University of Thessaly
Department of Mechanical Engineering

Diploma Thesis

Reliability Prediction using Neural Networks in Semiconductor Manufacturing

Evangelos Tsiakiris Christos Kalantaridis

Thesis Supervisor: Dr. George Liberopoulos, Professor

Volos, September 2019

Members of the Examination Committee:

First Examiner (Supervisor)	Dr. George Liberopoulos Professor, Department of Mechanical Engineering, University of Thessaly
Second Examiner	Dr. Dimitrios Pantelis 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, we would like to thank our supervisor Mr. George Liberopoulos for his valuable help and guidance during our work and for giving us the opportunity to examine such an interesting topic, as reliability and neural networks in manufacturing.

We would also like to thank Phd student Michalis Deligiannis for his constant scientific support and encouragement. Without his crucial contribution, we wouldn’t be able to carry through with our thesis on time.

Moreover, we would like to thank Dr. Dimitrios Pantelis and Dr. Georgios K. D. Saharidis for their participation in the examination committee.

Last but not least, we would like to thank our families for all the patience and guidance through all the difficult moments we had these years. Being at this stage of our life is all thank to them. In addition, we would like to thank our friends for the great time spent together and the experience shared.

Abstract

In this thesis, the reliability of specific machines from Bosch's semiconductor plant, which is located at Reutlingen in Germany, is investigated by using neural networks in order to make future failure predictions. The processing data of the machines, provided by Bosch's factory, was taken into consideration. More specifically, three cumulative descriptors were used for the creation of each neural network model. On the one side, the machines' execution time along with the number of maintenance, compose the input parameters. On the other side, the number of faults constitutes the output parameter. As a result, the general idea is that the cumulative number of failures can be estimated depending on machine's cumulative productive time and cumulative number of maintenance. Thus, in order to accomplish this conception, the data of each machine were imported from the Excel files into the Matlab software, to be deployed on neural networks training. After the training process, the validation and the performance of every model are tested, by making predictions regarding the future failures. It should be noted that besides the prediction part, another key feature of this thesis focuses on estimating the time that the first failure occurs for each machine. Having analyzed the results, a variety of diagrams that describe the data distributions is attached, in order to provide a better physical supervision and important statistical information. In the end, we consider some ways of how the predictive models could be applied in real life production chain circumstances, to help in the optimization of scheduling.

Keywords: Neural Networks, Data Analysis, Semiconductor Manufacturing, Reliability, Failure Prediction

ΠΕΡΙΛΗΨΗ

Σ' αυτή τη διπλωματική εργασία, διερευνάται η αξιοπιστία συγκεκριμένων μηχανών από το εργοστάσιο παραγωγής ημιαγωγών της Bosch, που βρίσκεται στο Reutlingen της Γερμανίας, χρησιμοποιώντας νευρωνικά δίκτυα για την πραγματοποίηση μελλοντικών προβλέψεων βλαβών. Τα δεδομένα λειτουργίας των μηχανών που παρασχέθηκαν από το εργοστάσιο της Bosch χρησιμοποιήθηκαν για το σκοπό αυτό. Πιο συγκεκριμένα, χρησιμοποιήθηκαν τρεις αθροιστικοί παράγοντες για τη δημιουργία των μοντέλων των νευρωνικών δικτύων. Από τη μία πλευρά, ο χρόνος λειτουργίας των μηχανών μαζί με τον αριθμό συντήρησης συνθέτουν τις παραμέτρους εισόδου. Από την άλλη πλευρά, ο αριθμός των βλαβών αντιπροσωπεύει την παράμετρο εξόδου. Έτσι λοιπόν, η γενική ιδέα είναι οτι ο αθροιστικός αριθμός βλαβών μπορεί να εκτιμηθεί συναρτήσει του αθροιστικού χρόνου παραγωγής και του αθροιστικού αριθμού συντήρησης της μηχανής. Για την επίτευξη αυτής της διαδικασίας, τα δεδομένα κάθε μηχανής εισήχθησαν από αρχεία Excel στο λογισμικό Matlab, έτσι ώστε να χρησιμοποιηθούν στην εκπαίδευση των νευρωνικών δικτύων. Μετά τη διαδικασία της εκπαίδευσης, ελέγχθηκε η εγκυρότητα και η απόδοση των μοντέλων πραγματοποιώντας προβλέψεις για μελλοντικές βλάβες. Πρέπει να σημειωθεί οτι εκτός απο το κομμάτι των προβλέψεων, ένα άλλο σημαντικό θέμα που πραγματεύεται η διπλωματική είναι η εκτίμηση του χρόνου στον οποίο συμβαίνει η πρώτη βλάβη, για κάθε μηχανή. Εφόσον έγινε η ανάλυση των αποτελεσμάτων, επισυνάπτεται ένα πλήθος διαγραμμάτων που περιγράφουν τις κατανομές των δεδομένων με σκοπό να δοθεί μία καλύτερη φυσική εικόνα και να ληφθούν σημαντικές στατιστικές πληροφορίες. Τέλος, παραθέτονται μερικοί τρόποι σχετικά με το πώς τα μοντέλα πρόβλεψης μπορούν να εφαρμοστούν σε περιπτώσεις πραγματικών αλυσίδων παραγωγής, με σκοπό να βοηθήσουν στη βελτιστοποίηση του εργοστασιακού προγραμματισμού.

Λέξεις Κλειδιά: Νευρωνικά δίκτυα, Ανάλυση δεδομένων, Κατασκευή ημιαγωγών, Αξιοπιστία, Πρόβλεψη βλάβης

Contents

1 Introduction

The objective of this thesis is to predict the cumulative number of failures of the equipment used in the semiconductor industry. This equipment is responsible for manufacturing chips and sensors that are widely applied in many cases, such as the automotive industry. The provided data are based on the SEMI E10 standards, which are specifically made for the equipment of the semiconductor plant. The SEMI E10 consists of a set of states the equipment must fall in, for users' convenience.

The motivation behind our research is the desire to learn more about data analysis and prediction making in real life industry situations. Moreover, the opportunity to be involved in neural networks' science was for us of great interest, as neural networks are considered to be cutting edge technology. Besides that, Bosch's broad reputation played a significant role for us to undertake such an interesting theme. These reasons gave us the tenacity needed to overcome the obstacles we found during our research.

Working with a world leading company like Bosch, was for us a really beneficial experience. This cooperation occurred as our department of Mechanical Engineering takes part in the research project «Productive 4.0 – Electronics and ICT as enabler for digital industry and optimized supply chain management covering the entire product lifecycle». The project aims at designing a user platform across value chains and as a result stimulating the digital networking of manufacturing industries.

The structure of this thesis is organized as follows:

In the second chapter will be discussed the procedure of semiconductor manufacturing with the aim to clarify the production chain. Afterwards, SEMI E10 standards will be analyzed so as to give a better understanding of the semiconductor industry basis.

In the third chapter will be described the data identity. Then we will represent our analysis regarding the selection process we followed.

In the fourth chapter the science of neural networks will be initiated. Their utility and application in semiconductor industry will be referred as well. Moreover, a comparison between neural networks and analytic methods will be also delivered.

In the fifth chapter we will introduce the prediction models. More specifically, we will present the general form of neural network models along with the training concept we used. On top of that, we pointed out the prediction results given by our models. Thus, as we assembled these prediction results we benchmarked them against analytic methods.

In the sixth chapter we estimate the data's distributions for the purpose of gaining a better view, concerning the input and output parameters of the models.

In seventh and final chapter we draw conclusions and set directions for future work.

2 Semiconductor Manufacturing

2.1 Description of the semiconductor manufacturing

Semiconductor applications are everywhere around our life and they affect much of the world's industries. Semiconductor material has an electrical conductivity value falling between that of a conductor and an insulator. As temperature increases, their resistance decreases. Usually, the materials used in semiconductor manufacturing belong to a group of materials with solid crystal structure. For example, some common semiconductors are silicon, germanium, and gallium arsenide. More specifically, silicon is the critical element for fabricating most electronic circuits. In addition, gallium arsenide is used in laser diodes, solar cells, microwave frequency integrated circuits, and others.

By themselves, intrinsic semiconductors are not of particular use. We can alter the properties of the material by introducing foreign substances or impurities into the crystal. These impurities are also known as dopants. A crystal with an added dopant is referred to as an extrinsic semiconductor or doped material.

Due to their properties, semiconductors are widely used in our lives. Their impact can be found either on simple diode, either on transistors or on microchips.

A semiconductor chip is an integrated electronic circuit, which is highly miniaturized, consisting of thousands of parts. The first step of every semiconductor manufacturing procedure includes the raw wafers that are thin discs made of gallium arsenide or silicon. Up to several thousand identical chips can be made on each wafer, depending on the diameter of the wafer. These chips can be made by building up the electronic circuits layer by layer in a wafer fab. It is worth mentioned that there are about forty layers for the most advanced technologies. Following this, the wafers are sent to sort or probe, where electrical tests identify the individual dies that are not likely to be in a good form when packaged. Historically, dysfunctional dies were physically marked, so that they would not take place in the packaging process. In recent years, this has been replaced by producing an electronic map to identify the defective dies. The probed wafers are sent to an assembly line, where the dies with fair quality are put into an appropriate package. Finally, the packaged dies are sent to a test facility where after testing, the packages are ready to be dispatched to the customers. Wafer fabs are often called front-end fabs, while assembly and test fabs are often called back-end. Although front-end operations usually take place in highly industrialized countries, back-end operations are typically carried out in countries, where the cost of labor is cheaper. The whole manufacturing process may require up to 700 single process steps and up to three months to be produced, considering the integration scale, the type of the chip and package, along with customer's specifications.

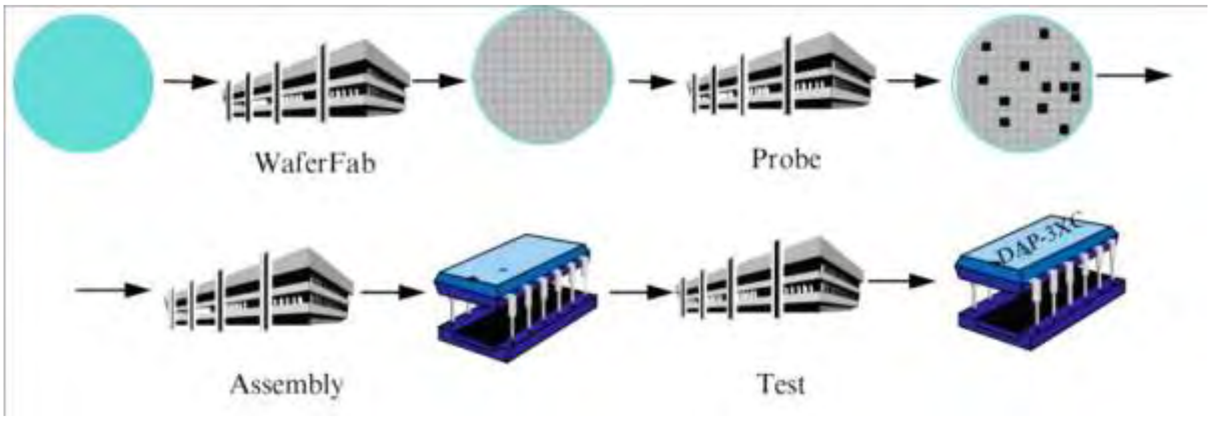


Figure 2.1The four main stages of semiconductor manufacturing. [6]

The four main stages of semiconductor manufacturing are shown in Figure 2.1. In the past, all that was needed for a semiconductor company in order to be profitable was a high quality design product. However, over the last decade, increased competition required semiconductor companies to become more efficient in order to provide the most cost effective products possible. Several performance measures are commonly used to describe and assess semiconductor manufacturing systems including machine utilization, production yield, throughput, cycle time, and on-time delivery performance-related measures. A high on-time delivery performance is crucial for customers' satisfaction. Usually, the competitiveness of a semiconductor manufacturer depends on the ability to rapidly incorporate advanced technologies in electronic products along with continuous improvement of manufacturing processes, and the capability of meeting customer due dates. In a situation where prices as well as the state of technology have settled at a certain level, the capability of meeting due dates along with the reduction of cycle time has become the most decisive factor in the fierce competition in the global market place. Consequently, short and predictable cycle times are highly desirable. Semiconductor companies have increasingly turned to data-intensive modelling and analysis tools and techniques because of their potential to significantly improve these performance measures. The semiconductor manufacturing modeling and analysis community has been working over the last twenty years on modifying general purpose manufacturing modeling tools and techniques. As a result, this could be helpful in order to handle the intricacies and complexity of semiconductor manufacturing.

Description of the Basic Process

Below is described the basic process steps of a wafer fab regarding the operations that can be performed in different work areas.

1) Oxidation and diffusion: Firstly, a layer of material is grown or deposited on the surface of a cleaned wafer. The aim of oxidation process is to grow a dioxide layer on a wafer. On the other hand, diffusion is a high temperature process that disperses material on the wafer surface. Diffusion furnaces which are typical batch machines and rapid thermal processing equipment, are placed at the oxidation - diffusion work area.

2) Film deposition: Deposition is used in order to deposit films onto wafers. In an advanced circuit, there can be a big amount of such deposition layers. Deposition can be performed by different processes, to name a few: a) physical vapor deposition (PVD) or chemical vapor deposition (CVD), b) metallization and c) epitaxy.

3) Photolithography: The basic steps of the photolithography procedure are exposure, coating, developing and process control. Initially, a thin film of photosensitive polymer (photo resist strip) coats the wafer. While an IC pattern is transferred by a photo mask, precise and accurate three-dimensional patterns are formed on the silicon wafer's surface. The pattern is transferred onto the wafer by exposure tools, called steppers. Steppers use ultraviolet light in order to complete this process by projecting the light through the reticle to expose the wafer. As a result, after removing polymerized sections of photo resist from the wafer, the exposed wafer is then developed. Due to the fact that the circuits are consisted of layers, every wafer needs to pass through the photolithography area many times.

4) Etch: This step is accountable for removing material from the wafer surface. After the photolithography step, the wafers are partly covered by photo resist strip while the non covered areas of the wafer are then removed. The etching process consists of two main categories: the wet (where liquids are used) and dry (where gases are used) etching.

5) Ion implantation: The dopant ions are deposited on the surface of the wafer in a discriminating way. Among all wafer's parts, the etched parts are the ones that are being deposited with the doping material.

6) Planarization: This step is responsible for cleaning and leveling the wafer's surface (chemical - mechanical polishing or CMP). As long as the chemical slurry is implemented to the wafer, the surface is equalized with the result of decreasing the wafer's thickness before adding a new layer.

2.2 SEMI E10

SEMI E10 is a Semiconductor Equipment & Materials International specification for definition and measurement of equipment Reliability, Availability, and Maintainability (RAM).

The semiconductor industry has been instrumental in developing a methodology for tracking and evaluating the application of information from equipment, regarding its operating condition (Trybula and Pratt 1994). Originally published in 1986 and revised in 1990, 1992, 1996, 1999, 2001, 2004 and 2011 SEMI E10-0304E which is the one we used in our analysis, is the most widely used of all SEMI standards. SEMI E10 defines the basic equipment conditions, RAM metrics, and equipment utilization measurement providing a common language and methodology between equipment suppliers and users.According to the Semi E10 specification, the considered equipment's total time is organized as described in the figure below.

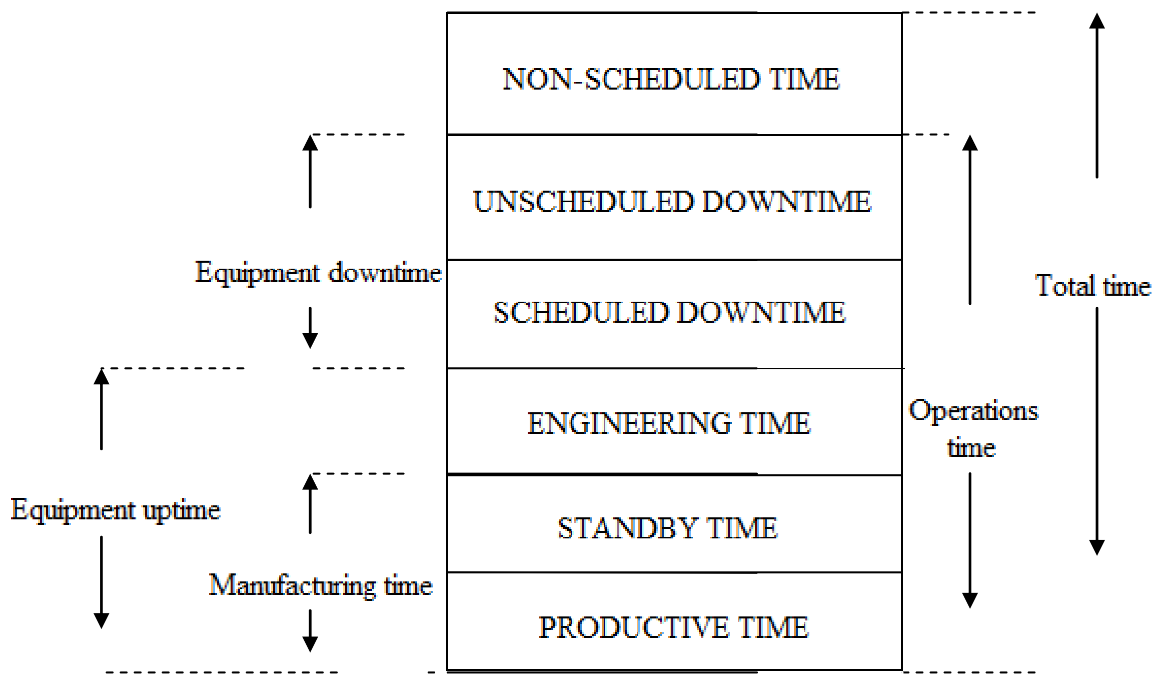


Figure 2.2 SEMI E10 distinctive times.

3 Data Analysis

3.1 Description of the data

Our research consists of four excel files (one for each machine) which contain information regarding time and date along with the corresponding state conditions. The provided data date range between 17/07/2015 – 31/12/2017 (4BSAXF18), 17/07/2015 – 28/12/2017 (2BSAXF20), 27/07/2015 – 30/12/2017 (NF671) and 15/07/2015 – 30/12/2017 (NM111). So, concerning the machines' time characteristics, we come across the following measurements :

- Uptime
- Processing time

- Productive time
- Time to restore
- Downtime
- Maintenance time
- Number of Maintenance

The mentioned above files originate from the diploma thesis of Apostolos Stamatakis and Spyridon Botsis who exported these data from the SQL server basis. A short report about each characteristic is outlined below.

- **Uptime:** The total time the equipment was up between the first and last failure.
- **Processing Time:** Sum of times that the equipment is either in process experiments and software qualifications or available - productive.
- **Productive Time**: Sum of times that the equipment is available - productive.
- **Time to Restore**: Sum of times that the equipment is in unscheduled downtime.
- **Downtime:** Sum of times that the equipment is in failure state.
- **Maintenance Time**: Total time that the equipment is in scheduled maintenance process.
- **Number of Maintenance:** The total number machine's scheduled maintenance.

The clarification of the time characteristics of machines is depicted in the subsequent chart.

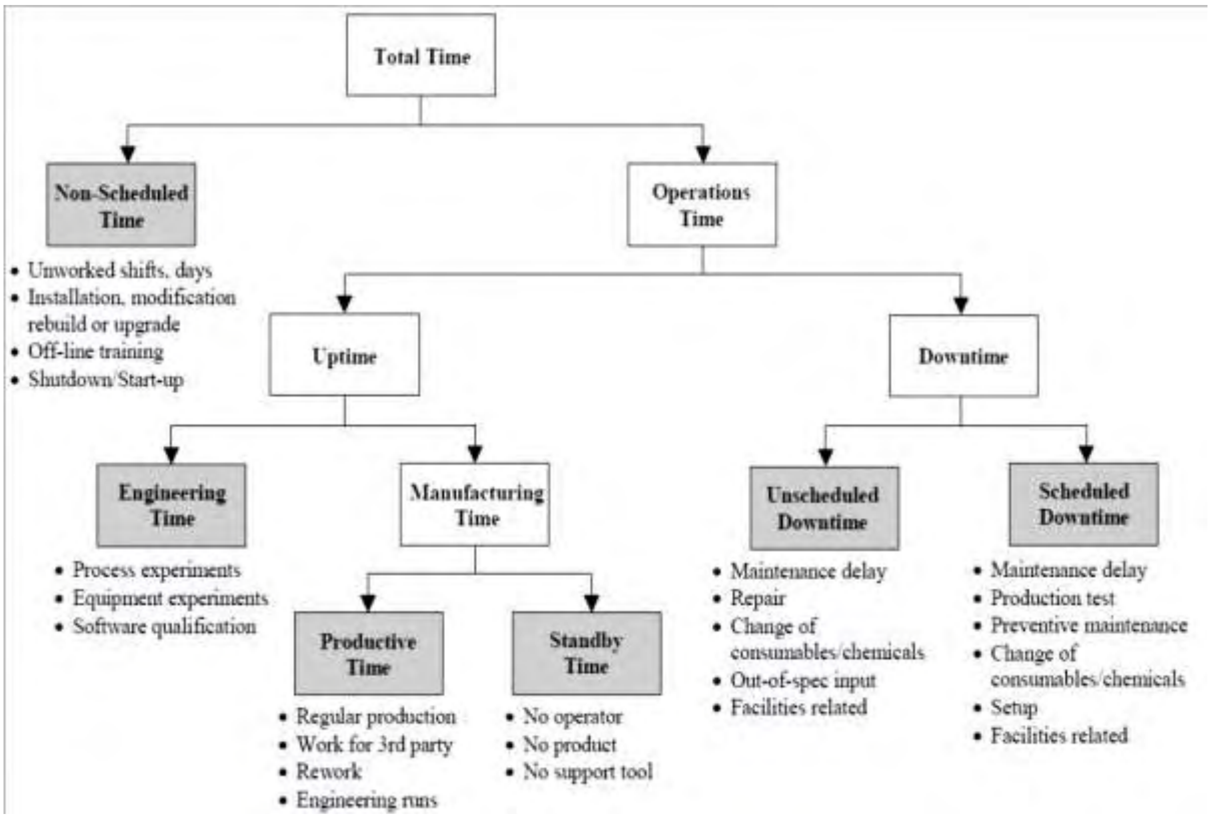
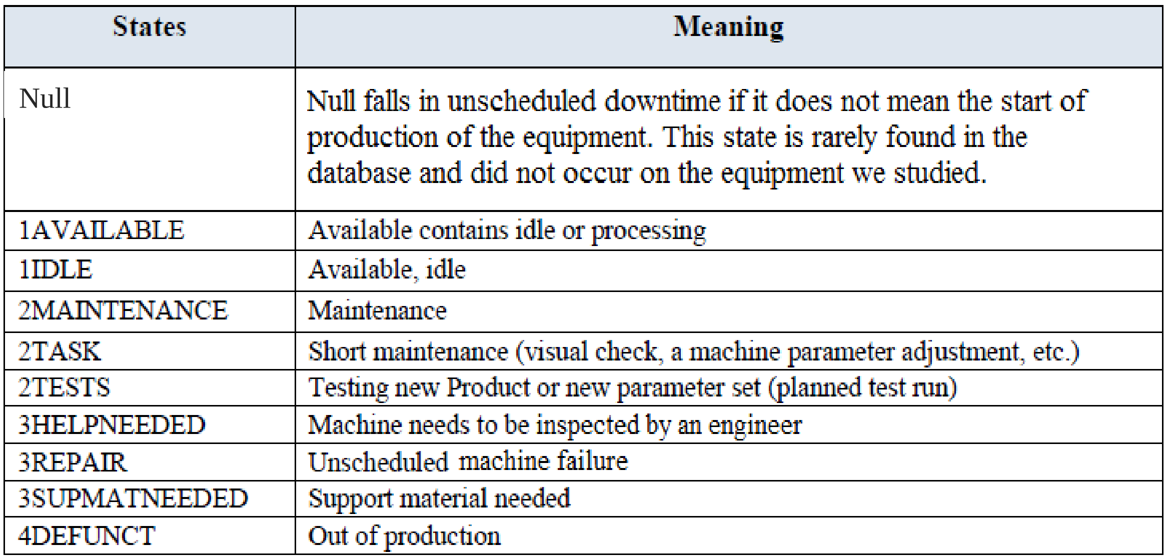


Figure 3.1 Equipment state scaling. [6]

3.2 Bosch states according to SEMI E10 standards

The SEMI E10 protocol to measure equipment performance defines nine equipment states. At any given moment each machine must fall in one and only of these states. However, in our case, we encountered only five of them, as these were the only to appear. Briefly, the equipment states are shown in the following table.



States	Meaning
Null	Null falls in unscheduled downtime if it does not mean the start of production of the equipment. This state is rarely found in the database and did not occur on the equipment we studied.
1AVAILABLE	Available contains idle or processing
1IDLE	Available, idle
2MAINTENANCE	Maintenance
2TASK	Short maintenance (visual check, a machine parameter adjustment, etc.)
2TESTS	Testing new Product or new parameter set (planned test run)
3HELPNEEDED	Machine needs to be inspected by an engineer
3REPAIR	Unscheduled machine failure
3SUPMATNEEDED	Support material needed
4DEFUNCT	Out of production

Figure 3.2 The description of the states. [6]

For better understanding of the states, we concluded a small classification of each state regarding the time characteristics.

1. The “1AVAILABLE” state contains productive time and standby time.
2. The “1IDLE” state falls in standby time.
3. The states “2TASK” and “2MAINTENANCE” fall in the scheduled downtime as both are maintenance procedures.
4. “2TASK” falls in engineering time (for planned test runs) as long as the machine is working properly.
5. The states “3REPAIR”, “3HELPNEEDED” and “3SUPMATNEED” all fall in the unscheduled downtime. However, “3REPAIR” is the only state that is equipment-related.
6. The “4DEFUNCT” state fits in the non-scheduled time as it is removed from the production network and there is no scheduled work for the equipment.

In conclusion, the attached diagram combines the states with the suitable - corresponding characteristics.

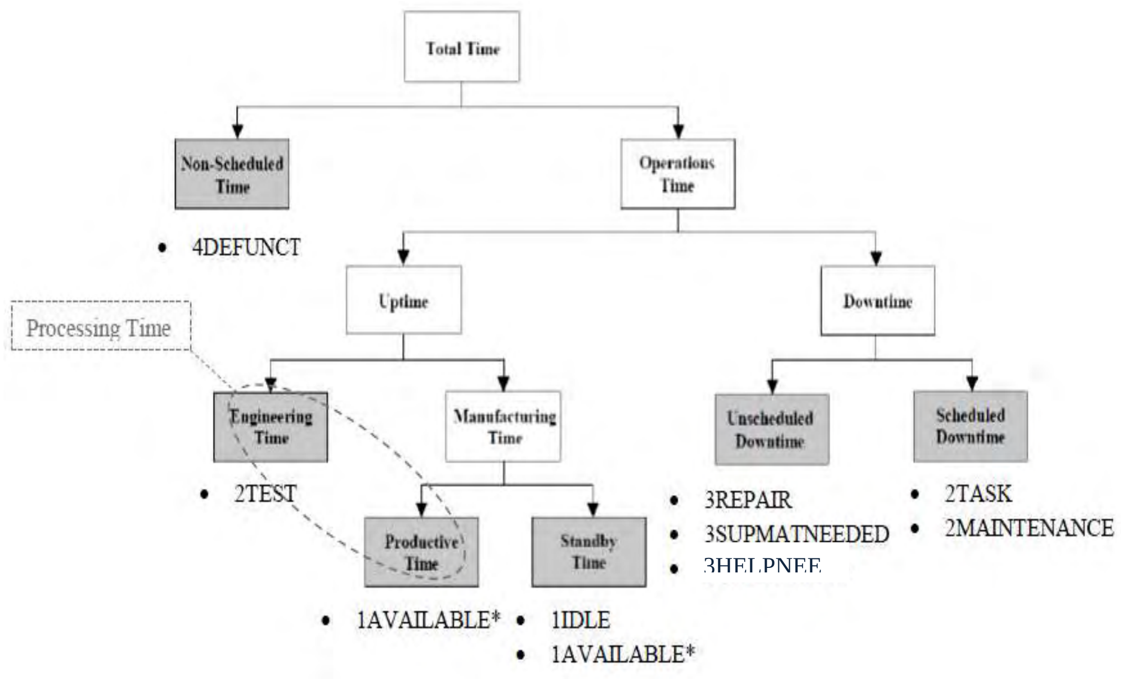


Figure 3.3 The classification of the states according to time characteristics. [6]

3.3 Useful Data Exportation

Initially, as mentioned in the previous section, from all the parameters contained in the excel files only the productive time and the number of maintenance was used on the neural network models. As a result, at first we had to measure the time that the machines were available ("1AVAILABLE" state). In order to achieve this, we calculated the differences between the time of each cell from its predecessor ($cell_{n+1} – cell_n$). Thereafter, we imported all the results into the Matlab software and then we unraveled the cases that were in available state. Consequently, after this separation process the input variable of the productive time was formed. As a next step, in order to compile the second input variable, we counted the number of maintenance state "2MAINTENANCE". Lastly, the number of faults defined by counting the amount of failure state "3REPAIR" in each data set.

The cycle used during the analysis is defined as the time period between two consecutive '3REPAIR' states. As a result, the discrete productive times and numbers of maintenance are calculated according to the mentioned above cycle. Consequently, the total number of cycles identifies with the total number of failures. The following table below shows how the initial form of data looked like:

Machine	Time and Date	Initial State	Last State
4BSAXF18	22/7/2015 2:20:46 am	1IDLE	1AVAILABLE
4BSAXF18	22/7/2015 2:28:19 am	1AVAILABLE	3REPAIR
4BSAXF18	22/7/2015 3:48:46 am	3REPAIR	1IDLE
4BSAXF18	22/7/2015 3:49:22 am	1IDLE	1AVAILABLE
….	….	….	….

Table 3.1 A sample of the initial form of the data.

The following tables below show how the final form of data looks like, after the analysis:

Machine	Time and Date	Last State	$Cell_{n+1} – cell_n$ (sec.)
4BSAXF18	22/7/2015 2:20:46 am	1AVAILABLE	453
4BSAXF18	22/7/2015 2:28:19 am	3REPAIR	4827
4BSAXF18	22/7/2015 3:48:46 am	1IDLE	36
4BSAXF18	22/7/2015 3:49:22 am	1AVAILABLE	….
….	….	….	….

Table 3.2 A sample of the edited data.

Number of Cycle	Productive Time (hours)	Number of Maintenance	Cumulative Number of Faults
1	86,191	1	1
2	0,146	0	2
3	94,744	1	3
4	35,319	0	4
….	….	….	….

Table 3.3 The exportation of parameters used.

As anyone can perceive, the initial form of the data could deliver much underlying information regarding the time characteristics (productive time, time to restore, processing time, number of maintenance, downtime, uptime, and maintenance time). This concept is finalized during our analysis, in the way that it is depicted in the final data form.

On the other hand, the final form of the data contains just the final state in which the machine was at a certain date and time, along with the subtraction results that refer to specific time parameters. Furthermore, both the discrete numbers of maintenance and the cumulative numbers of faults are also noted.

4 Neural Networks

4.1 Introduction to Neural Networks

In recent years great interest has been raised around the science of neural networks in the world of manufacturing. The inspiration behind artificial neural networks stems from the biological neural networks that constitute human brains. Their functionality tries to combine the human way of thinking with the abstract mathematical way. Thus, such networks are trained to perform tasks by taking into account examples. The origin of neural networks derives from the computational model of Warren McCulloch and Walter Pitts back in 1943 called threshold logic.The research paved the way for further penetration into this topic. Soon, the two main approaches arised. On the one hand, scientists focused on the biological processes of the brain, while on the other hand engineers focused on the application to artificial intelligence. The thing that sets artificial neural networks apart from the biological ones is that even though they learn exactly as the human brain, they generally aren't being programmed with any task – specific rules. The main aspiration of artificial neural networks

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.
A common form of a simple neural network looks as follows:

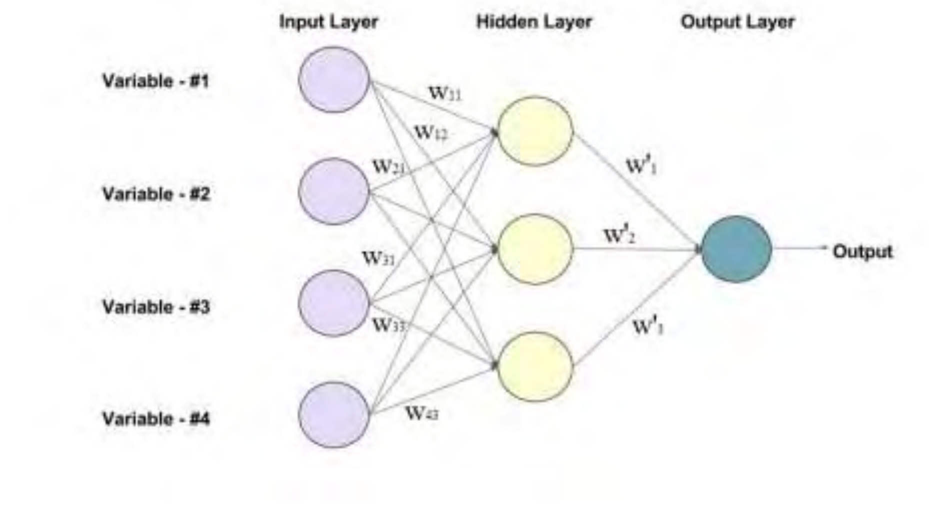


Figure 4.1 A neural network animation.

This typical neural network has four input variables, one hidden layer with three neurons and one output variable.
As we can see in the picture above, the main parts of a Feed-Forward NN (Neural Network) are:

- The input parameters
- The number of hidden layers and the number of neurons
- The output parameters

Secondarily, the following relative parts are also of particular interest:

- Weight values w_{ij}
- Biases values b_j
- Activation functions

Every neural network takes into consideration input values and calculates the corresponding output values, by forming thousand computations in a split of a second. So, at first the network should be provided with the pattern data, in which are included both the input and the desired output values (targets). In this way, by finding the intended internal format, it is capable of solving similar problems that it is not familiar with. However, it is worth noting that these problems should be of the same class with the template model. Regarding the number of hidden layers and neurons, there are no strict rules to define a good network topology just from the number of inputs and outputs. It depends critically 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. It is crucial for the operational qualification of the model to pay attention to the problem of overfitting. Overfitting occurs when the neural network becomes too complex, by having an unnecessarily big amount of neurons or hidden layers that makes it insensitive to input changes. In contrast, when neural network model is too elementary then we face the opposite

problem (underfitting). After conceiving the basic core of the articial neural networks, it is now a proper opportunity to go into details. Weights are the most essential factors in converting an input to impact the output. They constitute a connection between neurons. Weights are numerical parameters which determine how dynamically each of the neurons affects the other. This is inspired by the slope in linear regression where a weight is multiplied to the input to add up and shape the output. For instance, if a typical neuron contains the inputs x_1, x_2, x_3 then the synaptic weights are noted as: w_1, w_2, w_3.
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.

$$output = sum(weights \times inputs) + bias$$

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 as shown below:

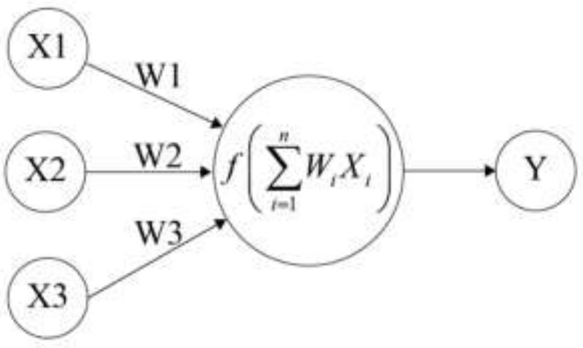


Figure 4.2 The connection between inputs - hidden layer – output.

Three of the most commonly used activation functions are:

- The linear function
- The log-sigmoid function
- The hard-limit function

The sigmoid transfer function shown below takes the input x that x ∈(-∞, +∞), and produces an output y that y ∈[0, 1]. This transfer function is commonly used in backpropagation networks. The general form of sigmoid's function is:

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

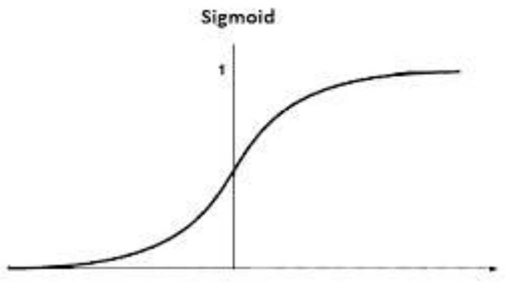


Diagram 4.1 The sigmoid function.

The linear transfer function shown below calculates the output of the neuron by simply returning the value that it was passed to it. This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. A linear network is unable to perform a nonlinear computation.

$f(x) = purelin(x)$

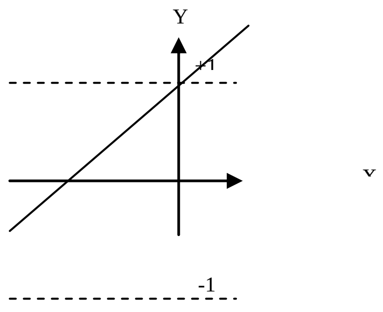


Diagram 4.2 The purelinear function.

The hard-limit transfer function shown below is used to form neurons that make classification decisions.

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

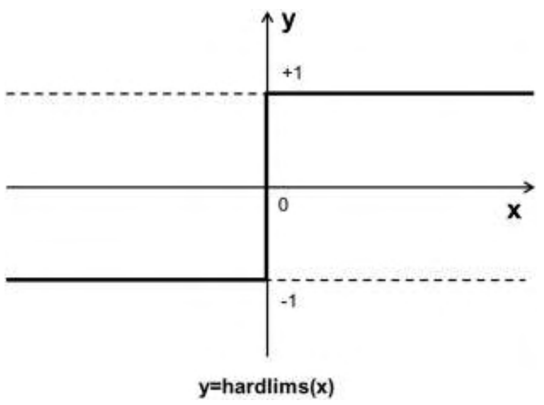


Diagram 4.3 The hard-limit function.

The most widely known training method for an artificial neural network is backpropagation. The key feature of this method is to redefine the weights, with the aim of minimizing the error between network outputs and target values. Backpropagation in a neural network is noted as the transmition of the information that relates to the error occurred by the network, when it makes a prediction about the data. The related learning algorithm can be divided into two parts and is described as follows:

- Propagation

  - The first step is called forward propagation in which the training pattern's input passes through the neural network to produce the propagation's output activations.
  - The second step is called backward propagation in which the mentioned above propagation's output activations go through the neural network the other way around, using the training targets in order to result in the corresponding errors of all output and hidden neurons.

- Weight update

  - Firstly, the output error and input activation must be multiplied to get the gradient of the weight.
  - Then, the weight must be brought in the opposite direction of the gradient by subtracting a learning rate from the weight, because the sign of the gradient of a weight indicates where the error is increasing.

Briefly, the back propagation steps can be summarized as follows:

1. Calculation of the network's error.

   $$error = \frac{1}{n} \times \sum_{i=1}^{n} (target_i - \ output_i)^2$$

   Where n = the number of outputs.

2. Calculation of the new weights.

   $$w_{ij_{new}} = w_{ij_{old}} - \ n \times \frac{\partial(error)}{\partial(w_{ij_{old}})}$$

   Where n = the learning rate and indexes i, j refer to each neurons' weight connection.

3. Redefinition of the network's error.
4. The loop procedure ends when the network achieves the intended error.

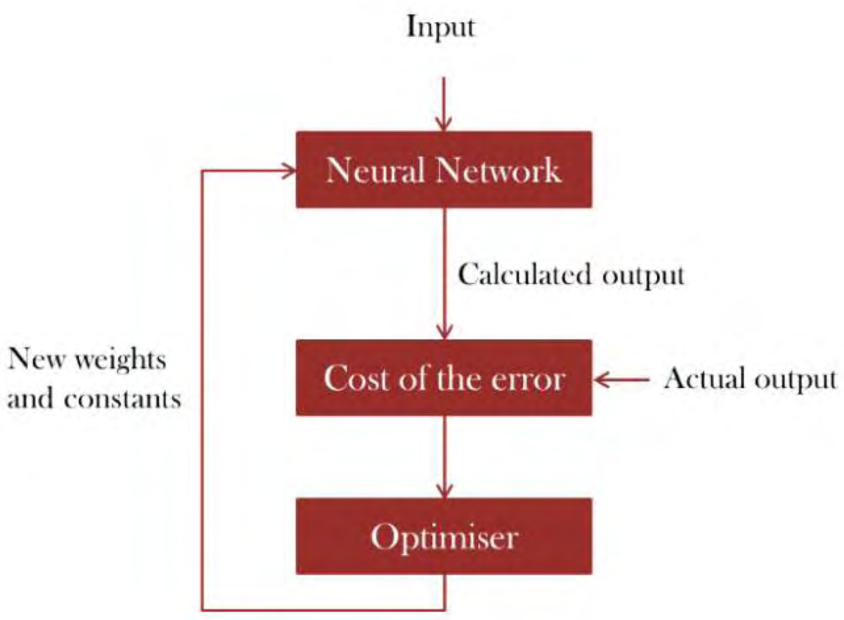


Figure 4.3 Backpropagation's algorithm.

4.2 Neural Networks in Manufacturing

Nowadays, new techniques are constantly introduced and adopted in the contemporary manufacturing sector. In recent times, there has been a raise of interest in applying artificial neural networks to manufacturing. Artificial neural networks have several upsides that are desired in manufacturing practice, including the ability to learn and adapt, performing computations, etc. There is the believe that neural networks and their techniques could lead to the grant of truly intelligent manufacturing systems.

In many cases, a NN gives the opportunity to predict the mechanical properties of processed products based on specific technological parameters. Therefore, the implementation of a NN benefits the industry by reducing the cost and saving the material resources. Neural networks can be used for monitoring, controlling and optimizing the production process.

The idea of using neural networks aims to map many inputs into multiple outputs without knowing the underlying function. One of the practical issues that goes with it is that the exact inputs which in fact affect the outputs are unknown. Often taking into account input variables that are considered as surplus parameters leads to an over complicated model. Unfortunately, a feed-forward neural network is like a "black box" and so it doesn't provide any insight for the correlation between the inputs and the outputs.

Neural networks deliver exceptional stability and robustness as well as efficient control of critical costs, such as WIP. Moreover, they enlighten many other manufacturing characteristics such as machine failure by making useful predictions.

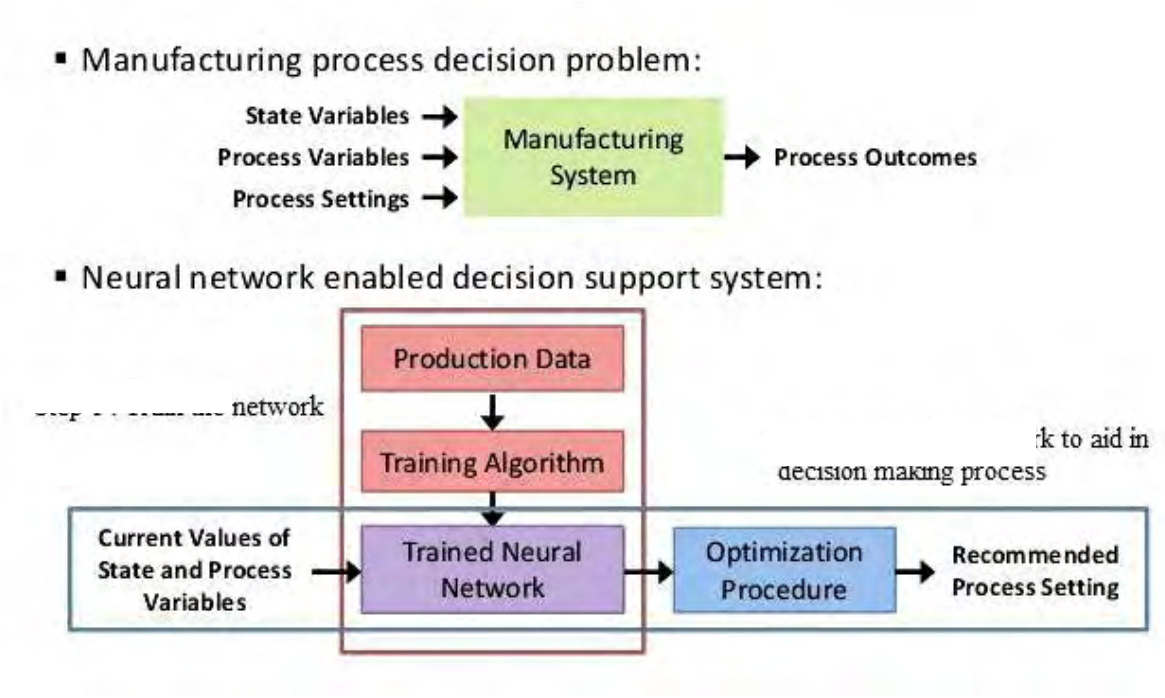


Figure 4.4 Neural networks in production decisions.

4.3 Analytic models compared to Neural Networks

Statistical modeling and neural networks are two independent but converging methods which carry through with the task of learning from data. Neural networks are a part of machine learning, which is focused on algorithmic approaches. On the other hand, statistical modeling, introduces the probability distribution fitting of the observed data. Although neural networks and statistical models share the same goal, learning from data, this goal is pursued differently. On the one side, NNs emphasize predictive accuracy, while on the other side statistical modeling targets interpretability and parsimony.

Neural Networks	
Advantages	*Disadvantages*
✓ The loss of data does not affect its functionality.	• Powerful hardware dependence.
✓ NN's ability to produce output even with incomplete information.	• It doesn't give a clue as to why and how it produces the output.
✓ Parallel processing capability – numerical strength and multitasking.	• Appropriate NN's architecture is achieved through experience and trial & error (no specific rules).
✓ Even though they are trained from one example, they are able to solve a great variety of similar problems.	• Vulnerable to over fitting and under fitting states as they depend a lot on training data.
✓ NN's flexibility as they can be used to solve both regression and classification problems.	
✓ They require only the machine's failure history no assumptions.	

Statistical methods	
Advantages	*Disadvantages*
✓ Patterns and correlations are direct and easily comprehensible.	• They require assumptions in order to perform predictions.
✓ They provide insight regarding the exact calculations that lead to the results.	• In most cases are less accurate than neural networks, as neural networks adjust model complexity to manage the complexity of the failure history.
✓ They usually resolve the problems through approaches that are easily understood.	• Sometimes, the existing statistical distributions are inadequate to solve the problem.
✓ A great range of bibliography and much research has been delivered through the years.	• A strong mathematical background is required.
✓ They provide a variety of useful metrics (mean value, variation etc.) that describe the data.	

5 Neural Network models in reliability prediction

5.1 Neural Network models' architecture

Initially, the architecture of the NN is the first basic key feature. In order to deliver trustworthy results, the model should have the right form, otherwise the learning procedure fails. Finding the right form of a neural network is not a very easy task as there are no strict rules to follow. This process requires multiple trial & error attempts along with user's experience to be accomplished. The main structure of the neural network consists of three parts:

- The input parameters
- The hidden layers and neurons
- The output parameters

The amount of input and output parameters is defined by the nature of the problem while the number of hidden layers and neurons is subject to user's decision. According to the complexity of the problem and the data's volume user is able to set the appropriate values.

The optimal structure for each machine model is depicted as follows:

4BSAXF18

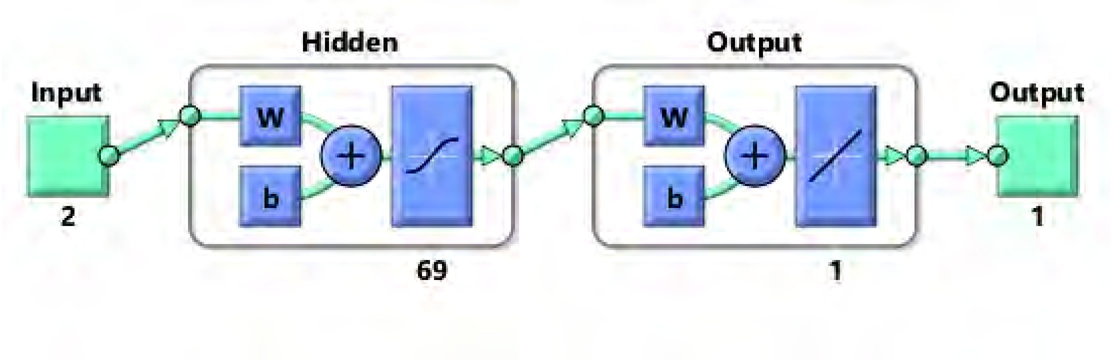


Figure 5.1 Architecture of the model for the machine 4BSAXF18.

2BSAXF20

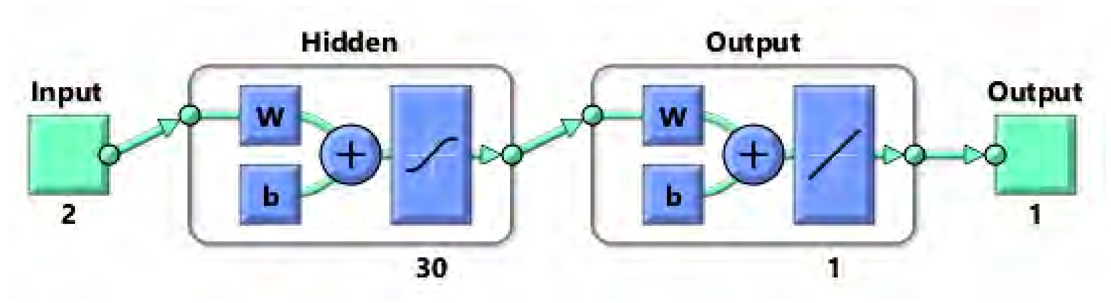


Figure 5.2 Architecture of the model for the machine 2BSAXF20.

NF671

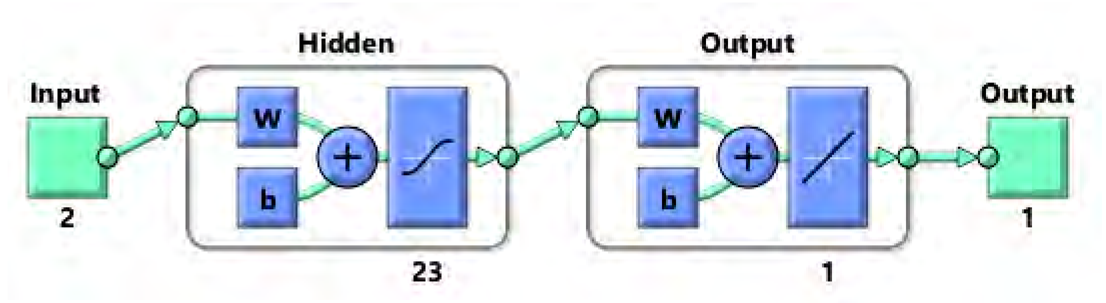


Figure 5.3 Architecture of the model for the machine NF671.

NM111

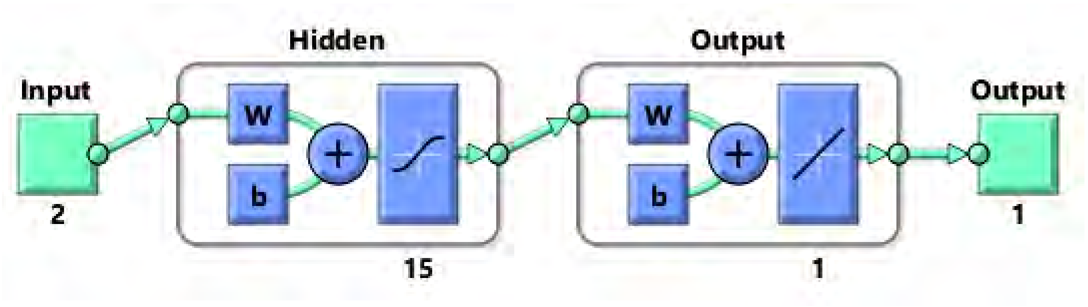


Figure 5.4 Architecture of the model for the machine NM111.

As shown above we used:

1. Cumulative productive time and cumulative number of maintenance as the two input parameters.
2. One hidden layer with different optimal amount of neurons for each machine.
3. Sigmoid transfer function in the hidden layer and pure linear function in the output layer for better accuracy.
4. Cumulative number of faults as the output parameter.

MACHINE	*Number of rows (volume of data)*	*HIDDEN LAYERS*	*NUMBER OF NEURONS*
4BSAXF18	51 868	1	69
2BSAXF20	34 334	1	30
NF671	11 461	1	23
NM111	3 960	1	15

Table 5.1 The structure of neural networks for each machine.

5.2 The training procedure

After neural network models creation the next step is the training process. As we have already mentioned this procedure is crucial for the network's proper function. The training is based on the provided data from which the model composes the suitable learning path, in order to produce the right outputs. What this means is that the weights and biases have reached optimal values. A typical training process consists of several iterations (epochs). During each epoch the model provides the network with a sequence of training pairs. The parts of each training pair are the input values and the corresponding target values. More specifically, in our case each training pair is formed by the cumulative productive time and the cumulative number of maintenance (as inputs) along with the cumulative number of machine's failure (as output). Typically, at the beginning of training, network weights and biases are initialized with a set of small random values.

Before training procedure starts, the user should provide the network with the main training characteristics which are:

- The training function
- The number of iterations (epochs)
- The tolerance limit – mean squared error
- The division of the data regarding training, validation and testing

Knowing which training algorithm will fit best in each case, is almost impossible. It depends on many factors, such as the complexity of the model, the error goal, the number of weights etc. In our situation we make use of the Bayesian regulation training function (trainbr) as it was the most accurate comparing to all the others.

Depending on the training function the number of epochs usually ranges between 0 – 1000 as default. However, it is worth mentioned that if the error rate increases continuously for more than six (default) epochs then the training procedure stops. Using the Bayesian regulation function the number of iterations reaches the maximum set value.

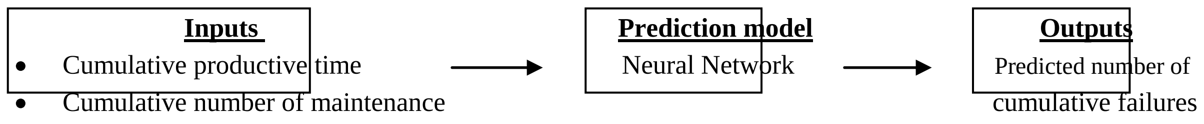
As the next step, the algorithm computes a cumulative squared error between the targets and the actual outputs. Then the gradient of the sum squared error is used to adjust the weights so that the error measure decreases in the future iterations. At that point, the training process is terminated when the cumulative squared error reaches a specific tolerance limit.

Last but not least it is necessary to set the percentage of data which are utilized for validation, training and testing. To avoid the danger of making the model too robust we subtracted the pairs that had productive time less than 1.2 minutes. Especially, for the machine 2BSAXF20 we additionally eliminated the first data pair for practical reasons.

5.3 Tailoring the models for prediction

After the training process, the neural network model should be ready for machine's failure prediction. Having obtained optimal values of weights and biases, the network is ready to deliver outputs for given inputs that are outside from the training data set.

In order to test the prediction model we calculated, the mean values of productive time and maintenance's number. This allows us to get a general understanding concerning the expected time period between each failure. The prediction process is summarized as follows:



Inputs
Cumulative productive time
Cumulative number of maintenance
Prediction model
Neural Network
Outputs
Predicted number of cumulative failures

Figure 5.5 The basic process of performing predictions.

At the beginning of prediction procedure the user must provide the model with the inputs. Then the model classifies the given inputs based on the training data set to perform the appropriate fitting. Basically, the network's algorithm passes the inputs through the existing learning path in which is involved the whole architecture of neural network (weights, biases, neurons, hidden layers etc.) and so it estimates the requested output values. It should be mentioned that this automatic process is very quick. Through the learning path the network is already able to identify the given inputs, as like classifying them into the training data set to compare them with similar values. Thus, the network is able to provide outputs in every circumstances.

5.4 Networks' performance and accuracy

Since the prediction process is performed and having the input – target pairs, we are able to examine the network's accuracy and performance. More specifically, these two factors are defined in general by the mean squared error and the R-value. At the end of the training procedure the model calculates the mean squared error of the neural network which characterizes model's performance. This error value should be close to zero in order to achieve better accuracy. The mean squared error is calculated as it is shown below:

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (target_i - output_i)^2$$

The network's MSE is estimated by using all the target values from the data set and the corresponding predicted outputs. The next diagrams show the correlations between the MSE and the number of iterations.

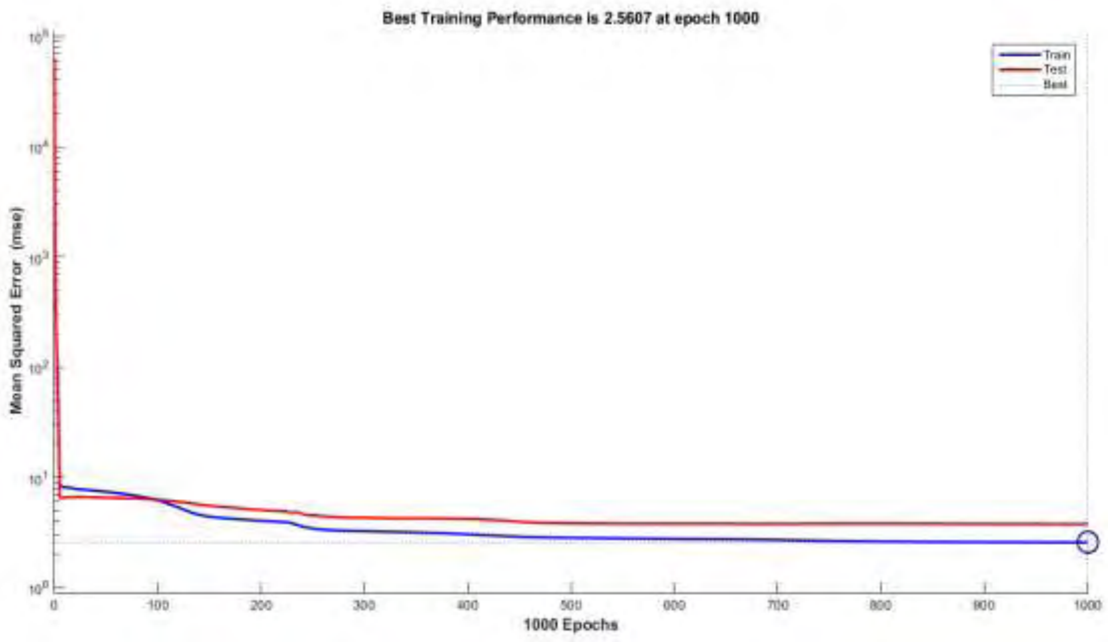


Diagram 5.1 MSE – epochs diagram for the machine 4BSAXF18.

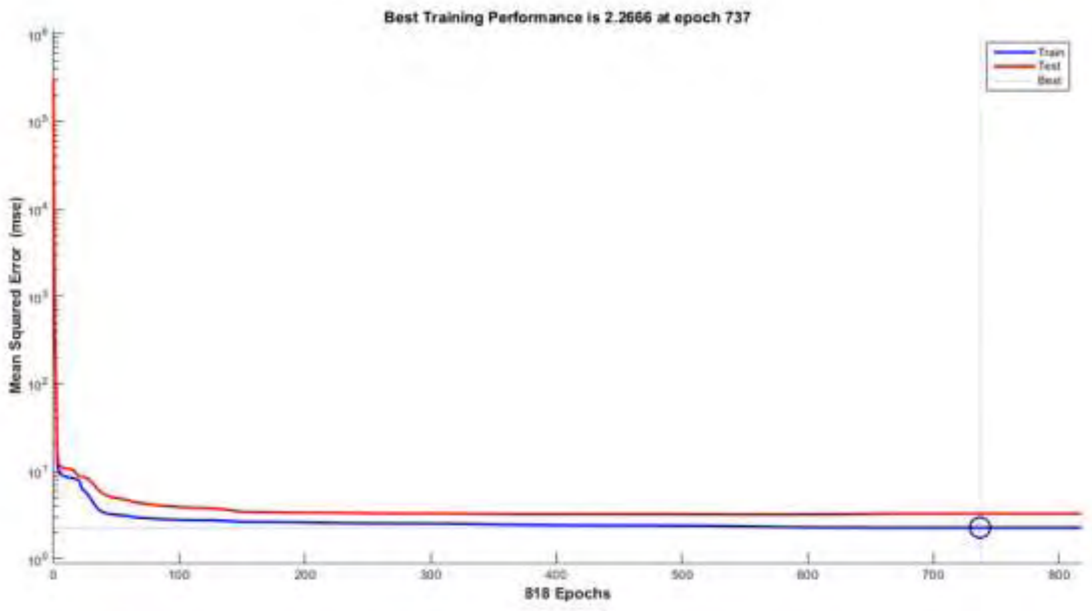


Diagram 5.2 MSE – epochs diagram for the machine 2BSAXF20.

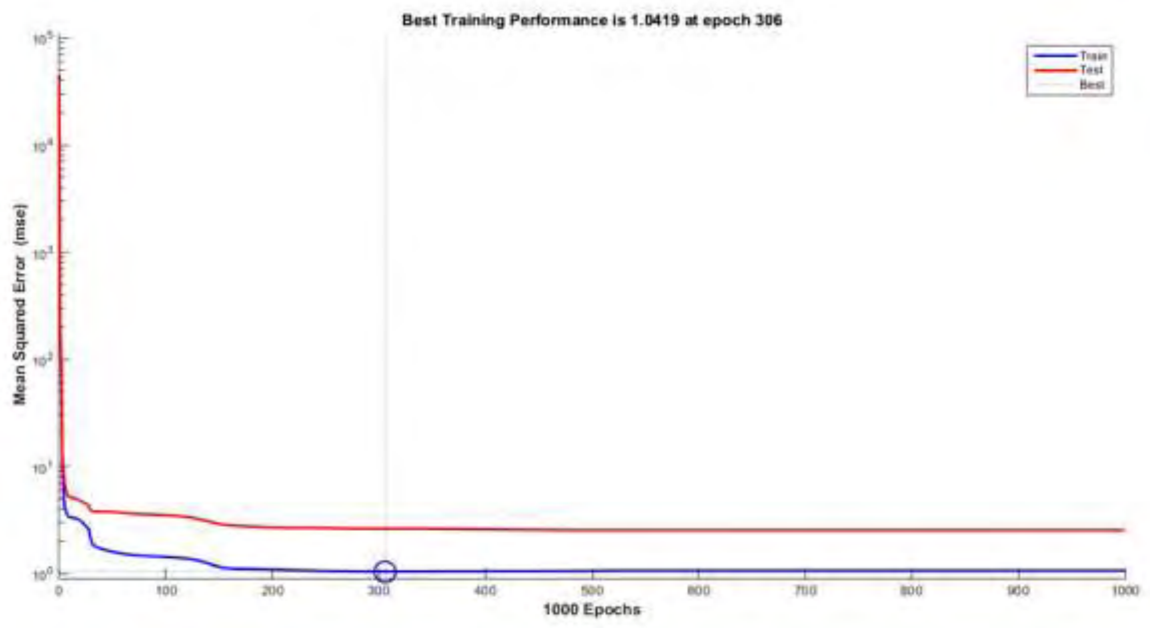


Diagram 5.3 MSE – epochs diagram for the machine NF671.

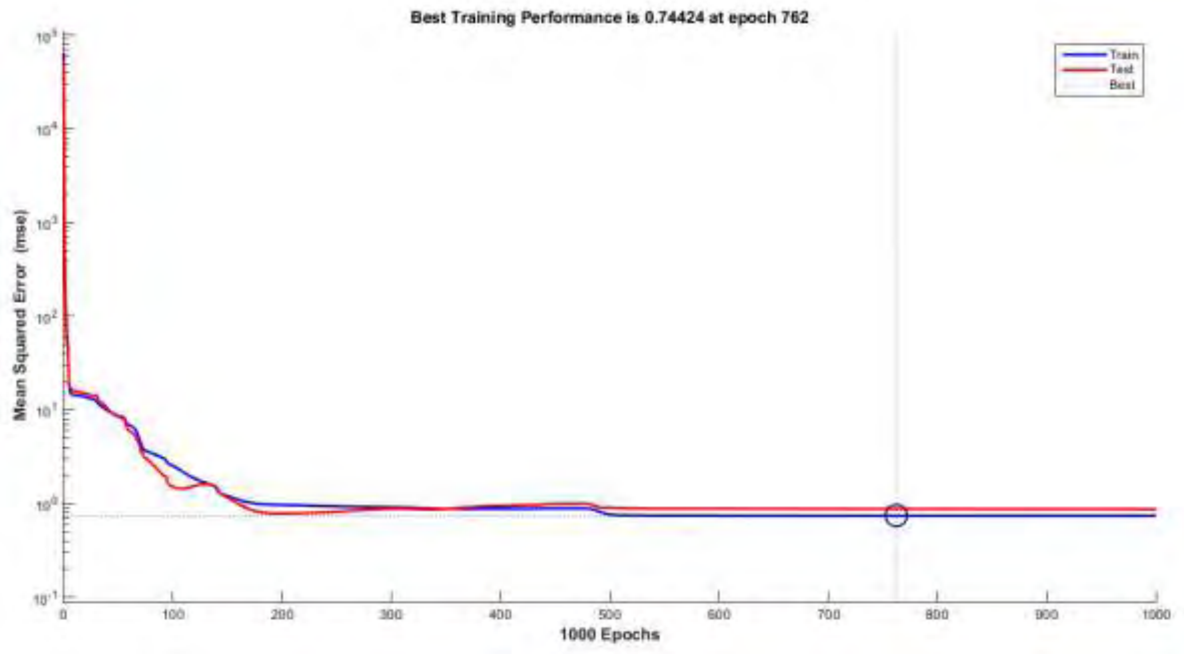


Diagram 5.4 MSE – epochs diagram for the machine NM111.

It is obvious that during the training process the mean squared error decreases, as the number of iterations increases. The best training performance is achieved at the end of the training.

Other critical factors that affect network's performance and accuracy are the R-value (coefficient of correlation) along with R^2 – value (coefficient of determination). The R-value is used to measure the correlation between actual and predicted values and it computes the direction and strength of the linear relationship between them. An R-value close to one is preferable as it shows that the network has achieved the best fitting performance.

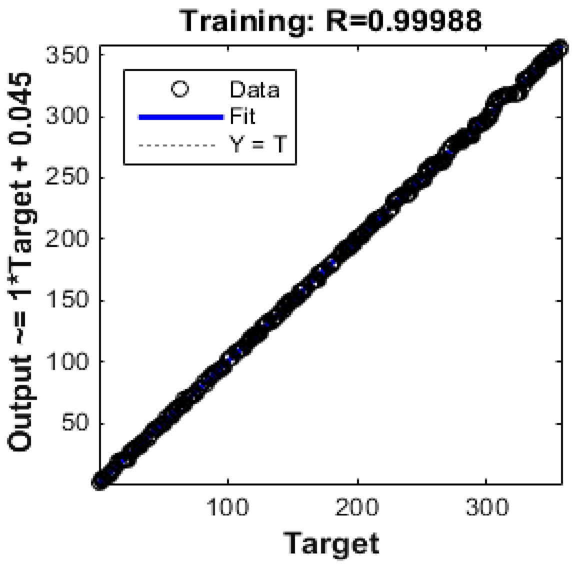


Diagram 5.5 The R – Value for the machine 4BSAXF18.

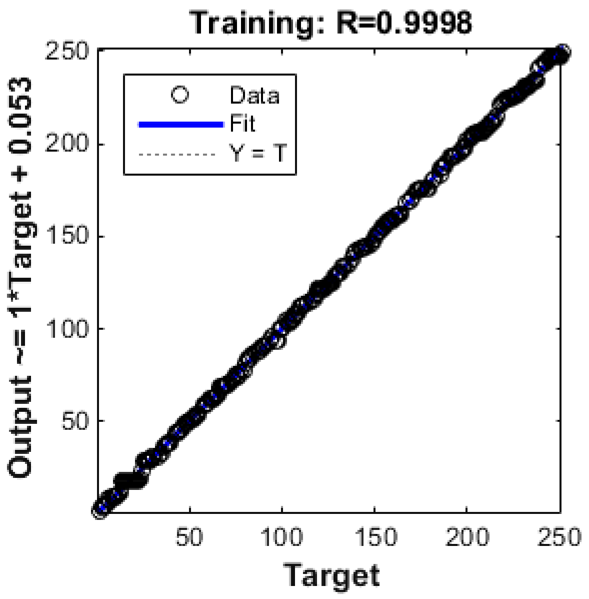


Diagram 5.6 The R – Value for the machine 2BSAXF20.

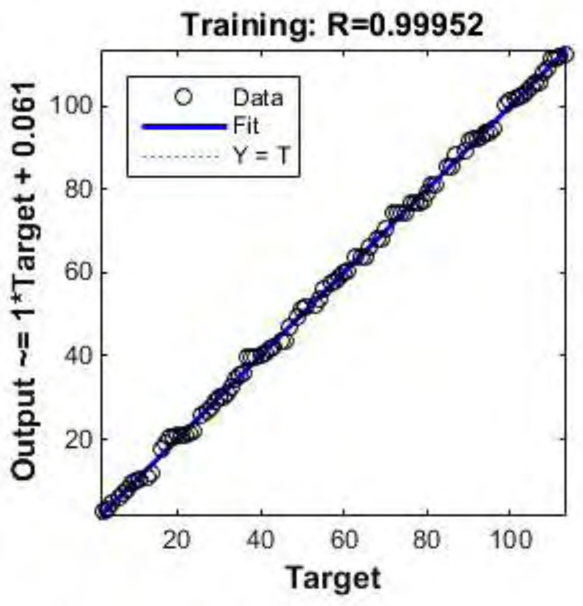


Diagram 5.7 The R – Value for the machine NF671.

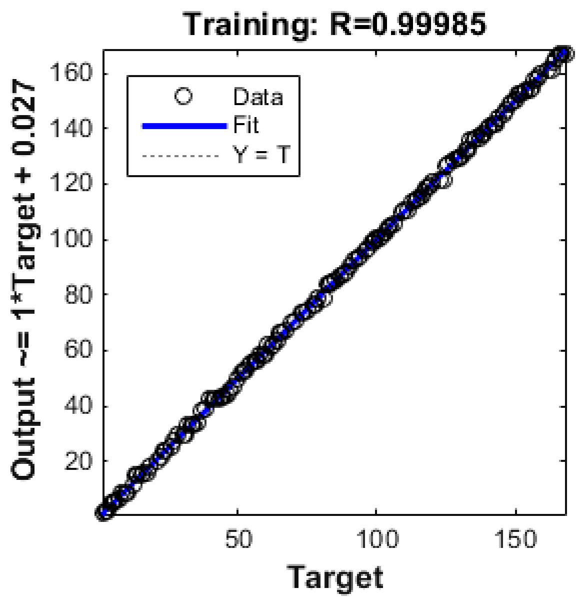


Diagram 5.8 The R – Value for the machine NM111.

Coming across an R-value close to one allows us to perform accurate predictions. However, it is necessary to supply the network with a proper amount of data, in order for it to function precisely even when the given inputs are out of the training bounds. Otherwise, the neural network becomes disoriented. Furthermore, the distributions of the data are playing crucial role in model's accuracy as the concentration of the training points characterizes the whole system.

It is worth mentioning that the output results of the neural network models consist of decimal values. These values describe the number of failures which obviously in practice should be integer numbers. However, these decimal numbers could be conceived in theory as a 'rate of failure' which differs to any known theoretical probability value. As a result, it can be received as a measurement of how close the machine is to failure. Having to deal with a data driven model, this is something distinguishable and should not be confused with the concepts derived from the probability theory.

Last but not least, the models perform well on short term predictions but as prediction periods build up, accuracy decreases (due to the lack of long term data). In order to test the prediction accuracy of every model, we perform the training procedure without using the last five provided data cells. These last five data cells were the ones used as target values for this comparative process.

<u>Machine 4BSAXF18</u>

Prediction (number of failures)	***Target (number of failures)***	***Error (%)***
354.700	353	0.48
354.900	354	0.25
355.300	355	0.08
356.400	356	0.11
365.900	357	2.40

Table 5.2 Prediction accuracy table.

<u>Machine 2BSAXF20</u>

Prediction (number of failures)	***Target (number of failures)***	***Error (%)***
245.300	247	0.68
245.310	248	1.00
245.314	249	1.40
245.316	250	1.87
246.700	251	1.70

Table 5.3 Prediction accuracy table.

<u>Machine NF671</u>

Prediction (number of failures)	***Target (number of failures)***	***Error (%)***
108.700	109	0.27
111.600	110	1.40
112.010	111	0.90
112.410	112	0.36
112.240	113	0.67

Table 5.4 Prediction accuracy table.

Machine NM111

Prediction (number of failures)	*Target (number of failures)*	*Error (%)*
165.500	164	0.91
169.150	165	2.50
172.250	166	3.70
172.530	167	3.30
173.150	168	3.00

Table 5.5 Prediction accuracy table.

Improving the network's accuracy is not an easy task but there are many steps which could lead us to the right direction. At first, changing the number of neurons and hidden layers via trial & error is a common way to start. As it is mentioned in a previous chapter, the number of these basic parts depends on the amount of the existing data. If that is not enough, the next step is to choose another training function that fits the data properly.

5.5 Implementation in production scheduling

On the topic of production optimization, scheduling is a key part. Manufacturing and production industries run on timelines, where it must be insured that all the processes are completed in proper sequence as efficiently as possible. The areas of production scheduling are the raw material procurement, the staff administration and the machinery operations which is the main focus of this thesis. The virtues of the created model are numerous and they have a great impact on manufacturing procedure. For instance, the ability to predict machine's failure gives the opportunity to optimize the production planning by adopting a suitable maintenance schedule. It should be pointed out that the main idea of this thesis concentrates on figuring out the estimated time of the first failure before and after the first maintenance. In practice, this is depicted in the following diagrams:

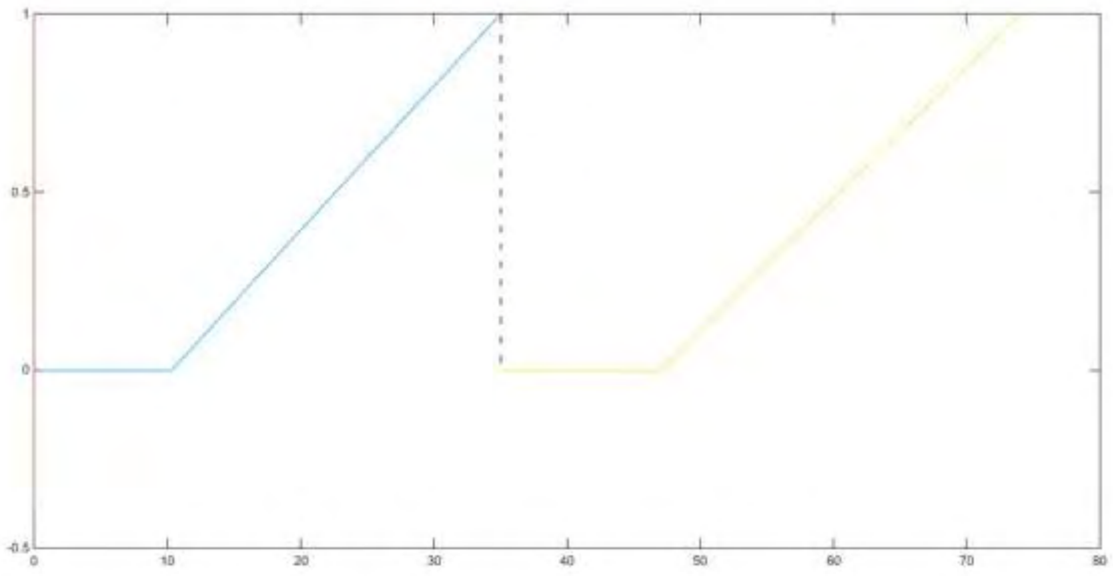


Diagram 5.9 Productive time – rate of failure diagram for the machine 4BSAXF18.

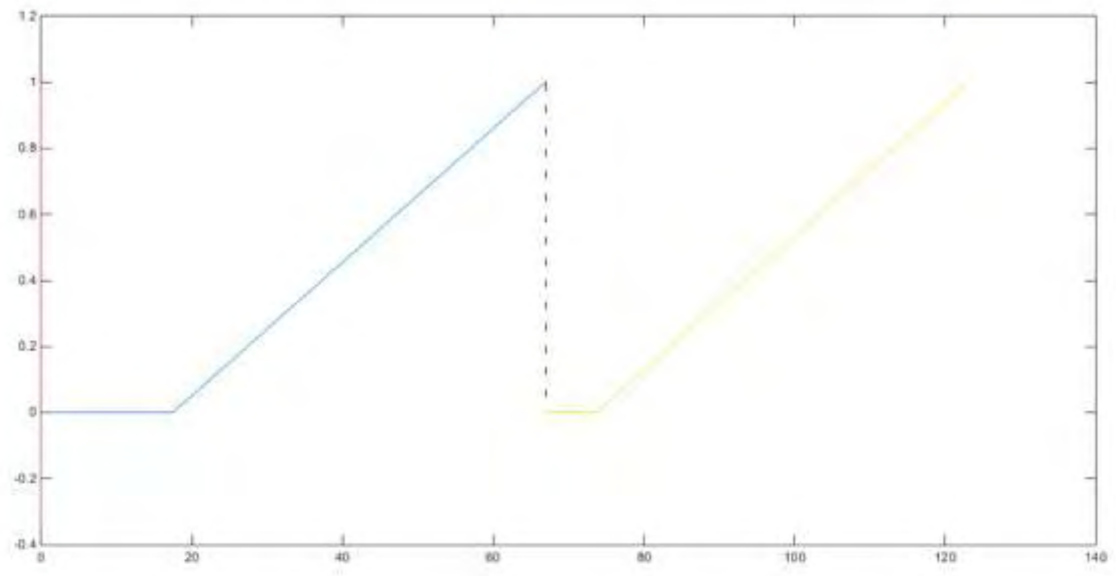


Diagram 5.10 Productive time – rate of failure diagram for the machine 2BSAXF20.

The X-axis refers to the machine's productive time, while the Y-axis refers to the failure rate. On the one hand, the blue line describes the way that the rate of failure changes as the productive time increases, when the number of maintenance is equal to zero. On the other hand, the orange line describes the way that the rate of failure changes as the productive time increases when the number of maintenance is equal to one. When the graph reaches the line Y=1, the machine fails so it needs to be repaired. The parallel to the Y-axis discontinuous line show how the failure rate falls, because maintenance occurs at this time. So, these diagrams can be used for providing information regarding the exact maintenance time schedule to prevent machine from failure. It is really valuable having this knowledge from the diagram, as usually the machine's maintenance time is less than the time to restore. As a result, the whole production system becomes more efficient and less expensive. Moreover, another significant information that can be received is that the user gets insight of how close the machine's state is to failure at every time of production. In cases that the industry owns a certain amount of capital for maintenance process, it is vital for the company to design an economic - timeline machine's plan. In other words, if the company owns a specific amount of money referring to maintenance cost then the machine's durability for this specific amount of money becomes known.

Lastly, it should be mentioned that for the machines NF671 and NM111, the quantity and the sequence of their data did not allow the formation of the corresponding diagrams.

6 Statistical distributions

6.1 Useful statistical distributions

At the beginning of this chapter, we are going to examine some useful statistical distributions concerning the productive time values. Through this process, the pattern behind these values will be defined, providing us with helpful information. In probability theory, the statistical distribution is a function which produces the probabilities of occurrence of different possible outcomes. As long as the input parameters are the productive time and the number of maintenance, conditional probability distribution can be used for the productive time when the number of maintenance is known to be a particular value.

The method used for finding the best distributions is described as follows:

1. Fitting different probability distributions to the data and selecting the best one.
2. Performing a Kolmogorov - Smirnov goodness-of-fit statistical test for the best fitted distribution.

The Kolmogorov - Smirnov test measures the distance between the cumulative distribution functions of the reference distribution and the empirical distribution function of the sample. The aim of this process is to find which empirical distribution (if any) is the most suitable for any given sample of data.

The candidate statistical distributions considered are:

- Beta
- Birnbaum-Saunders
- Exponential
- Extreme value
- Gamma
- Generalized extreme value
- Generalized Pareto
- Inverse Gaussian
- Logistic
- Log-logistic
- Lognormal
- Nakagami
- Normal
- Rayleigh
- Rician
- T-location scale
- Weibull

A brief description of some of the most common distributions follows below:

- Weibull distribution

The Weibull distribution is one of the most widely used lifetime distributions in reliability theory. It is a versatile distribution that can take on the characteristics of other types of distributions, based on the value of the shape parameter, k.
The Weibull distribution is given by the following function:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(\frac{x}{\lambda})^k}, & x \geq 0 \\ 0 \quad , & x < 0 \end{cases}$$

where the variable x and the parameters λ and k are positive real numbers.
The Weibull distribution is depicted in the following graph:

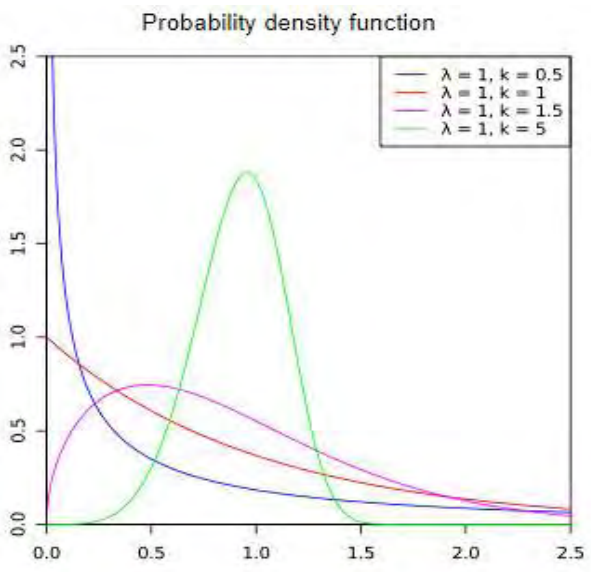


Diagram 6.1 The Weibull pdf.

- Exponential distribution

The exponential distribution is one of the most commonly used continuous distributions. It is used to define the elapsed time between two sequential events.
The exponential distribution is given by the following function:

$$f(x;\lambda) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}}$$

Where both variables x and α are positive real quantities.
The exponential distribution is depicted in the following graph:

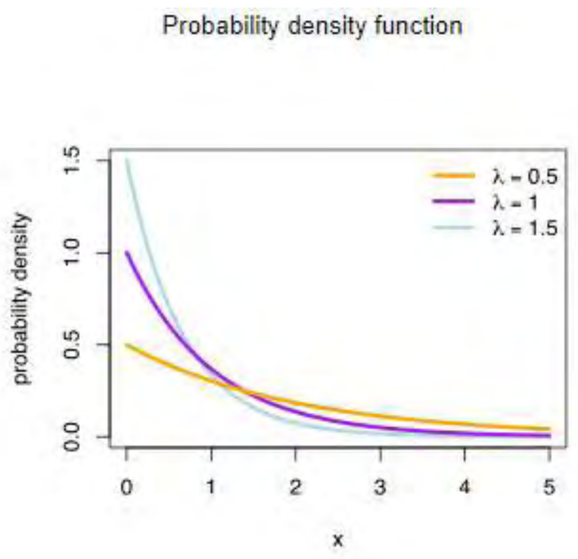


Diagram 6.2 The expontential pdf.

- Generalized Pareto distribution

The generalized Pareto distribution (GPD) is a part of continuous probability distributions. It is often used to model the tails of another distribution. It is defined by three parameters: location μ, scale σ, and shape ξ. Although sometimes it is specified by scale and shape, it could sometimes be described only by its shape parameter.
The generalized pareto distribution is given by the following function:

$$f(x;\mu,\sigma,\xi) = \frac{1}{\sigma}(1+\xi\frac{x-\mu}{\sigma})^{-(\frac{1}{\xi+1})}$$

Where $\mu, \xi \in R \; and \; \sigma \in (0,+\infty)$
The generalized pareto distribution is depicted in the following graph:

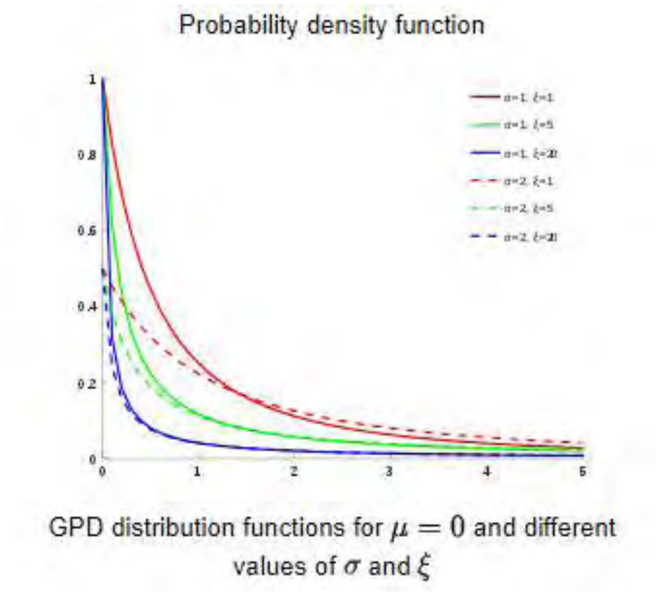


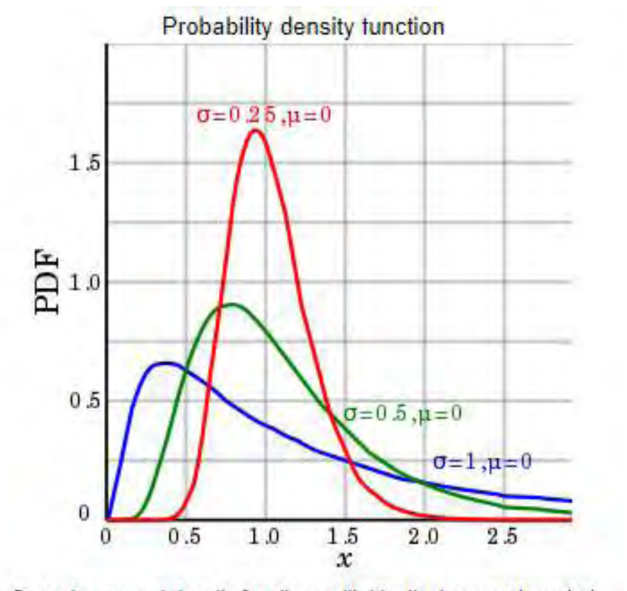
Diagram 6.3 The generalized pareto pdf.

- Log-normal distribution

The lognormal distribution is used in reliability analysis with its main application to be maintainability analysis of time to repair.
The log-normal distribution is given by the following function:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{(-\frac{(lnx-\mu)^2}{2\sigma^2})}$$

Where $\sigma > 0$ and μ ∈ R.
The log-normal distribution is depicted in the following graph:



Some log-normal density functions with identical parameter μ but differing parameters σ

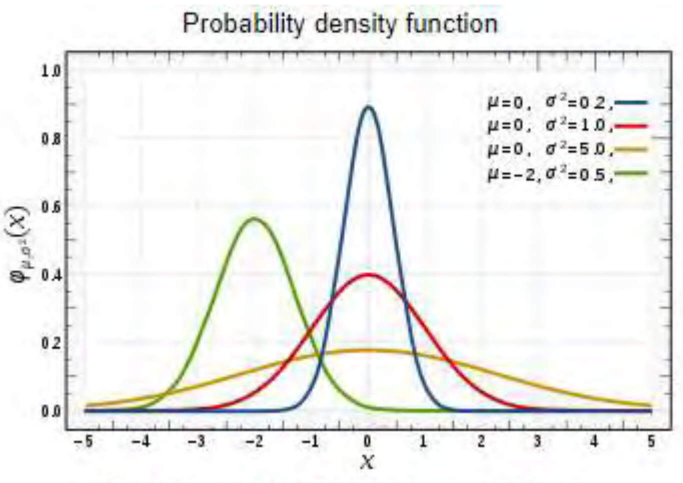
Diagram 6.4 Log-normal pdf.

- Normal distribution

In probability theory, the normal distribution is commonly used in the natural sciences to represent real valued random variables whose distribution is unknown.
The normal distribution is given by the following function:

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where $\mu \in (-\infty, +\infty)$, $x \in (-\infty, +\infty)$ and $\sigma^2 > 0$
The normal distribution is depicted in the following graph:



The red curve is the *standard normal distribution*

Diagram 6.5 Normal distribution.

We used Matlab software in order to observe which of the above distributions fits the data, based on maximum likelihood estimation. The Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) were used with the aim of choosing the best distribution model, by estimating the quality of each model compared to other models. From all the candidate distribution fitting models, the preferred model is the one which has the minimum AIC or BIC value. The equation that estimates the AIC value is given as follows: $AIC=2k-2ln\ (L^{\wedge})$, where $L^{\wedge}$ describes the maximum value of the likelihood function and k refers to the number of estimated parameters. In like manner, the BIC value is calculated by the equation: $BIC=(N)k-2ln\ (L^{\wedge})$. The AIC and BIC are not test models in the sense of testing a null hypothesis, as they do not provide any insight about the absolute quality of the model.

To find out which distribution describes the data the best, Kolmogorov-Smirnov goodness-of-fit statistical test was performed. This test introduces a significance p-value to examine the divergence between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution. The p-value must be larger than a significance level a, to accept the null hypothesis that the sample is drawn on the best distribution. The significance level used in the performed tests was α=0.05.

6.2 Relevant diagrams

In this section of our thesis, we are going to represent some helpful diagrams with the aim of raising reader’s understanding. These diagrams are comprised of the data’s distribution and several histograms for each machine. To be more precise, probability density function plots are going to be presented, regarding the discrete productive time and in cases of small data sample, these plots are replaced by histograms. To form the distribution plots we used the command ‘allfitdist’ in Matlab software which provides a number of possible fitting distributions along with the corresponding significant parameters (AIC, BIC etc.). Although this command produces many possible distribution fittings for every situation (machine), on each plot are depicted only the most probable ones. The following plots refer to the possible distribution fittings of the productive time values, at every situation. After examining the plots’ results we are able to decide which of the featured distributions are worthy of taking part to the Kolmogorov – Smirnov test.

Machine: **4BSAXF18**

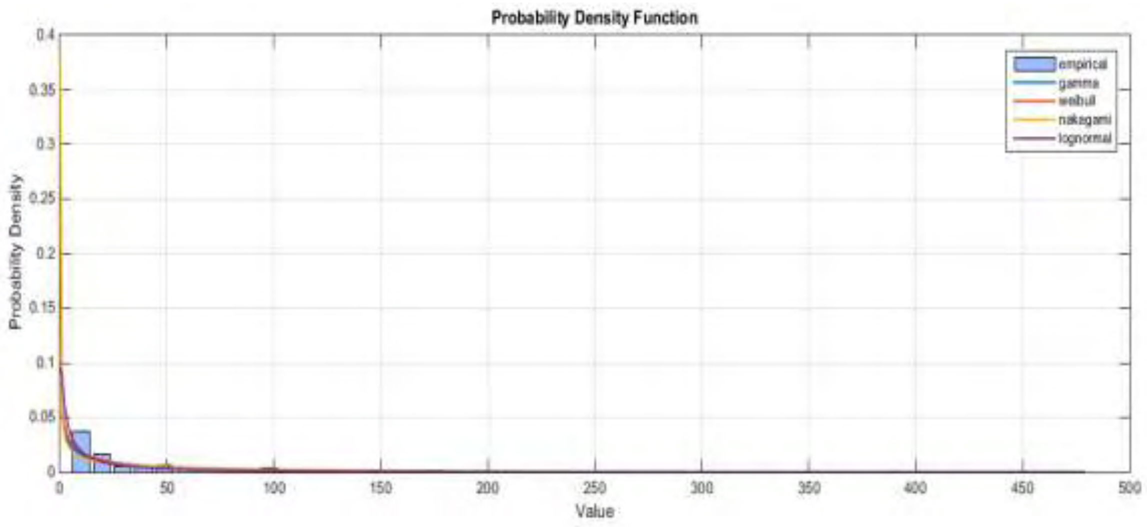


Diagram 6.6 Distribution fittings for the entire productive time data set.

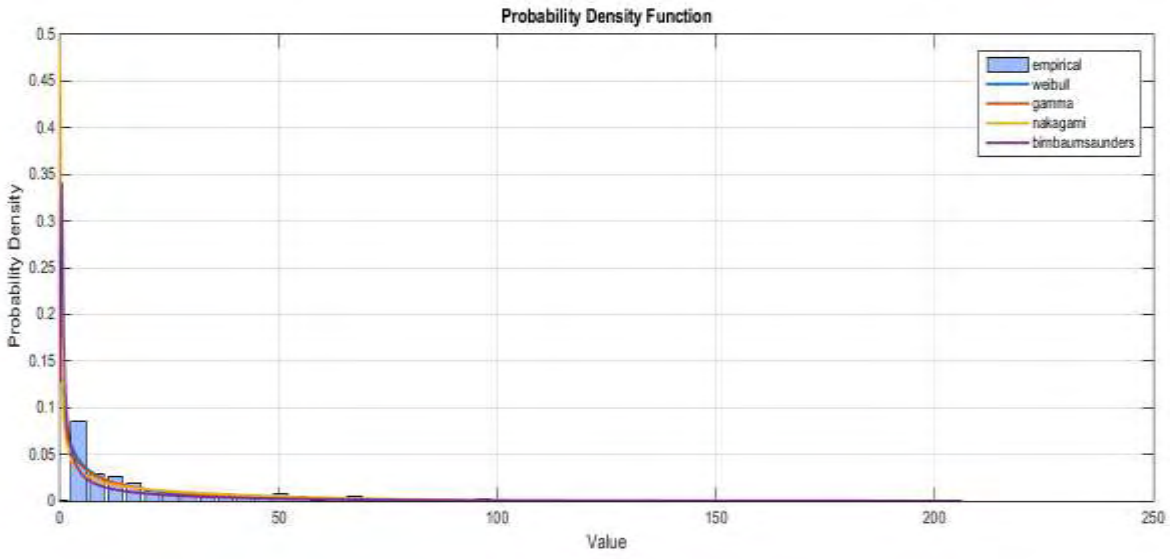
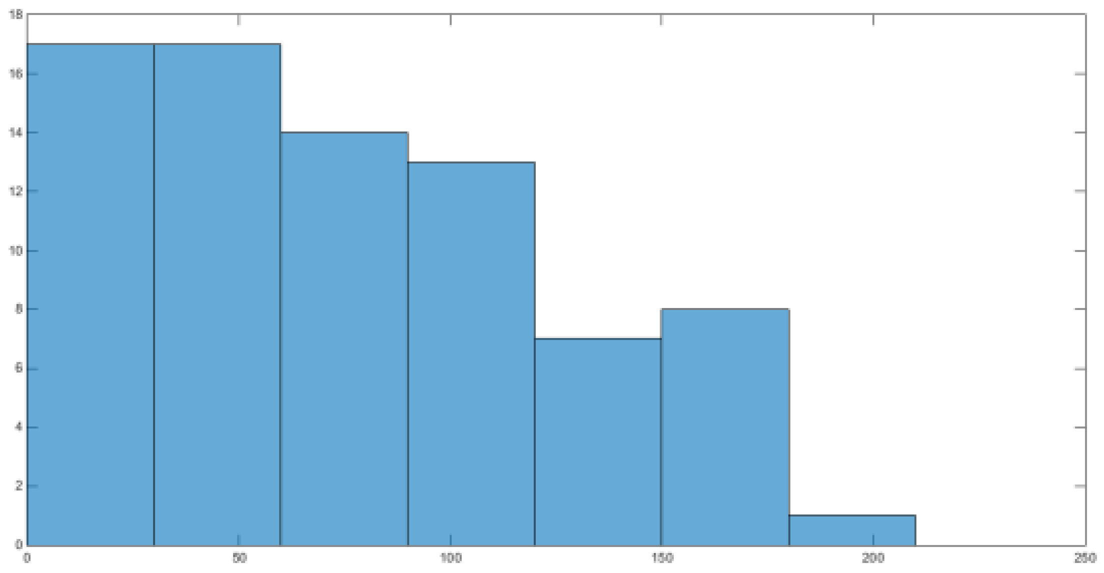
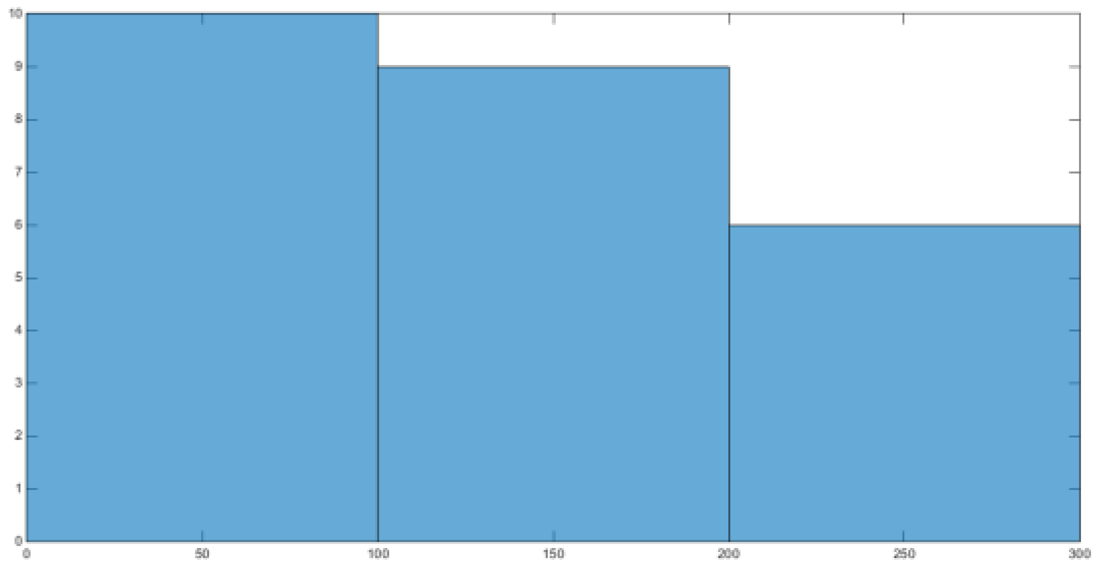


Diagram 6.7 Distribution fittings for the productive times that refer to zero maintenance.



Histogram 6.1 Productive time values when the number of maintenance is equal to 1.



Histogram 6.2 Productive time values when the number of maintenance is equal to 2.

Data set	Distribution	Mean value	Variance value
Entire productive times	Gamma	47.283	4 629.300
Productive times with zero maintenance	Weibull	18.798	706.771
Productive times with a single maintenance	-	74.340	2 353.200
Productive times with two maintenances	-	143.835	6 581.300

Table 6.1 Mean – var table.

Machine: **2BSAXF20**

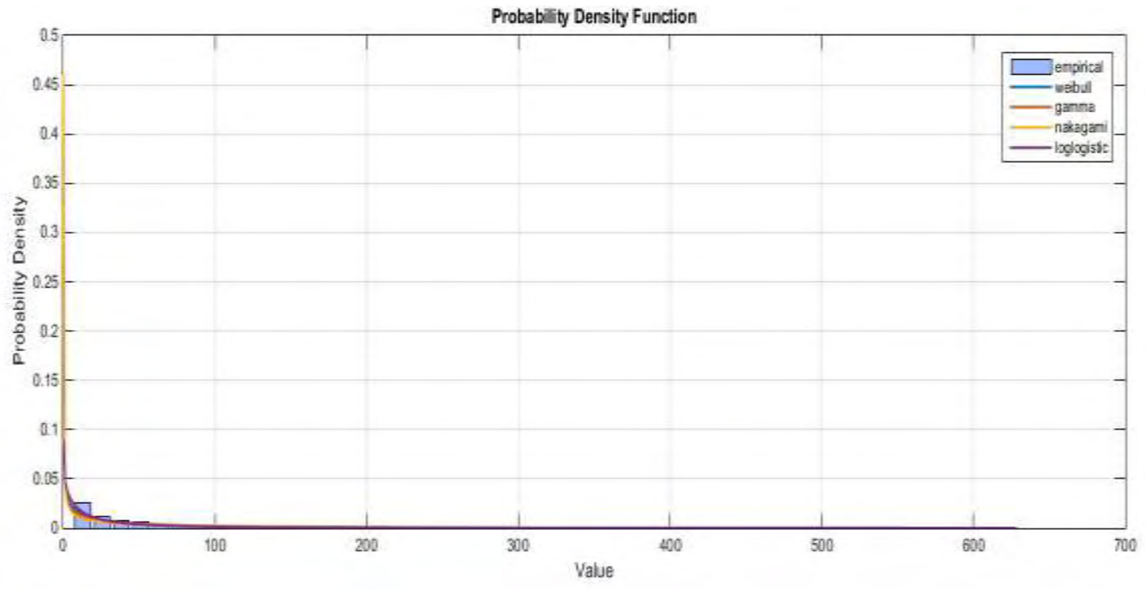


Diagram 6.8 Distribution fittings for the entire productive time data set.

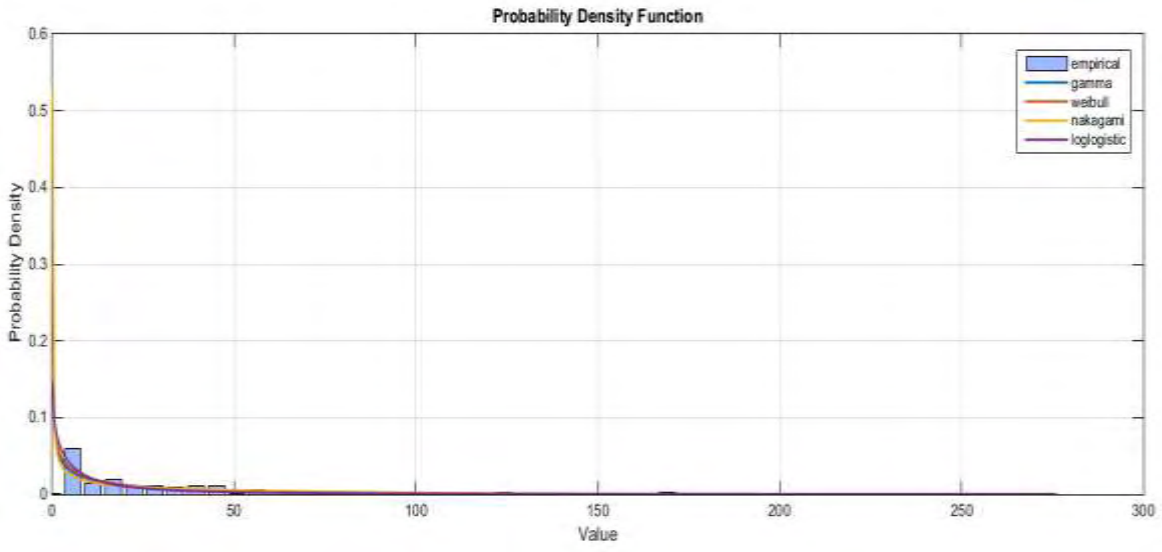
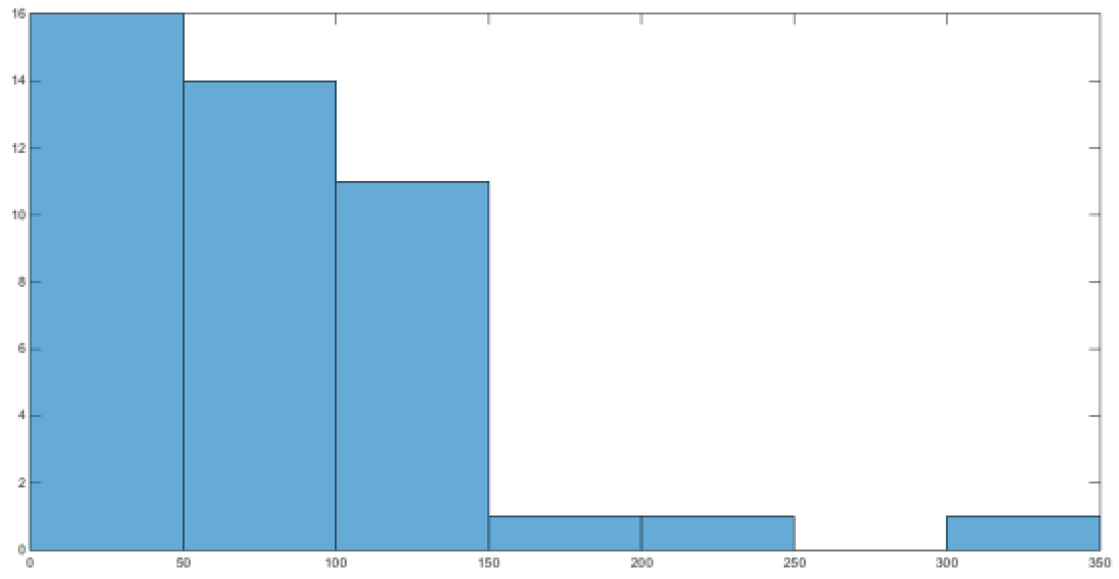
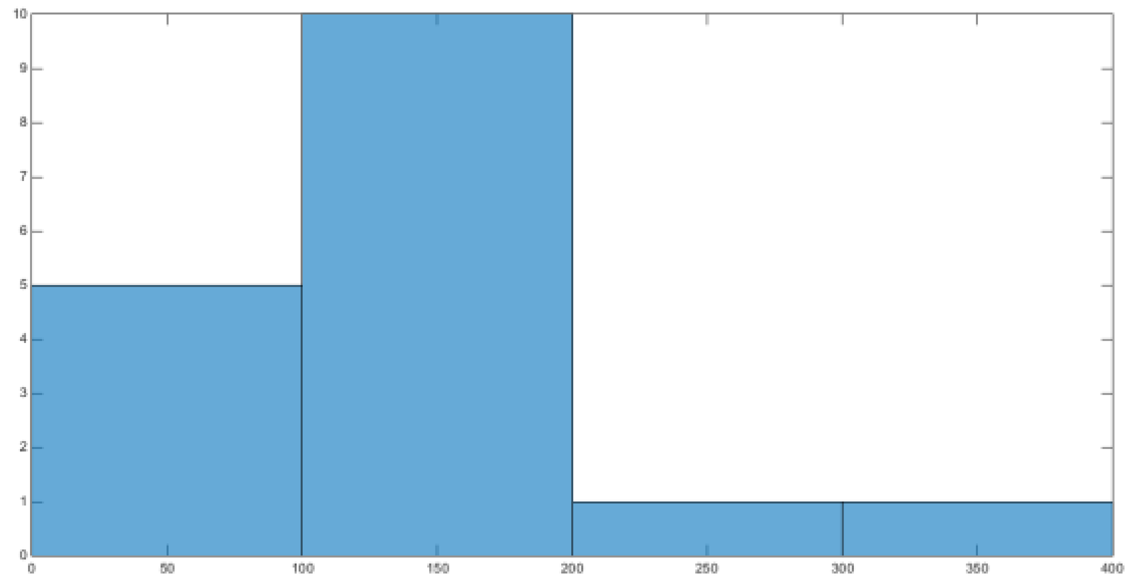


Diagram 6.9 Distribution fittings for the productive times that refer to zero maintenance.



Histogram 6.3 Productive time values when the number of maintenance is equal to 1.



Histogram 6.4 Productive time values when the number of maintenance is equal to 2.

Data set	Distribution	Mean value	Variance value
Entire productive times	Weibull	66.661	11 117
Productive times with zero maintenance	Gamma	29.564	1 772.5
Productive times with a single maintenance	-	80.180	3 972.5
Productive times with two maintenances	-	135.211	8 992.3

Table 6.2 Mean – var table.

<u>Machine: **NF671**</u>

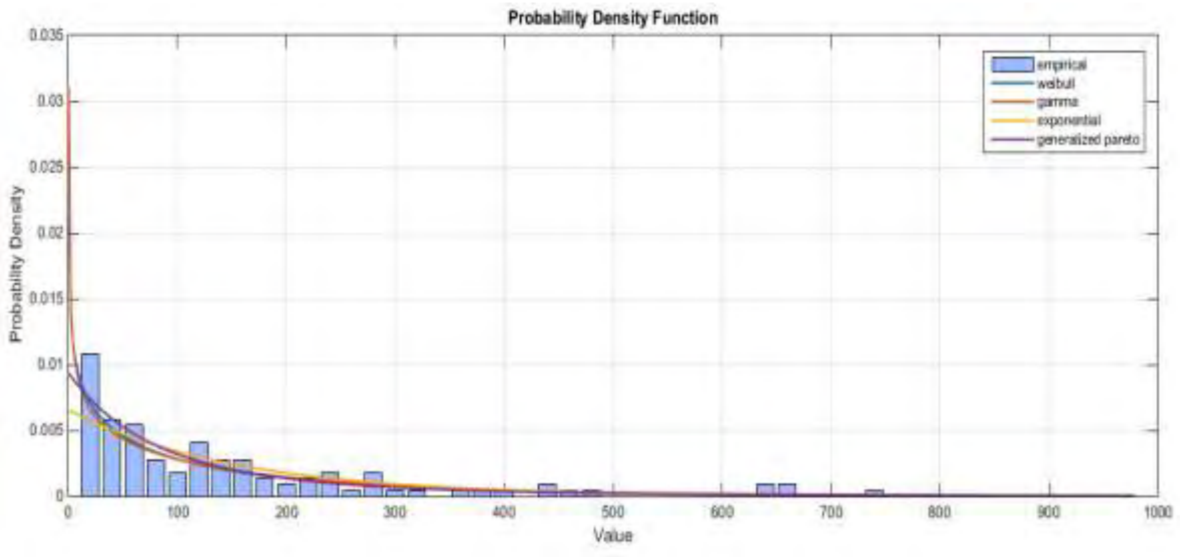


Diagram 6.10 Distribution fittings for the entire productive time data set.

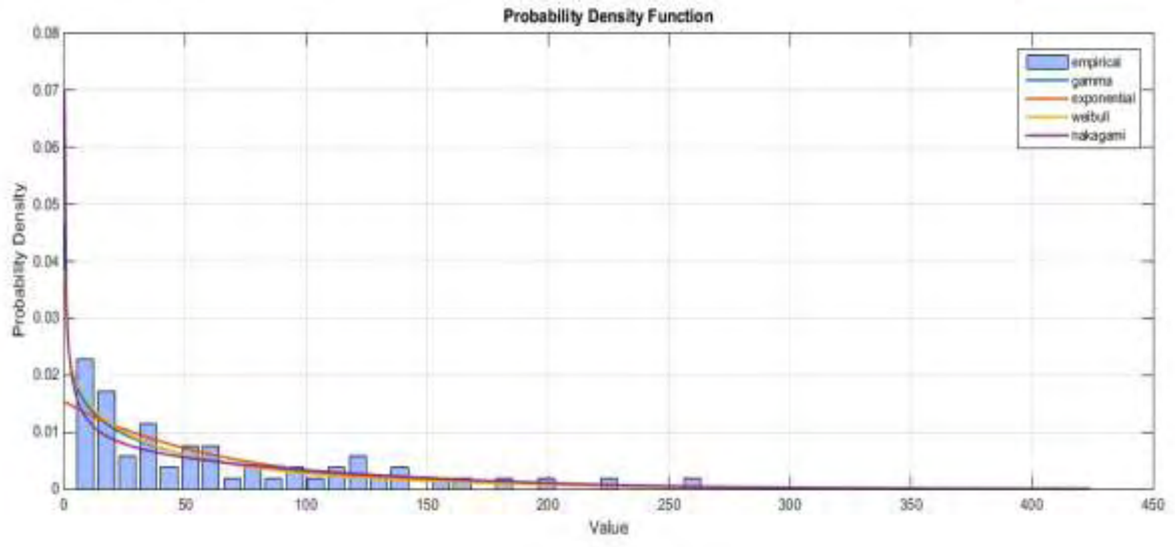
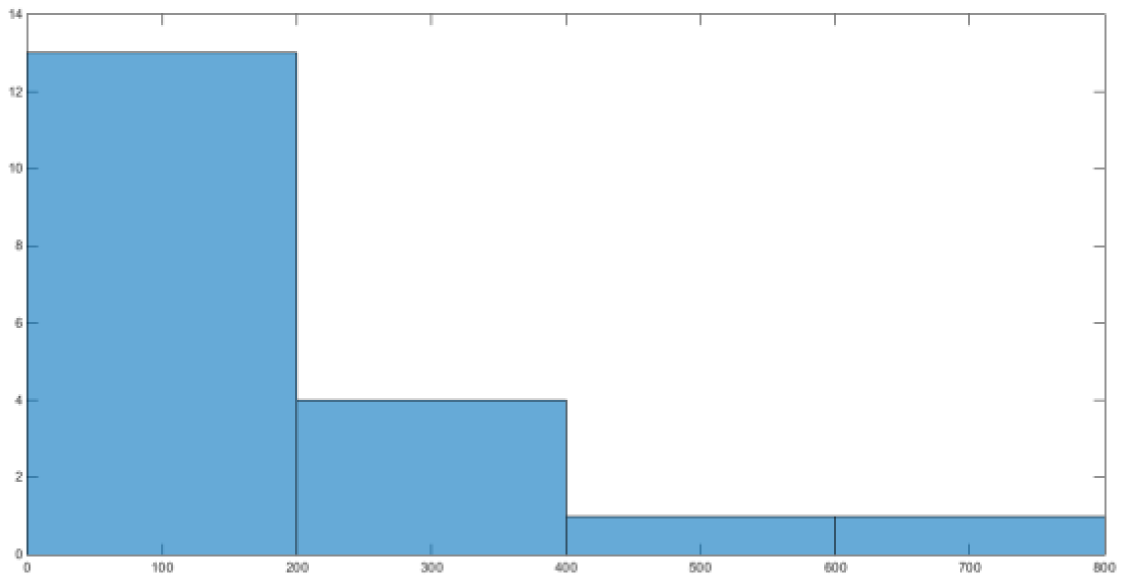
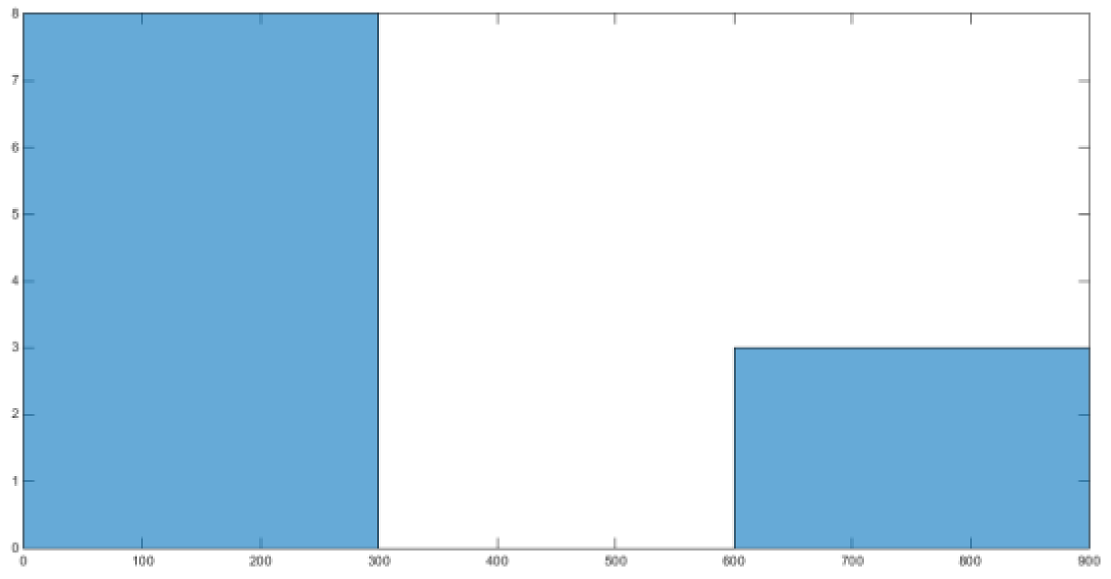


Diagram 6.11 Distribution fittings for the productive times that refer to zero maintenance.



Histogram 6.5 Productive time values when the number of maintenance is equal to 1.



Histogram 6.6 Productive time values when the number of maintenance is equal to 2.

Data set	Distribution	Mean value	Variance value
Entire productive times	Weibull	152.588	37 177
Productive times with zero maintenance	Gamma	65.202	5 732.3
Productive times with a single maintenance	-	176.347	27 586
Productive times with two maintenances	-	303.206	61 303

Table 6.3 Mean – var table.

Machine: **NM111**

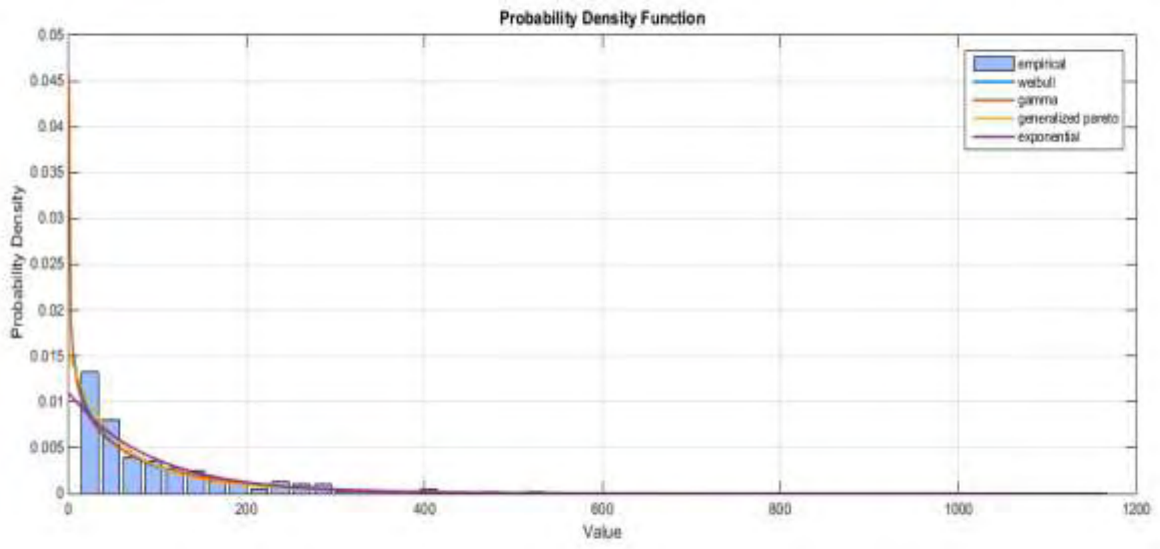


Diagram 6.12 Distribution fittings for the entire productive time data set.

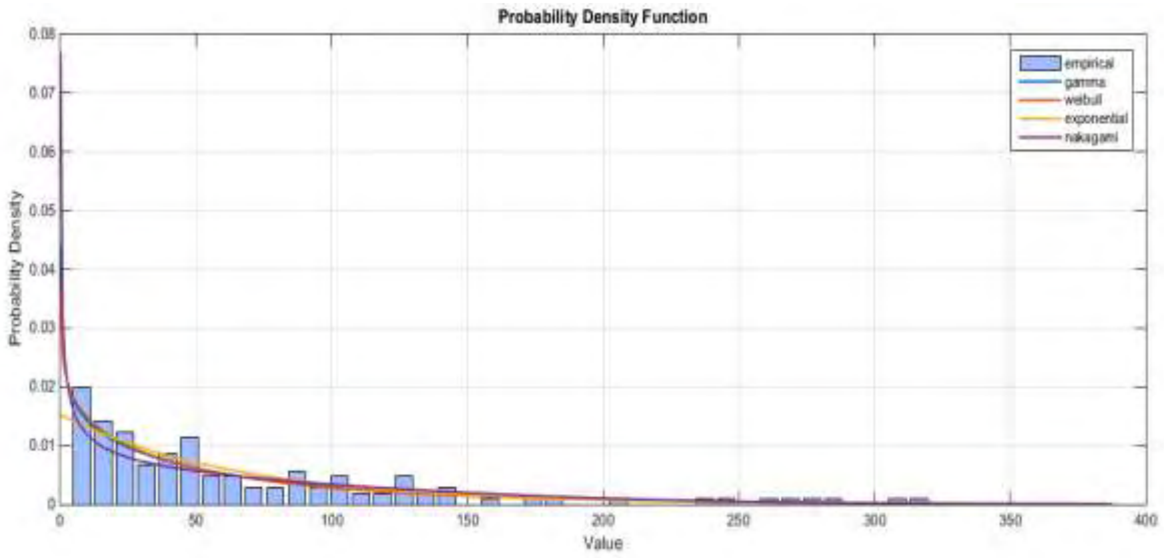
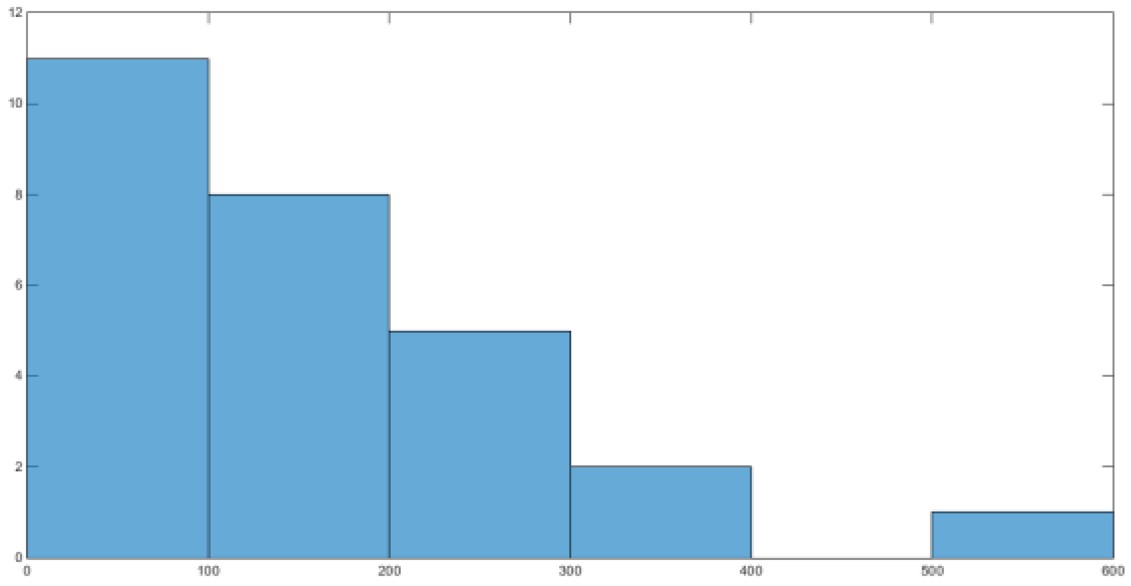
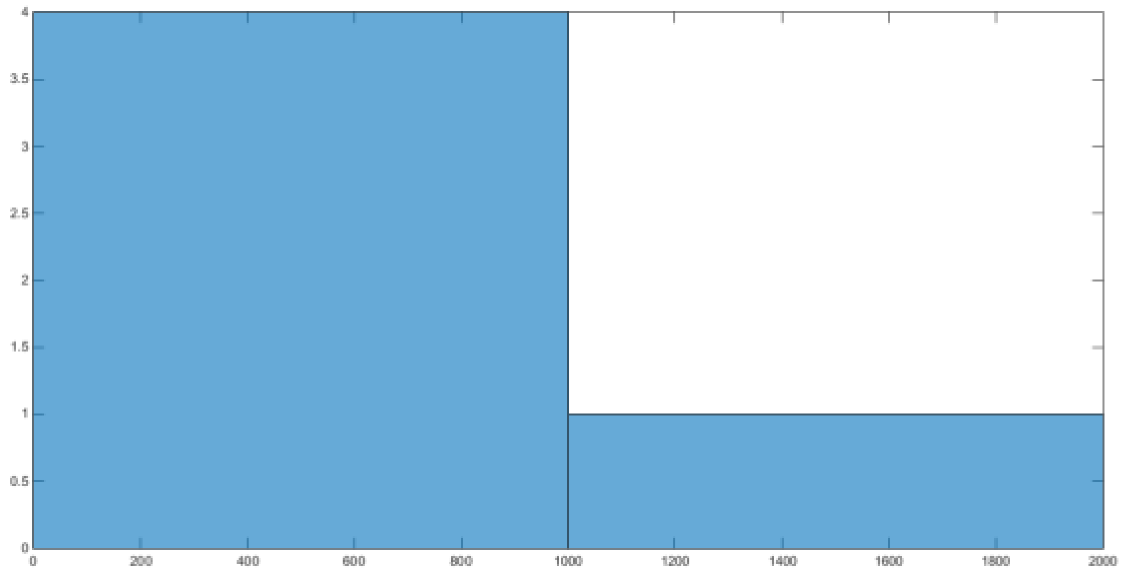


Diagram 6.13 Distribution fittings for the productive times that refer to zero maintenance.



Histogram 6.7 Productive time values when the number of maintenance is equal to 1.



Histogram 6.8 Productive time values when the number of maintenance is equal to 2.

Data set	Distribution	Mean value	Variance value
Entire productive times	Weibull	91.183	16 246
Productive times with zero maintenance	Gamma	65.620	5 528.700
Productive times with a single maintenance	-	153.941	15 873
Productive times with two maintenances	-	420.451	202 240

Table 6.4 Mean – var table.

7 Conclusions

The goal of this thesis was to generate neural network models, which could predict future machine failures by taking into account two input parameters (productive time and number of maintenance). Essentially, the models provide the user with the opportunity to decide the optimal time period in which the maintenance process should be performed.

The main idea, in which the models were developed, was to compute the cumulative number of faults according to the two input descriptors. However, the fundamental objective in manufacturing decision problems is to predict the proper time of maintenance in order to avoid machine's failure. This crucial question can be addressed by the interpretation of the productive time – rate of failure diagram (chapter 5.5) so to optimize the production chain planning.

Concerning the application of neural networks and their performance, it can be concluded that NN models are pretty reliable and trustworthy as the provided results are reasonable and accurate. Nevertheless, having a representative and a broad based data set, it's critical for the neural network's solid functionality.

Last but not least, some of the suggested future research plans could include the direct implementation of neural networks to production industries. For example, the real time production supervision through sensors which are connected in a wider network can be applied. In this way, this dynamic network could take efficient decisions regarding all the possible problematic situations. All this procedure is integrated into a further sector called internet of things, where the neural network science plays an instrumental role.

In the end, hundreds of industries could feasibly use neural networks not only to operate more efficiently, but also to target new audiences, develop new products or improve consumer's safety.

Bibliography

1. Nachimuthu Karunanithi, Darrell Whitley and Yashwant K. Malaiya, Colorado State University, ‘Using neural networks in reliability prediction’.
2. Anahid Jalali, Clemens Heistracher, Alexander Schindler, Bernhart Haslhofer, Austrian Institute of Technology, Tanja Nemeth, Robert Glawar, Wilfried Sihn, Fraunhofer Austria, Peter De Boer, Infineon Technologies Austria AG, ‘Predicting time to failure of plasma etching equipment using machine learning’.
3. Jorge Hurtado and Diego Alvares, Universidad National de Colombia, ‘Neural network based reliability analysis: a comparative study’.
4. Howard Demuth and Mark Beale, ‘Neural network toolbox for use with matlab’.
5. Primoz Potocnik, University of Ljubljana, ‘Neural networks: Matlab examples’.
6. Apostolos Stamatakis and Spyridon Botsis, University of Thessaly, ‘Reliability and maintainability analysis in semiconductor manufacturing’.
7. J. A. Anderson, ‘An Introduction to Neural Networks’, MIT Press, Cambridge (1995).
8. S. Haykin,’ Neural Networks: A Comprehensive Foundation’, Second Edition, Prentice Hall, Upper Saddle Point (1999).
9. R. Lippmann, ‘An Introduction to Computing with Neural Networks’, IEEE ASSP Magazine, 4-22 (1987).
10. M. McCord Nelson and W.T. Illingworth, ‘A practical guide to Neural Nets’, AddisonWesley (Reading, Mass), 1991.
11. N. K. Bose and P. Liang, ‘Neural Networks Foundamentals with Graphs, Algorithms and Applications’, McGraw-Hill, New York (1996).
12. Joarder Kamruzzaman, Monash University, Rezaul K. Begg, Victoria University, Ruhul A. Sarker, University of New South Wales, ‘Artificial neural networks in finance and manufacturing’.
13. Charu C. Aggarwal, IBM T.J. Watson Research Center ‘Data mining: the textbook’
14. Samuel H. Huang and Hong – Chao Zhang, ‘Artificial neural networks in manufacturing: Concepts, applications and perspectives’.
15. Joseph Rocca, ‘A gentle journey from linear regression to neural networks’, [Online]. Available:https://towardsdatascience.com/a-gentle-journey-from-linear-regression-to-neural-networks-68881590760e .
16. “Semiconductor,” [Online]. Available: https://en.wikipedia.org/wiki/Semiconductor.
17. L. Monch, J. W. Fowler and S. J. Mason, Production Planning and Control for Semiconductor Wafer Fabrication Facilities, vol. 52, S. Ramesh, Ed., Springer, 2013, pp. 11-13, 20-22.
18. T. Pomorski, “Major Revision Update for SEMI E10 Specification for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM),” [Online]. Available: http://www.semi.org/en/semi-e10-specification-equipment-reliability-availability-and-maintainability.
19. ‘List of probability distributions’, [Online]. Available: https://en.wikipedia.org/wiki/List_of_probability_distributions?fbclid=IwAR2Tpk1Yd1lEQfD9c08agsHLw-lQ9rl09bZmowqLOTOSF1GKDcdIIUEAb8k

20. Matthew Stewart, Phd researcher at Harvard University, ‘The actual difference between statistics and machine learning’.

Appendix

A.1 Matlab codes

In this section of thesis, it is going to be presented the majority of the Matlab codes used not only for the neural network models' creation but also for the proper exportation of the corresponding diagrams and calculations.

The following codes were used for the production of the neural networks and the prediction process.

Machine 4BSAXF18

```
clear all;
close all;
clc;
%--- Data insertion ---%
k=1;
m=0;
prod=0;
j=1;
[~,~,num]=xlsread('4B-PROC.xlsx');               % Data import from excel files
for i=1:51867
   if strcmp(num(i,3),'1AVAILABLE')==1;
      prod=prod+(cell2mat(num(i,6)))/3600.;     % Calculation of the cumulative
   end                                          % productive time (hours)
   if strcmp(num(i,3),'2MAINTENANCE')==1
      m=m+1;                          % Calculation of the cumulative number of maintenance
   end
   if strcmp(num(i,3),'3REPAIR')==1;
      faults(j)=j;                    % Calculation of the cumulative  number of faults
      productive(j)=prod;
      maintenance(j)=m;
      if j==1
         distprod(j)=productive(j);
         distmain(j)=maintenance(j);
      else                                          % Computing every discrete
         distprod(j)=productive(j)-productive(j-1);    % productive time and number of
         distmain(j)=maintenance(j)-maintenance(j-1);  % maintenance
      end
      if distprod(j)>0.02                       % We finally skip input values
         newprod(k)=productive(j);              % of discrete productive time
         newmain(k)=maintenance(j);             % that are less than 1.2 minutes
         newfaults(k)=k;
         if j==1
            newdistprod(k)=productive(j);
            newdistmain(k)=maintenance(j);
         else
```

```
                newdistprod(k)=productive(j)-productive(j-1);
                newdistmain(k)=maintenance(j)-maintenance(j-1);
            end
            k=k+1;
        end
        j=j+1;
    end
end
z=mean(newdistmain); % Mean value of number of maintenance
y=mean(newdistprod); % Mean value of productive time
x(1,:)=newprod;    % Cumulative productive time and cumulative number of maintenance
x(2,:)=newmain;    % are used as inputs to our neural network

%--- Definition of inputs,targets,hidden layer's neurons ---%
inputs=x;
targets=newfaults;     % The cumulative number of faults constitutes the output of NN
hiddenlayer1size=69;   % The network has one hidden layer with 69 neurons

%--- Creation of the neural network ---%
trainFcn = 'trainbr';      % Bayesian regulation training function is used in this case
net=fitnet(hiddenlayer1size,trainFcn );

%--- Deviation of samples ---%
 net.divideFcn = 'dividerand';       % Divide data randomly
 net.divideMode = 'sample';          % Divide up every sample
 net.divideParam.trainRatio = 70/100;   % 70 percent of samples are used for training
 net.divideParam.valRatio = 15/100;     % 15 percent of samples are used for validation
 net.divideParam.testRatio = 15/100;    % 15 percent of samples are used for testing

%--- Training procedure and results of the NN ---%
 [net,tr] = train(net,inputs,targets);       %Training session
 outputs = net(inputs);                      %Results of the NN
 errors = gsubtract(outputs,targets);        %Error values of NN
 performance = perform(net,targets,outputs);  %Mean square error value

%--- Mining final values of weights and biases ---%
 weights1=net.IW{1};  %The input to hidden layer weights
 weights2=net.LW{2};  %The hidden to output layer weights
 bias1=net.b{1};      %The input to hidden layer bias
 bias2=net.b{2};      %The hidden to output layer bias

%------- Prediction -------%
 prompt='Give the Cumulative Productive Time : ';
 o(1,1)=input(prompt);
 prompt='Give the Cumulative Number of Maintenance : ';
 o(2,1)=input(prompt);
 sample=net(o);
 if sample<0
  sample=0;
 end
```

```
predictions=sample
```

Machine 2BSAXF20

```
clear all;
close all;
clc;
%--- Data insertion ---%
k=1;
m=0;
prod=0;
j=1;
[~,~,num]=xlsread('2B.xlsx');             % Data import from excel files
for i=1:34333
   if strcmp(num(i,3),'1AVAILABLE')==1;
      prod=prod+(cell2mat(num(i,6)))/3600.;   % Calculation of the cumulative
   end                                        % productive time (hours)
   if strcmp(num(i,3),'2MAINTENANCE')==1
      m=m+1;                          % Calculation of the cumulative number of maintenance
   end
   if strcmp(num(i,3),'3REPAIR')==1;
      faults(j)=j;                    % Calculation of the cumulative  number of faults
      productive(j)=prod;
      maintenance(j)=m;
      if j==1
         distprod(j)=productive(j);
         distmain(j)=maintenance(j);
      else                                     % Computing every discrete
         distprod(j)=productive(j)-productive(j-1);    % productive time and number of
         distmain(j)=maintenance(j)-maintenance(j-1);  % maintenance
      end
      if  distprod(j)>0.02                     % We finally skip input values
         newprod(k)=productive(j);               % of discrete productive time
         newmain(k)=maintenance(j);               % that are less than 1.2 minutes
         newfaults(k)=k;
         if j==1
            newdistprod(k)=productive(j);
            newdistmain(k)=maintenance(j);
         else
            newdistprod(k)=productive(j)-productive(j-1);
            newdistmain(k)=maintenance(j)-maintenance(j-1);
         end
         k=k+1;
      end
      j=j+1;
   end
end
z=mean(newdistmain);  % Mean value of number of maintenance
```

```
y=mean(newdistprod);  % Mean value of productive time

% We start from the second data pair as the first cell contains a productive time value that
affects the network's proper functionality and accuracy

x(1,1:250)=newprod(2:251);    % Cumulative productive time and number of maintenance
x(2,1:250)=newmain(2:251);    % are used as inputs to our neural network

%--- Definition of inputs,targets,hidden layer's neurons ---%
inputs=x;
targets(1:250)=newfaults(2:251);     % The cumulative number of faults constitutes the output
hiddenlayer1size=30;   % The network has one hidden layer with 30 neurons

%--- Creation of the neural network ---%
trainFcn = 'trainbr';    % Bayesian regulation training function is used in this case
net=fitnet(hiddenlayer1size,trainFcn );

%--- Deviation of samples ---%
 net.divideFcn = 'dividerand';       % Divide data randomly
 net.divideMode = 'sample';          % Divide up every sample
 net.divideParam.trainRatio = 70/100; % 70 percent of samples are used for training
 net.divideParam.valRatio = 15/100;   % 15 percent of samples are used for validation
 net.divideParam.testRatio = 15/100;  % 15 percent of samples are used for testing

%--- Training procedure and results of the NN ---%
 [net,tr] = train(net,inputs,targets);      %Training session
 outputs = net(inputs);                     %Results of the NN
 errors = gsubtract(outputs,targets);       %Error values of NN
 performance = perform(net,targets,outputs); %Mean square error value

%--- Mining final values of weights and biases ---%
 weights1=net.IW{1};  %The input to hidden layer weights
 weights2=net.LW{2};  %The hidden to output layer weights
 bias1=net.b{1};      %The input to hidden layer bias
 bias2=net.b{2};      %The hidden to output layer bias

%------- Prediction -------%
 prompt='Give the Cumulative Productive Time : ';
 o(1,1)=input(prompt);
 prompt='Give the Cumulative Number of Maintenance : ';
 o(2,1)=input(prompt);
 sample=net(o);
 if sample<0
  sample=0;
 end
 predictions=sample
```

Machine NF671

```
clear all;
close all;
clc;
%--- Data insertion ---%
k=1;
m=0;
prod=0;
j=1;
[~,~,num]=xlsread('nf.671.xlsx');           % Data import from excel files
for i=1:11460
   if strcmp(num(i,3),'1AVAILABLE')==1;
      prod=prod+(cell2mat(num(i,6)))/3600.;   % Calculation of the cumulative
   end                                        % productive time (hours)
   if strcmp(num(i,3),'2MAINTENANCE')==1
      m=m+1;                          % Calculation of the cumulative number of maintenance
   end
   if strcmp(num(i,3),'3REPAIR')==1;
      faults(j)=j;                    % Calculation of the cumulative
      productive(j)=prod;                  % number of faults
      maintenance(j)=m;
      if j==1
         distprod(j)=productive(j);
         distmain(j)=maintenance(j);
      else                                          % Computing every discrete
         distprod(j)=productive(j)-productive(j-1);    % productive time and number of
         distmain(j)=maintenance(j)-maintenance(j-1);  % maintenance
      end
      if distprod(j)>0.02                      % We finally skip input values
         newprod(k)=productive(j);                 % of discrete productive time
         newmain(k)=maintenance(j);                 % that are less than 1.2 minutes
         newfaults(k)=k;
         if j==1
            newdistprod(k)=productive(j);
            newdistmain(k)=maintenance(j);
         else
            newdistprod(k)=productive(j)-productive(j-1);
            newdistmain(k)=maintenance(j)-maintenance(j-1);
         end
         k=k+1;
      end
      j=j+1;
   end
end
z=mean(newdistmain); % Mean value of number of maintenance
y=mean(newdistprod); % Mean value of productive time
x(1,:)=newprod;    % Cumulative productive time and cumulative number of maintenance
x(2,:)=newmain;    % are used as inputs to our neural network
```

```
%--- Definition of inputs,targets,hidden layer's neurons ---%
inputs=x;
targets=newfaults;     % The cumulative number of faults constitutes the output of NN
hiddenlayer1size=23;   % The network has one hidden layer with 23 neurons

%--- Creation of the neural network ---%
trainFcn = 'trainbr';  % Bayesian regulation training function is used in this case
net=fitnet(hiddenlayer1size,trainFcn );

%--- Deviation of samples ---%
 net.divideFcn = 'dividerand';       % Divide data randomly
 net.divideMode = 'sample';          % Divide up every sample
 net.divideParam.trainRatio = 70/100; % 70 percent of samples are used for training
 net.divideParam.valRatio = 15/100;   % 15 percent of samples are used for validation
 net.divideParam.testRatio = 15/100;  % 15 percent of samples are used for testing

%--- Training procedure and results of the NN ---%
 [net,tr] = train(net,inputs,targets);      %Training session
 outputs = net(inputs);                     %Results of the NN
 errors = gsubtract(outputs,targets);       %Error values of NN
 performance = perform(net,targets,outputs); %Mean square error value

%--- Mining final values of weights and biases ---%
 weights1=net.IW{1};  %The input to hidden layer weights
 weights2=net.LW{2};  %The hidden to output layer weights
 bias1=net.b{1};      %The input to hidden layer bias
 bias2=net.b{2};      %The hidden to output layer bias

%------- Prediction -------%
 prompt='Give the Cumulative Productive Time : ';
 o(1,1)=input(prompt);
 prompt='Give the Cumulative Number of Maintenance : ';
 o(2,1)=input(prompt);
 sample=net(o);
 if sample<0
  sample=0;
 end
 predictions=sample
```

Machine NM111

```
clear all;
close all;
clc;
%--- Data insertion ---%
k=1;
m=0;
prod=0;
j=1;
[~,~,num]=xlsread('nm.111.xlsx');              % Data import from excel files
for i=1:3959
    if strcmp(num(i,3),'1AVAILABLE')==1;
        prod=prod+(cell2mat(num(i,6)))/3600.;   % Calculation of the cumulative
    end                                         % productive time (hours)
    if strcmp(num(i,3),'2MAINTENANCE')==1
        m=m+1;                            % Calculation of the cumulative number of maintenance
    end
    if strcmp(num(i,3),'3REPAIR')==1;
        faults(j)=j;                      % Calculation of the cumulative
        productive(j)=prod;                   % number of faults
        maintenance(j)=m;
        if j==1
            distprod(j)=productive(j);
            distmain(j)=maintenance(j);
        else                                  % Computing every discrete
            distprod(j)=productive(j)-productive(j-1);    % productive time and number of
            distmain(j)=maintenance(j)-maintenance(j-1);  % maintenance
        end
        if distprod(j)>0.02                        % We finally skip input values
            newprod(k)=productive(j);                  % of discrete productive time
            newmain(k)=maintenance(j);                   % that are less than 1.2 minutes
            newfaults(k)=k;
            if j==1
                newdistprod(k)=productive(j);
                newdistmain(k)=maintenance(j);
            else
                newdistprod(k)=productive(j)-productive(j-1);
                newdistmain(k)=maintenance(j)-maintenance(j-1);
            end
            k=k+1;
        end
        j=j+1;
    end
end
z=mean(newdistmain); % Mean value of number of maintenance
y=mean(newdistprod); % Mean value of productive time
x(1,:)=newprod;    % Cumulative productive time and cumulative number of maintenance
x(2,:)=newmain;    % are used as inputs to our neural network
```

```
%--- Definition of inputs,targets,hidden layer's neurons ---%
inputs=x;
targets=newfaults;     % The cumulative number of faults constitutes the output of NN
hiddenlayer1size=15;   % The network has one hidden layer with 15 neurons

%--- Creation of the neural network ---%
trainFcn = 'trainbr';  % Bayesian regulation training function is used in this case
net=fitnet(hiddenlayer1size,trainFcn );

%--- Deviation of samples ---%
 net.divideFcn = 'dividerand';       % Divide data randomly
 net.divideMode = 'sample';          % Divide up every sample
 net.divideParam.trainRatio = 70/100; % 70 percent of samples are used for training
 net.divideParam.valRatio = 15/100;   % 15 percent of samples are used for validation
 net.divideParam.testRatio = 15/100;  % 15 percent of samples are used for testing

%--- Training procedure and results of the NN ---%
 [net,tr] = train(net,inputs,targets);       %Training session
 outputs = net(inputs);                      %Results of the NN
 errors = gsubtract(outputs,targets);        %Error values of NN
 performance = perform(net,targets,outputs); %Mean square error value

%--- Mining final values of weights and biases ---%
 weights1=net.IW{1};  %The input to hidden layer weights
 weights2=net.LW{2};  %The hidden to output layer weights
 bias1=net.b{1};      %The input to hidden layer bias
 bias2=net.b{2};      %The hidden to output layer bias

%------- Prediction -------%
 prompt='Give the Cumulative Productive Time : ';
 o(1,1)=input(prompt);
 prompt='Give the Cumulative Number of Maintenance : ';
 o(2,1)=input(prompt);
 sample=net(o);
 predictions=sample
```

The following code was used for the formation of the distribution diagrams.

```
function [D PD] = allfitdist(data,sortby,varargin)
%% Check Inputs
if nargin == 0
   data = 10.^((normrnd(2,10,1e4,1))/10);
   sortby='BIC';
   varargin={'CDF'};
end
if nargin==1
   sortby='BIC';
end
sortbyname={'NLogL','BIC','AIC','AICc'};
if ~any(ismember(lower(sortby),lower(sortbyname)))
   oldvar=sortby; %May be 'PDF' or 'CDF' or other commands
   if isempty(varargin)
      varargin={oldvar};
   else
      varargin=[oldvar varargin];
   end
   sortby='BIC';
end
if nargin < 2, sortby='BIC'; end
distname={'beta', 'birnbaumsaunders', 'exponential', ...
   'extreme value', 'gamma', 'generalized extreme value', ...
   'generalized pareto', 'inversegaussian', 'logistic', 'loglogistic', ...
   'lognormal', 'nakagami', 'normal', ...
   'rayleigh', 'rician', 'tlocationscale', 'weibull'};
if ~any(strcmpi(sortby,sortbyname))
   error('allfitdist:SortBy','Sorting must be either NLogL, BIC, AIC, or AICc');
end
%Input may be mixed of numeric and strings, find only strings
vin=varargin;
strs=find(cellfun(@(vs)ischar(vs),vin));
vin(strs)=lower(vin(strs));
%Next check to see if 'PDF' or 'CDF' is listed
numplots=sum(ismember(vin(strs),{'pdf' 'cdf'}));
if numplots>=2
   error('ALLFITDIST:PlotType','Either PDF or CDF must be given');
end
if numplots==1
   plotind=true; %plot indicator
   indxpdf=ismember(vin(strs),'pdf');
   plotpdf=any(indxpdf);
   indxcdf=ismember(vin(strs),'cdf');
   vin(strs(indxpdf|indxcdf))=[]; %Delete 'PDF' and 'CDF' in vin
else
   plotind=false;
end
```

```
%Check to see if discrete
strs=find(cellfun(@(vs)ischar(vs),vin));
indxdis=ismember(vin(strs),'discrete');
discind=false;
if any(indxdis)
    discind=true;
    distname={'binomial', 'negative binomial', 'poisson'};
    vin(strs(indxdis))=[]; %Delete 'DISCRETE' in vin
end
strs=find(cellfun(@(vs)ischar(vs),vin));
n=numel(data); %Number of data points
data = data(:);
D=[];
%Check for NaN's to delete
deldatanan=isnan(data);
%Check to see if frequency is given
indxf=ismember(vin(strs),'frequency');
if any(indxf)
    freq=vin{1+strs((indxf))}; freq=freq(:);
    if numel(freq)~=numel(data)
        error('ALLFITDIST:PlotType','Matrix dimensions must agree');
    end
    delfnan=isnan(freq);
    data(deldatanan|delfnan)=[]; freq(deldatanan|delfnan)=[];
    %Save back into vin
    vin{1+strs((indxf))}=freq;
else
    data(deldatanan)=[];
end

%% Run through all distributions in FITDIST function
warning('off','all'); %Turn off all future warnings
for indx=1:length(distname)
    try
        dname=distname{indx};
        switch dname
            case 'binomial'
                PD=fitbinocase(data,vin,strs); %Special case
            case 'generalized pareto'
                PD=fitgpcase(data,vin,strs); %Special case
            otherwise
                %Built-in distribution using FITDIST
                PD = fitdist(data,dname,vin{:});
        end

        NLL=PD.NLogL; % -Log(L)
        %If NLL is non-finite number, produce error to ignore distribution
        if ~isfinite(NLL)
            error('non-finite NLL');
        end
        num=length(D)+1;
        PDs(num) = {PD}; %#ok<*AGROW>
        k=numel(PD.Params); %Number of parameters
        D(num).DistName=PD.DistName;
        D(num).NLogL=NLL;
        D(num).BIC=-2*(-NLL)+k*log(n);
        D(num).AIC=-2*(-NLL)+2*k;
        D(num).AICc=(D(num).AIC)+((2*k*(k+1))/(n-k-1));
        D(num).ParamNames=PD.ParamNames;
        D(num).ParamDescription=PD.ParamDescription;
        D(num).Params=PD.Params;
        D(num).Paramci=PD.paramci;
        D(num).ParamCov=PD.ParamCov;
        D(num).Support=PD.Support;
    catch err %#ok<NASGU>
        %Ignore distribution
    end
end
warning('on','all'); %Turn back on warnings
if numel(D)==0
    error('ALLFITDIST:NoDist','No distributions were found');
end


%% Sort distributions
indx1=1:length(D); %Identity Map
sortbyindx=find(strcmpi(sortby,sortbyname));
switch sortbyindx
    case 1
        [~,indx1]=sort([D.NLogL]);
    case 2
        [~,indx1]=sort([D.BIC]);
    case 3
        [~,indx1]=sort([D.AIC]);
    case 4
        [~,indx1]=sort([D.AICc]);
end
%Sort
D=D(indx1); PD = PDs(indx1);
%% Plot if requested
if plotind;
    plotfigs(data,D,PD,vin,strs,plotpdf,discind)
end
end


function PD=fitbinocase(data,vin,strs)
%% Special Case for Binomial
% 'n' is estimated if not given
vinbino=vin;
```

```
%Check to see if 'n' is given
indxn=any(ismember(vin(strs),'n'));
%Check to see if 'frequency' is given
indxfreq=ismember(vin(strs),'frequency');
if ~indxn
   %Use Method of Moment estimator
   %E[x]=np, V[x]=np(1-p) -> nhat=E/(1-(V/E));
   if isempty(indxfreq)||~any(indxfreq)
      %Raw data
      mnx=mean(data);
      nhat=round(mnx/(1-(var(data)/mnx)));
   else
      %Frequency data
      freq=vin{1+strs(indxfreq)};
      m1=dot(data,freq)/sum(freq);
      m2=dot(data.^2,freq)/sum(freq);
      mnx=m1; vx=m2-(m1^2);
      nhat=round(mnx/(1-(vx/mnx)));
   end
   %If nhat is negative, use maximum value of data
   if nhat<=0, nhat=max(data(:)); end
   vinbino{end+1}='n'; vinbino{end+1}=nhat;
end
PD = fitdist(data,'binomial',vinbino{:});
end


function PD=fitgpcase(data,vin,strs)
%% Special Case for Generalized Pareto
% 'theta' is estimated if not given
vingp=vin;
%Check to see if 'theta' is given
indxtheta=any(ismember(vin(strs),'theta'));
if ~indxtheta
   %Use minimum value for theta, minus small part
   thetahat=min(data(:))-10*eps;
   vingp{end+1}='theta'; vingp{end+1}=thetahat;
end
PD = fitdist(data,'generalized pareto',vingp{:});
end


function plotfigs(data,D,PD,vin,strs,plotpdf,discind)
%Plot functionality for continuous case due to Jonathan Sullivan
%Modified by author for discrete case

%Maximum number of distributions to include
%max_num_dist=Inf;  %All valid distributions
max_num_dist=4;
```

```
%Check to see if frequency is given
indxf=ismember(vin(strs),'frequency');
if any(indxf)
    freq=vin{1+strs((indxf))};
end

figure

%% Probability Density / Mass Plot
if plotpdf
    if ~discind
        %Continuous Data
        nbins = max(min(length(data)./10,100),50);
        xi = linspace(min(data),max(data),nbins);
        dx = mean(diff(xi));
        xi2 = linspace(min(data),max(data),nbins*10)';
        fi = histc(data,xi-dx);
        fi = fi./sum(fi)./dx;
        inds = 1:min([max_num_dist,numel(PD)]);
        ys = cellfun(@(PD) pdf(PD,xi2),PD(inds),'UniformOutput',0);
        ys = cat(2,ys{:});
        bar(xi,fi,'FaceColor',[160 188 254]/255,'EdgeColor','k');
        hold on;
        plot(xi2,ys,'LineWidth',1.5)
        legend(['empirical',{D(inds).DistName}],'Location','NE')
        xlabel('Value');
        ylabel('Probability Density');
        title('Probability Density Function');
        grid on
    else
        %Discrete Data
        xi2=min(data):max(data);
        %xi2=unique(x)'; %If only want observed x-values to be shown
        indxf=ismember(vin(strs),'frequency');
        if any(indxf)
            fi=zeros(size(xi2));
            fi((ismember(xi2,data)))=freq; fi=fi'./sum(fi);
        else
            fi=histc(data,xi2); fi=fi./sum(fi);
        end
        inds = 1:min([max_num_dist,numel(PD)]);
        ys = cellfun(@(PD) pdf(PD,xi2),PD(inds),'UniformOutput',0);
        ys=cat(1,ys{:})';
        bar(xi2,[fi ys]);
        legend(['empirical',{D(inds).DistName}],'Location','NE')
        xlabel('Value');
        ylabel('Probability Mass');
        title('Probability Mass Function');
        grid on
    end
```

```
else

%Cumulative Distribution
    if ~discind
        %Continuous Data
        [fi xi] = ecdf(data);
        inds = 1:min([max_num_dist,numel(PD)]);
        ys = cellfun(@(PD) cdf(PD,xi),PD(inds),'UniformOutput',0);
        ys = cat(2,ys{:});
        if max(xi)/min(xi) > 1e4; lgx = true; else lgx = false; end
        subplot(2,1,1)
        if lgx
            semilogx(xi,fi,'k',xi,ys)
        else
            plot(xi,fi,'k',xi,ys)
        end
        legend(['empirical',{D(inds).DistName}],'Location','NE')
        xlabel('Value');
        ylabel('Cumulative Probability');
        title('Cumulative Distribution Function');
        grid on
        subplot(2,1,2)
        y = 1.1*bsxfun(@minus,ys,fi);
        if lgx
            semilogx(xi,bsxfun(@minus,ys,fi))
        else
            plot(xi,bsxfun(@minus,ys,fi))
        end
        ybnds = max(abs(y(:)));
        ax = axis;
        axis([ax(1:2) -ybnds ybnds]);
        legend({D(inds).DistName},'Location','NE')
        xlabel('Value');
        ylabel('Error');
        title('CDF Error');
        grid on
    else
        %Discrete Data
        indxf=ismember(vin(strs),'frequency');
        if any(indxf)
            [fi xi] = ecdf(data,'frequency',freq);
        else
            [fi xi] = ecdf(data);
        end
        %Check unique xi, combine fi
        [xi,ign,indx]=unique(xi); %#ok<ASGLU>
        fi=accumarray(indx,fi);
        inds = 1:min([max_num_dist,numel(PD)]);
        ys = cellfun(@(PD) cdf(PD,xi),PD(inds),'UniformOutput',0);
        ys=cat(2,ys{:});
        subplot(2,1,1)
        stairs(xi,[fi ys]);
        legend(['empirical',{D(inds).DistName}],'Location','NE')
        xlabel('Value');
        ylabel('Cumulative Probability');
        title('Cumulative Distribution Function');
        grid on
        subplot(2,1,2)
        y = 1.1*bsxfun(@minus,ys,fi);
        stairs(xi,bsxfun(@minus,ys,fi))
        ybnds = max(abs(y(:)));
        ax = axis;
        axis([ax(1:2) -ybnds ybnds]);
        legend({D(inds).DistName},'Location','NE')
        xlabel('Value');
        ylabel('Error');
        title('CDF Error');
        grid on
    end
end
end
```