

An Exploratory Analysis of the industrial machine's data for
predictive maintenance operations

by

Faisal Ahmed Shovon

16101144

Shahriar Raz Fahim

16101157

K.C. Zamiul Alam

16101143

Sajib Mohajan

16101137

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2021

© 2021. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The submitted thesis is original solid work as a mandatory part for obtaining degree at Brac University.
2. The thesis material is not published or written by a third party, while providing due credit to the stakeholder with appropriate citation through full and accurate referencing.
3. The thesis or its any content material is not used or accepted for accomplishing any other degree or diploma at a university or other institution.
4. Proper acknowledgement of all main sources of help.

Student's Full Name & Signature:



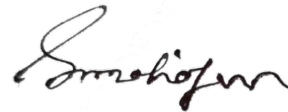
Faisal Ahmed Shovon
16101144



Shahriar Raz Fahim
16101157



K.C. Zamiul Alam
16101143



Sajib Mohajan
16101137

Approval

The thesis/project titled “Predicting optimum maintenance time interval of various industrial machines using different Machine Learning algorithms” submitted by

1. Faisal Ahmed Shovon (16101144)
2. Shahriar Raz Fahim (16101157)
3. K.C. Zamiul Alam (16101143)
4. Sajib Mohajan (16101137)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 8, 2021.

Examining Committee:

Supervisor:
(Member)



Amitabha Chakrabarty, PhD
Associate Professor, School of Data and Sciences
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Mahbubul Alam Majumdar, PhD
Professor and Dean, School of Data and Sciences
Department of Computer Science and Engineering
Brac University

Abstract

Modern industries nowadays heavily rely on hefty machineries which have lots of moving parts and contain sensor data. These sensor data are indexed in time order which are referred to as time series data. Industrial machines have a huge maintenance cost and failure risks involved with them. Sometimes, a lot is at stake for the companies for preserving the health of these machines. Also important machineries like airplanes need to be maintained on a regular basis in order to prevent any kind of disaster while in operation. Effective maintenance of these equipment are crucial to avoid several damage, downtime for repair and to prevent any mishap which is easily avoidable. Predictive Maintenance is a prominent strategy for dealing with maintenance issues given the increasing need to minimize downtime and associated costs. Time series data plays an important role in this field. We have implemented SVM, Logistic Regression and Random Forest model for classification on which we got 94% accuracy on an average after using different metrics. Moreover, we used LSTM and ARIMA for forecasting future values where LSTM performed better with an accuracy of 38.7%. Due to the imbalance on the data, the accuracy for classifying failure rate is very poor. Our goal is to explore and analyze different approaches of dealing with the time series data of industrial machines for using them to train different types of machine learning models and compare the performances of each approach.

Keywords: Predictive Maintenance, Machine Learning, Time Series Analysis, Failure Rate, LSTM.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption. Secondly, to our supervisor Amitabha Chakrabarty,(PhD) sir for his kind support and advice in our work. He helped us whenever we needed help. Thirdly, Md. Golam Rabiul Alam,PhD sir and the whole Thesis Coordinator panel of Fall 2019 and Spring 2020. All the reviews they gave helped us a lot in our later works. Finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
1 Introduction	2
1.1 Problem Statement	4
1.2 Objective and Motivation	4
1.3 Thesis Orientation	5
2 Literature Review	6
2.1 Background	6
2.1.1 Time Series Terminologies	11
2.1.2 Random Forest	17
2.1.3 ARIMA	19
2.1.4 Logistic Regression	20
2.1.5 Support-Vector Machine:	22
2.1.6 Recurrent Neural Network:	24
3 Methodology and Result	27
3.1 Workflow	27
3.2 Dataset Description	28
3.3 Data Preparation	30
3.4 Algorithm Workflow	40
3.4.1 SVM	40
3.4.2 ARIMA	42
3.4.3 Logistic Regression	43
3.4.4 RNN	45
3.4.5 Random Forest	48
3.5 Result and final thoughts	49

4 Conclusion	52
4.1 Conclusion	52
4.2 Future Plan	52
Bibliography	54

List of Figures

1.1	Reactive Maintenance	2
1.2	Preventive Maintenance	3
1.3	Predictive Maintenance	4
2.1	Time Series Data of Machine Voltage	12
2.2	Stationary Data	13
2.3	Non-Stationary Data	14
2.4	Lag Plot	15
2.5	Autocorrelation Example (lag=30)	16
2.6	Partial Autocorrelation Example (lag=30)	17
2.7	Methodology of Random Forest	18
2.8	ARIMA model forecasting	20
2.9	Logistic Function	21
2.10	SVM Algorithm	23
2.11	RNN Architecture	24
2.12	Types of RNN	25
2.13	Architecture of LSTM	26
3.1	Workflow Diagram	27
3.2	Sample Data	28
3.3	Machine Description	29
3.4	Error Count for MachineID 1	29
3.5	Scatter Plot of Machine 1 and Machine 2	30
3.6	Machine 3 data	31
3.7	Visualization of the Machine 5 data	32
3.8	ACF Plot for voltmean	35
3.9	ACF Plot for voltsd	35
3.10	ACF Plot for rotatemean	35
3.11	ACF Plot for rotatesd	36
3.12	ACF Plot for pressuremean	36
3.13	ACF Plot for pressuresd	36
3.14	ACF Plot for vibrationmean	37
3.15	ACF Plot for vibrationsd	37
3.16	PACF Plot for voltmean	37
3.17	PACF Plot for voltsd	38
3.18	PACF Plot for rotatemean	38
3.19	PACF Plot for rotatesd	38
3.20	PACF Plot for pressuremean	39
3.21	PACF Plot for pressuresd	39

3.22	PACF Plot vibrationmean	39
3.23	PACF Plot for vibrationsd	40
3.24	Classification report for SVM model	40
3.25	Confusion Matrix (Kernel = 'Linear')	41
3.26	Classification report for SVM model	41
3.27	Classification report (Kernel = 'Linear')	42
3.28	Forecast using ARIMA(0,1,0)	42
3.29	Forecast using ARIMA(1,1,2)	43
3.30	Confusion Matrix of Logistic Regression	44
3.31	AUC Curve	45
3.32	Feature Explanation	46
3.33	Percentage of data	46
3.34	Final Prediction	47
3.35	Final Prediction plot	48
3.36	Confusion Matrix for Random Forest model	49
3.37	Data Distribution	50
3.38	Performance Analysis(Classification)	50
3.39	MAE plot	51
3.40	Model loss plot	51

Chapter 1

Introduction

The increasing availability of data is changing the way decisions are taken in industry [12] in important areas such as scheduling [11], maintenance management [14] and quality improvement [9]. Machine Learning (ML) approaches have been shown to provide increasingly effective solutions in these areas, facilitated by the growing capabilities of hardware, cloud-based solutions, and newly introduced state-of-the-art algorithms. At the same time the efficient management of maintenance activities is becoming essential to decrease the costs associated with downtime and defective products [1], especially in highly competitive advanced manufacturing industries. Approaches to maintenance management can be grouped into three main categories which, in order of increasing complexity and efficiency [13], are as follows:

- i. Run-to-Failure (R2F) - where maintenance interventions are performed only after the occurrence of failures. It is also known as Reactive maintenance approach. This is obviously the simplest approach to deal with maintenance (and for this reason it is frequently adopted), but it is also the least effective one, as the cost of interventions and associated downtime after failure are usually much more substantial than those associated with planned corrective actions taken in advance. Moreover, this approach can not be implemented for highly essential machines like life support equipment for patients as the failure of the machine will be fatal. For non-critical machines like Fan, Air Conditioner, this model can be used to be cost effective and to use the full life cycle of the machine.

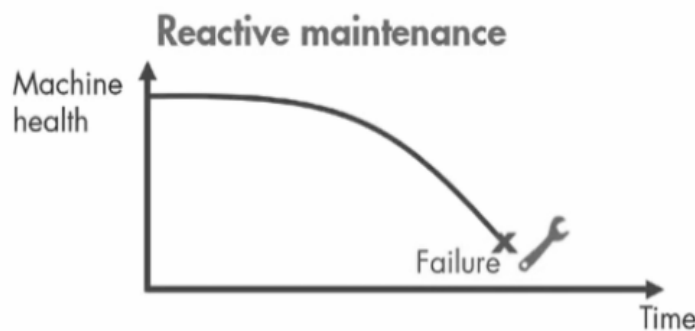


FIGURE 1.1: Reactive Maintenance

ii. Preventive Maintenance (PvM) - where maintenance actions are carried out according to a planned schedule based on time or process iterations. This approach follows a specific time frame for maintenance based on the type of machine and the significance of the machine's operability. With this approach, also referred to as scheduled maintenance, failures are usually prevented, but unnecessary corrective actions are often performed, leading to inefficient use of resources and increased operating costs. This ultimately wastes the remaining useful life (RUL) of a machine by performing early maintenance which leads to more time and resource consumption. Also it reduces the machine's operational life cycle significantly.

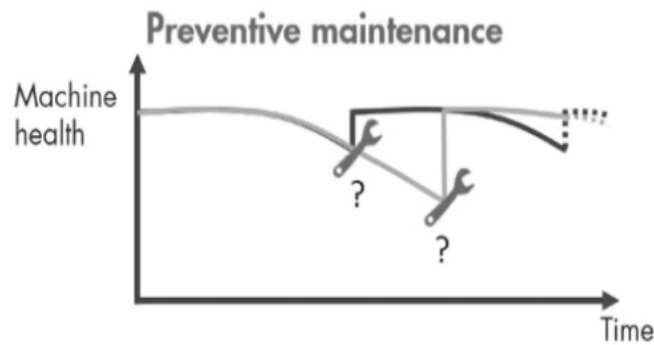


FIGURE 1.2: Preventive Maintenance

iii. Predictive Maintenance (PdM) - where maintenance is performed based on an estimate of the health status of a piece of equipment [3]. PdM systems allow advance detection of pending failures and enable timely prefailure interventions, thanks to prediction tools based on historical data, ad hoc defined health factors, statistical inference methods, and engineering approaches. Predictive maintenance uses condition-monitoring equipment to evaluate an asset's performance in real-time. A key element in this process is the Internet of Things (IoT). IoT allows for different assets and systems to connect, work together, and share, analyze and action data. IoT relies on predictive maintenance sensors to capture information, make sense of it and identify any areas that need attention [10]. Some examples of using predictive maintenance and predictive maintenance sensors include vibration analysis, oil analysis, thermal imaging, and equipment observation. A great difficulty in dealing with predictive maintenance is the processing of huge amounts of data. In order to be able to make reliable statements about the condition of machines and plants and thus to be able to detect malfunctions as quickly as possible, it is necessary to collect large amounts of data. These data must be stored, processed and analyzed using intelligent algorithms. Furthermore, even if the data is readily available, the feature extraction process to find the perfect classification model is an even more complex task to pull off.

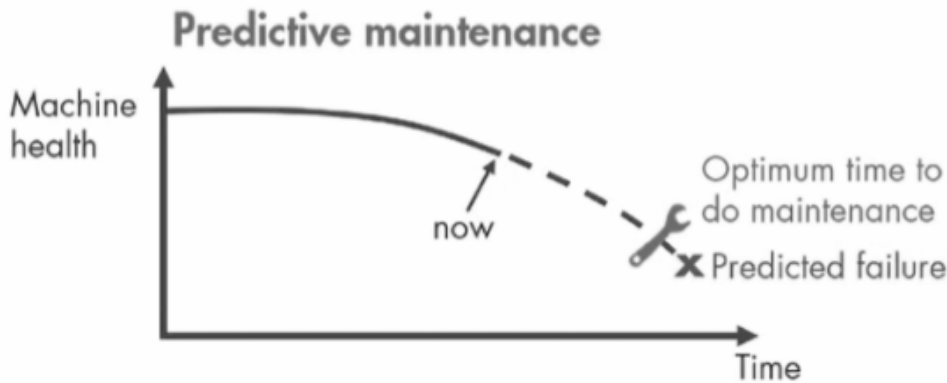


FIGURE 1.3: Predictive Maintenance

Overcoming the common obstacles faced in the predictive maintenance model is the most important task of this research.

1.1 Problem Statement

Our goal is to predict the optimum maintenance time interval for industry machineries using different machine learning algorithms and measure the effectiveness of each of the algorithms used on the dataset.

Saving cost on unnecessary maintenance is the main purpose of predictive maintenance techniques. If we do not use effective techniques to find out the maintenance time for these industrial machineries, the production of important goods will be hampered as well as the manufacturer will have to bear the unwanted expenses which will weaken their financial power.

We will use machine learning algorithms to predict when to perform maintenance for optimum results and find out which algorithm works best on which type of sensor data.

1.2 Objective and Motivation

A lot of research have been done in the field of Predictive Maintenance since the rise of popularity of machine learning algorithms. However, most of the work has been done using particular algorithms. Our motivation to work on this topic was to incorporate several machine learning algorithms for building classification and forecasting model of time series data of industrial machines for predictive maintenance and evaluate the performance level and share our findings. This research will contribute towards the ongoing development of predictive maintenance operations. Being able to predict when a machine might fail and identifying the optimum time of performing maintenance is a valuable asset when it comes to saving maintenance cost and expensive equipment.

1.3 Thesis Orientation

1. Chapter 1 is Introduction where motivation, problem statement, objectives are discussed.
2. Chapter 2 is literature review where background and related work are discussed. In the literature review part, it reviews the previous work.
3. Chapter 3 is Methodology and Result, here we discussed about the algorithms we implemented and reviewed our results.
4. Chapter 4 is Conclusion and Future works where we discussed our work till now and scope of improvement.

Chapter 2

Literature Review

2.1 Background

Predictive maintenance is a technique that uses condition-monitoring tools and techniques to track the performance of equipment during normal operation to detect possible defects and fix them before they result in failure. In this report we are trying to work on predictive maintenance in industrial machines to minimize the time the equipment takes for being maintained. We have already read some papers where there were some works based on similar topics.

In the very first paper [17] the authors used Multiple Classifier which effectively deals with the unbalanced datasets that arise in maintenance classification problems. Multiple classifiers are a set of classifiers whose individual predictions are combined in some way to classify new examples. The integration of the classifiers improves the overall prediction. The researchers labeled the failed state as ‘F’ for the last ‘m’ number of iterations, which provides more conservative maintenance recommendations by choosing larger values for the failure horizon. According to the researchers, the performance of this methodology increases with the number of classifiers.

One another paper [16] implements a Multi-Instance Learning (MIL) algorithm. As they discussed, the maintenance strategy was determined separately for different components and it was influenced by various types of factors like repair cost, failure intensity, failure severity and the business model. Therefore, enabling proactive functional response and preventing unexpected equipment failures will be ensured too. They compared this algorithm against others like All Instances, Aggregated, MILES, MI-SVM to justify that their approach significantly outperformed other algorithms.

Next paper [23] which we had gone through was about ‘Cost Sensitive Learning’ for predictive maintenance. It showed that by using algorithms it can result in remarkable cost reduction and fault tolerant policies. We saw a lot of explanations about illustrated data processing, feature extraction and model selection/evaluation in detail. In the experiment, they considered behavioral data collection for a total number of 15,924 devices. However, according to them they extracted 23 categorical as well as 7 numerical features. In addition, they derived cost function which led to

less performance in terms of F1-score and thus achieved much higher savings which was almost 16k devices with 1 week prediction interval. Finally, they discovered a mismatch between PDM performance criteria and the business requirements and also they have deployed their solution in production environments and extended it to different use cases in future.

This paper [20] describes the workflow of a slitting machine where the authors have indicated that the two most prominent factors responsible for producing low quality packaging rolls are tension and pressure. A preoccupied tension and pressure is provided to the system by an operator which is maintained by the system itself depending on the parameters- roll diameter, roll width and roll length. With the parametric changes in the system, the programmable logic controller (PLC) on board sends signals to the required actuators to keep the system running. Their system setup uses PLC for recording values which is later converted to TCP form and the values are stored in Industrial Personal Computer (IPC). Using MQTT protocol, the IPC was in charge of sending the data to the cloud in order to predict failures beforehand. However, while analysing their data, they replaced the null values by mean values and outliers were done with the usage of clustering. This analysis showed that a bad production cycle occurs when there is a sudden drop in pressure. They processed their data with various supervised models like Deep Neural Network, Support Vector Machine (SVM) and CART to train a classifier to detect failures in the production cycle among which deep neural network model was the most efficient in modeling the data. They also used Autoregressive Integrated Moving Average (ARIMA) for determining the future conditions of the machine. The researchers stated that the events of the bad quality cycles are low when contrasted with the good quality cycle; the model keeps on effectively learning with the recent datas and continues updating the logs. They are also working on the possible future extensions for this model which are predicting the Remaining Useful Life (RUL) of machines, anomaly detection and controlling machine parameters.

In another paper[15], the authors have worked with real-world test cases of aircraft behavior, collected from an aerospace corporation named ‘Airbus.’ The authors have proposed two data driven approaches of fault prediction, which are - 1) degradation detection and trending and 2) prediction of rare events based on classification. However, the objective of the continuous task in Airbus is to build up a robotized system for the early alerts for conceivable exorbitant flaws. To develop such a system, a few issues should be thought of. First of all, a multidimensional data tracer has to be formed to identify the degradation measures in the system. Then the second issue is a matter of rare event prediction to have the option to envision some particular groups of shortcomings. For detection of degradation, they used a Support Vector Machine (SVM) where if an abnormal activity is detected, the next step is to predict remaining time before the fault. In view of a few instances typical carried on airplanes, they prepared a one class SVM model and applied it to the out-of-test airplane with and without degradation. The discovery of abnormality (degradation measure) was performed on comparing time arrangement, expecting that if there arises an occurrence of debasement, it indicates that an anomaly has been detected. For the modeling of multidimensional degradation behavior in time they used the Autoregressive Integrated Moving Average (ARIMA) model which allows to predict

the number of flights left before the maintenance event is actually required. Lastly, to predict certain failures they used the parameter selection approach based on the Logistic Regression model. To assess the predictive capability and execution of the methodology, they compared it with simple thresholding and their approach had remarkably surpassed the simple thresholding. To demonstrate the proposed idea they analyzed two cases from AIRBUS. The main case is a utilization of multidimensional degradation detection and predicting whether there is any possibility of abnormal behavior in the airplane cooling system. Second case is an expectation of some gathering of particular failures in an airplane subsystem that occur with frequency of 0.2-1%.

Susto and Beghi [19] generated a methodology called Supervised Aggregative Feature Extraction (SAFE) to deal with the problems of predictive maintenance when using time series data. They pointed out some modelling difficulties such as dealing with high-dimensionality, data fragmentation etc. when using the regular regression techniques. Moreover, information loss persists as an issue when using the typical approach of trying to extract a homogenous set of features. According to them, with the help of bypassing the phase of feature extraction, SAFE- a supervised regression methodology has been tested in a predictive maintenance problem for the first time. Their drift from the conventional regression models depends on several techniques. Firstly, a cost function F has been generated which is basically responsible for determining the probable error of the training dataset, S . The weighted integration of a particular time series with its corresponding cost function produced the desired output of each time series. Moreover, a regularization function denoted by R was used for determining the complexity of the proposed model. A form of linear combination of Gaussian densities named Radial Basis Function was also in use for parameterizing the shape functions. Lastly, for tackling the problem of data points being irregular and noisy, Gaussian Process approximation has been utilized. The dataset they used was an actual industrial dataset of semiconductor Ion implantation processes. A total of 3671 observations were present in that dataset. With this input of time series data, they tried to predict the remaining useful time (RUL) of the component (filament). From a total of 124 regressors, 4 quantities were pulled out for each time series. In addition to that, 5 intervals and median values of each interval were generated too. The ratio of training data- test data split was 70 to 30. Even, to ensure the elimination of bias in splitting the data, Monte Carlo cross-validation had been employed too. Thus, with the help of this model, they showed that their proposed model performed better than classical feature extraction procedures. They did not fail to mention that by studying their findings respect to Type I and Type II errors, their experimental part of the study could get more furnished.

For dealing with a large size of time series data collected from a vehicle's engine control unit[21] used a data driven framework which processes data through pattern recognition and utilizes machine learning approaches called OXYCLOG. As oxygen sensor is a crucial part of automobile industry, they tried to predict the state of the oxygen sensor (clogged, unclogged or almost clogged). To achieve this, a set of recordings from the sensors over a significant amount of time span was considered. Two different programmes were used. Though both of these programmes gathered data from the same clock cycles, their key interests were different. With the help of

these two programmes, accelerator signals of the time when cutoff occurred could be monitored and analyzed. In the preprocessing of the data, the first objective was to reduce the number of variables. Exploiting Pearson Correlation, the correlation among different features were computed and thus the redundant features were identified. In addition to that, CORR-FS, an ad-hoc algorithm was made use of too. To make the number of data feasible, the signals were summed up by the means of standard derivation and average. Moreover, OXYCLOG made use of the distribution of the derivatives' values to process and use the time component to some degree. By taking the advantage of using the second programme (programme B) and SEMI-SUPERVISED-LABELING (SSL) algorithm, the cycles were labeled. Decision tree, Support Vector Machine and Artificial Neural Network- these three machine learning algorithms were in action for getting the actual result. Though the authors conceded that the accuracy of the classifiers is not unquestionable as the most interesting class (red class which indicates the oxygen sensor being clogged) possessed the lowest number of cycles, effort was put on making the classifiers to produce as accurate results as possible using techniques such as 10-fold cross validation and exhaustive grid search. Though decision tree performed the worst among the three models, still the results were fairly decent. Moreover, the internal mechanism of decision tree itself was quite helpful in providing insights regarding feature selection. As the cycles were classified in three classes - red (clogged), yellow (intermediate), green (unclogged), the predictions were classified accordingly too. A strong positive of the models is, the models misclassified only between adjacent classes. Thus, this study provides very useful insights on feature selection, turning a time dependent problem into a time independent one and labeling of data through exploiting diverse data sources. The authors did mention that the study could be developed more by studying large amounts of data collected from different contexts and thus a more generalized approach could be reached.

Another interesting approach of tackling predictive maintenance problem could be found in [18]. This proposed a model which is based on supervision and prognosis tools. They dealt with thermoregulator- an important industrial component. As every prognostic approach has its own pros and cons and generating a mathematical model for dynamic and complex systems is not quite easy that is why their proposed prognostic technique is a hybrid one. Moreover, Possibility Function by Episode (PFE) of components has been utilized in lieu of Possibility Function. According to their dataset and intention a semi supervised learning would be appropriate, thus AUDyC has been chosen among the available modern pattern recognition approaches as it is quite adequate for supervising dynamic and complex systems. Possibility function by episode has been calculated with the help of the estimation of degradation of the components based on real time. In addition to that, for building the architecture a dysfunctional analysis technique called FMECA has been utilized too as dysfunctional techniques are quite useful and common for defining maintenance strategies. Three modules are responsible for the supervision. The evaluation of the present state of all the components is monitored by the monitoring module. The diagnosis module is in charge of finding out the faults and then isolating them. Then, based on the analysis provided by FMECA, a Fault Tree Analysis (FTA) could be generated. This FTA considers all the environmental factors and the internal causes which affect the whole system and hence provides an overall better understanding of

the current state of the system. In addition to that, the components which might get affected by the faults and the classification of these faults could be found with the help of drift detection tool. After this step the prognosis was conducted accordingly. At this stage, the system will be under close observation and the modes in which the thermoregulator operates will be characterized by appropriate adaptive modeling tool. Determining PFE of each identified component could be conducted by the proposed detection and diagnosis approaches. At last, the prediction of PFE of each component is made with the help of the prognosis method. This approach produces an acceptable result with an error margin percentage of 20. This architecture could be further developed by applying it on other evolving systems and industry related systems.

This paper [24] discusses about carrying out predictive maintenance of an Air booster compressor (ABC) motor which shows the performance by applying RNN-LSTM (Long short-term Memory) algorithm. Instead of different neural network structures, in this paper recurrent neural network was chosen because of the time series dataset as the data is independent of each other. To improve data quality and make faster prediction the authors used data preprocessing and they gathered data over the period of two years. The data is categorized into two sections which are healthy and unhealthy data and again the unhealthy data is divided into three parts such as shutdown, outliers and out of range fault. In the development step they utilized the data into training, validation and testing to test the accuracy and for using the validation process they did make sure that the data is not overfitting. Their performance of the model depends on various parameters such as architecture of model, network parameters and training function which generates lower root mean square error. This paper shows that when the value for the process variable for the Air booster compressor motors is within the range then the motor is safe to be operated and performs in a fully functional way. However, when the value has exceeded the limit, there will be prediction alerts to remind that it needs to be stopped immediately and it would be quite harmful to operate a damage motor. The disadvantage of this paper is the long training time as it takes about 30 minutes to predict the signal. In order to get an optimal model with weights, they used fourteen parameters which are chronologically trained.

In this paper [7] condition-based support procedures for mechanical hardware and forms are depicted in this paper along with illustrations of their utilize and talk of their benefits. These procedures are partitioned here into three categories which employments signals from existing prepare sensors, such as resistance temperature finders (RTDs), thermocouples, or pressure transmitters to assist verify the execution of the sensors and process-to-sensor interfacing additionally to recognize issues within the handle and this study[4] also uses diagnosing strategies of the ball and cylindrical roller component bearing absconds were investigated by vibration observing and spectral investigation as a predictive maintenance apparatus. It was appeared that ball and cylindrical roller bearing absconds were advanced in indistinguishable manner without depending on rolling component sort.

The author centers about detection, area and determination of untrue in pivoting responding apparatus utilizing vibration investigation in this paper[2]. In addition, this

paper[6]improves an coordinates neural-network-based system for predictive maintenance of rotational hardware. The coordinates framework is platform-independent and is pointed at minimizing anticipated fetched per unit operational time and also in this paper[8] the author deals with a blame conclusion framework which incorporates a information collection work that procures time histories of chosen factors for one or more of the components, a pre-processing work that calculates indicated characteristics of the time histories

In [10], the authors focused on an online monitoring system which depends on sensor automated inputs. It creates a nonlinear model which will predict on time possible changes in vibration tendencies. They have chosen a mill fan which is from Maritsa East 2 power plant. The sensor consists of various types of functions like flow level, pressure, temperature, power, vibration and so on. The focused work is to differentiate the recurrent neural network with the Elman RNN architecture. Before that they prepare to compare two types of recurrent neural network-one is historical Elman architecture and another is Echo stet networks. Their initial experiments notice better approximation and rapid training abilities of Echo stet networks in difference with the Elman network. They use a mill fan from Maritsa power plant because in this power plant Experion process knowledge system which is a highly cost-effective system that provides secure, vigorous, adaptable, plant wide service with phenomenal network through all levels of the plant as outlined within the taking after high-level see of the design. Their mill fan system has four mills per boiler and four radial bearings. They consider prognostic maintenance which is based on the vibration of the nearest to the processor rotor bearing square. In their use of two algorithms, they use the hidden layer which is called reservoir or hidden neurons that have connections from input. As they use real industrial conditions, they use wavelet de-noising method for vibration sources. They categorized data into two sections which are training and testing sets. The algorithm is trained by the measurement with only one-minute time step. Then every ten minutes the number of training and testing data has been decreased ten times. Both RNN structures are prepared to predict vibrations adequacy of a process fan framework. The results demonstrated the prevalence of ESN structure with respect to the information fitting exactness and preparation time required.

2.1.1 Time Series Terminologies

Time series data: Time series is a set of observations on the values that a variable takes at different times. The series may be denoted by $X_1, X_2 \dots X_t$, where t refers to the time period and 'x' refers to the value. It is a sequence taken at successive equally spaced points in time.

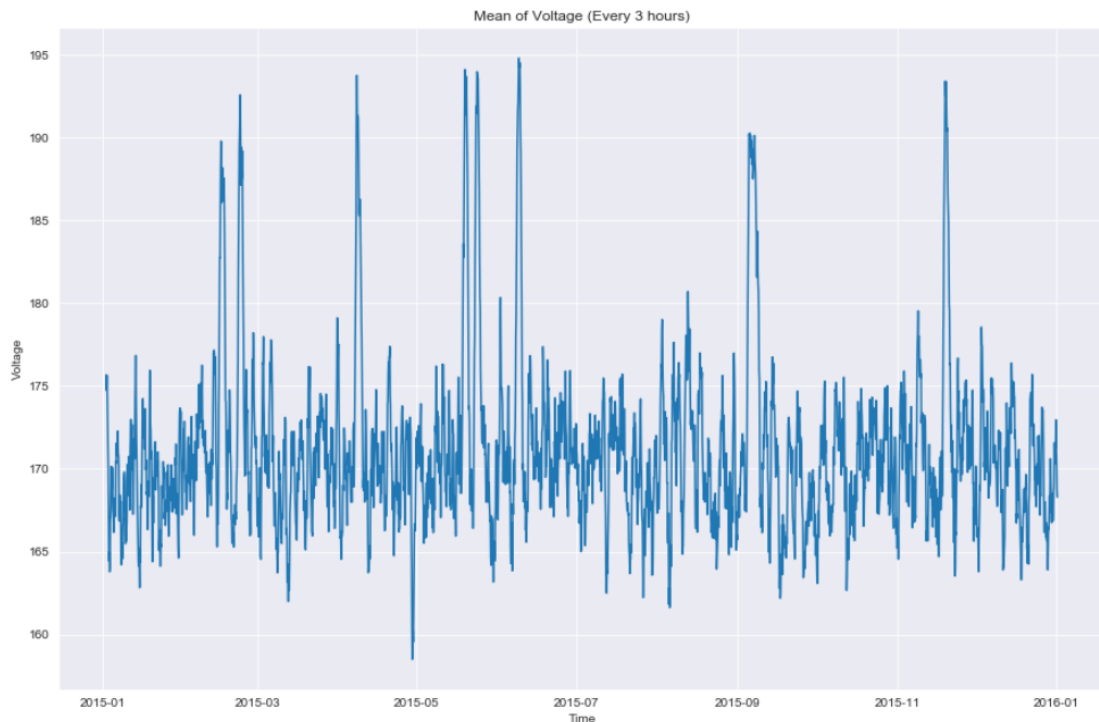


FIGURE 2.1: Time Series Data of Machine Voltage

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. Time series analysis is similar to another common approach, which is regression[5]. Like regression, time series analysis is often focused on identifying underlying trends and patterns, describing them mathematically and making a prediction or forecast about what will happen in near future.

Deterministic Time Series: If the X 's are exactly determined by a mathematical formula, the series is said to be deterministic.

Stochastic Time Series: If future values can be described only by their probability distribution, the series is said to be a statistical or stochastic process.

Stationarity: Stationarity shows the mean value of the series that remains constant over a time period; if past effects accumulate and the values increase toward infinity, then stationarity is not met. A time series has stationarity if a shift in time doesn't cause a change in the shape of the distribution. Basic properties of the distribution like the mean, variance and covariance are constant over time.

There are many types of stationary-

- **Strict stationarity** means that the joint distribution of any moments of any degree (e.g. expected values, variances, third order and higher moments) within the process is never dependent on time. This definition is in practice too strict to be used for any real-life model.

- **First-order stationary** series have means that never changes with time. Any other statistics (like variance) can change.
- **Second-order stationary** (weak stationarity) time series have a constant mean, variance and an auto covariance that doesn't change with time. Other statistics in the system are free to change over time. This constrained version of strict stationarity is very common.
- **Trend-stationary** models fluctuate around a deterministic trend (the series mean). These deterministic trends can be linear or quadratic, but the amplitude (height of one oscillation) of the fluctuations neither increases nor decreases across the series.
- **Difference-stationary** models are models that need one or more differencing to become stationary (see Transforming Models below).



FIGURE 2.2: Stationary Data

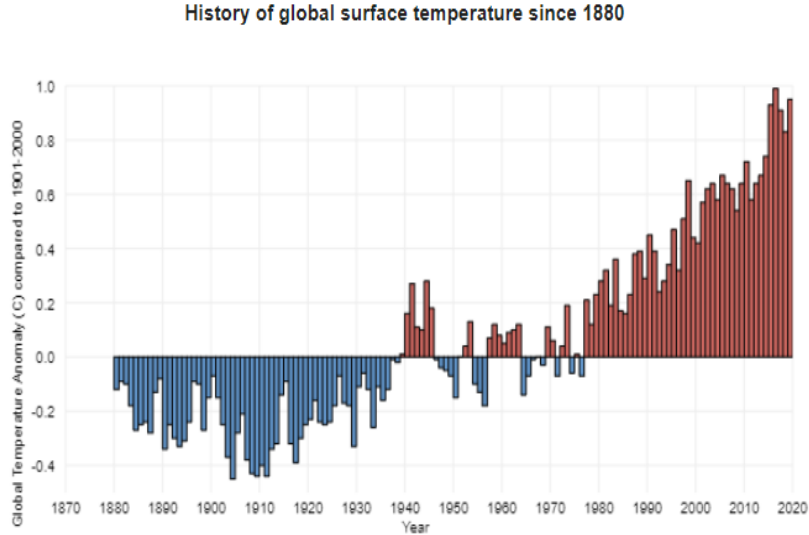


FIGURE 2.3: Non-Stationary Data

A time series with cyclic behavior can still be stationary if the cycle are not of a fixed length. Figure (2.2) shows an example of stationary data and figure (2.3) shows an example of non-stationary data. It is really important for a time series to be stationary in order to extract information through machine learning models. If the data is not stationary, it cannot be forecasted using traditional time series models. Trend is when there is a long-term increase or decrease in the data. Seasonality is a reoccurring pattern at a fixed and known frequency based on a time of the year, week, or day.

Differencing: Differencing is used to make the series stationary, to remove seasonality or De-trend and to control the auto-correlations. However, some time series analysis do not require differencing and over-differenced series can produce inaccurate estimates. Differencing can be performed by computing the differences between consecutive observations. Depending on the data, order of differencing can be one or more than one.

- **First order differencing** is when the data is differenced only once.

$$Y_t = Y_t - Y_{t-1} \quad (2.1)$$

- Most of the time, differenced data will not appear to be stationary and it may be necessary to difference the data a second time to obtain a stationary series. This is called **Second-order differencing**.

$$\begin{aligned} Y_t &= Y'_t - Y'_{t-1} \\ &= (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \\ &= Y_t - 2Y_{t-1} + Y_{t-2} \end{aligned} \quad (2.2)$$

- **Random walk model** assumes that in each period the variable takes a random step away from its previous value. A random walk is a time series

$$Y_t = Y_{t-1} + w_t \quad (2.3)$$

where w_t is a discrete white noise where all values are independent identically distributed with a mean of zero.

- Seasonality occurs when time series data exhibits regular and predictable patterns at time intervals that are smaller than a year. Seasonality can be removed from the time series. This process is called Seasonal Adjustment, or Deseasonalizing, which can be done using **Seasonal differencing**. It is the difference between an observation and the previous observation for the same season.

$$Y'_t = Y_t - Y_{t-m} \quad (2.4)$$

where m is the number of seasons. These are also called “lag- m differences,” as we subtract the observation after a lag of m periods.

Lag: Lag is essentially target values from previous periods in time series data. The k th lag is the time period that happened at k time points before time i . The most commonly used lag is 1, which is called first-order lag.

$$\text{Lag}_k(Y_{k+1}) = Y_k \quad (2.5)$$

Lag plots are one set of observations in a time series is plotted (lagged) against a second, later set of data. The shape of the lag plot can provide clues about the underlying structure of your data. For example:

- A linear shape to the plot suggests that an autoregressive model is probably a better choice.
- An elliptical plot suggests that the data comes from a single-cycle sinusoidal model.

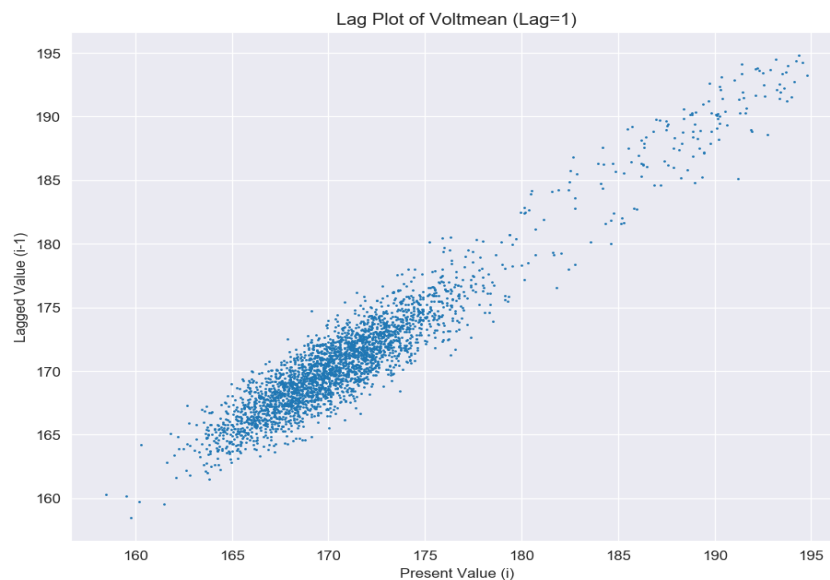


FIGURE 2.4: Lag Plot

Lag plots also enable us to check for outliers, randomness in the dataset and points out the correlation and seasonality. It depends on the dataset which lags work best and looking at correlation is one way to select the lag values.

Autocorrelation: Autocorrelation represents the degree of similarity between a given time series and a lagged version of itself over successive time intervals. It measures the relationship between a variable's current value and its past values. When computing autocorrelation, the resulting output can range from 1 to negative 1. Positive correlation is a relationship between two variables in which both variables move in the same direction. Negative correlation is a relationship between two variables in which one variable increases as the other decreases, that means they move in opposite directions.

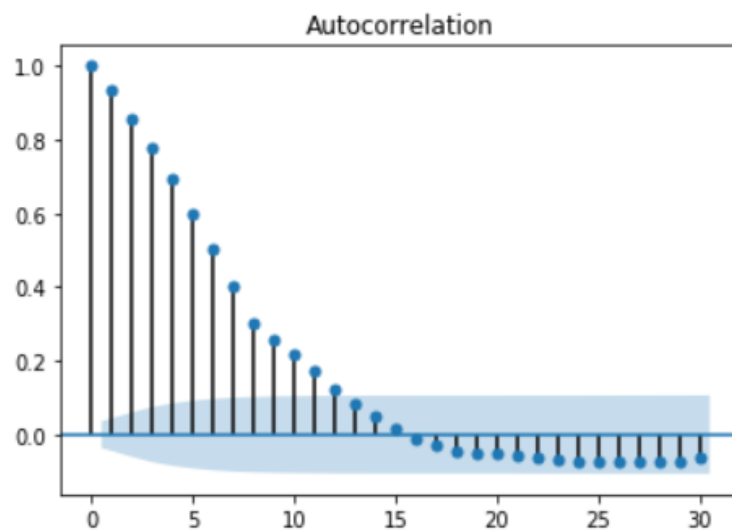


FIGURE 2.5: Autocorrelation Example (lag=30)

Autocorrelation is a time rescan be determined by Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF).

ACF is the complete auto-correlation function which gives us the value of the auto-correlation of any series with lagged values. It describes how well present values are related to its past values. When we plot these values along with a confidence band, we create an ACF plot. A time series has several components which includes seasonality, trend, cyclic and residual. The ACF takes all of these into account while finding correlations which is why it is the complete auto-correlation plot. In the Figure, we are seeing that values are highly correlated to the earlier lagged values, but the correlation fades as we go back further in time.

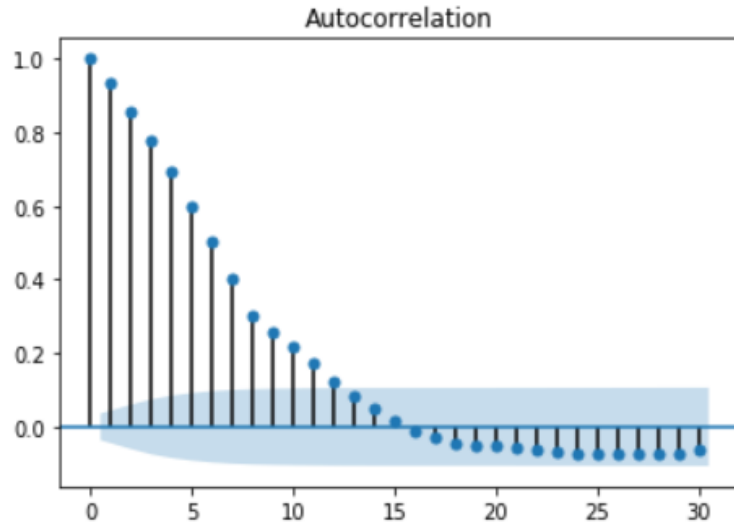


FIGURE 2.6: Partial Autocorrelation Example (lag=30)

Unlike ACF, the PACF finds correlations with the residuals (these are the values that remain after removing the other effects) with the next lag. So, if there is any remaining information which can be modeled by the next lag, we might get a good correlation and we will keep that lag as a feature when modelling.

When we predict a model, we do not want too many features which can create multicollinear issues. Therefore, we only keep the relevant features. Figure 2.6 shows that, the first 2 lagged values are have high positive correlation with the residuals and next lagged values have weaker correlation. In this case, taking two past lagged values of the current data in to account will be beneficial for classifying or forecasting future values.

2.1.2 Random Forest

Random Forest is one of the most prominent classification algorithms in data science. Machine learning would not be what it is today without the accumulation of these useful classification algorithms like Random Forest, Naive Bayes, Logistic regression, Support Vector Machine (SVM) etc. That is why it is important to take a thorough look in this algorithm and try to figure out why it would be efficacious for our own model of predictive maintenance.

A pre requisite for describing the mechanism of random forest is focusing on Decision Tree which is another classification algorithm and provides the base for Random Forest. Decision Tree's mechanism can be compared with the structure of nested if else conditions. A set of conditions need to be fulfilled before reaching the final decision. What decision Tree does is it sets up these relevant conditions by satisfying which the data gets categorized into an intermediary category. From that intermediary category with the help of more questions, the data gets further categorized and then reaches the final category which can be labeled as an attribute or a feature. The decision-making process is a binary one which means it can make decisions based

on yes/no answers.

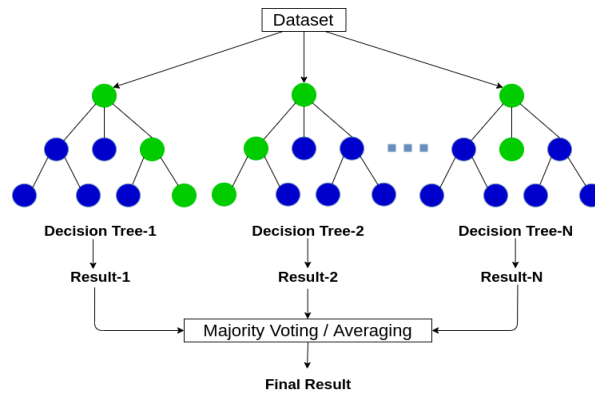


FIGURE 2.7: Methodology of Random Forest

Random Forest can be said to possess multiple Decision Trees. Multiple trees are needed because they differ in their decision-making conditions. Nonetheless, they do try to reach the same categorization through variable conditions. This is needed because variations of decision-making conditions help to categorize unclear data. Some datasets do not contain clear dependent features. Moreover, missing features or missing labels are one of the deadliest enemies for a researcher. This hazard can be solved by using multiple Decision Trees. As multiple trees have multiple decision-making conditions so it is possible to categorize the data through different routes and thus even if some labels are missing, best possible categorization can be achieved. Random Forest makes the best use of this multiple decision trees as it counts the category which prevails the most in different decision trees. For example, if we want to predict the color of a certain fruit which contains missing labels or features then different decision-making conditions can be set considering variations of useful parameters and thus different decision trees would predict the fruit color. Though at first glance it seems an inaccurate way of reaching a decision but it is safe to say that we can choose the color which has been predicted by the majority of the trees. By which margin the result dominates over the other categorizations can possibly indicate about the accuracy of the result. That is why this mechanism can be labeled as an efficient method to categorize data.

As we've already stated uncorrelated features processing is very much plausible in Random Forest and that is why we chose Random Forest as one of our classification algorithms. Our dataset does contain data which are not clearly labeled. So, it became really troublesome for us to classify the data and predict an outcome. Moreover, as the labels were missing, figuring out the correlation between different features became cumbersome too. Without having a clear notion about the correlation, we cannot actually build up an efficient model for our project. In addition to that, as random forest would allow us to experiment with our features, a trial-and-error methodology could be exploited. This would provide us a chance of finding out the necessary features to make our model a successful one.

For these reasons we thought Random Forest could be a good approach as it will be able to process uncorrelated data efficaciously and will not make decisions based on

a single Decision Tree which would accommodate only one kind of routes among the several rather the algorithm will try to reach the ultimate destination through a set of different conditions and then take the one which comes out as the most frequently occurred result.

2.1.3 ARIMA

ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is also known as the Box-Jenkins method, named after the statisticians George Box and Gwilym Jenkins. This is one of the easiest and effective machine learning algorithm to performing time series forecasting. Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models. ‘Auto Regressive’ in ARIMA means it is a linear regression model that uses its own lags as predictors. Linear regression models work best when the predictors are not correlated and are independent of each other. An ARIMA model with seasonal components becomes SARIMA model. This model is used when the time series exhibits seasonality. This model is similar to ARIMA models, just with few added parameters to account for the seasonality.

A standard notation is used which is ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used. The parameters of the ARIMA model are defined as follows:

- p is the number of ‘Auto Regressive’ (AR) term. It refers to the number of lags of Y to be used as predictors. If P= 3 then we will use the three previous periods of our time series in the autoregressive portion of the calculation.
- d is the number of differencing required to make the time series stationary,

If d = 0:

$$y_t = Y_t \tag{2.6}$$

If d = 1:

$$y_t = Y_t - Y_{t-1} \tag{2.7}$$

If d = 2:

$$y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} - Y_{t-2} \tag{2.8}$$

- q is the number of ‘Moving Average’ (MA) term. This variable denotes the lag of the error component, where error component is a part of the time series not explained by trend or seasonality. Here, q=1 means that there is auto-correlation with one lag.

In terms of y, the general forecasting equation is:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q} \tag{2.9}$$

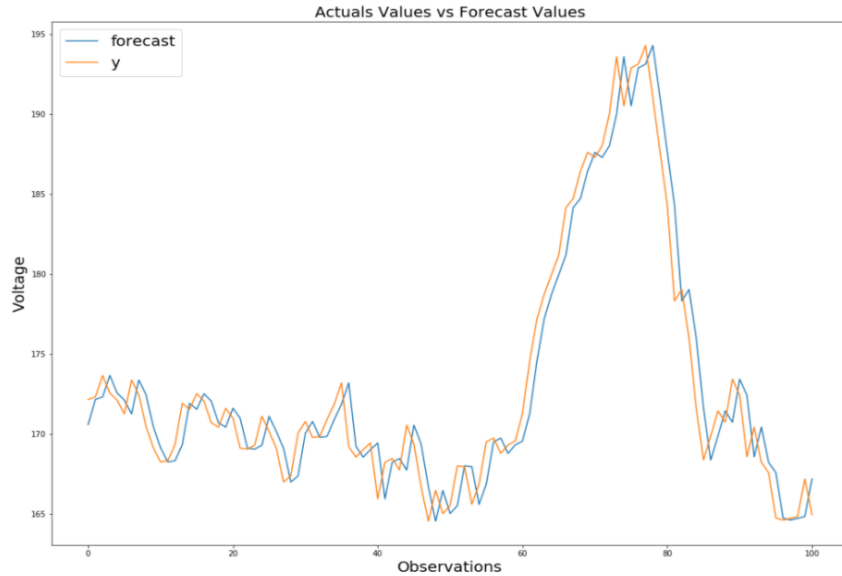


FIGURE 2.8: ARIMA model forecasting

A linear regression model can be constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the model. A value of 0 can be used for a parameter, which denotes not to use that element of the model.

The accuracy of ARIMA model is dependent on the sample size of the time series data. ARIMA model relatively works better on smaller or short term sample size and also it is comparatively easy to implement than most other machine learning algorithm.

2.1.4 Logistic Regression

Logistic regression is a classification algorithm which is used to predict a binary outcome based on a set of independent variables. A binary outcome refers to two possible results- either an event occurs (1) or it does not occur (0). On the other hand, independent variables are those factors which influence an outcome. It is the machine learning technique for the function used at the core of the method, the logistic function. Logistic regression can also be used to deal with the issues of classification. Generally, a logistic regression classifier can work with a linear combination consisting more than one feature value or explanatory variable as argument of the sigmoid function.

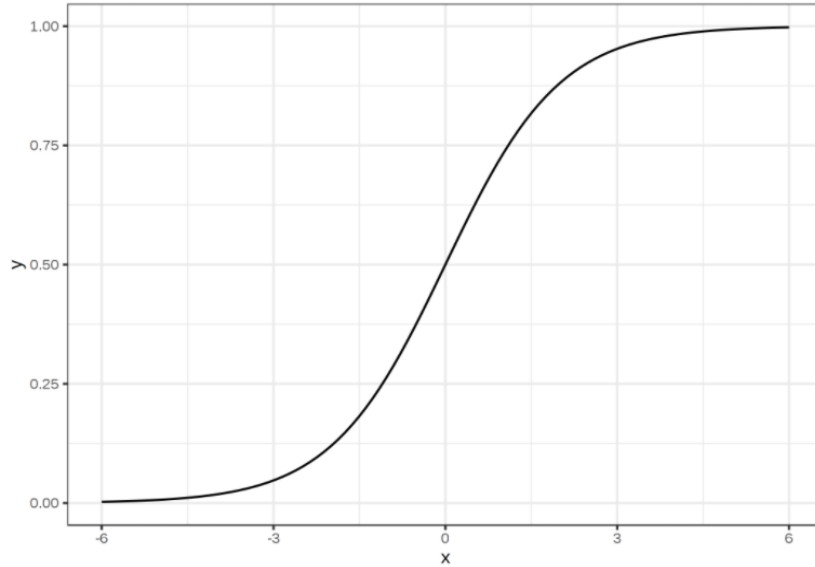


FIGURE 2.9: Logistic Function

The logistic function (Figure 2.9), also referred to as the sigmoid function was developed by statisticians to explain the fundamentals of increase in ecology. It's an S-shaped curve that will take any real-valued number and map it into a worth between 0 and 1, but never exactly on the same verge of those limits. The logistic function equation is given as:

$$S(x) = \frac{1}{1 + e^{-x}} \quad (2.10)$$

In this equation [2.10], e is the base of the natural logarithms and x is the actual numerical value that we want to transform. The equivalent output of that sigmoid function is a value between 0 and 1 whereas the middle value is considered to be a threshold value to establish what belongs to class 1 and to class 0. Particularly, an input generating a value greater than 0.5 is considered to belong to class 1 and if the output is less than 0.5, then the corresponding input is classified as belonging to the 0 class. On the other hand, if the input value is 0, it will give output as 0.5.

Types of Logistic Regression

Based on the possibilities of outcomes, logistic regression can be categorized into 3 classifications.

Binary Logistic Regression:

it is used to forecast the relationship between an independent variable (X) and a dependent variable (Y) where the dependent variable is by nature a binary number. For instance, the output can be 0/1, true/false, yes/no or success/failure.

Multinomial Logistic Regression:

it is often used when there is a categorized dependent variable with at least two or more discrete possibilities. It is quite similar to binary logistic regression except the fact that multinomial logistic regression has more than two possibilities. Example

of such variable can be transportation system, where transportation type is referred as the dependent variable whereas train, bus, car and bike are possible outcomes.

Ordinal Logistic Regression:

it is used when the dependent variable (Y) has an ordinal density and also has more than two suborders. For example, movie rating (1 star to 5 star), score on a test (Excellent/Good/Average/Poor/Very Poor) or an opinion poll (I agree/ I disagree/Neutral).

Pros and Cons of Logistic Regression

Pros:

1. Logistic regression is easier to train and implement as compared to other models.
2. Model coefficients can be interpreted as indicators of important features.

Cons:

1. Assumption of linearity between dependent variable and independent variable.

2.1.5 Support-Vector Machine:

Support-Vector machines are supervised learning models with an associated learning algorithm that analyze data used for classification and regression analysis. Mostly it is used for classification problems. This is a very popular classification algorithm. The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labeled as one class and all the points that fall on the other side will be labeled as the second. The approach is also known as the wide street approach, introduced for the first time by Vladimir Vapnik in the early 1990's. SVM uses 'support vectors' to draw a line between two classes. It refers to two position vectors drawn from the origin to the points which dictate the decision boundary. Hence, the name Support-Vector Machine.

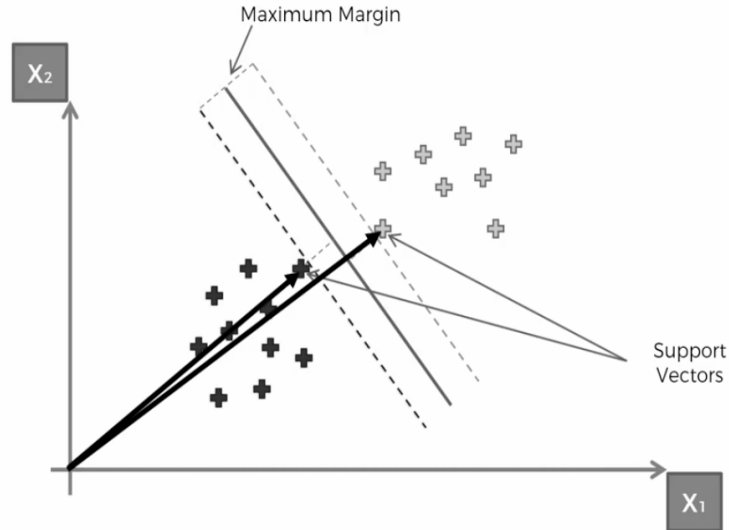


FIGURE 2.10: SVM Algorithm

There are 4 ways to or methods of drawing this line: line with High or Low Gamma and line with High or Low Regularization. The High Gamma line considers the nearest possible points as support vectors whereas the Low Gamma line considers the opposite. Similarly, a line with High regularization is drawn carefully to avoid any misclassification. On the contrary, a line with Low regularization is drawn to look smoother, accepting some errors. Lines with high regularization have better accuracy than most others but tend to overfit the model most of the time and lines with low regularization do not give the accurate result but are more flexible towards outliers. The SVM linear classifier is based on the linear discriminant function-

$$f(x) = w^T x + b \quad (2.11)$$

In this equation [2.11], w is the weight vector and b is called the bias. SVM uses kernel functions to effectively separate the data with high correlation factors. Some of these kernel functions are simple, others are very complex. Depending on the data, different kernel functions can be used to compute the relationship of the observation in a higher dimension. We can use the high dimensional relationships to find a Support Vector Classifier. This process makes it easier to identify the margin between two closely related features. Some widely used kernels are Polynomial kernel, Gaussian kernel, Hyperbolic tangent kernel, Sigmoid kernel etc.

Support vector machine is very efficient with high dimensional data. Also, in cases where number of features are greater than the sample size, SVM performs well. This algorithm can be used for both classification and regression analysis. Small changes in data does not affect the hyperplane as SVM model is very stable. SVM can handle non-linear data using Kernel trick. Performance is great for well separated data, however when data points are not well separated, which means overlapping classes are there, SVM does not perform well. Moreover, training time and hardware requirement to train this model is pretty high. Also, choosing the optimal kernel is a difficult task and lot of planning and data preprocessing needs to be done before implementing the model. When training SVM, we need to make a number of de-

cisions: which kernel to choose, how to process the data, what will be the value of soft margin constant (C) or gamma will be. Uneducated guess can be detrimental to the performance of the model and will yield inaccurate result.

SVM is a handy tool as a binary classification algorithm. It will be able to classify the faulty data and the healthy data into clusters and based on that predict the time of failure of any industry machine and calculate the optimum maintenance time interval of those machines to efficiently use the remaining useful life cycle of those machines to reduce the cost of maintenance and prevent avoidable mishaps on the workplace due to machine failure.

2.1.6 Recurrent Neural Network:

A recurrent neural network is the one kind of artificial neural network which is used for basically time series or sequential data implementation. Language translation, natural language processing (NLP), speech recognition, and image captioning- these types of temporal problems have been used by deep learning algorithms and they are joined into well-known applications such as Siri, voice look, and Google Interpret. RNN (Figure 2.11) is basically utilized for energetic data handling like time arrangement expectation, handling control, and so on. This neural network is the moment kind of ANN show, in which the outputs from neurons are utilized as criticism to the neurons of the past layer.

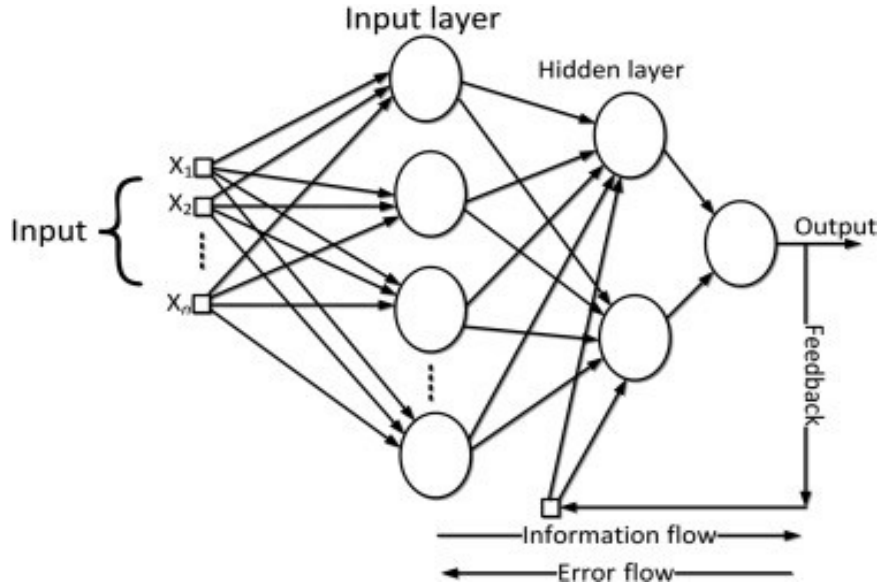


FIGURE 2.11: RNN Architecture

Recurrent neural networks allow previous outputs to be used as inputs while having hidden states. If we use time step t and activation $a^{<t>}$, the output $y^{<t>}$ are denoted as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (2.12)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \quad (2.13)$$

The steps 2.12 and 2.13 shows the expression of traditional RNN Where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 are activation functions.

Types of RNN:

RNN models are generally utilized within the areas of normal dialect preparing and discourse acknowledgment. The different applications (Fig:2.12) are as follows:

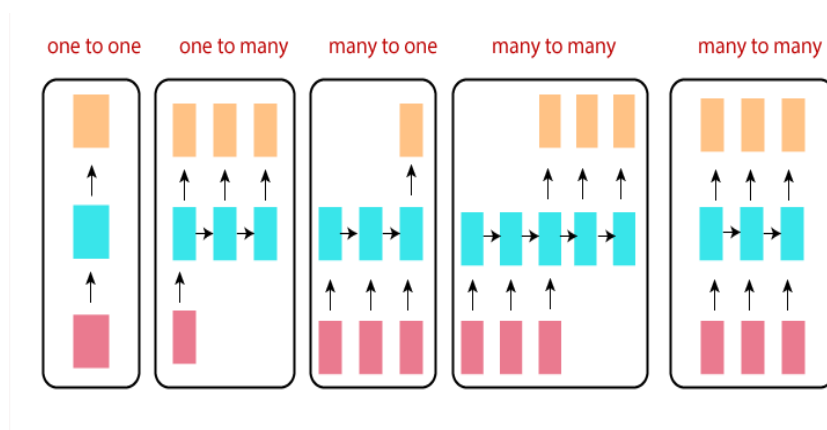


FIGURE 2.12: Types of RNN

One to one:

It is referred as Traditional Neural Network. It deals with a fixed size of the input to the fixed size of output, where they are independent of previous information/output.

One to many:

It deals with an exact size of information as input that gives a sequence of data as output. For example, image captioning.

Many-to-one:

It is more like Sentiment classification. It takes a grouping of data as input and yields a settled estimate of the output.

Many-to-many:

It is referred as a Machine translation. It takes a Arrangement of data as input and forms the repetitively yields as a Arrangement of information.

LSTM Structure:

Recurrent Neural network endure from short-term memory. On the off chance that a grouping is long sufficient, they'll have a difficult time carrying data from prior time steps to afterward ones. So, on the off chance that you're attempting to handle a section of content to do forecasts, RNN's may take off out critical data from the starting. Long Short-Term Memory (LSTM) systems are a sort of recurrent Neural network competent of learning arrange reliance in arrangement forecast problems.

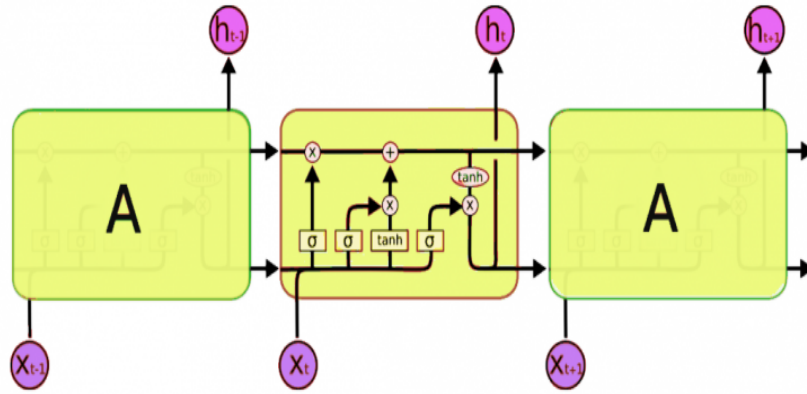


FIGURE 2.13: Architecture of LSTM

A commonplace LSTM arrange (Fig:2.13) is comprised of diverse memory squares called cells. There are two states that are being exchanged to another cell; the cell state and the hidden state. The memory pieces are capable for recalling things and controls to this memory is done through three major components, called doors.

Chapter 3

Methodology and Result

3.1 Workflow

The data that we have used is collected from the ALM Workshop and we will acquire data from other sources if there is a need for that. After acquiring the dataset we performed basic analysis on the entries like how many entries we have and what are the features. The details of the dataset will be discussed later on this paper.

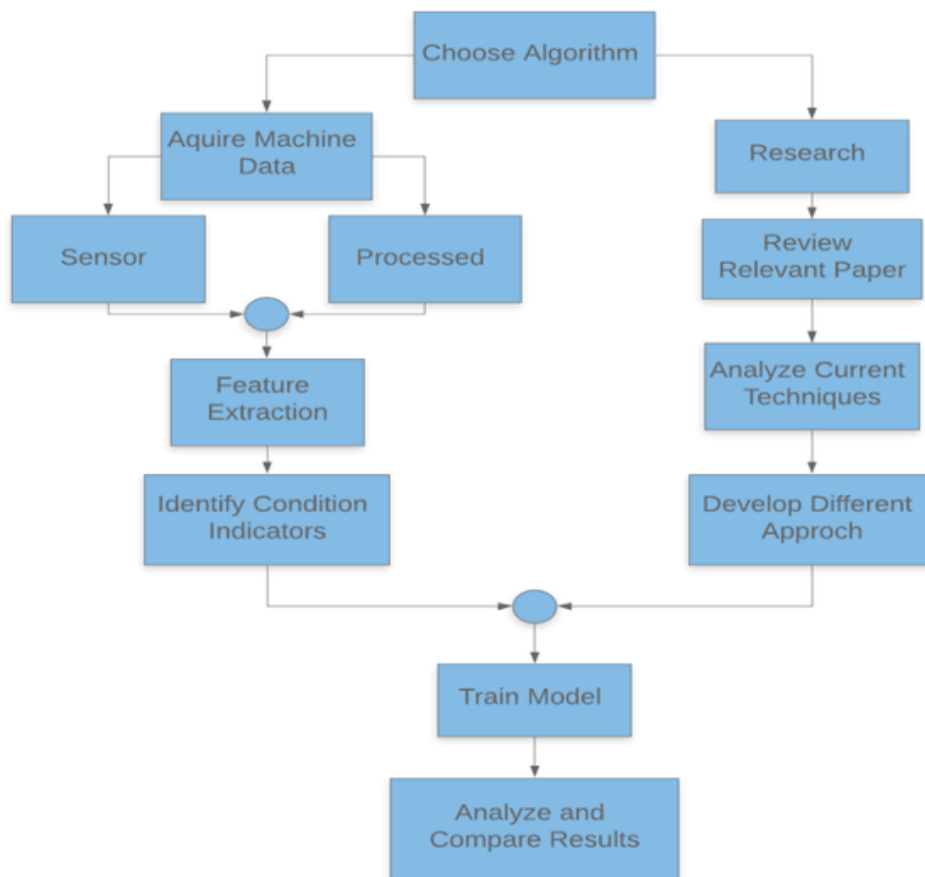


FIGURE 3.1: Workflow Diagram

After acquiring the data, we did our research on the related works in this field by other renowned researchers and found that the techniques that others used while confronting this task. Based on our literature review, we have found that different algorithms perform differently on the same data set as the data sets are machine specific. We have also noticed that some algorithms like Random Forest, SVM and Logistic Regression perform better at predicting failure times of the machines based on previous machine data. Accuracy is highly dependent on the pre-processing methods used to process the data set which is then used for training the machine learning model. We removed some of the features to reduce the complexity and then identified condition indicators. This was tricky as our data set did not define the error by naming them properly. The errors were categorized into five groups (Error1, Error2, Error3, Error4, and Error5). We did not know the type or nature of this error which would have helped us a lot for identifying the key condition indicator. Figure 3.1 shows our workflow diagram which we followed throughout our whole process.

3.2 Dataset Description

The data that we have used for this research so far is the machine log of hydro power turbines which was recorded in the energy industry during the year 2015. The data source is the telemetry time-series data which consists of voltage, rotation, pressure and vibration measurements collected from 100 machines in real time averaged every hour. Time series data are the quantities that represent or trace the values taken by a variable over a period such as a month, quarter, or year. Time series data occurs wherever the same measurements are recorded on a regular basis. The data is collected at different points of time. Figure 3.2 shows a small sample of our raw data set. Telemetry is the automatic measurement and wireless transmission of data from remote sources. It is the highly automated communications process by which measurements are made and other data collected at remote or inaccessible points and are transmitted to receiving equipment for monitoring.

datetime	machineID	volt	rotate	pressure	vibration
2015-01-01 06:00:00	1	176.2179	418.5041	113.07794	45.08769
2015-01-01 07:00:00	1	162.8792	402.7475	95.46053	43.41397
2015-01-01 08:00:00	1	170.9899	527.3498	75.23790	34.17885
2015-01-01 09:00:00	1	162.4628	346.1493	109.24856	41.12214
2015-01-01 10:00:00	1	157.6100	435.3769	111.88665	25.99051
2015-01-01 11:00:00	1	172.5048	430.3234	95.92704	35.65502
2015-01-01 12:00:00	1	156.5560	499.0716	111.75568	42.75392
2015-01-01 13:00:00	1	172.5228	409.6247	101.00108	35.48201
2015-01-01 14:00:00	1	175.3245	398.6488	110.62436	45.48229
2015-01-01 15:00:00	1	169.2184	460.8507	104.84823	39.90174

FIGURE 3.2: Sample Data

There are 291300 data entries of 100 machines. Each of the machines has about 3000 data entries for themselves. There are 4 types of machines used (Model 1-4). Total errors recorded are 3920 times among the 292300 entries. Errors are categorized into 5 different groups named Error1, Error2 and so on. The unlabeled features contain voltage, rotation, pressure and vibration.

machineID	model	age
1	1 model3	18
2	2 model4	7
3	3 model3	8
4	4 model3	7
5	5 model3	2
96	96 model2	10
97	97 model2	14
98	98 model2	20
99	99 model1	14
100	100 model4	5

FIGURE 3.3: Machine Description

This dataset also includes information about the machines. It contains machine ID, model type and the age which denotes the years in service of that particular machine Shown in Figure 3.3. There are a total of 100 machines used to gather the data. Age column indicates the number of months that the machine was being used.

error1count	error2count	error3count	error4count	error5count	model	age	failure
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
0	0	0	0	0	model3	18	FALSE
1	0	0	0	0	model3	18	FALSE

FIGURE 3.4: Error Count for MachineID 1

We also get the error count from the dataset which is recorded separately from unlabeled features. One sample of the error record is shown in Figure 3.4 of MachineID 1. Getting an error does not mean that the machine has failed. As the number of errors increase, the probability of the failure of the machine increases along with it.

After acquiring the data, we start to look for the best possible ways to train the model using the unlabeled features at first. First, we plot two of each feature for a particular machine. Figure 3.5 shows the scatter plot of Machine 1 and Machine 2 using voltmean and rotatemean on the X and Y axis. We can clearly see that only using unlabeled features cannot at all form clusters to distinguish the machines. We will need to perform feature engineering to process the data and convert the data in another intermediate state so that the features can be used to truly predict the failure time of the machines. In the next sections, we will discuss different approaches to prepare the dataset and train models using the prepared dataset.

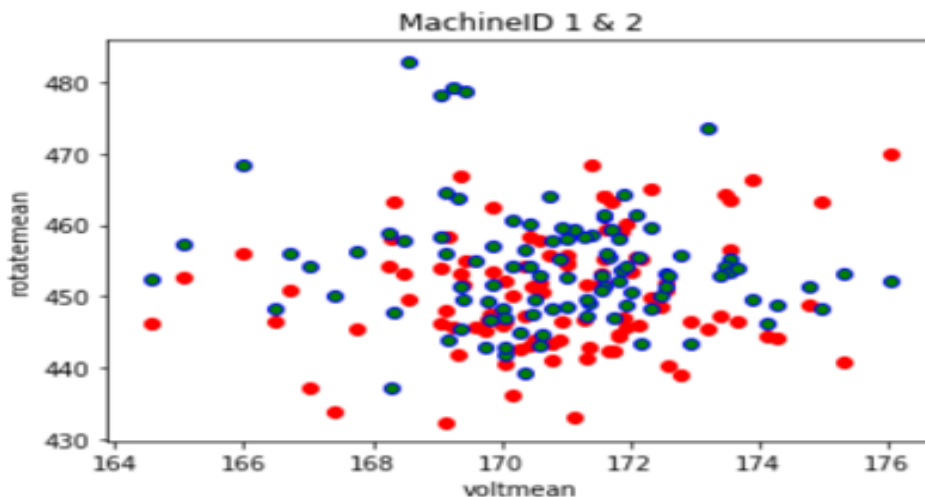


FIGURE 3.5: Scatter Plot of Machine 1 and Machine 2

3.3 Data Preparation

For keeping it simple, first we separated the entire dataset into machine wise data i.e. machine1 data, machine2 data etc. Because training the entire dataset with 291300 samples all at once will be very complex and will strain our limited hardware capacity. Also we have dropped the 'errorcount' columns as these columns do not add significant information to the training model. We are using the mean and standard deviation data of each feature column for training the model instead of the raw unlabeled data as it is more likely to provide meaningful information about patterns and trends in the dataset.

	datetime	machineID	voltmean	rotatemean	pressuremean	vibrationmean	voltsd	rotatesd	pressuredsd	vibrationsd	model	failure
0	2015-01-02T05:00:00Z	3.0	170.066825	460.956803	101.395264	37.989643	12.133703	50.054464	10.831107	5.845904	model3	False
1	2015-01-02T08:00:00Z	3.0	167.193146	457.407125	100.415560	38.557481	13.310902	50.098975	10.930315	5.433685	model3	False
2	2015-01-02T11:00:00Z	3.0	169.120617	460.992958	99.214805	38.863332	12.218718	47.527612	11.005281	4.967186	model3	False
3	2015-01-02T14:00:00Z	3.0	170.667376	464.491193	99.134247	39.658603	12.089424	44.312169	10.326532	4.606327	model3	False
4	2015-01-02T17:00:00Z	3.0	169.182447	472.409812	101.396497	40.188093	12.652348	46.191892	10.433913	3.983933	model3	False
5	2015-01-02T20:00:00Z	3.0	168.116799	473.222315	103.196659	40.326600	13.144143	46.590069	9.739985	3.841949	model3	False
6	2015-01-02T23:00:00Z	3.0	165.658495	479.284092	103.152755	40.004943	14.520519	40.932535	9.469833	4.272169	model3	False
7	2015-01-03T02:00:00Z	3.0	164.803837	478.622524	103.408181	39.396092	12.315427	42.427274	10.771449	3.883036	model3	False
8	2015-01-03T05:00:00Z	3.0	165.558850	471.408804	103.549486	39.127594	13.401385	45.472875	9.285569	3.809484	model3	False
9	2015-01-03T08:00:00Z	3.0	169.492729	468.776360	104.119234	39.284716	15.909475	46.988548	9.050613	3.767897	model3	False
10	2015-01-03T11:00:00Z	3.0	169.353956	461.648251	104.997225	38.914257	15.932795	55.766193	8.806063	3.646701	model3	False

FIGURE 3.6: Machine 3 data

First step of preparing the time series dataset is to determine whether it is stationary or not. That means we have to look for increase and decrease in the mean and variance of the values. The statistical properties of the series do not change over time if it is stationary. It does not mean that the series does not change over time, just that the way it changes does not itself change over time. If the mean or variance changes over time, which indicated that the time series is not stationary. Stationary datasets are easier to analyze and predict or forecast. Figure 3.6 shows a small sample data of Machine 3. Simplicity of the dataset can be very useful towards building an efficient model. Stationarity is also important because almost all the prominent approaches for classifying and forecasting time series data use stationary data. Stationarity has become a common assumption for many practices and tools in time series analysis. These include trend estimation, forecasting and causal inference, among others.

We can see the visualization of the data from Machine 5 on Figure 3.7. It is hard to determine whether the series is stationary or not just by looking at the plotted graph. That is why we need to check for stationarity by testing the dataset. A quickest way to check to see if time series is non-stationary is to review the summary statistics by splitting time series into two (or more) partitions and compare the mean and variance of each group. If they differ and the difference is statistically significant, the time series is likely to be non-stationary. But this approach is a slack approach to check for stationarity and not always reliable. The two most prominent approach are the Augmented Dickey-Fuller Test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test. These tests are used for testing a null hypothesis that an observable time series is stationary around a deterministic trend (i.e. trend-stationary) against the alternative of a unit root. We have opted to use the ADF testing of our dataset.

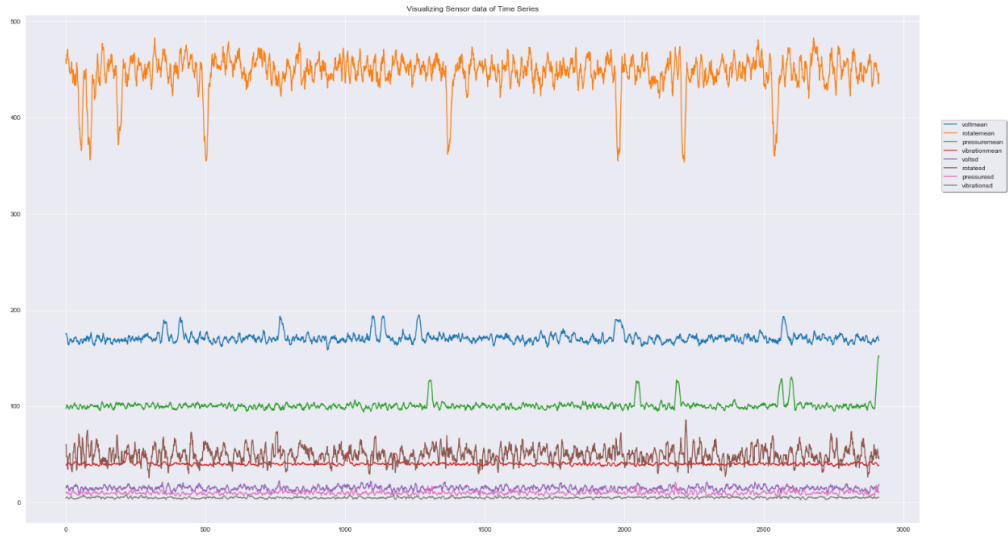


FIGURE 3.7: Visualization of the Machine 5 data

Augmented Dickey-Fuller test: It is a type of unit root test which is good for large, complex datasets. Unit root is a characteristic of a time series that makes it non-stationary. The presence of a unit root means the time series is non-stationary. Moreover, the number of unit roots contained in the series corresponds to the number of differencing operations required to make the series stationary.

$$Y_t = \alpha Y_{t-1} + \beta X_e + \epsilon \quad (3.1)$$

where, Y_t is the value of the time series at time 't' and X_e is a separate explanatory variable, which is also a time series.

ADF test calculates the p value which determines whether we accept or reject the **Null hypothesis**. The hypothesis which states that there is no difference is called null hypothesis. Without the null hypothesis, we would need some preliminary data in order to make a statement that we can test in the follow up experiments. In time series analysis, null hypothesis assumes the presence of a unit root, which we will accept or reject based on the p value[22].

For checking stationarity of time series -

- If $p > 0.05$: We accept the Null Hypothesis (H0) and determine the data has a unit root and it is non-stationary.
- If $p \leq 0.05$: We reject the Null Hypothesis (H0) and determine the data has no unit root and it is stationary. The more negative the value is, the stronger the rejection of the hypothesis that there is a unit root.

We have test the stationarity using ‘adfuller’ method which is a library function stated under Statsmodel in Python (Version 3.7). We conduct the method on each of the columns of the dataset and here are the results.

TABLE 3.1:

Feature	Voltmean
Test Statistic	-6.473288e+001
P-value	1.350427e-08
Critical Value(1%)	-3.433820e+00
Critical Value(5%)	-2.863073e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.2:

Feature	Rotatemean
Test Statistic	-5.275401
P-value	0.000006
Critical Value(1%)	-3.433818
Critical Value(5%)	-2.863072
Critical Value(10%)	-2.567586
Verdict	Stationary

TABLE 3.3:

Feature	Pressuremean
Test Statistic	-6.923090e+00
P-value	1.133246e-09
Critical Value(1%)	-3.433822e+00
Critical Value(5%)	-2.863074e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.4:

Feature	Vibrationmean
Test Statistic	-6.727698e+00
P-value	3.356931e-09
Critical Value(1%)	-3.433822e+00
Critical Value(5%)	-2.863074e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.5:

Feature	Voltsd
Test Statistic	-7.504866e+00
P-value	4.160300e-11
Critical Value(1%)	-3.433818e+00
Critical Value(5%)	-2.863072e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.6:

Feature	Rotatesd
Test Statistic	-8.582578e+00
P-value	7.657584e-14
Critical Value(1%)	-3.433822e+00
Critical Value(5%)	-2.863074e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.7:

Feature	Pressuresd
Test Statistic	-6.389406e+00
P-value	2.124239e-08
Critical Value(1%)	-3.433818e+00
Critical Value(5%)	-2.863072e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

TABLE 3.8:

Feature	Vibrationsd
Test Statistic	-7.881795e+00
P-value	4.681507e-12
Critical Value(1%)	-3.433820e+00
Critical Value(5%)	-2.863073e+00
Critical Value(10%)	-2.567586e+00
Verdict	Stationary

When the p-value is below 0.05 and the Test Statistic value is lower than all the critical values, we can reject the Null Hypothesis and determine with confidence that there is no unit root on the series and the series is stationary. If the p-value is greater than 0.05 and the Test statistic value is greater than the critical values, we accept the Null Hypothesis and determine that there is a presence of unit root in the series, therefore the series is not stationary. Table 3.1 to table 3.8 shows the ADF test results for all of the features. As the test results show that the data is already stationary, for the moment we will not need to apply any transformation (Differencing, Logarithmic Transformation etc.). Further feature engineering will be done if needed based on the requirements of specific algorithms.

ACF and PACF:

Autocorrelation represents the degree of similarity between a given time series and a lagged version of itself over successive time intervals. We can measure this using the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF).

ACF is the auto-correlation function which gives us values of auto-correlation of any series with its lagged values. The plot is sometimes called a correlogram or an autocorrelation plot. Confidence intervals are drawn as a cone. By default, this is set to a 95% confidence interval, suggesting that correlation values outside of this cone are very likely a correlation and not a statistical fluke.

Figure 3.8 shows the autocorrelation plot of the column Voltmean of our data set. This suggests that there is high correlation between data up to lag 12, which means we will need to consider data up to previous 12 values for prediction. Here the last significant lag is the 12th one.

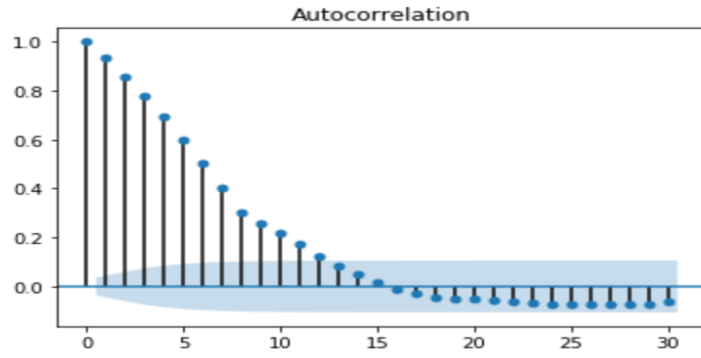


FIGURE 3.8: ACF Plot for voltmean

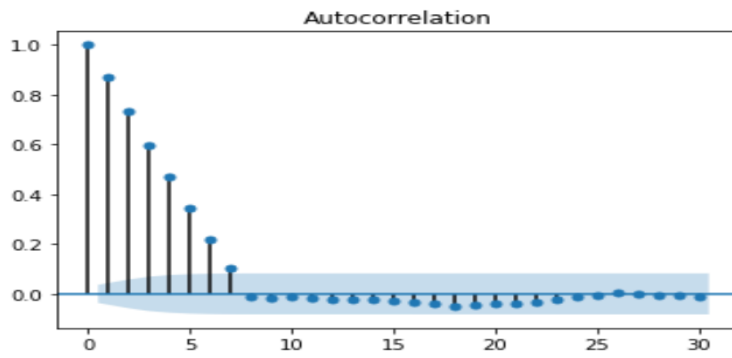


FIGURE 3.9: ACF Plot for voltsd

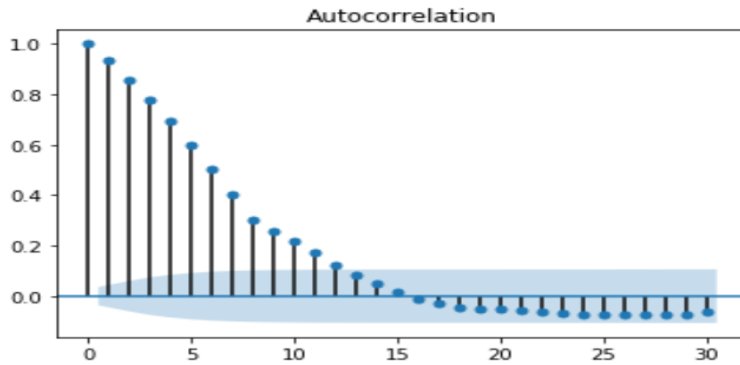


FIGURE 3.10: ACF Plot for rotatemean

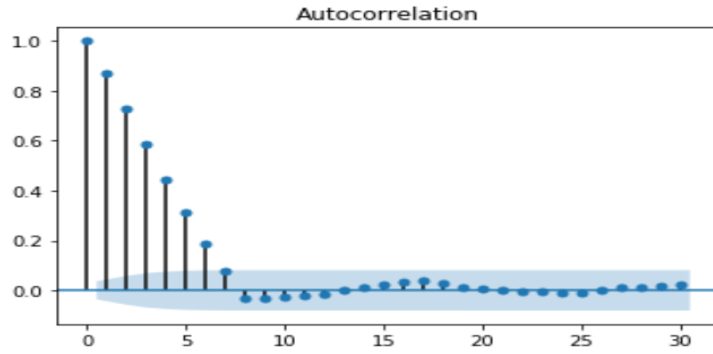


FIGURE 3.11: ACF Plot for rotatedsd

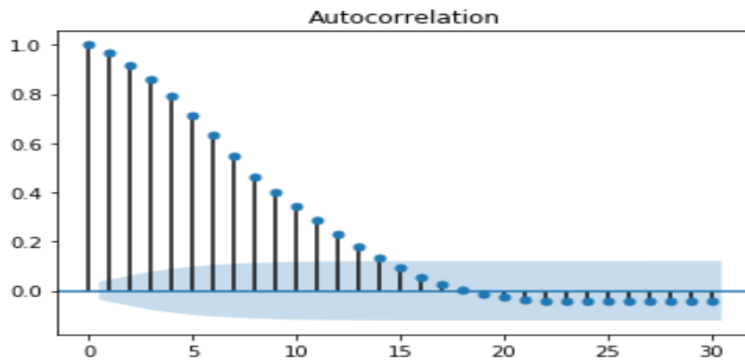


FIGURE 3.12: ACF Plot for pressuremean

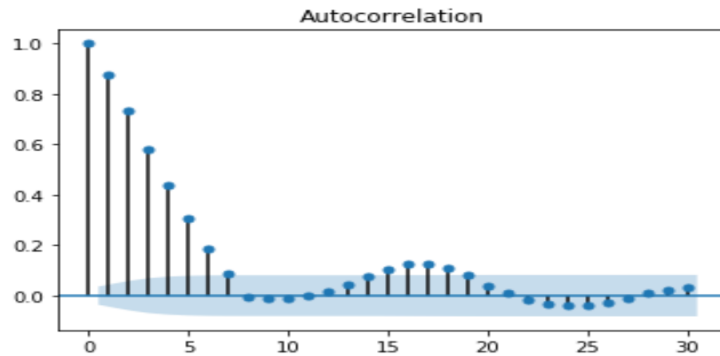


FIGURE 3.13: ACF Plot for pressuresd

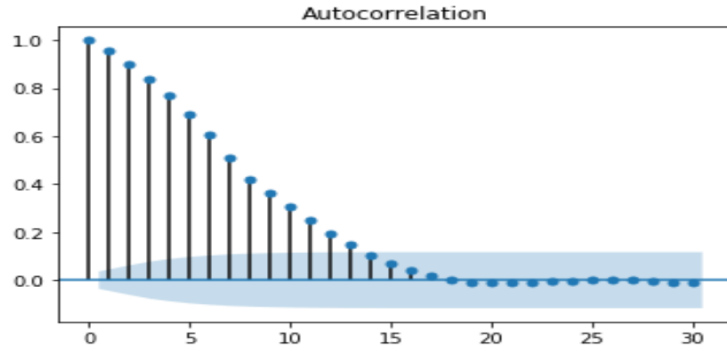


FIGURE 3.14: ACF Plot for vibrationmean

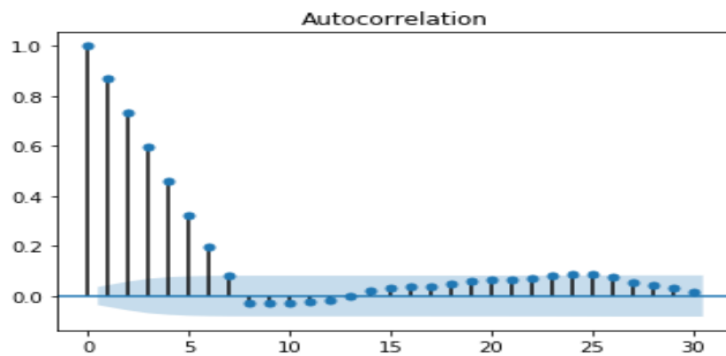


FIGURE 3.15: ACF Plot for vibrationsd

PACF only describes the direct relationship between an observation and its lag, which means the relationships of intervening observations is removed. The autocorrelation for an observation and an observation at a prior time step is comprised of both the direct correlation and indirect correlations. These indirect relations are removed on the PACF plot. Figure 3.16 shows the partial autocorrelation of the column ‘voltmean’. Here, we are seeing that the current value has direct relation with lag up to 25, but some intermediate lags are weakly correlated. In this case, the immediate previous value has the highest correlation among the past values. Here, the last significant lag is the also the 12th one as seen before on the ACF plot of voltmean column.

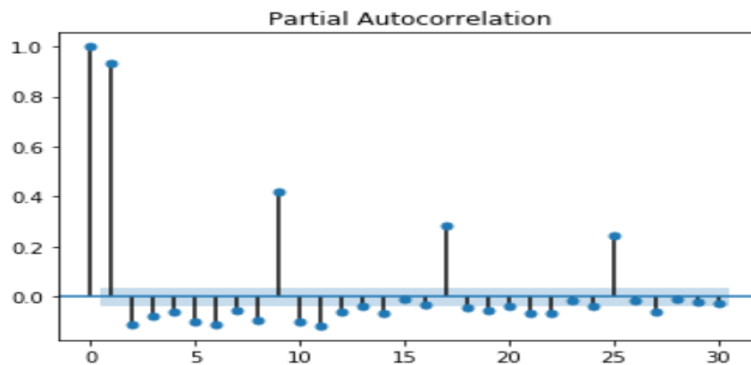


FIGURE 3.16: PACF Plot for voltmean

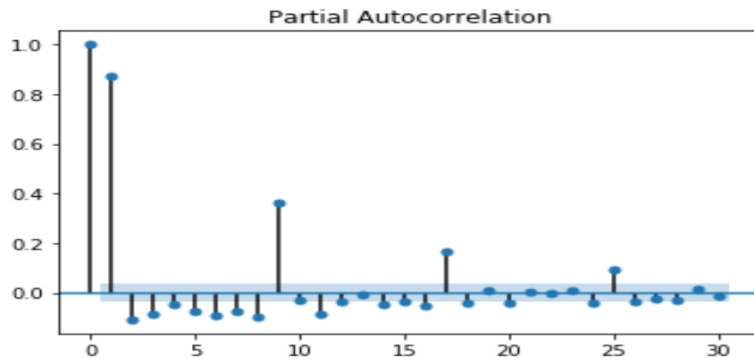


FIGURE 3.17: PACF Plot for voltsd

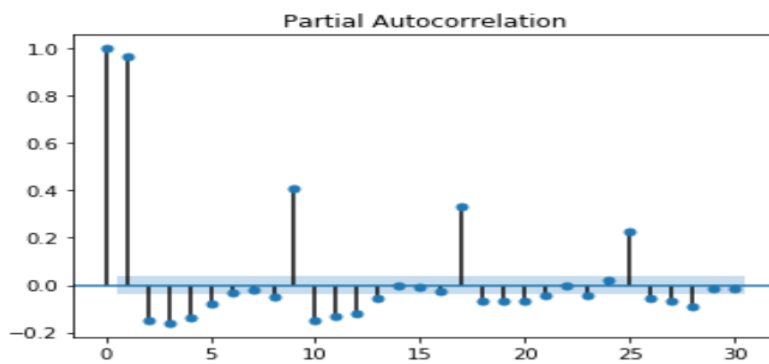


FIGURE 3.18: PACF Plot for rotatemean

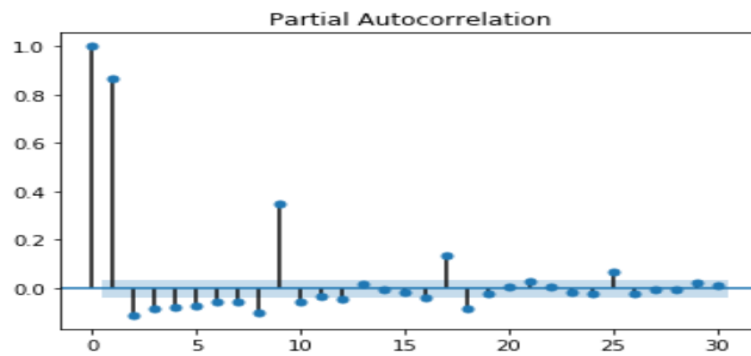


FIGURE 3.19: PACF Plot for rotatedsd

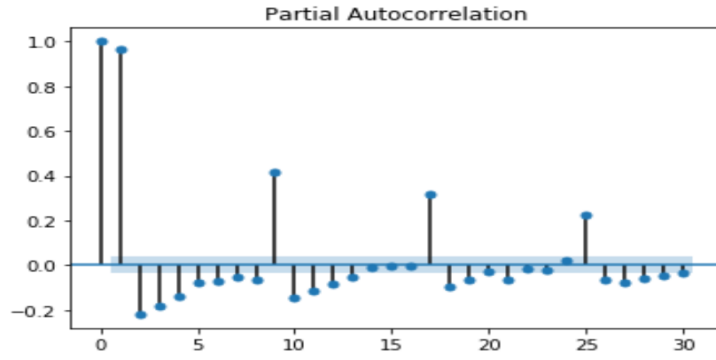


FIGURE 3.20: PACF Plot for pressuremean

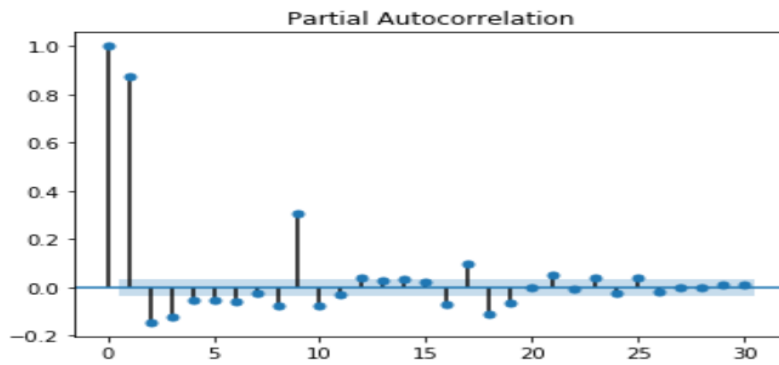


FIGURE 3.21: PACF Plot for pressuresd

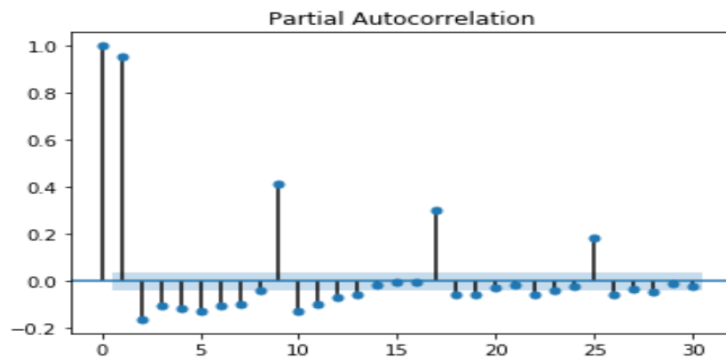


FIGURE 3.22: PACF Plot vibrationmean

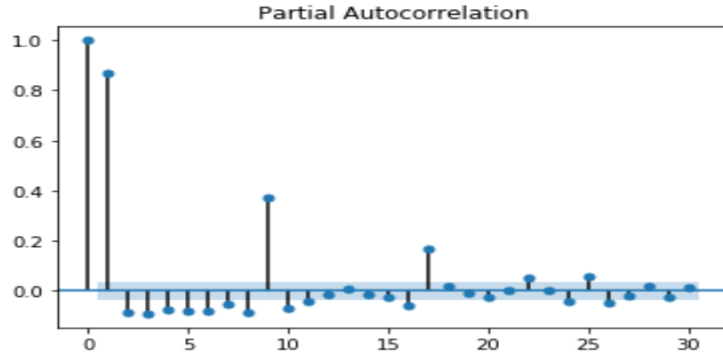


FIGURE 3.23: PACF Plot for vibrationsd

Both ACF and PACF yield values within the interval $[-1, 1]$. Positive value indicates that the previous value and the current value move in the same direction, that means if one increase the other one increased too. Negative value shows that if one increase the other decreases and vice-versa.

3.4 Algorithm Workflow

3.4.1 SVM

We use our dataset to train a SVM classification model to see whether the machines will show true value on failure or not. At first we run the model using the unedited raw data using 8 independent variable and one dependent variable ('Failure'). We used the linear kernel at first and later checked the result using polynomial kernel as well.

We used 80% of the available data for training and the rest of the data was reserved for testing. For, raw data using the linear kernel we get an accuracy score of 98.28%. But the classification report on Figure 3.24 shows that the f1-score for true values of the failure column is 29% (Kernel='linear') which indicates the model is not good at classifying true values at all. The accuracy is biased towards false values.

	precision	recall	f1-score	support
False	0.98	1.00	0.99	571
True	1.00	0.17	0.29	12
accuracy			0.98	583
macro avg	0.99	0.58	0.64	583
weighted avg	0.98	0.98	0.98	583

FIGURE 3.24: Classification report for SVM model

The confusion matrix on Figure 3.25 visualizes the results. Almost all false values are classified correctly, which is simply because the dataset is imbalanced towards the false values. Number of true values are significantly less than the false values. Using polynomial kernel outputs similar results in this case.

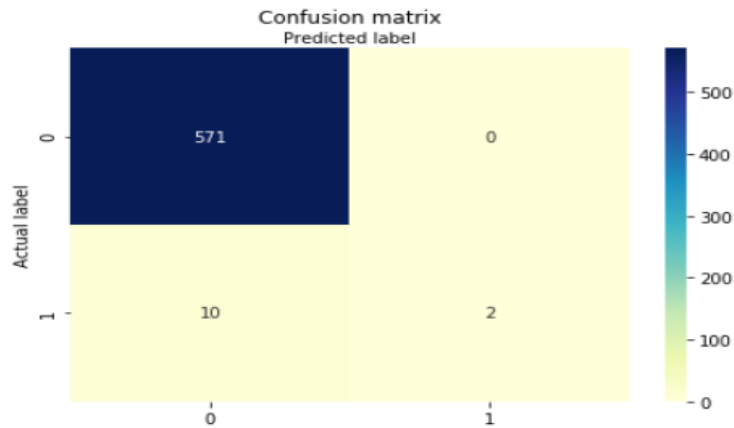


FIGURE 3.25: Confusion Matrix (Kernel = ‘Linear’)

Next we use the difference values of the column for train and testing. We calculated the difference of the present and previous values and store the result in place of present value, then remove the rows with null values. After training we get a similar result as before with an accuracy of 94.07% (Kernel='polynomial', C = 10000, Gamma = 0.1). But the precision for true values only is way below its acceptable margin on Figure 3.26. The false value prediction is higher because of the higher count to false values in the sample dataset (Kernel = ‘poly’, C=10000, gamma=0.1).

	precision	recall	f1-score	support
False	0.99	0.95	0.97	569
True	0.03	0.20	0.06	5
accuracy			0.94	574
macro avg	0.51	0.57	0.51	574
weighted avg	0.98	0.94	0.96	574

FIGURE 3.26: Classification report for SVM model

Another way we tried to overcome this is by taking the weighted moving average of lagged values of the past 24 hours. We determined the weight by taking into account the ACF and PACF values which we have calculated earlier. Then we calculated the weighted moving average for each row of the column and used the new weighted average values as dataset for training and testing. After removing the outliers from the dataset we split the data and then train the model. The accuracy for true value is even worse than the previous approaches (Shown on Figure 3.27). Only one true value was in the test set and the model was unable to classify it correctly.

	precision	recall	f1-score	support
False	1.00	1.00	1.00	581
True	0.00	0.00	0.00	1
accuracy			1.00	582
macro avg	0.50	0.50	0.50	582
weighted avg	1.00	1.00	1.00	582

FIGURE 3.27: Classification report (Kernel = ‘Linear’)

SVM model was not successful in classifying the data, mainly for two reasons. Firstly, the data set is highly imbalanced. The number of true values are significantly smaller than the false value, therefore leaving the model with no proper path to classify them correctly. Secondly, the true values are not distributed evenly throughout training data set. The sample on the third approach that we used had only 24 true values and most of them were at the later end of the time series data. This also might cause the model to perform poorly for classifying true values.

3.4.2 ARIMA

We also tried using the ARIMA model, the most popular algorithm for time series analysis and forecasting. But ARIMA model alone cannot solve classification problem. This model is mainly used for forecasting only. This model cannot be used by itself to classify or cluster. However, integrating it with other models can improve classification performance of the entire process.

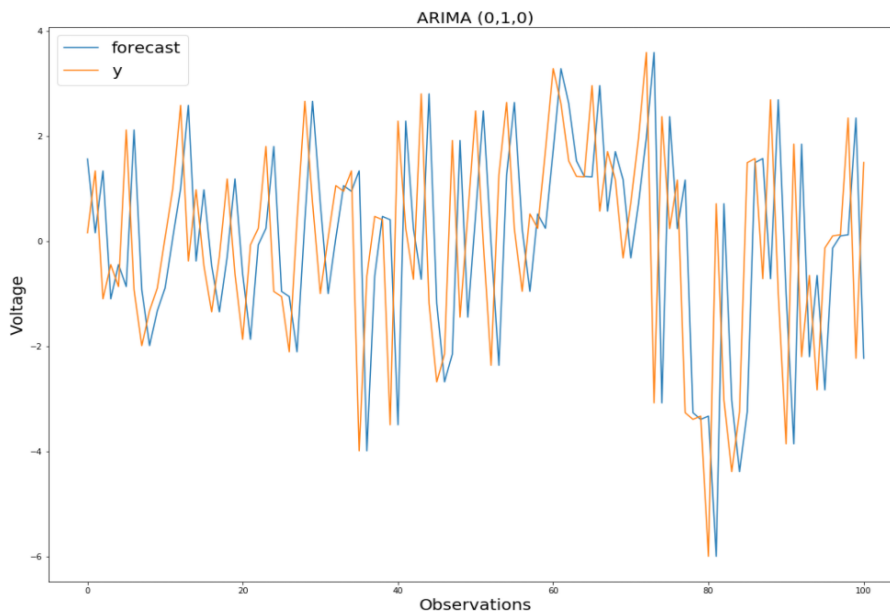


FIGURE 3.28: Forecast using ARIMA(0,1,0)

As our data has no seasonal attributes, we used ARIMA (0, 1, 0) the random walk model and ARIMA (1, 1, 2) the damped-trend linear exponential smoothing for

forecasting particular feature values. Figure 3.28 shows the actual values versus the forecast values which performed significantly better than ARIMA (1, 1, 2) which is shown in Figure 3.29.

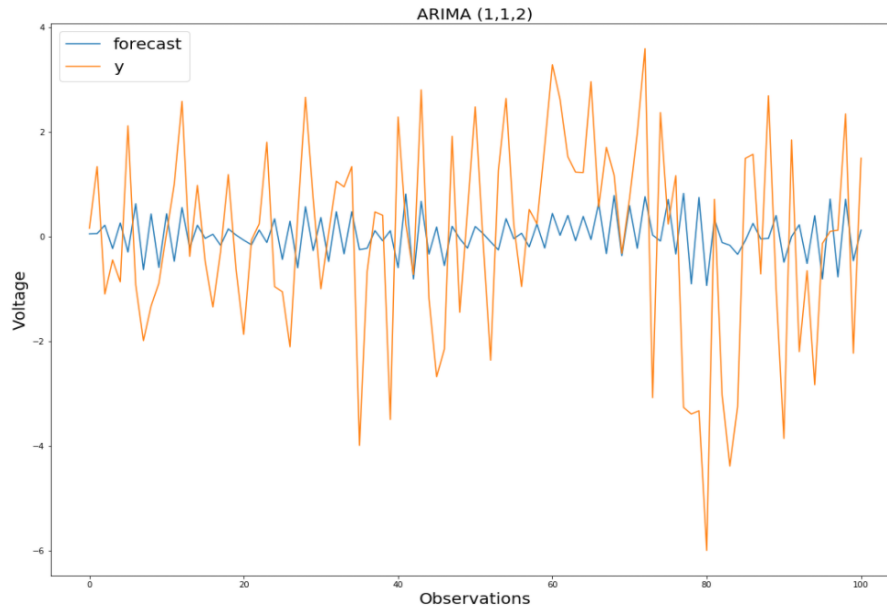


FIGURE 3.29: Forecast using ARIMA(1,1,2)

We can use forecasting methods to predict future data and then test that data on a classification model to predict the failure rate of any machine for predictive maintenance. Forecasting future values based on the future will increase the sample size of the dataset and will enable scope for using better clustering methods. We also tried implementing the Vector Autoregressive (VAR) method but the performance was poor for this particular dataset.

3.4.3 Logistic Regression

To train a logistic regression classification model, we initially divided the features into two variables- target variable and feature variable. There are 8 target variables and one is feature variable. In order to understand performance of the model, we split the available raw dataset into training set and a test set. The dataset is broken into a ratio of 75:25 which means 75% data is used for training set and 25% data is used for model testing. After training raw data using logistic regression function, we get an accuracy score of 97.80% which is more than what we have expected. However, against the accuracy, the precision and recall score both turned out to be 0 as the true positive values are 0.

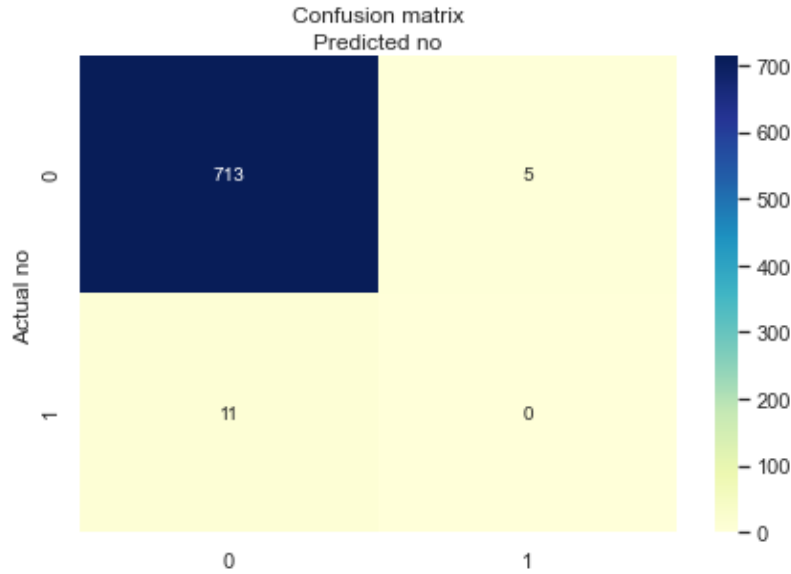


FIGURE 3.30: Confusion Matrix of Logistic Regression

From the confusion matrix on Figure 3.30 we can see that among 729 values, 713 values are classified towards the true negative. The confusion matrix also shows that there are no true positives. As a matter of fact it is happening because the dataset highly leans towards the negative values. To justify the accuracy, precision and recall, we can use the values and compute them from the confusion matrix.

Accuracy: $(TP+TN)/total = (0+713)/729 = 0.9780$

Error Rate: $(FP+FN)/total = (5+11)/729 = 0.0219$

Precision: $TP/predicted\ yes = 0/5 = 0$

Recall: $TP/actual\ yes = 0/11 = 0$

In our case, the numerator for calculating precision and recall is 0 as there are no true positives found while classification. In addition, precision, recall and f1-measure all gives null values if true positives are 0 and one or both of the two counters- false positive and false negative is larger than 0. Since precision and recall is very low, it is clearly understandable that the class is poorly handled by the model.

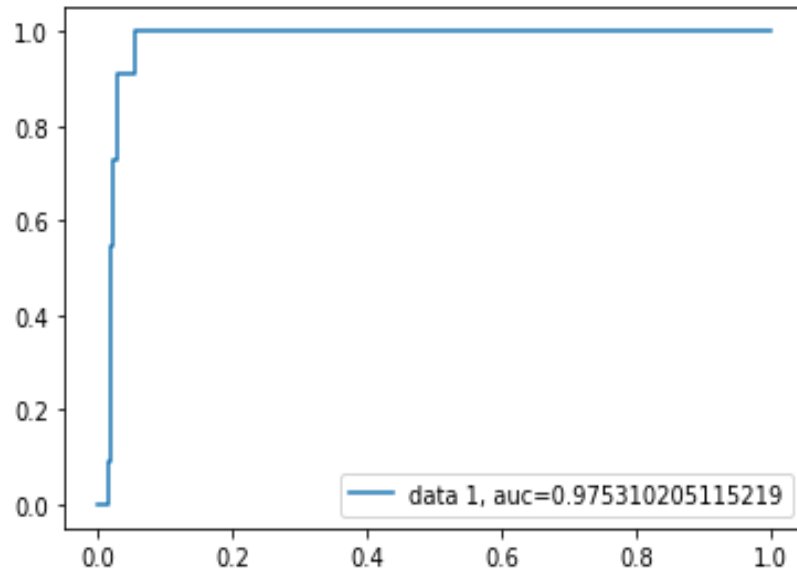


FIGURE 3.31: AUC Curve

However, Figure 3.31 shows the performance of the classification model that we plotted to distinguish between the positive and negative classes. It is called the AUC curve. Since the AUC value is more than 0.5 and less than 1, it is certain that the classifier is able to distinguish the positive classes from the negative classes. Nevertheless, as the dataset is biased towards the negative values, it was easier to distinguish the classes comparing to an evenly distributed dataset.

3.4.4 RNN

In our time series dataset, we separated the data machine wise. After separating we used LSTM (Long short-term-memory) to predict our future value. There were 18 features and after converting the datetime column from object to datetime format, we used datetime as index and picked 8 features primarily.

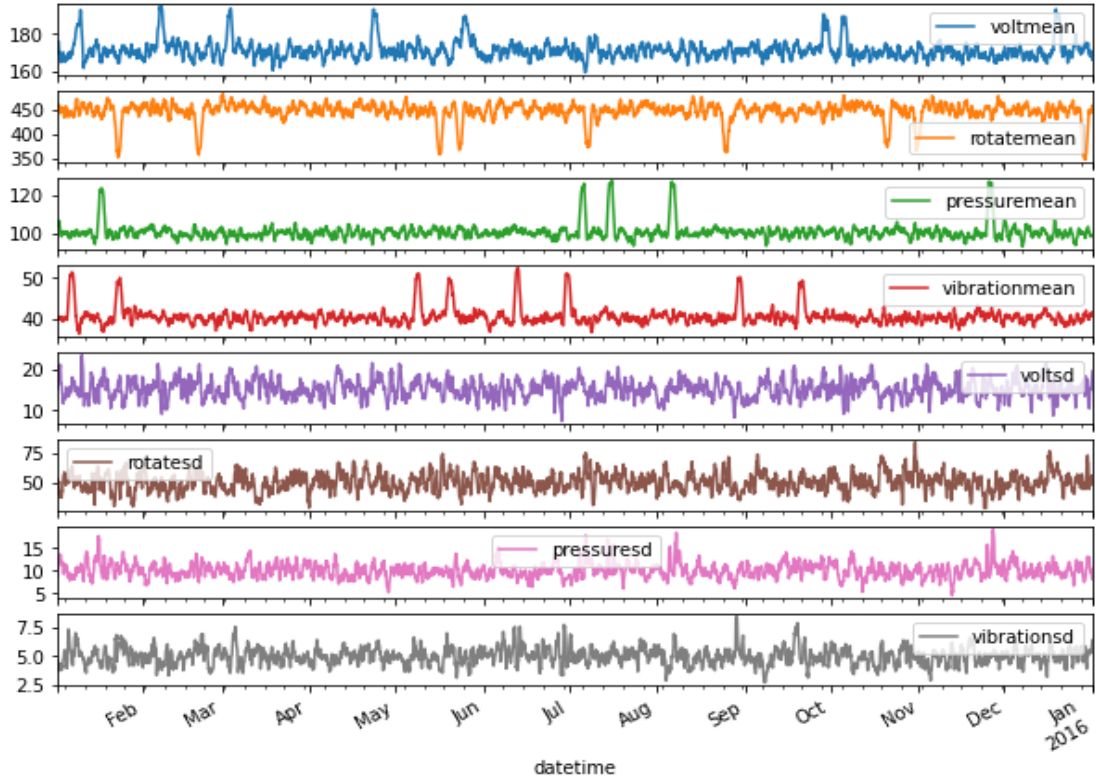


FIGURE 3.32: Feature Explanation

After that we took all those features as input and described those and then came out with the percentage of data as follows:

	voltmean	rotatemean	pressuremean	vibrationmean	voltsd	rotatesd	pressuresd	vibrationsd
count	2912.000000	2912.000000	2912.000000	2912.000000	2912.000000	2912.000000	2912.000000	2912.000000
mean	171.180190	446.076156	100.827758	40.490782	15.163814	50.641496	10.070617	5.006493
std	4.879168	18.806198	4.143935	2.128230	2.302057	7.646995	1.663109	0.750544
min	159.200937	346.559818	93.202416	36.202199	7.233255	28.362964	4.502620	2.658384
25%	168.197474	441.462151	98.872187	39.458699	13.560744	45.245075	8.958067	4.530929
50%	170.423058	448.987861	100.304986	40.151403	15.093639	50.431195	9.910021	4.984592
75%	172.886459	456.506272	101.740098	40.871834	16.711157	55.440617	11.031054	5.455985
max	194.825531	482.486087	127.065102	52.310798	23.429673	84.227878	19.065755	8.476983

FIGURE 3.33: Percentage of data

After visualizing the data, we found that almost 75% of the voltmean data was within the value of 172.88. So, we took the value of greater than 172 just to take a look at the number of outliers we have. After that there were 969 outliers having 8 columns and before fitting into the model, we scaled it because after scaling it had standard deviation and standard mean for the gradient to convert faster. After scaling we prioritized first index value as target and predicted the future value of

voltmean.

Secondly, the input values were multivariant data and output was the value of voltmean. Using time series generator, we used test size as 20% and train size as 80% and also, we took shuffle as false because in the time series data the order is very important. So, we just made sure that there won't be any shuffle and split the data. In our model we did not pass time at all, we just passed the feature. We used Sigmoid as an activation function where the input of the function was transformed into a value of 0.0 and 1.0. Then, we used the 19 days of data as window length. We trained the model using fit generator and we took each input of hidden layer. When we summarized our model, it looked like this:

Layer(type)	Output Shape	Param(#)
lstm(LSTM)	(None, 152,128)	70144
leaky.relu(LeakyRELU)	(None, 152,128)	0
lstm.1(LSTM)	(None, 152,128)	131584
leaky.relu.1(LeakyRELU)	(None, 152,128)	0
dropout(Dropout)	(None, 152,128)	0
lstm.2(LSTM)	(None, 64)	49408
dropout.1(Dropout)	(None, 64)	0
dense(Dense)	(None, 1)	65
dense.1(Dense)	(None, 1)	2
Total params		251,203
Trainable params		251,203
Non-trainable params		0

TABLE 3.9: Model Summary

Next when we defined the model using early stopping, it took 24 epochs out of 50 and we noticed that the mean absolute error of validation was 0.0446 and the mean absolute error of training was 0.0480. After evaluating the model, we created difference between the original and predicted value.

	voltmean	rotatemean	pressuremean	vibrationmean	voltsd	rotatesd	pressuresd	vibrationsd	voltmean_Pred
2482	176.849737	456.533986	97.007333	41.065584	14.990750	53.476383	7.349509	4.816302	172.986630
2483	173.150519	456.095481	96.011507	40.888623	16.390854	54.686848	9.670566	4.460066	174.086100
2484	170.222496	467.650767	95.993905	41.065175	15.527458	48.091938	9.372604	4.536985	174.370943
2485	169.923083	459.572124	97.685029	40.473364	14.594236	56.370823	9.971743	5.170654	172.721087
2486	171.493632	446.004956	98.602977	39.840799	13.865288	56.256145	10.958804	5.023743	170.235092
...
2908	169.478473	443.839825	99.203280	40.911942	15.900749	53.559810	9.614154	5.821800	168.338406
2909	168.454482	451.086112	98.667426	41.417374	17.100773	56.202362	8.357704	5.892207	168.697102
2910	166.080161	458.127506	98.494356	40.490408	17.838420	51.208494	8.234211	6.172692	169.080634
2911	170.914938	457.114192	98.400433	41.489346	19.627635	53.248743	8.439110	6.431580	168.789488
2912	167.747274	453.480750	98.956782	40.985822	19.583214	54.137639	8.092884	6.135890	169.106756

431 rows x 9 columns

FIGURE 3.34: Final Prediction

Our prediction shape was 431 which was basically our prediction frame (Shown in Figure 3.34). After counting the final prediction frame it matched with our prediction count. Here we see that the difference between the prediction value and our actual value is very minimal. After evaluating the model we found that the accuracy was 38.70%. The final predicted plot(Figure 3.35) is attached below where X axis represents the total numbers of data and Y axis shows the voltmean data.

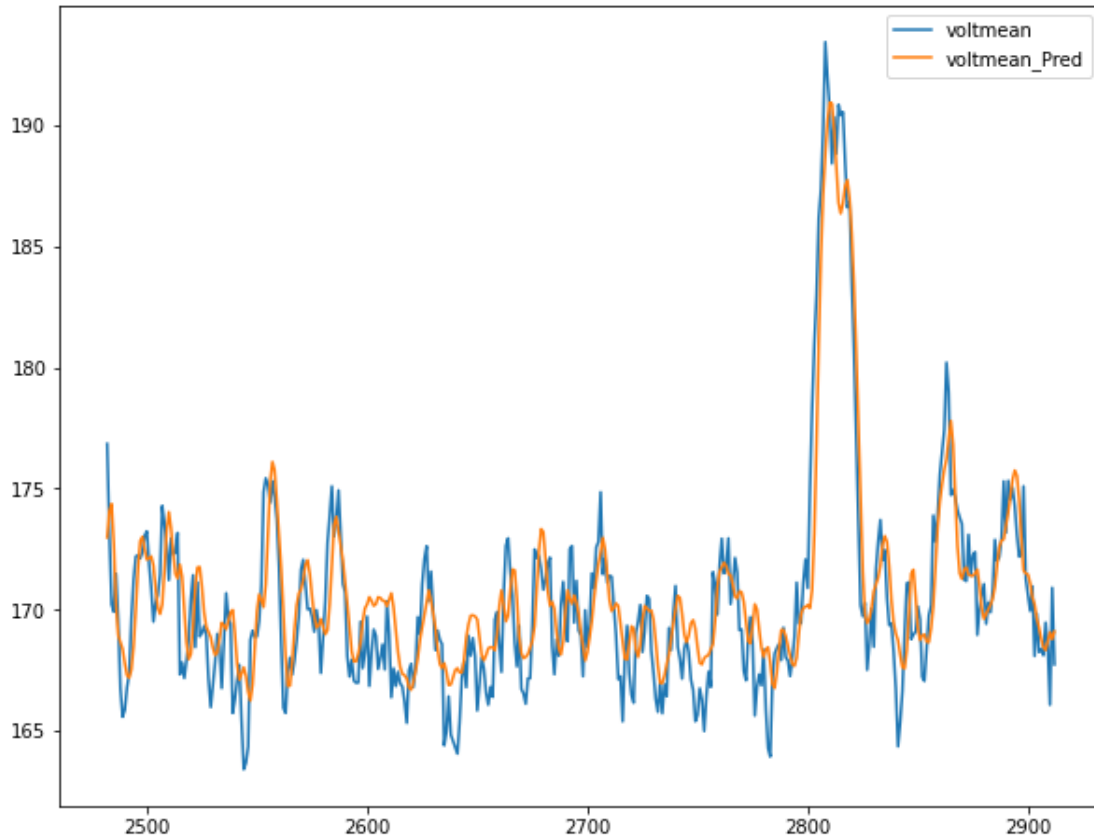


FIGURE 3.35: Final Prediction plot

The final LSTM model for our Machine had a MAE value of 1.5899, MSE value of 4.0165 and RMSE value of 2.00412. This high values of RMSE were due to small number of high error predictions.

3.4.5 Random Forest

Random Forest was exploited to make our desired prediction. At first, the model was trained using the raw data. Eight variables (voltmean, rotatemean, pressuremean, vibrationmean, voltsd, rotatesd, pressuresd, vibrationsd) were decided as independent and the only one variable (failure) which indicates whether the machine is functioning properly or not was labeled as dependent variable. The dataset was split into training set and test set. The training set contained 80% of the dataset and the test set consisted of the rest 20%. At first, 20 decision trees were taken as the estimator for the random forest. We experimented with this number but found out that the output does not drastically change because of it. So, randomly the estimator value was chosen as 35. In this way, the accuracy we got was 99.3%.

Clearly it was overfitting and that is why we moved to another strategy.

To tackle overfitting, weighted moving average of lagged values of the past 24 hours was considered. Weighted moving average for each row of the column was found out. Then with the help of these new generated values we again trained the model. The result of it could be found out from the following confusion matrix.

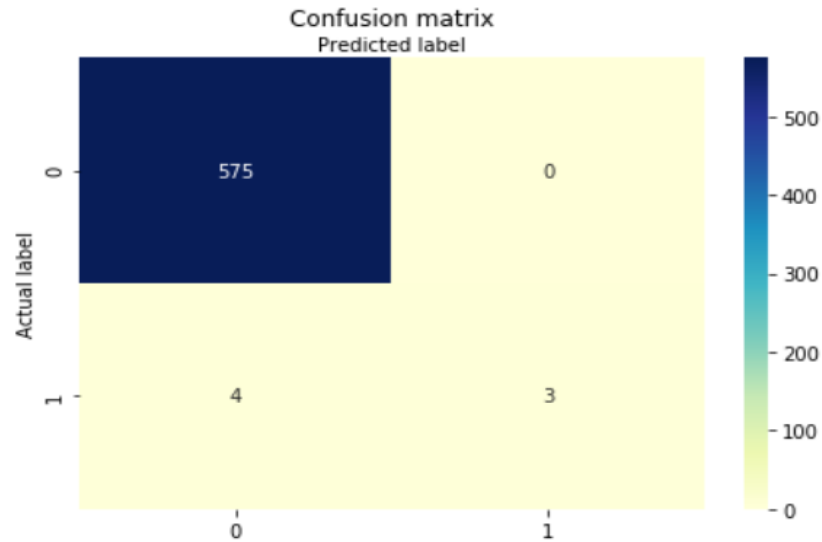


FIGURE 3.36: Confusion Matrix for Random Forest model

Figure 3.36 shows how imbalanced the dataset is. Because of this imbalance, 575 predictions fall under true negative. Only 4 predictions fall under the category false negative. Unfortunately, false positive contains 0. This again, draws attention towards the imbalance of the dataset. Similarly, only 3 predictions were categorized under true positive. Keeping the imbalance of the dataset in mind, this accuracy is acceptable.

3.5 Result and final thoughts

Our goal was to explore the use of different machine learning algorithms for classification of sensor data for predictive maintenance. In total, we looked through 5 machine learning models but none of them performed at a satisfactory level. This is not due to our approach or the modeling to the process. This is due to the limitations of the dataset that we used. Utilizing highly imbalanced datasets is a difficult task and when it comes to time series analysis, the job is even more demanding.

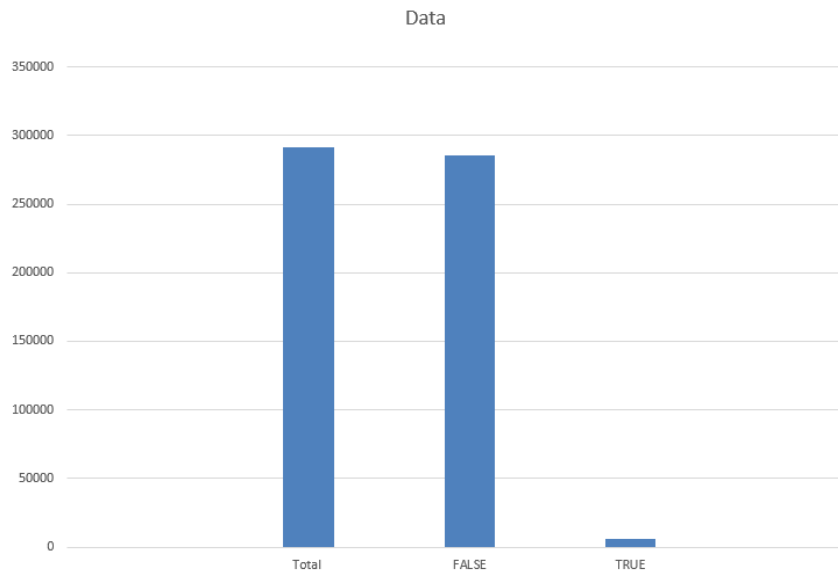


FIGURE 3.37: Data Distribution

The dataset that we used had a total sample size of 291300, of which only 5595 samples show true value on the dependent feature failure. Which is roughly 1.92% of the entire dataset (Shown in Figure 3.37). Rest of the data shows false true on the failure feature. Even if we used the entire sample size to train our model which we could not due to hardware limitations, there will be some flaws in the model and performance will not be satisfactory as our liking. The algorithms that we have used are commonly used for classification and performs well in almost all the cases where datasets are somewhat balanced in terms of different clusters.

The unusually high number of 'false' data in failure feature makes our model biased towards predicting every classification as 'false'. Therefore, the correct 'true' value classification is almost nonexistent in every classification model. Also dealing with time series data is a sensitive task as one misstep can ruin the performance of the model.

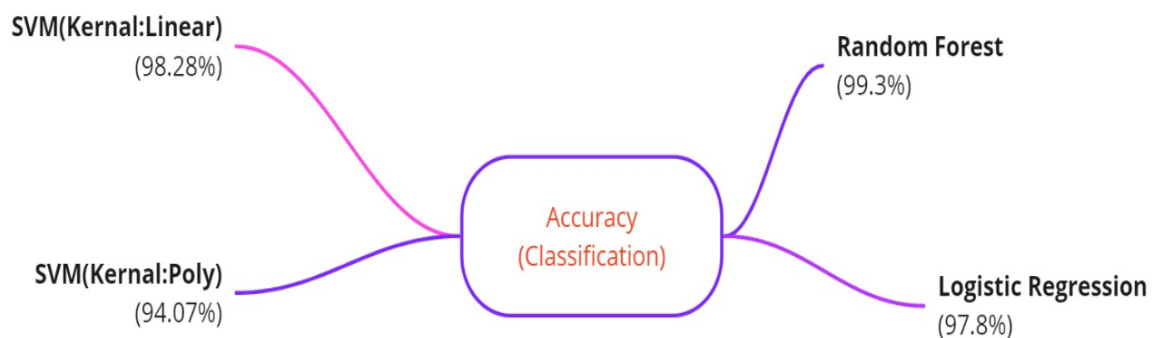


FIGURE 3.38: Performance Analysis(Classification)

Support Vector Machine, Logistic Regression, Random Forrest all performed well in terms of classifying false values in failure feature(Figure 3.38) because of the huge number of samples in the dataset. If there was more balance between two classes, these algorithms would have performed at an adequate level. Therefore, finding more balanced sample dataset will be a good start for implementing such models in future.

For the LSTM forecasting we tried to figure the lowest possible MAE error on (Figure 3.39). So, we have come up with the least possible error as follows:

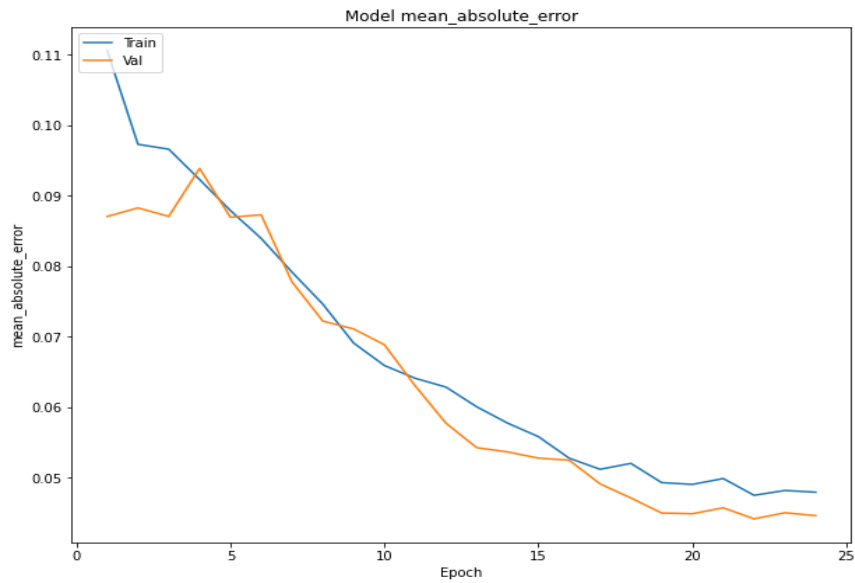


FIGURE 3.39: MAE plot

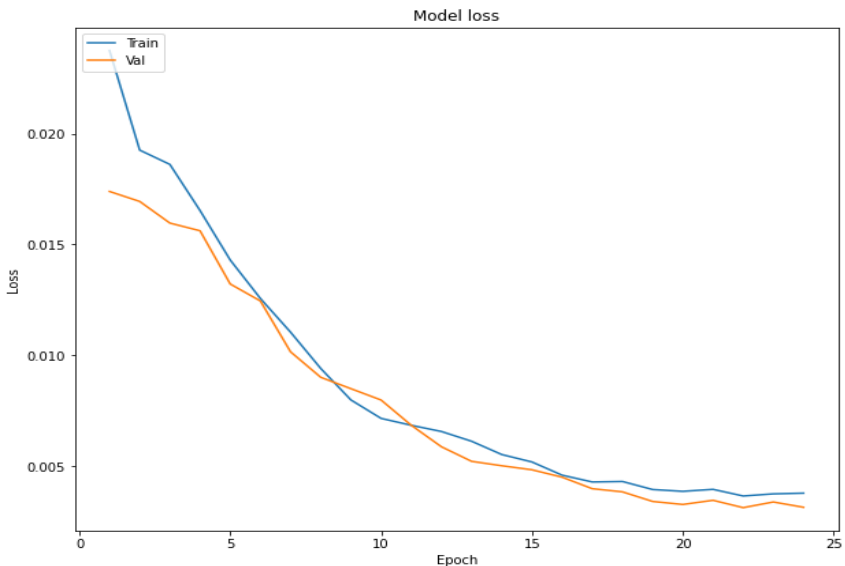


FIGURE 3.40: Model loss plot

We tried to reduce the loss but due to data set limitation we could not decrease the loss effectively. However, Figure 3.40 shows that we were able to reduce the loss from 0.0871 to 0.0446.

Chapter 4

Conclusion

4.1 Conclusion

The ultimate goal of the predictive maintenance approach is to perform maintenance at a scheduled point in time when the maintenance activity is most cost-effective and before the equipment loses performance within a threshold period. In recent times, a lot of progress has been made in this sector due to the easier availability of sensor data and uprising in the use to machine learning technologies. Most of the works in predictive maintenance was performed using one or two machine learning models. Hence, we tried implementing multiple machine learning models and data preparation approaches to classify and predict failure time from specific machines' sensor data. It was our key motivation for choosing this topic.

4.2 Future Plan

In future, we would like to work on other types of machines' sensor data and use different algorithms for classification and prediction of maintenance time and compare the performance for each type of algorithm and datasets. More work of this type will increase the scope of using the predictive maintenance approach and make will help making it a common practice in the industrial sector.

Bibliography

- [1] R. K. Mobley, *An introduction to predictive maintenance*. Elsevier, 2002.
- [2] C. Scheffer and P. Girdhar, *Practical machinery vibration analysis and predictive maintenance*. Elsevier, 2004.
- [3] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, “Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005, pp. 64–75.
- [4] S. Orhan, N. Aktürk, and V. Celik, “Vibration monitoring for defect diagnosis of rolling element bearings as a predictive maintenance tool: Comprehensive case studies,” *Ndt & E International*, vol. 39, no. 4, pp. 293–298, 2006.
- [5] W. W. Wei, “Time series analysis,” in *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*, 2006.
- [6] S.-j. Wu, N. Gebraeel, M. A. Lawley, and Y. Yih, “A neural network integrated decision support system for condition-based optimal predictive maintenance policy,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 2, pp. 226–236, 2007.
- [7] H. M. Hashemian, “State-of-the-art predictive maintenance techniques,” *IEEE Transactions on Instrumentation and measurement*, vol. 60, no. 1, pp. 226–236, 2010.
- [8] M. Hosek, J. Krishnasamy, and J. Prochazka, *Intelligent condition-monitoring and fault diagnostic system for predictive maintenance*, US Patent 7,882,394, Feb. 2011.
- [9] G. Köksal, İ. Batmaz, and M. C. Testik, “A review of data mining applications for quality improvement in manufacturing industry,” *Expert systems with Applications*, vol. 38, no. 10, pp. 13 448–13 467, 2011.
- [10] P. D. Koprinkova-Hristova, M. B. Hadjiski, L. A. Doukowska, and S. V. Beloreshki, “Recurrent neural networks for predictive maintenance of mill fan systems,” *International Journal of Electronics and Telecommunications*, vol. 57, no. 3, pp. 401–406, 2011.
- [11] L. Mönch, J. W. Fowler, S. Dauzere-Peres, S. J. Mason, and O. Rose, “A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations,” *Journal of scheduling*, vol. 14, no. 6, pp. 583–599, 2011.
- [12] G. I. Platforms, “The rise of industrial big data,” *GE Intelligent Platforms*, 2012.

- [13] G. A. Susto, A. Beghi, and C. De Luca, “A predictive maintenance system for epitaxy processes based on filtering and prediction techniques,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 4, pp. 638–649, 2012.
- [14] G. A. Susto, A. Schirru, S. Pampuri, D. Pagano, S. McLoone, and A. Beghi, “A predictive maintenance system for integral type faults based on support vector machines: An application to ion implantation,” in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2013, pp. 195–200.
- [15] S. Alestra, C. Brand, E. Burnaev, P. Erofeev, A. Papanov, C. Bordry, and C. Silveira-Freixo, “Rare event anticipation and degradation trending for aircraft predictive maintenance,” in *11th World Congress on Computational Mechanics, WCCM*, vol. 5, 2014, p. 6571.
- [16] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, “Log-based predictive maintenance,” in *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, 2014, pp. 1867–1876.
- [17] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, “Machine learning for predictive maintenance: A multiple classifier approach,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2014.
- [18] M. Traore, A. Chammas, and E. Duviella, “Supervision and prognosis architecture based on dynamical classification method for the predictive maintenance of dynamical evolving systems,” *Reliability Engineering & System Safety*, vol. 136, pp. 120–131, 2015.
- [19] G. A. Susto and A. Beghi, “Dealing with time-series data in predictive maintenance problems,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2016, pp. 1–4.
- [20] A. Kanawaday and A. Sane, “Machine learning for predictive maintenance of industrial machines using iot sensor data,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, 2017, pp. 87–90.
- [21] F. Giobergia, E. Baralis, M. Camuglia, T. Cerquitelli, M. Mellia, A. Neri, D. Tricarico, and A. Tuninetti, “Mining sensor data for predictive maintenance in the automotive industry,” in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2018, pp. 351–360.
- [22] S. Roy, S. Nanjiba, and A. Chakrabarty, “Bitcoin price forecasting using time series analysis,” in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, IEEE, 2018, pp. 1–5.
- [23] S. Spiegel, F. Mueller, D. Weismann, and J. Bird, “Cost-sensitive learning for predictive maintenance,” *arXiv preprint arXiv:1809.10979*, 2018.
- [24] T. Abbasi, K. H. Lim, and K. San Yam, “Predictive maintenance of oil and gas equipment using recurrent neural network,” in *Iop conference series: Materials science and engineering*, IOP Publishing, vol. 495, 2019, p. 012067.