Improving Vehicle Diagnostic Solutions for Heavy-Duty Vehicles through Machine Learning

A study on preventive fault detection in fuel injectors through the use of warranty claim data, operational data and fault codes

Master's thesis in Systems, Control and Mechatronics

GABRIELLA GALONJA

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

Master's thesis 2020

Improving Vehicle Diagnostic Solutions for Heavy-Duty Vehicles through Machine Learning

A study on preventive fault detection in fuel injectors through the use of warranty claim data, operational data and fault codes

GABRIELLA GALONJA

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communication Systems
Chalmers University of Technology
Gothenburg, Sweden 2020

Improving Diagnostic Solutions for Heavy-Duty Vehicles through Machine Learning
A study on preventive fault detection in fuel injectors through the use of warranty claim data, operational data and fault codes
GABRIELLA GALONJA

Supervisor: Shen Li, Electrical Engineering
Supervisor: Kausihan Selvam, Volvo Technology AB
Examiner: Tommy Svensson, Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Division of Communication Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Improving Diagnostic Solutions for Heavy-Duty Vehicles through Machine Learning
A study on preventive fault detection in fuel injectors through the use of warranty claim data, operational data and fault codes
GABRIELLA GALONJA
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Fault prediction in the automotive industry takes on an important role as hardware failure tends to stem from the hardware degrading over time. Thus, it is of interest to predict a failure before it occurs, for preventive measures. As data from the workshop is collected by the electronic control unit (ECU) from the sensors, deviations with regards to e.g. the engine can be detected, and are in turn saved as diagnostic trouble codes (DTCs). Based on these generated fault codes, operations are conducted on the heavy-duty vehicle. This thesis aims to investigate whether warranty claim data, together with operational data and fault codes generated in the workshop, can be used in order to predict failures in fuel injectors. Another aim is to gain insights of how the diagnostic solutions are used in the field when further replacements of fuel injectors are not required. This was achieved through the development of two machine learning algorithms: random forest and long-short term memory. The results imply that the warranty claim data, together with the operational data and generated fault codes, give useful insights to the failure of fuel injectors and how they can be prevented. The attained balanced accuracy of the random forest algorithm was at 82.8%, where the ability to predict a faulty injector was 66.1% as opposed to 32.2% of the long-short term memory network. Moreover, the vehicle age and mileage have a particularly strong association to the outcome of replacing fuel injectors, in addition to testing of various components related to fuel injectors (especially common rail injectors). Likewise, the generated fault codes tend to be associated with low/high fuel pressure.

Keywords: machine learning, neural networks, DTC, RNN, LSTM, random forest, decision trees.

Acknowledgements

I would like to thank my supervisor at Volvo who helped me and guided me through this work, as well as giving me the opportunity to do the thesis there. Additionally, I'd like to also thank my supervisor and examiner at Chalmers as they have been of help during the writing of my report.

Gabriella Galonja, Gothenburg, 2020

Contents

List of Figures

List of Tables

1

Introduction

This chapter presents a background to the importance of fault prediction, and enhances the importance of this work. Next, the aim, objectives as well as previous work relevant for this thesis, and the scope are presented. Lastly, the disposition of this thesis is outlined.

1.1 Background

As the automotive industry progresses towards innovative technological developments, the importance of a more sustainable future increases in order to keep up with the continuous progress. Hence, fault prediction is taking on an important role as it reduces unnecessary costs for both the consumers and the company itself. Additionally, it is providing a more sustainable future as faults are predicted correctly and in time. As hardware failures tend to originate from hardware degrading over time, where the degradation process can occur prematurely due to manufacturing defects [1], hardware faults are crucial when considering fault prediction as it reduces downtime for heavy-duty vehicles and thus reducing costs and inefficiency. Insights of the quality of the products as well as their reliability can be obtained by analysing warranty claim data as it includes repaired and replaced parts during the warranty period [2]. With this in mind, it becomes evident that it is critically important for the automotive industry to assess fault prediction in hardware in order to thrive even more.

A vehicle's performance is being tracked by on-board diagnostics. This eases data collection from the sensors of the network through diagnostic tools in the workshop, which can be used in order to determine what problem has occurred. On-board diagnostics were developed due to various reasons, e.g. emission control and electronic fuel injection, since fuel flow is being monitored through computer systems. The on-board diagnostics system consists of a central unit called electronic control unit (ECU), where the data from the sensors are collected which is used for either controlling the vehicle or monitoring various parts of the vehicle. The data collected provides information about the engine, chassis and electronic systems. This data contains information about the source and signal parameters, respectively, which is read by the ECU. If deviations are detected, the information is saved as a diagnostic trouble code (DTC). This causes a signal to be sent to the indicator light and

thereby a fault has been detected [3].

There is no one-to-one correspondence between a DTC and the caused fault. Instead, symptoms arises which could point to several different causes. The P0094 is an example of a DTC for powertrains, denoted by the first letter, where the four digits indicates a decrease in fuel pressure. Examples of ECUs are engine management system (EMS) and aftertreatment control module (ACM). As EMS is in control of the fuel flow that is being injected, the maximum engine power is affected by the performance of EMS. Thus, expenses could be lowered and sustainability is guaranteed if the wrongfully replaced parts caused by EMS would be detected [4]. Additionally, ACMs are critical to emission as they are working with exhaust aftertreatment systems [5].

As fuel injectors are expensive to replace, great investments are a necessity in order to give room for improvements in quality and performance. By improving the fault prediction in fuel injectors and associating it with the vehicle data, operational data and warranty claim data, knowledge could be gained for preventive measures. Thereby costs could be reduced while simultaneously investing in a better future for the environment. Hence, it is of foremost interest for Volvo Trucks to utilize this in the aftermarket aspect of the company for fuel injectors. By utilizing data concerning various performed operations in the workshop, insights could be gained about how the diagnostic tools are being used in the field. Data collected on performed operations involving replacement of fuel injectors in heavy-duty vehicles and operations that do not lead to further replacements give an indication of wrongful workshop actions, as well as flaws in the diagnostic tool. In turn, they can be improved by applying various machine learning techniques.

1.2 Aim

This thesis aims to develop algorithms with machine learning methods. The intent is to predict faults in fuel injectors, with the help of data from warranty claims, operations performed and vehicle information collected from diagnostic solutions which are retrieved from various databases. Two developed machine learning algorithms will be illustrated: random forest (RF) and long-short term memory (LSTM), respectively. The feature importance that comes with random RF, and its superiority in comparison to other decision trees models and due to LSTMs documented performance, make them suited for this type of work. Additionally, the goal is to determine what fault codes and types of operation contribute to the largest extent of wrongful replacements. Lastly, to evaluate what types of operations tend to be the most likely cause of ensuring correctly detected faults.

1.3 Objectives

– How are the diagnostic solutions being used in the instances when the operations do not lead to a replacement of a part?

- What DTCs are causing the largest amount of wrongfully replaced parts, and what are the potential causes?
- What faulty components tend to be detected wrongfully and is there any connection to what type of vehicles, e.g. the vehicle age, mileage, variant and usage type?
- What types of operations caused by what type of DTC are performed in order to prevent future replacements?

1.4 Previous work

Preventive fault detection has been used in e.g. IT and telecom environments, where preventive fault detection can be achieved by different means. These include anticipation of faults, detecting faults that impose an impossibility to predict, root cause analysis of unknown faults and fault prevention/recovery [6]. It has been utilized in the automotive industry with great success. In [7], an unsupervised DTC pattern learning framework has been suggested through the detection of new patterns in addition to applying known patterns that are being recognized accurately. This has yielded successful results by the usage of data gained from thousands of vehicles. Moreover, in [8], various machine learning techniques have been used including neural networks, e.g. convolutional neural network, in order to apply root cause analysis which was achieved by the use of time-series data analysis.

In [9], DTCs generated in the workshop have been used for anomaly detection. The paper aims to look at rarely observed fault codes generated for specific operation modes. The results of this paper points to successful results of detecting anomalies with the help of various machine learning techniques which shows promising results of speeding up future fault analyses and in turn, repairs. Although most failures occur when most vehicles are running, there are also some cases that point to faults occurring when the vehicle is cold. Additionally, there is more work on using DTCs and workshop repair data. One more such example is illustrated in [10], where the purpose is to predict vehicle maintenance. Both on-board and off-board databases were used as data for supervised machine learning algorithms, as well as repair history and in order to predict future failures in air compressors.

1.5 Scope

The fuel injector plays an important role when considering on-board diagnostics. As they are expensive to replace and it can be a tedious job to replace them, they impose a crucial role when improving diagnostic solutions. Therefore, only DTCs and operations conducted in the workshop that were involving fuel injectors are considered. Among all the DTCs available concerning fuel injectors, only EMS and ACM will be used. The reasons for this are the performance issues to the engine caused by such failures, as well as the emissions affected by the two, respectively.

Operations that lead to readouts containing general failure information were chosen, and this was due to two reasons. First, as such DTCs are generated due to a failure belonging to a unique category which cannot be expressed through a standard assignment or it can be through several subtypes within the same category. Instead, it consists of a new unique subtype from where no conclusions can yet be drawn. Second, it is of interest to uncover patterns from such types as the cause is unknown. Also, certain DTCs were excluded, despite them containing general failure information, as the cause is known to not be related to the fuel injectors.

Warranty claim data of heavy-duty vehicles from Volvo assembled between the years 2015 and 2017 are considered, since the data from these years are deemed to be the most stable and refined, regardless of delivery country. Also, the heavy-duty vehicle models that are included are Volvo FH and Volvo FM, which include more than 200 000 different trucks. As these are popular trucks from Volvo, they could provide some interesting and useful insights.

1.6 Disposition

This thesis is organized into six different chapters. The first chapter consists of the introduction which is divided into a background section, which lifts the importance of the thesis, as well as the aim and previous work related to the subject. The second chapter gives a brief background to machine learning, neural networks and algorithms relevant to the thesis. The third chapter contains the method, which describes the data cleaning as well as the data used and presents the whole process of the data extraction and creation of the algorithms. Chapter four and five present the results produced by the algorithms, and the discussion of the results, as well as further developments. Lastly, chapter six presents the conclusions that could be drawn from this work.

2

Machine Learning and Neural Networks

This chapter presents a background to machine learning and neural networks. Different categories and different algorithms relevant to this thesis will be explained, as well as how a neural network is built up. Lastly, methods of preprocessing the data will be briefly explained, as well as the ways of evaluating developed algorithms.

2.1 Supervised and unsupervised learning

Machine learning algorithms can be divided into two types: supervised and unsupervised learning. Supervised learning is done with already assigned classifications while unsupervised learning, on the other hand, is using unlabelled input data. Since unsupervised learning algorithms do not work as well as supervised learning algorithms, although they have benefits of their own [11], supervised learning algorithms will be focused on in this thesis.

Supervised machine learning algorithms are developed under the premise that it is being taught through examples, where the training data is already labelled into different categories. During the training process of the algorithm, the used input will be matched with its output while simultaneously looking for patterns connecting the input-output pair. When this is achieved, the next step for the supervised learning algorithm is to process new input data where the correct label will in turn be predicted. Supervised learning can be divided into classification and regression. When applying a classification algorithm, categories will be assigned to the data points as it is being trained. In turn, the algorithm will assign a class to each input value. In regression, one dependent variable is used together with a multitude of independent variables, from where a relationship is found in a predictive statistical manner [12].

Unsupervised learning, on the other hand, is about training unlabelled data in order to uncover unknown patterns and categories. As opposed to supervised learning, unsupervised learning is not limited by its training, and the system could potentially gain the same knowledge as a human being if the technology was to develop to a greater extent. Despite the fact it does not work as great as supervised learning, it has its benefits when considering reduction of the dimensionality of datasets, as

well as grouping similar objects and processing noisy data [13].

2.2 Neural networks

There are two main types of neural networks: feedforward networks and recurrent neural networks. The intention of a neural network is to mimic the knowledge of a human brain. It does so through neurons, where the neurons are interconnected and in turn build up the networks through a minimum of two layers as illustrated in Figure 2.1. The interaction between each layer is through weighted connections, and they are connected to neurons in other layers such as the input layer, output layer and the hidden layer [14]. The output of the nth neuron is given by:

$$y_n = \sigma\left(\sum_{i=1}^{m} w_{i,n} \cdot x_i + b_n\right), \tag{2.1}$$

where $w_{i,n}$ is the weight, x_i is the input of this neuron, b_n is the bias term, and σ is the activation function which can be linear or nonlinear [15]. The sum is the input to the activation function. If each input in the previous layer is connected to each one of the activation functions in the next layer, it is a fully connected layer. In order to determine the performance of a neural network, the loss function has to be computed, where a minimization of the loss is wanted in order to ensure the best performance. This is where the training process comes into the picture, as the purpose of training is to optimize the loss function, where the error value is used in backpropagation [16].

2.2.1 Deep feedforward networks

Deep feedforward networks are the simplest form of deep learning models that approximate a certain function f^*. By taking some classifier $y = f^*(\boldsymbol{x})$, the input $\boldsymbol{x}$ will be mapped to the category y and in turn a feedforward network will define the mapping $y = f(\boldsymbol{x}; \theta)$ where θ is the parameter that is supposed to be learnt by the network in order to obtain the best function approximation [17].

Deep feedforward networks can consist of several layers, as shown in Figure 2.1. The layers provide the depth of the model through the hidden layers placed between the input and output layer. The aim of the training part of the neural network is to force $f(\boldsymbol{x})$ to become equal to $f^*(\boldsymbol{x})$. The ultimate goal is for the learning algorithm to utilize the layers in order to obtain the best approximation of $f^*(\boldsymbol{x})$. The learning algorithm must decide how to use the applied layers to produce the desired output, but the training data does not tell each individual layer what they should do. Instead, the learning algorithm must decide how to use these layers to best implement an approximation of $f^*(\boldsymbol{x})$. As the training data does not show the desired output for each of these layers, they are called hidden layers [17].

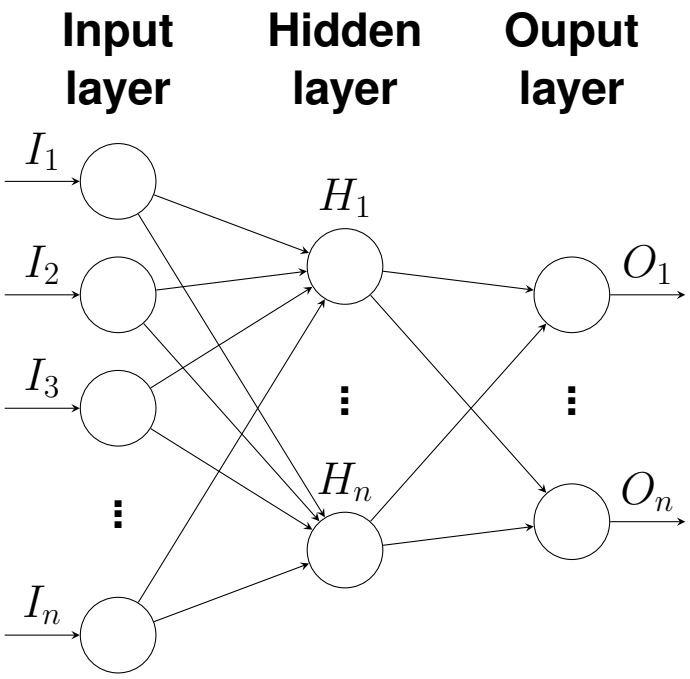

Figure 2.1: Visualization of a deep feedforward neural network, illustrating the various layers and how they are built up by n neurons.

2.2.2 Recurrent neural networks

When feedforward neural networks additionally consist of feedback connections, they are called recurrent neural networks (RNNs). This is what makes RNNs different from other neural networks: they are built up by a temporary (short) memory, a sort of feedback connection where propagation in opposite directions occur for the signals [18]. This makes them maintain all the previous data. An illustration of a RNN is shown in Figure 2.2. It is shown that the hidden state h_t, which contains all previous input information, is transferred from the previous time step to the next.

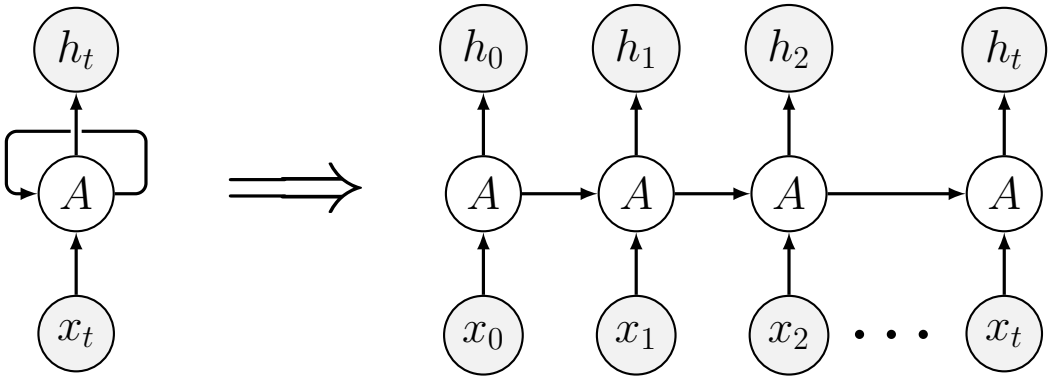

Figure 2.2: Illustration of an unrolled RNN, where x_t denotes the input and h_t denotes the output (hidden state) which contains all previous information.

A drawback with RNNs is their lack of long-term memory, which makes long temporal dependencies being forgotten. Also, this can cause a decrease in accuracy as

an optimal time lag has to be found. This has been proven to be a difficult task as the method is based on trial and error, and RNNs tend to also suffer from vanishing gradients due to this [19]. In order to solve this issue, different types of networks can be used instead, of which long-short term memory (LSTM) will be covered in this thesis.

2.2.3 Activation function

In order to add complexity to the neural network when considering its learning and performance measure, a non-linear transformation is being introduced into the output of the neural network through the use of an activation function. The input to an activation function consists of a weighted sum and an added bias, as shown in equation (2.1). The default choice of activation function tends to be the sigmoid function:

$$\sigma\left(v_i\right) = \frac{1}{1 + \exp(-v_i)}, \tag{2.2}$$

while the hyperbolic tangent function is an example of a function that converges faster than the former:

$$\sigma\left(v_i\right) = \frac{\exp(v_i) - \exp(v_i)}{\exp(v_i) + \exp(-v_i)}. \tag{2.3}$$

The most commonly used activation function is, due to its simplicity, the rectified linear unit (ReLU):

$$\sigma\left(v_i\right) = \max\left(0, v_i\right). \tag{2.4}$$

The values can be also be further rescaled to range between -1 and 1 through a softsign function [20]:

$$\sigma\left(v_i\right) = \frac{v_i}{(1 + |v_i|)}. \tag{2.5}$$

2.2.4 Regularization

Various regularization techniques can be used in order to prevent overfitting. When the algorithm performs well on the training data and not on the test data, which in turn can cause an increase of the training error, the algorithm could benefit from such a technique. When using features that do not have much impact on the target, it will result in an unnecessary increase of the variance which in turn will make the model perform worse than it should be. The performance of feature selection can be improved by using regularization, as the variance can be reduced.

2.2.4.1 Weight regularization

A norm penalty on parameters can be imposed with L^1 and L^2 regularization, respectively. By applying L^1, it will be of help when it comes to feature selection, as the coefficient of features that are deemed less important will be equal to zero. In L^2 regularization, large weights are being penalized in the cost function which results in a minimization of the variance which is the most prominent effect of L^2 regularization [21].

2.2.4.2 Batch normalization

The input layer can be normalized through batch normalization which is done by scaling the input as well as shifting it. Normalizing layer inputs can also be achieved by using mini-batches, which deals with each activation separately when estimating the mean and variance so they can be scaled and shifted independently of each other [22]. This is done since during the training process, the inputs do not attain the same distribution. By applying batch normalization, faster training can be achieved as the learning phase is speeding up as well as increasing stability, all thanks to a smooth optimization [23].

2.2.4.3 Dropout

Dropout can be used in order to prevent overfitting, as well as increasing the efficiency when combining a large amount of architectures. The technique is about temporarily removing both hidden and visible units together with their connections, in a probabilistic way during training. In turn, this leads to the hidden units being capable of creating features that are useful, as opposed to being dependent on other hidden units in order to fix their mistakes. During training, as it proceeds to the next training sample, a new part of the neurons is used in the dropout. During testing, the probability of the units' dropout will be multiplied with the weights [24].

2.2.5 Backpropagation

The purpose of backpropagation is to assess the weights and biases of a neural network in order to minimize the cost function, which is achieved through gradient descent. The initialization is done through random weights and biases, and as the training process proceeds, the weights and biases will be sequentially improved. However, this gives rise to the vanishing gradients problem for networks with a multitude of layers. This appears when useful gradient information is not able to transverse from the output back to the input layer of the model. In turn, this tends to lead to a too early convergence which worsens the performance [25].

2.3 Models

Two types of models will be illustrated: long-short term memory and random forest, respectively. The former is a neural network while the latter is a decision tree model.

2.3.1 Long short-term memory

A long short-term memory (LSTM) network is a type of RNN which has been altered for past data to be more easily remembered in its memory. In turn, this solves the issue of short-term memory in RNNs. LSTM recurrent networks consist of cells that have a self-loop present as opposed to having only outer ones which is characteristic for RNNs. Additionally, LSTMs have more parameters present when considering their inputs and outputs in the RNN as well as gating units where the system is in

control of the information's flow. In the LSTM there are three unit gates present: forget, input and output unit gate, respectively [17].

In order to determine what information could be discarded, the forget gate unit $f_i^{(t)}$, where i is the cell and t is the time step, consists of a sigmoid function where the weight can take on a value between 0 and 1:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right), \tag{2.6}$$

where $\boldsymbol{x}^{(t)}$ is the current input, $\boldsymbol{h}^{(t)}$ is the current hidden layer with all the outputs, and $\boldsymbol{b}^f$, $\boldsymbol{U}^f$ and $\boldsymbol{W}^f$ are the biases, input weights, and recurrent weights, respectively. That the parameters are from the forget gate is denoted by the superscript f. The state unit, $s_i^{(t)}$, is an important component when considering LSTMs and is dependent on the conditional self-loop $f_i^{(t)}$ as it is being updated:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right). \tag{2.7}$$

The input gate is given by:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right), \tag{2.8}$$

and consists of a sigmoid function as well which makes the possible values range from 0 to 1. The output $h_i^{(t)}$ is:

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}, \tag{2.9}$$

and can in turn be turned off through the output gate $q_i^{(t)}$:

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right), \tag{2.10}$$

which is also dependent on the sigmoid function [17].

The resulting LSTM system is shown in Figure 2.3. The external outputs, $c^{(t-1)}$ and $h^{(t-1)}$, are the memory and hidden states, respectively. The forget gate unit is shown to the far left where σ_1 is a fully connected layer, σ_2 is where the input gate is illustrated, whereas the candidate memory is shown after it, and the output gate is shown to the far right.

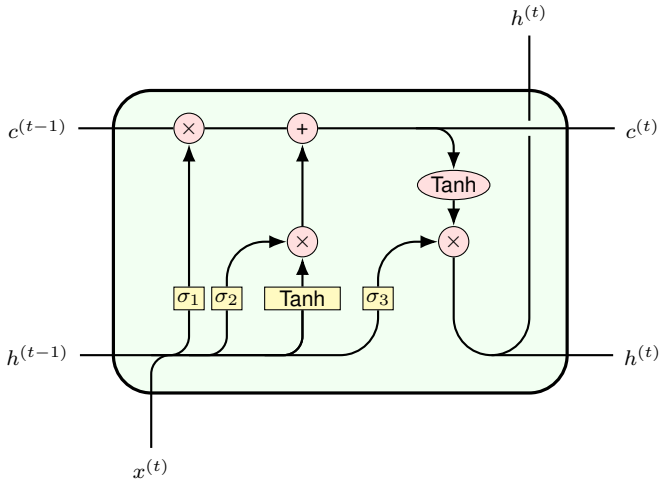

Figure 2.3: The resulting computations of the memory cells in an LSTM system.

2.3.2 Random forest

Random forest (RF) is an example of a supervised learning algorithm. In its essence, it is simply a collection of decision trees, that can be used for classification, as well as regression. An example of a decision tree is shown in Figure 2.4. As it is an ensemble learning method, the method results in trying a multitude of different decision trees with different splits, where the chosen model will be the one that is the most optimal which is shown in Figure 2.5. The introduction of RF is thanks to Leo Breiman [26], where RF today is essentially an extension of his bagging idea, where this "bagging" method is what RF is being trained with. The training data is used in bagging by building each decision tree by applying a replacement sample [27]. By combining and in turn applying a wide variety of learning models, the model will improve throughout this process. This is achieved by bootstrapping while simultaneously training the decision trains and lastly bagging. Bootstrapping is a statistical method where the point is to use small data samples of which the average estimate is taken in order to estimate the quantity, while bagging concerns reducing the complexity as there are certain models that lead to overfitting. Boosting, on the other hand, does the opposite since it helps with underfitting. The variance of RF classifiers is being minimized as the individual decisions trees remain unique throughout the classification. Lastly, when it is time for the last decision in the decision tree line, all results of the individual trees are gathered in order to determine what tree provides the highest average class probability. Due to this, RF tends to provide the best performance in comparison to other methods as well as preventing overfitting from being an issue [28].

Out-of-bag (OoB) error is a commonly used technique used in order to evaluate the RF accuracy. Also, it can be used for determining the values of the tuning parameters which are evaluated to be the most appropriate. The downside with this technique is that it can lead to an overestimation of the true prediction error when choosing the parameters [28]. RF algorithms can also be used in order

to obtain feature importance. There are two different types of feature importance methods: the gini variable importance, and permutation accuracy importance. The gini variable importance determines the probability that each feature is receiving an incorrect classification when chosen at random, while permutation feature importance randomly shuffles the values of single features. The permutation accuracy has been shown to provide more reliable results in comparison to the gini variable importance if there are a lot of features involved in the model, while the gini variable importance is also biased towards numerical features [29], [30].

2.4 Preprocessing

There exists a variety of different preprocessing techniques, whereas three will be presented in this section. The purpose of preprocessing is to e.g. prepare and transform raw data before being used in a machine learning algorithm. This is achieved by reducing the dimensionality of the data as well as normalizing it, searching for patterns and extracting features [31]. Feature extraction as well as feature selection are means of improving learning performance and lowering computational complexity [32].

2.4.1 Principal Component Analysis

One widely used technique for feature extraction and thus reducing the dimensionality of large datasets and at the same time preserving both statistical information and interpretability, is principal component analysis (PCA). New variables are found which are correlated with the original dataset by applying linear functions of them. Additionally, the new variables are uncorrelated between themselves while at the same time maximizing the variance. What PCA turns into is finding a solution to an eigenvalue/eigenvector problem [33]. However, when applying feature extractions such as PCA, it imposes problems with mapping the original features to the new features. Thus, it is more difficult to analyse the resulting features since they lack physical meaning [32].

2.4.2 Feature selection

Feature selection is used in order to minimize redundancy as well as maximizing the relevance to for instance class labels when classification is considered. The original features, as opposed to when applying feature extraction, maintain intact. It is done by selecting a subset of features without using any type of transformation in order to reduce overfitting. Additionally, it leads to an increase of accuracy as well as decreasing the training time [32].

2.4.3 Feature scaling

In order to scale numerical features, there are two popular methods: normalization and standardization. Normalization rescales the features to range from 0 to 1 or -1 to 1 for neural networks, while a mean of 0 and standard deviation of 1 is achieved for

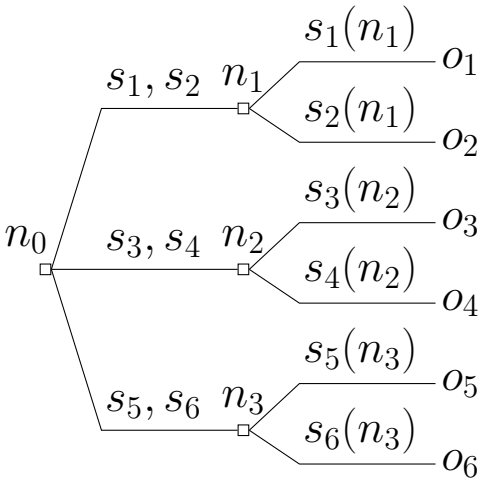

Figure 2.4: An example of a decision tree, with four nodes and six different outcomes.

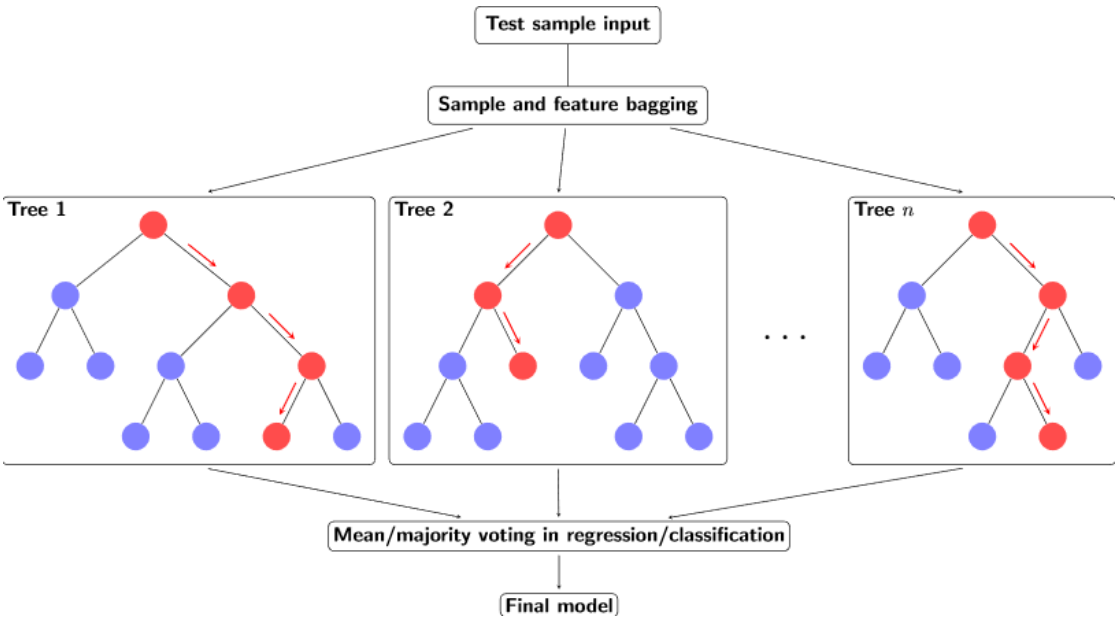

Figure 2.5: An illustration of an RF, where the sample and feature bagging result in different tree models, where the selected prediction will be the one with the majority of all votes for a classifier.

the features through standardization. Normalization is done through the following equation:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{2.11}$$

where $\boldsymbol{x}$ is the original features. Standardization is achieved through [34]:

$$x_{\text{std}} = \frac{x - \text{mean}(x)}{\text{std}(x)}. \tag{2.12}$$

2.5 Evaluation

In order to determine whether a high-performance model has been built, a confusion matrix can be computed. It consists of two rows and two columns, as shown in Table 2.1. The rows illustrate the predicted classes while each column represents the actual class. From this matrix, the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) can be obtained. This gives a better picture of the performance of a model, especially when considering imbalanced datasets [35]. The recall rate, or the sensitivity, is obtained through:

$$\text{recall rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.13}$$

which gives the percentage of how many of the positive class are classified correctly. Likewise, the precision is given by:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.14}$$

from where it can be determined how precise a model is when it comes to predicting the positive labels. The F1 score is dependent on recall rate and precision:

$$\text{F1 score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \tag{2.15}$$

since it is the harmonic mean of them. Balanced accuracy is an appropriate metric for imbalanced datasets as it takes into account the recall rate of both classes:

$$\text{balanced accuracy} = \frac{\text{TPR} + \text{TNR}}{2}, \tag{2.16}$$

where TPR is the true positive rate, and TNR is the true negative rate. By not taking the recall rate into account, the algorithm could give an accuracy of almost 100% if there is less than 1% of the positive class, since it will correctly predict all samples of the negative class.

Table 2.1: The outline of a confusion matrix.

		Actual value		
		p	**n**	**Total**
Prediction outcome	**p′**	True Positive	False Positive	P′
	n′	False Negative	True Negative	N′
	Total	P	N	

3

Methods

This chapter covers the methodology. The first part describes the data extraction from various databases as well as an illustration of the preprocessing steps. Next, the data selection, and how the datasets are built up, is described. Then, the developed algorithms are illustrated. Lastly, how the results are evaluated is depicted.

3.1 Data extraction and preprocessing

The data extraction involved extraction from databases storing vehicle data, workshop data and warranty claim data.

3.1.1 Vehicle data database

The vehicle database provides information about functionality in addition to transactional data and diagnostic session logs of products that have been generated by the attached workshop tool. It consists of services from where product session information (PSI) is received and provided to consumers. Thus, the logged data that is generated from ECUs and diagnostic session logs which are obtained from workshop tools can be extracted from the database containing the vehicle data. Through this, knowledge is gained regarding the usage of products as well as performed workshop actions in both the maintenance and repair process.

3.1.2 Workshop data

The workshop data was extracted through SQL queries from the databases which contained information from diagnostic sessions. The data was stored in DB2, which is a relational database management system developed by IBM [36]. This database contains vehicle information as well as information about the workshop sessions.

The vehicle data consists of one row per vehicle, that contains basic vehicle information about each vehicle, e.g. chassis number, chassis series, country and assembly date. The chassis number is what denotes each vehicle's uniqueness, from where e.g. the vehicle ID can be retrieved which can be used in order to obtain information from other tables in the database.

Product session information (PSI) contains information about diagnostic sessions

that are retrieved from the workshop. This information consists of what type of operation was performed in the workshop, including e.g. fault tracing, calibration and testing which are all relevant for this thesis. Faulty parts of a vehicle will only be replaced after fault tracing has been performed. Therefore, operations performed on unaffected vehicles which do not have any warranty claim give an indication of what actions tend to prevent unnecessary replacements. Moreover, each operation where testing or calibration was performed contains an operation identifier. This identifier is unique for what type of operation has been performed, and example of such identifiers are system testing of rail fuels and calibration of the injectors.

Each performed operation, which has been stored in the database, consists of a diagnostic path which indicates the path to the tree in the authoring environment, where only the variable path is visible from the data extraction. Typically, the last part of it contains the DTC readout when the performed operation is fault tracing. However, this is not always the case for older readouts and DTCs generated for certain sensors. When an older fault tracing is being done but not for that specific DTC, it can be linked to an old file. This tends to be the case when the hardware has barely been changed over the years.

3.1.3 Warranty claim data

A warranty claim is made as soon as a vehicle has been repaired or if it happens to have a part that is defective which was discovered during the workshop visit. The warranty claims data is collected from live data from the workshops containing logged session information from where fault tracings and operations can be retrieved. Fault tracing can be retrieved as this live data provides access to all fault codes that have been fault traced onto each claim, and in turn it becomes evident whether the fault tracing has been successful or not. This gives a large amount of DTCs which have been tested for in the diagnostic tracing and an indication of the performance of the diagnostic sessions.

Both chassis with repeated claims and with only one claim were considered faulty. If there are several claims for a particular vehicle, it gives an indication of faulty replacements since it has to return to the workshop. Likewise, it gives an indication of a defective component. Altogether, the warranty claim data consisted of 18000 claims from all around the world, although the workshop sessions consisted of many more operations as there were multiple DTC readouts during each session. As fuel injectors were to be considered, claims related to solely fuel supply related components were of importance. The fault date together with chassis number were used in order to link the claim data with the workshop data. Despite the fact that part number and name were included in the claim, they were not used as there was insufficient data containing this information in the workshop data. Furthermore, the emission level provides information about what vehicle variant the claims were connected to.

3.1.4 Preprocessing

Some of the columns in the dataset retrieved from the databases through SQL queries contained missing values. This occurred in values related to e.g. vehicle information such as mileage, as mileage is only present in diagnostic session logs when a part has been replaced. Numerical values as input data tend to be a requirement for most machine learning algorithms since neural networks and RF algorithms cannot handle missing data. Due to this, the numerical values had to be preprocessed while the duplicate entries were removed as there were some duplicates of the same operation which did not provide any new information. The missing values can be handled by applying an imputation technique which statistically will calculate a value that is most likely and in turn replace the value with this statistical value in each column. This is a common way of approaching this problem, due to the simplicity that comes with using the training dataset. Additionally, it tends to provide a good performance where the performance of such a technique is independent of the amount of missing values as well as the dataset [37]. The preferred way could be, to simply remove all the rows containing missing data. As there is close to a million readouts, this could be done and a large amount of data would still be present in the dataset. Both methods were tried, the imputation did however seem to give more weight to the numerical values, while less weight was put on other features of interest. Therefore, the final results did not involve imputation as there were only two numerical features.

Normalizing numerical values tends to give better results for neural networks as they are very sensitive to scaling, while this is usually not the case for RF algorithms as it tends to not change the performance of the algorithm. Vehicle information such as mileage and vehicle age which consisted of numerical values were normalized before being used as input to both the RF and LSTM algorithms. Since there were both categorical variables and continous variables, some type of encoding had to be done before feeding them as inputs to the models. If there would have only been categorical features in the dataset, the choice of using an embedded layer instead of dense layers would have been the better choice. The reason for this is that it is speeding up the training process due to the fact that it reduces the complexity. While using a method such as one-hot encoding would lead to more features and in turn a very sparse matrix, and thus slowing down the training speed as it involves more matrix multiplications which increases the computational complexity [38].

3.2 Data selection

Two datasets, one containing vehicles with warranty claims and another one with vehicles without warranty claims, were used when extracting data from the databases. In the end, two datasets were finalized: one with all types of fuel injectors and one with common rail fuel injectors.

3.2.1 Datasets

There were two datasets: one containing all the vehicles assembled between the years 2015 and 2017, and another dataset which consisted of all the warranty claims made on the vehicles in the first dataset. Since not all the vehicles assembled between those years have a warranty claim, those vehicles are unaffected by repairs, and should give an indication of what operations were performed that (possibly) prevented the vehicle from repeatedly going back to the workshop and in turn getting claims attached to them. Since workshop repairs are made within a 6 week interval before the claim is created, operations conducted within this time interval were considered and extracted from various databases for the created warranty claims. They were found by matching the chassis ID from the vehicle in the warranty claim data within that period of time, where also the fault codes were denoted. Although the warranty claim data also had all the fault codes generated for each diagnostic session, what types of operations that were performed are not visible. Moreover, Volvo has 2 years fitted parts warranty. As the heavy-duty vehicles used in the analysis were produced between the years 2015 and 2017, readouts that took place in 2019 the latest were therefore extracted. Also, the operations performed in the workshop for the unaffected vehicles include only fault codes which are overrepresented in the warranty claim data.

Each row in the dataset consists of one unique DTC, unique for a particular operation. This means one workshop session can contain several relevant DTCs which were collected during the fault tracing by the diagnostic tool, all of which were related to powertrain. Additionally, each row contained what types of operation were performed, operation identity, as well as the mileage, vehicle age, emission level in order to distinguish what type of vehicle each particular readout is related to, etc. Moreover, as the intended problem was to classify whether a vehicle was faulty which the vehicles with warranty claims are considered, each operation where a warranty claim was created afterwards was considered faulty while vehicles without a claim were considered not faulty. Hence, there was a binary classification problem at hand, where the faulty vehicles belonged to the positive class and the unaffected vehicles without claims belonged to the negative class.

As it was interesting to see what conclusions could be drawn from all parts of fuel injectors and of common rail fuel injection, two separate datasets were finalized. One of them consisted of a positive class of warranty claims of all parts of fuel injectors, while the second dataset consisted of only common rail fuel injection for the positive class. Since it is also of interest to uncover patterns between the failure of fuel injectors through the operations performed based on the generated DTCs, the different operations as well as the different types of generated DTCs were used as features as inputs to the RF algorithm and LSTM network. For RF and the LSTM network, some type of encoding had to be done for categorical variables in Python. One-hot encoding was chosen for this thesis, not only due to its simplicity, but also because it gives us a clear picture of what effects the operational data and generated fault codes have on the outcome.

3.2.2 Class imbalance

There was an existent class imbalance in the dataset, as the majority class which comprised of vehicles that had undergone operations which did not lead to a wrongful replacement of a component and did not have a warranty claim, was much larger in comparison to the affected vehicles. This causes an issue when looking at the training aspect of the classification system, as the prediction accuracy will be in favor for the majority class [39]. One way to deal with class imbalance is by imposing a class weight, which gives more weight to the minority class. Also, undersampling the majority class while simultaneously oversampling the minority class is another way. When oversampling, samples are added to the minority class by duplicating them, while it is the opposite in the case when undersampling is done. Instead, samples are removed from the majority class [40]. Additionally, a combination of both ways is also possible. In this thesis, several techniques were tested since imposing class weights was in some cases not sufficient, due to the fact that certain datasets consisted of less than 1% of the positive class. Therefore, undersampling was used.

3.3 Machine learning Frameworks

TensorFlow is an end-to-end open-source software which covers all aspects of machine learning: everything from development to training. On top of TensorFlow, Keras can be run, which is used for deep learning as it is a high-level API used for neural networks, as opposed to TensorFlow which is a lower level API. Keras cannot be run without a backend, however it does not necessarily have to be with TensorFlow as a backend framework as there are other frameworks that can be used as well. Scikit-learn is another machine learning library built on top of Python packages such as NumPy. It tends to be used for general machine learning problems whereas TensorFlow is usually used for deep learning and also provides more efficient training in comparison. However, due to its simplicity, it could be the preferred library, on the other hand it cannot be applied on deep learning methods. However, the flexibility and control of the networks that comes with TensorFlow is what makes TensowFlow the preferred choice in some cases.

3.4 Algorithms

Two algorithms were implemented: one through neural networks and one through decision trees.

3.4.1 Long-short term memory

LSTM has the benefit of containing far more hyperparameters that can be tuned in comparison to decision tree models, and it also has the possibility of adding more complexity to it. Two simple LSTM layers tend to be sufficient in order to detect complex features, whereas one layer is generally enough when considering datasets that do not contain any further complexity. For the architecture of this

problem, three LSTM layers were stacked on top of each other, where the first LSTM contained a ReLU activation function. Each LSTM layer contained 128-512 hidden states, and were wrapped up by dropout layers. This was followed by a multitude of deep layers stacked up on each other which were not connected horizontally. In turn, a dropout layer was applied between each fully connected layer in order to prevent overfitting, each with probability 0.2. However, applying too many hidden layers will lead to overfitting, since it results in unnecessary parameters. Additionally, the batch normalization layers were applied between linear and non-linear layers as it normalizes the input to the activation functions. The last layer consisted of a sigmoid activation function as the classification is considering binary classification, while a combination of ReLU and sigmoid activation functions were used for the other layers.

3.4.2 Random forest

The choice of an RF algorithm was due to the feature importance that comes with it, and due to its high performance. The RF algorithm was implemented through the Scikit-Learn library in Python. RF contains far less hyperparameters in comparison to LSTM networks. This includes choosing the number of decision trees which regulates its depth and tuning the number of features, as well as imposing class weights into the classifier. The number of decision trees did not have much effect when comparing a couple of hundred to a couple of thousand, at the cost of resulting in an increase of computing time. Thus, 400 decision trees were chosen. At each leaf node, the maximum number of features considered at each time the splitting was taking place could be chosen [41]. When considering each leaf node, a random subset of all features which can be tuned are considered.

3.5 Evaluation of results

In order to figure out how each individual feature affects the output of the model, the feature importance and the permutation feature importance were computed. These are means to determine what is going on in the machine learning model and what effect each feature has on it. Additionally, the balanced accuracy for the RF algorithm as well as the precision, recall rate and F1 score were taken into account, as the problem at hand is a class imbalance problem.

4

Results

In this chapter, the results from the developed algorithms are presented. This is done by first illustrating the results from the complete model setup with the performed operations in the workshop and the generated fault codes obtained from the fault tracing, from where the most important DTCs are obtained. The second part consists of results from the operations that were shown to be the most important from the complete model setup.

4.1 Results of fault code and operational data

Here, the results of the performance of the algorithms are presented, both for the all types of fuel injectors and common rail ones. Additionally, feature importance illustrating the faultcodes and operations with the most impact are shown.

4.1.1 Replacement of all types of parts in fuel injectors

In order to determine the performance of the models, balanced accuracy, precision and recall rate were computed, as well as the F1 score. As the accuracy does not provide sufficient information about the accuracy for highly imbalanced datasets, the accuracy for RF is in fact the balanced accuracy. While for the LSTM it is not the balanced accuracy that is being taken into account, thus it is heavily influenced by the class imbalance. The results from the dataset containing performed operations of all warranty claim data involving all parts of fuel injectors are labelled as faulty, whereas vehicles assembled from the same years that do not have a warranty claim are labelled as not faulty, are shown in Table 4.1.

Table 4.1: The performance of the RF and LSTM algorithms with operational and fault code data for all warranty claims involving all parts of fuel injectors. The accuracy for the LSTM network is not the balanced accuracy.

	Random Forest	Long-short term memory
(Balanced) accuracy	82.8%	(95.6 %)
Precision	86.3%	76.2%
Recall rate	66.1%	32.2%
F1 score	74.9%	45.4%

In Table 4.1, it is evident that RF most likely has the highest balanced accuracy as it has the highest precision and recall rate. As the dataset is highly imbalanced, the recall rate gives a better estimate of whether a faulty fuel injector can be estimated.

4.1.2 Replacement of common rail fuel injectors

Table 4.2 illustrates the performance of the RF and LSTM algorithms based on all operations with a warranty claim data only containing replacements of common rail fuel injectors.

Table 4.2: The performance of the RF and LSTM algorithms on both operational and fault code data for replacement of common rails of the fuel injectors. The accuracy for the LSTM network is not the balanced accuracy.

	Random Forest	Long-short term memory
(Balanced) accuracy	70.8%	(98.6%)
Precision	75.3%	74.1%
Recall rate	42.0%	41.6%
F1 score	53.9%	53.3%

Here, the performance is similar when considering the RF and LSTM algorithms, both when it comes to determining whether a not faulty fuel injector is not faulty, and whether a faulty fuel injector is in fact faulty.

4.1.3 Feature importance

The feature importance could be obtained from the RF algorithm, which was done for both the dataset with all features and the dataset which concerned parts related to common rail fuel injectors. As both operational data as well as the generated fault codes were considered, all of them are shown in the results. The most common DTCs were used later in order to deep dive into what operations are being performed that do not lead to replacement of fuel injectors. The results are illustrated in the table below.

Table 4.3: Feature visualization obtained from RF for all types of fuel injectors and common rail ones.

Feature visualization all types	Feature visualization on common rail fuel injector
Mileage	Vehicle age
Vehicle age	Mileage
DTC A	DTC A
Test common rail fuel injector	DTC B
DTC B	Test common rail fuel injector
Euro 6	Euro 6
Rail pressure system	DTC C
Emissionlevel Euro 6	Rail pressure system
DTC C	DTC D
DTC D	DTC G
DTC E	DTC F

In Table 4.3, it is evident that mileage and vehicle age have a great impact on the replacement of a part. Moreover, DTC A is a fault code that is generated when the fuel pressure regulator is too high, while DTC B is indicating that the fuel pressure has essentially decreased a severe amount. Operation identity A denotes testing of common rail fuel injectors. DTC C, on the other hand, simply indicates that the coolant level is low. The fault code D is triggered when the target angle and phase-control have a gap between the two, which in turn will cause the valve to stop working. DTC E occurs when fuel system with a low pressure is reaching a pressure that is way too low. DTC F is low fuel pressure while DTC G indicated that the oil level is fairly low. Lastly, the emission levels indicate what type of vehicle it is that these faults tend to be present in, where Euro 6 is for light-duty vehicle produced from 2015.

4.2 Results of the operational analysis

The operations performed on the most important fault codes were obtained by using only the most important fault codes. Therefore, insights could be gained about what types of operations have a big impact on whether a fuel injector will be wrongfully replaced.

4.2.1 Replacement of all types of parts in fuel injectors

Here, the accuracy, precision, recall rate and F1 score, respectively, were computed for the operations based on the most important fault codes. The results are shown in Table 4.4.

Table 4.4: The performance of the RF and LSTM algorithms on warranty claim data for both operational and certain fault code data for replacement of common rail fuel injectors.

	Random Forest	Long-short term memory
(Balanced) accuracy	75.1%	(94.4)%
Precision	73.1%	72.4%
Recall rate	51.7%	35.6%
F1 Score	60.6%	47.7%

In Table 4.4, it is evident that the performance has decreased in comparison to when considering only certain fault codes.

4.2.2 Replacement of common rail fuel injectors

Below is the performance of DTCs A-E on replaced parts of the fuel injector.

Table 4.5: The performance of the RF and LSTM algorithms on both operational and fault code data for replacement of of common rail fuel injectors.

	Random Forest	Long-short term memory
(Balanced) accuracy	69.6%	(97.3%)
Precision	84.2%	57.2%
Recall rate	39.1%	67.8%
F1 score	53.4%	62.1%

Interesting enough, the LSTM seems to be able to handle this problem better than the RF algorithm.

4.2.3 Feature importance

The DTCs A-E depicted in the previous section were used in order to see whether there is a strong association to the operations performed. The results are shown in the table below.

Table 4.6: Feature visualization of the operations performed on DTCs A-E, obtained from Random Forest for all types of fuel injectors and common rail ones.

Feature visualization all types	Feature visualization on common rail injectors
Testing of common rail fuel systems	Testing of common rail fuel systems
Rail pressure system	Rail pressure system
Testing of fuel consumption	Engine start
Fuel injector shut off	Testing of fuel pressure
Low fuel pressure	Testing of fuel consumption
Cylinder compression test	Fuel consumption data
High pressure sensor	Exhaust aftertreatment
Injector function	Testing of fuel pressure
Testing of fuel pressure	Testing of cylinder balancing

In Table 4.6, it is evident that tests have usually been performed related to the fuel injector, in order to observe its performance when it comes to fuel pressure. Also, it appears to have been done in order to see whether it works as intended. Mo

5

Discussion

The purpose of this work is to investigate whether warranty claim data together with operational data and fault codes provide sufficient information in order to predict whether a fuel injector has failed using machine learning methods. This has been achieved by the development of an RF and LSTM algorithm, which will be evaluated in this section, followed by the ethical aspects, the contribution to the field and suggestions for future work.

5.1 Model performance

The performance of the two algorithms, for all types of fuel injectors and for common rail fuel injectors, was measured. As the balanced accuracy was only obtained from the RF algorithm, recall rate and precision are in focus instead.

5.1.1 All types of fuel injectors

The results are pointing to a relatively high balanced accuracy of 82.8% for the RF algorithm, and the precision and recall rate are relatively high as well, 86.3% and 66.1%, respectively. The LSTM network did not outperform the RF algorithm, as the recall rate was much lower with 32.2%, while the precision was slightly lower in comparison but still acceptable. Possibly the RF algorithm was better at handling the class imbalance while the LSTM was more prone to it, as the recall and thus its ability to predict faulty fuel injectors was higher. Perhaps the architecture was not well-suited for this type of data, i.e. perhaps embedding layers, one handling the numerical variables, and one handling the categorical ones instead of using one-hot encoding, would have improved the model. Another reason could be that there were too many irrelevant features whose effect was not great enough, as it is enough that there is one operation performed on such a feature for it to be considered. Otherwise, one could expect the LSTM to outperform the RF algorithm as it has more potential in regards to the hyperparameters. When choosing only 5 DTCs with the intention to see what performed operations were being crucial, it was evident that the performance decreased in comparison, when it came to the balanced accuracy, precision and recall rate. Most likely this gives an indication of that more parameters have to be taken into account, alternatively handling the numerical and categorical parameters in a different way. Additionally, DTCs related to low/high fuel pressure

tend to be recurrent.

5.1.2 Common rail fuel injectors

The results for the replacement of common rail fuel injectors points to a large portion of the faulty injectors being misclassified since the recall rate was about 42% for both algorithms, while it was 68% for LSTM and 39% for RF when considering the operations performed. The decline in recall rate and thus balanced accuracy when considering only certain types of fuel injectors could be due to the heavy class imbalance. The negative class, which consists of the unaffected vehicles which do not have any warranty claims attached to them, was not filtered more than only containing the same fault codes that were present in the affected dataset. As the part name often is left out in the database from where the operation sessions were retrieved since they are not always replaced, they could not be taken into account and in turn all operations considering all parts of the fuel injector were included. If this would not have been the case, that the entire dataset could have contained only these types of parts, other conclusions could have been drawn as more relevant data is considered. Nevertheless, various methods such as over- and undersampling and class weights as well were used in order to deal with the class imbalance for this case, although this could potentially leave out some useful information. The LSTM algorithms higher recall rate when considering the operations performed, could additionally be due to a better class weight tuning.

5.2 Performed operations with respect to vehicle information and generated fault codes

The results point to a strong connection between the failure of fuel injectors and the vehicle information. The replacement of a fuel injector appears to be heavily influenced by the mileage and the vehicle age, which was expected due to the hardware degrading over time as it is being heavily used indicated by the mileage. However, it could also be due to numerical values being biased when considering feature importance. The results from the operations performed in the workshop give an indication that for the most part they have involved testing of the fuel injectors. As the negative class consists of the major part of the extracted operations that were conducted in the workshop, it indicates that performing testing of fuel injectors have lead to the fuel injector not being replaced. This could indicate that fault tracing is not being used as frequently as it should be, thus resulting in faulty fuel injectors as it comes into the workshop a multitude of times when it is not being replaced when it should be. Alternatively, the diagnostic solutions do not work to its full potential, as fuel injector are being replaced even though the same type of fault codes appear to have been generated even when there is only testing of the fuel injector.

5.3 Ethical aspects

As the data extraction involved vehicles used for private use as well, one aspect to consider is invasion of privacy. On the other hand, as the customers have actively chosen to seek help from a workshop which have plugged in the diagnostic solutions, the customer should be well aware of that this imposes a risk for invasion of privacy. Nevertheless, the data extracted for this thesis does not tie the vehicle in question to anyone on an individual level and as the vehicles have not been considered individually during the data extraction this has in turn been respected. Also, as the results of this work points to insufficient testing in the workshop, further research on this matter could in the end result in more work in the workshop. However, this could lead to less work for factory workers as the production of fuel injectors would decrease as a result of improved fault tracing, hence resulting in people losing their jobs.

5.4 Contribution

There has been related work concerning on-board and off-board diagnostics, where fault codes and operational data together with repair history have been used to predict future failure in order to improve maintenance with various machine learning techniques which have been achieved through both supervised and unsupervised methods. It does, however, not appear to have been done before with deep learning, and not with warranty claim data. The difference here is that the warranty claim data has been used as an indication of an existent fault in the component. The reason for this has been assumed to be either due to manufacturing faults or due to the wrong workshop actions as a result of either misinterpreting the action that should have been performed, alternatively the diagnostic solution is not working as fully intended. Instead, these types of data have been used either for anomaly detection in order to capture fault codes occurring in rare instances, or to predict upcoming failures based on all repair history in air compressors. It does not seem to take into account that the failure could be pre-determined by its production. Neither does it appear to have been done with fuel injectors in such a way.

5.5 Future work

Although there was lots of data included in this work, only fault codes in which very little about the cause of them is known, as the source cannot be pinpointed, it would be interesting to analyse the anomalies. As very little weight was given to them in this work, insights could be gained which could perhaps have an effect on the more frequently appearing faults. There are more factors based on the warranty claim data that could be taken into consideration but were not, in order to get a better understanding of why faulty components are being replaced. As the effects could be due to human error as well, and not only due to how the diagnostic solutions are working in the field. Lastly, by adding more data when considering the positive class for the LSTM network, an improvement in performance could be possible as

neural networks are best suited for large datasets.

Other ways of improving the results is by improving the quality of the data, which could be achieved by collecting the data differently. As the sensors and thus signals consist of a limited amount, implementation of more sensors could create more well-defined data, and in turn would make fault tracing more accurate. Also, by collecting data more frequently with e.g. a smartphone, fault prediction would be more easily accessible. In turn, issues that lead to unnecessary replacements could be prevented if faults prior to them are detected in time.

6

Conclusion

In conclusion, this thesis has explored factors that could help preventing wrongful replacements of fuel injectors by utilizing data collected from diagnostic solutions, together with vehicle information and warranty claim data, that in turn have been used as inputs to various machine learning techniques. This has been achieved through an RF algorithm and LSTM network, where the performance of the RF algorithm has shown better results with a balanced accuracy of 82.8%, with a better ability to correctly predict faulty fuel injectors as it is 66.1% in comparison to 32.2% of the LSTM network. The high balanced accuracy implies that there is a strong connection between the features and the failure of a fuel injector, and that warranty claim data therefore can be used in order to predict the failure of the component or whether it will be replaced a multitude of times. As the vehicle age as well as the mileage carry a strong connection to the replacement, it could indicate that it is not due to manufacturing faults that these replacements occur, instead it is due to either the software in the diagnostic solutions or the customer's usage of the trucks. Alternatively, it is just due to numerical features being biased. Moreover, testing of fuel injectors, especially of common rails fuel systems, seems to have a connection to fuel injectors not being replaced as they are over represented for operations performed on vehicles that do not have any warranty claims. Recurrent generated fault codes tend to be associated to low/high pressure in the fuel injector, also light-duty vehicles produced from 2015 and onward tend to be particularly affected.

Bibliography

[1] E. Sammer, *Hadoop Operations: A Guide for Developers and Administrators*. O'Reilly Media, 2012.

[2] P. H. A.Dai, Z. Zhang, "Warranty claims forecasting for new products sold with a two-dimensional warranty," vol. 28, 2019, pp. 715–730. [Online]. Available: https://link.springer.com/article/10.1007/s11518-019-5434-8

[3] Noregon, "What Does OBD Stand For?" 2020. [Online]. Available: https://www.noregon.com/what-is-obd/

[4] F. Wuhib and Y. Lemieux, "Automating fault detection for management systems using ML," 2019. [Online]. Available: https://www.sciencedirect.com/topics/engineering/engine-management-system

[5] V. Trucks, "Service Information," 2013. [Online]. Available: https://class8truckparts.com/media/file/volvo_truck_ecm_aftertreatment_control_module_dtc.pdf

[6] F. Wuhib and Y. Lemieux, "Automating fault detection for management systems using ML," 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/3/automating-fault-detection-for-management-systems-using-ml

[7] S. Kriebe, E. Kusmenko, B. Rumpe, and I. Shumeiko, "Learning error patterns from diagnosis trouble codes," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 179–184.

[8] Y. Sun, Z. Xu, and T. Zhang, "On-board predictive maintenance with machine learning," in *SAE Technical Paper 2019-01-1048*, 2019, p. 10.

[9] A. Theissler, "Multi-class novelty detection in diagnostic trouble codes from repair shops," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 2017, pp. 1043–1049.

[10] R. Prytz, "Machine learning methods for vehicle predictive maintenance using off-board and on-board data," Ph.D. dissertation, 09 2014.

[11] S. Goldman and Y. Zhou, "Enhancing supervised learning with unlabeled data," in *ICML*. Citeseer, 2000, pp. 327–334.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[13] M. Heller, "Unsupervised learning explained," 2019. [Online]. Available: https://www.infoworld.com/article/3429017/unsupervised-learning-explained.html

[14] A. Goh, "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143 – 151, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/095418109400011S

[15] M. Zakaria, M. Al-Shebany, and S. Sarhan, "Artificial neural network: a brief overview," *Int J Eng Res Appl*, vol. 4, pp. 7–12, 2014.

[16] F. Günther and S. Fritsch, "neuralnet: Training of neural networks," *The R journal*, vol. 2, no. 1, pp. 30–38, 2010.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] D. Pham and X. Liu, "Identification of linear and nonlinear dynamic systems using recurrent neural networks," *Artificial Intelligence in Engineering*, vol. 8, no. 1, pp. 67 – 75, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/095418109390032B

[19] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187 – 197, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X15000935

[20] P. Ramachandran, B. Zoph, and Q. Le, "Searching for activation functions," 2018. [Online]. Available: https://arxiv.org/pdf/1710.05941.pdf

[21] S. Lau, J. Gonzalez, and D. Nolan, in *Principles and Techniques of Data Science*, 2020.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[23] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" 2018.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[25] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[26] A. Cutler, D. R. Cutler, and J. R. Stevens, *Random Forests*. Boston, MA: Springer US, 2012, pp. 157–175. [Online]. Available: https://doi.org/10.1007/978-1-4419-9326-7_5

[27] K. Fawagreh, M. M. Gaber, and E. Elyan, "Random forests: from early developments to recent advancements," *Systems Science & Control Engineering*, vol. 2, no. 1, pp. 602–609, 2014. [Online]. Available: https://doi.org/10.1080/21642583.2014.956265

[28] S. Misra and H. Li, "Chapter 9 - noninvasive fracture characterization based on the classification of sonic wave travel times," in *Machine Learning for Subsurface Characterization*, S. Misra, H. Li, and J. He, Eds. Gulf Professional Publishing, 2020, pp. 243 – 287. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128177365000090

[29] C. Strobl and A. Zeileis, "Danger: High power! ? exploring the statistical properties of a test for random forest variable importance," 2008. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-2111-8

[30] scikit learn, "4.2.2. Relation to impurity-based importance in trees," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/permutation_importance.html#id2

[31] S. Alasadi, "Review of data preprocessing techniques in data mining," *Journal of Engineering and Applied Sciences*, vol. 12, pp. 4102–4107, 09 2017.

[32] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applications*, pp. 37–64, 01 2014.

[33] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0202

[34] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern recognition letters*, vol. 22, no. 5, pp. 563–582, 2001.

[35] D. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation," *Mach. Learn. Technol.*, vol. 2, 01 2008.

[36] "DB2 - Introduction," 2020. [Online]. Available: https://www.tutorialspoint.com/db2/db2_introduction.htm

[37] A. Jadhav, D. Pramod, and K. Ramanathan, "Comparison of performance of data imputation methods for numeric dataset," *Applied Artificial Intelligence*, pp. 1–21, 07 2019.

[38] P. Rodríguez, M. A. Bautista, J. Gonzàlez, and S. Escalera, "Beyond one-hot encoding: Lower dimensional target embedding," *Image and Vision Computing*, vol. 75, pp. 21 – 31, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885618300623

[39] M. Molinara, M. T. Ricamato, and F. Tortorella, "Facing imbalanced classes through aggregation of classifiers," in *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, 2007, pp. 43–48.

[40] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, p. 20–29, Jun. 2004. [Online]. Available: https://doi.org/10.1145/1007730.1007735

[41] scikit learn, "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html