel, and asphalt. A preflash unit reduces the pressure and provides the first vaporization, where the vapor goes to a debutanizer for distillation and the liquid mix goes to an atmospheric column for separation. On the digital side, we have servers and software for online and offline processing of received data. For offline data processing, the Hadoop Distributed framework [55] is used for providing the Extract, Transform and Load (ETL) services. This service gathers raw data from all sensor streams and presents them in a unified format. Using offline data, we can extract the steady-state models for physical systems and use this prior information to instantiate cyber models. Using the online data, we can tune the system models dynamically and detect gross errors in real-time. We deployed DR-GED models and GEC algorithms inside a CEP engine.

The benefits of using CEP engines for data stream analytics are at least threefold:

- They can turn raw data into actionable information quickly, thus helping oil refineries catch critical issues to avoid losses or detect alarming situations and operational inefficiencies in real-time.
- They can eliminate unwanted data early in the data pipeline, saving further CPU, memory, storage and energy costs.
- They can catch transient or emerging patterns, which never show up in an offline data mining analyses.

The client applications with a graphical user interface (GUI) can connect to data

services to receive reconciled data and alert about the existence of gross errors in sensors, as well as their types after classification.

Considering all input–output lines and the different types of measurements (water and vapor flow rates, temperature, pressure, fuel oil and fuel gas flow rates), there are about 1,000 sensors in the power plant and 60,000 sensors in the entire oil refinery, where a sample of one month real data measured every minute is made available for academic use at OpenML datasets web site [98]. Our goal is to process all of this raw, streaming sensor data and create valuable data services for generating clean and reconciled data, detecting and classifying gross errors (*i.e.* avoid false positives - FP), and raising alerts when the system is malfunctioning (*i.e.* true positives - TP). The system can also be used to track a set of KPI for the entire plant and report results in dashboards if the performances are below or above pre-defined thresholds.

3.2 Methodology in DREDGE

DR-GED requires a mathematical model to reduce inconsistency between measured data and industrial process model [27]. In this study, we use two main models for data reconciliation: mass balance constraints for systems in steady–state and Kalman filters [99] for time–varying processes. We also propose DREDGE, which is a special Kalman Filter implementation extended with unity-gain that incorporates system dynamics into mass balance constraints. After modeling system we apply a distribution–based outlier detection approach on the estimated system model as GED method. All methods are implemented into a CEP engine.

On the other hand, industrial systems can have multiple operational modes and there can be shifts among them during daily operation. Accordingly, the models are required to be reconstructed and time–varying parameters updated, which explains the emergence of local and window–based stream online analysis to be more reliable than offline (or batch data) analysis.

3.2.1 Steady-State and Instantaneous Mass Balance (IMB)

The first method used for GED is IMB. IMB enforces a mass balance constraint at every sensors' measurement instant and does not take into account the system dynamics or the memory effect. This is a standard DR-GED method, based on comparison of observed data to their statistics [8] as shown in Equation 22.

$$\hat{y}_t = y_t - VA^T(AVA^T)^{-1}A\hat{y}_t \quad (22)$$

If we assume y_t matrix contains the masses of input water and De-Super Heater (DSH), and the mass of output vapor, then ideally mass balance constraint should enforce $Ay_t = 0$, where matrix A represents the system input flow +1, output flow -1, and unused measurements 0. For our power plant system, y_t is a 3-dimensional vector of these masses and a sample is shown in Table 1. IMB method also uses a covariance matrix V (Eq.1), in order to find out how attributes vary together. We obtained covariance matrices by using historic data, where V is an $N \times N$ matrix ($N = 3$ for power plant and $N = 17$ for petrochemical plant) containing the variance-covariance of all the input-output attributes:

$$A = \begin{bmatrix} +1 & +1 & -1 \end{bmatrix}$$

$$V = \begin{bmatrix} 104.998 & 1.825727 & 128.1891 \\ 1.825727 & 0.236576 & 2.163642 \\ 128.1891 & 2.163642 & 163.9264 \end{bmatrix}$$

Next, the data is reconciled using Equation 22 and \hat{x} denotes the reconciled (or de-noised) data. Then, under the null hypothesis H_0 (H_0 : The system is working, so there is no gross error), a statistical test analogous to global test is applied for detecting gross errors known as Chi-Squared test (X^2), which is a nonparametric statistical test corresponding to cumulative probability, exceeding the 95% criterion

Table 1: Sample data for y_t , input/output flow values (*ton/hour*) of one of the boilers. The complete data spans 5 months worth of measurements from 12/2014 to 4/2015.

Water	DSH	Vapor
54.15951	0.737258	52.311
53.48025	0.74118	51.476
54.11722	0.740656	53.231
51.46956	0.742319	53.128
54.07272	0.738685	51.057

Table 2: Sample flow measurements for 17 sensors y_t (*ton/hour*) of the petrochemical system of Figure 6. The complete data spans 2 months worth of measurements from 08/2014 to 10/2014.

1	2	3	4	...	17
14136.54	34.91064	1530.896	12549.15	...	910.1204
14139.98	35.59276	1537.567	12557.17	...	909.0833
14122.35	35.73056	1533.281	12553.83	...	906.283
14113.85	36.53142	1529.905	12545.45	...	902.0441
14126.97	35.47277	1527.698	12555.25	...	909.6918

gross error is detected. Given r the vector of residuals of linear model that follows a normal distribution with zero mean and the covariance matrix V , the statistical global test is constructed. Statistics given by γ in Equation 23 follows a X^2 -distribution with v degree of freedom, where v is the rank of matrix A . The 1×3 Matrix A used in Equation 22 in IMB method has rank 1 and the Chi-Squared test for this process has 1 degree of freedom and 95% confidence ($X_{95\%}^2(m) < 3.84$) corresponding to cumulative probability for desired confidence interval. A result of data reconciliation and gross error detection using IMB method is shown in Table 3.

$$\phi = AVA^T, R = Ay_t, \gamma = R^T \phi^T R \quad (23)$$

For the petrochemical plant, the same approach is used for DR-GED on raw values. As depicted in Figure 6, this plant has 17 flow sensors over 3 main branches of material flows and the corresponding sensor data streams. Each branch has its own mass balance consideration as follows: $1=2+3+4$, $3=5+6+7$, $4=8+\dots+17$. Table 2

Table 3: A result of data reconciliation using IMB method.

Water	Vapor	Rec. Water	Rec. Vapor	γ
54.582	59.898	57.307	57.66	7.230
55.951	58.477	57.246	57.414	1.632
55.316	58.320	56.856	57.055	2.308
54.004	59.372	56.756	57.112	7.372
55.239	58.644	56.985	57.210	2.966
55.158	58.366	56.803	57.015	2.633

shows sample sensor measurements for Figure 6 and matrix A represents the mass balance equation ($V_{17 \times 17}$ not shown for brevity). The least squares estimates (LSE) method is used to obtain \hat{y}_t and a Chi-squared test with $rank(A) = 3$ degree of freedom is used for GED in this plant.

$$A = \begin{bmatrix} 1 & -1 & -1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots & -1 & -1 & -1 \end{bmatrix}$$

3.2.2 Capturing System Memory Using ARMA Model

Assuming plants are linear dynamical systems, their operation can be explained with the AutoRegressive Moving Average (ARMA) model [32]. This model can capture the “memory effect” of the systems with respect to sudden changes in the input. For example, in the power plant any sudden change in the amount of water input might not reflect itself at the vapor output instantaneously, as some water/vapor gets stored in the system during heating. The ARMA model is described by the recursive Equation 24:

$$y_k + \alpha_1 y_{k-1} + \dots + \alpha_n y_{k-n} = \beta_0 x_k + \dots + \beta_m x_{k-m} \quad (24)$$

where y_k are the output, and x_k are the input of the system at time k . For the power plant, y_k is the vapor output and x_k is the water input, both in *ton/hour*. For the petrochemical processes, y_k is the various by-products (fuel, diesel, kerosene) and

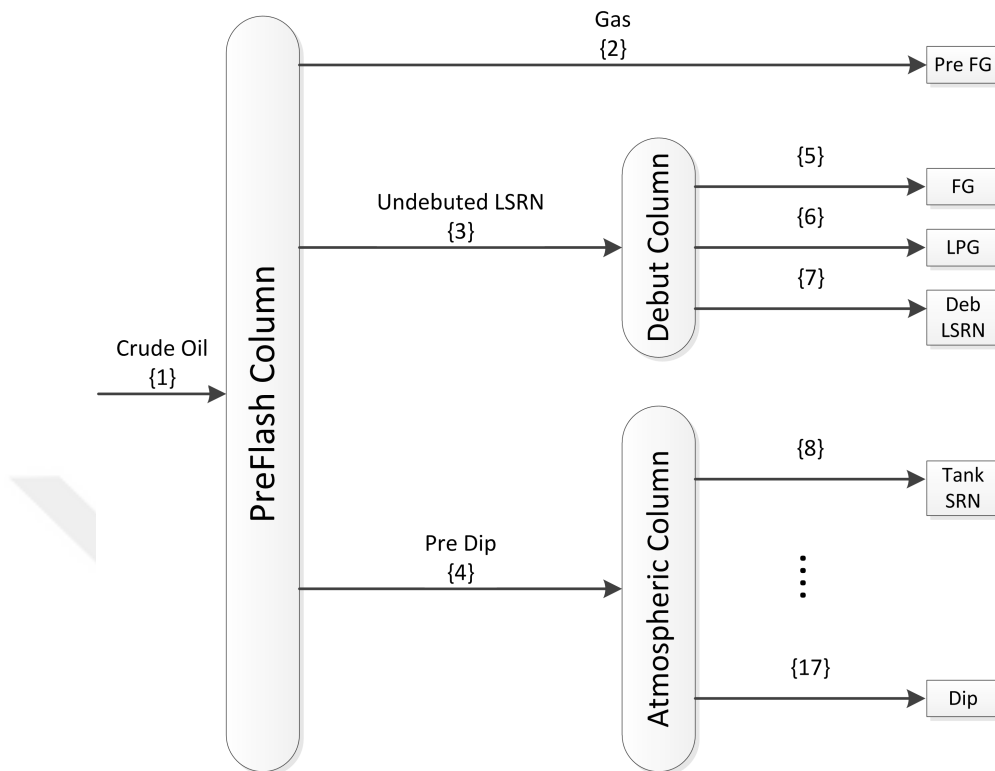


Figure 6: Petrochemical process showing crude oil input, preflash, debutanizer, atmospheric columns, and sensors (1-17) measuring input and output flows.

x_k is the crude oil input. Enforcing instantaneous mass balance (*i.e.* IMB) at the input and output might not work if the system is not in steady-state. ARMA model captures this system buffering, thus “memory effect”. The order of the model in Equation 24 is given by m and n parameters. The higher the order is, the bigger the memory of the system. In this system, discrete time is chosen, since the sensor values are obtained in discrete time intervals. A set of training data is chosen for fitting ARMA model and extracting the coefficients α_i and β_j to identify system order.

Then for a given system order and corresponding coefficients, the estimated model is trained and applied on test data. The system order that performs best on training data is selected, which resulted in $m = n = 1$ with both of our systems (power plant and petrochemical plant). We also compared ARMA with Holt-Winters time-series modeling technique [34], which is a special case of ARMA model used for time-series

smoothing (that integrates level, trend, seasonality of data). Since Holt-Winters does not take into account input/output autocorrelations, we observed the ARMA model to be more effective for our scenario.

3.2.3 Time-Varying Kalman Filter (KF)

A system's stability is preserved when the coefficients are all in a region of convergence, so, the step response and impulse response of the system are more reliable. These explanations show that the system is simulated reasonably and using Kalman Filter to remove random error from data and detect gross error is possible for this linear dynamic system. The ARMA model obtained for both systems (power plant and crude oil) is converted to state-space representation. Then given the observed input data, the system state is tracked using KF as explained in Equations 7, 8. The "innovation" (*i.e.* delta) is computed in the correction step of the KF, which is subjected to a Chi-Squared test similar to IMB. If the test fails, gross error is detected. This method incorporates system dynamics, however, mass balance is not watched for. To incorporate mass balance as well, we used the unity-gain for KF and will be referred as DREDGE¹ (Data REconciliation and Gross Error Detection) method.

KF is designed for noise detection and removal. The input, output, reconciled input and output and KF states are implemented and tested on a fraction of 2,000 of Water-input and Steam-output data. Constructing a statistical test to be applied in steady-state systems for GED in dynamic linear systems, is possible utilizing the properties of innovations.

3.2.4 DREDGE: Novel Approach

IMB method is ideally used with systems at steady-state. The time-varying Kalman Filter (KF) is a generalization of the steady-state models for time-varying systems, *i.e.* systems with nonstationary noise covariance. Although the ARMA model takes

¹The word "Dredge" means: to dig and remove rocks and earth continuously from a stream.

into account the dynamics of the boilers, it does not take into account the fact that, all the water that goes in must eventually come out of the system. This means the ARMA model for this process requires to have a D.C. gain of 1 which is a smoothing technique in moving-average model for reduction of random fluctuations in time-series [31]. Given the plants' state obtained by ARMA model, power plant and petrochemical plant are converted into a state-space representation [30]. In our approach initial KF method tracks systems' dynamic state. To incorporate system dynamics into mass balance constraints, we also added the unity-gain constraint to the KF method and called it DREDGE [100]. Unity-gain constraint in ARMA model is defined with D.C. gain of 1 allowing imbalance instantaneously, but enforcing mass balance to be preserved at the long run [31]. Equation 25 explains this additional linear constraint with respect to Equation 24.

$$-\alpha_1 - \alpha_2 - \dots - \alpha_n + \beta_0 + \beta_1 + \dots + \beta_m = 1 \quad (25)$$

Thus, we modified the above method to use a constrained least-squares step in estimating the system coefficients, to enforce mass balance.

To improve the quality of identification, it is intuitive to incorporate prior information about the system like stability or D.C. gain and system structure. In our system, we need to consider the unity-gain to achieve mass balance of input and output. For this purpose, the system needed to be modeled with ARMA model with prior information so the problem is to solve an LSE problem subject to unity-gain. The problems formulation is as follows:

$$\begin{aligned} & \text{Minimizing} \quad ||A\theta - b||^2 \\ & \text{Subject to} \quad C^T\theta = d \end{aligned} \quad (26)$$

where unity-gain is considered in $C^T\theta$ which is equal to one, and the C matrix is a vector of ones. Vector b is real output system and θ is to be estimated. The solution

to this problem is solving the Lagrangian relaxation with optimality condition [101].

$$L(\theta, \lambda) = 1/2\|A\theta - b\|^2 + \lambda^T(C^T\theta - d)$$

$$\theta = (A^T A)^{-1}(A^T b - C^T \lambda) \quad (27)$$

$$\text{where } \lambda = (C(A^T A)^{-1}C^T)^{-1}(C(A^T A)^{-1}A^T b - d)$$

Solving this problem, θ which is the systems characteristics is estimated such that error is minimized considering unity-gain. After estimating θ according to the above explanation, the dynamic system is converted to state-space model and combined with KF for GED.

Since DREDGE combines the best of both worlds, we expect it to give the most accurate results in GED. Then given the observed input data, the system state is tracked in a Kalman Filter. The “innovation” is computed in the correction step of the Kalman Filter, which is subjected to a Chi-Squared test. If the test fails, a gross error is detected. KF is used for random error detection for data reconciliation in sensor data is required to turn its output prediction error into a GED tool, similar to the statistical global test used in IMB. Constructing a statistical test in dynamic linear systems is possible by utilizing the properties of KF innovations (*i.e.* output prediction error), which is computed in the correction step of KF. Innovations have normal distributions with expected values and a covariance matrix V_k given by Equation 28, where $P_{k|k-1}$ is *a priori* estimate covariance, H_k is observation model and Q_k is noise covariance matrix [8].

$$V_k = H_k P_{k|k-1} H_k^T + Q_k \quad (28)$$

Equation 29 is used to obtain a γ value, where V_k is innovation covariance and v_k is innovation residual at time k . The γ value follows a Chi-squared distribution with 1 degree of freedom and if it exceeds the criterion 95% corresponding probability for the desired confidence interval, the test fails and the gross error existence, as well as its location, are detected [8].

$$\gamma = v_k^T V_k^{-1} v_k \quad (29)$$

Combining a Chi-squared test and unity-gain to KF, we obtained a GED method that considers mass balance in a time-varying process model.

3.2.5 Gross Error Classification (GEC): Novel Approach

The Second component of the proposed software service is classification of detected errors called GEC. In the previous chapter, we addressed models for GED in detail and in this section take the vision a step further by formally classifying oil refinery sensor errors into four types using data-driven techniques CDT, NN, and KNN classifiers for pattern recognition and classification of data. These machine learning algorithms were trained with the sudden changes of mean and variance properties of measured data in a window-based analysis according to the definition of each gross error type.

Different types of gross errors have their own natural data corruption behavior or characteristics. In relation to the physical defects of the sensors and their operational conditions, common types of gross errors are as follows:

- Bias: Due to uncalibrated sensors, the received values contain a constant shift with respect to the correct value.
- Drift: Due to uncalibrated sensors, the data contains increasing or decreasing amounts of error with time.
- Precision Degradation (PD): The sensor plates may wear out or get dirty over time, resulting in received data from sensors containing errors resembling random noise around the nominal values.
- Failure: Due to sensor failure or measurement boundaries, the received data is constant or completely random.

For comparison of gross error detection and classification accuracies of IMB, KF

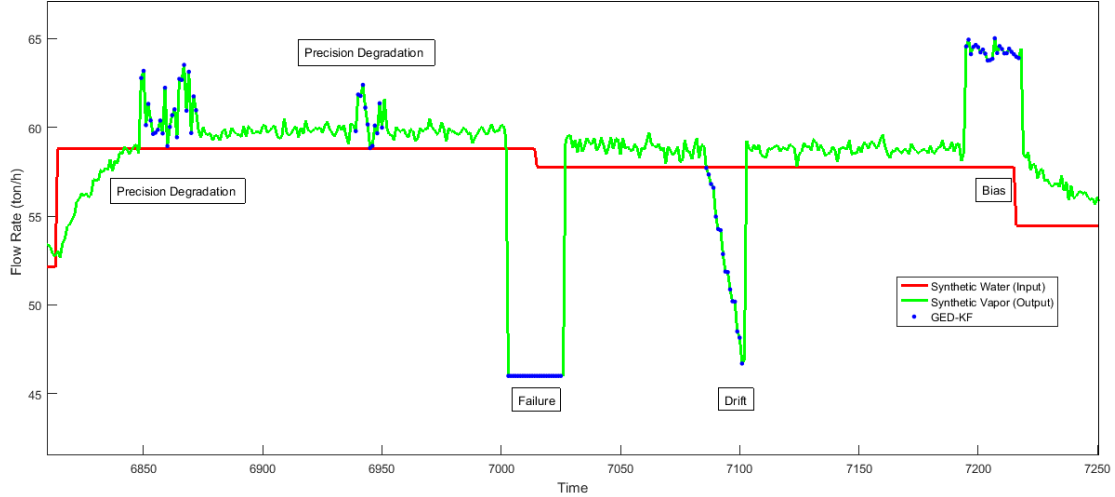


Figure 7: Different types of gross errors are inserted on top of refinery power plant real data. Blue dots show that gross errors can correctly detected using KF. These error are then classified by mean-variance tracking.

and DREDGE methods, we generated a synthetic dataset with 1 million data samples whose properties were extracted from real data obtained from the power plant. Next, error events representative of different gross error types (Bias, Drift, PD, and Failure) were added to the synthetic data as exemplified in Figure 7. After detecting the gross error, for each error type, the F-measure values of classifiers including CDT, NN, and KNN are calculated and compared. F-measure is a harmonic mean of precision ($TP/(TP+FP)$) and recall ($TP/(TP+FN)$) as described in Equation 30 which provides a unified score for the classification quality evaluation [36].

$$F - measure = \frac{2 * precision * recall}{precision + recall} \quad (30)$$

3.2.6 Rule-Based Algorithm for Labeling Gross Error Types

For applying a supervised learning algorithm to learn gross error behavior, data should be labeled. But, the real data obtained from our oil refinery did not have any extra information about types of gross errors. With statistical study done after GED, and evaluation of mean and variance for data points that detected as gross error point in

synthetic data, rule-based learning algorithm is used for data labeling purpose. The real data from oil refinery is unlabeled data and for error type distinction after GED, we developed a rule-based algorithm based on $3 - \sigma$ rule for normal distribution. Pseudo code in Algorithm 1 shows this rule-based learning algorithm.

Algorithm 1 Gross Error Classifier

```

1: procedure
2:  $3\sigma = MaxDev$  :  $\sigma$  is StdDev of gross errors extracted from historic data
3: maintain a sliding window
4: for each line do
5:    $M = Mean(previousWindow)$ 
6:    $Value = parseLine(line)$ 
7:   insert new value to current sliding window
8:    $\Delta M = |M - value|$ 
9:    $S = StdDev(window)$ 
10:   $case = "Normal"$ 
11:  if  $GED(value) == true$  then
12:    if  $(1.5\sigma < \Delta M < 3\sigma$  and  $1.5\sigma < S < 3\sigma)$  then  $case = "Bias"$ 
13:    if  $(\Delta M < 3\sigma$  and  $S < \sigma)$  then  $case = "Drift"$ 
14:    if  $(\Delta M < 1.5\sigma$  and  $\sigma < S < 3\sigma)$  then  $case = "PrecisionDegradation"$ 
15:    if  $(\Delta M > 3\sigma$  or  $S > 3\sigma)$  then  $case = "Failure"$ 
16:    else  $"UnClassified"$ 
17:  end if
18:   $print(line + case)$ 
19: end for

```

The algorithm is trained with mean (μ) and the variance (σ^2) values of offline data and obtained the model illustrated in Figure 8. This model is used for labeling data and by measuring the latest mean (μ) and the variance (σ^2) values of the time-series data in the current sliding window and classify gross errors accordingly. Given one method of GED that detects a gross error event, if the mean changes up to 3σ ($\Delta M < 3\sigma$) and variance does not change drastically ($S < \sigma$), then the gross error is classified as Drift. If both the mean and variance change significantly ($1.5\sigma < \Delta M & S < 3\sigma$), then it is classified as a Bias. If the mean does not change significantly ($\Delta M < 1.5\sigma$), but the variance increases ($\sigma < S < 3\sigma$), then it is classified this as Precision Degradation. Finally, if both variance and the mean changes significantly,

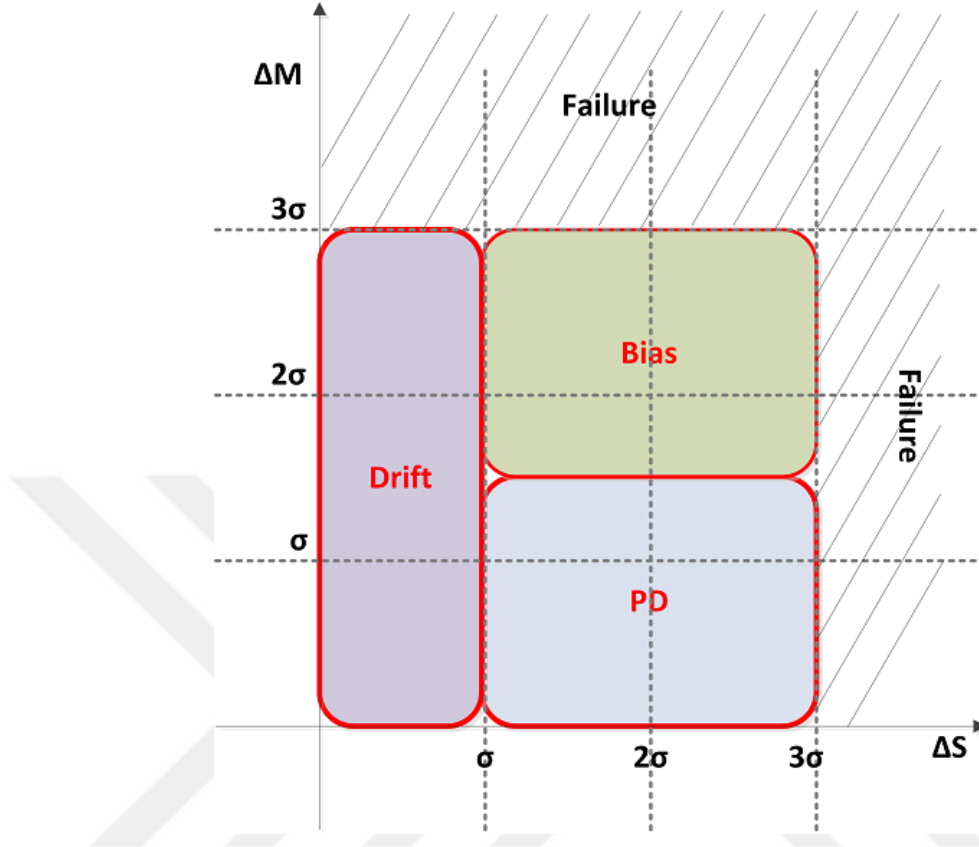


Figure 8: Categorization of gross error types with respect to changes in mean and variance of sensor values.

it is classified as a Failure–Random. In case there is no movement at all, the sensor is classified as Failure–Dead. The classification categories illustrated in Figure 8 are separated linearly without any overlap, but in real data, gross error types might have overlaps or some detected data points may contain more than one type of error. By training the classifiers using these rules, the unclassified or misclassified data points can be detected and their true classes can be extracted. These points are also revealed by the accuracy evaluation of supervised learning algorithms and F-measure values.

3.3 *Experimental Results*

In this section, the GED accuracy results of IMB, KF, and DREDGE methods for different gross error types are discussed over synthetic data. Next, data reconciliation

Table 4: GED results of IMB, KF, and DREDGE for Bias-type gross error over synthetic data.

	Gross Error Existance	GE Prediction	
		T	F
DREDGE	T	2755	0
	F	0	2756
KF	T	2741	14
	F	0	2742
IMB	T	2795	0
	F	40	2796

and error classification are applied over the real refinery dataset. Finally, the performance of different algorithms over different streaming window sizes are tested and compared to better understand the scalability and sustainability of these DR-GED techniques over fast moving sensor data.

Java Eclipse is used to implement DREDGE using ESPER engine and using linear algebra library for Java to support matrix operations. IMB, KF, DREDGE were first implemented and tested in Matlab. In all scenarios, GED methods were implemented in Java to be utilized in real-time GED and GEC. The first code block in Figure 9 shows the configuration of the ESPER engine for reading the data stream of the boiler sensor data in the correct format. The raw data stream is then started and is fed into the ARMA model, IMB and, KF by calling the related class, wrapped inside the KFLListener classes for ARMA and DREDGE and IMBListener for IMB method.

3.3.1 Results for Synthetic Data

Table 4 shows the GED accuracy results of IMB, KF, and DREDGE methods in detecting Bias type gross error over the synthetic dataset, to which $N = 2,755$ Bias *events/epochs* were synthetically inserted at random points. Logically, there will be $(N + 1)$ 2,756 periods where there is no gross error. Each Bias event contains a set of Biased values, the count of which randomly varies between 5 and 25 to provide statistical significance. A “True (T)” event signifies the existence and a “False (F)”

```

Configuration config = new Configuration();
    config.addEventType("Boiler",Boiler.class);
    EPServiceProvider epService =
EPServiceProviderManager.getProvider("Boiler",config);
String expression = "select FC062, FI066, FI061 from
Boiler.win:length_batch(100)";
    EPStatement statement =
epService.getEPAdministrator().createEPL(expression);
        //UpdateListener ;
        KFLListener listener = new KFLListener();
        statement.addListener(listener);
AdapterInputSource source = new AdapterInputSource("KBS-
rawm.csv");
    CSVInputAdapterSpec csvInputAdapterSpec = new
CSVInputAdapterSpec(source,"Boiler");
    CSVInputAdapter csvInputAdapter = new
CSVInputAdapter(epService, csvInputAdapterSpec);
    csvInputAdapter.start();

```

Figure 9: Snippet of the CEP + DVR development.

event signifies the non-existence of gross error in that period.

As seen in Table 4, our DREDGE method gives the most accurate prediction results for Bias type gross events, where the 2,755 synthetically inserted events were all correctly detected (true positive-TP) and no misdetections (false positive-FP or false negative-FN) were recorded. However, the KF algorithm misses 14 of the Bias events. The IMB method is the least accurate in this case, where 40 “non-gross error” events were detected as gross error events. Since the DREDGE method takes into account both system dynamics and mass balance constraint, it performs the best. Note that a FP increases the total number of both gross error and non-gross error events, whereas a FN decreases both since we have time-series event data. The experiments are repeated for all 4 types of errors (Failure, Bias, Drift, Precision

Table 5: GED results for IMB, KF, and DREDGE over different gross error types on synthetic data.

	Failure		Bias		Drift		PD	
	Precis	Recall	Precis	Recall	Precis	Recall	Precis	Recall
DREDGE	100%	100%	100%	100%	100%	99.14 %	99.8%	90.8%
KF	100%	100 %	100%	99.49 %	100%	99.37%	99.54%	78.05%
IMB	91.40%	100%	98.57%	100%	95.71%	100%	81.36%	100%

Degradation) and summarized results in Table 5.

Table 5 shows the precision and recall rates in confusion matrix for the GED methods over all gross error types. Precision of both KF and DREDGE are very high (99.54-100%), since they make little or no FP. Since IMB does not track the dynamics of the system, it has lower precision ratios (81.3-98.57%) due to FPs. One interesting phenomenon is the relatively low recall rates for KF (78.05%) and DREDGE (90.8%) during detection of PD type errors. This happens because they both suffer from over-fitting their models and include precision degradation errors as regular, non-gross error events. In summary, we learned IMB can cause a lot of false alarms and DREDGE and KF methods are more dependable in their predictions compared to IMB. Therefore, KF-based methods are preferable for GED over time-varying systems found in refineries.

Applying GEC technique on top of GED methods, Table 6 and Table 7 show the F-measure, precision, and recall of all classifiers per error type. The F-measure values are between $0 \leq F \leq 1$ and the higher F-measure shows that the classification has a higher predictive power. In Table 6, we generally observe that the F-measure values of classifiers over synthetic data are higher than the real data. This can be attributed to the higher number and separation of gross errors inserted in the synthetic data, whereas in the real data the errors may be overlapped.

Over synthetic data CDT has the highest F-measure values $0.994 \leq F \leq 1.0$ and the lowest values are achieved by KNN $0.965 \leq F \leq 0.995$. This shows that

Table 6: F-measure for GEC results of CDT, NN, and KNN over different gross error types on Real and Synthetic (Synt) data.

	Failure		Bias		Drift		PD	
	Real	Synth	Real	Synth	Real	Synth	Real	Synth
CDT	0.958	0.994	1.0	0.996	0.992	1.0	1.0	0.998
NN	0.869	0.915	0.867	0.964	0.989	0.974	0.953	0.973
KNN	0.851	0.967	0.836	0.976	0.982	0.995	0.984	0.965

Table 7: GEC results for CDT, NN, and KNN over different gross error types on real data.

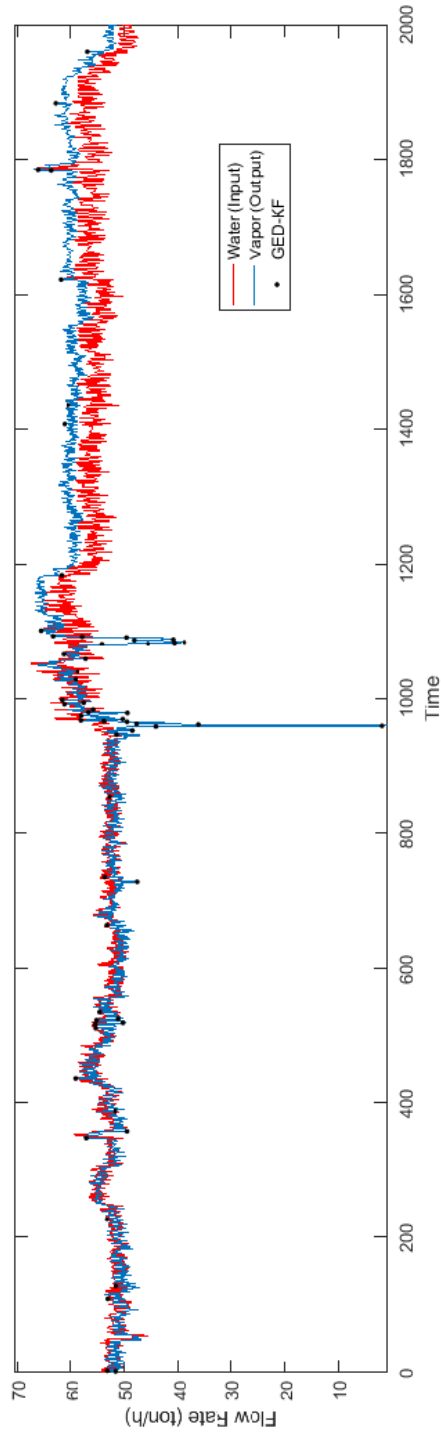
	Failure		Bias		Drift		PD	
	Precis	Recall	Precis	Recall	Precis	Recall	Precis	Recall
CDT	95.8%	95.8%	100%	100%	99.2%	99.2%	100%	100%
NN	90.9%	83.3 %	81.3%	92.9 %	99.3%	98.6%	96.8%	93.8%
KNN	86.9%	83.3 %	85.1%	82.1 %	99.2%	97.22%	100%	96.8%

the classifiers can learn the labeled data and they have high predictive power. Next, these classification algorithms are validated over real data.

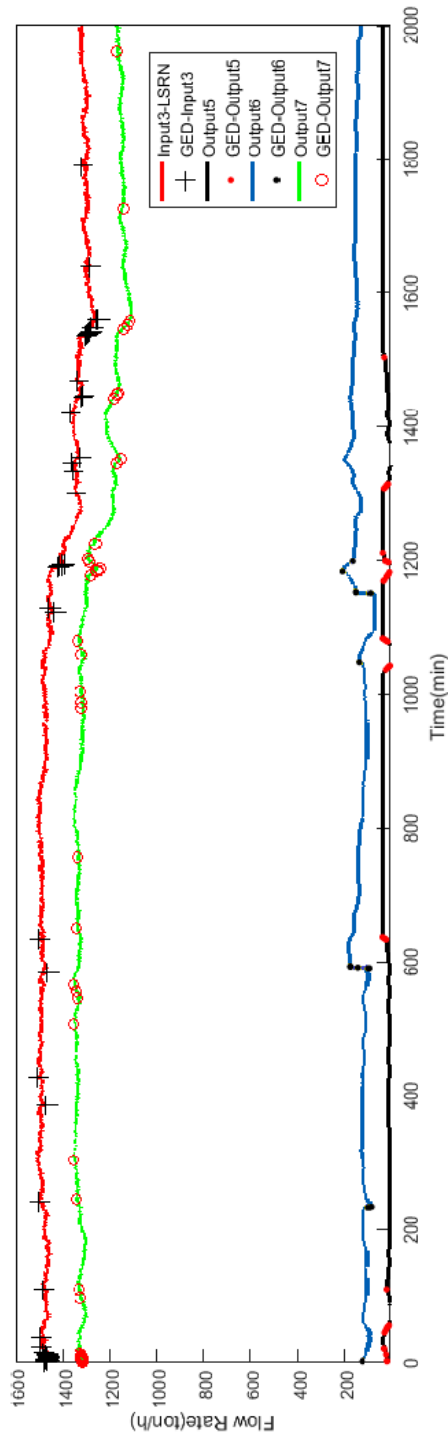
3.3.2 Results for Real Refinery Data

After evaluating the accuracy of GED and GEC methods on synthetic data, these methods were validated on real data. IMB, KF, and DREDGE models were trained using a common dataset from a single day (8/1/2013) and tested again on another common dataset from the following day (8/2/2013). The visualization in Figure 10-(a) shows the gross errors detected by KF for the flow rate measures (in *ton/hour*) on the two main lines (water/vapor) of the power plant (time = 2000 *mins*). Note that, the transient jumps in data are marked (*) by KF algorithm and there are not gross error marks in the steady-state regions.

In Figure 10-(b), we see the petrochemical material flow measurements and the gross errors detected therein. We first observe that GED can be applied on different lines simultaneously to identify and potentially locate which lines have which type of gross errors. We see some locality among the gross errors on different lines, but also



(a)



(b)

Figure 10: (a) GED for the water/vapor lines of the Power Plant data, (b) GED for debutanizer lines 3=5+6+7 of the Petrochemical process data.

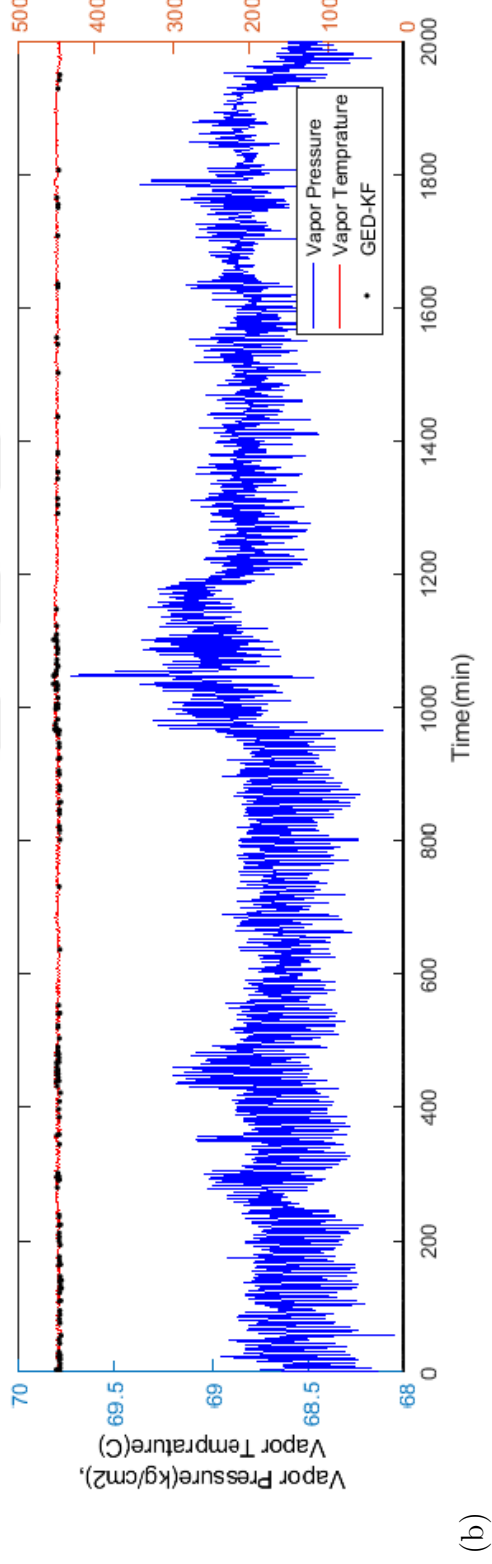
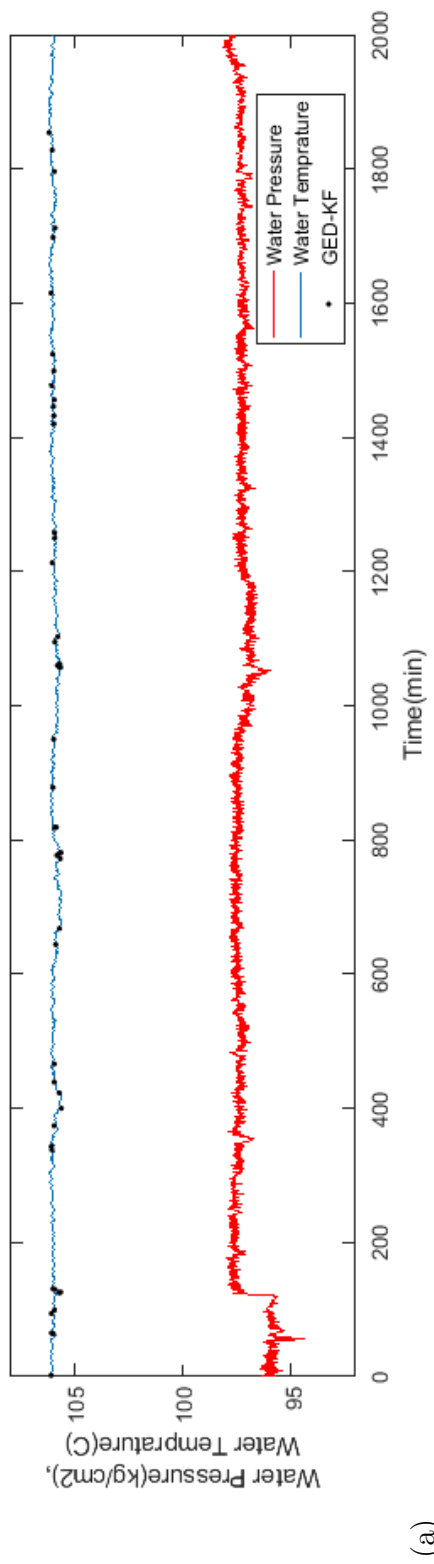


Figure 11: (a) Water temperature/pressure lines of power plant used for GED, (b) Vapor temperature/pressure lines of power plant used for GED, by DREDGE.

Table 8: GEC results of IMB, KF, and DREDGE over real dataset.

	Gross Error Existance	GE Prediction	
		T	F
DREDGE	F	9754	0
	T	5	241
KF	F	9781	0
	T	4	215
IMB	F	9477	0
	T	1	522

some independence. This proves that we can detect, locate, differentiate, classify, and fix gross errors on different sensors for different industrial processes. For sensor #3, where $3=5+6+7$, we used reconciled values obtained from line 1, $1=2+3+4$ (refer to Figure 6 for petrochemical plant’s details).

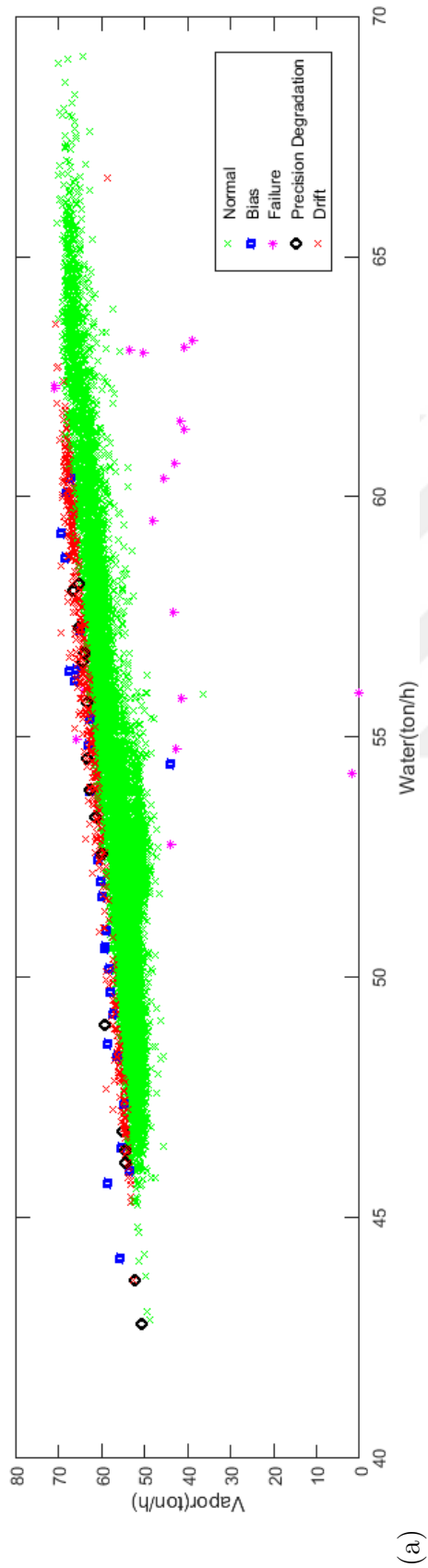
Our GED methods can also be applied on sensor measurements of different types such as temperature/pressure, temperature/flow, *etc.* Figure 11-(a) and Figure 11-(b) respectively show that, beyond the use of flow rates, water temperature/pressure and vapor temperature/pressure measurements can be utilized for GED purposes. Using multivariate data is beneficial if some gross errors are not detectable in one set of data, but can be extracted from among different sensor measurements.

Next, the GEC technique is applied on top of GED (IMB, KF, DREDGE) methods over real data from the power plant and report results in Table 8. IMB declared 522 measurements as gross errors, which was significantly more than KF (215) and DREDGE (241). We know from Table 5 that IMB has a low Precision (81.36%) for detecting PD types, therefore the higher GED numbers can be attributed to higher FPs for PD. In Figure 12-(a) we see that the water/vapor flow rates cluster around 45-70 *ton/hour* creating a concentrated green region, which is referred to as the normal range; GED algorithms detect and mark gross errors on top of this cluster. IMB acts like a multi-linear classifier by declaring errors detected above the normal range as

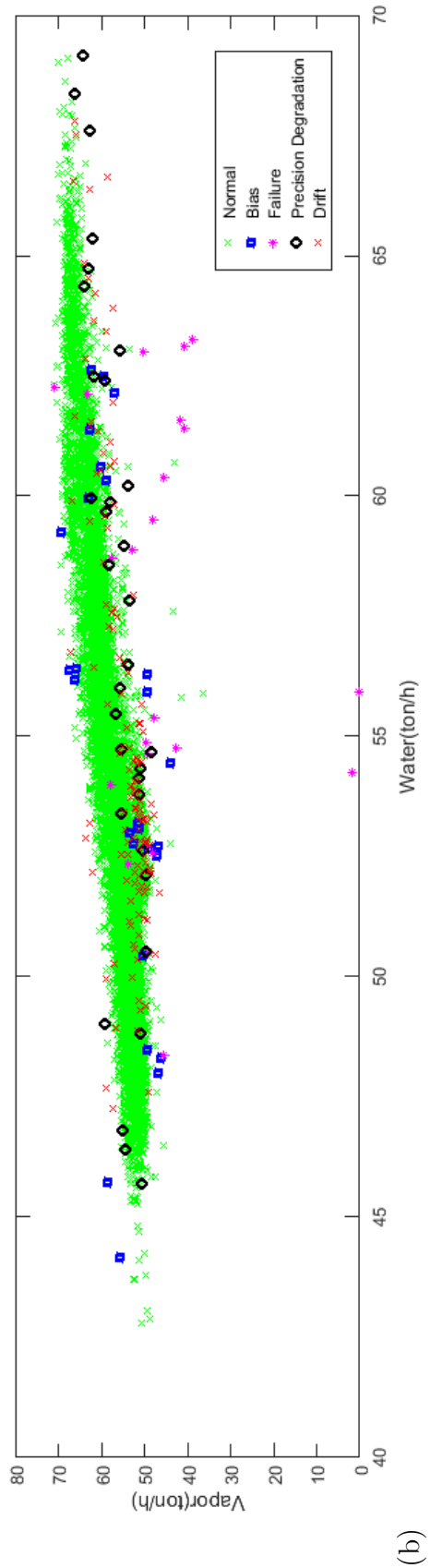
Drift, Bias, and PD and errors below this range as Failure types. As seen in Figure 12-(b), KF model also detects the same Failure types, as detected by IMB in the lower region, but does not agree in most of the Drift, Bias, PD types (FPs) declared by IMB in the upper region; IMB and KF only agree on a few of the Bias types on this upper region. However, Drift and PD errors detected by KF and DREDGE models are more dispersed among the normal values. This is because KF can track Drifts and PDs that are a part of the dynamic system behavior even inside normal ranges.

The F-measure obtained from CDT model over real data is $0.958 \leq F \leq 1.0$ for each type of gross error. NN's F-measure for gross error classification is $0.867 \leq F \leq 0.989$. The F-measure achieved by KNN are $0.836 \leq F \leq 0.984$ as shown in Table 6. The lowest F-measure values belong to Failure and Bias which is related to the overlaps between these two types due to their similar behavior. Table 7 also compares the GEC results, but reports the precision and recall details. Lower precision values can be attributed to FPs or "misdetections", whereas lower recall values can be attributed to FN or "missed detections". While it is desirable to have higher values in both precision and recall, having low recall values may have worse outcomes for oil refineries. As seen in Table 7, KNN may achieve a recall of 83.3% for Failure and 82.1% for Bias type errors. This means that the operators won't be informed 16-17% of the time when those sensor errors are happening, which is not acceptable. In comparison, the highest F-measure, precision and recall values are obtained by CDT. Also, the model's performance is an important concern in stream data processing. As a result, we trained and applied the CDT classifier on CEP engine because of its high accuracy and relatively lower time complexity (*i.e.* $O(\log n)$).

Figure 12 shows classification of errors detected by IMB vs. KF-based GED methods. We observe in both Figures 12(a)-(b) that water/vapor flow rates cluster around 45-70 *ton/hour* creating a concentrated green region, which we will refer to as the normal range; GED methods detect and mark gross errors on top of this cluster.



(a)



(b)

Figure 12: (a) GEC for errors detected by IMB, (b) GEC for errors detected by KF.

IMB works as a sharp, multi-linear classifier for error detection, where values above and below the normal ranges are declared as errors. CDT classifies those above the normal range as Drift, Bias, PD and errors below normal range as Failure. As seen in Figure 12-(b), KF-based models also detect the same Failure types, as detected by IMB in the lower region of cluster, but do not agree in most of the Drift, Bias, PD types (FPs) declared by IMB in the upper region; They agree on a few of the Bias types on this upper region. However, Drift and PD errors detected by KF-based models are more dispersed among the normal values. This is because KF and DREDGE can track Drifts and PDs that are a part of the dynamic system behavior even inside normal ranges.

The validated sensor data that is de-noised through DR-GED process can be used in modeling and other analysis process of the refinery and results of GEC is used for identification of sensor malfunction or system anomaly.

CDT was modeled and evaluated by 5-fold cross-validation technique and Gini's diversity index for the split criterion. NN was trained and tested using scaled conjugate gradient backpropagation, which was a two-layer feed-forward network, with sigmoid hidden and softmax output neurons. Similar to CDT, KNN model was trained and evaluated by 5-fold cross-validation technique, and the Euclidean distance metric for 1 nearest neighbor. Over synthetic data CDT has the highest F-measure values $0.994 \leq F \leq 1.0$ and the lowest values are achieved by KNN $0.965 \leq F \leq 0.995$. This shows that the classifiers can learn the labeled data and they have high predictive power. These classification algorithms were then validated over real data obtained from the oil refinery.

3.3.3 Computational Performance Evaluation

We learned that with careful selection of system analysis model (DREDGE, KF, or IMB), we can accurately detect and simultaneously classify gross errors. However,

the question is whether these methods are sustainable over fast data streams. This section aims to provide an answer by detailed observation of performances for each method. The performance evaluation is done using an open-source CEP engine for Event Stream Processing, called ESPER. The system specifications are ESPER v.4.6.0 installed on RedHat Enterprise Linux Server 5.4 64-Bit OS, Java 1.7.0-05 and JRE v7 on an IBM HS22 Blade Server with Intel Xeon E5530 CPUs (8 cores, 2.40 GHz) and 24 GB DRAM. Data used in these experiments are 200,000 records from (Water, DSH, Vapor) flow sensors sampled every 60 seconds for about 5 months in the TÜPRAŞ power plant. This data was replicated 5 times to form 1 million lines to better represent the real sensor loads. Every line has records of 3 flow sensors in power plant dataset and 17 flow sensors in the petrochemical dataset. Therefore, we have 3 million sensor “events” in the first and 17 million “events” in the second dataset. We publicly provided a 1-month sample of this real data at OpenML datasets site [98] for academic use. Performances of four representative continuous queries (a, b, c, d described below) were evaluated at different window sizes [100, 250, 500, 1000] using tumbling windows. Tumbling window is a sliding window, where the slide-size is equal to the window size. Continuous queries are:

- (a) Select(*) from Boiler: This query returns all sensor data for Boilers Water, DSH, and Vapor sensors. It is implemented as a reference query with the lowest computational and reference I/O loads.
- (b) GEC: We measured the impact of our GEC method that primarily uses statistical (stddev and average) library functions inside ESPER.
- (c) - (d) GED (IMB, KF-based, and DREDGE) methods using the JBLAS linear algebra library for matrix computations.

We start with performance (memory and CPU usage) analysis of queries using the water/vapor dataset from the refinery’s power plant. Each experiment is repeated

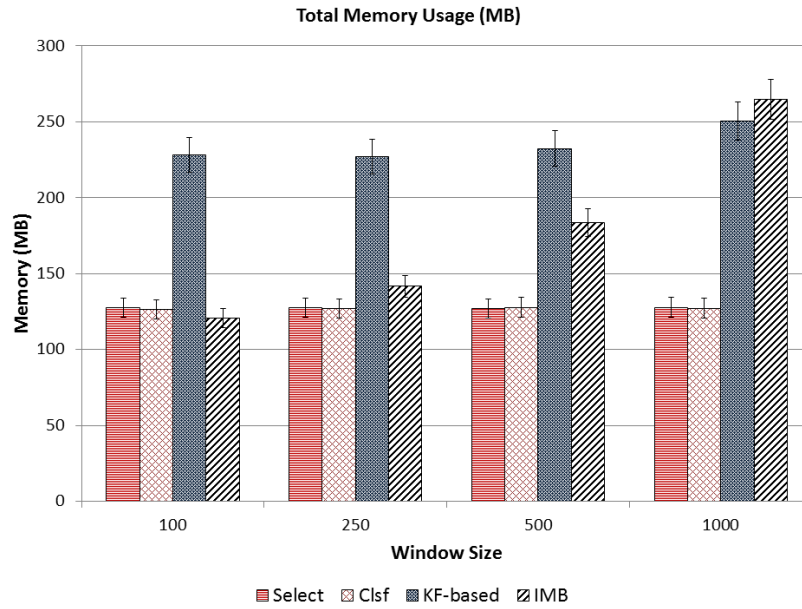


Figure 13: Power Plant: computational loads of reference queries and GED algorithms with respect to total memory usage (MB) for different window sizes over streaming data.

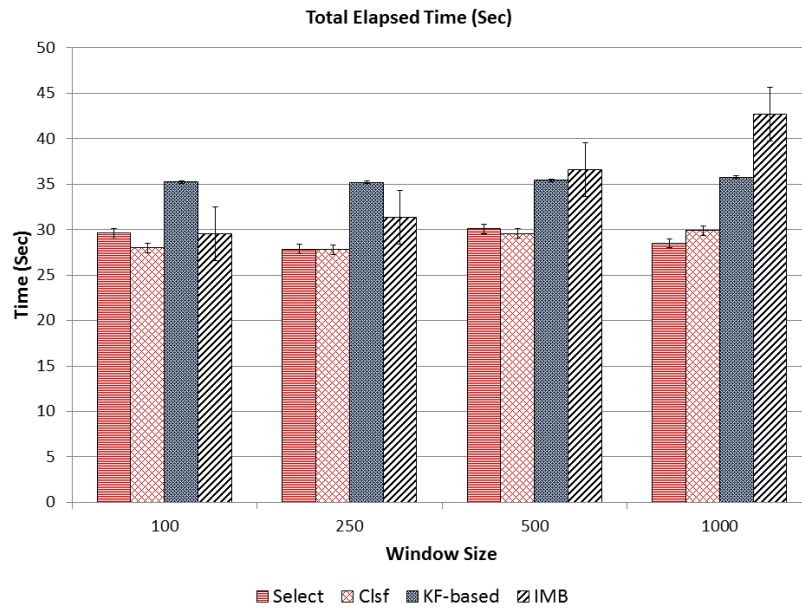


Figure 14: Power Plant: total processing time (sec) for different window sizes over streaming data.

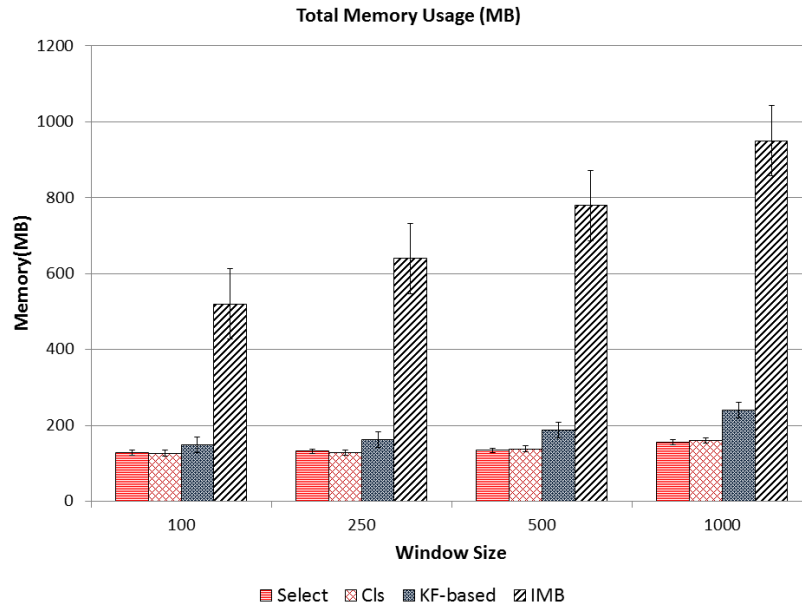


Figure 15: Petrochemical Plant: computational loads of reference queries and GED algorithms with respect to total memory usage (MB) for different window sizes over streaming data.

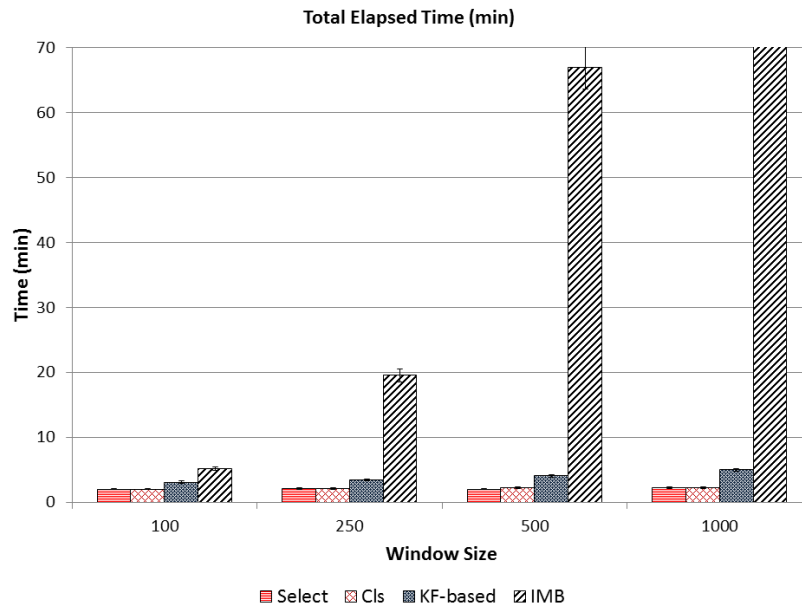


Figure 16: Petrochemical Plant: total processing time (*min*) for different window sizes over streaming data.

5 times for each window size and the average values (as well as min and max) are depicted in Figure 13-14. Figure 13 shows the total memory used by different window sizes. For the Select(*) and GEC queries the memory usage is almost constant *w.r.t* the growth of window length as the data is quickly moved in and out of the window. However, for KF-based and IMB GED methods, memory consumption increases linearly *w.r.t* window size. Figure 14 shows the total CPU time consumed for GED by different methods over the power plant data. We see that it is possible to process 1 million rows on average in 30 *secs* with a single core mapping to a rate of 100,000 *events/sec* ($1 \text{ Million events}/30\text{sec} = 3 \text{ events/row} \times 33,333 \text{ rows/sec}$). The total time consumed for processing all data with Select(*) and GEC methods shows a slight growth *w.r.t.* window size. IMB time increases almost exponentially *w.r.t.* window size, whereas KF-based only increases linearly. IMB method executes high-dimensional matrix computations for GED. At window size 1000, IMB takes 43 *secs* to complete the task (50% slower than for window size 100). Therefore, we conclude that large window sizes are less preferable for GED since we do not want to miss important events due to delays in response. In Figure 15-16, we continue with performance (memory usage and CPU time) analysis of queries tested with petrochemical processing plant (17-lines) dataset and for different sliding window sizes. In Figure 15, we see that the memory consumption of Select(*), Classification, and KF-based increase slightly *w.r.t.* windows size, but these memory loads are not demanding compared to the memory capacity of our server. Similarly, their CPU processing times show a slight linear increase Figure 16 from 2 to 3 *mins*. However, the processing time of the IMB algorithm grows exponentially as the window size increases from 5 *mins* to 310 *mins* for the 1000 window size; beyond our charts limits. Again, we conclude that smaller window sizes and use of KF-based for GED algorithms are preferable.

Table 9: Values are average of RMSE, comparing two models over real data, using 1-4 days data for modeling.

	1	2	3	4
Holt-Winters	4.7603	3.860467	3.4932	3.3099
ARMA	3.26	2.05	1.07	1.47

3.4 *Holt-Winters vs. ARMA*

Exponential smoothing and ARMA models are the two most widely-used approaches to time-series forecasting, and provide complementary approaches to time-series problem. While exponential smoothing models were based on a description of trend and seasonality in the data, ARMA models aim to describe the autocorrelations in the data. Holt-Winters is a special case of ARMA model for forecasting time-series.

ARMA(p,q), p is the number of lagged values which represents the autoregressive (AR) [a weighted sum of past values] nature of the model, q is the number of lagged values of the error term which represents the moving average (MA). For online and batch processing two methods: Holt-Winters and ARMA model is applied on data and two approaches are compared. For online processing, Holt-Winters and ARMA models are applied on epochs of 1 to 4 days long and validated on the next day (24 hours). The RMSE values of forecasting are then compared. As shown in Table 9, the RMSE values of ARMA model is less than Holt-Winters for all window lengths. We can conclude that ARMA model fits on data more accurately and it is because of taking autocorrelation between attributes into account. The forecasted values are closer to real values and consequently, the RMSE value of ARMA is less than Holt-Winters.

As shown in Figure 17–18, the predicted values of ARMA model are closer to real value in comparison to Holt-Winters.

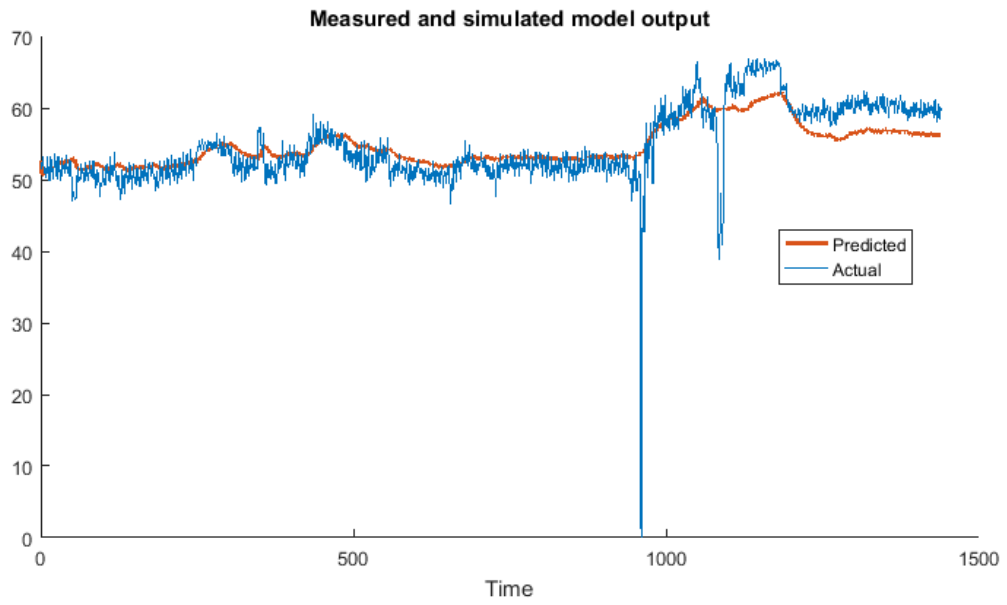


Figure 17: ARMA model: Real Data Results.

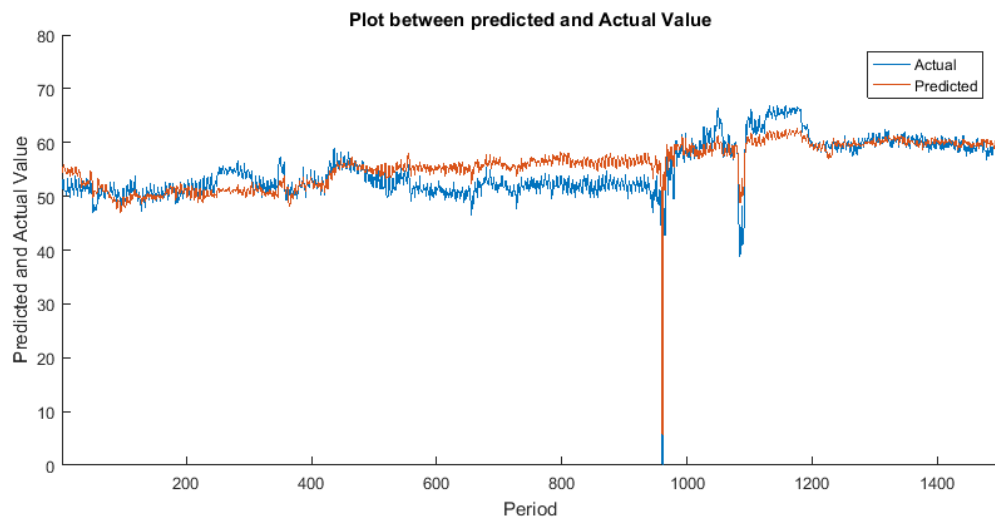


Figure 18: Holt-Winters Smoothing: Real Data Results.

3.5 Complexity of Algorithms

Algorithms developed for data stream processing must process data under certain time and space restrictions. Three algorithms were used in this thesis in GED and GEC approaches. The ARMA model is constructed on input–output data and the extracted characteristics of the system is then passed to KF-based algorithm for GED. If the gross error is detected then a classification algorithm is activated to classify the detected anomalies. The classification algorithm can be described as a decision tree, whose worst–case complexity is $\Omega(\log n)$ similar to binary trees, where n is the number of items in the tree. KF-based algorithm for tracking system dynamicity and GED is $\Omega(n^3)$ [102] in worst–case, where n is the matrices dimension. As we used the packaged JBLAS matrix multiplication and this package is an optimized library for Java, the complexity is close to $\Omega(mn^2)$.

CHAPTER IV

SYSTEM OPERATIONAL MODE IDENTIFICATION AND MODEL UPDATES

This section’s focus is on analyzing underlying system behavior using state parameters of the system by applying learning techniques on IIoT stream data in real-time. In general, a system’s functionality model is either in steady-state or dynamic, and it is crucial to select suitable models. On stream data, parameters extracted from these models frequently vary and operational conditions drift among steady-state modes [103]. Consequently, variations cause inefficiency in the decision making processes. Accordingly, the model is required to be reconstructed and time-varying parameters updated, which explains the emergence of sub-model analysis to be more reliable than batch models [104].

4.1 Sub-model Analysis and Parameter Estimation

Regression is a well known statistical data analysis technique that is applicable for system mode identification and operational sub-model construction on sensor data. Zheng, et al. [105] proposed parameter estimation with multi-operating conditions for data reconciliation in steady-state and Nadungodage, et al. [20] presented an Incremental Mathematical Stream Regression (IMSR) method for recomputing regression function online. Zhu, et al. [106] proposed a “multi-scenario” parameter estimation for dynamic systems. As the industrial systems consist of multi-operating conditions, the measurements collected change and identifying new states is useful in anomaly detection and data reconciliation [107]. Clustering methods such as K -means can be used for detecting the states of the system in comparison to RM, where data points

grouped in one cluster as one state and split in cluster estimates formation of new cluster which is interpreted as a drift to new operating condition in an emerging set of pattern. Accordingly, when a change in operating mode occurs and parameters vary, the anomaly properties of the new mode will be different than other mode. As such, a locality-based outlier detection approach is beneficial for each state of the system. Density-based clustering methods are suitable for this evaluation and DBSCAN [108] is applied on the sub-models in this thesis.

Another aspect of real-time stream analysis and parameter estimation is the frequency of sub-model update. In stream data analysis the entire data is not available at all time, the past data may have a large volume, and the cyber model needs to be constructed in real-time by using sub-models. Yet, it is challenging to detect the system's operational changes when the process is in a transient or drift mode from one operational state to another [109]. In model reconstruction, selecting an optimal window size is also critical, a window should neither be too large to miss the patterns and operational modes, not too small to make frequent, unnecessary updates. This window size is computable using historical data analysis, but it does not necessarily require a fixed length and can change over time based on system behavior.

In Section 4.2 and 4.3, we construct and evaluate a linear regression model for real sensor stream data obtained from TÜPRAŞ power plant, and compare the results with K -mean respectively and applied DBSCAN clustering method for outlier detection in window-based approach. In Section 4.4, we demonstrate how to also adapt the model by tuning window size based on TCP's congestion control algorithms and model errors.

4.2 Operational Mode Identification Methodology

Three mode identification techniques are discussed for sub-model analysis, starting with regression analysis and continuing with K -means and DBSCAN clustering.

4.2.1 Regression Analysis Method

Regression is a widely used technique that is applicable to time-varying systems for operational mode identification. Sensor measurements in industrial plants may be interdependent and fluctuations on one measurement may have effect on other measurements (*i.e.* input and output of a system). This information can be extracted from estimated parameters and used for operating mode analysis when conditions drift from a known state of the system.

The linear model is explained by Equation 20 where \hat{y}_i is the predicted response. It is clear that the predicted values will contain “prediction error” or “residual error”.

For measuring the fitness of a model RMSE and R^2 values are commonly used factors. The best line that fits the data minimizes the “sum of squared errors” by applying the least squares criterion as shown in Equation 31. RMSE value for linear mode is calculated according to this prediction error.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (31)$$

R^2 value explained in Equation 32 is a normalized measure of how well a model can predict the data and is valued as $0 < R^2 < 1$. The smaller RMSE value means the predicted model fits better to observed data, and the higher the R^2 value means the model can predict the data better. The variations of these two values are used here for operational mode identification in a window-based approach.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (32)$$

4.2.2 *K*-means Clustering

For operational mode identification, an unsupervised cluster analysis in a window-based approach using *K*-means algorithm is also applied on sensors data. With *K*-means, n sensor observations are partitioned into K clusters, where each observation belongs to a cluster with the closest mean. The centroid c is chosen by minimizing the objective function described in Equation 33 based on the squared Euclidean distance [110].

$$J = \sum_{j=1}^K \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2 \quad (33)$$

When a change occurs in data, the pattern and clusters will vary as well. A division in clusters can either be interpreted as mode change, or the system is in a transient state. This approach can be used for validation of RM analysis.

4.2.3 DBSCAN Clustering

Real-time clustering methods can be used for detecting the system's operating states, where data points would be grouped in one cluster denoting the steady-state and formation of new clusters are interpreted as drift to new operating conditions or emerging patterns. Note that the anomaly properties of transients will be different than steady-state modes. As such, a locality-based outlier detection approach, without specifying cluster numbers in advance is beneficial. Density-based clustering methods are suitable for this evaluation, therefore we employed DBSCAN.

The application for sub-model identification in real-time stream analysis are operating state identification and local outlier detection.

For all series of points q , that are density-reachable from p one cluster is formed from connected points described in Equation 21 and points that are not reachable are detected as outliers in a window-based analysis [42].

4.2.4 System Operational State Analysis in Real-time

Another aspect of real-time stream analysis and parameter estimation is the frequency of model update. In stream analysis the entire data is not available at all time, the past data may have a large volume, and the cyber model needs to be constructed using a window-based approach, *a.k.a.* sub-model identification. Yet, it is challenging to detect a system's operational changes when the process is in a transient or drift mode from one state to another. In model reconstruction, a window should neither be too large to miss the patterns and operational modes, not too small to make frequent, unnecessary updates. The performance evaluation shows that smaller windows sizes are preferable because of lower CPU time and memory usage. Optimal window size can be computed using historical data analysis, but it does not necessarily require a fixed length and can be changed over time based on the system's behavior [111].

Algorithm 2 SystemModeTracker

```
1: procedure  
2:  $W$  : initial window size  
3: offline:  
4:  $M = Model(W)$  //predict model on steady-state data  
5: online: DREDGE  
6: for all windows  $W$  over stream do  
7:    $GED(W)$   
8:    $RMSE_{currModel} = RMSE(M)$   
9:    $cluster = DBSCAN(W)$   
10:  if  $(RMSE_{currModel}) > (RMSE_{prevModel})$  &  $cluster > 2$  then  
11:     $detectNewMode$   
12:     $M = updateModel(W)$   
13:     $RMSE_{prevModel} = RMSE_{currModel}$   
14:  end if  
15:   $GEC(W)$   
16: end for
```

Stream data context fluctuations in industrial systems is a critical indicator for system modeling and the necessity for model updates. This information is extracted using the cleaned, high quality data obtained from DR-GED process. Here, the system is modeled using the analysis described for GED and GEC in a window-based

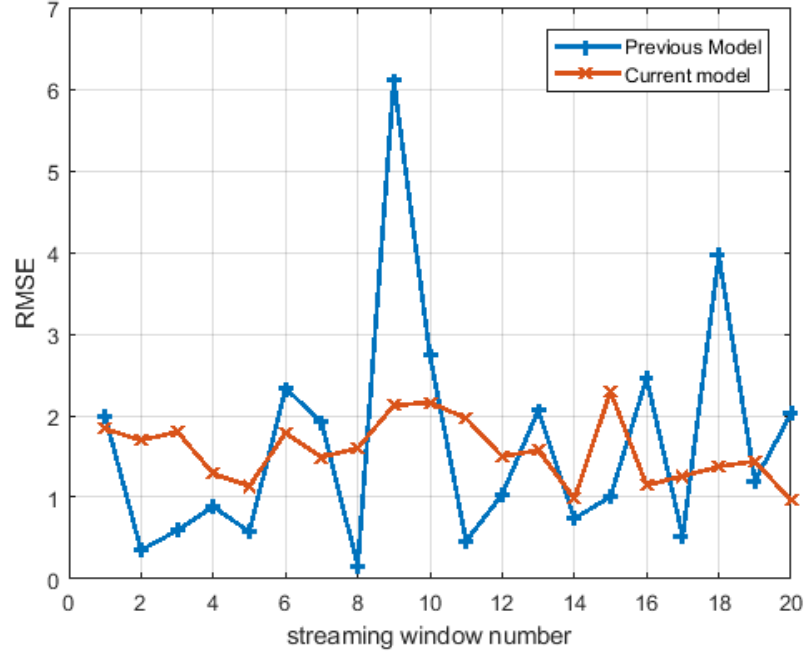


Figure 19: Model evaluation of pressure/temperature data.

fashion, with a fixed window size of 180 data points (6 hours) in consecutive tumbling windows. By applying the model from the previous window to the current window and measuring the RMSE from the predicted model, the RMSE value is evaluated for operating state identification. We observe that when there is a drift in data context the RMSE increases dramatically as shown in Figure 19 window #9. When the system works in one steady-state, the previous model is applicable to the current window; for example between window #6→#8. But when the RMSE increases suddenly, the system goes into a transient state and is an indication of state change. This extracted knowledge is interpreted as a requirement for a model update as described in Algorithm 2. The validation of this extracted knowledge is tested using DBSCAN clustering method as described next. Note that in *SystemModeTracker* Algorithm, offline models were used to get the sensor stream started; after that models are trained and tuned online.

4.3 Experiments and Results

In our experiments, we used a set of real data collected from sensors in power plant of an oil refinery. This power plant has 80 megawatts electric generation capacity. It consists of 8 boilers, converting hot water into high-pressure vapor at a maximum capacity of 100 *ton/hour*. The vapor is fed into steam turbine to convert thermo-kinetic energy to electricity. Hot water (input) and high pressure vapor (output) flow rates measured by the flow sensors are the main measurements for streaming sensor data analysis. The changes in modes can be due to operator-driven switching among the desired vapor pressure levels (low, high, very high) or system-driven and auto-controlled modes for heating, cooling, recycling, and condensing. By offline analysis, it was discovered that the system is time-varying, its operation is composed of several steady-states and the input/output values of the boiler have a correlation of about 90%.

This sensor data is modeled using linear regression in a window-based approach, a fixed-window size of 360 data points (sampled every minute over 6 hours) and used for sub-model construction. In linear time-varying systems, the underlying model requires to be reconstructed when the pattern and operational functions' parameters change. The main challenge is detecting the time for sub-model update. In addition, the amount of information from past data to be used in current window plays an important role in accurate system analysis. The approach used in this study follows the construction of an optimal linear regression sub-model for the current window. For the next window, the same RM from the previous window is applied first, and the RMSE is calculated, if the RMSE value is increasing, the sub-model is reconstructed and state parameters are updated according to current window observations. The RMSE and R^2 variation values used together for detecting the operational mode and the need for model update. Otherwise, the same model is retained. In summary, the sub-model trained for the previous window is tested with the current window to

Table 10: R^2 values of fixed-window size for regression model prediction performance.

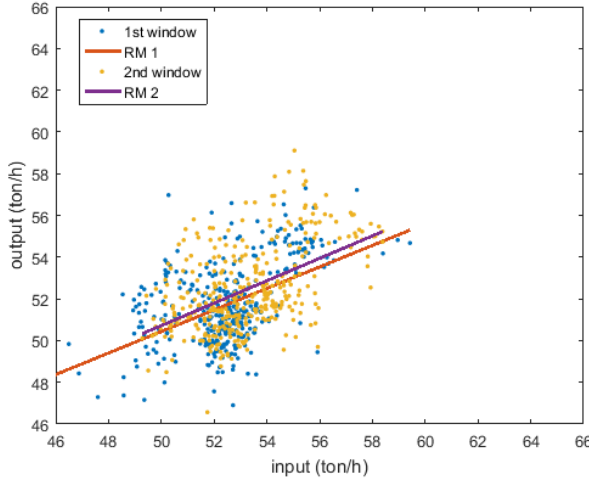
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Current RM	74.98%	75.03%	74.54%	92.74%	88.45%	25.95%	89.35%	99.64%	41.64%	78.44%
Previous RM	0%	74.01%	11.27%	0%	38.26%	59.42%	0%	81.39%	0%	89.56%

decide whether to maintain the model or update it.

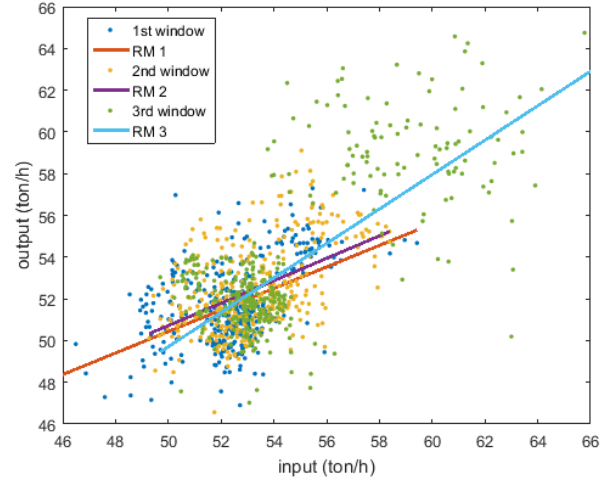
4.3.1 Regression Model Evaluation

In Figure 20(a)-(d) RM trained on input vs. output flow sensor data from one boiler are depicted. Figure 20-(a) shows distributions of two adjacent time window (6 hours) readings that have a high overlap and the sub-model for window #1 is applicable on window #2 since the RMSE values shown in Figure 21-(a) are very close. In Figure 20-(b) a new set of data are received and applying RM from the previous window increases the RMSE and R^2 decreases dramatically from 74.01% to 11.27% as shown in Table 10, column #2→#3. This change indicates that the current window readings are not represented well by the previous sub-model, a reconstruction is required and the system is in a transient state, therefore a drift from existing operational mode is detected. Receiving new data from window #4, Figure 20-(c), the system has clearly drifted into a new operational mode, the slope of the fitted line has varied and previous RM does not fit the current window. This transition in operational mode is extracted by the RMSE value evaluation where a local maxima is observable in Figure 21-(a) and the previous sub-model does not fit the data. R^2 is also decreased as shown in Table 10, columns #3→#4, where R^2 value 0% refers to unfitting RM.

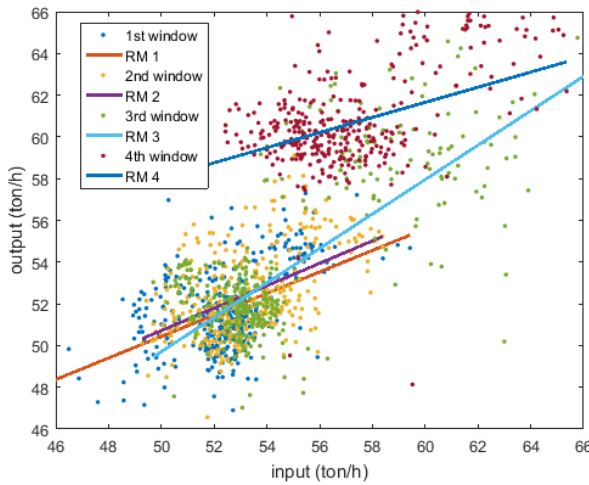
In Figure 21-(a), the RMSE values of RM from the last observed window is compared to the current window. This analysis shows that when the RMSE value is growing or has a sudden deviation, a new operational mode is identified. The global maximum at window #9 shows the formation of new operational mode and local maxima are related to the transitions or drifts among operational modes inside the system. In Figure 21-(b) around 3000 minutes and windows #8→#9 this drift is



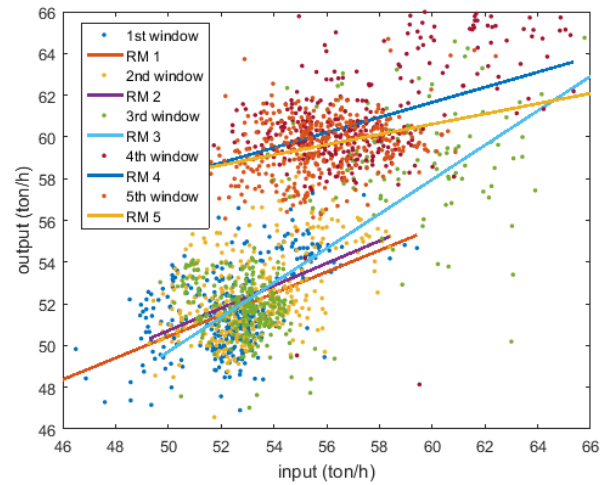
(a) RM: 1st, 2nd window



(b) RM: 3rd window



(c) RM: 4th window



(d) RM: 5th window

Figure 20: Regression model on streaming sensor data, (a) in windows #1→#2 RM models are close and system works in the same operational mode, but in (b) window #3 data requires a new RM sub-model since the distribution is changed and shows a transition among system's operational modes, (c) system clearly drifted to a new operational mode with updated RM in window #4, and (d) window #5 stays in the new operational mode.

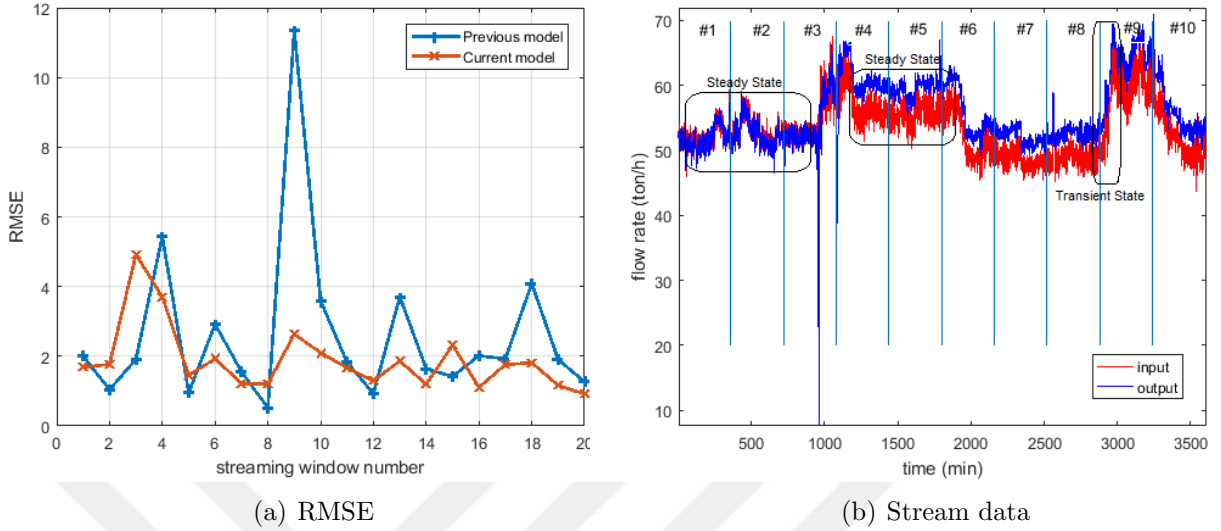


Figure 21: (a) Comparison of RMSE values of regression sub-model from previous window and current streaming window, (b) window-based stream data used for operational modes identification analysis depicted for windows #1→#10.

observable, while simultaneously R^2 shows a very low prediction performance and is decreased to 0% in Table 10 window #9. The steady-state and transient states are depicted in Figure 21-(b).

4.3.2 K -means Clustering vs. RM Mode Identification

The observations used for this experiment are the same stream data used in RM. The goal is to compare the formation of sub-models in transient states, and how an unsupervised clustering method reacts to drift in system state without knowing the underlying relations among variables. We used $K = 2$ for this experiment since the system is either in steady-state or transient mode drifting into a new steady-state. In Figure 22, windows #1→#2 (green and yellow colors) are clustered with very close centroid points, which confirms the results of RM where the previous window model was applicable to the current window according to RMSE, R^2 values. Similar to RM method on arrival of window #3, the observations are divided into two partitions with distant centroids as shown in Figure 22-(a). The window #4 approves this

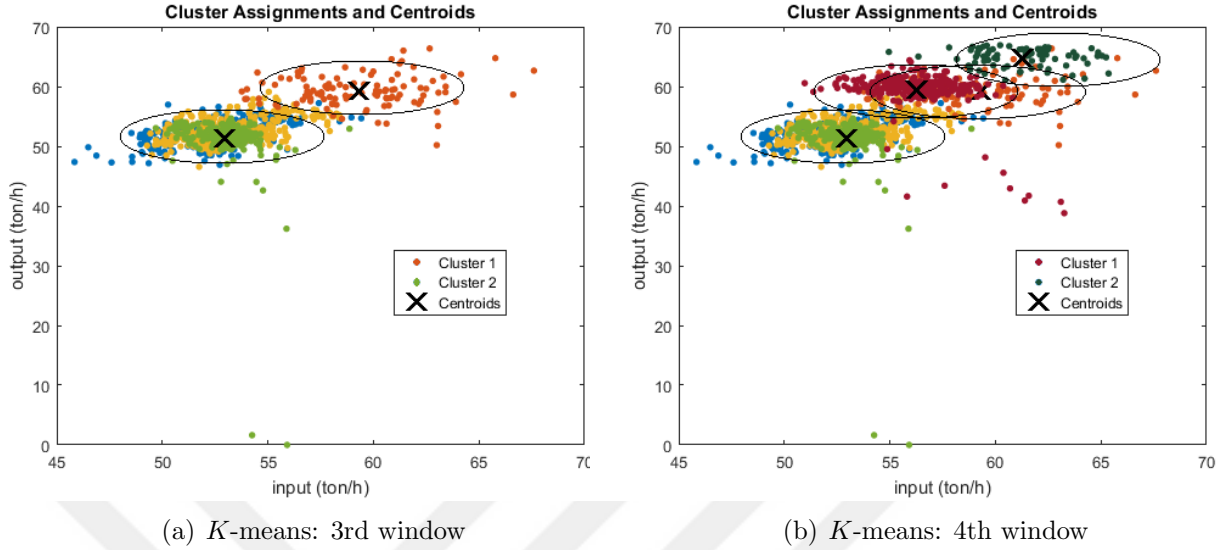


Figure 22: *K*-means clustering is also used for operational mode identification, in (a) window #3, observations are clustered into two partitions in a transient mode, (b) window #4 observations consist of two clusters where one centroid has a close distance to previous centroid from window #3.

transient mode by clustering data into two partitions by *K*-means shown in Figure 22-(b). Clusters formed in the successive windows are plotted on top of each other for reference, which matches the results of RMSE value in window #4 that reaches a local maxima and the R^2 value decreases to zero. In all cases, we also have a minimal number of outliers outside any clusters.

4.3.3 Density-Based Clustering

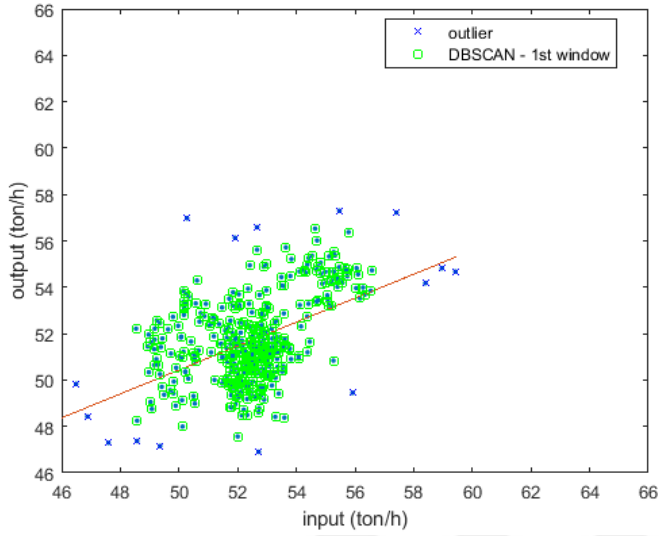
This algorithm is used on fixed-size window stream data. As data stream arrives DBSCAN algorithm is applied for outlier detection of the current window. As shown in Figure 23-(a) in window #1 data points are clustered into 2 clusters of inlier and outlier, on arrival of window #2 Figure 23-(b) the data is still in the steady-state as 2 clusters are detected. However, in window #3 the data split into 3 clusters, which indicates a transient in the system that is confirmed by RM and *K*-means algorithm depicted in Figure 23-(c). The window #4 stays in the new steady mode as shown in Figure 23-(d).

The benefit of using DBSCAN on top of sub-model identification is that it enables detecting the outliers within the window and that the system is working under a steady-state operational mode. If a transient in mode happens, DBSCAN can detect the drift by partitioning data into more than one cluster of inliers and outliers. In comparison, a fixed-count K -means with $K = 2$, would not be able to show us more than 2 clusters, yet DBSCAN would be able to detect multi-steady-states as shown in Figure 23-(c), where 3 clusters are formed in 1 window. For the modeled boiler device, still, DBSCAN did not remain in 3 clusters (2 + outlier) for too long and stabilized in another steady-state mode. Also, the parameters and outliers change nonlinearly between multi-operating states, thus clustering helps to decrease its chaotic effects.

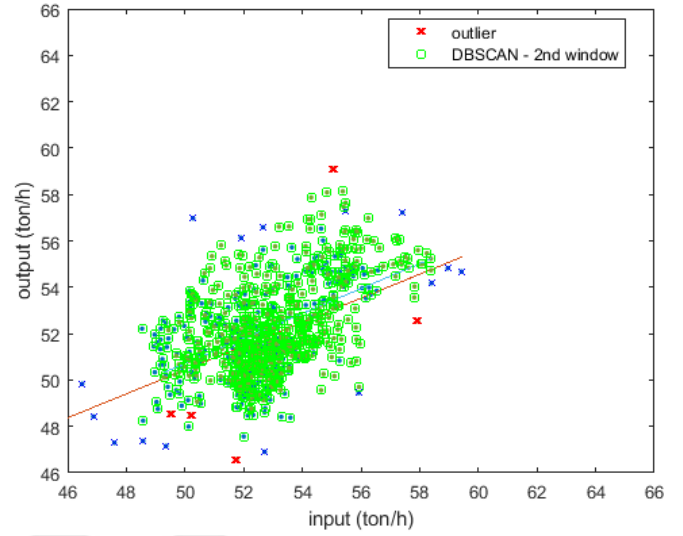
4.3.4 Using DBSCAN for Mode Change Detection

As the data stream arrives, DBSCAN algorithm is applied for operating state identification and outlier detection of the current window. In Figure 24, the behavior of the system is studied for water/vapor relation. As shown in the figure, until window #8, there is only one main cluster, but outliers are beginning to show the emergence of a second cluster. However, in window #9 the data split into 3 clusters, which indicates a transient in the system. The data received in window #10→#11 stay in the new steady-state mode. DBSCAN clearly enables detecting the outliers without any prior assumption about the distribution of data or any relationship among variables, and whether the system is working under a steady-state operational mode. If a transient happens, DBSCAN can detect the drift by partitioning data into more than one cluster of inliers and outliers.

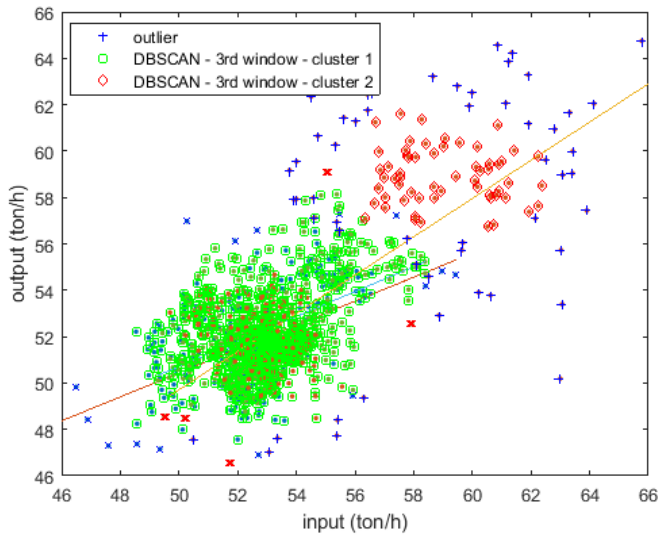
In the meantime, the operation of the system is evaluated from other sensor measurements of the boiler, pressure and temperature values. As shown in Figure 24, the behavior of the system using flow rate measurement is evaluated for operating state identification. However, using other sensor data such as pressure and temperature as



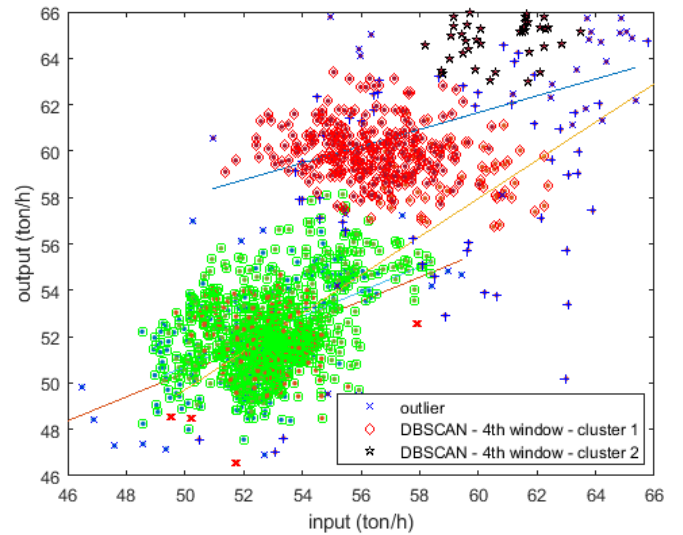
(a) DBSCAN: 1st window



(b) DBSCAN: 2nd window

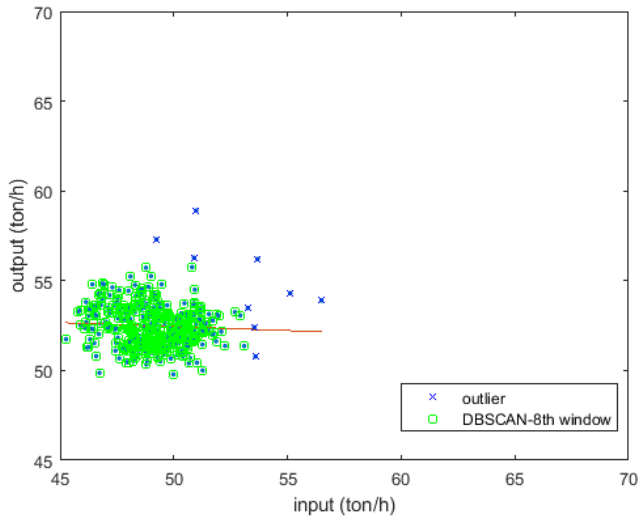


(c) DBSCAN: 3rd window

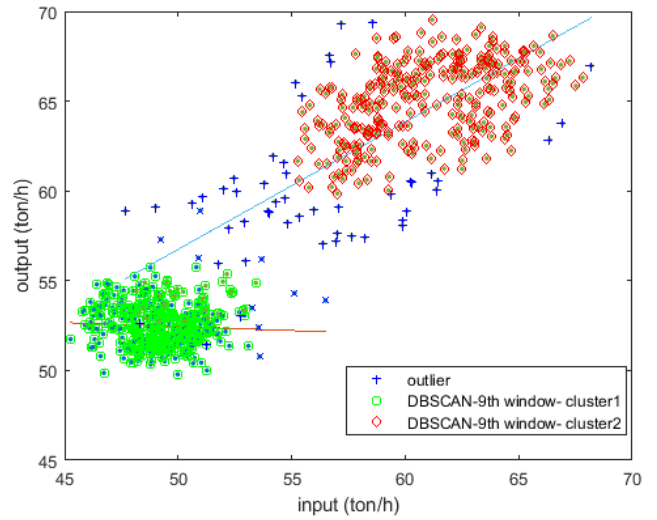


(d) DBSCAN: 4th window

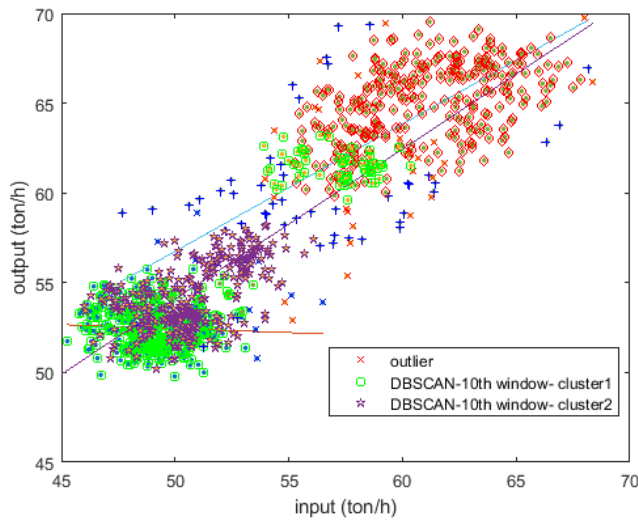
Figure 23: DBSCAN clustering, (a) in window #1 two clusters are identified in steady-state mode outliers and inliers, in (b) window #2 the system is under same operational state as window #1, but in (c) window #3 data points are partitioned into 3 separate clusters since the distribution is changed and a transient in system operational mode has occurred, (d) system clearly drifted to a new operational mode.



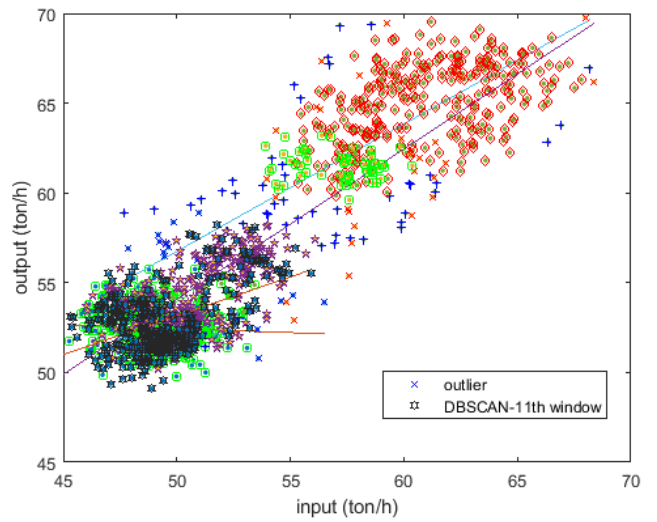
(a) DBSCAN: 8th window



(b) DBSCAN: 9th window



(c) DBSCAN: 10th window



(d) DBSCAN: 11th window

Figure 24: DBSCAN model on streaming sensor data water/vapor flow rate: (a) in window #8 (refer to Figure 19), DBSCAN model shows formation of one cluster of inliers indicating one operational state, (b) as new data samples are received from window #9, data gets split into 2 main clusters and some outliers that indicates a drift in data context and that the system is in a transient mode. (c)-(d) Windows #10→#11 show that the system is operating under a new steady-state mode.

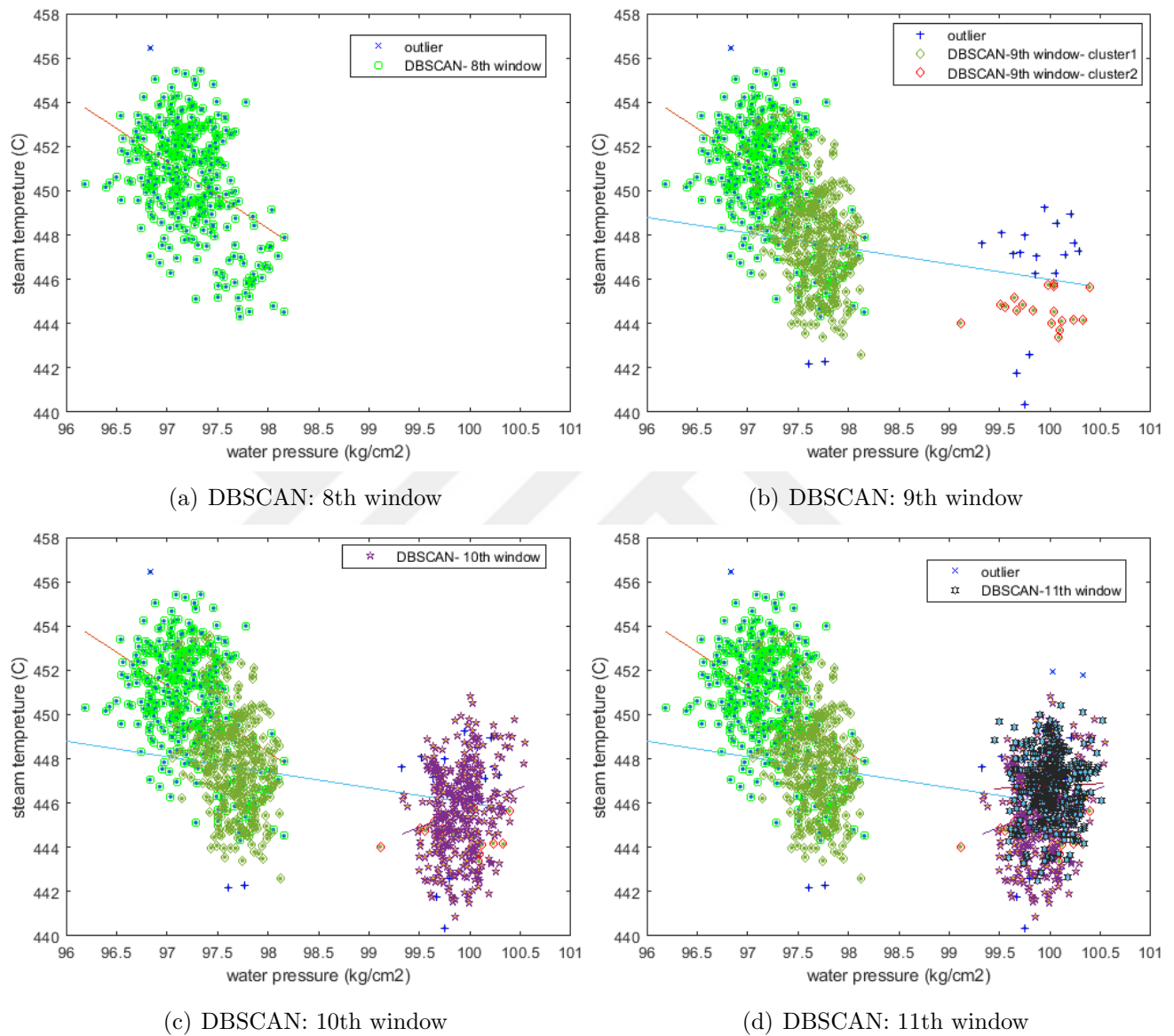


Figure 25: DBSCAN model on streaming sensor data water/vapor, pressure/ temperature: similar to flow rate, (a) in window #8 DBSCAN model shows formation of one cluster of inliers indicating one operational state, (b) in window #9, data split into 2 main cluster and some outliers that is a distinctive indication of drift in data context and the system goes into transient mode, requiring a model update. (c)-(d) Windows #10→#11 show that the system is operating under a new steady-state mode.

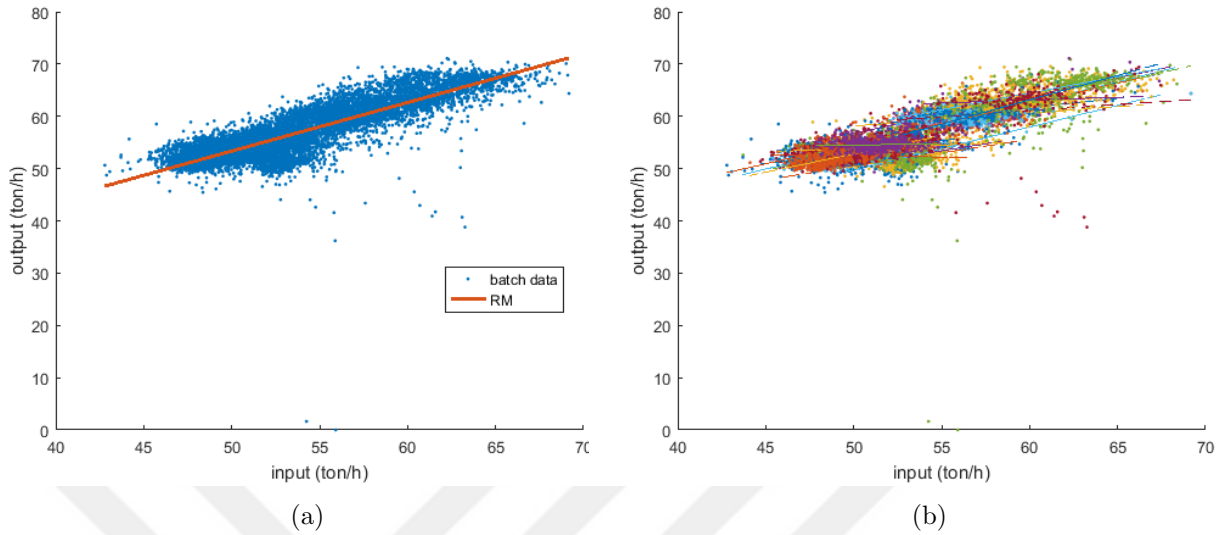


Figure 26: Comparison of regression model result on (a) batch analysis, (b) window-based analysis. Although a single regression model describes the data, operational modes of the system are not distinguished by batch analysis.

shown in Figure 25 we can observe the same behavior with more distinction. In window #8 the system is operating under one steady-state, and by receiving the new set of data in window #9 a transient is observable since the data is split into 3 different clusters. Windows #10→#11 approve this transition and the system is observed to stay in the new operating state.

For online data analysis all the streaming data is not available at a certain time, consequently window-based, sub-model construction is required. If a model is constructed on a relatively large window size, or batch data offline, the underlying model may not represent local phenomena accurately and some critical information about the system will never get extracted. In batch data analysis shown in Figure 26-(a) a large cluster of data points is formed by the accumulation of several small windows and the operational sub-modes may not be detected using one RM. However, in window-based or streaming analysis 26-(b), previously hidden information about the time-varying system behavior is revealed which is important for real-time decision making. In addition, processing large volumes of data is neither timely nor efficient.

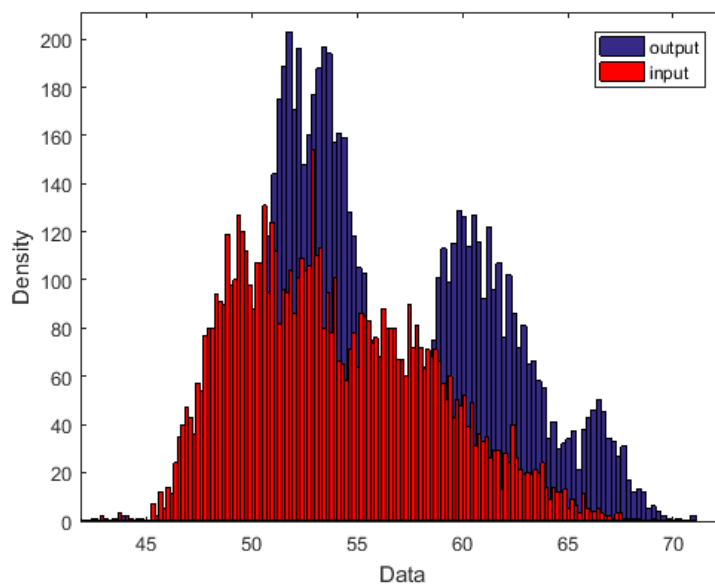


Figure 27: Histogram of input and output flow rates (*ton/hour*).

The frequency of running model updates and sub-model identification depends on the computational complexity of the algorithms and the computational power at hand. In modern distributed data mining approaches, all the data may not be available in one computational node and algorithms may need to be applied locally to summarize the results and share among the nodes. It is desirable for these approaches to be efficient and scalable. Moreover, the amount of information from past data to be used in current window plays an important role in accurate system analysis.

From Figure 21-(a), we observed that system changes states and operates under several operational modes, while the distribution of data clearly change during these drifts. This extracted knowledge from streaming data analysis on the same fraction of data is approved by studying the distribution as shown in Figure 27 where data distribution is a combination of several Normal distributions.

Table 11: Gaussian parameter analysis in fixed-window size for sub-model detection.

window #	peak 1	peak 2	<i>std</i>	<i>RMSE</i>	R^2
1	59.28	47.95	6.32	1.5784	0.3525
2	57.32	49.51	10.43	1.7209	0.2997
3	63.38	54.54	34.5	4.7573	0.3152
4	64.82	51.15	30.82	3.7044	0.0874
5	64.59	59.72	2.976	1.4312	0.15

4.3.5 Using Two-Term Gaussian Function for Mode Change Detection

For mode change detection Gaussian Distribution Function is applied on sensor data using the same window-based approach, a fixed-window size of 360 data points and utilized for sub-model construction. In this experiment a two-term Gaussian Distribution Function is applied on data. The Gaussian functions are often used to represent the probability density function of a normally distributed random variable with expected value $\mu = b$ and variance $\sigma^2 = c^2$. The parameter a is the height of the curve's peak, b is the position of the center of the peak and c (the standard deviation) controls the width of the bell curve. This function is formulated as follows:

$$f(x) = ae^{-\left(\frac{x-b}{c}\right)^2} \quad (34)$$

As shown in Figure 28 according to change in data distribution, the sub-models are recognizable. This information is also obtained by analysis of Gaussian model parameters provided in Table 11. In window #1→#2 the peak of both Gaussian functions are close and the standard deviations (*std*) do not fluctuate, but on the arrival of window #3→#4 the peaks shift and *stds* and RMSE values increase dramatically as shown with red color. This information is useful for sub-model identification and similar to RM, DBSCAN and K -means approaches, the drift in operational mode is extracted by applying this approach.

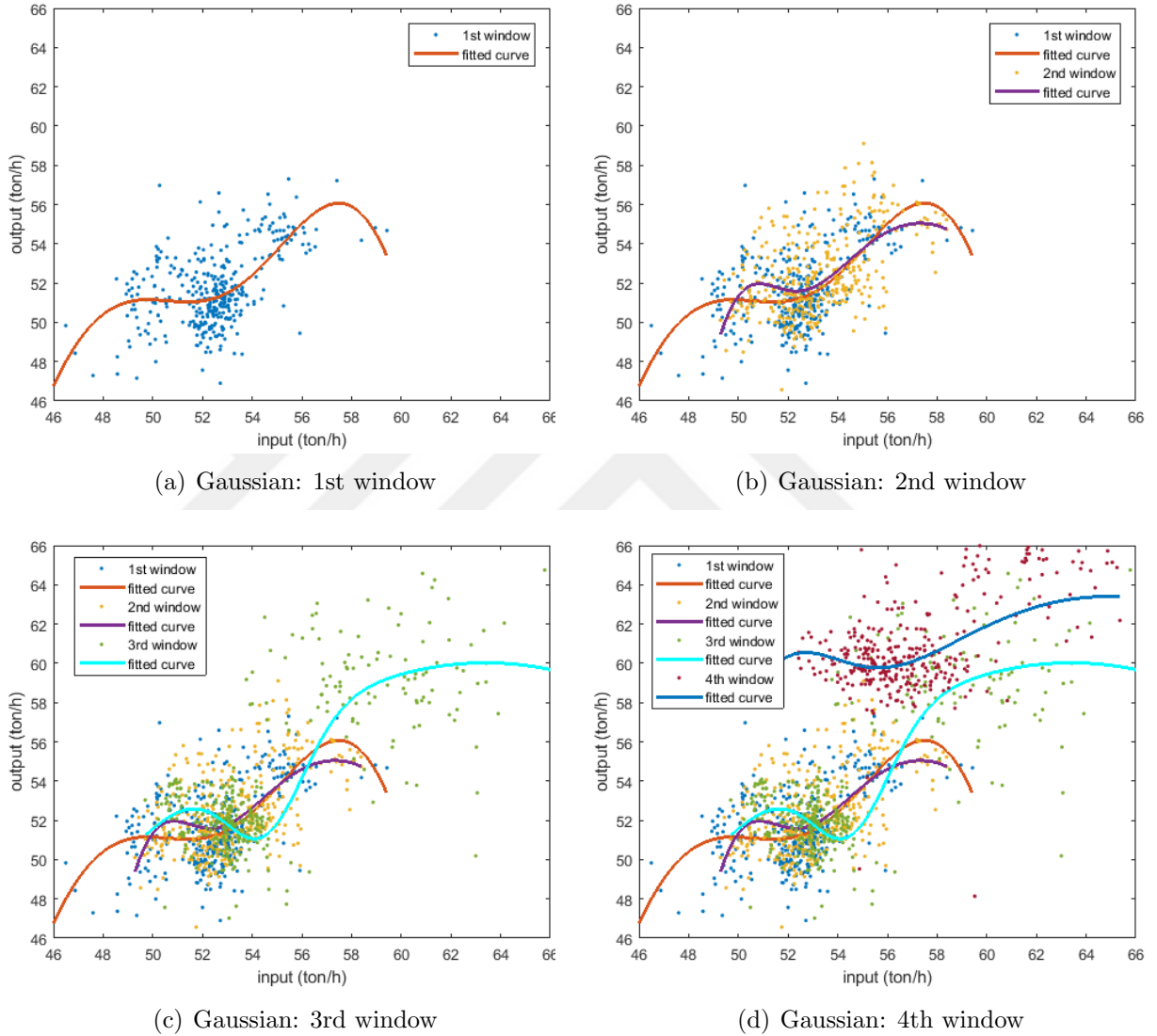


Figure 28: Gaussian distribution model on streaming sensor data, (a)-(b) in windows #1→#2 Gaussian curves are close and system works in the same operational mode, but in (c) window #3 data requires a new Gaussian sub-model since the distribution is changed and shows a transition among system's operational modes, (d) system clearly drifted to a new operational mode with updated curve in window #4.

4.3.6 Discussion: Sensor Error or System Anomaly

One crucial question to answer is whether an outlier measurement would occur due to a sensor malfunction or system anomaly. The identification of these correlated high-level events could be difficult [112], [113]. However, due to redundant sensors and laws of mass and energy preservation, the system can be monitored in multiple locations (in & out) as well as in multiple dimensions (flow, temperature, pressure) to differentiate sensor *vs.* system issues. In our experiments, we observed that the system will sometimes shift among regular operational modes and go through transient states, which get detected as gross errors in our scenarios. Consequently, in “Bias” and “Drift” types of gross errors, the system goes through a transient state until it reaches a new steady-state operation. In “Precision Degradation” type there is no steady-state and in “Failure” type there is no sensor or system operation at all.

The system is declared as operating in a steady-state when the model from the previous window that is applied on new window data fits well by RMSE value evaluation. However, a new operational mode formation can be identified when prediction error of the previous model on the current window increases dramatically. The operational mode identification compared and is confirmed by DBSCAN clustering method.

4.4 Adaptive Window Size Tuning

Until now, fixed-size windows were used for regression-based operational mode identification. However, the window size doesn't need to stay the same; yet finding an optimal size is also challenging. For time-varying data, based on the system behavior and received patterns, the window size may be updated. In this section, we propose and test an adaptive-window based regression model update schema. We utilize the time-tested TCP (Transfer Control Protocol) Congestion Control algorithm [114]. This algorithm uses slow-start and congestion avoidance phases for reliable data

Algorithm 3 :Adaptive window size tuning

```
1: procedure AWST(S)
2:   minWin : Minimum window size
3:   maxWin : Maximum window size (cwnd)
4:   win : Initial window size
5:   s = Swin : Sub-model window
   #Slow-start
6:   for each sub – model(s) do
7:     if (RMSEs < threshold) then
8:       win = win * 2
       #Congestion avoidance & Fast recovery
9:     elseif (RMSEs > threshold)
10:      win = win/2
11:     if (win > maxWin||win < minWin) then
12:       win = minWin
13:     s = Swin
   perform #Slow-start
```

transmission, but we utilize the methods for adapting the streaming window size selected for model updates. The RMSE values of consecutive windows are evaluated by applying the algorithm for tuning the window size as follows:

- Slow-start: doubles the window size each round RMSE variation value is less than threshold,
- Congestion avoidance: enters the linear growth phase,
- Fast recovery: during congestion avoidance mode, the congestion window size is reduced to slow-start threshold (ssthresh) rather than the much smaller initial value.

The proposed adaptive window size tuning algorithm is explained in the Algorithm 3 pseudo code.

The window is initialized with a length of 180 minute readings (3 hours). As shown in the Figure 29, when the RMSE value of the model is less than a threshold, the window size is doubled following slow-start. In contrast, if the obtained RMSE value of the current window is higher than the threshold value, the window length

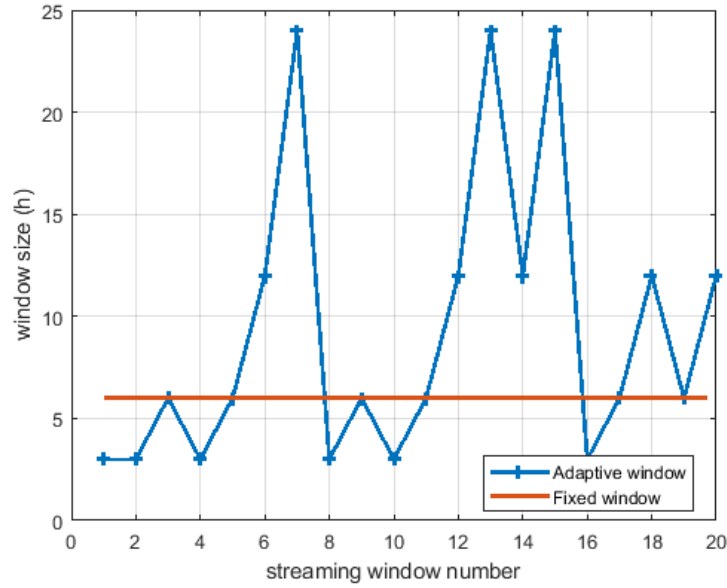


Figure 29: Adaptive window size tuning over data.

is decreased to half the size. Consequently, using adaptive window tuning algorithm has a significant positive impact on R^2 values shown in Table 12. The R^2 values obtained from adaptive-window based regression model update schema has increased in comparison to fixed-size window described in Table 10 which shows that adaptation can improve the model prediction performance and operational mode identification. As shown in Table 12 when RMSE values are less than the threshold the window size grows #2→#3 and #4→#7 meaning that the system is in steady-state but, when it has high variation, transition among operational modes is occurring. In window #7 the algorithm goes to fast-recovery phase as window size decreases to minWin. In window #3 and #13 the RMSE value has increased dramatically and according to fast-recovery, the window is decreased to half size. Meanwhile increasing RMSE values are interpreted as a transition to a new mode. As shown in Figure 29 dataset of length 186 hours is covered with 20 adaptive-sized windows *vs.* 31 fixed-sized (*i.e.* 6 hour) windows. The proposed adaptive algorithm therefore reduces modeling related computational costs by $\sim 35\%$ (11/31 updates).

Table 12: R^2 , RMSE values of adaptive-window size for regression model prediction performance.

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
R^2	98.04%	72.60%	74.54%	92.61%	88.46%	83.63%	13.60%	98.42%	41.64%	96.67%
RMSE	1.36	1.66	4.92	1.34	1.44	1.55	1.91	0.96	2.63	1.30

4.5 Summary and Conclusions

We showed that by evaluating the fluctuation of error values mode changes are detectable. The results were also verified by real-time K -means clustering. This approach provides information about steady-state, drifts and transient states of time-varying industrial systems and the requirement for real-time model updates. An outlier detection algorithm using windows-based DBSCAN clustering was also used as a novel strategy for multi-state operational mode identification. Finally, the proposed TCP congestion control-based adaptive window size tuning for streaming regression analysis resulted in reductions in RMSE values as well as saving modeling computational costs of frequent updates.

CHAPTER V

A CLOUD-BASED BIG DATA PROCESSING ARCHITECTURE FOR INDUSTRY 4.0

Industry 4.0 revolution manifests that IoT, Cloud, and Big Data Analytics technologies should be used together to deliver a digital transformation and establish smarter operation for all sectors including Oil & Gas industry. However, transforming existing industries at levels 2.0 and 3.0 to level 4.0 will entail *all* of the Big Data challenges: volume, velocity, variety, and veracity (4V) to be addressed, simultaneously. Fortunately, both of the manufacturing industries and digital technologies are ready to make this transformation. Yet, everyone is looking for quick and effective ways to complete the task. In this Section, we propose a cloud-based big data “*Lambda*” architecture based on the GED, GEC tools developed in the previous Sections. Refineries have already implanted thousands of sensors inside and around their physical systems. IoT and IIoT, sensor networks, digital media, and business transactions generate large amounts of data that continuously stream in via DCS and SCADA systems’ measurements and should get extracted, transformed, and processed efficiently. Knowledge discovery and decision making from such data are challenging when there are millions of various measurements feeding in [77], [115].

Industrial data-intensive computation and storage have been considered as an important paradigm for science: vast amount of data obtained from applications and the infrastructures are stored, shared and processed collaboratively [13]. Furthermore, rapid collection of big data everywhere introduced open-source distributed frameworks such as Apache Hadoop [55], which is a powerful framework for parallel computing of data gathered from scientific, industrial and business processes [116].

Hadoop runs MapReduce jobs [117] over the data stored in Hadoop Distributed File System (HDFS) [118], object-based systems (Ceph) [119] and even NoSQL databases (MongoDB, Cassandra) by creating parallel algorithms [120]. Hadoop framework created an opportunity for designing a completely open-source “*data architecture*” for processing Big Data coming from real-world resources and improve decision support processes (HBase [121], Hive [122]). Apache Mahout was the first project to enable running Machine Learning algorithms on top of MapReduce. More recently, Apache Spark [89] was introduced as an in-memory distributed alternative to MapReduce, which also boosted the ML algorithms to be executed with 100x performance. The ML library that runs over Apache Spark is simply called MLlib [123].

5.1 *Cloud Computing*

Cloud Computing is defined by National Institute of Standards and Technology (NIST) as: “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources” (e.g., servers, storage, network, services at all layers, and applications) “that can be rapidly provisioned and released with minimal management effort or service provider interaction” [124]. It also provides features needed for massive data streaming applications. This trend is in large part due to the development of new processes that allowed IT departments and data centers to ETL (Extract, Transform and Load) massive amounts of data efficiently and inexpensively. Examples of popular Public Cloud Services include Amazon AWS (<https://aws.amazon.com>), Microsoft Azure [125], Google Cloud [126], IBM Bluemix [127], and many more.

The Cloud architecture includes three layers as shown in Figure 30.

Software as a Service (*SaaS*): refers to end-user software provided via a browser and the Internet. It may be layered on top of public PaaS and IaaS or completely private data centers of the service provider. Examples of SaaS include:

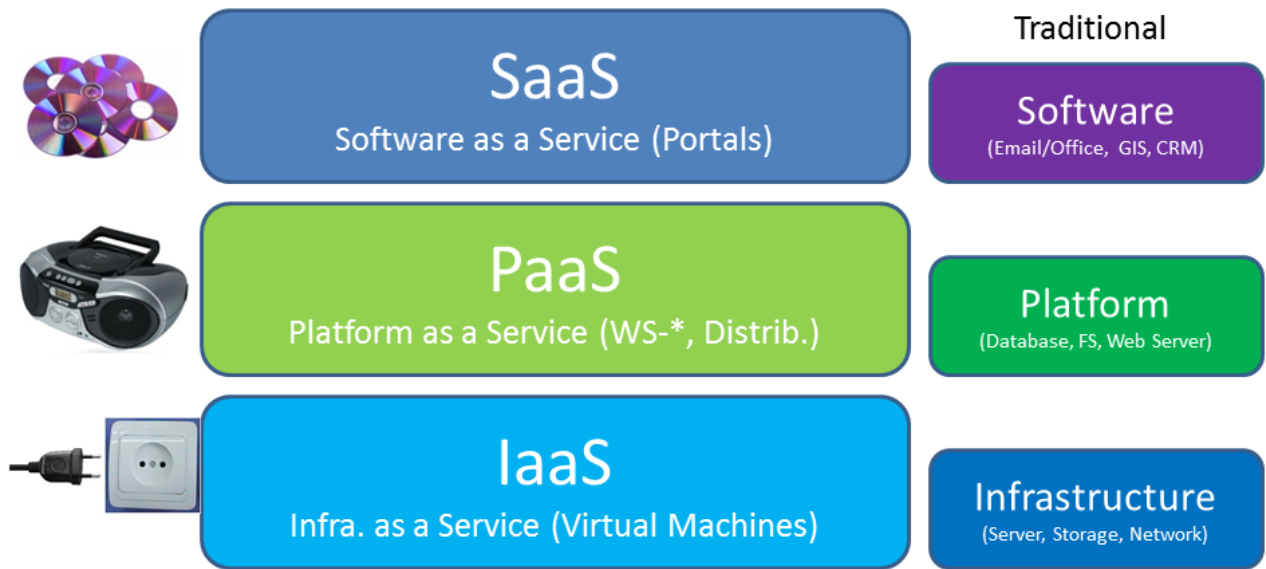


Figure 30: Three cloud service layers.

Microsoft Azure Office360, SalesForce [128], Dropbox, and many more of Email systems, and business data processing systems such as Customer Relationship Management (CRM), Enterprise Resource Planning (ERP) [129] and accounting services.

Platform as a Service (*PaaS*): this layer has the middleware software components on top of which other end-user software can be developed by programmers or SaaS service developers. The analogy made for PaaS, SaaS and IaaS in Figure 30 is that PaaS is the CD player platform, whereas SaaS are the different albums, and IaaS is the electricity that powers it all.

Infrastructure as a Service (*IaaS*): is about suitable computing, storage and networking hardware that is required on which the software could be run properly.

Cloud is also our medium for running Industry 4.0 big data solutions in scalable way. The industrial data used in this work is a streaming data that has to be processed in real-time, besides being stored for offline analysis as well. Algorithms

for pattern matching, anomaly detection using K -means clustering, dimension reduction procedures (PCA, SVD), *etc.* can be applied after the data preprocessing and reconciliation phases. These algorithms can be performed on either Apache Spark or Hadoop on top of a real cluster. We use Apache Spark [130] and integrate ML algorithms using Scala programming language.

We propose to use Lambda architecture as the unified data and analytics architecture for industrial plants specifically for Oil & Gas plants, which consists of three layers: *batch processing layer* for offline data, *serving layer* for preparing indexes and views, and *speed layer* for real-time processing [131]. In the following subsections, we introduce Lambda architecture implementation alternatives in private and public clouds.

5.2 Private Cloud Implementation

We propose to use Big Data processing architecture including a PaaS named in this thesis as DREDGE. Operations running inside an organization's data center is an example of a private cloud. An important advantage of private clouds is higher security, easier maintenance, and direct control over the deployments [132]. In Oil & Gas sector, data and applications may require a higher level of security and these companies can afford to build their own private cloud.

To manage data storage, processing, and analytics at scale Oil & Gas industry recently started experimenting with open-source distributed frameworks such as Apache Hadoop, Apache Spark and Apache Ignite [133]. Apache Hadoop consists of HDFS, MapReduce, HBase, Hive, and other system management modules. Apache Spark solves online processing problems of disk-based Hadoop by implementing MapReduce layer entirely in-memory [134], but its Resilient Distributed Datasets (RDDs) are immutable. Apache Ignite provides shared, mutable in-memory RDDs [135], called data grid, which is a NoSQL key-value store (partitioned hashmap)

that can be employed for providing distributed caching or complementing Apache Spark. While these technical developments in distributed systems and their ease of availability are useful for the Oil & Gas sector, there are tens of open-source data processing & mining projects with no unified view about how to leverage them coherently, because:

- Oil & Gas industry has not developed a comprehensive data architecture to complete its digital transformation neither in the drilling nor refining businesses, and,
- the integration with private and public clouds is not discussed clearly.

The proposed architecture is depicted in Figure 31 for a private cloud [6]. Apache Kafka [136] is an open-source publish-subscribe engine for building real-time data pipeline and stream processing applications. Sources of data streams such as sensors, log files, and IoT act as the publishers. Applications in the speed and batch layers subscribe to Kafka for pulling the data and processing them in real-time. Spark Streaming [89] receives live data streams and divides them into smaller batches. This data is also stored in a distributed database management systems such as Apache Cassandra [137], which also replicates it internally to several nodes to ensure reliability. Distributed algorithms can now have access to replicas via MapReduce framework of Apache Spark API. Complex analytics such as pattern recognition, classification, rule extraction, and fault detection are performed in the serving layer. MLlib [138] is an open-source machine learning library which works on top of Apache Spark as well as Cassandra. The resulting real-time (online) models in speed layer and precomputed (offline) models in the batch layer are merged for visualization and prediction purposes.

The private cloud data system can be integrated with public cloud services such as Amazon Web Services (AWS) [10]. Each open-source project shown in Figure 31 has a public cloud version, AWS-Kinesis is cloud version of Kafka, AWS-EMR (Elastic

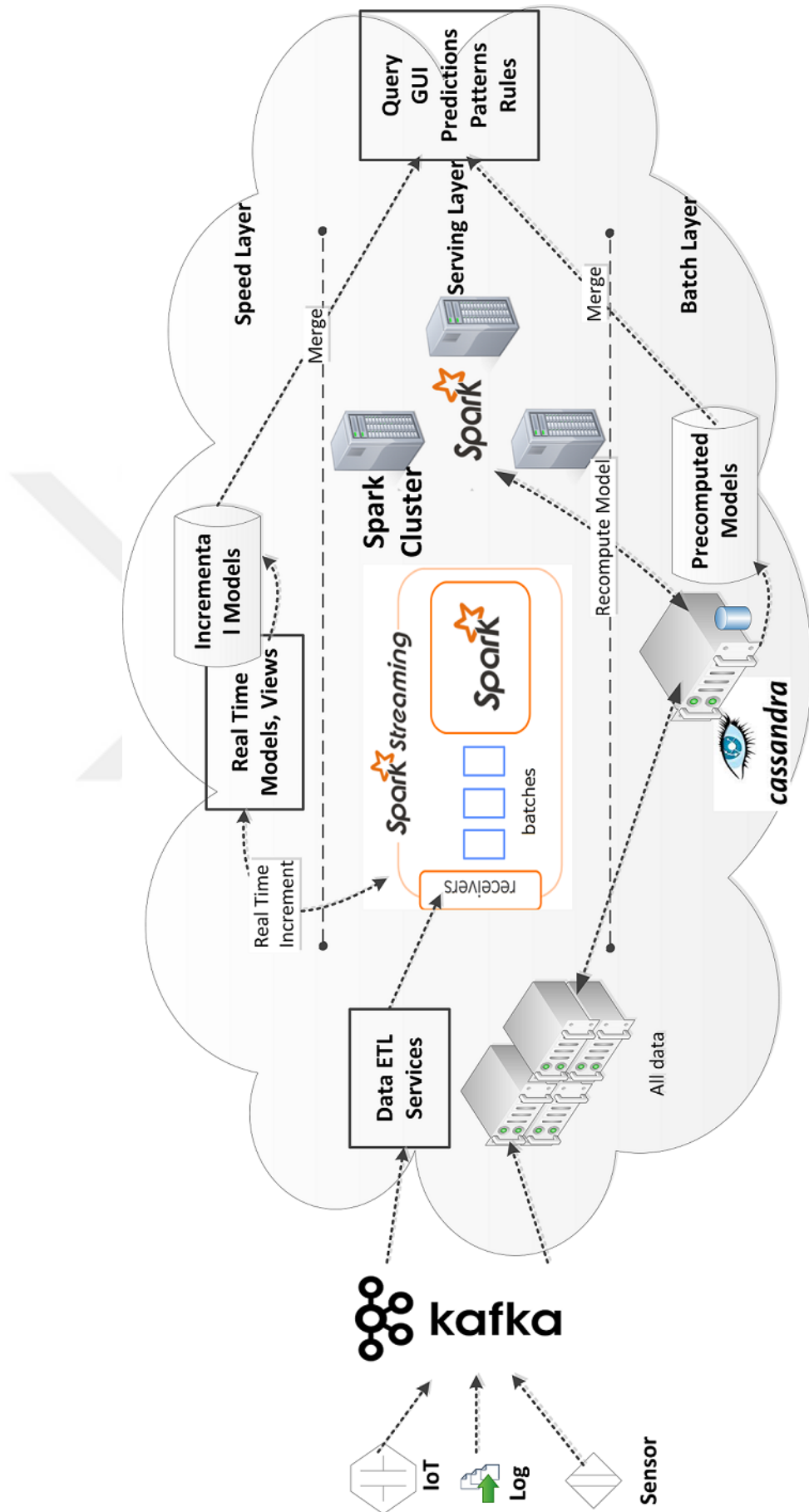


Figure 31: Proposed Lambda architecture for Oil & Gas data analysis.

MapReduce) is data processing framework for MapReduce. AWS-S3 (Simple Storage Service) is for storing batches and log files similar to Cassandra.

5.2.1 Spark Cluster

For high-speed computation, Zaharia et al. [139] designed Spark which improved MapReduce for Interactive Queries and Stream Processing efficiently. Spark cluster design allows several types of workloads such as batch applications, interactive streaming and query processing. There are three different Spark deployments, Standalone, Hadoop Yarn, Spark in MapReduce. In this work, Standalone deployment is used which means Spark is built on top of HDFS and it runs beside MapReduce for the batch jobs in the cluster. Spark components depicted in Figure 32 [135] are explained as follows:

- Spark Core is the general execution engine of spark platform that provides in-memory computation and access to external storage.
- Spark SQL is a component that supports data abstraction for structured and semi-structured data analysis.
- Spark Streaming is for the streaming data analysis and uses Spark Core's scheduling capabilities. It breaks the streaming data to mini-batches for converting to RDD and performing the demanded process.
- MLlib is a distributed and high speed machine learning framework in Spark.
- GraphX provides graph-processing in Spark [135].

5.2.2 Machine Learning Library: MLlib

MLlib [123] is developed as part of the Apache Spark project and is Sparks machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. It provides tools for supervised and unsupervised learning algorithms like classification, clustering, regression and provides feature extraction, selection, and dimension reduction [138]. It also provides constructing and evaluating pipelines for ML and,

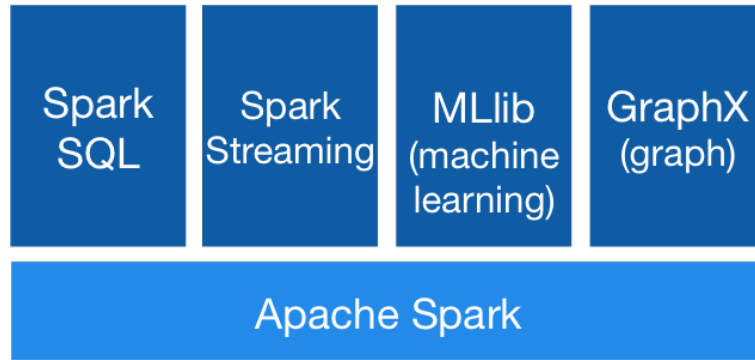


Figure 32: The different components of Spark.

saving, loading of trained models and algorithms. This library utilizes linear algebra and statistics packages and is included in Apache Spark.

5.2.3 Kafka Streaming Platform

Apache Kafka is a distributed platform for stream processing in real-time with three capabilities:

- It provides to publish/subscribe to streaming records, and is similar to messaging queue.
- It provides the possibility of storing the streaming records.
- As records happen, Kafka lets applications to process the streams.

Kafka is used for two classes of applications: (1) building pipelines for real-time streaming data for reliable interaction between systems or applications and; (2) building real-time application for taking action in presence of stream data [140]. Kafka is capable of running on one or more servers called broker and stores records of data streams in specific categories called topics, and each record consists of a key, a value, and a timestamp [141]. Topics are maintained in partitioned logs, distributed in the Kafka cluster where servers handle data and requests to logs. The log partitions are replicated on the cluster for fault tolerance purpose. Kafka has four main application programming interfaces (APIs) for automating tasks shown in Figure 33 [136]:

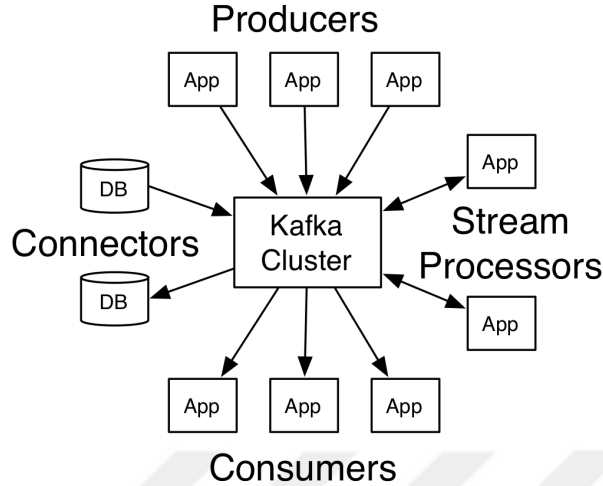


Figure 33: Kafka cluster's core API.

- *The Producer API* allows applications to publish messages to Kafka topics.
- *The Consumer API* sends notification messages to subscribing and matching applications.
- *The Streams API* allows applications to operate as a stream processor, consuming topics and producing output streams and, transform streams.
- *The Connector API* allows applications to run reusable producers/consumers that connect Kafka topics to existing applications [136].

Kafka stores data reliably by replicating logs in distributed servers and provides very low-latency pipelines. Kafka uses Zookeeper [142], which provides coordination of distributed services that maintains configuration information, naming, distributed synchronization, and group services.

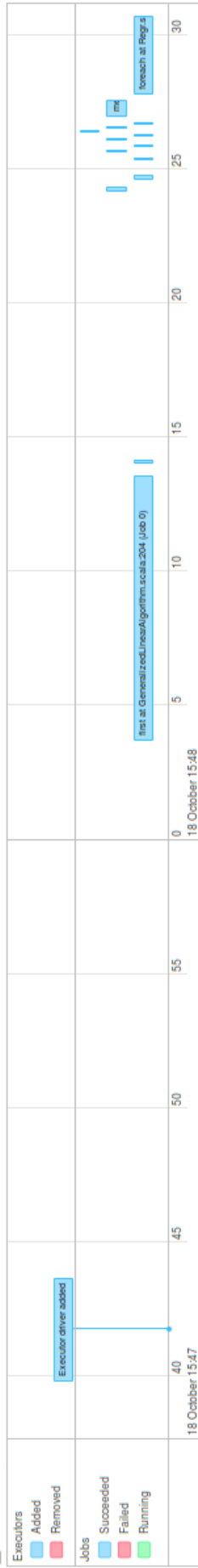
5.3 *Fault Detection and Classification Modules*

The DREDGE approach that is introduced in this thesis, integrates prior information of system to plant model using statistical pattern recognition to detect anomalies and error types in the industrial system. We have used statistical pattern recognition techniques including LSE, ARMA, and DREDGE to extract system properties and

Spark Jobs (?)

User: osboves
 Total Uptime: 1.6 min
 Scheduling Mode: FFO
 Completed Jobs: 14

Event Timeline
 Enable zooming



Completed Jobs (14)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
13	foreach at Regr.scala:48	2017/10/18 15:48:27	3 s	1/1	1/1
12	mean at Regr.scala:45	2017/10/18 15:48:26	0.6 s	1/1	1/1
11	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:26	45 ms	1/1	1/1
10	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:26	72 ms	1/1	1/1
9	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:26	61 ms	1/1	1/1
8	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:26	73 ms	1/1	1/1
7	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:26	82 ms	1/1	1/1
6	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:25	92 ms	1/1	1/1
5	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:25	0.1 s	1/1	1/1
4	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:25	73 ms	1/1	1/1
3	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:24	0.2 s	1/1	1/1
2	treeAggregate at GradientDescent.scala:239	2017/10/18 15:48:24	0.2 s	1/1	1/1
1	count at GradientDescent.scala:209	2017/10/18 15:48:13	0.2 s	1/1	1/1
0	first at GeneralizedLinearAlgorithm.scala:204	2017/10/18 15:48:03	10 s	1/1	1/1

Figure 34: Running fault detection application on Spark cluster.

develop fault detection models over real and synthetically-generated industrial data. Locating which sensor makes what type of error improves overall system reliability for the refinery and improves predictive maintenance performance [143]. Modeling the time-varying refinery plants, the anomalies can be detected with statistical hypothesis tests (Chi-squared test) for fault detection and noise removal.

In this section, again the ARMA model integrated to KF-based tracking model for online system identification and GED is used, over streaming sensor data. The selected techniques are based on the performance and accuracy evaluations in chapter 3. As mentioned in this chapter Apache Spark is an in-memory distributed alternative to MapReduce, which also boosted the ML algorithms to be executed with 100x performance. The Lambda architecture [131] that is proposed in this chapter is used for unified stream data processing and analytics for Oil & Gas industry. These techniques are implemented in Scala for “stream mining” in real-time and to be used in the proposed private cloud. The data stream into the applications that are running on top of Spark cluster through Kafka topics and the application subscribes to these topics for processing the sensor data for fault detection. Running these applications on Spark cluster are much faster than local machines. The jobs are distributed on the cluster with 1 master and 8 working nodes. Once this application is started, the master will print out a `spark://HOST:PORT` URL for itself, which can be used to observe workers performance via the masters web UI. In Figure 34 the execution of this application is depicted, which shows the successfully completed jobs on Spark cluster.

5.4 Public Cloud Implementation

The best definition for public cloud presented from NIST is: “The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination

of them. It exists on the premises of the cloud provider” [124]. Public cloud’s main properties are that it is a service allowing users to access the cloud by using their web browsers, by a payment policy of *pay-per-use*. Only amount of time a user spends for using a service is calculated for payment. This concept helps users reducing the costs of their IT expenditure by taking benefits of an almost infinite computing resources and pay only the amount used. The main disadvantage of public clouds are lower security and being prone to malicious attacks. Vendors and clients can implement security checks on both sides for validation and boundaries of operations [144].

5.4.1 Spark on Amazon EMR

Apache Spark is an open-source, distributed processing system used for big data processing. Using this service we can easily create Apache Spark clusters and use fast connectivity to other AWS services such as AWS-Kinesis for stream data processing and AWS-S3 [145].

Streaming data analytics is becoming more popular in large industries as the technology has become more user-friendly to implement. Spark-Streaming connected to an Amazon Kinesis stream processing is a typical model for real-time analytics that is available and easy to use. Combining Amazon Kinesis, Spark on Amazon EMR, and S3 together provides all the requirement for real-time stream processing. Amazon EMR clusters can read and process Amazon Kinesis streams directly, using familiar tools in the Hadoop such as Hive, Pig, MapReduce, the Hadoop Streaming API. We can join real-time data from Amazon Kinesis with existing data on Amazon S3, Amazon DynamoDB, and HDFS in a running cluster and directly load the data from Amazon EMR to Amazon S3 or DynamoDB for post-processing activities as well. A sample architecture is depicted in Figure 35 which are equivalent public services introduced for private cloud versions in the previous section.

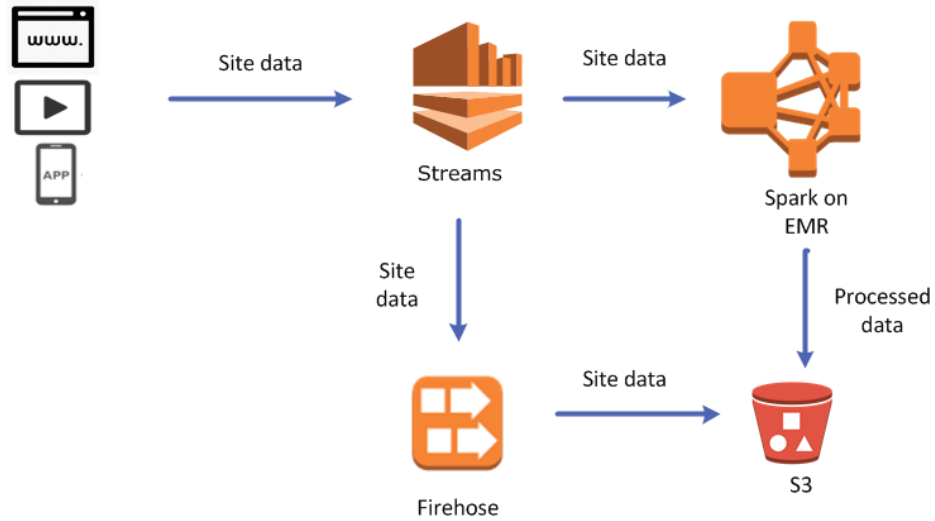


Figure 35: AWS architecture for stream processing.

5.4.2 AWS-Kinesis and AWS-S3

Amazon Kinesis provides collecting, processing, and analyzing streaming data in real-time. It offers cost-effective processing capabilities at scale with the flexibility to select tools according to requirements for applications. AWS Kinesis digests real-time data for analysis and machine learning processes. The benefits of using this service are processing streaming data in real-time, minimum infrastructure management, and handle the massive amount of data with low latency.

Amazon S3 is a storage object to store and access any amount of data from anywhere from different applications. It provides flexible data management and query-in-place functionality on data. In Figure 36 a sample execution of the distributed job on AWS-EMR is depicted, where the cluster consists of 1 master and 2 worker nodes.

5.5 Summary and Conclusion

Oil & Gas industry demands effective methods to complete its Industry 4.0 digital transformation. Open-source big data tools and their cloud counterparts provide us with distributed processing capabilities. Our proposed Lambda architecture utilizes

eu-west-1.console.aws.amazon.com/elasticmapreduce/home?region=eu-west-1#cluster-detailsj-8J25VP... ☆

Resource Groups Athar Khodabakhsh

Clone Terminate AWS CLI export

Cluster: newcluster Terminated Terminated by user request

Summary Application history Monitoring Hardware Events Steps Configurations Bootstrap actions

Add task instance group

Instance groups

Filter: Filter instance groups ... 2 instance groups (all loaded) ↻

ID	Status	Node type & name	Instance type
ig-2U52AKIWL8F6H	Terminated (2 Requested)	CORE Core Instance Group	m3.xlarge 8 vCPU, 15 GiB memory, 80 SSD GB storage EBS Storage: none
ig-2Q1WTJUPRP9KY	Terminated (1 Requested)	MASTER Master Instance Group	m3.xlarge 8 vCPU, 15 GiB memory, 80 SSD GB storage EBS Storage: none

(a)

Resource Groups Athar Khodabakhsh Ireland Support

Cluster: newcluster Terminating Terminated by user request

Summary Application history Monitoring Hardware Events Steps Configurations Bootstrap actions

YARN applications > application_1508491553427_0001 (Spark) ↻

Jobs Stages Executors

Jobs > Job 0

Status: Succeeded
Completed stages: 2

Stages (2)

Filter: Filter stages ... 2 stages (all loaded) ↻

Stage ID	Status	Description	Submitted (UTC+3)	Duration	Tasks succeeded / total
1	Complete	saveAsTextFile at NativeMethodAccessorImpl.java:0	2017-10-20 12:35 (UTC+3)	2 s	32 / 32
0	Complete	reduceByKey at wc.py:10	2017-10-20 12:35 (UTC+3)	6 s	2 / 2

(b)

Figure 36: Running application on AWS-EMR.

open-source software to solve Big Data problems for building analytical models using machine learning techniques.



CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this thesis, problems associated with erroneous sensor readings in oil refineries and we proposed a real-time data validation, gross error detection (GED) and classification (GEC) service are addressed, leveraging tools from statistics, signal processing, data mining and a CEP engine integrated with cyber-physical systems. For comparison of GED and GEC accuracies as well as computational performances, time-series data from the power and petrochemical plants of an oil refinery are obtained. We found that our proposed DREDGE method has accurate error detection (99.1-100%) and sustainable performance at smaller window sizes. IMB method had lower accuracy results and its performance degraded exponentially with increasing window size. After comparison of three gross error classification (GEC) methods, Complex Decision Tree (CDT) is found to have the highest precision and recall values (95.8-100%), where KNN had the lowest recall values (e.g. 82.1%) that would be unfit for oil refineries and their safety requirements. Therefore, CDT technique is implemented into CEP engine for real-time GEC.

The proposed approach combines data cleaning via gross error detection, steady-state system modeling, real-time operational mode identification and model updates, all at once. This study is applicable to many data-driven systems dealing with sensor streams to ensure data quality and improve system modeling. Devices in interconnected CPS can communicate for more accurate system monitoring and avoid total failure by predicting and detecting abnormal behavior of the system. We believe that our study has the potential to bridge the gap between generic big data software available in the market and real challenges faced by oil refineries: detecting erroneous

data and sensors with high accuracy (no false positives or negatives) over high volume stream data. Error detection and classification such as ours provide a data quality improvement for better system modeling and isolation of faulty processes.

To identify a system’s operational mode, regression analysis is applied on a time-varying industrial system. It was shown that by evaluating the fluctuation of error values mode changes are detectable. The results were also verified by real-time K -means clustering. This approach provides information about steady-state, drifts and transient states of time-varying industrial systems and the requirement for real-time model updates. An outlier detection algorithm using windows-based DBSCAN clustering was also used as a novel strategy for multi-state operational mode identification. Finally, the proposed TCP congestion control-based adaptive window size tuning for streaming regression analysis resulted in reductions in RMSE values as well as saving modeling computational costs of frequent updates. We believe that DREDGE service has the potential to bridge the gap between generic big data software available in the market and the challenges faced by oil refineries: detecting erroneous data and sensors with high accuracy (no false positives or negatives) and speed.

In future work, integration of models from our system into the real refinery, techniques from deep learning [146] will be studied and the proposed architecture will be applied to other production plants. In recent studies on complex and dynamic patterns that appear in time-series, Recurrent Neural Network (RNN) which are building blocks of Long Short-Term Memory (LSTM) structure is used for capturing long-term dependencies/correlations of data [147]. In our future work, this structure will also be tested for prediction and classification of multivariate time-series.

Bibliography

- [1] R. K. Perrons and J. W. Jensen, “Data as an asset: What the oil and gas sector can learn from other industries about big data,” *Energy Policy*, vol. 81, pp. 117–121, 2015.
- [2] E. Lughofer and M. Sayed-Mouchaweh, “Autonomous data stream clustering implementing split-and-merge concepts—towards a plug-and-play approach,” *Information Sciences*, vol. 304, pp. 54–79, 2015.
- [3] B. Andres, R. Sanchis, and R. Poler, “A cloud platform to support collaboration in supply networks,” *International Journal of Production Management and Engineering*, vol. 4, no. 1, pp. 5–13, 2016.
- [4] M. Bakir, B. Aydogan, M. Aydin, A. Khodabakhsh, I. Ari, and A. O. Ercan, “Real-time data reconciliation solutions for big data problems observed in oil refineries,” *In Signal Processing and Communications Applications Conference (SIU)*, vol. 22, pp. 1612–1615, IEEE, 2014.
- [5] C. Harrison, B. Eckman, R. Hamilton, and et al., “Foundations for smarter cities,” *IBM Journal of Research and Development*, vol. 54, no. 4, pp. 1–16, 2010.
- [6] A. Khodabakhsh, I. Ari, and M. Bakir, “Cloud-based fault detection and classification for oil & gas industry,” *SIAM International Conference on Data Mining, In DM4OG 2017 Workshop on Data Mining for Oil & Gas, arXiv:1705.04583*, 2017.
- [7] “TÜPRAŞ refinery information,” <http://tupras.com.tr/en/rafineries>, Accessed On: 20/11/2018.
- [8] S. Narasimhan and C. Jordache, *Data Reconciliation & Gross Error Detection, An Intelligent Use of Process Data*. Gulf Professional Publishing, 2000.
- [9] M. J. Bagajewicz and Q. Jiang, “Gross error modeling and detection in plant linear dynamic reconciliation,” *Computers & Chemical Engineering*, vol. 22, no. 12, pp. 1789–1809, 1998.
- [10] “AWS: Amazon Web Services,” <http://aws.amazon.com>.
- [11] M. Ferguson, “Architecting a big data platform for analytics,” *A Whitepaper Prepared for IBM*, 2012.
- [12] P. C. Evans and M. Annunziata, “Industrial internet: Pushing the boundaries of minds and machines,” *General Electric Reports*, 2012.

- [13] N. Bessis and C. Dobre, *Big data and internet of things: a roadmap for smart environments*, vol. 546. Basel, Switzerland: Springer International Publishing, 2014.
- [14] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and cloud computing: from parallel processing to the internet of things*. Morgan Kaufmann, 2013.
- [15] M. Diaz, C. Martin, and B. Rubio, “State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing,” *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, 2016.
- [16] B. Buntz, “The top 20 industrial IoT applications,” <http://www.ioti.com/industrial-iot-iiot/top-20-industrial-iot-applications>, Accessed On: 20/11/2018.
- [17] G. Dudek, “Pattern-based local linear regression models for short-term load forecasting,” *Electric Power Systems Research*, vol. 130, pp. 139–147, 2016.
- [18] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, “Log-based predictive maintenance,” in *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1867–1876, ACM, 2014.
- [19] B. Kroll, D. Schaffranek, S. Schriegel, and O. Niggemann, “System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants,” in *Emerging Technology and Factory Automation (ETFA)*, pp. 1–7, IEEE, 2014.
- [20] C. H. Nadungodage, Y. Xia, F. Li, J. J. Lee, and J. Ge, “Streamfitter: a real time linear regression analysis system for continuous data streams,” in *International Conference on Database Systems for Advanced Applications*, pp. 458–461, Springer, 2011.
- [21] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, “Ensemble learning for data stream analysis: a survey,” *Information Fusion*, vol. 37, pp. 132–156, 2017.
- [22] “Foghorn Industrial IoT use cases,” <https://www.foghorn.io/industries/>, Accessed On: 20/11/2018.
- [23] N. Askham and et al., “The six primary dimensions for data quality assessment,” tech. rep., DAMA UK Working Group, 2013.
- [24] C. Dobre and F. Xhafa, “Intelligent services for big data science,” *Future Generation Computer Systems*, vol. 37, pp. 267–281, 2014.
- [25] D. F. Ferguson and E. Hadar, “Optimizing the it business supply chain utilizing cloud computing,” pp. 1–6, IEEE, 2011.
- [26] J. Gertler, *Fault Detection and Diagnosis*. Springer London, 2015.

- [27] Z. Zhang and J. Chen, “Simultaneous data reconciliation and gross error detection for dynamic systems using particle filter and measurement test,” *Computers & Chemical Engineering*, vol. 69, pp. 66–74, 2014.
- [28] G. V. Reklaitis and D. R. Schneider, *Introduction to material and energy balances*. New York: Wiley, 1983.
- [29] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 1, pp. 35–45, 1960.
- [30] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1990.
- [31] S. Ao, *Applied time series analysis and innovative computing*, vol. 59. Springer Science & Business Media, 2010.
- [32] G. E. P. Box, G. M. Jenkins, and et al., *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [33] P. S. Kalekar, “Time series forecasting using holt-winters exponential smoothing,” in *Kanwal Rekhi school of information Technology*, 2004.
- [34] S. Gelper, R. Fried, and C. Croux, “Robust forecasting with exponential and holt-winters smoothing,” *Journal of Forecasting*, vol. 29, no. 3, pp. 285–300, 2010.
- [35] E. Alpaydin, *Introduction to machine learning*. Second Edition, Cambridge, Massachusetts London, England, MIT press, 2010.
- [36] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson International Edition, 2005.
- [37] M. H. Kutner, C. Nachtsheim, and J. Neter, *Applied linear regression models*. McGraw-Hill/Irwin, 2004.
- [38] C. C. Aggarwal, *Managing and mining sensor data*. Springer Science & Business Media, 2013.
- [39] R. B. Darlington and A. F. Hayes, *Regression analysis and linear models: Concepts, applications, and implementation*. Guilford Publications, 2016.
- [40] G. De’Ath, “Multivariate regression trees: a new technique for modeling species-environment relationships,” *Ecology*, vol. 83, pp. 1105–17, Apr. 2002.
- [41] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study,” *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

- [42] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “Dbscan revisited, revisited: Why and how you should (still) use dbscan,” *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, p. 19, 2017.
- [43] M. V. Shcherbakov, A. Brebels, and et al., “Outlier detection and classification in sensor data streams for proactive decision support systems,” *Journal of Physics: Conference Series*, vol. 803, no. 1, 2017, IOP Publishing.
- [44] D. Jankov, S. Sikdar, R. Mukherjee, K. Teymourian, and C. Jermaine, “Real-time high performance anomaly detection over data streams: Grand challenge,” in *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, pp. 292–297, ACM, 2017.
- [45] A. Varga, *Solving Fault Diagnosis Problems*. Studies in Systems, Decision and Control. Springer International Publishing, 2017.
- [46] B. Tang and H. He, “A local density-based approach for outlier detection,” *Neurocomputing*, vol. 241, pp. 171–80, 2017.
- [47] C. Guarnaccia, L. Elia, J. Quartieri, and C. Tepedino, “Time series analysis techniques applied to transportation noise,” in *Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), 2017 IEEE International Conference on*, pp. 1–6, IEEE, 2017.
- [48] B. Babcock, S. Babu, M. Datar, and et al, “Models and issues in data stream systems,” in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1–6, 2002.
- [49] G. Kreml, I. Zliobaite, and et al, “Open challenges for data stream mining research,” *ACM SIGKDD explorations newsletter*, vol. 16, no. 1, pp. 1–10, 2014.
- [50] M. Cruz, M. Bender, and H. Ombao, “A robust interrupted time series model for analyzing complex health care intervention data,” *Statistics in Medicine*, 2017.
- [51] G. Reikard, S. Haupt, and T. Jensen, “Forecasting ground-level irradiance over short horizons: Time series, meteorological, and time-varying parameter models,” *Renewable Energy*, no. 112, pp. 474–85, 2017.
- [52] A. Vasebi, E. Poulin, and D. Hodouin, “Dynamic data reconciliation in mineral and metallurgical plants,” *IBM Journal of Research and Development*, vol. 36, no. 2, pp. 235–243, 2012.
- [53] A. Rafiee and F. Behrouzshad, “Data reconciliation with application to a natural gas processing plant,” *Journal of Natural Gas Science and Engineering*, vol. 31, pp. 538–545, 2016.

- [54] S. Yin, X. Li, H. Gao, and O. Kaynak, “Data-based techniques focused on modern industry: An overview,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–67, 2015.
- [55] “Apache Hadoop Project,” <http://hadoop.apache.org>.
- [56] “EsperTech inc. Event Stream Intelligence,” <http://www.espertech.com/>.
- [57] C.-L. Yang, H. Sutrisno, Lo, and et al., “Streaming data analysis framework for cyber-physical system of metal machining processes,” in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 546–551, IEEE, 2018.
- [58] S. Ramirez-Gallego, B. Krawczyk, and et al., “A survey on data preprocessing for data stream mining: current status and future directions,” *Neurocomputing*, vol. 239, pp. 39–57, 2017.
- [59] E. Ölmezogullari and I. Ari, “Online association rule mining over fast data,” in *Big Data (BigData Congress), 2013 IEEE International Congress on*, pp. 110–117, IEEE, 2013.
- [60] E. C. do Valle, R. de Araújo Kalid, A. R. Secchi, and A. Kiperstok, “Collection of benchmark test problems for data reconciliation and gross error detection and identification,” *Computers & Chemical Engineering*, vol. 111, pp. 134–148, 2018.
- [61] Z. Zhang, Y. Y. Chuang, and J. Chen, “Methodology of data reconciliation and parameter estimation for process systems with multi-operating conditions,” *Chemometrics and Intelligent Laboratory Systems*, vol. 137, pp. 110–119, 2014.
- [62] S. Guo, P. Liu, and Z. Li, “Data reconciliation for the overall thermal system of a steam turbine power plant,” *Applied Energy*, vol. 165, pp. 1037–1051, 2016.
- [63] G. Ruan, P. C. Hanson, and et al., “Mining lake time series using symbolic representation,” *Ecological Informatics*, vol. 39, pp. 10–22, 2017.
- [64] X. Jiang, P. Liu, and Z. Li, “Data reconciliation and gross error detection for operational data in power plants,” *Energy*, vol. 75, pp. 14–23, 2014.
- [65] A. Alenany, H. Shang, M. Soliman, and I. Ziedan, “Improved subspace identification with prior information using constrained least squares,” *IET control theory & applications*, vol. 5, no. 13, pp. 1568–1576, 2011.
- [66] W. E. Larimore, “Maximum likelihood subspace identification for linear, non-linear, and closed-loop systems,” *American Control Conference, Proceedings of*, pp. 2305–2319, June IEEE, 2005.
- [67] M. Buchholz and W. E. Larimore, “Subspace identification of an aircraft linear parameter-varying flutter model,” *American Control Conference (ACC)*, pp. 2263–2267, IEEE, 2013.

- [68] T. McWhorter and M. C. M., “Robust subspace estimation using prior information,” *Signals, Systems & Computers, Conference Record of the Thirty-Second Asilomar Conference on*, vol. 2, pp. 1354–1358, IEEE, 1998.
- [69] Y. Zhu and D. Shasha, “Statstream: Statistical monitoring of thousands of data streams in real time,” in *Proceedings of the 28th international conference on Very Large Data Bases*, vol. 28, pp. 358–369, 2002.
- [70] W. Zhang, M. Hirzel, and D. Grove, “Aqua: adaptive quality analytics,” in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pp. 169–180, ACM, 2016.
- [71] E. Lughofer, M. Pratama, and I. Skrjanc, “Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation,” *IEEE Transactions on Fuzzy Systems*, 2017.
- [72] Z. Zhu, G. Geng, and Q. Jiang, “Multi-scenario parameter estimation for synchronous generation systems,” *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 1851–1859, 2017.
- [73] A. Shaker and E. Lughofer, “Self-adaptive and local strategies for a smooth treatment of drifts in data streams,” *Evolving Systems*, vol. 5, no. 4, pp. 239–257, 2014.
- [74] “Integrated Operations and the Oil & Gas Ontology,” *POSC Caesar Association (PCA), Norwegian Offshore Industry*.
- [75] C. C. Aggarwal and C. Charu, “Data streams: models and algorithms,” *Springer Science & Business Media*, vol. 31, 2007.
- [76] T. Sanislav, G. Mois, and L. Miclea, “An approach to model dependability of cyber-physical systems,” *Microprocessors and Microsystems*, vol. 41, pp. 67–76, 2016.
- [77] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, “The anatomy of big data computing,” *Software: Practice and Experience*, vol. 46, no. 1, pp. 79–105, 2016.
- [78] K. Kambatla and et al, “Trends in big data analytics,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, 2014.
- [79] D. E. O’Leary, “Artificial intelligence and big data,” *IEEE Intelligent Systems*, vol. 28, no. 2, pp. 96–99, 2013.
- [80] J. A. Miller, C. Bowman, V. G. Harish, and S. Quinn, “Open source big data analytics frameworks written in scala,” in *Big Data (BigData Congress), 2016 IEEE International Congress on*, pp. 389–393, IEEE, 2016.

- [81] J. Fu, J. Sun, and K. Wang, “Spark—a big data processing platform for machine learning,” in *Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), 2016 International Conference on*, pp. 48–51, IEEE, 2016.
- [82] A. N. Richter, T. M. Khoshgoftaar, S. Landset, and T. Hasanin, “A multi-dimensional comparison of toolkits for machine learning with big data,” in *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*, pp. 1–8, IEEE, 2015.
- [83] N. Cai, S. Wei, and et al., “A survey on stream distributed computing,” in *Intelligent Networks and Intelligent Systems (ICINIS), 2015 8th International Conference on*, pp. 94–97, IEEE, 2015.
- [84] M. D. Mauro and C. D. Sarno, “A framework for internet data real-time processing: A machine-learning approach,” *Security Technology (ICCST), International Carnahan Conference on*, pp. 1–6, IEEE, 2014.
- [85] L. Ramaswamy, V. Lawson, and S. V. Gogineni, “Towards a quality-centric big data architecture for federated sensor services,” in *Big Data (BigData Congress), 2013 IEEE International Congress on*, pp. 86–93, IEEE, 2013.
- [86] R. Y. Wang, H. B. Kon, and S. E. Madnick, “Data quality requirements analysis and modeling,” in *Data Engineering, 1993. Proceedings. Ninth International Conference on*, pp. 670–677, IEEE, 1993.
- [87] R. Kavitha and et al., “Cloud computing integrated with testing to ensure quality,” *Journal of Scientific & Industrial Research*, vol. 75, pp. 77–81, 2016.
- [88] “Apache Storm,” <http://storm.apache.org>.
- [89] “Apache Spark,” <http://spark.apache.org>.
- [90] S. Perera and S. Suhothayan, “Solution patterns for realtime streaming analytics,” in *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, pp. 247–255, 2015.
- [91] B. Gaunitz, M. Roth, and B. Franczyk, “Dynamic and scalable real-time analytics in logistics combining apache storm with complex event processing for enabling new business models in logistics,” in *Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on*, pp. 289–294, IEEE, 2015.
- [92] I. Santos, M. Tilly, B. C. B, and J. Goldstein, “DiAl: Distributed streaming analytics anywhere, anytime,” in *Proceedings of the VLDB Endowment*, vol. 6, pp. 1386–1389, 2013.
- [93] S. Fiore, A. DANca, and et al., “Ophidia: toward big data analytics for escience,” *Procedia Computer Science*, vol. 18, pp. 2376–2385, 2013.

- [94] J. Lee, B. Bagheri, and H. Kao, “Cyber-physical systems architecture for industry 4.0-based manufacturing systems,” *Manufacturing Letters*, pp. 18–23, 2015.
- [95] R. R. Rajkumar, I. LEE, and et al., “Cyber-physical systems: the next computing revolution,” in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pp. 731–736, IEEE, 2010.
- [96] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0,” *IEEE Industrial Electronics Magazine*, pp. 17–27, 2017.
- [97] J. Wan, H. Cai, and K. Zhou, “Industrie 4.0: enabling technologies,” in *Intelligent Computing and Internet of Things (ICIT), 2014 International Conference on*, pp. 135–140, IEEE, 2015.
- [98] “TÜPRAŞ, Boiler Data,” <https://www.openml.org/d/41170>, Accessed On: 20/11/2018.
- [99] L. Zhang and X. Peng, “Time series estimation of gas sensor baseline drift using arma and kalman based models,” *Sensor Review*, vol. 36, no. 1, pp. 34–39, 2016.
- [100] A. Khodabakhsh, I. Ari, M. Bakir, and A. O. Ercan, “Multivariate sensor data analysis for oil refineries and multi-mode identification of system behavior in real-time,” *IEEE Access*, vol. 6, pp. 64389–64405, 2018.
- [101] D. G. Luenberger, *Introduction to dynamic systems: theory, models, and applications*, vol. 1. Wiley New York, 1979.
- [102] V. Vaidehi and C. N. Krishnan, “Computational complexity of the kalman tracking algorithm,” *IETE journal of research*, vol. 44, pp. 125–134, 1998.
- [103] J. D. Kelly and J. D. Hedengren, “A steady-state detection (ssd) algorithm to detect non-stationary drifts in processes,” *Journal of Process Control*, vol. 23, no. 3, pp. 326–331, 2013.
- [104] D. Dochain, “State and parameter estimation in chemical and biochemical processes: a tutorial,” *Journal of process control*, vol. 13, no. 8, pp. 801–818, 2003.
- [105] Z. Zhang, Y.-Y. Chuang, and J. Chen, “Methodology of data reconciliation and parameter estimation for process systems with multi-operating conditions,” *Chemometrics and Intelligent Laboratory Systems*, vol. 137, pp. 110–119, 2014.
- [106] Z. Zhu, G. Geng, and Q. Jiang, “Multi-scenario parameter estimation for synchronous generation systems,” *IEEE Transactions on Power Systems*, vol. 32, pp. 1851–1859, 2017.
- [107] X. Jiang, P. Liu, and Z. Li, “Data reconciliation and gross error detection for operational data in power plants,” *Energy*, vol. 75, pp. 14–23, 2014.

- [108] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD*, vol. 96, pp. 226–231, 1996.
- [109] D. Hodouin, “Process observers and data reconciliation using mass and energy balance equations,” in *Advanced control and supervision of mineral processing plants*, pp. 15–83, Springer, 2010.
- [110] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.
- [111] A. Khodabakhsh, I. Ari, M. Bakir, and S. M. Alagoz, “Stream analytics and adaptive windows for operational mode identification of time-varying industrial systems,” in *2018 IEEE International Congress on Big Data (BigData Congress)*, pp. 242–246, IEEE, 2018.
- [112] A. Akbar, G. Kousiouris, H. Pervaiz, J. Sancho, P. Ta-Shma, F. Carrez, and K. Moessner, “Real-time probabilistic data fusion for large-scale iot applications,” *IEEE Access*, vol. 6, pp. 10015–10027, 2018.
- [113] I. Ari and Ö. F. Çelebi, “Finding event correlations in federated wireless sensor networks,” in *2011 7th International Wireless Communications and Mobile Computing Conference*, pp. 2052–2057, IEEE, 2011.
- [114] M. Allman, V. Paxson, and E. Blanton, “TCP congestion control,” tech. rep., 2009.
- [115] I. Ari, E. Olmezogullari, and Ö. F. Çelebi, “Data stream analytics and mining in the cloud,” in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pp. 857–862, IEEE, 2012.
- [116] I. A. Hashem, I. Yaqoob, N. B. Anuar, S. M. S, A. Gani, and S. U. Khan, “The rise of “big data” on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98–115, 2015.
- [117] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [118] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pp. 1–10, Ieee, 2010.
- [119] C. Maltzahn, E. Molina-Estolano, A. Khurana, A. J. Nelson, S. A. Brandt, and S. Weili, “Ceph as a scalable alternative to the hadoop distributed file system,” *The USENIX Magazine*, pp. 38–49, 2010.
- [120] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2012.

- [121] “Apache Hbase,” <https://hbase.apache.org/>.
- [122] “Apache Hive,” <https://hive.apache.org/>.
- [123] “MLlib: Machine Learning Library on Spark,” <http://spark.apache.org/mllib>.
- [124] P. Mel and T. Grance, “The nist definition of cloud computing,” <http://www.nist.gov>, *Accessed On: 20/11/2018*, 2011.
- [125] “Microsoft Azure,” <https://azure.microsoft.com/en-us/>.
- [126] “Google Cloud,” <https://cloud.google.com/>.
- [127] “IBM Bluemix,” <https://console.bluemix.net/>.
- [128] “Salesforce,” <https://www.salesforce.com/eu/>.
- [129] H. Qi and A. Gani, “Research on mobile cloud computing: Review, trend and perspectives,” in *Digital Information and Communication Technology and it’s Applications (DICTAP), 2012 Second International Conference on*, pp. 195–202, IEEE, 2012.
- [130] M. Zaharia and et al., “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 2–2, USENIX Association, 2012.
- [131] M. Kiran, P. Murphy, and et al., “Lambda architecture for cost-effective batch and speed big data processing,” in *Big Data (Big Data), 2015 IEEE International Conference on*, pp. 2785–2792, IEEE, 2015.
- [132] T. Dillon, C. Wu, and E. Chang, “Cloud computing: issues and challenges,” in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 27–33, IEEE, 2010.
- [133] “Apache Ignite,” <http://ignite.apache.org>.
- [134] Z. Han and Y. Zhang, “Spark: A big data processing platform based on memory computing,” in *Parallel Architectures, Algorithms and Programming (PAAP), Seventh International Symposium on*, pp. 172–176, IEEE, 2015.
- [135] “Apache Spark - introduction,” https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm, *Accessed On: 20/11/2018*.
- [136] “Apache Kafka,” <http://kafka.apache.org>.
- [137] “Apache Cassandra,” <http://cassandra.apache.org>.

- [138] X. Meng and et al, “Mllib: Machine learning in apache spark,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.
- [139] M. Zaharia, M. Chowdhury, M. J. Franklin, S. S. S, and I. Stoica, “Spark: Cluster computing with working sets,” *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.
- [140] M. Stonebraker, U. Cetintemel, and S. Zdonik, “The 8 requirements of real-time stream processing,” *ACM Sigmod Record*, vol. 34, no. 4, pp. 42–47, 2005.
- [141] J. Kreps, N. Narkhede, J. Rao, *et al.*, “Kafka: A distributed messaging system for log processing,” in *Proceedings of the NetDB*, pp. 1–7, 2011.
- [142] “Apache Zookeeper,” <https://zookeeper.apache.org/>.
- [143] R. K. Perrons and J. W. Jensen, “The big deal about big data in upstream oil and gas,” *White Paper, IDC Energy Insights*, 2012.
- [144] Y. Jadeja and K. Modi, “Cloud computing-concepts, architecture and challenges,” in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pp. 877–880, IEEE, 2012.
- [145] J. Varia and S. Mathew, “Overview of amazon web services,” *Amazon Web Services*, 2014.
- [146] M. Langkvist, L. Karlsson, and A. Loutfi, “A review of unsupervised feature learning and deep learning for time-series modeling,” *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.
- [147] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

VITA

Athar Khodabakhsh received the B.Sc. degree in software engineering from the Computer and Electrical Engineering Department, Islamic Azad University, Zanjan, Iran, in 2005. She pursued the Ph.D. degree with the Computer Science Department, Özyeğin University, Istanbul, Turkey between February 2013 - February 2019. She was also a Teaching and Research Assistant at Özyeğin University. Her fields of interest include data science, data mining, cloud computing, and distributed systems.

Academic Employment

- Lecturer, Department of Engineering, Özyeğin University, summer 2018. Teaching: Introduction to Programming.
- Graduate Teaching Assistant, Department of Computer Science, Özyeğin University, February 2013 - January 2019. Responsibilities include: assisting professors with grading, proctoring exams, and lab sessions.
- Research Assistant to Professor Dr. Ismail Ari, Department of Computer Science, Özyeğin University, February 2013 - January 2019.

Publications

- Athar Khodabakhsh, Ismail Ari, Mustafa Bakir, Ali Ozer Ercan, “Multivariate Sensor Data Analysis for Oil Refineries and Multi-mode Identification of System Behavior in Real-time”, *IEEE ACCESS*, vol. 6, pp. 64389-64405, 2018. DOI: 10.1109/ACCESS.2018.2877097
- Athar Khodabakhsh, Ismail Ari, Mustafa Bakir, Serhat Murat Alagoz, “Stream

Analytics and Adaptive Windows for Operational Mode Identification of Time-Varying Industrial Systems,” *IEEE International Congress on Big Data (Big-Data Congress)*, San Francisco, CA, USA, pp. 242-246, 2018. DOI:10.1109/BigDataCongress.2018.00042

- Athar Khodabakhsh, Ismail Ari, Mustafa Bakir, “Cloud-based Fault Detection and Classification for Oil & Gas Industry,” *SIAM International Conference on Data Mining, In DM4OG 2017 Workshop on Data Mining for Oil and Gas*, Houston, TX, USA, pp. 14-19, April 2017.
- Mustafa Bakir, Mehmet Aydin, Burak Aydogan, Athar Khodabakhsh, Ismail Ari, Ali Ozer Ercan, “Real-time data reconciliation solutions for big data problems observed in oil refineries”, *22nd Signal Processing and Communications Applications Conference (SIU)*, Trabzon, pp. 1612-1615, IEEE, 2014. DOI: 10.1109/SIU.2014.6830553
- Mustafa Bakir, Mehmet Aydin, Burak Aydogan, Ismail Ari, Ali Ozer Ercan, Athar Khodabakhsh. “A Real-Time Data Verification and Data Reconciliation System for Petroleum Refineries.” WO/2016/114736.

Academic Awards

- Best Teaching Assistant Award from Graduate School of Engineering for the 2016–2017 academic year, Özyeğin University, Istanbul, Turkey.
- Ph.D. Student Travel Award SDM17, SIAM International Conference on Data Mining, Houston, Texas, USA, April 2017.

Professional Membership

- IEEE Student member since 2015.