Stream Processing System for Product Quality Prediction using Sensor data

ASLAM Muhammad Najeeb

Delft University of Technology

Stream Processing System for Product Quality Prediction using Sensor data

Master's Thesis in Computer Science

Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

ASLAM, Muhammad Najeeb

16th October 2017

Author
Muhammad Najeeb Aslam

Title
Stream Processing System for Product Quality Prediction using Sensor data

MSc presentation
23 October, 2017

Graduation Committee

prof. dr. D. H. J. Epema	Delft University of Technology
dr. C. Hauff	Delft University of Technology
mr. S. Beniers (C.E.O)	Bright Cape (Small and Big data Solutions)

Abstract

Predicting product qualities using a stream of industrial sensor data emitted directly from the assembly lines is a complex data mining problem. Most of the traditional machine learning models take much time during the learning phase, which becomes a bottleneck in real time quality prediction. This problem arises more frequently for the manufacturing industry, primarily when the relationship between the real-time input data and the training data cannot be defined precisely. In this master thesis, we have investigated the possibility to use a machine learning approach for streaming data in an efficient way. We have designed a strategy to interconnect the training and the detection phases of prediction model into two separate layers. We have devised a strategy to filter out most of the meaningless dimensions from intermixed input sensor data, which makes the training phase much faster in our approach. We also have designed an architecture of data processing tool to develop quality prediction framework that uses inputs from multiple sources and capable of applying machine learning models dynamically in a batch layer to generates a prediction model. Further, this model is applied to the moving data to identify faulty products in a separate streaming layer. The evaluation of our proposed data-driven quality prediction solution shows that it is capable enough to replace the manual quality testing in the manufacturing industry.

Preface

The document before you is my Master of Science thesis and represents my final work as EIT Digital master school student. This research is performed at Bright Cape in conjunction with the Distributed Systems group at the Delft University of Technology. Bright Cape is a consultancy company in financial data, and now they are starting their services in data analytics. I was hired as their first employee in software engineering department in Bright Cape to develop computing infrastructure for streaming data analysis for that project.

This thesis describes my contribution in the creation of a distributed stream processing system (which is also my specialization as MSc graduate), to analysis sensor data in nearly real time. That will help Philips to predict the quality of their products in real time using the sensor data that they collect from the assembly lines during the production process. This document will also serve as a blueprint to develop a software product, that Bright Cape can sell to other customers having similar use cases in quality predictions.

I am more than grateful to the people and institutions that have made all of this possible. First of all, I am deeply thankful for dr. Dick Epema, for his advice, and guidance during the whole duration of my thesis and also throughout my stay in Netherlands for the second year of my masters. Staying out of the home country was not easy, but his support never let me felt down, and I continued to meet all the challenges in my way. Then I am thankful for Mr. Serge Beniers, the founder of Bright Cape company, who trusted on my skills and selected me for that high profile project at Bright Cape. Furthermore, I would like to say thanks, Mart Althuizen, Erlijn Linskens and Dori van Hulst who guided, supported and inspired me throughout the project, especially during my six months stay in Eindhoven and also due to their support to communicate with the parties involved in this project. I would also like to thank EIT as an organization regarding their financial support for my masters.

Muhammad Najeeb Aslam

Delft, The Netherlands
16th October 2017

Contents

Chapter 1

Introduction

The emerging big data paradigm has transformed our society. Due to the broader impacts, big data paradigm is rapidly increasing as a segment of the IT industry. We are now living in a data avalanche era, evidenced by the sheer volume of data collection from diversified sources and the growth rate of data generation. IDC annual report [17] prognosticates that the rate of increase of global data volume during the period of 2005 to 2020, will be by the factor of 300, and data volume will grow from 130 exabytes to 40,000 exabytes, and that represents a double growth every two years. Big data has become the most valuable global commodity, to accelerate the solutions for the most pressing challenges of our time and has generated significant interest in various fields, including the manufacturing, healthcare, banking, telecommunication, security, and satellite imaging [8].

The potential associated with Big data has attracted a diverse attention from academic and industrial experts. Big data has grown as one of the most substantial research domain not only in IT but also in other disciplines like mathematics, finance, and industrial engineering; reasoning that the techniques required to generate value from the massive amount of available data are expanding beyond the available technological equipment and challenging the existing capacity of computing infrastructure. These challenges are mainly due to the heterogeneous nature of data, originated from a variety of disparate sources with a sheer volume. These challenges demand an overhauling reexamination of the current data storage, processing and modeling techniques; ranging from the architectural designs to the implementation details.

1.1 Defining Big Data

Big data terminology was originated during the mid-1990s, and that term became widespread after 2010. From the fact that how the word big data sounds, it is presumed that Big data is data which is quite huge in volume and its large enough to fit into traditional data storage systems. Clearly, size is the first aspect of big data but

fundamentally big data is not only the data which is very large in volume, but other aspects of big data have also evolved recently. In 2001, Laney [12] suggested that three Vs. (volume, velocity, and verity) are the main characteristic of big data. To focus more on the aspects of big data other than its volume, an architectural definition suggested by The National Institute of Standards and Technology (NIST) [10] is presented below:

Big data is a data where the data volume, acquisition velocity, and data representation limits the ability to perform effective analysis using traditional relational approaches or requires the use of significant horizontal scaling for efficient processing.

Another definition presented by Gartner [34]:

Big data is high-volume, high-velocity and high variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.

Similarly, TechAmerica Foundation [1] defines big data as follows:

Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information.

By these definitions, we further categorized big data paradigm into two categories, big data science, and big data frameworks. Big data science is the study of techniques covering the acquisition, conditioning, and evaluation of big data. It falls under the category of data mining, machine learning, and statistical analysis to get value out of the data. Whereas big data frameworks are software libraries and computing frameworks used to implement these advanced analytical algorithms and also to deal with the storage and processing systems to enable distributed storage and massively parallel processing across distributed clusters of computer units.

1.2 Role of Sensor data in Big Data Paradigm

The sheer volume of big data available today depends heavily on the various sources of data generation that evolved in different stages. The first stage began in the 1990s. When database and information management systems were widely adopted, and numerous management systems in several organizations were storing large volumes of data, for instance, banking transactions, customers billing from shopping malls, and government sector archives. These datasets are structured and can be analyzed through relational database management systems. Then the second stage began with the growing popularity of web systems. The Web 2.0 systems, characterized by web search engines and e-commerce businesses, generated a massive amount of semi-structured and unstructured data including weblogs and user-generated data. Finally, we entered the third stage of data genera-

tion triggered by the emergence of smartphones, tablets and sensor-based Internet-enabled devices. These smart devices are creating abundant of machine-generated data from a verity of origins, including vehicles, buildings, machinery, medical equipment, production industries and many other sources. This stage of the Internet of Things (IoT) is completely transforming the way organizations collect information and process business intelligence.

Therefore, the vast expansion of IoT and more precisely Industrial Internet of Things (IIoT) promoting a system of electromechanics having more and more sensors and corresponding data associated with machines, which is giving birth to fourth industrial revelation, represented as smart industries or industries 4.0. It refers to the technological evolution of robotic systems to data-driven cyber-physical systems, which means that industrial production no longer simply processes and the product, but that the product communicates with the machinery to tell it exactly what to do. This paradigm shift is opening new avenues to apply big data analytical strategies to industrial technology to solve complex problems in the most optimal way.

1.3 Problem Statement

Among many use cases of sensor data usage for industrial improvement, predicting the product quality using data has been a very sensitive and profound interest in manufacturing industry due to many reasons. The main factors are to save manual testing cost, reducing the time required to assure product quality and also for saving the cost of defective product production. Statistics show that most semiconductor production equipment suffers at least 8% unscheduled downtime and loses another 7% to scheduled maintenance [33].

With the invention of low-cost sensor technology [20], sensor data collection from assembly lines of production units has become very easy. Using this sensor data, a mathematical model can be developed to correlate the manufacturing operational conditions with the quality of the product [30]. Philips production unit (source of sensor data used for this research) collects abounding sensor data during the production process from assembly lines. That includes the machine settings and environmental conditions, for instance, temperature, pressure, the thickness of sheets, angles; width; height of different metallic parts and more. Finally, on completing the production cycle, for monitoring the production quality of finished products, product quality parameters are also gathered by sampling testing. Thus at this stage, only the quality of monitored products is grasped. Therefore, a failure occurred during the non-sampling periods where other products are under process may rigorously deteriorate the final production quality, which results in a huge number of scrapped products and thus a substantial loss of yield of fabrication. Moreover, this manual random testing is time-consuming and expensive; and results in a less efficient quality assurance practice in the manufacturing industry. To overcome this problem, the goal of this research is to develop an adequate way to

predict the product quality concerning to the process parameters and sensor data collected at different stages.

As the number of installed sensors is enormous, the product quality prediction is very complicated. To achieve the goal requires intermediate steps to be investigated. Sensors are producing time series of data, and its quite erroneous, most of it can be vague, misleading and irrelevant. Moreover, the shortage of knowledge about the relevant variables that affect the quality makes the problem more difficult. It is useful to identify quality deviations as early as possible and in real-time by applying data analysis on streaming data, so a stream processing system is also required to integrate quality prediction model. Hence, in a broader context, the main research is divided into two parts for this thesis.

RQ1: How to develop a prediction model to filter out faulty products from unlabeled industrial sensor data?

RQ2: How to design a reusable layered system architecture to analyze assembly line streaming data for quality prediction in real time?

The first question leads us to develop a mathematical model that uses sensor data measurements to predict unique product quality. We will investigate different machine learning and data mining approaches to answer this question. For which we will have to integrate data coming from various sources and then identify critical parameters that influence the quality. Several types of methods for the prediction of the product quality are presented in the literature, which can be divided into three categories: expertize-based methods, model-based methods and data-based methods.

The second question leads us to develop a pipeline of a stream processing engine to store and process data and then aggregate key performance indicators. This architecture design for a stream processing system includes an investigation for a storage system for the raw data, then its integration with parallel stream processing engine to handle time series of data and finally to produce output in a meaningful and understandable format.

1.4 Thesis Structure

The remainder of this document exists of five chapters and is organized as follows: Chapter 2 consists of background knowledge about the problem statement and also present literature review of state of the art techniques in similar research. Chapter 3 and 4 jointly answer the first research question (RQ1). First, in Chapter 3, we present some work on data preprocessing to reduce the dimensional space and then Chapter 4 describes procedural steps to develop a prediction model. Chapter 5 is organized to answer the second research question (RQ2) which presents an architecture of the stream processing tool for anomaly detection. Finally, in Chapter 6, conclusion and future work is presented.

Chapter 2

Background

In the current decade, an increasing number of organizations have been adopting the best practices for Operational Technology (OT) combined with Information Technology (IT). During recent years, the waste expansion of Internet of Things (IoT) and more precisely Industrial Internet of Things (IIoT) which promotes a system of mechatronics having more and more sensors and corresponding data associated with machines, is opening up new avenues to apply Big data analytics strategies to improve industrial Operational Technology.

In this chapter, we will describe a brief background of the industrial use case for which we are developing quality prediction tool. In Section 2.1 we will explain manufacturing process in Philips production unit. In Section 2.1.2 we will describe problems in the current quality testing procedure. Then in Section 2.2 we will explain the business process and data description. Finally, in the Section 2.3 we will present a literature review of few state of the art techniques that we are considering to apply in hope to solve the problem. We mainly classify existing approaches into three categories, and in each category, we will describe a couple of techniques with some of its advantages and disadvantages.

2.1 Philips Production Process

As described in the Section 1.3 that we are using sensor data provided by Philips Electronics and using their case study to develop a tool for the quality prediction that would be reconfigurable for the related manufacturing industry. Philips Shaving factory in Drachten is the birthplace of innovation in Netherlands, on this site they produce millions of shavers for men around the world every day. To provide the best shaving experience, Philips producing V-Track precision blades and contour detects technology which makes their shaver series unique in the world. The key to success in these shavers is the unique shaver heads (see Figure 2.1). On V-Track there are two blades, the inner blades and the outer blades. Outer blades are designed with v joints for better hair catch, and the inner track teeth are tuned

for better catching and cutting of long and flat hairs.

Figure 2.1: Top of the shaver head with inner and outer blades

When the shaving head is taken off, the part appears, is the core of innovation, the cutter. The cutter consists of V-shaped cutting elements (see Figure(a) in the Table 2.1), that gives a smooth shave, fewer strokes and leaving the skin in a smooth state. The contour detects technology is an improvement in shaving unit. It starts with a pivot that they elevate to skin level for better gliding. On top of the pivot, a shaving unit is placed. That ensures a smooth movement in the back and forth direction, and in left and right as well. On top of that, a primary tilting is embedded again for back and forth movement (see Figure(b) in the Table 2.1). Then they added a unique part, that is another tilting inside the previous titling (see Figure(c) in the Table 2.1), it is hunch perpendicular to primary tilting, ensuring the movement in the opposite direction then of its parent tilting. All in the collection this fitting provide eight movements at a time. These eight movements on top of the V-track blade system made this shaver unique in the world and gave to the user the best experience and results, ever found in any electric shaver in the past.

(a) (b) (c)

Table 2.1: Innder parts of the shaver: (a) V-shaped inside cutting elements, (b) primary tilting, (c) secondary tilting

The purpose of describing the steps to illustrate the production process of even a

small product like shaver is not single step process, rather its multi-stage manufacturing system (MMS) and multiple machines are involved in the whole procedure, each unit is responsible for producing a tiny component. Most of the procedure is carried out automatically and controlled by robots. In total, Philips produces some around 600 shaver models, and all have different quality standards.

Although, machines are quite automotive in their operations as described above paragraph. However, the quality assurance is entirely independent of the production system, and quality parameters are not adjustable automatically. Tracking the quality is not an easy task on such huge and complex automatic production system. The production consists of some steps in very much predefined order to complete one model of the shaver. For the correct production of the new model, a setup time is required, for the adjustment of the accurate setting of each machine (see Figure 2.2).

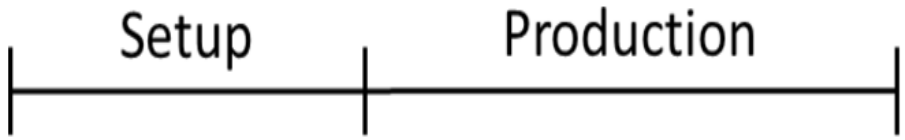

Figure 2.2: Setup time before the adjustment of machine settings

After the setup time for the adjustment of machine setting, it is expected that machine will produce exactly what is expected and quality is assured by monitoring the production by random sampling.

2.1.1 Quality Testing procedure

Philips produces millions of shavers, and It is almost impossible to test every product individually. Currently, they rely on offline measurements, and it works on random sampling. After the production of a certain number of products for instance 3600, they pick few random samples, and these samples undergo a complete measurements testing, that includes testing the angles of edges, circumstances of the rotating parts, curvatures of curved parts especially the rotating shafts and the width of lamellas from different positions. Some of these measures are very critical, and few are less sensitive. The products, having correct values of critical measurements and with minute differences of noncritical measurements are marked correct products.

2.1.2 Problems in current Quality Testing Procedure

On detecting a product, not passing the threshold quality measures, is an indication that error is detected and engineers are forced to perform following additional

actions.

- Detecting which part of the product is out of quality
- Tracking back to find out from which point error started
- Tracing back the root cause in assembly line (which machine, which settings)
- Replacing parts of the defected machine
- Adjusting the machine settings of that particular production unit
- Performing some cross-testing to assure quality in new parts

The above-described manual process is prolonged and not much reliable. It causes much time to reach the error pruning point. The time till error will be detected, main while production unit continues to produce faulty products (which is the most expensive part of the problem). Once an error is detected, engineers have no choice to stop the production process till the detection and correction of the error. With this quality testing practice, Philips faces many issues, which are not only making their manufacturing process slow but also creates bottlenecks to get full advantage of fully automated robot based production units.

Taking measurements of under construction products to find manufacturing defects as early as possible can reduce a lot of quality testing cost and also can help to automate the system. Additionally, it can also contribute to identifying any potential process or design flaws. Since defects are typically the result of many factors, analyzing long histories of assembly line sensor data can find subtle anomalies that signify product flaws. In recent years, different machine learning techniques have been employed to develop a quality prediction model for the manufacturing industry, and mostly these techniques relied on the historical data, but very less work has been done for developing a tool for quality prediction using live data coming directly from the production lines. The most popular techniques that are used for such predictions are the regression [21], clustering [21], association rules [27], and classification [9]. Mostly these techniques are applied in industries where products are manufactured in modules, i.e., injection modeling, for example, food processing industries and most of the models were developed for single stage quality predictions (SMS).

2.2 Business Process

Philips have quite complex multi-stage manufacturing system (MMS). The quality assurance is based on random sampling after regular intervals. To replace the hand-operated quality testing practice with a spontaneous data-oriented way, we closely

monitor and understand the mechanical system and identify following steps where we can introduce a data-oriented approach.

Figure 2.3 explain a very top and abstract level view of the shaver head production process (this is just to give an idea, the names of machines and flow have been changed due to privacy issue). First of all, a real of steel entered to the assembly line, where it undergoes various steps, and different operations are performed at different substations. That includes cutting, heating, colling, shaping parts to rotating blades and polishing, etc. All these operations are performed under different environmental conditions like temperature/pressure, and mechanical measurements are sensed using sensors placed at each sub-station.

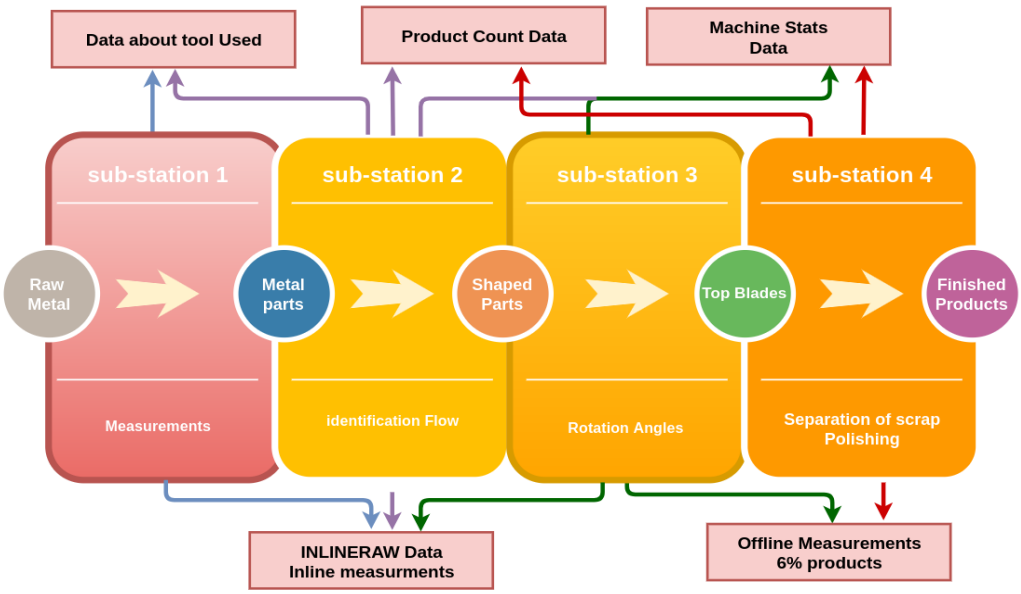

Figure 2.3: Overview of steps involved in assembly line production system of Philips (Figure is taken from the use case description of Philips)

According to the current quality assurance practice, after the production of a certain number of products (3600), the quality engineers pick two random products from the output of the last sub-stations. These samples undergo a complete measurements testing (offline testing). The offline measurement includes testing the angles, edges,circumstances of the rotating parts, curvatures of curved parts and the width of lamellas from different positions.Some of the measurements are very critical, and few are less sensitive. Quality Engineer repeats this step of picking samples and performing testing each after 10 minutes approximately. Products with correct values of critical measurements and with minute differences of non-critical measurements are marked correct products. On detecting faulty products during these offline measurements, quality engineer imitatively shut down the production line and take necessary steps according to maintenance SOPs as described in Section 2.1.

We propose a system for predictive maintenance, which will continuously scan the time series data while the products are under construction in different units. Then identify a pattern that the sensor data should follow for normal quality. Once the system observes any a deviation from the regular pattern, the quality engineers will be informed. Then they can re-examine machine settings to adjust perimeters and can take necessary actions to set the machine configuration. Using this predictive maintenance system, Philips will be able to replace manual offline testing cycle after 10 minutes with data-oriented quality assurance system. That will alert in case of out of quality production. Then will immediately remove that faulty product from the assembly line, so it will not proceed further in the production cycle.

2.2.1 Data Description

The data we are considering is of five types, captured by different sensors embedded at the different machines. Refereeing to Figure 2.3, the sensor on sub-station 1, 2, 3, and 4 are collecting data about tooling used on different machines. The sensor on sub-station 2 captures three types of data including, in-line measurements of every product, machines operational stats and also about product count information in bulk. At sub-station 4, offline measurement and some part of in-line measurement data is collected. We combine the sensor data of different sensors based on timestamps to get a full set of in-line data that we will use for the experiments in this research. Table 2.2 presents the description of the parameters in different data file.

Column Name	**Description**
Dotcode	Unique ID for a single product. Consists of a datestamp and increment counter like: 3-YY-MM-DD-hh-mm-ss-counter(3 digit)
M_01 to M_15	CTQ parameter 1 to 15, from a 2D measurement
M_mm_number	Measure for a single lamella on top of the cap
M_mm_min	Minimum value of a single lamella
M_mm_max	Maximum value of a single lamella
M_mm_dev	Standard deviation of a single lamella
M_mm_mean	Mean of the single lamella
M_mm_max_gleuf	Contains the number of the lamella with the highest measurement value
X_xx_min_gleuf	Contains the number of the lamella with the lowest measurement value

Table 2.2: Column names with attribute description of in-line assembly data

2.3 Literature Review

To identify faulty product data within the stream of sensor data, we will use anomaly or outlier identification. Anomaly detection is the identification of patterns in data set which do not follow a well-defined concept of normal behavior. In this section, we review different approaches to anomaly detection focusing on the broad categories of existing data analysis techniques. Here we present following techniques: Classification, Nearest Neighbor, Statistical anomaly detection, Information Theoretic, and Spectral Anomaly Detection.

2.3.1 Classification based Anomaly Detection

Classification [36] [14] is a systematic approach to build classification models from an input dataset. Neural networks, Bayesian networks, Support vector machines, and rule-based classifiers are few examples of classification techniques. Each method uses a learning algorithm to find a model that best fits the relationship between the label of input data and the attribute set. Classification based anomaly detection techniques work in two phases, the training, and the testing phase. The classifier first trains itself using the labeled data while in testing phase a test instance is fed in the trained classifier to find out either it is normal or anomalous. Classification based anomaly detection techniques are grouped into two categories according to the labels available for training data.

1. **Multi-class classification**: In this category, it is assumed that the training data contains labeled instances belonging to multiple classes [43] [6]. Such anomaly detection techniques learn a classifier model to distinguish different categories in the data set. A test instance is supposed as anomalous if it is not classified as a normal by any of the classifiers. Some techniques in this sub-category correlate a confidence score with the prediction made by the classifier. Further, if none of the classifiers classifying the test instance as normal, the instance is declared to be anomalous.

2. **One-class classification**: In this category, The main assumption is that the training data has only one class label. This technique learns a discriminative boundary around the normal instances using a one-class classification algorithm, e.g., one-class SVMs [42], one-class Kernel Fisher Discriminants [40] [41]. Then any test instance that does not fall within the learned boundary is reported as anomalous. Here are different classification algorithms for anomaly detection.

2.3.1.1 Neural Networks

Neural Networks techniques have been applied for both multi-class as well as one-class anomaly detection. In multi-class, a neural network is trained with data to learn the different classes. Then in the testing phase, each test instance is given

as an input to this trained network. If the network accepts the test, it is normal otherwise anomalous [43] [13]. Some multi-class neural network based techniques include Multi-Layered Perceptron [16] [28] [45] [18] [35], Hopfield Networks [24] [11], Auto-associative Networks [4] [25] etc. In one-class, Replicator Neural Networks have been used. Replicator Neural Networks for anomaly detection was first proposed by Hawskin et al [22]. Their network is a feed-forward multilayer perceptron that has the same number of input and output neurons with three hidden layers. If the network accepts the test input, it is normal.

2.3.1.2 Bayesian Networks

Bayesian Networks have been applied to anomaly detection in the multi-class category. In this technique, different attributes are assumed to be independent. A basic technique for a univariate categorical dataset uses a Naive Bayes network to estimate the posterior probability of observing a class label from a set of normal class labels and the anomaly class label. The class label with largest posterior is chosen as the predicted class for the given test instance. The likelihood of observing the test instance given a class and the prior on the class probabilities is estimated from the training dataset. Application of this technique has been recommended for network intrusion detection [7], novelty detection in video surveillance [38], anomaly detection in text data and disease outbreak detection [48].

2.3.1.3 Support Vector Machine

Support Vector Machines (SVMs) was first proposed by Vapnik [46] and have been applied to anomaly detection in the one-class category by Ratsch et al. 2002. SVM is a supervised method that needs some pre-knowledge before classification. SVMs learn a region that contains the training data instances (a boundary). Kernels, such as radial basis function (RBF) kernel, can be used to learn complex regions. For each test data instance, the basic technique decides if the test instance falls within the learned region or not. If a test instance falls within that range, it is considered as normal, else it is reported as anomalous. A similar technique Robust Support Vector Machines (RSVM) proposed by Song et al. in 2001, which is robust to the presence of anomalies in the training data and have been successfully applied to system call intrusion detection. This technique is mostly used in anomaly detection for audio signal data, novelty detection in power generation plants and system call intrusion detection.

2.3.1.4 Rule based Anomaly Detection

Rule-based techniques have been applied to anomaly detection in both multi-class as well as one-class category. This technique learns rules that capture the normal behavior of the system. A test instance that is not covered by any such rule is reported as an anomaly. A multi-class rule-based technique works in two steps:

First, it learns rules with a confidence level from the training data using a rule learning algorithm for example RIPPER. Second, it finds for each test instance the rule that best captures the test instance. Association rule mining has been used for one-class rule-based technique. Association rule mining for anomaly detection was first proposed by Agrawal and Srikant [47]. In that technique, association rules are created from a categorical data set in an unsupervised fashion. Application of association rule mining includes network intrusion detection, system call intrusion detection, credit card fraud detection and many other.

2.3.1.5 Computational Complexity

The computational complexity of classification based technique depends on which classification algorithm is used. For example, training decision trees are faster than training SVMs. SVMs are computationally more expensive as compared to the other methods we discuss below, to overcome expensive operation in traditional SVM, linear SVMs have been proposed that have linear training time but with certain limitations.

2.3.1.6 Advantages and Disadvantages

Classification based anomaly detection techniques use powerful algorithms to decide between data instances belonging to different classes. The testing phase is fast as each test instance gets compared against the pre-computed model.

In classification based anomaly detection techniques, assigning a label to each test instance can be a disadvantage as when a meaningful outlier score is required for the test instances. Also, multi-class classification based technique solely depends on the availability of accurate labels for various classes which are often not possible.

2.3.2 Nearest Neighbor based Anomaly Detection

Nearest neighbor based anomaly detection techniques exploit the idea of neighboring distance to calculate anomaly score. The techniques assume that the normal data instances occur in dense neighborhoods and anomalies are at a larger distance from their neighbors. Few methods based on Nearest neighbor approach compute the distance between two data instances, while other compute similarities between two instances to distinguish between anomalous and normal instances. This technique can be categories into two classes:

1. Methods in the first category estimate the distance of the data instances from their kth-nearest neighbor to evaluate the anomaly score. There are several ways to compute distance or similarity between two data instances. For continuous attributes, Euclidean distance is a common preference, but Euclidean distance can only be calculated of numerical attributes. For categorical attributes, a simple matching coefficient is used. Further, for multi-variant

data instances, distance or similarity is calculated for each single attribute, and then all individual distances are combined to estimate overall similarity. Application of this technique includes detecting landmines from satellite ground images and detecting irregularities in the winding fields of turbine generators.

2. Methods in the second category use a relative density of each data instance instead of the distance to compute its anomaly score. An instance that lies in a high-density neighborhood is designated normal while an instance that is in a neighborhood with low density is considered as anomalous. Density-based techniques perform poorly if the data has regions of varying densities. A set of procedures have been proposed to improve the performance. One of the most widely used technique is Local Outlier Factor (LOF) proposed by Breunig et al [19]. In this method, LOF score is calculated, which is equal to the ratio of the mean local density of the k-nearest neighbors of the instance and the local density of the data instance itself. The advantage of the LOF algorithm is its ability to detect all forms of anomalies, including those that cannot be detected by the distance-based method. Researchers have also proposed variants of LOF technique. They estimate the local density of an instance differently. Some variants have adapted the original technique to more complex data types. Since the original LOF technique is $\mathcal{O}N^2$(N is the data size), several techniques have been proposed that improve the efficiency of LOF. They include Connectivity-based Outlier Factor (COF), Outlier Detection using In-degree Number (ODIN) and Multi-granularity Deviation Factor (MDEF).

2.3.2.1 Computational Complexity

Nearest-neighbor based anomaly detection techniques have a computational complexity of $\mathcal{O}N^2$ for finding the nearest neighbors. These techniques also do not scale well as the number of characteristics increases. The computational complexity of the testing phase of nearest-neighbor based anomaly detection techniques is a significant challenge since it includes computing the distance of each test instance. The performance of a nearest neighbor based technique relies on a distance measure between a pair of data instances.

2.3.2.2 Advantages and Disadvantages

Nearest-neighbor based anomaly detection techniques are unsupervised and are purely data-driven. Semi-supervised techniques perform better than unsupervised techniques to find missed anomalies, since the likelihood that an anomaly will form a close neighborhood in the training data set is very low. Adapting methods in nearest neighbor based technique to a different data type is easy and only requires a suitable distance measure for the given data. However, Nearest-neighbor based anomaly detection techniques suffer from numerous drawbacks. For unsupervised

technique, if the data has normal instances that have not enough closer neighbors, or in another case, if the data has anomalies that have enough closer neighbors, the technique fails to label them correctly. For semi-supervised techniques, if the normal instances in test data do not have enough similar normal instances in the training data, the false positive rate for such techniques is high.

2.3.3 Statistical Anomaly Detection

Anscombe presented the concept of statistical anomaly detection techniques in 1960 [23]. The underlying principle is an anomaly is a data observation which is not generated by the stochastic model. Hence it can be presumed partially or wholly irrelevant concerning the data distribution. Statistical anomaly detection technique works on the assumption that normal data events occur in high probability regions of a stochastic model while anomalies fall in low probability area of the stochastic model. This technique can also be divided into two categories.

1. **Parametric Technique**: Parametric techniques works on the assumption that the normal data instances are generated from a parametric distribution, for example, Gaussian distribution. The parameters are estimated using Maximum Likelihood Estimates (MLE). The anomaly score of a data event is the distance of that data instance to the estimated mean. A threshold is applied to the anomaly scores to decide the anomalies. Other than Gaussian distribution there is two more parametric technique, Regression based model and Mixture of Parametric Distributions based model.

2. **Non-parametric Technique**: In nonparametric techniques, the model structure is not determined apriori but is instead determined dynamically from the given data instances. The nonparametric adopt model dynamically that is why this technique makes fewer assumptions about data as compared to parametric technique. One of the simplest non-parametric statistical-based anomaly detection techniques is to use histograms to maintain a profile of the normal data [15]. The technique involves two steps: The first step is building a histogram based on the different values of a single attribute in the training data. The second step checks if a data instance in the test data falls in any of the containers of the trained histogram. If an instance does not fit, it is reported as anomalous. Otherwise, it is considered as normal. Applications of the histogram based technique are intrusion and fraud detection.

2.3.3.1 Computational Complexity

Computational complexity completely dependent on the statistical model used to the data. Fitting single parametric distributions from the exponential family, e.g., Gaussian, Poisson, and Multinomial is typically linear in data size. Fitting complex distributions such as mixture models using iterative estimation techniques such as

Expectation Maximization are also typically linear per iteration, though they might be slow in converging depending on the complexity of the data patterns.

2.3.3.2 Advantages and Disadvantages

Statistical anomaly detection techniques are mathematically justified, and the methods are very efficient if a given probabilistic model will fit in the data set. This technique detects anomalies without storing the original datasets that are mostly of large sizes. However, the statistical anomaly detection techniques suffer from some key drawbacks. First, they depend on the supposition that the data is generated from a particular distribution and this supposition is not true for high-dimensional real data sets. Second, choosing the best statistical technique is not a straightforward task. Constructing hypothesis tests for complex distributions that are required to fit high-dimensional data sets is a complex task. Finally, histogram methods are effective for a single feature analysis, but they lose much of their effectiveness for multi or high-dimensional data because they cannot analyze multiple features at the same time.

Chapter 3

Quality Prediction Framework

In this Chapter along with next Chapter 4, we present our work to answer the first research question (RQ1). Anomaly detection is a technique that involves; first, identifying regular patterns in data and then finding subsequences that do not conform to the presumed behavior. These non-conforming patterns are often referred to as anomalies or an outlier. Figure 3.1 shows an example of an anomaly, where the normal frequency is between 50 to 70. The events, exceeding that range are anomalies. The major challenge in anomalies detection in a streaming context is that we cannot try complex processing in limited time to decide all the factors in the data. Rather we have to process the data on the fly very quickly and target only the most critical parameters in the data. This situation demands to develop a prediction model with quite refined data because the foundation of the success in any machine learning algorithm is based on the quality of the input data used.

In this chapter, we will describe few of the challenges we see to solve the RQ1 in Section 3.1. In Section 3.2 we will present a sketch of the procedural steps of our methodology to produce a prediction model that we can use for anomaly detection for streaming data. Then in Section 3.3, we will explain the first phase of our proposed procedure that is mainly a preprocessing part. In Section 3.3.1 we will start presenting our work from feature selection, we named it as dimension reduction. We present different possibilities for feature selection in each subsection.

3.1 Challenges in Anomaly Detection

At the conceptual level, an anomaly detection is a straightforward approach. By definition, we have to differentiation the observations, which are out of the boundaries of the normal data regions. For this, we need a profile or a model to compare new events to detect a subsequence that does not conform to expected normal behavior. However, several factors make this apparently simple approach very challenging: Few of the challenges are below:

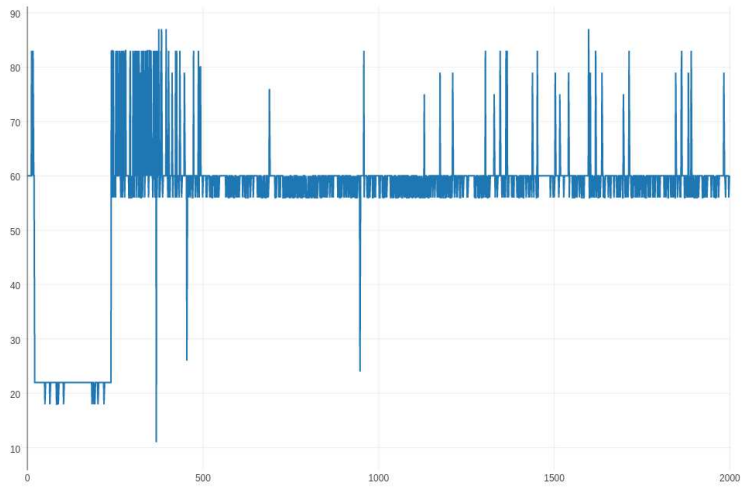

Figure 3.1: Example of anomalies in time series data

- Availability of labeled data for training/validation of models used by anomaly detection techniques is usually a major issue. In the current situation, there is no notion of labeling. All the data points we have are without any indication that either the product is normal or with faults. Establishing a profile for such data is quite complex.

- In stream processing, applying any machine learning algorithm is not much effective as in batch because the data arrive in small intervals and data ingestion is not as huge as most of the machine learning algorithms require to establish a solid concept about the patterns available in the data.

- In streaming context normal behavior keeps evolving, and a current notion of normal behavior might not be sufficiently representative in the future.

- Often the data contains noise which tends to be similar to the genuine anomalies. Therefore, it is troublesome to distinguish noise to remove it beforehand.

  In general, all of the challenges in common machine learning algorithms multiples in streaming context due to memory and time constraints and also due to the size of input data.

3.2 Procedure

In our research, we develop an unsupervised anomaly detection methodology by combining different techniques to handle various challenges in the data, few of them discussed in Section 3.1. Figure 3.2 shows the flow diagram of the steps involved in the process.
The first and significant challenge is high dimensionality. With over 700+ attributes, it is quite hard to use existing algorithms to generate output within a limited time allowed for streaming data processing. To cope with the high dimensionality, in the first phase shown as step I in Figure 3.2, we reduce the dimensions of the data by preserving the most significant meanings depicted by the original data set with raw dimensions. We reduced the dataset of 700+ features into a few dozen features, with the same value and meanings. In the next phase (from step II to step VI) shown with blue color in the Figure 3.2, we use a clustering model for anomaly detection. We identify the most promising attributes that participate the most in controlling the quality of the product. We perform various steps in this phase. In step II, we select an appropriate technique for clustering then in step III we chose a suitable value of K. Further, in step IV, we distribute the data in K clusters depending on the relevant data events. Then we perform cluster analysis in step V, to identify anomalous regions among the K clusters. Moreover, we focus our investigation on the normal clusters on determining the trends of most prominent feathers. Next, we establish upper and lower limits of these attributes within which a product can exist in the normal cluster. Finally in step VI, using the limits calculated in step V, we construct a profile vector with upper and bottom boundaries of each attribute. This profile vector is the output of this second phase. That profile vector will be utilized as a model to predict the quality level of unseen data observations received in the form of streaming.

In rest of the Chapter, we elaborate the first phase that is dimension reduction. The steps of the second phase will be explained in Chapter four. The third phase will be described in Chapter five, where we integrated the detection model into a pipeline of stream processing framework and will detect anomalies using profile vector in the streaming data.

3.3 Dimension Reduction

The recent explosion of data size has increased the number of records as well as the attributes to represent the state of each record. This increase of data size has triggered the development of many big data platforms and parallel data processing techniques. However, at the same time, it has pushed for the usage of data dimensionality reduction procedures to make the data more compact and concise to use the available computational power more efficiently. High dimensionality is always the first hardball for efficient data analysis. Furthermore, the way sensors are being employed in the industry to capture the state of events, dimensionality in the

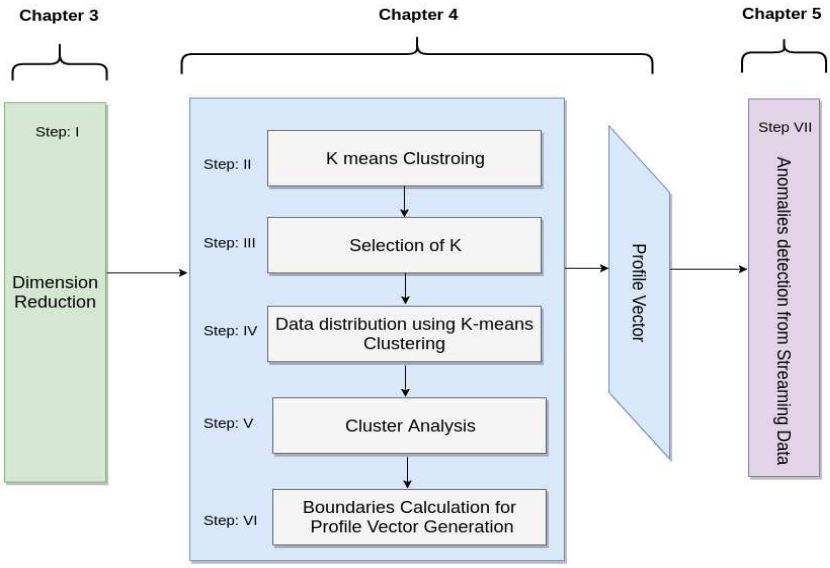

Figure 3.2: Steps involved in anomaly detection process for streaming data

raw sensor data became an even more prominent issue. There are many algorithms available for supervised and unsupervised classification; for instance, decision tree, naive Bayes, regression, clustering, logistic regression, random forest and more. Due to the internal learning methodology, most of these might find it difficult to deal with a very high number of columns. This is a problem inherent to the way the algorithms are designed and therefore a transversal problem to all data analytics tools. Here, the important thing to notice is that the problem is not the size of the data, rather the number of input columns. Internally many machine learning and data mining algorithms loops around the input columns due to which, the learning phase duration increases as quickly as the number of columns increases and could become prohibitive. Removing non-informative or dis-informative data columns indeed help to find more general classification rules and reducing dimensions speed up the algorithm execution to achieve overall better performances and better clustering accuracy on new data events.

From Philips production lines we collect machine positioning, off-line measurements, in-line measurements and a few other types of categorical data. A single sensor does not capture all the measurements, but multiple sensors are fixed at different positions to capture maximum state information from various angles. For instance, let's examine the inline measurement data set (described in section 2.4), that we use for our cluster analysis. These are the measurements of the top of the shaver heads during different stages of the production process. This data capture size, angles, spacing and thickness of cutting blades. These blades are named lamellas. In total there are 80+ lamellas on each shaver head. Nine sensing points

from different positions are used to measure the quality of one lamella, so there are 750+ measurements for all the lamellas (80+ * 9 = 750+) (exact number of lamella cannot be revealed due to design privacy). Additionally, there are fifteen other 2-D measurements as well. Hence, combining all this we have roughly 800+ dimensions to represent the state of each shaver head at one time of production process. This immense number of features causes not only high computational time and power for the convergence of any machine learning algorithm but also few exceptional/erroneous values of a less important feature may lead the algorithm to converge at a wrong deciding point. Additionally, it will impact the algorithmic performance. Now, the few questions we have about these attributes are:

- Are all these dimensions equally meaningful in the deciding the quality of shaver head?
- Are there measurements with little difference from one another so that we may use only one of them?
- Is there any repetition or overlap in the dimensions?
- Is there any intermediate value that can depict same meanings of more than one measurements?

To answer these questions, we explored a few of the state-of-the-art techniques to reduce the number of input features in the data set to make the feature space more concise and compact. After testing each of the reduction methods, we investigated resultant data to estimate how impactful this technique can be in our case. Next, we use a combination of best approaches systematically.

3.3.1 Baseline for Selecting Threshold Value

The goal of dimension reduction is to have a small and compressed data set with a fewer number of columns or features without damaging the original data distribution curves and patterns. In each technique described below, we refer threshold value many times to filter out attributes. Selecting threshold is not straightforward approach. Using higher threshold, we get smaller dimensional space, but remaining data set may not be the data required for our analysis. We apply the same detection model that we develop in Chapter four for anomaly detection on the original data set and data set after removing few of attributes with a particular threshold. Then we compare the results to test the effectiveness of attributes we selected. Then repeat the experiment by changing the threshold many times. In this way, we select a suitable threshold value for each technique. The complete procedure of a quantitative evaluation of Dimension Reduction is described in next Chapter after presenting prediction model in Section 4.4.

In Particular, we implement following techniques that are useful in our case.

1. High number of Missing values
2. Low Variance Filter
3. High Correlation Filter within the group of lamella and across the lamellas
4. Principal Component Analysis (PCA)

In the following section, these techniques are explained in detail.

3.3.2 Missing Values

Missing values is a widespread phenomenon in the data collected by sensors. We already considered this in data preliminary preprocessing part (which is not described in this thesis), which was about removing rows having various missing values. Here we examine it again with a different approach. Now we are more concerned about reducing the dimensions, not the data instances. There are different possibilities to deal with missing values:

1. **Mean Insertion**: Instead of removing rows we calculate the arithmetic mean of that column and insert the mean value to keep the size of the data same.

2. **Partial Removal**: If we have frequent missing values in some rows, then we remove these rows from the data set.

3. **Column deletion**: If a high number of values in columns are missing, then we drop the whole column.

The motive of the first approach is to preserve the data size by inserting an estimated value in missing cells. The other two approaches are more intended to drop the features or remove the data instances if the quality of data is low. In the current use case, we have a huge number of columns, and the goal is to reduce the dimensions, so we adopt the third approach first. Because, if a column has 70% to 80% missing values then remaining 30% to 20% possible values are likely not to be useful to classify the whole dataset. The above assertion motivates us to get rid of these columns to achieve our goal of compactness. We remove missing value columns in the following way. Read columns of each data set one by one and calculated missing value ratio with the formula:

$$Ratio \; = \; missing\, values \; / \; total\, values\, in\, the\, column$$

If the ratio is greater than a threshold, then we removed that column and evaluated this removal using baseline methodology for dimension reduction described in section 3.3.1. If the conditions there are satisfied, we drop this column. Otherwise, we apply Partial Removal method.

In this step, if the ratio of the total number of missing columns in a row is greater than a threshold, then we drop the whole row, as all artificially filled values in that row will not be effective during the analysis phase. After applying these two techniques, all the remaining missing values are filled with the mean of the associated column, and we end up with no missing values in the whole data set.

3.3.3 Low Variance Filter

The variance of the column is the expectation of squared deviation of the values from the average of the whole column. It measures how far values in the column are spread out from the mean of the column. In some cases, the sensor records the same value for all the instances, which results in a constant variable value throughout the column. Variance filter is a way to get rid of these persistent and nearly constant features. It determines how much information a data column has. Higher variance contains more knowledge associated with that column and a low the variance contains less or even no information. Dimension reduction using low variance have two constraints. First, it can only be implemented with the numerical data. Further for numerical data, it can only be applied on normalized data columns. The column ranges are normalized to make variance values independent from original ranges of columns. We used z-score normalization for data preprocessing.

To apply low variance filter, we loop through all columns. For the numerical data we replace column values with normalized values, and then we calculate the variance using the formula:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(X-\mu)^2}{N}$$

A column having variance zero depicts that only a constant value is filled in the whole column. With such high dimensionality, we believe that this constant variable will not help anyhow to improve our model. Of course, even a constant value can contribute in making a constant classification but in the present situation it will not be useful, and we drop these columns straight away. To further eliminate more columns we established a threshold (0.5) and the columns having a variance below than this threshold are selected. Passing through baseline test described in Section 3.3.1, we drop these columns as well. Remaining columns are replaced with the original values instead of z-score normalized values.

3.3.4 Highly Correlation Filter

In general, a dimension is a compelling feature in the data analysis if it is relevant to the class concept. However, it should not be redundant to any of the other related features. By theory, if data with multiple features are linearly separable in the original representation, it is still linearly separable if all but one of a group of linearly dependent features are removed [29].

Considering the above assertion, a feature is effective if it is not highly correlated to any of the other features. The high correlation between two features makes them depend on one another, and both carry similar information. The presence of both of these features will not add very much new information to the existing pool of input features. Any of the two or more related variables can predict a trend in the data without considering the other variables. This understanding of feature relation boils down to the problem of finding a suitable measure of correlations between features. It sounds like a procedure to select features based on this measure. High correlation filter proved very effective in our data set.

There exist broadly two approaches to measure the correlation between two features. The first approach is based on classical linear correlation and the second uses information theory. We selected linear correlation because of its two main benefits. First, it helps to remove features with near zero linear correlation to the class. Second, it contributes to reducing redundancy among selected features. We calculate the linear correlation for a pair of variables (X, Y) using the following formula:

$$Corr = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x^2)][N\sum y^2 - (\sum y^2)]}}$$

The value of Correlation lies between -1 and 1. If X and Y are completely correlated, it takes the value of 1 or -1. If X and Y are independent, then it is zero.

Based on the above theory, we remove highly correlated columns in the inline dataset. The dataset consists of 80+ lamellas, and each lamella has multiple measurements. This means that the data represents a grouping structure. To preserve that original grouping structure, we adopted correlation filter in such way that highly correlated columns within the same lamella are filtered first. As result of this, we found a smaller set of features to represent the state of each lamella. The reduction pattern is as follows:

$$[x\_1\_5, x\_3\_5, x\_5\_5, x\_6\_5, x\_8\_5, x\_9\_5] \Rightarrow [x\_5\_5, x\_8\_5, x\_9\_5]$$

Each entity in the comma-separated list is a column name in the in-line dataset. x_1_5 represents the first measurement of the lamella 5. Similarly, x_6_5 is the sixth measurements of lamella 5. For more details please refer to data description in Table 2.2 of Chapter 2.

Then the high correlation filter is applied to the remaining subset of the measurements of each lamella with the subset measures of the other lamellas. Then we filter out one complete subset of measures for one of the two highly correlated lamellas. We apply the correlation filter in following way:

First, all the features of a single lamella are selected, then a correlation matrix

for all pairs of columns is computed using correlation formula. Figure 3.3 shows a heat map of the correlation matrix of the columns in a single lamella. Where values range from intense green (+1 = full correlation), yellow (0 = no significant correlation), red (-1 = inverse correlation). The diagonal of the matrix is full green, which shows the self-correlation of a data column and of course, this should be a 1.0 correlation. Next, we use this correlation matrix as input and filter out all those columns correlating higher than the threshold and remove one of these columns. As result of this filter, we managed to drop 3 out of 6 columns for each lamella. We also found a very high correlation of each odd lamella with its next even lamella. This pattern is because, in the Philips production unit, these lamellas are not shaped individually rather these are shaped in the form of groups of two or four. Note that, correlation coefficients can only be computed for numerical columns and also require normal data ranges. So we first normalize all columns and then after filtering out columns we replaced remaining columns with the original values in the similar fashion as we do for variance filter.

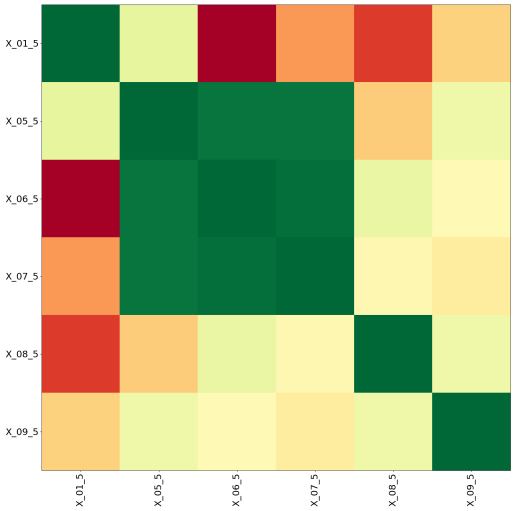

Figure 3.3: Heat-map of correlation matrix for lamella 5

3.3.5 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation) [29]. The number of principal components is always less than or equal to the number of original variables. This transformation is defined in such a way, that the first principal component has the largest possible variance (that is, accounts

for as much of the variability in the data as possible). Then each succeeding Component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding Components.

The goal of PCA is to find the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace with lower m dimensions, m is smaller than the original number of dimensions while retaining most of the data information. Therefore PCS is another strong candidate to reduce dimensionality in our case. Using the previous two techniques, columns with low variance and high correlation have already been removed so at this stage our dataset suits well to apply PCA. To apply PCA, we defined a strategy such that: all the remaining measures of each lamella, will be replaced with its first principal component. Below we show the columns names for the measurements of lamella five that remained left after the all previous techniques, and then these measures are replaced with first Principal The PCA conversion on let's say for lamella five is like this way:

$$[x\_5\_5, x\_8\_5, x\_9\_5] \Rightarrow [first\ principal\ component]$$

Each entity in the comma-separated list is a column name in the in-line dataset. x_1_5 represents a measurement of the lamella 5. Similarly, x_6_5 is the sixth measurements of lamella 5. For more details please refer to data description in Table 2.2 of Chapter 2.

An important point to notice before to apply PCA is that PCA transformation is sensitive to the relative scaling of the original variables, and the new principal components are not the real data values produced by the system. So any prediction for original data will not be valid anymore after PCA. PCA is not a suitable technique for any data set where the interpretation of values is important. Our goal is to distribute the data into clusters and then defining boundaries which give us leverage to use PCA for our dimension reduction problem. Data column ranges also need to be normalized before applying PCA.

3.4 Conclusion

By applying different techniques, a maximum possible number of dimensions have been reduced by applying different threshold values. In total, we preserved 92 attributes before to apply PCA and 46 after applying PCA. We selected best possible thresholds for each technique by repeating our experiments several times. Table 3.1 shows a summary of for each dimensionality reduction techniques and general patterns in which column have been removed.

In Table 3.1, it can be observed that only a few columns have been dropped using missing value filter which shows that the data collection accuracy is quite good. Then 7% of the columns have been removed in low variance filter which is un-

derstandable in huge data set collection. The high correlation filter proved to be the most successful approach here. This is because the individual lamella is a tiny area in the surface and sensing measurements that very close to one another will give high correlation. We improved this grand closeness factor up to some extent by removing similar measurements. Finally, PCA proved to get an intermediate value from two variables. The goal of this phase was to apply available techniques to reduce a maximum number of features to make a lighter data set. On one side, dimension reduction will help to achieve a better performance in machine learning algorithms. On the other hand, it will give insights to Philips to understand what type of redundant information is being captured and how they can collect a compressed but enriched data set instead of capturing features with redundant meanings.

Reduction Technique	**Reduction Rate**	**Threshold**	**Removal Pattern**
Missing Values	1%	25% Ratio of missing values with total number of values	Random attributes without any special pattern.
Low Variance	7%	0.5	Most of attributes with 2nd 3rd measurements of lamellas
High Correlation within Lamellas	35%	0.85	3rd and 5th measurement
High Correlation Across Lamellas	23%	0.80	Every even lamella removed
PCA	12%	First Principle component preserved	Combined 8th and 9th measurement

Table 3.1: A comparison of different dimension reduction techniques with success rate and threshold

Chapter 4

Prediction Model

Anomalies in time series data can be detected in several ways. One is by making predictions rules and comparing the observed values with these predictions rules. Another is constructing a profile of the time series and checking whether a new event fits into the profile. A third is by information theoretic approach where a point is an anomaly if removing it makes the fitting less erroneous. In this research, we focus on the second approach of constructing a profile of the normal or desired behavior and then checking the fitness level of new products with that profile.The major challenge in the current context is deal with streaming data, indicators to identify anomalies can change over the time as data moves. This issue can be tackled by dynamic updating the prediction model/profile and use the updated prediction model as new data points arrive.

In this chapter, we will explain an unsupervised machine learning methodology to distribute data into different clusters in Section 4.1. In Section 4.1.3 we will present some results of cluster analysis to distribute clusters into desired and undesired classes. In Section 4.2 a scheme will be explained to develop a profile constituting of few critical attribute values that effect the quality of the product, we use this quality profile as prediction model in the streaming tool. Finally, in Section 4.4 we will explain a scoring model to evaluate dimension reduction techniques to quantitatively test the quality of our selected attributes in the Chapter 3.

4.1 Clustering

Clustering is an unsupervised machine learning technique which is used for grouping entities based on similarities among those entities. The goal is to construct clusters such that entities in one cluster are more closely related. As opposed to classification and other supervised machine learning approaches described in literature review Section 2.3, where the goal is to learn based on data labels, clustering involves learning based on observation to understand the trends. Most of the

clustering algorithms do not assign all points to the clusters but account for noisy objects. In other words, clustering algorithms are optimized to find clusters rather than the outlier. After data distribution into distinct classes, outlier detection is a different phase in this approach which looks for outlier by applying one of the clustering algorithms and retrieve the noisy set. After the formation of clusters density or distance, estimation is performed on the clusters to identify properties of each cluster, which normally depends on the goal of cluster analysis. In our case, we do not have a filter to check the value ranges all columns of data and streaming context doest allow this preprocessing. Therefore, we focus on identifying clusters having a majority of normal values. Then we estimate the range of each attribute within the normal clusters. This helps us to have an estimate what can be a standard range of attribute values for a normal product, the products having attributes values deviating the established standard. In theory, every clustering algorithm can be used to cluster the data in a first step. However, in practice k-means is commonly used to take advantage of the low computational complexity, which is linear compared to the quadratic complexity of the nearest-neighbor search.

4.1.1 K-mean Clustering

For clustering step, we chose K-means clustering algorithm. It is one of the simplest clustering algorithms widely used in the unsupervised classification of machine learning tasks. James MacQueen first proposed this technique. It automatically recognizes groups of similar data instances. K-means algorithm classifies instances to a pre-defined number of clusters, this predefined number of clusters value is known as K parameter in that algorithm, and it is a mandatory parameter. The first important step is to choose a set of K instances as centroids (centers of each cluster) randomly. Next, the algorithm continues to read new data points and assigns it to the closest cluster. The cluster centroids are always recomputed after each insertion of a new data point. The process is iterated until no more changes are made. Following are the Key steps in the algorithm:

Step 1: Select the total number of clusters, K
Step 2: Randomly chose K centroids as centers of each cluster
Step 3: Calculate the distance from each instance to all the centroids using Euclidean method
Step 4: Assign each instance to the closest centroid
Step 5: Update the centroids by calculating new centers
Step 6: Repeat *step 3-5* until the centroids do not change

4.1.2 Selection of K

Determining the number of clusters (K) is a very important phase in k-means clustering. This is acknowledged as a distinct step in the process of actually solving

the clustering problem. There are different theories for optimal selection of k. One of them is a visual approach which focuses on visually determining how many clusters are required for a proper binding of the data. This method is manual and quite difficult especially in unsupervised learning problem where available data is not labeled. Another method is to set the number of clusters to the square root of the number of data points divided by two. This approach is again a rough guess. We opted a more scientific method known as elbow method. In the elbow method sum of squared error (SSE) is calculated for the possible values of K (for example 2, 4, 6, 8...). By definition, SSE is the sum of the squared distance between each entity of the cluster and its centroid. Mathematically is represented:

$$SSE = \sum_{i=1}^{k} \sum_{x \varepsilon c^i} dist(x, c^i)$$

Figure 4.1 shows a graph plotted for numbers of clusters (k) against the SSE. It can be observed, that the K gets larger error decreases as well. This is because when the number of clusters increases, the distance to the centroids also decreases, therefore distortion is also lower. The idea of the elbow method is to choose the K at which the SSE falls abruptly. Aforementioned produces an ”elbow effect” in the graph, that is why that process is known as elbow method. Observe that a first drop occurs when k = 4. We chose five clusters which one after the significant decline occurs.

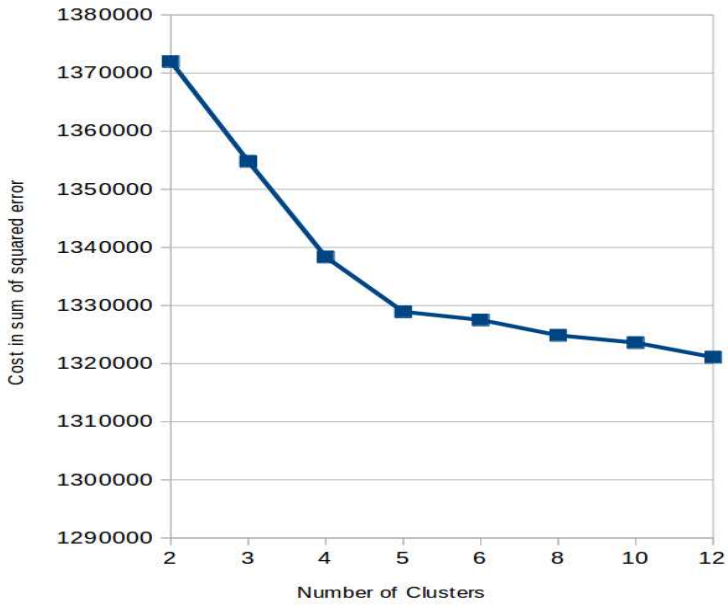

Figure 4.1: Elbow effect for the cost computation for optimal number of clusters

4.1.3 Cluster Analysis for Outlier Detection

In this phase, we further breakdown the clusters into two divisions. We radically investigation our formulated five clusters to draw a boundary between the clusters having normal products and the one with possible anomalies or outlier. Cluster analysis for outlier detection can be divided into two categories: distance-based outlier detection and density-based outlier detection. The first method, distance-based outlier detection assumes that the normal data instances have a dense neighborhood whereas anomalies are far away than its neighbors. Greater the distance of the object to its neighbor, the more likely it is to be an outlier. The second method assumes that the density around the normal data instance is similar to the density around its neighbors while the density around an anomaly is considerably different to the density around its neighbors. Distance-based methods require computing the distance of each data point to the centroid of its cluster, while density can be estimated with the range of values. For example, septation between max and min value within the cluster. The method we implement for subdivision of cluster analysis is explained in next section.

4.1.4 Methodology of Performing Cluster Analysis

For each single feature (column in the data set), an univariate k-means clustering is performed, and data is distributed into 5 clusters. Then we first sort the clusters according to their size into descending order. We use a heuristic method to classify clusters further into two groups for further analysis. Heuristic analysis is rooted in two assumptions of clustering like other unsupervised machine learning techniques. Next, we validate these two assumptions.

4.1.4.1 First Assumption Validation

As described in Section 4.1 the clustering based anomaly detection works on a strong assumption that most of the data observations are normal and anomalies are relatively rare in the data set. This hypothesis upholds in our case, and we validate that assumption with the help of ground realities of the data source. The analysis is performed using real data obtained from production unit of Philips shaving factory, generated just a few months ago. During that time there were no exceptional production or quality issues, and most of the production was up to the quality standards. Hence the majority of the data instances must fall within the normal region, and bigger clusters must consist of the normal data observations.

4.1.4.2 Second Assumption Validation

Further, we cross-validated our presumption with density estimation of normal clusters. The assumption is that the normal clusters are always with higher density as compared to anomaly clusters. Figure 4.2 shows a comparison between the size, against the density of each cluster. In this chart, the size of clusters is represented

by red color and density with a yellow color; density is computed by the difference between the largest and smallest values in the cluster. Clusters are sorted according to size. It can be seen that largest cluster (with the largest red bar) have a very short range of values dispersion while smallest cluster with the least red area has the most significant length of the yellow bar. Hence, It is clear that larger clusters are compact and events lie in dense ranges with a high density. Similarly, normal behavior starts from largest cluster and anomalies lies mostly in the smallest clusters. As there was a huge difference between the size of the cluster and the density so to present on the same scale in the graph, we divided the size with ten and multiplied the density by ten.

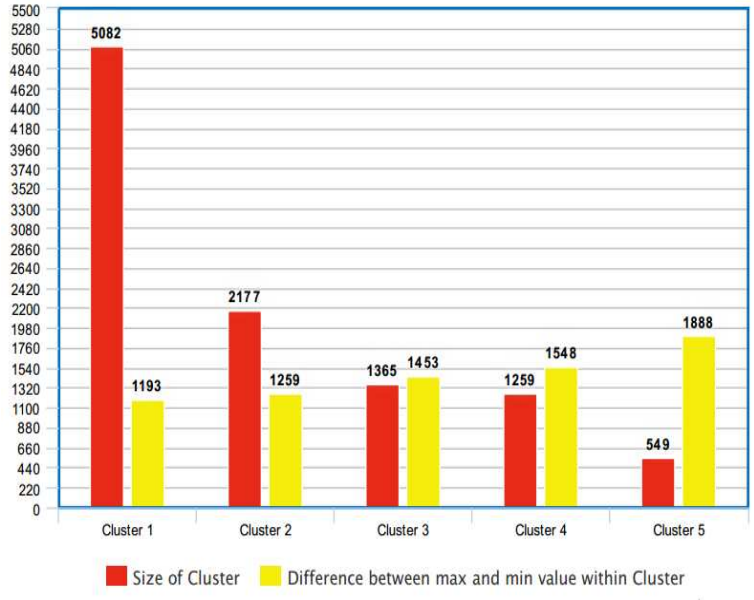

Figure 4.2: Comparison of size of each cluster against the dispersion of values within the cluster

4.1.5 Discovering a Boundry between Normal and Anomalous Clusters

After fortifying the hypothecal assumptions about data distribution in normal and anomaly clusters. The next challenge is to draw a sharp boundary line between both of groups so that we have a distinct separation to quantify the properties of normal and anomalies regions. In general, it is hard to achieve a clear boundary, especially with high dimensional data. Most of the approaches result in a blurred boundary. To avoid the effect of the overlapping region, we decided to use only the extremes of both types of clusters. For this, we sort clusters according to the size and holding our assumptions made in cluster analysis; the largest clusters have normal and smallest clusters contains anomalies. Although the edges of both zones are overlapped; but, the probability of finding a normal product on the extreme left

(in the largest cluster) is quite high. Similarly, any candidate anomalies must fall on the extreme right (smallest) cluster. Hence, from these two extremes, we can understand the properties of both groups.

4.2 Profile Vector Generation

The goal of cluster analysis is to find a compact region with high confidence that the products in that region are normal and acceptable for production quality. So that, we can use the properties of these products to assemble a standard scale of acceptable measurements in the product data. In this step, we generate a profile vector containing the list of attributes with the desired value of each attribute. The centroid of clusters symbolizes that this is the mean value of an attribute for all the points in one cluster. The centroid of the largest clusters intimates that it is the average value for the majority of the points in the data set. As the largest cluster also contains normal products, therefore centroid of the largest cluster is the most desired value for a normal or acceptable product. We use the same intuition to construct profile vector. To make it more generalized, for each attribute we use the centroid of top two largest clusters. Profile vector is a special data structure with a list of attributes (selected in dimension reduction process) with desired values of each attribute. Each attribute has following three properties.

$$
\begin{aligned}
&Mean(M) = mean\ of\ the\ centroids\ of\ top\ three\ clusters \\
&Standard\ Deviation(STD) = Highest\ standard\ deviation\ in\ top\ three\ clusters \\
&Marginal\ Threshold(MT) = \sqrt{Mean(M)}/10.
\end{aligned}
\tag{4.1}
$$

The marginal threshold is a constant factor to cover the broader range of acceptable values. As in this research, we do not have an environment to test the output, so to make it adoptable we use this marginal threshold. That helps to tune the algorithm for different acceptable quality standards. We try different formulas to calculate this. For one of the experiment, we divide the square root of mean by 10 to normalize all the attributes scales, a few of attributes have value range between 200 to 300 and few have between 1900 to 2500. To balance that margin we obtained that range with the above formula. Below table shows the structure of the Profile vector.

	F1	**F2**	**F3**							**F92**
Mean	2250	247	263							
SD	33	13	17							
MT	22	2	2							

Table 4.1: Profile Vector with 92 attributes and each attribute with three properties

4.2.1 Results after Cluster Analysis

Table 4.2 presents the results of first 30 attributes after cluster analysis on the one-month product data. The first column is the name of attribute, second and third columns are the mean and standard deviation of normal clusters identified in cluster analysis. To give a comparison, in the fourth and fifth columns are the mean and standard deviation in the anomalous cluster. The sixth column is a difference of the means in both types of clusters.

	Normal Cluster		Anomalous Cluster		
Attribute	**Mean**	**STD**	**Mean**	**STD**	**Difference**
X_05_1	258.16	1.26	237.28	3.41	20.88
X_08_3	247.34	0.82	240.78	1.70	6.55
X_08_1	266.63	0.88	254.69	1.86	11.94
X_09_1	250.04	0.97	236.06	2.19	13.98
X_09_3	234.64	0.77	227.59	1.91	7.05
X_08_5	268.12	0.74	256.13	1.64	12.00
X_09_5	255.36	0.93	238.87	2.55	16.49
X_08_7	245.46	0.83	238.93	1.60	6.53
X_09_7	235.67	0.78	226.16	1.73	9.51
X_08_9	264.59	1.00	256.60	2.06	7.99
X_08_11	245.43	0.91	253.71	2.11	-8.28
X_09_11	233.34	0.84	240.45	1.72	-7.11
X_08_15	244.93	0.85	237.82	1.92	7.10
X_09_15	234.50	0.78	228.13	1.66	6.37
X_08_17	258.14	1.05	269.88	1.51	-11.74
X_09_17	251.01	0.99	239.19	2.08	11.82
X_08_19	249.74	0.78	256.10	1.73	-6.36
X_09_19	239.26	0.70	230.99	1.54	8.27
X_08_21	258.39	0.82	251.94	1.52	6.45
X_09_21	248.05	0.82	236.92	2.16	11.13
X_08_23	245.55	0.84	238.60	1.91	6.95
X_09_23	235.93	0.68	220.63	2.62	15.30
X_08_25	253.63	0.98	266.31	1.88	-12.68
X_09_25	249.13	0.84	238.65	2.16	10.49
X_08_27	248.41	0.85	241.18	2.16	7.22
X_09_27	239.74	0.63	231.86	1.55	7.87
X_08_29	254.84	0.78	,244.43	2.11	10.41
X_09_29	243.96	0.69	,237.38	1.66	6.58
X_08_30	252.79	0.75	,226.85	1.65	25.94

Table 4.2: Comparison of the mean and the standard deviation in normal and anomalous clusters

4.3 Anomaly Detection

Finally, we have a standard profile vector to measure the correctness of new data events. On receiving a new streaming window, first, we perfume features selection and attributes transformations according to profile vector to make the observation fit for the comparison with the profile vector. Then we loop through all the attributes to predict its class. We calculate boundaries of $\imath\ th$ attribute with the formula below:

$$\begin{aligned} &Mean = Vector[i][Mean] \\ &Standard\ \ deviation(STD) = Vector[i][STD] \\ &Marginal\ Threshold(MT) = Vector[i][MT] \end{aligned} \tag{4.2}$$

$$\begin{aligned} &Upper\ limit(UL) = Mean + (STD\ \ *\ \ MT) \\ &Lower\ Limit(LL) = Mean - (STD\ \ *\ \ MT) \end{aligned}$$

Any instance having values out of the upper and lower limits is considered as candidate anomaly concerning $\imath\ th$ feature. There are total 92 entries in profile vector, and 92 comparisons are performed. For each comparison, if a value lies out of the boundary (upper and lower limits) a unit anomaly score is incremented to *Total_Score* of that data instance. After all 92 comparisons, *Total_Score* decides if data instance will be reported as an anomaly or not.

Further, we have two level of alerts. The first level of alert is a count of the products having anomaly score greater than first threshold and the second level alert is the count of the data instances having anomaly score over the second threshold level. A product can have maximum 92 anomaly score, as there is 92 profile vector comparison for each product. Finally, as an output of each streaming window, both level alert count with product ids and all the attribute measures are served. Complete implementation of streaming anomaly detection is explained in Section 5.5.1

4.4 Quantitative Evaluation of Reduced Dimension

The criteria we explained in Chapter 3 for dimensionality reduction is based on two points, execution time and accuracy. Reduced dimensions speed up the algorithm performance by maintaining high accuracy in the classification/clustering algorithms. In this section, we quantitatively evaluate dimension reductions to measure how accurately the selected attributes represents the meaning of original data concerning the execution time/cost.
In general, this type of validation is performed with the help of labeled data. If the labeled test data is provided, then it is quite a simple to compare the classification results of two set of attributes (reduced and original dimensions) with a testing

data. Then we can measure the accuracy of both sets of attributes. However, in the current study, the experimental data is not labeled at all. In the absence of labeled data, we adopt a classifier technique based on clustering model we developed in the first part of this Chapter to validate our reduced dimension results. First, we explain a strategy to label the data using a scoring method of K-means clustering in Section 4.4.1. Then in Section 4.4.2 we describe steps to evaluate the quality of reduced dimensions.

4.4.1 Calculating Anomaly Score

We use a formula inspired by Leonid Portnoy [37] to compute a confidence score per data instance from the clustering model that shows how likely a data instance is a candidate anomaly. The confidence score reflects the likelihood or an instance to be an anomaly or not. First, we distribute the whole data set into 5 clusters using K means clustering. The chosen value (5) for k is opted because of the experiment performed in Section 4.1.2. This clustering gives us all data instances with corresponding clusters (1 to 5) then we compute the size of each cluster. Finally, we calculate a confidence score for each data instance to assign it a label of an anomaly or a normal. Lets assume x_i is a data point and it is assigned to the cluster $C(x_i)$. We calculate anomaly score $S(x_i)$ using formula:

$$S(x_i) = \frac{N_{max} - N_{c(x_i))}}{N_{max} - N_{min}}$$

In this equation, N_{max} is the size of the largest cluster and N_{min} is the size of the smallest cluster. $N_{c(x_i)}$ represents the size of cluster assigned to x_i. Examining the equation carefully; it can be noticed that $S(x_i) = 1$ when x_i is assigned to the smallest cluster and $S(x_i) = 0$ when x_i is assigned to a largest cluster. This formula mathematically represents the intuition we build in our cluster analysis Section 4.1.3, that is the data points assigned to smaller clusters are more likely the anomalies, and the larger clusters represent normal data.
The anomaly score obtained from above formula is always between 0 to 1. Further, to assign a nominal label to each data event, we label each instance as an anomaly (Negative) if If the anomaly score is greater than 0.75, otherwise we label it as normal (Positive).

4.4.2 Evaluation Procedure

The procedure to label the data and then a comparison with reduced dimension is explained as follow:

- *Step 1:* First, we label all the instances in the original data including all the raw dimension/attributes as normal or anomaly using anomaly scoring method described in section 4.4.1. We name this labeled data as a base classifier.

- *Step 2:* Then we use lesser attributes (e.g., reducing 100 dimensions from the original data set by applying our dimension reduction techniques) and then label this data using the same method as in step 1. We name this data set the predicted classifier.

- *Step 3:* Finally, We compare the labels in the base classifier of *step 1* with labels in the predicted classifier of *step 2*. The results can be represented in a structure known as a confusion matrix or contingency table. The Table 4.3 representing confusion matrix which has four divisions: True positives (TP) are examples of base classifier correctly predictive as positives. False positives (FP) correspond to negative examples incorrectly predicted as positive. True negatives (TN) refer to negatives correctly labeled as negative. Finally, false negatives (FN) resembles positive examples incorrectly predicted as negative.

	Prediction Positive	**Prediction Negative**
Truth Positive	TP (Predicted Correctly)	FN (Prediction Error)
Truth Negative	FP (Prediction Error)	TN (Predicted Correctly)

Table 4.3: Confusion table with comparison possibilities

The question we address here is how many targets (true positives) or non-targets (false positives) will be identified correctly if we use less number of attributes? We try a different set of attributes and repeat step 2 and step 3 to measure the accuracy in each experiment. The Table 4.4 shows the accuracy achieved in 6 experiments (A to F) performed on one-month data set of 104031 products. We use following formulas to calculate accuracy.

$$
\begin{aligned}
True\ positive\ Rate &= \frac{TP}{TP+FN} \\
True\ Negative\ Rate &= \frac{TN}{TN+FP} \\
Total\ Accuracy &= \frac{TP+TN}{TP+TN+FP+FN}
\end{aligned}
\tag{4.3}
$$

For each experiment, we also measure time to calculate anomaly score. We are interested in the time needed to classify whole data set into anomaly or normal classes. Hence, after ignoring the time required to read data from the disk, we only measure the time taken for (i) performing K-means clustering (ii) calculating anomaly score and (iii) assigning labels to each data set.

4.4.3 Choosing optimal Set of Dimensions

The goal of dimension reduction is to get a minimum set of attributes (dimensions) that can represent closely the meanings of the original data set with the minimum computational time to apply clustering or other machine learning algorithms for anomaly detection. In Table 4.4 we show the results of our six experiments (A to F). The second column represents the numbers of attributes, the third column depicts accuracy achieved, and the fourth column refers to the time taken to perform clustering experiment. In the fifth column of that table, we compute a ratio of accuracy to the time required to achieve that accuracy (accuracy/time). This quantity gives us deciding point to chose the best set of attributes. It is evident that using the higher number of dimensions increase the accuracy but require more computational time which decreases that ratio factor. A set of smaller dimensions gives results in lesser time but accuracy degrees as well, the goal is to get the point with the maximum ratio.

Experiment	**Number of Attributes used**	**Accuracy in %**	**Time in minutes**	**Ratio**
A	485	99.20	150.35	0.66
B	365	97.60	113.15	0.85
C	225	95.40	69.75	1.37
D	92	92.80	29.14	3.18
E	74	70.10	22.90	3.05
F	49	60.30	27.90	2.69

Table 4.4: Representation of time required to achieve certain accuracy with certain number of dimensions

To understand the relationship better, we plot a graph shown in Figure 4.3. X-axes represent dimensions used, and Y-axes represents accuracy to time ratio. Starting from experiment A, we can see on the extreme left of the graph that despite the very high accuracy, the ratio of accuracy to time is quite low. Moving toward left, in experiment B and C, accuracy decreases a bit from 99.2 to 95.5 due to lesser attributes, and the less computational time also improves the value of the ratio. The best results we achieved in experiment D, in which we used the best possible thresholds in our dimension reduction techniques to get 92 dimensions that really satisfy metamathematical definitions. Referring to that Table 4.4, it can be seen that the accuracy is more than 90 in this experiment and results are achieved with 29.14 minutes with highest value of ratio. Referring to the last experiment E and F, we reduce more dimensions, that give results faster but accuracy is decreases and value of ratio is decreased as well.

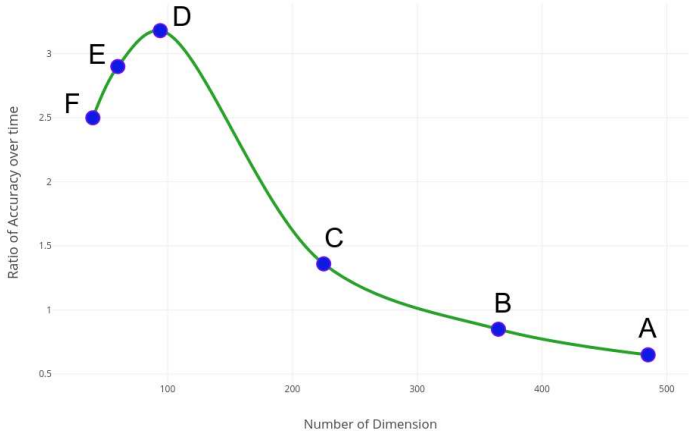

Figure 4.3: A graph representing the slop of ratio(accuracy/time) with increasing dimensions

4.5 Conclusion

In this chapter, we used unsupervised machine learning approach to make distinct classes of unlabeled data to distribute individual data events into different clusters. Then we performed cluster analysis to validate assumptions of clustering methodology. In these analyses, we found that majority of the normal data events fall in a dense region and also the size of the normal cluster is always more significant than the other clusters. After filtering out products in the normal cluster, we generated a profile vector that is just a value ranges of few critical attributes within which a product falls into the normal clustering region. Finally, we used the same prediction model to develop a scoring model to evaluate our dimension reduction strategy. We used this scoring model to validate that all the attributes present in original product data do not necessarily affect the product quality. The results produced by 500+ attributes can also be produced by using 92 carefully selected characteristics. This prediction model helped us to select these 92 attributes with expected value ranges.

Chapter 5

Software Architecture for Prediction Framework

Anomaly detection given the data at rest is a complex but well-studied topic. However, detecting anomalies for streaming applications remains an open problem. In that case, designing algorithm remains not the greatest hurdle, rather the infrastructure development to apply intelligent algorithms to moving data becomes the primary challenge. In the context of answering the second research question (RQ2), we investigate the system requirements to satisfy the needs for the industrial sensor data processing tool. To adhere these specifications, we impersonate a design and implementation of a proposal in a Big Data scenario using state-of-the-art streaming components. We devise an anomaly detection system using Apache Spark [39] based on Lambda architecture to provide a layered approach to process sensor data. Our system is based on two main principles: generality and scalability.

In this chapter, we will identify top-level system requirements for PQP tool in Section 5.1. Section 5.2 will compare different system architectures for such kind of big data processing. Then in Section 5.3 we will present a complete design of our proposed solution with integration of all the components. In Section 5.5 we will present an implementation of anomaly detection in one of the streaming application using user profile vector. Finally in Section 5.6 we will provide a quantitative evaluation of our proposed solution to give an overview of the processing capacity of the system.

5.1 System Requirements

This research is based on one use case of Philips production unit. However, the goal is to develop a software infrastructure that should be usable as ”software as service” for the other manufacturing industries. Hence, the system must exhibit to excel at a variety of real-time stream processing applications. Considering the goal for such generalized solution, we mainly focused on three principles to design this

software: generality, scalability, and decoupling. We outline six requirements for the quality prediction software as service.

5.1.1 Process moving Data without intermediate Storage

The first requirement for the processing unit is to process messages on the fly. The system must be able to perform data processing without having a costly storage operation in the processing path. This principle should remain true at the time of receiving the sensor data, and also during the intermediary processing as well. A storage operation costs a long and unnecessary latency in the overall processing operation that includes a disk write cost, committing a database and a log file writing. For an efficient stream processing applications, it is not acceptable nor necessary for such a time-intensive application.

5.1.2 Availability of Results

Processing large amounts of semi-structured data always take time, and there is invariably a delay associated between the point when data is gathered and visibility of results on the dashboard. This delay is mostly due to validation and confirmation of the results. However in the current case, being able to react immediately against some emergency pattern is more important than being 100 percent certain about the results. Hence, the system should be able to provide a speedy interface to generate results even with some compromises with precision.

5.1.3 Handling Delays and Process in order Data

In the traditional database for batch processing, data is always present, and it is accessible as per the desired order. However, in a streaming system as we have removed the database storage layer between data generation and processing units, so maintaining the missing, delayed or out of order messages can be an issue. The required infrastructure must make provision for handling data that is late or delayed, missing, or out-of-sequence.

5.1.4 Support of modern Data manipulation Libraries

Another fundamental requirement for production tool is to support a high-level language with built-in extensible stream-oriented primitives and operators such as SQL, streaming SQL and machine learning. The business logic that we present in chapter 3 and 4 is one kind of data modeling technique. For some other system, the business logic for the prediction can be even more complex and may require even more advanced data manipulation techniques, for instance, SQL support and machine learning. The system should have the ability to apply advanced libraries, and languages support on the moving data.

5.1.5 Data Filtering

Industrial sensors collect various types of environmental evidence in the production line, and all the recorded data may or may not be relevant and useful for some data modeling techniques. Filtering the interesting data and ignoring the messy or irrelevant data can be time-consuming. The data ingestion in our system must be smart enough to keep the data filtering decoupled from the stream processing engine. The processing component must be able to inject only those messages that are supposed to be processed to generate the desired output.

5.1.6 Decoupled Scalability

In the Big Data paradigm, distributed processing and scalability has become increasingly important and popular techniques due to the availability of low-cost commodity clusters with affirmative price-performance characteristics. The objective in designing the PQP tool is a scalable pipeline of different components combined with high decoupling. All the segments in the processing pipeline must of able to scale with elasticity. When heavy manufacturing industries use the system, the scalability is mandatory required. Different situations demand scalability of the various components. A spin-off for processing unit on multiple clusters is necessary for the compute-intensive jobs, especially for the use cases with little data ingestion but complex processing. Similarly, for systems with a huge volume of data generation but with simple data requires scalability only for data ingestion component.

5.2 Proposed Solution

To meet the requirement cultivated in Section 5.1, we explored different architectures for building big data processing pipeline. We mainly studied fully incremental architecture [32], used for twitter analysis; and Kappa architecture [44], used for smart cities IoT data management. However, we choose to design the system under the guidelines proposed in Lambda architecture [26]. The Lambda architecture espouses the same idea that we consolidate in the system requirement phase that a single application or service cannot be a viable solution for a Big Data problems with multiple inputs streams. Rather, various tools can be combined to build a processing pipeline. Within that pipeline, each layer should serve for a specific need. Nathan Marz presented this idea in his book, *Big Data Principles and Best Practices of Scalable Real-time Data Systems* [31]. The core definition we could find: *It provides a generic strategy to execute an arbitrary function on an arbitrary set of data with having an ability to return the results with low latency*. Lambda architecture is consist of three main components: a data source, a batch layer, and a speed layer as shown in the figure 5.1.
All data entering in the system is dispatched to both the batch layer and the speed layer for processing. The green lines in the diagram 5.1 represent the flow of the

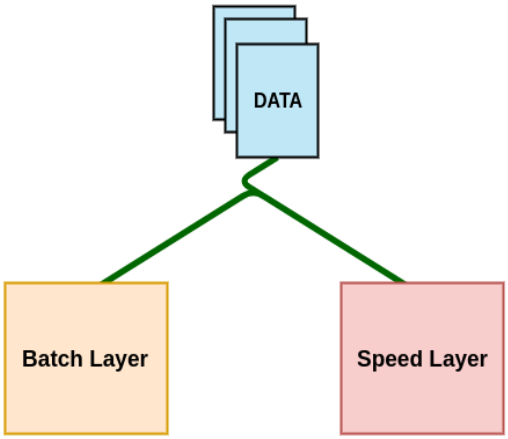

Figure 5.1: High-level overview of the Lambda architecture with two way flow of incoming data

data. We can see that data flow makes a Greek charter lambda, this the reason the architecture is famous a 'Lambda' architecture. Speed layer provides an active layer with freshness in the results for an alert system with the support of a quite extensive layer of batch processing.

5.3 Design of the System

Lambda architecture is not a solution rather; It is a software design pattern for big data processing that proposes a general direction to unify a real-time processing with a batch processing within a single framework. Fundamentally it is intended to improve the extract load and transformation (ETL) flow in the data warehouses to improve OLAP processing. In a data warehouse, newly arrived data takes much time to ingest into the master data set mainly because of several layers of ETL. To immediate incorporate the value in newly arrived data, the speed layer provides an initial estimate of the trends in newly arrived data before to passing it through the complete ETL pipeline. However, the core value of the output is generated in batch layer by the analyses on historical data. The streaming layer adds very little to the final output in serving layer. In the current study, we slightly modify the lambda architecture by amending the responsibility of layers to simplify and extend the capabilities of Lambda to make it fit to serve well for an online anomaly detection tool. In our current system, the core of the output must be generated from speed layer to get low latency response for immediate actions. Further, to help speed layer to monitor data without applying time-consuming, complex machine learning algorithms, batch layer helps speed layer to load an efficient detection model dynamically.
Figure 5.2 portrays an overview of the system we develop with all the components

by modifying the Lambda architecture. The figure represents the complete pipeline starting from the various sensor on the left-hand side to the anomaly detection results on the right-hand side. Sensor measurements are received from the temporary repositories and pushed into the dedicated queues for real-time consumption by the producer programs. To handle queues a message broker system, Kafka is integrated that takes care of maintaining the order of all types of sensor data into separate queues. On top of Kafka, a consumer program writes data into HDFS for permanent storage that we call master data. Both anomaly detection components the batch and the speed layers consume the data either in the real-time queue or from S3 storage, process it and push their respective outputs into the serving layer. Cassandra database represents serving layer. In next section, we explain the functionality of each component. We also discuss the design choices of the technologies for each component in the Lambda architecture.

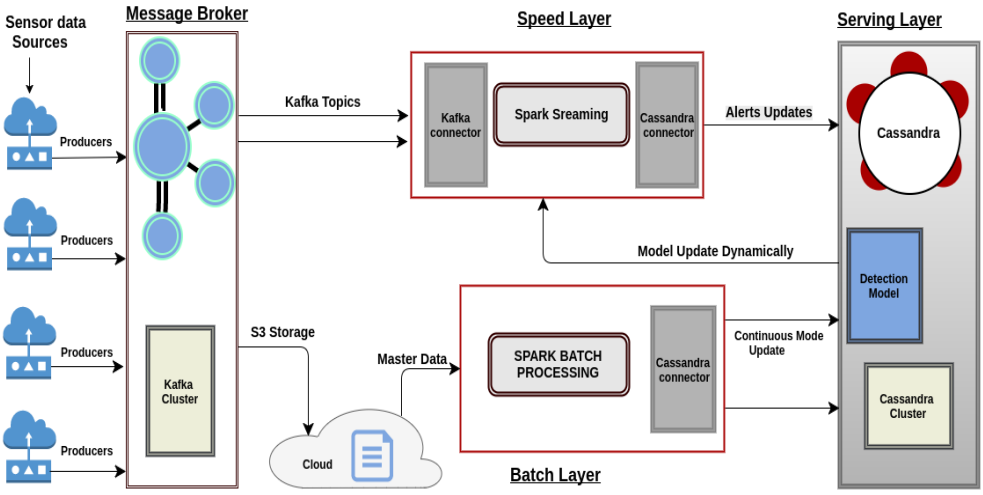

Figure 5.2: Big data processing pipeline with integration of receiving, processing and serving layers

5.4 System Components in Lambda Architecture

Multiple technologies are available to implement each layer. For each component, we select the most promising technology by considering the current application requirements. In this section, we briefly explain the functionality of each layer. Then a concise description of the selected technology with few arguments for preferring the particular choices over the others.

5.4.1 Message Broker Kafka

Apache Kafka [31] is an open-source distributed message broker system designed for handling Tara bits of streaming data. It is built to be fault-tolerant, high-throughput, and to be horizontally scalable. It allows to handle geographically distributing data streams and translates messages from the messaging protocol of the data producer to the messaging protocol of the processing application. The main abstraction in Kafka is a structured commit log of updates: Producers are running on top of production lines sensors, send a stream of messages which are appended to commit log. Then, any number of consumers can continuously stream these messages off the tail of the log with a millisecond latency. Each of these data consumers has its position in the log and advances independently while consuming the messages. This allows in-ordered message deliver to each consumer. Kafka is built as a synchronous distributed queuing system. Data is replicated and partitioned over the clusters that can expand and shrink transparently to the applications using the broker system. Being persistence is another essential feature of the Kafka. The producers on the production lines can go down in off hours or for maintenance and come back hours later trusting that all changes have been safely persisted in the up-stream Kafka cluster.

5.4.2 Batch Layer

The batch layer a core component of the Lambda Architecture is high latency layer that performs processing in the background on the full set of data known as master data. It enables to perform deeper analytics and heavier computations on the master data. A formula

$$BatchLayer \Rightarrow Function(Master\ Data)$$

best describes the functionality of batch layer.
The batch layer has no time constraints to produce its outputs. Also, it is shaped by the traditional approach of batch data processing and can use all the classical data modeling languages which enable our system to use the power of all the high/low-level data manipulation techniques and libraries including MySQL, Hadoop MapReduce or even multi-phase data processing. As a result, it provides very accurate, stable and reliable results. We integrated Apache spark for the implementation of batch layer.

Apache Spark

Apache Spark is a general purpose large-scale parallel data processing engine. It offers several advantages over MapReduce framework, which is the pioneering framework for big data processing. The bottleneck in MapReduce is it process data in multiple phases. It transfers the data after each stage of intermediate processing which involves an expensive disk operation. This multi-pass processing

makes it unsuitable for our time-sensitive streaming application. Alternatively, In addition to the properties of MapReduce, spark provides a faster in-memory execution, especially for cases where multiple passes are necessary for the same data such as when transforming, mapping and reducing operations are applied to the streaming data. Spark does not shuffle the data over the disk instead save the intermediate results within memory. This in-memory processing policy makes spark perfect choice for our anomaly detection framework. Sparks primary abstraction are resilient distributed datasets (RDDs) [13]. RDDs are the data handler in spark. Data is loaded into the form of RDDs, and used for distributing the data on multiple machines for parallel processing and remains as immutable abstractions of distributed data throughout the application lifecycle. All the operations on it are organized in a Directed Acyclic Graph (DAG) to perform transformations on top of the data. Spark evaluates all the transformations on the DAG lazily, and data operations are only applied when data shuffling becomes necessary. This approach provides spark core a significant leverage to schedule processing tasks of DAG in the most optimal way. DAG also keeps all the data transformation history which helps in state recovery in case of failure without expensive checkpointing. Here we provide a just brief overview of spark, for details we encourage to read details of spark here [39]. Broadly, the following properties of spark help us to chose it as execution engine in our application.

1. Execution Speed: Spark can execute batch processing jobs 10 to 100 times faster than MapReduce. Although spark provides a fast in-memory processing ability that doesnt mean it lags behind when disk operations are involved in writing or fetching the data. Spark is the world record holder for large-scale on-disk sorting [39].

2. General purpose Processing Engine: Apache Spark is general purpose processing framework. Most of the other big data frameworks are designed mainly considering specific domain requirements. For example, Strom is flexible only for streaming, Mahout for machine learning and Hive for SQL type processing. Spark provides a core framework for parallel processing; then it unifies all kind of big data processing tools including bash processing, streaming, machine learning and graph processing on top of spark core. This unification makes spark a perfect choice in our case. These tools are available in the form of APIs. These APIs are easy to use and come with higher level libraries that support SQL queries and machine learning on data streaming.

3. Unified Resource Scheduling: In a parallel processing framework, cluster management is a significant concern. Spark can run on top of Hadoop, making use of YARN cluster manager or with the help of Mesos resource scheduling. However, it can also be run independently without the support of explicit cluster manager; Spark offers standalone and even local mod deployment options. The freedom of

cluster management and resource scheduling gives flexibility for deploying PQP on any cloud infrastructure.

4. Programming Language Choices: Spark does not tie down to a particular programming language. Spark support five programming languages including Scala, Java, Python, Clojure, and R. Any of the languages can be chosen from the list. That property is also an essential concern, as PQP is designed to be deployed as software as service. In future, there can be a particular domain task that may get a solution in another language, so the choice of the programming language makes it flexible.

5. Active and expanding user community: Apache Spark initially started as a research project at UC Berkeley in the AMPLab. The code base along with the design was later donated to the Apache Software Foundation since then it is open source and has been one of the most active projects under Apache software foundation. It has over 40376 commits by more than 1513 regularly contributors [3]. Such active user community has led to a stable release of Spark 2.2 in early 2017. All of the major releases are quite stable and being used by major big data solution providers in including IBM, Amazon, Data Bricks and much more; please refer the link [2] for a complete list of companies using a spark. Spark has developed a worldwide community both in academia and industry, which is on the rise.

5.4.3 Speed Layer

The choice of using a stream processing framework is due mainly to the requirement of a maximum freshness of the results. The speed layer consists of micro-batches running every minute. Each micro batch reads the events directly from Kafka, performs the business logic for outlier detection and then stores pushes the results on the serving layer. There are many choices to select the framework for speed layer including Apache Flink, Apache Spark, and Apache Strom. However, we selected Apache Spark Streaming API for our application.

Spark Streaming

Spark includes a streaming library called Spark Streaming that can offer all the power of spark batch processing in the streaming mod by distributing the data into micro batches. Streaming API introduces another core abstraction, discretized streams (DStreams) [39] to give an abstraction of processing moving data. The streaming data is received in the form of DStreams. DStreams are continuous sequences of RDDs. One RDD contains all the data belonging to one micro-batch. Most of the functions available for RDDs in the Spark batch mode are also available for DStreams in the streaming mode. The only difference is DStreams need to be transformed first to uncover the Dstream wrapper; then native RDDs are available.

Further, any RDD operation can be applied abstracting away the special processing of data during streaming.
The same execution engine is used both in spark batch and spark streaming modes. The execution engine obtains one RDD from the DStream per micro-batch time interval and applies the transformations directly to the RDD. Since both RDDs and DStreams are immutable, the output of applying a transformation to a DStream is a new DStream representing a continuous sequence of transformed RDDs. Concerning the development effort, any part of the code can be reused between the batch and streaming modes. That makes Apache Spark well-suited for implementation of Lambda architecture where both batch and streaming data should be jointly processed.

5.4.4 Serving Layer Casandra

The responsibility of both batch and speed layer is to produce output using the given functions to them respectively. They are not meant to serve the results to the application. Speed layer perfume computations just on a fraction of data at a time and cannot answer application queries. Though batch layer performs computations on master data, however computing ad-hoc-queries on that layer takes much time. Hence responsibility of serving results to the application is dedicated to serving layer. This layer cache the output from batch-layer, so results are immediately available in the serving layer to answer the ad-hoc queries. Batch layer periodically re-compute data and the cached results are refreshed in the serving layer accordingly. Similarly, speed layer pushes its output to serving layer and on receiving alerts, the serving layer can trigger to the application. For the serving layer, we integrate Casandra database.

5.5 Implementation

We provide an implementation for anomalies detection using the profile vector we develop in the batch layer 4.2. This implementation is based on the event vise comparison of each data record representing the state of the product with the profile vector. This method is a variation of Moving Average Model for time series analysis [5]. In the moving average model, a data event is considered outlier if it deviates from a continuous mean value. In our case, we do not calculate moving average. Instead, we compare each record with an expected value of quality parameter identified by batch layer. The method we propose applicable for multidimensional and categorical data. Besides that, our solution does not make any assumptions about the data distribution and doesn't necessitate complex data preprocessing before to apply in the streaming form. Moreover, the strength of the method is, it is Big Data friendly. It can cope with large volumes of data at rest as well as for the data on the fly.

5.5.1 Profile Vector Detection

To simulate the environment for sensor data generation similar to assembly line production, we write an artificial sensor data producers to push data on Kafka broker continuously with different topics. The spark streaming application receives 'inline' topic data in the form of DStream. A complete data pipeline for computing profile vector technique is shown in the figure 5.3.

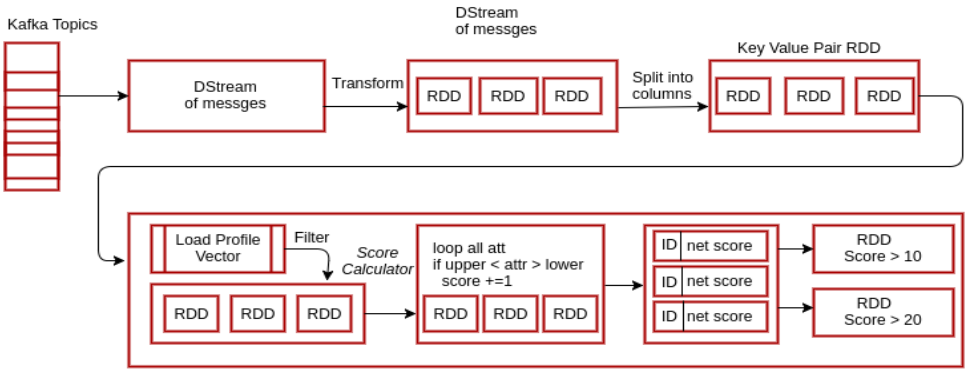

Figure 5.3: DStream transformations to detect anomalous data events by a comparison with profile vector

First of all, a micro-batch is fetched in the form of a streaming window; the data in that window is wrapped in a DStream abstraction of spark. Then we start applying different operation on the DStream data to get the desired output. First, the input DStream is mapped to a new DStream to separate the data part from the metadata of DStream message. The data part consists of product data in an RDD. Then the new DStream is transformed to apply operation on internal RDDs. In that transformation, each RDD (which represents a full state of a single product) is charted into a key-value pair (key, state parameters). Next, we dynamically load the profile vector from the serving layer. This profile is an output of the batch layer, which performs analysis on master data and keeps on updating this vector. That profile vector consists of the following information. (i) List of critical quality attributes that will be used to test product quality. (ii) Expected mean values of each attribute. (iii) The standard deviation for each attribute from arithmetic mean, (iv) and a marginal constant. After loading detection model (profile vector) dynamic, each RDD is passed through filter transformation to filter attributes only indicated in the profile vector list. Now the DStream is consist of a lightweight filtered RDDs having the attributes specified by the Profile vector. Next, to that, a final transformation is applied on DStream, where a user-defined function *anomalyScoreCalculator* is applied on each RDD. That function calculates upper and lower limits and checks whether each attribute of RDD falls within that boundary, if the parameter value is out of the bounds then it increments anomaly score for each product. After full comparison of one RDD with the profile vector,

each RDD has a sum of total anomaly score. That is a count of the total deviations of quality parameters than the expected values. Then, the RDD is returned in the form of a key-value pair, at this time the key is a unique id of the product and value is the total anomaly score *(Product ID, anomaly score)*.

Further, we apply two filters. First, the number of products having anomaly score higher than the first threshold and second the products having anomaly score higher than a second threshold. The threshold value is adjustable, and here we have 45 as first and 65 as second threshold values. A product can have maximum 92 anomaly score according to our current model, but that does not matter. The model can be updated at any time, and critical level of anomaly score can also change. We provide this information in the profile vector. Finally, the output of each DStream calculation is pushed to serving layer, and a new DStream is fetched from Kafka.

5.6 Evaluation

To demonstration the system capacity we made an experimental set up to show some results. The demonstration cluster consists of 4 virtual machines, one is dedicated to Kafka Cluster, one for stream processing, one for batch processing. The fourth machine is dedicated for serving layer, databases and web server are configured with that to aggregate and display results to the client dashboards. Following system configurations are same for all machines:

System Property	**Configuration**
Processor	Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
System Memory	8GB
Hard Disk Space	32 GB SCSI Disk
Network	Ethernet interface, capacity: 1Gbit/s
Programing Environment	Oracle Java 1.8.0, Scala 2.11.11, Sbt: 0.13.11
Resource manager	Zookeeper resource manager version: 3.4.10
Queuing System	Kafka version: 0.11.0

Table 5.1: Column names with attribute description of in-line assembly data

We evaluate following properties:

1. **Throughput**: This is the total throughput of the system. It is the amount of time required to process 100 Thousands data events, the metric unit seconds. To estimate throughput time, we simply calculate start and ending time of the 100 Thousands product data. This time includes the time required to push data on Kafka, network time to fetch product from one layer to another, and the processing time to apply prediction model on each product.

2. **Compute time**: This is the time required to process one product data through all the layers, the metric unit is milliseconds. To measure the compute time we first write a time stamp on each product when it is pushed to Kafka and second-time stump when streaming application is about to write product results on the serving layer. Then in a separate script, we calculate the average time difference between both time sumps.

Input Rate(events per sec)	**Products per window**	**Compute time per window in seconds**	**Per product Compute Time in millisecond**	**Throughput Time in second**
50	1000	11.08	10.40	1700.0
100	2000	11.50	10.11	875.7
200	4000	13.07	10.19	450.3
350	7000	14.00	11.14	291.6
400	8000	14.42	12.51	281.2
600	12000	15.21	14.36	267.6

Table 5.2: Comparison of Compute rate and Throughput time against different data generation rate

Table 5.2 shows the execution results performed on 100 Thousands. In all experiments, the length of the streaming window is 20 seconds. Which means streaming application fetches data from the Kafka queues each after 20 seconds and we test our system with different data generation rates The first column in the table is the rat of data generation per seconds; the second column represents the total number of products captured in each streaming window with the given data generation rate. The third column depicts the total compute time to process one streaming window. The fourth column corresponds the compute time per product represented in milliseconds, and the fifth column is the overall throughput of the system.

It can be observed from the table 5.2 that throughput of the system increases as the data generation rate increases. Which means that in the given experimental setup, the processing system has a more extensive capacity to process streaming events than the data generation source. We observed that the compute time per window increase when the data generation rate increases. This time expansion is because, with high data generation rate, the total number of products increase in each streaming window, which takes more time to process each window but in that case the total numbers of windows (to process full set of data) decreases. As result of this, we get high throughput (less time to process 100 thousand products).

5.7 Conclusion

In this chapter, we presented a modified concept of Lambda architecture to create a pipeline of sensor data processing for machine learning into two different layers. The goal of this proposal to uses two different processing layers for the learning and prediction phases of machine learning. That is to balance the latency, throughput and scaling for real-time prediction results. We interconnected the state-of-the-art Apache Spark batch data processing with streaming API for real-time sensor data analysis in an efficient way. The interconnection allows us to process the larger set of historical data to develop a prediction model by understanding critical quality parameters. Then the streaming API is used to analyze sensor data on the fly using simple comparison with the prediction model generated by the batch layer. That gives quality prediction results in near real-time without consuming time required for learning phase. We also integrate data ingestion layer which takes very little time to serve the data to the processing unit; as a result, we can process sensor data as soon as it is generated from the source without any delay.

Chapter 6

Conclusions and Future Work

This chapter summarizes the thesis, discusses our findings while answering the research questions of this thesis, points out limitations of the current work and also outlines directions for the future research. The chapter is divided into two parts. Section 6.1 presents a discussion on the main conclusion in our research. Then in Section 6.2, we describe future work.

6.1 Conclusions

In this thesis, we presented a pilot exploitation of stream processing systems for real-time quality prediction using assembly line sensor data for the manufacturing industries. We used anomaly detection approach to filter outlier in the streaming data. Most of the anomaly detection methods require complex machine learning algorithm to develop a prediction model that takes much time during the learning phase. That is not suitable for real-time quality prediction due to time constraints. In our research, we were able to interconnect state-of-the-art Apache Spark batch data processing with streaming API for real-time sensor data analysis in an efficient way. The interconnection allows us to process the larger set of historical data to develop a prediction model by understanding critical quality parameters and then use this model to identify out of quality products on the fly. We mainly addressed two research question in this thesis. The conclusion of both questions are illustrated below.

In our research to answer first research question (RQ1:*developing a suitable approach for quality prediction by utilizing the assembly line sensor data*) we concluded that anomalies detection from the sensor data collected during the production process could play an important role in product quality prediction, but finding these anomalies in assembly line sensor data is difficult task. Classification methods can be used to identify anomalies in general. However, due to unavailability of labels to differentiate the data events into different classes, these methods were not applicable in current context. On another hand, statistical methods

can also be proposed for such anomalies, but due to an imbalanced distribution of sensor data generation, most common statistical learning algorithms do not work as well in our case. Therefore to answer first research question we used an unsupervised clustering algorithm to identify anomalies from entirely unlabeled, imbalances and high dimensional data. In that part, our investigation mostly focused on developing a prediction model by distributing the sensor data into different classes. We concluded from our cluster analysis experiments that the majority of the normal data events fall in a dense region and also the size of these clusters is always more significant than the clusters having anomalous data. This observation helped us to study the properties of normal and faulty product's data separately, and we benchmarked the acceptable values of different critical attributes that play a vital rule in deciding the product quality. One of our other finding in sensor data analysis is that the sensor collects abundant of information about the under construction products and this can include much redundant information. For efficient working of machine learning algorithms, data must be refined systematically before to use it.

In the last part of the thesis we addressed the second research question (RQ2: *designing a reusable layered system architecture to analyze assembly line sensor data on the fly in the form of live streams*). During that investigation, we concluded that building a well-designed, reliable and functional sensor data processing application is fundamentally different and challenging than developing tradition software solutions. To cater a variety of requirements that includes sensing data from multiple sources, the dimension reductions and to apply learning model dynamically cannot be handled in a single stand-alone application. For this, a complete data pipeline with multiple components is required. In that processing pipeline separation of concern is the most important factor. Each component must have its well-defined functionality, and it should be able to scale up and down individually to perform its role within its time constraints. For example, in processing part, we use two layers for two phases of machine learning algorithms, i.e., the learning and prediction phases. To train the prediction model, we use a batch layer and then the profile vector comparison technique provides a very efficient way to detect anomalies in unseen data in streaming layer. As a result of this, just applying a few comparisons anomalies can be detected directly without time-consuming learning phase. It makes the system a suitable choice for real-time quality prediction.

The involvement of many layers in the system makes it difficult to mentally visualize and evaluate, but end-to-end connectivity gives a good understanding of the importance of the system. It is also worth to mention that the proposed system is a complicated architecture. The complexity of the code can be a 2 to 3 times more than the traditionally designed data processing tools. Furthermore, It demands much more effort for code base management as each of the layers, i.e., Kafka, streaming, batch and speed layer will have a separate code base and

most probably in the different programming languages as well. Moreover, many hardware resources are also required that can either be achieved with virtualization or by cloud base instances. We recommend that this architecture should only be adopted if the organization has long-term project planning and development budget allows acquiring more resources.

6.2 Future Work

The purpose of this MSc thesis research was to develop a test bed product to understand the needs of a quality prediction system for manufacturing industry and then develop a layered architecture to handle sensor data in real time. With an overall understanding of all stack holders of PQP product, the proposed architecture for streaming is capable enough to meet the requirements of Philips shaving production unit and in general as well. There is still few pending task to improve the proposed solution for both research question. Few of them are listed below.

6.2.1 Prediction Model

1. During the first phase of this research, there was no knowledge sharing mechanism with Philips to test the results. Now after developing processing framework, results can be tested in the production environment. Hence, the first future item is to test the prediction results to tune the models.

2. Currently we have configured only one streaming layer to process the moving data. However, our system architecture allows incorporating more than one streaming application without bearing delay. Next, we want to integrate multiple prediction algorithms in separate layers to predict from a different prospect and then define a scoring metric to select the prediction results of each algorithm.

3. We also have data about which machine/tool is being used at the different time slots. We want to introduce another streaming layer to associate time slice of each tool used on the individual product. That will help to indicate a sub-station in the assembly line that is responsible for producing faulty products.

4. For now, we are using 92 preselected critical variables. In future, the plan is to apply supervised machine learning; first to select X set of critical variables dynamically. Then use these X variables instead of always using the same set of variables.

6.2.2 Processing Framework

1. The proposed infrastructure will be deployed on cloud environment with separate instances of Kafka server, streaming and batch layers.

2. A web-based dashboard system will be developed to integrate serving layer of proposed solution with Casandra as the backend. These dashboards can also be configured on smart phones to give rapid alerts system.

3. In the current implementation, the prediction model is updated whenever a new streaming window is fetched. That is unnecessary because the frequency of prediction model updates cannot as frequent as the speed of new streaming windows. In future, the read functionality of prediction model can be improved so that streaming layer will update prediction model after long intervals or even only at the time when the batch layer will update the prediction model. This will improve the overall throughput time of streaming application.

Bibliography

[1] Demystifying bigdata: A practical guide to transforming the business of government. *TechAmerica Foundations Federal Big Data Commission*, 2012.

[2] The apache software foundation. *http://spark.apache.org/powered-by.html*, 2017.

[3] Black duck software, inc. *https://www.openhub.net/p/apache-spark*, 2017.

[4] Aeyels and Dirk. On the dynamic behaviour of the novelty detector and the novelty filter. *Analysis of Controlled Dynamical Systems-Progress in Systems and Control Theory*, 8:1–10, 1991.

[5] Jyoti Bansal. Time series anomaly detection using multiple statistical models. *Computer Associates Think, Inc.*, 20, 2007.

[6] Barbara, Daniel, Ningning Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. *SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*, 2001.

[7] V. CHANDOLA, A. BANERJEE, and V. KUMAR. Anomaly detection: A survey. *ACM Computing Surveys*, 2009.

[8] M. Chen, Y. Liu, and S. Mao. Big data: a survey. *Mobile Networks and Applications*, 19(2):170–210, 2014.

[9] P Chen and J. Luo. Data detection and pattern recognition on fms control charts. *International Conference on Industrial Technology*, 36(2), 2008.

[10] M. Cooper and P. Mell. Tackling big data [online]: Available: http://csrc.nist.gov/groups/sma/forum/documents/june2012presentations/ 2012.

[11] Crook, Paul, and G. Hayes. A robot implementation of a biologically inspired method for novelty detection. *Proceedings of Towards Intelligent Mobile Robots Conference*, 2001.

[12] Laney D. 3-d data management. *Controlling data volume, velocity and variety. Application Delivery Strategies by META Group Inc*, February 2001.

[13] Addison JF Dale, Stefan Wermter, and John MacIntyre. pages 976–981, 1999.

[14] Duda, Richard, Peter E. Hart, and D.G. Stork. Pattern classification. john wiley and sons. *Procedia Engineering*, 2012.

[15] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 255–262, 2000.

[16] Augusteijn M. F. and B. A. Folkert. Neural network classification and novelty detection. *International Journal of Remote Sensing*, 23(4):2891–2902, 2002.

[17] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,. *IDC iView, IDC Anal. Future*, 2012.

[18] Ghosh, Anup K., Aaron Schwartzbard, and Michael Schatz. Using program behavior pro les for intrusion detection. 20.

[19] M Goldstein and S Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS*, 2016.

[20] Vehbi C. Gungor and Gerhard P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10), 2009.

[21] X. Guo, F. Wang, and M Jia. A stage-based quality prediction and control method for batch processes. *In Proceeding of the International Conference on Machine Learning and Cybernetics*, 2005.

[22] Hawkins and Simon. Outlier detection using replicator neural networks. *DaWaK*, 2454, 2002.

[23] ANSCOMBE F. J. Rejection of outliers. *Technometrics*, 2(2):123–147, 1960.

[24] Jagota and Arun. Novelty detection on a very large number of memories stored in a hopfield-style network. *IEEE Neural Networks IJCNN-91-Seattle International Joint Conference*, 2, 1991.

[25] Japkowicz, Nathalie, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. *IJCAI*, 1, 1995.

[26] M Kiran, P. Murphy, I. Monga, J Dugan, and S. S. Baveja. Lambda architecture for cost-effective batch and speed big data processing. *Modeling and Processing for Next-Generation Big-Data Technologies*, 2011.

[27] H. Lau, G. Ho, K. Chu, W. Ho, , and C. Lee. Development of an intelligent quality management system using fuzzy association rules. *Expert Systems with Applications*, 36(2):18011815, 2009.

[28] LeCun and Yann. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 1990.

[29] Y Lei and L Huan. Feature selection for high-dimensional data: A fast correlation-based filter solution. *ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE*, 2003.

[30] Kano. M and Nakagawa. Y. Data-based process monitoring, process control, and quality improvement. *Recent Developments and Applications in Steel Industry Computers and Chemical Engineering*, 32(1):12–24, 2008.

[31] Nathan Marz. Big data: Principles and best practices of scalable realtime data systems. 1, 2013.

[32] G. Mishani., J. Dalton, Z. Li, and A. Sharma. Fast data in the era of big data: Twitter's real-time related query suggestion architecture. *ACM SIGMOD International Conference on Management of Data*, pages 1147–1158, 2013.

[33] Sathyan Munirathinam and Balakrishnan Ramadoss. Predictive models for equipment fault detection in the semiconductor manufacturing process. *IACSIT International Journal of Engineering and Technology*, 8(4), 2016.

[34] Gartner IT Glossary (n.d.). [online] available: http://www.gartner.com/it-glossary/big-data/. 2012.

[35] Barson P. The detection of fraud in mobile phone networks. *Neural Network World*, 6(4):477–484, 1996.

[36] Tan. Pang-Ning, M. Steinbach, and V. Kumar. Introduction to data mining. *Procedia Engineering*, 1, 2005.

[37] L. Portnoy, E. Eskin, and S Stolfo. Intrusion detection with unlabeled data using clustering. *ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[38] A. PrabakarMuniyandi, R.Rajeswari, and R.Rajaram. Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm. *Procedia Engineering*, 30:174–182, 2012.

[39] Josh Rosen, Ion Stoica, Patrick Wendell, Reynold Xin, and Matei Zaharia. Scaling spark in the real world: Performance and usability. *Databricks Inc. MIT CSAIL*, 2015.

[40] Roth and Volker. Outlier detection with one-class kernel fisher discriminants. *Advances in Neural Information Processing Systems*, 2005.

[41] Roth and Volker. Kernel fisher discriminants for outlier detection. *Neural computation*, 18(4):942–960, 2006.

[42] Schlkopf and Bernhard et al. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[43] De Stefano, C. Sansone, and Mario Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

[44] M. StrohbachEmail and Z. G. Akiva. Towards a big data analytics framework for iot and smart city applications. *Modeling and Processing for Next-Generation Big-Data Technologies*, 4:257–282, 2011.

[45] Sykacek and Peter. Equivalent error bars for neural network classifiers trained by bayesian inference. *ESANN*, 1997.

[46] V. Vapnik and C. Cortes. Support-vector networks. *Kluwer Academic Publishers, Boston*, 1995.

[47] V. Vapnik and C. Cortes. Mining quantitative association rules in large relation-altables. *ACM SIGMOD international conference on Management of data*, (1-12), 1996.

[48] W. Wong and A. Moore. Bayesian networks anomaly pattern detection for disease outbreak. *Procedia Engineering*, 30:174–182, 2012.