



**UNIVERSITÄT PADERBORN**

*Die Universität der Informationsgesellschaft*

Faculty for Computer Science, Electrical Engineering and Mathematics  
Department of Computer Science

## Project Group Report

Submitted to the Advanced Systems Engineering Research Group  
in Partial Fulfilment of the Requirements for the Degree of

Master of Science

# Anomaly Detection in Sensor Data

by

APOORVA RAVISHANKAR  
6868653

HARIS NAWAZ  
6865969

NOOR MAHAMMAD SYED  
6868576

SPOORTHY MADHAVAN  
6865947

Project Group Mentor:  
Prof. Dr. -Ing. Roman Dumitrescu

Paderborn, September 17, 2020

# Table of Contents

<b>Chapter 1: Introduction</b>	<b>2</b>
Dataset Characteristics	2
Problem Statement	2
<b>Chapter 2: Frontend</b>	<b>3</b>
Home Page View	3
Loading Screen View	4
Graph Visualizations View	5
Plots View	6
<b>Chapter 3: Backend</b>	<b>9</b>
Approach	9
Noise Removal	9
Un-Supervised Learning Model	10
Hyper-parameter optimization	11
Correlation Matrix	12
<b>Chapter 4: Setting up and running the project</b>	<b>12</b>
Application Usage	12
<b>Chapter 5: Technology Stack</b>	<b>13</b>
Frontend	13
Backend	13
<b>Chapter 6: UML Diagrams</b>	<b>14</b>
<b>Chapter 7: Conclusion</b>	<b>17</b>

# Chapter 1: Introduction

Anomaly detection is the process of identifying unexpected data points in the given dataset. The datasets could be the readings from machinery, sensors, etc., The unexpected data points are the events that deviate from the normal behavior of the dataset due to some failure or defect in the corresponding models or due to external factors. Understanding and analyzing these data points would help to prevent failures in the future. In most cases, anomaly detection is applied to unlabeled data. This method is called unsupervised anomaly detection. Unsupervised anomaly detection is performed on the basis of two assumptions that it is rare that the anomalies occur in data and their features differ from the normal instances significantly.

## Dataset Characteristics

The dataset in discussion in this project is - Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository [6].

Data sets consist of multiple multivariate time series. Each data set is further divided into training and test subsets. Each time series is from a different engine number i.e., the data can be considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation are considered normal, i.e., it is not considered a fault condition. The data is contaminated with sensor noise.

The data in each file has 26 columns of numbers, separated by spaces. Each row is a snapshot of data taken during a single operational cycle, each column is a different variable. The columns correspond to:

- 1) unit number
- 2) time, in seconds
- 3) operational setting 1
- 4) operational setting 2
- 5) operational setting 3
- 6) sensor measurement 1
- 7) sensor measurement 2
- ... ....
- 26) sensor measurement 21

## Problem Statement

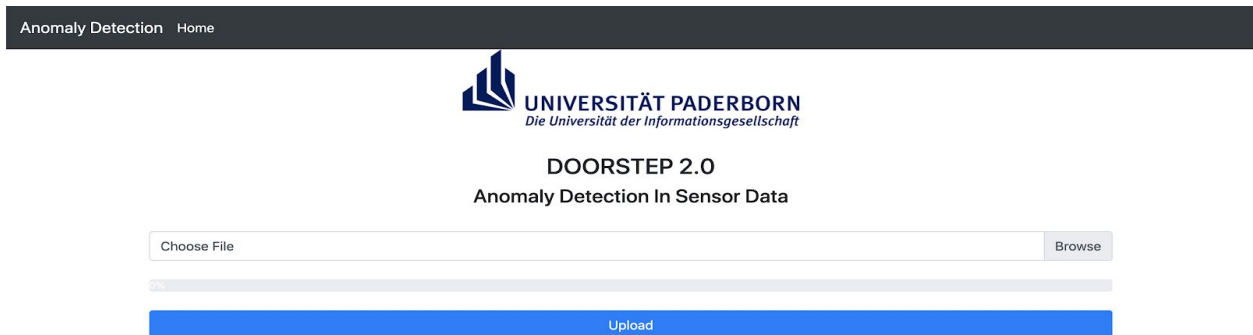
The challenge in this project was to understand the unlabeled dataset, break it down into chunks of useful information, remove noise present in the data and create a machine learning model that can convert data into analyzable information for domain experts.

A tool must be created that depicts the data points and its trend over time, show sensors with anomalous entries in the data, also tabularise any resemblance in the sensor readings by considering correlation metrics.

## Chapter 2: Frontend

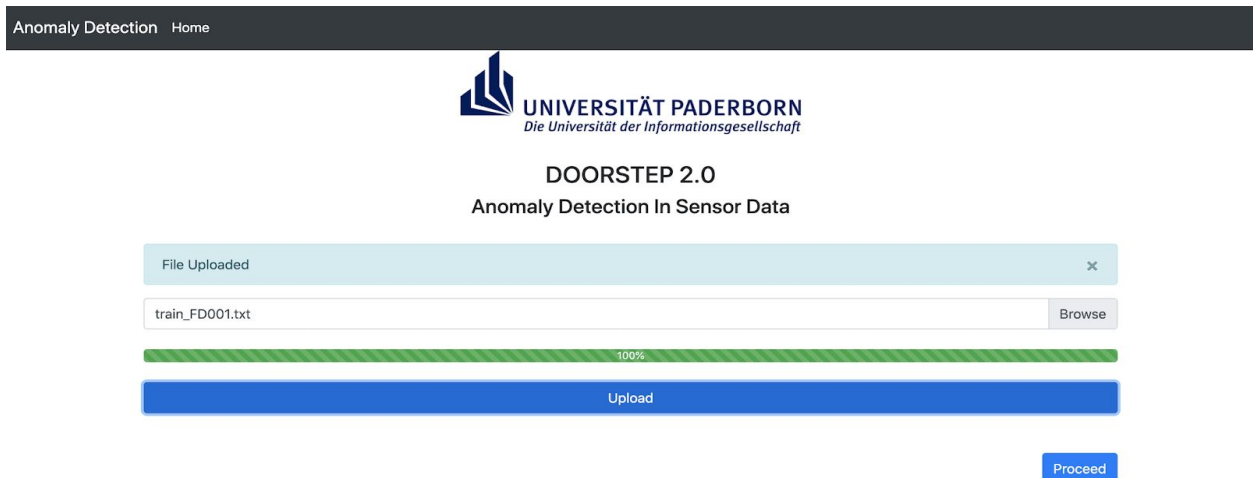
The task of the frontend was to depict the anomalies in sensor data in an understandable fashion to the user. In order to do so, a sequential flow is employed. Below is a presentation of all the views starting from the home page to graphs showing the detected anomalies and finally viewing the correlation between sensors as a table.

### Home Page View



The screenshot shows the home page of the DOORSTEP 2.0 application. At the top, a dark navigation bar contains the text "Anomaly Detection" and "Home". Below this, the logo of the University of Paderborn is displayed, followed by the text "UNIVERSITÄT PADERBORN" and "Die Universität der Informationsgesellschaft". The main heading is "DOORSTEP 2.0" with the subtitle "Anomaly Detection In Sensor Data". A file upload section includes a "Choose File" button, a "Browse" button, a progress bar showing 0%, and a large blue "Upload" button.

(Figure 1)



The screenshot shows the home page of the DOORSTEP 2.0 application after a file has been uploaded. The navigation bar and university logo remain the same. The main heading is "DOORSTEP 2.0" with the subtitle "Anomaly Detection In Sensor Data". A file upload section includes a "File Uploaded" message with a close button, a text input field containing "train\_FD001.txt", a "Browse" button, a green progress bar showing 100%, and a large blue "Upload" button. A "Proceed" button is located at the bottom right of the upload section.

(Figure 2)

(Figure 1) is where the user first lands when he opens the project.

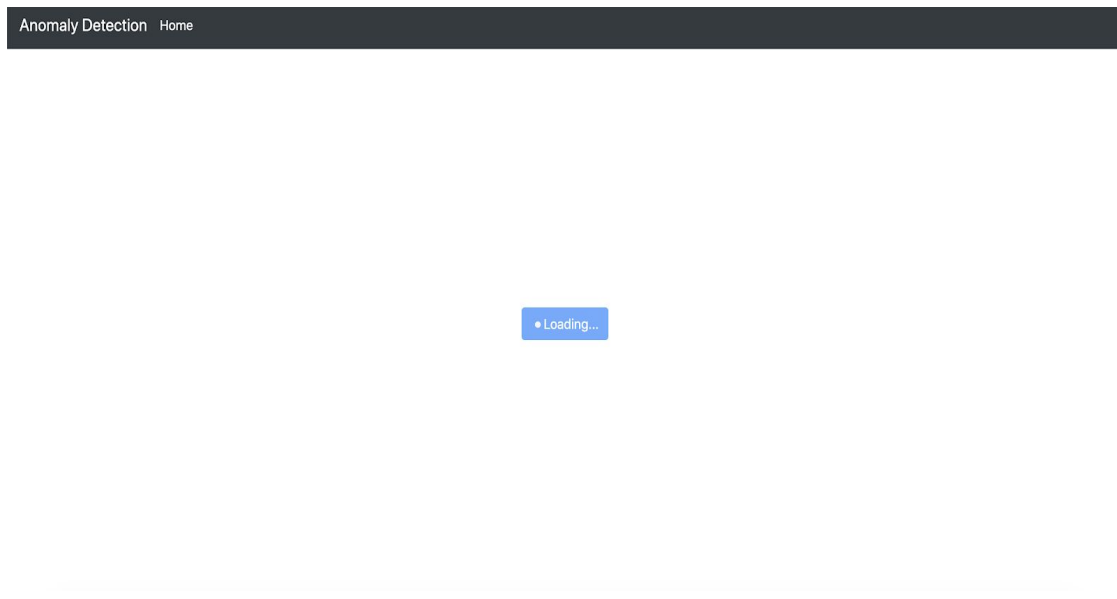
- The first screen is the home screen and clicking on the home button from any other page navigates to this page.
- Home screen has a “Choose File” option which opens a file browser to upload a file for further processing.
- On clicking Upload, a progress bar appears that indicates the percentage of the uploading process.

(Figure 2) shows the view after selecting a file and clicking the upload button

- When the progress bar shows 100%, the file selected was uploaded successfully to the backend for further processing.
- A message “File Uploaded” is being displayed to indicate the user that process is successful.
- In case of a failure, an error message “There was a problem with the server” will be displayed. In order to inspect it further, one can use the inspect option of their web browser which gives detailed message received from the backend.
- Once the upload is successful, a “proceed” button gets enabled to start backend modeling and to view different visualizations.

---

## Loading Screen View



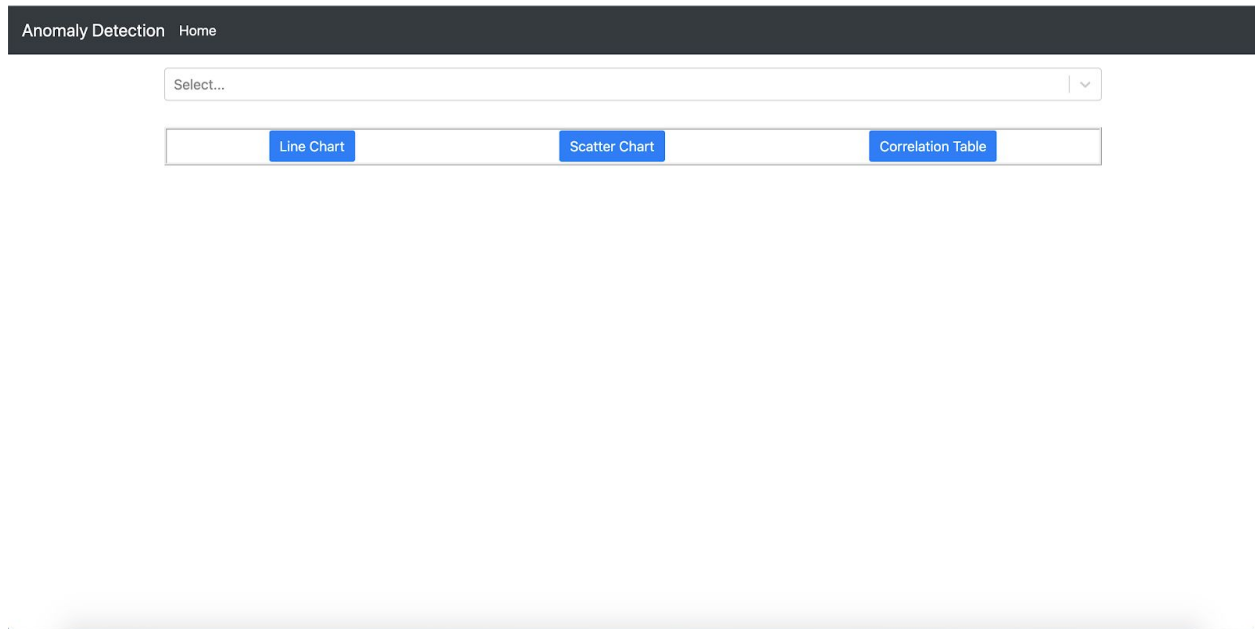
(Figure 3)

---

Loading screen (Figure 3) - After successfully uploading a file, the backend processes the data in the file which takes some time. Until the backend finishes the processing, user is shown this screen.

---

## Graph Visualizations View



(Figure 4)

---

Basic view of dropdown and visualisations (Figure 4): Upon successful completion of data cleaning, processing & modeling in the backend, JSON response is sent to the frontend. This changes the view from loading screen to view shown in Figure 4. The screen contains a dropdown menu and visualisation tabs.

---

DropDown Menu (Figure 5): Only the sensors which have anomalous data points are listed in the DropDown Menu. The user can click on one of the values to view the corresponding visualizations of the chosen sensor.

Select...

▼

sensor\_03

sensor\_04

sensor\_09

sensor\_14

(Figure 5)

## Plots View

When a sensor is chosen from the backend, 2 types of graphs visualizations are plotted. It is explained in detail below:.

Line Chart :



(Figure 6)

(Figure 6) - The Line Chart tab loads a line chart of the data values plotted against the duration of time the sensor ran. X-axis represents Time (in seconds) and Y-axis represents Sensor Data.

Line charts are drawn using [CustomizedDotLineChart](#) so as to represent an anomaly with a customized red dot.



(Figure 7)

A [Brush](#) tool has been used to zoom into the charts. The user can use the brush to resize the contents in the charts and exactly where the anomaly has occurred. An example is shown in Figure 7.

Scatter Chart:



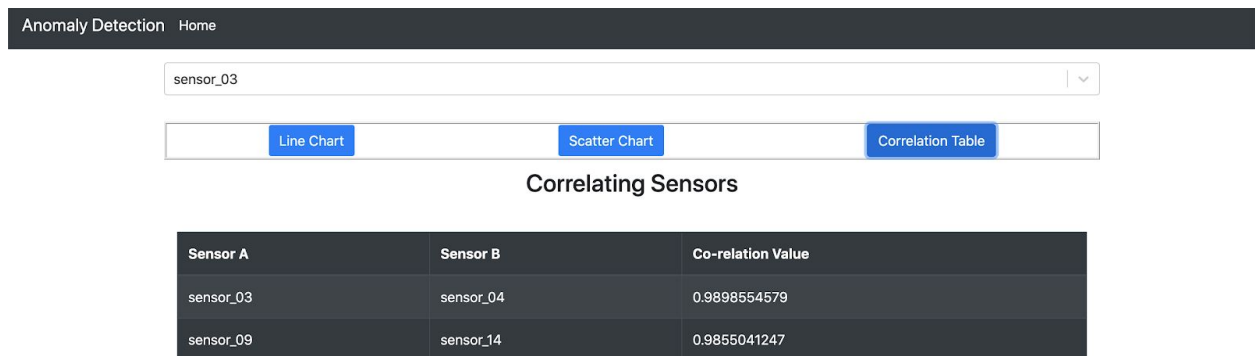
(Figure 8)



Figure 7 - When the user clicks on Scatter Chart tab, a band is calculated with upper and lower limit values for each data point. If the data point falls within this limit then it is considered as non-anomalous (Green colored points) else as anomalous (Red colored points).

All the Graphs are developed using recharts [3].

Correlation table:



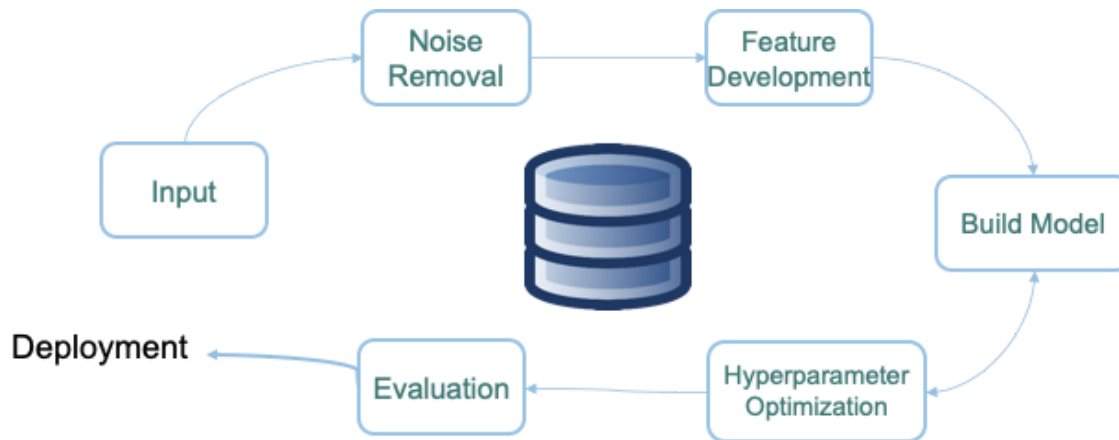
(Figure 9)

A high Correlating value says that corresponding sensors are similar. Here in the picture the pairs (sensor\_03 , sensor\_04) (sensor\_09 , sensor\_14) are correlated. We have set a high threshold (0.8) to qualify whether the sensors are related. Therefore the correlation table shows the user the pairs of related sensors.

# Chapter 3: Backend

## Approach

Our Machine Learning model follows the underlying CRISP-DM model for forecasting anomalies in time-series data. Each of the steps is explained in the further sections.



## Noise Removal

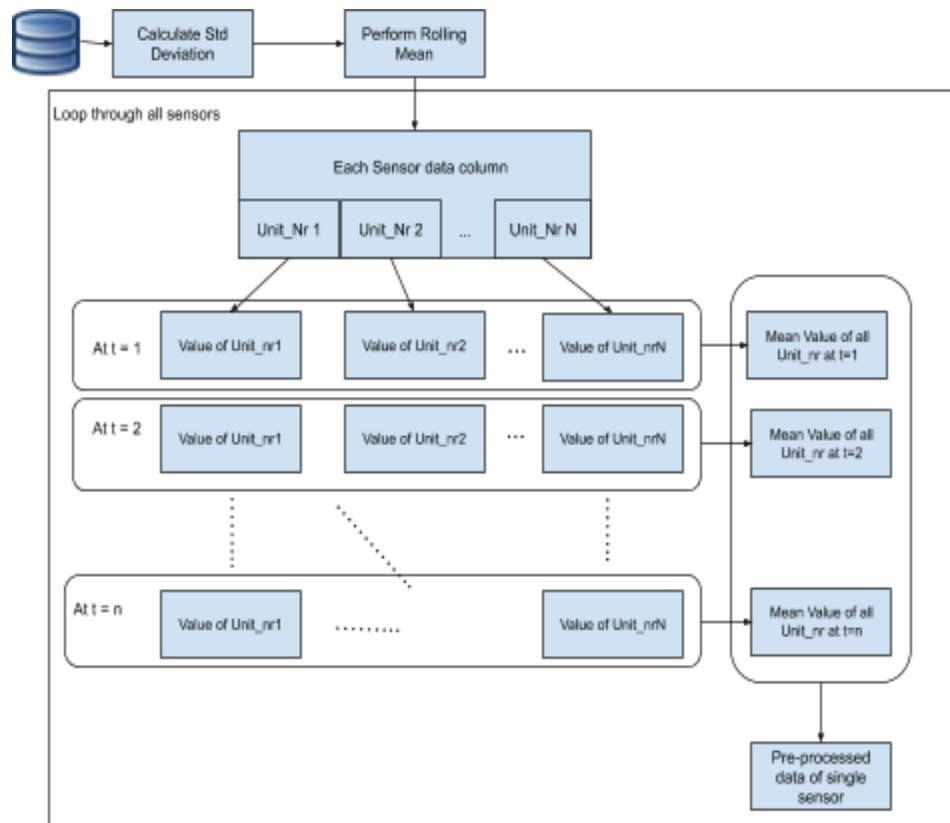


Figure 10: Explaining the pre-processing pipeline

The first step in data cleaning is to prune the data according to the needs of the supervised learning model. As explained above, the dataset has multiple test and training files. We consider only the training files to train the model. Each training file has 21 sensor columns with 100's of unit numbers.

Since the goal is to evaluate the irregularities in data, any data columns with negligible standard deviation are avoided. As they are considered to be perfectly recorded data & non-anomalous. Later a well-known filtering approach called Rolling Mean [1] is performed. Here any data is smoothened out of short-term fluctuations and longer-term trends or cycles are highlighted.

Once the smoothening is done, the columns still have many redundant values i.e At time  $t=1$ , N-Unit numbers produce N data points corresponding to one sensor. So, a mean of these values will speed up the computation time. Since all the units of the sensor start at the same time (but not necessarily end at the same time), the data points collected in the same second are averaged. The average of all data points from  $t=1,2,...,n$  is the cleaned & pre-processed output of one sensor. This process is carried out for all 21 sensor columns.

## Un-Supervised Learning Model

Since the data is unlabeled and no information is provided regarding what each sensor does, and if they are related. The goal of the learning model is to learn the history of the data points, analyze the trend, and predict a range for the upcoming data value. If a datapoint lays within this predicted range ( $\hat{y}_{upper} - \hat{y}_{lower}$ ) then such a data point is considered non-anomalous. In order to accomplish this task, we made use of an opensource library created by Facebook called - Prophet Library [2]. The library is used to generate a modular regression model with interpretable parameters that can be intuitively adjusted by analysts with domain knowledge about the time series. Working with the Prophet library is pretty straight forward.

It takes a data frame with 2 columns, time series, and value. An example is depicted below:

index	datetime	value
0	11:35:17	51.7
1	11:35:18	51.4
2	11:35:19	52.3
3	11:35:20	51.0

Note: Prophet requires that the DateTime column is named as "ds" and the value column as "y". Prophet follows a simple "instantiate  $\rightarrow$  fit  $\rightarrow$  predict" workflow for forecasting.

The result of the predict function yields a dataframe containing 4 columns: ['ds', 'yhat', 'yhat\_lower', 'yhat\_upper'].

They stand for:

- ds: forecast time steps
- yhat: forecast values
- yhat\_lower & yhat\_upper: confidence intervals

The output would look like:

index	ds	fact	yhat	yhat_lower	yhat_upper
0	11:35:17	51.7	52.1	51.6	52.2
1	11:35:18	51.4	51.9	51.4	52.4
2	11:35:19	52.3	51.5	51.0	52.0
3	11:35:20	51.0	51.3	50.8	51.8

Prophet follows an additive regression modeling [4] where it generates a piecewise linear or logistic growth curve trend.

Figure 11 shows the Sample working of the algorithm used by the prophet library.

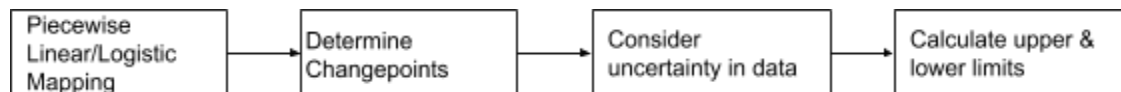


Figure 11: Algorithm Implementation of Prophet Library

Detailed working can be read here [5].

## Hyper-parameter optimization

Certain parameters can be adjusted in order to modify the model as per our needs, The predict function takes the following parameters:

- 1) [n\\_changepoints](#) - parameter can be used to manually set a number of potential changepoints.
- 2) [interval\\_width](#) - parameter to specify a custom uncertainty interval width.
- 3) [changepoint\\_prior\\_scale](#) - parameter determines the flexibility of the trend.
- 4) [changepoint\\_range](#) - Parameter that takes the proportion of the history in which the trend is allowed to change.

# Correlation Matrix

In general, correlation matrix shows the correlation of two given variables. In our solution, we have used correlation matrix to analyse the sensor data and show the sensors in a tabular form that are correlated to each other. The correlation matrix has been implemented using the Pearson Correlation coefficient [7] method where the covariance of two data values of sensors is divided by the product of their standard deviation. After removing null/empty values, predefined correlation values(0.8, -0.8) are applied to identify the strongly correlated and weakly correlated sensors. In our solution, we are showcasing only those sensors which are highly(>0.8) correlated and weakly(<-0.8) correlated. The predefined correlation value can be changed in the code as needed.

## Chapter 4: Setting up and running the project

### Setting Up the project

Following steps are necessary to set-up the project in the local environment.

1. Pull the project repository from [https://git.cs.uni-paderborn.de/users/sign\\_in](https://git.cs.uni-paderborn.de/users/sign_in).
2. Install all the technology stack mentioned above.
3. To avoid compatibility issues in windows systems, install Anaconda, create a virtual environment, and then install the mentioned libraries.  
(<https://docs.anaconda.com/anaconda/user-guide/getting-started/>)
4. Install dependencies by executing 'npm install' in command prompt.
5. Execute 'npm start' to run the application on localhost:3000 by default.

### Application Usage

Follow the below steps to run the application.

1. Once the server starts running, browser will open the application automatically on localhost:3000
2. Now, run the 'app.py' file to start the backend.
3. Once app.py starts running, upload the data file in the frontend using fileupload option.
4. Click 'Proceed' to start the analysis.

# Chapter 5: Technology Stack

## Frontend

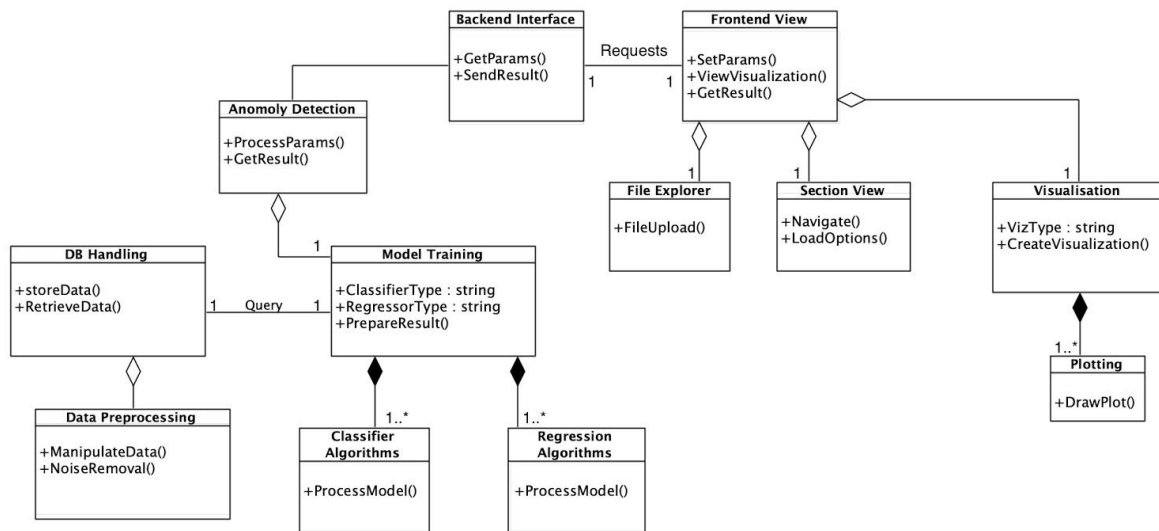
Technology / Frameworks / Library	Version	Description
React.JS	16.8.6	Used as frontend
Recharts	1.8.5	Used for data visualisation

## Backend

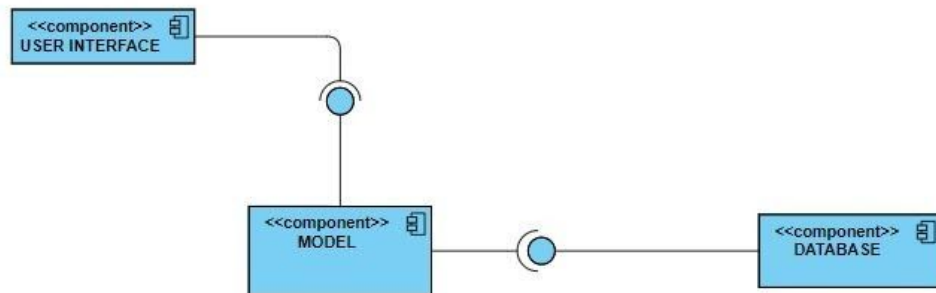
Technology / Frameworks / Library	Version	Description
Python	3.7	Used as backend
pandas library	1.0.5	Used for data analysis and manipulation
NumPy library	1.18.5	Used to perform mathematical operations
fbprophet library	0.6	Used for forecasting anomalies
C++	14.0	Used by fbprophet library

# Chapter 6: UML Diagrams

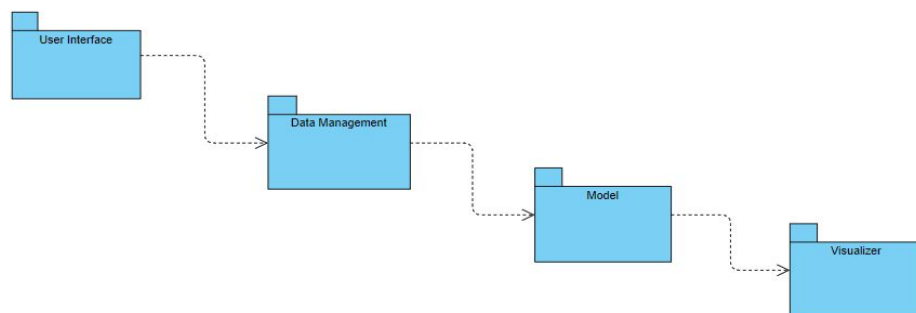
## a. Class Diagram



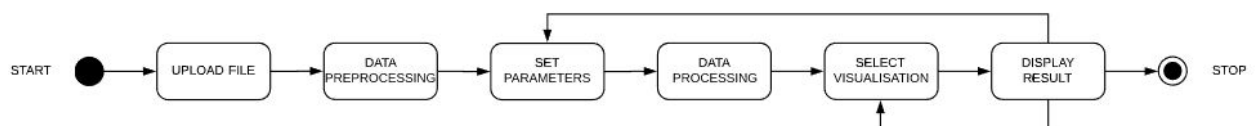
## b. Component Diagram



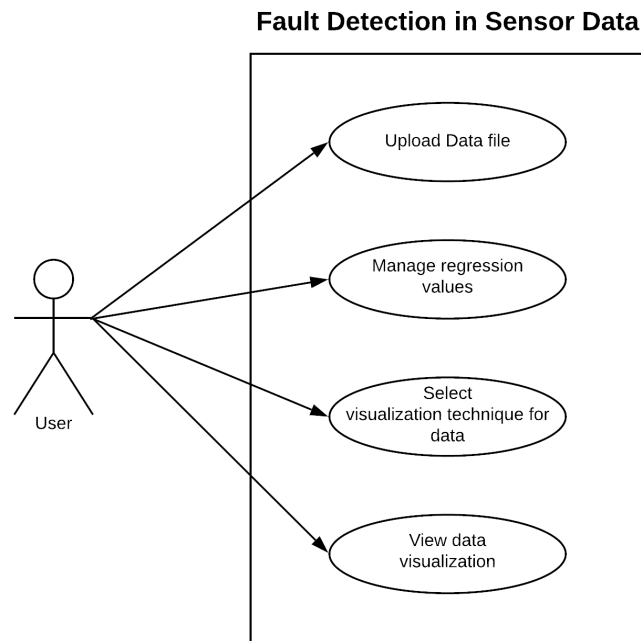
## c. Package Diagram



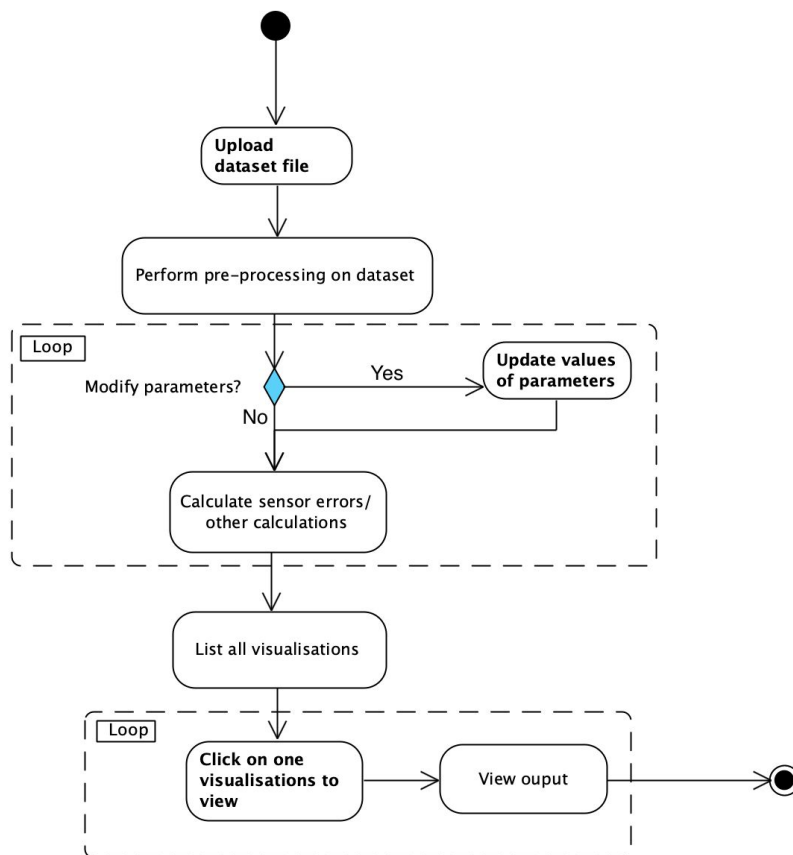
## d. State Machine Diagram



e. Use Case Diagram

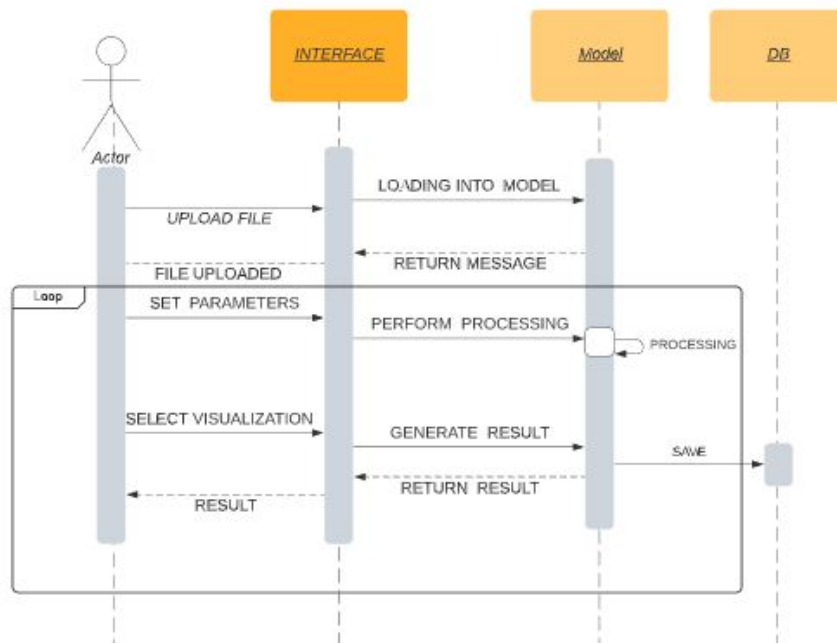


f. Activity Diagram

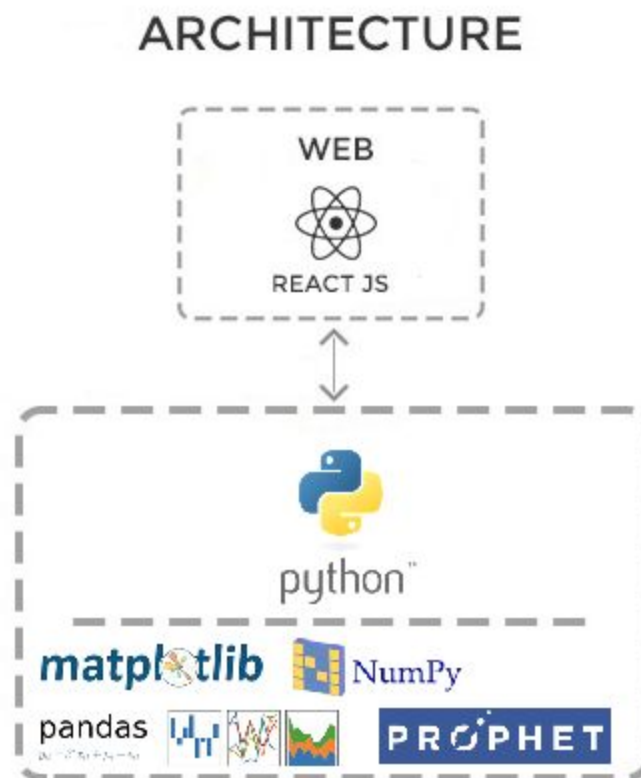




g. Sequence Diagram



h. Architecture Diagram



## Chapter 7: Conclusion

This project was successful in accomplishing the task of identifying faults and anomalies in turbofan dataset [6]. The tool created as a part of the project has been trained on the FD\_train files and they were tested on FD\_test files for accuracy. The Model created using the Prophet library gave us +3.47% accuracy over the model created using nearest-neighbor algorithm.

Further, Our tool is customised to show anomalies on the frontend in an easier understandable way to the user (who might not be a domain expert). Even though the dataset did not provide any information regarding the type/relation of sensors, our tool derived a correlation table showing the similarity between sensors. The correlation values were developed using Pearson correlation threshold [7]. The result shows the co-linear (positive or negative) dependency between 2 sensors, so with relationship we could easily deduce the potential candidates (sensors) which might have anomalous datapoints.

Some of the advantages, disadvantages and further improvements of our tool is listed below:

Pros:

- 1) Open Source library
- 2) Produces a specific range for every datapoint by considering data history.
- 3) Little to no use of the domain knowledge.
- 4) Tunable parameters for precision.

Cons:

- 1) Time-Consuming when dataset is huge.
- 2) Can process only one column of time-series data at a time.
- 3) The Prophet library currently only supports Python & R programming languages.

Further Improvements:

- 1) Passing dataframes to Prophet's predict function needs to be parallel programmed.
- 2) Different graphs can be added to analyze other functionalities like percentage of failure, remaining useful time (RUT) of the engine.

# References

- [1][https://en.wikipedia.org/wiki/Moving\\_average#:~:text=In%20statistics%2C%20a%20moving%20average,of%20finite%20impulse%20response%20filter](https://en.wikipedia.org/wiki/Moving_average#:~:text=In%20statistics%2C%20a%20moving%20average,of%20finite%20impulse%20response%20filter)
- [2]<https://facebook.github.io/prophet/>
- [3]<https://recharts.org/>
- [4][https://en.wikipedia.org/wiki/Additive\\_model](https://en.wikipedia.org/wiki/Additive_model)
- [5]<https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/>
- [6]<https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan>
- [7][https://en.wikipedia.org/wiki/Pearson\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient)