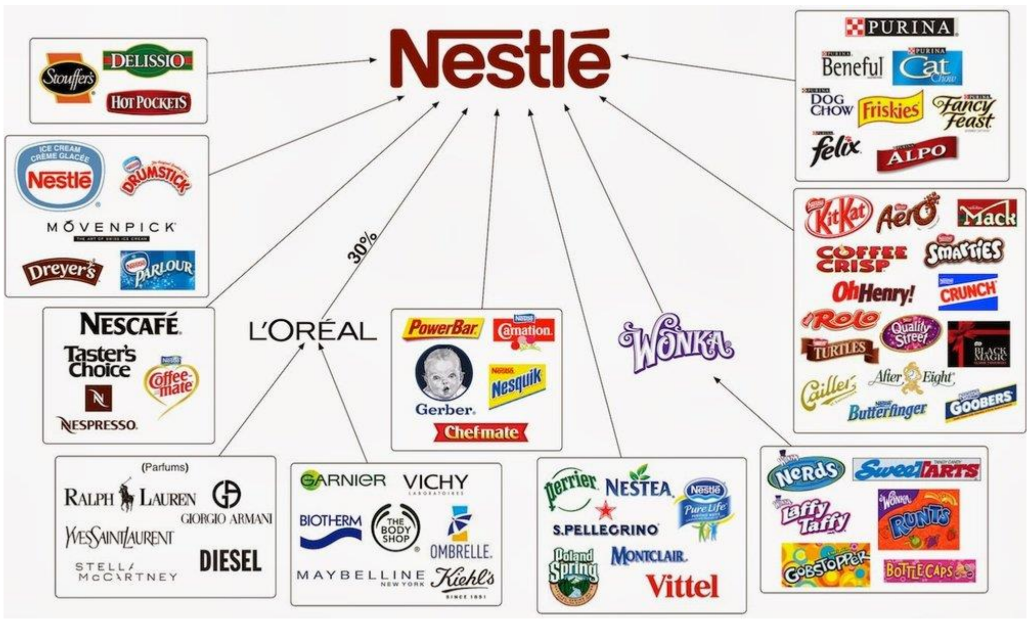
Predictive Maintenance Using Recurrent Neural Networks

Table of Contents:

Chapter 1: An introduction into Nestlé’s world

1- Introduction:

Nestle is one of the largest conglomerate corporations in the world. It specializes in the processing of foods and drinks. Some of their most sold beverages are Nescafe, Nespresso, Nesquik... And we find among their most famous instant foods: Neslac, Nestle and Cerelac. The company adopted the principle of diversification in the early years of the 20th century, and has currently over 2,000 brands with a wide range of products across a number of markets.



Nestle is best known for the high quality for its product. However, it shows strong political principles and acts even if it would be at risk of losing profit or even totally closing its factories. Not only does the company supply Ukraine with food during its recent conflict with Russia, but it also suspended shipment of non-essential items in Russia and makes sure to protect its employees on both sides of the war. In 2002, Nestle helped Ethiopia by lending it US$6 million before donating and reinvesting the debt repayment back into the country. In 2015, Nestle found out about the forced labor in Thai fishing industry, and has promised to impose new requirements on all potential suppliers, train boat owners and captains about human rights, and hire auditors to check for compliance with new rules.

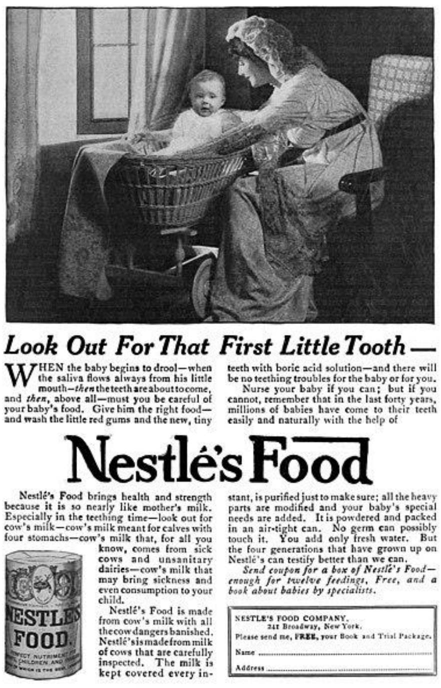
The company also sponsors many events in different countries around the world such as: *Walt Disney's EPCOT Center* from 1993 to 2009, the *Beijing Music Festival* from 2010 to 2013, *the Beijing Music Festival* in Austria from 1981 to 2015, *IAAF's Kids' Athletics Programme* from 2012 to 2016, in addition to sponsoring *the Great Britain Lionesses Women’s rugby league team* for the team's second tour of Australia with Nestlé’s Munchies product in 2002.

2- History:

Nestle was originally founded in the 1860's by the Swiss pharmacist Heinrish Nestle. In 1867, he developed milk-based baby food. Nestle then helped Daniel Peter in fixing a problem he faced with the removal of water from the milk he added to his chocolate. After the retirement of Henri in 1875, the company's new ownership decided to keep the name as "Société Farine Lactée Henri Nestle". During 1877 and 1878, 2 new products were added in the portfolio of Nestle's products: milk-based baby food and condensed milk. In the following year, Nestle merged with milk chocolate inventor Daniel Peter. Henri dies at the age of 75 in Montreux, Switzerland.

In the late 19th and early 20th century, Nestle's successors participated in the development of the chocolate industry in Switzerland, together with the Peter, Kohler, and Cailler families, before finally merging in 1929. An earlier alliance in 1904 between Peter and Nestlé also allowed the production of milk chocolate in the United States, at the Fulton plant. In 1947, another merger took place, Nestle's and Anglo-Swiss condensed milk company. The current name of the company was put in place in 1977. By the beginning of 20th century, it was already operating multiple factories in the United States, the United Kingdom, Germany, and Spain.

The First World War created demand for dairy products in the form of government contracts, and, by the end of the war, Nestlé's production had more than doubled. However, the Second World War had impacted negatively on the company. Profits dropped from US$20 million in 1938 to US$6 million in 1939. But luckily, the war helped in the introduction of a new product: Nescafe, that brought a promising customer for the company: the US military. The year 1945 was the beginning of a dynamic phase for Nestlé. Growth accelerated and numerous companies were acquired. It entered the era of diversification when Pierre Liotard-Vogt was its chairman and CEO by holding shares in L'Oreal in 1974 and by the acquisition of Alcon Laboratories Inc. in 1977 for $280 million.



Look Out For That First Little Tooth —

WHEN the baby begins to drool—when the saliva flows always from his little mouth—*then* the teeth are about to come, and *then*, above all—must you be careful of your baby's food. Give him the right food—and wash the little red gums and the new, tiny teeth with boric acid solution—and there will be no teething troubles for the baby or for you.

Nurse your baby if you can; but if you cannot, remember that in the last forty years, millions of babies have come to their teeth easily and naturally with the help of

Nestlé's Food

Nestlé's Food brings health and strength because it is so nearly like mother's milk. Especially in the teething time—look out for cow's milk—cow's milk meant for calves with four stomachs—cow's milk that, for all you know, comes from sick cows and unsanitary dairies—cow's milk that may bring sickness and even consumption to your child.

Nestlé's Food is made from cow's milk with all the cow dangers banished. Nestlé's is made from milk of cows that are carefully inspected. The milk is kept covered every instant, is purified just to make sure; all the heavy parts are modified and your baby's special needs are added. It is powdered and packed in an air-tight can. No germ can possibly touch it. You add only fresh water. But the four generations that have grown up on Nestlé's can testify better than we can.

Send coupon for a box of Nestlé's Food—enough for twelve feedings, Free, and a book about babies by specialists.

NESTLE'S FOOD COMPANY,
241 Broadway, New York.
Please send me, **FREE**, your Book and Trial Package.
Name ....
Address ....

This image is an advertisement for Nestlé's infant formula "Nestlé's Food" in 1905.

3- Nestle entering the Moroccan market :

Nestlé's first import operations in Morocco go back to the year 1927. The company's partnership in the kingdom is directly related to its purpose of the development of the global dairy industry while maintaining a good quality of life for everyone. Nestle has invested more than 420 million MAD in its local factories since 2010, proving jobs for more than 500 people as well as indirect employment for more than 7500 people. It works with 70 partner entities and distributes its products across 85,000 retail outlets, grocery shops, pharmacies, and the food service industry.

The company supports the idea of creating Shared Value in areas including nutrition and general health, rural development, local sourcing, environmental sustainability, education, and youth career development. It helps women fight literacy through their program in the region of Zagoura and envisions upgrading the quality of life of all woman and children in Morocco equally.

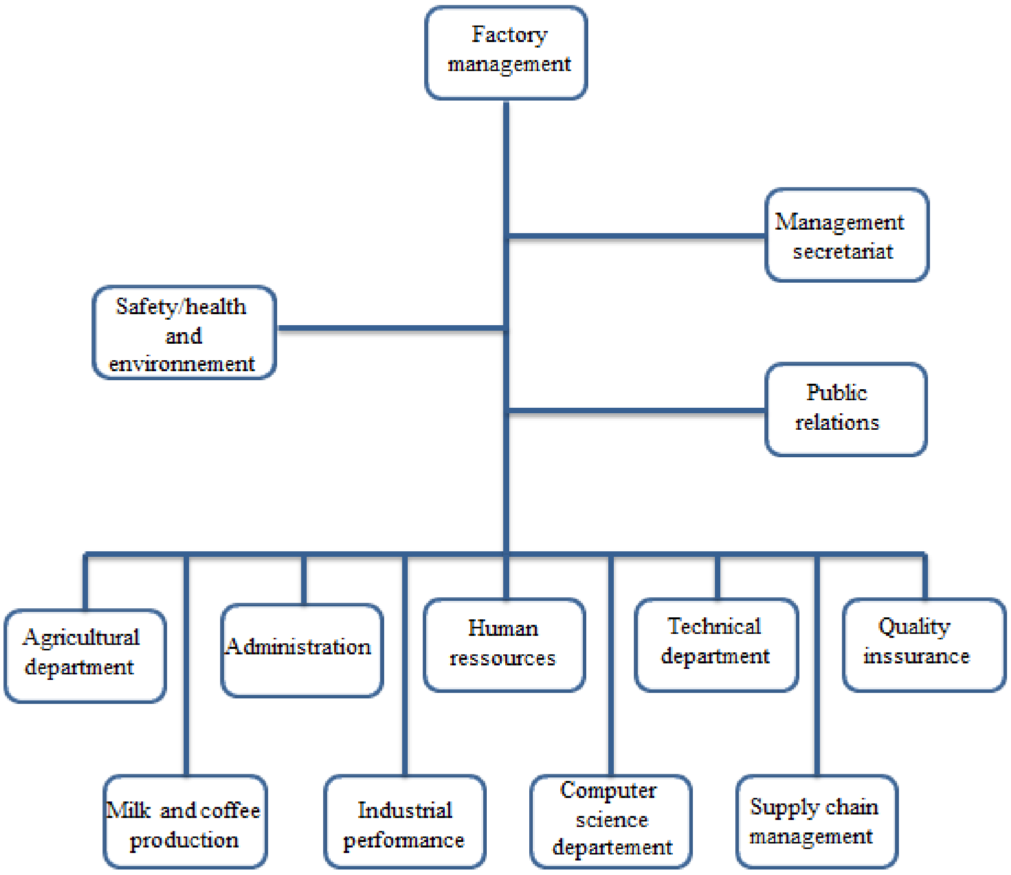
Among the most famous products that Nestle introduced to the Moroccan market we find: Nescafe, Nesquik, Chocapic, Cheerios, Gold Corn Flakes, Fitness, Nido, and Nespresso.

4- Nestlé's factory in El Jadida:

The El Jadida factory opened in 1992, with 3 strong principles: Delight consumers, Deliver competitive advantage and Excel in compliance. Nestlé sources more than 60,000 tons of raw milk per year directly from more than 6,000 small farms of various sizes in the Doukkala-Abda region, which currently produces 22% of the milk in Morocco, sourcing over 340 million liters a year. The "Model Village" milk collection farmer program helps farmers enhance safety, quality, quantity, water conservation, and many other environmental practices. The collaboration of Nestle with the local authorities aims to to increase milk production and improve the quality of fresh milk and encourage. The factory produces, as of last year, mainly two products: NIDO and Nescafe. They are then distributed in all regions of the kingdom, as well as being exported to the following Middle Eastern countries: Algeria, Tunisia, and Egypt.

It has its own solar station, which is the third solar plant of Nestle in the Middle East and North Africa (after Dubai and Jordan), and first in Morocco. The project uses the "fairly-light" approach and was brought into reality with the sponsoring of Nestlé's French partner Quadran with an estimated investment of 12 million MAD, in addition to a 1.3 million$ investment from renewable energy leader Qair. It brings together 2600 photovoltaic panels, which generates 1.7 GWh of electricity per year from a 7,000 m2 plot. The station was built close to Nestlé's local milk and coffee products factory. This will help in achieving a net zero goals by the year of 2030, as well as generating 52% of the Kingdom's electricity. Globally, Nestlé expects to complete the transition of its 800 sites in the 187 countries where it operates to 100 percent renewable electricity within the next five years.

5- Organizational chart of the factory :



Chapter 2: Production of milk and coffee

1- Department of milk:

1.1- Production of powdered milk :

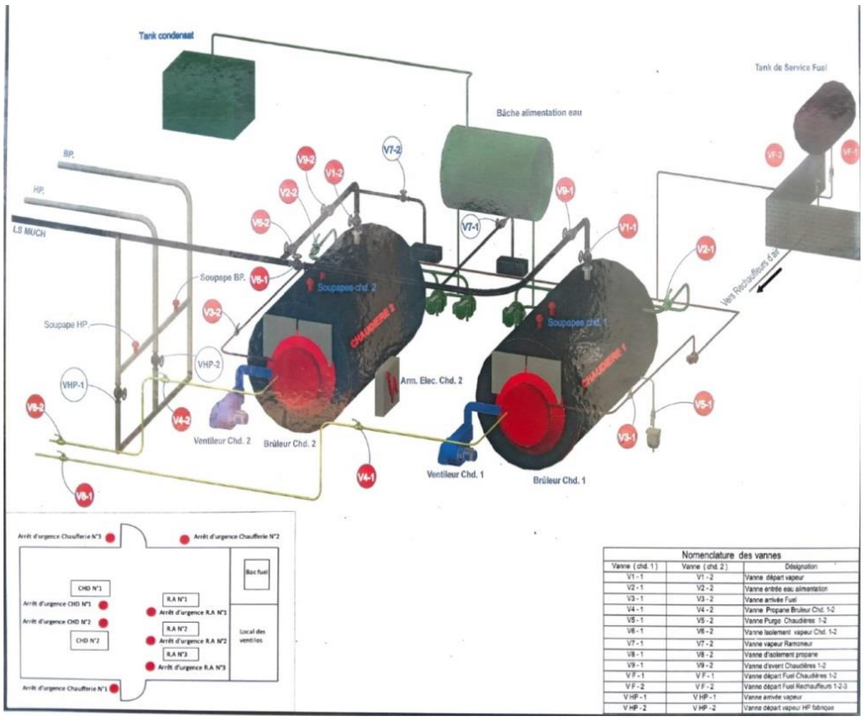
The milk comes first in cisterns, and then passes through a system of canalization to arrive inside the factory. The first and most important process in the production of powdered milk is the pasteurization of milk, which we will explain in details in the following paragraphs. The second process is the evaporation of the milk. Based on the percentage of water that is initially in the milk, the operator sets the machines on certain parameters, and sometimes to better the quality of the product, ,melted butter is added as well as some vitamins (especially when “NIDO plus” is being produced, because it is targeted to be fed to children in the first place) . Finally, the powered that will give us the product “NIDO” passes through a big sieve and is ready to be stocked into big tanks.

- **Process of pasteurization:**

This process relies on two essential operations, which only act on the temperature of the milk in order to not allow any bacteria to grow.

- Elevating the temperature of milk:

The factory of Nestlé El Jadida has two large boilers used to elevate to temperature of water. They start working when there is a detection of propane gas, through an electrode. We always need a natural gas as a heating source for the machine. Its existence has an only purpose of making the fuel circulate and so powering the mechanism. There are other choices for the energy that will be consumed by the system, such as gas and diesel, but fuel is the cheapest one of them. It is at first burned and the vapor resulting from the combustion passes through the many tubes composing the boiler. As the machine holds water into it, with its level arriving until 50% of all the height the heated tubes, its temperature keeps arising until it reaches its evaporating point. The output of the boiler is then water steam, which goes all the way through tubes to the department where the powdered milk is produced. The heated vapor is used to elevate the milk’s temperature to realize the first step of pasteurization process.



Nomenclature des vannes		
Vanne (chd. 1)	Vanne (chd. 2)	Désignation
V1 - 1	V1 - 2	Vanne départ vapeur
V2 - 1	V2 - 2	Vanne entrée eau alimentation
V3 - 1	V3 - 2	Vanne arrivée Fuel
V4 - 1	V4 - 2	Vanne Propane Bruleur Chd. 1-2
V5 - 1	V5 - 2	Vanne Purge Chaudières 1-2
V6 - 1	V6 - 2	Vanne Isolement vapeur Chd. 1-2
V7 - 1	V7 - 2	Vanne vapeur Ramoneur
V8 - 1	V8 - 2	Vanne d'isolement propane
V9 - 1	V9 - 2	Vanne d'event Chaudières 1-2
V F - 1	V F - 1	Vanne départ Fuel Chaudières 1-2
V F - 2	V F - 2	Vanne départ Fuel Rechauffeurs 1-2-3
V HP - 1	V HP - 1	Vanne arrivée vapeur
V HP - 2	V HP - 2	Vanne départ vapeur HP fabrique

Outline of the boilers at Nestlé's factory

- Cooling down the milk:

After elevating the temperature of milk, we have to cool it down in order for the bacteria to die. The reason behind this is enzymes continue to the cooling treatment and still unable to function in the process of helping bacteria to grow. We repeat the two operations (elevating and reducing the temperature) until we reach full pasteurization. The process of bringing the milk to a temperature of 20°C depends entirely on the presence of NH_3 gas. Choosing this gas among all other gazes relies on the fact that it has a small evaporating temperature. The factory has a closed circuit where Ammoniac gas circulates; it is called the Ice Builder. As a first step, the gas is sent to a compressor where both its temperature and pressure are modified and its physical state is now liquid. Then, it enters a tensioner that changes its state from liquid to vapor. The third step requires pumping the gas into vertical canals; the pump's pressure is measured using 4 barometers. The fourth step includes 5 evaporators where the NH_3 is in indirect contact with the water that was sent to the evaporating rooms using 2 distribution pumps. Finally, the gas should be sent back to the compressor through a tank that, for protection purposes, cannot contain NH_3 to a level exceeding 50% of its volume. The ice water goes afterwards in a tube that will be in wrapped up inside a bigger canal containing milk. Its temperature at the end of the process approaches 2°C.

1.2- Storing the powdered milk :

There are 3 lines for the storage of powdered milk into the metallic jars: Albro (for the 2500/2400g jars), Valpak (for the 400/50g jars) and Wolf (for the 60g packages). This department has multiple tanks with a capacity of 730kg called "Totbines" where the milk starts its journey. As a first step, an operator withdraws a sample from Totbine for quality tests and analysis in the factory's laboratory, once the tests come back as positive the tank can go on to the following step. There is a sieve with a 2mm diameter that makes it possible to remove all unwanted bodies. The powdered milk then passes to another big tank, a "Silo" this time. It goes through an endless screw to change its trajectory; this screw isn't launched until a level detector informs it there is no powder in the tank below it. There is another sensor for detecting metal and separating it from the powder using a check valve that throws it away. One of the operators has to always check the temperature as well as humidity of the area where the whole operation of storing takes place. The vertical Silo puts precise quantities of powdered milk into the plastic bags (for the quantity of 60g only) and the metallic jars. In order to eliminate any surpassing weights, a “Garvens Check-weighting” system is used. The factory contains a machine for the removal of all air from the bags/jars (in the same time as putting in the powder), using a vacuum pump, and a second machine to inject carbon dioxide CO2 and Nitrogen N2 into the packages. This operation prevents the bacteria in the milk from growing and so keeps the same quality of product for a long time. The last step of this process is the control of sealed packages through a tightness measurement with a pressure equal to 0.6 bars. And this is how the final product “NIDO” is ready to be exported to other countries and distributed in the Moroccan kingdom.

2- Department of coffee:

2.1- Production of Nescafe:

Big bags of green coffee are imported from different countries to El Jadida’s factory, where they get emptied and the coffee is cleaned from dust. The process of getting the industrial product ‘Nescafé’ from the raw coffee can be explained in 8 steps:

1) The green coffee beans are stocked in 4 big containers called Silo’s. A balance permits only to 300 kg of beans to enter the thermal oven, and depending on the color wanted on the final product; the operators set the temperature at around 187°C. Water in injected into the machine for purposes of cooling down the beans and elevating their humidity.

2) The final product which consists of clean, roast coffee beans is stored in 4 different large Silo’s. Its mass is now equal 258 kg instead of 300, because of the waste and pebbles that were separated from the beans, using a machine known as a “Destoner”.

3) The beans are put into a mill and come out as powder. Then, the operator performs a color test named “CTN” on a sample of the powdered roast coffee, to make sure that the quality required by the client is valid.

4) A portion of the roast beans goes to a new process called “VAX Mixer process”. At first, hot water vapor is injected into the coffee container in order to separate it from the aromas. A compressor turns the vapor aromas into liquids. They are stored at a temperature of 33°C and a pressure of 5 bars. The remaining beans get exported to Egypt.

5) For the coffee beans that do not get exported anywhere, they get distributed in 8 Silo's that work in cycles (with a mas of 180 Kg in total). The operators add to them percentages of liquid aromas; depending on the taste of the roast coffee and on the customer's wishes. Water is also added to get a light extract with a concentration rate of coffee equal to 12%. Its mass is now equal to 1030 Kg.

6) This product, which is liquid, is distributed in 3 cubic containers. Then it is sent through canals to a machine called the "Clarifier" to remove any unwanted bodies from the extract. At the end of the process, it is stored in one big tank.

7) Another process takes place in El Jadida's factory; the "VAX DGI process". The principal behind it is: lowering the pressure in the extract containers and as a result only needing a low temperature for the liquid vapor to evaporate. The conditions under which the concentration of the extract happens are T°<<100°C and P=25 mbar < 1000 mbar (1 bar = the atmospheric pressure). The final rate of concentration of the coffee in the liquid extract is 46%.

8) Drying the extract if the final step towards getting the industrial product "Nescafé". The process is called "Extract ELJ". Operators inject CO2 gas into the extract containers to unify its color, and then add hot air for the drying part. The factory has 2 big sieves for 2 different qualities of the product: agglomerate coffee and "Vile" for the powdered product.

2.2- Storing Nescafé:

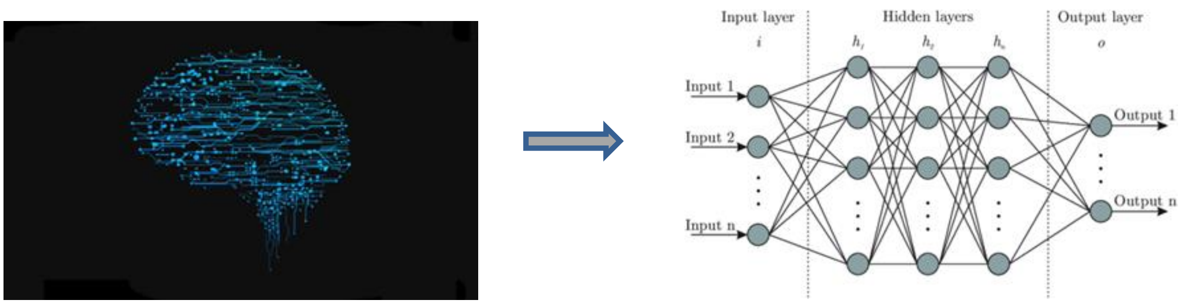
The jars used for the storage of the powdered coffee are brought from one of the collaborators of Nestlé; the Sévam factory. They arrive in big pallets that get put into a machine called "EMMETI". This machine is responsible for removing the plastic from the pallets and ejects the jars into the production line. There are numerous different sensors that make sure the dimensions of the jar are exactly as the ones required by the Nestlé factory. Then, they go through a visual inspection with an optical zoom lens; in case one of them is broken or represents a failure within any tolerance zone, it is immediately removed by the operator. Next step is cleaning the bottles using air pressure in order to remove any foreign body. The OPTIMAL machine, with a capacity to hold 30 glass bottles at once, fills them in with the powdered coffee, before closing them using pressure and a high temperature. Nestlé's factory uses, for quality measurements, a metal detector and an X-Ray machine to check if any unwanted body is present in the bottle. Then the labels for the industrial brand are glued to the jars using a special Superglue ENERCON, with the date of that day printed on them with a laser. The final step is gathering the bottles in a series of 24; they are put in storage pallets and wrapped in plastic to be ready for shipments.

This line of production is called the "Optimal line", the other containers of Nescafe require changing some of the machines and so have different lines of production: "SCMUCKER", "OCTABIN" and "Rovema and Mateer-Burt".

Chapter 3: Introduction to Neural Networks

1- Definition of a neural network:

As the human brain works by sending signals from one neuron to another, neural networks (or artificial/ simulated neural networks) use this exact principle for many applications in both the machine learning and deep learning fields. They contain an input and output layers, in addition to hidden/processing layers that vary depending on the application of the NN. The inputs may be weighted based on various criteria. We call artificial neurons, or nods, the components of each layer, and each connection, like the synapses in a biological brain, can transmit a signal to other neurons.



The reason behind introducing NN to artificial intelligence goes back to their ability to be trained. In supervised learning, the result is well known so the NN only has to work on the input to find patterns in data and generate a result close to ours. We call the difference between the result we want (target output) and the one we got (processed output): the error. NNs process the data communicated to them and based on the weights adjusted and associated to each input, the accuracy of the model is modified. After a certain number of adjustments, the training can be considered as completed if the processed output approached the target output within the wanted limit. On the other hand, in unsupervised learning, we can only look for patterns and structures within the data instead of finding pattern recognition from it. It deals with unlabeled data, which does not provide any real values of the output we are trying to predict.

2- Types of neural networks and their applications:

Neural networks have gained popularity in many fields, whether its image recognition, auto translation or the development of self-driving cars. We can define 3 types of NNs whose properties vary from one another:

- Convolutional Neural Network (CNN): They are regularized versions of multilayer perceptrons, which means each neuron in each layer is connected to all neurons in the next layer. The network is fully connected. The size of the input and the resulting output are fixed, and they are mainly composed from 3 types of building blocks: Convolutional layer, Pooling layer and Fully-connected layer. They are distinguished from other NNs by their little pre-processing, the optimization of kernels (filters) and their high performance in image and speech recognition.

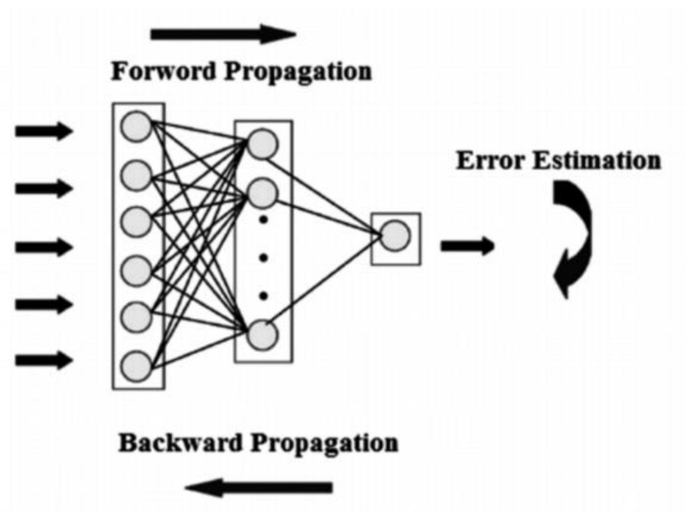
- Recurrent Neural Network (RNN): They are a type of conventional feed forward NNs whose algorithm deals best with sequential data or time series data. They use supervised deep learning combined with their property of 'memory'; where the output from the previous step is fed as input to the current step. The input layer processes the input data and passes it to the middle layer. This layer contains a loop where information cycles. There are variant RNN architectures: Bidirectional Recurrent Neural Networks, Gated Recurrent Units and Long Short Term Memory.
- Deep Neural Networks (DNN): They also use the method of feed forward propagation for machine learning. They contain multiple hidden layers between the input and output layers (that are independent from one another). They learn to model complex non-linear relationships (mostly high-level functions) based on the given data, and give output to solve real world problems like classification. DNN's main purpose is finding a way to transform the data into a more creative and abstract component with consideration to the error function.

Neural Network	CNN	RNN	DNN
Applications	• Digital images/photographs • Decoding facial recognition • Object classification • Auto translation • Understanding Climate • Self-driving or autonomous cars	• Handwriting recognition • Speech recognition • Machine translation • Call center analysis • Face detection • Prediction problems	• Fraud Detection • Healthcare • Virtual Assistants • Pixel Restoration • Demographic and Election Predictions • Detecting Developmental Delay in Children

3- Proposing a solution to our problem:

The main purpose of this study is to shed light on a new and effective way to realize predictive maintenance of an electric motor. The data we have from the sensor shows that the vibratory movement and all other parameters are not time independent. As a consequence, we propose to implement this data into a LSTM (abbreviation for Long Short Term Memory) recurrent neural network, that is most known for its "memory property". Each layer takes in consideration the information communicated by the previous layer of neurons.

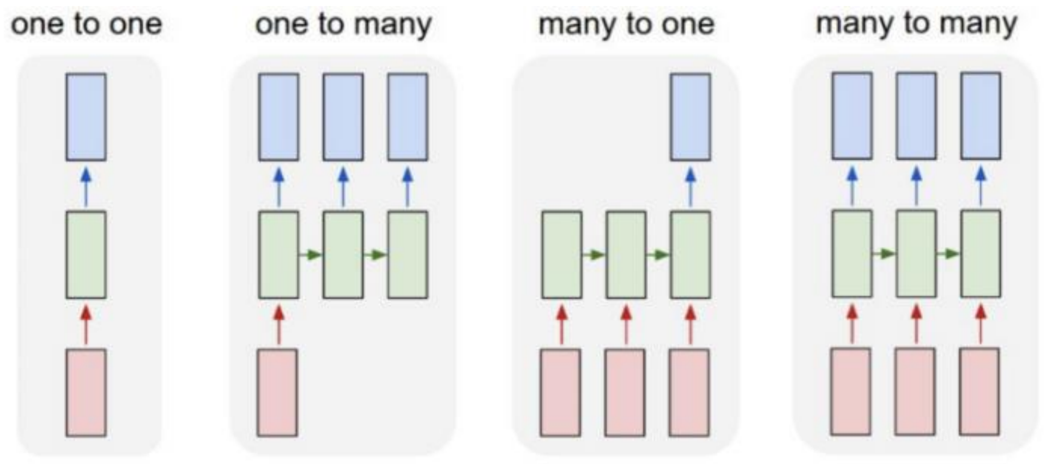
The LSTM RNN is theoretically Turing complete, which means in principle that it could be used to solve any computation problem. The main difference between a LSTM unit and a standard RNN unit is that the LSTM unit is more sophisticated. This goes back to the gates that compose LSTM that regulate better the flow of information through units. It is designed to lean spatial hierarchies of features through its back-propagation algorithm, which is the preferable method of adjusting the weights associated to inputs to reach the minimized loss function.



Forward and backward Propagation in neural networks

4- Types of RNN:

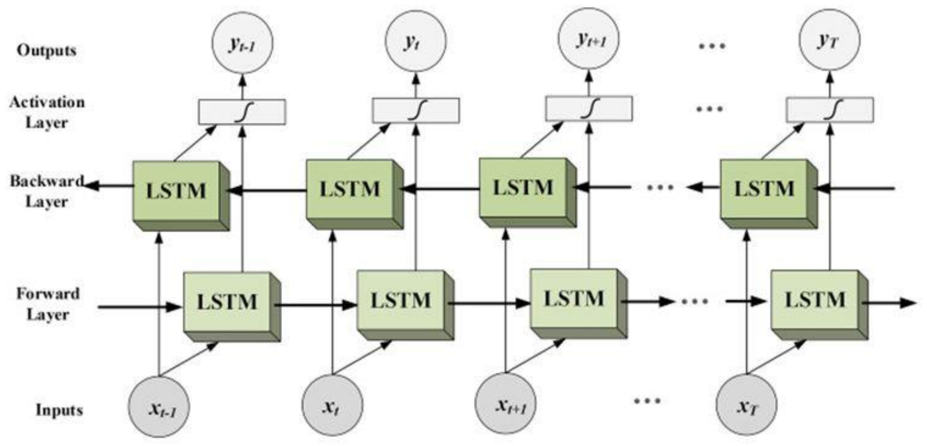
There are 4 types of recurrent neural networks:



Chapter 4: Building the LSTM RNN structure

1- Architecture of our neural network:

We will adopt a bidirectional LTSM network, because we need the properties of both forward and backward (hidden) layers. The image below clarifies the structure of its building blocks:



2- Identification of the LSTM's properties:

In the table below are some of the characteristics of our neural network:

Type of the neural network	Many to many
Input dimension	8*501(number of parameters times samples)
Output dimension	number of parameters
Activation functions	'relu'
Optimizer	Adam

3- The target output:

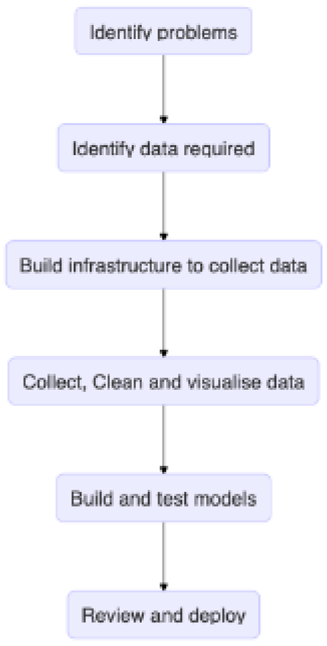
The data we have is composed from all properties of the electric motor. The platform also gives information about the times when the motor faced a failure, marked by a speed of 0 rpm and a temperature of 29°. By our neural network, we want to predict when the motor is going to stop working before it does so, for the main purpose of minimizing the costs resulting from its failure.

We propose that the target output will be a future sequence of the parameters that we put as inputs for the network. And so, the output dimension depends on the number of parameters whose values we want to predict. Since the platform that will provide us with data, has already set outliers for all parameters, we only have to predict their values and see for ourselves whether the outlier is still far to reach or the machine needs a quick intervention.

The LSTM will adjust the weights itself while training, based on the influence on each input on the processed output. And if the accuracy is still far from what we hope to get, we will add more hidden layers, as well as rely on the back-propagation method to minimize the error function. After cleaning and smoothing the data, we will consider 70% of it for the training phase of the neural network and the remaining 30% for testing the model. The sensor's given measurements will present an indication of all future possible problems the motor could be facing.

4- Structuring our solution:

In order for the solution to be effective, we have to follow certain steps:



As a first step, we identified the problem that the factory encounters while the electric motors work; which is the increasing values of overall vibrations and high bearing conditions as times. We aim to predict the next values for all parameters of these motors. That is how we can prevent the variables from ever reaching the outliers that we put for them, let alone the failure of the machine.

In the following chapters, we will go step by step in the building of our solution, from data identification to testing the model.

Chapter 5: Data Analysis

1- Identifying the sensor:

The sensor that we will be working on is a sensor attached to an electric motor, which provides mechanical energy to a fan in the department of coffee. It provides all information about the motor for the last 90 days.

In the table below are the properties of the ABB sensor:

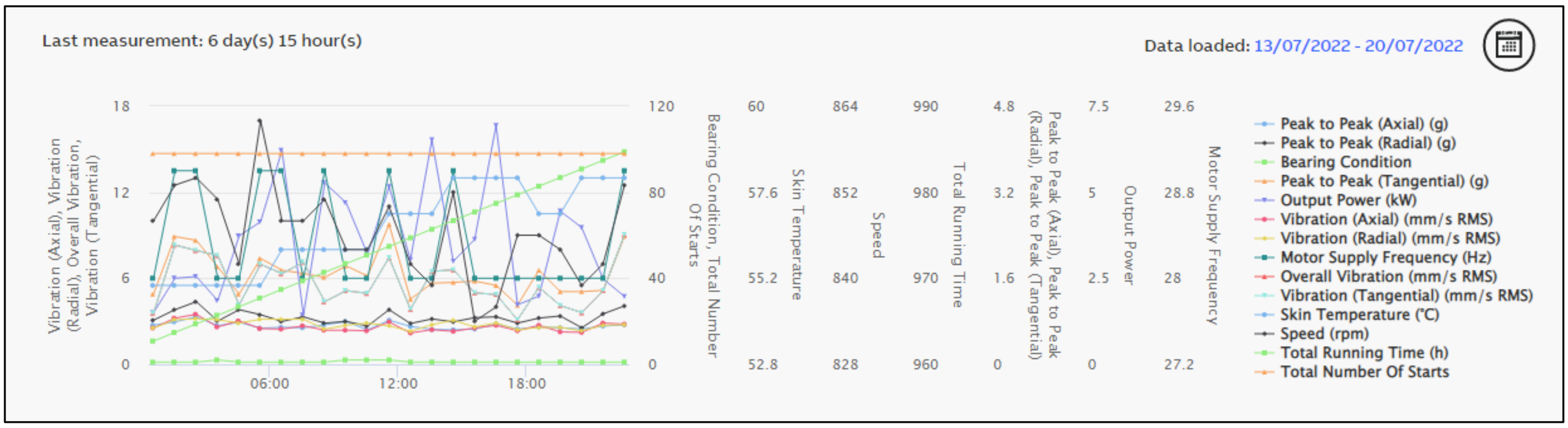
Sensor identifier	S2A0070577-0CF3
Sensor type	Smart Sensor For Motors
Firmware version	9.4
Hardware revision	F
Subscription type	ABB Ability™ Smart Sensor

2- Data collection:

The data that the ABB platform gives is a visualization of the following parameters:

- Health parameters: Overall vibration, Bearing conditions and Skin temperature.
- Operational parameters: Vibration (Axial / Tangential / Radial), speed, motor supply frequency, output power, peak to peak (Axial / Tangential / Radial), Total running time and the total number of starts.

Below is a visualization of these parameters in the last week:



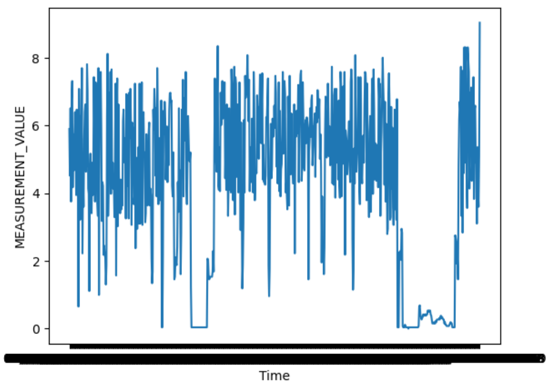
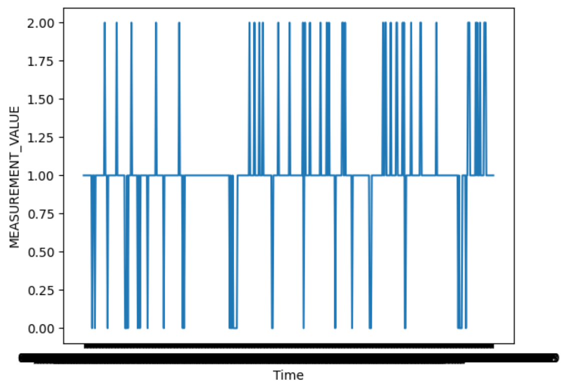
3- Plotting the data:

We will not take in consideration each vibration as an independent parameter; we will rather work with the overall vibration. We also do not need the total running time nor the total number of starts to predict future failures of the motor.

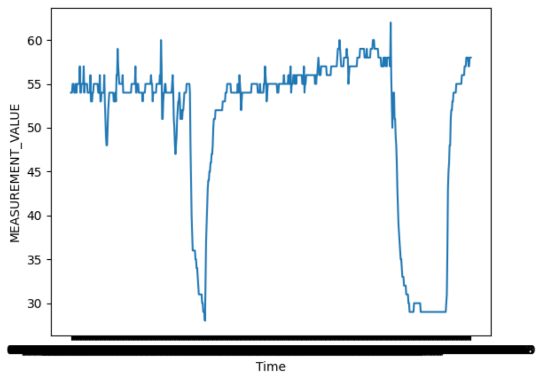
We will be computing all the following syntaxes using python language. As a first step, we download all the data from the platform as excel files. We get two files: a file containing all operational parameters data and another with health parameters data. We plot the data of each parameter on its own.

The syntax used is:

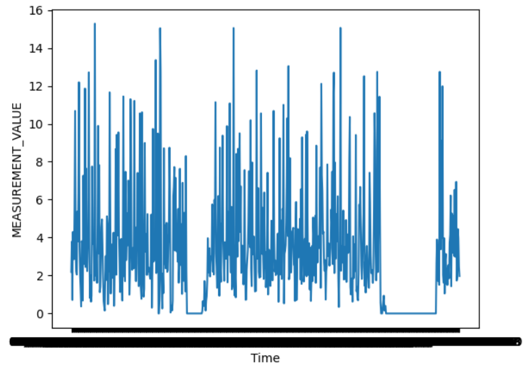
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Coffee = pd.read_excel(r'C:\Users\hp\Coffee.xls', 'Overall Vibration')
pdf = Coffee.Time
pdd = Coffee.MEASUREMENT_VALUE
sns.lineplot(x=pdf, y=pdd, data=Coffee)
plt.show()
```

• Overall Vibration	• Bearing Condition
	

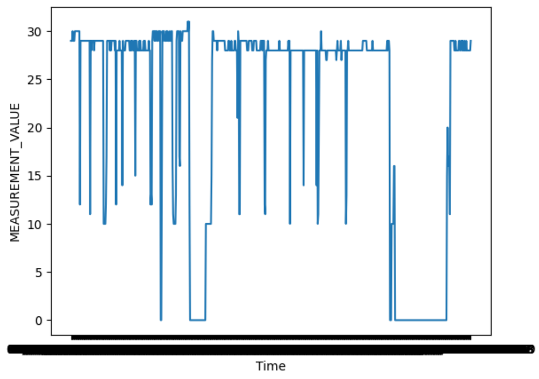
- Skin Temperature



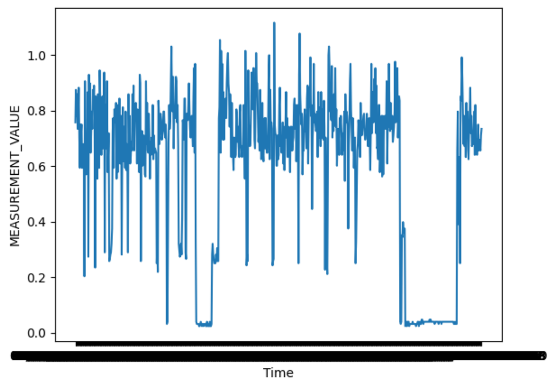
- Output Power



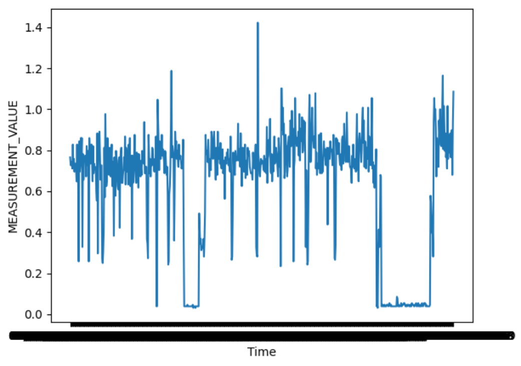
- Motor Supply Frequency



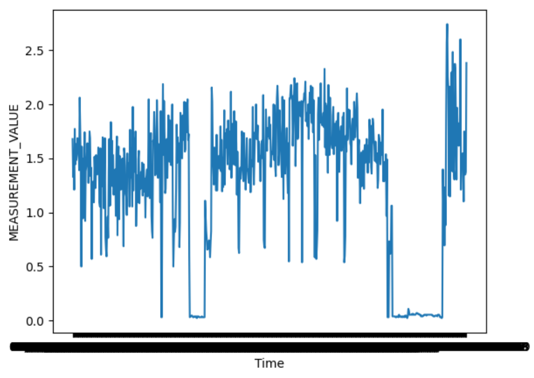
- Peak to Peak (Axial)



- Peak to peak (Radial)



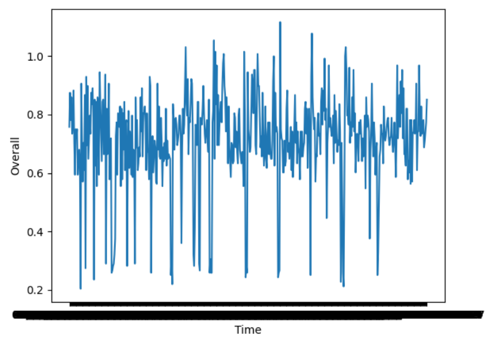
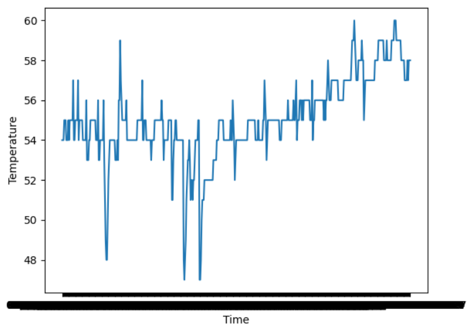
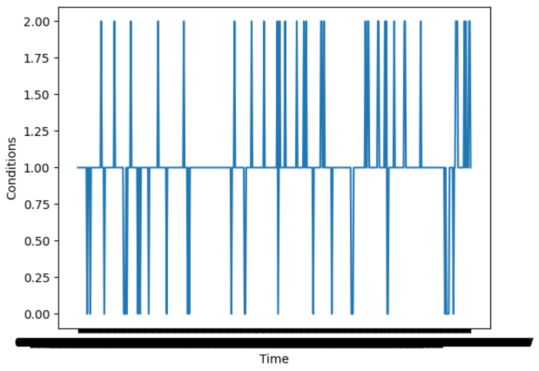
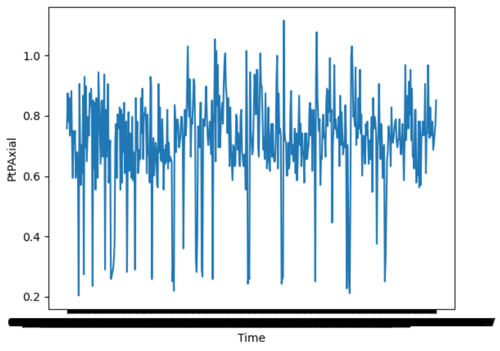
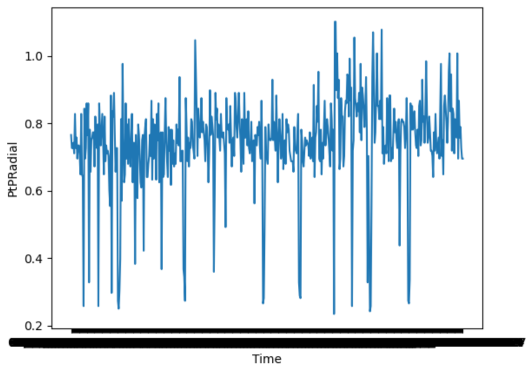
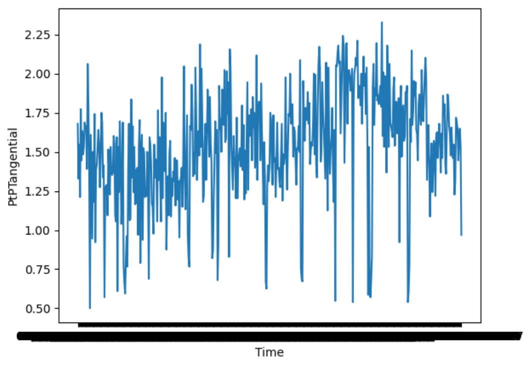
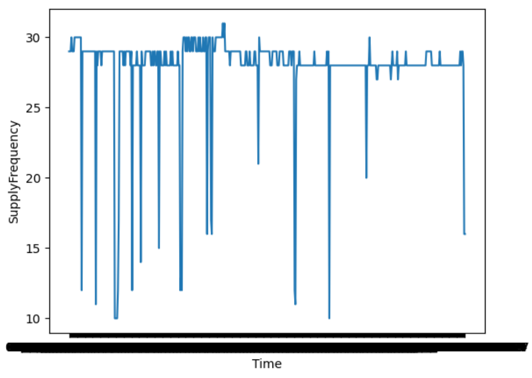
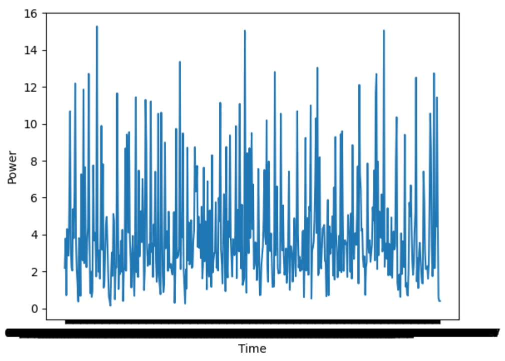
- Peak to peal (Tangential)



4- Data Smoothing:

This step requires going through all the data we got in our excel file, and cleaning it in a few steps. At first, we remove unnecessary, irrelevant data. Then we remove the unwanted observations. As a second step, we deal with missing data. And finally, we put outliers for data filtering.

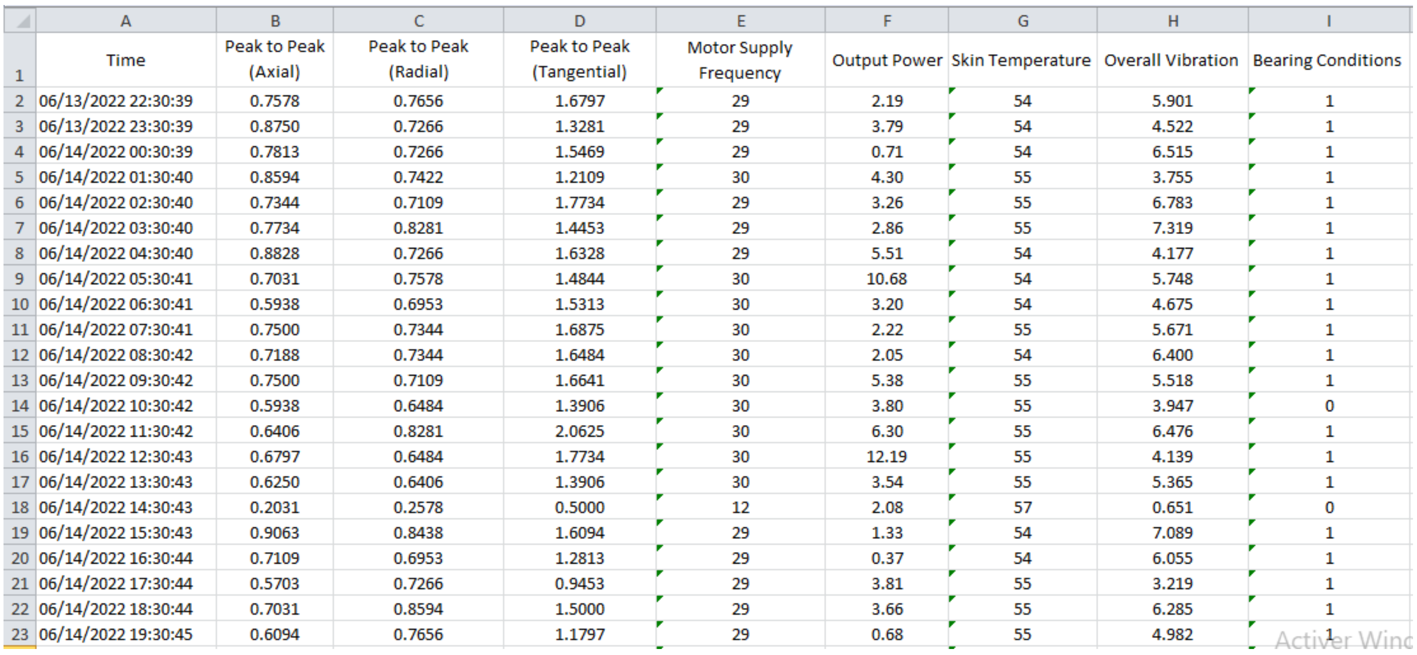
We plot the cleaned data in order to visualize how our parameters really vary through time:

• Overall Vibration  [redacted]	• Skin Temperature  [redacted]
• Bearing condition  [redacted]	• Peak to Peak (Axial)  [redacted]
• Peak to Peak (Radial)  [redacted]	• Peak to Peak (Tangential)  [redacted]
• Motor Supply Frequency  [redacted]	• Output Power  [redacted]

5- Correlation and dimensionality reduction:

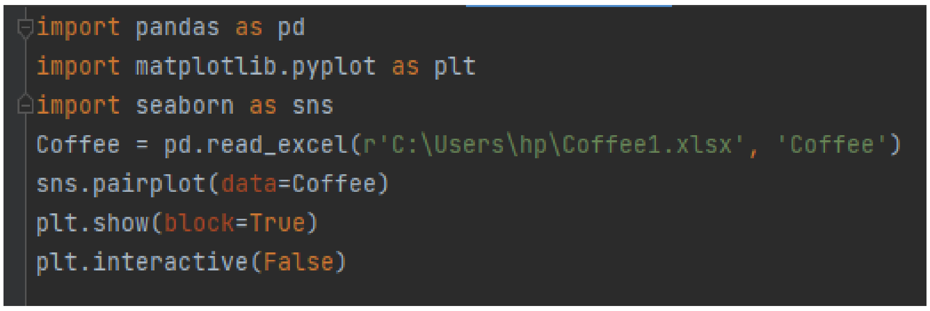
The data that we got from the platform will only give us a high accuracy of the model is we clean it. So before training our neural network on this data, we are going to set a limit for all input parameters and remove all irrelevant data. We also need to reduce dimensionality, which is why we will use Pearson's regression model. This model allows us to calculate the correlation between different input parameters and vibration of the motor (since the vibration is the main reason of failure). As a consequence, we will have only the necessary variables as inputs to our neural networks and let it adjust their weights.

We merge all the data of all the parameters we downloaded from the platform in one excel file. It has 721 files, from 06/13/2022 at 22:30:39 to 07/13/2022 at 21:34:29.

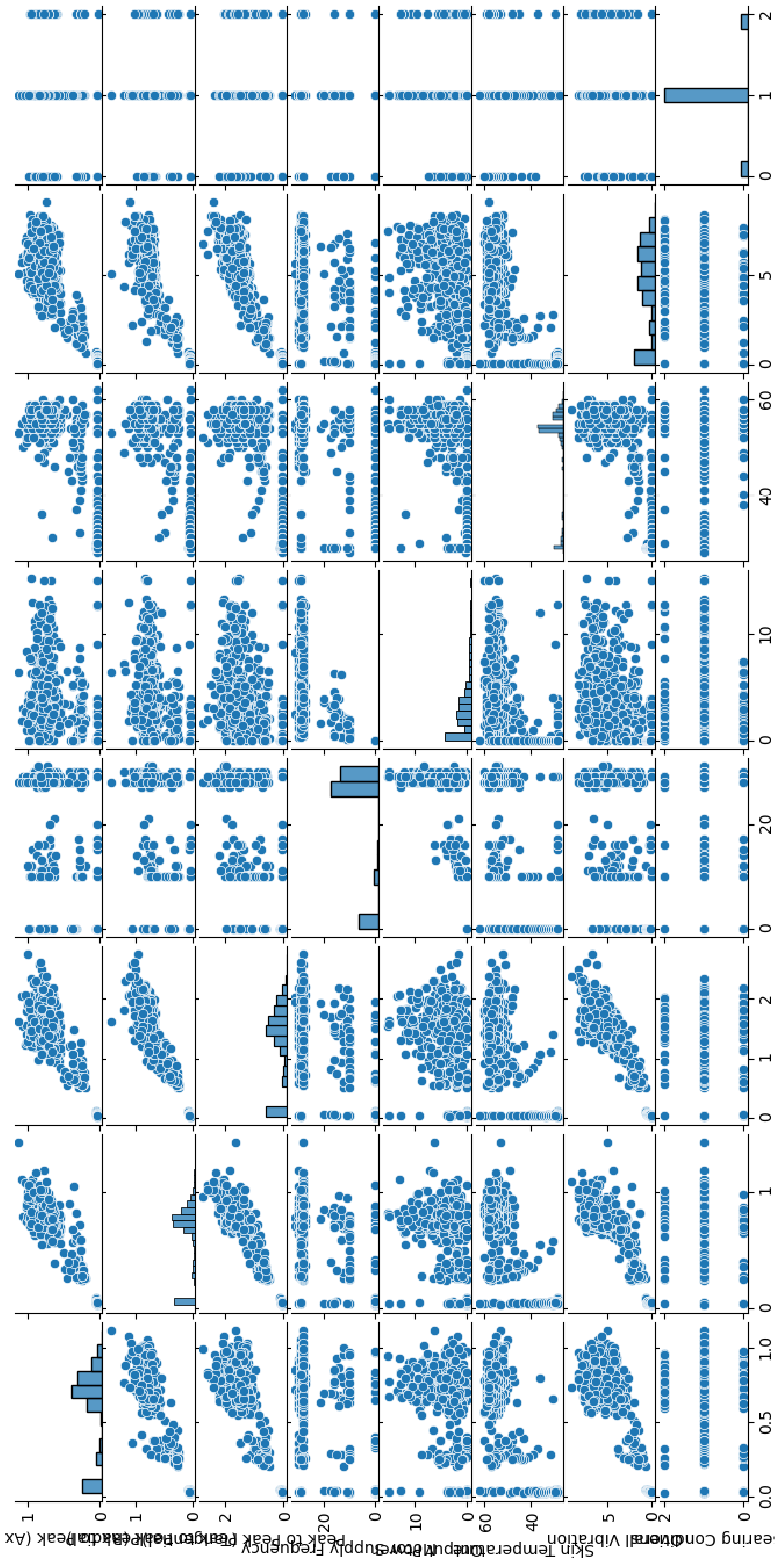


	A	B	C	D	E	F	G	H	I
1	Time	Peak to Peak (Axial)	Peak to Peak (Radial)	Peak to Peak (Tangential)	Motor Supply Frequency	Output Power	Skin Temperature	Overall Vibration	Bearing Conditions
2	06/13/2022 22:30:39	0.7578	0.7656	1.6797	29	2.19	54	5.901	1
3	06/13/2022 23:30:39	0.8750	0.7266	1.3281	29	3.79	54	4.522	1
4	06/14/2022 00:30:39	0.7813	0.7266	1.5469	29	0.71	54	6.515	1
5	06/14/2022 01:30:40	0.8594	0.7422	1.2109	30	4.30	55	3.755	1
6	06/14/2022 02:30:40	0.7344	0.7109	1.7734	29	3.26	55	6.783	1
7	06/14/2022 03:30:40	0.7734	0.8281	1.4453	29	2.86	55	7.319	1
8	06/14/2022 04:30:40	0.8828	0.7266	1.6328	29	5.51	54	4.177	1
9	06/14/2022 05:30:41	0.7031	0.7578	1.4844	30	10.68	54	5.748	1
10	06/14/2022 06:30:41	0.5938	0.6953	1.5313	30	3.20	54	4.675	1
11	06/14/2022 07:30:41	0.7500	0.7344	1.6875	30	2.22	55	5.671	1
12	06/14/2022 08:30:42	0.7188	0.7344	1.6484	30	2.05	54	6.400	1
13	06/14/2022 09:30:42	0.7500	0.7109	1.6641	30	5.38	55	5.518	1
14	06/14/2022 10:30:42	0.5938	0.6484	1.3906	30	3.80	55	3.947	0
15	06/14/2022 11:30:42	0.6406	0.8281	2.0625	30	6.30	55	6.476	1
16	06/14/2022 12:30:43	0.6797	0.6484	1.7734	30	12.19	55	4.139	1
17	06/14/2022 13:30:43	0.6250	0.6406	1.3906	30	3.54	55	5.365	1
18	06/14/2022 14:30:43	0.2031	0.2578	0.5000	12	2.08	57	0.651	0
19	06/14/2022 15:30:43	0.9063	0.8438	1.6094	29	1.33	54	7.089	1
20	06/14/2022 16:30:44	0.7109	0.6953	1.2813	29	0.37	54	6.055	1
21	06/14/2022 17:30:44	0.5703	0.7266	0.9453	29	3.81	55	3.219	1
22	06/14/2022 18:30:44	0.7031	0.8594	1.5000	29	3.66	55	6.285	1
23	06/14/2022 19:30:45	0.6094	0.7656	1.1797	29	0.68	55	4.982	1

We plot the regression matrix of the data using the following syntax:

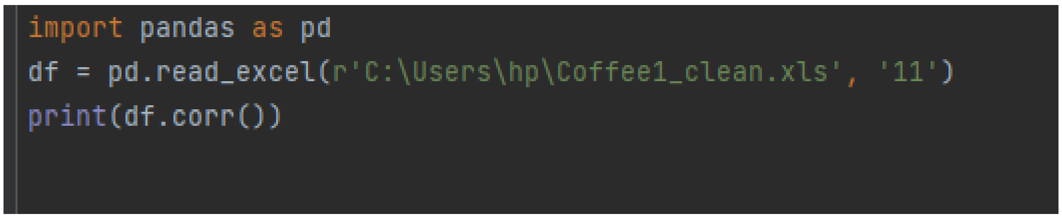


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Coffee = pd.read_excel(r'C:\Users\hp\Coffee1.xlsx', 'Coffee')
sns.pairplot(data=Coffee)
plt.show(block=True)
plt.interactive(False)
```



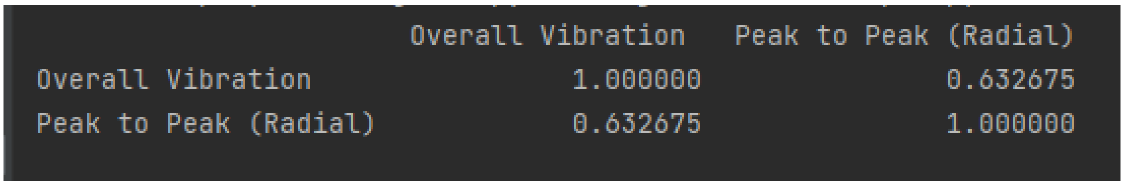
Regression Matrix

However we cannot depend only on the regression matrix to look for correlation between two parameters of the motor. We will use the following syntax to calculate the exact correlation coefficient between the overall vibration and the rest of the variables.



```
import pandas as pd
df = pd.read_excel(r'C:\Users\hp\Coffee1_clean.xls', '11')
print(df.corr())
```

And the results are as following:



```
                         Overall Vibration   Peak to Peak (Radial)
Overall Vibration                 1.000000                0.632675
Peak to Peak (Radial)             0.632675                1.000000
```

Vibration And	Peak to Peak (Axial)	Peak to Peak (Radial)	Peak to Peak (Tangential)	Output Power	Motor Supply Frequency	Skin Temperature	Bearing Conditions
Correlation Coefficient	0.617624	0.632675	0.760372	0.017974	0.159445	0.024795	0.0132487

We just reduced our system's dimension by 4 dimensions, which depending on their low correlation coefficient with the vibration parameter, are not very influencing on the motor's healthy state. As the Pearson model requires, we will only consider the parameters whose correlation coefficient to the vibration is higher than 60%.

Chapter 6: Computing the network model

1- Data structure:

As the collected, cleaned data is in an excel file, we have to implement it in python. All the following syntaxes will be computed in Google colab.

```
#Implementing data
!pip install xlrd>=2.0.1
import pandas as pd
import xlrd
from google.colab import files
uploaded=files.upload()
import io
df=pd.read_excel(io.BytesIO(uploaded['Coffee.xlsx']))
print(df)
```

The output is:

```
                    Time  PtPAxial  PtPRadial  PtPTangential  Vibrations
0    06/13/2022 22:30:39    0.7578     0.7656         1.6797       5.901
1    06/13/2022 23:30:39    0.8750     0.7266         1.3281       4.522
2    06/14/2022 00:30:39    0.7813     0.7266         1.5469       6.515
3    06/14/2022 01:30:40    0.8594     0.7422         1.2109       3.755
4    06/14/2022 02:30:40    0.7344     0.7109         1.7734       6.783
..                   ...       ...        ...            ...         ...
496  07/07/2022 06:33:33    0.7578     0.6953         1.6875       7.549
497  07/07/2022 07:33:34    0.7813     0.8672         1.6484       6.553
498  07/07/2022 08:33:34    0.6875     0.7578         1.4453       3.219
499  07/07/2022 09:33:34    0.7109     0.7891         1.5703       3.296
500  2022-07-07 10:33:34    0.7422     0.7266         1.6484       6.553

[501 rows x 5 columns]
```

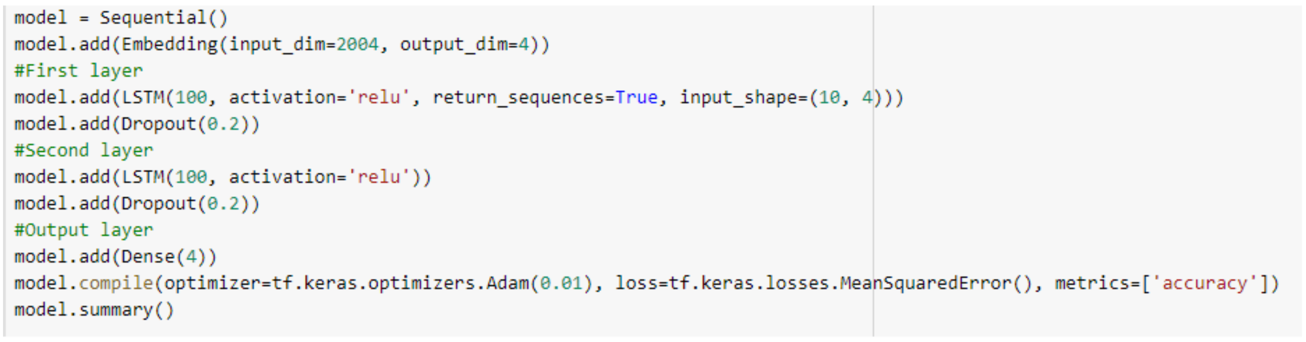
2- Importing libraries:

We are going to need more than a couple libraries to code our neural network:

```
#Importing libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy import hstack, array
from tensorflow import keras
from keras.layers import LSTM, Dense, Dropout
from keras.models import Sequential
!pip install xlrd>=2.0.1
import xlrd
from google.colab import files
import io
```

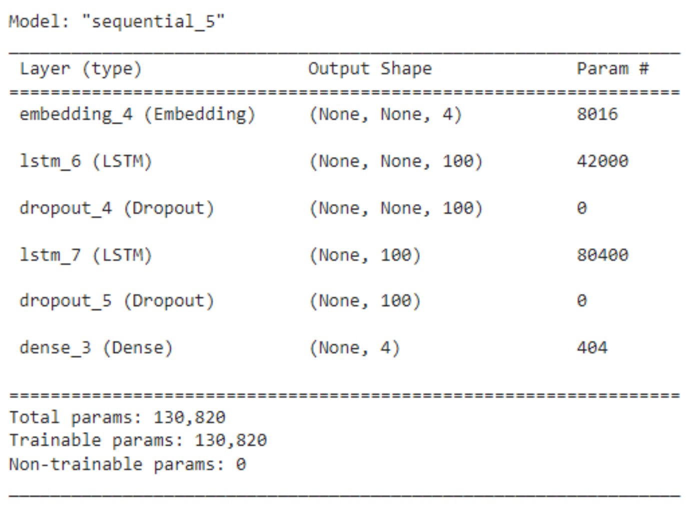
3- Model Summary:

The following code returns all the parameters our neural network's architecture:



```
model = Sequential()
model.add(Embedding(input_dim=2004, output_dim=4))
#First layer
model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(10, 4)))
model.add(Dropout(0.2))
#Second layer
model.add(LSTM(100, activation='relu'))
model.add(Dropout(0.2))
#Output layer
model.add(Dense(4))
model.compile(optimizer=tf.keras.optimizers.Adam(0.01), loss=tf.keras.losses.MeanSquaredError(), metrics=['accuracy'])
model.summary()
```

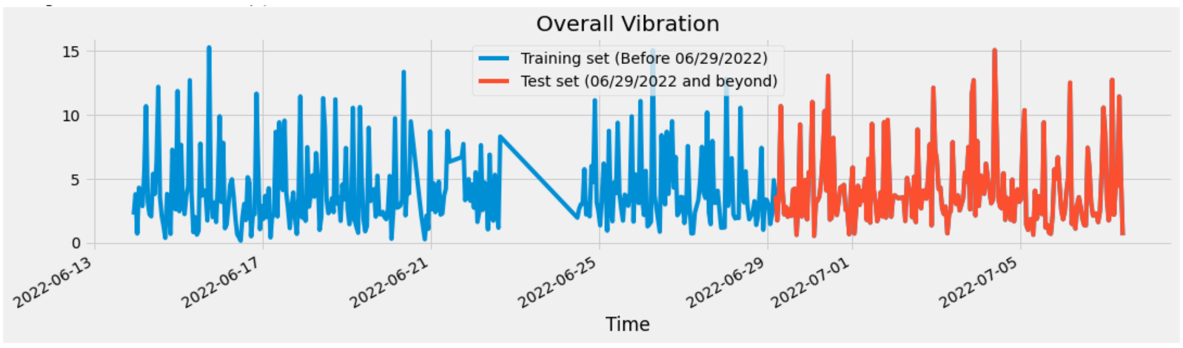
The output is:



```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, None, 4)           8016

 lstm_6 (LSTM)               (None, None, 100)         42000

 dropout_4 (Dropout)         (None, None, 100)         0

 lstm_7 (LSTM)               (None, 100)               80400

 dropout_5 (Dropout)         (None, 100)               0

 dense_3 (Dense)             (None, 4)                 404

=================================================================
Total params: 130,820
Trainable params: 130,820
Non-trainable params: 0
_________________________________________________________________
```

4- Training and testing data:

We split our data into two parts: one part for training the model, and the second part for testing the model. We plot only the overall vibration for the main purpose of visualizing this split:



5- Building the LSTM NN model:

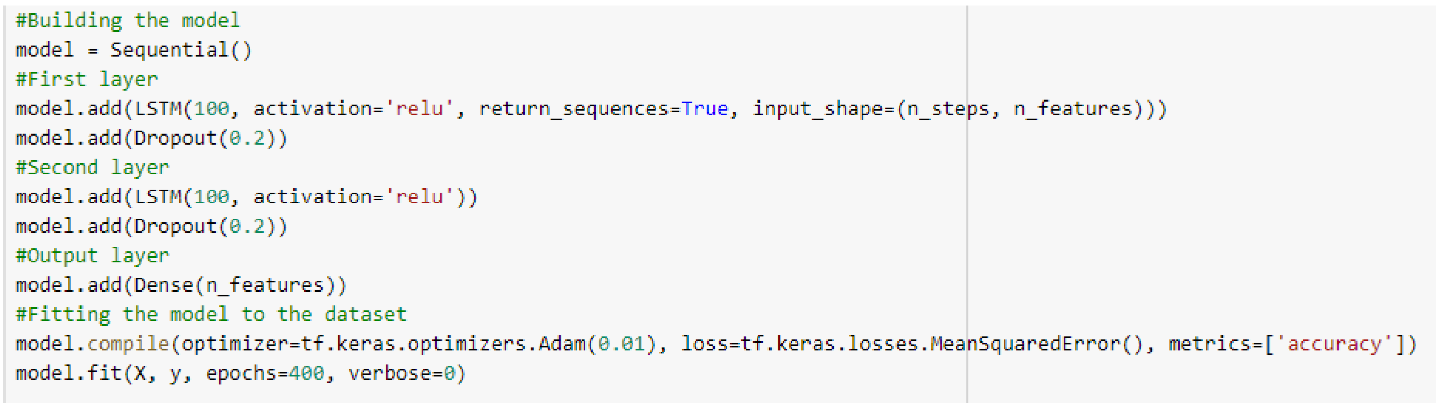
We define a function that will allow us to split data into samples:

```
#Adding a split_sequences function
def split_sequences(sequences, n_steps):
  X, y = list(), list()
  for i in range(len(sequences)):
    # find the end of this pattern
    end_ix = i + n_steps
    # check if we are beyond the dataset
    if end_ix > len(sequences)-1:
      break
    # gather input and output parts of the pattern
    seq_x, seq_y = sequences[i:end_ix, :], sequences[end_ix, :]
    X.append(seq_x)
    y.append(seq_y)
  return array(X), array(y)
```

And now that the data is already implemented, we reshape it and split it:

```
#Defining input sequences
in_seq1 = np.array(dataset.Vibrationss)
in_seq2 = np.array(dataset.PtPAxial)
in_seq3 = np.array(dataset.PtPRadial)
in_seq4 = np.array(dataset.PtPTangential)

#Reshaping data
in_seq1 = in_seq1.reshape((len(in_seq1), 1))
in_seq2 = in_seq2.reshape((len(in_seq2), 1))
in_seq3 = in_seq3.reshape((len(in_seq3), 1))
in_seq4 = in_seq4.reshape((len(in_seq4), 1))

#Splitting data
dataset = hstack((in_seq1, in_seq2, in_seq3, in_seq4))
n_steps = 10
X, y = split_sequences(dataset, n_steps)
n_features = 4
```

Finally, we define the model and its layers:

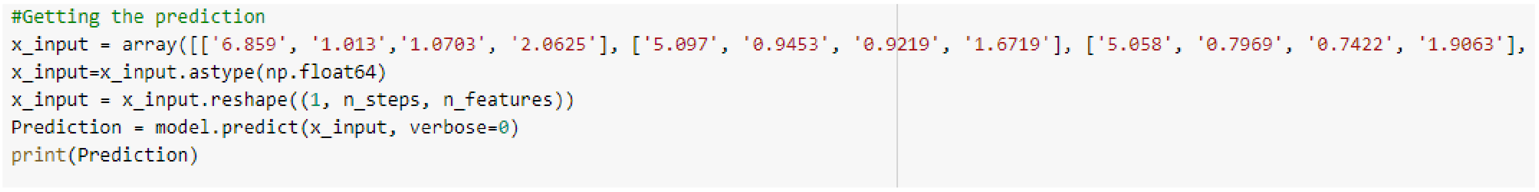


```
#Building the model
model = Sequential()
#First layer
model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(Dropout(0.2))
#Second layer
model.add(LSTM(100, activation='relu'))
model.add(Dropout(0.2))
#Output layer
model.add(Dense(n_features))
#Fitting the model to the dataset
model.compile(optimizer=tf.keras.optimizers.Adam(0.01), loss=tf.keras.losses.MeanSquaredError(), metrics=['accuracy'])
model.fit(X, y, epochs=400, verbose=0)
```

To avoid over fitting, which is usually a problem in the LSTM NN's, we will only have 2 layers because our dataset is relatively small. We will also use 400 epochs for our network, so that we maximize the number of times where the LSTM goes backwards and forwards through data.

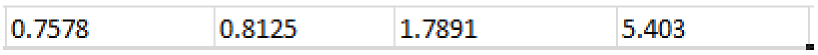
6- Predicting the target sequence:

The syntax below will allow any user to get the following sequence of the parameters that they entered as an input. Since we decided on 10 steps for our network, we have to give 10 sequences as an input.



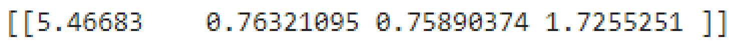
```
#Getting the prediction
x_input = array([['6.859', '1.013','1.0703', '2.0625'], ['5.097', '0.9453', '0.9219', '1.6719'], ['5.058', '0.7969', '0.7422', '1.9063'],
x_input=x_input.astype(np.float64)
x_input = x_input.reshape((1, n_steps, n_features))
Prediction = model.predict(x_input, verbose=0)
print(Prediction)
```

The target output is the following sequence:



0.7578	0.8125	1.7891	5.403

The processed output of our network is:



```
[[5.46683    0.76321095 0.75890374 1.7255251 ]]
```

We sum up in this table:

Parameter	Overall Vibration	Peak to Peak (Axial)	Peak to Peak (Radial)	Peak to Peak (Tangential)
Target output	5.403	0.7578	0.8125	1.7891
Processed output	5.4668	0.7632	0.7589	1.7255

7- Predicting all future sequences:

We are going to define a function that will allow us to get, at once, the processed output for a parameter for all data in the testing phase:

```
#Test_data prediction
for i in dataset['PtPAxial']['07/03/2022 08:33:07':]:
  test_data=np.array([i-10, i-9, i-8, i-7, i-6, i-5, i-4, i-3, i-2, i-1])
  test_data= test_data.astype(np.float64)
  test_data = test_data.reshape((1, n_steps, n_features))
  predictNextNumber = model.predict(test_data, verbose=1)
  print(predictNextNumber)
```

The output is as follows:

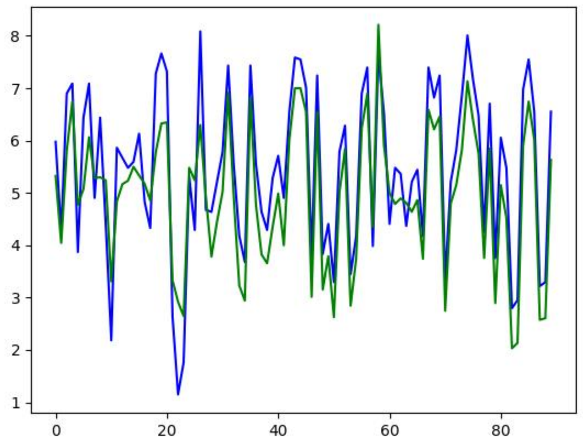
```
1/1 [==============================] - 0s 22ms/step
[[1.74102]]
1/1 [==============================] - 0s 23ms/step
[[0.78715503]]
1/1 [==============================] - 0s 22ms/step
[[1.5250895]]
1/1 [==============================] - 0s 21ms/step
[[1.3364959]]
1/1 [==============================] - 0s 22ms/step
[[0.49195898]]
1/1 [==============================] - 0s 22ms/step
[[0.5391894]]
1/1 [==============================] - 0s 32ms/step
[[1.8300214]]
1/1 [==============================] - 0s 22ms/step
[[2.0226798]]
1/1 [==============================] - 0s 21ms/step
[[1.6900632]]
1/1 [==============================] - 0s 24ms/step
[[0.6183994]]
1/1 [==============================] - 0s 20ms/step
[[0.64185685]]
```

8- The accuracy of the network:

We predict all values of the parameter 'Overall Vibration' in the testing phase. We put the values in the same excel file as the real values, and we plot them together in one graph to see the difference.

```
#Plotting the predictions
Vibrationss=np.array(dataset.Vibrationss)
Predictionss=np.array(dataset.Predictionss)
plt.plot(Vibrationss,color='blue')
plt.plot(Predictionss, color='green')
plt.show()
```

The blue line presents the real values and the green line the predicted ones:



The syntax for calculating the accuracy of the model is as follows. And to avoid the problem of continuous values, we around all real and predicted valued and then study the accuracy between the new arrays.

```
from sklearn.metrics import accuracy_score
Vibrationss=np.array(dataset.Vibrationss)
Predictionss=np.array(dataset.Predictionss)
acc=accuracy_score(Vibrationss, Predictionss)
print(acc)
```

The output is:

```
0.752643
```

This accuracy is relatively mediocre, due to the small sized data that we work on. But on the other hand, this provides us with an important advantage: the data is not noisy. And in order to avoid underfitting, we structure the capacity of our neural network as to have 400 epochs, to gather as much information as it can about our inputs. And to avoid overfitting, we minimize the hidden layers into only two.

Chapter 7: Bachmann-Landau notation:

Big O Complexity of our neural network

1- Introduction:

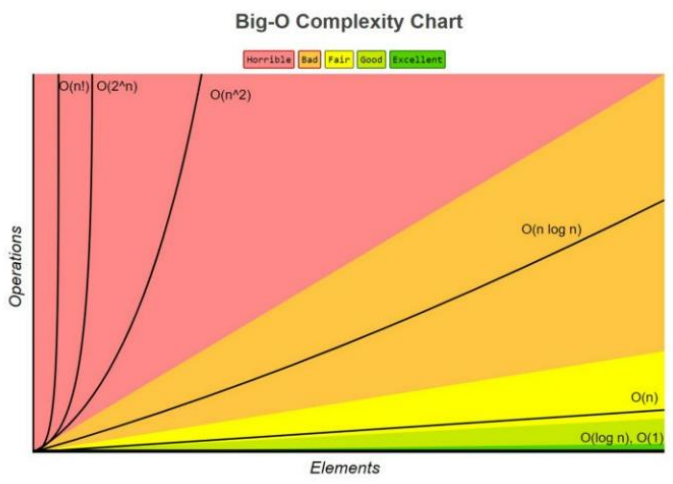
Big O is a member of a family of notations invented by Paul Bachmann, Edmund Landau, and others, collectively called Bachmann–Landau notation or asymptotic notation. It is often referred to as “asymptotic notation”, as it provides a description of the limiting behavior of a function when the argument tends towards a particular value or infinity.

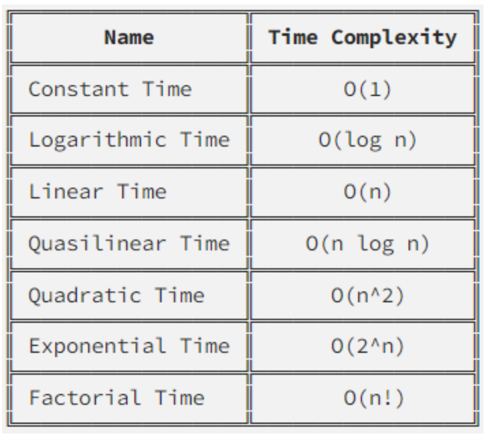
In computer science, Big O notation serves as a tool to classify algorithms according to the correlation between their inputs’ growth and the growth of their running time or space. In other words, it provides us with a method of scaling any algorithm in terms of the size of its input.

Associated with big O notation are several related notations, using the symbols o, Ω, ω, and Θ, to describe other kinds of bounds on asymptotic growth rates. These notations define three cases of the complexity of an algorithm: best-case, average-case and worst-case. In the following parts, we are going to focus our study only on the worst case, as defining it is the first step towards optimizing our algorithm. Nowadays, all systems come up with a large memory, so space complexity is not considered as important as time complexity. So, we are going to start by calculating the time complexity as it is more of an impediment to our network’s performance.

2- Calculating time complexity:

The time complexity of an algorithm is the computational complexity that describes the amount of time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform. Below is a chart illustrating the evolution of the most known Big-O Time Complexities depending on the growth of their inputs, and a table stating their names.





Name	Time Complexity
Constant Time	O(1)
Logarithmic Time	O(log n)
Linear Time	O(n)
Quasilinear Time	O(n log n)
Quadratic Time	O(n^2)
Exponential Time	O(2^n)
Factorial Time	O(n!)

Looking at your code, we have to take in consideration 4 steps to calculate ist total time complexity. We have considered our batch to be one sample at a time (samples that are fed to the neural network), so we do not take in consideration the batch size in our calculations.

- **O_1 for first layer:**

The embedding layer has 4 nodes or 4 neurons that we note m, and four inputs fed at one time to this layer. We note the inputs i. We multiply the dependencies against each other, and we get the time complexity for this layer: $O_1 = O(n = m * i)$.

- **O_2 for second layer:**

The first LSTM layer has 100 nodes that we note n, and 4 inputs fed at one time to each neuron of this layer. We note the inputs m. We multiply the dependencies against each other, and we get the time complexity for this layer: $O_2 = O(n * m)$.

- **O_3 for third layer:**

The second LSTM layer has 100 nodes that we note n, and 100 inputs fed at one time to each neuron this layer. We note the inputs n also. We multiply the dependencies against each other, and we get the time complexity for this layer: $O_3 = O(n * n) = O(n^2)$.

- **O_4 for fourth layer:**

The embedding layer has 4 nodes that we note m, and 100 inputs fed at one time to each neuron of this layer. We note the inputs n. We multiply the dependencies against each other, and we get the time complexity for this layer: $O_4 = O(n * m)$.

The time for computing the activation function “Relu” is linear, so we do not add it as it can be neglected compared to these four values of complexity.

We sum up all complexities of all operations: $\sum_{i=1}^{4} O_i = O(n * m * 2 + m * i + n^2)$.

- **Feedforward and Backpropagation:**

Since we have trained 65% of our data and tested the accuracy of the neural network for the remaining 35% of data. We must take in consideration the total time complexity for all the times where the NN went backwards and forward through our data. That is the number of epochs times 2 for the 65% data and times 1 for the 35% data (since for the testing part, the network only moves forward to generate the wanted prediction. This translates into the following equation: $O_{total} = \alpha * O(n * m * 2 + m * i + n^2)\, , \alpha \in R$.

This is the total time complexity of our NN. As we can see, the parameters m and i are so inferior the parameter n, but still, they contribute to the total time of the algorithm’s computing time. In other terms, our time complexity is higher than the quadratic complexity time $O(n^2)$.

3- Calculating space complexity:

Space complexity is the amount of memory used by an algorithm to execute and produce the targeted result. It is decomposed into two parts, the input space; taken by all inputs of the network, and the auxiliary space; which is the extra or temporary space used by the network during its computation. We get the total value by adding these two together:

Space Complexity = Auxiliary Space + Input space

The input space is the sum of all inputs to all layers of our network. Knowing that each float variable takes 4 bytes of space, we illustrate the input space for each layer in the table below:

Layers	Embedding layer	First LSTM layer	Second LSTM layer	Final output dense layer
Inputs	4	100	100	100
Amount of memory used these inputs	16	400	400	400

As we sum these values, the total input space of our network is: 1216 bytes.

If we consider n the number of inputs for the three last layers, the total memory requirement can be written as $O(3n + 4)$ which is increasing linearly with the increase in the input value n, hence it is called as Linear Space Complexity.

As the neural network does not take any temporal variables, we can neglect the term of auxiliary space as it is null.

So, the total space complexity of our network is $O(3n + 4)$. This result could have been bigger as there were more inputs to feed to the network, but due to our study of dimensionality reduction, our algorithm became more performant.

Conclusion

To sum up this report, we have focused in the first two chapters on an introduction into Nestle's world. We have dived into its history, and we presented the organizational chart of the factory. We then moved to the description of each of the processes behind the production of powdered milk and coffee. Finally, we talked about the storing of these products within the factory.

In the following chapters, we embarked on the study for the algorithm used to predict all future values of health parameters of the electric motor. We started off by a small introduction of neural networks in general and pointed out to how deep learning can be so useful to the predictive maintenance. We chose a LSTM RNN, whose structure we built in the fourth chapter.

A small data analysis has been made through multiple steps: identifying the sensor, data collection, plotting the data, data smoothing, correlation study and dimensionality reduction. We passed then to the computation of the network model. We defined the data on which the NN trained and used the testing data for the prediction of all future targeted sequences.

Our report ended with an interesting notation related to the complexity of algorithms: The Big O notation. We calculated both time and space complexity of our neural network as it is the first step towards optimizing its execution's time especially.