MBIT SCHOOL

ADVANCED DEEP LEARNING MASTER DISSERTATION

Deep Learning For Predictive Maintenance

Author:
Hugo PÉREZ BLASCO

Supervisor:
Dr. Manuel
SÁNCHEZ-MONTAÑÉS ISLA

August 4, 2020

MBIT SCHOOL

Abstract

Software Engineer

Deep Learning For Predictive Maintenance

by Hugo PÉREZ BLASCO

Predictive maintenance encompasses a variety of topics, including but not limited to: failure prediction, failure diagnosis (root cause analysis), failure detection, failure type classification, and recommendation of mitigation or maintenance actions after failure.

Predictive Maintenance is also a domain where data is collected over time to monitor the state of an asset with the goal of finding patterns to predict failures which can also benefit from certain deep learning algorithms. This study uses simulated aircraft sensor values to predict when an aircraft engine will fail in the future so that maintenance can be planned in advance.

The goal of this dissertation if to do a trade-off over the different deep learning architectures and models that compound the current state of the art concerning the prediction and classification over time series data.

Contents

List of Figures

List of Tables

List of Abbreviations

RUL	**R**emaining **U**seful **L**ife
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
GRU	**G**ated **R**ecurrent **U**nit
MAE	**M**ean **A**bsolut **E**rror
PCA	**P**rincipal **C**omponent **A**nalysis

Chapter 1

Objectives

1.1 Problem Description

Aircraft lifetime and maintenance is a recurrent problem for companies due to its high costs. Being able to predict when an aircraft is going to have failures or malfunctioning is a key topic for improving costs and performance of this machines.

The scenario uses data from simulated aircraft sensor to predict when an aircraft engine will fail in the future. Two different approaches are used for this problem:

- A regression models that manages to answer the question: How many more cycles an in-service engine will last before it fails?
- A binary classification model that manages to answer the question: Is this engine going to fail within a specific number of time cycles?

1.2 Data Summary

The data provided is divided in three different sets:

1.2.1 Training data

The training data consists of multiple multivariate time series with "cycle" as the time unit, together with 21 sensor readings for each cycle. Each time series can be assumed as being generated from a different engine of the same type. Each engine is assumed to start with different degrees of initial wear and manufacturing variation, and this information is unknown to the user. In this simulated data, the engine is assumed to be operating normally at the start of each time series. It starts to degrade at some point during the series of the operating cycles. The degradation progresses and grows in magnitude. When a predefined threshold is reached, then the engine is considered unsafe for further operation. In other words, the last cycle in each time series can be considered as the failure point of the corresponding engine. Taking the sample training data shown in the following table as an example, the engine with id=1 fails at cycle 192.

1.2.2 Test data

The testing data has the same data schema as the training data. The only difference is that the data does not indicate when the failure occurs (in other words, the last time period does NOT represent the failure point). Taking the sample testing data shown in the following table as an example, the engine with id=1 runs from cycle 1 through cycle 31. It is not shown how many more cycles this engine can last before it fails.

TABLE 1.1: Training Data

Id	Cycle	Setting 1	Setting 2	S1	S2	S3	...
1	1	-0.0007	-0.0004	100.0	518.67	641.82	...
1	2	0.0019	-0.0003	100.0	518.67	642.15	...
1	3	-0.0043	0.0003	100.0	518.67	642.35	...
...	...						...
1	191	0	-0.0004	100.0	518.67	643.34	...
1	192	0.0009	0	100.0	518.67	643.54	...
2	1	-0.0018	0.0006	100.0	518.67	641.89	...
2	2	0.0009	-0.0003	100.0	518.67	641.82	...
2	3	0.0018	0.0003	100.0	518.67	641.55	...
...	...						...

TABLE 1.2: Test Data

Id	Cycle	Setting 1	Setting 2	S1	S2	S3	...
1	1	0.0023	0.0003	100.0	518.67	643.02	...
1	2	-0.0027	-0.0003	100.0	518.67	641.71	...
1	3	0.0003	0.0001	100.0	518.67	642.46	...
...	...						...
1	30	-0.0025	0.0004	100.0	518.67	642.79	...
1	31	-0.0006	0.0004	100.0	518.67	642.58	...
2	1	-0.0009	0.0006	100.0	518.67	641.89	...
2	2	-0.0011	0.0002	100.0	518.67	642.51	...
2	3	0.0002	0.0003	100.0	518.67	642.58	...
...	...						...

TABLE 1.3: Ground Truth Data

RUL
112
98
69
82
91
...

1.2.3 Ground truth data

The ground truth data provides the number of remaining working cycles for the engines in the testing data. Taking the sample ground truth data shown in the following table as an example, the engine with id=1 in the testing data can run another 112 cycles before it fails.

1.3 Dissertation goals

Once the data and the problem to solve is known, the goal of this dissertation is to expose and analyze the different approaches and alternatives used for the problem resolution.

All of the different methodologies, technical solutions and results will be explained step by step and compared within each other.

Finally some conclusions must be extracted in order to clarify which one is the best approach to solve the problem concerning this dissertation context.

Chapter 2

Methodology and Techniques

2.1 Deep Learning for sequence processing

This dissertation is focused on the use of Deep Learning techniques to solve predictive maintenance problems.

Some of this techniques are going to be used to solve the problem exposed in chapter 1.

This methodologies and techniques are chosen based on the state-of-art of the topic and also on its frequency of use.

2.1.1 Study methodology

The methodology for the study follows the next steps:

- Data analysis and pre-processing.
- Technical solution and model definition.
- Model and data visualization.
- Analysis of the results.
- Hyper-parameters and model adjustment
- Conclusions

2.1.2 Trade-off

The results of each approach and technique will be compared in a trade-off extracting conclusion about its feasibility to solve the exposed problem.

Also, future directions and improvements will be discussed in agreement with the results of the dissertation.

Chapter 3

Data Analysis and Pre-processing

3.1 Data labelling

The first step is to generate the labels required to train the model. Two different labels must be generated for each question we want to answer:

- Remaining Useful Life (RUL) for the regression problem.
- Boolean flag indicating if the RUL is less than an specific cycle value.

3.1.1 Remaining Useful Life

In order to face the regression model to answer the question *how many more cycles an in-service engine will last before it fails?* each entrance of the train dataset must include the number of remaining cycles until the aircraft fails.

This way, all the rows associated with the same aircraft identifier will have a decreasing number for this field. In the last step, where the cycle number represents the moment the aircraft fails, the RUL must be 0. See table 3.1

3.1.2 Failure Cycle Window

In the case of the binary classification model, the question to answer is *is this engine going to fail within w1 cycles?*.

The w1 represents the window of cycles that we want to take in account to classify. In this dissertation the number of cycles will be set to 30.

TABLE 3.1: Training Data with RUL column

Id	Cycle	Setting 1	Setting 2	S1	...	RUL
1	1	-0.0007	-0.0004	100.0	...	191
1	2	0.0019	-0.0003	100.0	...	190
1	3	-0.0043	0.0003	100.0	...	189
...	...					...
1	191	0	-0.0004	100.0	...	1
1	192	0.0009	0	100.0	...	0
2	1	-0.0018	0.0006	100.0	...	286
2	2	0.0009	-0.0003	100.0	...	285
2	3	0.0018	0.0003	100.0	...	284
...	...					...

TABLE 3.2: Training Data with label for classification

Id	Cycle	Setting 1	Setting 2	S1	...	RUL	label1
1	1	-0.0007	-0.0004	100.0	...	191	0
1	2	0.0019	-0.0003	100.0	...	190	0
1	3	-0.0043	0.0003	100.0	...	189	0
...	...						...
1	191	0	-0.0004	100.0	...	1	1
1	192	0.0009	0	100.0	...	0	1
2	1	-0.0018	0.0006	100.0	...	286	0
2	2	0.0009	-0.0003	100.0	...	285	0
2	3	0.0018	0.0003	100.0	...	284	0
...	...						...

Along with the RUL, it is possible to generate a label (label1) for each entrance indicating if the aircraft failed on this cycle window. To calculate it, just set to 1 ("true") all the RUL values lesser than w1. See table 3.2

3.2 Feature Engineering and Normalization

The information of the aircraft for each time step is formed by 21 different sensor and 3 setting numerical values. The distribution of this values is shown in the figure 3.1.

The data distribution shows that most of the sensor data follows a gaussian distribution, avoiding the existence of outliers. With this premise and also taking in account that some of the value ranges can take negative values, the normalization method chosen is min max scaler.

Min max scaler normalization transforms the values in the distribution to the range [0-1]. This normalization is required to avoid the network to priorize on features with higher values and also to avoid exploding gradient problems.

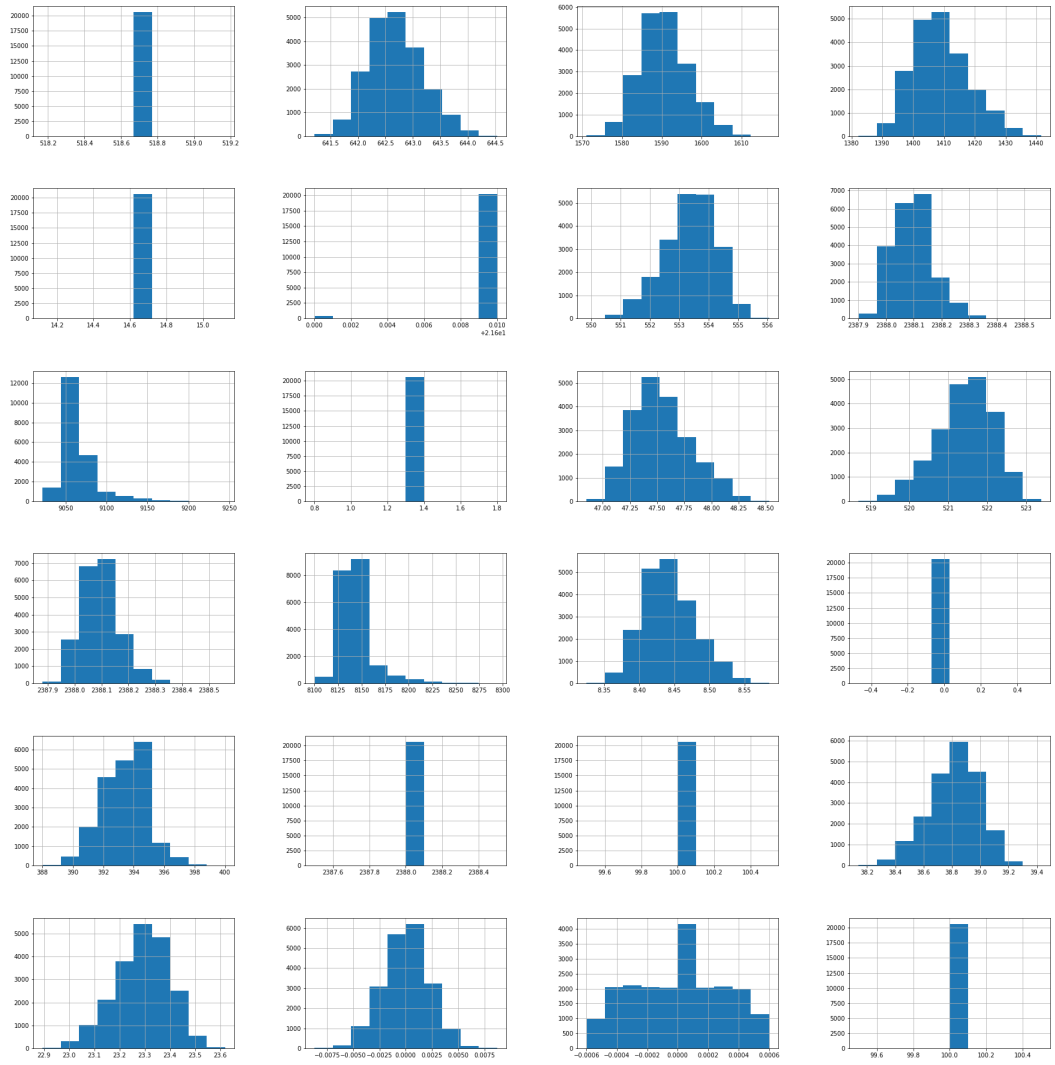

FIGURE 3.1: Sensor data distribution

Chapter 4

Long Short-Term Memory

4.1 Understanding LSTMs

Long Short Term Memory networks – usually just called *LSTMs* – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997).

LSTMs are explicitly designed to avoid the long-term dependency problem. This allows the network to remember information for long periods of time.

All recurrent neural networks have the form of a chain of repeating modules of neural network. But the core of the LSTMs is the *cell state*, that allows the network to keep a simple state information through the processing of all the time steps in the chain. An scheme of a LSTM architecture is described in figure 4.1.

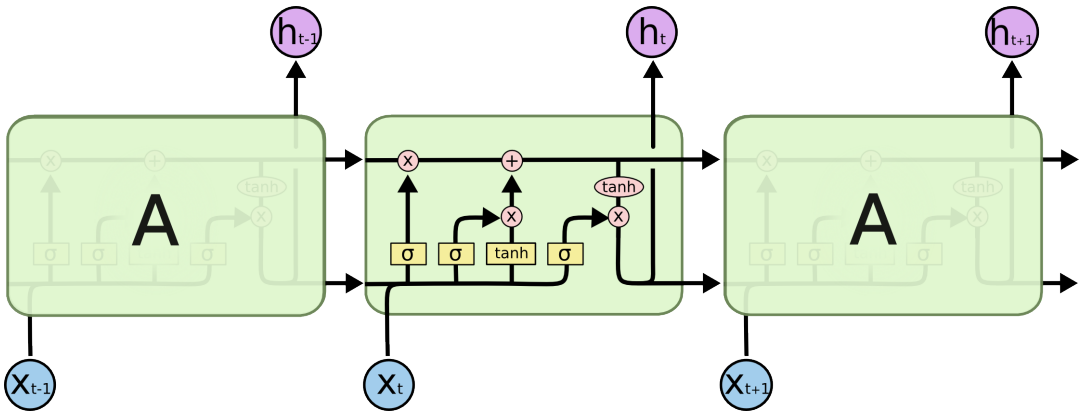

FIGURE 4.1: LSTM architecture.

4.2 Binary Classification

The next sections describe the methodology applied to solve the binary classification problem using LSTMs

4.2.1 Data Generation

Before the ingestion of the data into the LSTM model, it has to be adapted to the architecture of the network.

Being *samples* the total number of data, *timesteps* the amount of rows to be ingested for each aircraft and *features* the number of sensor data the input of the LSTM layer is defined by a tensor of shape:

```
(samples, timesteps, features)
```

The time steps used for the study will be 50, which means that for each aircraft batches of 50 time steps will be taken. Using the *zip* function over the data of the aircraft with id=1, which contains 192 cycles. The batches will be distributed this way:

```
(0, 50)     -> from row 0 to row 50
(1, 51)     -> from row 1 to row 51
(2, 52)     -> from row 2 to row 52
...
(111, 192)  -> from row 111 to row 192
```

After the pre-processing, the input tensor will have the shape:

```
(15631, 50, 25)
```

To generate the labels, the values of the column *label 1*, which was generated on the data preparation (see chapter 3), must be grouped by aircraft indicating if the engine failed within *w1* cycles.

The result label tensor has the shape:

```
(15631, 1)
```

4.2.2 Model definition

As explained above, the architecture of the neural network is based on the use of LSTM layers. In this model two different LSTM layers are used to achieve better results.

To accomplish the binary classification, the output layer of the network is a Dense layer with a single output value. This value classifies between two different options: the aircraft will fail or not within the specified cycle.

To prevent over-fitting, two Dropout layers are concatenated after the output of both LSTM layers. This adds a regularization method that drops random units after the LSTM layers, avoiding the network to learn interdependent set of feature weights.

Finally, the loss function selected for this kind of classification problem is *binary crossentropy* using *Adam* as optimizer.

The diagram for the neural network is described in figure 4.2

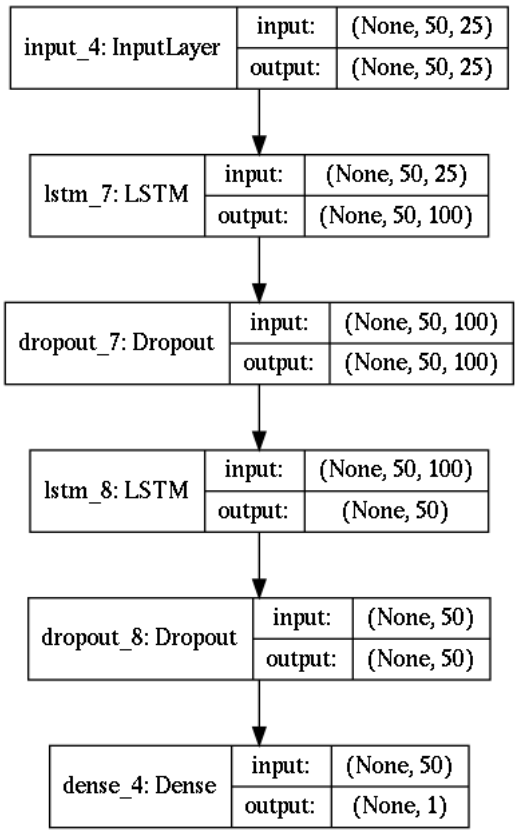

FIGURE 4.2: Neural Network model for LSTM binary classification.

4.2.3 Training visualization

Visualizing the output of the different layers of a network is one of the most useful tools to analyze the network performance. It could also help us to determine if the data analysis has been done correctly and allow us to see possible biases over the data distribution.

For the current network, the output of the last LSTM layer has been taken in account to do this analysis. The output of this layer is a tensor of size 50, which represents the hidden units of the LSTM. To extract the information contain by those celds, the *Principal Component Analysis* (PCA) function has been chosen.

PCA is a mathematical function that allows us to summarize huge features sets. For this dissertation, two principal components has been configured. In figure 4.3 you can see a graphical representation of this components.

In this figure we can clearly see how the values conforms to main clusters of data. This has a lot of sense since we are trying to classify the data in two different groups.

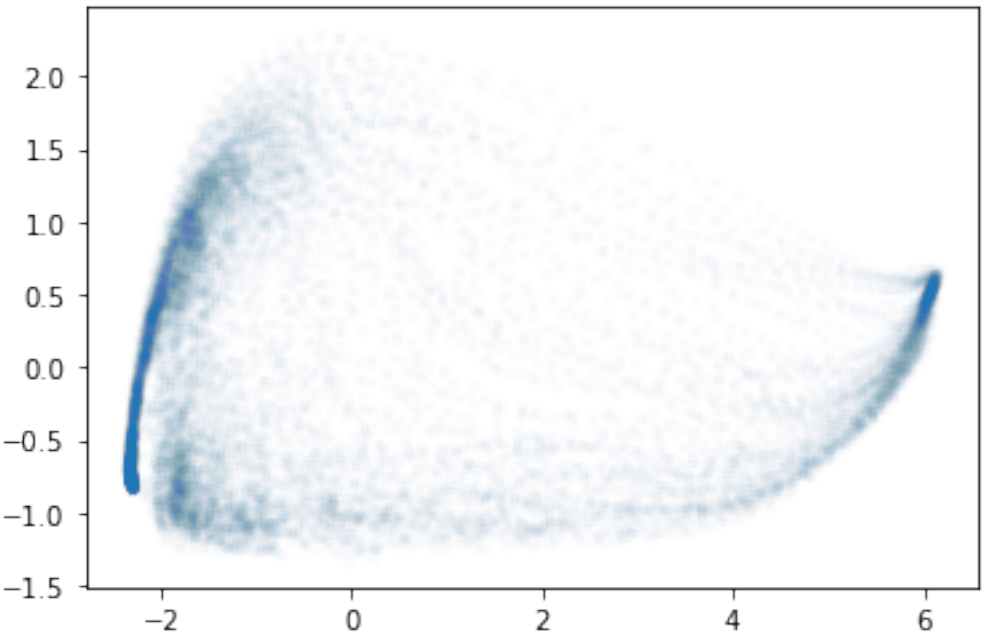

FIGURE 4.3: PCA applied over LSTM output.

4.2.4 Results and model evaluation

After the model is trained using batches of data of size 200. The validation split selected for the data is 5%.

The evolution of the loss and accuracy during the model training can be seen in the figures 4.4 and 4.5

The final accuracy obtained is 98% so we can conclude that the network works very well for this task. To evaluate the results against the ground truth data a comparison graph has been generated. This graph can be seen in figure 4.6.

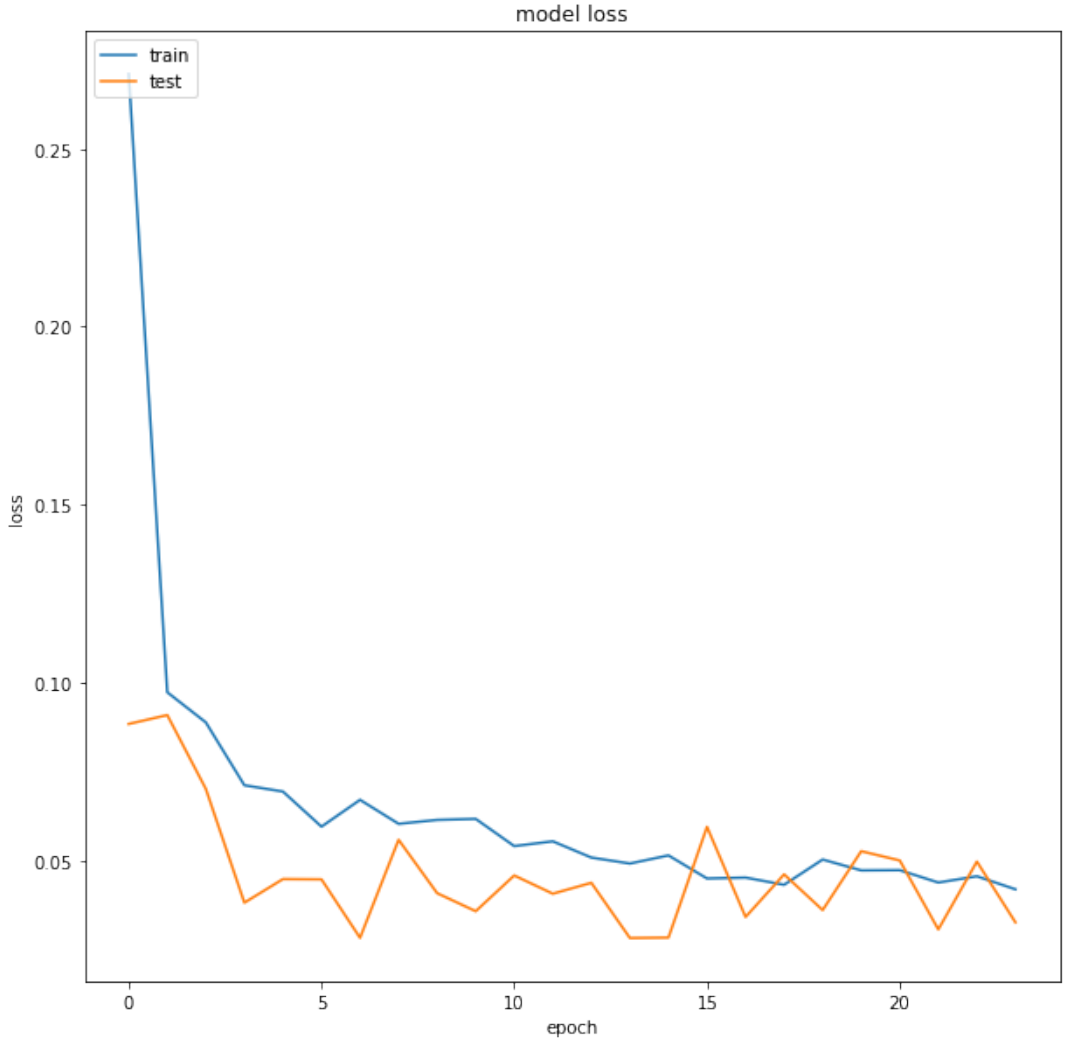

FIGURE 4.4: LSTM Binary Classification Model Loss.

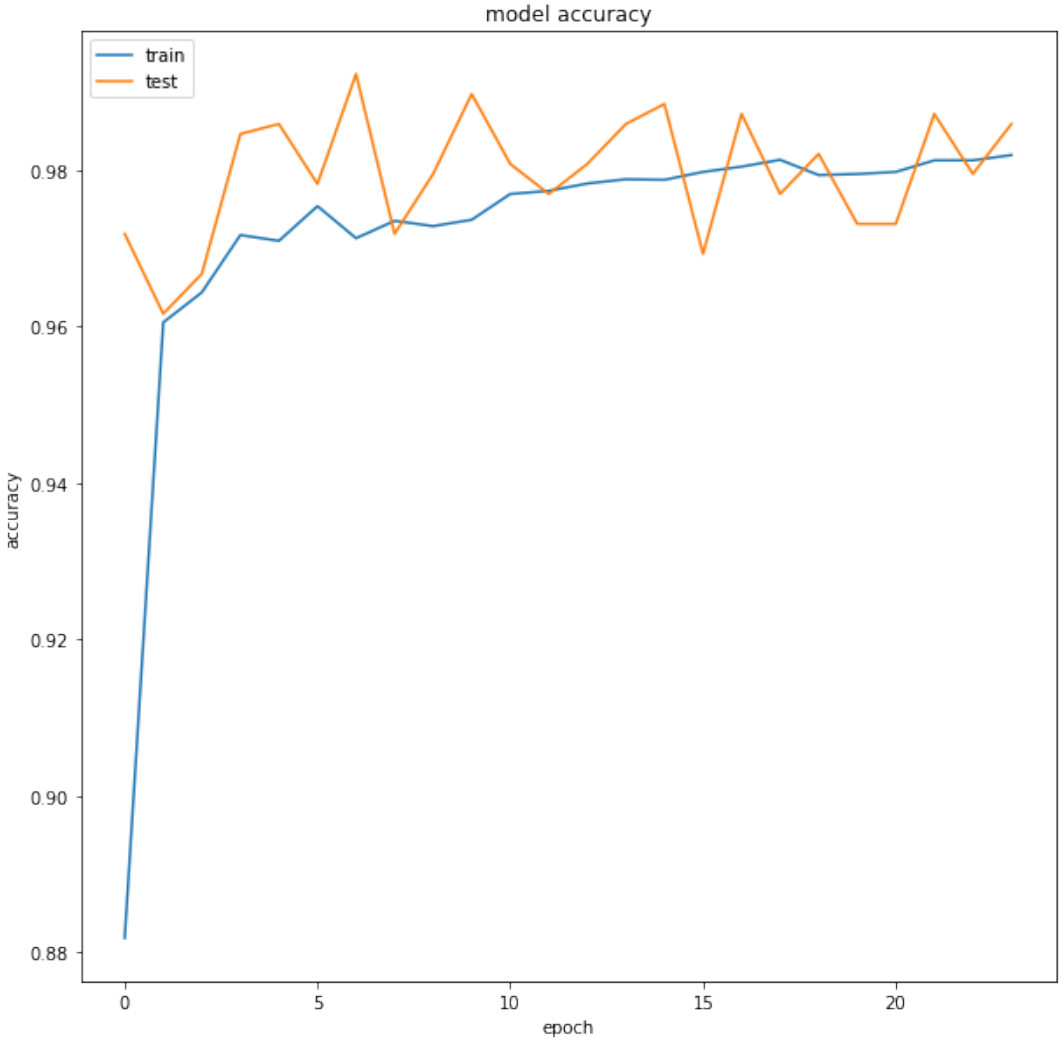

FIGURE 4.5: LSTM Binary Classification Model Accuracy.

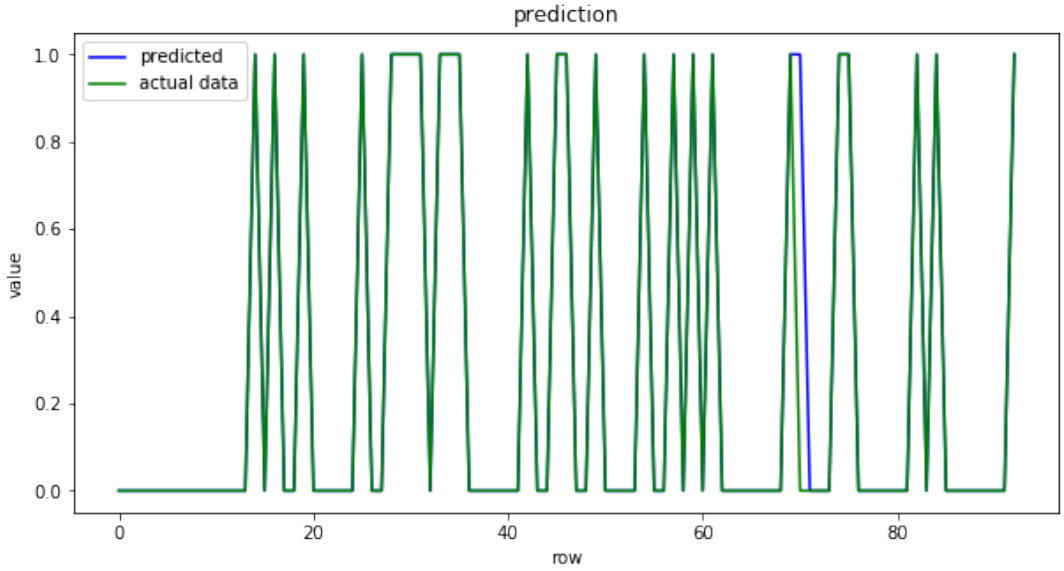

FIGURE 4.6: Result comparison between Binary LSTM Model and Ground Truth data.

4.3 Regression Model

The next sections describe the methodology applied to solve the regression problem using LSTMs

4.3.1 Data Generation

Before the ingestion of the data into the LSTM model, it has to be adapted to the architecture of the network.

Being *samples* the total number of data, *timesteps* the amount of rows to be ingested for each aircraft and *features* the number of sensor data the input of the LSTM layer is defined by a tensor of shape:

```
(samples, timesteps, features)
```

The time steps used for the study will be 50, which means that for each aircraft batches of 50 time steps will be taken. Using the *zip* function over the data of the aircraft with id=1, which contains 192 cycles. The batches will be distributed this way:

```
(0, 50)      -> from row 0 to row 50
(1, 51)      -> from row 1 to row 51
(2, 52)      -> from row 2 to row 52
...
(111, 192)   -> from row 111 to row 192
```

After the pre-processing, the input tensor will have the shape:

```
(15631, 50, 25)
```

For this type of regression model the labels are the values of the column *RUL*, which was generated on the data preparation (see chapter 3).

The result label tensor has the shape:

```
(15631, 1)
```

4.3.2 Model definition

As explained above, the architecture of the neural network is based on the use of LSTM layers. In this model two different LSTM layers are used to achieve better results.

To accomplish the binary classification, the output layer of the network is a Dense layer with a single output value. This linear value is generated by the network output and represents the *RUL* that the network calculates for an especific aircraft. Finally, an linear activation function is added to the last layer.

To prevent over-fitting, two Dropout layers are concatenated after the output of both LSTM layers. This adds a regularization method that drops random units after the LSTM layers, avoiding the network to learn interdependent set of feature weights.

The loss function selected for this kind of classification problem is *binary crossentropy* using *RMSProp* as optimizer.

For the regression model the metric used is *MAE*, also, in this dissertation the *Coefficient of Determination* (R squared) metric function has been implemented.

The diagram for the neural network is described in figure 4.7

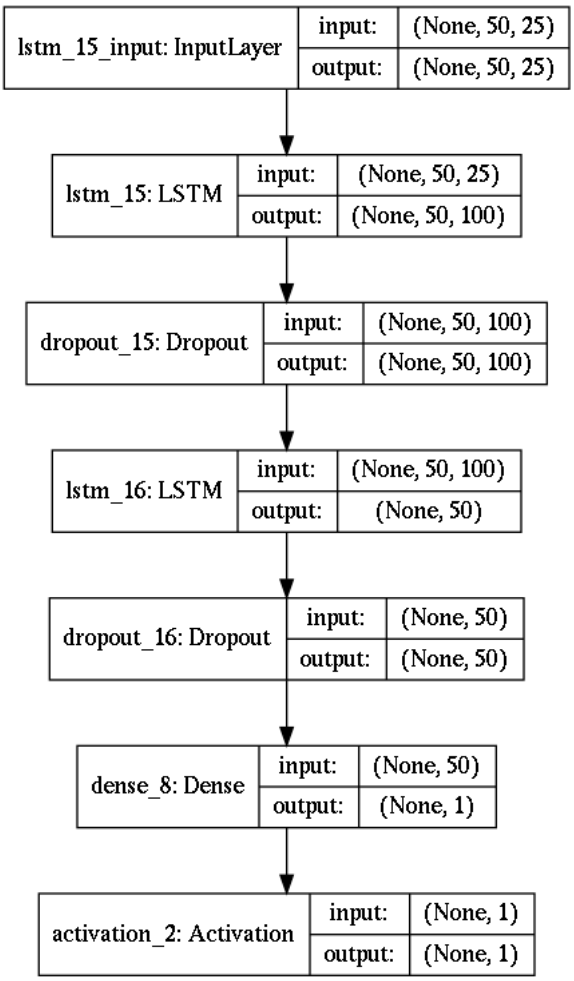

FIGURE 4.7: LSTM Neural Network for regression model.

4.3.3 Results and model evaluation

After the model is trained using batches of data of size 200. The validation split selected for the data is 5%.

The evolution of the loss and metrics during the model training can be seen in the figures 4.8, 4.9 and 4.10

The final results obtained for *MAE* and *R-square* are 11.93 and 0.81 respectively.To evaluate the results against the ground truth data a comparison graph has been generated. This graph can be seen in figure 4.11.

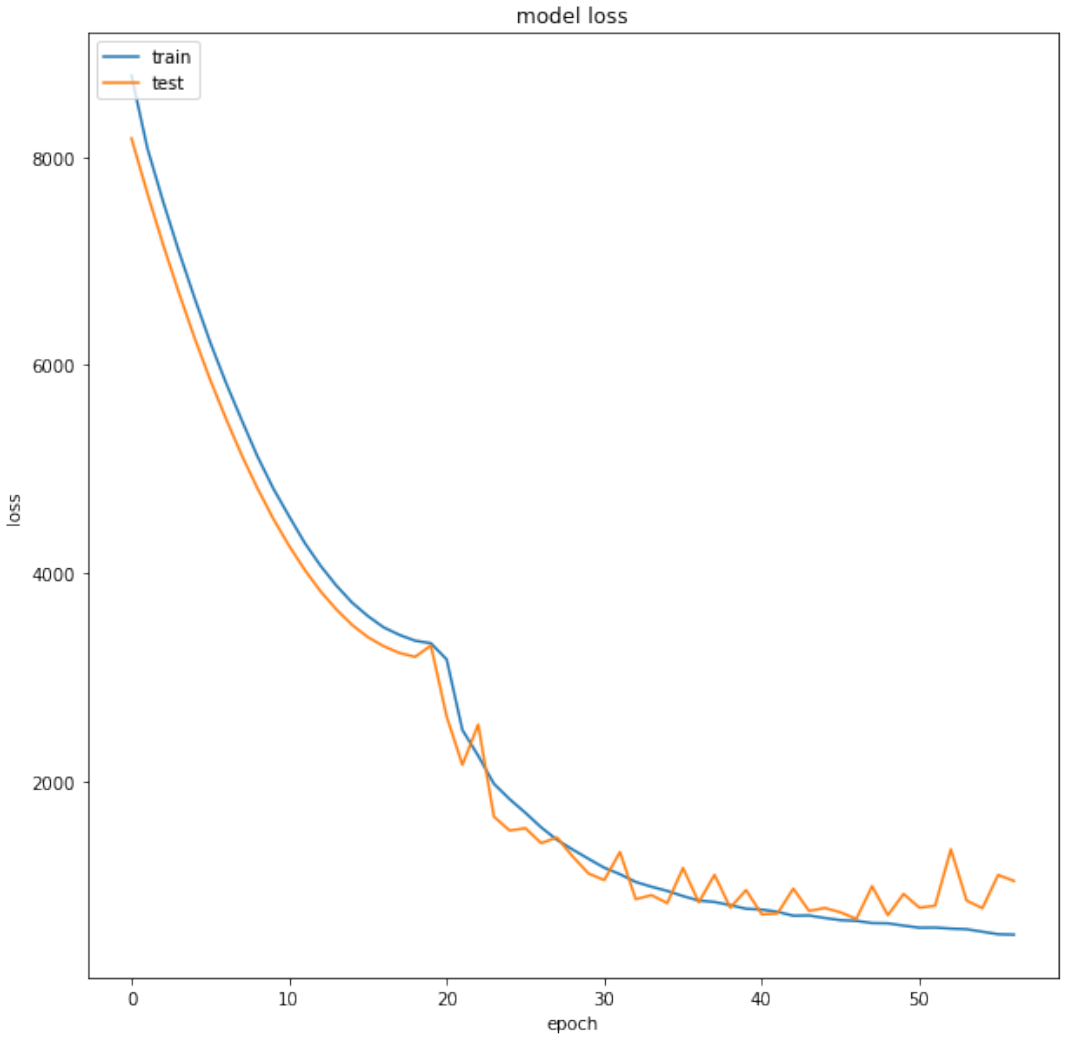

FIGURE 4.8: LSTM Regression Model Loss.

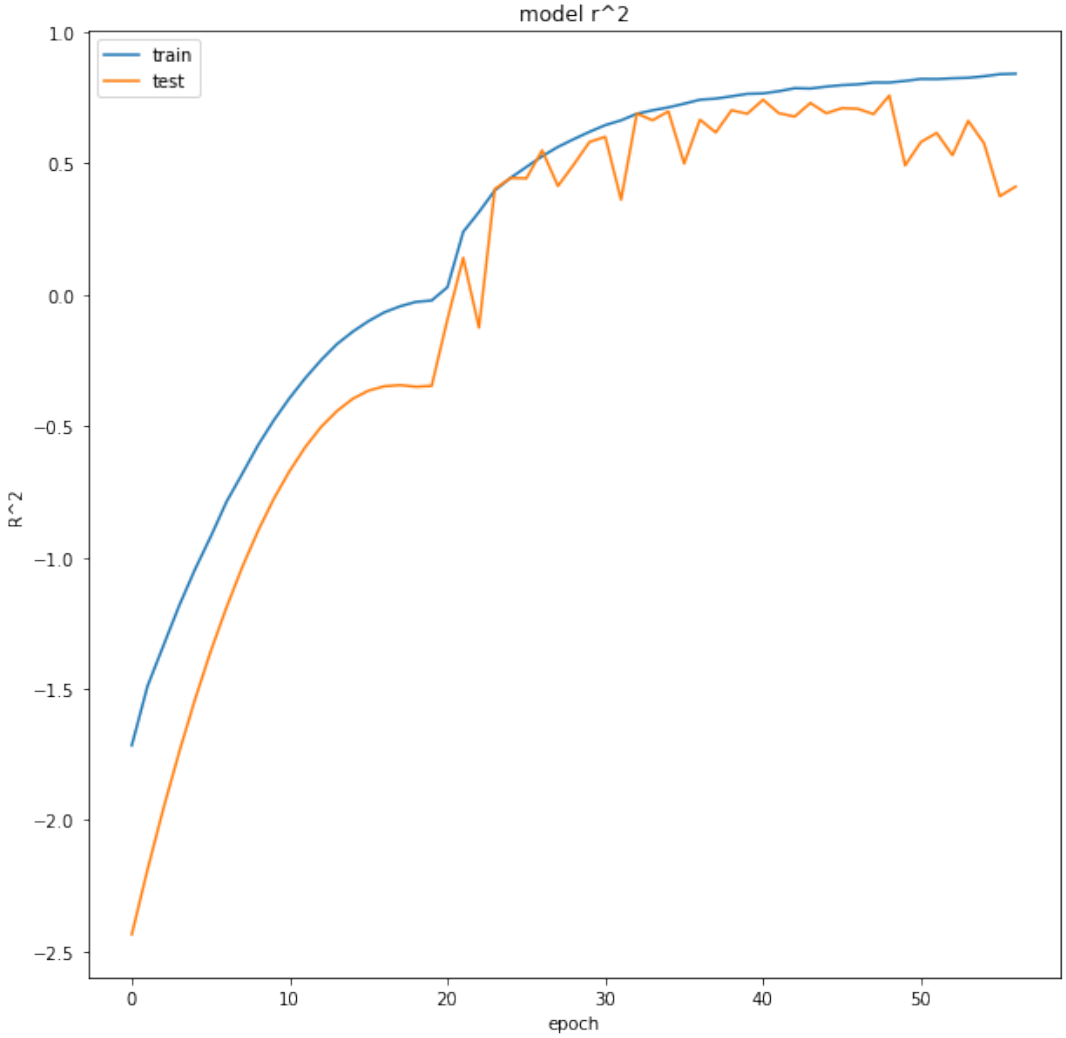

FIGURE 4.9: LSTM R2 model.

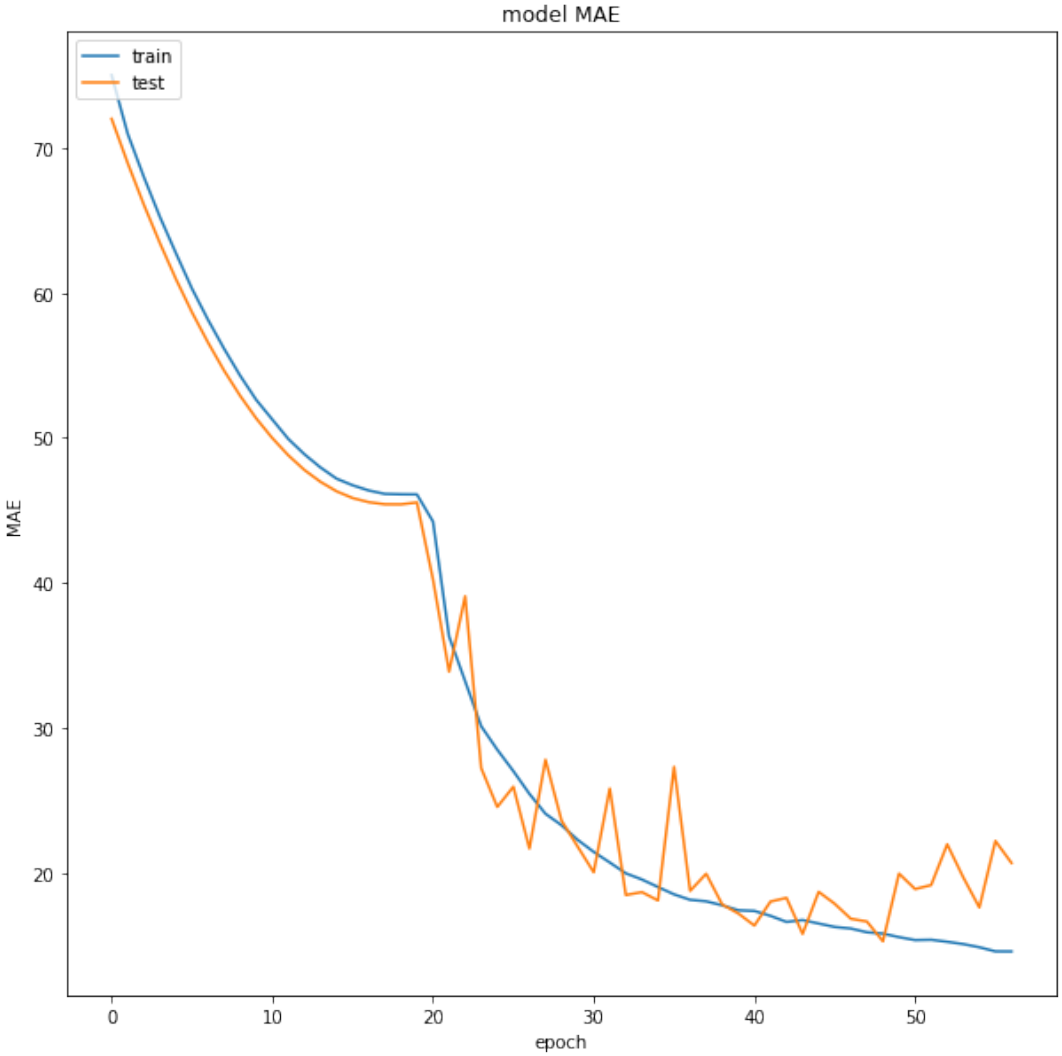

FIGURE 4.10: LSTM MAE model.

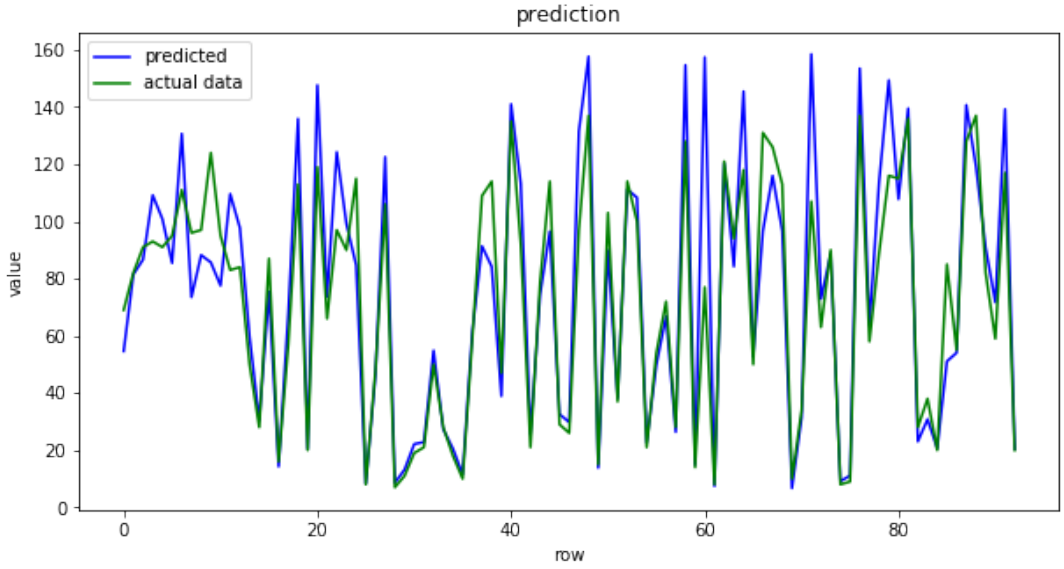

FIGURE 4.11: Result comparison between Regression LSTM Model and Ground Truth data.

Chapter 5

Gated Recurrent Unit

5.1 Understanding GRUs

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al .

The GRU is like a LSTM with a forget gate but has fewer parameters than LSTM, as it lacks an output gate.

An scheme of a GRU architecture is described in figure 5.1.

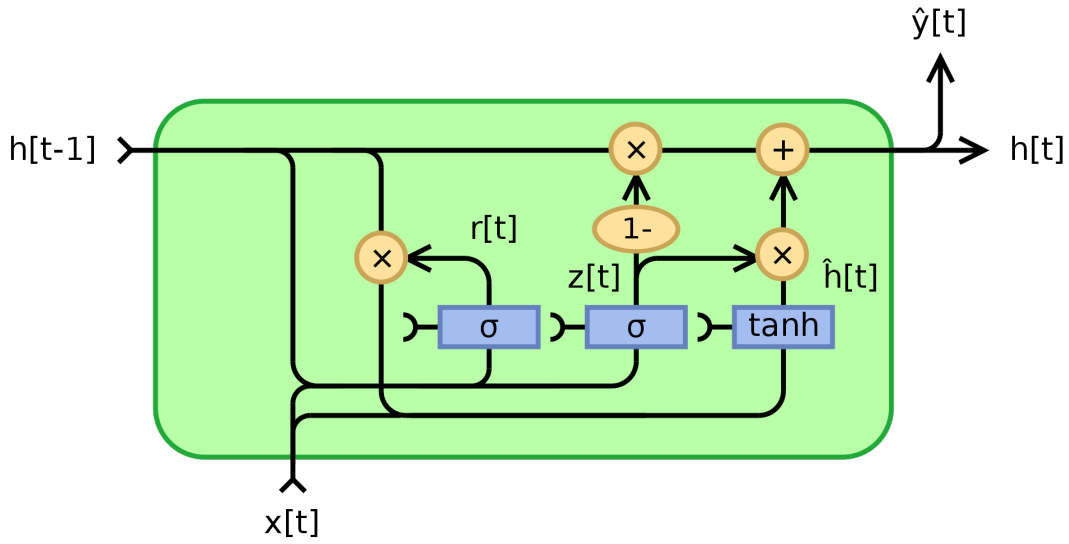

FIGURE 5.1: GRU architecture.

5.2 Binary Classification

The next sections describe the methodology applied to solve the binary classification problem using GRUs.

5.2.1 Data Generation

Before the ingestion of the data into the GRU model, it has to be adapted to the architecture of the network.

Being *samples* the total number of data, *timesteps* the amount of rows to be ingested for each aircraft and *features* the number of sensor data the input of the GRU layer is defined by a tensor of shape:

```
(samples, timesteps, features)
```

The time steps used for the study will be 50, which means that for each aircraft batches of 50 time steps will be taken. Using the *zip* function over the data of the aircraft with id=1, which contains 192 cycles. The batches will be distributed this way:

```
(0, 50)     -> from row 0 to row 50
(1, 51)     -> from row 1 to row 51
(2, 52)     -> from row 2 to row 52
...
(111, 192)  -> from row 111 to row 192
```

After the pre-processing, the input tensor will have the shape:

```
(15631, 50, 25)
```

To generate the labels, the values of the column *label 1*, which was generated on the data preparation (see chapter 3), must be grouped by aircraft indicating if the engine failed within *w1* cycles.

The result label tensor has the shape:

```
(15631, 1)
```

5.2.2 Model definition

This model is based on the previous one that focuses on the use of LSTM layers. This time the same architecture, loss and activation functions are used but all the LSTM layers are switched by GRU layers.

Instead of adding Dropout layers, the dropout field provided by the Keras API implementation of the GRU layers is used.

The diagram for the neural network is described in figure 5.2

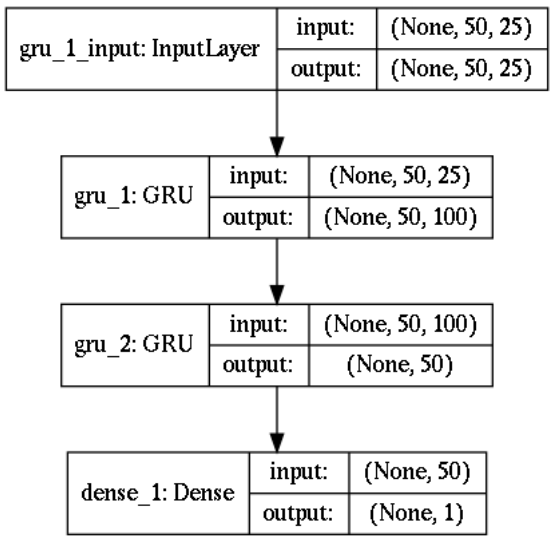

FIGURE 5.2: GRU Neural Network model for binary classification.

5.2.3 Training visualization

As done in the previous model, the output of the second RNN layer has been visualized. In this case the second GRU layer is the one selected. The output of this layer is a tensor of size 50, which represents the hidden units of the GRU. To extract the information contain by those celds, the *Principal Component Analysis* (PCA) function has been chosen.

In figure 5.3 you can see a graphical representation of this components.

As happened with the LSTM model, two separated clusters can be seen in the image. This follows to the same conclusion, the data tends to split into two different options: the aircraft fails in a cycle or it doesn't.

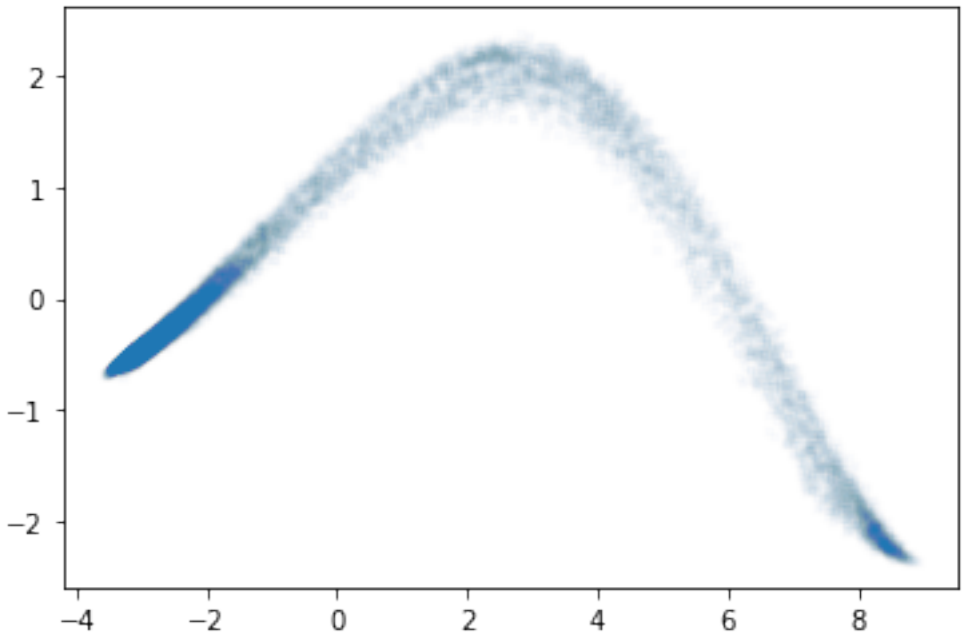

FIGURE 5.3: PCA applied over GRU output.

5.2.4 Results and model evaluation

After the model is trained using batches of data of size 200. The validation split selected for the data is 5%.

The evolution of the loss and accuracy during the model training can be seen in the figures 5.4 and 5.5

The final accuracy obtained is 97% so we can conclude that the network works very well for this task. To evaluate the results against the ground truth data a comparison graph has been generated. This graph can be seen in figure 5.6.

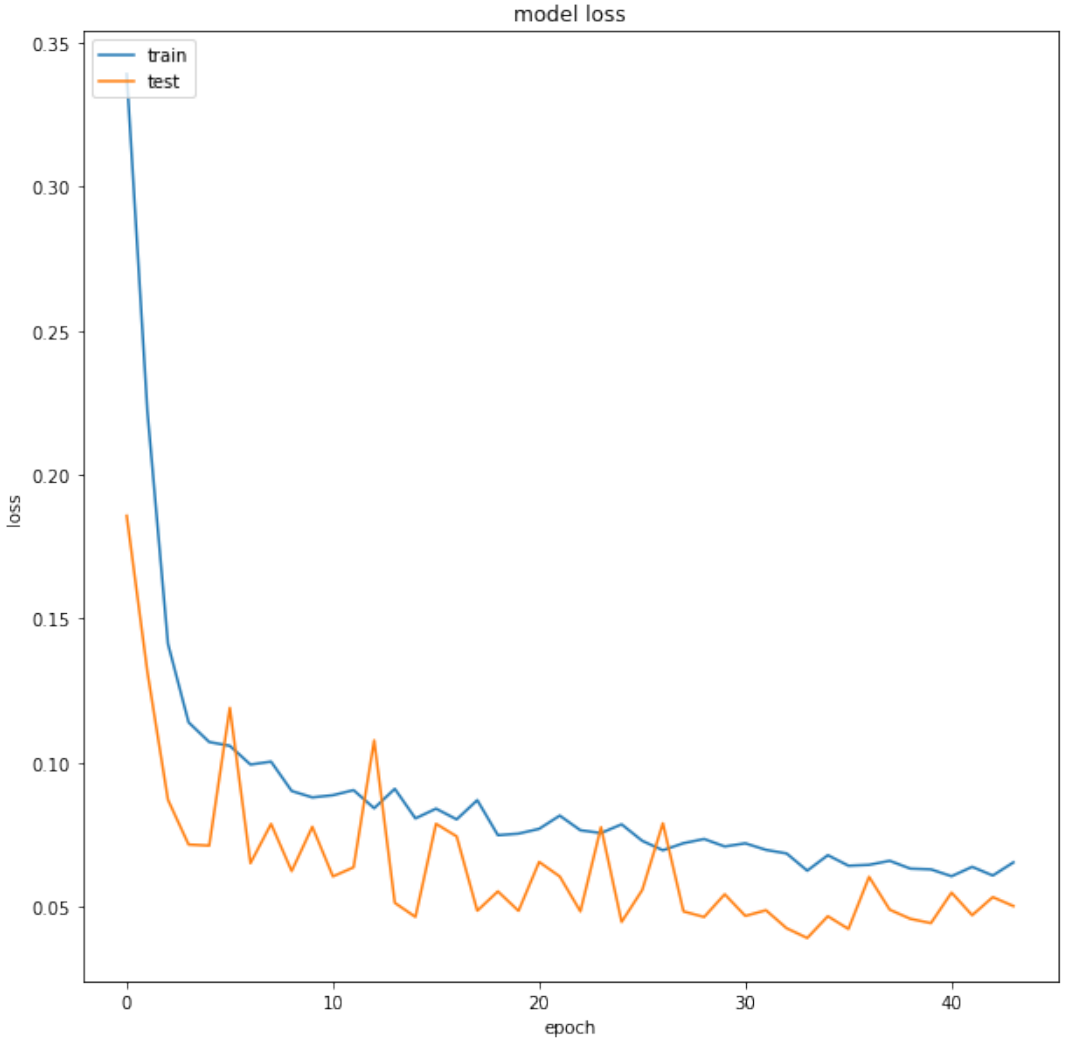

FIGURE 5.4: GRU Binary Classification Model Loss.

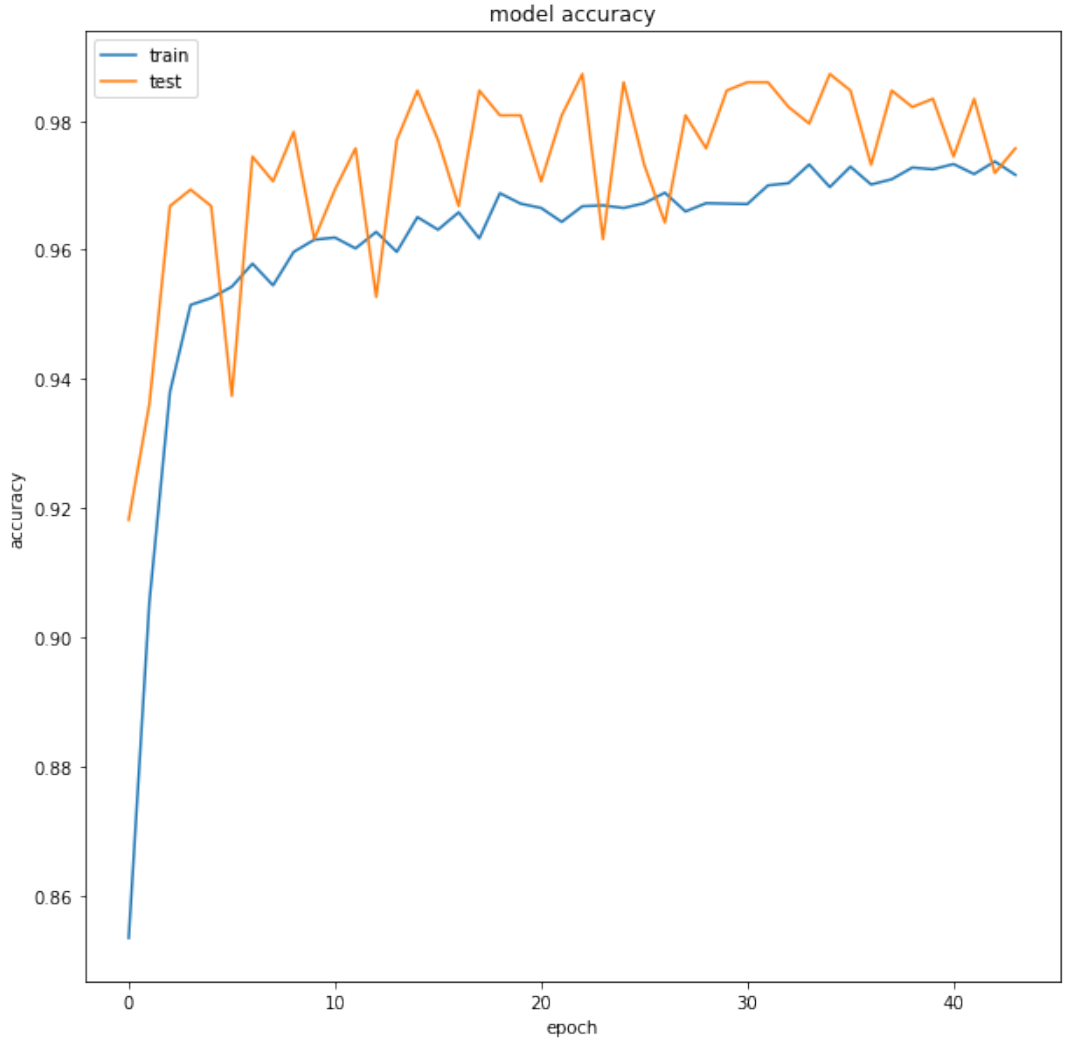

FIGURE 5.5: GRU Binary Classification Model Accuracy.

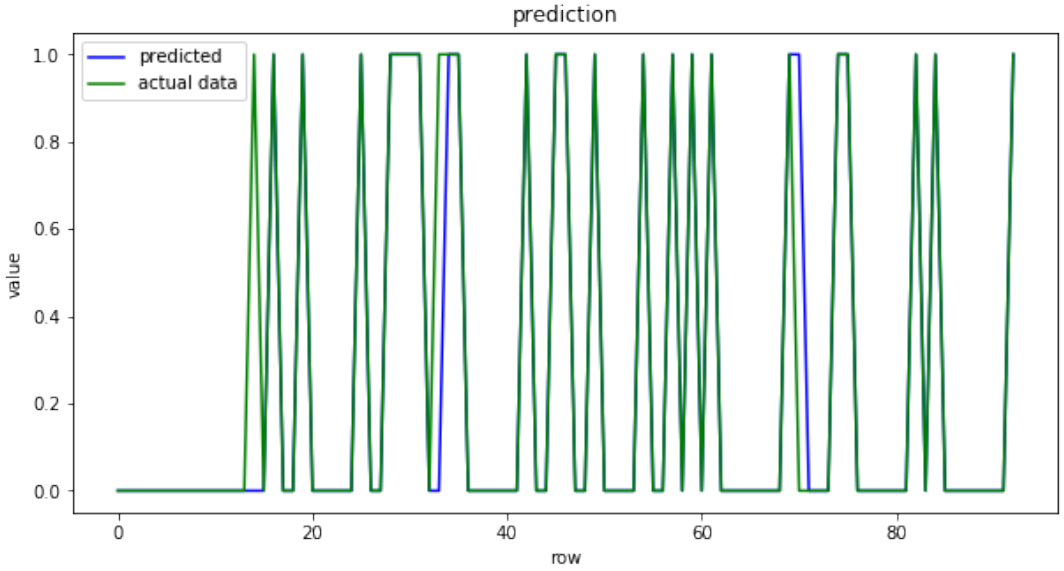

FIGURE 5.6: Result comparison between Binary GRU Model and Ground Truth data.

5.3 Regression Model

The next sections describe the methodology applied to solve the regression problem using GRUs

5.3.1 Data Generation

Before the ingestion of the data into the GRU model, it has to be adapted to the architecture of the network.

Being *samples* the total number of data, *timesteps* the amount of rows to be ingested for each aircraft and *features* the number of sensor data the input of the LSTM layer is defined by a tensor of shape:

```
(samples, timesteps, features)
```

The time steps used for the study will be 50, which means that for each aircraft batches of 50 time steps will be taken. Using the *zip* function over the data of the aircraft with id=1, which contains 192 cycles. The batches will be distributed this way:

```
(0, 50)      -> from row 0 to row 50
(1, 51)      -> from row 1 to row 51
(2, 52)      -> from row 2 to row 52
...
(111, 192)   -> from row 111 to row 192
```

After the pre-processing, the input tensor will have the shape:

```
(15631, 50, 25)
```

For this type of regression model the labels are the values of the column *RUL*, which was generated on the data preparation (see chapter 3).

The result label tensor has the shape:

```
(15631, 1)
```

5.3.2 Model definition

Like the binary one, this model has been created just replacing the LSTMs layers used in the previous chapter with GRU layers. The rest of the components of the model remain the same.

As a point, it is interesting to notice that in order to make the model converge, especific Dropout layers has been used instead that the features provided by the GRU layer API.

The diagram for the neural network is described in figure 5.7

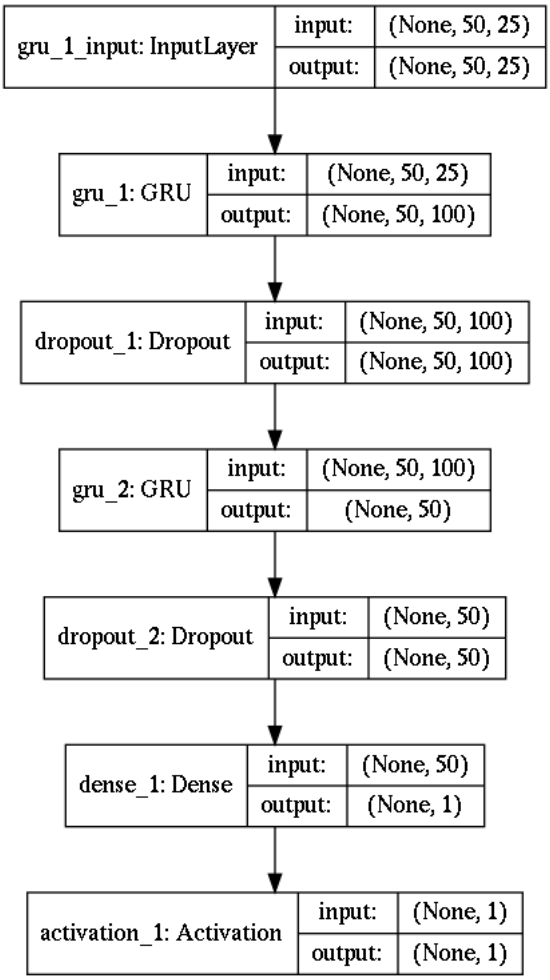

FIGURE 5.7: Neural Network for GRU regression model.

5.3.3 Results and model evaluation

After the model is trained using batches of data of size 200. The validation split selected for the data is 5%.

The evolution of the loss and metrics during the model training can be seen in the figures 5.8, 5.9 and 5.10

The final results obtained for *MAE* and *R-square* are 13.70 and 0.81 respectively. To evaluate the results against the ground truth data a comparison graph has been generated. This graph can be seen in figure 5.11.

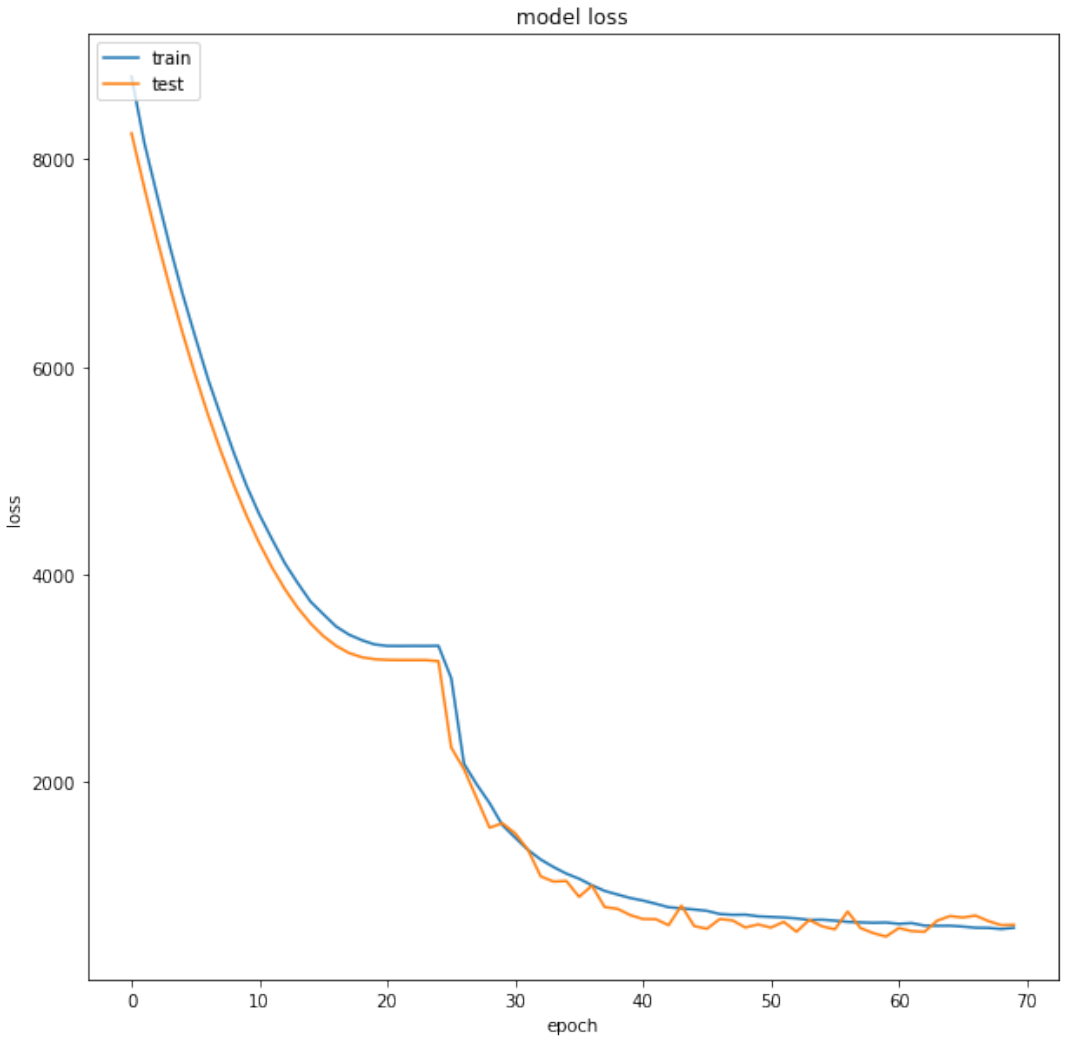

FIGURE 5.8: GRU Regression Model Loss.

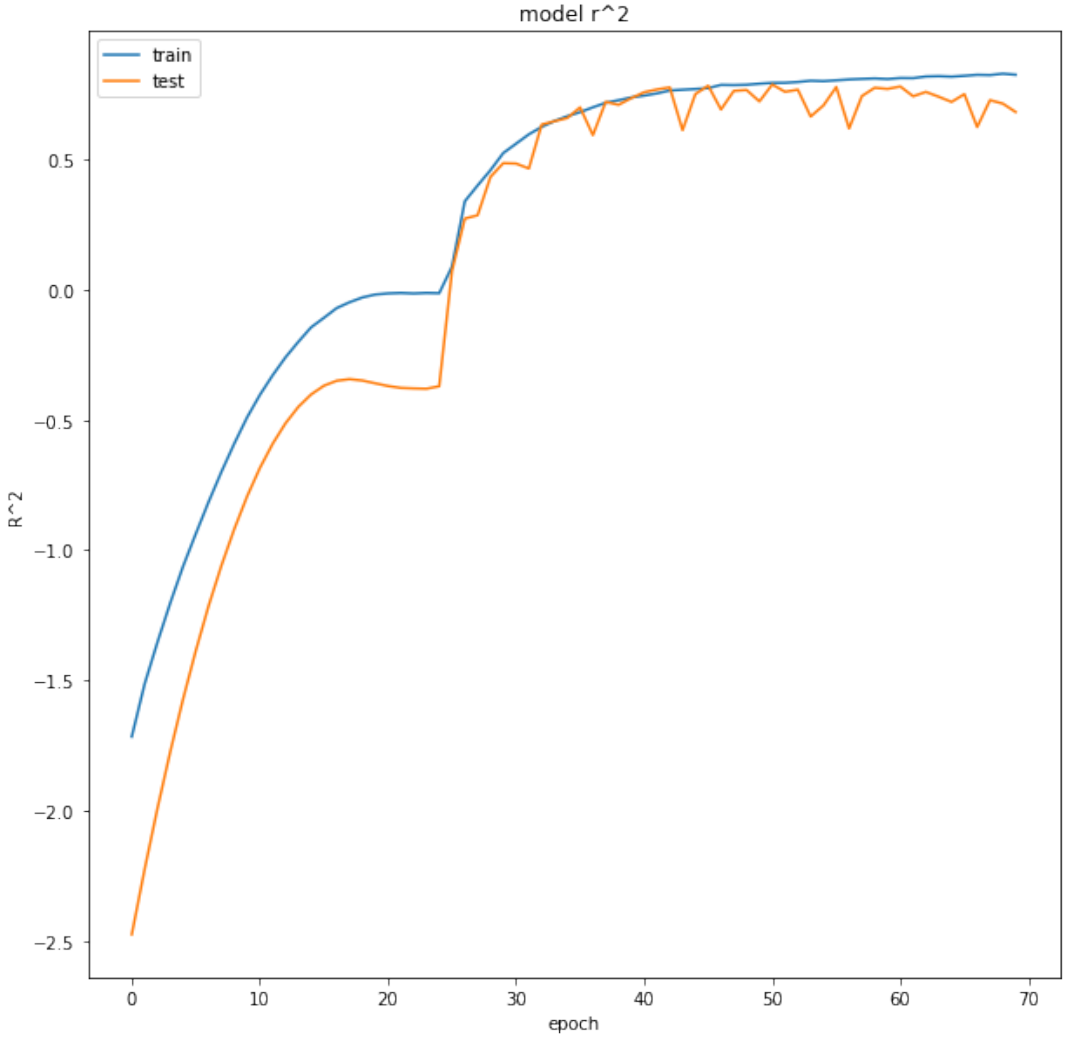

FIGURE 5.9: GRU R2 model.

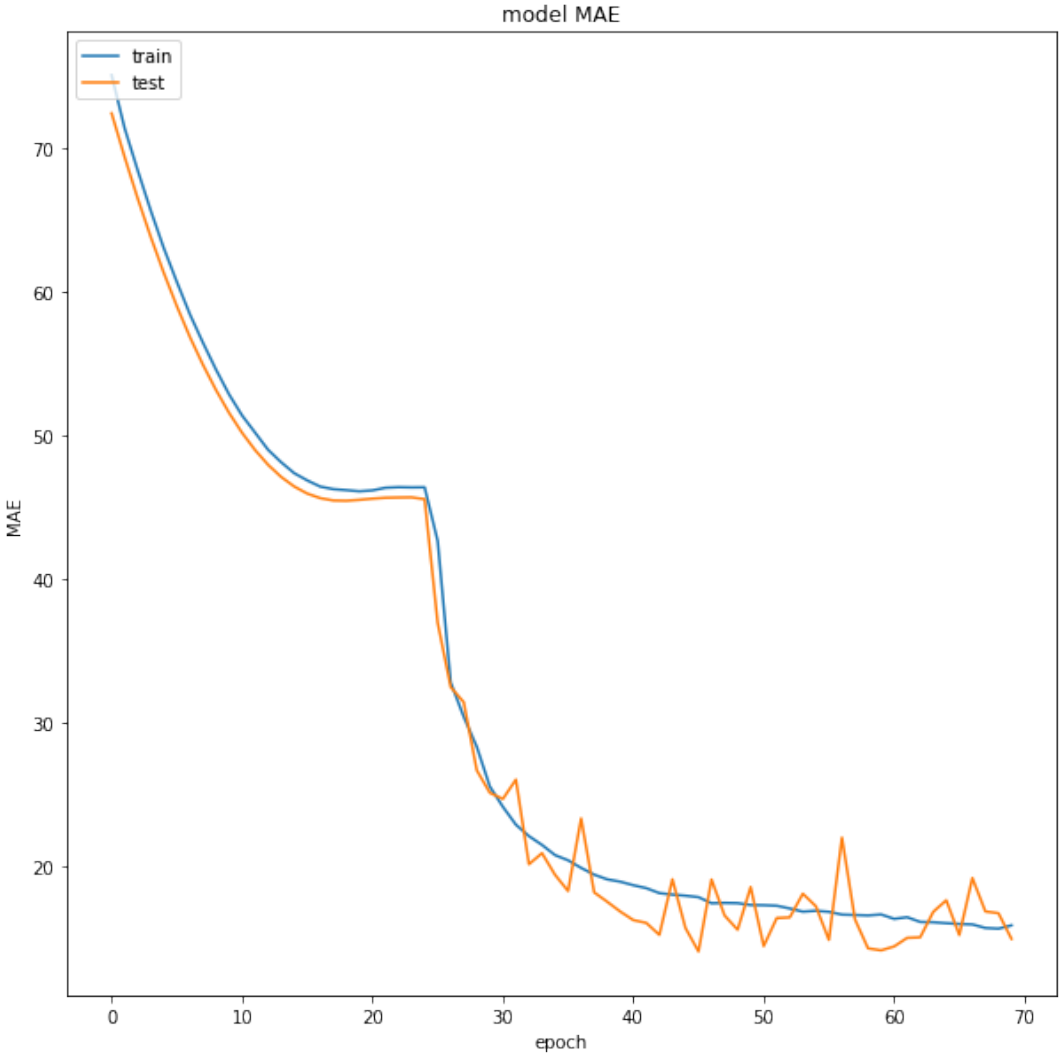

FIGURE 5.10: GRU MAE model.

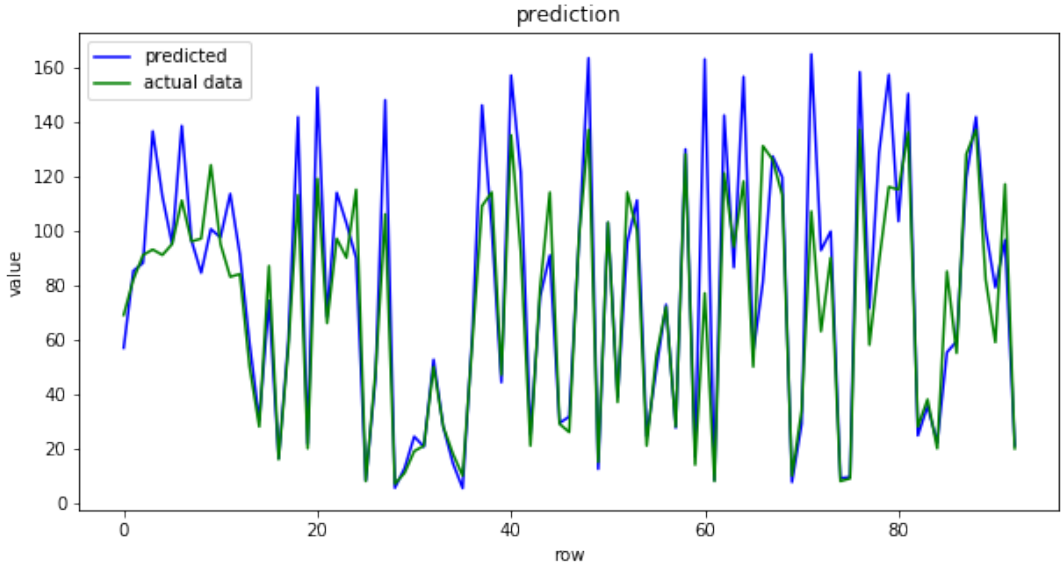

FIGURE 5.11: Result comparison between Regression GRU Model and Ground Truth data.

Chapter 6

Conclusions

6.1 Recurrent Neural Networks For Predictive Maintenance

As shown in the results obtained by the different models implemented in this dissertation, it can be concluded that RNNs are a very good solution to solve predictive maintenance problems using machine learning.

Both models (LSTM and GRU) have accuracy results higher that 97% for predicting if an aircraft is going to fail in defined number of cycles.

Also, both solutions work very well to predict the number of remaining cycles that an aircraft is able to operate without failing. A comparison of the results with the ground truth data can be seen in figures 4.6 and 5.6.

6.1.1 LSTM vs GRU

There are no great differences between the use of LSTM or GRU layers. The results are almost the same in both cases.

Also, the implementation costs using libraries like Tensorflow and Keras are disposable in terms of time or complexity.

The implementation of the GRU layer by Keras add more parameter configuration, like adding straight the dropout regularization without adding any extra layers.

On the other hand using GRU layers instead of LSTM adds a bit more cost concerning performance on the training phase, being the LSTM model a bit more faster.

Also the LSTM model tends to resolve earlier using *Early Stopping* callbacks provided by the Keras API.

6.1.2 Future steps

Currently, most of the traditional RNN models are moving to use *Transformer* architectures with *Attention layers*.

This kind of architectures are often used in NLP translation problems, but their applicability on time series data is something to take in account.

Applying this kind of architectures in predictive maintenance problems is a logic future step for this dissertation.

Appendix A

Code And Templates

A.1 Source code

All the code used for the implementation of this dissertation can be found on this GitHub repository.

A.2 Template

This dissertation was written using *LaTex* based on the Master/Doctoral Thesis template made by Steve R. Gunn and Sunil Patel. The template can be found here.

Bibliography

Cho, Kyunghyun (Sept. 2015). *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*. URL: https://arxiv.org/pdf/1406.1078v3.pdf.

Chollet, François (2018). *Deep Learning With Python*. Manning.

Fidan Boylu, Ahmad Afsar (July 2017). *LSTMs for Predictive Maintenance*. https://github.com/Azure/lstms_for_predictive_maintenance.

Griffo, Umberto (Apr. 2019). *Recurrent Neural Networks for Predictive Maintenance*. https://github.com/umbertogriffo/Predictive-Maintenance-using-LSTM.

Predictive Maintenance: Step 2A of 3, train and evaluate regression models (Apr. 2015). Microsoft Azure AI. URL: https://gallery.azure.ai/Experiment/Predictive-Maintenance-Step-2A-of-3-train-and-evaluate-regression-models-2 (visited on 07/01/2020).

Understanding LSTMs (Aug. 2015). Colah's Blog. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 07/01/2020).