Eteläpää, Arttu

WCD-20 spark diagnostic

School of technology and innovation management
Master's thesis in technology
Automation and computer science

Vaasa 2020

UNIVERSITY OF VAASA
School of technology and innovation management
Author: Eteläpää, Arttu
Title of the Thesis: WCD-20 spark diagnostic
Degree: Master of science in technology
Programme: Automation and computer science
Supervisor: Timo Mantere
Instructor: Leif Strandberg
Year: 2021 **Number of pages:** 67

ABSTRACT:

Spark plugs are used to ignite the fuel in the Wärtsilä's SG engine ignition process and over-time they can suffer from various conditions such as wearing and fouling. More diagnostic information about the health condition of the spark plugs is needed and machine learning can be used to train a model with data from spark plugs to find the underlying relationship between the created model's inputs and outputs. This thesis evaluates if machine learning can be used to provide such diagnostic information from the WCD-20 engine module data, and as a result a concept machine learning model is implemented and tested.

The machine learning model is first designed, and the chosen learning technique and algorithm are supervised learning and neural network, respectively. The designed machine learning model classifies spark plugs into three different classes based on the input features, and these classes present the health conditions of the spark plugs. The data for the model's training and validation processes is gathered by testing spark plugs in different conditions with a spark plug test rig machine. During this testing, the spark plugs are labeled into the three different classes according to their conditions. The machine learning model is implemented with Python programming language using Tensorflow library, and after implementing and training, the model is saved and downloaded into an engine module. The engine module's source code is programmed to be able to run the machine learning model.

The machine learning model's accuracy is tested, and it achieves an overall accuracy of 82% when testing it with unseen data. The model has a high recall value for the output class that presents the spark plugs in good condition, but the model does not classify the spark plugs that are in bad condition as well. The model increases the overall CPU usage of the used engine module by 4,3%, which is relatively high, and this is due to the many matrix multiplication that are performed in the model's dense layers for each spark plug separately. Based on these results it is evident that spark plug health condition can be generally diagnosed by using machine learning, but some misclassifications can still occur.

KEYWORDS: WCD-20, machine learning, spark plug

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Eteläpää, Arttu
Tutkielman nimi: WCD-20 kipinän diagnosointi
Tutkinto: Diplomi-insinöörin tutkinto
Oppiaine: Automaatio ja tietotekniikka
Työn valvoja: Timo Mantere
Työn ohjaaja: Leif Strandberg
Vuosi: 2021 **Sivumäärä:** 67

TIIVISTELMÄ:

Sytytystulppia käytettään Wärtsilän SG moottoreissa sytyttämään polttoaine, ja ajan kuluessa ne voivat kärsiä monenlaisista kuntoa heikentävistä asioista, kuten likaantumisesta ja kulumisesta. Sytytystulppien kunnosta tarvitaan lisää diagnostiikka informaatioita ja koneoppimisen avulla voidaan kouluttaa malli mikä käyttää sytytystulpista saatavaa dataa löytääkseen suhteen luodun mallin sisään- ja ulostulojen välillä. Tämä opinnäytetyö arvio koneoppimisen soveltuvuutta tuottamaan tarvittavaa diagnostiikka informaatioita sytytystulpista WCD-20 moottorimoduulista saatavalla datalla, ja lopputuloksena konsepti koneoppimismalli toteutetaan ja testataan.

Koneoppimismalli suunnitellaan ensimmäisenä ja valittu koneoppimisen oppimistekniikka ja algoritmi ovat valvottu oppiminen ja hermoverkko. Suunniteltu koneoppimismalli luokittelee sytytystulppia kolmeen eri luokkaan valittujen sisäänmeno piirteiden perusteella, ja nämä luokat edustavat sytytystulppien kunnon tiloja. Koneoppimismallin opetuksessa käytetty on kerätty testaamalla erikuntoisia sytytystulppia käyttäen sytytystulppien testaus laitteistoa. Näiden testien aikana sytytystulpat luokitellaan kolmeen eri luokkaan niiden kunnon perusteella. Koneoppimismalli toteutetaan ja koulutetaan käyttäen Python ohjelmointikieltä ja Tensorflow kirjastoa, mikä jälkeen malli tallennetaan ja ladataan moottorimoduulille. Moottorimoduulin lähdekoodia ohjelmoidaan siten että se pystyy käyttämään koneoppimismallia.

Koneoppimismallin tarkkuus testataan ja se saavuttaa 82 %:n kokonaistarkkuuden testattaessa sitä ennennäkemättömällä datalla. Mallilla on korkea herkkyysarvo ulostuloluokalle mikä edustaa sytytystulppia hyvässä kunnossa, mutta malli ei luokittele huonokuntoisia sytytystulppia yhtä hyvin. Malli kasvattaa prosessorin käyttöastetta 4,3 %, mikä on melko korkea lisäys. Tämä lisäys johtuu monista matriisien kertolaskuista mitkä suoritetaan mallin tiheissä kerroksissa jokaiselle sytytystulpalle erikseen. Näiden tuloksien perusteella koneoppimista voidaan yleisesti käyttää sytytystulppien kunnon luokittelemiseen, mutta vääriä luokittelutuloksia voi silti tapahtua.

KEYWORDS: WCD-20, koneoppiminen, sytytystulppa

Contents

Figures

Tables

Abbreviations

SG	Spark gas
CPU	Central processing unit

1 Introduction

Wärtsilä's SG engines use spark plugs in their combustion process, which require maintenance at certain time intervals. WCD-20 module controls and measures variables related to the spark plugs, but currently more diagnostic information about the health of the spark plugs could be used. Different types of spark plugs, spark plugs from different manufacturers and the engine configuration can all affect the behaviour of the spark phenomenon and the measurement values, thus adding complexity to the spark plug health diagnosis process.

Machine learning corresponds to the concept of teaching a system to learn and improve from experience and data, to build a model that can provide a relation between input data and an output result. Machine learning is used by many companies in their systems and applications because it can be used effectively to solve problems that would be very challenging or time consuming to do with standard programming (Hao Karen 2018). Considering the analysis and prediction capabilities of machine learning, it is worth of investigation if the spark plug health condition can be estimated by using machine learning with the data that the WCD-20 module can provide.

1.1 Objective of the thesis

The objective of this thesis is to evaluate the possibility to use machine learning to diagnose spark plug health condition in Wärtsilä's SG engines. The motivation for this is to increase the maintenance interval for the SG engines and improve the diagnostic information about the spark plugs, thus making it easier to detect spark plug failure.

The data for this thesis will be gathered from several spark plugs in different conditions and they will be tested using a spark plug test rig machine. The WCD-20 module data from this testing is used in the machine learning model, and appropriate parameters that

can indicate spark plug health condition from this data will be selected with the help of the information gathered in the theory part of this thesis. The suitable machine learning algorithms will be discussed and the best fitting algorithm for this problem will be chosen based on the ability to estimate the spark plug health condition. Finally, the performance of the developed machine learning model will be tested, and the results will be analysed to evaluate whether it is feasible to use machine learning in this issue.

1.2 Structure of the thesis

This thesis consists of six chapters. Chapter 1 is the introduction of the thesis and the chapters 2 and 3 focus on the relevant theory that provides support and information about the objective of the thesis. Chapter 4 presents the designing of the machine learning model and Chapter 5 is about the implementation process of the machine learning model. In Chapter 6 testing and analysis of the machine learning model is introduced. The Chapter 7 is the final chapter of this thesis and it will provide the conclusion to this thesis.

2 Machine learning

In general machine learning is artificial intelligence that is capable of learning, and generally data and algorithms are used to build and train a machine learning model that can be descriptive, predictive or both. When a system is in a changing environment it needs to learn and adapt in order to be intelligent, and the learning capabilities also relieve the system designer from thinking and designing every possible solution for every possible scenario and event that could occur. (Ethem Alpaydin 2020: 3-7)

Machine learning is considered to be a part of artificial intelligence, and it mainly differs from traditional definition of artificial intelligence in that passive observations from data are used to learn and to make the predictions. Artificial intelligence is a broader subject which involves machines and computers interacting with and learning from their surrounding environment intelligently, and one way to achieve this can be by utilizing machine learning. The definition of artificial intelligence can also change and grow overtime when technology advancements occur. (Roberto Iriondo 2018)

Machine learning algorithms are great for solving problems related to analyzing data and using that data to make, for example, predictions, classifications, optimizations, troubleshooting or controlling (Ethem Alpaydin 2020: 3-4). Some possible tasks could include finding the most satisfying solution for non-polynomial problem through optimization, troubleshooting a system by finding patterns and deviations from the patterns in data or classification of data samples into categories based on their individual features. Thanks to these features machine learning can be used in various applications and fields such as robotics, banking and medicine to solve myriads of problems. (Mehryar M., A. Rostamizadeh & A. Talwalkar 2018: 1-2)

The ability to learn is achieved by building a mathematical model based on collected sample data, also known as training data. This training data is used to train the model through different kinds of techniques and various iterations. An example of the training process could be using collected training data that has been labeled by a human with

labels such as “good” or “bad” sample, and then going through the training data and it’s features and training the machine learning model to be able to label future unlabeled data by evaluating the features of the data. During the learning process the machine learning model will try to predict the label of a training data sample by using the values of the features of the training data sample, and depending on the result the mathematical model that is used to calculate the value of the label will be adjusted accordingly. (Mehryar M., A. Rostamizadeh & A. Talwalkar 2018: 1-3)

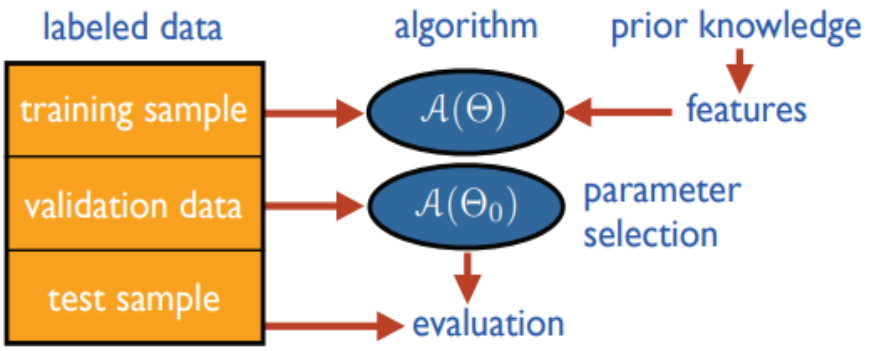

Figure 1 Example stages of a machine learning process (Mehryar M., A. Rostamizadeh & A. Talwalkar 2018: 5).

2.1 Learning techniques

The techniques to train a machine learning model can vary and depending on the data available the most suitable learning technique for a given problem can be chosen. The machine learning techniques include (Mehryar M., A. Rostamizadeh & A. Talwalkar 2018: 1-3):

- Supervised learning.
- Unsupervised learning.
- Semi-supervised learning.
- Transductive inference.
- On-line learning.
- Reinforcement learning.

- Active learning.

2.1.1 Supervised learning

Supervised learning is a learning technique that utilizes readymade training data during the training process. The training data can be labeled and during the training the machine learning model tries to learn the relation between the input training data and the labeled output. The input data can consist of many features and it can be represented for example as a feature vector. The relation between the input data and the output can be represented as a mathematical function, for example as a linear regression model. (Vladimir Nasteski 2017: 3-5).

Other parts of this learning technique include the validation of the model with part of the data saved for this phase, where the maker of the machine learning model validates it's performance and correctness with some part of the data that has been available to them. After the machine learning model is ready and working, it can be tested in the real environment to see how it performs. (Vladimir Nasteski 2017: 3-5)

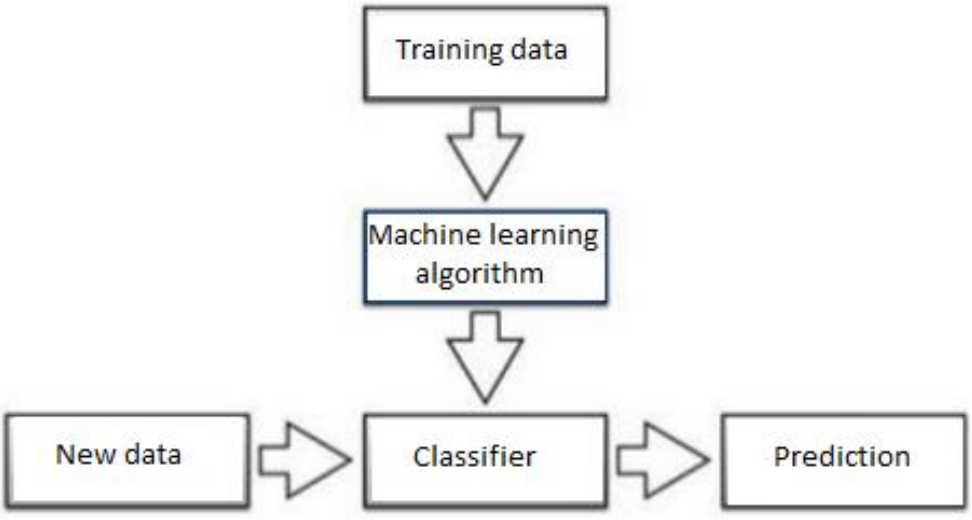

Figure 2 The process of supervised learning. (Vladimir Nasteski 2017: 4).

Supervised learning is most commonly utilized in neural networks and decision tree algorithms, which both depend on the information and data that is given to them by the pre-determinate classification. In the end, supervised learning is used in two different learning tasks, classification and regression. In regression problems the label is continuous and in classification scenarios the label is discrete. Applications that use supervised learning are such of a kind where historical data can likely predict feature events. (Vladimir Nasteski 2017: 5)

2.1.2 Unsupervised learning

Unsupervised learning aims to find regularities or patterns in the input data without readymade training data with labels or supervision that tells the machine learning model if it is right or wrong. Unsupervised learning is most commonly used in density estimation. Main methods that are used in unsupervised learning include principal component analysis and clustering. In clustering the goal is to group the input data into separate clusters according to their features. In a successful scenario this will result a division between data points that have different kinds of features and in contrast data points with similar features will be clustered together. (Ethem Alpaydin 2020: 11-12).

2.1.3 Reinforcement learning

Reinforcement learning focuses on optimizing the policy of a sequence of correct actions needed to reach the satisfactory output of a system. In other words, the machine learning algorithm learns a policy how to act and respond to specific events that happen in the world around it. These events have an impact on the environment, which in turn affects and provides feedback to the machine learning system that it can use and learn form. (Vladimir Nasteski 2017: 2)

Systems which output can be in such form include robots that are trying to navigate in an area to a desired destination or a game where the artificial intelligence tries to win. In such learning scenarios a single action or move is not that important, and these single

decisions are also only good if they are a part of the right policy that will eventually reach the desired goal. (Ethem Alpaydin 2020: 12-13)

2.2 Algorithms

The available data and the chosen learning technique affect the algorithm that can be used in a machine learning model. It is important to choose the right algorithm for a given model, because the core of the machine learning is the algorithm that is created to learn from the data that will be inputted to it. The algorithm can mimic human style learning in some tasks and in addition the algorithm can represent how difficult it is to learn in different environments. Currently many of the machine learning algorithms have already been developed and improved over the years and choosing a readymade algorithm and altering it for the desired application can be a part of the workflow of many machine learning systems. (Vladimir Nasteski 2017: 1-3)

Some of the machine learning algorithms are listed below, and in this thesis, I will focus on some of the most suitable candidates that could solve the WDC-20 spark diagnostic problem. (Vansh Jatana 2019: 1-4)

- Linear regression.
- Logistic Regression.
- Decision tree.
- Boosting.
- Naive Bayes.
- Neural networks.
- K-means.

Machine learning algorithm also behave differently and have unique properties to one another. Properties such as memory consumption and size, time to learn and to predict and overfitting tendency also separate algorithms from each other and must be considered when developing a machine learning model. (Vansh Jatana 2019: 1-4)

Each machine learning algorithm is typically used either in classification or regression problems, and classification problems can be also divided into single- and multiclass classification tasks. Algorithms can be transformed to perform multiclass classification from single class classification by using methods such as “one vs all” or “one vs one”, which split the single class classification problem into multiple different classification problems. These divided and new classification problems are then calculated and used to create output with multiple classes. (Mehryar M., A. Rostamizadeh & A. Talwalkar 2018: 213-230)

2.2.1 Linear Regression

Linear regression aims to find and model the relationship between some explanatory variables and some real valued outcome. When this is cast as a learning problem, the domain set *X* is a subset of R^d for some *d*, and the label y is the set of real numbers. The goal is to learn a linear function $h : R^d \rightarrow R$ that is the best approximation of the relationship between the models input and output variables, where input is vector x and output is $(w, x) + b$ where b is the added bias value and the symbol *R* corresponds to a set of real numbers. The hypothesis for this class of linear regression is the formula below. (Shai S. S. & B. D. Shai 2014: 123-125)

$$H_{reg} = L_d = \{x \rightarrow (w, x) + b : w \in R^d, b \in R\}$$

In addition to this a definition for a loss function is needed. In a classification task the definition is simple, and it can be defined as $l\big(h(x, y)\big)$, and this indicates if the $h(x)$ with x as an input value correctly predicts desired output y. In the case of regression, we need to define how much penalty we will have the further away our prediction $h(x)$ from the correct *y* value is. The formula for the squared-loss function is shown below, and this is one common way of calculating the loss-function. The empirical risk function for the squared-loss function is called Mean squared error, which is used to calculate the

expected value of a loss function, is also represented below the loss function formula. (Shai S. S. & B. D. Shai 2014: 123-124)

$$l\big(h(x,y)\big) = (h(x) - y)^2$$

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$$

The mathematical model for simple regression and multiple regression that has a linear combination of features is shown respectively in the formulae below. They depict linear regression between a continuous scalar dependent variable y and one or more explanatory variables x. Variables for β represent the regression coefficients for the explanatory variables and variable β_0 represents the intercept. Variable e is the error term. The variable *y* is often called a label or a target in machine learning terminology, and the explanatory variables are called features or input variables for example. (Vladimir Nasteski 2017: 6-7)

$$y = \beta_0 + \beta_1 x_1 + e$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. + e$$

The output of a simple linear regression can be similar to the picture 4, where the red line corresponds to the output of the linear regression model. This output has been calculated to fit the blue points, in other words the input data, as accurately as possible by trying to minimize a value of a loss function that is used in this case. (Vladimir Nasteski 2017: 7)

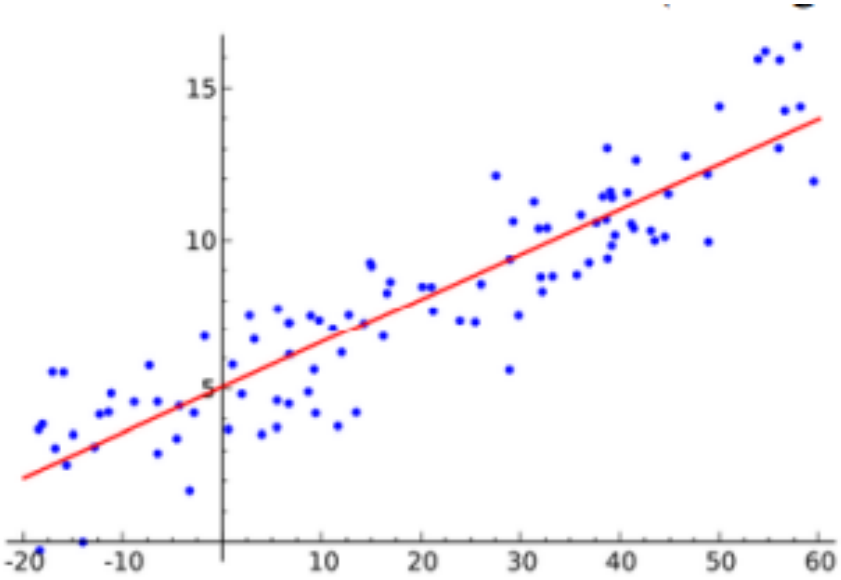

Figure 3 Output of linear regression. (Vladimir Nasteski 2017: 7).

Linear regression algorithm tends to have low memory consumption and size compared to other machine learning algorithms. It is also fast in the learning process and is generally fast when doing the prediction calculations. Simple linear regression algorithm also has low overfitting tendency, but can be prone to underfitting, and the parametrization is usually fairly straightforward. (Vansh Jatana 2019: 1-4)

2.2.2 Logistic regression

Logistic regression is a discriminative classifier that is used to predict the probability of an event by fitting the data to a logistic function. The hypothesis of the logistic regression is the first represented formula below, where the function g corresponds to a sigmoid function, which is also represented as a formula after the hypothesis, and θ is the vector of parameters which will be calculated to fit the classifier. Logistic regression function is best fitted to be used in classification problems. (Vladimir Nasteski 2017: 8)

$$h_\theta = g(\theta^T x)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

The basic logistic regression algorithm functions by first extracting a set of weighted features from the input data and then taking logarithms from the input data features and combining the results linearly. The input variables to the logistic regression can be either numerical or categorical and the output will be in the range of [0, 1] (Vladimir Nasteski 2017: 8). The logistic regression function is generally fast in doing the learning process and calculating the predictions, while having a small memory usage (Vansh Jatana 2019: 1-4).

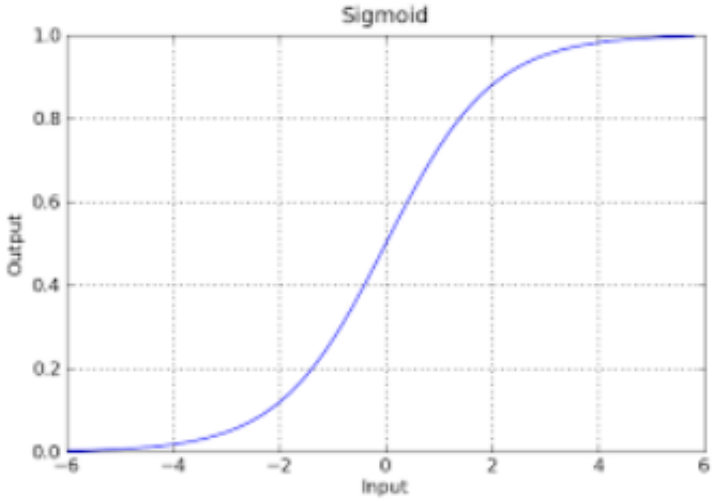

Figure 4 Output of logistic regression. (Vladimir Nasteski 2017: 8).

The logistic regression can also be extended to do a multiclass classification. The extended logistic regression formula is the formula that is represented below, and it is often called the Softmax equation. The W values correspond to the weights, x values are the input values, the b values are the bias or added error values and K represents the number of classes in the multi-class classifier. The Softmax equation divides the exponent of each input element with the sum of exponents of all of the input elements, thus creating an output with multiple classes and probabilities assigned to them. The sum of the output probabilities is equal to one and every single class will have an own output that is between [0, 1]. (Developers.google 2020)

$$p(y = j|x) = \frac{e^{(W_j^T x + b_j)}}{\sum_{k \in \mathrm{K}} e^{(W_k^T x + b_k)}}$$

2.2.3 Artificial neural networks

Artificial neural networks are based on the idea of multiple neurons that join together with communication links to carry out complex computations, which mimics the behaviour of human brain. The neurons themselves are all modelled as a simple scalar function, for example the sign-function, sigmoid-function or the threshold-function, and these functions of the neurons can be called the activation functions and are defined as $\sigma: \mathrm{R} \rightarrow \mathrm{R}$. The outputs of the neurons are then connected to the inputs of some other neuron and the input of a neuron is obtained by taking a weighted sum of the outputs of all the neurons connected to it. These weights related to the neurons are adjusted in the learning process of the artificial neural network according to the error of the output of the network, which is gotten from calculating the output of a selected loss function. Methods such as stochastic gradient descent can be used in the training process of a neural network to adjust the weights. Example structure of a feedforward neural network, which does not have any cycles, can be seen in the picture 4. (Shai S. S. & B. D. Shai 2014: 269-271)

The mathematical formula for a layered feedforward neural network is represented below. This is described as a directed acyclic graph $G = (V, E)$, where E represents the edges of the graph and V depicts the layers of the graph, and the weight function over the links between the neurons is $w : E \rightarrow \mathrm{R}$. When advancing from here, the set of nodes can be decomposed into a union of disjoint subsets $V = \bigcup_{t=0}^{T} V_t$, such that every edge in E connects some node in V_{t-1} to some node in V_t, for some $t \in [\mathrm{T}]$. The input layer V_0 contains $n + 1$ neurons, where n is the dimensionality of the input space, and for every $i \in [\mathrm{n}]$ the output neuron i is simply x_i. We then denote $v_{t,i}$ the i :th neuron of the t:th layer and by $o_{t,i}(X)$ the output of $v_{t,i}$ when the network is fed with the input vextor X. Therefore, for $i \in [\mathrm{n}]$ we have $o_{0,i}(X) = x_i$ and proceed calculations in layer by layer manner, which is represented in the form of calculations for the layer $t + 1$,

where $v_{t+1,j} \in V_{t+1}$, and let $a_{t+1,j}(X)$ denote $v_{t+1,j}$ when input vector X is fed into the network. (Shai S. S. & B. D. Shai 2014: 269-271)

$$a_{t+1,j}(X) = \sum_{r:(v_{t,r},v_{t+1,j})\in \mathrm{E}} w\left(\left(v_{t,r}, v_{t+1,j}\right)\right) o_{t,r}(X)$$

$$o_{t+1,j}(X) = \sigma(a_{t+1,j}(X))$$

In other words, the $v_{t+1,j}$ is a weigthed sum of the neurons outputs in V_t, which are connected to the $v_{t+1,j}$. Weighting is done according to w, and the output of $v_{t+1,j}$ is the application of the activation function on its input. (Shai S. S. & B. D. Shai 2014: 269-271, 281-282)

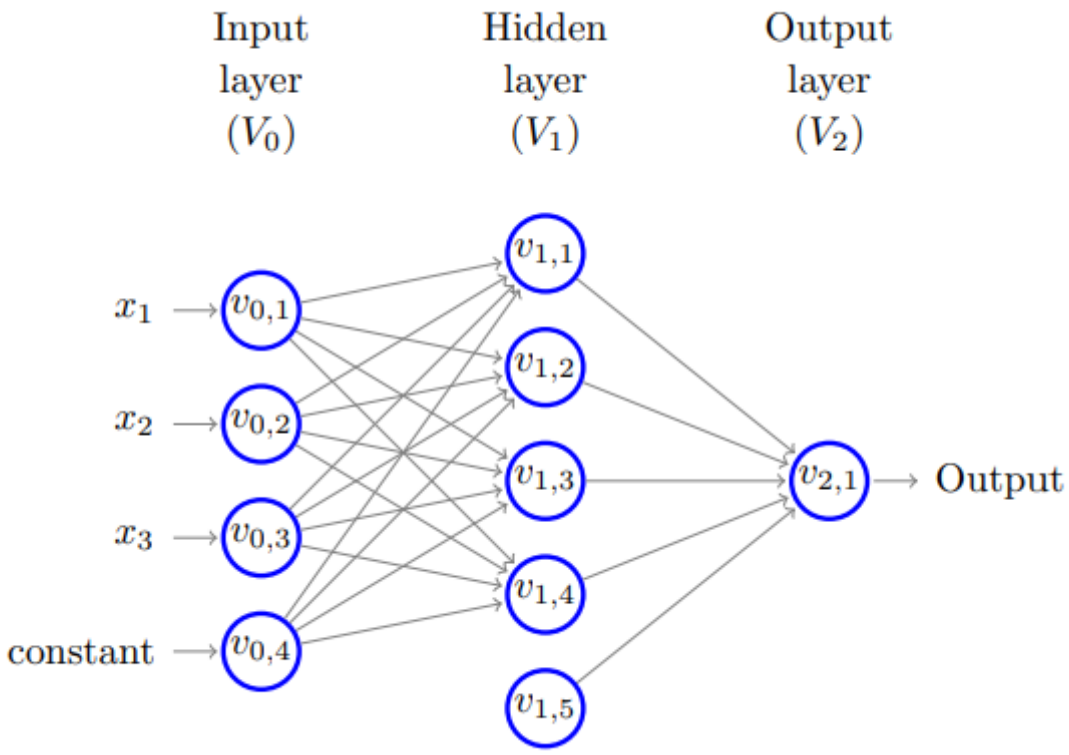

Figure 5 Artificial neural network with three layers (Shai S. S. & B. D. Shai 2014: 270).

Neural networks can be modified to support multiclass classification by using the “one vs all” or “one vs one” methods. In such a scenario the output layer can consist of multiple binary neurons instead of one which will output the result as a multiclass classification and the last layer of neurons can be represented as a Softmax function layer (Developers.google 2020). This concept is shown in the picture 6 below, and the x values are the input values to the network, w values are the weights of the neurons and the z values are the outputs of the neurons (Ashutosh S. & Y. Li 2017).

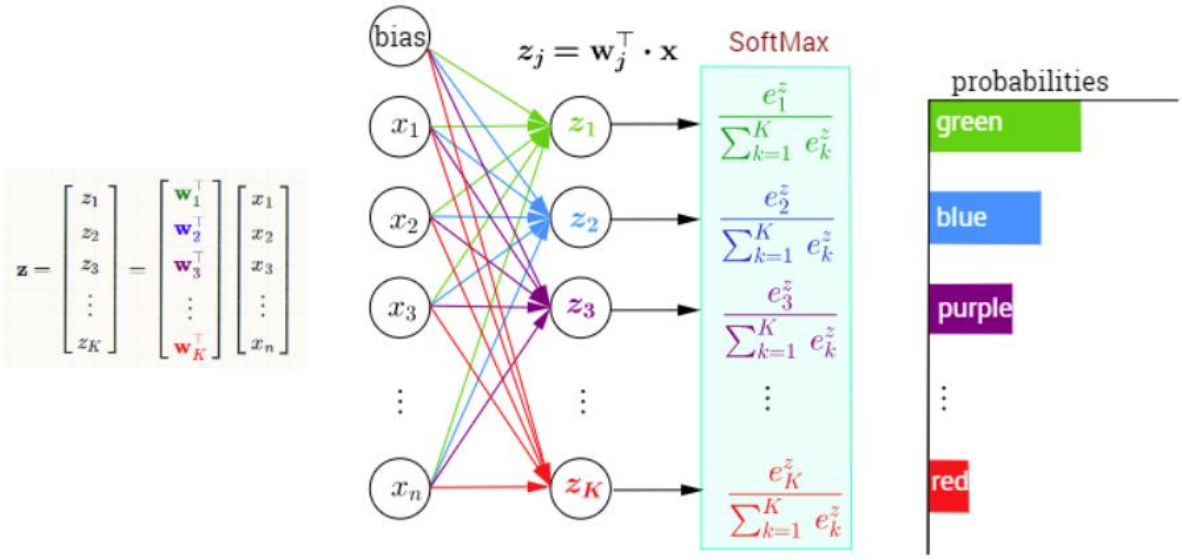

Figure 6 An example of an artificial neural network with multiple classification **(**Ashutosh S. & Y. Li 2017).

Artificial neural networks are usually fast when calculating the predictions, but time they need to train and learn can be quite high in comparison to other machine learning algorithms. They also consume more memory than the regression algorithms, but not as much as for example random forest or boosting algorithms, making them average in that regard. Artificial neural networks are mainly used in classification problems. (Vansh Jatana 2019: 1-4)

2.3 Data

Typically, the first requirement for the data that can be used in a machine learning model is that it is in a format that can be read by a computer. It can be represented for example as a vector or in tabular form where every row represents a particular data sample and every column represents a feature. The data can also be in such a format that is not obviously a ready table or vector, such as text, images or genomic sequences, thus making the feature selection and data preparation more demanding of a job. (Deisenroth M. P, A.A. Faisal & C. S. Ong 2020: 251-253)

After the data is in the right format, it still has to be converted into numerical format if it is not yet in it. For example, data that is in categories such as “up” or “down” should be converted to 0 and 1 for example. Numerical data must also be inspected in case of the scale, units and constraints of it are fit for the entire model. (Deisenroth M. P, A.A. Faisal & C. S. Ong 2020: 252-254).

One of the problems that the data can have is the noise. It is difficult for the machine learning model to learn for example the right relation between the input and the output from data that has too much noise. In such a scenario the model could learn some right parts about the actual model it tries to predict, but in addition to that it can learn the noise too (Kalapanidas E., N. M.Avouris, M. V. Craciun & D. Neagu 2003: 2-4). This leads to incorrect model or overfitting or underfitting. Ways to reduce noise in data include removing features that are not useful, regularization of the model, cross-validation using more data and early stopping. (Elite data science 2019).

2.4 Model validation and testing

After the machine learning model is complete the performance of the model needs to be validated in order to find out if the model can predict the expected outcome in a right manner. The basic idea of the validation process is to calculate how many times the machine learning model predicted the expected result right or wrong, or how much error

the model has between the values it predicted versus the expected result values. (Shai S. S. & B. D. Shai 2014: 146-150)

The validation and testing for the machine learning model can be done by splitting the training, validation and test data into different sets and after the training and model validation sets have been completed, the test set is used to calculate the error the machine learning model produces with unseen data. Other method to test the model is to use k-fold cross validation, which splits the data into k amount of folds and trains the data on k-1 folds and then tests on the fold that was left out . This is done for all of the combinations and the results are averaged for every instance. The advantage with this method is that every observation is used for both training and validation. In addition to these methods there are also other methods that can used in the validation and testing process of the machine learning model. (Shai S. S. & B. D. Shai 2014: 149-150)

3 Engine ignition

Engine ignition system is responsible for generating the spark that is used in the ignition of the fuel-air mixture. This energy created in the ignition is then used to move the engine cylinder piston thus also rotating the crankshaft of the engine, and general illustration of this process is represented in the picture 6. In Wärtsilä's spark gas engines the WCD-20 module is responsible for managing and measuring the parameters related to the spark plugs and the ignition. In 4-stroke combustion engines, the stages of the combustion cycle include induction, compression, power and emission. (University of Calgary 2019)

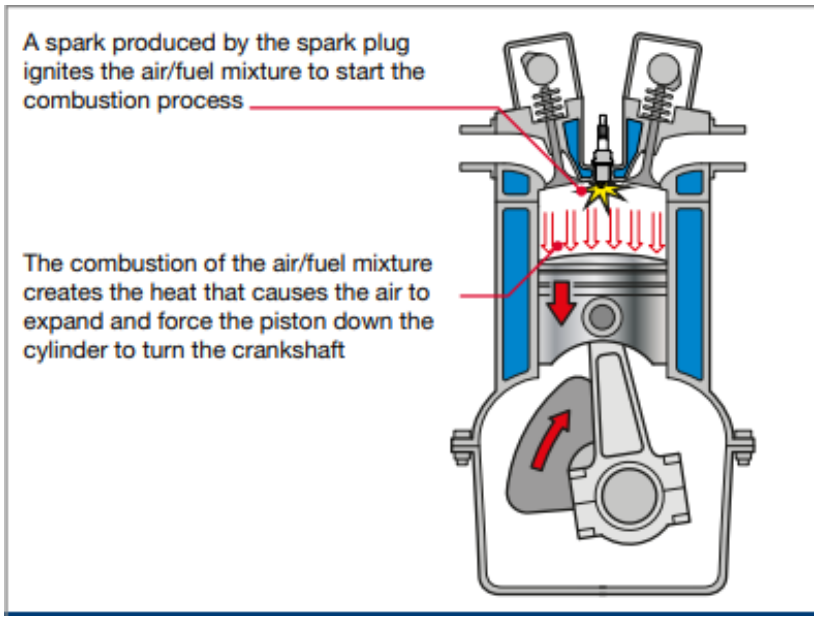

Figure 7 Combustion and spark (Wouter K., P. Coombes & G. Couvert 2019: 3).

During the intake process the intake valve opens and lets the air-fuel mixture inside the combustion chamber. After this the combustion part begins and the piston starts to move upward in the cylinder, thus compressing the air-fuel mixture and increasing the temperature, pressure and the density according to the ideal gas law. Right before the piston reaches the top dead center, a spark plug is used to generate the spark that ignites the air-fuel mixture. The stage preceding this point is called the power phase, and during

this phase the pressure of the gases in the combustion push the piston downward, thus decreasing the density, temperature and pressure of the combustion gases in the cylinder according to the ideal gas law. Right before the piston reaches the bottom dead center, the exhaust valve opens, thus causing the exhaust gases to expand. The piston continues to move upward after the bottom dead center pushing the exhaust gases out of the cylinder through the exhaust vent, and this last stage is called the exhaust phase. (University of Calgary 2019)

For the ignition to happen, the conditions in the cylinder must be correct and the equipment used in the combustion must be in good condition. The equipment will wear overtime and it is critical to notice the possible signs that might indicate a need for service, in order to maintain proper engine combustion cycle and avoid any severe downtime or damage to the engine. To increase awareness to these issues, different automated diagnostic systems and sensors can be used to detect and measure various parameters that help in the diagnosis of the engine. (Raman K Autar 2004: 1-5)

3.1 Spark plugs

Spark plugs are the components in the cylinder that receive a short burst of high voltage from the ignition system to generate a spark between the small gap in the tip of the spark plug. This generated spark is then used to ignite the air fuel mixture inside the cylinder, and the design and features of the spark plugs used in an engine have an impact on the ignition and combustion process in general (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 32–33). Despite having different properties, spark plugs should still be able to generate the spark in various operating conditions, where the temperature, air fuel mixture, pressure and engine speed and load can vary. (Wouter K., P. Coombes & G. Couvert 2019: 18).

The general structure of a spark plug can be seen in the picture 7. The tip of the spark plug is the area where the spark event itself occurs, and it consists of the centre electrode and the ground electrode. The housing of the spark plug provides support and

protection to the insulator, and also secures the spark plug assembly to the engine. The insulator part of the spark plug gives electrical insulation between terminal, centre electrode, housing and the centre shaft.

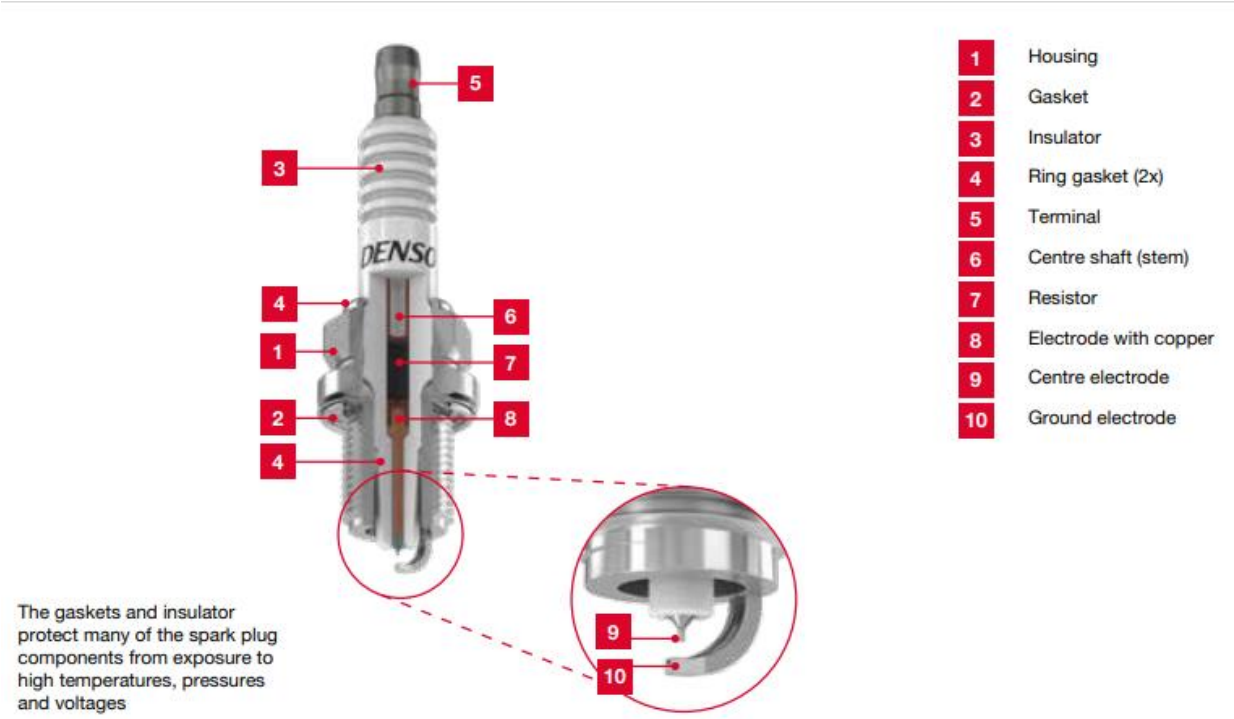

Figure 8 General structure of a spark plug (Wouter K., P. Coombes & G. Couvert 2019: 18).

The voltage and the electrical energy that discharges during the spark event is dependent on multiple factors, such as the spark plug gap length, internal resistance of the spark plug and the pressure of the gas between the spark plug gap. Generally increasing the pressure or the spark plug gap length will lead to higher required voltage to generate the spark. Additionally, in order for the ignition in the cylinder to happen, the voltage that is generated in the spark has to be high enough. Depending on the fuel type and it's mixture, the electrical conductivity inside a cylinder can vary, and typically gasoline requires less voltage for the ignition than compressed natural gas fuels. The shape of the electrodes in the spark plug also affect the required voltage, generally smaller electrodes decreasing the required voltage, but ultimately also raising the tip temperature, thus

leading to reduced lifetime. (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 32–33)

A general view of the behaviour of the voltages during a spark can be seen in the picture 8 below. At point ‘a’ the current applied to the primary winding of the ignition systems coil is cut off, thus inducing a high voltage which passes down to the spark plug. In point ‘b’ the voltage increases and between points ‘b’ and ‘c’ the gas between the spark plug ioinises, thus generating the spark. This phase is also known as capacitance spark. Between the points ‘c’ and ‘d’ a longer duration of the spark is maintained and this stage is called the inductance spark, which refers to the fact that the spark is generated and maintained by the electromagnetic energy of the coil in the ignition system, in which the current gradually reduces. This electromagnetic energy of the coil is not enough to maintain the spark after point ‘d’, thus ending the spark and the discharge. (Wouter K., P. Coombes & G. Couvert 2019: 20).

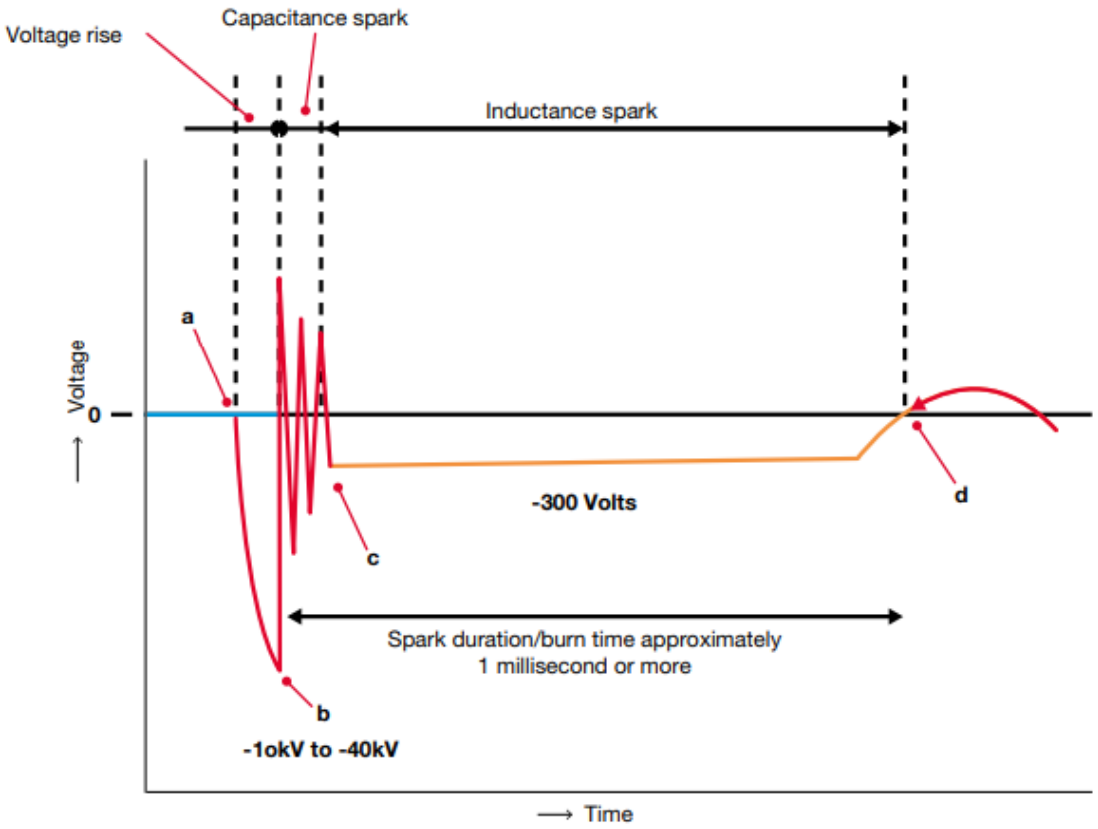

Figure 9 Voltages and phases of a spark plug's spark (Wouter K., P. Coombes & G. Couvert 2019: 20).

Spark plugs have to withstand high temperatures and pressures in the combustion chamber, thus causing the electrodes of the spark plug to erode. This erosion can lead to the increase in the spark plug gap length, which will increase the required voltage to generate the spark. If the required spark voltage grows too large, the ignition system will not be capable of producing enough voltage for the spark to occur, thus causing misfires. (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 32, 37)

While the spark plug gap growth is one of the main reasons that leads to increased voltage after several running hours, other factors such as electrical insulator deposits and oxide layers to the spark plug can also cause the required spark voltage to increase. This lessens the quality of the spark and can enable the spark to occur from a different path. This different path can be for example from the side of the electrode, which can lead to

increase in the cyclic variation of the indicated mean effective pressure, which also affects the required the spark voltage. In general, older and more used spark plugs have higher required voltage to generate the spark and worse spark quality compared to new and less used spark plugs, but the rate of the erosion decreases overtime, which implies that newer spark plugs suffer from it more than the older plugs. (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 32, 37)

In addition to the above conditions and effects, spark plugs can go through other various kinds of wearing and aging conditions during their running hours. For example, the air/fuel mixture, fuel type, mechanical damage and the general conditions in the combustion chamber can affect the health of a spark plug. Some of these conditions with causes and effects are represented in the table 1 below. (Bosch 2019: 1-2).

Table 1. Spark plug condition (Bosch 2019: 1-2).

Condition	Cause	Effects
Lead fouling	Lead additives in fuel. Glazing results from high engine loading after extended part-load operation.	At high loads, the glazing becomes conductive and causes misfiring.
Oil-fouled	Too much oil in combustion chamber.	Misfiring, difficult starting.
Formation of ash	Alloying constituents, particularly from engine oil, can deposit this ash in the combustion chamber and on the spark-plug face.	Can lead to auto-ignition with loss of power and possible engine damage.

Center electrode covered with melted deposits	Overheating caused by auto-ignition.	Misfiring, loss of power (engine damage).
Heavy wear on center electrode	Spark plug exchange interval has been exceeded	Misfiring, particularly during acceleration (ignition voltage no longer sufficient for the large electrode gap). Poor starting.
Heavy wear on ground electrode	Aggressive fuel and oil additives. Unfavorable flow conditions in combustion chamber, engine knock.	Misfiring, particularly during acceleration (ignition voltage no longer sufficient for the large electrode gap). Poor starting.
Insulator-nose fracture	Mechanical damage	Misfiring, spark arcs-over

3.2 Engine ignition system

Engine ignition systems are responsible for creating the high voltage that can produce the spark to the spark plugs. The basic structure of an ignition system consists of ignition coils, spark plugs, contact breaker switch and rotator arm in distributor body. The ignition coil typically contains primary and secondary windings, and the primary winding is supplied with electrical current. This produces a magnetic field around both of the windings, but when the current is switched off, the induced voltage to the secondary winding will be much higher because of its structure. This induced voltage is then used to generate the spark in the spark plugs. The contact breaker switch is used to open and close

the circuit that supplies the electrical current to the primary winding, and the rotator arm is used to distribute the induced voltage to the correct cylinders and spark plugs in the right cylinder firing order. (Wouter K., P. Coombes & G. Couvert 2019: 6-7).

In Wärtsilä's 34SG engines the ignition coil is located in the cylinder cover and is integrated in the spark plug extension. The ignition module communicates with the main control module of the engine control system to aid in determining the global ignition timing, and the ignition module controls the cylinder specific ignition timing based on the combustion quality. (Wärtsilä engines 2011: 6-8)

In the ignition process of the 34SG engine the lean mixture of gas fuel and air, which corresponds to the greater amount of air present in the cylinder than is needed for complete combustion, is first ignited in the pre-chamber before it sets the flame front for the main combustion chamber. This design is essential part of the learn-burn spark-ignited gas engine, which enables the generation of less NO_x emission, extended spark plug life and reliable and powerful ignition with high combustion efficacy and stability. (Wärtsilä engines 2011: 6-8).

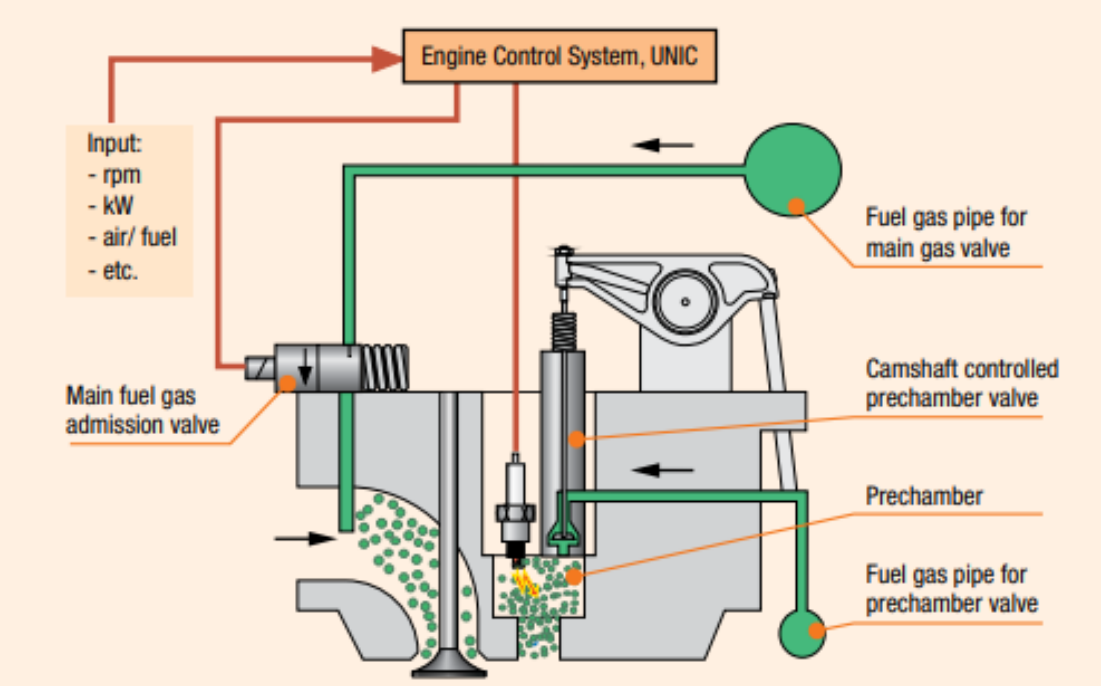

Figure 10 Overview of the 34SG engine (Wärtsilä engines 2011: 6).

WCD-20 module is a part of some of the Wärtsilä's SG-engine ignition systems, which controls and measures ignition and spark related parameters of the cylinders, such as ignition timing and spark energy level values, and the engine control module provides information to the module for it to operate as accurately as possible (Wärtsilä Engines 2019: 100-101). An example of another modular ignition system is the Altronic LLC' CPU-XL VariSpark ignition system for large gas engines, and it uses an improved capacitive discharge ignition technology where only a measured amount of a large capacitor is discharged to generate a spark. (Altronic 2020).

The capacitor discharge ignition functions by storing energy in an external capacitor, which is discharged into the ignition systems primary coil winding when the spark is required to be generated. The capacitor charging process is fast, thus enabling short transient response, fast voltage rise and a short spark peak duration. The high initial spark that this type of system generates allows combustion to occur in an engine that has excess oil or an over rich fuel air mixture in the combustion chamber. The high initial spark

voltage also avoids leakage across the spark plug insulator and electrode caused by fouling, but on the other hand, the short spark duration caused by the fast capacitor discharge leaves less energy for a longer spark duration to take place which might be needed for complete combustion in some cases. This short spark duration can cause misfires and increased exhaust emission, but the use of multi-spark ignition can alleviate this issue. In multi-spark the spark is generated multiple times in an engine cycle to achieve complete combustion, but this stresses the spark plug and can cause increased spark plug wearing. (Industrial gas engine controls 2020)

The structure of a generic capacitor discharge system can be seen in the picture 11. The basic operating principle of this system is that current is supplied to the circuit for example, through a battery or an alternator, and the supplied electricity charges the capacitor. The diode prevents the capacitor from discharging before the desired ignition timing, which is provided by the engine control unit. When the ignition timing is right the electronic switch is turned on and the capacitor discharges it's energy to the ignition coil. (eeweb 2020)

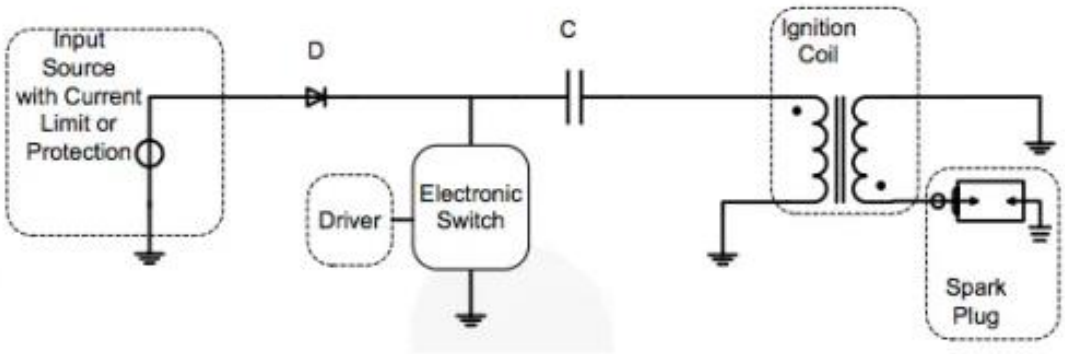

Figure 11 Generic structure of a capacitor discharge ignition system (eeweb 2020).

In general, an Altronic CPU-XL VariSpark system consists of four modules that are used to control, to generate and measure the spark and ignition, and this concept is visible in the picture 12. The logic/display module manages all inputs, communication and control functions used to maintain and generate the spark. Junction/diagnostic module houses

all of the spark discharge diagnostic logic and all cylinder assignments for the engine firing order are done by this module. The output module is installed on every cylinder bank on the engine, and this module accepts logic-level firing signals and generates the high energy electrical pulse for the ignition coil/EZRail modules. The ignition coils or the EZRail modules, which consist of multiple ignition coils, are the final part of the system, and they are used to generate the spark voltage. (Altronic 2020)

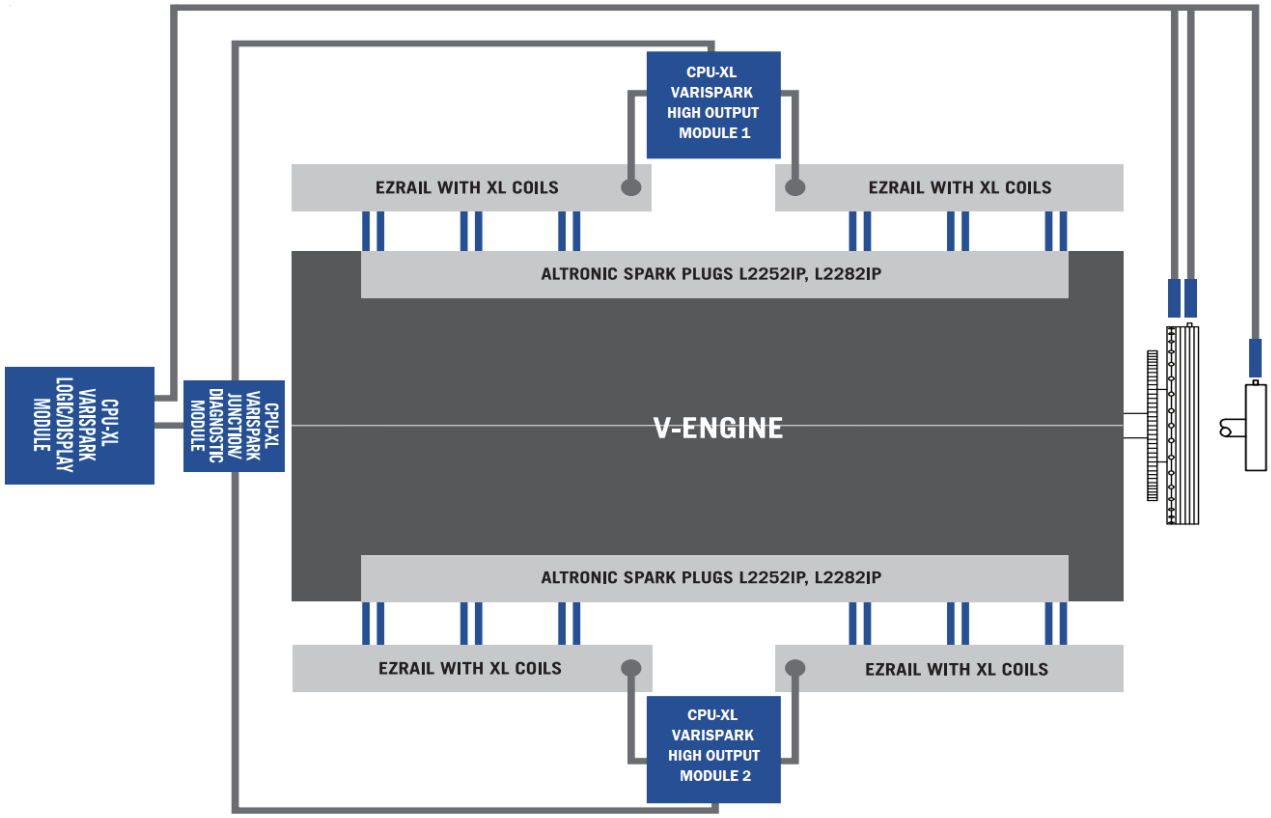

Figure 12 Generic structure of an Altronic CPU-XL VariSpark system (Altronic 2020).

The ignition timing signal in the Altronic's system is generated by using the angle of the engines flywheel to determine crankshaft's position. The magnetic sensing holes in the flywheel are monitored to calculate the angle of the flywheel and these values are matched with the programmed engine firing patterns and angles, thus allowing precise spark ignition timings. The adjustment of the spark energy level is also possible in this kind of a system. These adjustments can lower the emissions and increase the spark plug

lifetime by using less electrical energy to generate the spark with newer plugs and optimal cylinder conditions. Readjustments such as increasing the spark energy level to ensure that a spark occurs, and misfires are avoided can be made when the situation requires them. This can be for example, due to spark plug wear or transformed conditions in the cylinder. (Altronic 2020)

4 Model design

In this chapter the machine learning model's design and contributing factors to it are presented. In this thesis the general design of the machine learning model consists of the used machine learning technique, algorithm and data, while the contributing factors to this core design are the hardware architecture and designing of how to test the model's performance. The goal of the model is to estimate the spark plug health condition as accurately as possible, because the result value is needed to inform the user about the health state of the spark plugs, but also possibly adjust parameters related to ignition with this diagnostic estimation data.

The planned phases to develop the machine learning model in this thesis consists of the following parts, and these parts will be gone through later in this chapter:

- Collect the data
- Create the model
- Train the model
- Evaluate the model's performance

4.1 Hardware architecture design

The used hardware will affect the amount of available processing power that the model can use, thus contributing to the model's core design. Overall, there are three possible hardware architecture designs available in this case, which are:

- Wärtsilä engine module.
- Wärtsilä engine module and a pc.
- Wärtsilä engine module and a microcontroller.

Implementing the machine learning model directly to a Wärtsilä's engine module does not require any additional hardware and successful integration of the machine learning model to the engine module could be an effective way of implementing the model. In

this scenario the model could directly read the data from the WCD-20 module and use this data to calculate the spark plug condition predictions and based on the results control the module or inform the user about the results. Plausible limitations or disadvantages with this approach may arise if the CPU and memory usage become too high due to the implemented model, thus negatively impacting the engine module's performance.

The other options use other hardware to perform the processing related to the machine learning model. In these two scenarios the model would be implemented on to a pc or on to a microcontroller and the WCD-20 module would communicate with them about the spark plug measurement data and machine learning model prediction results. The microcontroller is an additional piece of hardware that would be required to be purchased and installed to implement this approach, but a pc can already be used in the spark plug measurements. These two approaches enable the usage of more processing power for the model, but at the same time they have a more complicated design due to the additional hardware component.

Considering these points, the chosen approach for this thesis is to implement the machine learning model directly to the Wärtsilä's engine module. This approach was chosen because no additional hardware is needed to implement it straight to the engine module and it will be useful to see if the engine module CPU and memory usage will increase too much or remain at a suitable level when the machine learning model is running on the module.

The final hardware architecture of the machine learning system will consist of the WCD-20 module and a COM module. The WCD-20 will provide the necessary measurement information about the spark plugs to the COM module, which is responsible for several control functions, communication, software and engine configuration update management (Wärtsilä Marine solutions 2017). The machine learning model is located in the

COM module, thus making the COM module responsible for running the machine learning algorithm and calculating the spark plug diagnostic information from the WCD-20 data. The plausible CPU limitations must also be considered when developing the machine learning model, meaning that the model cannot be too complex that it requires more CPU and memory resources than those that are available to it.

4.2 Used software and equipment

The machine learning model will be implemented and trained initially on a desktop pc, and the used programming language will be Python. Python has a wide support for multiple machine learning and data analysis packages and libraries, such as Tensorflow, Numpy and Matplotlib, which make it an excellent choice to be used in the development of machine learning applications (Nimshi V. & S. Konam 1.5.2020).

Tensorflow is Google's machine learning platform, which is used to build and deploy machine learning models for a wide variety of systems and devices, and it will be used in the machine learning model of this thesis (Tensorflow 2020). Wärtsilä also already uses Tensorflow in their Expert insight system, which makes Tensorflow a practical choice to be also used in the WCD-20 spark diagnostic machine learning model (Wärtsilä 2019). The Tensorflow platform will be a critical part of the machine learning model, because after the model has been implemented and trained on a desktop pc, it can be saved to a file, for example with ".h5"- or ".tflite"-format, and the saved machine learning model file can be transferred to be used in the engine module (Tensorflow 2020b). The saved model will contain the trained weight values and the architecture of the model, thus making it usable without the need to initially train it on the engine module (Tensorflow 2020b).

The positive fact about a model which is saved as a ".tflite"-file is that it requires less cpu and memory resources to operate (Tensorflow 2020b), thus making it the most suitable option to be used on an embedded device like the engine module. The model cannot be

retrained however if it is saved as a ".tflite"-file, but the retraining process is not necessary in this case because training process is generally a part of a machine learning model that requires a lot of CPU time which the module has a very limited amount of, and it would need an expert to inspect the condition of the spark plugs that a running engine uses and inputting the inspected spark plug conditions as classes to the system, which is not feasible in this case. Better option in comparison to this is to collect the data and train the machine learning model under supervised conditions and create a new "tflite"-file which can be put to the embedded device to be used.

The engine module source code is written using c-programming language and the machine learning model will be used inside a c-program that controls the ignition energy and WCD-20. The part which opens, initializes and controls the machine learning model file will be implemented using the c++-programming language, because it supports the Tensorflow library and can open saved Tensorflow machine learning model files. The model can be tested using Wärtsilä's UNITool software, which is the tool generally used for configuration, tuning of engine parameters, troubleshooting, and loading software into the engine modules (Wärtsilä 2020b).

4.3 Machine learning technique and algorithm

To create and train the machine learning model, the used learning technique and machine learning algorithm must be selected. The used machine learning technique will be supervised learning because the available data that can be used for training is labelled. This makes it possible to teach the machine learning model which is a good spark plug and which is bad based on the selected features of the training data, thus making the input of the model the selected features from the training data and the output of the model the condition estimate of the spark plug. A machine learning algorithm that is suitable for supervised learning will be used and it will find the most suitable function to depict the relation between the input and the output of the model. In this thesis the machine learning algorithms candidates that I will consider are the regression algorithms and neural networks.

4.3.1 Neural network

One option in this thesis is to use a neural network to model the spark plug condition. Inputs in this algorithm are the selected features of a spark plug and the outputs are classes that tell in which condition a spark plug is currently in. A multiclass neural network is the most suitable alternative to solve this problem, because multiple classes can provide more options to make adjustments in ignition related parameters during the lifetime of a spark plug, than two classes implying only that a spark plug is in a good condition or in a bad condition. The adjustments can be made for example, to try to increase the spark plugs lifetime or to increase engine ignition performance. Better classes would be “very good”, “good”, “average”, “bad” and “very bad” because this kind of a division provides more accuracy for the user and for the possible ignition parameter adjustments.

The designed neural network will consist of multiple neurons that are divided into multiple layers, and the number of neurons and layers will depend on the prediction accuracy results and performance impact. The layers are connected to each other and if multiple layers will be used, then the hidden layers before the output layer will use an activation function such as sigmoid function or rectified linear function. Rectified linear function is defined in the equation below and it returns input value x if it is greater than zero, otherwise it returns a zero (Prajit R, B. Zoph & Q V. Le 16.10.2017: 1-2).

$$f(x) = \max(x, 0)$$

The output layer will consists of as many neurons as there are different classes in the created model, and each neuron will get an output value that will tell how likely it is that the current input belongs to the class that an output neuron represents. A Softmax function layer can be used as the last layer of the neural network to get the output values in probabilities which are easy to interpret. The model’s training process will consist of calculating the output of the model using the real training data and updating the weights

of the neural network using the error value that a loss function outputs. This neural network will use a loss function that is suited for multiple classification problems, such as a cross entropy loss function (Feng L., S. L. Shu, Z. Lin, F. Lv, L. Li & B. An 2020: 2206-2210).

The formula for the cross entropy loss is represented below and the data set used is $D = \{(x_i, y_i)|1 \leq i \leq m\}$ where $x_i \in X$ ($X \in \mathrm{R}^d$) is a d-dimensional feature vector and $y_i \in \{1, \dots, k\}$ is the label associated with the symbol x_i. A classifier is a function that is going to map the feature space to the label space $f: X \to \mathrm{R}^k$. In the formula $f_y(x)$ depicts the y:th element of $f(x)$ and the symbol e_y is a one-hot vector where $e_{yj} = 1$ if $j = y$, and otherwise 0. (Feng L., S. L. Shu, Z. Lin, F. Lv, L. Li & B. An 2020: 2206-2210)

$$L_{CCE}(f(x), y) = -e_y \log f(x) = \log f_y(x)$$

A stochastic gradient descent method will be used in the training process of the network to update the weights, because of the fast training speed of these methods and a general formula for it is represented below (L´eon Bottou 2012: 421-425). Stochastic gradient descent algorithm estimates the gradient of empirical risk function with a randomly picked example z_t during each iteration. The Symbol Q corresponds to a loss function and γ_t is the chosen learning rate. Values of w are the weights of the neurons and these values are updated in each iteration based on the estimated gradient of the empirical risk function. (L´eon Bottou 2012: 421-425)

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$$

A more optimized method to perform stochastic gradient descent is called adaptive moment estimation, which is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments (Kingma D. P. & J. L. Ba 2015: 1). The idea of this method is to calculate individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients, and it only requires first-order gradients to do so, which means that this

method is fast but also has a small memory consumption (Kingma D. P. & J. L. Ba 2015: 1-2). Because of these points, this could be the best method to be used in the learning process of the model.

The mathematical process of adaptive moment estimation is represented below and it consists of first updating the biased moment estimates m_{t+1} and v_{t+1} which are then used to compute bias-corrected versions of themselves, marked as $\hat{m}_{t+1}$ and $\hat{v}_{t+1}$. The parameters of the model are then updated in the last formula and new parameters are stored to w_{t+1}. Symbol t depicts the timestep parameter which increases by one after every iteration, ϵ is a small scalar to prevent division by zero and γ_t is the defined learning rate. Parameters $\beta_1, \beta_2 \in [0,1]$ control the exponential decay rates of the moment estimates and $\nabla_w Q_{t+1}(w_t)$ represents the gradients of the used loss function Q with parameters w at timestep $t+1$. (Kingma D. P. & J. L. Ba 2015: 2-4)

$$m_{t+1} = \beta_1 * m_t + (1-\beta_1)\nabla_w Q_{t+1}(w_t)$$
$$v_{t+1} = \beta_2 * v_t + (1-\beta_2)(\nabla_w Q_{t+1}(w_t))^2$$
$$\hat{m}_{t+1} = m_t/(1-{\beta_1}^{t+1})$$
$$\hat{v}_{t+1} = v_t/(1-{\beta_2}^{t+1})$$
$$w_{t+1} = w_t - \gamma_t * \hat{m}_{t+1}/(\sqrt{\hat{v}_{t+1}} + \epsilon)$$

4.3.2 Regression algorithm

A regression algorithm can be used to generate a graph that most accurately fits the input and output data according to a specified mathematic criterion. In this thesis the input data is the feature data of the spark plugs and the output data is the spark plug health condition's estimate value. Based on a given input to the regression model the output value of the health condition estimate could then be used to adjust the ignition parameters or inform it to the user. The scale of the output could vary between zero and one, and a value of zero could indicate that the spark plug is in a very good condition and while the output value increases the estimated health condition decreases.

The general degradation process of a spark plug is fairly linear but it slows down the further the process proceeds, thus implying that a simple linear regression could be suitable enough to model this phenomenon with pleasing accuracy (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 34-36). On the other hand, other conditions in addition to regular wearing can cause spark plugs to behave differently, which might render simple linear regression insufficient, but in this case a polynomial regression model can also be used to model this process if so desired.

In any case, the simplest mathematical formula for this linear regression machine learning model is the formula represented below, where β values are the weigth coefficients which are altered in the training process. The variable y corresponds to the predicted spark plug health condition value, x variable is the used value of the selected spark plug measurement feature and e is the error value. If more than one input feature is used the model's formula will be similiar to the multiple regression formula presented in the chapter 2.2.1.

$$y = \beta_0 + \beta_1 x + e$$

The training process of this regression model will consists of calculating the output of the model with the training data, calculating the error value of the selected loss function and updating the weigths with the error value. A stochastic gradient descent method can also be used to update the weights in this model (L´eon Bottou 2012: 423-430). The used loss function in this design is either the mean squared error-function, which was also presented in the chapter 2.2.1, or the mean absolute error function.

Mean absolute error function differs from the mean squared error in the way that it does not take into account the direction of the error but only the magnitude of it (Prince Grover 5.6.2018). The absolute value loss function is the first formula represented below and the empirical risk function for it, which is called the mean absolute error, is the the second formula represented below (Shai S. S. & B. D. Shai 2014: 123-124). The $h(x)$ and

$h(x_i)$ represent the actual output values, y and y_i are the output values from the implemented machine learning model, and the symbol m is the total number of data points.

$$l\big(h(x,y)\big) = |h(x) - y|$$

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}|h(x_i) - y_i|$$

4.4 Data

The data that will be used to train and validate the model will be collected using a spark plug test rig machine. Different spark plugs in different health conditions will be tested and the collected results from the tests will be saved. During this process, the spark plugs will be labelled according to their health conditions. The saved data will be arranged in such a format that the columns will depict the different measured properties of the spark plugs and ignition, and the rows of the saved data are the individual measurement points over a certain amount of time. The amount of spark plugs that are available for testing is limited and the conditions of them don't necessarily represent every possible spark plug health condition, thus affecting the model's overall accuracy and performance.

The planned measured properties include the spark voltage, the coefficient of variation of the voltage, primary open current and a status indicating if the coefficient of variation of the voltage is high or not. These values are selected because of their established connection to a spark plug's health (Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi 2013: 32, 37), and these values are also obtainable to be used in the engine module. The coefficient of variation CoV is calculated by dividing the standard deviation of the spark voltage σ with the mean of the spark voltage μ. The formula is represented below.

$$CoV = \frac{\sigma}{\mu}$$

4.5 Testing

The accuracy of the machine learning model will be tested using the available spark plug data and performing predictions to it. This accuracy measure will be used in analysing the model's performance. The difference between the model's prediction value versus the real value will be analysed and this testing will determine the accuracy of the model. The $x_{number\ of\ errors}$ is the total amount of errors occurred during the testing of the model and $x_{number\ of\ all\ observations}$ is the total number of all observed data points.

$$Model\ accuracy = 1 - \frac{x_{number\ of\ errors}}{x_{number\ of\ all\ observations}}$$

The model's average absolute error value is also tested, meaning that the average absolute error will be calculated with test data. The regular average error will also be calculated using the test data, and this will tell whether the model classifies the mistakes too high or too low on average compared to the real class values. In the first formula below n corresponds to the number of errors, y depicts the predicted class value and x values are the real class values. The second formula below is for calculating the direction of the error and same variables are used there.

$$average\ absolute\ error = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

$$average\ error = \frac{\sum_{i=1}^{n} y_i - x_i}{n}$$

The precision and recall of every predicted class will be tested, and these are particularly useful when inspecting how well the model can classify certain classes correctly. Precision determines the proportion of results for a predicted class that really belong to this predicted class. The formula for precision is the first formula represented below and $True\ Positives$ is the amount of outputs that were correctly classified in to the

class under inspection, and the $False\ Positives$ depicts the amount of classification outputs that were incorrectly classified as the class that is currently being inspected. Recall on the other hand measures the proportion of correct classification outputs for a given class and the formula for Recall is the second formula below. $False\ Negatives$ in this formula presents the amount of false classification results for a class under inspection. (David M. W. Powers 2007: 1-3)

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

5 Model implementation

This chapter presents the implementation process of the machine learning model and the model design part of this thesis is used as a guideline during the implementation process. The implementation of the model starts with performing tests and gathering data from several spark plugs in different conditions. When the testing was complete, the machine learning model was developed with Python programming language using Tensorflow library, and this model is introduced later in this chapter.

5.1 Gathering data for the model

To get the data for the model, several spark plugs in different conditions were tested totaling in 15 different spark plugs. These conditions included spark plugs in normal good condition, lead fouled plugs, sooted-carbon-fouled plugs, plugs with worn electrodes, plugs with ash formation or a combination of these. This testing was done using a spark plug test rig machine, which simulates a real engine, in one of Wärtsilä's laboratories. The tests consisted of:

- Switching different spark plugs in to the spark plug test rig machine.
- Powering on the spark plug test rig machine and connecting it to a pc with ethernet cable
- Commencing the tests by setting the test rig machine into run mode.
- Monitoring and gathering data using Wärtsilä's UNITool software.

The gathered data during these tests consisted of spark voltage, the coefficient of variation of the spark voltage, primary open current, a status indicating if the coefficient of variation of the voltage is high or not, and in addition to these planned features the used spark profile was also measured. The spark profile indicates which kind of a profile is used to generate the spark, for example engine start has a different profile than average operation.

Before installing different spark plugs into the spark plug test rig machine, the spark plugs were categorized into three different classes, “Good”, “Average” and “Bad”, based on their current conditions and history of use. These new formed classes were then used to label the tested spark plugs accordingly so that each spark plug belonged to one of these three classes and the measured data for every spark plug could be associated with one of these three classes as well.

After these tests were finished the results were exported to a “.csv”-file to be able to be used in the machine learning model. The final form of the data consisted of five columns representing the measured features during the testing and approximately 1000 data points for every test. Every tested spark plug also had the label which indicated the class of the spark plug.

It was clear from the gathered data that typically spark plugs in bad condition had higher coefficient of variance of the ignition voltage and higher ignition current than spark plugs in good condition. The average coefficient of variance of the ignition voltage was 83% larger than the average value of the good spark plugs, and the average values of the ignition current were 4.4% larger than the values of the good spark plugs.

Exceptions still occurred, for example when simulating the engine start with the spark plug test rig machine spark plugs of all conditions had higher ignition voltage coefficient of variation values compared to running the test rig in normal mode, and the values of some good spark plugs could even be briefly on par with some bad spark plugs. This indicates that singular predicted values of a class for a given spark plug may not be enough to solely classify a spark plug into a category, but a larger number of classification results overtime could be averaged and based on that a spark plug could then be categorized with better accuracy.

5.2 Implementing the model

There were two algorithms that were presented as possible choices to be used in the machine learning model of this thesis, a neural network and a regression algorithm, and the chosen machine learning algorithm for this thesis was a neural network. This was mainly because the tested spark plugs were labeled into three different classes meaning that output classification based on the input data could be done accordingly to the formed classes and also because different kinds of health conditions that spark plugs may have can produce different types of behavior to the input features than regular wearing. Neural networks are also generally good algorithms to be used in classification problems (Vansh Jatana 2019: 1-4), which further encouraged to choose a neural network to be used in this thesis. Regression based algorithm could have also been a plausible choice, but regarding these points related to the spark plug labeling and multiple types of conditions spark plugs may have, the neural network was chosen as a more suitable algorithm for this machine learning model.

The model was implemented with Python programming language and the Tensorflow library was used in the implementation process. The first part of the implementation consists of reading the measured test data from ".csv"-files into variables and concatenating all data into a one array that contains all of the data that is used the training process of the model. Some data is also reserved to do some testing with the machine learning model and this same procedure is done separately to this testing data as well.

The input is first normalized when it enters the machine learning model. This normalization is done by a Tensorflow preprocessing layer which is inserted into the "normalizer" variable with a correct input shape (Tensorflow 2020e). This "normalizer"-variable is then adapted to the input data using "adapt()"-function with variable "df_allplugs", which contains all data that will be used in the training of the model. The normalization is done to scale all the inputs, because some values of the features are much larger than

the values of other features, for example, ignition voltage versus high coefficient variance status. This will prevent these larger values from dominating the other values in the training process, thus making the neural network more accurate in this case.

```
#prepare the neural network with normalization of data
normalizer = preprocessing.Normalization()
normalizer = preprocessing.Normalization(input_shape=[5,])
normalizer.adapt(np.array(df_allplugs))
```

The neural network model is constructed using Tensorflow-library's "keras.Sequential()" function, which groups a linear stack of layers into Tensorflow keras model, which also provides training and inference features on the model (Tensorflow 2020c). The created model has four layers, and this amount was chosen because it was the minimum amount that is required for a multiple classification Tensorflow model of this kind to work properly, thus also requiring the least amount of processing power to operate.

The first layer is the normalization layer which normalizes the input data according to the data that has been adapted to it. The second layer is a densely connected neural network layer of 16 neurons per feature, which uses a rectified linear activation function that was considered in the design part of this thesis. This second layer of the neural network gets it's input from the normalized input feature vector from the normalization layer and the output of the second layer is connected to the third layer which is also a densely connected neural network layer with three neurons. Each of these three neurons corresponds to one of the three classes "Good", "Average" and "Bad" that were formed during the spark plug test rig machine testing. The final layer of the neural network is a Softmax-layer as was planned during the design part of this thesis. The Softmax-layer is used to calculate the Softmax-activation function values, which are the probabilities for each of the three classes represented as an array of three values. The highest probability out of the three is then chosen to be the right classification result.

```
#create the model, 3 Dense outputs = 3 classes => attached
to a #Softmax-layer to get the final output as probabilities
neural_network_model = tf.keras.Sequential([
    normalizer,
```

```
    layers.Dense(16, activation='relu'),
    layers.Dense(3),
    tf.keras.layers.Softmax()
])
```

The next step is to configure the training process of the machine learning model. This is done by using the Tensorflow library's "Model.compile()"-function, which allows to select the optimizer or in other words the way the model's weights are updated, loss function and the metrics which provide information about the training and testing of the model (Tensorflow 2020e). In the model of this thesis the "optimizer" that is used is the adaptive moment estimation method, which is called "Adam" in Tensorflow's syntax (Tensorflow 2020c). Adaptive moment estimation was reintroduced in the design part of this thesis and it was chosen to be used because of the performance and speed this method provides. The used loss function is the "SparseCategoricalCrossentropy"-function, which calculates the regular cross-entropy loss values between the predictions and the labels during the training process of the model (Tensorflow 2020c). This loss function was chosen because it is suitable for multiclass classification problems and the model of this thesis does that. Finally, the information about the training process is also configured to display as "accuracy" metrics.

```
#configure the training process of the model
neural_network_model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.01),
    loss=tf.keras.losses.SparseCategoricalCrossen
tropy(from_logits=True),
    metrics=['accuracy'])
```

When the model is constructed and the training process is configured, the model's training can be performed. The Training is done by using Tensorflow libarary's function "Model.fit()", where the number of epochs, or in other words iterations over the entire input and output of the model, training data, training labels, logging features and validation of the trained model are configured (Tensorflow 2020c). The variables "df_allplugs" and "train_labels" correspond to the training data and training labels respectively, and "verbose" corresponds to the logging information that is displayed. The "validation_split" in this model is configured to be 20% of the data used in the training process, and this

proportion is randomly taken out of the training data to be solely used in the validation process.

```
#configure the training process of the model
#perform the training of the model
epochs_number=100
history = neural_network_model.fit(
    df_allplugs, train_labels,
    epochs=epochs_number,
    # suppress logging
    verbose=0,
    # 20% training data used in validation
    validation_split = 0.2)
```

When the model is trained the performance statistics about the training can be inspected and the model is ready to do classifications on real data. The classifications are done by using Tensorflow library's function "Model.predict()"-function, and inserting the desired input data of right dimensionality to this function (Tensorflow 2020e). The model of this thesis outputs an array of three numbers as probabilities for the classes "Good", "Average" and "Bad" and these probability values can be inspected to choose the greatest of them, when the input is an array containing five columns that represent the features. This trained model can be saved in "tflite"-file format so that it can be used later or so that it can be put into an embedded device (Tensorflow 2020f).

5.3 Engine module implementation

To run the created tflite-file in the engine module, which contains the machine learning model, the Tensorflow library must be included into the project which contains the engine module source code. The model also must be downloaded separately into the engine modules filesystem, where it is stored in its own folder. After these steps the model can be loaded to be used by creating an object from "FlatBufferModel" class and using a function "BuildFromFile(filename)", where the filename corresponds to the used model's filename and entire path to the model (Tensorflow 2020f).

```
pModel = tflite::FlatBufferModel::BuildFromFile(filename);
```

To put input and get the output classification results from the loaded model, an “Interpreter” object is needed. This object is constructed by calling “InterpreterBuilder” from the “tflite”-library and inputting the model and resolver into it. The resolver is used to register the interpreter operations with the used kernel library. Then the “Interpreter”-objects “AllocateTensors()”-function is used to allocate memory for the inputs and outputs of the model. The model’s input size and output size can be specified separately, but in this case the size of the input is five and the size of the output is three, according to the used input features and output classes. (Tensorflow 2020f)

```
 tflite::InterpreterBuilder(*pModel,*pResolver)(&pInter-
preter);
```

To make the model do classification, a separate function called TensorFlowRun() is used. This function takes a struct that contains the required input features and the number of current cylinder as it’s inputs. This function then outputs the classification result as a number that present the inspected spark plugs condition class, and these numbers are either 0 meaning good, 1 meaning average and 2 meaning bad.

In this function an object of Tensorflow class is used to put the input into the model and get the output from the model, and this object is called ”TensorFlow”. The input is created by using the struct that contains the relevant data and taking these values into the variables which are added as inputs to the input tensor in the AddInput()-function below. After this the RunInference()-function is called where the interpreter object’s Invoke()-function is used to calculate the output from the input. If this succeeds the the calculated output is taken into the tensorflowOutput variable from the output tensor in the GetOutput() -function. Finally, to get the highest probability value for a predicted class, the three output values are gone through in the for-loop and the result are compared to each other. The index of the highest probability value is then stored into the maxindex variable and returned.

```
TensorFlow.AddInput(
    std::array<float,  5>{SparkVoltage,  covVoltage,  pri-
maryOpenCurrent, highCOVstatus, code});

if (TensorFlow.RunInference())
{
     tensorflowOutput = gTensorFlow.GetOutput<float>();
}

// go through 3 classes and return the index of the highest
//probability value
for (i = 0; i < 3; i++)
{
     if (tensorflowOutput[i] > tensorflowOutput[maxindex])
     {
          maxindex = i;
     }
}

return maxindex;
```

The model is first initialized and after this it is used in a diagnostic loop that calculates cylinder related diagnostics and values. The TensorFlowRun()-function is called in each iteration of the cylinder diagnostic loop and the spark plug condition classification results are calculated for each cylinder separately.

6 Model training and testing

The testing of the implemented machine learning model occurs right after the training process as the validation of the trained model with the validation part of the training data and the other part of the training includes testing the model with new unforeseen data from spark plug rig machine measurements. In this chapter these test results are gone through in detail and the model's performance is analyzed.

6.1 Training and validation

The data that was used in the training process of the model consisted of 19200 different data points of measurements from the spark plug test rig machine. The values were from all three types of labeled spark plugs, "Good", "Average" and "Bad", and the data was stored in "csv"-format. 80% of this data was used in the training process and 20% of this data was used in the validation process of the model. This split is widely used in many different areas such as economics and it generally provides good results (Mike Vladimer 2018).

The model's training process was done multiple times with different values for the number of epochs, learning rate and the number of neurons, but initially the learning rate was 0.10 and the number of neurons per feature was 64. With these numbers the model was able to achieve accuracy between 50%-70% in the training and validation parts of the process. The model most likely suffered from the high learning rate and found a local maximum too quickly and started converging towards it too soon. When lowering the learning rate to 0.01 the model started to achieve better results of around 75%-85%. Lowering the number of neurons to 16 per feature to make the training process faster and the model simpler did not reduce the accuracy and the final model was trained with such parameters. Most likely the reason why the reduction of neurons did not impact the accuracy was because the spark plug diagnostic problem with these parameters does

not require more than 16 neurons to be solved. There are a limited number of parameters that affect the outcome of the neural network and a certain number of neurons in relation to these parameters is needed to perform classifications with good accuracy.

During the training process the model is validated with 20% of the training data that is reserved to do this validation. The picture 13 below depicts the training process and different performance metrics can be seen from it. These metrics include the loss, validation loss, accuracy, and validation accuracy of the model. The model is trained over 140 epochs and the accuracy that it reaches with the training data is 85%. The accuracy with the validation data is 82%, which is a little lower than the training data accuracy. These results imply that the model can correctly classify 82%-85% of the spark plug data that it is given to it, and that the model has most likely been able to find a suitable generalization for the spark plug classification diagnostic problem. Based on these training results no undertraining has occurred and most likely no over-training, at least not a significant amount.

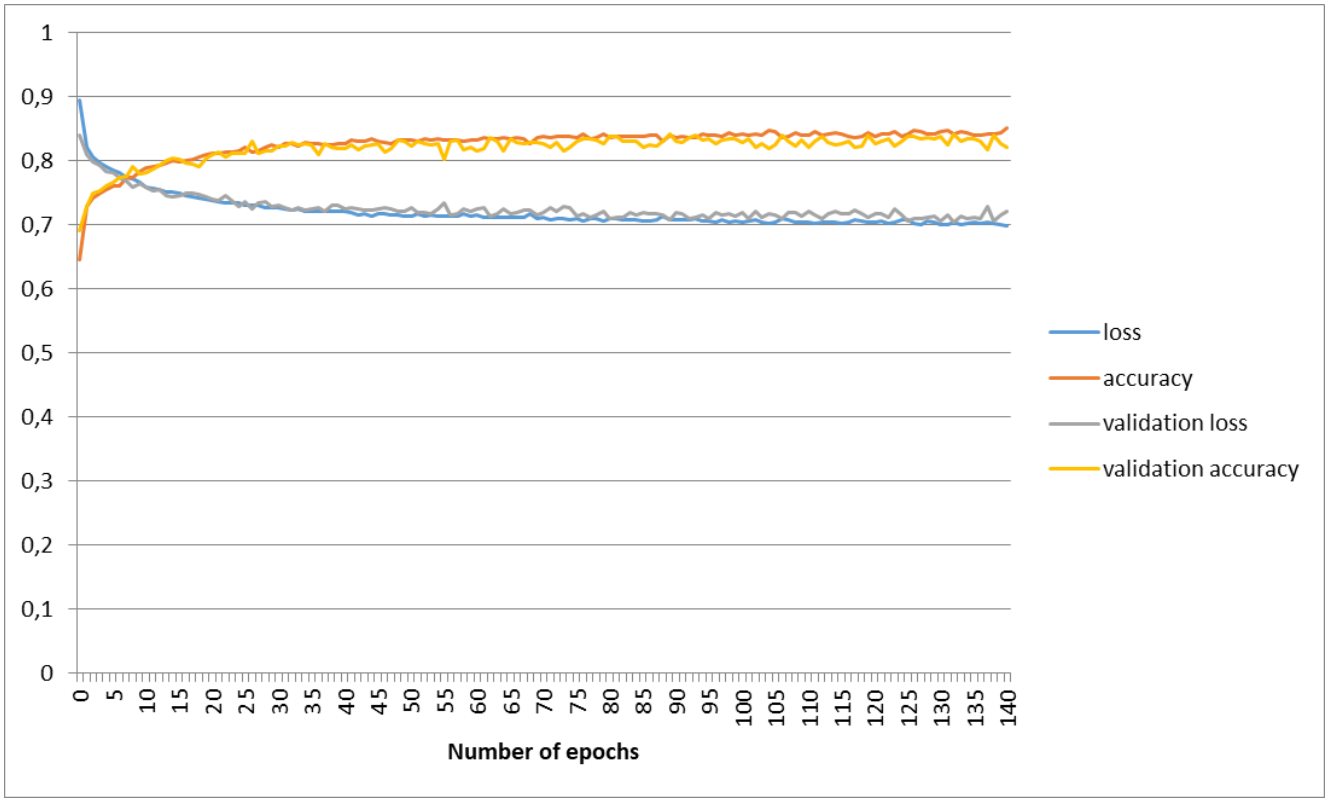

Figure 13 The model's training and validation processes.

6.2 Testing the model with real data

The trained model is tested with real spark plug test rig measurement data that was not used in the training process, thus being unforeseen data for the model. This testing data consists of 3180 data points of spark plugs in all three conditions. This testing is performed by constructing a new array that holds feature data in the columns from spark plugs of all three classes and creating a label array that has the corresponding labels for each entry in the new data array. The model is then used to predict the class of every entry in the data array and this result is compared to the label array's value.

The accuracy that was obtained from this testing was 82%, which is in line with the accuracy of the model validation. This further suggests that the model is indeed able to correctly classify that percentage of the spark plugs and it has found a way to generalize this relation between the model's inputs and outputs.

The average error that the model produces is -0,937 and the average absolute error is 1.50. The average error value implies that when the model misclassifies the output it on average classifies it better as it is, meaning that if the true output should be “Bad” the model classifies it as “Average”. On the other hand, the average absolute error value tells that on average the model misclassifies the outputs as either one or two classes different than the real value should be.

The precision and recall values of the model are presented in the table 2 below. The class “Good” has the highest recall value of 99,8%, which means that the model classifies that amount of actually good spark plugs into the “Good” class, and the remaining 0.2% is misclassified to the other classes. The lowest value for the recall is in the class “Bad”, and this tells that real bad spark plugs are being misclassified into the other classes the most. The precision values are more even, but the highest precision value of 87,8% is in the “Average” class. This implies that 87,8% of output results from this testing that were classified as “Average” were average spark plugs. The precision of 78,9% from the class “Good” was the lowest by a small margin and it means that most of the misclassifications that the model does are from the classes “Average” and “Bad” to the class “Good”.

Table 2. Precision and recall of the model.

Class	Precision	Recall
Good	78,9%	99,8%
Average	87,8%	85,2%
Bad	80,4%	61,4%

When further analyzing these recall and precision values, it becomes clear that the model can distinguish a good spark plug from the average and bad spark plugs with decent accuracy of 99,8%. This is a good thing because it means that when the model outputs a result of either class “Average” or “Bad” the result most likely is either of those classes and not misclassified value of a class “Good”. This should prevent false alarms about spark plug being bad while it is good, thus avoiding a spark plug being replaced in vain.

The lowest recall value of 61,4% from the class “Bad” is worrisome. The 48,6% of the bad spark plugs are being misclassified into the other two categories and this can cause some bad spark plugs to be unnoticed by this model. It is not that critical if a bad spark plug is miscategorized as an average spark plug while being a bad spark plug by the model, but because the absolute average error was 1.50 it implies that a bad spark plug can be classified as a good spark plug as well. However, the precision values for each of the classes are relatively good and close to the model’s overall accuracy of 82% that was obtained in this test, and they provide some indication about the model’s overall decent performance in this test.

Overall, this model produced mixed results from the testing. On one hand, the model’s overall accuracy of 82% is pretty good and the model can classify good spark plugs into the right category with satisfying accuracy, thus giving the model some credibility when it outputs that a spark plug is in an average or bad condition. On the other hand, when the model outputs that a spark plug is in a good condition, there is a 21,1% chance that the spark plug actually is in an average or bad condition, which can be negative and lead to problems in some diagnostic cases.

6.3 Engine module performance

The Tensorflow model’s performance impact was tested on engine module. This was done by monitoring the total CPU usage during a one-minute period three times with and without Tensorflow model in Wärtsilä’s UNITool program and calculating the average

of these measured values. The CPU usage increased by 4,3% when using the Tensorflow model, which is a moderate increase. This moderate impact is caused by the Tensorflow model's matrix multiplications in the dense layers which are performed for every spark plug in every cylinder separately during a single ignition diagnostic calculation loop.

One straightforward way to decrease this impact to the CPU's performance still exists. This way is to perform the spark plug health condition classification less often, for example only in every 10^{th} ignition diagnostic loop. This way the calculations needed in the Tensorflow model are not performed as frequently and this decreases the CPU usage.

7 Conclusions

This thesis presented a way to classify spark plug health conditions into three different classes by utilizing machine learning and the available data from the WCD-20 engine module. This was done because more diagnostic information about the spark plugs in the engine is needed and investigating if machine learning could be an effective way of doing this could give more information about machine learning's capabilities in engine applications.

The machine learning model was designed by first introducing machine learning overall and focusing on the most suitable approaches from plausible learning techniques and algorithms. The chosen learning technique was the supervised learning because of the labeled data that was gathered to be used in the training process and the selected algorithm was a neural network. The algorithm choice was mainly done due to the neural network's ability to do well in classification problems.

Implementation process consisted of first gathering the data from spark plugs in different conditions and developing the designed machine learning model. After this the model was trained and transferred into the engine module. The engine module implementation was chosen because no additional hardware was required to run the machine learning model, even though the CPU was a plausible a bottleneck. In the testing part the model increased the engine module's CPU usage by 4,3%, which indicates that the model might be too demanding to be operated directly in the engine module during every ignition diagnostic loop. This problem however can be alleviated by running the model less often in the diagnostics calculation loop.

The developed machine learning model was tested and the overall accuracy from this testing was 82%. This indicates that the developed model was able to correctly classify 82% spark plugs from the test data. Other test metrics such as recall and precision were also measured and on one hand, the high recall value 99,8% of the "Good" spark plug class was a positive outcome and indicates that the model is able to separate good spark

plugs from the other with high accuracy. On the other hand, the lower recall value of 61,4% of the “Bad” spark plug class tells that the model is not as good in distinguishing bad spark plugs, leading to more misclassifications for the bad spark plugs. The relatively high precision values for every class still gives some indication of the model’s overall decent performance.

Overall based on the training validation and final testing accuracy values, it can be concluded that the model has learned to classify spark plugs by using the given input features with the tested accuracy of 82% and that it is possible to use machine learning to diagnose spark plug health condition. It still does misclassifications in some cases, but the accuracy could possibly be improved by getting more spark plugs and gathering more data for the training process.

Nevertheless, this developed spark plug health condition diagnostic machine learning model can be used in the future, or a different model can be implemented by utilizing the same design and development process, and slightly modifying the engine module implementation, if so desired. If the approach of this thesis is further developed, the next development step from this point on would be to use the machine learning model’s output to adjust the ignition related parameters. This could be done so that for example, the spark plugs that are classified as bad will have increased ignition voltage values and those that are classified as good could have lower ignition voltage values to prolong the life of the spark plugs.

If a simpler and less demanding way to diagnose spark plugs is required than a machine learning version, one possibility is to define different thresholds for the inspected values, such as the coefficient of variation of the voltage, and these could be monitored without machine learning. If these defined thresholds are exceeded for example a certain amount of times or too many times consecutively a spark plug could be diagnosed as being “Bad” for example.

References

Ahmed A. Abdel-Rehim (1.6.2013 Impact of spark plug number of ground electrodes on engine stability. *Ain Shams Engineering Journal* [Online document] 4: 2 [3.9.2020], 307-316. Availabe: https://www.researchgate.net/publication/271603800_Impact_of_spark_plug_number_of_ground_electrodes_on_engine_stability

Altronic (2020). CPU-XL VariSpark ADVANCED DIGITAL IGNITION SYSTEM FOR LARGE GAS ENGINES (Altronic) [online]. [15.9.2020] Available: http://www.altronicinc.com/pdf/cpuXLvarispark/CPU-XL%20VariSpark%2011-12.pdf

Ashutosh S. & Y. Li (3.12.2017). To Study Implementation of Gradient Descent for Multi-class Classification Using a SoftMax Regression and Neural Networks (rstudio) [online]. [8.9.2020] Available: https://rstudio-pubs-static.s3.amazonaws.com/337306_79a7966fad184532ab3ad66b322fe96e.html

Bosch (2019). Spark plug condition identification (Bosch) [online]. [3.9.2020] Available: https://www.boschautoparts.com/documents/101512/0/0/ec0f482d-e566-490d-9132-1dadd0d5def6

David M. W. Powers (2007) Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation *Technical Report SIE-07-001* [Online document] 1: 1 [16.11.2020], 1-24. Availabe: https://web.archive.org/web/20191114213255/https://www.flinders.edu.au/science_engineering/fms/School-CSEM/publications/tech_reps-research_artfcts/TRRA_2007.pdf

Deisenroth M. P, A.A. Faisal & C. S. Ong (1.4.2020). *Mathematics for machine learning*. London: Cambridge University Press. ISBN 978-1108455145.

Developers.google (10.2.2020). Multi-Class Neural Networks (Developers.google) [online]. [4.9.2020] Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/video-lecture

eeweb (2020). Introduction to Automotive Ignition Systems. (eeweb) [online]. [15.9.2020] Available: https://www.eeweb.com/introduction-to-automotive-ignition-systems/

Elite data science (2019). Overfitting in Machine Learning: What It Is and How to Prevent It (Elite data science) [online]. [1.9.2020] Available: https://elitedatascience.com/overfitting-in-machine-learning

elprocus (2020). Capacitor Discharge Ignition(CDI) Working Principle, Its Advantage and Disadvantage (elprocus) [online]. [15.9.2020] Available: https://www.elprocus.com/capacitor-discharge-ignition-cdi-system-working/

Ethem Alpaydin (17.3.2020). *Introduction to Machine Learning fourth edition.* London: The MIT Press. ISBN 9780262043793.

Feng L., S. L. Shu, Z. Lin, F. Lv, L. Li & B. An (2020) Can Cross Entropy Loss Be Robust to Label Noise? *Joint Conference on Artificial Intelligence* [Online document] 1: 29 [7.10.2020], 2206-2212. Availabe: https://www.ntu.edu.sg/home/boan/papers/IJCAI20_Entropy.pdf

Hao Karen (17.11.2018). What is machine learning? (MIT Technology Review) [online]. [4.9.2020] Available: https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/

Industrial gas engine controls (2020). Ignition technology explained. (GILL Sensors & controls) [online]. [15.9.2020] Available: https://www.gillsc.com/assets/Uploads/Ignition-Technology-Explained.pdf

Javan S., S. V. Hosseini, S. S. Alaviyoun & F. Ommi (9.6.2013). Effect of electrode erosion on the required ignition voltage of spark plug in CNG spark ignition engine. *The Journal of Engine Research* [Online document] 26: 1 [2.9.2020], 31-39. Availabe: https://www.semanticscholar.org/paper/Effect-of-electrode-erosion-on-the-required-voltage-Javan-Hosseini/eadd43d1dac9fd943c751b9f774fe07f0e886f41

Kalapanidas E., N. M.Avouris, M. V. Craciun & D. Neagu (1.1.2003). Machine Learning algorithms: a study on noise sensitivity. *ResearchGate* [Article] [1.9.2020], 1-11. Availabe: https://www.researchgate.net/publication/238478263_Machine_Learning_algorithms_a_study_on_noise_sensitivity

Kingma D. P. & J. L. Ba (2015) ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION *International Conference for Learning Representations* [Online document] 1: 3 [9.10.2020], 1-15. Availabe: https://arxiv.org/abs/1412.6980

L´eon Bottou (2012) Stochastic Gradient Descent Tricks? *Lecture Notes in Computer Science* [Online document] 1: 1 [7.10.2020], 421-436. Availabe: https://link.springer.com/chapter/10.1007/978-3-642-35289-8_25

Mehryar M., A. Rostamizadeh & A. Talwalkar (25.12.2018). *Foundations of Machine Learning, second edition.* London: The MIT Press. ISBN 9780262351362.

Milan S., M. Kucera, M. Gutten & D. Korenciak (1.1.2019). THERMAL AND ELECTRICAL ANALYSIS OF AUTOMOTIVE IGNITION SYSTEM. *The Archives of Automotive Engineering* [Online document] 83: 1 [1.9.2020], 113-122. Availabe: https://www.researchgate.net/publication/238478263_Machine_Learning_algorithms_a_study_on_noise_sensitivity

Nimshi V. & S. Konam (1.5.2020). Abridging clinical conversations using Python (Python software foundation) [online]. [29.9.2020] Available: https://www.python.org/success-stories/abridging-clinical-conversations-using-python/

Prajit R, B. Zoph & Q V. Le (16.10.2017) Searching for Activation Functions. *Cornell University* [Online document] 1: 1 [7.10.2020], 1-13. Availabe: https://arxiv.org/abs/1710.05941

Prince Grover (5.6.2018). 5 Regression Loss Functions All Machine Learners Should Know (Heartbeat) [online]. [5.10.2020] Available: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

Raman K Autar (1.5.2004). Diesel Engine Maintenance – An Expert System Approach Utilizing the Multi-Sensor Criterion. *ResearchGate* [Online document] [2.9.2020], 1-22. Availabe: https://www.researchgate.net/publication/261835392_Diesel_Engine_Maintenance_-_An_Expert_System_Approach_Utilizing_the_Multi-Sensor_Criterion

Roberto Iriondo (16.8.2018). Machine Learning (ML) vs. Artificial Intelligence (AI) — Crucial Differences (medium) [online]. [16.12.2020] Available: https://medium.com/towards-artificial-intelligence/differences-between-ai-and-machine-learning-and-why-it-matters-1255b182fc6

Shai S. S. & B. D. Shai (15.5.2014). *Understanding Machine Learning From Theory to Algorithms.* London: Cambridge University Press. ISBN 9781107057135.

Tensorflow (2020). Tensorflow (Tensorflow) [online]. [29.9.2020] Available: https://www.tensorflow.org/

Tensorflow (2020b). Save and serialize models (Tensorflow) [online]. [29.9.2020] Available: https://www.tensorflow.org/guide/keras/save_and_serialize

Tensorflow (2020c). Module: tf.keras (Tensorflow) [online]. [9.11.2020] Available: https://www.tensorflow.org/api_docs/python/tf/keras/

Tensorflow (2020d). tf.keras.layers.Dense (Tensorflow) [online]. [9.11.2020] Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

Tensorflow (2020e). Basic classification: Classify images of clothing (Tensorflow) [online]. [10.11.2020] Available: https://www.tensorflow.org/tutorials/keras/classification

Tensorflow (2020f). TensorFlow Lite guide (Tensorflow) [online]. [10.11.2020] Available: https://www.tensorflow.org/lite/guide

University of Calgary (4.1.2019). Four stroke engine (University of Calgary) [online]. [1.9.2020] Available: https://energyeducation.ca/encyclopedia/Four_stroke_engine

Vansh Jatana (1.6.2019). Machine Learning Algorithms. *ResearchGate* [Online document] [25.8.2020], 1-4. Availabe: https://www.researchgate.net/publication/332902498_Machine_Learning_Algorithms

Vladimer Mike (20.3.2018) This '20/80 Rule of Big Data' has huge implications for IoT tech (Medium) [online]. [17.1.2021] Available: https://medium.com/internet-of-things-from-osv/this-20-80-rule-of-big-data-has-huge-implications-for-iot-tech-e8e7cdf42387

Vladimir Nasteski (11.12.2017). An overview of the supervised machine learning methods. *ResearchGate* [Online document] [25.8.2020], 1-11. Availabe: https://www.researchgate.net/publication/328146111_An_overview_of_the_supervised_machine_learning_methods

Wouter K., P. Coombes & G. Couvert (2019). Spark plugs (Denso) [online]. [3.9.2020] Available: https://www.denso-am.eu/media/1108386/m28410_spark_plug_manual_all_x.pdf

Wärtsilä (2019). Wärtsilä 31SG (Wärtsilä) [online]. [2.9.2020] Available: https://www.wartsila.com/marine/build/engines-and-generating-sets/pure-gas-engines/wartsila-31sg

Wärtsilä Engines (2011). Wärtsilä 34SG Engine Technology (Wärtsilä) [online]. [3.9.2020] Available: http://www.gastopowerjournal.com/documents/34SG_ET_082011.pdf

Wärtsilä Engines (30.10.2019). Wärtsilä 31SG Product guide (Wärtsilä) [online]. [3.9.2020] Available: https://www.wartsila.com/docs/default-source/product-

files/engines/wartsila-31sg-product-guide.pdf?utm_source=engines&utm_medium=puregasengine&utm_term=w31sg&utm_content=product+guide&utm_campaign=msleadscoring

Wärtsilä Marine solutions (2017). Wärtsilä Unic engine system overview (Wärtsilä) [online]. [29.9.2020] Available: https://cdn.wartsila.com/docs/default-source/product-files/engines/brochure-o-e-unic-ms.pdf

Wärtsilä (2019). Boosting asset reliability with proactive high-level support (Wärtsilä) [online]. [29.9.2020] Available: http://www.wartsilareports.com/en-US/2019/ar/stories/boosting-asset-reliability-with-proactive-high-level-support/

Wärtsilä (2020b). Wärtsilä UNIC for gas engines (Wärtsilä) [online]. [29.9.2020] Available: https://cdn.wartsila.com/docs/default-source/service-catalogue-files/wärtsilä-unic-for-gas-engine.pdf?sfvrsn=6600b844_6