



**FAULT DETECTION IN DIESEL ENGINES USING DEEP LEARNING AND
QARMA ALGORITHMS**

Lappeenranta–Lahti University of Technology LUT

Master's Programme in Mechanical Engineering, Master's thesis

2022

Oleg Rogov

Examiners: Post-doctoral Researcher, Grzegorz Orzechowski

Associate Professor, Pedro Juliano Nardelli

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Energy Systems

Mechanical Engineering

Oleg Rogov

Fault Detection in Diesel Engines using Deep Learning and QARMA algorithms

Master's thesis

2022

46 pages, 10 figures, 11 tables and 1 appendix

Examiners: Post-doctoral Researcher Grzegorz Orzechowski and Associate Professor Pedro Juliano Nardelli

Keywords: neural networks, qarma, quantitative, association, rule, mining, algorithm, diesel engines, classification, faults detection, comparison

QARMA, known as quantitative associated rule mining algorithm, is one of the recently developed algorithms for classification, that needs more testing on the real data. Neural Networks, that nowadays only tend to improve, are very common in solving different classification problems.

This work compares the state-of-the-art algorithm of QARMA with the established approach of Neural Networks by training them on the open-source dataset of the diesel engine faults, which contains 4 types of different engine conditions. Deep Neural Network was created by me using Python language with its different data processing libraries, while QARM algorithm was ran in a special graphical user interface under the control of the copyright holder. The evaluation of this algorithms is performed with the usage of accuracy metrics and confusion matrix. One of the goals of the work was also to improve the accuracy for diesel faults detection in comparing to the existing results, that were described in one of the articles – 5% improvement was reached with the usage of QARMA and Deep Neural Network. Few recommendations on how to improve the research algorithms are proposed.

ACKNOWLEDGEMENTS

I would like to say big thanks to all of the people who surrounded and helped me to get through my experience of studying abroad.

Special thanks to:

Grzegorz Orzechowski for fast and furious acceptance me as a thesis writer

Pedro Juliano Nardelli for the guidance through all this work and being pro-active supervisor

Ioannis Christou for opening the world of QARMA for me and contributing to my research

Daniel Gutierrez Rojas for helping me out with polishing the Neural Networks

Table of contents

Abstract

Acknowledgements

| | | |
|-------|---|----|
| 1 | Introduction..... | 8 |
| 1.1 | Research Idea | 8 |
| 1.2 | Thesis Goals | 9 |
| 1.3 | Thesis Structure..... | 9 |
| 2 | Main Concepts of Machine Learning Algorithms..... | 10 |
| 2.1 | Artificial Neural Networks..... | 10 |
| 2.1.1 | The Beginning and Linear Classifier | 10 |
| 2.1.2 | What is an Artificial Neural Network..... | 12 |
| 2.1.3 | Architectures of Neural Networks..... | 13 |
| 2.1.4 | Applications of Neural Networks..... | 15 |
| 2.2 | Association Rule Mining..... | 17 |
| 2.2.1 | What is ARL? | 18 |
| 2.2.2 | Concepts of ARL..... | 18 |
| 2.2.3 | Quantitative Association Rule Mining (QARM) | 21 |
| 3 | Diesel Engine Faults Dataset Explanation | 24 |
| 3.1 | Physical System Description | 24 |
| 3.2 | Dataset Description..... | 26 |
| 4 | Research Procedure | 28 |
| 4.1 | Operating environment..... | 28 |
| 4.2 | Dataset Processing | 29 |
| 4.3 | QARMA experiment set-up | 29 |
| 4.4 | Neural Networks experiment set-up | 31 |
| 5 | Obtained Results..... | 33 |
| 5.1 | QARMA performance | 33 |
| 5.2 | Deep Neural Network Performance..... | 35 |
| 5.3 | QARMA vs Neural Network..... | 38 |
| 5.4 | Improvements of Fault Identification | 39 |
| | Conclusions | 40 |

| | |
|-----------------|----|
| References..... | 41 |
|-----------------|----|

Appendices

Appendix A: Python code for creation and analysis of Deep Neural Network and post-processing of QARM algorithm

Figures

Figure 1: Hyperplanes in 2- and 3- dimensional space

Figure 2: Artificial Neural Network Example

Figure 3: Some Neural Network Architectures

Figure 4: Automatic Text Processing

Figure 5: Computer for experiments

Figure 6: QARM algorithm GUI

Figure 7: QARM Rules visualization

Figure 8: Confusion Matrix for QARM algorithm

Figure 9: Confusion Matrix for NN classifier

Figure 10: Confusion Matrix for NN classifier with balanced dataset

Tables

Table 1: Function parameters description

Table 2: Computer specifications

Table 3: Amount of values in training dataset, according to each class before data resampling

Table 4: Amount of values in training dataset, according to each class after data resampling

Table 5: Accuracy scores for QARM algorithm

Table 6: Classification results for each class instance with the usage of QARMA

Table 7: Accuracies of developed Neural Network

Table 8: Classification results for each class instance with the usage of Deep Neural Network

Table 9: Accuracies with dataset balancing for Neural Network

Table 10: Classification results for each class instance with the usage of Deep Neural Network and data balancing

Table 11: Accuracy results for all methods

SYMBOLS AND ABBREVIATIONS

| | |
|-------|--|
| ARL | Association Rules Learning |
| ARM | Association Rules Mining |
| BERT | Bidirectional Encoder Representations from Transformers |
| BFS | Breadth-First Search |
| CPU | Central Processing Unit |
| DNN | Deep Neural Network |
| ECLAT | Equivalence Class-Clustering and Bottom-Up Lattice Traversal |
| FAQ | Frequently Asked Questions |
| FP | Frequent Pattern |
| GPT | Generative Pre-trained Transformer |
| GUI | Graphical-User Interface |
| MWM | Motoren Werke Mannheim |
| NAS | Neural Architecture Search |
| NN | Neural Network |
| QARM | Quantitative Association Rule Mining |
| QARMA | Quantitative Association Rule Mining Algorithm |
| TCE | Turbo Control Efficiency |

1 Introduction

Currently, artificial intelligence methods are developing rapidly. A fairly large number of machine learning and artificial intelligence algorithms have been created, and their general essence is to work with a dataset, that is related to a certain task and generate results, thanks to which many similar tasks are analysed. Such algorithms are built based on mathematical statistics and probability theory, as well as various numerical methods, optimization methods and other chapters of mathematics. Applications can be completely different - from speech recognition to bioinformatics. In this research paper, attention is focused on the ability to detect faults in engines by using a diesel engine dataset containing 4 kinds of different engine conditions. The determination of faults happens through the application of several machine learning methods. The first method is the Quantitative Association Rule Mining Algorithm (QARMA), developed recently and studied quite a little. The second method under investigation is the use of an artificial neural network, which has received a widespread acceptance in the world community and is a standard example of artificial intelligence. The work contains comparative analysis of the "success" of these methods, thus showing the strengths and weaknesses of each of the sides in faults diagnosis problem.

1.1 Research Idea

The research is based on the comparison of two different classification approaches – QARMA and Neural Networks. The comparison is done by training these algorithms on the 0 dB noise level dataset of the diesel engine faults, that was obtained in the paper of D. Pestana-Viana et al. (2019).

Diesel engines are now widely used, at least in the automotive industry, but methods for diagnosing their faults are relatively outdated. The creation and analysis of high-precision fault detection methods can reduce the human resources for diagnostics and can create a fully automated diagnostic system that will not require the presence of a human specialist.

1.2 Thesis Goals

Since this master's thesis is research thesis, its main point is the analysis. The analysis occurs by comparing the accuracy of the two methods used for classification using different metrics.

Another important goal of this work is to improve the existing results on the classification of diesel engine failures.

If there are any recommendations for improving the methods used in the work, then it makes sense to offer them.

1.3 Thesis Structure

Chapter 1 is the literature review, which describes the main concepts of used in this work machine learning algorithms.

Chapter 2 explains how the experimental dataset was obtained and what data is going to be used for as a feature vector for classification (fault detection) problem.

Chapter 3 describes the research procedure for QARMA algorithm and for the Deep Neural Network.

Chapter 4 is finalizing the research with the analysis of the obtained results

2 Main Concepts of Machine Learning Algorithms

This chapter is the literature review of the approaches, that are used in the research – Neural Networks approach and Association Rule Mining approach.

2.1 Artificial Neural Networks

Subchapter provides the reader with the theoretical background of the artificial neural networks.

2.1.1 The Beginning and Linear Classifier

Attempts to describe the principles by which a brain-like machine could solve very complex problems, scientists began to do in the middle of the 20th century. In 1943, neuropsychologist Warren McCulloch and mathematician Walter Pitts proposed a mathematical model for the functioning of an artificial neuron (Galushkin, 2016).

The model turned out to be equivalent to the threshold linear classifier, that was already known by that time (Mitchell, 1997, Duda et al., 2001). Suppose there are objects of two classes: in the problems of medical diagnostics, these are sick and healthy, and in the problems of issuing bank loans - bona fide borrowers and unscrupulous ones. Objects are described by signs - some of their characteristics, which can be both quantitative and qualitative. For example, in the case of a patient, this is his age, gender, test results, complaints, records from the anamnesis, reaction to drugs. In the case of a borrower - socio-economic data and answers to questionnaire questions, such as salary, education, profession. The set of features is called a vector. All possible vectors form a space of dimension n equal to the number of features.

The linear classifier cuts this space into two halves by a hyperplane. On one side should be vectors of healthy people, on the other - vectors of sick people. Similarly with borrowers, only this will be a different space, because the meaning of the signs in it is different: on one

side there should be vectors of conscientious borrowers, on the other - vectors of those who do not like to repay debts. For this to be true, the position of the hyperplane must be correctly oriented, deployed in an optimal way so that there are as few errors as possible on the vectors known to us. We call this process learning, and vectors with known classifications - training set. The constructed hyperplane has the ability to classify new objects, that is, to make decisions automatically instead of an expert: the patient is already healthy or still sick; the borrower will be able to repay the debt to the bank or not. Planes, as ordinary examples of hyperplane are shown at the Figure 1.

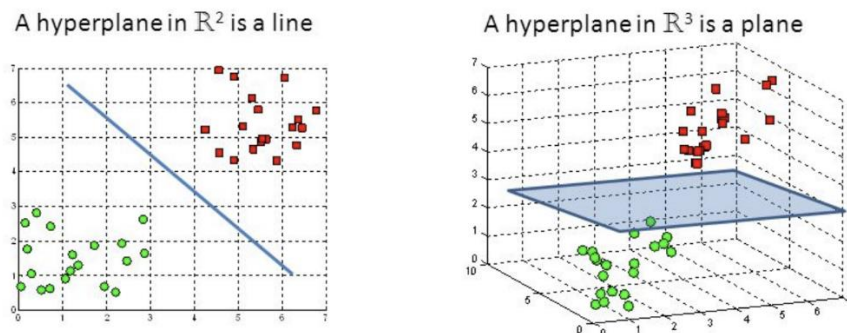


Figure 1. Hyperplanes in 2- and 3- dimensional space (DeepAI, 2019)

The term "hyperplane" used is because ordinary planes are in three-dimensional space, and our space is n -dimensional. But we are forced to imagine ordinary planes when we turn to geometric intuition. From the point of view of programming, everything is even simpler: a linear neuron (aka a linear classifier) receives n features at the input and calculates their sum, multiplying each of them by its own weight coefficient. Then it applies the so-called activation function to the resulting sum. It can be arranged very simply: for example, if the sum turned out to be greater than zero, then the object belongs to the first class, and if less, then to the second. Finding the optimal position of the separating hyperplane is the same as determining such weight coefficients for which the training sample is classified in the best way, that is, with a minimum number of errors.

Classification tasks are encountered in various fields of human activity. An artificial neural network consists of a huge number of linear neurons that work together and therefore are able to separate objects with much more complex surfaces than hyperplanes. So, what artificial neural network really is and how to teach it?

2.1.2 What is an Artificial Neural Network

A neural network is a superposition of a huge number of linear neurons. A superposition in mathematics is a usage of one function as an input to the other function (MathIsFun, 2017). In our case, functions should be understood as neurons. There are neurons that take n features of the object as input. They form the first layer of the network. At the output, each such neuron gives one number, so all together they form a vector of new features of the object. The dimension of this vector is equal to the number of neurons in the first layer. Continuing to act in the same spirit, you can build a second layer - these are neurons that take as input not the original features, but those formed by the first layer. For each specific task, we decide how many layers to set. Usually, the more difficult the task, the more layers we will need, but we will also need more data to train all the weights in all neurons. The visual representation of artificial neural network is shown at the Figure 2.

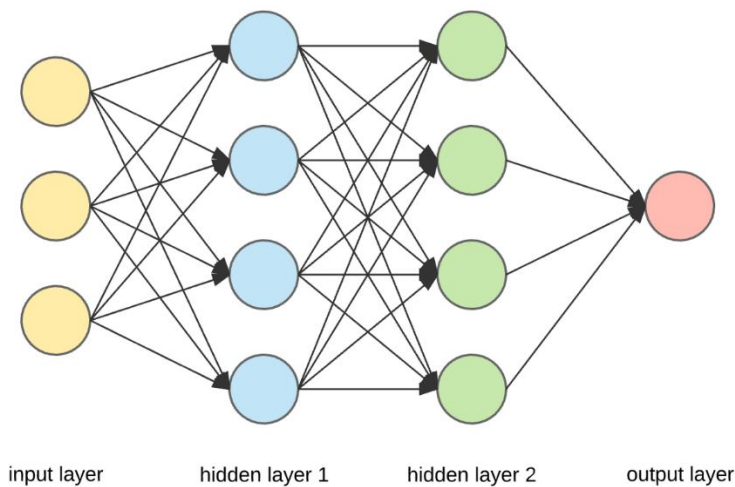


Figure 2. Artificial Neural Network Example (TextileValueChain, n.d.)

Machine learning in general is built on solving optimization problems. We set a criterion that we want to minimize - for example, the total network error on the training sample vectors - and run a tricky algorithm that changes the coefficients throughout the network step by step, gradually turning the hyperplane of each neuron to the best position. This is usually done using the gradient descent method (Chong et al., 2013), which can be illustrated with a simple example.

All of us at school solved simple optimization problems. To find the minimum of the function $f(x)$, need to find its derivative, equate it to zero and, having solved the resulting

equation, find the minimum point (perhaps there will be several of them or at the same time the maximum points will be with them, and then need to figure out what each of them is).

When training a neural network, we are looking for a minimum point of a very complex function (criterion) in a multidimensional space (Mitchell, 1997). The school trick with equating the derivative to zero no longer works here, but the technique used is very similar. We must find the derivative of the criterion with respect to each weight coefficient. The derivative shows how quickly the criterion increases by a given coefficient. And if all these derivatives are collected in a vector, which is called a gradient, then it will show us the direction of the fastest increase of our criterion in the space of coefficients. If we are going to minimize the criterion, then we will have to go in the opposite direction, in the direction of the anti-gradient. It is also important to adjust the step length, which determines how much we go down, and this is related to such a parameter as the learning rate. Over the past decades, many methods have been invented to help determine the length of the step, as well as to coordinate subsequent steps with the previous ones, thanks to which neural networks can now be trained very quickly (Wang, 2017, Congcong, 2019)

2.1.3 Architectures of Neural Networks

Architecture of a Neural Network is a form of Network creation. Possible architectures are shown at the Figure 3.

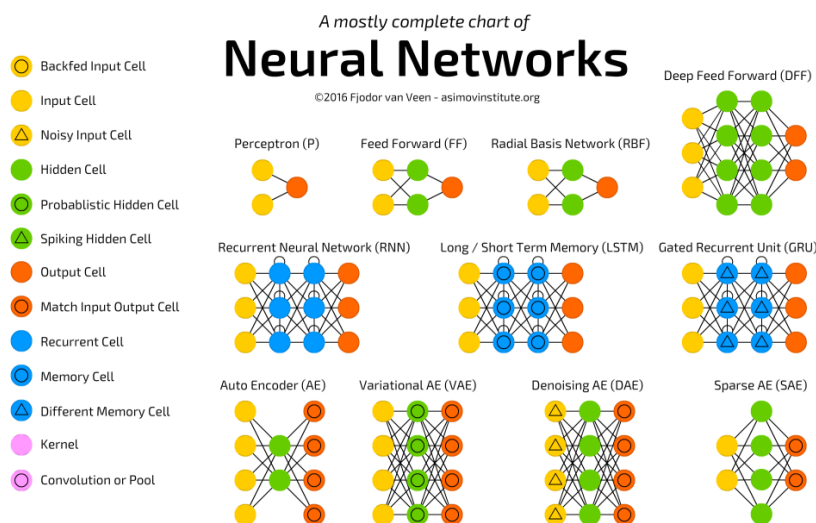


Figure 3. Some Neural Network Architectures (Sheidaei, 2019)

Each layer of neurons transforms the input vector into an output vector, and the dimension of the output vector is equal to the number of neurons in the layer. Many such transformations can be performed sequentially, and this is called a multilayer (deep) neural network (Mitchell, 1997). Why is it so difficult and why do you need many layers? The fact is that each layer is a set of linear classifiers, which are individually quite primitive. Each layer specializes in one relatively simple vector transformation. Together they form something like a conveyor. The vectors that enter the input of the pipeline are inconvenient, and the problem cannot be solved with them. However, after a series of processing, the output vector becomes convenient, and the problem is going to be solved. When does this “convenience” happens? The answer is simple: when all layers are trained together according to the final quality criterion, this forces them to adapt to each other and form the desired transformation pipeline. If the task is complex, then all the necessary transformations over the vector may simply not fit in a short pipeline of a small number of layers.

Similar logic is used in convolutional neural networks for image processing (Zhang, 1990). A convolution is a neuron that performs weighted averaging of brightness over neighboring pixels. If you apply convolution uniformly to all the pixels in an image, you may end up with a slightly blurry image, or an image that highlights certain color transitions brighter. As a rule, dozens or hundreds of different convolutions are applied simultaneously to one image, so that the output is an array of images. Layers of convolutional neurons alternate with pooling layers, which are more primitive neurons that do not have weight coefficients: they coarsen the image, reducing its size by several times. What is the essence of the transformations that such a pipeline implements? Each next pair of "convolution + pooling" layers reduces the size of the image, increases the dimension of the vector in each pixel, and detects larger image elements than the previous pair. At the output of the pipeline, the image size shrinks to a 1×1 point, but it is represented with a vector of a rather impressive dimension, for example 4000, which encodes information about all the "interesting" objects in the image. The mechanism of this miracle is the same: such a network is usually trained to recognize thousands of different objects in millions of images. Therefore, in the process of gradient optimization, the convolutional layers that form the pipeline adjust to each other so that the final image vector is as useful as possible for error-free recognition (Congcong, 2019).

Recurrent neural networks are very popular for processing signals and texts (Chatterjee, 2020). For these networks, it is fundamentally important that they process a sequence of input vectors. At the output, such a network generates a special state vector, which is fed to it as input when processing the next input vector. Passing a state vector between neighboring vectors allows the network to store important information about the history of the sequence being processed. Recurrent networks can classify not only entire sequences, but also their individual fragments, as well as transforming one sequence into another. Such tasks include machine translation, speech recognition and synthesis, and dialogue support (Chatterjee, 2020).

Another important type of architecture is called an autoencoder. This is an ordinary deep neural network, in which the vector first gradually, layer by layer, reduces its dimension, and then restores it (Nelson, 2020). The goal is to have the reconstructed vector match the original as closely as possible. The benefit of this learning principle is that the network learns to compress (encode) the vector so that it contains all the necessary information to decompress (decode) it without significant loss of accuracy. That is, we learn to form compressed, but the most informative vector representations of objects. Autoencoders as an element of neural network architecture are used in many tasks for the most rational vectorization of objects.

2.1.4 Applications of Neural Networks

A serious catalyst for the currently observed boom in neural networks was the international competition for object recognition systems in ImageNet images. In 2012, Jeffrey Hinton and two of his graduate students, Alex Krizhevsky and Ilya Sutskever, created the AlexNet convolutional deep learning network for this competition (d2l, 2020). ImageNet then included around million images that were collected from the Internet and manually marked up by users using Amazon's Mechanical Turk crowdsourcing. At that time, this was a huge amount of data for training neural networks, since earlier recognition algorithms were trained on datasets that included tens of thousands of images at best. At the beginning of the competition, such algorithms gave about 30% of errors. ImageNet currently contains approximately 15 million images divided into 22,000 categories. The AlexNet convolutional network allowed a sharp decrease in the error rate to 16% in the ImageNet competition. Since then,

only deep neural networks have led the competition. 2015 was marked by overcoming the barrier of 5% - this is the level of human errors, and the final mark stopped at about 2%. This means that image recognition technologies have been created to solve a huge number of automation tasks - from face and license plate recognition to scene analysis for unmanned driving.

Another type of problem that has large training sets and where deep neural networks perform amazingly well is natural language processing (Andenko, 2021). In computational linguistics, people have been doing language modeling for a long time and building probabilistic models of language. A language model can be represented mathematically as a prediction of a word in a text given its context. The context is understood in different ways: one can look at the previous ten words, or at the previous three sentences, or at the text from its very beginning. Possible text processing pipeline is shown at Figure 4.

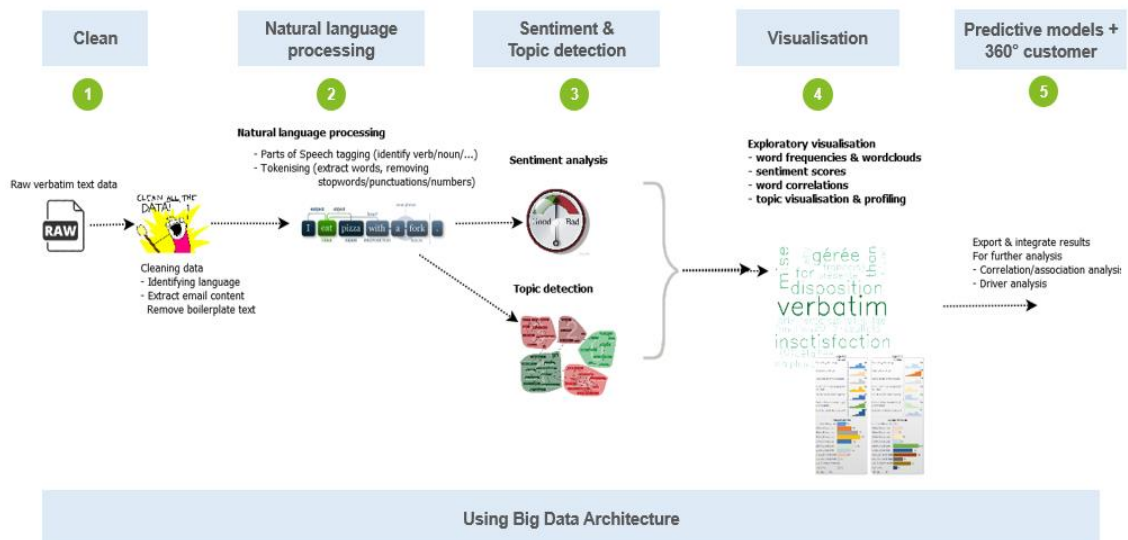


Figure 4. Automatic Text Processing (Bnosac, 2017)

During experiments in which participants were closed one word in a sentence and asked to guess this word, it turned out that a person correctly guesses it on average with a perplexity of 10. Perplexity is an assessment of the quality of a language model and shows how well the model predicts words: if perplexity is 1000, this means that any word out of a thousand could be in place of the gap; if perplexity is 10, then any word out of ten. The less perplexity, the higher the quality of the language model (Brown et al., 1992).

Recently emerging neural network language models with complex architectures - for example, BERT, GPT-1 (scheme, GPT-2, GPT-3 - have learned to predict words in sentences with the same accuracy as a person does (DesertFlow, 2019). The reason for this progress is that these are models that are trained on a huge array of texts, with a volume of half a terabyte (Habr, 2019).

However, it cannot be said that an artificial neural network understands any meaning of the texts it analyzes: no models of understanding are laid here. Nevertheless, neural networks are capable of generating fake news and even jokes that are very similar to real ones. This is because “the network has seen everything in the language”, having learned from such a huge corpus of texts. Now the efforts of the scientific community are aimed at learning how to combine such models with extralinguistic information (knowledge of the surrounding world), with logical reasoning algorithms, associative thinking and other elements necessary for a true understanding of the language. So far, neural networks remain only an imitation of intelligence.

Deep neural networks have made an important breakthrough in machine learning technologies by automating the vectorization of complexly structured objects. In the past, each image, signal, or text processing task required a team of feature engineers, and this work often took the lion's share of resources in projects. Deep neural networks have actually automated this type of work, drawing them into a common circuit with model training. Recently, another similar breakthrough is planned. Methods like NAS (Neural Architecture Search) are ready to automate the processes of choosing architectures and selecting model hyperparameters, including for other neural networks (Weng, 2020).

2.2 Association Rule Mining

Chapter describes the association rule mining algorithms, starting on the simplest one and ending on QARMA – quantitative association rule mining algorithm.

2.2.1 What is ARL?

The volumes of modern databases, which are quite impressive, have caused a steady demand for new high-scalable data analysis algorithms. Such algorithms are quite popular nowadays and are called algorithms for searching for association rules (Hahsler, 2005, Agrawal et al., 1993) They allow you to find different patterns in terms of related events to each other. Example could be the statement that a person who buys "Beer" will also buy "Chips" in a 40% probability. This is what is called "Learning on associative rules" (Associations rules learning – ARL, ARM – M stands for mining) is the topic of discussion in this chapter.

The first association rule search algorithm was developed in 1993. IBM Almaden Research Center took care of it and algorithm was called AIS [Agrawal et al., 1994]. Since this pioneering work, interest in association rules has increased. In the mid-1990s research became high intense in this area, and since then, several algorithms have appeared every year.

2.2.2 Concepts of ARL

The problem of finding association rules was proposed to solve one of the basic marketing questions: find typical patterns of purchases in supermarkets. That is why sometimes it is also called market basket analysis. But first, let us define the needed notions in ARL – *support, confidence, and lift*:

$$supp(X) = \frac{|\{t \in T; X \in t\}|}{|T|} \quad (1)$$

$$supp(X \Rightarrow Y) = \frac{|\sigma(X \cup Y)|}{|T|} \quad (2)$$

$$conf(X \Rightarrow Y) = \frac{supp(X \Rightarrow Y)}{supp(X)} \quad (3)$$

$$lift(X \Rightarrow Y) = \frac{supp(X \Rightarrow Y)}{supp(X) \times supp(Y)} \quad (4)$$

Formulas 1 and 2 stand for support, while 3 and 4 stand for confidence and lift. X and Y are the analyzing sets, T is the total number of transactions and σ is the number of transactions that contain both sets.

Now let us consider a database, that consists of customer transactions. Transaction represents a set of goods bought by the person in one shop visit. Other term for this transaction is a market basket. Case: 60% of transactions that contain "Coffee" also contain "Cream". Total of 5% of all the transactions contain these things together. Reliability (confidence) of the rule is 60%, support is 5%, or "Coffee" "Cream" with a probability of 60%.

It is possible to say, that the purpose of such analysis is to establish the dependencies: if a specific element set X was encountered in a transaction, by knowing this we can make a conclusion that another element set Y must also be in this transaction. Establishing this kind of dependencies give us possibility to find quite simple and intuitive rules (Piatesky-Shapiro, 2006).

Association rule search algorithms are designed to find all XY rules. Support and confidence of such rules must be above certain predetermined thresholds. The first one is the minimum support, which shortens to minsupport. The second one is the minimum confidence, which also shortens to minconfidence.

How to find association rules? Such task is divided into 2 subtasks:

1. Find all element sets, which satisfy the minimum support threshold. These sets of elements are called frequent sets;
2. Generation of rules from element sets found with a confidence that satisfies the minimum confidence threshold.

APriori algorithm was one of the pioneer algorithms, that was able to efficiently solve such class of problems. Other algorithms have been developed in addition to it: DHP, Partition, DIC.

Minimum Support and Minimum Confidence parameters are chosen to make the number of rules, that were found by an algorithm, limited. When we consider support as an important

parameter, then the algorithms will search for the rules which analysts comprehend or for the ultra-obvious rules, to which there is no reason in doing this analysis.

In contrary, low support value generates a huge number of obtained rules, what requires significant calculations. Nevertheless, rules of the main interest have the low value of the support threshold. Too low support value generates unreasonable rules from the statistical point.

The search for association rules is not a simple task at all, even if it looks like this. Complexity in finding frequently occurring sets of elements, is one of the main problems. That is because with the increase of the elements, potential sets of elements are starting to grow exponentially.

By the way, when searching for association rules, we assumed that the elements under analysis are homogeneous. Returning to an analysis of the market basket: the name is the only different thing for these goods, while they have completely the same attributes. However, it is not a difficult task to supplement the transaction with the information of which product group the product belongs to and build a hierarchy of products. Example of such a grouping (taxonomy) is in the form of a hierarchical model. Let us be given a database of transactions D and we know which groups (taxons) the elements are included in. Then you can extract from the data the rules that link groups to groups, individual elements to groups, and so on.

Example: if the Buyer bought a product from the "Soft Drinks" group, then he will also buy a product from the "Dairy Products" or "Juice" group "Dairy Products". Such kind of rules are named as generalized association rules.

Hierarchy grouping of elements, that will contain additional information, will provide benefits as:

1. Association rules will also be created between levels of the hierarchy, as well as between individual elements.
2. Individual elements could have insufficient support, but in general group may satisfy the threshold of minsupport.

To find such rules, it is needed to use any of the above algorithms. To do this, each transaction must be supplemented with all the ancestors of each element included in the transaction. However, the use of these algorithms "head on" will inevitably lead to the following problems:

1. Elements at higher levels of the hierarchy tend to have significantly higher support values compared to elements at lower levels.
2. With the addition of groups to transactions, the number of attributes has increased and, accordingly, the dimension of the input space has increased. This complicates the task and also leads to the generation of more rules.
3. The appearance of redundant rules that contradict the definition of a generalized association rule, for example, "Juice" "Soft drinks". Obviously, the practical value of such a "discovery" is zero at 100% certainty. Therefore, special operators are needed to remove such redundant rules.

To find generalized association rules, it is desirable to use a specialized algorithm that eliminates the above problems and also works 2-5 times faster than the standard APriori (Wang et al., 2017, Congcong, 2019). It is possible to group the items not only by being included in a certain product group, but also by other characteristics, for example, by price (cheap, expensive), brand, etc.

2.2.3 Quantitative Association Rule Mining (QARM)

When searching for association rules, the task was significantly simplified. In fact, it all came down to whether the element is presented in the transaction or not. If we consider the case of a market basket, then we considered two states: a product was bought or not, ignoring, for example, information about how much was bought, who bought it, the characteristics of the buyer, etc. And we can say that we considered "Boolean" associative rules. If you take any database, each transaction consists of various types of data: numeric, categorical, and so on. A search algorithm was proposed to process such records and extract numerical association rules.

An example of a numerical association rule: [Age: 30-35] and [Marital status: married] [Monthly income: 1000-1500 dollars].

In addition to the association rules described above, there are indirect association rules, association rules with negation, temporal association rules for time-related events, and others.

As mentioned, the problem of finding association rules was first introduced for market basket analysis. Association rules are effectively used in customer segmentation by shopping behavior, analysis of customer preferences, planning the location of goods in supermarkets, cross-selling, direct mailing. However, the scope of these algorithms is not limited to just one trade. They are also successfully used in other areas: medicine, for analyzing web page visits (Web Mining), for text analysis (Text Mining), for analyzing census data, in analyzing and predicting failures of telecommunications equipment, etc.

The pipeline and details of the QARMA work is (Christou, 2018):

1. Construction of the variables subsets by user with the target variable (whether it is a fault indicator or something else) of specified length. Such subsets are, as previously, named “itemsets”;
2. Producing of all the correctly working QARs (quantitative association rules) from each itemset of the specified size. In parallel way all the frequently occurring itemsets of size equal to 1 are considered by an algorithm within every phase of obtaining all correct rules of size 1. All possible rules for a given itemset are produced. Such rules have each attribute, that is unquantified in the beginning – for that, a different CPU core will run a procedure that is called `QUANTIFY_RULE()`, which will maintain a set of local rule R (at initial that is empty). After that, the special modified breadth-first procedure is being run, which firstly will assign the consecutive attribute to the value, which has the highest possibility to be assigned. As long as the resultant rule, which is quantified partially, has `minsupport` above the required threshold – it is being added to a data structure, that named T (queue data);
3. The first inserted into this structure rule is retrieved and then removed out of the structure, while it is not empty. Algorithms creates as many rules as possible, what depends on the amount of distinguished dataset values for the examined in an ascending order attribute value and finally enters the T queue in that order. But – it only happens if the new created rule meets the `minconfidence` (any needed metric). With that rule is going to be checked against the state local rules set R to understand

- does it has mastery over in R by another rule or not. If there is no any other rule in R, which could dominate the current state rule – current state rule is being added to R. As this process for all the itemsets of specified length that are frequent runs in parallel, different CPUs are acting in it, for them to synchronize the obtained data rules from other ones before they will move to handle the frequent itemsets of length specified+1;
- 4. Theoretical property of the resultant rule: coverage of the whole processed dataset. There is no rule exists, that is beyond the produced dataset in the above-described form, which is able to cover something, while having the minsupport and minconfidence on required level.

After, computation of all the rules which tend to be non-dominated is performed and a special classifier is constructed, which will be on the basis of their overall ensemble works as:

1. Rules selection is performed. The only selected rules are the ones, which previous conditions are fulfilled by this scenario and then adding operation of them to the set F is performed;
2. Decreasing order sorting of the rule-set F by confidence and by the training set support;
3. Top hundred rules of the whole sorted set F are now in usage;
4. Every rule in F has a weight – the training set confidence, that is equal to it;
5. The class of the instance is assigned with the usage of the specially weighted majority vote, which works on F.

According to the works, the QARMA produces the set of quantitative rules, which are easily explainable, so that a human won't be "afraid" of it (Christou, 2018, Gutierrez-Rojas et al., 2022). The extracted rules are trivially on a basic level checked against the training dataset for validation, so it becomes easy to understand "what does that mean", because pre-conditions of the rule are not more than just a restrictions compound of the attributes, which envelop previous rules to specific intervals. QARMA could be a good participant in a competition of making the artificial intelligence more explainable. More widely, pros and cons of QARMA will be considered in the research, by applying this method to the Diesel Engine Faults dataset, explanation of which is going to be in the next chapter.

3 Diesel Engine Faults Dataset Explanation

This chapter explains, how the training dataset was obtained and which data of it is used as a features vector.

3.1 Physical System Description

Usual types of condition in diesel engines are: no-fault condition, fault, that is caused by compression, fault, that is caused by injected fuel mass and fault, because of the intake manifold pressure (Pestana-Viana et al., 2019). The chosen engine for experiment is Acteon 6.12 TCE by MWM Diesel Motors with four stroke cycle engine.

With engine failure parameters modification, signals of faults are created, and the model is obtained, which can represent the operational state of the engine. The model is represented with the following equations.

$$P_i(\theta) = f(P_a, P_r, T_p, r_i, m_{a_i}, m_{c_i}, \theta_{inj_i}) \quad (5)$$

$$T_i(\theta) = f(P_a, P_r, T_p, r_i, m_{a_i}, m_{c_i}, \theta_{inj_i}) \quad (6)$$

$$V(t) = f(P_a, P_r, T_p, \{r\}, \{m_a\}, \{m_c\}, \{\theta_{inj}\}, \{K\}) \quad (7)$$

Parameters of the formulas 5-7 described in Table 1.

Table 1. Operating parameters description

| Parameter | Description |
|---------------|---|
| $P_i(\theta)$ | instantaneous pressure curve in the interior of each cylinder i |
| $T_i(\theta)$ | instantaneous temperature curve inside each cylinder |
| $V(t)$ | instantaneous torsional vibration response |
| P_a | pressure in the intake manifold |
| P_r | common rail pressure |

| | |
|------------------|---|
| T_p | cylinder walls temperature |
| r_i | compression ratio |
| m_{a_i} | air mass admitted to cylinders |
| m_{c_i} | fuel mass injected into cylinders |
| θ_{inj_i} | angle of injection start in each cylinder |
| K | cranks stiffness |

$$P_i(\theta) = f(\Delta P_a, \Delta P_r, \Delta T_p, \Delta r_i, \Delta m_{a_i}, \Delta m_{c_i}, \Delta \theta_{inj_i}) \quad (8)$$

$$T_i(\theta) = f(\Delta P_a, \Delta P_r, \Delta T_p, \Delta r_i, \Delta m_{a_i}, \Delta m_{c_i}, \Delta \theta_{inj_i}) \quad (9)$$

$$V(t) = f(\Delta P_a, \Delta P_r, \Delta T_p, \{\Delta r\}, \{\Delta m_a\}, \{\Delta m_c\}, \{\Delta \theta_{inj}\}, \{\Delta K\}) \quad (10)$$

Generation of the simulated signals is performed via solving the equations 8-10 with the usage of severity conditions input parameters, which are going to emulate the needed operating condition.

Severities allocation, which needs some input variables, is represented in the following equation.

$$\Delta p(\%) = \frac{P_n - P_f}{P_n} \cdot 100 \quad (11)$$

Where in equation 11: $\Delta p(\%)$ is the failure parameter variation, so that p_n – normal operating condition and p_f – fault condition.

Now it is possible to describe the engine conditions via equations, obtained previously.

- Normal (no faults): no fault situation. Performance curves created by manufacturer show the operational condition. No severity added into the input variables of the equations 8-10 for simulation purposes;
- Intake manifold pressure reduction. Occurs due to the change in ambient conditions, deposits in the intake valves, incorrect turbocharger operation, strange variations of

the intake valves angles (open/close). Severity inserted into variable ΔP_a of the equations 8-10 for simulation purposes;

- Cylinders compression ratio reduction. Occurs if - cylinder head gasket has some variation or combustion chamber dead volume is changing because of corrosion or deposits. Kinematic components variation could also make it. Severity inserted into variable Δr of the equations 8-10 for simulation purposes;
- Reducing of the fuel intake into the cylinder. It could happen because of segment rings/cylinder walls corrosion. Intake valves bad sealings because of corrosion. Spring deposits and failures or some injection pump problems. Severity is being inserted into variable Δm_c of the equations 8-10 for simulation purposes.

3.2 Dataset Description

The obtained dataset comprises with all the things, described in previous few chapters and contains these operating conditions:

- Normal (no faults). No fault situation is reproduced and presented in 250 instances total. Adding 0.1% of the maximum possible severity, that has Gaussian normal distribution (probability coverage from 0 to 0.1 %) allowed to create 51 distinguished realizations. The idea is to emulate the normal operating condition, where machine variables fluctuate within a small range;
- Intake manifold pressure reduction. Few instances with severities of [1, 2, ..., 50] % related to variable ΔP_r are considered. 250 instances in total;
- Cylinders compression ratio reduction. All the cylinders create the instances. Couple of instances with severities of [1, 2, ..., 50] % that relate to variables Δr_i and cylinders from 1 to 6 are considered. 300 instances for each cylinder, what is 1500 cases in total;
- Reducing of the fuel intake into the cylinder. All the cylinders create the instances. Couple of instances with severities of [1, 2, ..., 50] % that relate to variables Δm_{c_i} and the cylinders from 1 to 6 are considered, so that there are 300 cases for each cylinder and 1500 in total.

The entire dataset consists of 3500 fault cases, which represent 4 distinct working conditions. 250: normal condition; 250: 1-fault class; 1500: 2-fault class and 1500 from the 3-fault class. This is the 0 dB noise level dataset, that can be accessed from an open-source (Pestana, n.d.).

4 Research Procedure

Chapter describes the idea of how the research procedure was conducted. The procedure of run of QARMA and the development of Neural Network with its description.

4.1 Operating environment

The experiments will be conducted on a machine, that has the following technical specifications, which are shown in the Table 2.

Table 2. Computer specifications

| | |
|------------------|--|
| CPU | Intel i9 10920X CPU 4.47 GHz (12/24 cores) |
| Graphics card | Nvidia 2070 RTX SUPER GPU |
| RAM | 64 GB |
| Data storage | SSD 2 TB + HDD 4 TB |
| Operating system | Windows 11 Pro |

An image of a computer is shown at the Figure 5.



Figure 5. Computer for experiments

4.2 Dataset Processing

Dataset in overall contains 3500 rows with different faults, that represented by 4 classes of operational conditions. “Normal” class with the 2-fault class has 250 scenarios each, while 3-fault class and 4-fault class has 1500 scenarios each. Dataset is represented as a MATLAB storage file with extension “.mat” (Pestana, 2019). For the processing of such dataset, the data was transferred into CSV format.

For the needs of basic machine learning experiment, the dataset was then randomly shuffled and splitted into 90% of the training data and 10% of the validation data. The used columns for machine learning algorithms of the dataset are the first 84 ones, that are correspond to the “features vector”, according to the dataset description.

The description of each experimental set-up will be described in the next sections.

4.3 QARMA experiment set-up

According to the above information about QARM algorithm, it will produce the rules of the form

$$[a > feature_i > b] \rightarrow \text{class X}$$

OR

$$[c > feature_k > g] \text{ AND } [d > feature_l > u] \rightarrow \text{class Y}$$

Rules are going to be hold with support of at least 0.5% and confidence at least 80%. Having collected all the rules, given a new (unclassified) data instance, it will be classified according to the majority of the votes of the top 100 rules (sorted in descending order of confidence) that fire on that instance.

The 84-dimensional vector (first 84 columns of dataset) is fit into training data and the classes (last 98th column of dataset) are acting as a target data for multiclass classification.

Dataset is uploaded into the QARMA GUI, which has different customization parameters, and the experiment could be conducted. It is shown on the Figure 6.

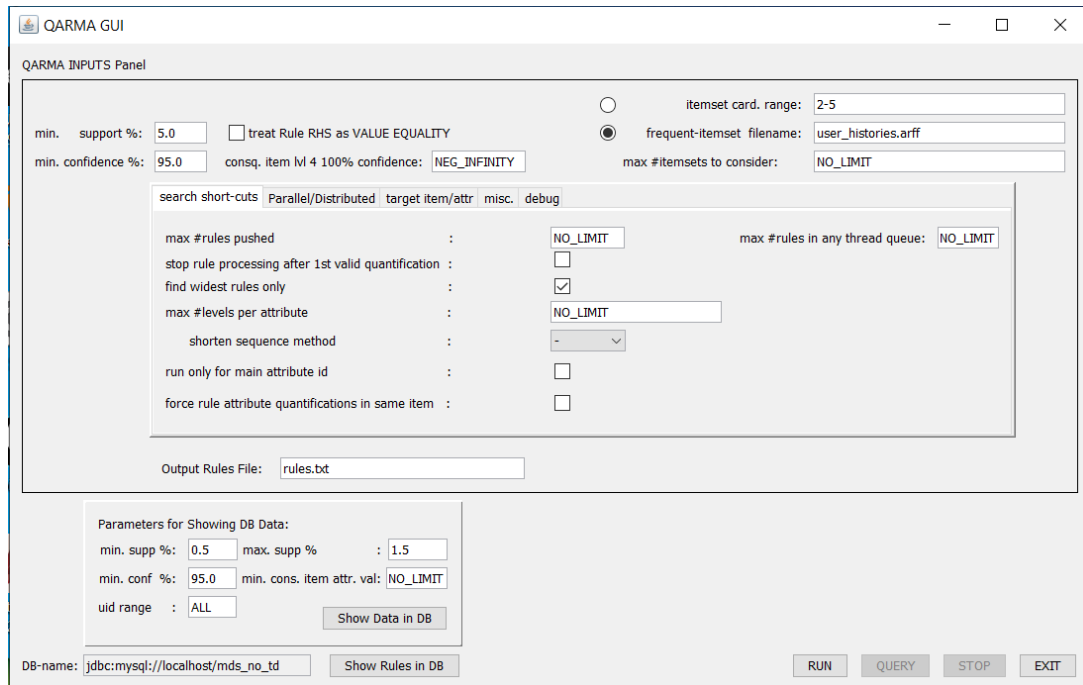


Figure 6. QARM algorithm GUI

Support and Confidence (see 1.2.2) are set to 5% and 95% accordingly to provide accurate enough results. Also, there’s no reason for creating small rules, because the feature vector is big and the rules won’t be small anyways – because of that, the checkbox at “find widest rules only” is ticked.

At the Figure 7 the obtained results are represented with the rules. Results are going to be described in the next chapter 4.

[ACCEL-1_0_2D_p == 0.025075178] * [CSI_1_1p_p == 1.220954327] * [Z1_1_1p_p == 0.142101950] * [Z1_1_1p_price_negated == -0.569087148] -> [RULParts = 1196.0] [Supp=3.4375% Conf=95.93023258129054%]

[ACCEL-1_0_3D_p == -0.063339922] * [ACCEL-1_0_3D_price_negated == 0.006150723] * [CSI_1_1p_p == 0.966550529] * [CSI_2_1p_p == 1.204809904] * [CSI_2_1p_price_negated == -1.611307228] -> [RULParts = 4788.0] [Supp=1.3958333333333332% Conf=95.71428571428571%]

[ACCEL-1_0_1D_p == 0.041204623] * [CSI_1_1p_p == 0.296807983] * [CSI_1_1p_price_negated == -0.275651641] * [CSI_1_0_1D_p == 0.0123201818] * [CSI_1_0_1D_price_negated == -1.170349673] -> [RULParts = 1196.0] [Supp=3.4375% Conf=95.93023258129054%]

| ID | Secs_after... | Z1_1 | Z1_0 | CSI_2 | BIS_ACC-1_0 | CSI_0 | CSI_1 | BIS_ACC-1_1 | ACCEL-1_0_1D | ACCEL-1_0_2D | ACCEL-1_0_... | CSI_0_1p | CSI_1_1p | CSI_2_1p | Z1_1_1p | Z1_0_1p | CSI_AWG | RULParts |
|------|---------------|-------|-------|-------|-------------|-------|-------|-------------|--------------|--------------|---------------|----------|----------|----------|---------|---------|---------|----------|
| 4865 | 4799723 | 0.556 | 0.589 | 0.647 | 0.964 | 0.414 | 1.212 | 0.044 | 0.018 | 0.04 | 0.038 | 0.341 | 1.24 | 0.724 | 0.497 | 0.632 | 0.757 | 114 |
| 4895 | 4179254 | 0.546 | 0.57 | 0.923 | 1.016 | 0.197 | 1.306 | 0.085 | 0.038 | 0.033 | 0.033 | 0.197 | 0.242 | 0.743 | 0.536 | 0.576 | 0.809 | 1594 |
| 4932 | 1481238 | 0.52 | 0.569 | 0.77 | 1.033 | 0.19 | 1.234 | 0.073 | -0.005 | 0.033 | 0.028 | 0.252 | 0.298 | 0.87 | 0.564 | 0.63 | 0.738 | 2886 |
| 4956 | 4261778 | 0.498 | 0.716 | 0.854 | 1.016 | 0.415 | 1.288 | 0.068 | -0.028 | 0.027 | 0.037 | 0.273 | 0.298 | 0.952 | 0.497 | 0.725 | 0.852 | 1594 |
| 4976 | 4262045 | 0.491 | 0.672 | 0.871 | 1.044 | 0.155 | 1.299 | 0.085 | 0.022 | 0.046 | 0.028 | 0.26 | 0.328 | 0.987 | 0.497 | 0.676 | 0.776 | 1594 |
| 5078 | 4287840 | 0.538 | 0.705 | 0.785 | 1.057 | 0.491 | 1.25 | 0.107 | 0.029 | 0.047 | 0.012 | 0.343 | 0.323 | 0.972 | 0.488 | 0.668 | 0.842 | 1594 |
| 5081 | 4131867 | 0.517 | 0.634 | 0.88 | 1.028 | 0.384 | 1.295 | 0.07 | 0.056 | 0.031 | -0.001 | 0.065 | 0.245 | 0.774 | 0.529 | 0.541 | 0.853 | 1594 |
| 5093 | 4872303 | 0.526 | 0.587 | 0.76 | 1.172 | 0.332 | 1.242 | 0.198 | 0.093 | 0.126 | 0.08 | 0.679 | 0.321 | 0.901 | 0.509 | 0.663 | 0.778 | 1594 |
| 5151 | 4752013 | 0.517 | 0.651 | 1.713 | 1.115 | 1.5 | 1.562 | 0.152 | 0.085 | 0.094 | 0.027 | 0.54 | 0.291 | 0.802 | 0.527 | 0.632 | 0.591 | 1594 |
| 5182 | 4766937 | 0.537 | 0.68 | 0.814 | 1.018 | 0.42 | 1.287 | 0.064 | -0.018 | 0.034 | 0.007 | 0.443 | 0.39 | 1.102 | 0.461 | 0.705 | 0.841 | 1594 |
| 5204 | 4260339 | 0.586 | 0.684 | 0.892 | 1.031 | 0.696 | 1.192 | 0.079 | -0.012 | 0.031 | -0.014 | 0.465 | 0.398 | 1.094 | 0.502 | 0.725 | 0.927 | 1594 |
| 5210 | 4761697 | 0.507 | 0.674 | 1.317 | 1.084 | 0.865 | 1.45 | 0.148 | 0.061 | 0.07 | 0.019 | 0.571 | 0.359 | 0.957 | 0.536 | 0.657 | 1.211 | 1594 |
| 5211 | 4830223 | 0.577 | 0.663 | 0.761 | 1.153 | 0.527 | 1.243 | 0.182 | -0.026 | 0.04 | 0.015 | 0.521 | 0.323 | 0.948 | 0.547 | 0.673 | 0.844 | 1594 |
| 5215 | 4772427 | 0.441 | 0.612 | 1.135 | 1.078 | 0.404 | 1.404 | 0.115 | 0.014 | 0.052 | 0.07 | 0.558 | 0.315 | 0.867 | 0.472 | 0.618 | 0.981 | 1594 |
| 5233 | 4261770 | 0.497 | 0.725 | 0.952 | 1.044 | 0.273 | 1.298 | 0.1 | 0.055 | 0.055 | 0.013 | 0.579 | 0.274 | 0.787 | 0.535 | 0.654 | 0.841 | 1594 |
| 5238 | 4777824 | 0.577 | 0.659 | 0.931 | 1.095 | 0.821 | 1.292 | 0.12 | 0.027 | 0.035 | 0.076 | 0.516 | 0.32 | 0.977 | 0.564 | 0.734 | 1.015 | 1594 |
| 5266 | 4128383 | 0.538 | 0.582 | 0.937 | 1.033 | 0.063 | 1.288 | 0.075 | 0.059 | 0.058 | 0.014 | 0.267 | 0.235 | 0.702 | 0.55 | 0.645 | 0.763 | 1594 |
| 5287 | 4918946 | 0.516 | 0.682 | 0.857 | 1.15 | 0.466 | 1.266 | 0.189 | 0.099 | 0.046 | 0.047 | 0.574 | 0.283 | 0.823 | 0.517 | 0.623 | 0.863 | 1594 |
| 5337 | 4760905 | 0.551 | 0.758 | 0.992 | 1.083 | 0.431 | 1.33 | 0.137 | 0.091 | 0.084 | -0.117 | 0.543 | 0.346 | 0.964 | 0.47 | 0.61 | 0.918 | 1594 |
| 5488 | 4843663 | 0.585 | 0.592 | 1.005 | 1.256 | 0.449 | 1.361 | 0.213 | 0.203 | 0.18 | 0.02 | 0.442 | 0.295 | 0.845 | 0.518 | 0.715 | 1.005 | 1594 |
| 5554 | 4836785 | 0.486 | 0.594 | 1.287 | 1.283 | 0.722 | 1.451 | 0.319 | 0.104 | 0.184 | 0.054 | 0.289 | 0.236 | 0.75 | 0.511 | 0.661 | 1.156 | 1594 |
| 5614 | 3728214 | 0.2 | 0.535 | 0.656 | 1.014 | 0.467 | 1.202 | 0.057 | -0.022 | 0.038 | 0.041 | 0.289 | 0.279 | 0.843 | 0.203 | 0.389 | 0.775 | 1594 |
| 5621 | 4221970 | 0.508 | 0.605 | 1.055 | 1.063 | 0.266 | 1.338 | 0.096 | 0.067 | 0.069 | 0.096 | 0.353 | 0.344 | 0.701 | 0.694 | 0.612 | 0.887 | 1594 |

Rules found for: 26850

Figure 7. QARM Rules visualization

The green highlighted columns show the antecedents of the selected rule. The green column is the consequent of the rule. The first row after the header (not shown here) shows the range of accepted values for each of the columns that participate in the selected rule. Feature works similar to “filters” in Excel, but with the added constraint, that for all the rows that obey the “filters” of the antecedents and are shown in the screenshot, the consequent value column also obeys the consequent target constraint with whatever confidence the rule specifies. After the results have been achieved – they were extracted and placed into .csv format file, which contains the classification results on the test dataset.

4.4 Neural Networks experiment set-up

Neural Network for the classification purposes is created with the usage of programming language Python and its Tensorflow library (Tensorflow.org, n.d.). Other libraries, such as pandas, sklearn and numpy are also being used mostly for data processing.

As the dataset is a .csv file, it is being read and processed with the pandas.read_csv method. After obtaining dataset in the needed format it was divided into training data (X), which is being used in our Neural Network for creating the classification and is being equal to 84-dimensional feature vector, that was described earlier, and target data (y), on which the Network is focused, as it is the object according classes. Method “iloc” from pandas is used for that. Next step was to do a split of our data into training (90% of the whole dataset) and test data (10%) – it was reproduced by the usage of “train_test_split” from the scikit-learn library. Number of instances for training is shown at Table 3.

Table 3. Amount of values in training dataset, according to each class before data resampling

| Class | Amount of values |
|-------|------------------|
| 0 | 230 |
| 1 | 223 |
| 2 | 1360 |
| 3 | 1337 |

Resampling is done with the “RandomOverSampler” method, that does the oversampling for our data. Main idea of this method is to make our dataset balanced. Amount of values of minority classes is being increased with artificial close values, so that the percentage of each class is becoming more or less the same – in our case exactly 25% from now on each class will represent in terms of the whole data (see table 4).

Table 4. Amount of values in training dataset, according to each class after data resampling

| Class | Amount of values |
|-------|------------------|
| 0 | 1360 |
| 1 | 1360 |
| 2 | 1360 |
| 3 | 1360 |

Sequential model from Tensorflow was used. Good performance was shown with the usage of 4 layers with amount of neurons equal to 42 for an input layer, 21 and 12 for hidden layers and 4 (amount of classes) for output layer accordingly. First three layers were using Rectified Linear Unit (ReLU) activation function, while the output one was with Softmax activation. After creating the Neural Network the model of it was compiled with “adam” optimizer (showed fastest achievement of high training accuracy) and loss function as categorical cross-entropy for multiclass classification.

5 Obtained Results

In this chapter the obtained results after running the algorithms are evaluated and intermediate conclusions are made.

5.1 QARMA performance

The main attributes, that are showing the QARM algorithm performance on fault identification problem are the classification accuracies (table 5) and confusion matrix.

Table 5. Accuracy scores for QARM algorithm

| | |
|-------------------|--------|
| Accuracy | 79.5 % |
| Balanced accuracy | 65 % |

Accuracy shows, that the amount of correctly classified objects is around 80%, while incorrectly classification was made for remaining 20%. It does not sounds bad at first, but now it is possible take a look at the confusion matrix (figure 10) and check the amount of correctly classified objects in accordance with each class. Confusion matrix is shown at Figure 8.

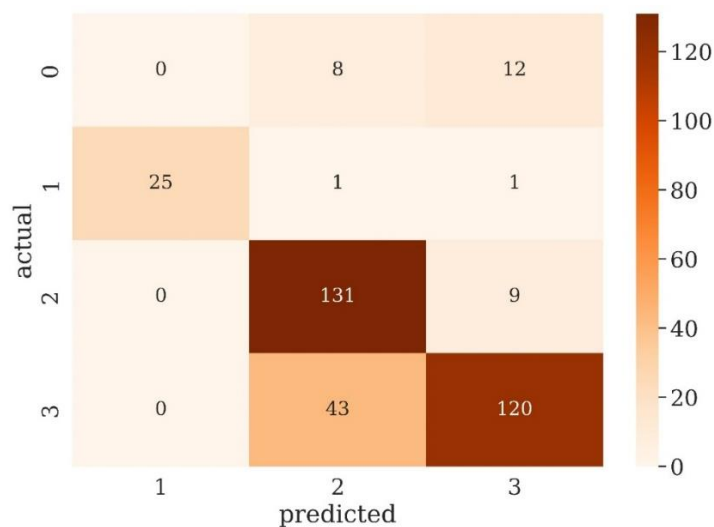


Figure 8. Confusion Matrix for QARM algorithm

As it can be seen from the confusion matrix, the algorithm was not able to classify any object of class 0 (0% accuracy), while for class 1 the amount of correctly classified objects is 25 out of 27, what is 93% accuracy – these class subsets of test data were equally small. For class 2 the amount of correctly classified objects is 131, so that accuracy for obtaining class 2 equals to 94%. And for the class 3 correctly amount of correctly classified points is 120, but 43 instances were assigned to class 2 – accuracy in that case is equal to 74%. This data is arranged into Table 6.

Table 6. Classification results for each class instance with the usage of QARMA

| Class | Amount of instances | Correctly classified | Accuracy |
|-------|---------------------|----------------------|----------|
| 0 | 20 | 0 | 0 % |
| 1 | 27 | 25 | 93 % |
| 2 | 140 | 131 | 94 % |
| 3 | 163 | 120 | 74 % |

In terms of the fast development of different Deep Learning methods with the accuracy up to 100%, the overall accuracy of 80% of the proposed new QARM method is not good at all, but it must be stated out, that some amount of optimization conditions has not been met for running the dataset of diesel engine through it. Also, huge number of modern methods could have very good accuracy, but mostly they were not tested on the “real” datasets, that could be used to different classification problems, such as the analyzed one.

From now it is possible to check the results of the Neural Network and then compare them with the results of QARMA.

5.2 Deep Neural Network Performance

Same indicators were used to evaluate the performance of Neural Network as for QARM algorithm. The accuracies are shown at the Table 7.

Table 7. Accuracies of developed Neural Network

| | |
|-------------------|------|
| Accuracy | 80 % |
| Balanced accuracy | 69 % |

Accuracy of Neural Network shows that 81% amount of data was classified correctly, while remaining 19 % were classified improperly. Balanced accuracy showed that in terms of balanced classification – it is only 69 % accurate. Confusion matrix for Neural Network is shown at the Figure 9.

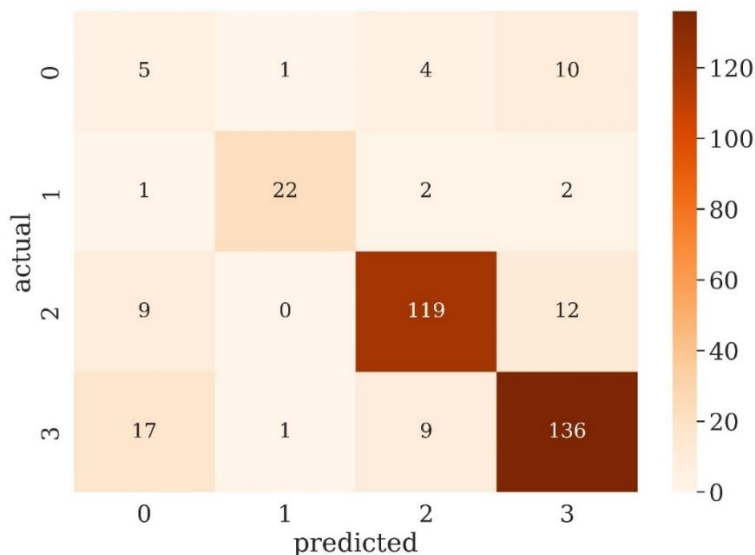


Figure 9. Confusion Matrix for NN classifier

At this time the amount of correctly classified data for class 0 is equal to 5 what is at least something, but in overall it is only 25% accuracy achieved for that class. Class 1 was

classified at 82 % accuracy with 22 correct values out of 27. Classes 2 and 3 have accuracy on around 85 % and 83% accordingly. Data is arranged into Table 8.

Table 8. Classification results for each class instance with the usage of Deep Neural Network

| Class | Amount of instances | Correctly classified | Accuracy |
|-------|---------------------|----------------------|----------|
| 0 | 20 | 5 | 25 % |
| 1 | 27 | 22 | 82 % |
| 2 | 140 | 119 | 85 % |
| 3 | 163 | 136 | 83 % |

Since in previous sub-chapter it was mentioned that nowadays Neural Networks could achieve accuracy on around 95% and accuracy of the proposed neural network is nearly 81% there are ways to explain why it is not as accurate as it should be. First of all, the dataset is imbalanced (the amount of instances of classes 1 and 2 is 250 for each, and for classes 3 and 4 it is 1500 for each) and none of the balancing techniques was used to prevent the overfitting with biased data. Because of that some amount of accuracy is lost. But since it does not requires a lot of time to do with the needed Python libraries, to compare the results with the simplest first realization, one balancing technique was implemented – the oversampling. It is done with the usage of the “RandomOverSampler” method from the imblearn library. The results of balancing the dataset with artificial data to prevent overfitting are shown at the Table 9. Same exact Neural Network was used as before to run the experiment with.

Table 9. Accuracies with dataset balancing for Neural Network

| | |
|-------------------|--------|
| Accuracy | 79.5 % |
| Balanced accuracy | 73 % |

Balanced accuracy has improved on 4 % since the amount of correctly classified results for minority classes is increased. With that overall accuracy slightly decreased – majority classes have less accurate classification results. The confusion matrix is shown at the Figure 10.

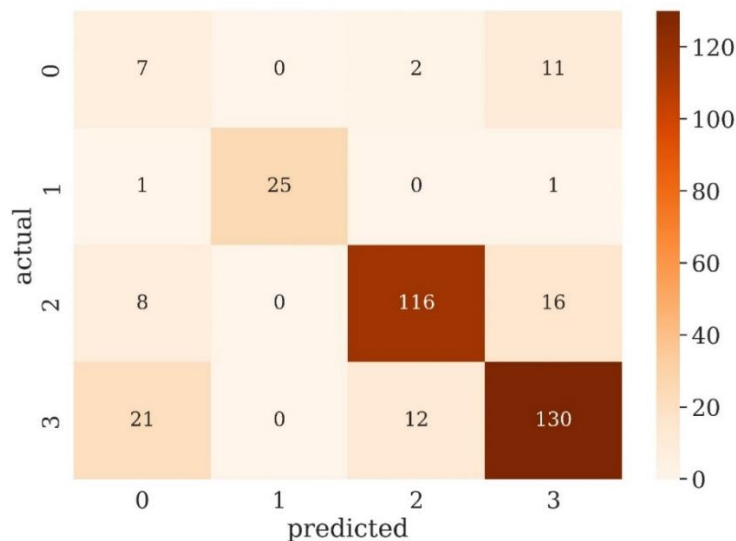


Figure 10. Confusion Matrix for NN classifier with balanced dataset

At Table 10 the data from confusion matrix is arranged and the accuracy for each class prediction is calculated.

Table 10. Classification results for each class instance with the usage of Deep Neural Network and data balancing

| Class | Amount of instances | Correctly classified | Accuracy |
|-------|---------------------|----------------------|----------|
| 0 | 20 | 7 | 35 % |
| 1 | 27 | 25 | 93 % |
| 2 | 140 | 116 | 83 % |
| 3 | 163 | 130 | 80 % |

With data balancing the accuracy of classification of class 0 increased up to 35 % from 25 % and for class 1 it has increased up to 93 % from 82 %. But because of that, the classification accuracy for classes 2 and 3 slightly decreased to 83 % from 85 % and to

80 % from 83 % accordingly. Data balancing did not help to make the overall accuracy better but helped to achieve more accurate results with minority classes classification.

All in all, it is now possible to compare the results of Neural Network with QARMA.

5.3 QARMA vs Neural Network

At first, it is important to point out, that the experimental achieved results are better than the achieved accuracy for the artificial neural network with 0 dB noise level (Pestana-Viana et al., 2019). Comparison is shown at the Table 11.

Table 11. Accuracy results for all methods

| “paper” ANN | QARMA | Deep NN | Deep NN with dataset balancing |
|-------------|--------|---------|--------------------------------|
| 75.7 % | 79.5 % | 80 % | 79.5 % |

It is clear, that both QARMA and Deep Neural Network developed for the experiment are better than the NN method, proposed in the paper of D. Pestana-Viana (2019). The goal to achieve the better experimental results is done. But what to choose for fault identification problem out of these two – QARMA or Deep Neural Network?

QARMA is a new method, that wasn't tested enough, to be used as a main method for hard machine learning tasks. This research work showed, that even since it is a different technique method, than Neural Network for classification problems, it still has nearly the same results as the Deep Neural Network, that was developed in this work to make the comparison. But when the talk is about minority classes – the QARMA algorithms is not the good choice. The data must be balanced – so as for QARMA and so as for Neural Network. And for QARMA especially, since in our experiment it was not able to identify any object of class 0, while the Neural Network was able to identify from 25 % to 35 % of class 0 test data. Nevertheless, the QARM algorithm is an explainable algorithm, since the outcomes of it provide us with explicit internal relations between features, while ANN is not doing that – it is like a black box at some point.

As a conclusion - the most reasonable way to identify faults in diesel engines now is the usage of Neural Networks. But in the long-term, if QARM algorithm will be developing and most of the limitations will be eliminated – because of its “explainability” it is probably going to be a better choice, since the faults in diesel engines must be understandable, so that we could know and identify the factors and the chain of consequences, which caused the fault condition.

5.4 Improvements of Fault Identification

The proposed methods for fault identification in diesel engines can possibly be further improved in different ways. Here are some recommendations.

- a) Starting with the QARMA the points are stated below, that are mostly related to its developer:
 - a.1) Solve the issue with minority classes classification
 - a.2) Create more information papers about the algorithm, FAQ and a website, so that it will be easier to understand and implement this algorithm in practice
 - a.3) Make a user-friendly GUI for ergonomic efficiency of working with the algorithm
- b) Neural Networks possible improvements:
 - b.1) Add more hidden layers to the model, so that it could find dependencies better
 - b.2) Adjust the learning rate by creating a rule for it
 - b.3) Use different Neural Network creation framework
 - b.4) Thoroughly adjust the remaining hyperparameters, according to the probable output

Such recommendations could make these existing methods better or not and that’s why they are called “recommendations”.

Conclusions

A comparative analysis of two different machine learning methods for classification problems was carried out - QARMA and a deep neural network. As a result of the analysis, it turned out that the accuracy of these methods when used on a dataset for diesel engine faults is relatively close and equals 79.5% and 80%, respectively, which is already a better result than was presented in the literature with an accuracy of 75.7% for a neural network (Pestana-Viana et al., 2019).

Recommendations were proposed to improve these methods for working with the dataset under study, such as improving the behaviour of QARMA on minority classes and more professional parameter settings adjustment for the neural network.

As a result, we can say that machine learning algorithms can show decent results when working with data taken from engines, in particular, from diesel engines, which are widely used at least in the automotive industry. Further development of these algorithms will lead to the fact that most of the problems of fault diagnosis will be solved computationally, based on the available data on the current state of the system.

References

Andenko, E., 2021. Natural Language Processing. [Online] Available at: <https://medium.com/nuances-of-programming/обработка-естественного-языка-b1e1cf606929> [Accessed: 12 March 2022].

Agrawal, R., Srikant, R., 1994. Fast Algorithms for Mining Association Rules. VLDB Conference, pp. 487-499.

Agrawal, R., Imielinski, T., Swami, A., 1993. Mining Association Rules between Sets of Items in Large Databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207-216.

Brown, P. F., Della Pietra V. J., Mercer, R. L., Della Pietra, S. A., Lai, C. J. 1992. An Estimate of an Upper Bound for the Entropy of English. Computational Linguistics, March, pp. 31-40.

Bnosac, 2017. Text Analytics. [Online] Available at: <https://www.bnosac.be/index.php/services/data-science-activities/text-mining> [Accessed: 13 March 2022].

Chong, K. P. E., Zak, S. H., 2013. An Introduction to Optimization. Wiley-Interscience, New York, 4th edition.

Chatterjee, S., 2020. What is Recurrent Neural Network | Introduction of Recurrent Neural Network. [Online] Available at: <https://www.mygreatlearning.com/blog/recurrent-neural-network/> [Accessed: 7 March 2022].

Christou, I. T., 2018. A Parallel/Distributed Algorithmic Framework for Mining All Quantitative Association Rules.

Congcong, L., Huaming, W., 2019. Channel pruning based on mean gradient for accelerating Convolutional Neural Networks. *Signal Processing*, March, pp. 84-91.

Duda, R. O., Hart, P. E., and Stork, D. G., 2000. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition.

d2l – Dive into deep learning, 2020. Deep Convolutional Neural Networks (AlexNet). [Online] Available at: https://d2l.ai/chapter_convolutional-modern/alexnet.html [Accessed: 12 March 2022].

DeepAI, 2019. Hyperplane. What is a Hyperplane?. [Online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/hyperplane> [Accessed: 28 February 2022].

Habr, 2019. GPT-2 нейросеть от OpenAI. Быстрый старт. [Online] Available at: <https://habr.com/ru/post/440564/> [Accessed: 15 March 2022].

Galushkin, A. I., 2016. NEURAL NETWORKS. [Online] Available at: https://bigenc.ru/technology_and_technique/text/4114009 [Accessed: 27 January 2022].

Gutierrez-Rojas, D., Christou, I. T., Dantas, D., Narayanan, A., Nardelli, P. H. J., and Yang, Y., 2022. Performance evaluation of machine learning for fault selection in power transmission lines. *Knowledge and information systems* [Online], 64(3), pp. 859–883. Available from: <https://doi.org/10.1007/s10115-022-01657-w>

Hahsler, M., Bettina. G., Hornik, K., 2006. Introduction to arules – Mining Association Rules and Frequent Item Sets. *Journal of Statistical Software*.

Mitchell, T. M., 1997. *Machine Learning*. McGraw-Hill Science/Engineering/Math.

MathIsFun, n.d. Composition of Functions. [Online] Available at: <https://www.mathsisfun.com/sets/functions-composition.html> [Accessed: 28 February 2022].

Nelson, D., 2020. What is an Autoencoder?. [Online] Available at: <https://www.unite.ai/what-is-an-autoencoder/> [Accessed: 7 March 2022].

Pestana-Viana, D., Gutierrez, R.H.R., de Lima, A.A., e Silva, F.L., Vaz, L.F.H., Prego T.M. and Monteiro, U.A. , 2019. Application of Machine Learning in Diesel Engines Fault Identification. Springer Nature Switzerland AG, Tom MMS 61, pp. 74-89.

Pestana, D., 2019. Diesel Engine Faults Features Dataset (3500-Default). [Online] Available at: <https://data.mendeley.com/datasets/k22zxxz29kr/1> [Accessed: 5 April 2022].

Piatetsky-Shapiro, G. 1991. Discovery, Analysis, and Presentation of Strong Rules. Knowledge Discovery in Databases.

Sheidaei, M., 2019. AI is the future of Process Control and Automation. [Online] Available at: <https://pr-cloud.com/aiautomation.html> [Accessed: 3 March 2022].

Tensorflow.org, n.d. Why Tensorflow?. [Online] Available at: <https://www.tensorflow.org/about> [Accessed: 15 April 2022].

TextileValueChain, n.d. Artificial Neural Network - 2. [Online] Available at: <https://textilevaluechain.in/in-depth-analysis/articles/textile-articles/artificial-neural-network-2/> [Accessed: 28 February 2022].

Weng, L., 2020. Neural Architecture Search. [Online] Available at: <https://lilianweng.github.io/posts/2020-08-06-nas/> [Accessed: 17 March 2022].

Wang, L., Yang, Y., Min, M.R. and Chakradhar, S.T., 2017. Accelerating deep neural network training with inconsistent stochastic gradient descent. Neural Networks, 16 June, pp. 219-229.

Zhang, W., Itoh, K., Tanida, J. and Ichioka, Y. 1990. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. Applied Optics, 29 32, 4790-7, 10 November, pp. 110-123.

Appendix A. Python code for creation and analysis of Deep Neural Network and post-processing of QARM algorithm

```
# import of libraries and its elements
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sn
import matplotlib.pyplot as plt
from tensorflow import keras
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.metrics import accuracy_score,
balanced_accuracy_score
from tensorflow.keras import layers

# load .csv dataset into variable
diesel_engine = pd.read_csv(
    "drive/MyDrive/Master's Thesis/diesel_engine_1.csv",
    header = 0
)

# obtaining the feature vector X and target data y
X = diesel_engine.iloc[:, :84]
y = diesel_engine.iloc[:, 97]

# rescaling the data into input range from 0 to 1
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# applying one-hot encoding to the class column and use
# the result as the desired output
encoder = OneHotEncoder(sparse=False)
Y = encoder.fit_transform(df[['CLASS']])

# splitting the dataset into training and test set in 90 % and
10 % accordingly
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.1, stratify=df['CLASS']
)

# oversampling performed to avoid overfitting on the majority
classes
ros = RandomOverSampler(random_state=42)
X_train_res, y_train_res = ros.fit_resample(X_train, y_train)
```

```

# neural network model creation. Input is the training feature
# vector, consisting of 84 dataset columns. It has 2 hidden lay-
# ers # with ReLU activation and the output layer has softmax
# activation, which is going to give us the probabilities of
# classification
def create_model():
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(42, input_shape=(84,) , acti-
vation = 'relu'))
    model.add(tf.keras.layers.Dense(21, activation = 'relu'))
    model.add(tf.keras.layers.Dense(12, activation = 'relu'))
    model.add(tf.keras.layers.Dense(4, activation = 'softmax'))

    model.compile(loss = 'categorical_crossentropy',
optimizer = 'Adam', metrics = ['accuracy'])
    return model

# training the model
model = create_model()
#model.fit(X_train, y_train, epochs = 256)
model.fit(X_train_res, y_train_res, epochs = 256)

# having trained the model now it is possible to verify whether
it # generalizes correctly. We run the model on the testing data
y_pred = model.predict(X_test)

# we transform the probabilities back to class labels
y_pred = encoder.inverse_transform(y_pred)[: , 0]
y_true = encoder.inverse_transform(y_test)[: , 0]

# displaying the accuracies, based on test data
NN_accuracy = accuracy_score(y_true, y_pred)
NN_balanced_accuracy = balanced_accuracy_score(y_true, y_pred)
print("Accuracy: ", NN_accuracy)
print("Balanced Accuracy: ", NN_balanced_accuracy)

# displaying the confusion matrix in a simple manner
confusion_matrix = pd.crosstab(y_true, y_pred, rownames=['actu-
al'], colnames=['predicted'])
print(confusion_matrix)

# printing the confusion matrix in a nice looking way
csfont = {'fontname':'Times New Roman'}
plt.rcParams["font.family"] = "Times New Roman"
sn.set(font_scale=1.7)
plt.rcParams.update({'font.size': 16})
sn.set_style({'font.family':'serif','font.ser-
if':'Times New Roman'})
plt.figure(figsize = (10,7))
sn.heatmap(confusion_matrix, cmap='Oranges', an-
not=True, fmt='.0f')
plt.save-
fig('fault_class_NN.jpg', bbox_inches='tight', pad_inches=0.1, d
pi=600)

```

```

plt.show()
# loading QARMA results
QARMA_res = pd.read_csv(
    "drive/MyDrive/Master's Thesis/results_by_QARMA_84.csv",
    header = None
)

# making results as np.array class labels
QARMA = QARMA_res.to_numpy()
QARMA
res = []
for x in QARMA:
    for y in x:
        res.append(y)
res = np.array(res)
res

# assigning variable to evaluate the QARMA performance
y_true_for_qarma = y_test

# computing and printing QARMA accuracies
print("Accuracy: ", QARMA_accuracy)
print("Balanced Accuracy: ", QARMA_balanced_accuracy)

# confusion matrix creation
confusion_matrix_QARMA = pd.crosstab(y_true_for_qarma, res, rownames=['actual'], colnames=['pre-
dicted'])
print(confusion_matrix_QARMA)

# displaying the confusion matrix for QARMA
csfont = {'fontname':'Times New Roman'}
plt.rcParams["font.family"] = "Times New Roman"
sn.set(font_scale=1.7)
plt.rcParams.update({'font.size': 16})
sn.set_style({'font.family':'serif','font.ser-
if':'Times New Roman'})
plt.figure(figsize = (10,7))
sn.heatmap(confusion_matrix_QARMA, cmap='Oranges', an-
not=True, fmt='.0f')
plt.save-
fig('fault_class_QARMA.jpg', bbox_inches='tight', pad_inches=0.1
, dpi=600)
plt.show()

```