UNIVERSITY OF THESSALY

DEPARTMENT OF MECHANICAL ENGINEERING

Diploma Thesis

Cycle Time Prediction in the Wafer Test Fab of a Semiconductor Manufacturing Plant using an Artificial Neural Network Model

by

GEORGIOS LOGOTHETIS

Submitted in partial fulfillment of the requirements

for the Diploma in Mechanical Engineering

February 2020

The Diploma Thesis of Georgios Logothetis has been examined and approved by a three-member Examination Committee as satisfactory for the thesis requirement for the Mechanical Engineering degree.

Approved by:

First Examiner:	Dr. George Liberopoulos
(Supervisor)	Professor, Department of Mechanical Engineering, University of Thessaly
Second Examiner:	Dr. Dimitrios Pandelis
	Associate Professor, Department of Mechanical Engineering, University of Thessaly
Third Examiner:	Dr. Georgios K.D. Saharidis
	Assistant Professor, Department of Mechanical Engineering, University of Thessaly

Acknowledgements

First of all, I owe special thanks to my Supervisor, Professor George Liberopoulos. My early involvement in [redacted]-funded project [redacted], while being on my 3rd year of studies, as well as the chance to be an intern at our project collaborators, KIT and [redacted], were some of the opportunities he generously offered me. These chances to perform undergraduate research and work for such partners, provided me with valuable lessons and insights for my career.

I also would like to thank Professor Liberopoulos' PhD student and a personal friend of mine, Michalis Deligiannis, for his scientific support throughout my work.

In addition, I am grateful to my intern supervisor at [redacted] plant in [redacted], [redacted], for the precious suggestions in the direction of my thesis topic. During my internship at [redacted], I had the opportunity to work as a shift employee in a Wafer Test Fab and gain valuable insights about fab operations. Many thanks to the [redacted] people who helped me understand the physical system that was modeled in this dissertation.

Last but not least, I thank my family, which stands by my side in all my endeavors.

Abstract

The large production scale [1], high stochasticity and complexity [2] that characterize the manufacturing processes of semiconductor plants, make the analysis and modelling of the physical system very challenging. Products that may re-enter the same machine types through the course of their production, as well as the routing flows and the job sizes that dynamically change [3] during the process, make it hard to gain valuable insights and Key Performance Indicators (KPIs) of the production process.

One of these KPIs is the Cycle Time (CT) of a product through any fab of the plant or, more importantly, through the entire plant.

Knowledge of CT information plays a very important role in semiconductor manufacturing industries. It allows them to promise accurate delivery times to their customers, and hence reduce penalties for lateness, as well as handle their inventory in a just-in-time way.

In order to achieve an accurate prediction of this important KPI, we must take into account the factors that affect it, such as the WIP of the fab, the utilization of the equipment, the size of queues before the equipment, etc. [4]

In the literature, several model-based and data-based approaches for predicting CT are suggested. Among the data-based approaches are the methods of Multiple Linear Regression (MLR) and Artificial Neural Networks (ANNs) [5]. Data-driven methods, enabled by recent technological advances, are particularly promising for accurate CT prediction, as they are based on historical data.

In this thesis, we present an ANN-based method for CT prediction. Roughly described, an ANN treats the system as a black-box, using massive input data and a properly designed forecast model to predict CTs, without much knowledge of the structure of the system.

We applied this method to the Wafer Test Fab of the [redacted] semiconductor plant in [redacted]. To do this, several steps had to be taken. First, we had to comprehend the physical system of the semiconductor manufacturing plant, and more specifically the Wafer Test Fab. Then, we had to identify and define the most important factors that potentially influence CT. These comprised the input factors of the ANN model. Next, we had to manipulate the raw data in order to form the proper aggregated inputs. This step turned out to be the most challenging one, because it involved dataset transformation through a very concrete data analysis. Finally, we developed the ANN model, using MATLAB's Neural Network Toolbox (currently, Deep Learning Toolbox) [6]. In the end, we evaluated the results and the model efficiency, and we gave directions for improvements. The results that we obtained were reasonable and encouraging, suggesting that our implementation approach is reliable and promising.

Περίληψη

Η ανάλυση και η μοντελοποίηση του φυσικού συστήματος των εργοστασίων παραγωγής ημιαγωγών είναι ιδιαίτερα σύνθετη. Ο λόγος έγκειται στο γεγονός ότι αποτελεί μία παραγωγή μεγάλης κλίμακας [1], όπου οι διαδικασίες της χαρακτηρίζονται από υψηλή στοχαστικότητα και πολυπλοκότητα [2]. Τα προϊόντα μπορούν να εισέλθουν στον ίδιο τύπο μηχανής περισσότερο από μία φορά κατά τη διάρκεια της επεξεργασίας τους. Επίσης, οι αλλαγές στις διαδρομές και τα μεγέθη τους είναι δυναμικές [3], και ως εκ τούτου καθιστούν δύσκολη την απόκτηση πληροφοριών για Βασικούς Δείκτες Απόδοσης της παραγωγικής διαδικασίας.

Ένας από τους Βασικούς Δείκτες Απόδοσης είναι ο χρόνος που δαπανά ένα προϊόν κατά την παραγωγή του, είτε σε έναν τομέα του εργοστασίου, είτε συνολικά σε όλο το εργοστάσιο.

Η γνώση του χρόνου παραγωγής (*Cycle Time* – CT) διαδραματίζει σημαντικό ρόλο για τις βιομηχανίες κατασκευής ημιαγωγών. Τους επιτρέπει να υπόσχονται ακριβείς χρόνους παράδοσης των προϊόντων στους αγοραστές, και κατά συνέπεια μειώνουν τα κόστη που προκύπτουν από τυχόν καθυστερήσεις. Επίσης, τους δίνει τη δυνατότητα να διαχειρίζονται τα αποθέματά τους με τρόπους *just-in-time*.

Η πρόβλεψη του παραπάνω δείκτη, εξαρτάται από παράγοντες, όπως ο αριθμός των εργασιών σε εξέλιξη (*Work In Process* – WIP) στο εργοστάσιο, η απόδοση των μηχανών, το μέγεθος των ουρών πίσω από κάθε μηχανή κ.α. [4]

Στη βιβλιογραφία προτείνονται διάφορες προσεγγίσεις για την πρόβλεψη του χρόνου παραγωγής. Μεταξύ αυτών είναι οι μέθοδοι πολλαπλής γραμμικής παλινδρόμησης (*Multiple Linear Regression* – MLR), η προσομοίωση της παραγωγής και τα τεχνητά νευρωνικά δίκτυα (*Artificial Neural Networks* – ANN) [5]. Τα μοντέλα που η ανάπτυξή τους βασίζεται σε ιστορικά δεδομένα, δύνανται να προβλέψουν αποτελεσματικά τέτοιους χρόνους.

Σε αυτή τη διατριβή, παρουσιάζουμε μια μέθοδο που βασίζεται σε τεχνητά νευρωνικά δίκτυα για την πρόβλεψη του χρόνου παραγωγής. Περιληπτικά, ένα τεχνητό νευρωνικό δίκτυο αντιμετωπίζει το σύστημα ως μαύρο κουτί, χρησιμοποιώντας μεγάλο αριθμό δεδομένων για τις μεταβλητές εισόδου και ένα κατάλληλα σχεδιασμένο μοντέλο πρόβλεψης για την εκτίμηση των μεταβλητών εξόδου, χωρίς γνώση της δομής του φυσικού συστήματος.

Πιο συγκεκριμένα, εφαρμόσαμε την παραπάνω μέθοδο στον Τομέα Δοκιμών Δισκετών/Πλακιδίων Ημιαγωγών (*Wafer Test Fab*) της βιομηχανίας παραγωγής μικροεπεξεργαστών [redacted] στο [redacted]. Για την επίτευξη του συγκεκριμένου στόχου, έπρεπε πρώτα να προηγηθούν κάποια σημαντικά βήματα. Αρχικά, έπρεπε να κατανοήσουμε το φυσικό σύστημα του εργοστασίου, και ειδικότερα τον Τομέα Δοκιμών των Δισκετών Ημιαγωγών. Επιπλέον, έπρεπε να εντοπίσουμε και να καθορίσουμε τους σημαντικότερους παράγοντες που πιθανώς επηρεάζουν το χρόνο

παραγωγής. Αυτοί οι παράγοντες θα αποτελέσουν και τις μεταβλητές εισόδου του τεχνητού νευρωνικού δικτύου. Στη συνέχεια, έπρεπε να διαχειριστούμε και να επεξεργαστούμε τα ακατέργαστα δεδομένα που λάβαμε από την παραγωγή, ώστε να διαμορφώσουμε τις κατάλληλες μεταβλητές εισόδου. Αυτό το βήμα αποδείχθηκε το πιο δαπανηρό χρονικά, εφόσον απαιτούσε μετασχηματισμό των δεδομένων μέσω της κατάλληλης ανάλυσης των αρχικών. Τέλος, αναπτύξαμε και εκπαιδεύσαμε το μοντέλο του τεχνητού νευρωνικού δικτύου χρησιμοποιώντας το εργαλείο “Neural Network Toolbox” της MATLAB (πλέον, “Deep Learning Toolbox”) [6]. Εν κατακλείδι, αναλύσαμε και αξιολογήσαμε τα αποτελέσματα και την απόδοση του μοντέλου και παρείχαμε κατευθύνσεις για μελλοντική έρευνα επί του θέματος. Αξίζει να σημειωθεί πως τα αποτελέσματα που λάβαμε ήταν λογικά, υποδηλώνοντας ότι η εφαρμογή μας είναι αξιόπιστη και ιδιαίτερα χρήσιμη.

Contents

CHAPTER 1 Introduction 11

1.1 Motivation and Background 11

1.2 Literature Review 12

1.3 Thesis Outline 13

CHAPTER 2 Semiconductor Manufacturing 15

2.1 Manufacturing Process 15

2.2 The Wafer Test Fab 16

CHAPTER 3 Data Analysis 18

3.1 Description 18

3.2 Data Validation & Preprocessing 19

3.3 Data Analysis Strategy 22

3.4 SQL Code Description 25

3.5 MATLAB Code Description 26

CHAPTER 4 Artificial Neural Network (ANN) Model 32

4.1 Introduction to ANNs 32

4.2 ANNs in Manufacturing & Competitive Advantage 37

4.3 ANN Model (NN-Tool) 37

CHAPTER 5 Results 47

CHAPTER 6 Summary and Future Directions 52

BIBLIOGRAPHY 53

APPENDICES 55

APPENDIX 1: Code of Data Preprocessing (SQL) 55

APPENDIX 2: Code of Data Preprocessing (MATLAB) 81

APPENDIX 3: Neural Network Training for All Product Families 98

Figures

Figure 1: Factor Analysis for Cycle Time 13
Figure 2: Stages of Semiconductor Manufacturing (source: Mönch et al., [15]) 15
Figure 3: Sample of txt file of Flow Table raw data [redacted] 19
Figure 4: Sample of txt file of Equipment State Table raw data [redacted] 20
Figure 5: Flow Table data visualization & validation 20
Figure 6: Union of Flow Table txt files 21
Figure 7: Each equipment type executes more than one recipe 22
Figure 8: Each Recipe is executed in a single Equipment type 22
Figure 9: Recipe-based database 22
Figure 10: Average Priority Calculation 23
Figure 11: 10Eqptype Lots 23
Figure 12:Parts to be analyzed 23
Figure 13: Number of lots of part [redacted] that pass from the 10 selected equipment types 24
Figure 14: The equipment-based dataset for the 14 parts 24
Figure 15: Entry and departure timestamps of lots 25
Figure 16: Priorities & CT of the lots 26
Figure 17: Lots' CT Calculation 26
Figure 18: Recipe-based Processing Time 27
Figure 19: Equipment-type-based Processing Times 27
Figure 20: WIP Level of Fab upon Entry 28
Figure 21: Expected Queue Times for entering Lots 28
Figure 22: Connection of MATLAB to the Database 28
Figure 23: SQL-based MATLAB Script for WIP 29
Figure 24: MATLAB SQL-based script for Projected Time in Queue 29
Figure 25: MATLAB SQL-based script for Equipment' Utilization 30
Figure 26: Creating the Final Tables 30
Figure 27: Final Table Layout 31
Figure 28: ANN Philosophy [6] 32
Figure 29: NN's basic Math Operation [6] 32
Figure 30: Multilayer Network [17] 33
Figure 31: Algorithm Cheat Sheet (Source: scikit-learn.org) 34
Figure 32: A typical NN Layout 34
Figure 33: The sigmoid function 35
Figure 34: Backpropagation's algorithm. 36
Figure 35: A Network that depicts the architecture of the first two cases 38
Figure 36: The 1-Hidden Layer Network Set 42
Figure 37: The 2-Hidden Layer Network Set 42
Figure 38: The NNtool of MATLAB 42
Figure 39: NNtool Set Up of 6 nets for 1st Product Family 43

Figure 40: The training Parameters....43
Figure 41: Product Family 1 Train (1 Hidden Layer of 16 Neurons)....44
Figure 42: Product Family 1 Train (1 Hidden Layer of 32 Neurons)....44
Figure 43: Product Family 1 Weights (1 Hidden Layer of 32 Neurons)....45
Figure 44: Product Family 1 Train (2 Hidden Layers of 32 Neurons)....45
Figure 45: Neural Net on the whole Data....46
Figure 46: SQL Code for EQPS Merge....49
Figure 47: MATLAB Script for EQPS Merge....49
Figure 48: Dataset with Equipment States....49

Tables

Table 1: MAPE of predicted CTs for the three ANN architectures for each product family and all products....47
Table 2: Prediction results for the 1st product family....48
Table 3: Influencing factors tracing based on best & worst prediction for three product families....50

CHAPTER 1 Introduction

1.1 Motivation and Background

The strength of semiconductor manufacturing in Europe as mainly a supplier for many different applications (automotive, automation, etc.) lies in the possibility to quickly react to demand changes and provide a large variety of products on time. Therefore, it is necessary to be able to predict throughput and especially Cycle Times (CTs) with high accuracy weeks ahead of the actual delivery dates.

The accurate prediction of production CTs is of great importance for operations, sales and inventory managers, because it allows them to make better workflow management and release decisions, quote more accurate delivery times, reducing late delivery penalties and contract losses while improving customer satisfaction, and handle their inventories more effectively, implementing minimum stock policies.

Furthermore, accurate CT forecasts, empowered by production automation and Industry 4.0 technological advances, can greatly benefit the supply chain as a whole, especially if these forecasts are shared with the interested supply chain partners. For instance, if a change in the production times of a partner upstream the chain is communicated to a partner downstream the chain, the latter can adjust its schedule accordingly to absorb fluctuations.

Semiconductor manufacturing processes, however, have specific characteristics which make CT prediction particularly difficult: complex product flows, random yields, diverse equipment characteristics, equipment downtime, production and development in shared facilities, and data availability and maintenance. In the past decades, several methods have been proposed for CT estimation, including analytical, simulation, statistical analysis-based, and Artificial Neural Network (ANN) methods.

The objective of this thesis is to implement an ANN method that takes as inputs important factors that potentially influence CT to predict CTs in the Wafer Test Fab of an actual semiconductor manufacturing plant.

The work in this thesis was carried out in part within project [redacted]” which is funded by the European Research and Innovation Program Horizon 2020 and National Funding Authorities, through the [redacted] program. The project coordinator is [redacted], and more than 100 other industrial and research partners are involved. The main purpose of the project is to bring about a significant improvement in digitalization of European industry through electronic and Information and Communication Technology (ICT). [redacted] is the largest European research effort to date under the initiative of automation and data exchange in Industry4.0 industrial technologies [7].

The industrial partner that provided the use case and data to pursue our goal is [redacted] [redacted], through its Semiconductor Manufacturing plant in [redacted], which primary focuses on [redacted].

The author was involved in [redacted] as an undergraduate student at UTH MIE, during his 3rd year of studies, and so managed to obtain a holistic view of the project since the beginning. His two internships at Karlsruhe Institute of Technology (KIT), one of the main research partners, and in [redacted], gave him the opportunity to gain insight in the different problem-solving policies that each organization use while dealing with the project tasks.

Most of our work was carried out in the Production Management Laboratory (PML) of the Department of Mechanical Engineering (MIE) at the University of Thessaly (UTH), which is one of the research partners of [redacted]. More specifically, PML is involved in Work Packages 4 and 5, entitled [redacted]" and "[redacted]," respectively.

1.2 Literature Review

Several approaches for CT estimation have been proposed in the literature. These approaches can be divided into two categories: model-based and data-based.

Examples of approaches in the first category are queueing models, Markov Chains, and simulation. Shanthikumar et al. [8] surveyed research efforts in applications of queueing theory in semiconductor manufacturing, focusing on the improvement of model assumptions and model input, mainly in the mean and variance of the processing times. They acknowledged that practice had shown that implementation of classical queueing theory in semiconductor industry had been unsatisfactory, and they discussed open problems on queueing modeling semiconductor manufacturing systems.

Among the data-based methods, ANNs have been attracting increasing attention in recent years [9]. Chen & Wang [10] incorporated the fuzzy c-means (i.e., classifying the jobs into several categories) and back propagation network (FCM–BPN) approach with a nonlinear programming model to predict the completion time of the jobs. Moreover, Sha et al. [5] and Tirkel [11]found that ANN models outperform MLR and decision tree methods, which are also data-based. Furthermore, the large and relatively clean data that the industrial partner provided, permits the use of such data-driven models. For all these reasons, the choice to develop such a model is made.

In order for an ANN to return an accurate prediction of the desired metric, it has to be fed with the proper inputs. Thus, an analysis must be made on which factors should be treated as input factors for the model. The determination and calculation of these factors is in the core of this dissertation.

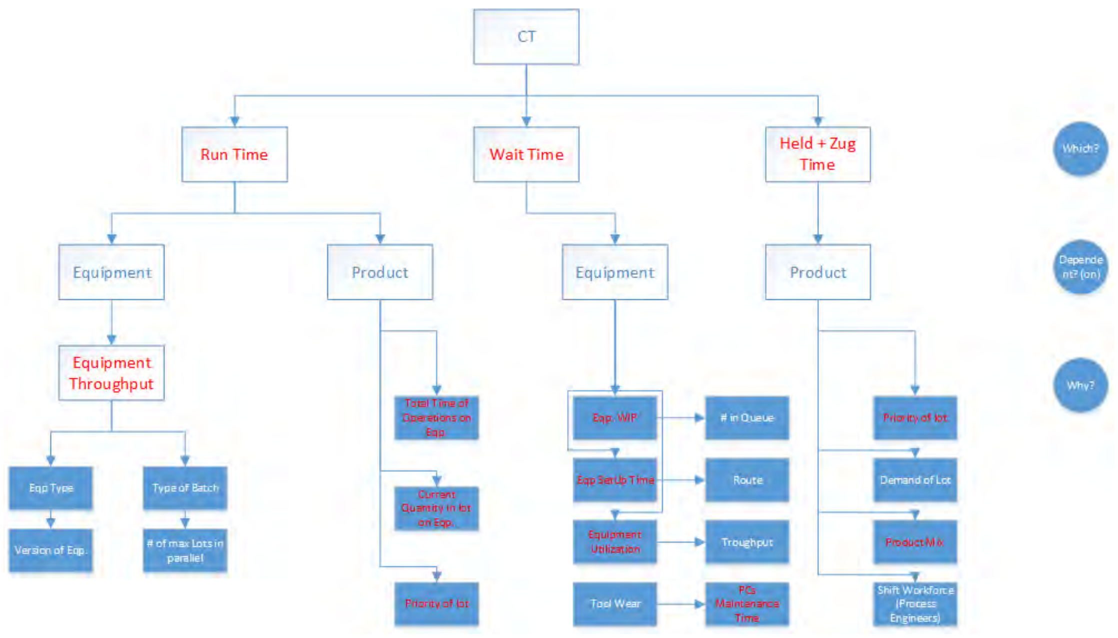


Figure 1: Factor Analysis for Cycle Time

As a starting point, let us break down the CT of a wafer lot into Processing (Run) Time, Wait Time and Transportation Time (see Figure 1). Which factors influence these times? Wang et al. (2018) [12], which inspired much of the work in this thesis, suggest in that CT is affected mainly by following input factors:

1. **"The processing time of each operation.** The processing of each operation is the difference between the track in time and track out time of the operation.
2. **The priority of each wafer lot.** The priority of a wafer lot is exactly equal to the attribute 'Priority' in the raw dataset.
3. **The utilization of each machine.** The utilization of a machine means the average utilization of a machine in a whole date (24 hours).
4. **The WIP level.** The WIP level can be measured by counting the wafer lot whose first process has been completed and the final process has not been completed.
5. **The size of waiting queue for each machine.** The Qi can be calculated by summing the next procedure's processing time of all wafer lots waiting before the machine i."

Going back to the diagram in Figure 1, we mark in red the factors that are considered more important.

1.3 Thesis Outline

The remainder of the thesis is organized as follows.

In CHAPTER 2, we briefly describe semiconductor manufacturing processes, focusing on the Wafer Test Fab stage.

In CHAPTER 3, we describe the dataset analysis. Initially, we describe the three-year data sample given to us by [redacted] plant. We focus on the use case of the Wafer Test Fab, for which we describe the preprocessing, validation and transformation of the related data. We complement this description with samples of Microsoft SQL Server and MATLAB code that was developed to acquire the factors that most likely affect CTs and constitute the input data of the ANNs.

CHAPTER 4 is devoted to ANNs. First, we introduce ANNs and discuss their usefulness and competitive advantage in analyzing complex manufacturing systems. Then we describe the MATLAB model and program that we developed for our specific use case.

In CHAPTER 5, we present and evaluate the results regarding the prediction of the ANN model. A discussion of the results and a few problematic cases follows.

Finally, CHAPTER 6 summarizes the works and gives directions for future research.

CHAPTER 2 Semiconductor Manufacturing

2.1 Manufacturing Process

Semiconductors are on the cutting edge of the electronics and technology revolution. Being among the fastest growing industries in the world, semiconductor manufacturing is perhaps the most complicated industry. The main raw material of a semiconductor is silicon, which is the main substrate used to manufacture Integrated Circuits (ICs). An IC is a device made of interconnected electronic components that can hold millions of circuits that are capable of high performance. The main attribute of a semiconductor is that it conducts electricity under some conditions and alternatively acts as an insulator in others [13]. This type of capability is widely exploited by products like computers, transistors and the mobile phones, among others [14].

In semiconductor manufacturing there are four basic stages: (1) Wafer Fabrication (Wafer-Fab), (2) Probe or Sort (Wafer-Test), (3) Assembly and (4) Final Testing, as it is depicted in Figure 2. These manufacturing processes can be grouped into two basic categories: “Front-End Operations” and “Back-End Operations”. Wafer Fabrication and Probe/Sort are included in “Front-End Operations” while Assembly and Final Testing belong to “Back-End Operations”.

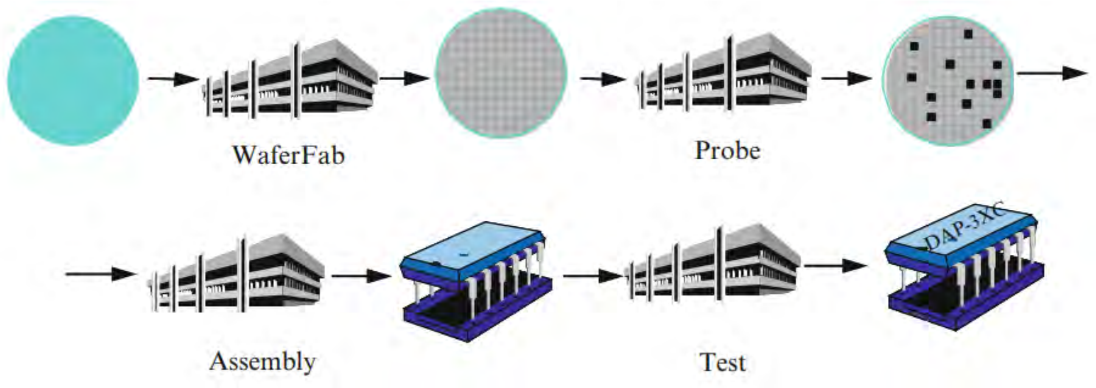


Figure 2: Stages of Semiconductor Manufacturing (source: Mönch et al., [15])

During the manufacturing process in the “Front End”, the products are grouped in lots, which in the [redacted] use case, contain a maximum of [redacted] wafers. The high degree of complication in the semiconductor manufacturing process stems partly from the fact that the process flow is highly re-entrant. Specifically, lots may pass more than once from the same equipment types, either for rework or as a part of their main flow.

As lots of raw wafers are deployed to the Wafer Fab, they are processed layer by layer, with the most advanced technologies, accumulating as many as 40 layers. The base process steps of this stage are the following [15]:

1. Oxidation/diffusion: A layer of material is grown or deposited on the surface of a cleaned wafer. Oxidation aims at growing a dioxide layer on a wafer. Diffusion is a high-temperature process that disperses material on the wafer surface.

2. Film Deposition: Deposition is used to deposit films onto wafers. The corresponding steps deposit dielectric or metal layers.
3. Photolithography: Coating, exposure, developing, and process control are the main steps of the photolithography process. In the first step, the wafer is coated with a thin film of a photosensitive polymer, called photoresist strip. Accurate and precise three-dimensional patterns are produced on the silicon wafer’s surface when an IC pattern is transferred via a photo mask, i.e., reticle, onto the photosensitive polymer, which replicates the pattern in the underlying layer. Exposure tools, called steppers, transfer the pattern onto the wafer by projecting light through the reticle to expose the wafer using ultraviolet light. The exposed wafer is then developed by removing polymerized sections of photoresist from the wafer. Every wafer passes through the photolithography area up to 40 times because the circuits are made up of layers. The photolithography work area is a typical example of a bottleneck in a wafer fab because steppers are very expensive machines.
4. Etch: This step is responsible for removing material from the wafer surface. The wafers are partially covered by photoresist strip after the photolithography step. Areas on the wafer that are not covered are then removed from the wafer.
5. Ion implantation: Dopant ions are selectively deposited on the surface of the wafer. Doping material is deposited where parts of the wafer have been etched.
6. Planarization: This step cleans and levels the wafer surface. It is called chemical-mechanical polishing (CMP).

Next, the wafers are sent to Probe/Sort, where tests identify the individual dies that are not likely to be good when packaged. Historically, bad dies were physically marked so that they would not be put in a package. Today, this has been replaced by producing an electronic map to identify the bad dies. The checked wafers are sent to an assembly facility where the dies with a reasonable quality are put into an appropriate package. Finally, the packaged dies are sent to a test facility where they are tested in order to ensure that only good products are sent to customers.

2.2 The Wafer Test Fab

This thesis focuses on the Probe/Sort stage, or more commonly the Wafer Test Fab. This fab is the final stage of the “Front End” operations, where products are still treated as wafer lots. Throughout this study, the objective is to create a tool that manages to efficiently predict CT values of these wafer lots.

In this fab, tests are run on the individual dies of each wafer in a wafer lot. The tests depend on the use that the dies are going to have when they will be formed into final products. In this manner, while the primary tests that run in this clean room are electrical, there also exist heat tests, pressure tests and even gas tests. During these tests, an electronic map of each wafer is updated. This map shows the distribution of dies on the wafer, and more importantly the ones that are characterized as defective. After the tests are performed, the wafers are forwarded for Visual Inspection. There, microscopes are used to determine if during the tests, dies were falsely marked as defunctive. Finally, in the Dot Marking area, the defective dies are marked physically by machines, so as to be scrapped when the wafers will be cut.

The equipment types that perform the electrical tests are composed of two main parts: The Prober, which executes the motion part, and the Tester, which executes the measurement part. The allocation of wafer lots to the machines is performed through a software that takes into account factors, such as the type of product, its priority and the availability of auxiliary parts for the testing, such as the PCs. PCs are hardware that have needles through which electrical tests are run on the dies of a wafer. These needles get worn out by use and need to be restored. This procedure is performed in a special area within the fab and seems to cause one of the main bottlenecks in the process flow.

Another delay is caused when a lot fails to perform the test on a machine, in which case, it is stored in a buffer until a process engineer deals with it. This results in a significant time of the lot in this machine, which may be recorded as a long waiting time or a long processing time, depending on whether in the data entry it is considered behind (queue) or in (processing) the machine. As it will become clear in the next chapters, cases like these can greatly affect the CT forecasting model.

Moreover, in the Wafer Test Fab, the flow of products observed is usually low, which means that equipment there has low utilization. The reason for this is that the Wafer Fab usually pushes wafer lots at a lower throughput than the Wafer Test Fab can handle. This is reasonable, since the former fab, being more expensive and complex that the latter, has lower capacity. The low utilization of equipment in the Wafer Test Fab was also observed in the data analysis, as we will see later.

Nevertheless, the utilization of certain machines in the Wafer Test Fab may see a substantial increase due to the relatively high intake volume of a certain product that requires to be tested in same type of equipment. This is a phenomenon that can be attributed to the Product Mix that is initially deployed in the Wafer Fab.

The above observations are important for comprehending the physical system and assessing the results that come out of the ANN.

CHAPTER 3 Data Analysis

3.1 Description

The dataset provided by the industrial partner contains 3 years [redacted] of records, organized in seven different tables that contain information about the material flow through the fabs, the states of the equipment, the flow and the equipment states specifically in the photolithography stage, lists of new and removed equipment throughout the 3 years, as well as data about the yield in the dicing area. We also had access to the database of lot priorities.

For our work, we used the Flow and Equipment State tables in the Wafer Test Fab.

More specifically, the Flow Tables contain the following information:

1. **LOTID**: Lot id (Lot Name) Lots that start with T are Test Wafers.
2. **LOTTYPE**: Highest aggregation of products (product cluster, technology) (part ⊆ product group ⊆ LOTTYPE). Lot types that start with Z are test wafer lot types.
3. **Obfuscated Part**: Anonymized name of product (without version) for all products ran in [redacted] (it is a “primary key” together with RECPID and STAGE, i.e., each combination of these three attributes is unique in the database).
4. **Obfuscated Prod.Area**: Anonymized name of the production area (Factory).
5. **STAGE**: Workflow section (it is a primary key together with RECPID and obfuscated Part).
6. **Obfuscated Eqptype**: Anonymized name of tool / machine / equipment group (group of same kind of equipment (equipment capability can vary due to equipment qualification status).
7. **EQPID**: Name of Equipment (Equipment ⊆ Equipment Type ⊆ Location ⊆ Production Area ⊆ Plant).
8. **RECPID**: Name of Recipe (Machine Program) (it is a primary key together with Stage and obfuscated Part).
9. **QUEUETIME**: Track out time stamp of lot in previous process step.
10. **TRACKINTIME**: Track in time stamp of lot in process step.
11. **TRACKOUTTIME**: Track out time stamp of lot in process step.
12. **TRACKINMAINQTY**: Lot size at lot track in (in Frontend <= [redacted] Wafer) (often 1 in case of Test lots).
13. **CURMAINQTY**: Lot size at lot track out. It is equal to TrackOutMainQty.
14. **CURPRCDKIND**: Kind of procedure being currently executed. [redacted] [redacted] and [redacted] are rework procedures, [redacted] and [redacted] are normal procedures.
15. **DUEDATE**: Schedule delivery date.

Due to the big amount of raw data (more than fifty million lines of input) and files, the tables were transferred to a single table in an SQL database.

The Equipment State Table contains the following information:

1. **EQPID**: Name of Equipment (Equipment ⊆ Equipment Type ⊆ Location ⊆ Production Area ⊆ Plant)
2. **LASTSTATE**: State before state change. NULL for new equipment.
3. **STATE**: State after state change.
4. **CHANGEDT**: Date of state change (from LASTSTATE to STATE).
5. **SERIALNOBYTE**: Serial Number of state change event. SerialNoByte is equal to 0 for current state and 1, 2, 3, ..., for historical record.

In addition to the two above types of tables, we also used a table that contained a list of LOTIDs and their priorities. A problem with this table is that it had multiple entries of the same LOTIDs with different priorities and no time stamps to identify the LOTIDs from the Flow Tables.

From the above data, we proceeded to calculate the factors described in the previous chapter, namely, the processing time of each operation, the priority of each wafer lot, the utilization of each machine, the WIP level and the size of waiting queue for each machine.

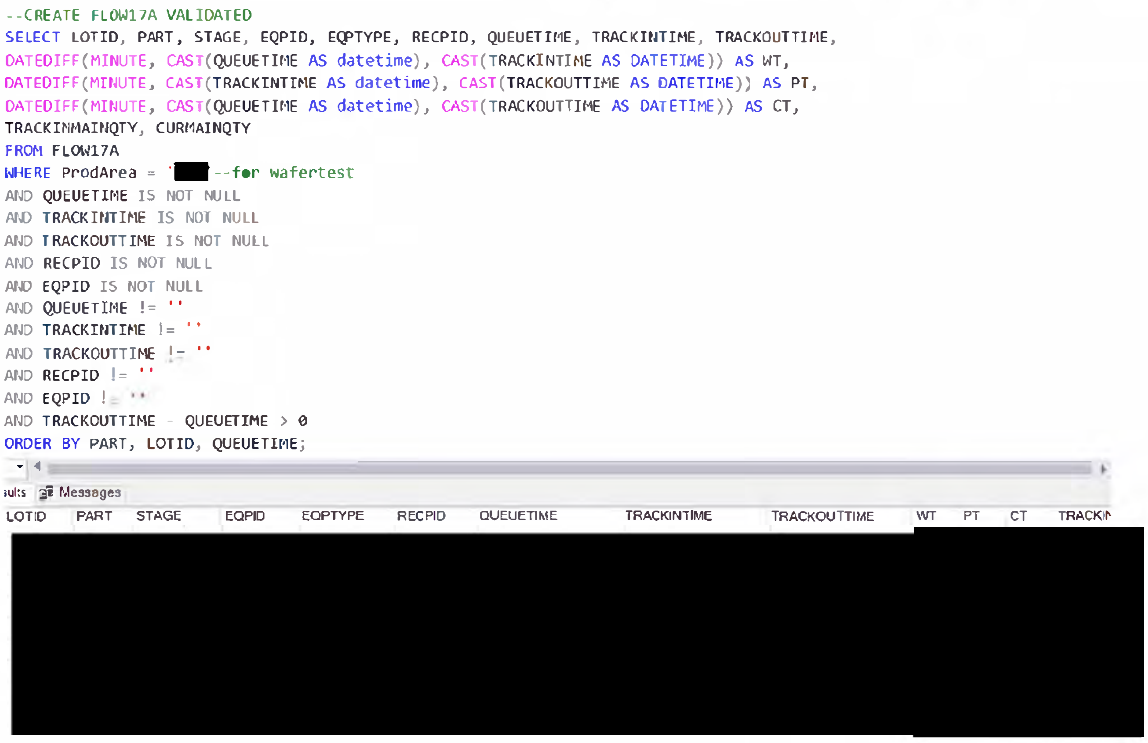
In the following sections, we describe the steps we took to reach this calculation.

3.2 Data Validation & Preprocessing

The raw data for both tables came in a txt format, as shown in Figure 3 and Figure 4. To validate the data and be able to visualize and manipulate it, we used Microsoft SQL Server. Figure 5 shows the SQL commands used to convert the Flow Table raw data of the Wafer Test Fab into a table format and validate them.

Figure 3: Sample of txt file of Flow Table raw data [redacted].

Figure 4: Sample of txt file of Equipment State Table raw data [redacted]



```
--CREATE FLOW17A VALIDATED
SELECT LOTID, PART, STAGE, EQPID, EQPTYPE, RECPID, QUEUETIME, TRACKINTIME, TRACKOUTTIME,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKINTIME AS DATETIME)) AS WT,
DATEDIFF(MINUTE, CAST(TRACKINTIME AS datetime), CAST(TRACKOUTTIME AS DATETIME)) AS PT,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKOUTTIME AS DATETIME)) AS CT,
TRACKINMAINQTY, CURMAINQTY
FROM FLOW17A
WHERE ProdArea = '[redacted]' --for wafertest
AND QUEUETIME IS NOT NULL
AND TRACKINTIME IS NOT NULL
AND TRACKOUTTIME IS NOT NULL
AND RECPID IS NOT NULL
AND EQPID IS NOT NULL
AND QUEUETIME != ''
AND TRACKINTIME != ''
AND TRACKOUTTIME != ''
AND RECPID != ''
AND EQPID != ''
AND TRACKOUTTIME - QUEUETIME > 0
ORDER BY PART, LOTID, QUEUETIME;
```

```
sults  Messages
LOTID  PART  STAGE  EQPID  EQPTYPE  RECPID  QUEUETIME  TRACKINTIME  TRACKOUTTIME  WT  PT  CT  TRACKIN
```

[redacted]

Figure 5: Flow Table data visualization & validation

As can be seen from Figure 5, we performed the following steps for validating the data:

1. We chose records concerning only the Wafer Test Fab by setting the Production Area equal to the anonymized name of that fab.
2. We eliminated records with "Null" and empty values for the Datetime data, recipe and equipment ids.
3. We omitted production year [redacted] and only kept [redacted] which the data was more reliable.

4. We eliminated records with zero processing times (whose departure times are equal to their arrival times), because they added computational burden, without carrying important information. A version of the table without this limitation was also studied. The original version with the elimination is called Validated and the second version without the elimination is called Semi-Validated.
5. The records were ordered alphabetically by Part (Product Family) name and LOTID (lot identification) and then in increasing order of the time that the lot enters the queue of the machine (QueueTime).

As can also be seen from Figure 5, the differences between the three key datetime instances of each record were defined and calculated. These differences are:

WT (Waiting Time) = Trackintime – QueueTime

PT (Processing Time) = TrackOutTime – Trackintime

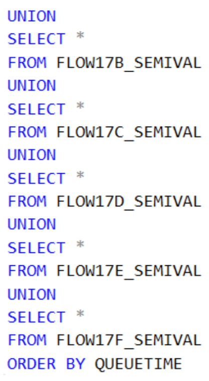
CT (Cycle Time) = TrackOutTime – QueueTime

The above differences were calculated in minutes and were used in the subsequent analysis.

It is worth noting that the above validation steps were partly suggested by our project collaborators who dealt with the same dataset and our common industrial partner that provided that set.

To minimize the size of the data, we only kept the important fields, namely: LOTID, PART, STAGE, EQPID, EQPTYPE, RECPID, QUEUETIME, TRACKINTIME, TRACKOUTTIME, PT, WT, CT, TRACKINMAINQTY, CURMAINQTY.

Finally, we merged all the validated files (there were originally 10 txt files for production years [redacted]) into one big unified file using the union command, as shown in Figure 6.



```
UNION
SELECT *
FROM FLOW17B_SEMIVAL
UNION
SELECT *
FROM FLOW17C_SEMIVAL
UNION
SELECT *
FROM FLOW17D_SEMIVAL
UNION
SELECT *
FROM FLOW17E_SEMIVAL
UNION
SELECT *
FROM FLOW17F_SEMIVAL
ORDER BY QUEUETIME
```

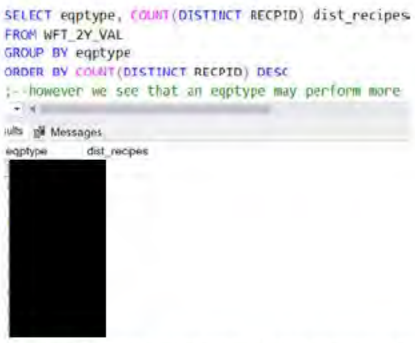
Figure 6: Union of Flow Table txt files

The unified file contained 775,415 records for the Wafer Test Fab which were ordered in increasing Queue Time.

3.3 Data Analysis Strategy

The analysis of the data is based on the methodology developed in Wang et al. [4], who implemented that methodology on a simulation-generated wafer fab model. Although that model produces 3 types of wafers with different numbers of production "procedures" (operations), only one of these types is analyzed. The operations of these type are performed on specific machines.

Initially, we wanted to see if we can perform a similar type of experiment. To this end, we checked if all or at least the majority of products (LOTIDs) have the same number of recipes (operations) in the Wafer Test Fab. By analyzing the data, we found that the majority of lots have [redacted] recipes, with more than one recipe being executed on the same type of equipment (see Figure 7). We verified that each recipe is performed in exactly one type of equipment, as Figure 8 shows. A screen-shot sample of the recipe-based database is shown in Figure 9.

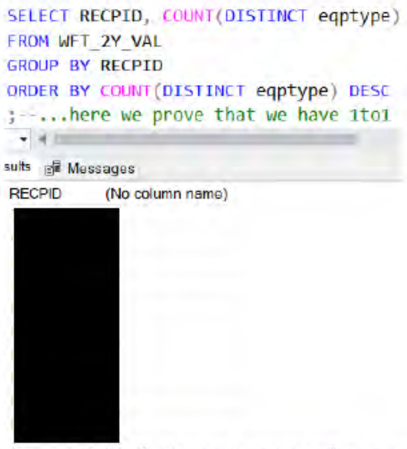


```
SELECT eqptype, COUNT(DISTINCT RECPID) dist_recipes
FROM WFT_2Y_VAL
GROUP BY eqptype
ORDER BY COUNT(DISTINCT RECPID) DESC
;--however we see that an eqptype may perform more
```

```
eqptype     dist_recipes
```

[redacted]

Figure 7: Each equipment type executes more than one recipe

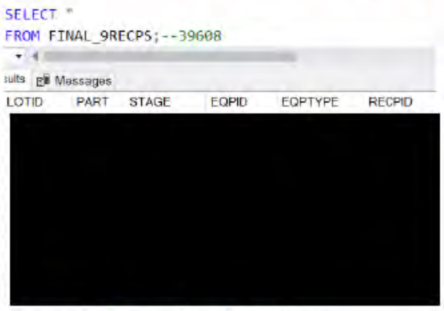


```
SELECT RECPID, COUNT(DISTINCT eqptype)
FROM WFT_2Y_VAL
GROUP BY RECPID
ORDER BY COUNT(DISTINCT eqptype) DESC
;--...here we prove that we have 1to1
```

```
RECPID     (No column name)
```

[redacted]

Figure 8: Each Recipe is executed in a single Equipment type



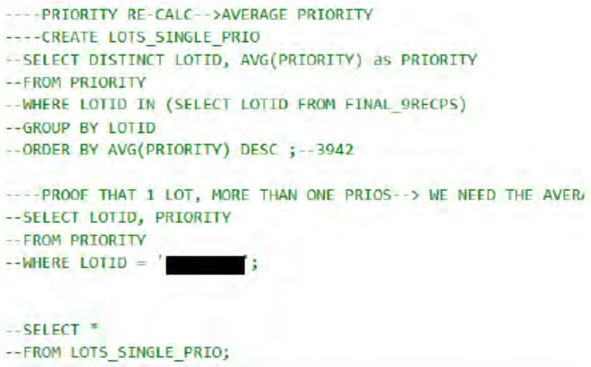
```
SELECT *
FROM FINAL_9RECPS;--39608
```

```
LOTID   PART   STAGE   EQPID   EQPTYPE   RECPID
```

[redacted]

Figure 9: Recipe-based database

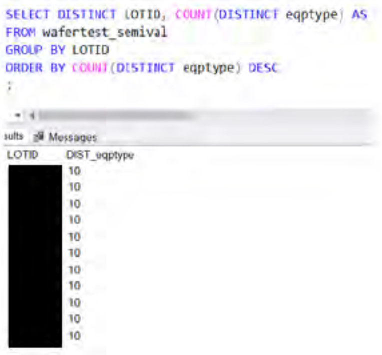
Because the raw data that contained the priorities of the lots had multiple entries of the same LOTIDs with different priorities, as was mentioned earlier, for each LOTID, we used its average priority for our calculations (see Figure 10).



```
----PRIORITY RE-CALC-->AVERAGE PRIORITY
----CREATE LOTS_SINGLE_PRIO
--SELECT DISTINCT LOTID, AVG(PRIORITY) as PRIORITY
--FROM PRIORITY
--WHERE LOTID IN (SELECT LOTID FROM FINAL_9RECPS)
--GROUP BY LOTID
--ORDER BY AVG(PRIORITY) DESC ;--3942

----PROOF THAT 1 LOT, MORE THAN ONE PRIOS--> WE NEED THE AVER
--SELECT LOTID, PRIORITY
--FROM PRIORITY
--WHERE LOTID = '[redacted]';


--SELECT *
--FROM LOTS_SINGLE_PRIO;
```

Figure 10: Average Priority Calculation

Next, we checked if all or at least the majority of products (LOTIDs) are processed in the same equipment types in the Wafer Test Fab. By analyzing the data, we found that the vast majority of lots were processed in the same 10 equipment types. Figure 11 shows a sample screen-shot of LOTIDs and the respective number of equipment types.



```
SELECT DISTINCT LOTID, COUNT(DISTINCT eqptype) AS
FROM wafertest_semival
GROUP BY LOTID
ORDER BY COUNT(DISTINCT eqptype) DESC
;
```

```
LOTID       DIST_eqptype
[redacted]  10
            10
            10
            10
            10
            10
            10
            10
            10
            10
            10
```

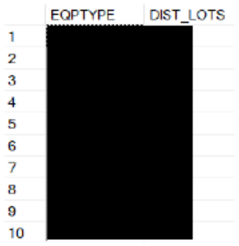
Figure 11: 10Eqptype Lots

Because the calculation of the factors that potentially affect CTs are equipment-based, we chose to sort and aggregate the data based on the 10 most used equipment types. To this end, first, we selected the parts (product families) that contain a sufficient number of lots for our analysis. We found 14 parts that had more than [redacted] lots, which was considered sufficient (see Figure 12). The total number of lots of these parts was [redacted].

```
PART        DIST_LOTID
[redacted]
```

Figure 12:Parts to be analyzed

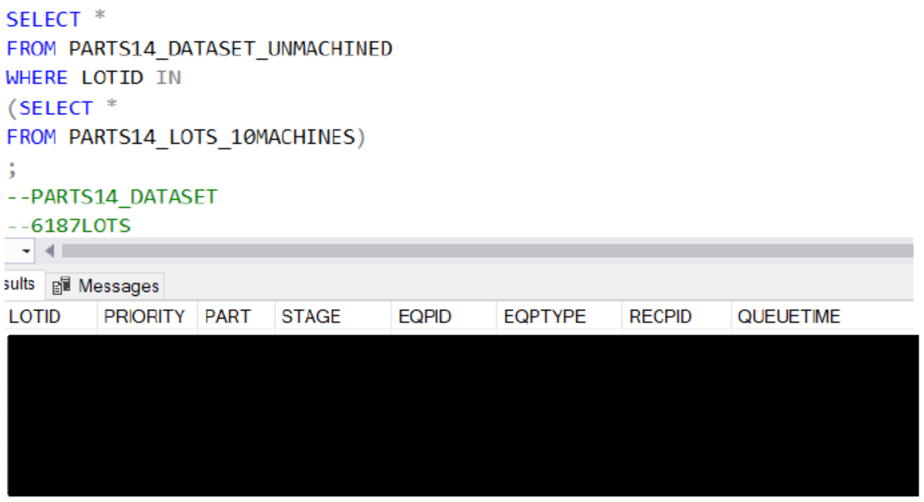
For validation purposes, we made sure that all the lots of these parts passed from the 10 selected equipment types. Figure 13 shows an example of a query of the number of lots of the first part type in Figure 12 ([redacted]) that pass from the 10 selected equipment. As can be seen by comparing the two figures, all the lots pass from all the equipment types.



	EQPTYPE	DIST_LOTS
1	[redacted]	[redacted]
2	[redacted]	[redacted]
3	[redacted]	[redacted]
4	[redacted]	[redacted]
5	[redacted]	[redacted]
6	[redacted]	[redacted]
7	[redacted]	[redacted]
8	[redacted]	[redacted]
9	[redacted]	[redacted]
10	[redacted]	[redacted]

Figure 13: Number of lots of part PMVBL that pass from the 10 selected equipment types

Based on the above analysis, we created the equipment-based dataset that was then used to shape the final table to be used as input of the ANN model. This set contained the following attributes: LOTID, PRIORITY, PART, STAGE, EQPID, EQPTYPE, RECPID, QUEUETIME, TRACKINTIME, TRACKOUTTIME, PT, WT, CT, TRACKINMAINQTY, CURMAINQTY (see Figure 14).



```
SELECT *
FROM PARTS14_DATASET_UNMACHINED
WHERE LOTID IN
(SELECT *
FROM PARTS14_LOTS_10MACHINES)
;
--PARTS14_DATASET
--6187LOTS
```

LOTID	PRIORITY	PART	STAGE	EQPID	EQPTYPE	RECPID	QUEUETIME
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Figure 14: The equipment-based dataset for the 14 parts

The entries of the final table to be used as input of the ANN model are:

1. The LOTID.
2. The part that the LOTID belongs to.
3. The LOTID priority.
4. The cumulative processing time that the LOTID spends on each equipment type.
5. The sum of the processing times of all the lots waiting to be processed in each equipment type at the time of entry of the LOTID.
6. The utilizations of the equipment-types, during the last 24 hours before the entry of the LOTID.

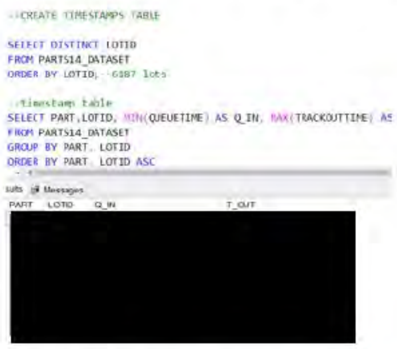
7. The total WIP (Work in Process) in the entire Wafer Test Fab at the time of entry of the LOTID.
8. The actual CT of the LOTID.

The first 7 entries comprise the factors that potentially influence the CT while the last entry is the actual CT that will be used as the Target Value for training the ANN.

In the remainder of this section, we will describe how we generated the final table using SQL and MATLAB code.

3.4 SQL Code Description

To calculate the entries of the final table, for each lot, we computed two timestamps: the time that it first entered the Wafer Test Fab and the time that it departed from it. The SQL commands for these calculations are shown in Figure 15.



```
--CREATE TIMESTAMPS TABLE

SELECT DISTINCT LOTID
FROM PARTS14_DATASET
ORDER BY LOTID; --6187 lots

--timestamp table
SELECT PART,LOTID, MIN(QUEUETIME) AS Q_IN, MAX(TRACKOUTTIME) AS
FROM PARTS14_DATASET
GROUP BY PART, LOTID
ORDER BY PART, LOTID ASC
```

```
PART   LOTID   Q_IN                T_OUT
```

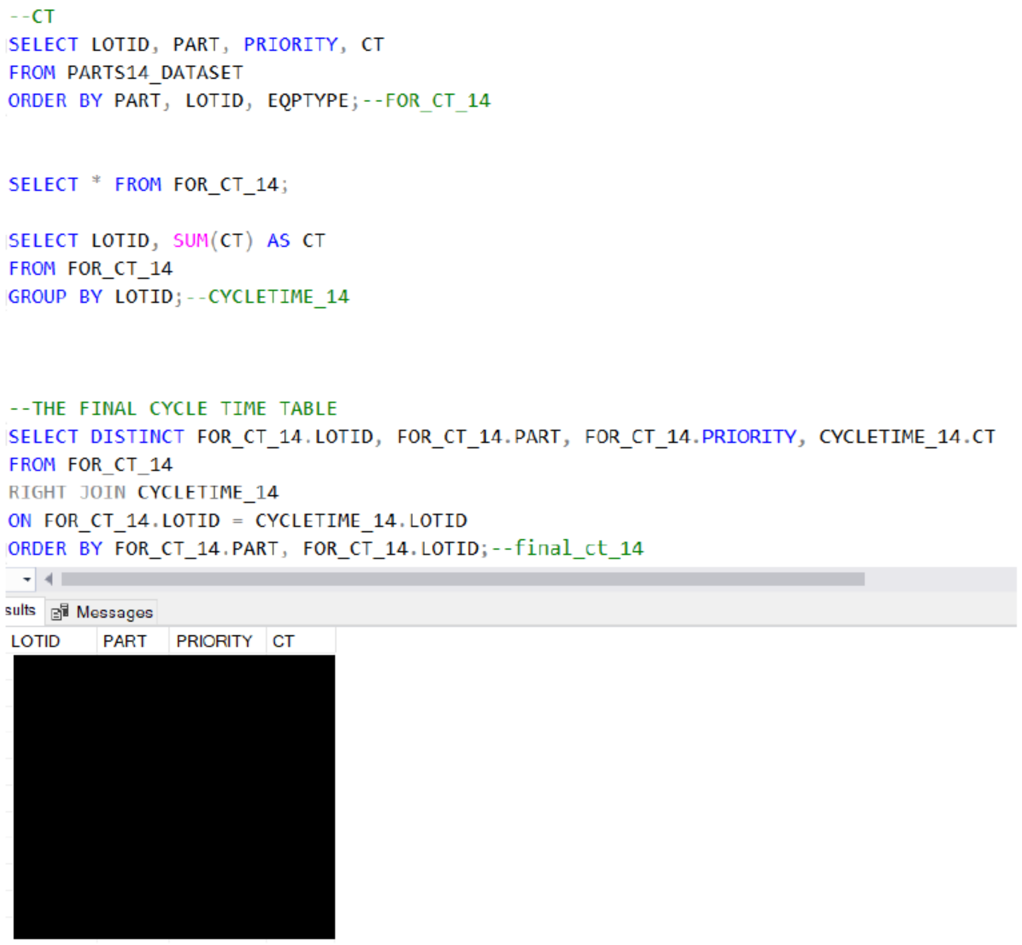
[redacted]

Figure 15: Entry and departure timestamps of lots

Having calculated the entry and departure timestamps, we proceeded to calculate the CT of each lot. Figure 16 shows the SQL commands to generate the CTs for each lot.

The calculation of the rest of the data, i.e., entries 4-7 of the final table, was done using MATLAB code, as we will see in the next subsection. SQL was still used to validate these data. In addition, an SQL query was also developed to track the state of the equipment upon the QueueTime, TrackInTime and TrackOutTime for each lot. This query was primarily used to validate the MATLAB script that computes these states.

The queries developed for the above purposes can be found in APPENDIX 1: Code of Data Preprocessing (SQL).



```sql
--CT
SELECT LOTID, PART, PRIORITY, CT
FROM PARTS14_DATASET
ORDER BY PART, LOTID, EQPTYPE;--FOR_CT_14


SELECT * FROM FOR_CT_14;

SELECT LOTID, SUM(CT) AS CT
FROM FOR_CT_14
GROUP BY LOTID;--CYCLETIME_14



--THE FINAL CYCLE TIME TABLE
SELECT DISTINCT FOR_CT_14.LOTID, FOR_CT_14.PART, FOR_CT_14.PRIORITY, CYCLETIME_14.CT
FROM FOR_CT_14
RIGHT JOIN CYCLETIME_14
ON FOR_CT_14.LOTID = CYCLETIME_14.LOTID
ORDER BY FOR_CT_14.PART, FOR_CT_14.LOTID;--final_ct_14
```

LOTID	PART	PRIORITY	CT
[redacted]			

Figure 16: Priorities & CT of the lots

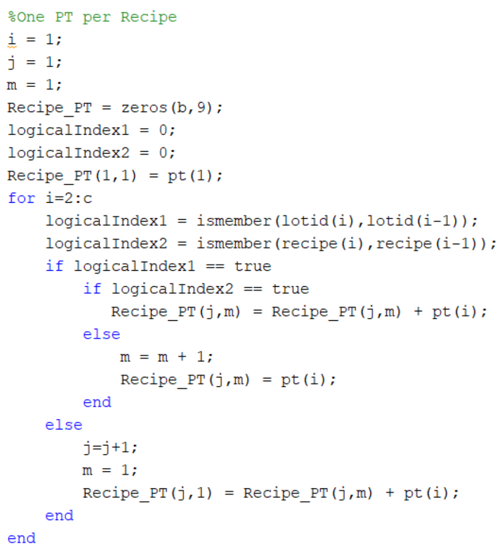
3.5 MATLAB Code Description

As was mentioned in Section 3.3, we first tried to sort and aggregate the lots based on recipe. To this end, we used MATLAB's "Database Explorer" tool, which allows MATLAB to connect with local databases and acquire data directly. After loading the proper data in the proper data type, we validated the CT of each lot in the fab (see Figure 17).

```matlab
%Create the LotID_CT Table
Lot_CT(1)=ct(1);
for i=2:c
    logicalIndex=ismember(lotid(i),lotid(i-1));
    if logicalIndex==true
        Lot_CT(j)=Lot_CT(j)+ct(i);
    else
        j=j+1;
        Lot_CT(j)=ct(i);
    end
end
```

Figure 17: Lots' CT Calculation

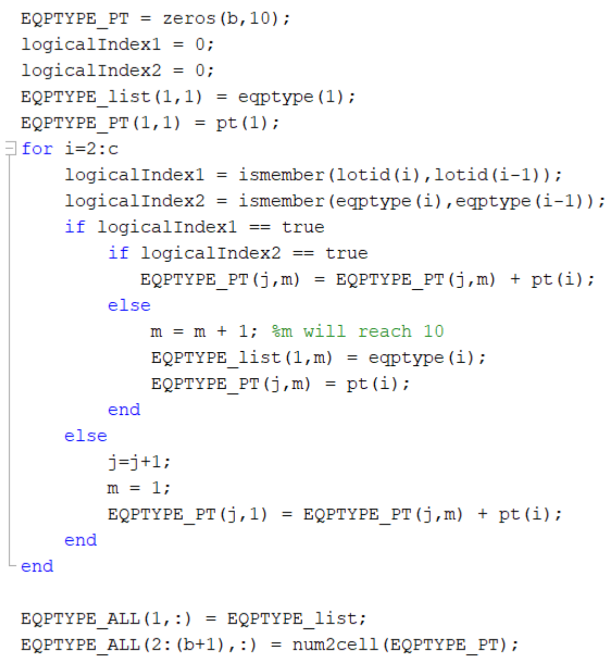
Then, we calculated the total processing time of each lot on each equipment type, accounting for the fact that a lot might have passed from the same equipment type more than once (see Figure 18).



```
%One PT per Recipe
i = 1;
j = 1;
m = 1;
Recipe_PT = zeros(b,9);
logicalIndex1 = 0;
logicalIndex2 = 0;
Recipe_PT(1,1) = pt(1);
for i=2:c
    logicalIndex1 = ismember(lotid(i),lotid(i-1));
    logicalIndex2 = ismember(recipe(i),recipe(i-1));
    if logicalIndex1 == true
        if logicalIndex2 == true
           Recipe_PT(j,m) = Recipe_PT(j,m) + pt(i);
        else
            m = m + 1;
            Recipe_PT(j,m) = pt(i);
        end
    else
        j=j+1;
        m = 1;
        Recipe_PT(j,1) = Recipe_PT(j,m) + pt(i);
    end
end
```

Figure 18: Recipe-based Processing Time

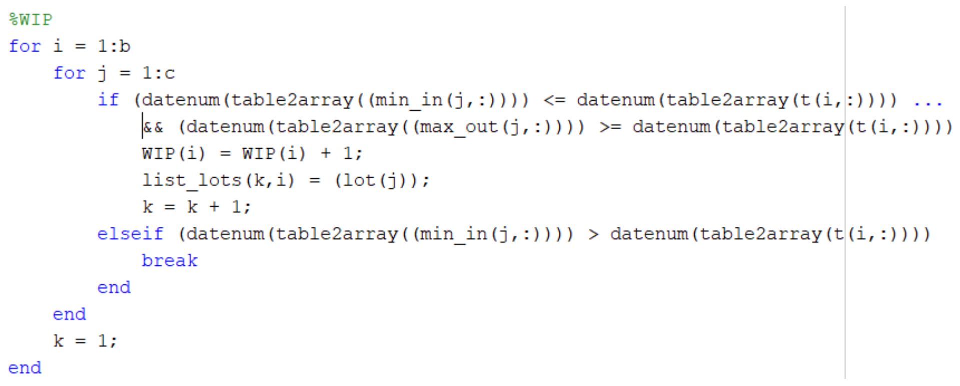
We repeated a similar procedure for the equipment-type-based model, which as was mentioned in Section 3.3 was easier to handle (see Figure 19).



```
EQPTYPE_PT = zeros(b,10);
logicalIndex1 = 0;
logicalIndex2 = 0;
EQPTYPE_list(1,1) = eqptype(1);
EQPTYPE_PT(1,1) = pt(1);
for i=2:c
    logicalIndex1 = ismember(lotid(i),lotid(i-1));
    logicalIndex2 = ismember(eqptype(i),eqptype(i-1));
    if logicalIndex1 == true
        if logicalIndex2 == true
           EQPTYPE_PT(j,m) = EQPTYPE_PT(j,m) + pt(i);
        else
            m = m + 1; %m will reach 10
            EQPTYPE_list(1,m) = eqptype(i);
            EQPTYPE_PT(j,m) = pt(i);
        end
    else
        j=j+1;
        m = 1;
        EQPTYPE_PT(j,1) = EQPTYPE_PT(j,m) + pt(i);
    end
end

EQPTYPE_ALL(1,:) = EQPTYPE_list;
EQPTYPE_ALL(2:(b+1),:) = num2cell(EQPTYPE_PT);
```

Figure 19: Equipment-type-based Processing Times

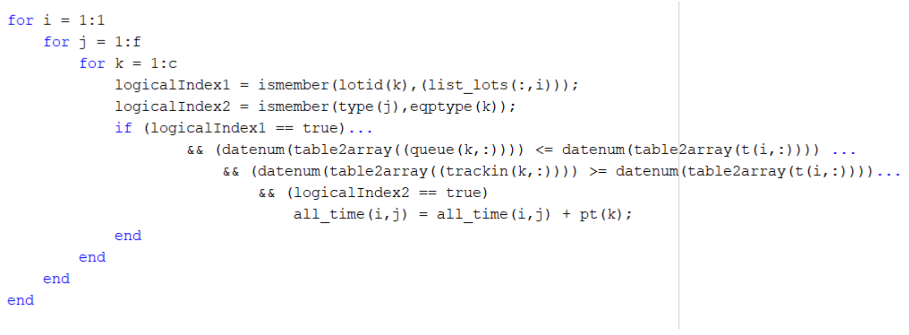
To calculate the Work in Process upon entry of each lot x, we used the timestamps table shown in Figure 15 that was created in SQL. For this calculation, we added all the lots that entered the fab before the arrival of lot x and departed the fab after that arrival. The MATLAB code is shown in Figure 20.



```
%WIP
for i = 1:b
    for j = 1:c
        if (datenum(table2array((min_in(j,:)))) <= datenum(table2array(t(i,:)))) ...
            && (datenum(table2array((max_out(j,:)))) >= datenum(table2array(t(i,:))))
            WIP(i) = WIP(i) + 1;
            list_lots(k,i) = (lot(j));
            k = k + 1;
        elseif (datenum(table2array((min_in(j,:)))) > datenum(table2array(t(i,:))))
            break
        end
    end
    k = 1;
end
```

Figure 20: WIP Level of Fab upon Entry

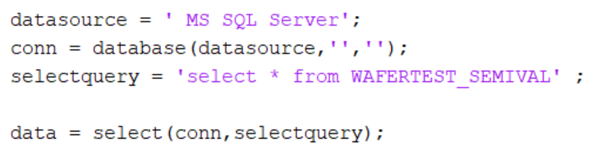
Next, for each lot x, we calculated the sum of the processing times of all the lots waiting to be processed in each equipment type at the time of entry of x. Figure 21 shows the MATLAB code for doing this.



```
for i = 1:1
    for j = 1:f
        for k = 1:c
            logicalIndex1 = ismember(lotid(k),(list_lots(:,i)));
            logicalIndex2 = ismember(type(j),eqptype(k));
            if (logicalIndex1 == true)...
                    && (datenum(table2array((queue(k,:)))) <= datenum(table2array(t(i,:)))) ...
                        && (datenum(table2array((trackin(k,:)))) >= datenum(table2array(t(i,:))))...
                            && (logicalIndex2 == true)
                                all_time(i,j) = all_time(i,j) + pt(k);
            end
        end
    end
end
```

Figure 21: Expected Queue Times for entering Lots

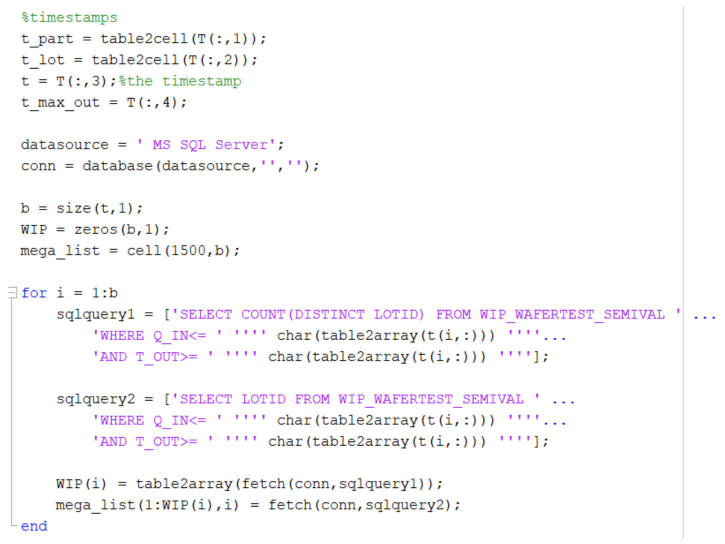
Although the above two codes are correct, they are extremely slow and hence impractical. A much faster alternative was to repetitively execute SQL query statements from within the MATLAB environment. To access SQL from MATLAB, we used the code shown in Figure 22.



```
datasource = ' MS SQL Server';
conn = database(datasource,'','');
selectquery = 'select * from WAFERTEST_SEMIVAL' ;

data = select(conn,selectquery);
```

Figure 22: Connection of MATLAB to the Database

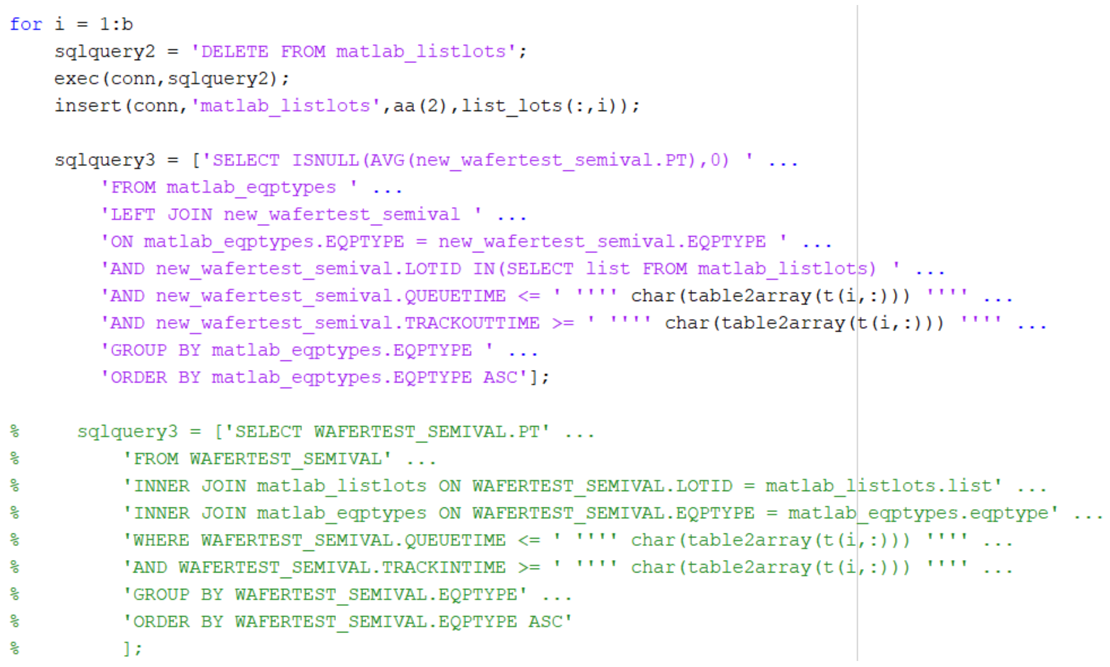
Then, we used the SQL ‘count’ command within a simple “for” loop to deliver the outcome extremely quickly (on average, it took approximately in 50 seconds). The main idea for calculating the WIP upon entry remains the same. Figure 23 shows the MATLAB code.



```
%timestamps
t_part = table2cell(T(:,1));
t_lot = table2cell(T(:,2));
t = T(:,3);%the timestamp
t_max_out = T(:,4);

datasource = ' MS SQL Server';
conn = database(datasource,'','');

b = size(t,1);
WIP = zeros(b,1);
mega_list = cell(1500,b);

for i = 1:b
    sqlquery1 = ['SELECT COUNT(DISTINCT LOTID) FROM WIP_WAFERTEST_SEMIVAL ' ...
        'WHERE Q_IN<= ' '''' char(table2array(t(i,:))) ''''...
        'AND T_OUT>= ' '''' char(table2array(t(i,:))) ''''];

    sqlquery2 = ['SELECT LOTID FROM WIP_WAFERTEST_SEMIVAL ' ...
        'WHERE Q_IN<= ' '''' char(table2array(t(i,:))) ''''...
        'AND T_OUT>= ' '''' char(table2array(t(i,:))) ''''];

    WIP(i) = table2array(fetch(conn,sqlquery1));
    mega_list(1:WIP(i),i) = fetch(conn,sqlquery2);
end
```

Figure 23: SQL-based MATLAB Script for WIP

Similarly, Figure 24 shows the MATLAB code for the calculation of the sum of the processing times of all the lots waiting to be processed in each equipment type at the time of entry of each individual lot x.



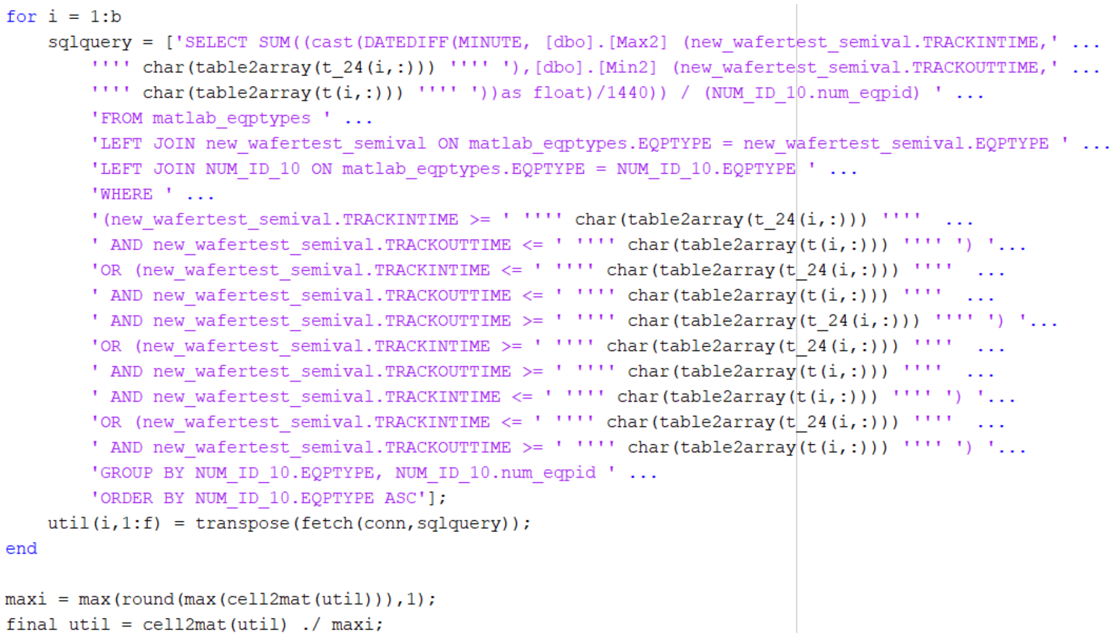
```
for i = 1:b
    sqlquery2 = 'DELETE FROM matlab_listlots';
    exec(conn,sqlquery2);
    insert(conn,'matlab_listlots',aa(2),list_lots(:,i));

    sqlquery3 = ['SELECT ISNULL(AVG(new_wafertest_semival.PT),0) ' ...
        'FROM matlab_eqptypes ' ...
        'LEFT JOIN new_wafertest_semival ' ...
        'ON matlab_eqptypes.EQPTYPE = new_wafertest_semival.EQPTYPE ' ...
        'AND new_wafertest_semival.LOTID IN(SELECT list FROM matlab_listlots) ' ...
        'AND new_wafertest_semival.QUEUETIME <= ' '''' char(table2array(t(i,:))) '''' ...
        'AND new_wafertest_semival.TRACKOUTTIME >= ' '''' char(table2array(t(i,:))) '''' ...
        'GROUP BY matlab_eqptypes.EQPTYPE ' ...
        'ORDER BY matlab_eqptypes.EQPTYPE ASC'];

%     sqlquery3 = ['SELECT WAFERTEST_SEMIVAL.PT' ...
%         'FROM WAFERTEST_SEMIVAL' ...
%         'INNER JOIN matlab_listlots ON WAFERTEST_SEMIVAL.LOTID = matlab_listlots.list' ...
%         'INNER JOIN matlab_eqptypes ON WAFERTEST_SEMIVAL.EQPTYPE = matlab_eqptypes.eqptype' ...
%         'WHERE WAFERTEST_SEMIVAL.QUEUETIME <= ' '''' char(table2array(t(i,:))) '''' ...
%         'AND WAFERTEST_SEMIVAL.TRACKINTIME >= ' '''' char(table2array(t(i,:))) '''' ...
%         'GROUP BY WAFERTEST_SEMIVAL.EQPTYPE' ...
%         'ORDER BY WAFERTEST_SEMIVAL.EQPTYPE ASC'
%         ];
```

Figure 24: MATLAB SQL-based script for Projected Time in Queue

Because the number of equipment (eqpid_number) of each individual equipment type might be more than one, the sum of the processing times was divided by eqpid_number to yield an average value, as follows:

final_time_queue = table2array(cell2table(TIME_QUEUE))./eqpid_number;

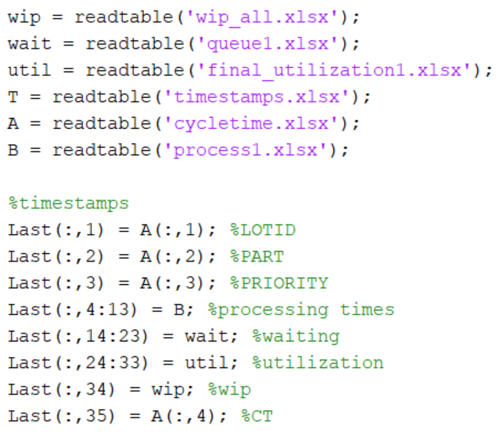
For the utilization of the equipment, a similar calculation method was used. First, an additional timestamp was calculated, signaling the beginning of a 24-hour window before the entry time of each lot x. Then, the utilization of each equipment-type, during this window was calculated by adding the part of the interval between the arrival and departure times of each lot that falls within the window and dividing the total by the number of equipment for each individual equipment type. The MATLAB code is shown in Figure 25.



```
for i = 1:b
    sqlquery = ['SELECT SUM((cast(DATEDIFF(MINUTE, [dbo].[Max2] (new_wafertest_semival.TRACKINTIME,' ...
        '''' char(table2array(t_24(i,:))) '''' '),[dbo].[Min2] (new_wafertest_semival.TRACKOUTTIME,' ...
        '''' char(table2array(t(i,:))) '''' '))as float)/1440)) / (NUM_ID_10.num_eqpid) ' ...
        'FROM matlab_eqptypes ' ...
        'LEFT JOIN new_wafertest_semival ON matlab_eqptypes.EQPTYPE = new_wafertest_semival.EQPTYPE ' ...
        'LEFT JOIN NUM_ID_10 ON matlab_eqptypes.EQPTYPE = NUM_ID_10.EQPTYPE ' ...
        'WHERE ' ...
        '(new_wafertest_semival.TRACKINTIME >= ' '''' char(table2array(t_24(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKOUTTIME <= ' '''' char(table2array(t(i,:))) '''' ') '...
        'OR (new_wafertest_semival.TRACKINTIME <= ' '''' char(table2array(t_24(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKOUTTIME <= ' '''' char(table2array(t(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKOUTTIME >= ' '''' char(table2array(t_24(i,:))) '''' ') '...
        'OR (new_wafertest_semival.TRACKINTIME >= ' '''' char(table2array(t_24(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKOUTTIME >= ' '''' char(table2array(t(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKINTIME <= ' '''' char(table2array(t(i,:))) '''' ') '...
        'OR (new_wafertest_semival.TRACKINTIME <= ' '''' char(table2array(t_24(i,:))) ''''  ...
        ' AND new_wafertest_semival.TRACKOUTTIME >= ' '''' char(table2array(t(i,:))) '''' ') '...
        'GROUP BY NUM_ID_10.EQPTYPE, NUM_ID_10.num_eqpid ' ...
        'ORDER BY NUM_ID_10.EQPTYPE ASC'];
    util(i,1:f) = transpose(fetch(conn,sqlquery));
end

maxi = max(round(max(cell2mat(util))),1);
final_util = cell2mat(util) ./ maxi;
```

Figure 25: MATLAB SQL-based script for Equipment' Utilization

The final table to be used in the ANN model was finally created by using the code in Figure 26.



```
wip = readtable('wip_all.xlsx');
wait = readtable('queue1.xlsx');
util = readtable('final_utilization1.xlsx');
T = readtable('timestamps.xlsx');
A = readtable('cycletime.xlsx');
B = readtable('process1.xlsx');

%timestamps
Last(:,1) = A(:,1); %LOTID
Last(:,2) = A(:,2); %PART
Last(:,3) = A(:,3); %PRIORITY
Last(:,4:13) = B; %processing times
Last(:,14:23) = wait; %waiting
Last(:,24:33) = util; %utilization
Last(:,34) = wip; %wip
Last(:,35) = A(:,4); %CT
```

Figure 26: Creating the Final Tables

This final table, has the layout shown in Figure 27.

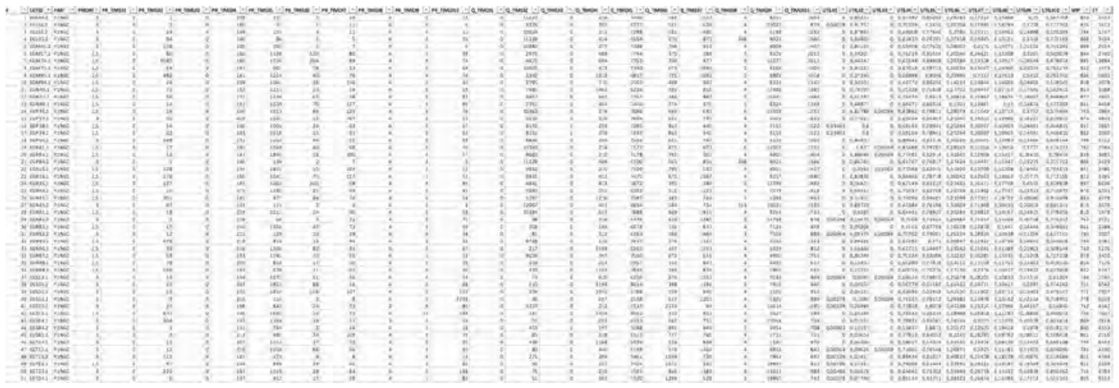


Figure 27: Final Table Layout

The code developed for the above purposes can be found in the APPENDIX 2: Code of Data Preprocessing (MATLAB).

CHAPTER 4 Artificial Neural Network (ANN) Model

4.1 Introduction to ANNs

As it is stated, the prediction methodology that is followed on this dissertation makes use of ANNs. So, it is of some value for the reader to be introduced into the ANNs and their features [16].

ANNs are computational models trained to perform tasks by taking into account examples. The main aspiration of ANNs is to carry through with the given assignments after the appropriate training procedure. These assignments could be image and voice recognition, data classification, prediction optimization, etc.

More specifically, ANNs are composed of simple elements operating in parallel. These elements are inspired by biological neural networks. As in nature, the network function is determined largely by the connections between elements. One can train an ANN to perform a particular function by adjusting the values of the connections (weights) between elements. Typically, ANNs are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Figure 28. There, the ANN is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are used, in this supervised learning, to train a network. The User Guide of MATLAB for its Neural Network Tool [6] is very explanatory on this topic.

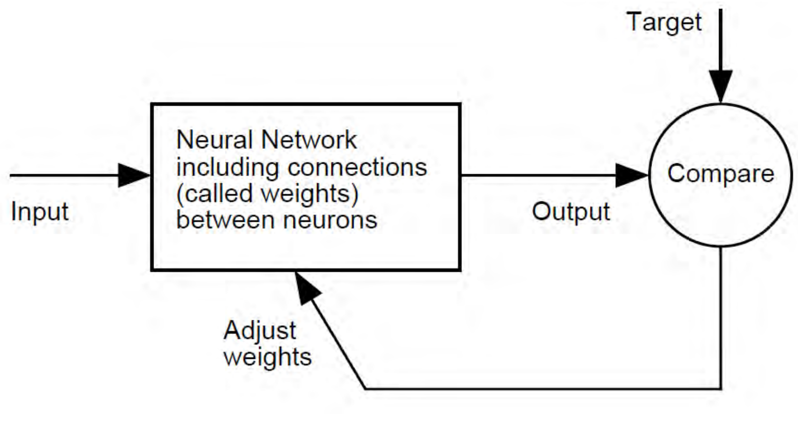

Target
Input
Neural Network including connections (called weights) between neurons
Output
Compare
Adjust weights

Figure 28: ANN Philosophy [6]

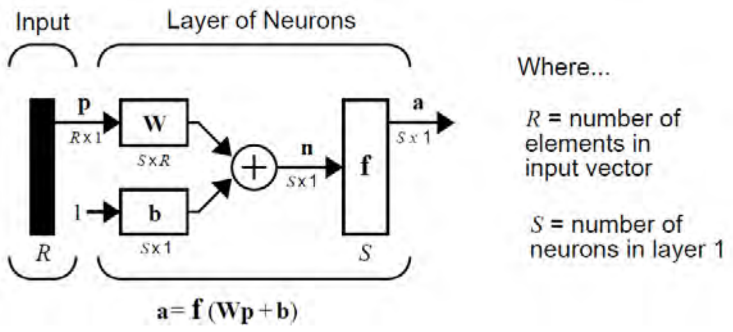


Figure 29: NN's basic Math Operation [6]

A layer of a network is defined in Figure 29. A layer includes the combination of the weights, the multiplication and summing operation (here realized as a vector product $W * p$), the bias b, and the transfer function f. The vector of inputs, p, is not included in the layer. In Figure 29, $\boldsymbol{w}$ and $\boldsymbol{b}$ are both adjustable scalar parameters of the neuron. The central idea of ANNs is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, we can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.

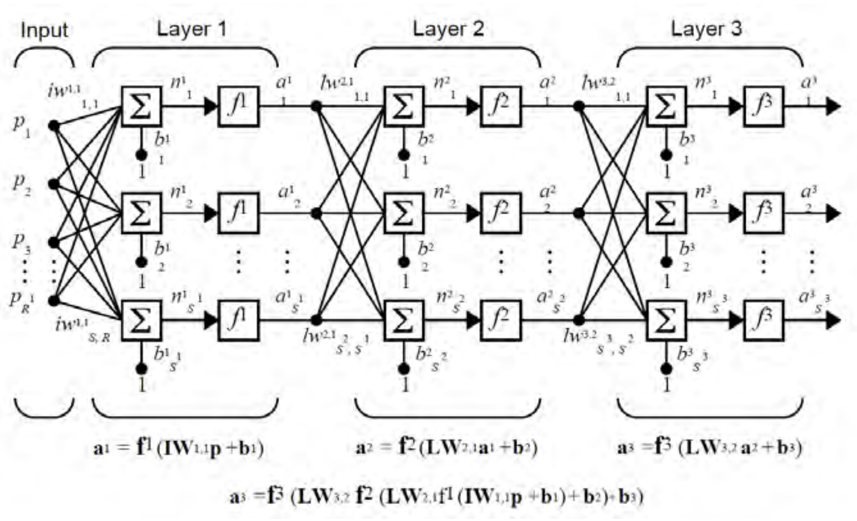

Input
Layer 1
Layer 2
Layer 3
$\mathbf{a}_1 = \mathbf{f}^1(\mathbf{IW}_{1,1}\mathbf{p} + \mathbf{b}_1)$
$\mathbf{a}_2 = \mathbf{f}^2(\mathbf{LW}_{2,1}\mathbf{a}_1 + \mathbf{b}_2)$
$\mathbf{a}_3 = \mathbf{f}^3(\mathbf{LW}_{3,2}\mathbf{a}_2 + \mathbf{b}_3)$
$\mathbf{a}_3 = \mathbf{f}^3(\mathbf{LW}_{3,2}\mathbf{f}^2(\mathbf{LW}_{2,1}\mathbf{f}^1(\mathbf{IW}_{1,1}\mathbf{p} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3)$

Figure 30: Multilayer Network [17]

The architecture of a network consists of a description of how many layers a network has, the number of neurons in each layer, each layer's transfer function, and how the layers connect to each other. Figure 30 shows a multilayer ANN.

Aside from the number of neurons in a network's output layer, the number of neurons in each layer is up to the designer. Except for purely linear networks, the more neurons in a hidden layer, the more powerful the network. Multiple feed-forward layers give a network greater freedom. For example, any reasonable function can be represented with a two-layer network: a sigmoid layer feeding a linear output layer [6]. The cheat-sheet presented on Figure 31 may be of use when choosing the ANN proper architecture.

There are 3 main cases that can be found in ANNs:

1. Classification: All classification tasks depend upon labeled datasets; that is, humans must transfer their knowledge to the dataset in order for a neural network to learn the correlation between labels and data. This is known as supervised learning.
2. Clustering: Clustering or grouping is the detection of similarities. Deep learning does not require labels to detect similarities. Learning without labels is called unsupervised learning.

3. Predictive Analytics (Regressions): Deep learning is able to establish correlations between present events and future events. It can run regression between the past and the future. The future event is like the label in a sense.

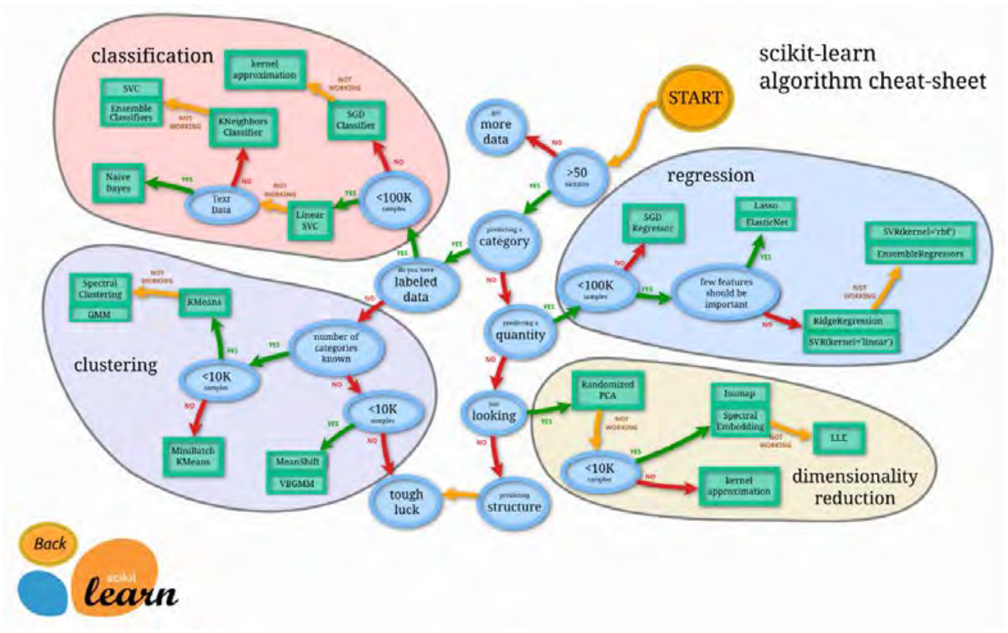


Figure 31: Algorithm Cheat Sheet (Source: scikit-learn.org)

In supervised learning (Classification), there is a known number of classes, it is based on a training dataset and used to classify future observations. In unsupervised (Clustering), there is an unknown number of classes, no prior knowledge and the model is used to understand (explore) the data.

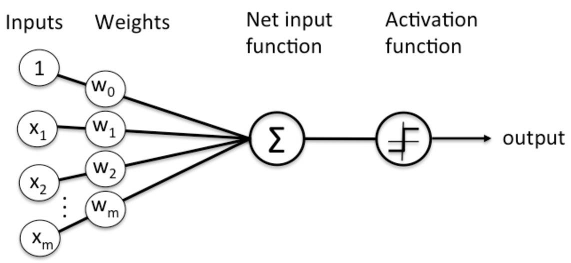


Figure 32: A typical NN Layout

The case of this thesis falls in the supervised learning class, since the data provided by the industrial partner are structured.

In essence, a neural network, as presented in Figure 32, is a collection of neurons connected by synapses. This collection is organized into three main layers: the input layer, the hidden layer, and the output layer. One can have many hidden layers, which is where the term deep learning comes into play. The role of a synapse is to multiply the inputs and weights. One can think of weights as the “strength” of the connection between neurons. Then, an activation function is applied to return an output. Deep learning maps inputs to outputs, by finding correlations. It is known as a "universal approximator", because it can learn to approximate an unknown function $f(x) = y$ between any input x and any output y, assuming they are related at all (by correlation or causation, for example).

A neuron or node is just a place where computation happens. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input. These input-weight products are summed. Sum is passed through a node's so-called activation function. If the signals pass through, then the neuron has been "activated".

The rules through which the number of hidden layers and neurons is determined, are a bit vague, as it depends between other on the number of training examples and the complexity of the desired classification. The most common strategy is a combination of trial and error with empirical methods.

A side note is that it is very important to keep in mind the overfitting problem, i.e. when the neural network becomes too complex and insensitive to input changes.

Weights are the most essential factors in converting an input to impact the output. The output is determined by the equation below:

$$y = f(x) = \sum x_i w_i$$

Bias is an additional parameter which is added to adjust the output along with the weighted sum of the inputs. Therefore, it helps the model to fit best for the given data.

$$\text{Layer } Output = \sum x_i w_i + Bias$$

Then a function called activation function is applied on this output so that the input of the next layer is the output of the neurons in the previous layer.

The most frequently used activation functions are the ReLU (Rectified Linear Unit) and the Log-Sigmoid Function, as shown in Figure 33. The general form of the sigmoid function is:

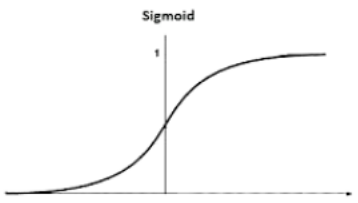


Figure 33: The sigmoid function.

$$f(x) = \frac{1}{1 + e^{-bx}}$$

One defines a learning rule as a procedure for modifying the weights and biases of a network. (This procedure may also be referred to as a training algorithm.) The learning rule is applied to train the network to perform some particular task. Learning fall into two broad categories: supervised learning, and unsupervised learning. In supervised learning, the learning rule is provided with a set of examples (the training set) of proper network behavior where is an input to the network, and is the corresponding correct (target) output. As the inputs are applied to the network, the network outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets [6]. Also, it is of worth to be mentioned that long training times can be caused by the presence of an outlier input vector whose length is much larger or smaller than the other input vectors.

Backpropagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by the user, as it can be seen in Figure 34. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Standard backpropagation is a gradient descent algorithm, as is the Widrow-Hoff learning rule, in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks.

Properly trained backpropagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input/ target pairs and get good results without training the network on all possible input/output pairs [17].

These are the reasons that Backpropagation is opted for this dissertation.

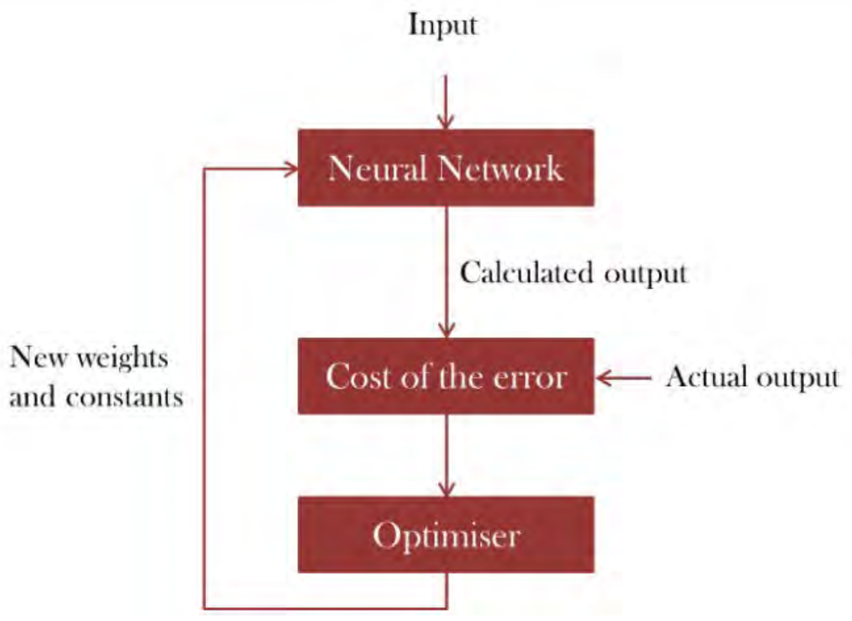


Figure 34: Backpropagation's algorithm.

4.2 ANNs in Manufacturing & Competitive Advantage

Modern manufacturing systems, especially in the semiconductor industry, are extremely complex. ANNs provide a promising alternative for making accurate predictions based on data, by mapping inputs into outputs, without knowing the underlying dependence function.

Furthermore, as was already mentioned in Section 1.2, the competitive advantage of ANNs is that they emphasize on predictive accuracy, while statistical modeling targets interpretability and parsimony. More specifically, ANNs can produce output even with incomplete information and are able to solve a great variety of similar problems. On the other hand, statistical methods may be more comprehensible, but they are usually stiff as they are limited to specific statistical distributions and require some assumptions in order to perform predictions.

4.3 ANN Model (NN-Tool)

In this sub-chapter we explain how to use the backpropagation training functions in MATLAB Neural Network toolbox to train feedforward ANNs to solve specific problems. There are generally four steps in the training process:

1. Assemble the training data
2. Create the network object
3. Train the network
4. Simulate the network response to new inputs

The first step was thoroughly described in the previous chapters. The remaining three steps will be analyzed here.

As was described in the Data Analysis on CHAPTER 3, there are 32 inputs to our model: The processing times, the queue times upon entry and the last-24-hour utilizations of the 10 equipment types, plus the wafer lot priorities and the total WIP upon entry of the fab. The only output is the CT of each wafer lot. The class of the problem is “Regression”, and so the proper architecture of the hidden layers and hidden neurons should be decided.

Based both on the above presented literature and on trial and error procedures, the architecture of the network is decided. First as discussed, a Feed-Forward Backpropagation Network, as shown in Figure 35, will be utilized. More specifically, three network architectures will be tested on each product family data, as well as on the whole dataset. These three cases are inspired by the number of inputs (32) and are the following:

1. 1 Hidden Layer of 16 neurons (half the inputs).
2. 1 Hidden Layer of 32 neurons.
3. 2 Hidden Layers of 32 neurons each.

The activation function from each layer to the next is chosen to be the tangent sigmoid (tansig), and from the last hidden layer to the output layer the pure linear (ReLU). Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors.

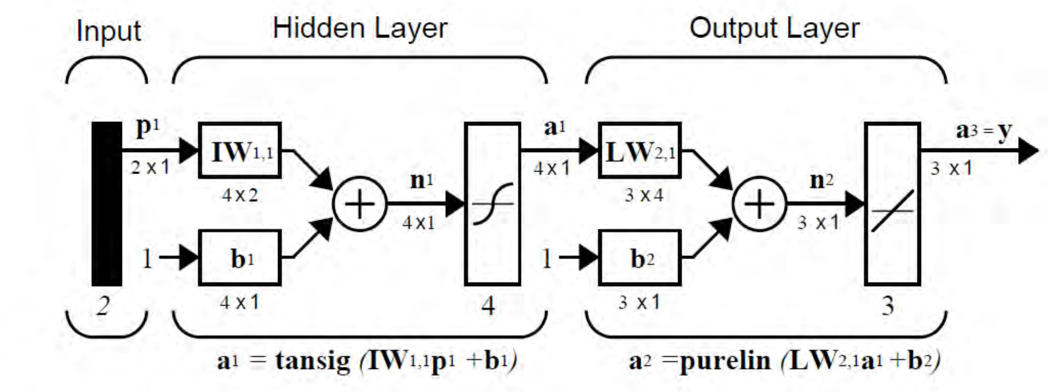


Figure 35: A Network that depicts the architecture of the first two cases

Once the network weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The current case is the first one. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function *net.performFcn*, which is chosen to be regularized mean square error mse - the regularized average squared error between the network outputs a and the target outputs t. There are several different training algorithms for feedforward networks. All of these algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called backpropagation, which involves performing computations backwards through the network. The backpropagation computation is derived using the chain rule of calculus.

The simplest implementation of backpropagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly - the negative of the gradient.

One iteration of this algorithm can be written as: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{\alpha}_k \boldsymbol{g}_k$

where $\mathbf{x}_k$ is a vector of current weights and biases, $\boldsymbol{g}_k$ is the current gradient, and $\boldsymbol{\alpha}_k$ is the learning rate.

There are several training functions that can be used to train the network. An example of them is the Batch Steepest Descent (TrainGD), where the weights and biases are updated in the direction of the negative gradient of the performance function, i.e. the mse.

Also, there are seven training parameters associated with TrainGD:

1. Epochs
2. Show
3. Goal
4. Time
5. min_grad
6. max_fail
7. The learning rate (LR)

The learning rate LR is multiplied times the negative of the gradient to determine the changes to the weights and biases. The larger the learning rate, the bigger the step. If the learning rate is made too large, the algorithm becomes unstable. If the learning rate is set too small, the algorithm takes a long time to converge. With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate.

The training status is displayed for every show iteration of the algorithm. The other parameters determine when the training stops. The training stops if the number of iterations exceeds epochs, if the performance function drops below goal, if the magnitude of the gradient is less than min_grad, or if the training time is longer than time seconds. Also, max_fail is associated with the early stopping technique.

When all the parameters set, one is ready to train the net. After training, they can simulate the net by providing new data and examining how good the outcome is.

However, one should mention here that the Batch Steepest Descent (TrainGD) method is often too slow [16] and as analyzed a couple of lines before, depends a lot on the correct LR setting. Thus, another algorithm should be selected.

In this quest for the proper algorithm selection, the choices are many. Some of them are the Resilient Backpropagation (TrainRP), the Conjugate Gradient Algorithms such as the Fletcher-Reeves Update (TrainCGF), the Polak-Ribiére Update (TrainCGP), the Powell-Beale Restarts (TrainCGB) and the Scaled Conjugate Gradient (TrainSCG). Also, there are Line Search Routines as well as Quasi-Newton Algorithms, such as the BFGS Algorithm (TrainBGF) and the One Step Secant Algorithm (TrainOSS).

Newton’s method is an alternative to the conjugate gradient methods for fast optimization. The basic step of Newton’s method is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{A}_k - \boldsymbol{1} * \boldsymbol{g}_k$$

where $\boldsymbol{A}_k$ is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton’s method often converges faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feedforward neural networks.

With this in mind, the focus falls on the Levenberg-Marquardt Method (TrainLM) and more precisely on the Bayesian Regularization Method (TrainBR), which is actually a modification of the Levenberg-Marquardt training algorithm to produce networks that generalize well. It reduces the difficulty of determining the optimum network architecture [6].

For the Levenberg-Marquardt Method (TrainLM) the following are true. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as $\mathbf{H} = \mathbf{J}^{\mathbf{T}}\mathbf{J}$ and the gradient can be computed as $\mathbf{g} = \mathbf{J}^{\mathbf{T}}\mathbf{e},$ where $\mathbf{J}$ is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and $\mathbf{e}$ is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [\mathbf{J}^{\mathbf{T}}\mathbf{J} + \mu\mathbf{I}] - 1 * \mathbf{J}^{\mathbf{T}}\mathbf{e}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

The training parameters for TrainLM are:

1. Epochs
2. Show
3. Goal
4. Time
5. min_grad
6. max_fail
7. mu
8. mu_dec
9. mu_inc
10. mu_max
11. mem_reduc

The parameter mu is the initial value for μ. This value is multiplied by mu_dec whenever the performance function is reduced by a step. It is multiplied by mu_inc whenever a step would increase the performance function. If mu becomes larger than mu_max, the algorithm is stopped. The parameter mem_reduc is used to control the amount of memory used by the algorithm. It's used to determine how many rows of the Jacobian are to be computed in each submatrix. If mem_reduc is set to 1, then the full Jacobian is computed, and no memory reduction is achieved.

In many cases, TrainLM is able to obtain lower mean square errors than any of the other algorithms tested. However, as the number of weights in the network increases, the advantage of the TrainLM decreases.

One of the problems that occurs during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.

One method for improving network generalization is to use a network that is just large enough to provide an adequate fit. The larger a network you use, the more complex the functions the network can create. A method, also, for improving generalization is called regularization. This involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. The typical performance function that is used for training feedforward neural networks is the mean sum of squares of the network errors.

$$F = mse = \frac{1}{N}\sum_{i=1}^{N} e_i^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2$$

It is possible to improve generalization if we modify the performance function by adding a term that consists of the mean of the sum of squares of the network weights and biases:

$$msereg = \gamma * mse + (1-\gamma)msw$$

where γ is the performance ratio and:

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2$$

Using this performance function will cause the network to have smaller weights and biases, and this will force the network response to be smoother and less likely to overfit.

It is desirable to determine the optimal regularization parameters in an automated fashion. In this framework, the weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to the unknown variances associated with these distributions. We can then estimate these parameters using statistical techniques.

One feature of this algorithm is that it provides a measure of how many network parameters (weights and biases) are being effectively used by the network, indicated by #Par in the printout.

For the TrainBR, it is important to let the algorithm run until the effective number of parameters has converged. The training may stop with the message “Maximum MU reached.” This is typical and is a good indication that the algorithm has truly converged. You can also tell that the algorithm has converged if the sum squared error (SSE) and sum squared weights (SSW) are relatively constant over several iterations. When this occurs, you may want to push the “Stop Training” button in the training window [6].

Moreover, all in all, a two-layer network, with tan-sigmoid transfer function in the hidden layer and a linear transfer function in the output layer, is a useful structure for function approximation (regression) problems.

Also, the training function is chosen to be the Bayesian Regularization one and the performance metric is set to be the MSEreg, i.e. the Regularized Mean Square Error. The network architecture for a 2-layer and a 3-layer version with 32 neurons in each of the hidden layers can be seen in Figure 36 and Figure 37 respectively.

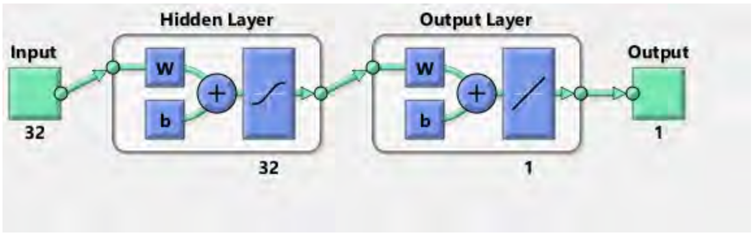


Figure 36: The 1-Hidden Layer Network Set

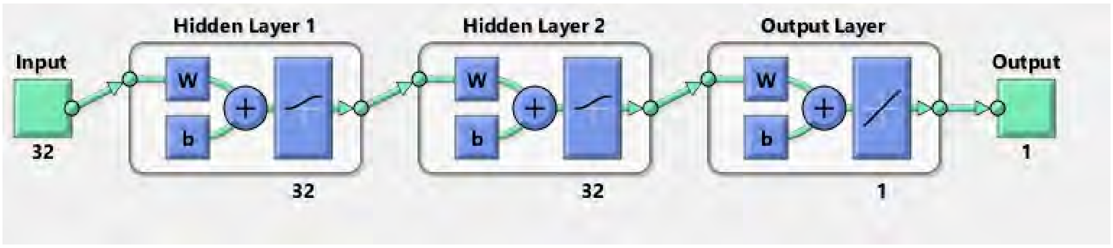
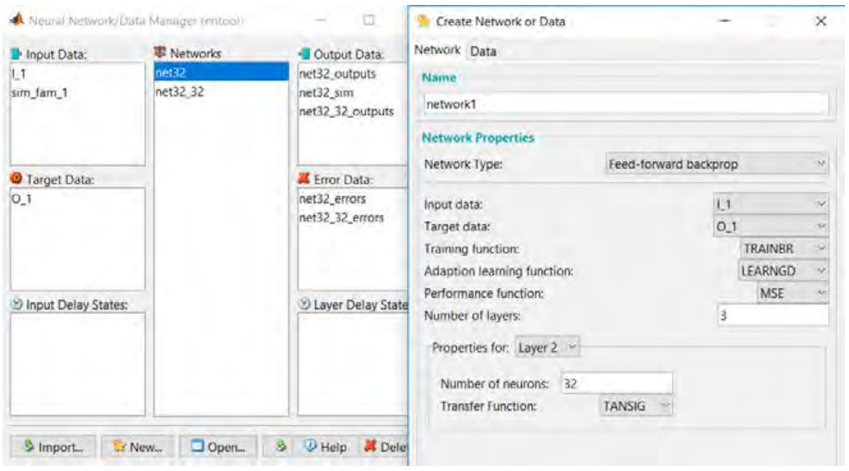


Figure 37: The 2-Hidden Layer Network Set

So, in this point follows a ty to apply the two Neural Network sets through MATLAB’s NNtool on all the product families by their self. In Figure 38 is depicted the setting of an ANN in NNtool.



Input Data:
I_1
sim_fam_1

Networks
net32
net32_32

Output Data:
net32_outputs
net32_sim
net32_32_outputs

Target Data:
O_1

Error Data:
net32_errors
net32_32_errors

Input Delay States:

Layer Delay State

Name
network1

Network Properties

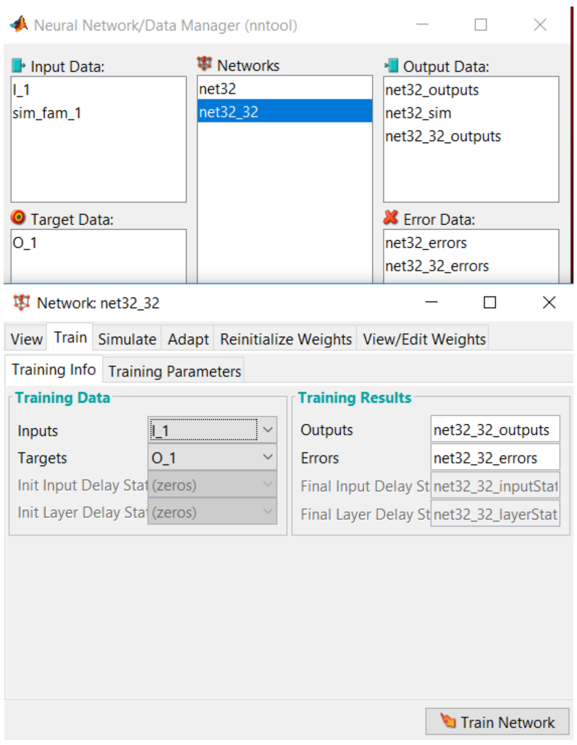
Network Type:	Feed-forward backprop
Input data:	I_1
Target data:	O_1
Training function:	TRAINBR
Adaption learning function:	LEARNGD
Performance function:	MSE
Number of layers:	3

Properties for: Layer 2

Number of neurons:	32
Transfer Function:	TANSIG

Figure 38: The NNtool of MATLAB

In Figure 39 lies the setup of the three networks for the case of the first product family.

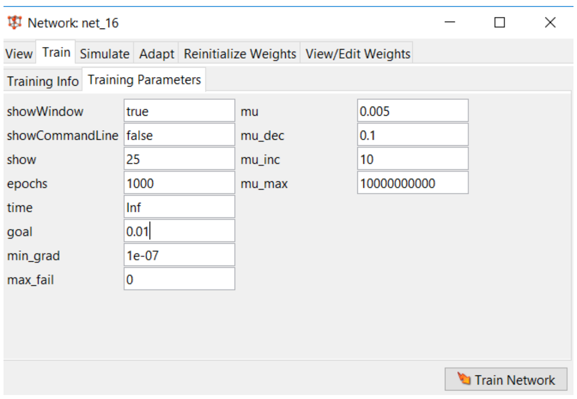


Input Data: I_1, sim_fam_1
Networks: net32, net32_32
Output Data: net32_outputs, net32_sim, net32_32_outputs
Target Data: O_1
Error Data: net32_errors, net32_32_errors

Training Data		Training Results	
Inputs	I_1	Outputs	net32_32_outputs
Targets	O_1	Errors	net32_32_errors
Init Input Delay Stat	(zeros)	Final Input Delay St	net32_32_inputStat
Init Layer Delay Sta	(zeros)	Final Layer Delay St	net32_32_layerStat

Figure 39: NNtool Set Up of 6 nets for 1st Product Family

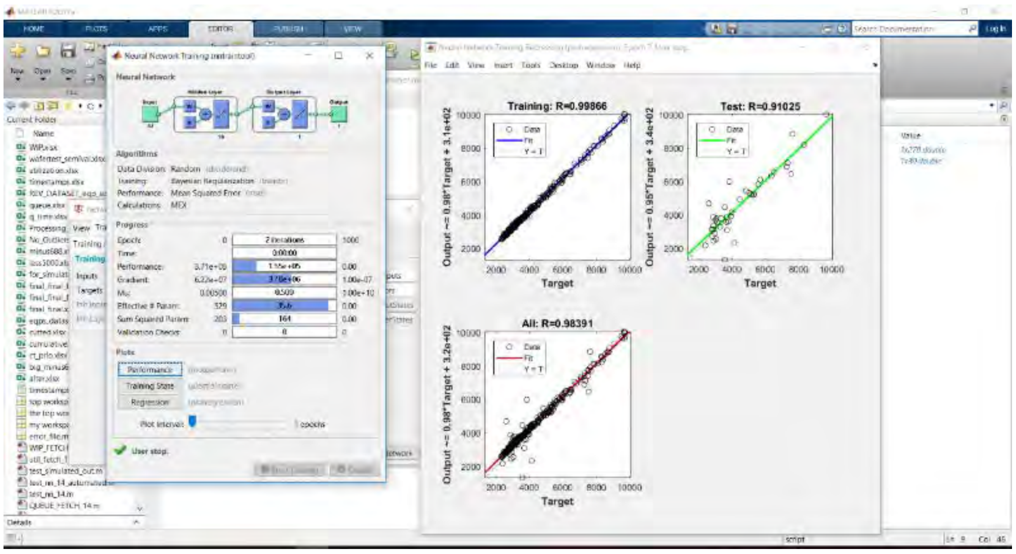
Also, in Figure 40 is shown how the training parameters of each net are set. The goal for the Mean Square Error is set to 0.01.



showWindow	true	mu	0.005
showCommandLine	false	mu_dec	0.1
show	25	mu_inc	10
epochs	1000	mu_max	10000000000
time	Inf		
goal	0.01		
min_grad	1e-07		
max_fail	0		

Figure 40: The training Parameters

Moving forward, the full picture is delivered for the first product family, and accordingly follow the rest of them in APPENDIX 3: Neural Network Training for All Product Families. In Figure 41 and Figure 42 the results of a training based on the first product family are shown concerning the one-hidden-layer strategy. More specifically, in Figure 43 the trained weights for the 32 neurons network as acquired by the NNtool are shown. Moreover, in Figure 44 the training based on the first product family of the two-hidden-layer strategy is depicted. As it will be analyzed, some of them seem promising and some not. We followingly ran a simulation for each network-dataset couple, using a random sample of records for each of the products, so as to assess the forecasting capability of the model.


Neural Network Training (nntraintool)

Algorithms
Data Division: Random (dividerand)
Training: Bayesian Regularization (trainbr)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

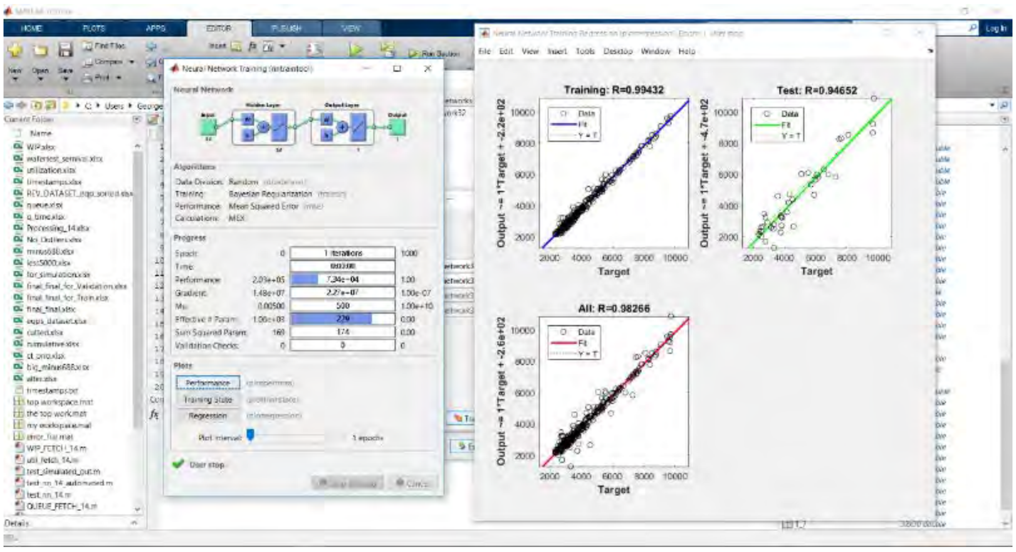
Epoch:	0	2 iterations	1000
Time:		0:00:00	
Performance:	3.71e+08	1.15e+05	0.00
Gradient:	6.22e+07	3.78e+06	1.00e-07
Mu:	0.00500	0.500	1.00e+10
Effective # Param:	329	35.6	0.00
Sum Squared Param:	203	164	0.00
Validation Checks:	0	0	0

Plot Interval: 1 epochs

User stop.

Training: R=0.99866
Test: R=0.91025
All: R=0.98391
Output ~= 0.98*Target + 3.1e+02
Output ~= 0.95*Target + 3.4e+02
Output ~= 0.98*Target + 3.2e+02
Target
Data
Fit
Y = T

Figure 41: Product Family 1 Train (1 Hidden Layer of 16 Neurons)


Neural Network Training (nntraintool)

Algorithms
Data Division: Random (dividerand)
Training: Bayesian Regularization (trainbr)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1 iterations	1000
Time:		0:00:00	
Performance:	2.03e+05	7.34e-04	1.00
Gradient:	1.48e+07	2.27e-07	1.00e-07
Mu:	0.00500	500	1.00e+10
Effective # Param:	1.06e+03	229	0.00
Sum Squared Param:	168	174	0.00
Validation Checks:	0	0	0

Plot Interval: 1 epochs

User stop.

Training: R=0.99432
Test: R=0.94652
All: R=0.98266
Output ~= 1*Target + -2.2e+02
Output ~= 1*Target + -4.7e+02
Output ~= 1*Target + -2.6e+02
Target
Data
Fit
Y = T

Figure 42: Product Family 1 Train (1 Hidden Layer of 32 Neurons)

Network net32
View Train Simulate Adapt Reinitialize Weights View/Edit Weights
Select the weight or bias to view: iw{1,1} - Weight to layer 1 from input 1
Revert Weight
Set Weight
Mean Sq
Performance goal met.
Target
326 Epochs

Figure 43: Product Family 1 Weights (1 Hidden Layer of 32 Neurons)

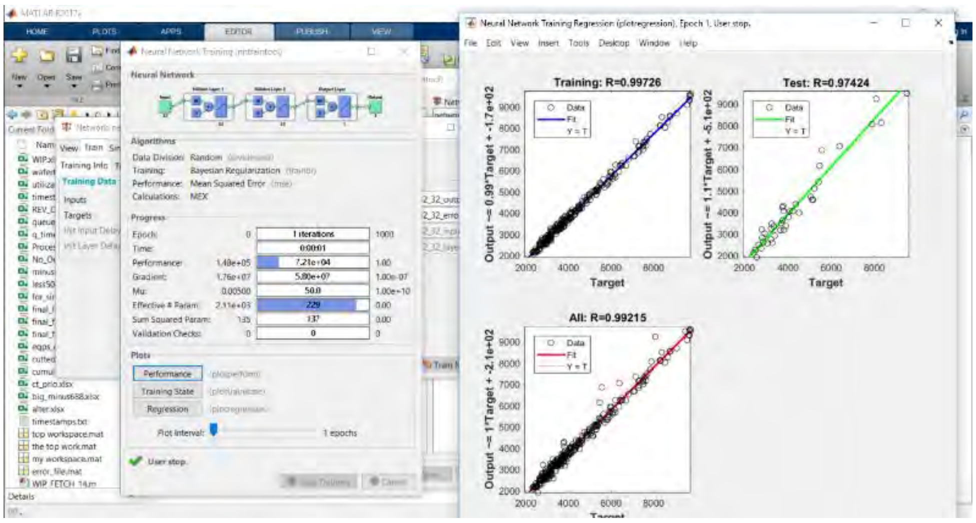
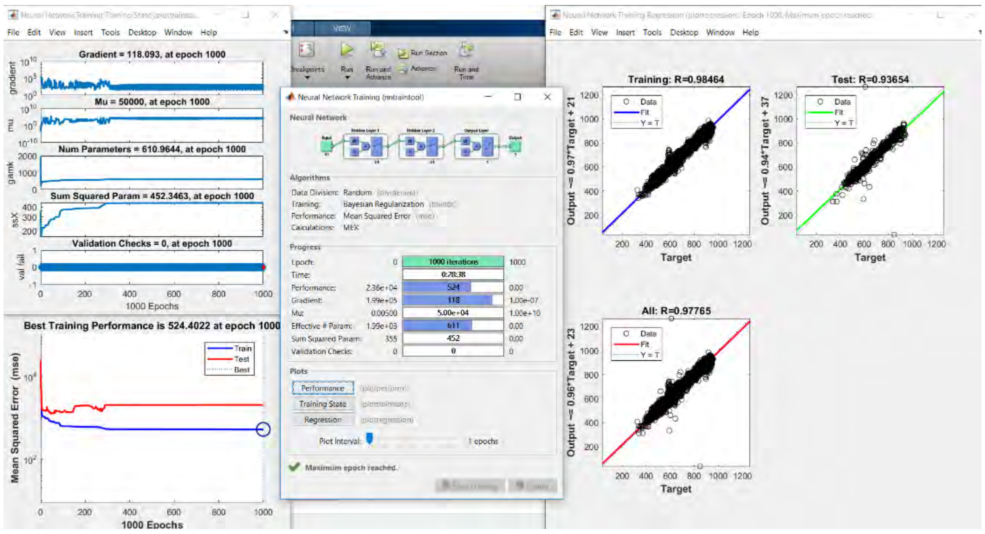

Neural Network Training (nntraintool)
Neural Network
Hidden Layer 1
Hidden Layer 2
Output Layer
Algorithms
Data Division: Random (dividerand)
Training: Bayesian Regularization (trainbr)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 1 iterations 1000
Time: 0:00:01
Performance: 1.48e+05 7.21e+04 1.00
Gradient: 1.76e+07 5.80e+07 1.00e-07
Mu: 0.00500 50.0 1.00e+10
Effective # Param: 2.11e+03 229 0.00
Sum Squared Param: 135 137 0.00
Validation Checks: 0 0 0
Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval: 1 epochs
User stop.
Stop Training
Cancel
Neural Network Training Regression (plotregression), Epoch 1, User stop.
Training: R=0.99726
Test: R=0.97424
All: R=0.99215
Output ~= 0.99*Target + -1.7e+02
Output ~= 1.1*Target + -5.1e+02
Output ~= 1*Target + -2.1e+02
Data
Fit
Y = T
Target

Figure 44: Product Family 1 Train (2 Hidden Layers of 32 Neurons)

Applying similarly for all the parts as well as on the whole data, one can get the results that are analyzed in CHAPTER 5. The weights and biases of the nets are saved, consisting actually the tool for each product family. The results can be found in APPENDIX 3: Neural Network Training for All Product Families. Also, in Figure 45 is shown the outcome when the model is applied on the whole data.


Gradient = 118.093, at epoch 1000
Mu = 50000, at epoch 1000
Num Parameters = 610.9644, at epoch 1000
Sum Squared Param = 452.3463, at epoch 1000
Validation Checks = 0, at epoch 1000
1000 Epochs
Best Training Performance is 524.4022 at epoch 1000
Mean Squared Error (mse)
Train
Test
Best
Training: R=0.98464
Test: R=0.93654
All: R=0.97765
Output ~= 0.97*Target + 21
Output ~= 0.94*Target + 37
Output ~= 0.96*Target + 23
Target
Data
Fit
Y = T

Neural Network Training (nntraintool)

Algorithms

```
Data Division:  Random  (dividerand)
Training:       Bayesian Regularization  (trainbr)
Performance:    Mean Squared Error  (mse)
Calculations:   MEX
```

Progress

```
Epoch:                 0      1000 iterations   1000
Time:                         0:28:38
Performance:           2.36e+04   524           0.00
Gradient:              1.99e+05   118           1.00e-07
Mu:                    0.00500    5.00e+04      1.00e+10
Effective # Param:     1.99e+03   611           0.00
Sum Squared Param:     355        452           0.00
Validation Checks:     0          0             0
```

Maximum epoch reached.

Figure 45: Neural Net on the whole Data

These results seem very promising of the analysis and the chosen method, and so are analyzed in the next chapter.

CHAPTER 5 Results

To test the effectiveness of the three ANNs in predicting CTs, we trained them and simulated them. As was mentioned earlier, the inputs of each ANN are:

1. The LOTID.
2. The part that the LOTID belongs to.
3. The LOTID priority.
4. The cumulative processing time that the LOTID spends on each equipment type.
5. The sum of the processing times of all the lots waiting to be processed in each equipment type at the time of entry of the LOTID.
6. The utilizations of the equipment-types, during the last 24 hours before the entry of the LOTID.
7. The total WIP (Work in Process) in the entire Wafer Test Fab at the time of entry of the LOTID.

The output is the CT of the LOTID.

At the early phases of training, we observed that there was an overfitting problem. To alleviate it, we monitored the training process and stopped it on purpose before it started to overfit the training data at the expense of underfitting the test data.

Table 1 shows the mean absolute percent errors (MAPE) of the predicted CTs, for the training and simulated data, for the three ANN architectures, for each product family and all products (last row). The number of training data for each product is shown in the second column. The number or LOTIDs used for the simulation was 10 for each product family.

Table 1: MAPE of predicted CTs for the three ANN architectures for each product family and all products

PART	# Tr. Data	16 ERR TRAIN	16 ERR SIM	32 ERR TRAIN	32 ERR SIM	32 32 ER R TRAIN	32 32 E RR SIM
[redacted]	[redacted]	7,38	14,92	10,52	21,53	10,48	22,41
[redacted]	[redacted]	11,58	25,54	10,15	27,97	11,84	16,15
[redacted]	[redacted]	22,55	9,76	22,51	8,43	16,21	8,89
[redacted]	[redacted]	9,65	8,85	7,37	9,87	8,13	4,25
[redacted]	[redacted]	15,01	8,79	15,14	19,68	67,83	11,68
[redacted]	[redacted]	12,19	22,29	11,52	32,96	13,23	18,36
[redacted]	[redacted]	6,93	17,90	14,48	19,39	4,64	18,12
[redacted]	[redacted]	7,52	21,39	7,48	13,33	6,37	12,33
[redacted]	[redacted]	28,75	10,52	18,32	17,33	10,14	26,72
[redacted]	[redacted]	9,19	24,44	7,16	20,67	5,97	15,78
[redacted]	[redacted]	7,40	18,21	3,78	17,13	2,42	19,23
[redacted]	[redacted]	11,89	18,96	7,43	30,12	5,31	28,26
[redacted]	[redacted]	5,13	22,76	6,69	17,36	6,03	32,05
[redacted]	[redacted]	3,02	14,77	3,94	24,70	4,39	15,61
[redacted]	[redacted]	28,55	18,07	25,42	9,61	21,89	10,11

In each row of Table 1, the ANN that resulted in better CT predictions for the simulated data is highlighted in yellow color. For example, in the last row, the ANN with 1 hidden layer and 32 neurons outperformed the other two ANNs with an average error of 9.61%, which is relatively good, considering the complexity of the problem. For three product families, the smallest MAPE value was less than 10% while for the remaining 11 families it was between 10% and 20%.

Table 2 shows the actual and predicted CTs, as well as the absolute percent error, for the 10 lots of the first part family ([redacted]), using the ANN with 1 Hidden Layer and 16 neurons.

Table 2: Prediction results for the 1st product family

LOTID	PART	CT	Sim 16	Absolute % Error
[redacted]	[redacted]	[redacted]	[redacted]	0,6402
[redacted]	[redacted]	[redacted]	[redacted]	1,2259
[redacted]	[redacted]	[redacted]	[redacted]	4,2666
[redacted]	[redacted]	[redacted]	[redacted]	6,8614
[redacted]	[redacted]	[redacted]	[redacted]	7,1785
[redacted]	[redacted]	[redacted]	[redacted]	9,9316
[redacted]	[redacted]	[redacted]	[redacted]	18,7692
[redacted]	[redacted]	[redacted]	[redacted]	30,7623
[redacted]	[redacted]	[redacted]	[redacted]	33,1850
[redacted]	[redacted]	[redacted]	[redacted]	36,3305

As can be seen from Table 2, the absolute percent error varies quite a bit from 0,36% to 36,33% between the lots. This large variation holds also for all the product families, the results of which can be found in APPENDIX 3: Neural Network Training for All Product Families.

At first, we thought that this variation, which led to poor CT predictions, was perhaps due to corrupted raw data and in particular data that indicated that the lots were being processed and the equipment was utilized when in fact it was incapacitated, resulting in long waiting/processing times.

To test this hypothesis, we used the information from the Equipment State (EQPS) Table. More specifically, we wrote an SQL script that draws the equipment state at the key time instances QUEUETIME, TRACKINTIME and TRACKOUTTIME and creates three new values in our MATLAB code, namely, Q_State, In_State and Out_State, describing the states of the equipment at these three time instances. The codes for this task are shown in Figure 46 and Figure 47, respectively.

However, as is shown in Figure 48, all the equipment turned out to be available at the times of interest and hence this hypothesis was not accepted.

```
SELECT EQPID, STATE, CHANGEDT
FROM EQPS
ORDER BY EQPID, CHANGEDT
;


--create eqps_dataset
select EQPID, CHANGEDT, LASTSTATE, STATE
FROM EQPS
WHERE EQPID IN
(SELECT DISTINCT EQPID
FROM REV_DATASET)
AND LASTSTATE IS NOT NULL
ORDER BY EQPID, CHANGEDT; --SADDLY, WE GET DATA ONLY FOR 16 OUT OF THE 95 EQUIPMENT
```

EQPID	CHANGEDT	LASTSTATE	STATE
[redacted]	[redacted]	[redacted]	[redacted]

Figure 46: SQL Code for EQPS Merge

```
for i=1:a
    for  j=1:c
        logicalIndex = ismember(eqpid(i),id(j));
        if logicalIndex == true
            if table2array(stamp(j,1))<table2array(q_time(i,1))
                q_state(i) = current_state(j);
                in_state(i) = current_state(j);
                out_state(i) = current_state(j);
            elseif table2array(stamp(j,1))>table2array(q_time(i,1)) && table2array(stamp(j,1
                in_state(i) = current_state(j);
                out_state(i) = current_state(j);
            elseif table2array(stamp(j,1))>table2array(in_time(i,1)) && table2array(stamp(j,
                out_state(i) = current_state(j);
            elseif table2array(stamp(j,1))>table2array(out_time(i,1))
                if states(i,:)==zeros(1,1:3)
                    q_state(i) = last_state(j);
                    in_state(i) = last_state(j);
                    out_state(i) = last_state(j);
                end
            else
                break
            end
        else
            continue
        end
    end
end
```

Figure 47: MATLAB Script for EQPS Merge

#	LOTID	PRIORITY	PART	STAGE	EQPID	EQPTYP	RECPID	QUEUETIME	TRACKINTIME	TRACKOUTTIMI	WT	PT	CT	TRACKINMAINQ	CURMAINQ	Q_State	In_State	Out_State
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Figure 48: Dataset with Equipment States

So, to get an idea of which input factors might be responsible for these large deviations in the errors, we compared the LOTIDs with the smallest and largest absolute percent error for three product families. The criterion for choosing these families was that the actual CTs of the LOTIDs with the smallest and largest absolute percent error of these families should be close, allowing us to better identify the factors that cause these errors. Table 3 shows the input factors for these LOTIDs as well as the actual and predicted CTs and the absolute percent error between them in the last three rows.

Table 3: Influencing factors tracing based on best & worst prediction for three product families

						OVER
						UNDER
LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
AVG_Error	0,575974	15,00909	3,933143	50,02806	6,132744	22,14989

From Table 3, it seems that certain factors, which are highlighted in yellow color, have a higher positive correlation with the errors in the CT forecasts generated by the ANNs than other factors. These factors are mainly the total WIP upon entry and the processing times and utilizations of certain equipment-types, with types 2 and 7 being the prominent ones.

CHAPTER 6 Summary and Future Directions

We developed a data-driven tool for predicting the CTs of wafer lots in the Wafer Test Fab of a Semiconductor Manufacturing Plant.

From our industrial partner's production data covering a 2-year period, we pulled and/or calculated for each LOTID information about the part family that the LOTID belongs to, the LOTID average priority, the cumulative processing time that the LOTID spends on each equipment type, the sum of the processing times of all the lots waiting to be processed in each equipment type at the time of entry of the LOTID, the utilizations of the equipment-types, during the last 24 hours before the entry of the LOTID, the total WIP in the entire Wafer Test Fab at the time of entry of the LOTID, and the actual CT of the LOTID. These calculations were performed using MATLAB in combination with Microsoft SQL Server.

This information was then used to train three ANN variants developed in MATLAB's NNtool for CT prediction.

This methodology was implemented on 14 product families processed on 10 equipment types in the Wafer Test Fab. The results showed that for three product families, the smallest MAPE value of the CT forecast was less than 10% while for the remaining 11 families it was between 10% and 20%. Considering all the product families together, the smallest MAPE value of the CT forecast was slightly less than 10%.

There are several directions for future work.

First, the raw data can be improved. There exist a lot of problematic data records, with excessively large processing times or waiting times, which made CT prediction difficult.

Moreover, Wang et al. [4] suggest a factor selection method as an extra preprocessing step. This method seems very effective and its application in the examined use case is promising.

There is also space for improving the design of the ANN as a predictive model. Some of the important features that it should have are analyzed in this dissertation, but the selection of its key parameters such as the number of hidden layers and neurons in each layer can be further researched.

Finally, it would be worthwhile to apply the developed methodology to the Wafer Fab, which is certainly more complex than the Wafer Test Fab. The data provided by our industrial partner covers all the fabs in the front and back end of the supply chain.

BIBLIOGRAPHY

[1] J. Zhang, W. Qin, L. H. Wu, and W. B. Zhai, “Fuzzy neural network-based rescheduling decision mechanism for semiconductor manufacturing,” Comput. Ind., vol. 65, no. 8, pp. 1115–1125, 2014.

[2] S. C. H. Lu, D. Ramaswamy, and P. R. Kumar, “Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants,” IEEE Trans. Semicond. Manuf., vol. 7, no. 3, pp. 374–388, 1994.

[3] Y. T. Tai, W. L. Pearn, and J. H. Lee, “Cycle time estimation for semiconductor final testing processes with Weibull-distributed waiting time,” Int. J. Prod. Res., vol. 50, no. 2, pp. 581–592, 2012.

[4] J. Wang, J. Zhang, and X. Wang, A Data Driven Cycle Time Prediction with Feature Selection in a Semiconductor Wafer Fabrication System. IEEE Trans. Semicond. Manuf., vol. 31, no. 1, 173-182, 2018.

[5] D. Y. Sha and S. Y. Hsu, “Due-date assignment in wafer fabrication using artificial neural networks,” Int. J. Adv. Manuf. Technol., vol. 23, nos. 9–10, pp. 768–775, 2004.

[6] H. Demuth and M. Beale, Neural Network Toolbox for Use with MATLAB®, User’s Guide, Version 4, The Mathworks, 2002.

[7] University of Thessaly, Department of Mechanical Engineering News. ([redacted]).

[8] J. G. Shanthikumar, S. Ding S, M. T. Zhang, “Queueing theory for semiconductor manufacturing systems: A survey and pen problems,” IEEE Transactions in Automation Science and Engineering vol. 4, no. 4, pp. 513-522, 2007.

[9] A. Kusiak and G. Xu, “Modeling and optimization of HVAC systems using a dynamic neural network,” Energy, vol. 42, no. 1, pp. 241–250, 2012.

[10] T. Chen and Y.-C. Wang, “Incorporating the FCM–BPN approach with nonlinear programming for internal due date assignment in a wafer fabrication plant,” Robot. Comput. Integr. Manuf., vol. 26, no. 1, pp. 83–91, 2010.

[11] I. Tirkel, “Forecasting flow time in semiconductor manufacturing using knowledge discovery in databases,” Int. J. Prod. Res., vol. 51, no. 18, pp. 5536–5548, 2013.

[12] J. Wang, J. Yang, J. Zhang, X. Wang & W. (C.) Zhang, “Big data driven cycle time parallel prediction for production planning in wafer manufacturing,” Enterprise Information Systems, vol. 12, no. 6, 714-732, 2018.

[13] A. Stamatakis and S. Botsis, “Reliability and maintainability analysis in semiconductor manufacturing, ”Diploma in Mechanical Engineering Thesis, University of Thessaly, 2018.

[14] A. R. Shinde, “Modeling and simulation of a Semiconductor Manufacturing Fab for cycle time analysis,” MSc. in Systems Engineering Dissertation, University of Maryland, 2018.

[15] L. Mönch, J. W. Fowler, S. J Mason, “Production Planning and Control for Semiconductor Wafer Fabrication Facilities - Modeling, Analysis, and

Systems,” Operations Research/Computer Science Interfaces Series, vol. 52, Springer, Ney York, 2013.

[16] Y. Tony, Understanding Neural Networks, Towards Data Science, 2019, [https://towardsdatascience.com/understanding-neural-networks-19020b758230].

[17] M.T. Hagan, H.B. Demuth, M.H. Beale, Neural Network Design, PWS Publishing Company, Boston, MA 1996.

[18] E. Tsiakiris and C. Kalantaridis, “Reliability and Prediction using Neural Networks in Semiconductor Manufacturing,” Diploma in Mechanical Engineering Thesis University of Thessaly, 2019.

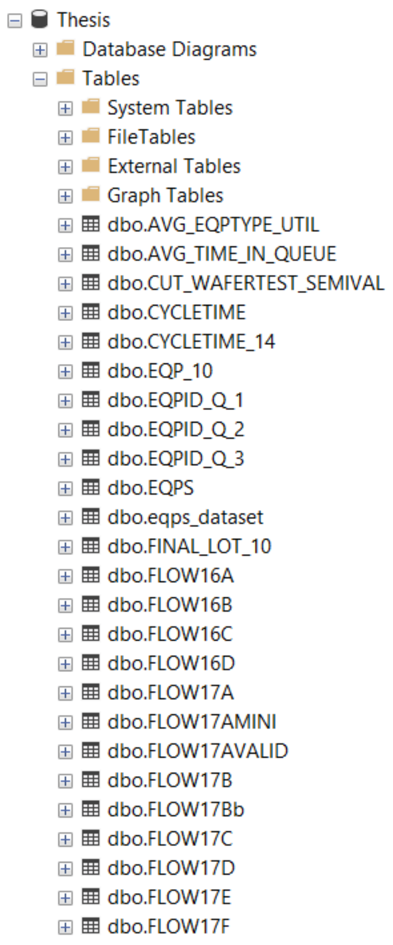
[19] K. Papadimitropoulos, “Implementation of a queueing network modeling method for analyzing semiconductor manufacturing systems,” Diploma in Mechanical Engineering Thesis, University of Thessaly, 2018.

[20] C. Kuo, C. Chien and J. Chen, "Manufacturing Intelligence to Exploit the Value of Production and Tool Data to Reduce Cycle Time," IEEE Transactions on Automation Science and Engineering, vol. 8, no. 1, pp. 103-111, 2011.

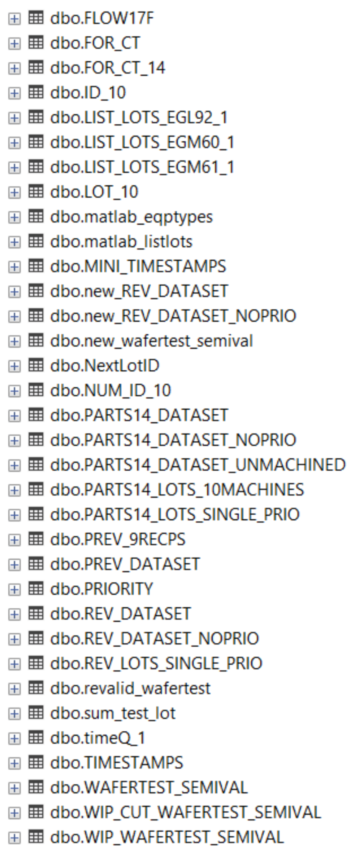
APPENDICES

APPENDIX 1: Code of Data Preprocessing (SQL)

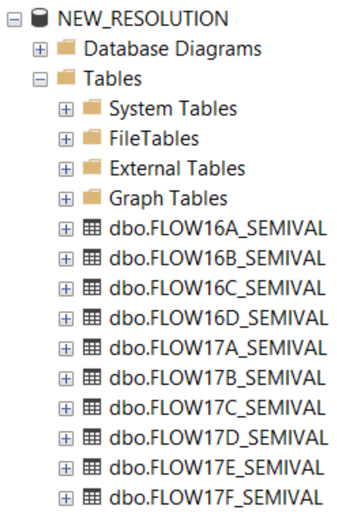
Here is a matrix with the total of databases we created through this analysis.



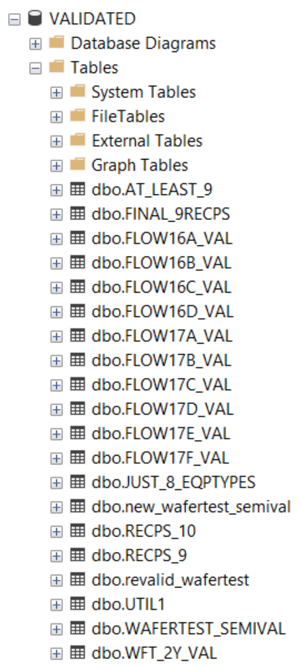
```
⊟ Thesis
  ⊞ Database Diagrams
  ⊟ Tables
    ⊞ System Tables
    ⊞ FileTables
    ⊞ External Tables
    ⊞ Graph Tables
    ⊞ dbo.AVG_EQPTYPE_UTIL
    ⊞ dbo.AVG_TIME_IN_QUEUE
    ⊞ dbo.CUT_WAFERTEST_SEMIVAL
    ⊞ dbo.CYCLETIME
    ⊞ dbo.CYCLETIME_14
    ⊞ dbo.EQP_10
    ⊞ dbo.EQPID_Q_1
    ⊞ dbo.EQPID_Q_2
    ⊞ dbo.EQPID_Q_3
    ⊞ dbo.EQPS
    ⊞ dbo.eqps_dataset
    ⊞ dbo.FINAL_LOT_10
    ⊞ dbo.FLOW16A
    ⊞ dbo.FLOW16B
    ⊞ dbo.FLOW16C
    ⊞ dbo.FLOW16D
    ⊞ dbo.FLOW17A
    ⊞ dbo.FLOW17AMINI
    ⊞ dbo.FLOW17AVALID
    ⊞ dbo.FLOW17B
    ⊞ dbo.FLOW17Bb
    ⊞ dbo.FLOW17C
    ⊞ dbo.FLOW17D
    ⊞ dbo.FLOW17E
    ⊞ dbo.FLOW17F
```



```
⊞ dbo.FLOW17F
⊞ dbo.FOR_CT
⊞ dbo.FOR_CT_14
⊞ dbo.ID_10
⊞ dbo.LIST_LOTS_EGL92_1
⊞ dbo.LIST_LOTS_EGM60_1
⊞ dbo.LIST_LOTS_EGM61_1
⊞ dbo.LOT_10
⊞ dbo.matlab_eqptypes
⊞ dbo.matlab_listlots
⊞ dbo.MINI_TIMESTAMPS
⊞ dbo.new_REV_DATASET
⊞ dbo.new_REV_DATASET_NOPRIO
⊞ dbo.new_wafertest_semival
⊞ dbo.NextLotID
⊞ dbo.NUM_ID_10
⊞ dbo.PARTS14_DATASET
⊞ dbo.PARTS14_DATASET_NOPRIO
⊞ dbo.PARTS14_DATASET_UNMACHINED
⊞ dbo.PARTS14_LOTS_10MACHINES
⊞ dbo.PARTS14_LOTS_SINGLE_PRIO
⊞ dbo.PREV_9RECPS
⊞ dbo.PREV_DATASET
⊞ dbo.PRIORITY
⊞ dbo.REV_DATASET
⊞ dbo.REV_DATASET_NOPRIO
⊞ dbo.REV_LOTS_SINGLE_PRIO
⊞ dbo.revalid_wafertest
⊞ dbo.sum_test_lot
⊞ dbo.timeQ_1
⊞ dbo.TIMESTAMPS
⊞ dbo.WAFERTEST_SEMIVAL
⊞ dbo.WIP_CUT_WAFERTEST_SEMIVAL
⊞ dbo.WIP_WAFERTEST_SEMIVAL
```



```
⊟ NEW_RESOLUTION
  ⊞ Database Diagrams
  ⊟ Tables
    ⊞ System Tables
    ⊞ FileTables
    ⊞ External Tables
    ⊞ Graph Tables
    ⊞ dbo.FLOW16A_SEMIVAL
    ⊞ dbo.FLOW16B_SEMIVAL
    ⊞ dbo.FLOW16C_SEMIVAL
    ⊞ dbo.FLOW16D_SEMIVAL
    ⊞ dbo.FLOW17A_SEMIVAL
    ⊞ dbo.FLOW17B_SEMIVAL
    ⊞ dbo.FLOW17C_SEMIVAL
    ⊞ dbo.FLOW17D_SEMIVAL
    ⊞ dbo.FLOW17E_SEMIVAL
    ⊞ dbo.FLOW17F_SEMIVAL
```



```
⊟ VALIDATED
  ⊞ Database Diagrams
  ⊟ Tables
    ⊞ System Tables
    ⊞ FileTables
    ⊞ External Tables
    ⊞ Graph Tables
    ⊞ dbo.AT_LEAST_9
    ⊞ dbo.FINAL_9RECPS
    ⊞ dbo.FLOW16A_VAL
    ⊞ dbo.FLOW16B_VAL
    ⊞ dbo.FLOW16C_VAL
    ⊞ dbo.FLOW16D_VAL
    ⊞ dbo.FLOW17A_VAL
    ⊞ dbo.FLOW17B_VAL
    ⊞ dbo.FLOW17C_VAL
    ⊞ dbo.FLOW17D_VAL
    ⊞ dbo.FLOW17E_VAL
    ⊞ dbo.FLOW17F_VAL
    ⊞ dbo.JUST_8_EQPTYPES
    ⊞ dbo.new_wafertest_semival
    ⊞ dbo.RECPS_10
    ⊞ dbo.RECPS_9
    ⊞ dbo.revalid_wafertest
    ⊞ dbo.UTIL1
    ⊞ dbo.WAFERTEST_SEMIVAL
    ⊞ dbo.WFT_2Y_VAL
```

In this chapter lies all the SQL Code we created throughout this dissertation.

To begin with, the following 7 queries were developed on the recipe-based analysis.

1.Data Preprocess

```
SELECT LOTID, COUNT(DISTINCT QUEUETIME)
FROM FLOW17A
WHERE ProdArea = '[redacted]'--WAFERTEST
GROUP BY LOTID
ORDER BY COUNT(DISTINCT QUEUETIME) DESC;

--THIS IS A TEST FOR THE VALIDATION RULES OF THE WHOLE DATASET
--CREATE FLOW17A VALIDATED
SELECT  LOTID,  PART,  STAGE,  EQPID,  EQPTYPE,  RECPID,  QUEUETIME,  TRACKINTIME,
TRACKOUTTIME,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKINTIME AS DATETIME)) AS
WT,
DATEDIFF(MINUTE, CAST(TRACKINTIME AS datetime), CAST(TRACKOUTTIME AS DATETIME))
AS PT,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKOUTTIME AS DATETIME)) AS
CT,
TRACKINMAINQTY, CURMAINQTY
FROM FLOW17A
WHERE ProdArea = '[redacted]'--for wafertest
AND QUEUETIME IS NOT NULL
AND TRACKINTIME IS NOT NULL
AND TRACKOUTTIME IS NOT NULL
AND RECPID IS NOT NULL
AND EQPID IS NOT NULL
AND QUEUETIME != ''
AND TRACKINTIME != ''
AND TRACKOUTTIME != ''
AND RECPID != ''
AND EQPID != ''
AND TRACKOUTTIME - QUEUETIME > 0
ORDER BY PART, LOTID, QUEUETIME;

--SELECT TOP 5004 *
--FROM FLOW17AVALID;

--SELECT *
--FROM FLOW17AMINI
--INNER JOIN PRIORITY
--ON FLOW17AMINI.LOTID = PRIORITY.LOTID;

--SELECT *
--FROM PRIORITY
--WHERE LOTID = '6583332.1';

SELECT COUNT(DISTINCT RECPID) --flow17aValid for 442 lotIDs
FROM FLOW17AMINI;

SELECT *
FROM FLOW17AVALID
where RECPID = '[redacted]' --sos exw diaforetika ct gia to idio recipe!!!
ORDER BY LOTID ASC;

select distinct RECPID, count(RECPID), count(distinct LOTID)
FROM FLOW17AMINI
group by RECPID
```

```
ORDER BY count(RECPID) DESC;

select *
from FLOW17AVALID
where LOTID = [redacted];

SELECT *
FROM FLOW17B;
```

2. Flow Validation

```
--FLOW VALIDATED
--i run this validated query for all the 2month raw data groups
--i reload them as FLOW##X_VAL in the VALIDATED Database
SELECT  LOTID,  PART,  STAGE,  EQPID,  EQPTYPE,  RECPID,  QUEUETIME,  TRACKINTIME,
TRACKOUTTIME,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKINTIME AS DATETIME)) AS
WT,
DATEDIFF(MINUTE, CAST(TRACKINTIME AS datetime), CAST(TRACKOUTTIME AS DATETIME))
AS PT,
DATEDIFF(MINUTE, CAST(QUEUETIME AS datetime), CAST(TRACKOUTTIME AS DATETIME)) AS
CT,--MINUTES ALL OF THEM
TRACKINMAINQTY, CURMAINQTY
FROM FLOW17F
WHERE ProdArea = [redacted]'
AND PART IS NOT NULL
AND QUEUETIME IS NOT NULL
AND TRACKINTIME IS NOT NULL
AND TRACKOUTTIME IS NOT NULL
AND RECPID IS NOT NULL
AND EQPID IS NOT NULL
AND QUEUETIME != ''
AND TRACKINTIME != ''
AND TRACKOUTTIME != ''
AND RECPID != ''
AND EQPID != ''
--AND  DATEDIFF(MINUTE,  CAST(QUEUETIME  AS  datetime),  CAST(TRACKOUTTIME  AS
DATETIME)) > 0
--AND  DATEDIFF(MINUTE,  CAST(TRACKINTIME  AS  datetime),  CAST(TRACKOUTTIME  AS
DATETIME)) > 0
ORDER BY QUEUETIME;
```

3.Tables Connect

```
--VALIDATED
SELECT *
FROM FLOW16A_VAL
UNION ALL
SELECT *
FROM FLOW16B_VAL
UNION ALL
SELECT *
FROM FLOW16C_VAL
UNION ALL
SELECT *
FROM FLOW16D_VAL
UNION ALL
SELECT *
FROM FLOW17A_VAL
```

```
UNION ALL
SELECT *
FROM FLOW17B_VAL
UNION ALL
SELECT *
FROM FLOW17C_VAL
UNION ALL
SELECT *
FROM FLOW17D_VAL
UNION ALL
SELECT *
FROM FLOW17E_VAL
UNION ALL
SELECT *
FROM FLOW17F_VAL
;--714182

SELECT *
FROM FLOW16A_VAL
UNION
SELECT *
FROM FLOW16B_VAL
UNION
SELECT *
FROM FLOW16C_VAL
UNION
SELECT *
FROM FLOW16D_VAL
UNION
SELECT *
FROM FLOW17A_VAL
UNION
SELECT *
FROM FLOW17B_VAL
UNION
SELECT *
FROM FLOW17C_VAL
UNION
SELECT *
FROM FLOW17D_VAL
UNION
SELECT *
FROM FLOW17E_VAL
UNION
SELECT *
FROM FLOW17F_VAL
ORDER BY QUEUETIME
;--710501 no douplicates
--from here we create wafertest2yearsvalid

SELECT *
FROM WFT_2Y_VAL;

--CREATE ONLY FOR 2 YEARS
SELECT *
FROM WFT_2Y_VAL
WHERE QUEUETIME > '[redacted]'
AND TRACKOUTTIME < '[redacted]';

--SELECT *
--FROM FLOW16A;
--UNION ALL
```

```
--SELECT *
--FROM FLOW16B;
----ON FLOW16A.QUEUETIME = FLOW16B.QUEUETIME;
----FULL JOIN FLOW16C ON FLOW16A.LOTID = FLOW16C.LOTID
----JOIN FLOW16D ON FLOW16A.LOTID = FLOW16D.LOTID
----FULL JOIN FLOW17A ON FLOW16A.LOTID = FLOW17A.LOTID
----FULL JOIN FLOW17B ON FLOW16A.LOTID = FLOW17B.LOTID
----FULL JOIN FLOW17C ON FLOW16A.LOTID = FLOW17C.LOTID
----FULL JOIN FLOW17D ON FLOW16A.LOTID = FLOW17D.LOTID
----FULL JOIN FLOW17E ON FLOW16A.LOTID = FLOW17E.LOTID
----FULL JOIN FLOW17F ON FLOW16A.LOTID = FLOW17F.LOTID
----ORDER BY FLOW16A.QUEUETIME, FLOW16B.QUEUETIME, FLOW16C.QUEUETIME;

--SELECT count(*)
--FROM COMBO_FLOW;

--SELECT *
--FROM COMBO_FLOW;

--SELECT count(*)
--FROM FLOW16B;

----INSERT INTO COMBO_FLOW
--  SELECT *
--    FROM FLOW16A
--UNION ALL
--  SELECT *
--    FROM FLOW16B;

--With TEMP as
--(
--select *
--from FLOW16A
--UNION ALL
--select *
--from FLOW16B)

--select *
--from TEMP;
```

4.Recipes PT

```
--DATABASE: VALIDATED
SELECT *
FROM WFT_2Y_VAL;--710501 records

SELECT DISTINCT LOTID, COUNT(DISTINCT RECPID)
FROM WFT_2Y_VAL
GROUP BY LOTID
ORDER BY COUNT(DISTINCT RECPID) DESC
;--we see that each lot passes from 1 up to 14 different recipes in the wafertest

SELECT DISTINCT RECPID, COUNT(DISTINCT LOTID)
FROM WFT_2Y_VAL
GROUP BY RECPID
ORDER BY COUNT(DISTINCT LOTID) DESC
;--101

SELECT DISTINCT LOTID, COUNT(DISTINCT eqptype)
FROM WFT_2Y_VAL
```

```
GROUP BY LOTID
ORDER BY COUNT(DISTINCT eqptype) DESC
;--we see that each lot passes from 1 up to 12 different eqpytypes in the wafertest

--and as we can guess...
SELECT RECPID, COUNT(DISTINCT eqptype)
FROM WFT_2Y_VAL
GROUP BY RECPID
ORDER BY COUNT(DISTINCT eqptype) DESC
;--...here we prove that we have 1to1 match of a recipe to an eqptype

SELECT eqptype, COUNT(DISTINCT RECPID) dist_recipes
FROM WFT_2Y_VAL
GROUP BY eqptype
ORDER BY COUNT(DISTINCT RECPID) DESC
;--however we see that an eqptype may perform more than one recipes

SELECT RECPID, COUNT(DISTINCT eqpid)
FROM WFT_2Y_VAL
GROUP BY RECPID
ORDER BY COUNT(DISTINCT eqpid) DESC
;--

SELECT eqpid, COUNT(DISTINCT RECPID)
FROM WFT_2Y_VAL
GROUP BY eqpid
ORDER BY COUNT(DISTINCT RECPID) DESC
;--

--RECIPES
SELECT COUNT(DISTINCT LOTID)
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
--OR RECPID = 'MAPOV.05'
--OR RECPID = 'MCCVM.02'
--OR RECPID = 'TRINK.01'--77137
--OR RECPID = 'INKN1.06'--77185
--OR RECPID = 'MAKRS.02'--77196
--OR RECPID = 'ICHT1.03'--77333
--OR RECPID = 'OFFRV.01'--77342
--OR RECPID = 'LTPDS.07'--77369
--OR RECPID = 'SIGHT.03'--77372
;

SELECT DISTINCT RECPID, LOTID--PER LOT -->RECIPES
FROM WFT_2Y_VAL
--GROUP BY LOTID
ORDER BY LOTID;

--TO BE IMPROVED BY SELECTING LOTS WITH PT>0!!!
--RECP1
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;--75664

--RECP2
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = [redacted]'
ORDER BY QUEUETIME;--69470
```

```
--RECP3
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;

--RECP4
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;

--RECP5
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;

--RECP6
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;

--RECP7
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;

--RECP8
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;--3/16-8/17

--RECP9
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME;--FOR SOME REASON THE LOTIDS HERE DO NOT APPEAR IN OTHER RECPS

--RECP10
SELECT DISTINCT LOTID,QUEUETIME
FROM WFT_2Y_VAL
WHERE RECPID = '[redacted]'
ORDER BY QUEUETIME; --2/17-8/17

--SELECT *
--FROM recp10
--WHERE LOTID ='EFA56.1';

--AT_LEAST_9
select recp1.LOTID
from recp1
INNER JOIN recp2
ON recp2.LOTID = recp1.LOTID--68597
INNER JOIN recp3
ON recp3.LOTID = recp1.LOTID--66232
INNER JOIN recp4
ON recp4.LOTID = recp1.LOTID--47876
INNER JOIN recp5
```

```
ON recp5.LOTID = recp1.LOTID--39334
INNER JOIN recp6
ON recp6.LOTID = recp1.LOTID--22199
INNER JOIN recp7
ON recp7.LOTID = recp1.LOTID--16214
INNER JOIN recp8
ON recp8.LOTID = recp2.LOTID--11767
INNER JOIN recp9
ON recp9.LOTID = recp2.LOTID--5648
;

SELECT COUNT(DISTINCT LOTID)
FROM WFT_2Y_VAL
WHERE LOTID IN(SELECT * FROM AT_LEAST_9);--CHECK AT LEAST 9RECPS, 5648

SELECT DISTINCT LOTID, COUNT(DISTINCT RECPID)
FROM WFT_2Y_VAL
WHERE LOTID IN(SELECT * FROM AT_LEAST_9)
GROUP BY LOTID
ORDER BY COUNT(DISTINCT RECPID)
;--CHECK FOR MORE THAN 9RECPS, 3942

--COMBINING WE GET JUST 9--> JUST_9
SELECT COUNT(DISTINCT LOTID)
FROM WFT_2Y_VAL
WHERE LOTID IN(SELECT * FROM JUST_9);--3942

--FINAL_9RECPS
SELECT *
FROM WFT_2Y_VAL
WHERE LOTID IN(SELECT * FROM JUST_9)
ORDER BY LOTID, QUEUETIME;--39608

SELECT *
FROM FINAL_9RECPS;--39608
```

5.EQP

```
--VALIDATED

SELECT DISTINCT(EQPTYPE)
FROM FINAL_9RECPS;

SELECT DISTINCT(EQPTYPE), COUNT(DISTINCT RECPID)
FROM FINAL_9RECPS
GROUP BY EQPTYPE;

SELECT DISTINCT(RECPID), COUNT(DISTINCT EQPID)
FROM FINAL_9RECPS
GROUP BY RECPID;

SELECT *
FROM FINAL_9RECPS;
```

6.UTIL+Q_TIME

```
--validated
```

```
--utilization
SELECT   *   --SUM(PT)   AS   TOTAL_PT   --/1052640?     --2YEARS(IN   MINS)   =
731DAYS*24HOURS*60MINS
FROM WFT_2Y_VAL
WHERE
--EQPID IN (SELECT EQPID FROM WFT_2Y_VAL WHERE
EQPTYPE = '[redacted]'
--OR EQPTYPE = 'EDDCA'
--OR EQPTYPE = 'EACFFZ'
--OR EQPTYPE = 'EAFDBF'
--OR EQPTYPE = 'EZDEAF'
--OR EQPTYPE = 'EAEECA'
--OR EQPTYPE = 'EACDCC'
--OR EQPTYPE = 'ECAAFC'
--OR EQPTYPE = 'ECA-80/88'
--AND PT < 5000
--GROUP BY EQPID
order by eqpid, TRACKINTIME, pt desc
;

SELECT *
FROM WFT_2Y_VAL
WHERE EQPID = '[redacted]'--12.000
ORDER BY PT DESC
;

SELECT *
FROM UTIL1;

--AVG UTIL EQPTYPE
SELECT WFT_2Y_VAL.EQPTYPE, AVG(UTIL1.UTIL) AS AVG_UTIL
FROM UTIL1
INNER JOIN WFT_2Y_VAL
ON UTIL1.EQPID = WFT_2Y_VAL.EQPID
GROUP BY WFT_2Y_VAL.EQPTYPE;

--time in queue
SELECT EQPTYPE, AVG(WT) AS TOTAL_WT
FROM WFT_2Y_VAL
WHERE
--EQPID IN (SELECT EQPID FROM WFT_2Y_VAL WHERE
EQPTYPE = '[redacted]'
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
OR EQPTYPE = ''
--AND WT < 5000
GROUP BY EQPTYPE
ORDER BY EQPTYPE
;
```

7.Dataset Creation

```
--THESIS
--+PRIORITY

--SELECT DISTINCT LOTID, COUNT(DISTINCT PRIORITY)
--FROM PRIORITY
```

```
--GROUP BY LOTID
--ORDER BY COUNT(DISTINCT PRIORITY) DESC ;--3942
----AND I SELECT FROM HERE THE 2442 LOTS WITH 1 PRIO FROM THE 3942

--SELECT DISTINCT FINAL_9RECPS.LOTID
--FROM FINAL_9RECPS;--3942

----SO WE GET THIS
--SELECT DISTINCT FINAL_9RECPS.LOTID, LOTS_SINGLE_PRIO.PRIORITY
--FROM FINAL_9RECPS
--JOIN LOTS_SINGLE_PRIO
--ON LOTS_SINGLE_PRIO.LOTID = FINAL_9RECPS.LOTID;--2442

--select *
--from LOTS_SINGLE_PRIO;--2442

SELECT DISTINCT (LOTID)
FROM FINAL_9RECPS;--3942

--PRIORITY RE-CALC-->AVERAGE PRIORITY
--CREATE LOTS_SINGLE_PRIO
SELECT DISTINCT LOTID, AVG(PRIORITY) as PRIORITY
FROM PRIORITY
WHERE LOTID IN (SELECT LOTID FROM FINAL_9RECPS)
GROUP BY LOTID
ORDER BY AVG(PRIORITY) DESC ;--3942

--PROOF THAT 1 LOT, MORE THAN ONE PRIOS--> WE NEED THE AVERAGE ABOVE
SELECT LOTID, PRIORITY
FROM PRIORITY
WHERE LOTID = '[redacted]';

SELECT *
FROM LOTS_SINGLE_PRIO;

--AND FINALLY
--LAST TABLE OF 3942 WITH PRIORITY-->DATASET
SELECT FINAL_9RECPS.LOTID,
LOTS_SINGLE_PRIO.PRIORITY,
FINAL_9RECPS.PART,
FINAL_9RECPS.STAGE,
FINAL_9RECPS.EQPID,
FINAL_9RECPS.EQPTYPE,
FINAL_9RECPS.RECPID,
FINAL_9RECPS.QUEUETIME,
FINAL_9RECPS.TRACKINTIME,
FINAL_9RECPS.TRACKOUTTIME,
FINAL_9RECPS.WT,
FINAL_9RECPS.PT,
FINAL_9RECPS.CT,
FINAL_9RECPS.TRACKINMAINQTY,
FINAL_9RECPS.CURMAINQTY
FROM FINAL_9RECPS
INNER JOIN LOTS_SINGLE_PRIO
ON LOTS_SINGLE_PRIO.LOTID = FINAL_9RECPS.LOTID
ORDER BY FINAL_9RECPS.LOTID, FINAL_9RECPS.QUEUETIME;--39608

SELECT DISTINCT LOTID, PRIORITY
FROM DATASET
ORDER BY LOTID;

--VALIDATION FOR MATLAB TIME IN QUEUE
```

```
select LOTID,EQPTYPE,WT
from dataset
ORDER BY LOTID, RECPID;

SELECT LOTID,RECPID,PT
FROM DATASET
ORDER BY LOTID, RECPID;

SELECT EQPTYPE ,SUM(PT) AS TOTAL_PT
FROM DATASET
GROUP BY EQPTYPE
;

SELECT *
FROM DATASET
ORDER BY pt;

--UTIL calc on VALIDATED - WFT_2Y_VAL
--SELECT DATEDIFF()--[redacted]

SELECT *
FROM EQPTYPE_UTIL;
```

Continuing on, as explained in the main course of this dissertation, the equipment-based modeling of the fab was selected. Hence, below lie the queries developed on this basis.

First of all, the united semi validated data are used.

1.Data Reunion

```
--DATABASE:NEW_RESOLUTION
SELECT *
FROM FLOW16A_SEMIVAL
UNION
SELECT *
FROM FLOW16B_SEMIVAL
UNION
SELECT *
FROM FLOW16C_SEMIVAL
UNION
SELECT *
FROM FLOW16D_SEMIVAL
UNION
SELECT *
FROM FLOW17A_SEMIVAL
UNION
SELECT *
FROM FLOW17B_SEMIVAL
UNION
SELECT *
FROM FLOW17C_SEMIVAL
UNION
SELECT *
FROM FLOW17D_SEMIVAL
UNION
SELECT *
FROM FLOW17E_SEMIVAL
UNION
SELECT *
FROM FLOW17F_SEMIVAL
ORDER BY QUEUETIME
;
```

2. Equipment based Analysis

```
--VALIDATED

select *
from wafertest_semival;--the new uncutted data

SELECT DISTINCT LOTID, COUNT(DISTINCT eqptype) AS DIST_eqptype
FROM wafertest_semival
GROUP BY LOTID
ORDER BY COUNT(DISTINCT eqptype) DESC
;

--LOTS WITH 10 RECPS ->RECPS_10 TABLE
SELECT DISTINCT LOTID
FROM wafertest_semival
```

```
--WHERE PT>0, IF WE PUT IT, WE MAY RUIN THE FAMILY(PART)
--WHERE PART = 'PHC8T' --CHECK FOR RANDOM PART, THAT IS A RESULT OF THE 2ND NEXT
QUERY
GROUP BY LOTID
HAVING COUNT(DISTINCT RECPID) = 10 --ALL WITH JUST 10 RECIPES
;
--CHECKING THE ABOVE QUERY FOR A LOTID
SELECT DISTINCT(RECPID)
FROM WAFERTEST_SEMIVAL
WHERE LOTID = '[redacted]';

--from this Query i'll get 3 or 14 PARTS, depending on the analysis

SELECT DISTINCT PART, COUNT(DISTINCT LOTID) AS DIST_LOTID
FROM wafertest_semival
WHERE
--PT>0
--AND
LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY PART
ORDER BY COUNT(DISTINCT LOTID) DESC
;--NOW I CAN CHOOSE TO KEEP THE PARTS WITH SUFFICIENT AMOUNT OF DISTINCT LOTS

--CHECK_2:TO SEE THESE PART HOW MANY LOTS THEY HAVE, IE WHAT IS THE PERCENTAGE
OF THE 9RECP LOTS
SELECT DISTINCT EQPTYPE --LOTID, COUNT(DISTINCT EQPTYPE) AS DIST_EQPTYPE
FROM WAFERTEST_SEMIVAL
WHERE PART ='[redacted]';
--GROUP BY LOTID
--ORDER BY COUNT(DISTINCT EQPTYPE) DESC;--1126/1218 = 0,9244
--I CAN KEEP THE FIRST 3 PARTS
--PART DIST_LOTID RATIO
--PMVBL       1386       0,9878
--PNA30       1126       0,9244
--P4VV4       923        0,8969

SELECT DISTINCT EQPTYPE
FROM wafertest_semival;

--FROM THIS QUERY I GET THE 10 standard eqptypes
SELECT DISTINCT EQPTYPE, COUNT(DISTINCT LOTID) AS DIST_LOTS --THIS ONE IS TO SHOW
THAT ALL LOTS OF THE PART PASS FROM ALL EQPPTYPES
FROM wafertest_semival
WHERE
--PT>0 AND
PART ='[redacted]' AND
LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY EQPTYPE
ORDER BY COUNT(DISTINCT LOTID) DESC
;
SELECT DISTINCT EQPTYPE, COUNT(DISTINCT LOTID) AS DIST_LOTS
FROM wafertest_semival
WHERE
--PT>0 AND
PART ='[redacted]' AND
LOTID IN
(SELECT *
FROM RECPS_10)
```

```
GROUP BY EQPTYPE
ORDER BY COUNT(DISTINCT LOTID) DESC
;
SELECT DISTINCT EQPTYPE, COUNT(DISTINCT LOTID) AS DIST_LOTS
FROM wafertest_semival
WHERE
--PT>0 AND
PART ='[redacted]' AND
LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY EQPTYPE
ORDER BY COUNT(DISTINCT LOTID) DESC
; --I ADE THE CHECK ALSO FOR THE TOP 14 PARTS
--SOS:SINCE FOR THE 3 PARTS, EXCACTLY THE SAME, WE KEEP THESE 10 EQPTYPES
--EACDCC
--EACFFZ
--EAEECA
--EAFDBF
--ECA-80/88
--ECAAFC
--EDBFDFCDZ
--EDDCA
--EEAADFFCZ
--EZDEAF

--FOR ALL PARTS, WE PROVE THE MOST POPULAR MACHINES
SELECT DISTINCT EQPTYPE, COUNT(DISTINCT PART) AS DIST_PARTS
FROM wafertest_semival
WHERE
--PT>0 AND
LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY EQPTYPE
ORDER BY COUNT(DISTINCT PART) DESC
;
--AS WE CAN SEE, WE HAVE CHOSEN PARTS THAT PASS THROUGH THE 10 MOST POPULAR TYPES

--HERE WE SEE FOR A SPECIFIC PART, THAT IN AN EQPTYPE MORE THAN ONE RECIPES ARE
EXECUTED
SELECT DISTINCT EQPTYPE, COUNT(DISTINCT RECPID) AS DIST_RECPS
FROM wafertest_semival
WHERE
--PT>0 AND
PART ='[redacted]' AND
LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY EQPTYPE
ORDER BY COUNT(DISTINCT RECPID) DESC
;

--WE SEE THAT A LOT MAY PASS MORE THAN ONCE FROM A SPECIFIC EQPTYPE-->WATCH OUT
FOR THE PROCESSING TIMES
SELECT *
FROM wafertest_semival
WHERE PART ='[redacted]
order by LOTID;

--HERE WE PROVE THAT LOTS OF THE SAME PART DOES NOT HAVE THE SAME # OF RECIPES
THAT THEY EXECUTE, BUT MOST OF THEM DO...GOOD RATIO
```

```
SELECT DISTINCT LOTID, COUNT(DISTINCT RECPID) AS DIST_RECPID
FROM wafertest_semival
WHERE PART = '[redacted]' --A RANDOM PART AS A CHECK VALUE
--AND PT>0
GROUP BY LOTID
ORDER BY COUNT(DISTINCT RECPID) DESC--78119
;

--REVISED DATASET, WITHOUT PRIOS -->REV_DATASET_NOPRIO
SELECT *
FROM WAFERTEST_SEMIVAL
WHERE (PART = '[redacted]' OR PART = '[redacted]' OR PART = '[redacted]')
AND (EQPTYPE = '[redacted] OR EQPTYPE = [redacted]' OR EQPTYPE = [redacted]' OR EQPTYPE =
'[redacted]' OR EQPTYPE = '[redacted]' OR EQPTYPE = '[redacted]'
OR EQPTYPE = '[redacted]' OR EQPTYPE = [redacted]' OR EQPTYPE = '[redacted]' OR EQPTYPE
= [redacted]')
--AND LOTID IN
--(SELECT *
--FROM RECPS_10)      --!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ORDER BY PART, LOTID, QUEUETIME
;--37892 ROWS

--NOTE:: IN 7.14PARTS, I GET THE SAME ANALYSIS FOR 14PARTS.

--WE GET 3435 DISTINCT LOTS
SELECT DISTINCT LOTID, COUNT(DISTINCT EQPTYPE)
FROM WAFERTEST_SEMIVAL
WHERE (PART = '[redacted]' OR PART = '[redacted]' OR PART = [redacted]')
AND (EQPTYPE = '[redacted] OR EQPTYPE = '[redacted]' OR EQPTYPE = [redacted]' OR EQPTYPE =
'[redacted]' OR EQPTYPE = '[redacted] OR EQPTYPE = [redacted]'
OR EQPTYPE = '[redacted]' OR EQPTYPE = [redacted]OR EQPTYPE = [redacted]' OR EQPTYPE
= [redacted]')
AND LOTID IN
(SELECT *
FROM RECPS_10)
GROUP BY LOTID
ORDER BY COUNT(DISTINCT EQPTYPE);

---with new_wafertest_semival (LIMIT TO 7200MINS)

--NEW REVISED DATASET, WITHOUT PRIOS -->REV_DATASET_NOPRIO
SELECT *
FROM new_wafertest_semival
WHERE (PART = [redacted]' OR PART = [redacted] OR PART = [redacted])
AND (EQPTYPE = [redacted] OR EQPTYPE = '[redacted]' OR EQPTYPE = [redacted]' OR EQPTYPE =
'[redacted]' OR EQPTYPE = [redacted]' OR EQPTYPE = [redacted]'
OR EQPTYPE = '[redacted] OR EQPTYPE = '[redacted] OR EQPTYPE = '[redacted]' OR EQPTYPE
= [redacted]')
--AND LOTID IN
--(SELECT *
--FROM RECPS_10)
ORDER BY PART, LOTID, QUEUETIME
;--37892 ROWS
```

3.Dataset creation and Data Analysis Validation of MATLAB

```
--THESIS
--+PRIORITY

--PRIORITY RE-CALC-->AVERAGE PRIORITY
--CREATE REV_LOTS_SINGLE_PRIO
```

```
SELECT DISTINCT LOTID, AVG(PRIORITY) as PRIORITY
FROM PRIORITY
WHERE LOTID IN (SELECT DISTINCT LOTID FROM REV_DATASET_NOPRIO)--EDW PAIZOYN OI
ALLAGES KATHE FORA
GROUP BY LOTID
ORDER BY AVG(PRIORITY) DESC ;--2338/3435 WE LOST DATA FOR WHICH NO PRIORITY INFO
EXIST

SELECT *
FROM REV_LOTS_SINGLE_PRIO;

select *
from WAFERTEST_SEMIVAL
where LOTID = '[redacted]'
order by QUEUETIME asc;

--AND FINALLY
--LAST TABLE OF 2338 WITH PRIORITY-->REV_DATASET
SELECT REV_DATASET_NOPRIO.LOTID,
REV_LOTS_SINGLE_PRIO.PRIORITY,
REV_DATASET_NOPRIO.PART,
REV_DATASET_NOPRIO.STAGE,
REV_DATASET_NOPRIO.EQPID,
REV_DATASET_NOPRIO.EQPTYPE,
REV_DATASET_NOPRIO.RECPID,
REV_DATASET_NOPRIO.QUEUETIME,
REV_DATASET_NOPRIO.TRACKINTIME,
REV_DATASET_NOPRIO.TRACKOUTTIME,
REV_DATASET_NOPRIO.WT,
REV_DATASET_NOPRIO.PT,
REV_DATASET_NOPRIO.CT,
REV_DATASET_NOPRIO.TRACKINMAINQTY,
REV_DATASET_NOPRIO.CURMAINQTY
FROM REV_DATASET_NOPRIO
JOIN REV_LOTS_SINGLE_PRIO
ON REV_LOTS_SINGLE_PRIO.LOTID = REV_DATASET_NOPRIO.LOTID
ORDER        BY        REV_DATASET_NOPRIO.PART,        REV_DATASET_NOPRIO.LOTID,
REV_DATASET_NOPRIO.QUEUETIME;--25614

SELECT DISTINCT EQPTYPE
FROM REV_DATASET;

SELECT DISTINCT RECPID
FROM REV_DATASET;

SELECT DISTINCT LOTID, PART
FROM REV_DATASET
ORDER BY PART;

SELECT DISTINCT LOTID, PRIORITY
FROM REV_DATASET
ORDER BY LOTID;--2338 lots

SELECT *
FROM WAFERTEST_SEMIVAL;--775415

--ANALYSIS STARTS

--CREATE TIMESTAMPS TABLE

SELECT DISTINCT LOTID
FROM REV_DATASET
```

```sql
ORDER BY LOTID;--2338 lots

--timestamp table
SELECT PART,LOTID, MIN(QUEUETIME) AS Q_IN, MAX(TRACKOUTTIME) AS T_OUT
FROM REV_DATASET
GROUP BY PART, LOTID
ORDER BY PART, LOTID ASC
;--2338
SELECT *
FROM REV_DATASET;

SELECT *
FROM TIMESTAMPS
ORDER BY Q_IN ASC
;--SOS: WE SEE THAT TIMESTAMPS START FROM THE [redacted]"

----HENCE, WE WILL KEEP THE DATA FROM 3 months before THAT STAMP AND ON FROM THE
WAFERTEST_SEMIVAL
--SELECT *
--FROM WAFERTEST_SEMIVAL
--WHERE QUEUETIME > '[redacted]'
--ORDER BY PART, LOTID, QUEUETIME
--;--CUT_WAFERTEST_SEMIVAL

SELECT LOTID, MIN(QUEUETIME) AS Q_IN, MAX(TRACKOUTTIME) AS T_OUT
FROM WAFERTEST_SEMIVAL
GROUP BY LOTID
ORDER BY MIN(QUEUETIME) ASC
;--WIP_WAFERTEST_SEMIVAL

SELECT *
FROM WAFERTEST_SEMIVAL
ORDER BY  PART, LOTID,EQPTYPE ASC;
--CT
SELECT LOTID, PART, PRIORITY, CT
FROM new_REV_DATASET
ORDER BY PART, LOTID, EQPTYPE;--FOR_CT

SELECT * FROM FOR_CT;

SELECT LOTID, SUM(CT) AS CT
FROM FOR_CT
GROUP BY LOTID;--CYCLETIME

--THE FINAL CYCLE TIME TABLE
SELECT DISTINCT FOR_CT.LOTID, FOR_CT.PART, FOR_CT.PRIORITY, CYCLETIME.CT
FROM FOR_CT
RIGHT JOIN CYCLETIME
ON FOR_CT.LOTID = CYCLETIME.LOTID
ORDER BY FOR_CT.PART, FOR_CT.LOTID;

--select *
--from new_wafertest_semival;-- 7200 limit

select *
from REV_DATASET
ORDER BY PART, LOTID, EQPTYPE;--processing

--BIG TEST FOR WIP
select *
from mini_timestamps;
```

```
--WIP OF 1ST LOT CHECK
SELECT COUNT(DISTINCT LOTID)
FROM WIP_WAFERTEST_SEMIVAL
--WHERE LOTID = '[redacted]'
WHERE Q_IN<='[redacted]'
AND T_OUT>='[redacted]'
;--750

--WHILE  ((SELECT  Q_IN  FROM  WIP_WAFERTEST_SEMIVAL)  <=  (SELECT  Q_IN  FROM
MINI_TIMESTAMPS))  and
--((SELECT   T_OUT   FROM   WIP_WAFERTEST_SEMIVAL)   >=   (SELECT   Q_IN   FROM
MINI_TIMESTAMPS))
--BEGIN
--     SELECT LOTID FROM WIP_WAFERTEST_SEMIVAL
--         IF  (SELECT  Q_IN  FROM  WIP_WAFERTEST_SEMIVAL)  >  (SELECT  Q_IN  FROM
MINI_TIMESTAMPS)
--          BREAK;
--END

--WIP OF 2nd LOT CHECK
SELECT *
FROM WIP_WAFERTEST_SEMIVAL
--WHERE LOTID = '[redacted]
WHERE Q_IN<='[redacted]'
AND T_OUT>='[redacted]'
order by Q_IN
;--800

--WIP OF 3rd LOT CHECK
SELECT *
FROM WIP_WAFERTEST_SEMIVAL
--WHERE LOTID = [redacted]'
WHERE Q_IN<='[redacted]
AND T_OUT>='[redacted]
order by Q_IN
;--802

--waiting time testing
select LOTID, EQPID, EQPTYPE,QUEUETIME,TRACKINTIME,TRACKOUTTIME,PT
from WAFERTEST_SEMIVAL
where LOTID in (select LOTID from LIST_LOTS_EGL92_1)
AND QUEUETIME <= '[redacted]0000'
AND TRACKINTIME >= '[redacted]'
ORDER BY PART, LOTID,EQPTYPE ASC
;--create preprocessed data for train lot_1 waiting in matlab --timeQ_1

--#1
select distinct eqpid,eqptype
from WAFERTEST_SEMIVAL
where EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
order by eqptype asc;--175 -->ID_10

SELECT EQPID, SUM(PT) AS TIME_IN_QUEUE
FROM WAFERTEST_SEMIVAL
WHERE LOTID IN (SELECT LOTID FROM LIST_LOTS_EGL92_1)
AND QUEUETIME <= '[redacted]'
AND TRACKINTIME >= '[redacted]'
GROUP BY EQPID
ORDER BY EQPID ASC
;--EQPID_Q_1

SELECT *
```

```
FROM ID_10;

SELECT ID_10.EQPID, ISNULL(AVG(WAFERTEST_SEMIVAL.PT),0)
FROM ID_10
left JOIN WAFERTEST_SEMIVAL
ON ID_10.EQPID = WAFERTEST_SEMIVAL.EQPID
AND WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1)
AND WAFERTEST_SEMIVAL.QUEUETIME <= '[redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]'
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY ID_10.EQPID
ORDER BY ID_10.EQPID ASC
;--EQPID_Q_1

SELECT     WAFERTEST_SEMIVAL.lotid,   EQP_10.EQPTYPE,   WAFERTEST_SEMIVAL.pt   --
ISNULL(AVG(WAFERTEST_SEMIVAL.PT),0)
FROM EQP_10
left JOIN WAFERTEST_SEMIVAL
ON EQP_10.EQPTYPE = WAFERTEST_SEMIVAL.EQPTYPE
--AND WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1)
AND WAFERTEST_SEMIVAL.QUEUETIME <= '[redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]'
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
--GROUP BY EQP_10.EQPTYPE
ORDER BY EQP_10.EQPTYPE ASC
;--EQPID_Q_1

select *
from new_WAFERTEST_SEMIVAL
where lotid = '[redacted]'
order by eqptype asc;

select *
from NUM_ID_10;

--the golden one
SELECT ISNULL(AVG(WAFERTEST_SEMIVAL.PT),0)
FROM matlab_eqptypes
LEFT JOIN WAFERTEST_SEMIVAL
ON matlab_eqptypes.EQPTYPE = WAFERTEST_SEMIVAL.EQPTYPE
--AND WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1)
AND WAFERTEST_SEMIVAL.QUEUETIME <= '[redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]'
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY matlab_eqptypes.EQPTYPE
ORDER BY matlab_eqptypes.EQPTYPE ASC
;--EQPID_Q_1

SELECT EQP_10.EQPTYPE, ISNULL(AVG(WAFERTEST_SEMIVAL.PT), 0)
FROM EQP_10
LEFT JOIN WAFERTEST_SEMIVAL
ON EQP_10.EQPTYPE = WAFERTEST_SEMIVAL.EQPTYPE
WHERE WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGL92_1)
AND WAFERTEST_SEMIVAL.QUEUETIME <= [redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]'
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY EQP_10.EQPTYPE
ORDER BY EQP_10.EQPTYPE ASC
;--EQPID_Q_1

select *
from matlab_eqptypes;
```

```
select *
from matlab_listlots;

select distinct lotid
from WAFERTEST_SEMIVAL
where  WAFERTEST_SEMIVAL.QUEUETIME <= '[redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]';

SELECT SUM(PT)
FROM WAFERTEST_SEMIVAL
WHERE LOTID IN (SELECT LOTID FROM matlab_listlots)
AND QUEUETIME <= '[redacted]'
AND TRACKINTIME >= '[redacted]'
AND EQPTYPE IN(SELECT EQPTYPE FROM matlab_eqptypes)
GROUP BY EQPTYPE
ORDER BY EQPTYPE ASC
;--EQPID_Q_1
SELECT COUNT(DISTINCT EQPTYPE)
FROM WAFERTEST_SEMIVAL
WHERE LOTID IN (SELECT LOTID FROM matlab_listlots)
AND EQPTYPE IN(SELECT EQPTYPE FROM matlab_eqptypes)
AND QUEUETIME <= '[redacted]'
AND TRACKINTIME >= '[redacted]'
;
SELECT SUM(WAFERTEST_SEMIVAL.PT)
FROM WAFERTEST_SEMIVAL
INNER JOIN matlab_listlots
ON WAFERTEST_SEMIVAL.LOTID = matlab_listlots.list
INNER JOIN matlab_eqptypes
ON WAFERTEST_SEMIVAL.EQPTYPE = matlab_eqptypes.eqptype
where WAFERTEST_SEMIVAL.QUEUETIME <= '[redacted]'
AND WAFERTEST_SEMIVAL.TRACKINTIME >= '[redacted]'
GROUP BY WAFERTEST_SEMIVAL.EQPTYPE
ORDER BY WAFERTEST_SEMIVAL.EQPTYPE ASC
;

--#1
SELECT DISTINCT EQPTYPE
FROM WAFERTEST_SEMIVAL
WHERE EQPID IN (SELECT EQPID FROM EQPID_Q_1)
ORDER BY EQPTYPE ASC
;

SELECT DISTINCT EQPID_Q_1.EQPID, EQPID_Q_1.TIME_IN_QUEUE AS TIME_IN_QUEUE
FROM EQPID_Q_1
INNER JOIN WAFERTEST_SEMIVAL
ON WAFERTEST_SEMIVAL.EQPID = EQPID_Q_1.EQPID
WHERE (EQPTYPE = '[redacted]' OR EQPTYPE = [redacted] EQPTYPE = '[redacted]' OR EQPTYPE
= '[redacted] OR EQPTYPE = [redacted] OR EQPTYPE = '[redacted]
OR EQPTYPE = 'E[redacted] OR EQPTYPE = [redacted] EQPTYPE = [redacted] OR EQPTYPE
= [redacted]')
;--AND WE KEEP ONLY THE 10 THAT WE ARE INTERESTED ABOUT

--#2
SELECT EQPID, SUM(PT) AS TIME_IN_QUEUE
FROM WAFERTEST_SEMIVAL
WHERE LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1)
AND QUEUETIME <= '[redacted]'
AND TRACKINTIME >= '[redacted]'
GROUP BY EQPID
ORDER BY EQPID ASC
;--EQPID_Q_2
```

```
--#3
SELECT EQPID, SUM(PT) AS TIME_IN_QUEUE
FROM WAFERTEST_SEMIVAL
WHERE LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM61_1)
AND QUEUETIME <= '[redacted]'
AND TRACKINTIME >= '[redacted]'
GROUP BY EQPID
ORDER BY EQPID ASC
;--EQPID_Q_3

--PRE UTILIZATION
--PRE UTILIZATION

select *
from matlab_eqptypes;
select *
from matlab_listlots;

SELECT *
FROM ID_10;

--CREATE TABLE NUM_ID_10
select matlab_eqptypes.EQPTYPE, count(distinct ID_10.eqpid) as num_eqpid
from matlab_eqptypes
LEFT JOIN ID_10
ON matlab_eqptypes.EQPTYPE = ID_10.EQPTYPE
group by matlab_eqptypes.EQPTYPE
order by matlab_eqptypes.EQPTYPE asc;

select eqptype --, count(ID_10.eqpid)
from ID_10
--group by ID_10.EQPTYPE
--order by ID_10.EQPTYPE asc
;

SELECT * FROM NUM_ID_10;
--UTILIZATION
--UTILIZATION
--UTILIZATION

--it works
SELECT NUM_ID_10.EQPTYPE, SUM((cast(DATEDIFF(
MINUTE,          [dbo].[Max2]         (WAFERTEST_SEMIVAL.TRACKINTIME,'[redacted]
[redacted]),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME,  '[redacted]'))as
float)/1440)) as sum1
FROM NUM_ID_10
LEFT JOIN WAFERTEST_SEMIVAL
ON NUM_ID_10.EQPTYPE = WAFERTEST_SEMIVAL.EQPTYPE
WHERE
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1) AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >=    '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=    '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    >=   '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >=    '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKINTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
```

```
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY NUM_ID_10.EQPTYPE
ORDER BY NUM_ID_10.EQPTYPE ASC
;--sum_test_lot
select sum_test_lot.sum1 / NUM_ID_10.num_eqpid
from sum_test_lot
left join NUM_ID_10
on sum_test_lot.eqptype = NUM_ID_10.eqptype;

--the one
SELECT NUM_ID_10.EQPTYPE, SUM((cast(DATEDIFF(
MINUTE,          [dbo].[Max2]         (WAFERTEST_SEMIVAL.[redacted]
[redacted]'),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME, '[redacted]'))as
float)/1440)) / (NUM_ID_10.num_eqpid)
FROM matlab_eqptypes
       LEFT    JOIN    WAFERTEST_SEMIVAL    ON    matlab_eqptypes.EQPTYPE    =
WAFERTEST_SEMIVAL.EQPTYPE
       LEFT JOIN NUM_ID_10 ON matlab_eqptypes.EQPTYPE = NUM_ID_10.EQPTYPE
WHERE
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1) AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >=    '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   >=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKINTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY NUM_ID_10.EQPTYPE, NUM_ID_10.num_eqpid
ORDER BY NUM_ID_10.EQPTYPE ASC
;

--ID_10.EQPID
SELECT ID_10.EQPID, SUM((cast(DATEDIFF(
MINUTE,          [dbo].[Max2]         (WAFERTEST_SEMIVAL.TRACKINTIME,'[redacted]
[redacted]),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME, [redacted]'))as
float)/1440))
FROM ID_10
LEFT JOIN WAFERTEST_SEMIVAL
ON ID_10.EQPID = WAFERTEST_SEMIVAL.EQPID
WHERE
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1) AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >=    '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   >=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKINTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <=   '[redacted]'   AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY ID_10.EQPID
ORDER BY ID_10.EQPID ASC
;
```

```
--ID_10.eqptype
SELECT  ID_10.eqptype, sum(cast(DATEDIFF(
MINUTE,         [dbo].[Max2]         (WAFERTEST_SEMIVAL.TRACKINTIME,'[redacted]
[redacted]'),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME, '[redacted]'))as
float)/1440)
FROM ID_10
LEFT JOIN WAFERTEST_SEMIVAL
ON ID_10.EQPID = WAFERTEST_SEMIVAL.EQPID
WHERE
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1) AND
--WAFERTEST_SEMIVAL.EQPTYPE IN (SELECT EQPTYPE FROM matlab_eqptypes) AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >=     '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= [redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=     '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= [redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    >=   '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >=   [redacted]'    AND
WAFERTEST_SEMIVAL.TRACKINTIME <= [redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY ID_10.eqptype
ORDER BY ID_10.eqptype ASC
;

--ONLY WAFERTEST_SEMIVAL, TO VALIDATE THE VALUES THROUGH EXCEL WHEN SUMING BY
EQPTYPE
SELECT LOTID, eqptype, EQPID, TRACKINTIME, TRACKOUTTIME, (cast(DATEDIFF(
MINUTE,         [dbo].[Max2]         (WAFERTEST_SEMIVAL.[redacted]
[redacted]'),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME, '[redacted]'))as
float)/1440)
--,COUNT()
FROM WAFERTEST_SEMIVAL
WHERE
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1) AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >=     '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=     '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    >=   [redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >=     '[redacted]'    AND
WAFERTEST_SEMIVAL.TRACKINTIME <= [redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME    <=   [redacted]'    AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
--GROUP BY matlab_eqptypes.EQPTYPE
ORDER BY eqptype, EQPID, TRACKINTIME ASC
;

--ONLY WAFERTEST_SEMIVAL, SUM
SELECT  eqptype, sum(cast(DATEDIFF(
MINUTE,         [dbo].[Max2]         (WAFERTEST_SEMIVAL.TRACKINTIME,'[redacted]
[redacted]'),
[dbo].[Min2] (WAFERTEST_SEMIVAL.TRACKOUTTIME, '[redacted]'))as
float)/1440)
--,COUNT()
FROM WAFERTEST_SEMIVAL
WHERE
```

```
--WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGM60_1)
--AND
(WAFERTEST_SEMIVAL.TRACKINTIME    >= [redacted]'      AND
WAFERTEST_SEMIVAL.TRACKOUTTIME <= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <= [redacted]      AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    <=   '[redacted]'      AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   >= [redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME    >= [redacted]'      AND
WAFERTEST_SEMIVAL.TRACKINTIME <= [redacted]
OR   (WAFERTEST_SEMIVAL.TRACKINTIME   <=  '[redacted]'     AND
WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]')
--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY eqptype
--ORDER BY eqptype, EQPID, TRACKINTIME ASC
;

--VALIDATION FOR MATLAB processing TIME
select PART,LOTID,EQPTYPE,PT
from REV_DATASET
ORDER BY PART,LOTID,EQPTYPE;

--VALIDATION FOR MATLAB TIME IN QUEUE
select LOTID,EQPTYPE,WT
from REV_DATASET
ORDER BY LOTID, RECPID;

SELECT LOTID,RECPID,PT
FROM REV_DATASET
ORDER BY LOTID, RECPID;

SELECT EQPTYPE ,SUM(PT) AS TOTAL_PT
FROM REV_DATASET
GROUP BY EQPTYPE
;

SELECT *
FROM REV_DATASET
ORDER BY pt;

--UTIL calc on VALIDATED - WFT_2Y_VAL
--SELECT DATEDIFF()--[redacted]

SELECT *
FROM AVG_EQPTYPE_UTIL;
```

4.Revalidated Dataset with a time limitation of 7200mins.

```
--re validate big data
--max time in state, 5 days = 7200mins

SELECT LOTID, PART, EQPID, EQPTYPE, QUEUETIME, TRACKINTIME, TRACKOUTTIME,
[dbo].[MINIMUM]   (WT,7200)   AS   WT,   [dbo].[MINIMUM]   (PT,7200)   AS   PT,
(([dbo].[MINIMUM] (WT,7200)) + ([dbo].[MINIMUM] (PT,7200))) AS CT
FROM WAFERTEST_SEMIVAL
ORDER BY QUEUETIME ASC; --ALL DATA IS KEPT

--we get new_wafertest_semival

SELECT DISTINCT LOTID, AVG(PRIORITY) as PRIORITY --new_REV_DATASET_NOPRIO
FROM PRIORITY
```

```
WHERE LOTID IN (SELECT LOTID FROM LOT_10)
GROUP BY LOTID
ORDER BY AVG(PRIORITY) DESC ;--2185 FINAL_LOT_10

--AND FINALLY
--LAST TABLE OF 2338 WITH PRIORITY-->REV_DATASET
SELECT new_REV_DATASET_NOPRIO.LOTID,
REV_LOTS_SINGLE_PRIO.PRIORITY,
new_REV_DATASET_NOPRIO.PART,
new_REV_DATASET_NOPRIO.EQPID,
new_REV_DATASET_NOPRIO.EQPTYPE,
new_REV_DATASET_NOPRIO.QUEUETIME,
new_REV_DATASET_NOPRIO.TRACKINTIME,
new_REV_DATASET_NOPRIO.TRACKOUTTIME,
new_REV_DATASET_NOPRIO.WT,
new_REV_DATASET_NOPRIO.PT,
new_REV_DATASET_NOPRIO.CT
FROM new_REV_DATASET_NOPRIO
JOIN REV_LOTS_SINGLE_PRIO
ON REV_LOTS_SINGLE_PRIO.LOTID = new_REV_DATASET_NOPRIO.LOTID
ORDER      BY      new_REV_DATASET_NOPRIO.PART,      new_REV_DATASET_NOPRIO.LOTID,
new_REV_DATASET_NOPRIO.QUEUETIME;
--25614

---we get new_REV_DATASET

--ALSO THE THRESHOLD SHOULD BE HERE

SELECT EQP_10.EQPTYPE, ISNULL(AVG(WAFERTEST_SEMIVAL.PT), 0)
FROM EQP_10
LEFT JOIN WAFERTEST_SEMIVAL
ON EQP_10.EQPTYPE = WAFERTEST_SEMIVAL.EQPTYPE
WHERE WAFERTEST_SEMIVAL.LOTID IN (SELECT LOTID FROM LIST_LOTS_EGL92_1)
AND WAFERTEST_SEMIVAL.QUEUETIME <= [redacted]'
AND WAFERTEST_SEMIVAL.TRACKOUTTIME >= '[redacted]'

AND PT<1440   ----SOS: OLO TO ZOUMI EDW

--AND EQPTYPE IN(SELECT EQPTYPE FROM EQP_10)
GROUP BY EQP_10.EQPTYPE
ORDER BY EQP_10.EQPTYPE ASC
;--EQPID_Q_1
```

5.EQPS Merge

```
SELECT EQPID, STATE, CHANGEDT
FROM EQPS
ORDER BY EQPID, CHANGEDT
;

--create eqps_dataset
select EQPID, CHANGEDT, LASTSTATE, STATE
FROM EQPS
WHERE EQPID IN
(SELECT DISTINCT EQPID
FROM REV_DATASET)
AND LASTSTATE IS NOT NULL
ORDER  BY  EQPID,  CHANGEDT;  --SADDLY,  WE  GET  DATA  ONLY  FOR  16  OUT  OF  THE  95
EQUIPMENTS
```

```
SELECT DISTINCT EQPID
FROM REV_DATASET;--95

select EQPID, CHANGEDT,LASTSTATE, STATE
FROM EQPS
WHERE EQPID IN
(SELECT DISTINCT EQPID
FROM WAFERTEST_SEMIVAL) --test for WAFERTEST_SEMIVAL
AND LASTSTATE IS NOT NULL
and EQPID != '[redacted]'
and EQPID != '[redacted]
ORDER BY EQPID, CHANGEDT; --SADDLY, WE GET DATA ONLY FOR 34 OUT OF THE 236 EQPs

SELECT DISTINCT EQPID
FROM WAFERTEST_SEMIVAL;--236

SELECT * FROM EQPS WHERE EQPID = '[redacted];

--SELECT * FROM WAFERTEST_SEMIVAL;

SELECT EQPID, QUEUETIME, TRACKINTIME, TRACKOUTTIME FROM REV_DATASET
WHERE EQPID IN
(SELECT DISTINCT EQPID
FROM EQPS)
order by EQPID, QUEUETIME
;--WE WANT TO MERGE THE EQPS DATA HERE.

select * --REV_DATASET.LOTID, eqps_dataset.EQPID, REV_DATASET.QUEUETIME,
eqps_dataset.LASTSTATE
from eqps_dataset
left join REV_DATASET
on eqps_dataset.EQPID = REV_DATASET.EQPID
--and REV_DATASET.QUEUETIME > eqps_dataset.CHANGEDT
order by REV_DATASET.LOTID, REV_DATASET.QUEUETIME
;

--FOR BIG PTs
SELECT WAFERTEST_SEMIVAL.LOTID, WAFERTEST_SEMIVAL.EQPID,
WAFERTEST_SEMIVAL.EQPTYPE, WAFERTEST_SEMIVAL.TRACKINTIME,
WAFERTEST_SEMIVAL.TRACKINTIME, WAFERTEST_SEMIVAL.TRACKOUTTIME, EQPS.CHANGEDT,
EQPS.STATE
FROM WAFERTEST_SEMIVAL
INNER JOIN EQPS
ON WAFERTEST_SEMIVAL.EQPID = EQPS.EQPID
WHERE WAFERTEST_SEMIVAL.TRACKINTIME > EQPS.CHANGEDT
ORDER BY WAFERTEST_SEMIVAL.EQPID, WAFERTEST_SEMIVAL.TRACKINTIME
;

--FOR BIG QTs
SELECT WAFERTEST_SEMIVAL.EQPID, WAFERTEST_SEMIVAL.EQPTYPE,
WAFERTEST_SEMIVAL.QUEUETIME, EQPS.CHANGEDT, EQPS.STATE
FROM WAFERTEST_SEMIVAL
INNER JOIN EQPS
ON WAFERTEST_SEMIVAL.EQPID = EQPS.EQPID
ORDER BY WAFERTEST_SEMIVAL.EQPID, WAFERTEST_SEMIVAL.QUEUETIME
;

select *
from NextLotID
where dest_prodarea = '[redacted]'
or src_prodarea = '[redacted]'
;
```

APPENDIX 2: Code of Data Preprocessing (MATLAB)

1.Recipe-based First Strategy

```
%Table Creation from DataBase (FLOW17AMINI)
%  A->Cell  B->Num

%SOS for A(cell),B(numeric) from (Dataset): INSERT LOTID,RECPID
ASC ORDER!!
%SOS for U,(dataset) from (eqp_util): INSERT IN ASC ORDER!!
%SOS  for  PR(numeric)  from  (LAST_LOTS_prio):  INSERT  IN  ASC
ORDER!!

lotid = A(:,1);
prio = B(:,1);
part = A(:,2);
stage = A(:,3);
eqpid = A(:,4);
eqptype = A(:,5);
recipe = A(:,6);
% queue = C(:,1);
% trackin = C(:,2);
% trackout = C(:,3);
wt = B(:,2);
pt = B(:,3);
ct = B(:,4);
% trackinQ = B(:,5);
% currentQ = B(:,6);

%LotID_CT initialize
b = size(unique(lotid),1);
Lot_CT = zeros(b,1);
%so as to change when sequencial lotids differ
logicalIndex = ismember(lotid(1),lotid(2));

c = size(A,1);
i = 1;
j = 1;

%Create the LotID_CT Table
Lot_CT(1)=ct(1);
for i=2:c
    logicalIndex=ismember(lotid(i),lotid(i-1));
    if logicalIndex==true
        Lot_CT(j)=Lot_CT(j)+ct(i);
    else
        j=j+1;
        Lot_CT(j)=ct(i);
    end
end

%Big Table Paper & List LotID-CT (1.2)
d   =   1   +   1   +   1   +   size(unique(recipe),1)   +   2   *
size(unique(eqptype),1) + 1;%5*1 == LOTID + PART + CT + PRIO
K = cell(b,d);
K(:,1) = unique(lotid);
```

```
K(:,2)= num2cell(Lot_CT);

%One PT per Recipe
i = 1;
j = 1;
m = 1;
Recipe_PT = zeros(b,9);
logicalIndex1 = 0;
logicalIndex2 = 0;
Recipe_PT(1,1) = pt(1);
for i=2:c
    logicalIndex1 = ismember(lotid(i),lotid(i-1));
    logicalIndex2 = ismember(recipe(i),recipe(i-1));
    if logicalIndex1 == true
        if logicalIndex2 == true
           Recipe_PT(j,m) = Recipe_PT(j,m) + pt(i);
        else
            m = m + 1;
            Recipe_PT(j,m) = pt(i);
        end
    else
        j=j+1;
        m = 1;
        Recipe_PT(j,1) = Recipe_PT(j,m) + pt(i);
    end
end

K(:,3:11) = num2cell(Recipe_PT);

%One Utilization per EQPTYPE
%unique(eqptype); SOS: INSERT UTIL IN ASC ORDER!!
k = size(unique(eqptype),1);
Eqp_Util = zeros(b,k);
for i=1:k
    Eqp_Util(:,i) = U(i,2);
end

K(:,12:20) = num2cell(Eqp_Util);

%TIME IN QUEUE
i = 1;
j = 1;
m = 1;
Eqp_Queue = zeros(b,9);
logicalIndex11 = 0;
logicalIndex22 = 0;
Eqp_Queue(1,1) = wt(1);
for i=2:c
    logicalIndex11 = ismember(lotid(i),lotid(i-1));
    logicalIndex22 = ismember(recipe(i),recipe(i-1));%RECIPE 1-
1 WITH EQPTYPE
    if logicalIndex11 == true
        if logicalIndex22 == true
           Eqp_Queue(j,m) = Eqp_Queue(j,m) + wt(i);
        else
            m = m + 1;
            Eqp_Queue(j,m) = wt(i);
        end
    else
        j=j+1;
        m = 1;
        Eqp_Queue(j,1) = Eqp_Queue(j,m) + wt(i);
    end
end

K(:,21:29) = num2cell(Eqp_Queue);

%PRIORITY
K(:,30) = num2cell(PR(:,2));

% xlswrite('K.xls',K);

% NEURAL NETWORK
% input(:,1) = cell2mat(K(:,1)); %lotid
input = cell2mat(K(:,3:30)); %input(:,2:29)
output = cell2mat(K(:,2));

% xlswrite('input.xls',input);
% xlswrite('output.xls',output);
```

2.Processing Time

```
% INSERT A(cell),B(numeric) from (new_Rev_Dataset)
% TIP: PART, LOTID,eqptype ASC ORDER!!

lotid = A(:,1);
% prio = B(:,1);
% part = A(:,2);
% stage = A(:,3);
% eqpid = A(:,4);
eqptype = A(:,2);
% recipe = A(:,6);
% queue = C(:,1);
% trackin = C(:,2);
% trackout = C(:,3);
% wt = B(:,2);
pt = B(:,1);
ct = B(:,2);
% trackinQ = B(:,5);
% currentQ = B(:,6);

%LotID_CT initialize
b = size(unique(lotid),1);
% Lot_CT = zeros(b,1);
% %so as to change when sequencial lotids differ
% logicalIndex = ismember(lotid(1),lotid(2));
%
c = size(A,1);
i = 1;
j = 1;
%
% %Create the LotID_CT Table
% Lot_CT(1)=ct(1);
% for i=2:c
%     logicalIndex=ismember(lotid(i),lotid(i-1));
%     if logicalIndex==true
%         Lot_CT(j)=Lot_CT(j)+ct(i);
%     else
%         j=j+1;
%         Lot_CT(j)=ct(i);
%     end
% end
%
% %Big Table Paper & List LotID-CT (1.2)
% d = 1 + 1 + 1 + 3 * size(unique(eqptype),1) + 1;%5*1 == LOTID
+ PART + CT + PRIO
% K = cell(b,d);
% K(:,1) = unique(lotid);
%
% % Now I put the part in K(:,2)
% i = 1;
% j = 1;
% K(1,2)= part(1);
% for i=2:c
%     logicalIndex=ismember(lotid(i),lotid(i-1));
%     if logicalIndex==true
%         K(j,2) = part(i);
%     else
%         j=j+1;
%         K(j,2) = part(i);
%     end
% end
%
% K(:,3)= num2cell(Lot_CT);

%One PT per EQPTYPE
% i = 1;
% j = 1;
m = 1;
EQPTYPE_PT = zeros(b,10);
logicalIndex1 = 0;
logicalIndex2 = 0;
EQPTYPE_list(1,1) = eqptype(1);
EQPTYPE_PT(1,1) = pt(1);
for i=2:c
    logicalIndex1 = ismember(lotid(i),lotid(i-1));
    logicalIndex2 = ismember(eqptype(i),eqptype(i-1));
    if logicalIndex1 == true
        if logicalIndex2 == true
           EQPTYPE_PT(j,m) = EQPTYPE_PT(j,m) + pt(i);
        else
            m = m + 1; %m will reach 10
            EQPTYPE_list(1,m) = eqptype(i);
            EQPTYPE_PT(j,m) = pt(i);
        end
    else
        j=j+1;
        m = 1;
        EQPTYPE_PT(j,1) = EQPTYPE_PT(j,m) + pt(i);
    end
end

EQPTYPE_ALL(1,:) = EQPTYPE_list;
EQPTYPE_ALL(2:(b+1),:) = num2cell(EQPTYPE_PT);

% K(:,4:13) = num2cell(EQPTYPE_PT);
%
%
% %PRIORITY
% % Now I put the priority in K(:,34)
% i = 1;
% j = 1;
% K(1,34)= num2cell(prio(1));
% for i=2:c
%     logicalIndex=ismember(lotid(i),lotid(i-1));
%     if logicalIndex==true
%         K(j,34) = num2cell(prio(i));
%     else
%         j=j+1;
%         K(j,34) = num2cell(prio(i));
%     end
% end
%
% % xlswrite('K.xls',K);
%
%
% % NEURAL NETWORK
% % input(:,1) = cell2mat(K(:,1)); %lotid
% input = cell2mat(K(:,3:30)); %input(:,2:29)
% output = cell2mat(K(:,2));

% xlswrite('input.xls',input);
% xlswrite('output.xls',output);
```

3.WIP

```
% INSERT T_1(CELL),T_2(Table) from (TIMESTAMPS)
% INSERT W_1(CELL),W_2(Table) from (WIP_WAFERTEST_SEMIVAL)

%timestamps
t_part = T_1(:,1);
t_lot = T_1(:,2);
t = T_2(:,1);%the timestamp
t_max_out = T_2(:,2);

% WIP
lot = W_1(:,1);
min_in = W_2(:,1);
max_out = W_2(:,2);

i = 1;
j = 1;
k = 1;
b = size(t,1);
c = size(min_in,1);
WIP = zeros(b,1);
```

```
list_lots = cell(1000,b);%i have to find how to make this a
dynamic table for 802

%WIP
for i = 1:b
    for j = 1:c
        if       (datenum(table2array((min_in(j,:))))       <=
datenum(table2array(t(i,:)))) ...
            &&     (datenum(table2array((max_out(j,:))))     >=
datenum(table2array(t(i,:))))
            WIP(i) = WIP(i) + 1;
            list_lots(k,i) = (lot(j));
            k = k + 1;
        elseif      (datenum(table2array((min_in(j,:))))      >
datenum(table2array(t(i,:))))
            break
        end
    end
    k = 1;
end
```

4.Queue Time

```
% INSERT T_1(CELL),T_2(Table) from (TIMESTAMPS)
% INSERT C(cell),D(Table),E(Numeric) from (WAFERTEST_SEMIVAL)
% TIP: PART, LOTID,EQPTYPE ASC ORDER!!

wip = readtable('3lots_wip.xlsx');
list_lots = table2cell(readtable('3lots_list.xlsx'));
T = readtable('timestamps.xlsx');

%timestamps
t_part = table2cell(T(1:3,1));
t_lot = table2cell(T(1:3,2));
t = T(1:3,1);%the timestamp
t_max_out = T(1:3,2);

% wafertest_semival
lotid = C(:,1);
eqpid = C(:,2);
eqptype = C(:,3);
queue = D(:,1);
trackin = D(:,2);
trackout = D(:,3);
pt = E(:);

% QUEUE TIME
i = 1;
j = 1;
k = 1;
m = 1;

equipment = unique(eqpi
```
[redacted]

```
b = size(wip,2);
c = size(lotid,1);
d = size(equipment,1);
f = size(type,2);

time = zeros(b,d);
all_time = zeros(b,f);

% for i = 1:b
%     for j = 1:d
%         for k = 1:c
%             if (ismember(lotid(k),(list_lots(:,i)))...
%                     && (datenum(table2array((queue(k,:)))) <=
datenum(table2array(t(i,:)))) ...
%                      && (datenum(table2array((trackin(k,:))))
>= datenum(table2array(t(i,:))))...
%                             && (equipment(j) == eqpid(k))
%                                time(i,j) = time(i,j) + pt(k);
%             end
%         end
%     end
% end
%
% for m = 1:f
%     if type(m) == 1
%         all_time(i,f) = time(i,j)/(size(time,2));%avg waiting
time for the eqptype
%     end
% end

for i = 1:1
    for j = 1:f
        for k = 1:c
            logicalIndex1                                      =
ismember(lotid(k),(list_lots(:,i)));
            logicalIndex2 = ismember(type(j),eqptype(k));
            if (logicalIndex1 == true)...
                    &&  (datenum(table2array((queue(k,:))))  <=
datenum(table2array(t(i,:)))) ...
                        &&
(datenum(table2array((trackin(k,:))))                       >=
datenum(table2array(t(i,:))))...
                            && (logicalIndex2 == true)
                                all_time(i,j) = all_time(i,j) +
pt(k);
            end
        end
    end
end
```

5.Database Setting

```
datasource = ' MS SQL Server';
conn = database(datasource,'','');
selectquery = 'select * from WAFERTEST_SEMIVAL' ;

data = select(conn,selectquery);
```

6.WIP_Fetch

```
T = readtable('timestamps.xlsx');

%timestamps
t_part = table2cell(T(:,1));
t_lot = table2cell(T(:,2));
t = T(:,3);%the timestamp
t_max_out = T(:,4);

datasource = ' MS SQL Server';
conn = database(datasource,'','');

b = size(t,1);
WIP = zeros(b,1);
mega_list = cell(1500,b);

for i = 1:b
    sqlquery1   =   ['SELECT   COUNT(DISTINCT   LOTID)   FROM
WIP_WAFERTEST_SEMIVAL ' ...
        'WHERE Q_IN<= ' '''' char(table2array(t(i,:))) ''''...
        'AND T_OUT>= ' '''' char(table2array(t(i,:))) ''''];

    sqlquery2 = ['SELECT LOTID FROM WIP_WAFERTEST_SEMIVAL ' ...
        'WHERE Q_IN<= ' '''' char(table2array(t(i,:))) ''''...
        'AND T_OUT>= ' '''' char(table2array(t(i,:))) ''''];

    WIP(i) = table2array(fetch(conn,sqlquery1));
    mega_list(1:WIP(i),i) = fetch(conn,sqlquery2);
end

list = cell2table(mega_list);
wip = array2table(WIP);
% filename = 'megalist.xlsx';
% writetable(wip,filename,'Sheet',1);
% writetable(list,filename,'Sheet',2);

%run in 51 seconds
```

7.Queue_Fetch

```
%load workspace for easier

wip = readtable('wip_all.xlsx');
list_lots = table2cell(readtable('megalist.xlsx'));
T = readtable('timestamps.xlsx');
D    =    readtable('wafertest_semival.xlsx');    %put    the
new_wafertest_semival

%timestamps
t_part = table2cell(T(:,1));
t_lot = table2cell(T(:,2));
t  = table(datetime(table2array(T(:,3)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));%the timestamp
t_max_out = T(:,4);

% wafertest_semival
```

```
lotid = table2cell(D(:,1));
eqpid = table2cell(D(:,2));
eqptype = table2cell(D(:,3));
queue = D(:,4);
trackin = D(:,5);
trackout = D(:,6);
pt = table2array(D(:,7));

% QUEUE TIME
i = 1;
j = 1;
k = 1;
m = 1;

[redacted]

b = size(wip,1);
c = size(lotid,1);
f = size(type,1);

all_time = zeros(b,f);
%%
datasource = ' MS SQL Server';
conn = database(datasource,'','');

% for i = 1:b
%     for j = 1:f
%         for k = 1:c
%           logicalIndex1 = ismember(lotid(k),(list_lots(:,i)));
%              logicalIndex2 = ismember(type(j),eqptype(k));
%              if (logicalIndex1 == true)...
%                      && (datenum(table2array((queue(k,:)))) <=
datenum(table2array(t(i,:)))) ...
%                       && (datenum(table2array((trackin(k,:))))
>= datenum(table2array(t(i,:))))...
%                              && (logicalIndex2 == true)
%                                  all_time(i,j) = all_time(i,j)
+ pt(k);
%              end
%         end
%     end
% end

TIME_QUEUE = cell(b,f);

aa = {'eqptype','list'};
% remove(conn,'matlab_eqptypes',"{}");
% remove(conn,'matlab_listlots',"{}");

sqlquery1 = 'DELETE FROM matlab_eqptypes';
exec(conn,sqlquery1);
insert(conn,'matlab_eqptypes',aa(1),type);

for i = 1:b
    sqlquery2 = 'DELETE FROM matlab_listlots';
    exec(conn,sqlquery2);
    insert(conn,'matlab_listlots',aa(2),list_lots(:,i));

    sqlquery3                    =                    ['SELECT
ISNULL(AVG(new_wafertest_semival.PT),0) ' ...
        'FROM matlab_eqptypes ' ...
        'LEFT JOIN new_wafertest_semival ' ...
        'ON           matlab_eqptypes.EQPTYPE              =
new_wafertest_semival.EQPTYPE ' ...
        'AND  new_wafertest_semival.LOTID  IN(SELECT  list  FROM
matlab_listlots) ' ...
        'AND    new_wafertest_semival.QUEUETIME    <=    '    ''''
char(table2array(t(i,:))) '''' ...
        'AND   new_wafertest_semival.TRACKOUTTIME   >=   '   ''''
char(table2array(t(i,:))) '''' ...
        'GROUP BY matlab_eqptypes.EQPTYPE ' ...
        'ORDER BY matlab_eqptypes.EQPTYPE ASC'];

%     sqlquery3 = ['SELECT WAFERTEST_SEMIVAL.PT' ...
%          'FROM WAFERTEST_SEMIVAL' ...
%         'INNER JOIN matlab_listlots ON WAFERTEST_SEMIVAL.LOTID
= matlab_listlots.list' ...
%       'INNER JOIN matlab_eqptypes ON WAFERTEST_SEMIVAL.EQPTYPE
= matlab_eqptypes.eqptype' ...
%               'WHERE  WAFERTEST_SEMIVAL.QUEUETIME  <=  '  ''''
char(table2array(t(i,:))) '''' ...
%               'AND  WAFERTEST_SEMIVAL.TRACKINTIME  >=  '  ''''
char(table2array(t(i,:))) '''' ...
%          'GROUP BY WAFERTEST_SEMIVAL.EQPTYPE' ...
%          'ORDER BY WAFERTEST_SEMIVAL.EQPTYPE ASC'
%          ];

%          sqlquery4  =  ['SELECT  COUNT(DISTINCT  EQPTYPE)  FROM
WAFERTEST_SEMIVAL '...
%           'WHERE LOTID IN(SELECT list FROM matlab_listlots) '
...
%          'AND EQPTYPE IN(SELECT eqptype FROM matlab_eqptypes)
' ...
%           'AND QUEUETIME <= ' '''' char(table2array(t(i,:)))
'''' ...
%          'AND TRACKOUTTIME >= ' '''' char(table2array(t(i,:)))
'''');

%     ab = table2array(fetch(conn,sqlquery4));
%     aa = table2array(fetch(conn,sqlquery3));
    TIME_QUEUE(i,1:f)= fetch(conn,sqlquery3);
end

final_time_queue                                             =
table2array(cell2table(TIME_QUEUE))./eqpid_number;

% q = cell2table(TIME_QUEUE);
% filename = 'queue.xlsx';
% writetable(q,filename,'Sheet',1);
%           'AND (EQPTYPE = "EACDCC" OR EQPTYPE = "EACFFZ" OR
EQPTYPE = "EAEECA" OR EQPTYPE = "EAFDBF" OR EQPTYPE = "ECA-
80/88" '...
%           'OR EQPTYPE = "ECAAFC" OR EQPTYPE = "EDBFDFCDZ" OR
EQPTYPE = "EDDCA" OR EQPTYPE = "EEAADFFCZ" OR EQPTYPE = "EZDEAF")
' ...
```

8.Utilization_Fetch

```
%load workspace for easier

wip = readtable('wip_all.xlsx');
list_lots = table2cell(readtable('megalist.xlsx'));
T = readtable('timestamps.xlsx');
D = readtable('wafertest_semival.xlsx');

%timestamps
t_part = table2cell(T(:,1));
t_lot = table2cell(T(:,2));
t  = table(datetime(table2array(T(:,3)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));%the timestamp
t_max_out = T(:,4);

%t-24
t_24  = table(datetime(table2array(t),'InputFormat','yyyy-MM-dd
HH:mm:ss.SSS')- days(1));

% wafertest_semival
lotid = table2cell(D(:,1));
eqpid = table2cell(D(:,2));
eqptype = table2cell(D(:,3));
queue = D(:,4);
trackin = D(:,5);
trackout = D(:,6);
pt = table2array(D(:,7));

% util
i = 1;
j = 1;
k = 1;
m = 1;

[redacted]

b = size(wip,1);
c = size(lotid,1);
f = size(type,1);

all_time = zeros(b,f);
%%

datasource = ' MS SQL Server';
conn = database(datasource,'','');

util = cell(b,f);
```

```
aa = {'eqptype','list'};

sqlquery1 = 'DELETE FROM matlab_eqptypes';
exec(conn,sqlquery1);
insert(conn,'matlab_eqptypes',aa(1),type);

for i = 1:b

    sqlquery = ['SELECT SUM((cast(DATEDIFF(MINUTE, [dbo].[Max2]
(new_wafertest_semival.TRACKINTIME,'                             ''''
char(table2array(t_24(i,:)))        ''''        '),[dbo].[Min2]
(new_wafertest_semival.TRACKOUTTIME,'                            ''''
char(table2array(t(i,:)))    ''''    '))as    float)/1440))    /
(NUM_ID_10.num_eqpid) ' ...
        'FROM matlab_eqptypes ' ...
        'LEFT        JOIN        new_wafertest_semival        ON
matlab_eqptypes.EQPTYPE = new_wafertest_semival.EQPTYPE ' ...
        'LEFT  JOIN  NUM_ID_10  ON  matlab_eqptypes.EQPTYPE  =
NUM_ID_10.EQPTYPE ' ...
        'WHERE ' ...
        '(new_wafertest_semival.TRACKINTIME    >=    '    ''''
char(table2array(t_24(i,:)))       ''''              '       AND
new_wafertest_semival.TRACKOUTTIME       <=        '       ''''
char(table2array(t(i,:))) '''' ') '...
        'OR   (new_wafertest_semival.TRACKINTIME   <=   '   ''''
char(table2array(t_24(i,:)))       ''''              '       AND
new_wafertest_semival.TRACKOUTTIME       <=        '       ''''
char(table2array(t(i,:)))        ''''              '       AND
new_wafertest_semival.TRACKOUTTIME       >=        '       ''''
char(table2array(t_24(i,:))) '''' ') '...
        'OR   (new_wafertest_semival.TRACKINTIME   >=   '   ''''
char(table2array(t_24(i,:)))       ''''              '       AND
new_wafertest_semival.TRACKOUTTIME       >=        '       ''''
char(table2array(t(i,:)))        ''''              '       AND
new_wafertest_semival.TRACKINTIME       <=        '        ''''
char(table2array(t(i,:))) '''' ') '...
        'OR   (new_wafertest_semival.TRACKINTIME   <=   '   ''''
char(table2array(t_24(i,:)))       ''''              '       AND
new_wafertest_semival.TRACKOUTTIME       >=        '       ''''
char(table2array(t(i,:))) '''' ') '...
        'GROUP BY NUM_ID_10.EQPTYPE, NUM_ID_10.num_eqpid ' ...
        'ORDER BY NUM_ID_10.EQPTYPE ASC'];

    util(i,1:f) = transpose(fetch(conn,sqlquery));

end

maxi = max(round(max(cell2mat(util))),1);
final_util = cell2mat(util) ./ maxi;

% u = cell2table(final_util);
% filename = 'final_utilization.xlsx';
% writetable(u,filename,'Sheet',1);
```

```
%1,021527778    25,35833333 0,004166667 17,56458333 0,999629968
3,208134921 3,502083333 2,077083333 1,688194444 1,106597222
```

9.Final_Countdown

```
wip = readtable('wip_all.xlsx');
wait = readtable('queue1.xlsx');
util = readtable('final_utilization1.xlsx');
T = readtable('timestamps.xlsx');
A = readtable('cycletime.xlsx');
B = readtable('process1.xlsx');

%timestamps
Last(:,1) = A(:,1); %LOTID
Last(:,2) = A(:,2); %PART
Last(:,3) = A(:,3); %PRIORITY
Last(:,4:13) = B; %processing times
Last(:,14:23) = wait; %waiting
Last(:,24:33) = util; %utilization
Last(:,34) = wip; %wip
Last(:,35) = A(:,4); %CT

filename = 'final_final_1.xlsx';
writetable(Last,filename,'Sheet',1);
%%
% NEURAL NETWORK
% input = Last(:,1:34);
% output = Last(:,35);
%
% filename1 = 'last_input.xlsx';
% writetable(input,filename1,'Sheet',1);
%
% filename2 = 'last_output.xlsx';
% writetable(output,filename2,'Sheet',1);
```

10.EQPS_Merge

```
%EQPS Merge

%load rev_dataset
%load eqps specific

data = readtable('REV_DATASET_eqp_sorted.xlsx');
machine = readtable('eqps_dataset.xlsx');

%                           q_time                              =
table(datetime(table2array(data(:,8)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));

%data
eqpid = table2cell(data(:,1));
q_time                                                          =
table(datetime(table2array(data(:,2)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));
in_time                                                         =
table(datetime(table2array(data(:,3)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));
```

```
out_time                                                              =
table(datetime(table2array(data(:,4)),'InputFormat','yyyy-MM-
dd HH:mm:ss.SSS'));
%eqps
id = table2cell(machine(:,1));
stamp                                                                 =
table(datetime(table2array(machine(:,2)),'InputFormat','yyyy-
MM-dd HH:mm:ss.SSS'));
last_state = table2cell(machine(:,3));
current_state = table2cell(machine(:,4));

%initialize
a = size(data,1);
b = size(unique(eqpid),1);
c = size(machine,1);
d = size(unique(id),1);

states = cell(a,3);
q_state = states(:,1);
in_state = states(:,2);
out_state = states(:,3);

i = 0;
j = 0;

%testing the datetime comparison
aaa=0;
if table2array(stamp(1,1))<table2array(q_time(1,1))
    aaa = 1;
end

for i=1:a
    for  j=1:c
        logicalIndex = ismember(eqpid(i),id(j));
        if logicalIndex == true
            if table2array(stamp(j,1))<table2array(q_time(i,1))
                q_state(i) = current_state(j);
                in_state(i) = current_state(j);
                out_state(i) = current_state(j);
            elseif
table2array(stamp(j,1))>table2array(q_time(i,1))           &&
table2array(stamp(j,1))<table2array(in_time(i,1))
                in_state(i) = current_state(j);
                out_state(i) = current_state(j);
            elseif
table2array(stamp(j,1))>table2array(in_time(i,1))           &&
table2array(stamp(j,1))<table2array(out_time(i,1))
                out_state(i) = current_state(j);
            elseif
table2array(stamp(j,1))>table2array(out_time(i,1))
                if states(i,:)==zeros(1,1:3)
                    q_state(i) = last_state(j);
                    in_state(i) = last_state(j);
                    out_state(i) = last_state(j);
                end
            else
```

```
                break
            end
        else
            continue
        end
    end
end

states(:,1) = q_state;
states(:,2) = in_state;
states(:,3) = out_state;

ff = cell2table(states);
filename = '3states.xlsx';
writetable(ff,filename,'Sheet',1);l
```

11.Test_NN

```
data = readtable('final_final.xlsx');

num_sim = 30;%number of samples for simulation
i = 1;
j = 1;
k = 1;
c = size(data,1);
logicalIndex1 = 0;
family(1,1:33,1) = table2array(data(1,4:36));
for i=2:c
    logicalIndex1 = ismember(data(i,3),data(i-1,3));
    if logicalIndex1 == true && k<=num_sim
            family_valid(k,1:33,j)    =    table2array(data(i-
1,4:36));
            k = k + 1;
    elseif logicalIndex1 == true && k>num_sim
            family_train(k-30,1:33,j)   =   table2array(data(i-
1,4:36));
            k = k + 1;
    else
        k = 1;
        j=j+1;
        continue;
    end
end

family_1_valid = family_valid(:,:,1);%P1N6C
family_1_train                                                =
family_train(any(family_train(:,:,1),2),:,1);%P1N6C

family_2_valid = family_valid(:,:,2);%P2UGU
family_2_train                                                =
family_train(any(family_train(:,:,2),2),:,2);%P2UGU

family_3_valid = family_valid(:,:,3);%P4VV4
family_3_train                                                =
family_train(any(family_train(:,:,3),2),:,3);%P4VV4
```

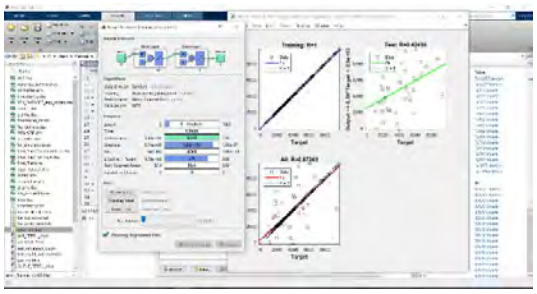
```
family_4_valid = family_valid(:,:,4);%P6380
family_4_train                                                  =
family_train(any(family_train(:,:,4),2),:,4);%P6380

family_5_valid = family_valid(:,:,5);%PF2PB
family_5_train                                                  =
family_train(any(family_train(:,:,5),2),:,5);%PF2PB

family_6_valid = family_valid(:,:,6);%PFR8T
family_6_train                                                  =
family_train(any(family_train(:,:,6),2),:,6);%PFR8T

family_7_valid = family_valid(:,:,7);%PHVB5
family_7_train                                                  =
family_train(any(family_train(:,:,7),2),:,7);%PHVB5

family_8_valid = family_valid(:,:,8);%PLVR1
family_8_train                                                  =
family_train(any(family_train(:,:,8),2),:,8);%PLVR1

family_9_valid = family_valid(:,:,9);%PMVBL
family_9_train                                                  =
family_train(any(family_train(:,:,9),2),:,9);%PMVBL

family_10_valid = family_valid(:,:,10);%PNA30
family_10_train                                                 =
family_train(any(family_train(:,:,10),2),:,10);%PNA30

family_11_valid = family_valid(:,:,11);%PNMXH
family_11_train                                                 =
family_train(any(family_train(:,:,11),2),:,11);%PNMXH

family_12_valid = family_valid(:,:,12);%PV7PP
family_12_train                                                 =
family_train(any(family_train(:,:,12),2),:,12);%PV7PP


family_13_valid = family_valid(:,:,13);%PW68U
family_13_train                                                 =
family_train(any(family_train(:,:,13),2),:,13);%PW68U

family_14_valid = family_valid(:,:,14);%PX933
family_14_train                                                 =
family_train(any(family_train(:,:,14),2),:,14);%PX933

K_1 = transpose(family_1_train(:,1:32));
L_1 = transpose(family_1_train(:,33));
T_1 = transpose(family_1_valid(:,1:32));

K_2 = transpose(family_2_train(:,1:32));
L_2 = transpose(family_2_train(:,33));
T_2 = transpose(family_2_valid(:,1:32));

K_3 = transpose(family_3_train(:,1:32));
L_3 = transpose(family_3_train(:,33));
T_3 = transpose(family_3_valid(:,1:32));
```

```
K_4 = transpose(family_4_train(:,1:32));
L_4 = transpose(family_4_train(:,33));
T_4 = transpose(family_4_valid(:,1:32));

K_5 = transpose(family_5_train(:,1:32));
L_5 = transpose(family_5_train(:,33));
T_5 = transpose(family_5_valid(:,1:32));

K_6 = transpose(family_6_train(:,1:32));
L_6 = transpose(family_6_train(:,33));
T_6 = transpose(family_6_valid(:,1:32));

K_7 = transpose(family_7_train(:,1:32));
L_7 = transpose(family_7_train(:,33));
T_7 = transpose(family_7_valid(:,1:32));

K_8 = transpose(family_8_train(:,1:32));
L_8 = transpose(family_8_train(:,33));
T_8 = transpose(family_8_valid(:,1:32));

K_9 = transpose(family_9_train(:,1:32));
L_9 = transpose(family_9_train(:,33));
T_9 = transpose(family_9_valid(:,1:32));

K_10 = transpose(family_10_train(:,1:32));
L_10 = transpose(family_10_train(:,33));
T_10 = transpose(family_10_valid(:,1:32));

K_11 = transpose(family_11_train(:,1:32));
L_11 = transpose(family_11_train(:,33));
T_11 = transpose(family_11_valid(:,1:32));

K_12 = transpose(family_12_train(:,1:32));
L_12 = transpose(family_12_train(:,33));
T_12 = transpose(family_12_valid(:,1:32));

K_13 = transpose(family_13_train(:,1:32));
L_13 = transpose(family_13_train(:,33));
T_13 = transpose(family_13_valid(:,1:32));

K_14 = transpose(family_14_train(:,1:32));
L_14 = transpose(family_14_train(:,33));
T_14 = transpose(family_14_valid(:,1:32));
K                                                              =
horzcat(K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_8,K_9,K_10,K_11,K_12,K_1
3,K_14);
L                                                              =
horzcat(L_1,L_2,L_3,L_4,L_5,L_6,L_7,L_8,L_9,L_10,L_11,L_12,L_1
3,L_14);
T                                                              =
horzcat(T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9,T_10,T_11,T_12,T_1
3,T_14);
nntool
```

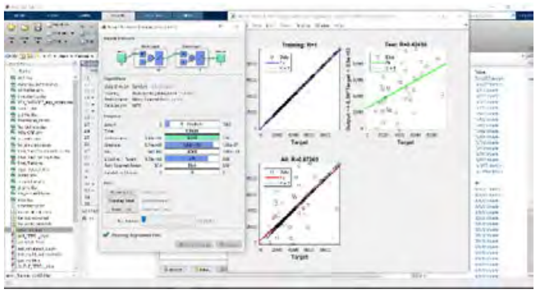
APPENDIX 3: Neural Network Training for All Product Families

For the rest 13 families, as well as the total data, the results are the following:

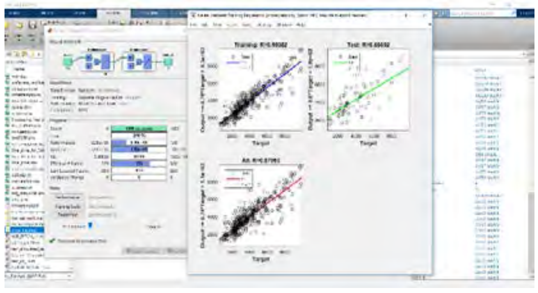
1. [redacted]

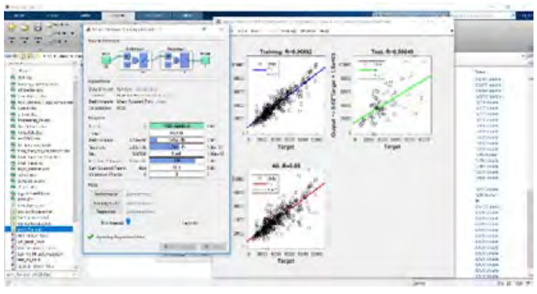


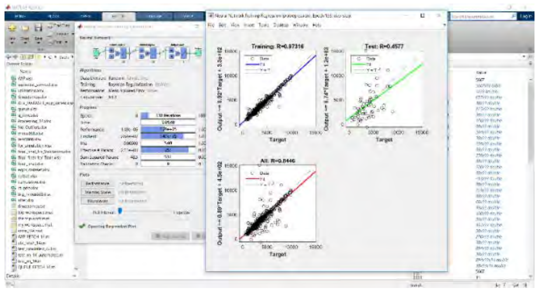
2. [redacted]



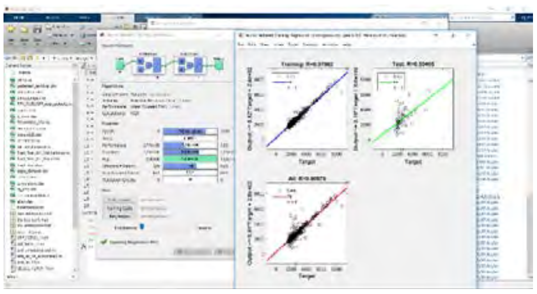
3. [redacted]

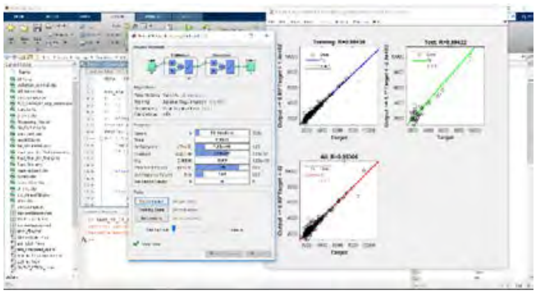


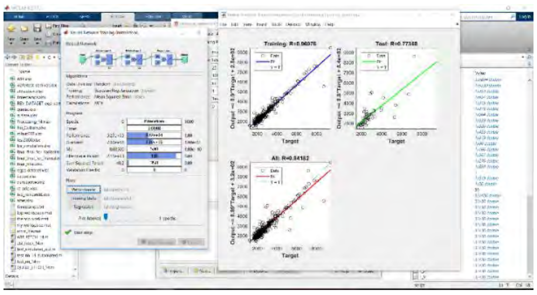




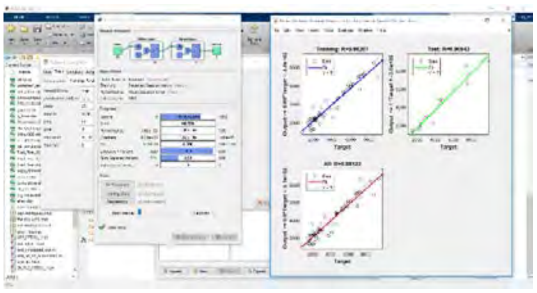
4. [redacted]

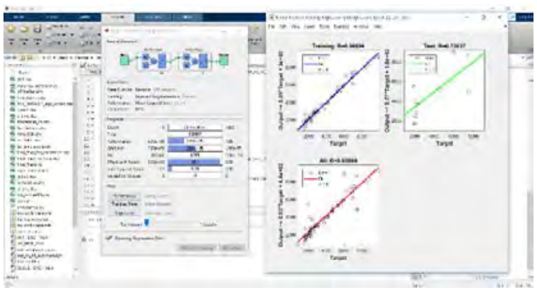


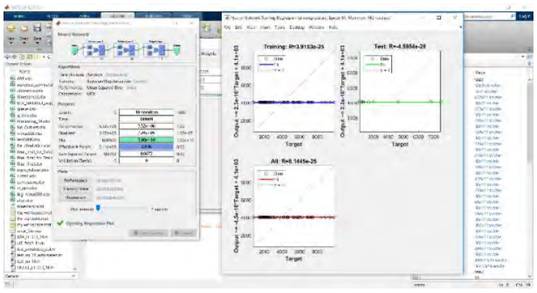




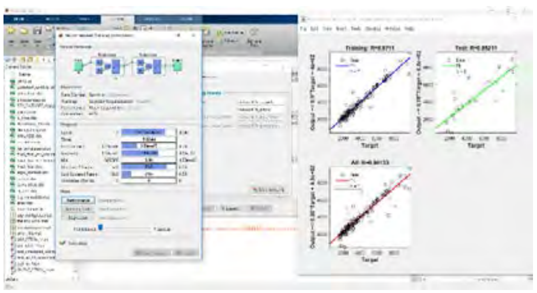
5. [redacted]

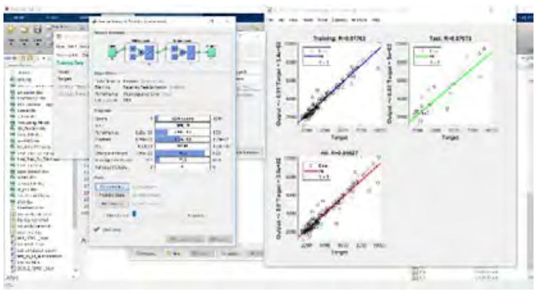


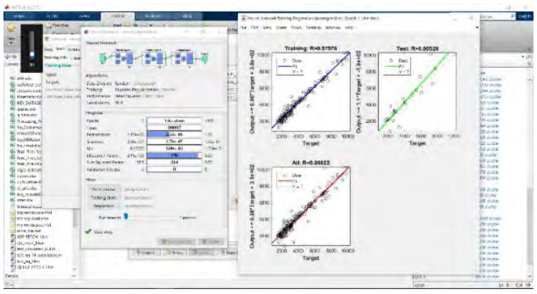




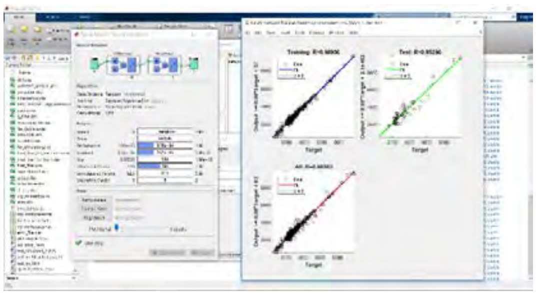
6. [redacted]

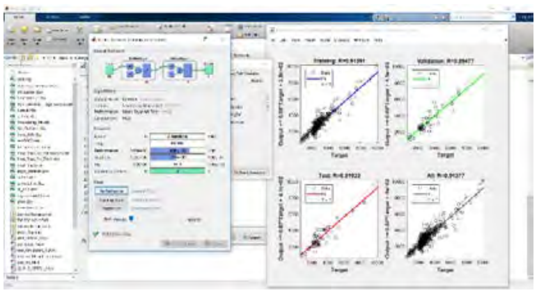


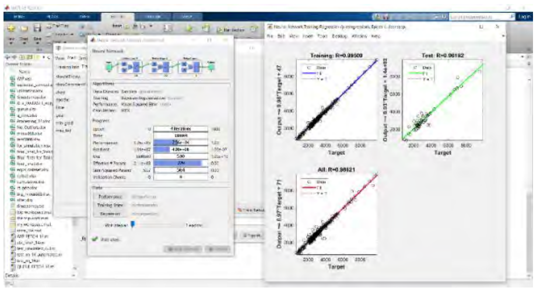




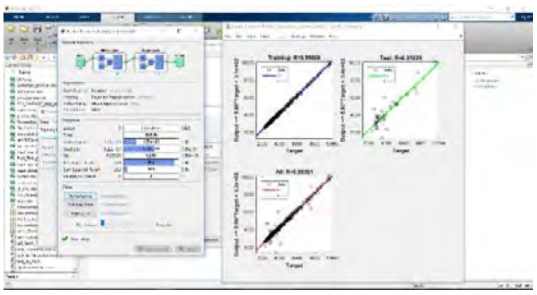
7. [redacted]

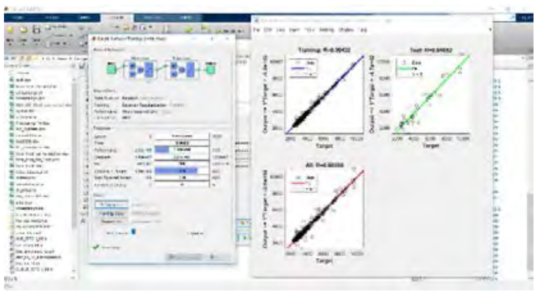


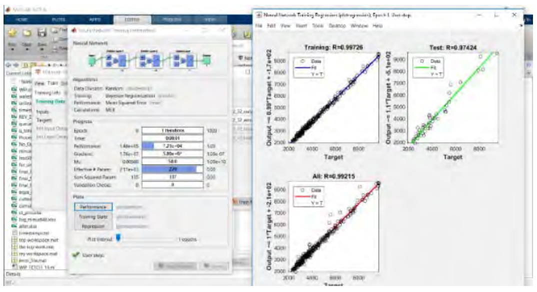




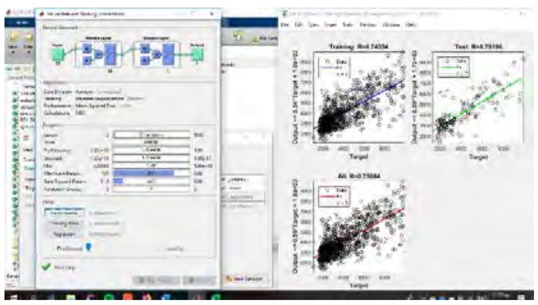
8. [redacted]

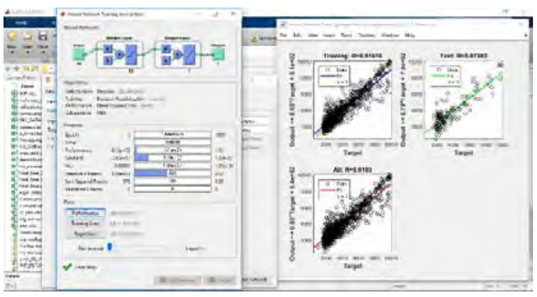


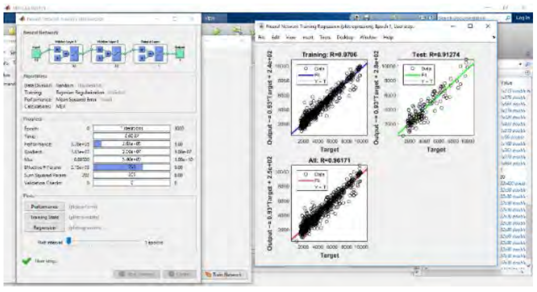



Neural Network Training Regression (plotregression), Epoch 1, User stop.
Training: R=0.99726
Test: R=0.97424
All: R=0.99215
Data
Fit
Y = T
Target

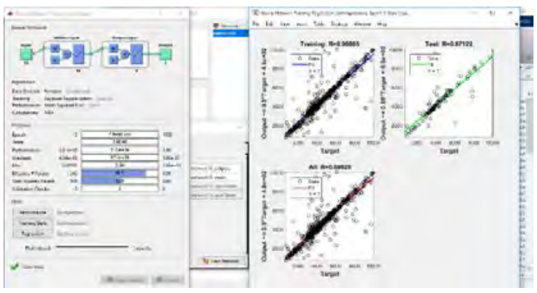
9. [redacted]

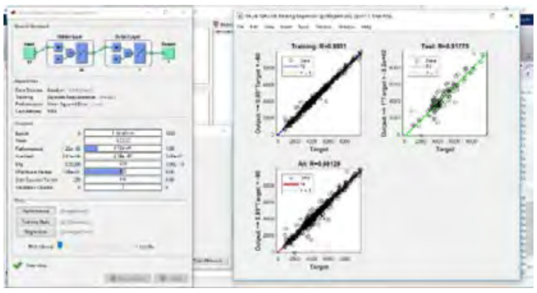


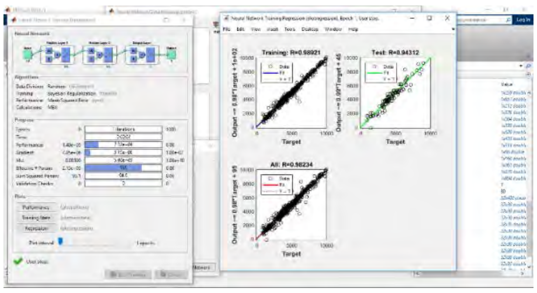



Neural Network Training Regression (plotregression), Epoch 1, User stop.
Training: R=0.9706
Test: R=0.91274
All: R=0.96171
Data
Fit
Y = T
Target

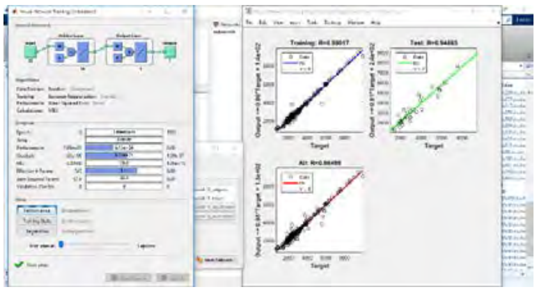
10. [redacted]

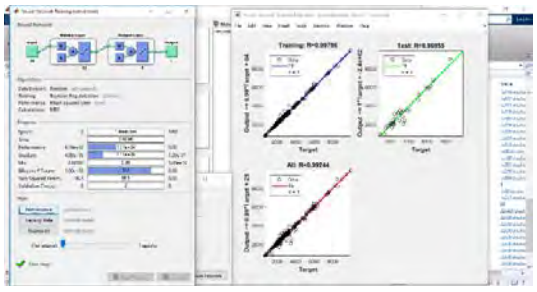


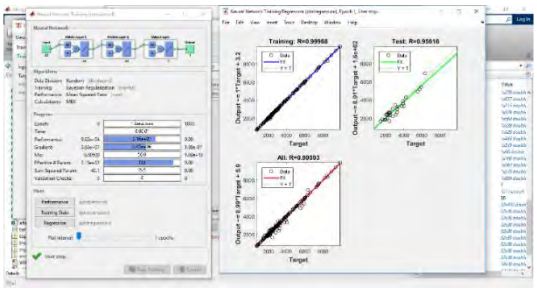




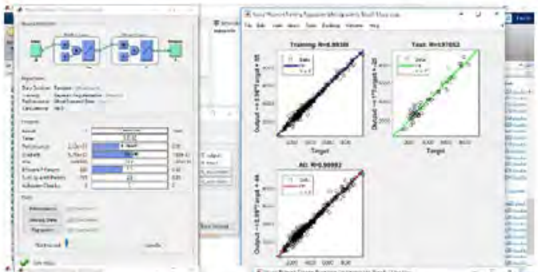
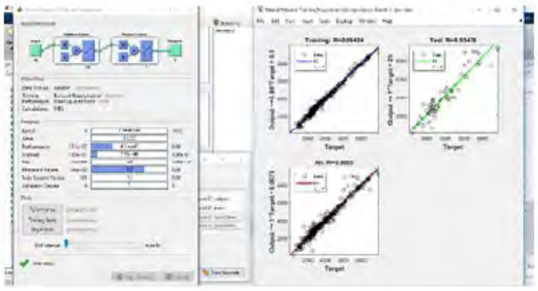
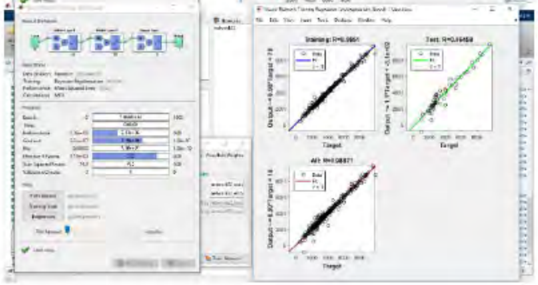
11. [redacted]



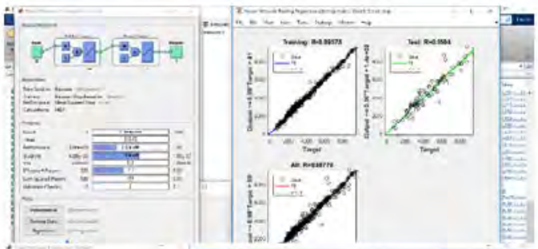
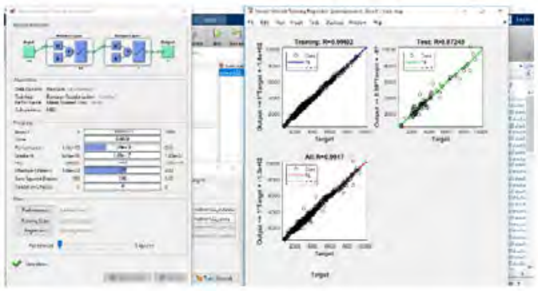
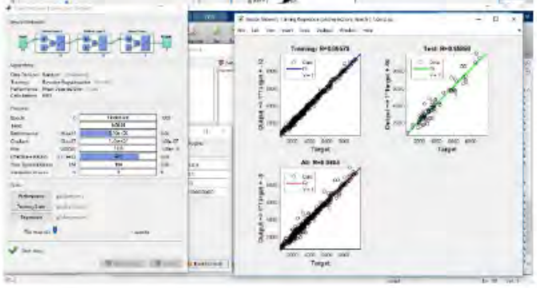




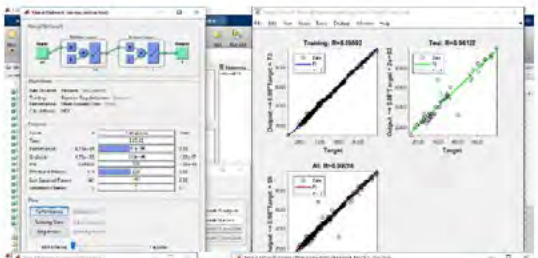
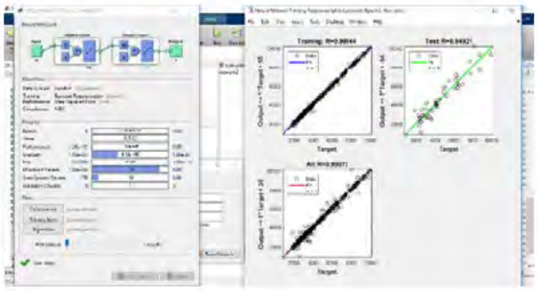
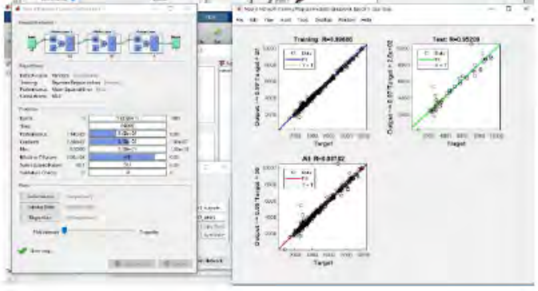
12. [redacted]

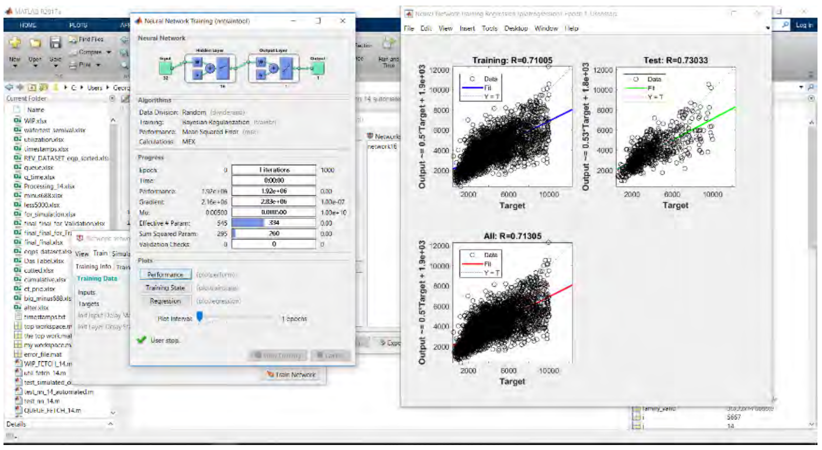
13. [redacted]

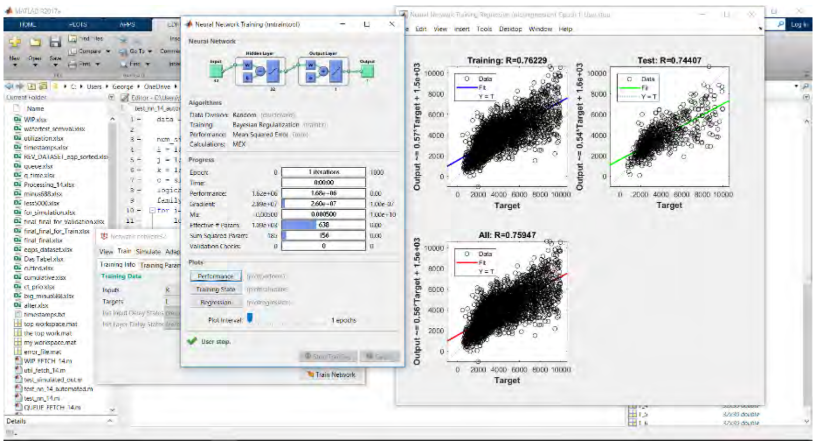
14. [redacted]

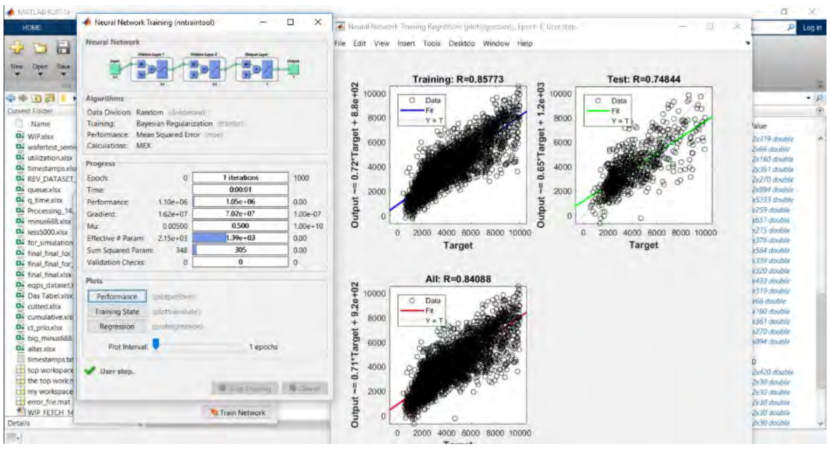
ALL Data


Training: R=0.71005
Test: R=0.73033
All: R=0.71305
Target
Output ~= 0.5*Target + 1.9e+03
Output ~= 0.53*Target + 1.8e+03
Output ~= 0.5*Target + 1.9e+03
Data
Fit
Y = T

```
Neural Network Training (nntraintool)

Algorithms
Data Division:  Random  (dividerand)
Training:       Bayesian Regularization  (trainbr)
Performance:    Mean Squared Error  (mse)
Calculations:   MEX

Progress
Epoch:                    0   1 iterations   1000
Time:                         0:00:00
Performance:       1.92e+06   1.92e+06       0.00
Gradient:          2.16e+06   2.83e+06       1.00e-07
Mu:                 0.00500   0.000500       1.00e+10
Effective # Param:      545   334            0.00
Sum Squared Param:      295   260            0.00
Validation Checks:        0   0              0

User stop.
```


Training: R=0.76229
Test: R=0.74407
All: R=0.75947
Target
Output ~= 0.57*Target + 1.5e+03
Output ~= 0.54*Target + 1.6e+03
Output ~= 0.56*Target + 1.5e+03
Data
Fit
Y = T

```
Neural Network Training (nntraintool)

Algorithms
Data Division:  Random  (dividerand)
Training:       Bayesian Regularization  (trainbr)
Performance:    Mean Squared Error  (mse)
Calculations:   MEX

Progress
Epoch:                    0   1 iterations   1000
Time:                         0:00:00
Performance:       1.62e+06   1.68e+06       0.00
Gradient:          2.89e+07   2.60e+07       1.00e-07
Mu:                 0.00500   0.000500       1.00e+10
Effective # Param: 1.08e+03   638            0.00
Sum Squared Param:      185   156            0.00
Validation Checks:        0   0              0

User stop.
```


Training: R=0.85773
Test: R=0.74844
All: R=0.84088
Target
Output ~= 0.72*Target + 8.8e+02
Output ~= 0.65*Target + 1.2e+03
Output ~= 0.71*Target + 9.2e+02
Data
Fit
Y = T

```
Neural Network Training (nntraintool)

Algorithms
Data Division:  Random  (dividerand)
Training:       Bayesian Regularization  (trainbr)
Performance:    Mean Squared Error  (mse)
Calculations:   MEX

Progress
Epoch:                    0   1 iterations   1000
Time:                         0:00:01
Performance:       1.10e+06   1.05e+06       0.00
Gradient:          1.62e+07   7.02e+07       1.00e-07
Mu:                 0.00500   0.500          1.00e+10
Effective # Param: 2.15e+03   1.39e+03       0.00
Sum Squared Param:      348   305            0.00
Validation Checks:        0   0              0

User stop.
```

Simulation Results

#	LOTID	PART	CT	Simulation	AVG_Error
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,640206944
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	1,225865618
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	4,266538124
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	6,861420678
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	7,178500198
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	9,931581162
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	18,76918295
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	30,76229461
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	33,18497089
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	36,330545
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,949221467
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	7,068351023
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	9,772566254
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	12,61932083
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	14,66592153
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	18,4262844
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	20,39314275
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	21,28825788
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27,21590548
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	29,13364857
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,575974214
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	1,682550023
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	2,534036518
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	6,947349525
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	10,14762215
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	10,76508626
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	11,18380309
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	11,83639111
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	13,64258692
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	15,0090916
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,00916986
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,069837295
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	1,395549134
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	3,449238524
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	3,927341001
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	4,105678902
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	4,542021042
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	6,252471166
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	6,960906601
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	11,74516618
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,495228207
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,49832968
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	2,415019438
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	5,472973574
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	9,179301516
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	9,270018309
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	12,7384369
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	13,0308283
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	15,69216376
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	19,12618547
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,744913652
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	0,970122235
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	12,24479795
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	18,61909199
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	19,09616178
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	21,77309722
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	23,11758199
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	23,73498294
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	26,27192241

[redacted]

36,9864557
1,726910537
2,153257033
8,654362512
10,79636296
15,78878818
16,06980662
24,23701338
28,63327694
35,10564547
35,85187119
0,793155923
6,232568396
7,820118891
10,55590652
13,86140449
14,22378827
15,86598887
15,97382234
18,69984116
19,32157118
0,639074026
2,150276783
3,517738532
5,276267224
11,05422007
11,31384934
14,57556893
15,10427391
20,57406
20,964617
1,013188286
10,8109631
12,76952663
13,40374293
15,04952844
15,68326905
16,78525133
21,53477977
25,24559912
25,53240817
6,632212477
7,309449239
11,80194705
16,20053935
16,34064026
19,55892376
19,85529236
20,95896727
21,52504934
31,15559041
3,93314318
4,546800227
8,520664176
14,35555879
23,37341019
29,01152091
35,92081286
40,66600294
47,35816161
50,02805944
1,082575306
3,081419576

[redacted]	5,24618759
[redacted]	6,989157255
[redacted]	13,02902257
[redacted]	15,76791913
[redacted]	25,92303435
[redacted]	28,19271736
[redacted]	31,17750848
[redacted]	43,07821157
[redacted]	6,132743896
[redacted]	6,504570189
[redacted]	8,358070261
[redacted]	10,14597853
[redacted]	12,24630217
[redacted]	16,87884763
[redacted]	21,4788173
[redacted]	21,79061117
[redacted]	22,03531746
[redacted]	22,14988841

Problematic Case Comparison

#	1	2	3	4	5	6	7	8
LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
AVG_Error	0,640207	36,33054	0,949221	29,13365	0,575974	15,00909	0,00917	11,74517

#	9	10	11	12	13	14	15	16
LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Q_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
AVG_Error	0,495228	19,12619	0,744914	36,98646	1,726911	35,85187	0,793156	19,32157

#	17	18	19	20	21	22	OVER
LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	UNDER
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
PR TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	
AVG_Error	0,639074	20,96462	1,013188	25,53241	6,632212	31,15559	
#	23	24	25	26	27	28	

LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
AVG_Error	3,933143	50,02806	1,082575	43,07821	6,132744	22,14989
						OVER
						UNDER
LOTID	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PART	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PRIORITY	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
PR_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Q_TIME#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Q_TIME#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
UTIL#10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
WIP	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
CT	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Simulation	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
AVG_Error	0,575974	15,00909	3,933143	50,02806	6,132744	22,14989

Typical MATLAB Neural Network Code

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 04-Mar-2020 12:14:55.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 32xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 1xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset                                                                 =
[2;0;0;0;180;417;1;22;0;0;38;0;31;0;0;9.86861313868613;8.42857142857143;12.6;0;0;9.833
33333333333;0;0.043482905982906;0;0.366820987654321;0.556143552311435;0.09427910052910
06;0.0949652777777778;0.021875;0.000231481481481482;0.262307098765432;333];
x1_step1.gain                                                                    =
[2;2;0.000369822485207101;2;0.00246609124537608;0.000265780730897010;0.000642880102860
816;0.00809716599190283;0.25;0.0350877192982456;0.000735294117647059;0.000277777777777
78;0.000465549348230912;0.4;0.000277777777777778;0.205243445692884;0.00295795478554828
;0.00271223216707350;0.000832986255726781;0.000833333333333333;0.0120240480961924;1.990
32480995163;2.13716927819163;480;3.37324310255075;4.71814985769975;2.18726266681133;2.
56890553920257;1.96923076923075;2.40300375469337;2.36528721996624;0.00326264274061990];
x1_step1.ymin = -1;

% Layer 1
b1       =       [-1.3592143147378476e-15;-0.16571209615763197;1.0187228313644613e-15;-
1.0225882533627232e-15;-1.3673920619115801e-15;-5.4446833705065246e-
16;0.18480325559385447;-1.3020142574846626e-
15;0.097054005745429425;0.17019717981536261;1.7717649369148558e-
15;0.066440219929966848;-0.095320536443100790;-7.8855166561016021e-16;-
0.065959835789589402;5.2403655872971608e-16;-
0.021583598268865682;0.082270879249972562;1.1664944158426322e-
15;0.050999296188244087;0.048173206428687818;0.24131908242620148;1.2382723458317663e-
15;7.2525336030752341e-16;-1.5101510623137780e-15;-5.7175290686636004e-16;-
0.055248061035780831;-1.5770533950214030e-15;0.065245426728686454;-8.5779627518763435e-
16;0.17746887144953630;1.5375039326523159e-15];
IW1_1   =   [4.6667777870765683e-15   -2.0053704724794636e-15   -2.0052213459637510e-15
1.3592143147271611e-15      7.5127121449285888e-17      -1.2437105486176589e-15      -
2.1435097950363594e-15         -1.4761729070445856e-15         -2.4364633161093104e-15
6.3831452692921919e-16          -9.9106208171189087e-16          9.2087195243441406e-16
3.7815183413164324e-15      1.1444651817329498e-15      1.7219361127945814e-15      -
1.9056215302138240e-15     -5.6076595178160183e-15     -3.7855205026437702e-15      -
3.4223553209222807e-16      -3.2789659105247535e-15      3.2954996649278457e-15      -
```

```
1.7904217741698344e-15          2.0709452095197915e-15          -2.6475681227332205e-15
2.0870873953555249e-15      3.6686832609427319e-15      -1.7808105924277697e-15      -
2.9612230218195576e-15      -1.0486615346669919e-15      2.8382245751743495e-15
3.0123501338042925e-15 -1.0811413339012171e-15;0.25093875906290214 0.099711720725119038
-0.22644446351174433   0.16571209615718774   0.38347657879932084   0.17060802501167102
0.10262786701296733   -0.26148106745062888   -0.10825018029100604   0.48593023702972155
0.30774334161613537   0.1524665188654532   -0.037394278489230832   -0.11677291354173457
0.30809195919391991   0.3554851385076222   -0.38668851948963595   0.00227362861944048
0.1682859616019767  -0.33457910691947246  0.56980395418596708  0.20558517242353314  -
0.61971668224960619 -0.12543289266299495 7.7449726864261973e-05 0.23209388325725838 -
0.6038806064788329   -0.32058583798901297   0.038928376564704749   -0.31304147537782834
0.20246818322875118 0.061160992468680737;-3.4980430366811649e-15 1.5031682982367328e-15
1.5030377540494489e-15          -1.0187228313671847e-15          -5.6284698204516604e-17
9.3226582086467953e-16          1.6067092331859865e-15          1.1064107663221644e-15
1.8262574429555899e-15      -4.783770719947575e-16      7.428930912701033e-16      -
6.9018510640797272e-16      -2.8344736887340808e-15      -8.5774055883661319e-16      -
1.2906256654177644e-15          1.4283940127483226e-15          4.2032074260419097e-15
2.8374317603077462e-15      2.5658828546708351e-16      2.457761900540223e-15      -
2.4700977081128428e-15          1.3420529620292524e-15          -1.5521874920201842e-15
1.9845163387104274e-15          -1.5643459487589303e-15          -2.7499309007921007e-15
1.3348121475914291e-15      2.2195938979285747e-15      7.8608060574553344e-16      -
2.1272822007199126e-15          2.2579400685929504e-15          8.1050741583140427e-
16;3.5102966158959457e-15        -1.5084114978568603e-15        -1.5083604921123946e-15
1.0225882533598284e-15      5.6534223443297901e-17      -9.35473713656782e-16      -
1.6123287308181453e-15          -1.1105886205763939e-15          -1.8327605663405919e-15
4.8029866416837236e-16          -7.4543002203640041e-16          6.9279469436690156e-16
2.8443764302392806e-15      8.6110317956264077e-16      1.2953635056432303e-15      -
1.4333305199057099e-15      -4.2182709398000494e-15      -2.8475984048893574e-15      -
2.5730233925841536e-16      -2.4665240878490459e-15      2.4790126629708789e-15      -
1.3467393456172781e-15          1.5580323713763422e-15          -1.9915227656716219e-15
1.5700390164703116e-15      2.7594553971693471e-15      -1.3395848613408776e-15      -
2.2274869709740629e-15      -7.8871136431439152e-16      2.135235994970758e-15      -
2.2658819256038167e-15          -8.1281370008463621e-16;4.6952340200384061e-15          -
2.0176087751368164e-15 -2.0174302755449298e-15 1.3673920619201513e-15 7.5563555642936e-
17    -1.2513171084667894e-15    -2.1565856164410396e-15    -1.4850649600580436e-15    -
2.4512829758856455e-15          6.4211464464513738e-16          -9.9713158019822261e-16
9.2641543693646488e-16          3.8045843972169595e-15          1.1513115260043689e-15
1.7323549687416088e-15      -1.9172654777532008e-15      -5.6417309001349908e-15      -
3.8085223809619545e-15          -3.4439198201843986e-16          -3.2989055193278706e-15
3.3154967139528102e-15      -1.8013458165574007e-15      2.0834240579068494e-15      -
2.6636942632102932e-15      2.0997363702993052e-15      3.6910925690549889e-15      -
1.7916371292292235e-15          -2.9792380347557776e-15          -1.0551029293730891e-15
2.8553467139163129e-15          -3.030708409154236e-15          -1.0879264583719338e-
15;1.8696086814017943e-15        -8.0340885273595262e-16        -8.0333954740361494e-16
5.4446833704664455e-16      3.0074471604784791e-17      -4.9827750121667234e-16      -
8.5874877404296098e-16          -5.913468578596128e-16          -9.7608514819414803e-16
2.5566904899786261e-16          -3.9706346745110867e-16          3.6887459661448749e-16
1.5149388819024254e-15      4.584280880075027e-16      6.8979316491338942e-16      -
7.6343430752559417e-16      -2.2464987310249614e-15      -1.5165310344720734e-15      -
1.3714797086095149e-16      -1.313611683328314e-15      1.3201880309125131e-15      -
7.1729973620872827e-16          8.2959362338785908e-16          -1.0606761124136714e-15
8.3609780569768014e-16      1.4697605972961713e-15      -7.1342406395801682e-16      -
1.1863125334241496e-15      -4.2014541404172276e-16      1.1369624308604739e-15      -
1.2068085435394665e-15 -4.3318956682733483e-16;-0.39236989060373073 -0.2026778149805458
-0.19410484420972415  -0.18480325559436273  -0.2466486834652464  0.34577318193176826  -
0.056356825422113677  -0.52748695722881267  0.3480393570227428  0.095807252299884627  -
0.27737814207562306  -0.19654463351073453  0.85171960269295921  0.28746443456893839  -
0.25527216935872987   0.028493867116261999   1.1584230761841057   0.70283893437284239
0.1640626552367889  -0.30594183055102098  -0.045943010668224891  -0.35257670057717383  -
0.11144270756762537  -0.12529283242875205  -0.062780883544893459  0.27413279797876355  -
0.0077590854301919279   -0.33524641632043845   0.035557001758275295   0.33715962714897618
0.71000347944887254 -0.75340637790107001;4.4717355039857943e-15 -1.9215948989817364e-15
-1.9213501803647451e-15      1.3020142574920389e-15      7.1910365836701057e-17      -
1.1918131114529301e-15      -2.0539436344077679e-15      -1.4140903971125581e-15      -
2.3345003466894548e-15          6.1130925398779976e-16          -9.4973490362461042e-16
8.8212807132994992e-16          3.6234989737317404e-15          1.0961622941088578e-15
1.6496801809226458e-15      -1.8260622155360798e-15      -5.3728582923409272e-15      -
3.6270201675253791e-15          -3.281796152255706e-16          -3.1417340929615904e-15
3.1574190970890947e-15      -1.7156206762023732e-15      1.9838609509518452e-15      -
2.5368532885366642e-15      1.9995844539293825e-15      3.5154926010196546e-15      -
1.7062657326088226e-15          -2.8373126417428204e-15          -1.0050003455912186e-15
2.7189472656350699e-15          -2.8864144277839125e-15          -1.0366379043184611e-
15;0.7919067644017882  0.11398958514658217  0.0030409627196034518  -0.097054005745246585
0.090530177523261632  0.2156664201014957  -0.0028186934637353732  0.16308928837649939  -
0.31225690595654132   0.050979209219458047   0.11349571927290875   0.085478338761951295
0.70702143622818991  -0.34544119658301436  0.22443974270442735  0.14452508019391508  -
0.32747532411474495  -0.15869093113012339  0.028937195013126991  -0.24961266173010399  -
```

```
0.067810718160366629  -0.56537596428129722  0.1756800172894073  -0.0010536050388541909
0.53430059601517044   -1.473713891890321   -0.28617162495291759   0.12807077036746214
0.13195642151905942  -0.23043258647033002  0.059556782038817049  -0.13890838006390482;-
0.51997100311926847   0.046791585486201617   0.20933464228472157   -0.17019717981546392
0.50500726262617224  0.018207644517591992  -0.04772332198316253  -0.70927034426457258  -
0.54456247686713932   0.095965341029023779   0.14402543228664422   -0.10180447050804636
0.18020765579473452  -0.084274701407462085  0.02596349045558019  0.46051717121920072  -
0.0068173978459033511  -0.56196486792523992  0.018647918615431958  -0.098970824188675699
0.44588819312654859   0.38674769983336121   0.45024868045834643   -0.10280244648493225
0.51789409499253347   1.5305482427212622   0.3339396291433942   0.69022470866793451   -
0.11853621357383359  -0.51978323177629315  -0.020029475375626889  0.030217099596306682;-
6.0831990327362367e-15       2.6140352251315463e-15       2.6138523078242254e-15       -
1.7717649369149205e-15           -9.7917622278017665e-17           1.6211988347046613e-15
2.7941034291664536e-15       1.9242480135537774e-15       3.175974300516441e-15       -
8.320584607761729e-16       1.2918696384526506e-15       -1.2003695204805546e-15       -
4.9292287032524207e-15           -1.4918455811323814e-15           -2.2445675603144034e-15
2.4839857427379799e-15           7.3096911736903288e-15           4.9345025973308997e-15
4.4610379972120906e-16           4.2741994836414561e-15           -4.2957298833181826e-15
2.3338515102997793e-15       -2.6995343028050516e-15       3.451156358513841e-15       -
2.7205646291191699e-15           -4.7821531285754853e-15           2.321329007237015e-15
3.8600048726208684e-15           1.3669427652286722e-15           -3.6996968224707521e-15
3.9266416606838138e-15 1.4092059043488865e-15;0.52740157094023499 -0.046032063787954047
0.16892442336638314   -0.066440219929717673   0.43429641284795578   -0.24835204624703386
0.036810382793745447  -0.63635707643329986  0.26098116941823102  0.17708738039835345  -
0.10509755571866319   0.016124511211014887   -0.20464451493230498   0.50951818868172916
0.11451789324436878  -0.12690315366748844  -0.28765265089864267  -0.081295733937898254
0.2179227986853795   0.96762092597973159   -0.80726724484517098   0.75538698482832078
0.3028458542525832   -0.43749332794136225   0.15791949333209332   -0.69283257807434906
0.057819672230940151  -0.37885913599318577  0.1477249812458151  -0.65611770740588238  -
0.27831715318857758    -0.024261323146501184;0.12253461515801699    0.11873088496260066
0.17191274967615158   0.095320536442944151   0.32046089219167578   -0.33667844331733598
0.0037880975916943279   0.38619068580471105   -0.21233369104204872   0.32470953367320737
0.046231226969971922  -0.028305138125816399  -0.1353744373548007   -0.60724495096368902
0.23565344773209723   -0.044787420245362836   -0.42466038883150276   -0.1160307071155363
0.13918125083168154   0.49211455186606862   0.24286344499776102   -0.53122455909822008
0.27358812883655981   -0.49906905239060984   0.0073223360082992905   0.13829428919923312
0.67065095989235435  0.060203757292319762  -0.050518830053121355  -1.0812134853756317  -
0.42744434386250901 -0.25746500503548719;2.7083988537360849e-15 -1.1638613479020553e-15
-1.1637049532742315e-15       7.8855166560910441e-16       4.3541654240882244e-17       -
7.2185790675358518e-16       -1.244019737652113e-15       -8.5644082242233765e-16       -
1.4139276997930211e-15           3.7021787826888144e-16           -5.752394957535613e-16
5.342507639266591e-16           2.1946432539097531e-15           6.6386700958043617e-16
9.9913164207798843e-16      -1.1059975619899167e-15      -3.2541461890679412e-15       -
2.1967569222007724e-15           -1.9879527877789226e-16           -1.9028435262622853e-15
1.9123195913582879e-15      -1.0391092582273418e-15      1.2015172513758365e-15       -
1.5364967254578438e-15       1.2110656782872711e-15       2.1292408069231322e-15       -
1.0334282067165249e-15           -1.7184651996799625e-15           -6.0871784696261232e-16
1.6467243077215899e-15           -1.7482145836466182e-15           -6.2791569120665302e-
16;0.47993000787216955  0.099541795920587525  -0.23582751682927108  0.065959835789294763
0.15435230801023322   0.33402315468479638   -0.10700469142826778   -0.21408465612382113
0.19223789221257251  -0.79098594469083894  -0.30393797066836431  7.8559784628226721e-06
0.69288428802505542  0.72297647711800084  -0.0024722847595141919  0.43392333098540226  -
0.22480052757243363   -0.17143133317608977   0.038371166988442444   -0.53047200443968934
0.32257336845427786  0.10679907131769494  -0.31032203664033869  -0.30596206448458124  -
0.31125268602561107   0.36039816440726363   -0.64116790355581299   -0.36480866725602878
0.023219322624267579   0.4343199703914507   0.78866111346176315   -0.74574472586412777;-
1.798135836310853e-15       7.7266265337306109e-16       7.7269627627234941e-16       -
5.2403655872990215e-16           -2.8996901594673143e-17           4.7915138832458295e-16
8.2590395904709404e-16       5.691218022132204e-16       9.389029652564778e-16       -
2.4621341642844836e-16      3.8179591272253072e-16      -3.5502185768192982e-16       -
1.4569973957057789e-15           -4.413625066780646e-16           -6.6370411817962957e-16
7.3416630496009392e-16           2.1610469322869591e-15           1.458841535021087e-15
1.3166530138347791e-16           1.2635840155994901e-15           -1.2700581580873414e-15
6.8985467378217794e-16      -7.9839904467016472e-16      1.0201923080064277e-15       -
8.044058306771479e-16           -1.4134379774063501e-15           6.8626878653637379e-16
1.1411088331385095e-15           4.039232350318881e-16           -1.094139852859583e-15
1.1607122146239952e-15 4.1595849574009819e-16;0.43792708913253164 0.011915888798079789
0.27316121797514492   0.02158359826957763   -0.46268066497290483   -0.018664966905354803
0.13479629489235054  -0.94780649628936331  0.47668689810021819  -0.49692458686440233  -
0.022341466570222096   0.0038958693298829878   -0.63660886131477656   0.6419038790933691
0.10146509448987992   0.06689745938346868   -0.0308809962762066   -0.5289247700926647
0.12348785779603029  -0.79171971588535484  0.095333933420051753  0.047877895029340811  -
0.12752573718539115   0.7643999800842578   -0.34850835190466067   -0.5290449526914196
0.5151090756944251   -0.37229752526512344   0.49320199498275868   -0.17249888510245559
0.67701435231287854   -0.3239470395762975;-0.18331892377473782   0.16104742392247148   -
0.075282619095164538  -0.082270879249876597  -0.43607730519423027  -0.013105145461292782
0.069563662463332662   -0.16247749341220136   0.39219146819332351   -0.14269581134433262
```

```
0.2523661727082307    -0.19500629405170372    0.752924129839041    -0.022087045777031036
0.2763401926329016   0.050279589205128539   -0.65812700664378787   0.037672619517786582
0.071072449521750125  -0.31177978152083463  0.32407046766001413  0.20309998981387617  -
0.77581929427940544  -0.37394453322432725  -0.88812519792401512  -0.34561625019228442  -
0.57644916638659816   0.39624279000487439   0.088055420694656245   -0.53141160049890057
0.14968361512905237 0.18481530606475408;-4.0051720698345474e-15 1.7210743154048142e-15
1.7209404702235279e-15          -1.166494415843441e-15          -6.4469983589312663e-17
1.0673969254713437e-15           1.8396287900907542e-15           1.2668736732647507e-15
2.0910407146842608e-15      -5.4779973896364187e-16      8.5056727298296602e-16      -
7.9030416530030947e-16      -3.2454165522633287e-15      -9.8218546202314198e-16      -
1.477800068805907e-15           1.6354684585448713e-15           4.8126407103082229e-15
3.2488339424491308e-15      2.9373365422851689e-16      2.8140984271376505e-15      -
2.8282769865057726e-15           1.5365972252632658e-15          -1.7773165026715146e-15
2.2722219111058853e-15          -1.7911864225033669e-15          -3.1485854298653562e-15
1.5283415179701335e-15      2.5414049488895419e-15      9.0000528347968741e-16      -
2.435810867553175e-15 2.5852908020298071e-15 9.2791049745768613e-16;0.45741922587430311
-0.058645776333149102  -0.35557144881541441  -0.050999296188287087  0.23929573894271314
0.46675707433013414   0.044578578868894138   -0.30679173859671172   0.067727419577432391
0.18894247077402224  -0.061253903454757175  0.067617720165652179  -0.29449918932678404  -
0.60253299831239604   0.20963685844838209   0.086123869031502209   -0.021582368733052717
0.47276387291586319  -0.016926513234679766  0.33311107842875548  0.097934115764931579  -
0.45285716868362241   0.13807353522561788   -0.20046032686040702   0.065104372546881673
0.16689880675899246  -0.41993112852438486  -0.98408287590191057  0.015281900506291921  -
0.051887068453032849     0.25719433922696294     0.63727497105513109;0.77963666724380809
0.18077969938142893   0.60861921263835217   -0.048173206428523595   -1.147893408652282l
0.11359926281612986   0.2137823953211086   0.68165365950137358   0.43774067729761096   -
0.13075479109468557  -0.01090166962099735  0.050254621623791254  0.041088622720110983  -
0.086157009600588169  0.11439425604619181  -0.86732863951467876  0.46721600671721203  -
0.44828396570404355   0.26147891049393251   -0.10661929549099181   -0.15217572897160936
0.56184618808179365   -0.11041794023759173   -0.29736433528106665   0.16113295110367173
0.5032236321216873    0.84271821126355562    -0.32148240955672541    0.30745441275158736
0.45009817551983378   -0.49871239055799649   0.2595178504316395;-0.78750455834060862   -
0.056761452345777674   0.65299570724074585   -0.24131908242616734   1.0621511121807834
0.44577671932170482  0.13591224759687565  0.82548057862674884  -0.58978564310265913  -
0.51239421791421447   -0.20485141395996889   -0.23293753357930067   -0.68009627783073823
0.1556677494867541  -0.083687234661736262  0.36814979790024055  -0.15176029411535716  -
0.20417301113590058  0.0088790380023027471  0.16836237625440686  0.89943034594151627  -
0.63576000421843182  -0.68145983397824905  -0.51153431780722969  0.42992170992092782  -
0.29741912099741252  -0.29580118876421591  -0.082445155385751145  -0.32006607022433481
0.63300744784593455  -0.68354826137962155  0.20385064097616387;-4.2500687267170483e-15
1.8262898724754952e-15      1.8262789143644956e-15      -1.2382723458336174e-15      -
6.8475483738435353e-17           1.132587296870516e-15           1.9521156142308522e-15
1.3448301942716662e-15           2.2190687396530249e-15          -5.8166767086414503e-16
9.0248769017814039e-16      -8.3890983815280605e-16      -3.4437854704120153e-15      -
1.0427946469918579e-15          -1.5684819784995233e-15           1.7353492087733527e-15
5.1074586174052951e-15           3.4478557380265745e-15           3.1141543162316234e-16
2.9864299496100316e-15      -3.0016064305466109e-15      1.6305552480059808e-15      -
1.8866313015547161e-15      2.411262720920547e-15      -1.9010484979365579e-15      -
3.3409208151033555e-15           1.6219557805521531e-15           2.6969894795832946e-15
9.5485301761180878e-16          -2.5855335476809728e-15           2.7434213955732357e-15
9.8376416171357675e-16;-2.4892030390442427e-15                    1.0696108308764926e-15
1.0695969218371685e-15          -7.2525336030735873e-16          -4.0129287030445969e-17
6.6332078053822979e-16           1.1433037993126976e-15           7.8762368024509602e-16
1.2996662440408942e-15      -3.4069072341788092e-16      5.2855348190335594e-16      -
4.9135797030540319e-16      -2.0170088764513079e-15      -6.1075884432654728e-16      -
9.1865751321955817e-16           1.0163835143461883e-15           2.9913410484423547e-15
2.0193367458912377e-15      1.8237625977643398e-16      1.7490804522568513e-15      -
1.7580162104537662e-15          9.5496375521135867e-16          -1.1049690631375612e-15
1.4122161279360174e-15          -1.113407666290803e-15          -1.9567442367865005e-15
9.4993610151534335e-16      1.5795773303419495e-15      5.5922940857786377e-16      -
1.5143107949034235e-15            1.6067748295882609e-15            5.7622906399668999e-
16;5.1847605106357992e-15        -2.2279477483327991e-15        -2.2278012954356723e-15
1.51015106231391e-15       8.3479223813179433e-17       -1.3817397253319017e-15      -
2.3814229457112952e-15          -1.6400907892489636e-15          -2.7069210039673306e-15
7.0922340225432839e-16          -1.1010473007790728e-15           1.0231305713945195e-15
4.2012384116808369e-15      1.2715806506053396e-15      1.9131137056939619e-15      -
2.1171180770939448e-15      -6.2301558567064775e-15      -4.2057436737839004e-15      -
3.8017632949919862e-16      -3.6429459152616665e-15      3.6613436609887801e-15      -
1.9891418505091847e-15           2.3009059101375384e-15          -2.9414437083787079e-15
2.3187921232752325e-15      4.075858392759386e-15      -1.9784928329381036e-15      -
3.289928763454278e-15      -1.1650263017192377e-15      3.153371222466852e-15      -
3.3467101781943387e-15         -1.2010149189294967e-15;1.9624584634200791e-15      -
8.4327827304704859e-16          -8.4326431270357229e-16           5.717529068663202e-16
3.1624156327670918e-17      -5.2296601446794578e-16      -9.0137644797699886e-16      -
6.209396277817082e-16      -1.0246384385376675e-15      2.6857140399619352e-16      -
4.1671859840201926e-16           3.8735887397190317e-16           1.5901752000853033e-15
4.8148441904727373e-16      7.2423464995154736e-16      -8.013049981020697e-16      -
```

```
2.3583215925939799e-15      -1.5920131800592786e-15      -1.4380341341431986e-16      -
1.3789532976861556e-15          1.3859721536891916e-15          -7.5289327877407544e-16
8.7111462807440199e-16          -1.1133801189523934e-15          8.7778452380910453e-16
1.5426768393436005e-15      -7.4891970853769712e-16      -1.2453157497137788e-15      -
4.4090513818938948e-16      1.193826310159441e-15      -1.2667635669860883e-15      -
4.5431384253165164e-16;-0.069361857474368946  0.23188993241729564  0.17381093651003635
0.055248061035735048   -0.34516914771627011   0.13401522077389688   0.22339002020290866
0.030872552830216636  -0.048373784710533455  0.51990076769215776  0.30489500868696362
0.016565756148530515 -0.42276780824069543 -0.030144379803423705 -0.11052742589190565 -
0.11585559234223439  -0.16412343882092445  0.44316222490425689  0.06306432324354827  -
0.43944789940124068  -0.24442851934663409  0.38044329947906574  -0.10275359698078401  -
0.13558704573773758  0.12701279865688744  0.32737789864877875  -0.75920458504966681  -
0.13353517076771174   0.11678254183689742   -0.6598597414761127   0.22331133595712099
0.65926207411169202;5.4143577632949003e-15          -2.3266121873081208e-15          -
2.3264705454633726e-15      1.5770533950216657e-15      8.7174006365711412e-17      -
1.4429270830619093e-15      -2.486885681047702e-15      -1.7127603526408529e-15      -
2.826806564156681e-15          7.4065220125866427e-16          -1.1498042079797676e-15
1.0684525982781927e-15          4.3872682747633357e-15          1.3279271207561374e-15
1.9978498204864723e-15      -2.2108581136367992e-15      -6.506091513768262e-15      -
4.3920192298808e-15 -3.969958884962507e-16 -3.8042929653317535e-15 3.8235023467936018e-
15      -2.0772349759052182e-15      2.4028438292911437e-15      -3.0717141310998453e-15
2.4215019118492628e-15      4.2563326519740319e-15      -2.0661235074171608e-15      -
3.4356332543157132e-15      -1.2166079132158621e-15      3.2930677288177295e-15      -
3.4949184870005492e-15                    -1.2541227360077918e-15;0.14238158532303535
0.00054964094627982295  0.12928919765642433  -0.065245426728667164  0.71746827995799645
0.40299818631828133   0.16000430917202771   0.68952403195344181   0.083614206986458758
0.35640057869631531  -0.12054550123871396  -0.067317367972657222  0.21133590286972151  -
0.10442226833851878   -0.26971389179085875   -0.63146746266470233   -0.23490782804330756
0.16641557238147234  -0.034062960291616567  0.15972461016736675  -0.39399558372491739  -
0.57723009975423445   0.056144057195782492   0.60869414522878396   -0.66658448939540482
0.12404216096547524 -0.66534380684806027 -0.028772300372562981 -0.10432900467088491 -
0.0300222090664247  0.30194080529263939  -0.38401499722429355;2.9450662448717583e-15  -
1.2655347936330859e-15          -1.2654546322764824e-15          8.5779627518758938e-16
4.7408913004675214e-17      -7.8486802953513288e-16      -1.3527128829833942e-15      -
9.3162002471222151e-16      -1.5375988718443852e-15      4.0284990659479774e-16      -
6.2542848897392809e-16          5.8115473485704817e-16          2.3863896791360724e-15
7.2228514911738814e-16      1.0866864782271256e-15      -1.2025671694490117e-15      -
3.5388852532089709e-15      -2.3889706665377811e-15      -2.1595456970016204e-16      -
2.0692902860646547e-15          2.0797229438925837e-15          -1.1298906316336952e-15
1.3069713436734964e-15          -1.670819315295421e-15          1.3171328355656129e-15
2.3151775448839589e-15      -1.1238384266294733e-15      -1.8687612761675873e-15      -
6.6176754189326218e-16      1.7911898872687662e-15      -1.901013096767539e-15      -
6.82183208637057e-16;0.012167573124046869  -0.12182060563133314  -0.19982138604834357  -
0.17746887144952259  -0.28542618375807272  0.031851545843189985  -0.046663889229768607
0.33237505730675371   0.050952770447509295   -0.1275098582008117   0.22895778398147884
0.081412489616566663   0.61173953327227704   -0.14329893413149353   0.1674020860868087
1.1378631568531554   -0.77217564149683537   -0.33270435996880704   -0.60401850321976902
0.43575937833174583  0.6118474167221275  0.15903154623420357  0.40195431866926806  -
0.17448885877300044    -0.80139171414628019    1.2027206886919883    0.1667338543564407
0.57162409909762046  -0.52687127068732365  0.31466337731054628  -0.09377288161031061  -
1.0060982799374596;-5.2796647321249738e-15 2.2687674478329163e-15 2.2685497620835802e-
15      -1.5375039326524526e-15      -8.4938277583761517e-17      1.4071017649597381e-15
2.4250403589220622e-15      1.6698488100282127e-15      2.7563805345790614e-15      -
7.2196200514855546e-16      1.1212799647252756e-15      -1.041663008489905e-15      -
4.2781384426796808e-15          -1.2945144132775064e-15          -1.947913764225004e-15
2.155923489297793e-15          6.3438982559463064e-15          4.2825347916843866e-15
3.8732107256991788e-16          3.7095074116045796e-15          -3.7281026809385633e-15
2.0255878453133973e-15      -2.342644454399559e-15      2.9952549302955726e-15      -
2.3610454951095404e-15          -4.1505552100546801e-15          2.0146355671314445e-15
3.3500482968695479e-15          1.1864789454052829e-15          -3.2106221220855614e-15
3.4079472172059267e-15 1.2234573962810453e-15];

% Layer 2
b2 = 0.054484046404546654;
LW2_1   =   [-1.0182238311503452e-14   -1.2559374118626356   1.2720945359798932e-14   -
1.4257740637023221e-14          -1.5923006541499374e-14          -8.1935526184934338e-15
1.6913094804646052   -1.3181370345553694e-14   1.5767131547843165   1.7306030244477704
2.0433327121647149e-14 1.5542533842199235 -1.4620080283238945 -1.0245003446798691e-14 -
1.7797814327925032   7.5455726251605344e-15   1.8299031520350832   1.5907820621472193
1.3057308216577734e-14    1.4976593800412838    1.75791981408737    1.9964451834364629
1.5193155774009948e-14      7.8093034157762883e-15      -1.5828473489417164e-14      -
6.2622649465776493e-15 1.3484656600414684 -1.5368084231350398e-14 1.5605873671917656 -
9.1212172905951956e-15 1.8082913083986045 1.9259444117309521e-14];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.000222000222000222;
```

```
y1_step1.xoffset = 983;

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
```