UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

UNIVERSIDAD DEL PAIS VASCO

FACULTY OF ENGINEERING VITORIA-GASTEIZ

DEPARTMENT OF MECHANICAL ENGINEERING

Diploma Thesis

Machine learning methods for predictive maintenance using real-time data and time-frequency analysis

Dimitrios Iason Papadopoulos

Supervisors: Georgios Chamilothoris, Vanessa Garcia, Saioa Etxebarria

Vitoria-Gasteiz, May 2023

GRADO EN INGENIERÍA MECÁNICA

TRABAJO FIN DE GRADO

Machine learning methods for predictive maintenance using real-time data and time-frequency analysis

Alumno/Alumna: PAPADOPOULOS DIMITRIOS IASON

Director/Directora (1): GARCIA VANESSA

Director/Directora (2): ETXEBARRIA SAIOA

Fecha: 29-05-2023

Acknowledgments

First of all, I would like to thank my supervisor from the mechanical engineering department of the University of West Attica, Georgios Chamilothoris, for guiding me throughout the duration of my thesis and my supervisors from the mechanical engineering department of the University of the Basque Country, Vanessa Garcia and Saioa Etxebarria for their important contribution in my effort. Moreover, I would like to thank my family and friends, for supporting me for the whole duration of my studies.

Abstract

In this diploma thesis, different techniques of Predictive Maintenance based on Machine Learning are compared. In particular, the Remaining Useful Life of a ball bearing of the shaft of a Wind Turbine was predicted with different methods: Classification algorithms, degradation models and real time updates using a Kalman Filter. In the first half, the theory of ball bearing failure mechanisms, predictive maintenance and machine learning is analyzed. At the second half, different methods are implemented for the prediction of the remaining useful life. Last, the writer comes to a conclusion about the efficiency of each method.

Key words: Machine Learning, Predictive Maintenance, Remaining Useful Life, Degradation Models, Classification, Kalman Filter

Περίληψη

Σε αυτή τη διπλωματική εργασία, γίνεται σύγκριση μεθόδων μηχανικής μάθησης για προδεικτική συντήρηση. Ειδικότερα, γίνεται πρόβλεψη για την εναπομένουσα ωφέλιμη ζωή ενός εδράνου, στον άξονα μιας ανεμογεννήτριας, με τις εξής μεθόδους, αλγόριθμους classification, μοντέλα degradation και συνεχής ανανέωση, με χρήση του φίλτρου Kalman. Σε πρώτη φάση, αναλύεται η θεωρία, σχετικά με τους μηχανισμούς αστοχίας των εδράνων, την μηχανική μάθηση και την προδεικτική συντήρηση. Στη συνέχεια, αυτές οι μέθοδοι, χρησιμοποιούνται για τον υπολογισμό της εναπομένουσας ωφέλιμης ζωής. Τέλος, ο συγγραφές αναλύει τα συμπεράσματά του για την αποδοτικότητα της κάθε μεθόδου.

Key words: Μηχανική μάθηση, Προδεικτική συντήρηση, Εναπομένουσα ωφέλιμη ζωή, Degradation Models, Classification, Φίλτρα Kalman

Table of Contents

Table of figures

1.Introduction

1.1 Problem Statement

Without a doubt, ball bearings play a crucial role on the function of a machine because they reduce friction and they absorb the applied loads. Because of these reasons, they are some of the first parts that are checked when a machine malfunctions. Depending on the application, they are considered expendable because they fail after a certain number of cycles of use and they are changed frequently. But, in some cases, ball bearings may be replaced with unfavourable results, because they might possibly come with big costs and their replacement could stop the whole production process. For these reasons, in specific applications the failure of a ball bearing has to be avoided and a distinctive example is that of a wind turbine. A wind turbine can not function without a ball bearing in its shaft, so it is crucial that failure has to be avoided. Failure can be avoided, by monitoring the condition of the ball bearing and by implementing predictive maintenance techniques, to ensure that it functions properly. Moreover, a model that calculates its remaining useful life (RUL), can be used so that the engineer responsible for the ball bearing, can be informed with detail about its health condition and most importantly, know when it is needed to take action. Knowing when action is needed is very important and it increases productivity, because maintenance or any kind of interference can be scheduled very close to the predicted time of malfunction. As a result, the ball bearing is utilized for the maximum available time and also the needs for possible replacements and maintenance are minimized. Thus, time wasted for maintenance and breakdowns and costs for maintenance are greatly reduced.

1.2 Objectives

It is now understood, how beneficial it is to implement predictive maintenance in the ball bearing of a wind turbine and because of that, the objective of this diploma thesis is to implement machine learning methods for predictive maintenance using real time data and time-frequency analysis. Specifically, the remaining useful life of a ball bearing will be predicted and estimated with different methods in order to compare them and highlight their benefits.

1.3 Bearing failure

A sizable share of wind turbine breakdowns is caused by bearing problems. It is not unexpected that a variety of reasons and circumstances might result in premature failure. But, using recommended procedures, applying the right lubricant, and employing the correct materials all significantly increase lifespans.

As it was mentioned before, bearings are necessary for wind turbine functionality, these precise parts are frequently subjected to a range of unfavorable and even harsh working situations and environments. Thus, endangering the dependability and productivity of a turbine as well as the performance and its lifespan. It might be difficult to fully comprehend typical failure mechanisms in turbine bearings since each premature bearing failure will be distinct owing to the numerous potential reasons in the context of specific operating circumstances. Notwithstanding the difficulties, there are ways to decrease the possibility of early bearing failures, lower maintenance and operating costs, encourage longer bearing service life, and, ultimately, support maintaining turbines in operation as planned. The dependability of equipment is a constant source of difficulty for those in charge of running wind farms, and the reliability of bearings throughout a wind turbine is a crucial component of the equation. Failure can occur, due to various root causes and threaten bearing performance and reliability at every turn. [1],[2]

Reason of failure

The most important reason that a bearing can fail is the improper condition monitoring. It is crucial that a bearing is monitored so that the details regarding its condition are known to the operator. If it is not, a variety of problems can occur and the operators will not be able to act in order to prevent such problems. Moreover, an important reason of failure is sudden stops of the wind turbine, because they greatly strain the bearing and can help begin the deterioration early. Generally, rapid changes in torque are the cause of strain development and that the bearing was not designed for. In addition, ineffective lubrication causes significant friction between the roller and the raceway thus creating excessive strain. Last but not least, environmental conditions, such as moisture, can have detrimental consequences for wind turbine bearings. If bearings are not effectively protected form moisture and too much is present within a turbine, rust can occur and lubricants will become ineffective thus causing premature failure. Furthermore, moisture will corrode the bearing and can damage electrical equipment. [3]

Kinds of Failure

In various wind turbine bearing's positions, white etching cracks can appear, as in figure 1-1. These cracks are more common in bigger wind turbines of the megawatt and multi-megawatt classes. These cracks, found at the end of the failure chain, appear white when acid-etched and form within the microstructure of the steel. Microscopically, this may be seen on the surface. Based on failure analysis, their genesis is frequently traceable to the rolling contact fatigue of a bearing and to physical factors that might hasten rolling contact fatigue. High moment loads, friction, heat, misalignment, and other physical factors may result in higher than expected stresses, which may result in fatigue, or environmental factors, such as water contamination, corrosion, and stray electrical currents, which may result in lower than expected material strengths, which may also result in fatigue. An example of fatigue can be seen in figure 1-2. [3]

Figure 1-1 White etching Cracks [3]

Figure 1-2 Material Fatigue [3]

When two improperly lubricated surfaces glide against one another, material is transferred from one surface to the other, generating adhesive wear, the results of this are visible in figure 1-3. In addition, the friction that results can heat the substance to levels that lead to rehardening. Thus, changing the microstructure of the rollers and raceways in a bearing which accelerates the deterioration, because of the increased stress, excessive friction, and generated heat that isn't needed. These elements will wear a bearing down over time until it is no longer usable. [2],[3]

Micropitting, also known as surface distress, is characterized by little cracks that progressively become larger and obstruct a bearing's smooth operation. This deterioration, is typically brought on by insufficient lubrication and affects not just the bearings but also the gear teeth. The ensuing damages result in concentrated stresses and excessive frictional heat, which compromise a bearing's ability to operate. Once it starts, the cracks propagate fast, resulting in failure, spalling (the flaking of the bearing material), and loss of bearing function.[1],[2],[3]

Figure 1-3 Inadequate lubrication [3]

When water or other corrosive substances enter a bearing's inside, corrosion develops like in figure 1-4. Rust can start to form on a bearing's steel surface when lubrication is not providing enough protection, harming the bearing. A bearing is highly vulnerable to water, and only a tiny amount of water is enough to severely reduce service life.[3]

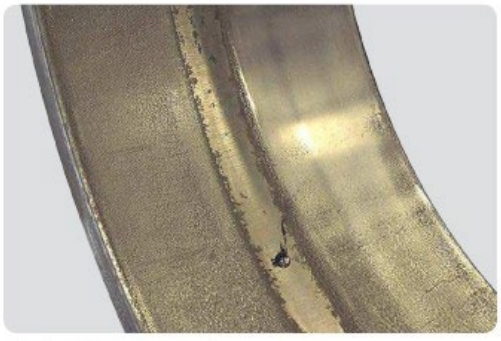

Figure 1-4 Corrosive substance [3]

Failure Prevention

High stresses that cause fatigue can be mitigated by bearings made of premium steel and with compressive residual stresses. Various protective heat treatments, surface treatments, coatings and hybrid bearings incorporating incredibly hard and durable ceramic rolling elements, high-strength stainless steel for corrosion resistance, and other measures can be taken to strengthen the material strength of a bearing depending on the circumstances. The rolling components and inner and outer rings of a bearing may benefit from a specialized black oxidation treatment done by the bearing manufacturer to increase resistance and guard mainly against adhesive wear, as well as for several other failure modes. As the dimensions will remain the same as the bearings that were first placed, bearings with such surface treatments can be used as upgrades and replacements in existing turbine systems. These situations can be avoided with proper lubricant management and regular habits. When it comes to lubricating correctly and keeping an eye out for degrading grease or oil, contaminated water, and particle pollution, maintenance personnel should take care to avoid over- or under-greasing, using the incorrect lubricant, and/or combining incompatible lubricants. Need of lubrication may be significantly reduced, by using suitable sealing design to ensure adequate amounts of lubricant remain. Correct sealing of the places where bearings are positioned helps prevent corrosion. As preventative measures, implementing a humidity management system and using components that are properly designed to prevent condensation inside a system are important.[1],[2],[3]

Predictive maintenance benefits

Early identification of operational issues in wind turbines, is now made possible by Predictive Maintenance. Measurements of numerous physical operational characteristics, such as vibration, temperature, displacements, and others, are used to identify abnormalities. By using the data, bearing and other component issues may be identified before they become more serious and require corrective action.

2.Predictive Maintenance

2.1 What predictive maintenance is

In order to forecast breakdowns well, in advance of the need for immediate action, predictive maintenance makes extensive use of process data and sophisticated analytical techniques. More process data becomes available with the use of ideas like Industry 4.0 or Smart Factory. As a result, it is possible to predict the runtime of assets with increasing precision. This maintenance strategy is typically used when substantial expenditures are incurred as a result of maintenance or downtime. While maintenance tasks are complicated, it can also make scheduling simpler. With this kind of industrial maintenance, businesses are able to foresee problems before they happen and prepare the appropriate maintenance interventions and processes. The development of data processing, analytics, and artificial intelligence has made it possible for maintenance specialists to plan predictive maintenance based on foreseeing errors and malfunctions.[5],[6]

Predictive maintenance keeps an eye on the functionality and state of the equipment while it's in operation. The idea behind it is to be able to anticipate when machinery is likely to break down, based on a variety of parameters, and then lower the risk of failure by preventing failure. Correct prediction requires condition monitoring, which is defined as continuous monitoring of equipment throughout process conditions to guarantee optimum machine use. The application of artificial intelligence has opened up new possibilities for predictive maintenance, since data analysis enables not only the prediction of probable failures but also the formulation of suggestions for modifying operating conditions to obtain the desired production outcomes. Prescriptive maintenance is the term used to describe this maintenance approach.[5],[6]

In predictive maintenance, the engineers in charge of maintaining industrial machinery employ a technique to forecast precisely when a piece of machinery will break down and then carry out repair to keep the production machines operating as efficiently as possible. This makes sure that a piece of equipment in need of repair is turned off just before it breaks, allowing the equipment to function for the duration of the maintenance period for the maximum feasible time. This maintenance strategy's key benefit is the cost savings, by lowering unexpected downtime and raising production rates. The current status of equipment components is presented in real-time statistical data, minimizing production hiccups. The time spent doing maintenance activities is also optimized, in addition to the usage of replacement components. Since it needs the procurement of extremely precise equipment as well as suitable software that can support the data generated during equipment operation, predictive maintenance is regarded as the most difficult maintenance technique. [5],[6]

Predictive maintenance also aims to extend the life span of equipment, as is the case with preventive maintenance. The condition of machinery is monitored, using both overall and component level analysis. This enables replacement parts to be ordered when required and maintenance teams to continuously optimize machinery, thus increasing its longevity. Finally, a significant benefit of predictive maintenance is the provision of an auditable documentation trail.

Because predictive maintenance involves the collection of vast amounts of data, it provides a robust paper trail that can support warranty claims and compliance with Good Manufacturing Practice (GMP) or ISO standards. This documentation trail provides greater transparency, accountability and reliability for companies seeking to demonstrate the quality and reliability of their products and services.[7]

2.2 Comparison to other maintenance approaches

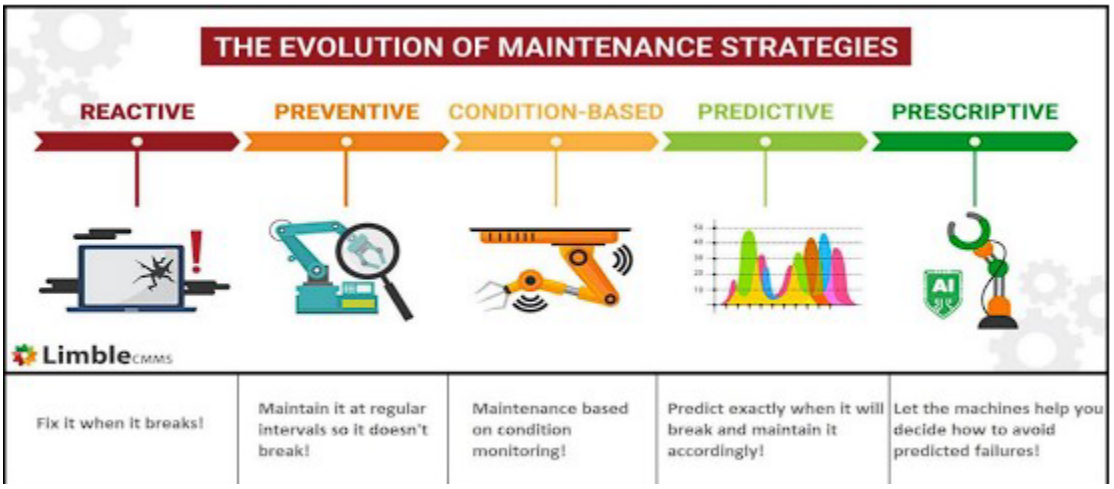

Figure 2-1 The evolution of maintenance strategies [5]

The adoption of predictive maintenance over traditional preventive maintenance is proving to be highly advantageous for businesses across many sectors. The benefits that come with predictive maintenance are significant and numerous. One of the primary benefits is a reduction in maintenance costs. Predictive maintenance enables the allocation of resources and labor only when needed, by analyzing when a machine or device actually requires attention. This is in stark contrast to preventive maintenance, which relies on a set schedule that may not reflect the actual status of the equipment. Another advantage is the reduction in the frequency of major equipment failures. Predictive maintenance quickly identifies issues with equipment, enabling maintenance crews to address the problem before it escalates and causes productivity losses. As such, major equipment failures are greatly reduced, or avoided altogether, compared to traditional maintenance techniques. [7]

2.3 PM applications in industry

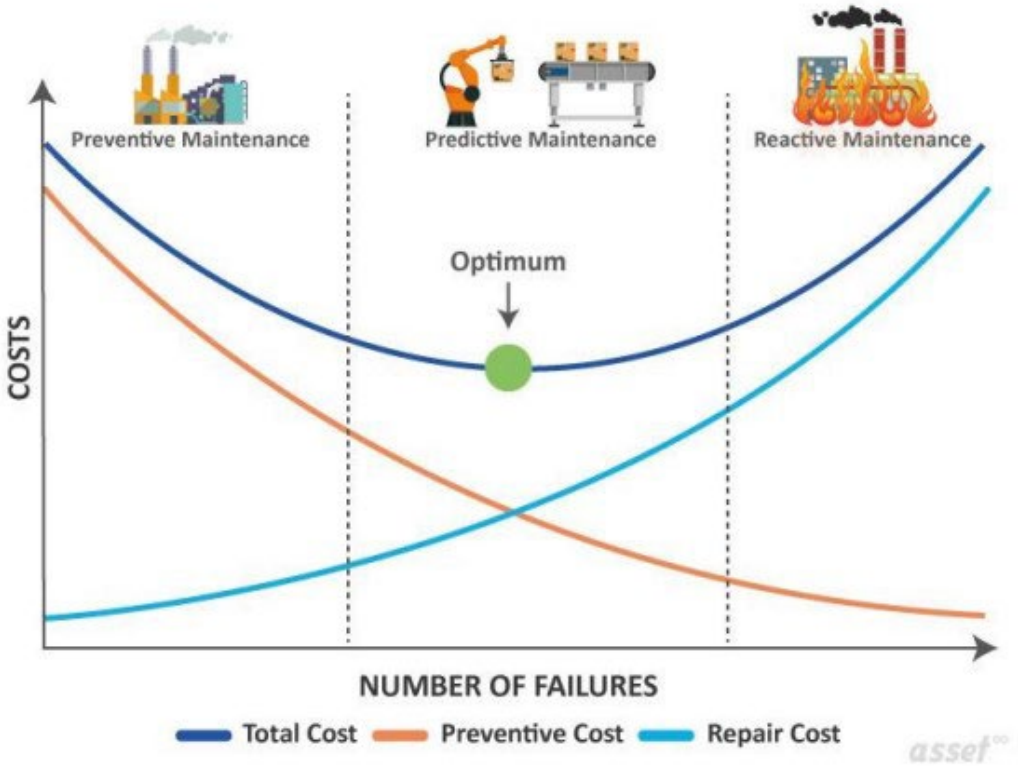

Figure 2-2 Comparison of maintenance approaches [5]

The global predictive maintenance market is projected to reach a value of $6.3 billion in the next years, as indicated in a report conducted by "Market Research Future". This growth can be attributed to the increasing adoption of predictive maintenance techniques in various industries. It is already being utilized or planned to be implemented by 83% of manufacturing companies within the next two years. A report titled "Digital Industrial Revolution with Predictive Maintenance" revealed that 91% of manufacturers implementing predictive maintenance experience a significant reduction in repair time and unplanned downtime. Additionally, 93% of these manufacturers reported improvements in aging industrial infrastructure. According to another report, the adoption of predictive maintenance in factories can yield several benefits, including a 12% reduction in costs, a 9% improvement in uptime, a 14% decrease in safety, health, environment, and quality risks, and a 20% extension in the lifespan of aging assets. Also, the report provides examples of how companies like EasyJet, Transport for London (TfL), and Nestle have leveraged predictive maintenance to enhance the efficiency of their technicians, improve the customer experience, and minimize unplanned downtime. These real-world examples highlight the tangible benefits that can be achieved through the implementation of predictive maintenance strategies.[9]

Predictive maintenance has been proven to be highly cost-effective, according to research conducted by the US Department of Energy. By implementing predictive maintenance software, companies can achieve significant financial gains and enjoy a remarkable return on investment (ROI). The benefits include a substantial reduction of maintenance costs by 25% to 30%, a drastic decrease in breakdowns by 70% to 75%, and a notable decline in downtime by 35% to 45%. In contrast, reactive maintenance is a traditional maintenance strategy where equipment or parts are repaired or replaced only after they have broken down or failed. Surprisingly, many companies still rely on reactive maintenance and organize their maintenance schedules accordingly. This means they wait for failures or breakdowns to occur before taking action to restore the equipment's functionality. However, this approach can be highly costly. Compared to proactive measures such as predictive maintenance, reactive maintenance can result in expenses that are four to five times higher. Immediate costs incurred with reactive maintenance include lost productivity due to unexpected failures, lack of inventory backup for quick repairs, and inefficient communication among maintenance teams. These consequences can be avoided by adopting predictive maintenance strategies, which enable early detection of potential issues and allow for proactive measures to be taken before failures occur.[10]

Leaders at any business that depends on complex machinery or devices know that regular maintenance is essential to smooth and efficient operations. Without timely maintenance, machinery breaks down, leading to downtime, costly repairs and sometimes even replacement. The common practice of preventive maintenance entails regularly inspecting equipment and tuning it up, before it needs repairs. But the emerging practice of predictive maintenance aims to build upon preventive approaches and make them more efficient and cost-effective. When a breakdown occurs, unexpected equipment downtime is the most dreaded consequence, along with poor workplace performance and unplanned expenses. Due to the early engagement and proactive service approach, downtimes can be avoided or planned ahead of time with minimal impact on the customer. In other words, by integrating predictive maintenance, equipment conditions are optimized which ultimately reduce machine downtime and directly influence the bottom line.[10]

2.4 Condition Monitoring

Initially, in order to ensure optimal performance and minimize downtime, it is crucial to identify the conditions that need to be monitored for each machine. This analysis may involve visual inspections, auditory checks, thermal observations, or a combination of these and other criteria. The next technological step is to determine the correct sensors and monitoring tools that need to be installed for each machine.[8]

Vibration analysis is an effective method for predicting potential problems with machinery. By analyzing small changes in vibration patterns, imbalances or misalignments can be identified, while high levels of vibration may indicate issues with bearings or other machine components. Vibration analysis can detect these issues early on, allowing for prompt corrective action to be taken. Sound and ultrasonic analysis can also provide valuable insights by identifying changes in

normal sound patterns that may indicate wear or other types of deterioration. Ultrasonic analyses can further help in determining the overall health of a system by translating high-frequency sounds, such as those produced by steam or air leaks, into the audible range. Infrared analysis is another method that can uncover hidden issues. By using thermography to translate temperature changes into a visible spectrum, even slight deviations from normal operational temperatures can provide early warning signs of impending problems. Fluid analysis is essential for the proper maintenance of machinery. Beyond simply monitoring fluid levels and temperature, a physical and chemical analysis of fluids can provide critical information about the condition of mechanical components. By detecting the rate of degradation in coolants and lubricants, preventive steps can be taken when necessary. Other predictive maintenance technologies are available to cater to specific industrial needs. These include laser alignment, electrical circuit monitoring, crack detection, corrosion monitoring, and electrical resistance measurements. [8]

3.Machine Learning

Machine Learning is a subset of Artificial Intelligence (AI) and as a component of AI, machine learning enables computers to automatically learn from experience and develop without having to be programmed for anything. The creation of computer processes that can utilize data and learn from it independently is the focus of machine learning. In order to detect patterns in the data and build a knowledge base that can be utilized to make judgments in the future, learning as a process begins with the observation of data in the form of examples or some other sort of instruction.[11]

It is understood that machine learning's primary goal is to enable computer systems to automatically learn without human intervention or input while also enabling them to adapt to the demands of various situations. Computers may learn automatically without special programming or human interaction thanks to machine learning, which allows them to modify their behavior accordingly. They are able to learn and develop fundamental behavioral patterns for various classes, train the data, and provide predictions for them through exploratory data analysis and the use of computer algorithms. To identify patterns in the data and make the best judgments moving forward based on the examples, the learning process starts with observations that are examples or empirical outcomes. Given that each of the aforementioned concepts employ classification and regression, machine learning and exploratory data analysis are complementary ideas. Hence, Machine Learning retains the benefits that computers provide while also being tied to mathematical optimization approaches. As a result, machine learning is the process of building models or patterns using a dataset and a computer system. Classification and regression are the most well-known approaches that have been created and employed, depending on the nature of the problem. [11]

Supervised learning

The ability of a machine learning model to learn the function that translates an input into an output based on examples of input-output pairs is called Supervised Learning. A model of this type takes the function from a labeled data set made up of training samples. In supervised learning, each sample consists of an input item and a target value. The training data are examined by such an algorithm, which then deduces a function that may be applied to assign fresh samples. The program should ideally be able to correctly predict the label from unidentified samples.[12]

Unsupervised Learning

Unsupervised learning is a sort of algorithm that looks for patterns in uncategorized data. The objective is to force the machine to imitate, which is a crucial aspect of learning for humans, in order to construct an internal perspective of its surroundings, and then use that internal view to make content. It differs from supervised learning in that the computer organizes the data instead of a human expert by identifying patterns in the form of probability density functions. [12]

Partial supervision learning

A large family of machine learning algorithms known as semi-supervised learning makes use of both labeled and unlabeled data concurrently. This makes it a combination of supervised and unsupervised learning techniques. Partial supervised learning's central concept is to use data in different ways depending on whether or not they are labeled. In the case of labeled data, the method updates the model weights using supervised learning; in the case of unlabeled data, on the contrary, the approach minimizes the difference in predictions across comparable data. [12]

3.1 Data Analysis and Feature Engineering

3.1.1 Data preprocessing

The inductive links between the variables are significant. This implies that when one variable changes, it is only natural for another to do so as well. Finding the connections between the data and the speed at which these connections may be made are two difficulties that arise. The precise use of the appropriate tools and models for each data set, is essential to the validity of these linkages. Data preprocessing consists of a set of procedures that help to obtain a better picture of the data. The most common problem found in data encountered by analysts is the presence of noise. Noise in data is the existence of incorrect values within a data set resulting from data mining. Data that has errors and outliers, i.e., unhelpful information, is classified as noisy and can confuse the mining algorithms, and therefore needs to be addressed at the preprocessing stage.[5]

3.1.2 Mathematical transformations

The preprocessing of data plays a crucial role in building an effective prediction model. It is essential to ensure that the data is in a manageable and suitable form for accurate analysis. Different models may have varying requirements for data processing, ranging from no processing at all to specific data transformations. One common aspect of data processing is the need for all the variables to be on the same scale or possess certain characteristics, such as symmetry. This ensures that the model can effectively interpret and utilize the data. To achieve this, several widely used methods are employed, including centering, scaling, and skewness removal.[13],[14]

Centering involves subtracting the average value from each data point of a predictor variable. This process effectively shifts the distribution to have a zero mean. By doing so, the model can focus on the relative differences between data points rather than absolute values. [13],[14]

Scaling, on the other hand, aims to normalize the variables by dividing each value by the standard deviation. This results in the predictor variables having a common standard deviation of one. Scaling is particularly useful when the variables have different scales or units, as it brings them

to a consistent level for analysis. It ensures that no single variable dominates the model due to its larger scale.[13],[14]

Skewness removal is employed when the distribution of the data is skewed, meaning it is asymmetrical. In such cases, transforming the data using mathematical functions can help achieve a more symmetric distribution. Common transformations include applying the logarithm, square root, or inverse functions to the data. These transformations can mitigate the impact of extreme values and promote a more balanced distribution, which is beneficial for prediction models.[13],[14]

When the values are distributed symmetrically or closely resemble a symmetric distribution, they exhibit a more desirable behavior in prediction models. Symmetric distributions tend to align with the assumptions made by many statistical models, leading to more reliable and accurate predictions.[13],[14]

In summary, data processing techniques like centering, scaling, and skewness removal play a vital role in preparing the data for prediction models. These methods ensure that the data is in a manageable form, with variables on the same scale and possessing desired characteristics, ultimately enhancing the performance and interpretability of the prediction model. [13],[14]

3.1.3 Feature Engineering

The act of choosing, modifying, and converting unprocessed data into features is known as feature engineering. It is important, to create and train better features, in order to make machine learning effective. A machine learning approach, called feature engineering, uses data to generate new variables, that are not present, in the training set. With the aim of streamlining and accelerating data transformations, while also improving model accuracy, it may generate new features for both supervised and unsupervised learning. For machine learning models, feature engineering is necessary. No matter the architecture or the data, a bad feature will directly affect the model. A crucial stage in machine learning is feature engineering. The process of incorporating artificial features into an algorithm is referred to as feature engineering. This algorithm then makes use of these fake traits to enhance performance or, in other words, to provide better outcomes. As data scientists work with data almost exclusively, accuracy of the models becomes crucial. When feature engineering tasks are carried out properly, the final dataset is ideal and includes all of the significant elements that have an impact on the business problem. The most precise prediction models and the most beneficial insights are generated as a result of these datasets.[15]

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a powerful and straightforward method that can greatly enhance our understanding of data by examining its qualities. This approach is often employed when the objective is to generate new hypotheses or identify patterns within the data. It is particularly useful when dealing with large volumes of unanalyzed qualitative or quantitative data.[15]

Outlier Treatment

One important aspect of feature engineering, is outlier treatment, which involves eliminating outliers from a dataset. By removing outliers, one can obtain a more accurate representation of the data across various scales, ultimately impacting the performance of the models. The magnitude of this impact may vary depending on the specific model being used. For example, linear regression is highly sensitive to outliers. Therefore, it is crucial to address outliers before training the models. Some models are not affected by outliers, such as Tree-based models and Support Vector Machines. For example, in Tree-based models, the prediction is based on logical statements made with smaller portions of the data, such as "if predictor A is greater than X, predict the class to be Y". In Support Vector Machines, some parts of the data aren't regarded and such values that are very far from the rest of the data influence the prediction threshold very little. [13],[14],[15]

There are several strategies available for handling outliers in data analysis. The first strategy is removal, which entails deleting data entries that contain outliers. However, it is important to consider that if outliers are present across multiple variables, this approach may result in a significant loss of data, potentially impacting the overall analysis. An alternative approach is replacing values. In this method, outliers are treated as missing values and substituted with suitable imputation techniques. By replacing these extreme values with estimated values based on the remaining dataset, one can maintain the overall distribution while mitigating the influence of outliers. Another technique is capping, where outlier values are replaced with either an arbitrary value or a value obtained from the distribution of the variable. This method ensures that extreme values are substituted with more reasonable values, reducing their impact on subsequent analyses.[13],[14],[15]

By employing these strategies, analysts can effectively manage outliers in their datasets, improving the accuracy and reliability of their analysis. It is essential to carefully consider the specific characteristics of the data and the goals of the analysis when selecting the appropriate outlier treatment method.[15]

Discretization

Another technique commonly used, is discretization. Discretization is the process of converting continuous variables, models, or functions into discrete ones. This is achieved by constructing a series of continuous intervals, also known as bins, that span the range of the desired variable, model, or function. Discretization can be particularly useful when dealing with data that contains a large number of distinct values, as it simplifies the analysis by reducing the number of unique categories.[15]

Overall, exploratory data analysis, coupled with outlier treatment and discretization techniques, provides valuable insights into the data, facilitates the discovery of patterns, and enables more

accurate modeling and analysis. By thoroughly understanding the characteristics and peculiarities of the data, we can extract meaningful information and make informed decisions. [15]

Principal Component Analysis

The aim of Principal Component Analysis (PCA) is to find a set of linear combinations of the predictors that have the highest variance, otherwise known as the principal components. To achieve this, the first principal component is determined by finding the linear combination of predictors with the greatest possible variability. Following this, subsequent principal components are calculated to capture any remaining variation while remaining uncorrelated to all previous principal components. The key advantage of PCA is that it produces uncorrelated components, which is why it is a popular method for data reduction. In some predictive models, uncorrelated or low correlation predictors are preferred for improved numerical stability and optimized solutions.[13]

PCA is a method that generates new predictors suited to these types of models. Initially, PCA prioritizes predictors with greater variation by searching for linear combinations that maximize variability. In the case of predictors with different orders of magnitude, the first few components will summarize higher magnitude predictors, while the later components will summarize lower variance predictors. Thus, higher variability predictors will have larger PC weights on the initial components. It is noteworthy that PCA identifies data structure based on measurement scales instead of significant relationships that suit the current problem. As most data sets consist of predictors on different scales and skewed distributions, it is recommended to transform skewed predictors and center and scale the predictors beforehand to prevent PCA from summarizing distributional differences and predictor scale information. Centering and scaling allow PCA to uncover the underlying relationships within the data without bias from its original measurement scales. After selecting the appropriate predictor variable transformations, PCA can be applied. Lastly, for data sets with numerous predictor variables, it is necessary to determine the components to retain. [13]

One popular technique for determining the optimal number of components to retain when performing PCA is to use a heuristic approach that involves generating a scree plot. This type of plot displays the ordered component number on the x-axis and the corresponding amount of summarized variability on the y-axis. Typically, the first few principal components will capture a large proportion of the total variability in the data, resulting in a sharp drop-off in the scree plot. Beyond this point, additional components will contribute less and less to the overall variability until the plot levels off.

By examining the scree plot, analysts can determine the point at which the rate of decrease in summarized variability slows down and the curve begins to level off. In general, the optimal number of components to retain can be selected by identifying the component number just before this leveling off point. For example, in Figure 3-1, it is clear that the curve begins to level off after

component 5, suggesting that the first four principal components would be most useful in summarizing the variability in this dataset.

Using a heuristic approach such as this can help to simplify the PCA process by providing a straightforward method for identifying the most informative principal components. By retaining only the most informative components, analysts can reduce the dimensionality of their data and provide a more concise summary of the underlying patterns and relationships within the original dataset. [13]

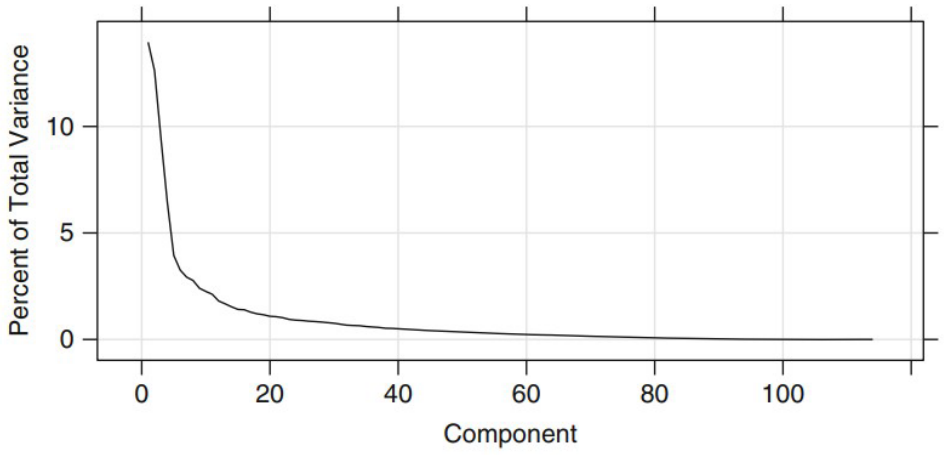

Figure 3-1 Plot of the percentage of variance of each component [13]

Another exploratory use of PCA is characterizing which predictors are associated with each component. Recall that each component is a linear combination of the predictors and the coefficient for each predictor is called the loading. Loadings that are near zero imply that the predictor variable had minimal effect on that particular component. This feature of PCA enables one to identify and interpret the significance of individual predictors in the identification of a given principal component. Therefore, exploring the loadings of a dataset can lead to a better understanding of its underlying structure and provide valuable insights into the relationship between inputs. [13]

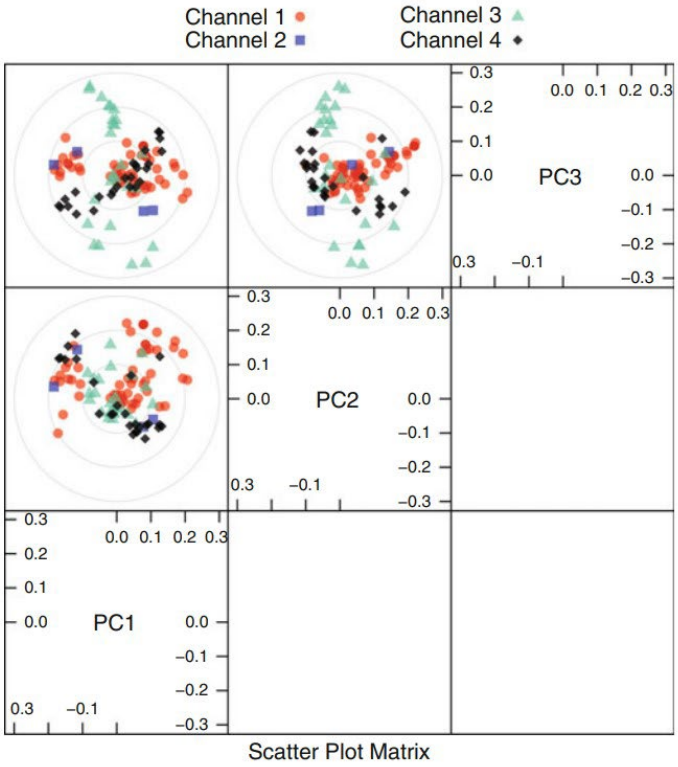

Figure 3-2 Example of PCA with three components [13]

Missing Values

Another problem that occurs in predictors is missing values. In some cases, the values are missing because the values are under a limit of detection and in these cases, they are often given a random value from zero to that limit. The existence of missing values is mostly connected to the predictor, rather than the sample. If the number of missing values is substantial, then that predictor is removed from the model. Though, some models are unaffected by missing values. Moreover, when missing values burden the model, it is possible that they are imputated and this can be achieved by constructing a predictive model for the imputation of the data. A usual imputation method is K-Nearest Neighbours that will be thoroughly analysed later. [13]

Removing Predictors

Removing predictors can potentially improve the model. That happens because fewer predictors decrease the complexity of the model, thus the computational time is also decreased. In addition, the removal of predictors that are greatly correlated, is beneficial because providing the same

information only burdens the model. Also, the removal of predictors with problematic values, increases the model's efficiency and stability. [13]

Correlations between predictors

Sometimes, there is a relationship between predictors and they are correlated. In order to calculate that, a correlation matrix can be made, which shows the correlation between the predictors. Generally, strongly related predictors are avoided because they make the model more complex without improving it. [13]

Adding Predictors

Mostly for categorical predictors, like gender or race etc, predictors can be decomposed in the form of dummy variables. In order to form the dummy variables, the data is disintegrated and categorised. The new categories formed is matched with a different dummy variable that usually has a value of 1 or 0. [13]

Binning

The main motivation of binning is to make the model more robust and prevent overfitting, however, it has a cost to the performance. Every time something is binned, information is sacrificed and the data is more regularized. The trade-off between performance and overfitting is the key point of the binning process. For numerical columns, except for some obvious overfitting cases, binning might be redundant for some kind of algorithms, due to its effect on model performance. However, for categorical columns, the labels with low frequencies probably affect the robustness of statistical models negatively. Thus, assigning a general category to these less frequent values helps to keep the robustness of the model. For example, if the data size is 100,000 rows, it might be a good option to unite the labels with a count less than 100 to a new category like 'Other'.[14],[15]

One-hot encoding

One-hot encoding is a widely used technique in machine learning for encoding categorical variables. This method transforms categorical data, which can be difficult for algorithms to interpret, into a numerical format that facilitates analysis and modeling. It achieves this by spreading the values in a column across multiple flag columns and assigning binary values of 0 or 1 to indicate the presence or absence of a particular category.[14]

The process of one-hot encoding involves creating new binary columns, each corresponding to a unique category in the original categorical column. If there are N distinct values in the column, it is sufficient to map them to N-1 binary columns. The reason for this is that the absence of a

category can be inferred from the absence of a 1 in all the binary columns. Hence, the missing value can be deduced without the need for an additional column.[14]

The term 'one-hot' in one-hot encoding refers to the representation of each category as a vector with a single element set to 1, indicating its presence, and all other elements set to 0. This binary representation effectively captures the relationship between the original categorical column and the encoded columns. One-hot encoding allows algorithms to process categorical data and leverage the information it contains without losing any significant details. By converting categorical variables into a numerical format, it enables the grouping and comparison of categories, as well as the calculation of distances between different categories. This encoding technique is particularly useful in various machine learning tasks, such as classification and clustering, where categorical variables need to be incorporated into models that require numerical inputs. It helps algorithms interpret and analyze categorical data more effectively, enhancing the performance and accuracy of machine learning models.

Grouping Operations

In most machine learning algorithms, the training dataset follows a structure where each instance is represented by a row, and each column corresponds to a distinct feature of that particular instance. This type of organized data is commonly referred to as "Tidy" data. However, it is important to note that many datasets often do not conform to the tidy data definition mentioned earlier. This deviation arises due to instances having multiple rows associated with them. In such scenarios, a common approach is to group the data based on the instances, resulting in each instance being represented by a single row. The key focus during these group by operations is to determine the appropriate aggregation functions for the features. When dealing with numerical features, it is often convenient to use aggregation options such as the average or sum functions. On the other hand, handling categorical features presents a greater complexity in selecting suitable aggregation methods. [14]

Categorical Column Grouping

When dealing with categorical columns, there are several options available for grouping and encoding the data. The first option involves selecting the label with the highest frequency, essentially performing a 'max' operation for categorical columns. This approach identifies the most common label within each group and assigns it as the representative value. It is a simple yet effective method for encoding categorical data. Another approach is to create a pivot table. This method is similar to the encoding technique discussed earlier but with a slight difference. Instead of using binary notation, it utilizes aggregated functions to define values for the grouped and encoded columns. This approach proves beneficial when the goal is to move beyond binary flag columns and consolidate multiple features into aggregated features, which often provide more

informative representations of the data. Lastly, one can apply a group by function after performing one-hot encoding. This approach retains all the data while simultaneously transforming the encoded column from a categorical to a numerical representation. This can be particularly useful when further analysis or modeling requires numerical input rather than categorical data.[15]

Numerical Column Grouping

When it comes to grouping numerical columns, two commonly used aggregation functions are the sum and mean functions. The choice between these functions depends on the specific meaning and purpose of the feature being analyzed. The sum function is often preferred when the numerical column represents a quantity that can be accumulated or added up. For example, if one is dealing with a dataset of sales transactions and has a numerical column representing the total revenue generated from each transaction, using the sum function would allow the user to group and calculate the total revenue for different categories or subsets of the data. This can be useful when trying to understand the overall revenue contribution from various factors or when comparing the total values among different groups. On the other hand, the mean function is commonly used when the numerical column represents an average or a measure of central tendency. For instance, if the user has a dataset of student grades and a numerical column represents the scores achieved in a particular subject, using the mean function would allow them to calculate the average score for different groups or categories. This can provide insights into the performance level or the typical scores attained within each group. Ultimately, the selection of the sum or mean function depends on the specific nature of the numerical column and the analytical goals. Both functions serve different purposes, with the sum function focusing on the accumulation or total value, while the mean function emphasizes the average or central tendency of the data.[15]

3.2 Classification algorithms

Classification is a predictive modelling technique that involves utilizing input variables to estimate a mapping function, enabling the identification of discrete output variables such as labels or categories. The primary task of a classification algorithm is to predict the label or category associated with a given set of input variables. While classification methods can incorporate both discrete and real-valued variables, the key requirement is that instances must be assigned to one of two or more distinct classes.[16]

Classification algorithms are employed to make predictions about data. These algorithms operate by utilizing a provided dataset that has already been categorized into two or more distinct classes. This labeled dataset serves as the input, enabling the generation of a classification model. Once the model is created, it can be utilized to assign new, unlabeled data to the appropriate class based on the learned patterns and characteristics observed in the labeled dataset. [17]

The initial dataset is typically divided into two main groups: the training dataset and the test dataset. The training dataset is used to construct the classification model, while the test dataset is employed to evaluate the performance of the model. To ensure the validity of the model, various partitioning techniques, such as Cross-Validation, are used for validation. Cross Validation involves dividing the dataset into a predetermined number (k) of groups or folds. One of these groups is designated as the test set, while the remaining groups serve as the training set. This process is repeated k times, with each group being used as the test set once. Cross-Validation is a versatile approach that works effectively on different types of datasets. However, alternative partitioning methods, such as fixed partitioning, can be employed when dealing with exceptionally large datasets. [17]

It's worth noting that different classification approaches can vary significantly from one another. There are various algorithms and techniques available, each with its own distinct characteristics and suitability for different types of problems. To assess the performance of various classification algorithms, several metrics are computed to evaluate the quality of the constructed model and determine if any adjustments are needed. One key metric is accuracy, which measures the proportion of correctly classified data compared to the total data in the test dataset. Additionally, for each class, metrics like recall and precision are determined. Recall refers to the ratio of data accurately classified in a specific class to the total data belonging to that class. It focuses on capturing how well the model identifies instances of a particular class correctly. On the other hand, precision represents the ratio of data correctly classified in a specific class to the total data assigned to that class. Precision measures the model's ability to accurately label instances for a given class. It is crucial to consider recall and precision alongside accuracy, as accuracy alone may not fully characterize the model's output. This becomes particularly important when dealing with imbalanced datasets, where the distribution of classes is uneven. In such cases, evaluating recall and precision helps provide a more comprehensive understanding of the model's performance and its ability to handle the imbalanced nature of the data. [17]

3.2.1 SVM (Support Vector Machines)

The support vector machine is likely the most widely used supervised learning approach for classification and regression problems. Nonetheless, it is primarily largely employed in machine learning for classification challenges. As seen in the figure 3-3, the purpose of this technique is to discover the best line, or decision boundary, for classifying the n-dimensional space into classes so that more data points may be conveniently placed in the correct class in the future. The hyperplane is the ideal decision boundary, and the support vector machine selects the outermost vectors that contribute to build it.[5]

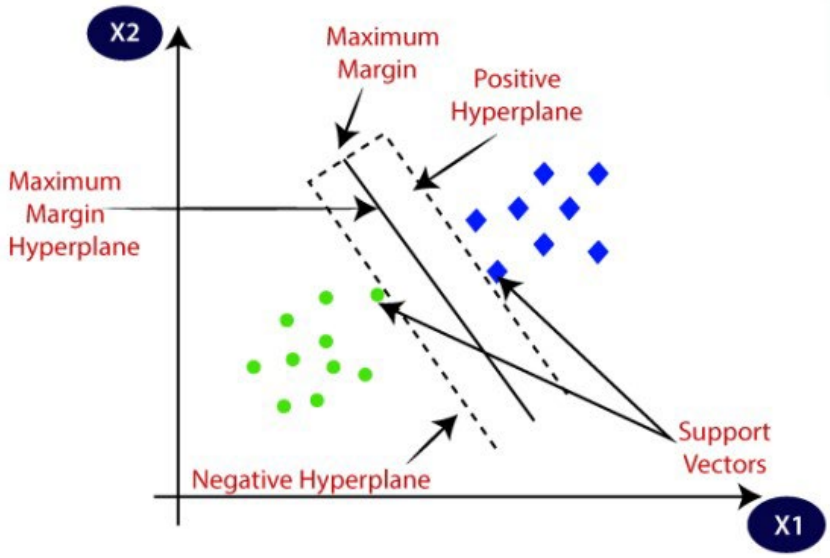

Figure 3-3 Support Vector Machines algorithm [5]

The linear type is a classifier for linearly separable data, which implies that if a dataset can be classified into two classes using a single straight line, it is linearly separable data, and the classifier is linear. Non-linear data is non-linearly separable data, which means that if a dataset cannot be classified using a straight line, it is non-linear data, and the classifier employed is non-linear. The benefits of employing support vector machines include the fact that they operate well in high-dimensional spaces and that even when the number of dimensions exceeds the number of samples, the approach remains effective. Furthermore, because it only employs a subset of training points in the decision function, the support vector machine is cheap in terms of computing memory and adaptable because alternative functions may be given for the choice function.[5],[13]

One downside of utilizing a support vector machine is that it is vital to avoid overfitting when selecting the normalization functions and conditions if the number of features exceeds the number of samples. Lastly, because they are obtained from a time-consuming five-fold cross-validation procedure, support vector machines do not provide direct probability estimates.[5],[13]

3.2.2 K – Nearest Neighbours

Figure 3-4 K nearest neighbours algorithm [18]

The K Nearest Neighbors (KNN) method is used to address classification and regression issues. The approach used to produce these predictions works by computing the distance between the test and training data, assuming that the difference between comparable data is modest. Based on the previously registered locations from the training data, it enables the identification and categorization of new data while taking their features into consideration. The algorithm will register these features of the new sites and categorize them based on their proximity to other places, which can be seen in the figure 3-4. The parameter "K" in KNN refers to the number of nearest neighbors, i.e. the space produced by these nearest points. The distance between the new and training data points is determined by the value of K. K is a positive integer that is normally small and should be odd.[12]

3.2.3 Naive Bayes

Bayesian networks are a type of probabilistic machine learning technique that is based on the Bayes theorem and may be used for a variety of classification problems. Bayes' theorem is a straightforward mathematical method for calculating probability. Conditional probability is a measure of the likelihood of an event occurring if another event has already occurred. It is a classification strategy based on Bayes' theorem, which assumes prediction independence. Simply expressed, the Bayes classifier believes that the presence of one feature in a class has no bearing on the presence of other attributes.[5]

Bayes' Theorem is a method that calculates the probability of an event based on a set of other probabilities. Specifically, it determines the posterior probability of event A given that event B has

occurred. Utilizing Bayesian networks offers several benefits. First, it is a simple and efficient technique for accurately predicting multiple classifications and determining the order of a test data set. Second, when the independence criteria are met, Bayesian networks outperform models like logistic regression while using less training data. Additionally, they demonstrate superior performance when dealing with input variables that have classes, especially in comparison to numerical input variables. Despite these advantages, there are drawbacks to employing Bayesian networks. For instance, if a category variable in the test data set includes a category not present in the training data set, the model will automatically assign a probability of zero, rendering it unable to make accurate forecasts. This issue is known as zero frequency. Moreover, Bayesian networks are often considered poor estimators, necessitating caution when interpreting prediction probabilities. Lastly, the assumption of independent predictions poses another weakness, as obtaining a group of entirely independent predictors is challenging in practical applications.[5]

3.2.4 Decision Trees

A dynamic and well-liked technique for classification and regression applications is the decision tree. Various properties are examined retroactively when building decision trees, and the attribute that best separates the data is utilized in each node. By employing the characteristics that are most effective for the job, the training sample is segregated using decision trees. The shortest tree that can be used to correctly represent the input-output connections while avoiding overfitting. Data and the best feature are first placed into the root node of decision trees so that they may be segregated based on their metrics.[11]

In contrast to other supervised learning algorithms, the decision tree method may be utilized to resolve regression and classification issues. The purpose of employing a decision tree is to build a training model that can be used to predict the order or value of the target variable by learning fundamental choice rules gleaned from previous data. The 'root' of the decision tree serves as the beginning point for predicting a class label in a record. The root attribute and the record's attribute values are contrasted. Based on the comparison, the branch associated with that value is followed and moved on to the following node. There are two different forms of decision trees, depending on the type of variable. the continuous variable decision tree, where a continuous variable decision tree is one that has a continuous target variable, and the categorical variable decision tree, which is a categorical variable decision tree with a categorical target variable.[5]

Decision trees have a problem in that they are extremely scalable, especially if a database has a lot of columns. A procedure called as decision tree pruning is used to address this tree overload. Pruning a decision tree involves deleting decision nodes one at a time, beginning with the leaf node, to preserve overall correctness. This is accomplished by splitting the actual training set into training and validation portions.[5]

3.2.5 Random Forest

Using decision tree techniques, the random forest is a supervised machine learning approach. It uses machine learning to address classification and regression problems. It makes use of ensemble learning, a method for mixing many classifiers to solve challenging issues. A random forest algorithm is made up of several decision trees and based on the decision tree's predictions, this algorithm decides the result. By averaging the results of several trees, the prediction is made and when there are more trees, the accuracy of the results get better. When coping with missing data, the random forest method performs more accurately than the decision tree approach. Moreover, it avoids the issue of overfitting the decision tree by randomly selecting a subset of features at each random forest tree's node split point.[5]

The main difference between random forest and decision tree methods is that the latter randomly constructs root nodes and divides nodes. The random forest uses the storage approach to produce the necessary forecast. Storage necessitates the usage of several samples as opposed to a single sample of data. A collection of attributes and observations used to generate predictions is known as a training dataset. The output of decision trees from the random forest method varies depending on the training data used.[5]

The outputs will be graded, and the best one will be chosen as the ultimate output. An ensemble approach is used by random forest classification to provide the desired outcome. The training data is used to train different decision trees. After the nodes are separated, a random selection of observations and characteristics are included in this dataset. The random forest algorithm's capacity to be applied to both regression and classification with accurate predictions is one of its benefits. The random forest algorithm is better at predicting outcomes than the decision tree approach and is able to handle big data sets with ease. Nevertheless, it has certain drawbacks, including the need for more time and advanced processing resources as compared to the decision tree approach.[5]

3.2.6 Logistic Regression

Despite its name, logistic regression is not a regression model; rather, it is a classification model. In terms of problems involving binary and linear classification, it is a fairly efficient method. A classification model with linearly separable categories that is easy to use and yields outstanding results. It is a commonly used classification technique in the industrial sector. The logistic regression model is a statistical method for binary classification that may be expanded to multi-class classification, just like Adaline and Perceptron.[5]

The training data must be used to construct the logistic regression algorithm's coefficients. For this, maximum likelihood estimate is employed. Maximum likelihood estimate is a common learning approach utilized by many machine learning algorithms, despite the fact that assumptions are made on the distribution of the data. Using the best coefficients, a model could forecast values for the default class that were very near 1, and for the other class, values that were very near 0. Maximum likelihood logistic regression seeks coefficient values that minimize

the difference between the model's projected probability and the actual probabilities found in the data. The emphasis of the methodology, the logistic function, led to the naming of the technique logistic regression. In order to describe the features of population expansion in ecology, such as how it expands quickly and finally approaches the carrying capacity of the ecosystem, statisticians developed the logistic function, often known as the sigmoid function. Each real-valued integer may be converted to a number between 0 and 1, but never exactly between those two points, using this S-shaped curve.[5]

3.3 Regression algorithms

Regression methods are employed to predict continuous values by establishing a mapping function between the input variables and the output variable. The main focus of regression problems is to estimate this mapping function accurately. Regression models are particularly useful when the target variable is a numerical value, such as the remaining useful life of a component, or a probability, such as the likelihood of failure for a part in a machine.[16]

A wide range of regression algorithms exists to tackle different types of problems and accommodate various data characteristics. These algorithms offer diverse approaches and techniques to capture the underlying relationships between the input and output variables. Some commonly used regression algorithms include linear regression, polynomial regression, support vector regression, decision tree regression, random forest regression, and neural network regression. These algorithms work with the same principles in Regression as in Classification and they have small differences to adapt to the different goal of each method.

Linear regression is a straightforward and widely applied algorithm that assumes a linear relationship between the input and output variables. Polynomial regression, on the other hand, extends the linear model by introducing higher-degree polynomial terms to capture non-linear relationships.[16]

These are just a few examples of regression algorithms, and the choice of algorithm depends on the specific problem, the nature of the data, and the desired accuracy and interpretability of the model.

3.3.1 Simple linear regression

In the context of quantitative variables, simple linear regression is a fundamental statistical technique used to estimate the relationship between an independent variable and a dependent variable. It aims to establish a linear connection between the two variables by fitting a straight line to the data. The independent variable, also known as the predictor or explanatory variable, is the variable that is believed to have an impact on the dependent variable. The dependent variable, also known as the response variable or outcome variable, is the variable being predicted or explained by the independent variable.[16]

The objective of simple linear regression is to find the best-fitting line that minimizes the difference between the observed data points and the predicted values on that line. This line is determined by estimating the slope and intercept parameters. The slope represents the change in the dependent variable for each unit change in the independent variable, while the intercept indicates the value of the dependent variable when the independent variable is zero. By analyzing the relationship between the independent and dependent variables through linear regression, one can quantify the strength and direction of the association. The resulting linear model can then be used to make predictions or draw inferences about the dependent variable based on specific values of the independent variable.[16]

It is important to note that the assumptions of simple linear regression should be carefully considered, such as linearity, independence, constant variance (homoscedasticity), and normality of residuals. Violations of these assumptions may impact the accuracy and reliability of the regression analysis, requiring further exploration and potential adjustments. Simple linear regression serves as a foundational tool in data analysis and provides a basis for more advanced regression techniques. It is commonly used in various fields, including economics, social sciences, finance, and engineering, to uncover relationships and make predictions based on quantitative variables.[16]

3.3.2 Multiple linear regression

Multiple linear regression is an extension of simple linear regression that allows for the prediction of a dependent variable based on the values of two or more independent variables. It is a statistical technique that enables the examination of the relationship between multiple predictors and a single response variable.

In multiple linear regression, the goal is to establish a linear equation that best describes the relationship between the dependent variable and the independent variables. The equation takes the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

Here, Y represents the dependent variable, β_0 is the intercept, β_1, β_2, ..., β_p are the regression coefficients (also known as slopes), X_1, X_2, ..., X_p represent the independent variables, and ε denotes the error term.

The coefficients (β_1, β_2, ..., β_p) in the equation provide the estimated change in the dependent variable associated with a one-unit change in each respective independent variable, holding other variables constant. The intercept (β_0) represents the estimated value of the dependent variable when all independent variables are zero.[16]

Multiple linear regression assumes that the relationship between the dependent variable and the independent variables is linear. It also assumes that the error term (ε) follows a normal distribution with a mean of zero and constant variance. Additionally, it assumes that the independent variables are not highly correlated with each other (multicollinearity). By fitting the multiple linear regression model to the data, analysts can examine the statistical significance of each independent variable and assess their individual contributions to the prediction of the dependent variable. The model can be used to make predictions, test hypotheses, and gain insights into the relationships between the variables. Before conducting multiple linear regression, it is important to assess the assumptions of the model, such as linearity, independence of errors, constant variance, normality of residuals, and absence of multicollinearity. Violations of these assumptions may require additional steps, such as transforming variables or considering alternative regression techniques.

Multiple linear regression is widely used in various fields, including social sciences, economics, finance, marketing, and healthcare, to analyze complex relationships and make predictions based on multiple independent variables.

[16]

3.3.3 Polynomial regression

Polynomial regression is a regression technique used to model and identify nonlinear relationships between dependent and independent variables. While simple linear regression assumes a linear relationship, polynomial regression allows for more flexible modeling by introducing polynomial terms. In polynomial regression, the relationship between the dependent variable and independent variable(s) is represented by a polynomial equation. The equation takes the form:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_n X^n + \varepsilon$$

Here, Y represents the dependent variable, X represents the independent variable, β_0, β_1, β_2, ..., β_n are the coefficients, X^2, X^3, ..., X^n are the polynomial terms of X up to the nth degree, and ε represents the error term.

By including polynomial terms in the equation, polynomial regression can capture and model nonlinear relationships that cannot be adequately represented by a straight line. The degree of the polynomial (n) determines the complexity of the model and the number of bends or curves it can accommodate. The coefficients (β_0, β_1, β_2, ..., β_n) in the equation are estimated using techniques such as least squares, which minimizes the sum of the squared differences between the observed and predicted values. These coefficients indicate the contribution and direction of each polynomial term in the model.[16]

Polynomial regression can be beneficial when the relationship between the variables is expected to exhibit curvature or when simple linear regression fails to capture the underlying pattern. However, it is important to avoid overfitting the data by selecting an appropriate degree of the polynomial. Overfitting occurs when the model fits the training data too closely but performs poorly on new, unseen data. Polynomial regression is widely used in various fields, including physics, biology, finance, and engineering, where nonlinear relationships are prevalent. It allows for more flexible modeling and provides a better fit for complex data patterns, enabling researchers and analysts to gain insights and make predictions beyond the constraints of linear relationships.[16]

The efficiency of a model is often assessed using some measure of accuracy when it predicts a numeric outcome. Yet, there are other metrics for gauging accuracy, each with subtle differences. Relying only on one statistic makes it difficult to grasp a model's benefits and drawbacks. Understanding if the model is suitable for purpose requires looking at visualizations of the model fit, particularly residual plots. This chapter talks about these methods.[13]

3.3.4 Regression vs Classification

The main distinction between classification and regression is that although classification aids in the prediction of discrete class labels, regression assists in the prediction of continuous quantities. The two categories of machine learning algorithms also have certain similarities. An integer-formatted discrete value can be predicted using a regression technique. A continuous value can be predicted by a classification method as a class label probability. [16]

For example, a dataset that includes student data from a specific university. Each student's height may be predicted in this situation using a regression algorithm based on factors including weight, gender, food, and field of study. As height is a continuous variable, regression is used in this situation. The height of a person can have any number of different values. [16]

On the other hand, classification may be used to determine whether or not an email is spam. To determine the likelihood that an email is spam, the algorithm examines the sender's address and the email's keywords. A classification method may be used to identify whether it will be cold or hot based on the provided temperature measurements, similar to how a regression model can be used to forecast the temperature for the following day.[16]

4.Application of PM in Bearing Failure

4.1 Predictive failure models /Remaining Useful Life (RUL)

Without a doubt one of the most important values in predictive maintenance is Remaining Useful Life. Remaining Useful Life (RUL), estimates the time a machine is able to operate before it fails. This way, the maintenance plan can be scheduled in order to ensure the machine's efficient operation and avoid any possible malfunctions and minimize downtime. Depending on the available data, RUL can be calculated with various methods, by comparing its lifetime data to similar machine's lifetime, by comparing it to Run-to-failure histories of similar machines and by comparing it to an indicator's threshold value that detects failure

The survival function plot in Figure 4-1 shows the probability that a battery will fail based on how long it has been in operation. The plot shows, for example, that if the battery is in operation for 75 cycles, it has a 90% chance of being at the end of its life time.[19]

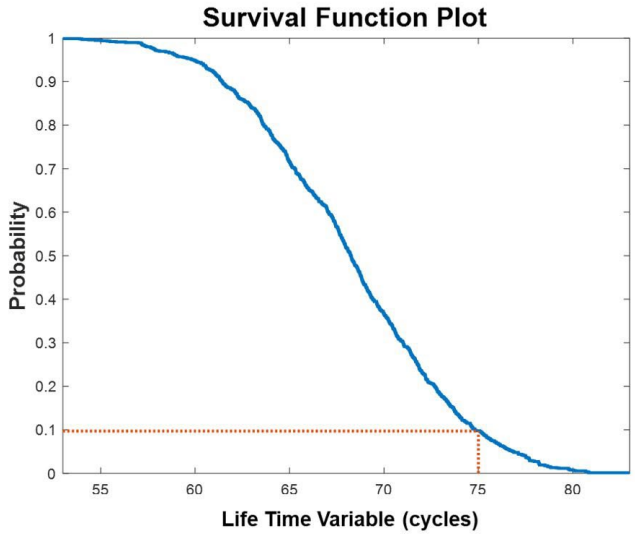

Figure 4-1 Survivor Function plot of a battery [19]

In the figure 4-2, the degradation profiles of historical run-to-failure data sets from an engine are shown in blue and the current data from the engine is shown in red. Based on the profile the engine most closely matches, the RUL is estimated to be around 65 cycles.[19]

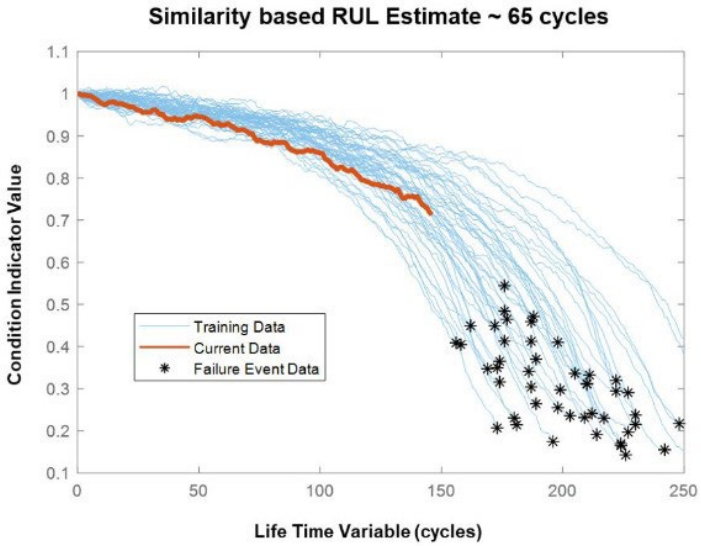

Figure 4-2 Similarity based plot of an engine [19]

Figure 4-3 shows an exponential degradation model that tracks failure in a high-speed bearing used in a wind turbine. The condition indicator is shown in blue. The degradation model predicts that the bearing will cross the threshold value in approximately 9.5 days. The region shaded in red represents the confidence bounds for this prediction.[19]

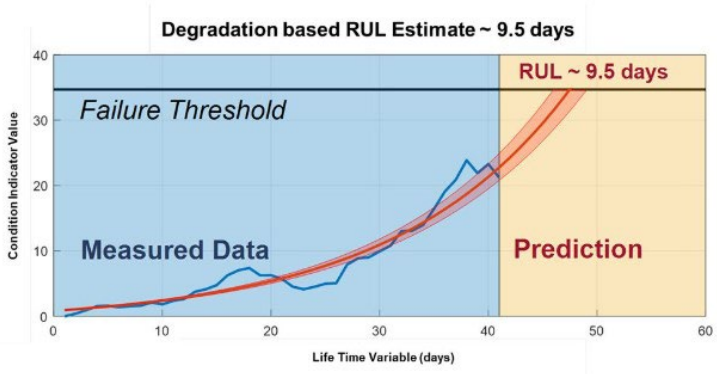

Figure 4-3 Degradation plot of a High-speed bearing in a wind turbine [19]

4.2 Application of Feature Engineering

As it was explained thoroughly before, for the calculation of the remaining useful life of a component and any predictive maintenance technique, the raw data have to be properly processed to be in a manageable from. That includes both, doing the proper mathematical transformations needed and selecting the predictors with the biggest merit.

The dataset is collected from a 2MW wind turbine high-speed shaft driven by a 20-tooth pinion gear. A vibration signal of 6 seconds was acquired each day for 50 consecutive days. [20]

The dataset is available on:

https://github.com/mathworks/WindTurbineHighSpeedBearingPrognosis-Data

First of all, plotting the data in the time domain gives as an initial impression about them. From the figure 4-4, the vibration signals from the wind turbine bearing dataset exhibit a noticeable increasing trend in signal impulsiveness when analyzed in the time domain. This implies that the signals are becoming more impulsive or abrupt over time. To quantify the impulsiveness of these signals and potentially use them as prognostic features, various indicators can be employed. These indicators provide numerical measures that capture different aspects of impulsiveness in the signal. [20],[21]

One such indicator is kurtosis, which measures the heaviness of the tails of a distribution. A higher kurtosis value indicates a more impulsive or heavy-tailed distribution, suggesting the presence of abrupt changes or extreme values in the signal. Another indicator is the peak-to-peak value, which measures the difference between the highest and lowest points in a signal. A larger peak-to-peak value suggests greater variations or spikes in the signal, indicating increased impulsiveness. Crest factors can also serve as useful indicators of impulsiveness. The crest factor represents the ratio of the peak amplitude of a signal to its root mean square (RMS) value. A higher crest factor indicates sharper peaks and a more impulsive signal. By calculating these indicators for the wind turbine bearing dataset, researchers and analysts can obtain quantitative measures of impulsiveness that can be used as prognostic features. These features can provide valuable insights into the condition and health of the bearings, allowing for the detection of potential faults or degradation over time. Machine learning algorithms can be trained on these features to develop predictive models for bearing performance and remaining useful life estimation, aiding in maintenance planning and preventing unexpected failures.[20],[21]

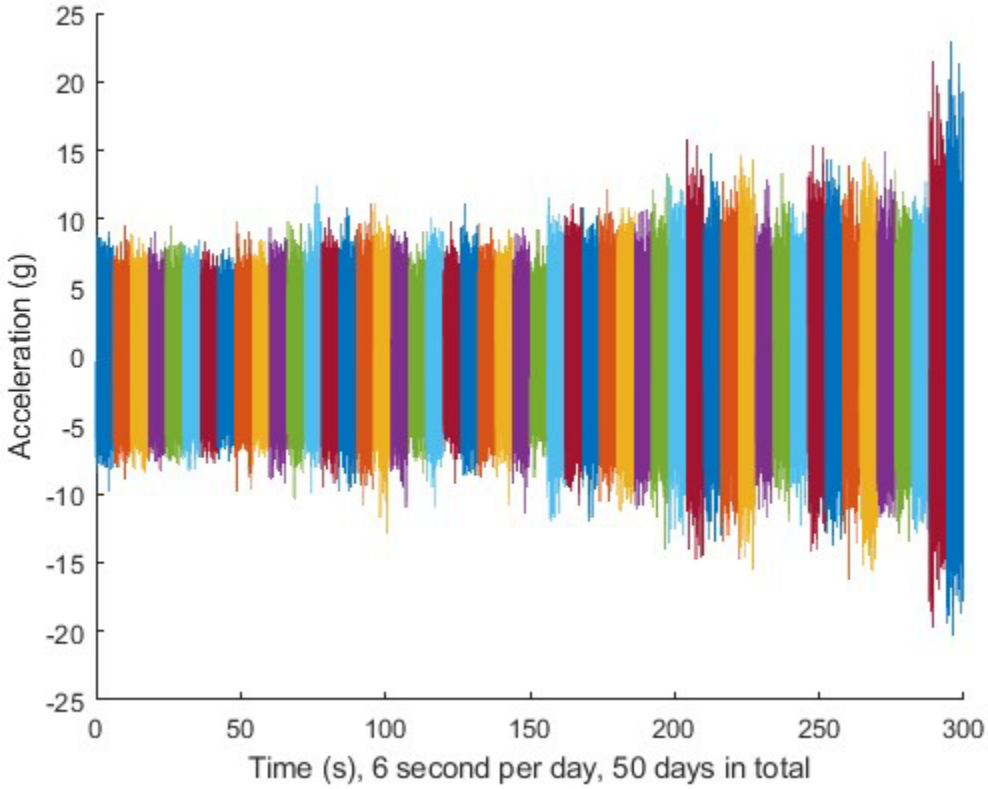

Figure 4-4 Time domain representation of the collected data [20]

Spectral kurtosis is considered a powerful tool for wind turbine prognosis in frequency domain. To visualize the changes in spectral kurtosis along time in the frequency domain, a plot can be created showing the spectral kurtosis values as a function of frequency and the measurement day.[20],[21]

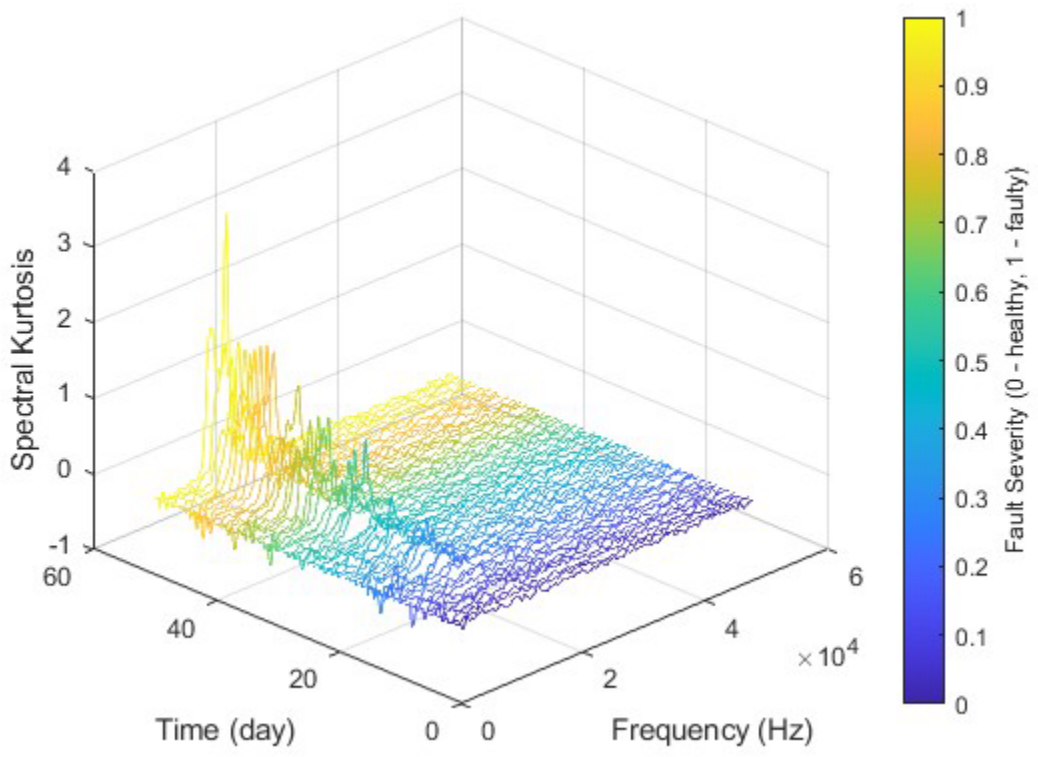

Figure 4-5 Frequency domain representation of Spectral Kurtosis [20]

It is obvious from the figure 4-5, that an increasing mean value of spectral kurtosis at a specific frequency, such as around 10 kHz, suggests a gradual deterioration of the bearing condition as the machine operates.

To mitigate the potential impact of noise and improve the robustness of the extracted features for remaining useful life (RUL) prediction, a causal moving mean filter is applied. The purpose of this filter is to smooth out the noise while preserving the trend and important information in the features. The causal moving mean filter operates by calculating the mean value of a lag window of 5 steps for each data point in the extracted features. The term 'causal' indicates that only past or current values are used in the filtering process, ensuring that no future values are included. This approach prevents any leakage of future information into the filtering process, maintaining the integrity of the time series data. By applying the causal moving mean filter, the noise with an opposite trend that could potentially harm RUL prediction is reduced. The filter effectively attenuates high-frequency variations or outliers that may introduce undesired fluctuations or distortions in the features. It is important to note that the choice of the lag window size (in this case, 5 steps) should be determined based on the characteristics of the data and the desired level

of smoothing. A larger lag window can provide a smoother output but may also introduce more delay in capturing changes in the data. The results of smoothing are visible in the figure 4-6. [20],[21]

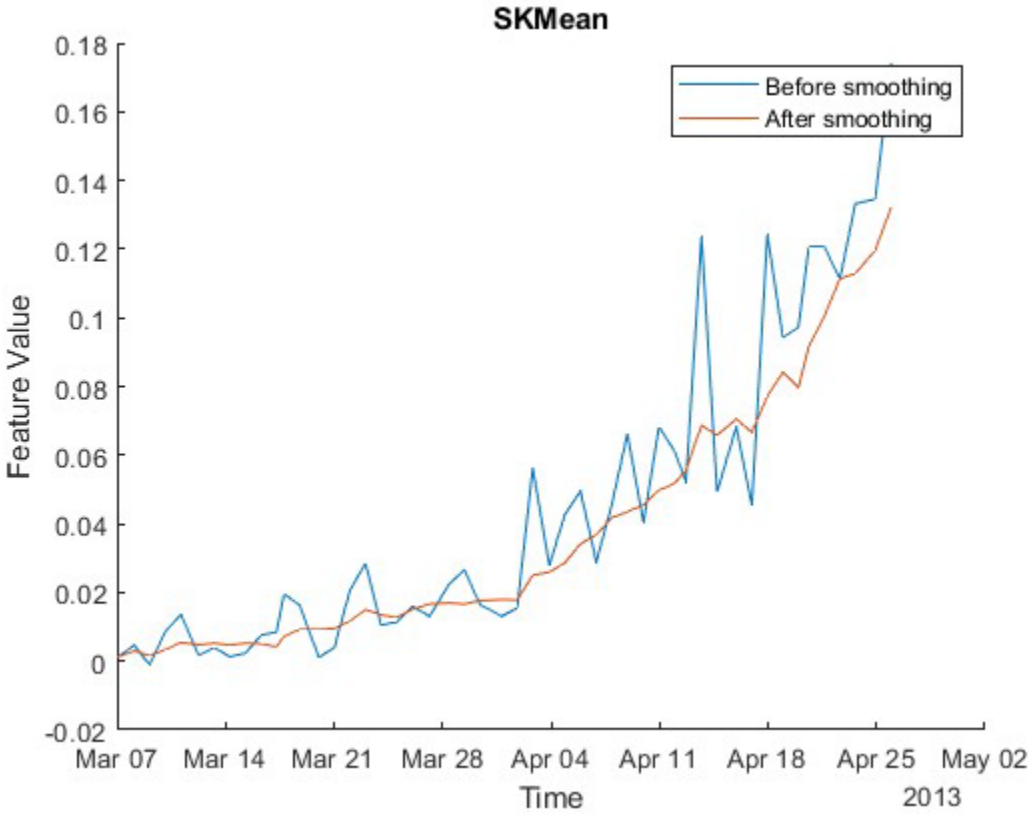

Figure 4-6 Example of smoothing of SKMean [20]

The application of the causal moving mean filter helps enhance the performance of feature-based analysis, particularly for metrics like monotonicity. Monotonicity, which measures the trend direction of a feature, may be adversely affected by noise. Smoothing the data with the causal moving mean filter improves the robustness of the monotonicity metric, allowing for more accurate assessment of the trend direction and aiding in the prediction of the remaining useful life of the system. In addition to visualizing the changes in spectral kurtosis over time, statistical features derived from the spectral kurtosis can serve as potential indicators of bearing degradation in the wind turbine system. Common statistical features, such as mean, standard deviation, and other descriptive statistics, can be calculated from the spectral kurtosis values at each frequency and measurement day. These features provide quantitative measures that summarize the distribution and behavior of the spectral kurtosis over time. By analyzing these statistical features, trends and patterns can be identified that are indicative of bearing degradation. The standard deviation of spectral kurtosis values can also provide valuable information. An increasing standard deviation may indicate higher variability or inconsistency in

the spectral kurtosis, potentially reflecting irregularities or changes in the bearing condition.[20],[21]

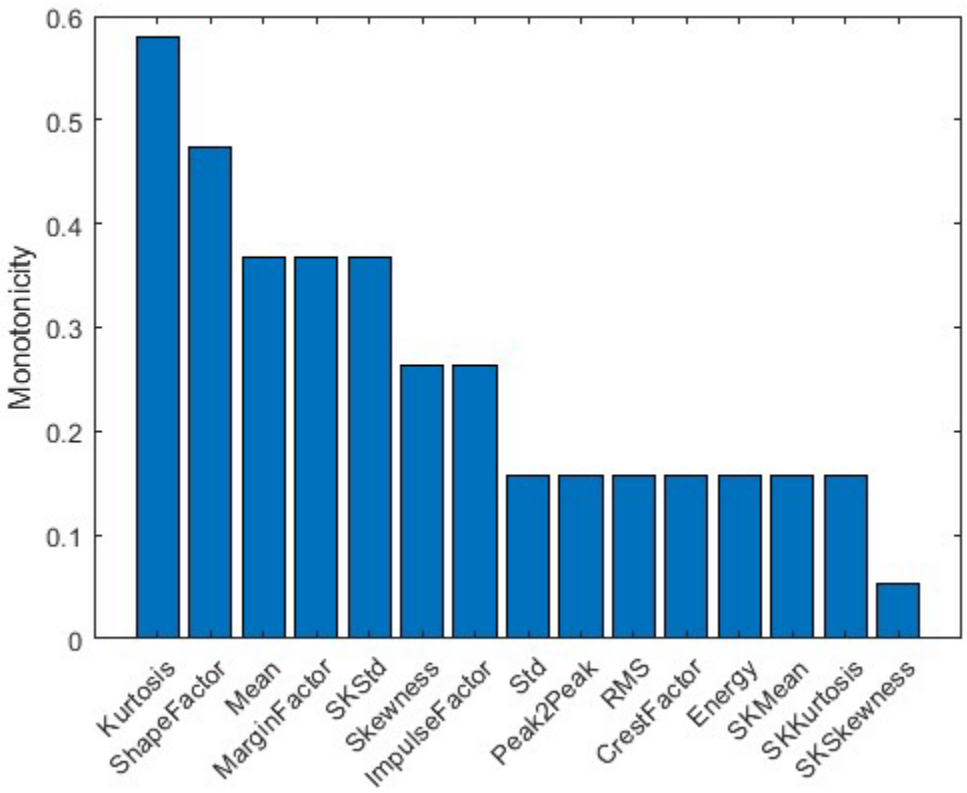

Figure 4-7 Monotonicity of the extracted features [20]

In this example, Principal Component Analysis (PCA) is utilized for dimension reduction and feature fusion. The chosen features are selected from the figure 4-7 and they are those with monotonicity higher than 0.3. However, prior to performing PCA, it is important to normalize the features to ensure they are on the same scale. Normalization helps prevent any bias or undue influence from features with larger magnitudes. A good practice is to normalize the features using the mean and standard deviation obtained from the training data. By subtracting the mean and dividing by the standard deviation, the features are centered around zero with a standard deviation of one. This normalization process ensures that each feature contributes equally during the PCA analysis. Additionally, PCA coefficients, which represent the weights or loadings of each original feature in the principal components, are calculated based on the training data. These coefficients capture the relationship and importance of each feature in the dimension reduction process. It is worth noting that the normalization and PCA steps are applied to the entire dataset together to maintain the relationship between features and capture the most important information during dimension reduction. This consistent processing across the entire dataset

helps maintain the integrity and coherence of the data, ensuring reliable and meaningful results in subsequent analysis or prediction tasks.[20],[21]

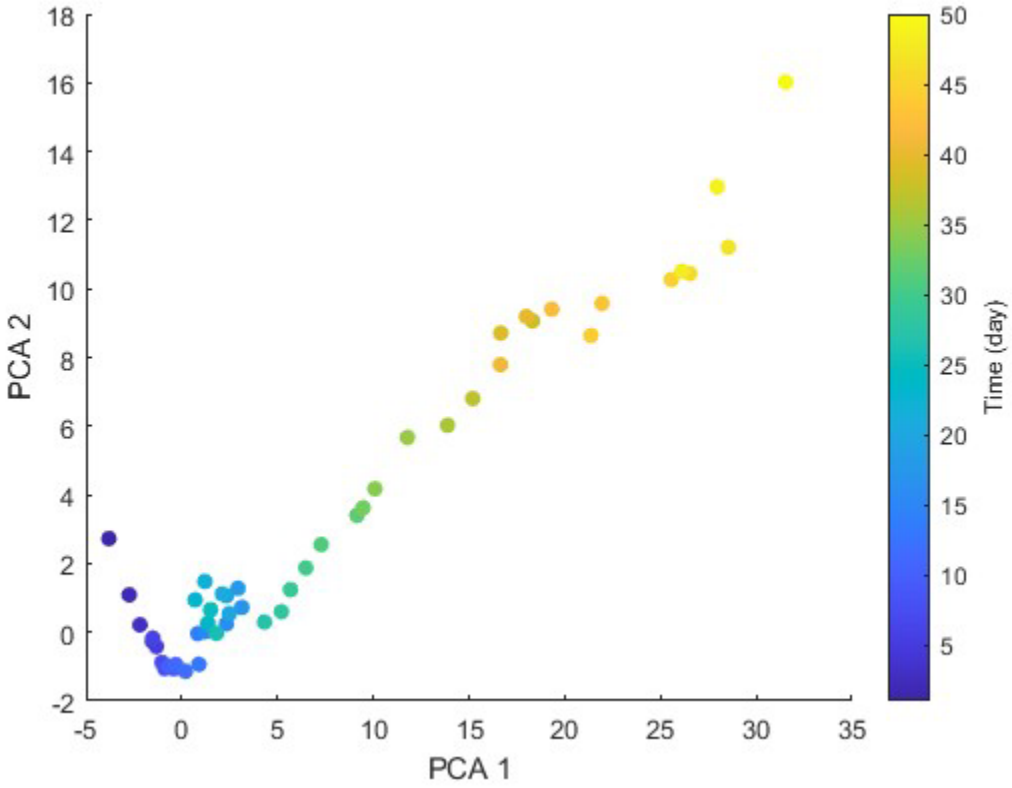

Figure 4-8 PCA plot [20]

It is obvious from the figure 4-8 that PCA 1 can properly describe the deterioration of the ball bearing because its values increase with the same rhythm as the degradation process advances. Thus PCA 1 can be used as a health indicator for the ball bearing. [20],[21]

4.3 Classification approach

In order to predict the remaining useful life of the ball bearing, two different classification algorithms were used. In this case, a specific value is not predicted, but the health indicator data is divided in different classes and each class represents a state of the health of the ball bearing. The two algorithms that were used, are the Support Vector Machines and the K-Nearest Neighbours and the data was divided in four classes, which are: 'good condition', 'medium condition', 'bad condition repair soon' and 'repair asap'. The thresholds that were chosen to divide the classes, were a percentage of the maximum value of the health indicator. The threshold of the "good condition" class is 45% of the maximum value of the health indicator, the threshold of

the "medium condition" class is 70% of the maximum value of the health indicator, the threshold of the "bad condition repair soon" class is 86% of the maximum value of the health indicator and above that value the class is "repair asap". First, the Support Vector Machines algorithm was made with the 'fitcecoc' function using MATLAB's Statistics and Machine Learning Toolbox that is used for training multi-class classification models using the Error Correcting Output Codes (ECOC) framework. The ECOC framework is a strategy for decomposing a multi-class classification problem into multiple binary classification subproblems

The results of the SVM algorithm were:

```
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'good condition'           }
{'medium condition'         }
{'medium condition'         }
{'medium condition'         }
{'medium condition'         }
{'medium condition'         }
{'medium condition'         }
{'medium condition'         }
{'bad condition repair soon'}
{'bad condition repair soon'}
{'bad condition repair soon'}
{'bad condition repair soon'}
{'repair asap'              }
{'bad condition repair soon'}
{'repair asap'              }
{'repair asap'              }
```

Figure 4-9 Results of the SVM algorithm

The K nearest neighbours algorithm was made with the 'fitcknn' function which is a function in MATLAB's Statistics and Machine Learning Toolbox that is used for training a k-nearest neighbours classifier

The results of the KNN algorithm were:

```
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
good condition
medium condition
medium condition
medium condition
medium condition
medium condition
medium condition
medium condition
bad condition repair soon
bad condition repair soon
bad condition repair soon
bad condition repair soon
repair asap
bad condition repair soon
repair asap
repair asap
```

Figure 4-10 Results of the KNN algorithm

4.4 Degradation model approach

As products are increasingly designed to have higher reliability and developed within shorter timeframes, it often becomes impractical to test new designs until they fail under normal operating conditions. In such cases, it is sometimes possible to estimate the reliability of unfailed test samples by relying on the accumulated test time data and making assumptions about the distribution. However, this approach typically introduces a significant level of uncertainty in the results. An alternative option in this scenario is to employ degradation analysis. Degradation analysis entails measuring performance data that directly correlates with the anticipated failure of the product in question. Numerous failure mechanisms can be directly linked to the degradation of specific product components, and by analyzing the degradation over time, analysts can extrapolate an estimated failure time. [22]

In certain scenarios, it is feasible to directly assess the decline of a physical attribute over time, such as the wear of brake pads, the growth of a crack, or the deterioration of performance characteristics like battery voltage or the luminous flux of an LED bulb. These instances fall into the category of Non-Destructive Degradation Analysis. Conversely, in other situations, direct measurement of degradation may not be viable without invasive or destructive techniques that would impact the product's subsequent performance. Therefore, only a single degradation measurement can be obtained. For instance, measuring corrosion in a chemical container or evaluating the strength of an adhesive bond. These cases fall into the category of Destructive Degradation Analysis. Nevertheless, in both cases, it remains crucial to establish a threshold level of degradation or performance that indicates failure. [22]

The analysis of Non-Destructive Degradation pertains to testing situations where multiple degradation measurements for each sample can be taken over time. To determine the appropriate failure level or degradation point leading to failure, basic mathematical models are utilized to extrapolate each sample's degradation measurements towards the point in time when failure is expected to occur. The extrapolated failure times are subsequently analyzed in the same manner as conventional time-to-failure data, as the number of samples being tested increases, so the level of confidence in the results increases . Following the recording of degradation information, the next step involves aligning the measurements with the defined failure level so as to estimate the failure time. [22]

The exponential model is a widely used methodology in model-based studies. Originally, a Bayesian approach updated the model parameters, allowing for measured information integration. There have been various modifications and applications of the exponential model in health management and Remaining Useful Life (RUL) prediction. An improvement to the model involves updating parameters with multiple historical degradation signals acquired through condition monitoring. This integration leads to a more comprehensive analysis and model refinement. Researchers have also combined Bayesian updating with the Expectation Maximization (EM) algorithm to efficiently estimate the model parameters. This integration provides a closed-form RUL distribution, offering valuable insights into the remaining useful life of the system. [22]

The exponential model is effective in predicting RUL for systems with exponential like degradation processes, making it a versatile and robust tool in multiple industries where accurate RUL prediction is critical for maintenance planning and decision-making. [22]

A comparative study was conducted to analyze the efficacy of sensorless and sensor-rich strategies in Prognostics health management for ball screw systems. The researchers aimed to achieve early diagnosis, health assessment, and remaining useful life (RUL) prediction of the ball screw. The findings of their investigation revealed the significant value of torque signals in fault diagnosis and the identification of incipient failures, while the vibration signals exhibited a clear exponential degradation trend in the system. [23],[24],[25]

Upon determining the degradation behavior, through the trend of the Health Indicator, the proposed method focused on estimating the RUL using an exponential model. This model effectively captured the time evolution of the Health Indicator and provided predictions of how long it would last before crossing the Failure Threshold. To construct the exponential degradation model, a Degradation Detection Threshold was established, enabling the prediction of future values of the Health Indicator. This model characterized the degradation behavior as an exponential stochastic process with an offset term. A comprehensive degradation model typically comprises both stochastic and deterministic components. The stochastic part accounts for the variation in the degradation process, while the deterministic part represents a constant physical phenomenon. [23],[24],[25]

4.4.1 Exponential degradation model

It was proven above that the model that best describes the degradation process of a ball bearing is an exponential degradation model. In further detail, the function that can calculate the health indicator of the bearing in the most efficient way is

$$h(t) = \phi + \theta \exp\left(\beta t + \epsilon - \frac{\sigma^2}{2}\right)$$

where $h(t)$ is the health indicator as a function of time. ϕ is the intercept term considered as a constant. θ and β are random parameters determining the slope of the model, where θ is lognormal-distributed and β is Gaussian-distributed. At each time step t, the distribution of θ and β is updated to the posterior based on the latest observation of $h(t)$. ϵ is a Gaussian white noise yielding to $N(0,\sigma^2)$. The $-\frac{\sigma^2}{2}$ term in the exponential is to make the expectation of $h(t)$ satisfy. The selection of threshold is usually based on the historical records of the machine or some domain-specific knowledge. Since no historical data is available in this dataset, the last value of the health indicator is chosen as the threshold. In order, to calculate the remaining useful life, the predictive maintenance toolbox in matlab will be used. The prior of the slope parameters

are chosen arbitrarily with large variances ($E(\theta)=1, \text{Var}(\theta)=10^6$, $E(\beta)=1, \text{Var}(\beta)=10^6$) so that the model is mostly relying on the observed data. Based on $E[h(0)] = \phi + E(\theta)$, intercept ϕ is set to -1 so that the model will start from 0 as well. An exponential degradation model is built with these parameters and using the built in functions, of the predictive maintenance toolbox, the remaining useful life is predicted and the figure is created.

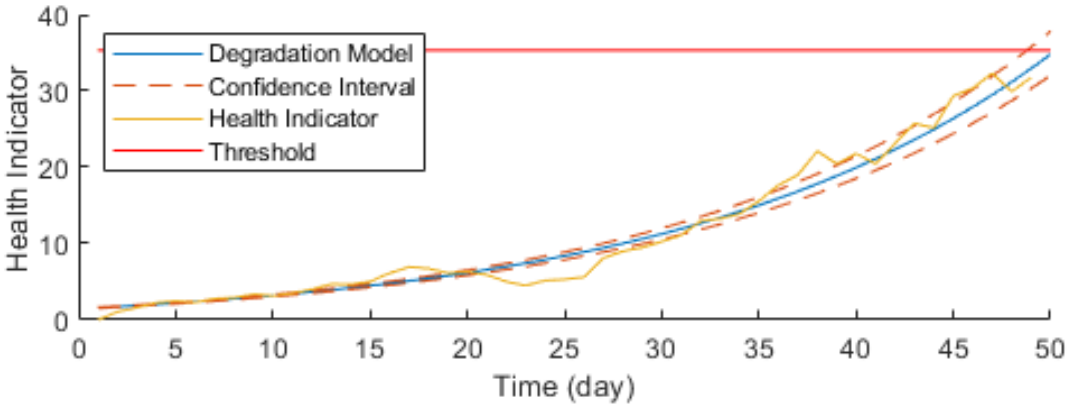

Figure 4-11 Exponential Degradation model plot

4.4.2 Linear Degradation Model

As explained in the previous chapter, the correct degradation model for the prediction of the remaining useful life of a ball bearing, is an exponential degradation model. Though, in some cases a linear model might be implemented to predict the remaining useful life. The main benefits are that the linear model is simpler and it requires less computational time. Using the least squares method, the line that best fits in the data can be found, and the slope and the intercept of the linear model can be estimated. A linear model based on the health indicator data can be constructed in MATLAB and the results are shown in figures 4-12 and 4-13

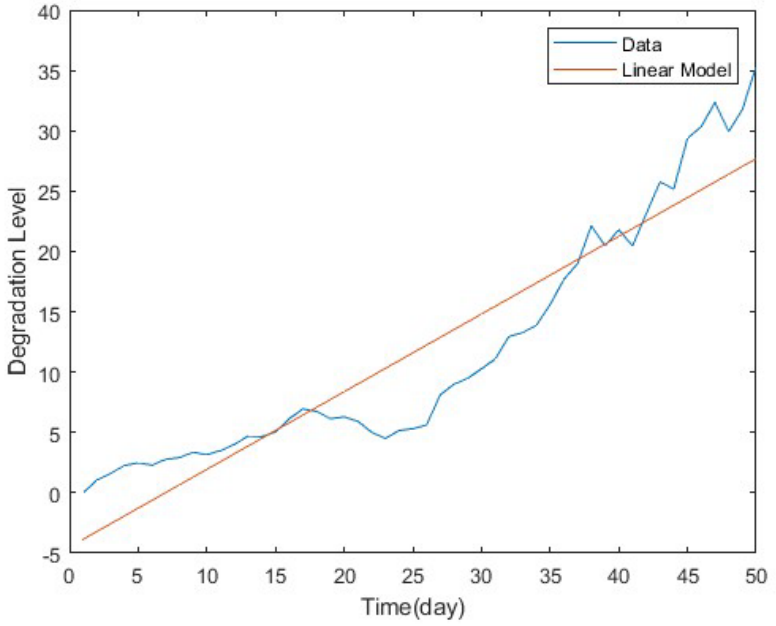

Figure 4-12 Plot of the linear degradation model

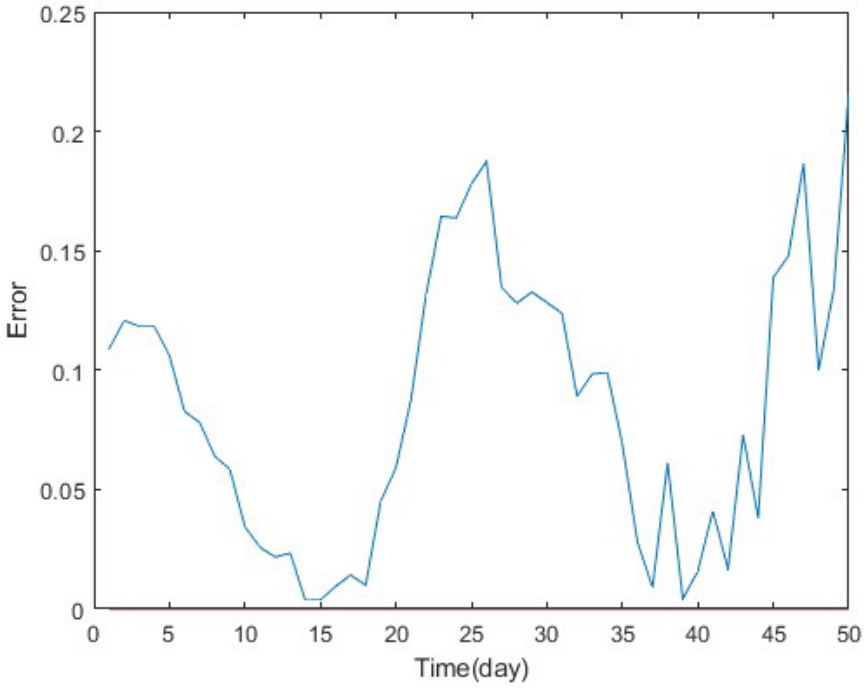

Figure 4-13 Plot of the Error in the Linear degradation model

4.5 Real time update approach using Kalman Filter

Modern systems typically incorporate multiple sensors to estimate hidden states based on a series of measurements. For instance, a GPS receiver estimates the location and velocity, with these being the hidden states, while the differential time of arrival of satellite signals serves as the measurements.[26]

One of the major challenges in tracking and control systems is achieving accurate and precise estimation of hidden states amid uncertainty. In GPS receivers, the uncertainty in measurements arises from various external factors such as thermal noise, atmospheric effects, slight variations in satellite positions, receiver clock precision, and more. To address this challenge, the Kalman Filter has emerged as a fundamental and widely used estimation algorithm.[26]

The Kalman Filter generates estimations of hidden variables based on imperfect and uncertain measurements. It also predicts future system states based on previous estimations. The filter derives its name from Rudolf E. Kálmán (May 19, 1930 – July 2, 2016), who introduced a recursive solution to the discrete-data linear filtering problem in his influential paper published in 1960. Today, the Kalman Filter finds applications in target tracking (radar), location and navigation systems, control systems, computer graphics, and numerous other domains. [26]

It is possible that RUL can be updated on real time by implementing the Kalman filter. In order to use Kalman filter, a series of equations is needed, for the construction of the algorithm. The variables of the equations are, 'X' which is the state vector, 'P' which is the estimate covariance and 'K' which is the Kalman Gain. The algorithm consists of the repetition of two steps, the 'Prediction' and the 'Correction', during the Prediction step the time update equations are used and in the Correction. step the equations used are the measurement update.

During the Prediction step the state extrapolation equation is used to predict the state in the next step and then the estimate covariance is predicted. In order to correct the values from the previous step the Kalman gain is calculated. Then the state vector is corrected and last the estimate covariance is corrected. For the next iteration the same cycle starts from the beginning of the prediction step until the end. In order to initiate the process, the values needed are, the initial state vector 'X' and the initial estimate covariance 'P'. In some cases, if the initial values are unknown, they can be chosen randomly and the Kalman filter, since it is a repetitive process, will converge on the correct values but with more iterations.[26],[27]

Figure 4-14 below contains all the variables used for the calculations.

Term	Name	Alternative term	Dimensions
$\boldsymbol{x}$	State Vector		$n_x \times 1$
$\boldsymbol{z}$	Measurements Vector	$\boldsymbol{y}$	$n_z \times 1$
$\boldsymbol{F}$	State Transition Matrix	$\boldsymbol{\Phi}, \boldsymbol{A}$	$n_x \times n_x$
$\boldsymbol{u}$	Input Variable		$n_u \times 1$
$\boldsymbol{G}$	Control Matrix	$\boldsymbol{B}$	$n_x \times n_u$
$\boldsymbol{P}$	Estimate Covariance	$\boldsymbol{\Sigma}$	$n_x \times n_x$
$\boldsymbol{Q}$	Process Noise Covariance		$n_x \times n_x$
$\boldsymbol{R}$	Measurement Covariance		$n_z \times n_z$
$\boldsymbol{w}$	Process Noise Vector	$\boldsymbol{y}$	$n_x \times 1$
$\boldsymbol{v}$	Measurement Noise Vector		$n_z \times 1$
$\boldsymbol{H}$	Observation Matrix	$\boldsymbol{C}$	$n_z \times n_x$
$\boldsymbol{K}$	Kalman Gain		$n_x \times n_z$
$\boldsymbol{n}$	Discrete-Time Index	$\boldsymbol{k}$	

Figure 4-14 Variables used in a Kalman Filter [26]

Figure 4-15 contains all the equations used in a Kalman filter

	Equation	Equation Name
Predict	$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	State Extrapolation
	$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	Covariance Extrapolation
Update (correction)	$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$	State Update
	$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	Covariance Update
	$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$	Kalman Gain
Auxiliary	$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$	Measurement Equation
	$\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$	Measurement Covariance
	$\boldsymbol{Q}_n = E\left(\boldsymbol{w}_n\boldsymbol{w}_n^T\right)$	Process Noise Covariance
	$\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right) = E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)^T\right)$	Estimation Covariance

Figure 4-15 Equations used in a Kalman Filter [26]

The whole calculation process can be summarized in the figure below.

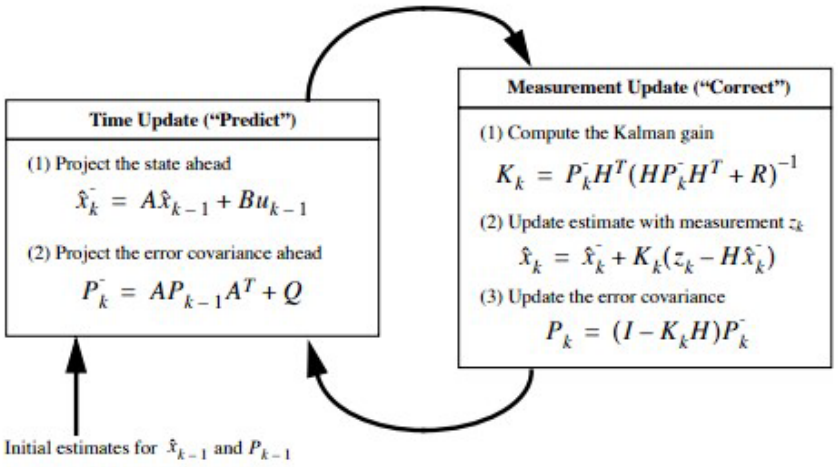

Figure 4-16 A complete figure of the operation of the Kalman filter [27]

In order to implement the Kalman filter the equations above have to be adapted to the known Degradation models.

The degradation model that describes the health indicator 'h' is the following [20]:

$$h(t) = \Phi + \theta exp(\beta t + \varepsilon - \frac{\sigma^2}{2}) \qquad (1)$$

By differentiating, the equation above, with respect to time, the result is this equation:

$$\frac{dh(t)}{dt} = \beta\theta exp(\beta t + \varepsilon - \frac{\sigma^2}{2}) \qquad (2)$$

Equation (2) can be rewritten as

$$h - \Phi = \theta exp(\beta t + \varepsilon - \frac{\sigma^2}{2}) \qquad (3)$$

Thus, the equation (3) is

$$\frac{dh(t)}{dt} = \beta(h - \Phi) \qquad (4)$$

Also

$$\frac{dh(t)}{dt} = \frac{h_{k+1} - h_k}{dt} \qquad (5)$$

In that case dt can be ignored and by combining equations (4) and (5) the result is

$$h_{k+1} - h_k = \beta(h_k - \Phi) \qquad (6)$$

Which is

$$h_{k+1} = h_k(1 + \beta) + \beta\Phi \qquad (7)$$

The two variable β,Φ are parameters that determine some characteristics of the degradation model and they do not have a physical meaning . So, in order to further simplify the last equation, two new variables will be introduced 'a' and 'g'. Where,

$$a = (1 + \beta) \qquad (8)$$

and

$$g = \beta\Phi \qquad (9)$$

By combining equations (7),(8) and (9) the final form of the equation of the health indicator is

$$h_{k+1} = ah_k + g \qquad (10)$$

Last but not least the variables a and g are not constant so their values will be calculated with the equations

$$a_{k+1} = a_k \qquad (11)$$

$$g_{k+1} = g_k \qquad (12)$$

The equations (10),(11) and (12) will be used as the state extrapolation equations in order to create a Kalman filter. The state vector is $x = [\, h; a; g]$

The measurement vector is the health indicator values

The initial values of the state vector are $x_0 = [\, h_0; a_0; g_0] = [0; 1; 1]$

The estimate covariance $P = [10{,}0{,}0\,;\, 0{,}10{,}0\,;\, 0{,}0{,}10]$

The process noise covariance $Q = [5,0,0\,;0,5,0\,;0,0,5]$.

The process noise matrix could be calculated using a theoretical model, but there was not such a model available, so it was estimated intuitively. Using a trial and error method the value that best represents the system was found.

The measurement covariance $R = [155,0,0\,;0,155,0\,;0,0,155]$

The measurements, in this case, the health indicator values are noisy and their covariance was calculated and it is 102. Initially this value was used, but the error of the model was big, so using a trial and error method the value of the measurement covariance was changed.

The initial value of the health indicator $h_0 = 0$ because the health indicator is a measure of how damaged the ball bearing, at the beginning the ball bearing is supposed to be at its best condition. The other values of the state vector and the estimate covariance are chosen randomly and if even if they are far from the correct values the Kalman filter will still operate properly and after some iterations their values will converge on the correct ones. A Kalman filter was constructed, in MATLAB with all the equations above and the results are shown in the figures 4-17 and 4-18 below

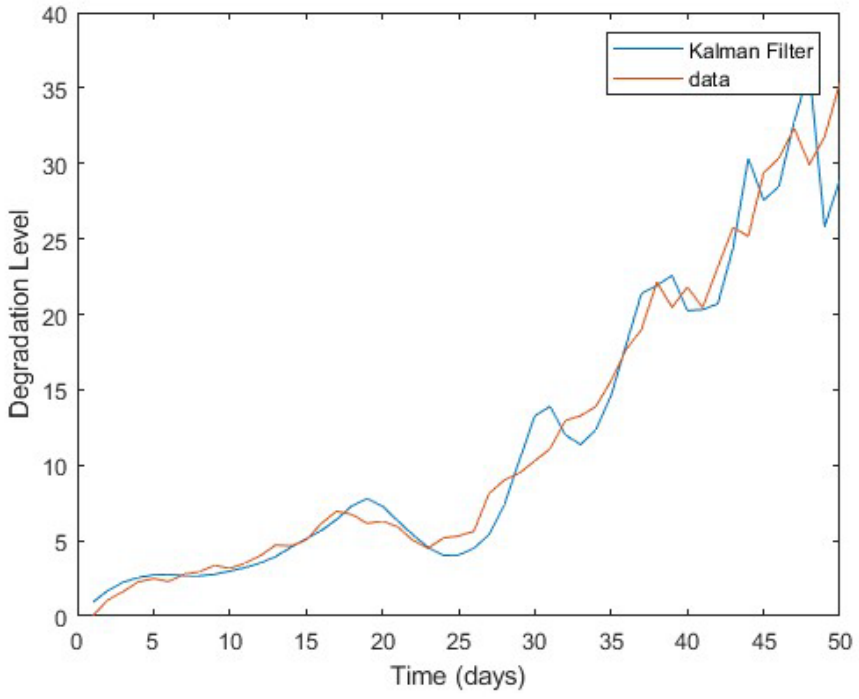

Figure 4-17 Plot of the model made with the Kalman Filter

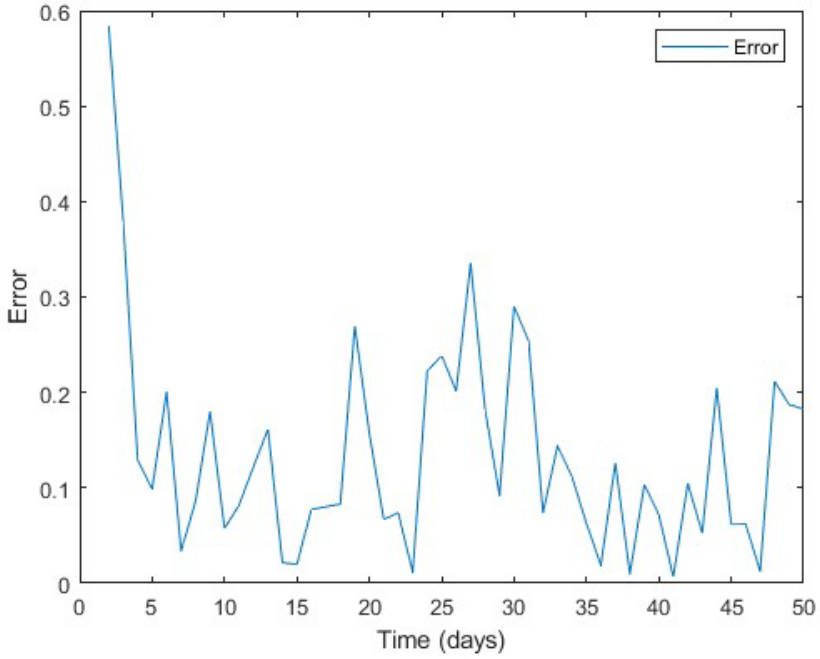

Figure 4-18 Plot of the error of the model made by the Kalman Filter

5.Conclusions and future work

Comparison between approaches

To begin with, the classification approach does not make a prediction about the accurate remaining useful life, but it is a very useful tool because it consults the user of how damaged the ball bearing is. In practice, the user wants to be consulted when to repair the ball bearing in order to avoid any unwanted breakdowns. Thus, using this algorithm, the user knows when the health indicator gets above a certain threshold, which means that it is close to failing so that the user makes any needed actions. The two algorithms give the same results because only on predictor was used. All the other features were rejected during the feature engineering, because they were inaccurate. That means that if any other predictor was added the accuracy of the algorithms would decrease immensely.

The Linear degradation model can be used to predict the remaining useful life, in the specific case, with relatively high accuracy. It is obvious from the first figure that the linear model fits in the data and according to the second figure the error is relatively low. At its maximum, the error is 20% of the maximum value of the health indicator and the mean value of the error is 8.5%. In a situation where great accuracy is not required, a linear model can be implemented, instead of an exponential model, in order to reduce computational time. Moreover, a linear model is simpler than an exponential, thus it is easier to construct and depending on the needed accuracy, it can produce satisfying results.

The exponential degradation model, was proven in the theoretical part, that can accurately predict the remaining useful life of a ball bearing. In the figure 4-11 it is visible that the exponential model fits in the data and can accurately predict the Remaining Useful Life of a ball bearing. The model, is relatively easy to make, because it does not require very complex information about the problem. Also, using the Predictive Maintenance toolbox in MATLAB simplifies greatly the problem because all the calculations and figures are made with the built in functions.

Last but not least, it is proven that the Kalman filter can be implemented in order to update the remaining useful life in real time. The values predicted by the Kalman filter converge on the actual values of the health indicator after few iterations, which means that the initial values that were used in the Kalman filter were accurate. Also, the values of the error have an oscillating behaviour with a steady period and around the same maximum and minimum values. Moreover, the maximum value of the error is 35% of the maximum value of the health indicator and the mean value of the error is 20% of the maximum value. It is possible that, if more accurate information is available about the model, the Kalman filter that is built can produce results with smaller error. Which means that, the filter, will converge on the model by oscillating between maximum and minimum values that are very close to the measurements. Nevertheless, the Kalman filter that was constructed, gave out satisfying and accurate results and it can be used to update in real time, the remaining useful life of a ball bearing.

To conclude, the prediction of the Remaining Useful Life is both very important and very interesting. The importance was greatly analyzed, in the previous chapters and its interest is found in the different methods that can be chosen and how different they operate but still give similar results. Also, the Kalman filter is a very fascinating tool that can be used to solve a vast amount of different problems. Its beauty is hidden in the fact that it can operate even with limited and noisy measurements and still give accurate results.

Directions for future work

The classification algorithms with some adjustments can be implemented to give more accurate results and give answers to more challenging problems. First, in a similar problem where more information is available, there would be more predictors in order to divide the data, thus making the algorithm more complex. Also, more classes could be added, in order to inform the user, with greater detail about the condition of the ball bearing, so that they take immediate action.

The remaining useful life can be updated in real time with greater accuracy by using a more complex Kalman Filter. Specifically, a dual extended Kalman filter would give greatly accurate results, because one filter would be used to simultaneously estimate the parameters and the other the state of the model. Although, in that case more information about the model is required. In another direction, a simple Kalman filter, could be used to predict the remaining useful life with limited data by calculating the health indicator. In the problem that was studied, the value of the error has a repetitive behaviour, similar to oscillation with almost stable period and maximum values and the Kalman gain has a smooth declining behaviour. Thus, the error can be replicated as a function and the gain as another function. These functions can be applied, in the equation for the state vector correction, in order to make a model that predicts the values of the health indicator.

Concluding, predictive maintenance, is a virgin industry and when it is absorbed by the industries, the whole production process will be greatly improved. The whole process will be sped up because of the reduced unwanted downtimes, the products will be cheaper because of the reduced maintenance costs and the quality of the products will be increased because at every point of the production the condition of the machines will be monitored and any unwanted malfunctions will be predicted. Also, there is a plethora of other techniques, that can be implemented, in order to calculate the remaining useful life of a component and predict when maintenance is needed in general. Predictive maintenance is a very interesting object of studies and engineers spend countless hours to improve their knowledge bringing humanity closer to the fifth industrial revolution.

6.Bibliography

1.Protecting Wind-Turbine Bearings, Dayananda Raju,

https://www.windsystemsmag.com/protecting-wind-turbine-bearings/

2.WHITE ETCHING CRACKS – A CONSEQUENCE, NOT A ROOT CAUSE OF BEARING FAILURE, Kenred Stadler, Reinder H. Vegter, David Vaes

https://evolution.skf.com/white-etching-cracks-a-consequence-not-a-root-cause-of-bearing-failure/

3.https://www.skf.com/binaries/pub12/Images/0901d1968064c148-Bearing-failures---14219_2-EN_tcm_12-297619.pdf

4.The leading causes of wind turbine bearing failures, Philipp Schmidd

https://www.linkedin.com/pulse/leading-causes-wind-turbine-bearing-failures-philipp-schmid

5.George Aslanidis, Predictive maintenance in Industry 5.0, UNIVERSITY OF WEST ATTICA,2021

6.Deloitte , Predictive Maintenance Taking pro-active measures based on advanced data analytics to predict and avoid machine failure ,2017

7.Can Predictive Maintenance Protect Your Business?, Ross Mudrick

https://www.businessnewsdaily.com/10920-predictive-maintenance-business.html

8.What is predictive maintenance

https://www.sap.com/insights/what-is-predictive-maintenance.html

9.Predictive Maintenance: What's the Economic value?, Amir Kupervas

https://www.anodot.com/blog/predictive-maintenance/

10.5 advantages of using predictive maintenance in Field Service Management, Eirini Saranti

https://fieldcode.com/en/resources/blog/5-advantages-of-predictive-maintenance-and-how-to-leverage-on-them

11.Stafylas Demetrios, Wildfire Prediction Using Machine Learning, UNIVERSITY OF WEST ATTICA,2022

12.Vafeiadis Vasileios, Using Artificial Intelligence to model rainfall induced damages, UNIVERSITY OF WEST ATTICA,2022

13.Max Kuhn, Kjell Johnson, Applied Predictive Modeling, Springer Science+Business Media, New York 2013, ISBN 978-1-4614-6848-6

14.Fundamental Techniques of Feature Engineering for Machine Learning, Emre Rencberoglu

https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114#3abe

15.What is Feature Engineering - Importance, Tools and Techniques for Machine Learning, Harshil Patel

https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10

16.Regression VS. Classification in Machine Learning: What's the Difference?, Sakshi Gupta

https://www.springboard.com/blog/data-science/regression-vs-classification/

17.Manfredi Manfre, Creation of a Machine Learning model for the Predictive Maintenance of an engine equipped with a rotating shaft, POLITECNICO DI TORINO,2020

18.K-Nearest Neighbor(KNN) Algorithm for Machine Learning

K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint

19.Three Ways to Estimate Remaining Useful Life for Predictive Maintenance, Aditya Baru, Rachel Johnson

Three Ways to Estimate Remaining Useful Life for Predictive Maintenance - MATLAB & Simulink (mathworks.com)

20.Wind Turbine High-Speed Bearing Prognosis

https://www.mathworks.com/help/predmaint/ug/wind-turbine-high-speed-bearing-prognosis.html

21.Mohamad Danish Anis, 'Towards Remaining Useful Life Prediction in Rotating Machine Fault Prognosis: An Exponential Degradation Model', *International Conference on Condition Monitoring and Diagnosis* – Perth – Australia, 2018

22.Degradation Data Analysis

https://reliawiki.org/index.php/Degradation_Data_Analysis

23.Naipeng Li, Yaguo Lei, Jing Lin, Steven X. Ding, 'An Improved Exponential Model for Predicting Remaining Useful Life of Rolling Element Bearings', *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,*2015

24.Islem Bejaoui, Dario Bruneo, Maria Gabriella Xibilia, A Data-Driven Prognostics Technique and RUL Prediction of Rotating Machines Using an Exponential Degradation Model, 7th International Conference on Control, Decision and Information Technologies (CoDIT'20), 2020

25.Pin Li, Xiaodong Jia, Jianshe Feng, Hossein Davari, Guan Qiao, Yihchyun Hwang, Jay Lee, Prognosability study of ball screw degradation using systematic methodology, Mechanical Systems and Signal Processing, 2018

26.Kalman Filter, Alex Becker

https://www.kalmanfilter.net/default.aspx

27.Greg Welch, Gary Bishop, An Introduction to the Kalman Filter, University of North Carolina at Chapel Hill Chapel Hill, NC 27599-3175, 2006

7.Annex

7.1 Support Vector Machines algorithm

```
clc

clear

data = [0; 1.0634698965758003; 1.6198648856633131; 2.2619072656147585;

2.4778802985226918; 2.2943845251715773; 2.7740425761314293;

2.9180203015730326; 3.367736654366372; 3.159961869377379;

3.4998663197029094; 3.998420146295548; 4.7016055877799339;

4.6474796899727275; 5.0296419995753423; 6.13871875384978;

6.9484578507084684; 6.7415437845933468; 6.1400964755969554;

6.2934825206386513; 5.9268511844752494; 5.0148520152985068;

4.5008137340201113; 5.1675760428773723; 5.3093600729497954;

5.6159501705990778; 8.1250934863113446; 9.0049394472183568;

9.4804881601481554; 10.280265933803911; 11.083833378254168;

12.957521954684625; 13.268755548065453; 13.888865664639129;

15.593199227412203; 17.68485152051311; 19.000952262294874;

22.121072555510139; 20.458639240135312; 21.807860406793182;

20.447913063968951; 23.125561318160997; 25.753613005143702;

25.170528605148871; 29.37452173787678; 30.335214444785265;

32.340198658042972; 29.930206533282856; 31.76226266865903;

35.339371181804907];
```

```
max_val = max(data);


labels = cell(length(data),1);
for i=1:length(data)
if data(i) <= 0.45*max_val
labels{i} = 'good condition';
elseif data(i) > 0.45*max_val && data(i) <= 0.70*max_val
labels{i} = 'medium condition';
elseif data(i) > 0.70*max_val && data(i) <= 0.86*max_val
labels{i} = 'bad condition repair soon';
else


labels{i} = 'repair asap';
end
end


svm_model = fitcecoc(data,labels);


predicted_labels = predict(svm_model,data)
```

7.2 K Nearest Neighbours algorithm

```
clc

clear


data = [0; 1.0634698965758003; 1.6198648856633131; 2.2619072656147585;
        2.4778802985226918; 2.2943845251715773; 2.7740425761314293;
        2.9180203015730326; 3.367736654366372; 3.159961869377379;
        3.4998663197029094; 3.998420146295548; 4.7016055877799339;
        4.6474796899727275; 5.0296419995753423; 6.13871875384978;
        6.9484578507084684; 6.7415437845933468; 6.1400964755969554;
        6.2934825206386513; 5.9268511844752494; 5.0148520152985068;
        4.5008137340201113; 5.1675760428773723; 5.3093600729497954;
        5.6159501705990778; 8.1250934863113446; 9.0049394472183568;
        9.4804881601481554; 10.280265933803911; 11.083833378254168;
        12.957521954684625; 13.268755548065453; 13.888865664639129;
        15.593199227412203; 17.68485152051311; 19.000952262294874;
        22.121072555510139; 20.458639240135312; 21.807860406793182;
        20.447913063968951; 23.125561318160997; 25.753613005143702;
        25.170528605148871; 29.37452173787678; 30.335214444785265;
        32.340198658042972; 29.930206533282856; 31.76226266865903;
        35.339371181804907];
```

```
% Define the maximum value

maxval = max(data);


% Define the thresholds for each class

good_thres = 0.45*maxval;

medium_thres = 0.70*maxval;

bad_thres = 0.86*maxval;


% Initialize labels

labels = cell(size(data));


% Classify data

for i = 1:length(data)
    if data(i) <= good_thres
        labels{i} = 'good condition';
    elseif data(i) <= medium_thres
        labels{i} = 'medium condition';
    elseif data(i) <= bad_thres
        labels{i} = 'bad condition repair soon';
    else
        labels{i} = 'repair asap';
    end
```

```
end

% Prepare input and output variables for k-nearest neighbor algorithm
inputs = data;
outputs = categorical(labels);

% Create and train the k-nearest neighbor algorithm model
k = 3; % Number of neighbors to consider
mdl = fitcknn(inputs,outputs,'NumNeighbors',k);

% Test the model on the same input data
predictions = predict(mdl,inputs)

% Compare the predicted labels to the true labels
accuracy = sum(predictions == outputs)/length(outputs)
```

7.3 Linear degradation model

```
clc

clear

data = [0; 1.0634698965758003; 1.6198648856633131; 2.2619072656147585;

2.4778802985226918; 2.2943845251715773; 2.7740425761314293;

2.9180203015730326; 3.367736654366372; 3.159961869377379;

3.4998663197029094; 3.998420146295548; 4.7016055877799339;

4.6474796899727275; 5.0296419995753423; 6.13871875384978;

6.9484578507084684; 6.7415437845933468; 6.1400964755969554;

6.2934825206386513; 5.9268511844752494; 5.0148520152985068;

4.5008137340201113; 5.1675760428773723; 5.3093600729497954;

5.6159501705990778; 8.1250934863113446; 9.0049394472183568;

9.4804881601481554; 10.280265933803911; 11.083833378254168;

12.957521954684625; 13.268755548065453; 13.888865664639129;

15.593199227412203; 17.68485152051311; 19.000952262294874;

22.121072555510139; 20.458639240135312; 21.807860406793182;

20.447913063968951; 23.125561318160997; 25.753613005143702;

25.170528605148871; 29.37452173787678; 30.335214444785265;

32.340198658042972; 29.930206533282856; 31.76226266865903;

35.339371181804907];


x = 1:length(data);
```

```
A = [x', ones(length(x),1)];

b = data;

lin_params = A \ b;


m = lin_params(1) % slope

b = lin_params(2) % intercept


y = m*x + b;

figure

plot(x,data,x,y)

xlabel('Time(day)')

ylabel('Degradation Level')

legend('Data', 'Linear Model')

Y = y';


Err = abs((Y-data)/data);

figure

plot(Err)

xlabel('Time(day)')

ylabel('Error')

mean(Err)
```

7.4 Kalman Filter

```
clear

clc

% times tou health indicator pou prokuptoun apo feature engineering

data = [0; 1.0634698965758003; 1.6198648856633131; 2.2619072656147585;

2.4778802985226918; 2.2943845251715773; 2.7740425761314293;

2.9180203015730326; 3.367736654366372; 3.159961869377379;

3.4998663197029094; 3.998420146295548; 4.7016055877799339;

4.6474796899727275; 5.0296419995753423; 6.13871875384978;

6.9484578507084684; 6.7415437845933468; 6.1400964755969554;

6.2934825206386513; 5.9268511844752494; 5.0148520152985068;

4.5008137340201113; 5.1675760428773723; 5.3093600729497954;

5.6159501705990778; 8.1250934863113446; 9.0049394472183568;

9.4804881601481554; 10.280265933803911; 11.083833378254168;

12.957521954684625; 13.268755548065453; 13.888865664639129;

15.593199227412203; 17.68485152051311; 19.000952262294874;

22.121072555510139; 20.458639240135312; 21.807860406793182;

20.447913063968951; 23.125561318160997; 25.753613005143702;

25.170528605148871; 29.37452173787678; 30.335214444785265;

32.340198658042972; 29.930206533282856; 31.76226266865903;

35.339371181804907];

%initial values
```

```
h = 0;

a = 1;

g = 1;

i = 1;

HI = [];

Err = [];

% eklego times gia P , Q ,R


P = 10*eye(3);

Q = 5*eye(3);

R = 155*eye(3) ;


while i<51

z = data(i)

h = [h 1]*[a;g];

a = a;

g = g;

A = P+Q; % nea timh p

Z = A / (A+R); %Gain

G = Z * [1;1;1]

d1 = (z-h)*G; % ypologismos error*Gain

d = [1 0 0]*d1;

%nees times hat
```

```
h = h+d

g = g+d;

a = a+d;

HI(i) = h;

%ypologismos neou P

D = (eye(3) - G);

P = D.*P.*D' + G.*R.*G';

e = abs((data(i)-h)/data(i))

Err(i)=e;

i = i+1

end


figure

plot(HI)

hold on

plot(data)

hold on

xlabel('Time (days)')

ylabel('Degradation Level')

legend('Kalman Filter', 'data')


figure

plot(Err)
```

```
xlabel('Time (days)')

ylabel('Error')

legend('Error')

mean(Err')
```

7.5 Feature Engineering and Exponential model [20]

7.5.1 Main code [20]

```
timeUnit = 'day';


hsbearing = fileEnsembleDatastore(...
    fullfile('.', 'WindTurbineHighSpeedBearingPrognosis-Data-master'), ...
    '.mat');
hsbearing.DataVariables = ["vibration", "tach"];
hsbearing.IndependentVariables = "Date";
hsbearing.SelectedVariables = ["Date", "vibration", "tach"];
hsbearing.ReadFcn = @helperReadData;
hsbearing.WriteToMemberFcn = @helperWriteToHSBearing;
tall(hsbearing)
fs = 97656; % Hz
reset(hsbearing)
tstart = 0;
figure
hold on
while hasdata(hsbearing)
    data = read(hsbearing);
    v = data.vibration{1};
```

```
    t = tstart + (1:length(v))/fs;
    % Downsample the signal to reduce memory usage
    plot(t(1:10:end), v(1:10:end));
    tstart = t(end);
end
hold off
xlabel('Time (s), 6 second per day, 50 days in total');
ylabel('Acceleration (g)');

hsbearing.DataVariables = ["vibration", "tach", "SpectralKurtosis"];
colors = parula(50);
figure
hold on
reset(hsbearing)
day = 1;
while hasdata(hsbearing)
    data = read(hsbearing);
    data2add = table;

    % Get vibration signal and measurement date
    v = data.vibration{1};

    % Compute spectral kurtosis with window size = 128
```

```
    wc = 128;
    [SK, F] = pkurtosis(v, fs, wc);
    data2add.SpectralKurtosis = {table(F, SK)};

    % Plot the spectral kurtosis
    plot3(F, day*ones(size(F)), SK, 'Color', colors(day, :));

    % Write spectral kurtosis values
    writeToLastMemberRead(hsbearing, data2add);

    % Increment the number of days
    day = day + 1;
end
hold off
xlabel('Frequency (Hz)')
ylabel('Time (day)')
zlabel('Spectral Kurtosis')
grid on
view(-45, 30)
cbar = colorbar;
ylabel(cbar, 'Fault Severity (0 - healthy, 1 - faulty)')
hsbearing.DataVariables = [hsbearing.DataVariables; ...
    "Mean"; "Std"; "Skewness"; "Kurtosis"; "Peak2Peak"; ...
```

```
    "RMS"; "CrestFactor"; "ShapeFactor"; "ImpulseFactor"; "MarginFactor"; "Energy"; ...
    "SKMean"; "SKStd"; "SKSkewness"; "SKKurtosis"];
hsbearing.SelectedVariables = ["vibration", "SpectralKurtosis"];
reset(hsbearing)
while hasdata(hsbearing)
    data = read(hsbearing);
    v = data.vibration{1};
    SK = data.SpectralKurtosis{1}.SK;
    features = table;

    % Time Domain Features
    features.Mean = mean(v);
    features.Std = std(v);
    features.Skewness = skewness(v);
    features.Kurtosis = kurtosis(v);
    features.Peak2Peak = peak2peak(v);
    features.RMS = rms(v);
    features.CrestFactor = max(v)/features.RMS;
    features.ShapeFactor = features.RMS/mean(abs(v));
    features.ImpulseFactor = max(v)/mean(abs(v));
    features.MarginFactor = max(v)/mean(abs(v))^2;
    features.Energy = sum(v.^2);
```

```
    % Spectral Kurtosis related features
    features.SKMean = mean(SK);
    features.SKStd = std(SK);
    features.SKSkewness = skewness(SK);
    features.SKKurtosis = kurtosis(SK);

    % write the derived features to the corresponding file
    writeToLastMemberRead(hsbearing, features);
end
hsbearing.SelectedVariables = ["Date", "Mean", "Std", "Skewness", "Kurtosis", "Peak2Peak", ...
    "RMS", "CrestFactor", "ShapeFactor", "ImpulseFactor", "MarginFactor", "Energy", ...
    "SKMean", "SKStd", "SKSkewness", "SKKurtosis"];
featureTable = gather(tall(hsbearing));
featureTable = table2timetable(featureTable)
variableNames = featureTable.Properties.VariableNames;
featureTableSmooth = varfun(@(x) movmean(x, [5 0]), featureTable);
featureTableSmooth.Properties.VariableNames = variableNames;

%featureTableSmooth = featureTable;
%featureTableSmooth.Properties.VariableNames = variableNames;


figure
```

```
hold on

plot(featureTable.Date, featureTable.SKMean);

plot(featureTableSmooth.Date, featureTableSmooth.SKMean);

hold off

xlabel('Time')

ylabel('Feature Value')

legend('Before smoothing', 'After smoothing')

title('SKMean')

breaktime = datetime(2013, 3, 27);

breakpoint = find(featureTableSmooth.Date < breaktime, 1, 'last');

trainData = featureTableSmooth(1:breakpoint, :);

% Since moving window smoothing is already done, set 'WindowSize' to 0 to

% turn off the smoothing within the function

featureImportance = monotonicity(trainData, 'WindowSize', 0);

helperSortedBarPlot(featureImportance, 'Monotonicity');

trainDataSelected = trainData(:, featureImportance{:,:}>0.3);

featureSelected = featureTableSmooth(:, featureImportance{:,:}>0.3)

meanTrain = mean(trainDataSelected{:,:});

sdTrain = std(trainDataSelected{:,:});

trainDataNormalized = (trainDataSelected{:,:} - meanTrain)./sdTrain;

coef = pca(trainDataNormalized);

PCA1 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 1);

PCA2 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 2);
```

```
%PCA2 = (featureSelected{:,:} - meanTrain) ./ sdTrain * coef(:, 1);

figure

numData = size(featureTable, 1);

scatter(PCA1, PCA2, [], 1:numData, 'filled')

xlabel('PCA 1')

ylabel('PCA 2')

cbar = colorbar;

ylabel(cbar, ['Time (' timeUnit ')'])

healthIndicator = PCA1;

figure

plot(featureSelected.Date, healthIndicator, '-o')

xlabel('Time')

title('Health Indicator')

healthIndicator = healthIndicator - healthIndicator(1);

threshold = healthIndicator(end);

mdl = exponentialDegradationModel(...

    'Theta', 1, ...

    'ThetaVariance', 1e6, ...

    'Beta', 1, ...

    'BetaVariance', 1e6, ...

    'Phi', -1, ...

    'NoiseVariance', (0.1*threshold/(threshold + 1))^2, ...

    'SlopeDetectionLevel', 0.05);
```

```
% Keep records at each iteration
totalDay = length(healthIndicator) - 1;
estRULs = zeros(totalDay, 1);
trueRULs = zeros(totalDay, 1);
CIRULs = zeros(totalDay, 2);
pdfRULs = cell(totalDay, 1);

% Create figures and axes for plot updating
figure
ax1 = subplot(2, 1, 1);
ax2 = subplot(2, 1, 2);

for currentDay = 1:totalDay

    % Update model parameter posterior distribution
    update(mdl, [currentDay healthIndicator(currentDay)])

    % Predict Remaining Useful Life
    [estRUL, CIRUL, pdfRUL] = predictRUL(mdl, ...
                                [currentDay healthIndicator(currentDay)], ...
                                threshold);
    trueRUL = totalDay - currentDay + 1;
```

```
    % Updating RUL distribution plot
    helperPlotTrend(ax1, currentDay, healthIndicator, mdl, threshold, timeUnit);
    helperPlotRUL(ax2, trueRUL, estRUL, CIRUL, pdfRUL, timeUnit)

    % Keep prediction results
    estRULs(currentDay) = estRUL;
    trueRULs(currentDay) = trueRUL;
    CIRULs(currentDay, :) = CIRUL;
    pdfRULs{currentDay} = pdfRUL;

    % Pause 0.1 seconds to make the animation visible
    pause(0.1)
end
```

7.5.2 Helper Functions [20]

7.5.2.1 helperAlphaLambdaPlot [20]

```
function alphaBoundProbablity = helperAlphaLambdaPlot(alpha, trueRULHist, estRULHist, ...
    CIRULHist, pdfRULHist, degradationTime, breakpoint, timeUnit)
%HELPERALPHALAMBDAPLOT create alpha-lambda plot and the probability metric

%  Copyright 2018 The MathWorks, Inc.

N = length(trueRULHist);
t = 1:N;
t2 = t((degradationTime+1):end);

% Compute the alpha bounds
alphaPlus = trueRULHist + alpha*trueRULHist;
alphaMinus = trueRULHist - alpha*trueRULHist;

% ---------------- Alpha-Lambda Plot --------------------
figure
hold on
grid on

% Plot true RUL and its alpha bounds
```

```
plot(t, trueRULHist)
fill([t fliplr(t)], [alphaPlus(t)' fliplr(alphaMinus(t)')], ...
    'b', 'FaceAlpha', 0.2, 'EdgeColor', 'none')

% Plot the estimated RUL and its confidence intervals
plot(t2, estRULHist(t2), '--')
fill([t2 fliplr(t2)], ...
    [CIRULHist(t2, 1)' fliplr(CIRULHist(t2, 2)')], ...
    'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none')

% Plot the train-test breakpoint
ylow = 0;
yup = 80;
plot([breakpoint breakpoint], [ylow yup], 'k-.')

% Add labels and legends
ylim([ylow yup])
hold off
xlabel(['Time (' timeUnit ')'])
ylabel(['RUL (' timeUnit ')'])
legend('True RUL', ['\alpha = +\\-' num2str(alpha*100) '%'], ...
    'Predicted RUL After Degradation Detected', ...
    'Confidence Interval After Degradation Detected', 'Train-Test Breakpoint')
```

```
% ---------------- Probability Metric --------------------

% Compute the probability of predicted RUL within alpha bounds

alphaBoundProbablity = zeros(N, 1);

for i = 1:N

    pdfRUL = pdfRULHist{i};

    idx = (pdfRUL{:, 1} > alphaMinus(i)) & (pdfRUL{:, 1} < alphaPlus(i));

    prob = sum(pdfRUL{idx, 2});

    alphaBoundProbablity(i) = prob;

end

end
```

7.5.2.2 helperPlotRUL [20]

```
function helperPlotRUL(ax, trueRUL, estRUL, CIRUL, pdfRUL, timeUnit)

%HELPERPLOTRULDISTRIBUTION helper function to refresh the distribution plot


%  Copyright 2018 The MathWorks, Inc.

cla(ax)

hold(ax, 'on')

plot(ax, pdfRUL{:,1}, pdfRUL{:,2})

plot(ax, [estRUL estRUL], [0 pdfRUL{find(pdfRUL{:,1} >= estRUL, 1), 2}])

plot(ax, [trueRUL trueRUL], [0 pdfRUL{find(pdfRUL{:,1} >= trueRUL, 1), 2}], '--')

idx = pdfRUL{:,1} >= CIRUL(1) & pdfRUL{:,1}<=CIRUL(2);

area(ax, pdfRUL{idx, 1}, pdfRUL{idx, 2}, ...

    'FaceAlpha', 0.2, 'FaceColor', 'g', 'EdgeColor', 'none');

hold(ax, 'off')

ylabel(ax, 'PDF')

xlabel(ax, ['Time (' timeUnit ')'])

legend(ax, 'pdf of RUL', 'Estimated RUL', 'True RUL', 'Confidence Interval')
```

7.5.2.3 helperPlotTrend [20]

```
function helperPlotTrend(ax, currentDay, healthIndicator, mdl, threshold, timeUnit)
%HELPERPLOTTREND helper function to refresh the trending plot

%  Copyright 2018 The MathWorks, Inc.
t = 1:size(healthIndicator, 1);
HIpred = mdl.Phi + mdl.Theta*exp(mdl.Beta*(t - mdl.InitialLifeTimeValue));
HIpredCI1 = mdl.Phi + ...
    (mdl.Theta - sqrt(mdl.ThetaVariance)) * ...
    exp((mdl.Beta - sqrt(mdl.BetaVariance))*(t - mdl.InitialLifeTimeValue));
HIpredCI2 = mdl.Phi + ...
    (mdl.Theta + sqrt(mdl.ThetaVariance)) * ...
    exp((mdl.Beta + sqrt(mdl.BetaVariance))*(t - mdl.InitialLifeTimeValue));

cla(ax)
hold(ax, 'on')
plot(ax, t, HIpred)
plot(ax, [t NaN t], [HIpredCI1 NaN, HIpredCI2], '--')
plot(ax, t(1:currentDay), healthIndicator(1:currentDay, :))
plot(ax, t, threshold*ones(1, length(t)), 'r')
hold(ax, 'off')
if ~isempty(mdl.SlopeDetectionInstant)
```

```
    title(ax, sprintf('Day %d: Degradation detected!\n', currentDay))
else
    title(ax, sprintf('Day %d: Degradation NOT detected.\n', currentDay))
end
ylabel(ax, 'Health Indicator')
xlabel(ax, ['Time (' timeUnit ')'])
legend(ax, 'Degradation Model', 'Confidence Interval', ...
    'Health Indicator', 'Threshold', 'Location', 'Northwest')
end
```

7.5.2.4 helperReadData [20]

```
function data = helperReadData(filename, variables)
% Read data variables for the fileEnsemble
%
% Inputs:
% filename  - a string of the file name to read from.
% variables - a string array containing variable names to read.
%             It must be a subset of DataVariables specified
%             in fileEnsembleDatastore.
% Output:
% data      - return a table with a single row

% Copyright 2017-2018 The MathWorks, Inc.

data = table;
mfile = matfile(filename); % Allows partial loading
for ct = 1:numel(variables)
    if strcmp(variables{ct}, "Date")
        % Extract the datetime information from the file names
        % as the independent variable of the ensemble datastore
        [~, fname] = fileparts(filename);
        token = regexp(fname, 'data-(\w+)', 'tokens');
```

```
        data.Date = datetime(token{1}{1}, 'InputFormat', 'yyyyMMdd''T''HHmmss''Z''');
    else
        val = mfile.(variables{ct});
        % Convert non-scalar values into a single cell
        if numel(val) > 1
            val = {val};
        end
        data.(variables{ct}) = val;
    end
end
end
```

7.5.2.5 helperSortedBarPlot [20]

```
function sortedIdx = helperSortedBarPlot(tbl, ylbl)

% HELPERSORTEDBARPLOT helper function to create sorted bar plot


%  Copyright 2018 The MathWorks, Inc.

[~, sortedIdx] = sort(tbl{1,:}, 'descend');

tblSorted = tbl(:, sortedIdx);

figure

bar(tblSorted{1,:})

xticks(1:size(tblSorted,2))

xticklabels(tbl.Properties.VariableNames(sortedIdx))

xtickangle(45)

ylabel(ylbl)

end
```

7.5.2.6 helperWriteToHSBearing [20]

```
function helperWriteToHSBearing(filename, data)

% Write data to the fileEnsemble

% Inputs:

% filename - a string of the file name to write to.

% data     - a structure

save(filename, '-append', '-struct', 'data');

end
```