Master's Degree Thesis

CNC end effector tools wear SOH prediction: big data approach

Candidate:
Mudassar Hussain

Mentor:
Prof. Daniele Apiletti

Supervisor:
Domenico Gatto
Brain technologies srl.

Spring session 2023

Abstract

Pertaining to the high cost of maintenance of various machines in the industrial sector, there is a growing need of efficient and cost saving techniques to predict the state of health(or remaining useful life of a machine). It is obvious that applying efficient techniques can lead to a competitive advantage in the industrial sector. Our thesis extends the work done previously on the MorePRO project based on the estimation of the state of health of CNC machines. The contrast with the previous work is to focus on cloud based computing rather then previously performed analysis with edge computing which is related to the data collected in the local environment from the machine. Both of these techniques have there own pros and cons. Mainly edge computing being local and thus has low latency while cloud computing encompasses a general analysis consequently leads to better accuracy.

The goal is to understand the model pertaining the CNC machine, identifying the useful features that need to be extracted during the ETL process and then apply efficient machine learning algorithm on the extracted dataset using a cloud based approach, deploying the model using real time streaming. Main steps include loading dataset and performing analytic in a spark based environment. The data exploratory analysis is performed using python based library along with visualization. Various regression models are trained and validated using test data. Finally designing pipeline integrating spark streaming with Apache Kafka

Our work focuses on performing regression task to predict the state of health (assumed as a continues variable). Since the data is coming from various models, we need to concatenate these data sets and then perform analysis on a cloud based platform given that the volume of data collected from various models requires more memory and consequently high computation power. We would like to mention that these computations are performed on the cluster provided by Politecnico di Torino.

Contents

Chapter 1

Introduction

1.1 Maintenance

System dependability is one of the most important challenges in today's business, the development of improved system maintenance approaches based on data obtained through system or component monitoring (or system state estimate) and equipment failure prognostics is an emerging topic (or system state forecasting). Such procedures can be classified into two groups, according to the EN 13306 (2001) standard. The first is corrective maintenance, which entails replacing a component and fixing any damage that has occurred as a result of a catastrophic failure. When the repercussions of a failure aren't as serious and field intervention doesn't need a lot of money or time, this strategy is adopted.

1.1.1 Palliative maintenance

Palliative maintenance is used when the repair is temporary, while curative maintenance is used when the repair is permanent.

1.1.2 Preventative maintenance

The second is preventative maintenance, which refers to the provision of an alarm before defects reach critical levels in order to avoid system performance decline, malfunction, or even catastrophic failure. Predetermined maintenance is when a maintenance intervention is time-based, meaning that components are changed according to a predetermined timetable based on the component's working hours. Obviously, this strategy is inefficient since components are changed before they reach the end of their useful life, raising expenses.

1.1.3 Condition based maintenance

Condition-based maintenance, which refers to the analysis of real-time data in order to detect a probable failure in the changing of their characteristics, is one such option. However, this method does not guarantee that a maintenance strategy will be designed with confidence. Predictive maintenance, on the other hand, uses more dynamic algorithms to estimate the machine's SoH.

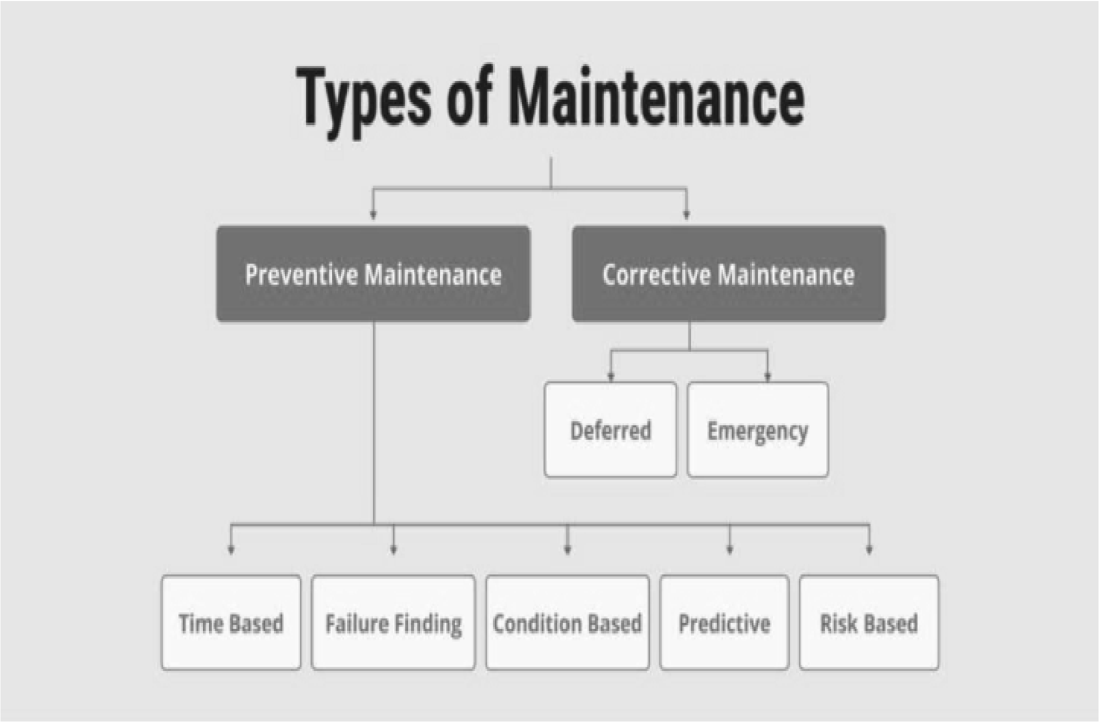


Figure 1.1: Types of predictive maintenance

1.1.4 Predictive maintenance

Predictive maintenance is a technique that uses data analysis tools and techniques to detect anomalies in the operation and possible defects in equipment and processes so as to fix them before they result in failure.

Ideally, predictive maintenance allows the maintenance frequency to be as low as possible to prevent unplanned reactive maintenance, without incurring costs associated with doing too much preventive maintenance.

Predictive maintenance uses historical and real-time data from various parts of the operation to anticipate problems before they happen. There are several key elements to predictive maintenance with technology and software being one of these critical pieces. Namely, the Internet of Things (IoT), artificial intelligence, and integrated systems allow for different assets and systems to connect, work together and share, analyze, and action data.

These tools capture information using predictive maintenance sensors, industrial controls, and businesses systems. They then make sense of it and use it to identify any areas that need attention. Some examples of using predictive maintenance and predictive maintenance sensors include vibration analysis, oil analysis, thermal imaging, and equipment observation.

Identifying critical assets, creating a database for historical data, analyzing failure modes, making failure predictions, and finally deploying predictive maintenance technology to a group of pilot equipment to validate the program are all steps in the implementation of a predictive maintenance program.

We have the following benefits of predictive maintenance:

- Reduce the number of unanticipated breakdowns.
- Increase asset dependability and maximize asset up time.
- Reduce operating costs by only doing maintenance when it is absolutely essential, increase production hours, and enhance safety.
- Reduce maintenance expenses by reducing equipment, inventory, and personnel costs.

1.2 SoH of CNC machine

CNC (Computer Numerical Control) machines are high-precision machines that automate material-subtraction industrial operations that require computerized control to ensure high precision and efficiency. In a subtractive manufacturing process, machine tools are used to remove layers of material from a stock component known as the blank or work-piece, resulting in a custom-designed product. This form of processing is essentially unaffected by the material used to make the workpiece: plastics, metals, foam, glass, and so on. This is why CNC machines are used in almost every aspect of industrial processing.

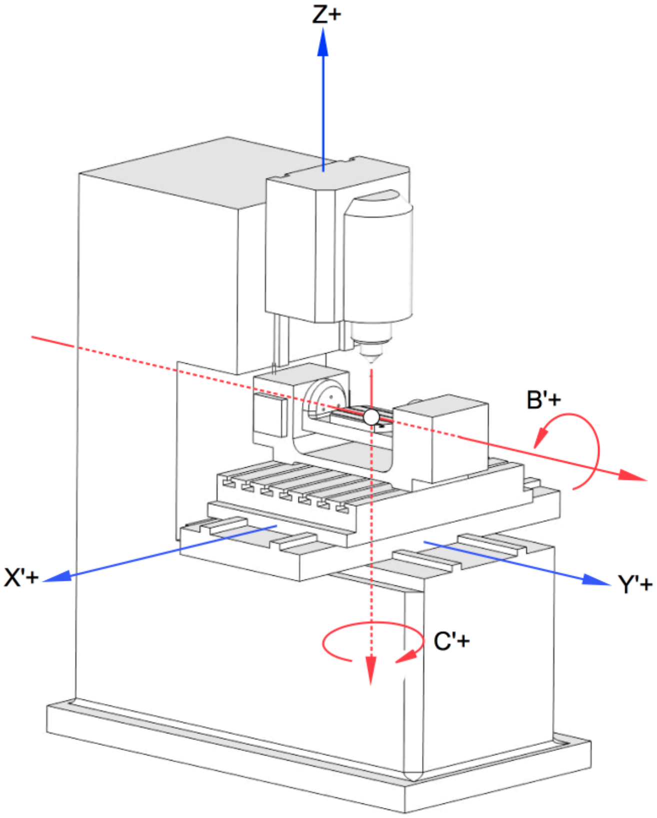


Figure 1.2: Schematic diagram of a CNC machine

These machines often feature a SCARA or cartesian robotic arrangement with an end-effector, which is commonly a cutter, as seen in figure. Modeling of the cutter contact is complex because of the large number of variables that must be taken into account, the most important of which are:

- Robotic configuration.
- Environmental parameters.
- Wear condition of the machine: SoH.

All of these factors must be kept under regular surveillance in order to ensure the machine's efficiency and accuracy. There is no direct means to check the state of health of the end-effector in particular. It may be feasible to install sensors to monitor temperature, voltage, and pressure, as well as assess the tools' potential SoH. However, even with knowledge of variables that can be measured by sensors, extracting information about the machine's actual condition is difficult, firstly because sensors are unlikely to be installed in the correct position, and secondly because knowledge of those parameters may not be sufficient to comprehend the real situation. For example, a temperature sensor cannot be placed close enough to the cutter to accurately detect the temperature, thus it must be placed further away, resulting in continual and unavoidable measurement inaccuracies.

The bulk of methods that estimate the SoH nowadays recommend digital-twin solutions. The difficulty of isolating tool wear from other observed effects is a shortcoming of such a system. Furthermore, such monitoring systems estimate SoH using machine learning approaches and digital-twin simulation without accounting for processing needs.The most influential parameters in the SoH estimation are those related to the cutting process of the end-effector: the most stressed mechanical elements. Digital-twin models are very useful when the variables that need to be controlled are numerous, but the most influent parameters in the SoH estimation are those related to the cutting process of the end-effector: the most stressed mechanical elements.

As a result, there are several parameters that are directly connected to the SoH, with the following being some of the most important:

- Friction coefficients.

- Temperature.
- Chip load.

Because those factors are inextricably linked to contact forces, knowing and modeling them is critical for estimating the state of health of CNC machine end-effectors.

1.3 Previous projects reference

This project research, as well as the entire thesis, is part of a growing list of projects managed by brain Technologies srl. Because MorePRO evolved from certain concepts created in prior projects, it is vital to have a preliminary review of the ideas and principles that make up the preceding initiatives.

The projects prior to this thesis are following:

- The BAT-MAN research and development project, which is a brain Technologies-owned industrial project, is the starting point for the application of an innovative approach based on an EKF bank, with the main goal of creating an electronic device capable of detecting and forecasting the working conditions of a Lead-Acid battery in real-time.
- The ERMES (Extendible Range MultiModal Estimator Sensing) algorithm was created by Brain Technologies, and its innovative value is to develop approaches to apply to the problem of accumulation system diagnostics, and in particular to the problem of battery SoH estimate. The suggested ERMES method for estimating the state of health (SoH) and state of charge (SoC) is based on a model with augmented state, which implies that the unknown parameters associated with SoH and SoC are treated as states rather than outputs.The approach entails creating a battery model using an analogous circuit and a bank of N EKF (Extended Kalman Filter) based on distinct SoH hypotheses.
- Thesis written by students of politecnico in collaboration with Braintech based on development of the machine model and applying edge computing algorithms for the SoH estimation of end-effector.

1.4 Cloud computing

Cloud computing is the supply of on-demand computing services through the internet and on a pay-as-you-go basis, ranging from apps to storage and processing power. Companies can rent access to everything from apps to storage from a cloud service provider rather than having their own computing equipment or data centers.

One advantage of cloud computing is that businesses may avoid the upfront costs and complexity of building and maintaining their own IT infrastructure by paying only for what they need, when they need it.

As a result, cloud-computing service providers may achieve enormous economies of scale by providing the same services to a diverse set of consumers.

1.4.1 Cloud computing services

Cloud computing services today include everything from basic storage, networking, and processing power to natural language processing and artificial intelligence, as well as common office programs. Almost any service that doesn't require you to be physically adjacent to the computer gear you're using, including quantum computing, may now be offered over the cloud.

A large number of services rely on cloud computing. Consumer services such as Gmail and cloud backups of your smartphone images are examples, as are services that allow major organizations to host all of their data and operate all of their programs in the cloud. Netflix, for example, uses cloud computing to power its video-streaming service as well as its other business operations.

Software providers are increasingly delivering their programs as services through the internet rather as separate goods as they aim to transition to a subscription model, and cloud computing is becoming the default choice for many apps.

1.4.2 Background

The location of the service, as well as many other variables such as the hardware or operating system on which it is running, are essentially irrelevant

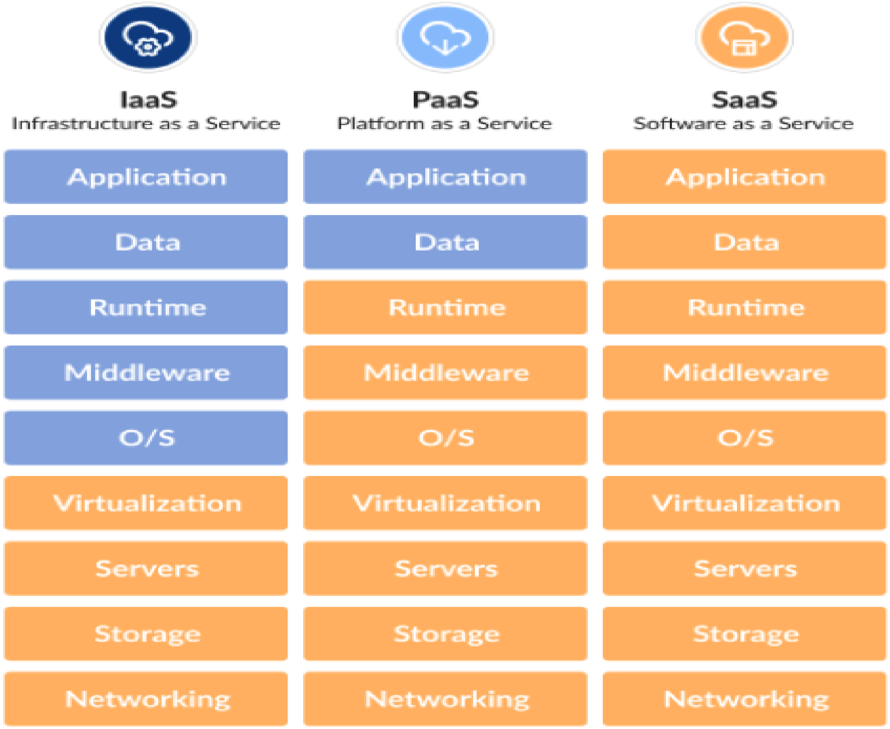


Figure 1.3: Cloud Services

to the user in cloud computing. The cloud metaphor was drawn from ancient telecommunications network designs, in which the public telephone network (and subsequently the internet) was typically shown as a cloud to indicate that the location didn't matter — it was simply a cloud of stuff. Of course, this is an oversimplification; for many consumers, the location of their services and data is still a major concern.

The name ”cloud computing” has been used since the early 2000s, although the notion of ”computing as a service” dates back to the 1960s, when computer bureaus offered firms the option of renting time on a mainframe rather than purchasing one.

These 'time-sharing' services were mostly replaced by the PC, which made owning a computer much more cheap, and subsequently by the growth of corporate data centers, which allowed firms to store massive quantities of data.

However, in the late 1990s and early 2000s, the notion of renting access to computer power reappeared in the form of application service providers, utility computing, and grid computing. The introduction of software as a service and hyperscale cloud-computing providers like Amazon Web Services was followed by cloud computing, which truly took off with the advent of software as a service and hyperscale cloud-computing providers like Amazon Web Services.

1.4.3 Significance of cloud computing

As computing workloads continue to migrate to the cloud, whether through public cloud services offered by vendors or private clouds built by enterprises themselves, infrastructure to support cloud computing now accounts for a significant portion of all IT spending, while spending on traditional, in-house IT continues to decline.Indeed, the cloud is winning when it comes to corporate computing platforms.

According to Gartner, by 2025, up from 41% in 2022, up to half of all investment in the application software, infrastructure software, business process services, and system infrastructure sectors would have gone to the cloud. Cloud computing is expected to account for nearly two-thirds of application software spending by 2022, up from 57.7% in 2022.

This is a shift that increased in 2020 and 2021 as firms advanced their digital transformation plans in the aftermath of the epidemic. Throughout the epidemic, the lock downs demonstrated how critical it was for businesses to be able to access their computer infrastructure, apps, and data from everywhere their employees worked - not just from an office.

1.5 Spark

Apache Spark is a big data and machine learning analytics engine that runs at breakneck speed. It was created in 2009 at the University of California, Berkeley.It is the largest open source data processing project. Apache Spark as the unified analytics engine, has seen significant adoption by businesses across a wide range of sectors since its debut. Internet powerhouses like Netflix, Yahoo, and eBay have utilized Spark at large scale, with clusters of over 8,000 nodes processing several petabytes of data. With over 1000 contributions from 250+ firms, it has quickly become the largest open source community in big data.

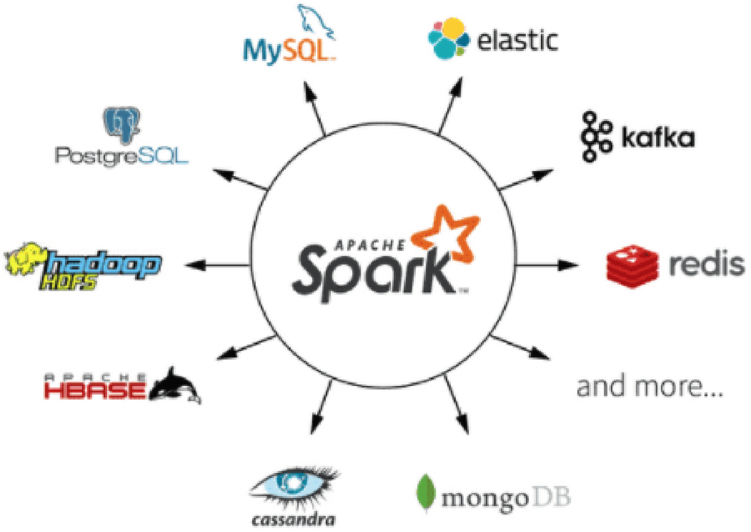


Figure 1.4: The spark open-source project

1.5.1 Pyspark

Apache spark has been developed using scala as a programming language. PySpark is a Python API for Apache Spark that was published to enable the collaboration of Apache Spark with Python. Furthermore, PySpark allows to interact with Resilient Distributed Datasets (RDDs) in Apache Spark and Python. This has been accomplished by utilizing the Py4j library. Py4j is a popular library that is built into PySpark that allows Python to interact with JVM objects dynamically. PySpark comes with a number of libraries that can help you write more efficient programs. There are also a number of other libraries that are compatible. Here we report some of them:

PysparkSQL

A Python module for doing SQL-style analysis on large amounts of structured or semi-structured data. With PysparkSQL, we can also utilize SQL queries. It's also possible to connect it to Apache Hive. HiveQL may be used as well. PysparkSQL is a wrapper around the core of Pyspark. PySparkSQL introduced the Data Frame, which is a tabular representation of structured data that looks like a table in a relational database management system.

Figure 1.5: pyspark

MLlib

MLlib is Spark's machine learning (ML) library and is a wrapper around PySpark. To store and operate with data, this library employs the data parallelism approach. The MLlib library's machine-learning API is straightforward to use. For classification, regression, clustering, collaborative filtering, dimensionality reduction, and underlying optimization primitives, MLlib provides a wide range of machine-learning methods.

Graph frames

Graph frames is a graph processing toolkit that uses the Pyspark core and PysparkSQL to provide a set of APIs for doing graph analysis quickly. It's designed for high-performance distributed computing.

The following are some of the benefits of utilizing Pyspark:

- Python is a fairly simple language to learn and use.
- It has a user-friendly and extensive API.
- Python provides considerably greater code readability, maintenance, and familiarity.
- It has a variety of data visualization capabilities, which is tough to do with Scala or Java.

Chapter 2

CNC machine model

Mathematical model design is the skill of transforming physical problems from an application field into tractable mathematical equations whose theoretical and numerical analysis gives insight, solutions, and direction valuable for the original application. However, due to the complexity and large number of aspects that such techno-logic instruments may achieve, modeling a CNC machine might be a very difficult task. Given the great complexity of a CNC machine, it has been chosen to start with a minimal model to allow extraction of data as soon as possible and to gain some useful findings from a basic simulation environment. The simulation's major goal is to understand and simulate the behavior of a certain manufacturing system on a computer prior to real production, decreasing the number of testing and experimentation on the shop floor. Less material is wasted when a virtual system is used, and disruptions in the functioning of a real machine on the job site are avoided.

2.1 Mechanical part

In terms of the mechanical element, a very basic milling machine model is employed. In specifically, a rotating disc is considered in the image below, which moves in the pieces direction in order to cut it.

Regarding the mechanical part of the model, we need to specify variables related to the model as follows:

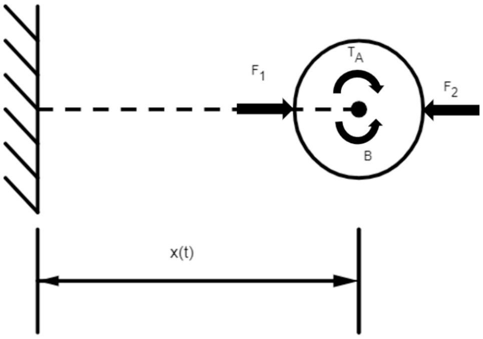


Figure 2.1: Basic milling machine model

- $\dot{\theta}$: Rotational velocity.
- $\dot{x}$: Linear velocity.
- F_1 : Horizontal forces that move the cutter.
- F_2 : Normal Force due to contact.
- f_c : Binary function that defines the presence of contact. Assumes 1 value when the work piece is present or 0 otherwise.
- T_a : DC motor torque applied to the cutter.
- I_n : Inertia of the motor and the cutter.
- β :Contact rotational friction.
- Δ_x Depth of cutting.
- *cost* : Minimum contact force (introduced in order to avoid model discontinuities).

Given the above mentioned parametrs we can deduce the following equations:

$$\begin{cases} \ddot{\theta} = \frac{T_a - \beta\dot{\theta}F_c}{I_n} \\ \ddot{x} = \frac{F_1 - f_c(F_2\Delta_x + cost)}{m} \end{cases} \tag{2.1}$$

2.2 Electrical part

Modern CNC machines, are powered by brushless or servo motors for the electrical portion. Fast responsiveness to commands, strong acceleration and deceleration qualities, the ability to manage velocity safely in all velocity ranges, and extremely accurate position control are the most critical attributes required for servo motors that drive CNC machines. Machines that use computer numerical control require high-resolution controllers with great accuracy. Classical and current control approaches, such as PID controllers, feedback control, feedforward control, adaptive control, and auto tuning methods, are employed at this time. A DC motor is used to drive and communicate with the mechanical portion, making the fundamental structure easier to maintain.

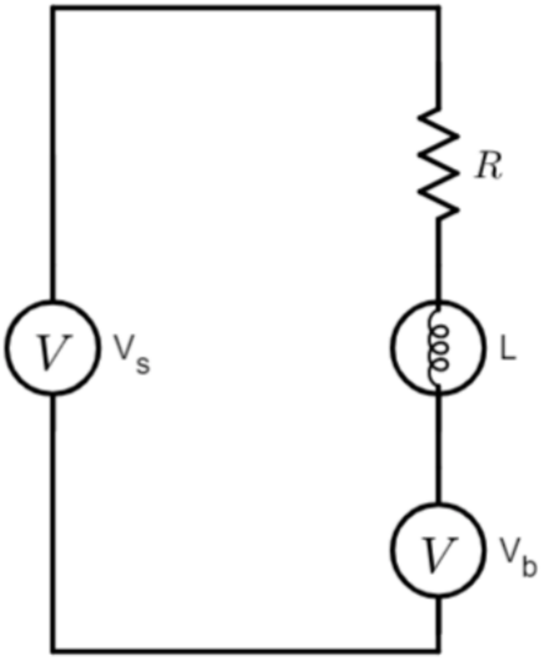


Figure 2.2: Simplified schema of DC motor

We can define the following parameters regarding the electrical part of the model:

- V_s : supply voltage.
- i_a : Armature current.
- T_a : DC motor torque applied to the cutter.
- k_t : Motor torque proportionality constant.
- L : Inductance.
- R : Resistance.
- V_b : Back E.M.F
- Att_mot : Engine friction.
- k :proportionality constant.
- b : Total flux.
- I_L : Motor inertia.

we have the following relations between these parameters:

$$\begin{cases} V_s = Ri_a(t) + L\frac{di_a(t)}{dt} \\ V_b = kb\dot{(}\theta) \\ T_a = k_t i_a(t) - Att_{mot}\dot{\theta} \end{cases} \tag{2.2}$$

where as for the angular velocity we have the following relation:

$$\omega = \frac{V_s}{k} - \frac{T_a}{k^2}R \tag{2.3}$$

2.3 Plant model

Finally we can derive the equation that represent both the electrical and mechanical parts combined. Please note that exact derivation of these equations is beyond the scope of this thesis, however they are mentioned for completeness.

$$\begin{cases} \ddot{\theta} = \frac{k_t i_a - Att_{mot}\dot{\theta} - {}_c\dot{\theta}}{I_n} \\ \ddot{x} = \frac{F_1 - f_c(F_2\alpha + cost)}{m} \\ i_a = V_s - Ri_a - k_v\dot{\theta} \end{cases} \tag{2.4}$$

2.4 β as a significant parameter

Literature research suggests that the friction coefficient plays a crucial role in the interaction between the work-piece and the end-effector tool, according to a literature review of the factors that most impact end-effector wear. The friction coefficient, abbreviated as β, is chosen as the parameter on which the filter wear hypothesis is based in this first phase of development. β depends on various factors such as temperature, the used material, relative speed, applied forces, cooling media, etc.

the key factor is that the analysis carried out during the model formulation operates independently of the precise choice of the parameter in such a manner that a complication of it might result in findings that are not too dissimilar from those produced when considering as representative.

2.5 Contact logic

Another important feature is the contact logic that demonstrates the contact of the machine with the work piece. The function has been implemented using Boolean operators. The following figure shows the design of such function in simulink as used in the project. Correlation of this feature with other features such as voltage and current will be discussed in the later chapter.

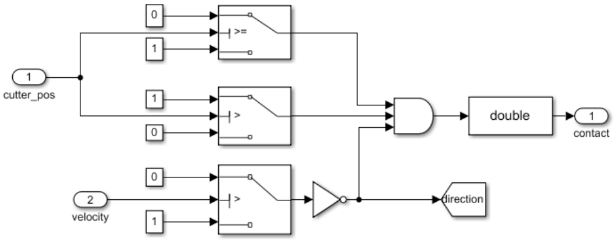
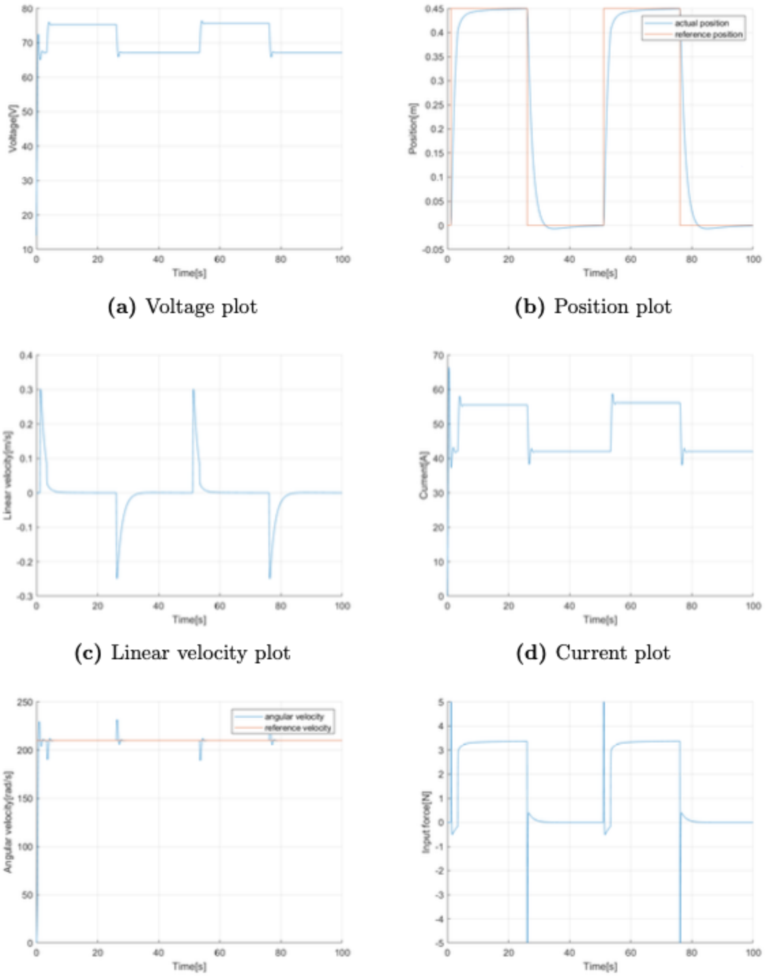


Figure 2.3: Contact logic

2.6 Trends in important features

In the following figure are shown the important features of the model. As an observation we can see that most of the features are periodic in nature for instance current and voltage. These features can be further refined in the data pre-processing phase as suggested in the later section.



(a) Voltage plot

(b) Position plot

(c) Linear velocity plot

(d) Current plot

(e) Angular velocity

(f) Input force plot

Figure 2.4: Trends in important features

Chapter 3

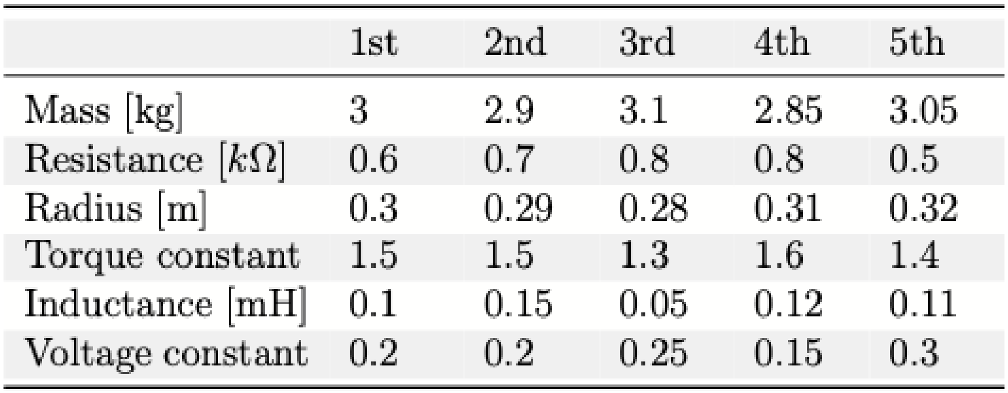
Exploratory data analysis

3.1 The data set

The input is a collection of datasets collected from the simulation of CNC machine model.The analysis has been performed using 5 datasets with parameters set according to the table in fig.3.1. Each dataset contains 9 columns and 10,000 instances. Following are the features of the dataset:

- **voltage**: Input voltage to the CNC machine.
- **current**: Current associated to the provided voltage.
- **contact_force**: Binary function that defines the presence of contact. Indeed, it assumes value when the work piece is present or 0 otherwise.
- **angular_velocity**: Rotation velocity of the cutting piece.
- **linear_velocity**: Velocity related to the movement of cutting piece.
- **cutter_position**: The displacement with respect to the initial placement of the cutting edge.
- **F1**: Horizontal force that moves the cutter.
- **F2**: Normal Force due to contact.
- **beta** : Contact friction co-efficient.
- **usurage** : Inverse of remaining useful life of the CNC machine.

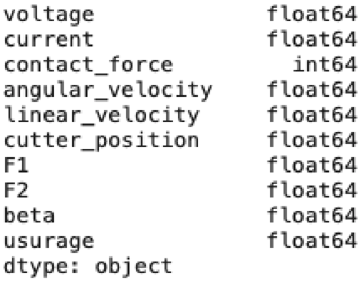
The major goal here is to collect data from numerous machines in the manufacturing facility, taking into account that each machine is slightly different from the others on a parametric level. This assumption provides for more realistic data and eliminates edge device estimate problems caused by a mismatch between the plant and the filter bank. Furthermore, it is expected, as previously, that data is collected for four alternative wear configurations for each CNC machine (and therefore of the friction coefficient). As a result, the algorithm will be based on the five machines stated in the table below.



	1st	2nd	3rd	4th	5th
Mass [kg]	3	2.9	3.1	2.85	3.05
Resistance [$k\Omega$]	0.6	0.7	0.8	0.8	0.5
Radius [m]	0.3	0.29	0.28	0.31	0.32
Torque constant	1.5	1.5	1.3	1.6	1.4
Inductance [mH]	0.1	0.15	0.05	0.12	0.11
Voltage constant	0.2	0.2	0.25	0.15	0.3

Figure 3.1: Parameters of different CNC machines

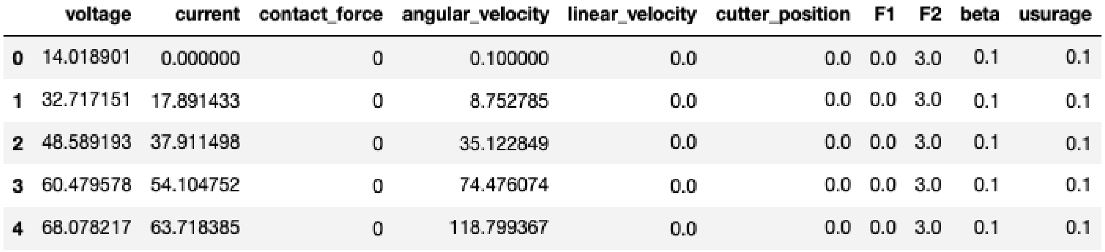
All of the features are numerical as depicted in the following figure. Considering the types of features are important as many different type of features may require a different set of preliminary analysis such as encoding, vectorization etc.



```
voltage             float64
current             float64
contact_force         int64
angular_velocity    float64
linear_velocity     float64
cutter_position     float64
F1                  float64
F2                  float64
beta                float64
usurage             float64
dtype: object
```

Figure 3.2: Data types of the features

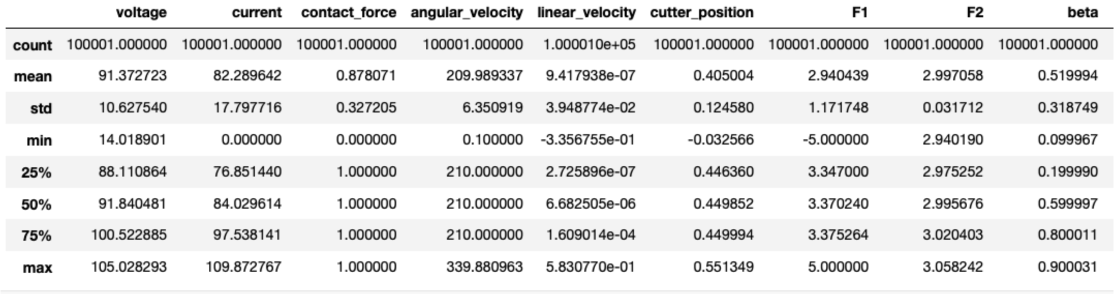
The figure below displays above mentioned features in a tabular form along with some instances from the dataset. The column on the right is to be predicted by the algorithm during the testing phase.



	voltage	current	contact_force	angular_velocity	linear_velocity	cutter_position	F1	F2	beta	usurage
0	14.018901	0.000000	0	0.100000	0.0	0.0	0.0	3.0	0.1	0.1
1	32.717151	17.891433	0	8.752785	0.0	0.0	0.0	3.0	0.1	0.1
2	48.589193	37.911498	0	35.122849	0.0	0.0	0.0	3.0	0.1	0.1
3	60.479578	54.104752	0	74.476074	0.0	0.0	0.0	3.0	0.1	0.1
4	68.078217	63.718385	0	118.799367	0.0	0.0	0.0	3.0	0.1	0.1

Figure 3.3: Features of the data set

The following figure shows a description of the features. The description reveals that the values of different features are in diverse ranges in term of min and max values. For example contact_force has max value of 1 where as angular_velocity can assume a max value of 339. Thus feature scaling is required so that the algorithm designed does not bias towards a particular feature based on its high range of values. Feature scaling can be very useful when applied keeping in mind that it may affect the appearance of the dataset in terms of the outliears or variability. Many algorithm can take advantage of this step and others are not affected. We have discussed this feature later in great detail in our later chapter.



	voltage	current	contact_force	angular_velocity	linear_velocity	cutter_position	F1	F2	beta
count	100001.000000	100001.000000	100001.000000	100001.000000	1.000010e+05	100001.000000	100001.000000	100001.000000	100001.000000
mean	91.372723	82.289642	0.878071	209.989337	9.417938e-07	0.405004	2.940439	2.997058	0.519994
std	10.627540	17.797716	0.327205	6.350919	3.948774e-02	0.124580	1.171748	0.031712	0.318749
min	14.018901	0.000000	0.000000	0.100000	-3.356755e-01	-0.032566	-5.000000	2.940190	0.099967
25%	88.110864	76.851440	1.000000	210.000000	2.725896e-07	0.446360	3.347000	2.975252	0.199990
50%	91.840481	84.029614	1.000000	210.000000	6.682505e-06	0.449852	3.370240	2.995676	0.599997
75%	100.522885	97.538141	1.000000	210.000000	1.609014e-04	0.449994	3.375264	3.020403	0.800011
max	105.028293	109.872767	1.000000	339.880963	5.830770e-01	0.551349	5.000000	3.058242	0.900031

Figure 3.4: Statistics of all the numerical features

3.2 Heat map

Heat Map chart, or heat map is a two-dimensional visual representation of data, where values are encoded in colors, delivering a convenient, insightful view of information. Heat map for the dataset is given in figure . The map reveals a strict correlation between current and voltage which is expected according to the ohm's law considering the load as equivalent resistance. There can be seen a very strong direct relation between beta and the usage. Other potent features affecting the usage are voltage, current and F2. Other features are not so effective in relation with the variable to be predicted. We can verify this impact on the feature to be predicted from the feature importance of the decision tree in the later chapter.

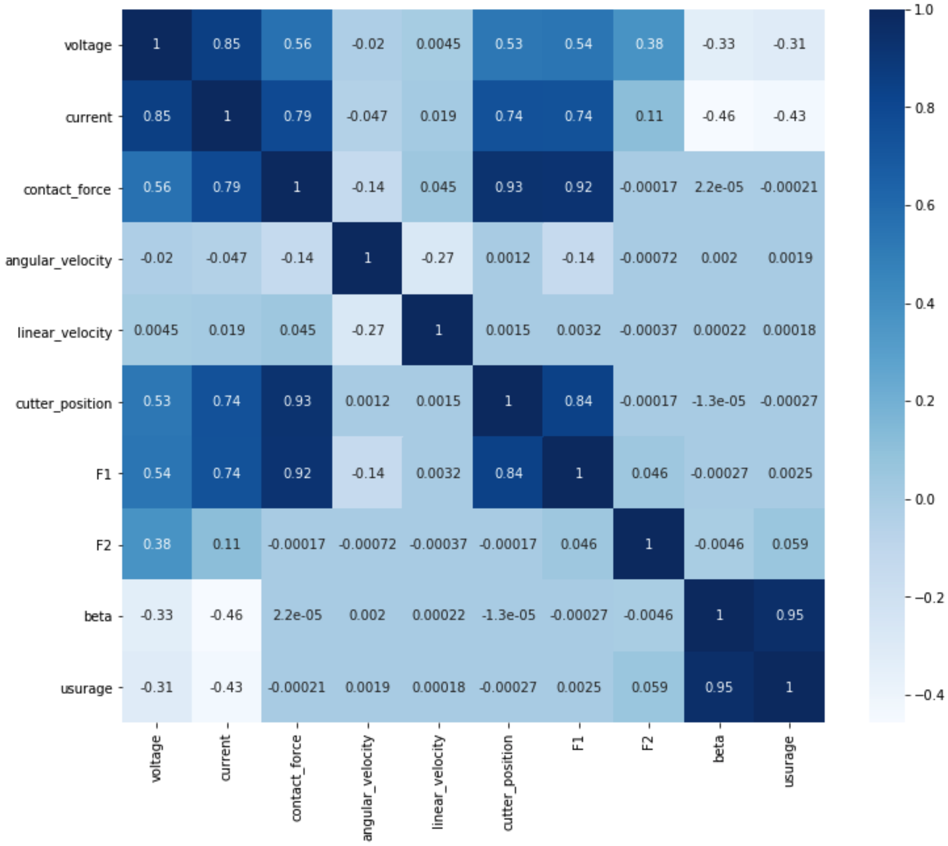


Figure 3.5: Heat map

3.3 Scatter plot

Scatter plot demonstrates the relation between the features of the dataset. While on the diagonal we have the histograms showing the values a variable can assume with the height depicting the frequency of that attribute.

In the figure, we can see a direct relation between voltage and current as well as between beta and usurage. The histogram reveals interesting facts such as contact force assuming only binary values. These results are congruent with the information given in the heat map. This plot is more insightful interms of the visualization of the instances projected on a 2 dimensional plane.

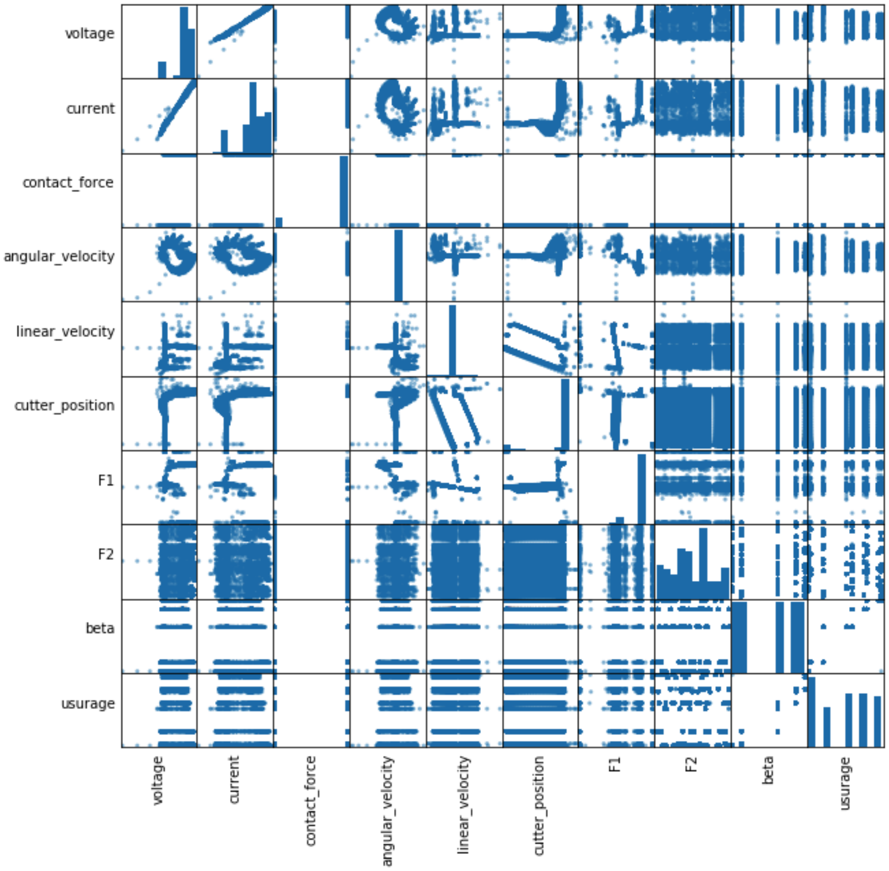


Figure 3.6: Scatter plot

3.4 Box plot

A boxplot is a standardized method of depicting data distributions using a five-number summary (”minimum”, first quartile (Q1), median, third quartile (Q3), and ”maximum”). It can give information about outliers and associated values. It can also determine if the data is symmetrical, how closely the data is packed, and whether or not the data is skewed. The following figure reveals boxplot for the features of the data set. It is clear from the figure that most of the variability lies in the features such as current, voltage and angular velocity. These features will play an important role in PCA as we will see in the later chapter.

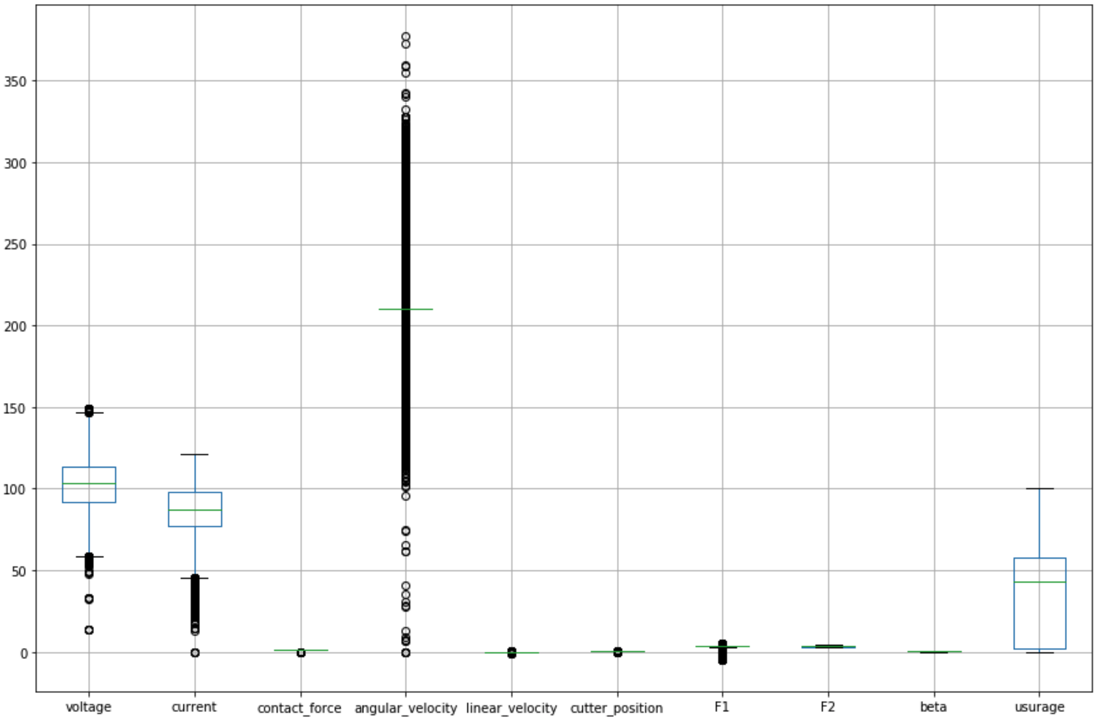


Figure 3.7: Box plot

3.5 KDE plot

The following kde plot reveals the distribution of the sample space for the labels of the dataset. One observation is that the plot is skewed towards the initial values. This indicates that even if the data was collected in equal intervals of time, the machine was estimated to have stable working conditions for the initial intervals of time and hence more samples are available relevant to those interval.

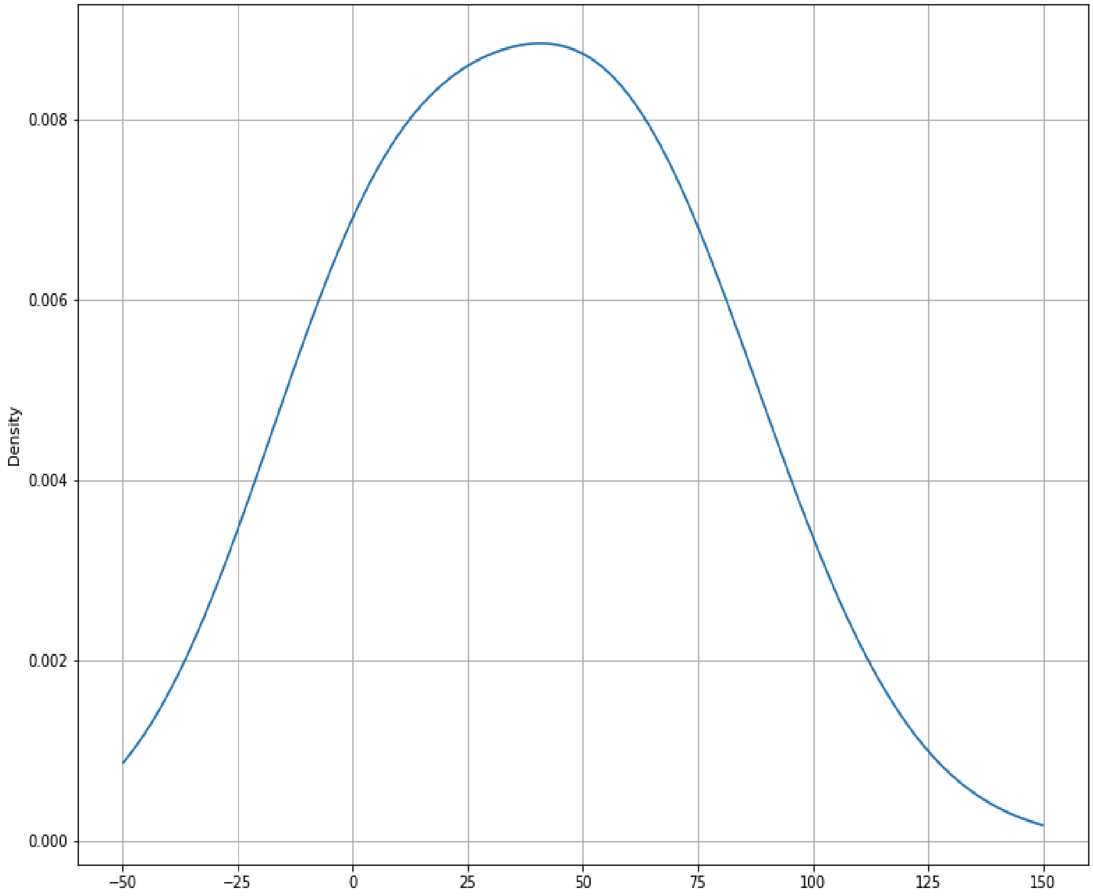


Figure 3.8: KDE plot

3.6 Moving average

We have observed previously that the features of the dataset are mostly periodic in nature. This can be easily related to the working of the CNC machine and of course the current and voltage signals are periodic in nature. Instead of feeding these featurs directly to the algorithm we could perform some pre conditioning on these features. The idea is to capture the long time trends in the signal. We must keep in mind that we should retain as much information contained in the feature as possible. Moving average can perform exactly this job. Essentially we are transforming our existing feature into a new useful feature. Thus we can think of moving average applied to a signal as a feature transformation. The figure below shows a moving average performed on the voltage contained in the dataset. Yellow signal being the original and green signal as a transformation.

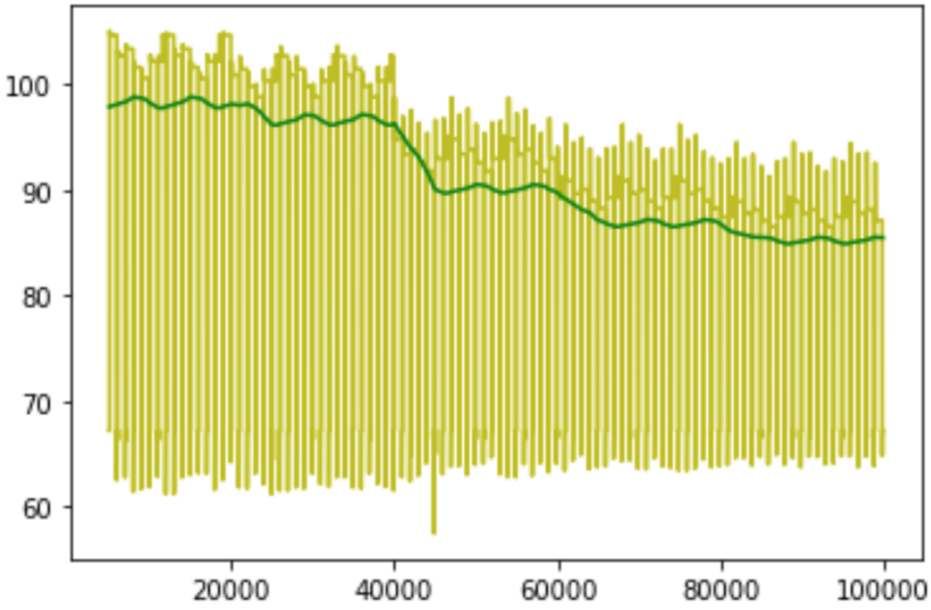


Figure 3.9: Moving average for voltage

3.7 Principal component analysis

In real world data analysis tasks we analyze complex data i.e. multi dimensional data. We plot the data and find various patterns in it or use it to train some machine learning models. One way to think about dimensions is that suppose we have a data point x , if we consider this data point as a physical object then dimensions are merely a basis of view, like where is the data located when it is observed from horizontal axis or vertical axis. As the dimensions of data increases, the difficulty to visualize it and perform computations on it also increases. So, how to reduce the dimensions of a data-

- Only keep the most important dimensions

PCA finds a new set of dimensions (or a set of basis of views) such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means more important principleaxis occurs first. (more important = more variance/more spread out data).

How does PCA work -

- Compute the vector μ containing mean of all the features in the dataset.
- Compute the centered data Matrix: $B = X - \mu$
- Calculate the covariance matrix of data: $S = \frac{1}{n}BB^T$
  Where S has a property of being symmetric and hence can be diagonalized. $S = PDP^T$
- Calculate the eigenvalues and eigenvectors over covariance matrix. Where we require that: $\lambda_1 \geq \lambda_2 .... \geq \lambda_n \geq 0$
- Choose the principal components according to given threshold of explained varience (e.g 0.9) or number of components to be kept.

3.7.1 Covariance

It is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction.

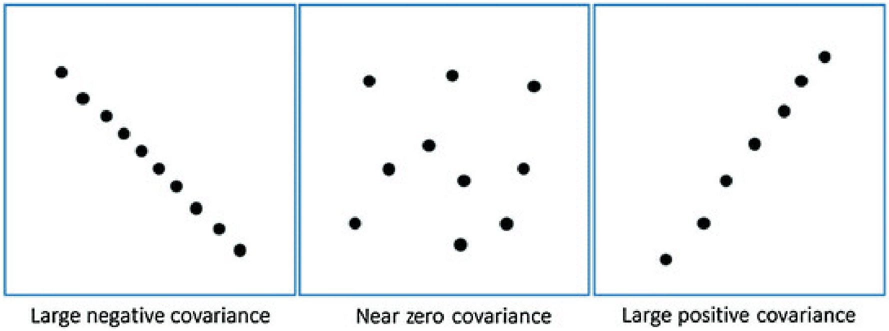


Figure 3.10: Covariance

Positive covariance means X and Y are positively related i.e. as X increases Y also increases. Negative covariance depicts the exact opposite relation. However zero covariance means X and Y are not related.

3.7.2 Eigenvectors and eigenvalues

The eigenvectors and eigenvalues of a covariance (or correlation) matrix represent the “core” of a PCA: The eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude. In other words, the eigenvalues explain the variance of the data along the new feature axes.

3.7.3 Dimensionality reduction

PCA is an unsupervised dimensionality reduction technique that focuses on capturing most of the variance in a datset. Usually mapping the data from a higher dimensional space to a lower one and thus reducing the dimensions. This mapping can be very useful since many algorithms in machine learning suffer from what is called ”curse of dimensionality”. Apart from this issue the algorithm can grow really complex and gives high latency during training and inference as well. The features obtained after applying this technique are independent. On the other hand mapping the data to a new set of feature has the issue of interpretability. It is not so obvious which features were important during the inference. We have various visualization techniques to describe these new feature and their relation with the original features of the dataset.

3.7.4 Explained variance

As PCA is meant to capture variance of the data. We can choose the set of features based on this parameter. Implementations mostly provide the choice to either choose the number of principal components we want to keep or the cumulative percentage of variance that these components must represent. So choosing the right number of components can be a hit and trial method until a certain threshold is not satisfied. The following figure shows explained variance of the principle components for the dataset. As an observation we can see that the first principle component explains the variance in the dataset upto 86% where as the second principle component explains variance upto 7% hence both of these component can explain upto 95% of the total variance which is remarkably good compared considering the dimensionality reduction from 9D to a 2D space.

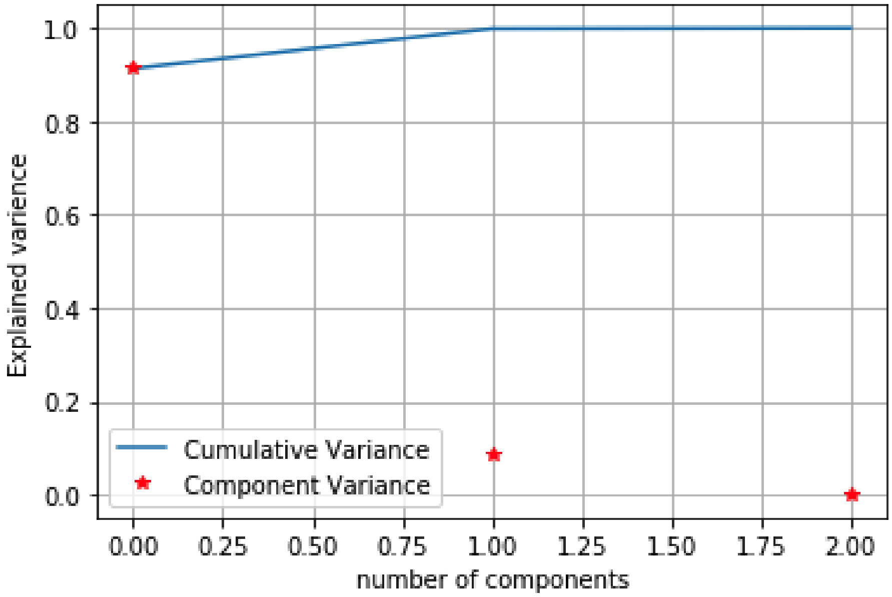


Figure 3.11: Principle component analysis (PCA)

3.7.5 Visualizing PCA

The visualisation of the principle components provides assistance in understanding the contribution of the original features. This alleviates the problem of interpretability to some extent. The following bar graph shows the relation between the original features and the principle components. The contributing features were expected this way since the velocity and current as well as angular velocity had a wide range of values, these feature play a dominant role in determining the new axis for mapping the dataset.

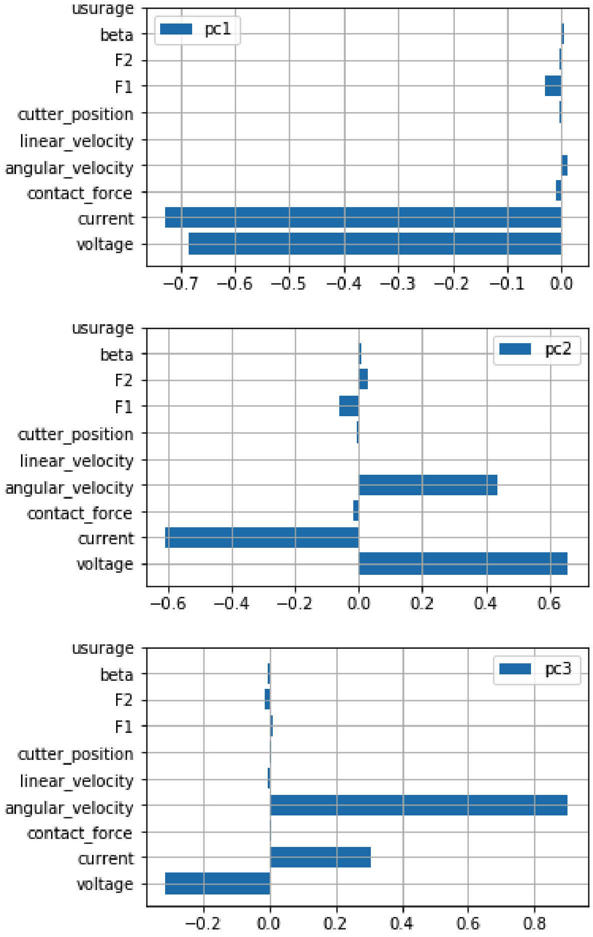


Figure 3.12: Principle components

3.7.6 Scatter plot

The application of PCA technique, allows a mapping to new feature space possibly in lower dimensional space. The data set can be visualised by mapping the original dataset onto the new axis. Figure below represents the transformation of some instances on the first two principle components. This figure is similar to the correlation figure between current and angular velocity represented in scatter matrix since these two feature played dominating role in determining the new axis.

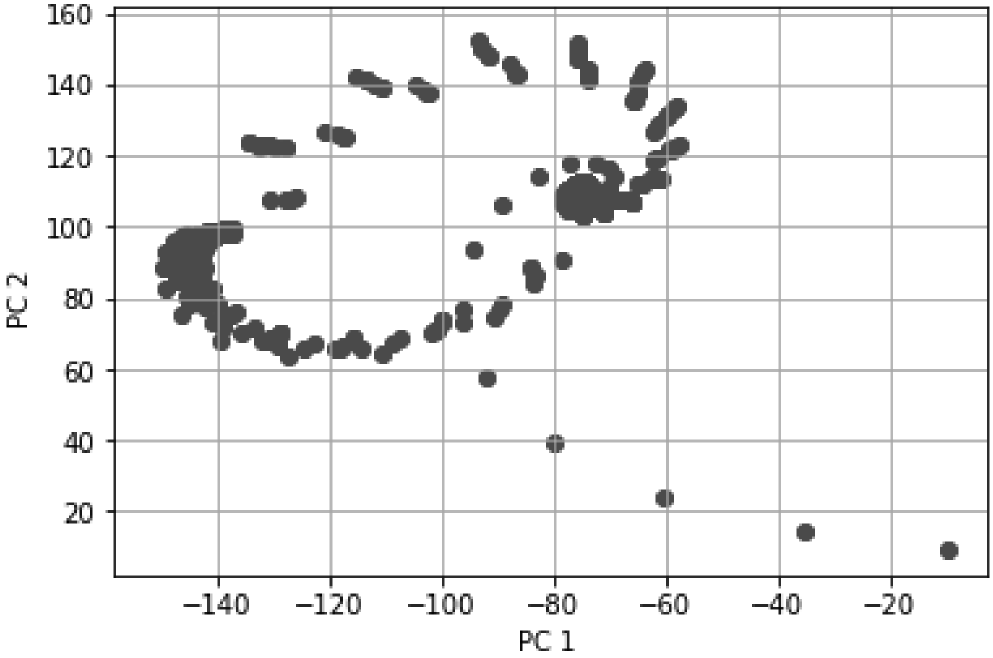


Figure 3.13: scatter plot: PC1 vs PC2

Chapter 4

Data pre-processing

4.1 Spark dataframe

A DataFrame is a named-column Dataset. It's similar to a table in a relational database or a data frame in R/Python in terms of principle, but it has more advanced optimizations. Structured data files, Hive tables, external databases, and existing RDDs are all examples of sources for DataFrames. Scala, Java, Python, and R are supported by the DataFrame API. A Dataset of Rows represents a DataFrame in Scala and Java. DataFrame is just another name for Dataset[Row] in the Scala API.

4.2 R formula

R formula model Fits a dataset to a R model formula by implementing the necessary transformations.RFormula generates a features vector column and a label double or string column. String input columns will be one-hot encoded, and numeric columns will be converted to doubles, same like when formulae are used in R for linear regression. If the label column is a string, it will be converted to a double using StringIndexer first. The output label column will be produced from the provided response variable in the formula if the label column is not present in the DataFrame.

4.3 Feature scaling

One of the most important phases in the pre-processing of data prior to developing a machine learning model is feature scaling. Scaling may make the difference between a weak and a better machine learning model.

Normalization and Standardization are the most frequent feature scaling approaches. When we wish to limit our data to a range between two integers, such as [0,1] or [-1,1], we utilize normalization. Standardization makes our data unitless by transforming it to have a zero mean and a variance of 1.

4.3.1 Why feature scaling

Machine learning algorithms just look at numbers, and if there is a significant difference in range, such as a few ranging in the thousands against a few ranging in the tens, it assumes that greater ranging numbers have some form of superiority. As a result, these larger numbers begin to play a larger part in the model's training.

Another rationale for using feature scaling is that some algorithms, such as Neural network gradient descent, converge more quicker with it than without it. One more reason is saturation, which may be avoided via scaling, as in the case of sigmoid activation in Neural Networks.

4.3.2 When to perform feature scaling

Machine learning methods that determine distances between data require feature scaling. When computing distances, if the feature with a higher value range is not scaled, the feature with a higher value range takes precedence, as discussed intuitively in the ”why?” section. The ML method is sensitive to ”relative scales of features,” which occurs when the features' numeric values are used instead of their rank. Scaling is a must in many algorithms that need quicker convergence, such as Neural Networks.

Due to the broad range of values in raw data, objective functions in some machine learning algorithms do not perform correctly without normalization. The majority of classifiers, for example, use the distance to compute the distance between two locations. If one of the characteristics has a large range of values, it is governed by the distance. As a result, all features' ranges should be normalized such that each contributes about equally to the final distance.

Even if the requirements outlined above are not met, we may need to rescale your features if the machine learning algorithm anticipates some scale or a saturation issue. Another suitable example is a neural network with saturating activation functions (e.g., sigmoid).

- Algorithms that involve measuring distance are sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
- When doing Principal Component Analysis, scaling is crucial (PCA). PCA aims to extract the features with the most volatility, and high magnitude features have a lot of variance, which skews the PCA towards high magnitude features.
- We can accelerate gradient descent by scaling, Because θ drops fast on short ranges and slowly on wide ranges, and oscillates inefficiently down to the optimum when the variables are highly unequal.

4.3.3 When not to perform feature scaling

Rules-based algorithms are those that do not require normalization or scaling. Any monotonic modifications of the variables would have no effect on them. A monotonic transition is scaling. All tree-based algorithms, such as CART, Random Forests, and Gradient Boosted Decision Trees, fall within this group. These methods do not need normalization and rely on rules (series of inequalities). Algorithms such as Linear Discriminant Analysis (LDA) and Naive Bayes are built to deal with this and assign weights to the features appropriately. In these algorithms, doing features scaling may not have much of an impact.

4.3.4 Scalers available in spark

Min-Max scaler

Min-max scaler scales each feature to a certain range to transform it. This estimator scales and translates each feature independently such that it falls inside the training set's predefined range, such as zero to one. If there are negative values, this Scaler compresses the data to a range of -1 to 1. The range can be specified to [0,1], [0,5], or [-1,1]. When the standard deviation is modest and the distribution is not Gaussian, this Scaler performs well. Outliers are sensitive to this Scaler.

Standard scaler

The Standard Scaler considers that data inside each feature is normally distributed and scales it so that the distribution is centered around 0 with a standard deviation of 1. By computing the necessary statistics on the samples in the training set, each feature is individually centered and scaled. This is not the ideal Scaler to use if the data is not regularly distributed.

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{4.1}$$

Normalizer

The whole feature vector is scaled as though it were of unit length. This generally entails dividing each component by the vector's Euclidean length (L2 Norm). The L1 norm of the feature vector may be more practical in particular applications (e.g., histogram features). The Unit Vector method, like Min-Max Scaling, creates values in the range [0,1]. This is quite handy when working with features that have strict bounds.

$$x' = \frac{x}{||x||} \tag{4.2}$$

Robust Scaler

In case of robust scaler median is removed, and the data is scaled according to the quantile range (defaults to IQR: Interquartile Range). The interquartile range (IQR) is the distance between the first and third quartiles (25th and 3rd quantiles) (75th quantile). Because this Scaler's centering and scaling statistics are based on percentiles, they are unaffected by a few large marginal outliers. It's worth noting that the outliers remain in the modified data. A non-linear modification is necessary if distinct outlier clipping is desired.

This Scaler is resilient against outliers, as the name implies. The mean and standard deviation of the data will not scale well if our data contains several outliers.

Max abs scaler

Scales each feature to its absolute maximum value. This estimator scales and transforms each feature independently so that the training set's maximum absolute value for each feature is 1.0. It does not relocate or center the data,

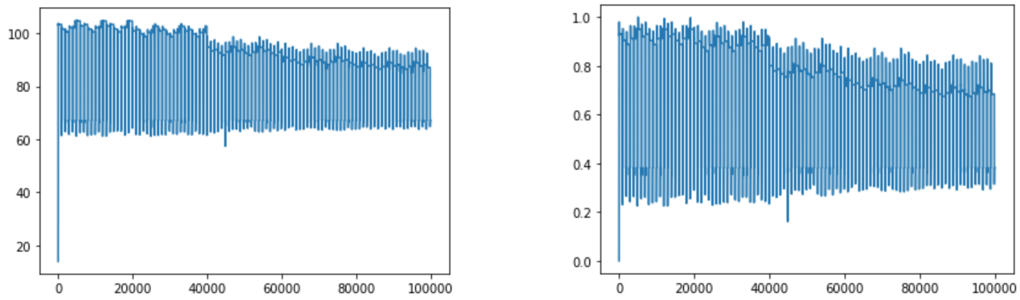


Figure 4.1: min-max scaling for the voltage

therefore there is no loss of sparsity. This Scaler acts similarly to Min Max Scaler on positive-only data and, as a result, suffers from severe outliers.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

We have chosen to use min max scaler to scale the features of the dataset. In this way correponding values are with in the range of [0,1]. As discusses before, many algorithm may show biasness towards an attribute based on wide range of values, by perform feature scaling we can alleviate this kind of biasness towards any particular feature.

4.4 Pipeline

In machine learning, it is common to run a sequence of algorithms to process and learn from data. For example in this case we require to perform the following tasks on the dataset:

- Apply formula model to the train dataset thus extracting a feature and label columns.
- Apply feature scaling so that all the values in the feature column are in the same range.
- Apply the machine learning model to learn from the data.

A Pipeline, which consists of a sequence of PipelineStages (Transformers and Estimators) to be performed in a certain order, is how MLlib expresses such a process.
A pipeline is defined as a series of steps, each of which is a Transformer or an Estimator. The input DataFrame is modified as it passes through each stage, which is done in sequence. The transform() method on the DataFrame is used for Transformer phases. The fit() method is used to create a Transformer (which becomes part of the PipelineModel, or fitted Pipeline), and the transform() method of that Transformer is used on the DataFrame.A Pipeline acts as an Estimator. Thus, after a Pipeline's fit() method runs, it produces a PipelineModel, which is a Transformer. This PipelineModel is used at test time in order to make predictions.
The PipelineModel has the same number of stages as the original Pipeline, but all of the Estimators have been transformed into Transformers. The data are fed through the fitted pipeline in sequence when the PipelineModel's transform() function is performed on a test dataset. The transform() method of each step modifies the dataset before passing it on to the next. Pipelines and PipelineModels, as a result, assist in ensuring that training and test data undergo the same feature processing stages. The following figure shows the working for the pipeline used:

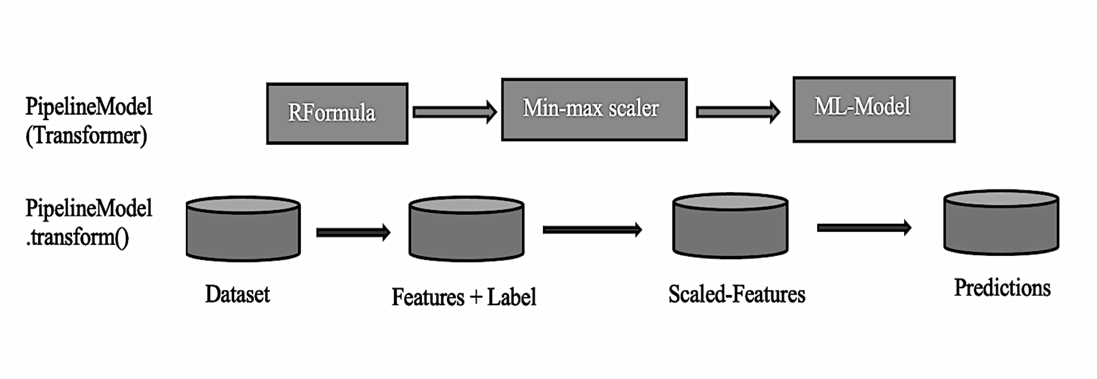


Figure 4.2: Model pipeline

Chapter 5

Machine learning techniques

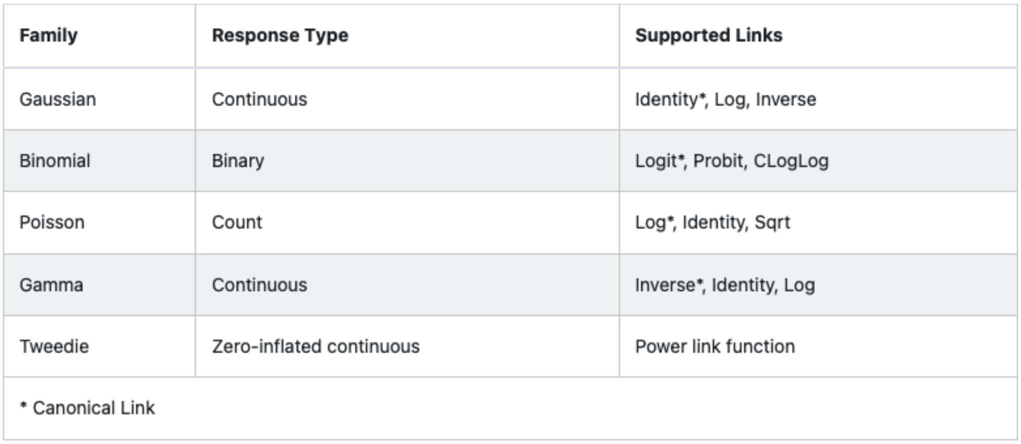
5.1 Problem definition

Continuing our analysis with the data set extracted from the Simulated model of the CNC machine. We would like to explore the possible data analytic techniques that we can apply in the Big data environment. The idea is to explore the possible options available in spark for the analysis. As for the SOH machine is concerned, we can say that the variable to be predicted is continuos in nature following a normal distribution as shown in the exploratory data analysis phase. In this regard, the problem at hand can be posed as a regression problem. So we would like to implement and compare the results of various regression techniques provided by apache spark. We will explore the following regression techniques:

- Generalised linear regression (Linear regression as a case).
- Decision tree.
- Gradient boosted tree.
- Random forest.
- Survival regression.

5.2 Generalised linear regression

GLMs (generalized linear models) are linear model formulations in which the response variable Y_i follows one of the exponential family of distributions. The Generalized Linear Regression interface in Spark enables for flexible GLM specification, which may be used for a variety of prediction issues such as linear regression, Poisson regression, logistic regression, and more. Only a subset of the exponential family distributions are supported in spark.ml until now, and they are given below. GLMs need natural exponential family distributions, which are exponential family distributions that may be represented in their ”canonical” or ”natural” form. A natural exponential family distribution has the following form:



Family	Response Type	Supported Links
Gaussian	Continuous	Identity*, Log, Inverse
Binomial	Binary	Logit*, Probit, CLogLog
Poisson	Count	Log*, Identity, Sqrt
Gamma	Continuous	Inverse*, Identity, Log
Tweedie	Zero-inflated continuous	Power link function
* Canonical Link		

Figure 5.1: Family of GLM in spark

$$f_y(y|\theta,\tau) = h(y,\tau)exp(\frac{\theta y - A(\theta)}{d\tau}) \tag{5.1}$$

where θ is the parameter of interest and τ is a dispersion parameter. In a GLM the response variable Y_i is assumed to be drawn from a natural exponential family distribution:

$$Y_i \approx f(.|\theta_i,\tau) \tag{5.2}$$

θ_i can be related to the response variable μ by:

$$\mu_i = A'(\theta_i) \tag{5.3}$$

The shape of the chosen distribution is used to define $A'(\theta_i)$. GLMs additionally enable you to provide a link function, which describes the connection between the response variable μ_i predicted value and the so-called linear predictor η:

$$g(\mu_i) = \eta_i = x_i^T.\beta \tag{5.4}$$

$A = g^{-1}$, which offers a simpler connection between the parameter of interest and the linear predictor, is a common choice for the link function. The link function $g(\mu)$ is considered to be the "canonical" link function in this case.

$$\theta_i = A'^{-1}(\mu_i) = g(g^{-1}(\eta)_i)) = \eta_i \tag{5.5}$$

A GLM finds the regression coefficients β which maximize the likelihood function described as follwos:

$$max_\beta L(\theta|y, X) = \prod_{i=1}^{N} h(y_i, \tau) exp(\frac{y_i\theta_i - A(\theta_i)}{d\tau}) \tag{5.6}$$

and the parameter of interest θ_i is related to the regression coefficients β by:

$$\theta_i = A'^{(} - 1)(g^{(} - 1)(x_i, \beta) \tag{5.7}$$

Spark's generalized linear regression interface also provides summary statistics for diagnosing the fit of GLM models, including residuals, p-values, deviances, the Akaike information criterion, and others.

Note

Spark's Generalized Linear Regression interface currently supports up to 4096 features and will throw an exception if this limit is exceeded.However Models with a larger number of features may still be trained using the Linear Regression and Logistic Regression estimators for linear and logistic regression. In our case, the feature are within the limit and hence we can confidently apply the Generalized Linear Regression Model

5.2.1 Linear regression

We may study Linear regression as a specific case of generalized linear regression where the distribution of the output follows a Gaussian distribution (i. a bell shaped curve). We can define the linear regression problem as follows:

A linear model is one in which the input variables (x) and the single output variable (y) are assumed to have a linear relationship (y). That y can be determined using a linear combination of the input variables is more detailed (x).

The method is known as simple linear regression when there is just one input variable (x). When there are several input variables, the technique is referred to as multiple linear regression in statistics literature.

To construct or train the linear regression equation using data, many strategies may be utilized, the most popular of which is termed Ordinary Least Squares. Ordinary Least Squares Linear Regression, or simply Least Squares Regression, is a term used to describe a model created in this manner.

5.2.2 Model representation

The representation is a linear equation that combines a collection of input values (x), with the solution being the projected output for that set of input values (y). As a result, both the input (x) and output (y) values are numeric.

Each input value or column is assigned one scale factor, referred to as a coefficient and denoted by the capital Greek letter Beta in the linear equation (β). One more coefficient is added, which gives the line an extra degree of freedom (for example, going up and down on a two-dimensional plot) and is known as the intercept or bias coefficient.

In a simple regression issue (with only one x and one y), the model would have the following form:

$$Y = \beta_0 + \beta_1.X \tag{5.8}$$

When there are several inputs (x) in higher dimensions, the line is termed a plane or a hyper-plane. As a result, the representation is the equation's form as well as the coefficients' precise values (e.g. B0 and B1 in the above example).

The complexity of a regression model, such as linear regression, is frequently discussed. This refers to the number of coefficients used in the model.

When a coefficient hits zero, it essentially removes the input variable's impact on the model and, as a result, the model's prediction (0 * x = 0).This is important to consider when discussing regularization strategies, which alter the learning process to minimize the complexity of regression models by exerting pressure on the absolute magnitude of the coefficients and pushing some to zero.

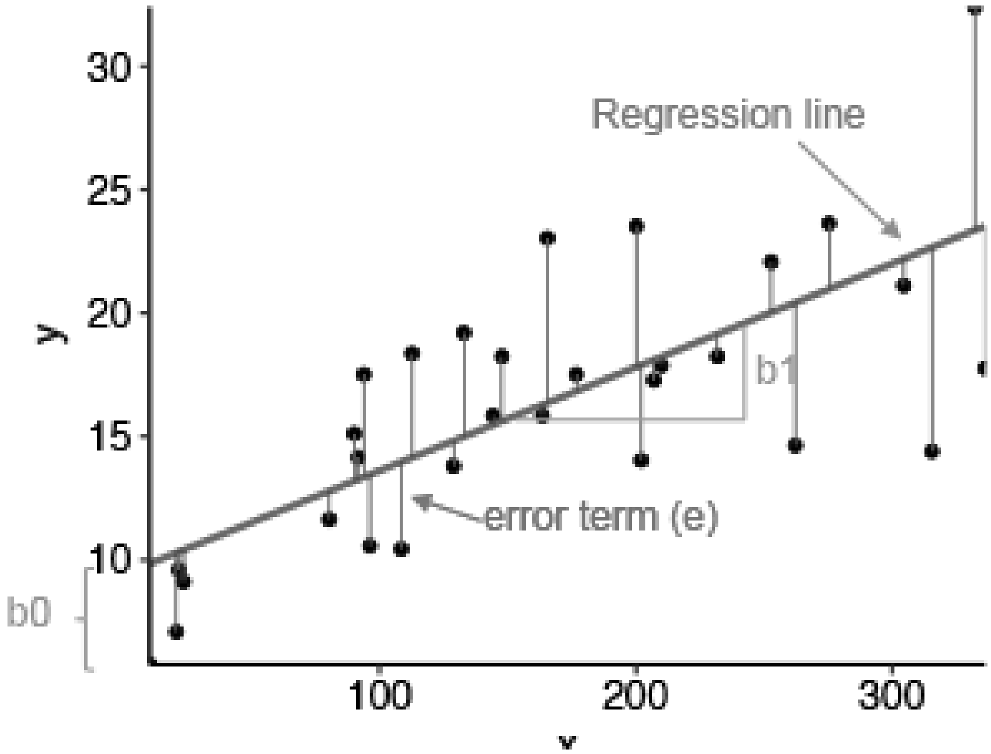


Figure 5.2: Linear Regression

5.2.3 Methods for learning linear regression model

Linear regression technique has been around for more than a century and therefor the technique has been well studied. Hence there are many methods for learning the model. We shall discuss them in this section:

Simple linear regression

When using basic linear regression with a single input, statistics may be used to estimate the coefficients. This necessitates the computation of statistical features such as means, standard deviations, correlations, and covariance from the data. To traverse and calculate statistics, all of the data must be available. Given more attributes present in our dataset, we need a more sophisticated technique for the analysis.

Ordinary least squares

When there are several inputs, we may utilize Ordinary Least Squares to estimate the coefficient values.

The sum of the squared residuals is minimized using the Ordinary Least Squares approach. This implies that given a regression line across the data, we square the distance between each data point and the regression line, then add all of the squared errors together. Ordinary least squares attempts to minimize this amount.

This method considers the data as a matrix and estimates the best coefficient values using linear algebra techniques. It implies access to all of the data and sufficient RAM to fit the data and conduct matrix operations.

Gradient descent

When there are one or more inputs, we may utilize an iterative method to optimize the coefficient values by reducing the model's error on the training data.

This method is known as gradient descent and it operates by starting with random values for each coefficient. For each pair of input and output values, the total of squared errors is determined. As a scale factor, a learning rate is utilized, and the coefficients are updated in the direction of minimizing the error. The procedure is continued until the sum squared error is reduced to a minimum or no more improvement is achievable. It allows choosing a learning rate (alpha) parameter that sets the magnitude of the improvement step to take on each iteration of the procedure while utilizing this approach.Because it is very simple to grasp, a linear regression model is frequently used to apply gradient descent. It's handy in practice when having a large dataset, either in terms of the number of rows or the number of columns, that won't fit in memory.

Regularization

Regularization methods are extensions of the linear model training process. These aim to lower the model's complexity while minimizing the model's sum of squared errors on the training data (using ordinary least squares) (like the number or absolute size of the sum of all coefficients in the model).

popular examples of regularization procedures for linear regression are:

- **Lasso regression:** where the Ordinary Least Squares method is adjusted to reduce the absolute total of the coefficients as well (called L1 regularization).

- **Ridge regression:** Ordinary Least Squares is modified to minimize the squared absolute sum of the coefficients in Ridge Regression (called L2 regularization).

5.2.4 Assumption on data for applying linear regression

To effectively apply this techniques we need to make sure thata the following conditions are evaluated during the preliminary analysis:

- **Linearity** The relation between the input and output is assumed to be linear in linear regression. Satisfying this condition may require performing logrithmic transformation to the non linear features of the dataset.

- **Noise removal** The input features and output variables are assumed to be noise-free in linear regression. This assumption can be satisfied by eliminating outliers. Visualizing the features using a boxplot help spotting the outliers as discussed in the chapter before.

- **Collinearity** The algorithm tends to overfit on data if the features involved are dependent. This phenomina can be spotted using heatmaps or scatter plots. The feature collinearity can be eliminated by feature elimination or transformations such as Principle Component Analysis (PCA).

- **Gaussian distribution** This technique is more effective if the data distribution follows gaussian distribution.

- **Data scaling** For better performance, Data scaling is required so that all the feaures are in the same range of values.

5.3 Decision tree

Decision tree is a simple machine learning algorithm. The objective is to learn basic decision rules from data characteristics to construct a model that predicts the value of a target variable. Decision Tree can be used both in classification and regression problem

5.3.1 Important terminology

Before diving into details, we would like to remind some basic terminology used when describing a decision tree.

- **Root Node:** Points to the entire population or sample and further gets divided into two or more homogeneous sets down the tree.
- **Splitting:** The process of dividing a node into two or more sub-nodes. Usually we have to follow some splitting criteria that we will discuss later.
- **Decision node:** A node that provides link to further sub nodes.
- **Leaf/Terminal node:** Node at the end of the tree and has can be split anymore.
- **Pruning:** The process of eliminating some nodes or branching.
- **Branch/Sub-Tree** A subsection of the tree is referred to as a branch or sub tree.
- **Parent and child node:** A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

5.3.2 Working principle

A decision tree generates an estimate by asking the data a set of questions, each of which narrows the range of potential answers until the model is confident enough to produce a single forecast. The model determines the

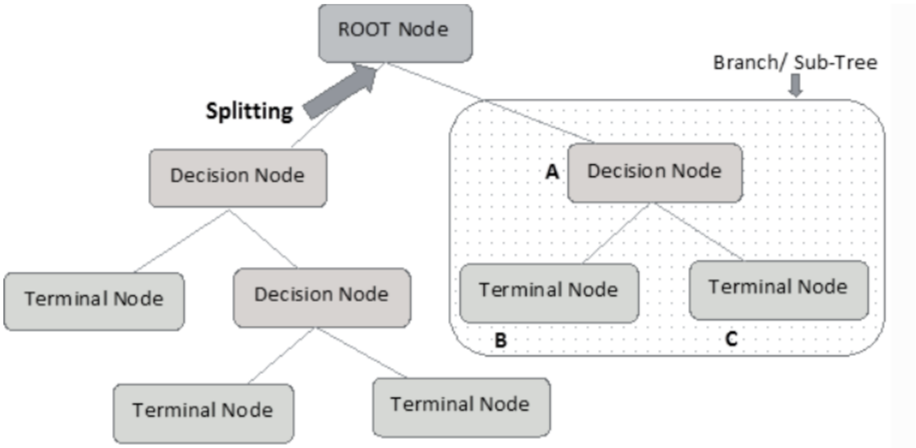


Figure 5.3: Decisioin tree: example

sequence of the questions as well as their substance. Furthermore, all of the questions are True/False in nature.

This is a bit difficult to comprehend since it is not how humans think naturally, and the simplest way to demonstrate this difference is to design a true decision tree from scratch. x1 and x2 are two aspects in the preceding issue that allow us to generate predictions for the target variable y by asking True/False questions.
There are different branches for each True and False response. We get to a forecast regardless of the answers to the questions (leaf node). Begin at the top, at the root node, and work your way down the tree, answering the questions as you go. As a result, any pair of X1 and X2 can be used.

We should highlight one feature of the decision tree: how it learns (how the 'questions' are produced and how the thresholds are set). In the training phase of model construction, a decision tree learns to map data to outputs as a supervised machine learning model. During training, all past data relevant to the issue domain and the real value we want the model to learn to predict is fed into the model. Any associations between the data and the target variable are learned by the model.

Following the training phase, the decision tree generates a tree similar to the one shown in the figure below, determining the best questions to ask as well as the sequence in which they should be asked in order to create the most accurate predictions possible. When we want to create a forecast, we should provide the model the same data format in order to produce a prediction. The forecast will be a guess based on the train data on which it was trained.

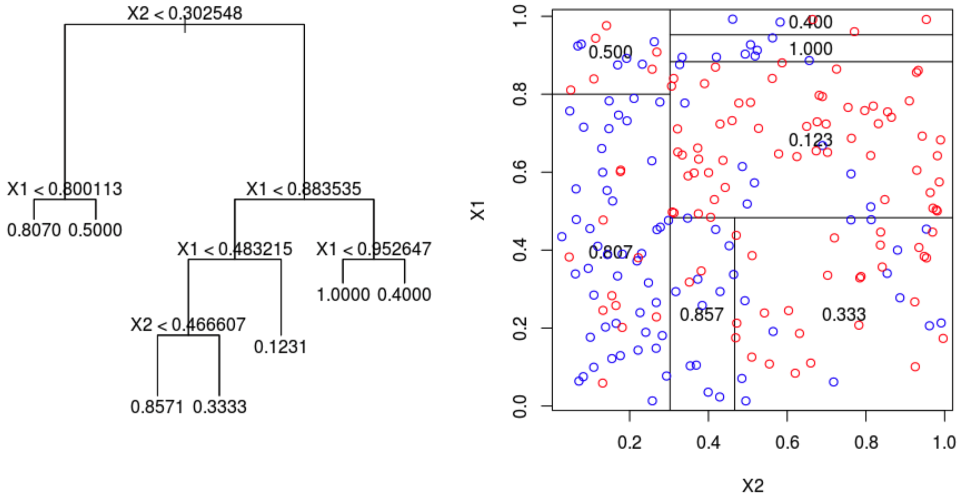


Figure 5.4: Decisioin tree: example(2)

5.3.3 Splitting criteria

The major question here is : How is the splitting decided for a decision tree. In fact the decision to make tactical splits has a significant impact on a tree's accuracy. The decision criteria for classification and regression trees are different. The mean squared error (MSE) is commonly used in decision trees regression to determine if a node should be divided into two or more sub-nodes. If we're making a binary tree, the algorithm will first choose a value and divide the data into two subsets. It will compute the MSE independently for each subgroup. The tree selects the value that produces the lowest MSE value. Let's take a closer look at how Splitting for Decision tree regressor is decided. The first stage in building a tree is to make a binary decision. How are we going to go about doing it?

- We need to choose a variable and a value to divide on such that the two groups are as dissimilar as feasible.
- for every variable, see if the value of the split for that variable is better.
- How can you tell whether it's better? (mse*num samples) take the weighted average of two new nodes.

So to summarize we can say that:

- The weighted average of the mean squared errors of the two groups that make up a split is a single value that shows how good a split is.
- A method for determining the best split is to test each variable and every potential value of that variable to determine which variable and value produces the best split.

At each step, choice of the best predictor to split is made according to some criteria. These measures represent the purity of a split. The most common ones are:

- Gini index: $1 - \sum_{i=1}^{n}(P_i)^2$
- Cross Entropy: $-\sum_{i=1}^{n} P_i log_2(P_i)$

Where Pi denotes the probability of an element being classified for a distinct class. Here is a visual example of gini.

5.3.4 Advantages

- Decision tree may be used to solve problems in both classification and regression.
- Simple to comprehend, interpret, and visualize When it comes to data exploration. One of the quickest ways to determine the most important factors and relationships between two or more variables is to use a decision tree for instance feature importance.
- We may use decision trees to develop additional variables / features that have a higher predictive potential for the target variable. This algorithm may also be useful during the data investigation stage.

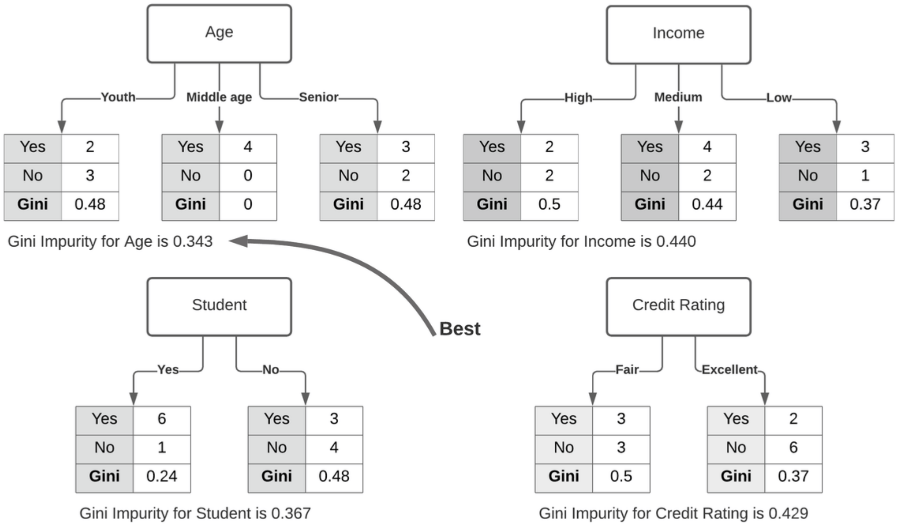


Figure 5.5: Splitting criteria

- Data preparation isn't as time-consuming as it formerly was: It is relatively unaffected by outliers and missing values.

- Decision tree can handle both numerical and categorical variables, thus data type isn't an issue.

5.3.5 Disadvantages

- Decision-tree learners can create over-complex trees that do not generalise the data well thus creating the problem of overfitting.

- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

- Decision-tree learning algorithms are based on heuristics such as the greedy method where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.

- Small differences in the data might result in an entirely different tree being produced. This is known as variance, and it may be reduced using techniques such as bagging and boosting.

5.3.6 Feature importance

A very useful aspect of decision tree is the interpretability of this algorithm. The implementation of this algorithm allows us to seek what is called the feature importance. In which case, the decrease in node impurity is weighted by the likelihood of accessing that node to compute feature importance. The number of samples that reach the node divided by the total number of samples is the node probability. The more significant the characteristic, the more is the importance of the feature. The following figure shows the feature importance for the variables used in the given dataset.

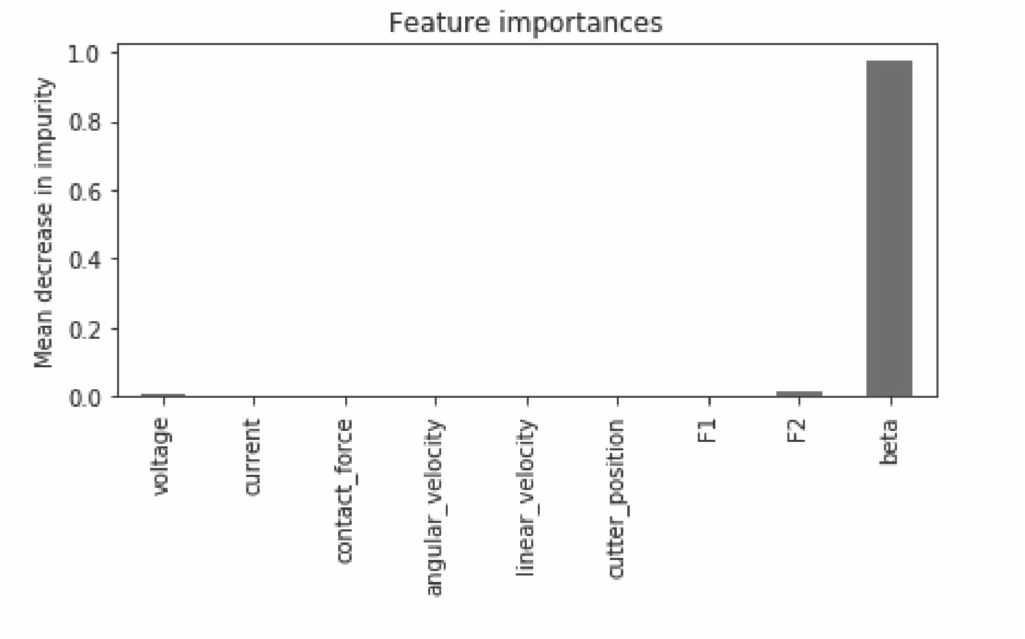
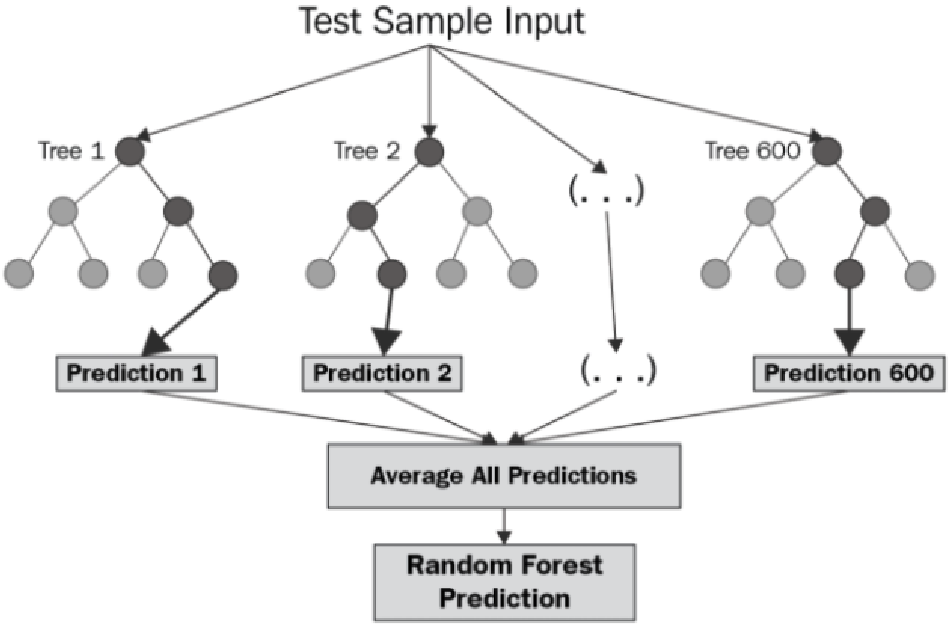


Figure 5.6: Feature importance

The figure reveals that the variable beta is the most significant feature for determining the usurage while F2 and voltage are important as well. This was as aspected by looking at the correlations fro the heatmap in the preprocessing phase.

5.4 Random forest

Random forest classifier belong to the ensemble class. Several decision trees are trained, each of them being trained on a different set of bootstrapped training data. At each split, only m predictors participate (usually $\sqrt{p}$). This splitting results in uncorrelated trees and hence obtaining a more robust model. Spark supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. Spark implements random forests using the existing decision tree implementation.



5.5 Ensemble learning

Ensemble learning is the technique of combining different models that have been trained on the same data and average their findings to provide a more powerful regression/classification result. The goal of ensemble learning is for the mistakes of each model (in this example, the decision tree) to be independent and varied from tree to tree.

5.5.1 Bootstrap aggregation

The Random Forest algorithm uses Bootstrap aggregating, also called bagging, as its ensembling method. It gains accuracy and combats overfitting by not only averaging the models but also trying to create models that are as uncorrelated as possible by giving them different training data sets. It creates the data-sets using sampling with replacement a straightforward but sufficient data sampling technique. Sampling with replacement means that some data-points can be picked multiple times.To further decrease the correlation between individual trees, each decision tree is trained on different randomly selected features. The number of features used for each individual tree is a hyperparameter, often called *max_features or n_features.*

So each decision tree in a random forest is not only trained on a different data-set (thanks to bagging) but also on different features/columns. After the individual decision trees are trained, they are combined together. For classification, max voting is used. That means the class, which most trees have as the output, will be the Random Forest's output. For regression, the outputs of the Decision Trees are averaged.

5.5.2 Hyper parameters

Random forests have several hyper parameters and it is important to understand their role in order to tune the model.

- **numTrees** The number of trees in the forest.
  - Increasing the number of trees will decrease the variance in predictions, improving the model's test-time accuracy.
  - Training time increases roughly linearly in the number of trees.
- **maxDepth:** The maximum depth of a tree.
  - Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting.
  - In general, it is acceptable to train deeper trees when using random forests than when using a single decision tree. One tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest).

- **subsamplingRate:** This parameter specifies the size of the dataset used for training each tree in the forest, as a fraction of the size of the original dataset. The default (1.0) is recommended, but decreasing this fraction can speed up training.

- **featureSubsetStrategy:**At each tree node, the number of features to utilize as candidates for splitting. The number is expressed as a percentage or as a function of the total number of features. This value can be reduced to speed up training, but if it is too low, it can have an influence on performance.

5.6 Gradient boosted trees

Gradient boosted trees (GBTs) are decision tree ensembles. GBTs train decision trees iteratively in order to minimize a loss function. GBTs, like decision trees, handle categorical features, can handle multiclass classification as well as regression, don't need feature scaling, and can capture non-linearities and feature interactions.

GBTs are supported by spark for binary classification and regression, and they may use both continuous and categorical information. The decision tree implementation in spark is used to implement GBTs.

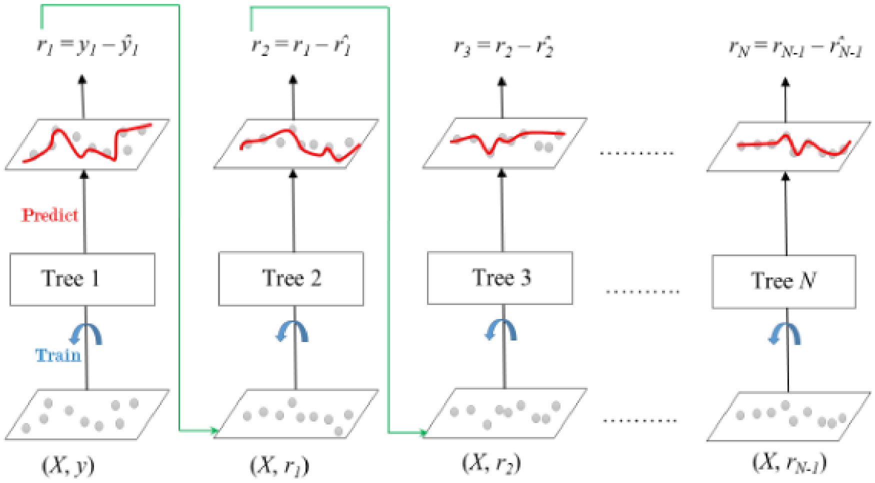


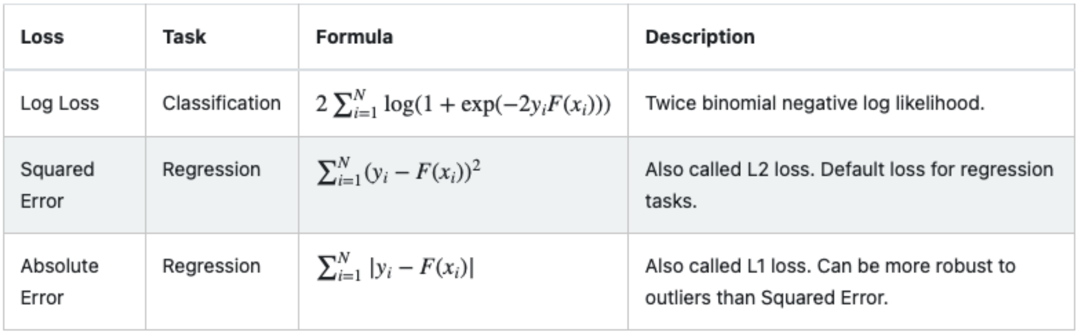
Figure 5.7: Gradient Boosting Tree

5.6.1 Algorithm working

Gradient boosting is a technique for repeatedly training a series of decision trees. The method utilizes the current ensemble to forecast the label of each training instance on each iteration, and then compares the prediction to the real label. The dataset has been relabeled to place a greater emphasis on training cases with poor predictions. As a result, the decision tree will assist in correcting earlier errors in the following iteration.

5.6.2 Loss functions

A loss function defines the specific technique for re-labeling instances. GBTs lower the loss function on the training data with each iteration. The following table shows the losses that can be used for training gradient boosted trees.



Loss	Task	Formula	Description
Log Loss	Classification	$2 \sum_{i=1}^{N} \log(1 + \exp(-2y_i F(x_i)))$	Twice binomial negative log likelihood.
Squared Error	Regression	$\sum_{i=1}^{N} (y_i - F(x_i))^2$	Also called L2 loss. Default loss for regression tasks.
Absolute Error	Regression	$\sum_{i=1}^{N} \lvert y_i - F(x_i) \rvert$	Also called L1 loss. Can be more robust to outliers than Squared Error.

Figure 5.8: Loss function

5.6.3 Validation while training

Validation during training is beneficial in preventing overfitting. To make advantage of this option, the method runWithValidation has been supplied. It accepts two RDDs as parameters, the first of which is the training dataset and the second of which is the validation dataset.

When the improvement in the validation error is less than a specific tolerance, the training is discontinued (supplied by the validationTol argument in BoostingStrategy). In practice, the validation error lowers at first, then rises afterwards.If the validation error does not change monotonically, the user should specify a big negative tolerance and check the validation curve using evaluateEachIteration (which returns the error or loss per iteration) to fine-tune the number of iterations.

5.7 Gradient boosted trees vs. Random forests

Both Gradient boosted trees (GBTs) and Random Forests are algorithms for learning ensembles of trees, but the training processes are different. There are several practical trade-offs:

- GBTs train one tree at a time, taking longer than random forests to complete. Random Forests may simultaneously train several trees. GBTs, on the other hand, generally allow for the use of smaller (thinner) trees than Random Forests, and training smaller trees takes less time.
- Over fitting is less likely with Random Forests. Over fitting is less likely when more trees are trained in a Random Forest, but it is more likely when more trees are trained with GBTs. (In statistical terms, Random Forests use more trees to minimize variance, whereas GBTs use more trees to reduce bias.)
- Because performance improves monotonically with the number of trees, Random Forests may be easier to adjust (whereas performance can start to decrease for GBTs if the number of trees grows too large).

5.8 Survival regression

The accelerated failure time (AFT) model is a censored data parametric survival regression model. It's also known as a log-linear model for survival analysis since it presents a model for the log of survival time. Because each instance contributes to the objective function individually, the AFT paradigm is easy to parallelize.
Given the values of the covariates x‘, for random lifetime ti of subjects i = 1, . . . , n, with possible right-censoring, the likelihood function under the AFT model is given as:

$$L(\beta,\sigma)=\prod_{i=1}^{n}[\frac{1}{\sigma}f_0(\frac{logt_i-x'\beta}{\sigma})]^{\delta_i}S_0(\frac{logt_i-x'\beta}{\sigma})^{1-\delta_i} \tag{5.9}$$

Where δ_i is the indicator of the event has occurred i.e. uncensored or not. Using $\epsilon_i = \frac{logt_i - x'\beta}{\sigma}$, the log-likelihood assumes the form:

$$l(\beta, \sigma) = \sum_{i=1}^{n} [-\delta_i log\sigma + \delta_i logf_o(\epsilon_i) + (\delta_i) logS0(\epsilon_i)] \tag{5.10}$$

Where $S0(\epsilon_i)$ is the baseline survivor function and $f_0(\epsilon_i)$ is the corresponding density function. The Weibull distribution of survival time is the basis for the most widely used AFT model. The $S0(\epsilon)$ functor corresponds to the extreme value distribution for the log of the lifetime, and the Weibull distribution for lifetime corresponds to the extreme value distribution for the log of the lifetime:

$$S_0(\epsilon_i) = exp(-e^{\epsilon_i}) \tag{5.11}$$

and the $f_0(\epsilon_i)$ function is:

$$f_0(\epsilon_i) = e^{\epsilon_i} exp(-e^{\epsilon_i}) \tag{5.12}$$

The log-likelihood function for AFT model with a Weibull distribution of lifetime is given as:

$$l(\beta, \sigma) = -\sum_{i=1}^{n} [\delta_i log\sigma - \delta_i \epsilon_i + e^{\epsilon_i}] \tag{5.13}$$

Due to minimizing the negative log-likelihood equivalent to maximum a posteriori probability, the loss function we use to optimize is $-l(\beta, \sigma)$. The gradient functions for β and $log\sigma$ respectively are:

$$\frac{\partial(-l)}{\partial\beta} = \sum_{i=1}^{n} [\delta_i - e^{\epsilon_i}] \frac{x_i}{\sigma} \tag{5.14}$$

$$\frac{\partial(-l)}{\partial(log\sigma)} = \sum_{i=1}^{n} [\delta_i + (\delta_i - e^{\epsilon_i})\epsilon_i] \tag{5.15}$$

The AFT model is a convex optimization problem, in which the goal is to find the minimizer of a convex function $-l(\beta, \sigma)$ that depends on the coefficients vector β and the log of scale parameter $log\sigma$. L-BFGS is the optimization method that underpins the implementation. The implementation is identical to the result of R's survreg survival function. The Weibull distribution of survival time is the basis for the most widely used AFT model.

Chapter 6

Model selection

The process of selecting a set of ideal hyperparameters for a learning algorithm is known as hyperparameter tuning. A hyperparameter is a model argument whose value is determined prior to the start of the learning process. Hyperparameter tuning is the cornerstone to machine learning algorithms.

6.1 Parametric grid search

As we said before, model selection, or using data to determine the optimum model or parameters for a particular job, is an essential issue in machine learning. Tuning is another term for this process as well. Individual Estimators, such as LogisticRegression, may be fine-tuned, as can complete Pipelines that comprise numerous algorithms, featurization, and other processes. Instead of tweaking each element in the Pipeline individually, users may tune the entire Pipeline at once. This selection can be performed by performing an exhaustive search over the parameters provided. This way of trying each combination for the best measure is known as grid search. The following figure depicts the grid-search space:

Spark supports model selection using tools such as CrossValidator and TrainValidationSplit. These tools require the following :

- *Estimator*: algorithm or Pipeline to tune Set of *ParamMaps* (or parameter grid).
- *Evaluator*: metric to measure how well a fitted Model does on held-out test data.

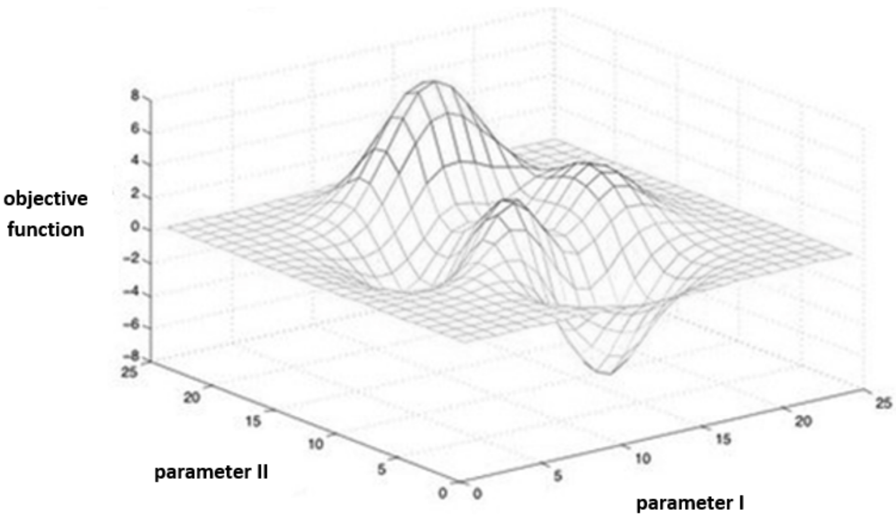


Figure 6.1: Grid-search space

In general these model selection tools work as follows:

- Split the data into training and test sets.
- For each tuple (training,test) they fit the Estimator using parameters provided in the in the parameter grid iteratively, get the fitted Model, and evaluate the Model's performance using the Evaluator.
- Model is selected that is produced by the best-performing set of parameters.

The evaluator can be either for regression problems, binary or multiclass classification problem or even for ranking problems. The default metric used to choose the best parameter grid can be overridden by the customized evaluation metric in each of the evaluators.

Working in a big data environment allows parallelism, In spark, parameter evaluation can be done in parallel by setting *parallelism* with a value of 2 or more (a value of 1 will be serial) before running model selection with eith CrossValidator or TrainValidationSplit.In order to maximize parallelism without exceeding cluster resources, the amount of parallelism should be carefully set, and greater values may not necessarily result in enhanced performance. In general, a number of up to ten should suffice for most clusters.

6.2 Cross validation

CrossValidator starts by dividing the dataset into folds, each of which serves as a distinct training and test dataset. CrossValidator, for example, will construct three (training, test) dataset pairings with k=3 folds, each using 2/3 of the data for training and 1/3 for testing. CrossValidator computes the average evaluation metric for the 3 Models built by fitting the Estimator on the 3 separate (training, test) dataset pairings to assess a given paramMap.

CrossValidator ultimately re-fits the Estimator using the best ParamMap and the complete dataset after determining the best ParamMap.

6.3 Train validation split

In addition to CrossValidator, Spark also provides TrainValidationSplit for hyper-parameter tuning. In contrast to CrossValidator, TrainValidationSplit only examines each combination of parameters once, rather than k times. As a result, it is less costly, but it will not generate as trustworthy results if the training dataset is not large enough.

TrainValidationSplit, unlike CrossValidator, provides a single (training, test) dataset pair. The trainRatio parameter divides the dataset into these two sections. TrainValidationSplit, for example, with trainRatio=0.75, will create a training and test dataset pair with 75% of the data utilized for training and 25% for validation.

TrainValidationSplit, like CrossValidator, fits the Estimator using the best ParamMap and the full dataset.

Chapter 7

Evaluation metrics

The prediction error is used to define model performance in regression issues. The discrepancy between the actual and expected values is characterized as the prediction error, often known as residuals.The regression model tries to fit a line that produces the smallest difference between predicted and actual(measured) values. When determining the quality of a model, residuals are crucial. You may look at residuals to see how big they are and if they create a pattern.

- The model predicts exactly when all of the residuals are zero. The model becomes less accurate as the residuals get further away from zero.
- When residuals contain patterns, it means the model is qualitatively incorrect, since it fails to explain some of the data's features.

Residual = actual value — predicted value
$error(e) = y|\hat{y}$

7.1 Mean absolute error (MAE)

It is the average of the absolute differences between the actual value and the model's predicted value.

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|y_i - \hat{y_i}| \tag{7.1}$$

where
N = total number of instances in the dataset
y_i = actual value
$\hat{y_i}$ = predicted value

The mean absolute error (MAE) is measured in the same units as the original data, and thus can only be compared with models that have the same error units.The larger the MAE, the more serious the mistake. It can withstand outliers. As a result, MAE can cope with outliers by using absolute numbers. Because a large error does not overwhelm a large number of little mistakes, the output gives us a fairly impartial picture of how the model is working. As a result, it fails to penalize the more serious errors.Because MAE is not differentiable, we must use differentiable optimizers such as gradient descent.

7.2 Mean squared error(MSE)

It is the average of the squared differences between the actual and the predicted values.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(y_i - \hat{y_i})^2 \tag{7.2}$$

N = total number of instances in the dataset
y_i = actual value
$\hat{y_i}$ = predicted value

To eliminate the sign of each mistake value and penalise excessive errors, MSE employs the square operation. Because the influence of greater errors is more evident when we take the square of the error, the model may now focus more on the larger faults.On the other hand, if all of the mistakes are minor, or even less than one, we may overestimate the model's shortcomings.

7.3 Coefficient of determination ($\mathbf{R}^2$)

The R-squared value indicates how much the variation of one variable explains the variance of the other. In other words, it calculates the fraction of the dependent variable's variation that can be explained by the independent variable. The R squared metric is a widely used statistic for determining model correctness. It indicates how near the data points are to the regression algorithm's fitted line. A better fit is indicated by a higher R squared value. This aids us in determining the link between the independent and dependent variables. It is the ratio of the sum of squares and the total sum of squares:

$$R^2 = 1 - \frac{SSE}{SST} \tag{7.3}$$

where SSE is the sum of the square of the difference between the actual value and the predicted value:

$$SSE = \sum_{n=1}^{m} (y_i - \hat{y_i})^2 \tag{7.4}$$

and SST is the total sum of the square of the difference between the actual value and the mean of the actual value:

$$SSE = \sum_{n=1}^{m} (y_i - \bar{y_i})^2 \tag{7.5}$$

where y_i is the observed target value, $\hat{y_i}$ is the predicted value, and $\bar{y_i}$ is the mean value, m represents the total number of observations.

Adding more features to the dataset, the R2 score begins to rise or remain constant, but it never falls, since it thinks that as more data is added, the variance of the data rises. The issue is that when we add an unimportant feature to the dataset, R2 occasionally starts to increase, which is wrong.

7.4 Explained variance

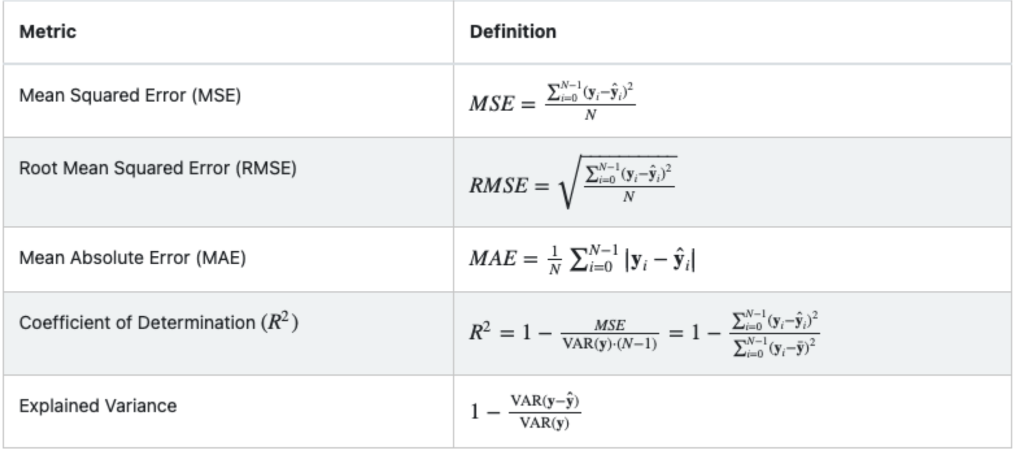
The explained variance score explains the dispersion of errors of a given dataset, and the formula is written as follows:

$$explainedvariance(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)} \tag{7.6}$$

where $Var(y - \hat{(y)})$ and $Var(y)$ are the variance of prediction errors and actual values respectively. Scores close to 1.0 are highly desired, indicating better squares of standard deviations of errors.

7.4.1 Summary and results

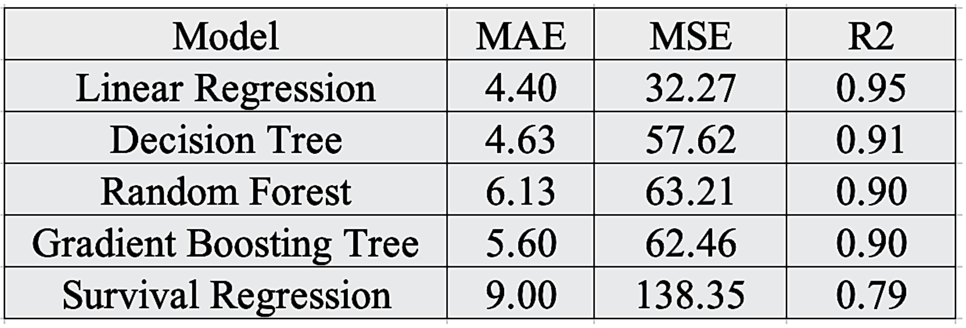
The following table summarizes evaluations metrics discussed in this chapter:



Metric	Definition
Mean Squared Error (MSE)	$MSE = \frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}}$
Mean Absolute Error (MAE)	$MAE = \frac{1}{N}\sum_{i=0}^{N-1}\lvert\mathbf{y}_i - \hat{\mathbf{y}}_i\rvert$
Coefficient of Determination (R^2)	$R^2 = 1 - \frac{MSE}{\mathrm{VAR}(\mathbf{y})\cdot(N-1)} = 1 - \frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{\sum_{i=0}^{N-1}(\mathbf{y}_i - \bar{\mathbf{y}})^2}$
Explained Variance	$1 - \frac{\mathrm{VAR}(\mathbf{y} - \hat{\mathbf{y}})}{\mathrm{VAR}(\mathbf{y})}$

Figure 7.1: Evaluation metrics

The following table shows the results, applying techniques mentioned in chapter 5. We can observe that linear regression model has the best performance overall:



Model	MAE	MSE	R2
Linear Regression	4.40	32.27	0.95
Decision Tree	4.63	57.62	0.91
Random Forest	6.13	63.21	0.90
Gradient Boosting Tree	5.60	62.46	0.90
Survival Regression	9.00	138.35	0.79

Figure 7.2: Summary of ML-model

Chapter 8

Spark streaming with kafka

We have so far performed data analysis as well as applied machines learning techniques on the data evaluating these analytical algorithms using various evaluation techniques. Now the question comes to the deployment of these application is the real time industrial situation. We would like to demonstrate how such machine learning model can be applied to a CNC machine operating in the industry to perform predictive maintenance. The following figure shows the pipeline of the project we would like to implement.In the later sections lets explain each bloch separately.

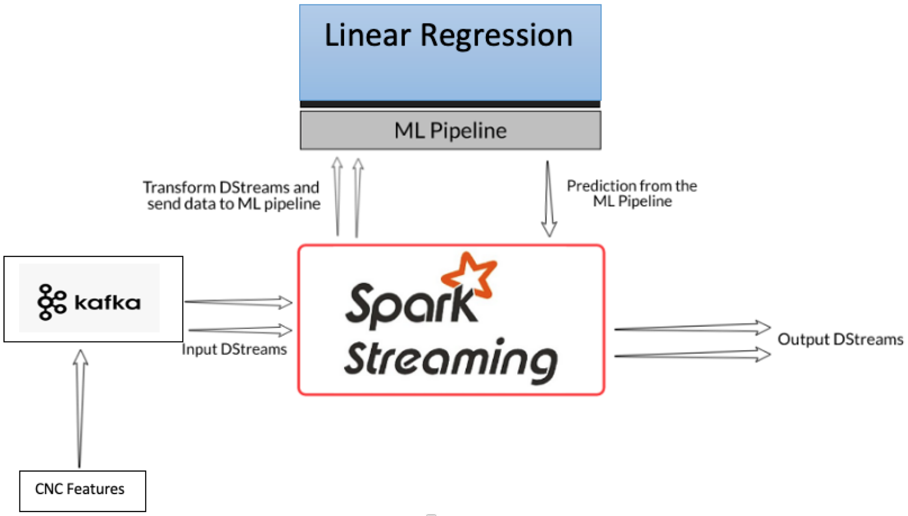


Figure 8.1: Spark Streaming with kafka

8.1 Apache kafka

Apache kafka is a distributed publish-subscribe messaging platform that has been designed specifically to handle real-time streaming data for distributed streaming, pipelining, and replay of data feeds for rapid, scalable processes. This framework was designed by linkdin and later open sourced.

Kafka is a broker-based system that works by storing data streams as records in a cluster of computers. By storing streams of records (messages) across several server instances in topics, Kafka servers may span different data centers and provide data permanence. A topic stores records or messages as a series of tuples, a sequence of immutable Python objects, which consist of a key, a value, and a timestamp.

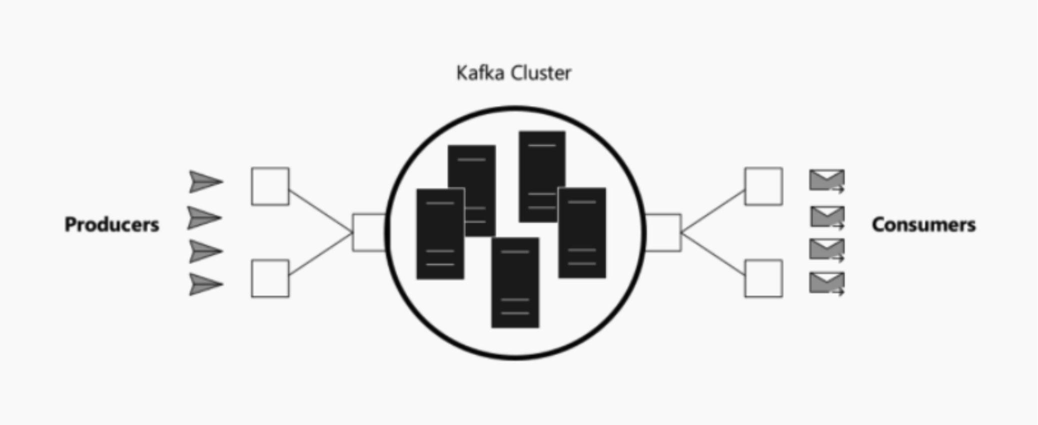


Figure 8.2: Kafka: Overview

8.1.1 Kafka use case

Apache kafka is one of the most popular open source messaging systems available today. This is owing to the architectural design pattern's improved logging mechanism for distributed systems.Being purpose-built for real-time log streaming, Kafka is ideally suited for applications that need:

- Data communication between separate components that is reliable
- The flexibility to divide messaging tasks as application needs vary.
- For data processing, real-time streaming is available.

- Data/message replay is natively supported.

8.1.2 Terminology

Kafka topic

In publish/subscribe communications, a topic is a generally ubiquitous idea. A topic is an addressable abstraction used to demonstrate interest in a specific data stream (series of records/messages) in Apache Kafka and other messaging platforms. A topic is an abstraction layer that the application uses to demonstrate interest in a certain stream of data. It may be published and subscribed to.

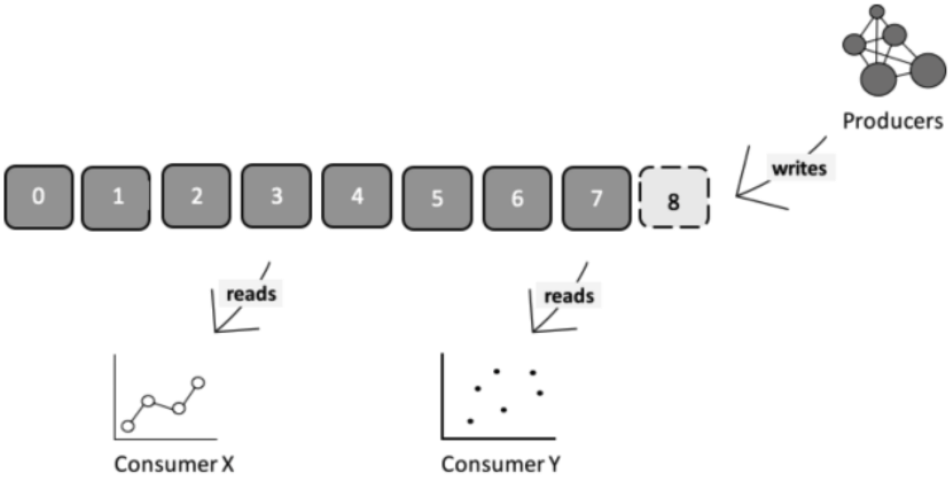


Figure 8.3: Kafka Topics

8.1.3 Partition

Topics in Apache Kafka may be separated into partitions, which are a series of order queues. A sequential commit log is formed by repeatedly appending these segments. Each record/message in the Kafka system is given a sequential ID called an offset, which is used to identify the message or record in a certain partition.

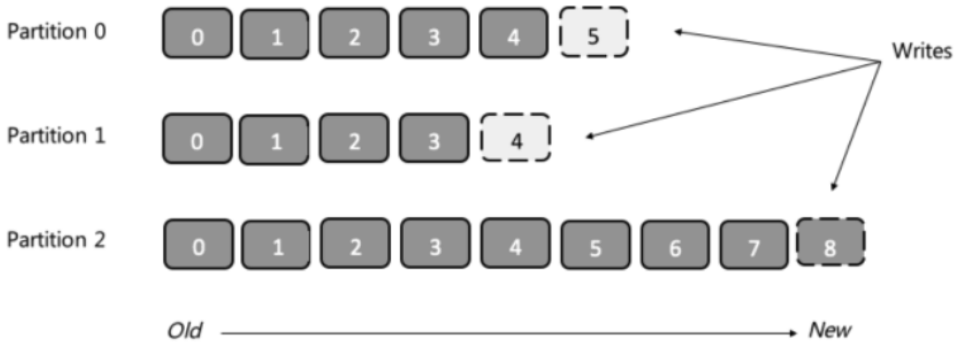
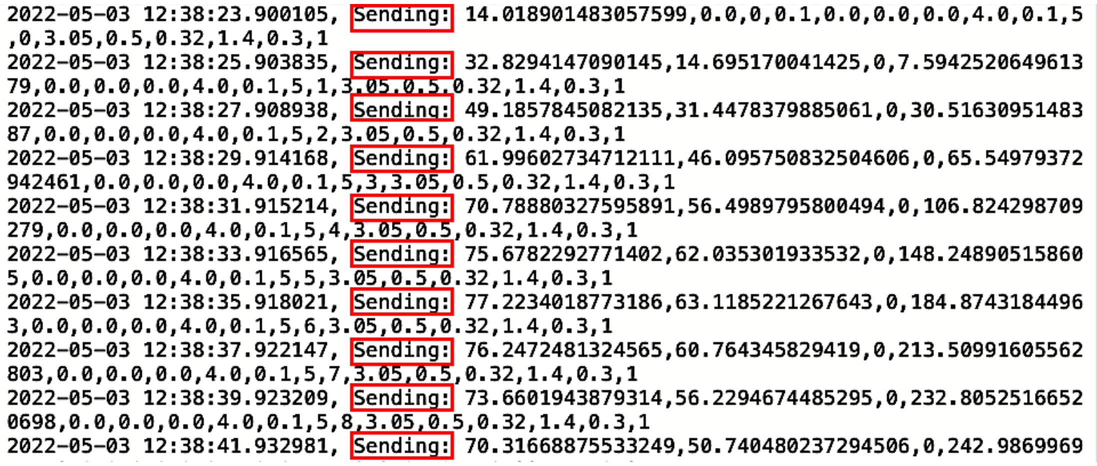


Figure 8.4: Partitions

Kafka producer

The idea of a producer in Apache Kafka is similar to that of other messaging systems. A data producer (records/messages) specifies the subject (data stream) on which a particular record/message should be published. Because partitions are used to increase scalability, a producer can choose which partition a specific record/message is published to. Producers are not required to identify a partition, and by doing so, load balancing between topic divisions can be performed in a round-robin fashion.

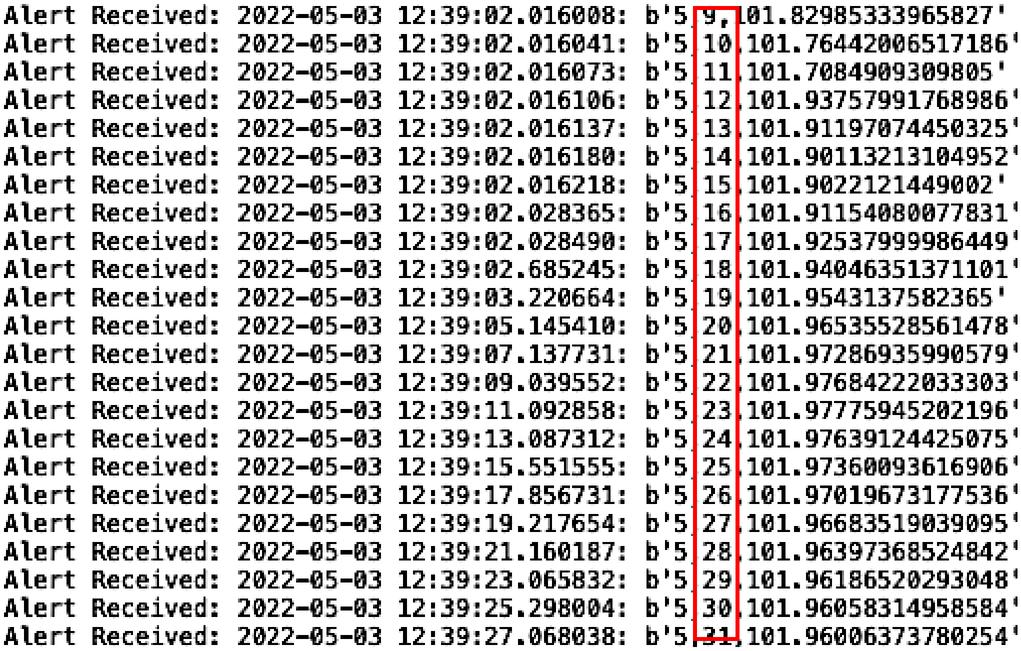


```
2022-05-03 12:38:23.900105, Sending: 14.018901483057599,0.0,0,0.1,0.0,0.0,0.0,4.0,0.1,5
,0,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:25.903835, Sending: 32.8294147090145,14.695170041425,0,7.5942520649613
79,0.0,0.0,0.0,4.0,0.1,5,1,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:27.908938, Sending: 49.1857845082135,31.4478379885061,0,30.5163095148З
87,0.0,0.0,0.0,4.0,0.1,5,2,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:29.914168, Sending: 61.99602734712111,46.095750832504606,0,65.54979372
942461,0.0,0.0,0.0,4.0,0.1,5,3,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:31.915214, Sending: 70.78880327595891,56.4989795800494,0,106.824298709
279,0.0,0.0,0.0,4.0,0.1,5,4,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:33.916565, Sending: 75.6782292771402,62.035301933532,0,148.24890515860
5,0.0,0.0,0.0,4.0,0.1,5,5,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:35.918021, Sending: 77.2234018773186,63.1185221267643,0,184.8743184496
3,0.0,0.0,0.0,4.0,0.1,5,6,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:37.922147, Sending: 76.2472481324565,60.764345829419,0,213.50991605562
803,0.0,0.0,0.0,4.0,0.1,5,7,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:39.923209, Sending: 73.6601943879314,56.2294674485295,0,232.8052516652
0698,0.0,0.0,0.0,4.0,0.1,5,8,3.05,0.5,0.32,1.4,0.3,1
2022-05-03 12:38:41.932981, Sending: 70.31668875533249,50.740480237294506,0,242.9869969
```

Figure 8.5: Kafka producer: Sending CNC Ddata

Kafka consumer

Consumers are the entities that handle records/messages in Kafka, as they are in most messaging systems. Consumers can be set up to work independently on their own tasks or collaboratively on a particular workload with other consumers (load balancing). Consumers manage their task processing based on the consumer group they belong to. Consumers can be dispersed inside a single process, across many processes, and even across multiple systems by using a consumer group name. Consumers can use consumer group names to load balance record/message consumption across the consumer set (multiple consumers with the same consumer group name), or process each record/message individually (multiple consumers with unique consumer group names), where each consumer subscribed to a topic/partition receives the message for processing.



```
Alert Received: 2022-05-03 12:39:02.016008: b'5,9,101.82985333965827'
Alert Received: 2022-05-03 12:39:02.016041: b'5,10,101.76442006517186'
Alert Received: 2022-05-03 12:39:02.016073: b'5,11,101.7084909309805'
Alert Received: 2022-05-03 12:39:02.016106: b'5,12,101.93757991768986'
Alert Received: 2022-05-03 12:39:02.016137: b'5,13,101.91197074450325'
Alert Received: 2022-05-03 12:39:02.016180: b'5,14,101.90113213104952'
Alert Received: 2022-05-03 12:39:02.016218: b'5,15,101.9022121449002'
Alert Received: 2022-05-03 12:39:02.028365: b'5,16,101.91154080077831'
Alert Received: 2022-05-03 12:39:02.028490: b'5,17,101.92537999986449'
Alert Received: 2022-05-03 12:39:02.685245: b'5,18,101.94046351371101'
Alert Received: 2022-05-03 12:39:03.220664: b'5,19,101.9543137582365'
Alert Received: 2022-05-03 12:39:05.145410: b'5,20,101.96535528561478'
Alert Received: 2022-05-03 12:39:07.137731: b'5,21,101.97286935990579'
Alert Received: 2022-05-03 12:39:09.039552: b'5,22,101.97684222033303'
Alert Received: 2022-05-03 12:39:11.092858: b'5,23,101.97775945202196'
Alert Received: 2022-05-03 12:39:13.087312: b'5,24,101.97639124425075'
Alert Received: 2022-05-03 12:39:15.551555: b'5,25,101.97360093616906'
Alert Received: 2022-05-03 12:39:17.856731: b'5,26,101.97019673177536'
Alert Received: 2022-05-03 12:39:19.217654: b'5,27,101.96683519039095'
Alert Received: 2022-05-03 12:39:21.160187: b'5,28,101.96397368524842'
Alert Received: 2022-05-03 12:39:23.065832: b'5,29,101.96186520293048'
Alert Received: 2022-05-03 12:39:25.298004: b'5,30,101.96058314958584'
Alert Received: 2022-05-03 12:39:27.068038: b'5,31,101.96006373780254'
```

Figure 8.6: Kafka consumer: alert

8.2 Apache zookeeper

Kafka uses Zookeeper to store metadata about brokers, topics and partitions. Thus we need to have some information about Apache ZooKeeper which is yet another fascinating service provided by Apache foundation. Zookeeper is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination. ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications.

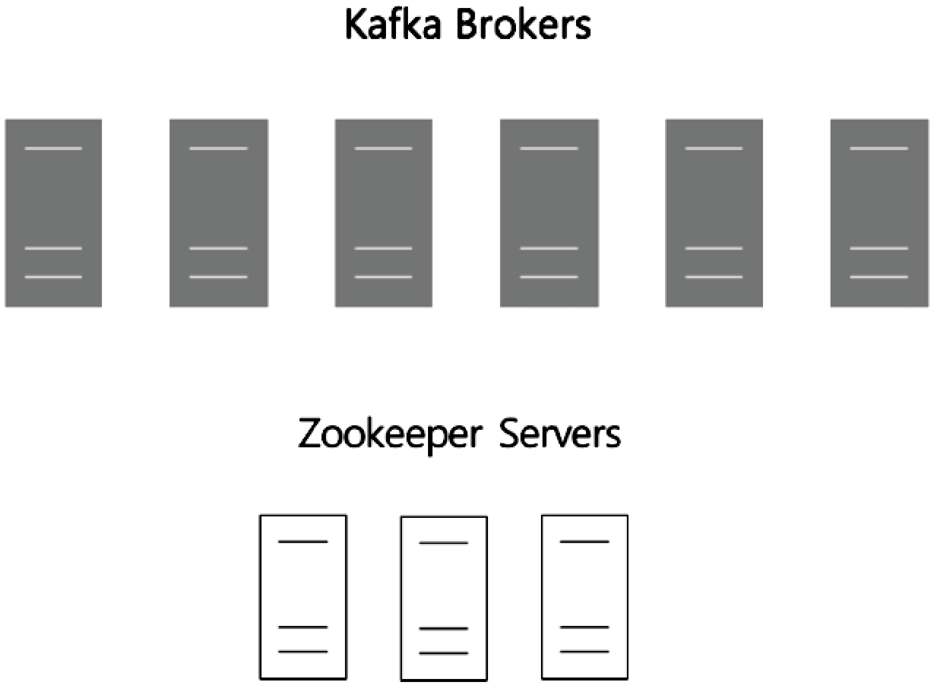


Figure 8.7: Zookeeper as a service

8.3 Spark streaming

Spark Streaming is an extension of the core Spark API that allows for scalable, high-throughput, and fault-tolerant live data stream processing. Data may be ingested from a variety of sources, including Kafka, Kinesis, and TCP connections, and processed using complicated algorithms described using high-level functions like map, reduce, join, and window. Finally, data may be written to file systems, databases, and live dashboards.

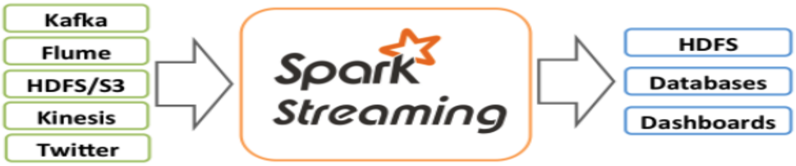


Figure 8.8: Spark streaming: overview

The internal working of the streaming API is as follows.Spark Streaming takes live incoming data streams and separates them into batches, which are then processed by the Spark engine to provide the final batch of results. Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data. DStreams can be produced by performing high-level operations on existing DStreams or by using input data streams from sources like Kafka and Kinesis. A DStream is internally represented as a succession of RDDs. Spark Streaming can be written in python, java or in scala with a few exceptions of APIs that are either different or not available in Python.

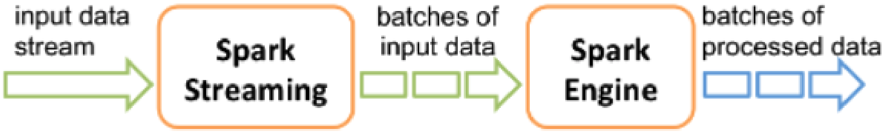


Figure 8.9: Spark Streaming in action

Chapter 9

Conclusion

This section summarizes the thesis work in a concise manner. In the beginning we have discussed maintenance, explaining various types of it with their pros and cons. However, focus of the thesis remained predictive maintenance which allows to predict the remaining useful life of a component during its life cycle. We mentioned the benefits of using predictive maintenance there by stressing the fact that such methods can help reduce unanticipated breakdown and maximize asset up time.

Another dimension of the thesis is to apply prediction using cloud based model which is in contrast with edge computing. The choice was made following the previous work done at brain technologies srl leveraging big data architecture to obtain accurate and generalized prediction models. For this purpose we explained cloud computing and its significance. Basically it is based on pay-as-you-go paradigm, where computing services are provided based on business needs. Moreover we explained the responsibility model (i.e IaaS, PaaS, SaaS) which shows how various layers of cloud architecture can be customised based on service model being used. That means the spectrum of responsibility shared between cloud provider and business user vary among service models ranging from hardware resource management to network security and application development.

In chapter 2, we discussed the CNC model in great detail. Starting from its mechanical and electrical parameters, Features such as linear and rotational velocity of the shaft, horizontal and normal forces acting on the machine, torque, supply voltage, armature current, inductance, resistance, inertia, friction co-efficient beta were considered for the mathematical model. We mentioned the equations of the plant model, However the derivation of these equations is out of scope of this thesis and is based on the work previously done on the same project. We also depicted the different features in graphical form which helps visualize various trends in those features. Moreover we discusses the significance of contact logic and β that depends on factors related to temperature, physical material etc.

Chapter 3 is devoted to exploratory data analysis which is considered to be the basic step in any data analytics pipeline. Understanding the type of data is crucially important for the choice of algorithm in the next step. On top of that, we need to look for any missing values or anomalies in the dataset. We covered that in great detail in that chapter. Starting from feature description, choice of machine parameters, we went on to depict these features using various visualization techniques. The techniques such as heat map, scatter plots, box plot are very useful for data analysis. They help spot feature correlation, data outliers etc. Another important technique introduced in that chapter is PCA which is very useful for dimensionality reduction. However the purpose of applying PCA was to understand the variance of the data and the features that contribute the most to it. This approach provides an additional support in understanding feature importance.

Next step in the knowledge discovery pipeline is to perform data preprocessing. This phase is further divided into different techniques aimed at preparing the data for training predictive models. As we developed the pipeline in spark based setting. these steps included creating spark data frame from extracted data sets, applying feature scaling using transformer model which can also be thought of as an automation process to join all steps in the knowledge discovery phase. Moreover we discussed what actually feature scaling is, different types of feature scaling such as min-max, standard, max-abs and robust scaler etc. Moreover we explained when and when not to perform this step for optimal results in the training phase.

Chapter 5 covers machine learning techniques applied to the data. Considering SOH estimation as a continuous variable, we applied various regression techniques provided by the pyspark framework. Techniques such as linear regression, decision tree, random forest and survival regression have been discussed in detail, providing working principles, pros and cons etc. Relevant concepts are explained using diagrams to help understand them in an easy manner.

Next chapter discusses techniques to train the model in an optimal way. Using techniques such as cross validation and train-validation splitting over fitting of the model can be avoided. Furthermore we discussed grid search which is an exhaustive way of finding best parameters of the model. Grid search is useful for a small number of selection parameters however for algorithms requiring more time fro training, other heuristics could be applied for the estimation of optimal parameters. Afterwords we presented evaluation metrics like mean absolute error (MAE), mean standard error (MSE) etc. We demonstrated the performance of each trained model on the test set using these metrics.

In chapter 8, we demonstrated how a pre-trained machine learning model can be applied to real time data. The pipeline consists of a broker, in this case its kafka. The broker is a distributed publish subscribe messaging platform. Each kafka topic stores consists of a key, value and a timestamp. On one side, kafka producer publishes a record on a data streams. On the other end is the kafka consumer that receives data streams from the subscribed topics only. Note that a consumer can also publish data to another topic at the same time. In the pipeline the stream of data is fed to the machine learning model and evaluated. Once The predicted value (RUL) passes a certain threshold, Alarm is raised via published message on a kafka topic. For technical details, docker is used to run containerized applications, Kafka runs as a container applications and provides pub/sub services for data streaming. Zookeeper on the other hand stores metadata about kafka topics and partitions etc.

Finally relevant links to the thesis work are mentioned as references. This includes softwares used to build the pipeline such as spark, kafka, zookeeper java jdk etc.

References

- **Spark documentation for machine learning:**
  https://spark.apache.org/docs/latest/ml-guide.html

- **Predictive maintenance documentation:**
  https://en.wikipedia.org/wiki/Predictive_maintenance

- **Predictive maintenance for turbo fan engine degradation:**
  https://github.com/sabderra/predictive-maintenance-spark

- **Maven repository containing jar files for kafka integration:**
  https://mvnrepository.com/artifact/org.apache.spark/spark-streaming-kafka-0-10

- **Java jdk download:**
  https://www.oracle.com/it/java/technologies/javase/jdk11-archive-downloads.html

- **Docker download:**
  https://www.docker.com/products/docker-desktop/

- **Spark download:**
  https://spark.apache.org/downloads.html

- **Zooker and Kafka image:**
  https://hub.docker.com/

- **Relevent for debugging code:**
  https://stackoverflow.com/