Big Data in aeronautics: Application to the predictive maintenance of the landing gear

Francisco José Gutiérrez García[1]
Student ofUniversity of Seville, Seville, Spain

José Ángel Jiménez Doblado[2]
Student ofUniversity of Seville, Seville, Spain

José Manuel Montilla García[3]
Student ofUniversity of Seville, Seville, Spain

Daniel Núñez Álvarez[4]
Student ofUniversity of Seville, Seville, Spain

Juan Ramón Parra Vilar[5]
Student ofUniversity of Seville, Seville, Spain

Tomás Sánchez Sánchez[6]
Student ofUniversity of Seville, Seville, Spain

The aim of this project is to apply Big Data techniques into the aeronautic systems maintenance. Using the actual amount of data available from the use of these systems and all the historical patterns, it is possible to define the components performance and its real condition. This knowledge can be use to find a better way to accomplish the service maintenance by findind correlations between the components anomalies and the data obtained from the different sensors of the aircraft.

Nomenclature

SQL = Structured Query Language.
NoSQL = No Structured Query Language.
P2P = Peer-to-peer.
RDD = Resilient Distributed Dataset.
PDF = Probability density function.
CQL = Cassandra Query Language.

I. Introduction

In the past few years the term 'Big Data' has increased its popularity in the business world due to the competitive advantage it could give. Hence, a great number of companies are focussing more and more on improving their Data Science departments. But, what does Big Data actually mean? Big Data is a term used to describe the storage, analysis and interpretation of immense and diverse amount of data. Due to its size and complexity, the management of Big Data is the new challenge that many companies are facing these days.

[1] frangut94@gmail.com
[2] joseangeljimenez102@gmail.com
[3] garcimonti@gmail.com
[4] dnunezalvarez@hotmail.com
[5] juan.ramon.parra.vilar@gmail.com
[6] tomas94sansan@gmail.com

One of the difficulties lies in the data storage. Data is constantly growing, at a pace of dozens of terabytes per day; and, thus, a great digital storage capacity is needed. However, not only the storage capacity is important, but also the data architecture is something to bear in mind while working with Big Data.

As far as analysis and interpretation are concerned, it is the capacity and velocity in reading and computing data what constrain the management of Big Data. In order to solve the problem of reading and computing enormous amount of data there are, in the market, some computer software specialised in working with Big Data. In the following sections, it will be explained what software has been used in this project.

Big Data can have many applications in very diverse fields inside the business world. One of them is predictive maintenance; that is, using all the information and data to predict accurately the exact moment when the maintenance of a product part should be performed. Taking into account the difficulties inherent to Big Data mentioned before, its application to predictive maintenance can be profitable for the company in terms of the costs reduction related to spare parts, people and product availability. This concept has been conceptually developed, as shown in [1], [2], [3]; however, it has not been actually applied to any maintenance application, and that is the point addressed in this paper.

In this project, it will be analyzed the application of Big Data to predictive maintenance of aircrafts; more specifically, to the predictive maintenance of the landing gear. In Chapter II, the way to present the data collected is described. Then, in Chapter III, it is explained how this data is stored in a database and the way to access to it. In Chapter IV, the fundamentals of Big Data analysis tools are explained; and in Chapter V, the algorithm used to detect anomalies is described in detail. In Chapter VI it is explained how the results are presented to the users. Finally, in Chapter VIII and Chapter VIII, there is a brief description of the conclusions and next steps of the actual project. In the Figure 1 it is shown a flowchart with the different parts of the process developed in this paper.

Figure 1. Flowchart of the process of a Big Data application.

II. Data gathering

The main objective of this project is to handle with real information about different variables related with a specific flight. In order to achieve that, the first step of the project is to collect that information; nevertheless, due to the little, or no, data available it is necessay to recreate certain variables with approximate values to test the algorithm and show the great potential it has.

To save data, it is necessary to distinguish between 5 types of variables:

- Binaries. Those which could have two values, zero or one.
- Discrete. Those variables that have a decimal value, but just once per flight.
- DiscreteU. Those discrete variables that have more than one value per flight. The number of times they are measured could be unknown or it could have a nominal number of times of occurrence.
- Discontinuous. Those variables that are measured in function of the timing of a sensor and have an array of values during a period. Can be measured more than once in each flight.
  Example: 'RW'* = ([1.5,101,305,604,737]] , [[…] […]])
  The first list shows time. The first element is the timing of the sensor. The other elements represent the initial and final value of the interval measured (each pair of numbers is an interval). The second list has inside as many lists as intervals.
- Continuous. Measured with the timing of a sensor during the whole flight.
  Example: 'OSAT'* = ([1.5] , […])

To narrow down the problem, it will be focused on 36 variables that correspond to those five sets. Each variable has associated an id code. For instance, 'TW' is the wheel temperature. As far as discontinuous variables are

concerned, it is important to underline the moments in which the intervals are measured in a flight. There will be enclosed an array with the information of the sensor timing, the initial and final time value for each interval of measurements.

To see the complexity of the problem to solve, it will be set the biggest dimension to treat. Due to the great amount of data, there are up to seven 'dimensions' to treat and relate. Those seven dimensions are shown in Table 1.

Day-time	Model	Plane	Flight	Type of variable	Variable	Arrays
02/03/2018	A-320	XRY-45	I-8000	Discontinuous	'RW'*	• Time • Values

Table 1. Example of the seven dimensions of the data gathered.

Each field is shown like a tree diagram. When certain values of the variables are wanted to be visualized or treated, the whole 7-dimensional system has to be treated.

III. Data storage

Once the data of a flight is available for study, it must be stored on a data base. There are many ways to do it, but as a consequence of the great amount of information to deal with it is necessary to use a structured data base. The identification of items must be easy and as quickly as possible to treat the data before the following flight starts.

SQL or NoSQL are different types of structured data base. SQL data base will be chosen in the first approach to the collection of data. SQL can be seen as a table with columns and rows, in which each element is a name, a value or an array. To proceed with the savings there will be used the SQlite mode in python. This is a relational database into a library in C adapted to different programming languages, Python included.

First of all, the database has to be created, it means that the format has to be determined previous to the storage of the information gathered. Each plane will have its own database archive, relating the name of the plane to the name of the database. Each row matches with a flight, and each column to a variable type defined in 'data gathering'. Due to the fact that, with time, just one database for a plane would be unproductive because it would tend to infinite, there will be a database for each day. This means that present and past information can be obtained from the SQL archives with the same quickness, having them stored in different places, avoiding using extra time when only present values are wanted.

There will be dictionaries with the name of planes and flights to have them completely situated into the database. Inside a plane database, the first column corresponds to the name of the flight, with its identification name. The second column corresponds to the 'id' or identification number that will be a autoincrement entire key. The other columns will have the clue name of the different variables used in the problem.

Flight	ID	'TW'*	'LV'*	...	'LW'*
I-8000	1	250	121	...	1.5
I-8001	2	243	115		1.3

Table 2. Example of the database of a plane.

The 'id' will be the primary key of the element, which means that when relating an entire row with other SQL archives, the 'id' would be the best way to create a relationship with another row different to this one but with data of the same flight. It also sets the natural order of flights during the day stored.

The format assigned to each variable depends on its type. For binaries an entire ('int') format will be chosen. For discrete variables 'float' is the format chosen. However, SQLite3 does not allow to save lists in a slot into the database, therefore it will be necessary to codify the information by serializing and saving it as a binary code.

Serializing means that an array, including multiple lists in the case of discontinuous, is converted to an argument which involves all that data but can be treated as a unique element to store it in the SQL database. Due to that, discreteU, continuous and discontinuous will be set as a 'blob' format variable. 'Blob' just means that despite being

an element, it represents an array, and it will be necessary to transform the element into its original state to treat that information.

IV. Data analysis with Big Data tool

As it was mentioned before, the final purpose of any Big Data application is to infer some conclusions about a subject based in an enormous data source, in comparison with what any human could possibly handle by himself. In doing that, the computational approach cannot be what you could call a classic approach; that is, to compute the information in a serial way, one instruction at a time. The approach that we look for in this matter is that of the multi CPU computation, but in a big way. To do this, there has been a development in the software side to make it possible and easy to implement. Some important data processing engines are Spark, Presto, Qubole, BigML and Statwing. Each one has its own characteristics that make them more suitable for different applications (some are pay to use, some are focused in machine learning, each one is implemented with different programming languages, etc), and, for the present project, we have chosen Spark for various reasons that are about to be explained.

Apache Spark is a fast, free to use and general engine for large-scale data processing, you can write applications quickly in Java, Scala, Python and R. Python is well known for being an easy to learn programming language, which has been a decisive factor when we chose the tool we would use. Spark can be used in practically any platform, such as Hadoop, Mesos, standalone, or in the cloud. Many organizations are using Spark for their business, some of them are the *NASA JPL - Deep Space Network*, *Yahoo!*, *Nokia Solutions and Network*, *eBay Inc.*, *Amazon* and many others.

As we mentioned before, the way data is handle in big data application is conceptually different from classical computing, meaning that in this kind of application the main goal is to parallelize the operations, in order to speed up the way you process some data. This way, given a cluster of computers, if there is a need to accelerate the application (or there is more data to process), with this technology you just have to add more processing power to the cluster, and the parallelization capabilities will handle the extra CPU computational resources. Of course, this is easier said than done, but as we will explain, Spark presents itself as a powerful tool in this kind of computational tasks.

The main idea behind the parallelization proficiency of Apache Spark is the concept of RDD (*Resilient Distributed Dataset*), a fundamental data structure of Spark which is an immutable collection of objects. Each dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster. Decomposing the name RDD:

- ***Resilient***, i.e. fault-tolerant with the help of RDD lineage graph (DAG) and so able to recompute missing or damaged partitions due to node failures.
- ***Distributed***, since data resides on multiple nodes.
- ***Dataset*** represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or a database.

RDD in Apache Spark supports two types of operations: Transformation, functions that take an RDD as an input and produce one or many RDDs as the output, and Actions, return final results of RDD computations. As all the capabilities of Spark and RDDs are quite complex and extensive, we won´t focus any more on this subject, since it lies out of the scope of this project, but there will be more comments about the actual tools that has been used to address our goal.

V. Data processing and analysis for the maintenance procedure

The main purpose of this project is to develop an algorithm that use the accumulated information of previous flights and the maintenance history of the fleet to detect anomalies; that is, behavior different from the expected, and to achieve a more efficient use of parts; which result in cost reduction. To do this, we are going to distinguish among two types of processed variables:

- Type 1: variables that can indicate anomalies.
- Type 2: accumulative or wear variables.

Firstly, we will address the anomaly detection side of the algorithm. The idea is that, for any process, there are some variables that behave following a pattern when the system does act as expected and a deviation from it could

indicate that something is wrong. A good example of this would be the relation between requests per minute and online users in a web page. The expected behavior is that the more online users the more requests per minute the page will handle.

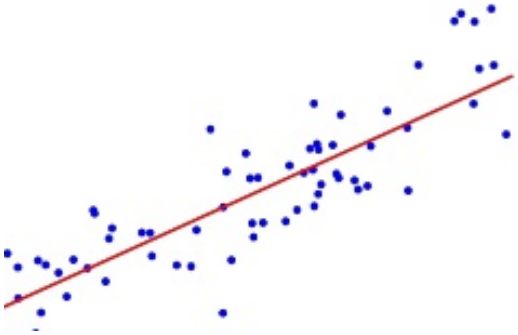

Figure 2. Example of the relation between requests per minute and online users in a webpage.

This can be used to detect anomalies, because if the number of request is above the expected it can be an indicative of malicious behavior of some users, such as the use of bots for some reason; and if it is below the expected it can indicate the page is no longer used as it was in the past; for instance, if the users are bored of the service it provides. This is the idea that it will be used to detect abnormal behavior in the landing gear and its parts, but as this system is quite complex and has many failure modes, modifications and additions will be implemented to indicate where is the failure taking place and how important it is.

Then, it is needed to identify variables or group of variables that are expected to follow a pattern. To do this a previous step could be to analyze data in order to find correlations. This is not the approach that has been followed, as it would have given the correlations that common sense already reveals (for a bigger system, as the whole airplane, this step is possibly mandatory). In our case, the groups and variables that will be used are:

- Impact force – Weight – Velocity (at landing).
- Landing gear retraction time (at takeoff).
- Landing gear extension time (at landing).
- Weight – Shock absorber contraction (before takeoff).
- Maximum breaks temperature – Velocity (after landing).
- Oil level in shock absorber (at landing and takeoff).
- Maximum oil temperature in shock absorber – Impact force (at landing).
- Tyre temperature – Tyre internal pressure (after landing).

Now, as it is necessary to detect when new data does not fit the previous history, we will consider that it follows a normal distribution or multivariate normal distribution when it is a group of variables.

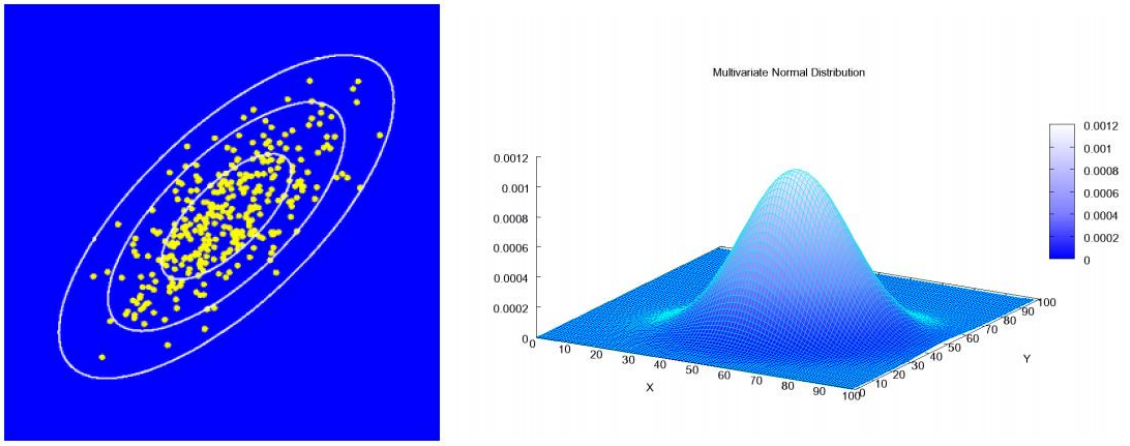

Figure 3. Examples of multivariate normal distributions [Left: survo.fi ; Right: Wikipedia.org]

As we can see in the last image, for a given cloud of points (the data used as history), the properties of the multivariate normal distribution are completely defined, meaning that we can compute the mean, one for each variable, and the covariance matrix, or the variance for just one variable. With that information it is easy to calculate the probability density function (PDF) for a new data point, and compare with the limit that must be stablished in order to spot the anomaly:

$$f_{\mathbf{x}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \leq \delta_i \quad \Rightarrow \quad \boxed{f_i = 1}$$

This means that if the PDF is under a certain value it will be consider an anomaly, and thus, a flag will be set to one for coming calculations. Notice that, graphically, we are considering anomalies the points that lie out of a certain boundary, being this a level curve of the PDF.

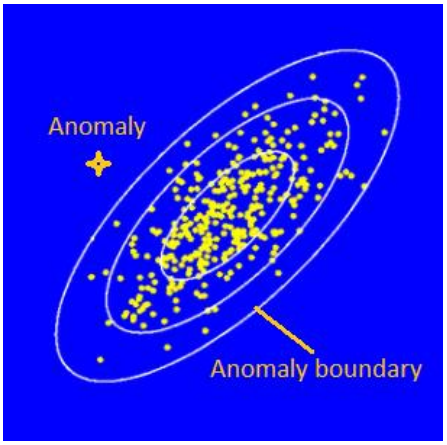

Figure 4. Example of an anomaly in the graphic of a multivariate normal distribution.

There will be one value of δ for each type 1 variable. There are eight variables in this case but the algorithm has no limit. Choosing the exact value of δ is not trivial, as it will determine when something could be considered anomalous or not. A value too high will result in many false positive warnings, and a low one will cover up the real anomalies. Neither of those cases are desirable, but it will be part of the maintenance operators responsibilities to tune the values for the anomaly boundary, as it will be seen from here on out.

The previous one only tells if something is wrong, but it doesn´t give any information about what is going wrong and how serious it is, which is exactly what the maintenance operators really need. To achieve this, we have to correlate the appearance of failure modes with the anomalies, which will be done throw another set of coefficients. For every failure mode there will be as many as type 1 variables:

$$Mode\ 1 \longrightarrow [c_1^1, c_2^1, c_3^1, c_4^1, c_5^1, c_6^1, c_7^1, c_8^1]$$

Each one represents the number of times that an anomaly of its corresponding type 1 variable came with the corresponding failure mode. Of course, for this being representative of the real behavior of the system there has to be as many cases of every failure mode as possible, as the appropriate dataset for a machine learning algorithm is a crucial part for the final performance of it. Using this information is easy to calculate the probability of a failure mode that is happening; or about to happen, if the dataset is good enough:

$$P_{mode1} = [c_1^1 f_1 + c_2^1 f_2 + \cdots + c_8^1 f_8]\frac{1}{\sum_i c_i^1}$$

P_{mode1} is then a statistical indicator of failure mode 1 happening, but it doesn´t tell anything about how serious the possible failure is. For that task, we will use the next function:

$$G_{mode1} = [a_1^1 f_1 + a_2^1 f_2 + \cdots + a_8^1 f_8]\frac{1}{\sum_i f_i}$$

$$a_i^1 = \left(\frac{\delta_i - f_x}{\delta_i}\right)^{1-\frac{c_i^1}{\sum_j c_j^1}}$$

The coefficient a_i^1 tells how important is the deviation in the anomaly i for the failure mode 1, it considers two aspects. The first is the actual difference between the PDF value and the anomaly boundary value, presented inside the parenthesis. This takes into consideration how far are we from the expected. In other words, the further away, the higher a_i^1 will be. The second aspect is the statistical importance of the anomaly group of variables i over the failure mode 1, and for that is the exponent of the parenthesis. The higher c_i^1 is, the higher a_i^1 will be for the same $\delta_i - f_x$.

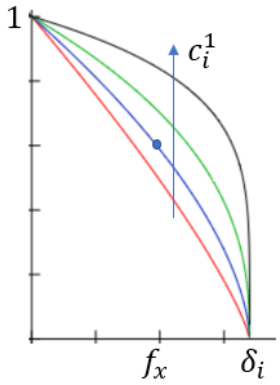

Figure 5. Behaviour of coefficient a_i with c_i.

Once P_{mode1} and G_{mode1} are calculated we have to present the operators some indicator that includes information regarding both the probability of a failure mode happening and the seriousness of it. One possible indicator, and the one that has been chosen is the product $P_{mode1} \cdot G_{mode1}$, which, pretty much like in a risk analysis process, the outcome grows with the likelihood and severity. This process has to be repeated for every failure mode defined in the database, and each time a new flight is asked to be analyzed.

The anomaly detection side of the algorithm reaches this far. There are other methods, some of them simpler and others more sophisticated, but regarding the scope of this project and the time limitations this is what has been chosen as the most appropriate and effective for the original goal. The last part of the algorithm is yet to be explained, and it closes the maintenance process by estimating how much a piece of the landing gear has lived, compared with pasts ones, and how much life is left at current pace of usage.

As mentioned at the beginning of this chapter, this side of the algorithm is after the more efficient use of the pieces that are part of the landing gear, as, traditionally, the maintenance of some of them is only referred to a certain number of cycles of usage, not to the real condition in which they happen to be when the manual says it must be changed. This can be interpreted as a money lost.

In order to do this, the previous maintenance history of the pieces must be considered in some way, but more importantly, we have to consider the real cause for a piece to degrade until it is no longer usable. Common sense tells us that a tyre will degrade with the distance it travels on ground, but not on air. It also reveals that high asphalt temperature will accelerate the process, and friction, and weight, and weather, and accelerations, and probably many more things. It goes the same with every piece of the system, so, whichever approach we chose to determine the actual life a piece has left, it has to be as general as possible, consider that anything can affect a certain piece, even if it has no real effect, and be left with the decision of what is of real importance to calculate the data required.

We have previously talked in this paper about all the data that has been decided to be measured and kept. Using part of that information and processing it, we have come to a type of variable that has been named as accumulative or wear variables (type 2 variables), some examples are as follows:

- Cumulative takeoff distance, in the actual cycle of usage for a specific piece:

$$V_1 = \sum_{\substack{from\ last\ change \\ of\ piece}} TakeoffDistance$$

- Cumulative product of takeoff distance and runway temperature, in the actual cycle of usage for a specific piece:

$$V_2 = \sum_{\substack{from\ last\ change \\ of\ piece}} TakeoffDistance \cdot RunwayTemperature$$

- Cumulative braking slip distance, in the actual cycle of usage for a specific piece:

$$V_3 = \sum_{\substack{from\ last\ change \\ of\ piece}} BrakingSlipDistance$$

Following this schema, we would be able to define almost as many type 2 variables as we wanted:

$$V_4 = Salinity$$
$$V_5 = ParticlesPerMillion$$
$$V_6 = RunwayInclination \cdot Distance$$
$$V_7 = RunwayFrictionCoefficient \cdot Distance$$
$$V_8 = TaxiDistance$$

The more type 2 variables there are, the easier it will be to estimate each piece degradation, and with a higher degree of accuracy. Having defined this, the method would be developed as follows:

- For each piece subject to maintenance:
  - Calculate the value of every type 2 variable in the present cycle of the piece.
  - Calculate the mean and standard deviation for each variable using the maintenance history of every plane of the same type than the one that is being analyzed.

Maintenance history for piece 1
-Previous cycles (include all the aircraft of the same type):
$[12, 15, \dots, 10, 11, 14]$
-Present cycle: 5

Figure 6. Example of maintenance history for piece 1.

  - The last has three outcomes for every type 2 variable, in case of piece 1:

$$(V_1^1, \mu_1^1, \sigma_1^1)$$
$$(V_2^1, \mu_2^1, \sigma_2^1)$$
$$\dots$$
$$(V_n^1, \mu_n^1, \sigma_n^1)$$

  - Finally, this information can be used to calculate the statistical life of the piece using the following:

$$V_{piece1} = f(V_1^1, \mu_1^1) \cdot K_1^1 + \cdots + f(V_n^1, \mu_n^1) \cdot K_n^1$$

$$K_1^1 = \frac{1/(\sigma_1^1/\mu_1^1)}{\sum_j 1/(\sigma_j^1/\mu_j^1)}$$

$$f(V_1^1, \mu_1^1) = \begin{cases} 100\dfrac{V_1^1}{\mu_1^1}\ ;\ if\ V_1^1 \le \mu_1^1 \\ 100 + 20 \cdot \sqrt{\dfrac{V_1^1 - \mu_1^1}{V_1^1 + \mu_1^1}}\ ;if\ V_1^1 > \mu_1^1 \end{cases}$$

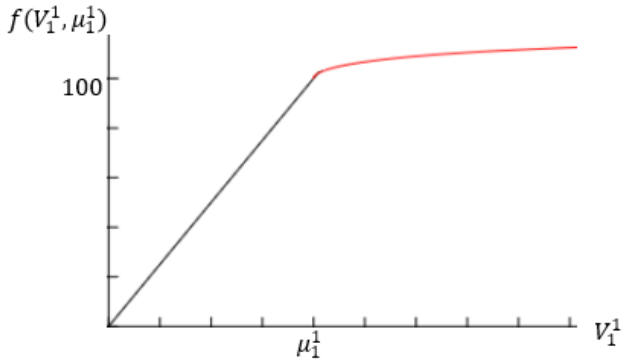

Figure 7. Evolution of the contribution of each variable to the life percentage with the value relative to the mean.

As can be seen in the formula of V_{piece1} , the value of life assigned to a piece is a weighing among what every variable tells about itself. If the value of V_1^1 is halfway to μ_1^1, then the function f will return a 50% of the life, but the actual fraction of this that is taken into account depends on how regular the maintenance of the piece 1 is in relation with V_1^1. This is considered using the value σ_1^1, normalized using the mean, or its inverse, resulting in a higher fraction when this standard deviation is low. That means that the piece is changed or maintained very regularly with the value of V_1^1, even if this hasn't been noticed before.

Then, this method does exactly what we wanted it to do: to consider everything as a wear factor for a piece and value more what has been, statistically more important for that piece in particular. The final value is an estimation of the percentage of life lived by the piece, and the number of cycles left can also be estimated:

- Using the variable with the lower σ/μ (the more important for the piece) calculate:

$$C_{left} = C_{present}\left(\frac{\mu}{V + 2\sigma} - 1\right)$$

Where C_{left} is the estimated number of cycles left and $C_{present}$ the present number of cycles. The last expression is not useful when $C_{present}$ is 0; in other words, when the piece has just been changed or maintained, and there is no information regarding the pace at which it is being used. Notice that, when knowing the value of $C_{present}$ and V, the mean rate at which V grows is given. As for the 2σ in the expression, it is a way of considering the variability of the wearing process, in the safe side, by adding it to V. When V surpasses the value μ the expression will tell that the number of cycles left is negative, which only means that the maintenance routine should focus on inspecting that piece and the ones that also present a similar outcome.

To take advantage of this algorithm and actually achieving a more efficient usage of the pieces, the philosophy of maintenance must be changed a bit; otherwise, the program will tell exactly what it has been done until now. The reason for that is simple: this algorithm is as intelligent as the people that perform the maintenance. Every machine learning algorithm is first trained with a dataset already in your possession, which gives it certain properties; that is, a level of intelligence. Supposing that we used all the data generated until now regarding airplane maintenance and that all that data includes what we need in the calculations, which is not true, and fed it to the defined algorithm, the values returned by it will be far from the actual values of the pieces. This is because, up until now, many pieces are being maintained in a regular basis, not based in their real condition.

On the other hand, this can be progressively fixed, by instructing the algorithm in order to elevate its intelligence. Every time that the operator makes the decision of acting on a piece, and this is registered, it will affect the next time the algorithm is asked about the life of the piece. For this reason, it is very important the operators focus their efforts on discerning the actual condition of the piece, and keep it if the piece can perform without evident risk of failure. Only this way the algorithm will start finding a pattern in the wearing process, and will return a more usage based value of the life. This has to be done carefully, not putting at risk the whole system, but as we have been commenting, there is margin to work with.

VI. User interface

The last step of the whole process described throughout this document is to present the results obtained from the data processing and analysis in a graphic software interface, which is friendlier for the user to visualize. This graphic software interface shows the different parts of the landing gear in three colours depending on the urgency of maintenance. The coulour code chosen is the following one:

- Green: apparently no maintenance would be needed.
- Yellow: an evaluation of the part would be recommended, although change is not compulsory.
- Red: an exhaustive examination and, probably, a change of the part would be needed.

VII. Conclusions

In this project, it has been addressed an innovative approach to the maintenance of aircrafts. This approach based the maintenance performance not in fixed period of times established by the manufacturer, as in the classical approach; but in the actual condition of the different pieces. This condition is evaluated by the algorithm detailed previously. One of the innovative points of that algorithm is that it is constantly improving thanks to the feedback produced in every maintenance decision; in other words, it learns from the experience.

Additionally, this project also combines different fields of study such as the analysis of massive amounts of data, or Big Data, and the detection of anomalies. Trying to deal with this through a classical computer software is difficult due to the immense amount of data and the memory capacity of the computer; nevertheless, thanks to the use of Big Data software tools as Apache Spark, explained in Chapter IV, it has been possible to handle a thousand million pieces of data in just one computer. Moreover, by parallelizing the algorithm and the data in a cluster of several computers the amount of data treated can be increased enormously.

VIII. Next steps

The main purpose of this project was to create a functional interface with its own database and an algorithm implemented to improve the maintenance and make this task easier to workers. However, the great potential of the application is all the upgrades that can be made in a near future, enhancing its capabilities.

- Migration from SQL database to CQL used in Cassandra. The advantage is that implementation in SQL and CQL is formally the same, although internally, datasets are different. Cassandra works with hosts communicated by P2P, without real information in the computer in use, and maximal redundancy.
- Interface improvements to make it friendlier to workers in their day to day tasks.
- Improve the robustness of the algorithm and create complementary ones to improve security and control in maintenance by giving less responsibility to workers and reducing human errors that could cause accidents or, at least, a worse performance.

To conclude, just comment that the algorithm and the user interface can be extrapolated to every single problem related with maintenance, as well as the field of study or the pieces, just having the knowledge of fails and parts of the set. Therefore, here there is definitely a market niche.

References

[1] E. (. N. Ph.D., "Anomaly Detection 101," 2015.

[2] P. A. Ng., "AnomalyDetection - Machine Learning," 2012.

[3] K. G. M. C. K. &. H. H. Mehrotra, "Anomaly Detection Principles and Algorithms," 2017.

[4] Assetivity-Asset Management Consultant, "https://www.assetivity.com," [Online]. Available: https://www.assetivity.com.au/article/maintenance-management/big-data-predictive-analytics-and-maintenance.html.

[5] W. Elrifai, Interviewee, *The Impact of Big Data on the Evolution of Predictive Maintenance.* [Interview]. 30 September 2016.

[6] A. Marfatia, "IBM-Big Data and analysis hub," 14 September 2012. [Online]. Available: http://www.ibmbigdatahub.com/blog/predictive-maintenance-machines-best-friend.

[7] Valmet, "http://www.valmet.com," [Online]. Available: http://www.valmet.com/media/articles/services/more-intelligent-maintenance-with-big-data-analytics/.

[8] B. Rapolu, "http://dataconomy.com," 27 April 2015. [Online]. Available: http://dataconomy.com/2015/04/predictive-maintenance-big-data-on-rails/.

[9] H. Davies, "The rise of 'Big Data' and how it can improve maintenance processes and fleet reliability," 06 April 2016. [Online]. Available: http://blog.rusada.com/blog/the-rise-of-bigdata-and-how-it-can-improve-maintenance-processes-and-fleet-reliability.

[10] N. Agarwal, "The Three Things You Need for a Big Data Project to Succeed," *Big Data Quarterly*, vol. 3, no. 1, 2017.

[11] J. Pilkington, "Diving Into Data Lakes With Self-Service Data Prep," *Big Data Quarterly*, vol. 3, no. 1, 2017.

[12] R. N. C. a. A. V. D. Rajkumar Buyya, Big Data. Principles and paradigms., Elsevier Inc, 2016.

[13] J. Wells, "BIG DATA 50 COMPANIES DRIVING INNOVATION," *Big Data Quarterly*, vol. 2, no. 3, 2016.

[14] F. P. a. T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big Data*, vol. 1, no. 1, pp. 51-59, 2013.

[15] O'Reilly Media, Big Data now, O'Reilly Media, 2016.

[16] M. A. Abbasi, Learning Apache Spark 2. Process big data with the speed of light!, Packt Publishing, 2017.

[17] Apache Spark, "https://spark.apache.org/," [Online]. Available: https://spark.apache.org/.