Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer, Communication and Electronic Engineering

Final Dissertation

Title

Sub-title (optional)

Supervisor
Alberto Montresor

Student
Marco Morandin

Academic year 2023/2024

Acknowledgements

...thanks to...

Contents

Abstract

https://gpttutorpro.com/predictive-maintenance-with-machine-learning-regression-models/

1 Introduction

2 Background

2.1 Industry 4.0

The Fourth Industrial Revolution, or Industry 4.0, denotes a new stage in the industrial sector defined by the introduction of digital technologies into manufacturing processes. It represent a change from conventional industrial and production methods to a completely digitalised, networked industry. Industry 4.0 is applied thanks to some key technologies such as: industrial internet of things (IIOT), big data, cloud computing, cyber physical systems, advanced robotics, machine learning and augmented reality.

2.1.1 Industrial Internet Of Things

IIOT represents a transformation in manufacturing and industry, enabling connections between machinery, control systems and operators. This network facilitates real-time data collection and analysis, improves the efficiency of manufacturing processes, reduces downtime, and collects data for predictive maintenance. In addition, IIOT enables better adaptation to market changes through a scalable and flexible infrastructure and can improve the efficiency of stock management to make the delivery of new products faster thanks to a network of small but distributed interconnected warehouses.

Key difference between IIOT and IOT

The Industrial Internet Of Things (IIOT) is a subgroup of IOT, which means that they share the same concept of interconnectivity and interaction between devices, but differ in their goals, scales, and operational requirements. IOT is generally oriented toward customers; therefore, it usually includes smart home appliances, wearable and home security devices, and every other accessory focused on the improvement of daily life. IIOT is highly specialised in manufacturing, where reliability, security, and scalability are critical. In addition, the IIOT is designed to work under extreme conditions and ensure business continuity without interruption.

2.1.2 Big Data

The collecting and handling of data flows that are more complicated than normal data in terms of volume, velocity, and diversity is referred to as big data. Big data management becomes essential in the context of Industry 4.0, as the number of linked devices and sensors rises. Imagine a large factory with numerous pieces of machinery and sensors that send data every few seconds to each other. Furthermore, reliability is a need in a professional setting to prevent data loss that could taint analytics.
Sensor telemetry data is time-indexed in order to be utilised for analytics so time series data is the specific data type designed to manage time-indexed data. Time series data cannot be stored in a

standard database, instead, they need appropriate database.

Time Series Database

Time series database (TSDB) are designed to handle high ingestion rates without losing any data, they offer time stamp-based indexes that are more effective at managing time series data than standard databases, which have more generalised structures and indexing techniques.
The main differences between standard database and time-series database are:

- Query optimisation

  TSDB are designed to handle large volumes of data with specialised indexing and query capabilities like time aggregation and down-sampling in the other hand standard database offer general porpoise query language that are less efficient with time series.

- Write and Read Performance

  TSDB are optimised for high throughput and rapid access to time-based data that make them suitable for continuous data inflow. Standard database offer balanced read and write performance for diverse use cases.

- Data Retention

  While typical databases do not have retention policies and require the user to build them on his own, TSDB frequently contains features for down sampling, efficient storing of big historical datasets, and automatically summarising data based on age.

- Data Compression

  TSDB reduces storage footprint and improves retrieval speeds by utilising sophisticated compression techniques designed specifically for time-series data. Standard database uses generic compression techniques that may result in higher storage needs for extensive historical data since they are not especially tailored for time-series data.

- Scalability

  Due to the growth of the volumes of the data TSDB are design to scale horizontally with distributed architecture while traditional databases encounter certain challenges when attempting to do the same because they are designed to scale vertically.

2.2 Industry 5.0

Since industry 5.0 goes beyond efficiency and productivity as the only goals in the economic and social transformations we are living through, it is a relatively new idea that the European Union has utilised on multiple occasions. While industry 4.0 focuses on economic and production improvement industry 5.0 focuses on social value and well-being. One of the key concepts of this new industrial revolution is that the industry must be sustainable in order to respect planetary boundaries. It must create circular processes that minimise waste and their negative effects on the environment while recycling, repurposing, and reusing natural resources. Reducing energy use and greenhouse gas emissions is what is meant by sustainability, which aims to prevent the depletion and destruction of natural resources. By maximising resource efficiency, waste reduction, and energy efficiency and consumption, technologies like artificial intelligence and additive manufacturing can play a significant part in this. Waste could be reduced with the retrofitting of legacy devices and the adoption of the predictive maintenance.

2.2.1 Retrofitting of Legacy Devices

Retrofitting is the process of equipping machinery with new or enhanced IIoT functions, like predictive maintenance, always-on monitoring, or data analytics. Because of this procedure, manufacturers could avoid purchasing new devices, reducing waste connected to discarding functional legacy ones. Additionally, prolonging the life of current machinery lowers the cost of adopting new machinery, which includes some associated costs related to the device's lifetime, such as the purchase price, staff training costs, and termination fees.

2.2.2 Maintenance

Maintenance is defined in the UNI EN 13306:2018 standard as:

"The combination of all technical, administrative and management actions, during the life cycle of an entity, intended to maintain or restore it to a state where it can perform the required function"

From basic techniques used when machinery has a blocking failure to a complex strategy strongly integrated with other production processes playing a crucial strategic role, the concept of maintenance has changed over time to meet the changing demands of the industrial sector.
Maintenance strategy could be subdivided into three groups, corrective maintenance, cyclic maintenance and preventive maintenance.

Corrective Maintenance

Corrective maintenance is conducted after the identification of a fault. This approach is the simplest and least strategic when a failure has already occurred, as it entails a lengthy period of machinery downtime until repairs or replacement are completed. **This prolonged downtime results in significant financial and energy losses due to the halt of production. Additionally, the implementation of corrective maintenance may compromise machine safety, as it may lead to dangerous failures that could harm employees. Another disadvantage of corrective maintenance is that the potential for resource waste increases, as there is a risk of creating incomplete or broken products.**

Cyclic Maintenance

In cyclic maintenance, regardless of the current operational status of the equipment, maintenance tasks are carried out at predefined intervals. This approach is advantageous in that the tasks are prearranged and standard methods are followed for inspection, cleaning, lubrication, and component replacements. **However, the potential for excessive maintenance exists, which could result in a time and resource waste. This indicates a non-sustainable approach and financial waste. Cyclic maintenance does not consider the condition of the machinery, which may result in the maintenance period becoming excessively long over time. This can lead to unexpected errors and the need for corrective maintenance.**

Predictive Maintenance

Predictive maintenance is an anticipatory approach that employs tools and methods for data analysis to predict potential irregularities in equipment performance. This enables the implementation of maintenance procedures prior to an equipment breakdown. The data is processed using machine learning algorithms, which identify patterns and anomalies suggestive of possible breakdowns. Predictive models identify the optimal time for maintenance and predict future equipment failures based on historical data. Predictive maintenance has several advantages, including decreased downtime, lower costs, longer equipment life, enhanced efficiency, safety, and data-driven decision-making. Predictive maintenance offers more dependable and economical operations than traditional main-

tenance methods by utilizing technology and data analytics.

3 Methodology

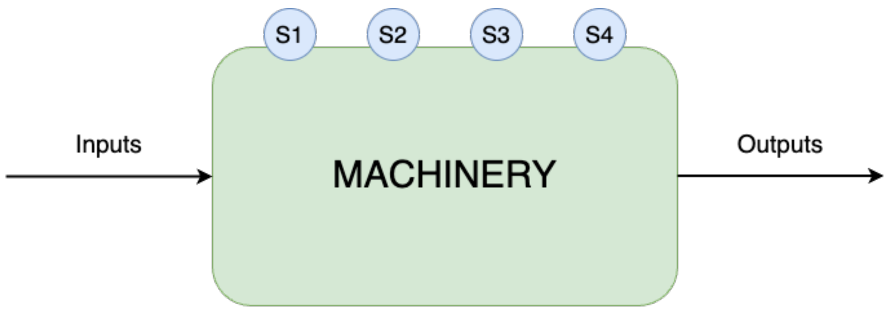
This chapter is about how the implementation of predictive maintenance of legacy devices is done in this dissertation. The data collection part of the project was done during my internship ad Thinkin, a software house situated in Trento.

3.1 Problem Definition

The goal is to retrofit legacy equipment that is working today but does not have support for sensor data collection due to lack of sensors or inability to access data, and add some data analytics such as predictive maintenance.

3.2 Operational Framework

In order to implement predictive maintenance, it was determined that a framework should be employed whereby external sensors are installed on a device and monitor its inputs and outputs. This allows for the understanding of whether the device in question requires maintenance or not.



So the framework can be generalised as follows:

$$Remaining\ Useful\ Life\ =\ f(Input,\ Output, \overline{S}\ )$$

The framework presented above allows for the observation of input and output as well as sensor data, in order to identify a function that can approximate the remaining useful life of the equipment in question.

3.3 Requirements

Given the intended real-world context of this project, which involves multiple users, each with multiple machines and sensors that transmit data at regular intervals, it is essential to implement a robust software architecture capable of handling high throughput and a vast volume of data without compromising data integrity or introducing errors. Furthermore, the objective was to develop an architecture that could be readily scaled horizontally, allowing for the project's growth to be managed effectively.

3.3.1 Used Technology

This section introduces the software technologies utilised in the project to fulfil the requirements of the preceding section. The objective is to underscore the decision to employ professional technologies that are readily scalable on a large scale, achieve optimal performance and are reliable. Additionally, they possess the benefit of being all open source technologies.

MQTT

MQTT is a publish-subscribe network protocol that is widely utilised in the field of Internet of Things due to its minimalistic design. It is an open OASIS standard and an ISO recommendation.
MQTT defines two actors in the network: a message broker, which receives all messages from clients and routes them to the appropriate destination clients; and clients, which are all devices that are connected to a message broker. The communications are structured into a hierarchical topic model, in which clients can publish or subscribe.
MQTT is designed to operate on constrained bandwidth and reliably transmit messages, even across unstable or high-latency networks. The protocol's keep-alive feature and minimal header size make it an optimal choice for IoT since it could be implemented in devices with microcontrollers. The lightweight nature of MQTT enables it to connect a considerable number of clients simultaneously to the same message broker.

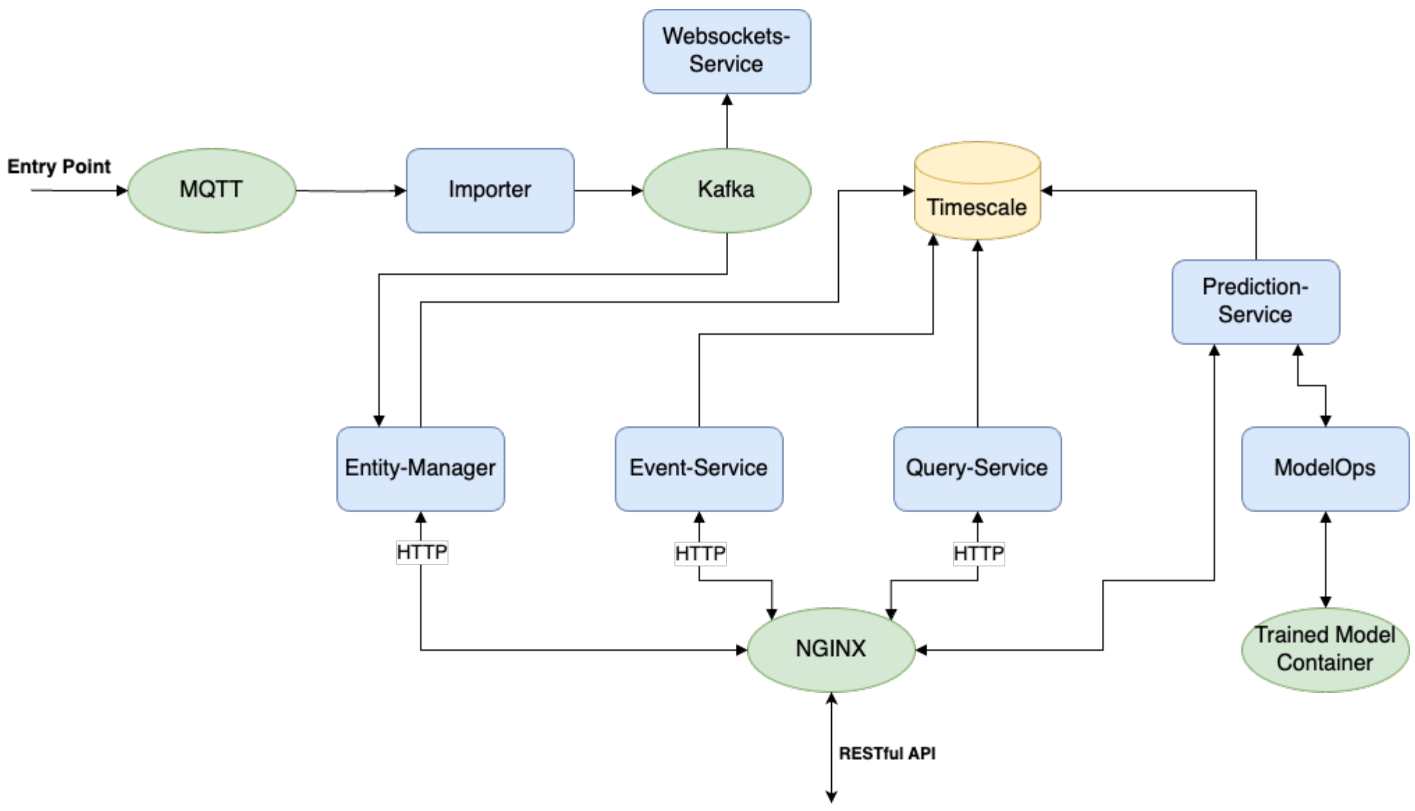
Kafka

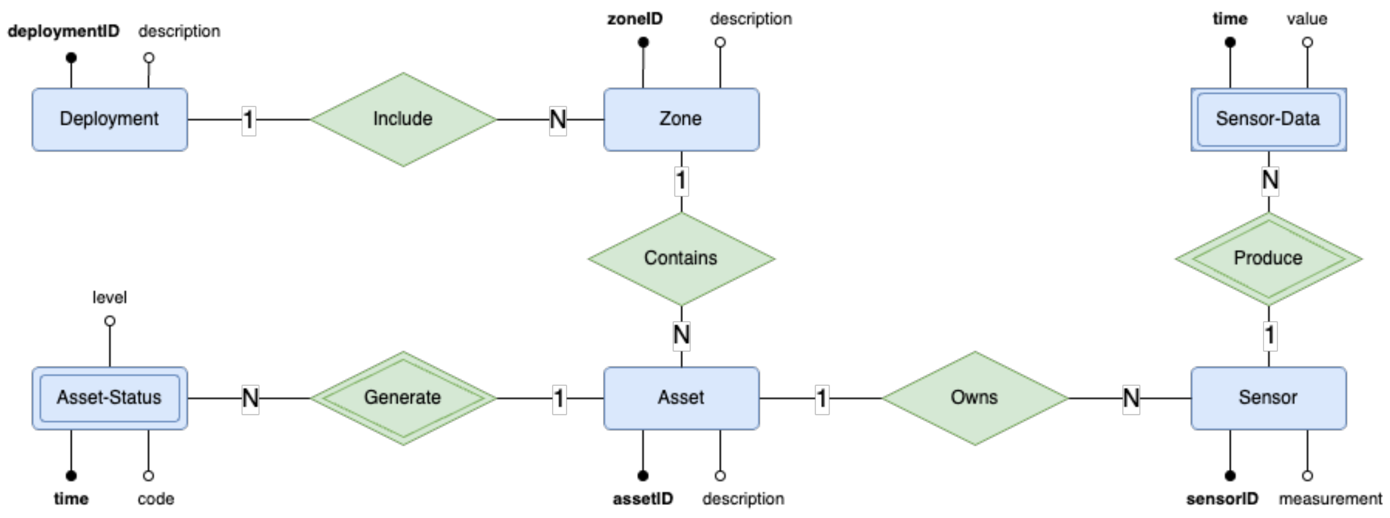
Timescale DB

Timescale is a time series database built on PostgreSQL, extending it with time-series optimisations. In comparison to PostgreSQL, Timescale offers a number of additional features, including automatic data partitioning, high ingest rates, and compression techniques optimised for time series data. The result is that queries on time series are thousands times faster in comparison with PostgreSQL. Due to the fact that timescale uses SQL with some additional function make it simpler to learn compared to other time series databases. Moreover, as an extension of PostgreSQL, it benefits from the reputation built up over twenty-five years of existence. This makes it an excellent choice, as it is a guarantee of a widely known, solid database management system. Furthermore, between versions, there are no radical changes that make the work already done inapplicable.

3.4 Software Architecture and Implementation

The software architecture designed to facilitate predictive maintenance of legacy machinery is structured with micro services to enable horizontal scalability and facilitate maintenance of the various software components.
The following image depicts the software architecture in use. Rectangles represent developed services, ovals represent used services, and the cylinder represents the database.
The data flow begins with MQTT, where data from all sensors registered in the system are published. From there, a micro service called "importer" is responsible for reading the data published on MQTT and publishing it to a topic on Kafka. The importer is a straightforward module that ensures optimal reliability and efficiency in data persistence. Subsequently, a micro-service designated as the "entity-manager" is responsible for database management and data storage from Kafka to Timescale. This service provides a suite of RESTful APIs that facilitate control over the entities stored in the database. In the system, data is divided into deployments, which represent the various clients. Deployments are divided into zones, which represent areas of a factory or locations. In the various zones, there are different assets, which represent machinery. Finally, each zone contains the various sensors installed. In the picture below you can see the ER schema of the database. It is intended to point out that the sensor-data and asset-status tables are Timescale hyper tables and are weak relationships, which means that the data are identified by the instant at which they were recorded and by the relationship they have to the other entity to which they are linked.
With the API, these entities can be created, deleted, and visualised. Additionally, for each asset, the status is saved in a dedicated table, representing any malfunctions encountered. Finally, a table containing the data of all sensors with a reference to the corresponding sensor is available. The "entity-manager" service is also responsible for reading data from Kafka and saving it to Timescale in such a way that it is saved persistently and efficiently. This is because complex queries can be made on Timescale and data compression is much more efficient. Furthermore, a service designated





as the "query service" has been developed to facilitate the execution of queries on sensor data via RESTful APIs. This service aggregates the results into buckets of a size selected by the user. Additionally, an "event service" has been created to generate events on sensor data through the creation of RESTful API rules comprising mathematical formulas. The latter two services are beneficial for the generation of statistical data on the operation of a machine. Finally, with regard to predictive maintenance, a service called "prediction-service" has been created that, through an HTTP endpoint, provides the prediction of the remaining life of the machinery. This service takes care of making a query on Timescale to obtain the useful data of the requested machinery and then forwards it through an HTTP request to another service that, with the received data, makes the prediction. Finally, a service has been developed that collects sensors data in real time and posts them into web-sockets, thereby providing near real-time data for potential visualisations.

3.5 Predictive Maintenance Implementation

This section presents a detailed account of the implementation of predictive maintenance.

3.5.1 Long Short Term Memory Model

3.5.2 LSTM Operationalisation

4 Use case

To test the operation of the previously exposed implementation, we selected an example dataset generated through a script implemented by Microsoft Azure. This choice was made due to the scarcity of online datasets that were complete enough to obtain an acceptable result. This scarcity is likely due to the sensitivity of the information contained in them.

4.1 Sensor Data Generator

The data generator is an open-source physics-inspired simulation framework that can be customised for the generation of data for model training. Generated data are a combination of telemetry and maintenance records.
Generated telemetry data contains:

- Timestamp
- MachineID
- Ambient pressure
- Ambient temperature
- Rotational speed
- Desired rotational speed
- Temperature
- Pressure

Maintenance generated record contains:

- Timestamp
- MachineID
- Level (INFO, WARNING, ERROR, CRITICAL)
- Code (identifies event/failure type)

This data does not represent a particular machinery but it is supposed that it could represent rotational motors or hydraulic pumps.

4.2 Data Preparation

4.2.1 Data Ingestion

In order to align the generated dataset with the data collection architecture depicted above, preliminary operations are necessary. First, a suitable structure must be created in the system. This involved the creation of a deployment, a zone, and two machines with six sensors each. Next, the generated data must be divided by sensor. This is to simulate sending the data by individual sensor. This is done by going to create records containing the timestamp, the value read from the sensor, and the sensor ID. Then, the data must be sent on the MQTT topic where the system listens. This allows the data to be uploaded to Timescale.

4.2.2 Feature Engineering

In the context of machine learning, a feature is a single measurable attribute of a physical phenomenon; the process of extracting features from raw data is called feature engineering.
In this case, the data is aggregated into machine cycles, which are periods of time when the machine is in a particular state; for example, the operation of an electric motor can be divided into cycles based on the states of on, rotating, and off.
This aggregation is necessary because the raw data cannot be used directly to train a predictive maintenance model.
Aggregation into cycles usually works because it is rather unlikely that there will be a sudden degradation during an operating cycle.
Cycles are dynamic in that they do not have a fixed duration, but adapt to the duration of the machine cycle, separating cycles when there is an interruption of at least 30 seconds in the data.
In addition, it is necessary to merge maintenance type data with telemetry type data aggregated by time and machine. This allows us to go in and calculate the Remaining Useful Life (RUL) based on the position of the various telemetry data in the sequence; this way we get the number of cycles for which the machine will continue to operate smoothly. For example, the cycle just before a problem will have a RUL equal to 1, the cycle before that will have a RUL equal to 2, and so on.
These operations to group the data for training were done through a complex query on Timescale, which allows us to get a data set composed in this way.

4.3 Model Training

In order to train the model, a number of preliminary operations were performed. Initially, a query was issued to the database, as previously described, and the data was then stored in Pandas data frames. These are a highly efficient two-dimensional data structure commonly used in data science. They were selected because they provide a pre-defined set of data manipulation operations and are highly efficient data structures.
Subsequently, sensor datasets belonging to the same machine with the same timestamp were merged into a single row, resulting in a structure that contains

- asset_id
- cycle
- speed_desired_max
- speed_avg
- temperature_avg
- temperature_max
- pressure_avg
- pressure_max
- rul

Subsequently, the dataset is partitioned into two distinct data structures: "sequences" and "labels." The sequences are two-dimensional NumPy arrays that encompass all data except the rule, while the labels are NumPy arrays that exclusively contain the rule. This process results in the creation of two subdatasets, with the sequences serving as the input data for the model and the labels representing the output data that the model is tasked with predicting. The selection of a NumPy array as the data structure was driven by the necessity of aligning with the requirements of the utilised model.

4.3.1 Features Scaling

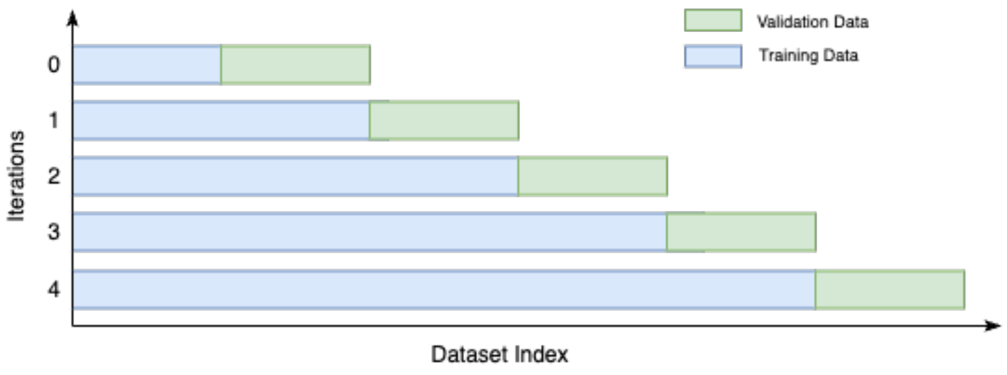
The utilisation of a deep learning model, such as LSTM, necessitates the normalisation of features. This optimises the training of the model, resulting in enhanced speed and accuracy.
Specifically, the normalisation of features facilitates a faster and more efficient convergence of the gradient. This, in turn, influences the training of the LSTM model, which exploits the descending vanishing problem in order to 'learn'. Furthermore, neural networks are susceptible to numerical instability issues that can be mitigated by scaling the values to eliminate those that are excessively large or small. Finally, feature normalisation enhances the model's generalisability by improving its ability to learn from diverse data.
In our case, we employed an object provided by the "Scikit-Learn" library called "MinMaxScaler." This object scales the data in a range from zero to one by applying a linear scaling transformation. It should be noted that this normalisation technique does not reduce the effects of outliers, as the smallest value in the dataset corresponds to 0 and the largest value in the dataset corresponds to 1. The scaled values are obtained with the following formula:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where X is the value that has to be scaled, X_{min} is the smallest value in the dataset and X_{max} is the biggest value in the dataset.

4.3.2 Dataset Splitting

The k-fold cross-validation technique was selected for training the model. The rationale behind this choice is to prevent overfitting, which occurs when a model performs well on training data but struggles with new data. The k-fold technique involves dividing the dataset into k subsets and selecting one subset for validation while the others are used for training. This process is repeated k times, with each iteration involving a different subset chosen for validation. Finally, the results of each step are averaged in order to obtain an accurate performance estimate. In this case, the "TimeSeriesSplit" object from the "scikit-learn" library was utilized to perform the split. This object enables the implementation of cross-validation, in contrast to the conventional k-fold approach, wherein the data is mixed and then split into subsets. In the case of the timeseries split, the data is split while maintaining the temporal order. The dataset is divided into consecutive subsets, with each iteration utilising data that has already been employed for training and data that is new to the model, which is then used for testing. This process is analogous to the real-world operation of training a model on historical data and then testing it on future data.



4.3.3 Model structure

The model comprises multiple layers, each with a distinct function. To achieve this, a sequential model from the Keras library was employed, which enables the construction of machine learning models comprising various layers. The layers utilized are as follows:

- Input: This layer instructs the model regarding the size of the dataset to be passed for training.
- LSTM: This layer will undergo training with the LSTM model.

- Dropout: The dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. In this case, after some trials, the chosen drop rate is 0.2.
- LSTM: another layer with the pattern, using two layers gives better performance in recognizing complex patterns by increasing the generalisation capabilities of the pattern. However, the problem is that having two layers makes training computationally more intensive.
- Dropout
- Batch Normalization: This process normalizes the activation of each layer by subtracting the mean and dividing by the standard deviation. It improves the training speed and helps to prevent overfitting.
- Dense: this layer applies the following operation

  $$output = activation(input \cdot kernel) + bias$$

  where activation is the activation function passed as parameters, in this case "linear", kernel is a weights matrix created by the layer and bias is a vector created by the layer. It is the layer that effectively gives us an output.

Furthermore, a callback was incorporated into the model via the "EarlyStopping" object, which enables the training process to be terminated when the observed metric fails to demonstrate a continued increase. In this instance, the observed metric is the training loss, which serves to prevent overfitting. Finally, the training was conducted with a batch size of 64 and 200 epochs, resulting in a total of five repetitions of the training process, or the number of subsets created with the TimeSeriesSplit. The values in question were selected following an exhaustive series of tests designed to identify the optimal parameters, given that there is no established formula for defining them in advance.

5 System evaluation

5.1 Training Evaluation

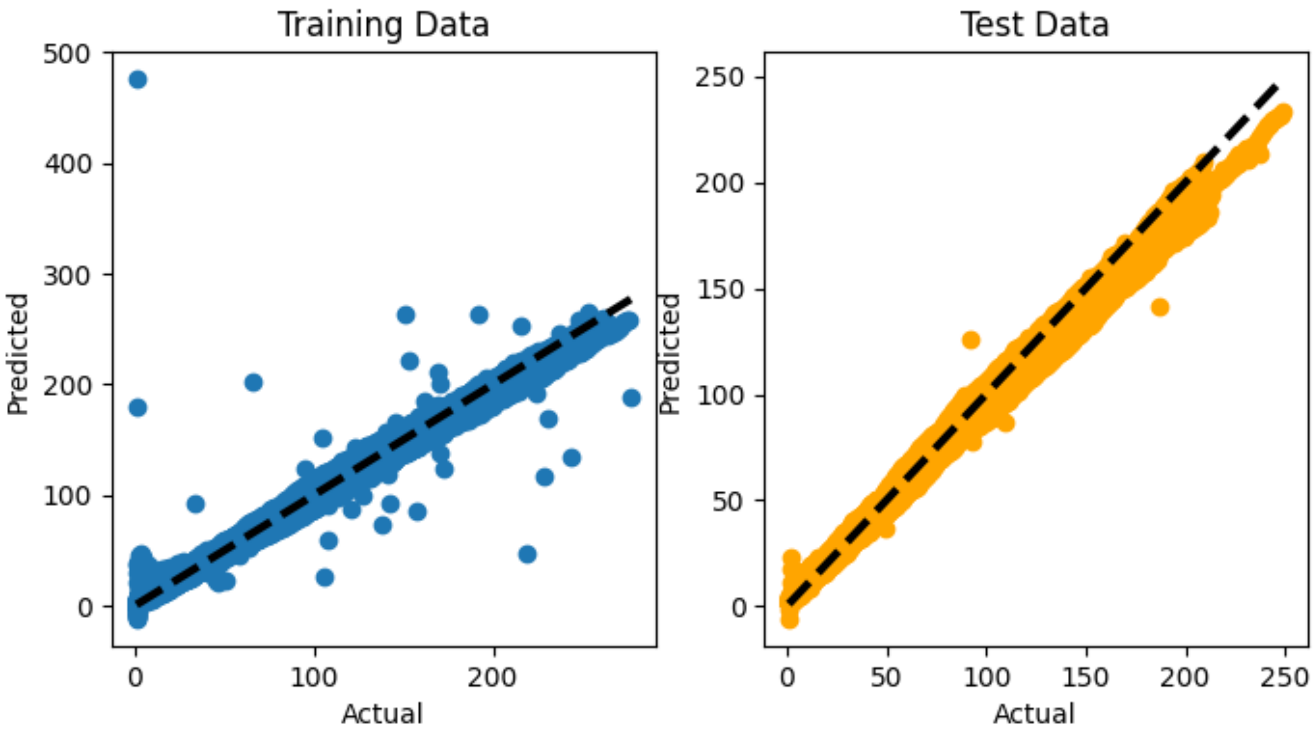
5.2 Project Results

6 Conclusions

6.1 Future Work

6.2 Limitations and Simplifications

6.3 Conclusions



Training Data
Test Data
Predicted
Actual

Bibliography