Supervised Methods for Fault Detection in Vehicles

Master’s Thesis in Electrical Engineering

Gao Xiang
Jiang Nan
Supervised Methods for Fault Detection in Vehicles

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

May 2010
Acknowledgement

The project and thesis was supervised by Stefan Byttner. During our research, he helped us with developing research ideas and improving our results every single week. We are very grateful to him for the help he gave us. We also want to thank our parents who gave us this opportunity to study in Sweden where we have a good learning environment to learn more advanced knowledge and technology.
Abstract

Uptime and maintenance planning are important issues for vehicle operators (e.g. operators of bus fleets). Unplanned downtime can cause a bus operator to be fined if the vehicle is not on time.

Supervised classification methods for detecting faults in vehicles are compared in this thesis. Data has been collected by a vehicle manufacturer including three kinds of faulty states in vehicles (i.e. charge air cooler leakage, radiator and air filter clogging). The problem consists of differentiating between the normal data and the three different categories of faulty data. Evaluated methods include linear model, neural networks model, 1-nearest neighbor and random forest model. For every kind of model, a variable selection method should be used. In our thesis we try to find the best model for this problem, and also select the most important input signals. After we compare these four models, we found that the best accuracy (96.9% correct classifications) was achieved with the random forest model.

Keywords:

Fault Detection, Supervised Methods, Machine Learning
1. Introduction

1.1 Problem Statement

Unplanned downtime for a bus can be very costly for a commercial operator (i.e. a bus company). It is of benefit to be able to detect faults before they become too serious. There are two methods in machine learning to approach the fault detection problem: the supervised method and the unsupervised method. The difference between supervised and unsupervised learning is that with supervised methods we have access to labeled data (categories of data are known) while with unsupervised methods, the categories are unknown. Experiments with unsupervised methods have shown that faults such as air filter clogging are difficult to detect. This thesis intends to investigate whether supervised methods can detect such faults. The purpose of our project and thesis is to use supervised method to detect the faults (charge air cooler leakage, radiator and air filter clogging) in vehicles.

At Halmstad University, there is a research project (Data-Driven Modeling, DDM) dealing with fault detection for improved up-time for vehicles. In this project there is a large amount of data collected from a real bus that has been used by different drivers and driving conditions. A few different faults have been introduced into a test-bus, such as a charge air cooler leakage, radiator filter clogging and air filter clogging. These faults represent issues that could cause downtime for a city bus, and it is therefore beneficial to be able to detect when these faults have occurred.

In our project, we have four different kinds of data collected from the car company's laboratory. One set of data is normal data, and the other three are fault data (charge air cooler leakage, radiator and air filter clogging). Also for every set of data, we have 15 signals, so we want to use these 15 input signals to detect the faults or the normal situation.

The faults in the data that have been collected (charge air cooler leakage, radiator and air filter clogging) are faults where there is currently no diagnostic function available to detect them.

1.2 Project Goal

The goal of this thesis project is to evaluate if supervised methods can detect (and if
so, how well) different faults (charge air cooler leakage, radiator and air filter clogging). We also want to find which is the best model for detecting faults in our project and determine which are the most important signals for each model.

1.3 Project Scope

There are some limitations related to the data in the project. All the signals have had their labels removed and a transformation has been performed on each signal to make them difficult to identify. This was a request from the company which has supplied the data. This limitation means, for example, that there is no physical knowledge that can be applied when doing variable (signal) selection for the models, and the data we use is all collected in buses, not all different kinds of vehicles. In our project, we also limit the scope to testing linear regression, k-nearest neighbors, neural networks and random forest.
2. Earlier Research

In [KH], they conducted research about early warning fault detection. In their research, they use artificial intelligence methods to detect faults. They use Multi-Layer Feed forward (MLF) network as the network architecture in thesis project. The MLF network is a network of neurons and synapses organized in the form of layers. Their research showed us how to use the neural networks model to detect the faults.

In [MJ], they conducted research about fault detection and isolation (FDI) in dynamic data from an automotive engine air path using artificial neural networks. In their case, several faults are considered, including leakage, EGR valve and sensor faults, with different fault intensities. RBF neural networks are trained to detect and diagnose the faults, and also to indicate fault size, by recognizing the different fault patterns occurring in the dynamic data. Three dynamic cases of fault occurrence are considered with increasing generality of engine operation. The approach is shown to be successful in each case.

In [NW], they researched about on-line fault detection for bus-based field programmable gate arrays. In their article, they introduce an online technology built self-testing (BIST) of bus-based field-programmable gate arrays (FPGA’s). The system detects the desired function from a deviation of FPGA without using special hardware, hardware peripherals, and system operation without interruption. It is about the fault detection for programmable gate, but it can also give us some good ideas about fault detection in vehicles.

In [G], Gancho Vachkov researched about intelligent data analysis. He proposes an efficient computational strategy for remote performance analysis and diagnosis of construction machines and other complex systems. A special information compression (IC) method is used to send the information obtained from various sensors to the maintenance center in a compact and economical way.

In [R1], R.Isermann published article entitled ‘Supervision, Fault-Detection and Fault-Diagnosis methods-An introduction’ in 1997. The operation of technical processes requires increasingly advanced supervision and fault diagnosis to improve reliability, safety and economy. In his paper, he gives an
introduction to the field of fault detection and diagnosis. Then different methods of fault detection are considered which extract features from measured signals and use process and signal models. These methods are based on parameter estimation, state estimation and parity equations.

In [GF], they present a method to increase the reliability of unmanned aerial vehicle (UAV) sensor Fault Detection and Identification (FDI) in a multi-UAV context. In their article, reliability is a key issue in aerial vehicles, where Fault Detection and Identification (FDI) techniques play an important role in the efforts to increase the reliability of the system. Most FDI applications to UAVs that appear in the literature use model-based methods which try to diagnose faults using the redundancy of some mathematical description of the system dynamics.

In [SA], they present a method based on continuous wavelet transform to detect the faults of vehicle suspension systems. They used a full vehicle dynamic model which had been simulated in ADAMS/CAR and validated it by laboratory test results. In their paper, the incapability of the spectral analysis by using fast Fourier transform in the analysis of the signals is revealed through applying the inputs that include transient characteristics and then wavelet transform was employed to achieve more proper results.

In [ST], they have done the research for networked vehicles for automated fault detection with unsupervised methods. Unsupervised methods (COSMO) have previously been tested for this kind of problem, but resulted in low accuracy (only air filter clogging reliably detected). It would be interesting to see if supervised methods can classify these faults.
3. Data Sets

In this part, we will introduce the data we use, and describe how we use it.

3.1 Data Description

All data we use are collected by a bus company within about six months. There are four sets of data (normal data, charge air cooler leakage data, radiator filter clogging data and air filter clogging data). Every set of data used the same sampling frequency, 1 Hz. Then, for different fault and normal set, the laboratory of the bus company collected different amount of data, which we can see in Table 3.1:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Total time(hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amat</td>
<td>Air fault data</td>
<td>18.7</td>
</tr>
<tr>
<td>Cmat</td>
<td>Cooler fault data</td>
<td>23.7</td>
</tr>
<tr>
<td>Rmat</td>
<td>Radiator fault data</td>
<td>20.0</td>
</tr>
<tr>
<td>Nmat</td>
<td>Normal data</td>
<td>42.2</td>
</tr>
</tbody>
</table>

Table 3.1: Measurement duration for the different vehicle states; it shows how long the signals are collected for each kind of data.

All the signals have had their labels removed and a transformation has been performed on each signal to make them difficult to identify. This was a request from the company which supplied the data.

Measurements have been made under varying conditions; the data from each driving run (stored in a matrix) have been made with different drivers and ambient conditions. We can see the detailed data structure in Table 3.2:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Number of Matrices</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amat</td>
<td>Air filter clogging fault data</td>
<td>10</td>
<td>$15 \times 67346$</td>
</tr>
<tr>
<td>Cmat</td>
<td>Charge air cooler fault data</td>
<td>13</td>
<td>$15 \times 85187$</td>
</tr>
<tr>
<td>Rmat</td>
<td>Radiator filter clogging fault data</td>
<td>12</td>
<td>$15 \times 71897$</td>
</tr>
<tr>
<td>Nmat</td>
<td>Normal data</td>
<td>22</td>
<td>$15 \times 151861$</td>
</tr>
</tbody>
</table>

Table 3.2: Size of data; it shows the number of observations when measuring with 1Hz sampling frequency.
3.2 Data Preparation

Before using different models to test our result, we prepared our data first. The work to prepare the data can be mainly divided as following:

- Selecting the data
- Normalizing the data
- Dividing the data

3.2.1 Selecting the data

The reason we select the size of the data is that we can use the same size of each kind of data when train and test the data. Then we find the minimum of the four sets. It is air filter fault data and the size of it is $15 \times 67346$, so we should get the same size of other three kinds of data. However, which part to select is a question for us to address.

Because each matrix has different feature, when we use some matrices to train and other matrix to test, it will give a low accuracy.

Here the simplest and most effective way to solve this problem is to randomize the data before we use it.

So we randomize all data and then take the first 67346 samples from each class. We can see it in Figure 3.1:

![Figure 3.1: Size of data we use to train and test; we use the same size of each kind of data.](image)

Then we combine four different matrices into one big matrix. The size of it becomes $15 \times 269384$.  

6
3.2.2 Normalizing the data

We want to put all signals on the same scale, so we normalize the data before we divide it.
We can use the formula as follows to deal with our data:

\[ x_n = \frac{z_n - \mu_n}{\sigma_n} \]  

(3.1)

Where \( x_n \) is the n-th normalized component, \( z_n \) is the n-th component in the original signal space, \( \mu_n \) and \( \sigma_n \) is the mean and standard deviation, respectively, for the n-th component in the original signal space.

3.2.3 Dividing the data

After we normalized the data, then we should divide the data for training and testing.
In the linear model, k-nn model and random forest model, we divide the data 66% for training and 34% for testing.
In the neural networks model, we use 33% for training data, 33% for validation data and 34% for testing and the validation data is also a part of training data. The data have been divided allows a comparison between the different models. In each kind of data, we use the same way to divide the data. It is benefit for us to test the performance of the each kind of model with the same condition.
4. Methods

In our project, the emphasis in this thesis is on methods from the machine learning field. Here we do a brief introduction to machine learning.

Machine learning is a sub-field of artificial intelligence; the main focus is put on building models that can automatically “learn” e.g. classification or regression tasks. Machine learning has a very wide range of applications such as biometric identification, search engines, medical diagnosis, detection of credit card fraud, stock market analysis, DNA sequencing, speech and handwriting recognition, computer vision, strategy games, and so on.

In our project, we use machine learning for classification tasks.

In machine learning, supervised learning means that it has a teacher to sign the label of the class. It is often posed as a function approximation problem. In our case, we build and train the model to make the output we get close to the target.

In the following sections we will describe the principles of various models that are used in the project.

4.1 Linear Model

At first, we want to try linear model for our problem and evaluate how well it performs.

Linear regression is an approach to modeling the relationship between a between the target $y$ and the input $X$. Also with linear model, we can use least-square solution to solve the problem of building the model.

$$ y = w \cdot X + e $$

Least square means that the overall solution minimizes the sum of the squares of the errors made in solving the equation. In this equation, $y$ is the target value, and $X$ is our input matrix, $e$ is the error vector, $w$ is the weight vector of every input. We can use least-square solution to get the weight of each input $X$, then we can use the following formula to calculate the weight vector value $w$.

$$ w = (X^T \cdot X)^{-1} \cdot X^T \cdot y $$

In our case, we should use the training data to calculate the weight for our model.
Then the $X$ should be a $D \times N$ matrix where $D$ is the number of signals and $N$ is the number of observations. $X^T$ is the transpose matrix of $X$.

We have four kinds of data sets, so we should use four linear models for each kind of data; also we make the target 1 when the input belongs to the model otherwise we make it be 0. The target can be found in the Table 4.1:

<table>
<thead>
<tr>
<th>Data belongs \ Model</th>
<th>AFC model</th>
<th>CAC model</th>
<th>RFC model</th>
<th>Normal model</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CAC</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RFC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 4.1**: Target value for linear model; AFC means air filter clogging fault, CAC means charge air cooler leakage fault, RFC means radiator filter clogging fault. When the data belongs to the class, we set the target to be 1, otherwise to be 0.

After we set up the target value, we can train our linear model and calculate the weight of each model and we can build four linear models for our project. We can see it in Figure 4.1:

**Figure 4.1**: The linear model; we use the training data and target value to calculate the weight of the linear model.

Then we can use our testing data and linear model to get the output, and we can get
4 output values from 4 linear models. We can use the maximum value for our linear model’s result, the target value is put to one if the model is trained on data corresponding to its fault, and otherwise it is zero.

In the linear model and other models we need to use confidence interval. We use 10 sets of random data to test the confidence interval. Here we use 95% for confidence interval in normal distribution.

\[
\mu \pm \frac{1.96 \cdot \sigma}{\sqrt{n}}
\]  

(4.3)

Here \( \mu \) is the average accuracy for the datasets, and \( \sigma \) is the standard deviation, \( n \) is the number of observations (of accuracies) used for computing the average value and standard deviation.

### 4.2 Neural Network Model

Neural networks provide a unique computing architecture that is used to address problems that are intractable or cumbersome with traditional methods. Neural networks are nonlinear mapping systems whose structure is loosely based on principles observed in the nervous systems of humans and animals. These new computing architectures are radically different from the computers that are widely used today. Networks are massively parallel systems that rely on dense arrangements of interconnections and surprisingly simple processors.

In general terms, an artificial neural network consists of a large number of simple processors linked by weighted connections. The processing units may be called neurons. Each unit receives inputs from many other nodes and generates a single scalar output that depends only on locally available information, either stored internally or arriving via the weighted connections.

In general, the processing units have responses like:

\[
y = f(\sum_k (w_k x_k))
\]  

(4.4)

Where \( x_k \) are the output signals of other nodes or external system inputs, \( w_k \) are the weights vector of the connecting links and \( f(\cdot) \) is a simple nonlinear function. Here, the unit computes a weighted linear combination of its inputs and passes this through the nonlinearity \( f \) to produce a scalar output.

In our case, we use one hidden layer, make the inputs signals into a large matrix of a
size of $15 \times 269384$. Then we train the neural networks and the target is almost like the linear model, but we use only one model and the target is a $4 \times 1$ matrix. We can see Table 4.2 as follows:

<table>
<thead>
<tr>
<th>Training data belong</th>
<th>AFC data</th>
<th>CAC data</th>
<th>RFC data</th>
<th>Normal data</th>
</tr>
</thead>
</table>
| Target vector        | \[
\begin{bmatrix}1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\] | \[
\begin{bmatrix}0 \\ 1 \\ 0 \\ 0 \end{bmatrix}
\] | \[
\begin{bmatrix}0 \\ 0 \\ 1 \\ 0 \end{bmatrix}
\] | \[
\begin{bmatrix}0 \\ 0 \\ 0 \\ 1 \end{bmatrix}
\] |

Table 4.2: Target vector for NN model (orthogonal coding); one position in the vector is set to 1 depending on which class the data belongs to, otherwise 0.

Then we can use the input data and the target to build our neural networks model. After we build the model we can test the output through the model. The output we get should be a $4 \times 1$ matrix, and the value should be between zero and one. We can choose the maximum value from the four vectors in the matrix, and use the category of the vector for the output of the model.

### 4.3 K-NN Model

In pattern recognition, the k-nearest neighbor’s algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by the majority of votes of its neighbors’; with the object being assigned to the most common class amongst its k nearest neighbors (k is a positive integer, typically small). We can see that in Figure 4.2:

![Figure 4.2: Example of k-NN models; the left one is 1-NN model, the right one is 3-NN model.](image)

In Figure 4.2, we have 2 classes, one is circle, and the other is triangle. When we choose 1-nearest neighbor model, we find the triangle one is the closest one to the
new input, and we can insert the new one into triangle ones’ class (left side in Figure 4.2). If we choose 3-nearest neighbor model, we should choose three closest distances from the new input, then we find there are two circles and only one triangle, and then we can think the new input falls in the circle class (right side in Figure 4.2). So there will be different results with different k-values.

As far as the procedure to use the data is concerned, we can use all the data in database, and then choose different number of nearest neighbor model. However, in our project, we have a large number of data in our database. If we use all the data to test our result, it will take a long time to calculate every value. So here we use one vector for each kind of data set and the vector is average vector for each category. We get an input vector and calculate the distance from this point to these four average vectors, and choose the closest for our result.

4.4 Random Forest Model

Random forest is an ensemble classifier that consists of many decision trees. It has shown good performance in many areas, so we want to know if it can give us good performance in our project. Random Forests grows many classification trees. To classify a new object from an input vector, we put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree votes for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

![Decision Tree](image)

**Figure 4.3:** Example of a decision tree; a random forest model consists of many trees.

We can also see in Figure 4.3, here \( x_1, x_2, \ldots \) are input signals, and \( a_1, a_2, \ldots \) are some value we use to classify the different kinds of data. And the figure here is only the example for a decision tree, not our model, our model has 50 trees.

In the random forest model, we can calculate the variable importance. That can tell
us which input signals have a great effect on the result. The importance score is calculated based on how much the forest is affected by randomly permuting a variable. That means we can get a performance with all 15 input signals, then we can add some noisy signals to input signal no.1 and get another performance using 14 original input signals and signal no.1 with noise. Then we can calculate the difference between them and get the variable importance of input signal no.1. Finally, we can get the variable importance of other input signals in the same way, and then we can decide the importance of each input signal.
5. Result and Discussion

In this part, we will introduce our result with different models and discuss the performance.

5.1 Linear Model

In the linear model, we have tried two different ways to divide the data. As it is described in the data set part, we use 66% for training and 34% for testing data. Now we have fifteen input signals, and with different combinations of input signals, we can get different models. We try to find the signals that are relevant for solving the classification problem.

We can try different numbers of input signals, if we only use one input signal for our linear model, there will be fifteen models and if we use two input signals from fifteen signals, we can get \( \binom{15}{2} = 105 \) models. We also try to choose three and four signals from fifteen input signals and we can get the accuracy in Figure 5.1:

![Figure 5.1](image)

**Figure 5.1**: Accuracy in the linear model; every part is divided by the thick dotted line, the first part is accuracy with 1 input signal, the second part is with 2 input signals, the part is with 3 input signals, the last part is with 4 signals.
The total number of model combinations with an exhaustive search for important input signals can be large. Therefore we chose to use the forward selection technique for finding the most important input signals and we can see it in Table 5.1:

<table>
<thead>
<tr>
<th>Average correct rate of different numbers of inputs</th>
<th>1 (15 models)</th>
<th>2 (105 models)</th>
<th>3 (455 models)</th>
<th>4 (1365 models)</th>
<th>15 (1 model)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27.4%</td>
<td>30.1%</td>
<td>31.8%</td>
<td>33.2%</td>
<td>42.8%</td>
</tr>
</tbody>
</table>

**Table 5.1:** Average accuracy of different numbers of input signals

We find the number of model will be huge if we use this way to test different models, so we decide to use forward selection. We use five input signals together with highest accuracy when using only one input signal and then we add another input signal from the other 10 input signals, test their accuracy, choose the best one, and continue add one input signal from the other 9 input signals and repeat the same procedure as before until all input signals are added. Then we can get the result in Figure 5.2:

**Figure 5.2:** Forward selection for the linear model; the first part is the accuracy with only one input signal; in the second part, we use 5 signals which give us higher accuracy in the first part; the rest part we add one other signal based on the input signals which give us highest accuracy in the previous every time until we add all 15 signals.

We can find the best performance in this way and it is model no.66. The accuracy is 42.7%. The input signals are [1 2 3 4 5 6 7 8 10 11 12 13 15]. However, we can use the
input signals [3 4 5 7 8 10 11 13 15] for our best model, the performance of it is 42.5%. In the following discussion part we will describe the reason we choose it for the best linear model.

The confusion matrix for this model is shown in the Table 5.2:

<table>
<thead>
<tr>
<th>Tar</th>
<th>AFC</th>
<th>CAC</th>
<th>RFC</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>76.0%</td>
<td>16.4%</td>
<td>3.8%</td>
<td>3.9%</td>
</tr>
<tr>
<td>CAC</td>
<td>22.0%</td>
<td>72.3%</td>
<td>3.0%</td>
<td>2.7%</td>
</tr>
<tr>
<td>RFC</td>
<td>60.4%</td>
<td>24.4%</td>
<td>9.0%</td>
<td>6.2%</td>
</tr>
<tr>
<td>Normal</td>
<td>41.6%</td>
<td>36.4%</td>
<td>8.3%</td>
<td>13.6%</td>
</tr>
</tbody>
</table>

Table 5.2: Confusion matrix table for the linear model

**Discussion:**

We can find that we can get the accuracy about 42% with linear model in our project, so the linear model does not have sufficient accuracy.

We use 10 sets of data with all 15 input signals to see the accuracy. We also calculate the confidence interval to see if the model gives us sufficient accuracy.

We also calculate the confidence interval for linear model is almost 0.25%, then we find that when we add input signal to 10, 11, 12, 13, 14, 15, the accuracy does not increase much. Nine input signal can be sufficient to get the same accuracy.

Through Table 5.1, we find that for air filter clogging fault and charge air cooler leakage data, we can get a high accuracy by this model, but very low accuracy for radiator filter clogging data and the normal data.

Confidence interval of linear models is about ±0.25%, then the best performance with least input signals is model no.46, in which the input signals are [3 4 5 7 8 10 11 13 15]. The accuracy is 42.5%, and it has only used 9 input signals.

### 5.2 Neural Networks Model

In our neural networks model, we use one hidden layer in the network, but we do not know how many hidden nodes are sufficient for our model in this case. So at first, we test different numbers of hidden nodes. We calculate the confidence interval for
different numbers of hidden nodes and we can see the result in Figure 5.3:

![Figure 5.3](image)

**Figure 5.3:** Accuracy with different number of hidden nodes in the NN model; here we use all 15 input signals to test it and the length in y-axis for every point means the confidence interval for the point.

We can see in the Figure 5.3, when the number of hidden nodes is 8, 9 or 10; we almost get the same performance, and not much improvement. Here 10 hidden nodes are sufficient for our problem, so we decide to use 10 hidden nodes for our research.

Then we use the forward selection to improve the performance, and we get the result shown in Figure 5.4:

![Figure 5.4](image)

**Figure 5.4:** Forward selection for the NN model; the same way to select input signals as the linear model.

We can find the best performance in this way it is model no.62. The accuracy is
77.68%. The input signals are [1 3 4 5 6 7 9 10 11 13 14 15]. Considering of the confidence interval, we use the input signals [6 7 10 11 13 14 15] for our best neural networks model. Its accuracy is 76.60%. Then we can get the confusion matrix in Table 5.3:

<table>
<thead>
<tr>
<th>Target</th>
<th>AFC</th>
<th>CAC</th>
<th>RFC</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>72.4%</td>
<td>4.7%</td>
<td>14.0%</td>
<td>8.9%</td>
</tr>
<tr>
<td>CAC</td>
<td>9.2%</td>
<td>87.1%</td>
<td>2.7%</td>
<td>1.1%</td>
</tr>
<tr>
<td>RFC</td>
<td>17.4%</td>
<td>6.7%</td>
<td>67.5%</td>
<td>8.5%</td>
</tr>
<tr>
<td>Normal</td>
<td>4.4%</td>
<td>2.7%</td>
<td>9.4%</td>
<td>83.5%</td>
</tr>
</tbody>
</table>

**Table 5.3:** Confusion matrix for the NN model

We calculate the confidence interval and it is almost 0.5%. And the part in cycle almost get the same performance as the point which the arrow points since their accuracy are in the confidence interval of this point. Then we choose the top 5 most accurate neural network models and check correspondingly what the input signals are. We can see it in Table 5.3:

<table>
<thead>
<tr>
<th>Index of the model</th>
<th>Accuracy</th>
<th>Input signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>77.6%</td>
<td>[4, 6, 7, 9, 10, 11, 13, 14, 15]</td>
</tr>
<tr>
<td>2</td>
<td>77.4%</td>
<td>[4, 5, 6, 7, 9, 10, 11, 13, 14, 15]</td>
</tr>
<tr>
<td>3</td>
<td>77.7%</td>
<td>[1, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15]</td>
</tr>
<tr>
<td>4</td>
<td>77.2%</td>
<td>[3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15]</td>
</tr>
<tr>
<td>5</td>
<td>77.1%</td>
<td>[1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15]</td>
</tr>
</tbody>
</table>

**Table 5.4:** Best five performance of neural networks

Thereafter we use 10 different set of random data and we can calculate the average accuracy and confidence intervals for top five neural networks models. The results can be seen in Figure 5.5:
Figure 5.5: Best 5 performance with confidence interval in the NN model; the length in y-axis for every point means the confidence interval for the point.

We find the best model is no.2, the input signals are [6 7 10 11 13 14 15] and the accuracy is 76.60%.

Discussion:
In addition to model no.1, for each model we almost get the same performance. That means when we use different data, the model with the best performance also changes. In some scenarios, fewer input signals can provide the same level of efficiency, and then we can choose that model, so we also can choose model no.2 for our best neural networks model, because it needs fewer input signals than others and achieve the same performance.

5.3 K-NN Model

Next, we try to use the K-NN model to test the data. We use the training data to calculate one average vector point for each kind of different categories and we can get four average vector points for the four categories.

With these points, we can calculate the distances to the four points with testing data. That means every time we get a testing data, we can calculate four distances between the testing data and four categories vectors. Then we choose the shortest one for our result. Using the same approach as linear and NN models’, we begin the procedure with only one input signal and then can see the result in Figure 5.6:
We can choose five input signals with high accuracy to continue our forward selection. The input signals should be $[4 \ 7 \ 10 \ 11 \ 13]$, then we add another input signal to improve our performance. However, we add input signals from six to seven, we found that the performance does not improve. Then we try to test all fifteen input signals for our 1-nn model and get a lower accuracy, so we stop it at six input signals, and accept it is the best model for 1-nn model. This gives us the result in Figure 5.7:

Figure 5.7: Forward selection for the 1-NN model; every part is divided by the thick dotted line, the first part use 5 input signals, the second part use 6 input signals, the third part use 7 input signals, the last part use all 15 input signals.

In Figure 5.7, the first part is the accuracy when we use 5 input signals, which give us higher accuracy while we only one of them (we call it top 5 accuracy input signals in the rest part of thesis). In the second part, we add one other signal, meaning there
are 6 input signals all together. In the third part, we use 7 inputs and the last one stands for using all 15 input signals.

Use this way, the best performance we can get is 48.4%, and the input signals are [1 4 7 10 11 13].

In our project, the performance of 1-nn is not so good and we can see the confusion matrix in Table 5.5:

<table>
<thead>
<tr>
<th>Tar\Out</th>
<th>AFC</th>
<th>CAC</th>
<th>RFC</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>47.5%</td>
<td>7.6%</td>
<td>22.2%</td>
<td>22.7%</td>
</tr>
<tr>
<td>CAC</td>
<td>26.7%</td>
<td>22.7%</td>
<td>15.1%</td>
<td>35.5%</td>
</tr>
<tr>
<td>RFC</td>
<td>13.0%</td>
<td>7.3%</td>
<td>57.8%</td>
<td>21.9%</td>
</tr>
<tr>
<td>Normal</td>
<td>15.6%</td>
<td>6.9%</td>
<td>11.9%</td>
<td>65.5%</td>
</tr>
</tbody>
</table>

Table 5.5: Confusion matrix for the 1-NN model

Discussion:
We have a large number of data in our database. If we use all the data to test our result, it will take a long time to calculate every value, but it may give us a little higher accuracy. We may also get a better performance if we use bigger number of k-value.

5.4 Random Forest Model

In random forest model, we calculate the variable importance; here is the importance of the input signal in Figure 5.8:

Figure 5.8: Variable importance in the random forest model
Next, we use the program to calculate the accuracy for each model. Same as before, we use the training data to build our random forest and then use the random forest model to test the accuracy of the testing data.

We can see that with this model we have a very high accuracy (here is 95.1%) and we can get the confusion matrix in Table 5.6:

<table>
<thead>
<tr>
<th>Tar\Out</th>
<th>AFC</th>
<th>CAC</th>
<th>RFC</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>97.7%</td>
<td>1.3%</td>
<td>0.1%</td>
<td>0.9%</td>
</tr>
<tr>
<td>CAC</td>
<td>3.3%</td>
<td>92.4%</td>
<td>0.7%</td>
<td>3.6%</td>
</tr>
<tr>
<td>RFC</td>
<td>0.3%</td>
<td>0.9%</td>
<td>97.8%</td>
<td>1.1%</td>
</tr>
<tr>
<td>Normal</td>
<td>2.9%</td>
<td>3.8%</td>
<td>0.7%</td>
<td>92.6%</td>
</tr>
</tbody>
</table>

Table 5.6: Confusion matrix for the random forest model (with 15 input signals)

Because we know the importance of our input signals, we use only these four input signals ([6 7 11 13]) and then we can find the accuracy is very high (96.9%), and the confusion matrix in Table 5.7:

<table>
<thead>
<tr>
<th>Tar\Out</th>
<th>AFC</th>
<th>CAC</th>
<th>RFC</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFC</td>
<td>98.3%</td>
<td>0.8%</td>
<td>0.1%</td>
<td>0.8%</td>
</tr>
<tr>
<td>CAC</td>
<td>1.1%</td>
<td>96.0%</td>
<td>0.4%</td>
<td>2.5%</td>
</tr>
<tr>
<td>RFC</td>
<td>0.1%</td>
<td>0.4%</td>
<td>99.2%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Normal</td>
<td>1.3%</td>
<td>4.1%</td>
<td>0.5%</td>
<td>94.2%</td>
</tr>
</tbody>
</table>

Table 5.7: Confusion matrix for the random forest model (with 4 input signals)

So we can use less input signals to get a higher accuracy and then it will be our best random forest model. The accuracy is 96.9%; the input signals are [6 7 11 13].

We could have got very high accuracy with this model since we have already built a lot of trees, we choose 50 trees. It will give a good performance, but it takes up more memory resources, so we try fewer trees. Here we try different numbers of trees and get the result in Figure 5.9:
Figure 5.9: Accuracy with different number of trees in the random forest model; here we use all 15 input signals.

With fewer trees, we also can get a high accuracy for performance. The results we got before use all part of original database for train data and test data, so we get a very high accuracy. Subsequently, we want to test the performance under the case when the train data and test data are not the same part of original database and we can get result in Figure 5.10:

Figure 5.10: Accuracy with different set of train data and test data; we use 9 sets of data and test another set, we do it for four different kinds of data.

Here we test matrix 1 to 10, and every time we use 9 matrices to build the forest model, then test the accuracy of the other matrix.
**Discussion:**
We find that different runs give us low accuracy. Different data sets are affected by external conditions; these conditions may influence the data more than the faults themselves.

**5.5 Comparison of Models**

In the random forest model, we can get a very high accuracy. With its variable importance, we can see input signals [6 7 11 13] have high importance, so we want to know whether these four important inputs work well in other models we build. Consequently, we test these four input signals for other model we build and we can compare them with the best performance we get and their confidence interval. We can see it in Table 5.8:

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy with best performance</th>
<th>Confidence interval</th>
<th>Accuracy with these four input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>42.5%</td>
<td>± 0.25%</td>
<td>38.3%</td>
</tr>
<tr>
<td>NN</td>
<td>76.6%</td>
<td>± 0.50%</td>
<td>74.2%</td>
</tr>
<tr>
<td>1-NN</td>
<td>48.4%</td>
<td>± 0.11%</td>
<td>48.1%</td>
</tr>
</tbody>
</table>

**Table 5.8:** Comparison of 4 input signals performance and best performance in each model.

We can find that the result we get is better than using only these four important input signals. However only using these four input signals also gives us high accuracy.
6. Summary and Conclusion

In this thesis, we evaluate the performance of fault detection with supervised methods. The objective is to find out whether there exists information in the measured signals to detect faults.

We get four best performances in different models: linear model, neural networks model, k-nn model and random forest model in our project. We can find that the best input signals we used are also different from each other among our best models. This is shown in Table 6.1:

<table>
<thead>
<tr>
<th>Model</th>
<th>Best Input Signals</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>[3,4,5,7,8,10,11,13,15]</td>
<td>42.5%</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>[6 7 10 11 13 14 15]</td>
<td>76.6%</td>
</tr>
<tr>
<td>K-NN</td>
<td>[1 4 7 10 11 13]</td>
<td>48.4%</td>
</tr>
<tr>
<td>Random Forest</td>
<td>[6,7,11,13]</td>
<td>96.9%</td>
</tr>
</tbody>
</table>

Table 6.1: Summary of the performance of different models

These are results that we get by using different models and we can find that the random forest model gives us the best performance.

With the performance in the table, we believe that supervised methods can detect different faults (charge air cooler leakage, radiator filter clogging and air filter clogging) in vehicles. And its performance is good enough when we use the random forest model to detect the fault. However, when the distribution of the test data is different from the training data, the accuracy will be lower. In our project, we also find the four input signals [6 7 11 13] are much more important than all others.
Bibliography


*Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*
Publisher: MIT Press Cambridge, MA, USA ISBN: 0262181908


*Neural Network Architectures: An Introduction*
Van Nostrand Reinhold, New York, American


*Supervision, Fault-Detection and Fault-Diagnosis Methods: An Introduction*


*Intelligent Data Analysis for Performance Evaluation and Fault Diagnosis in Complex Systems*
INSPEC Accession Number: 9701146


*Fault Diagnosis in HVAC Chillers*


*Networked Vehicles for Automated Fault Detection*
Early Warning Fault Detection Using Artificial Intelligent Methods
Universities Power Engineering Conference ’96, Iraklio, Crete, Greece.

On-line Fault Detection for Bus-based Field Programmable Gate Arrays
IEEE Educational Activities Department Piscataway, NJ, USA
Pages: 656 - 666 Year of Publication: 1998 ISSN:1063-8210

Neural Network Principles
Lincoln Laboratory Massachusetts Institute of Tecnology
Prentice Hall; 1st edition (January 15, 1994)

Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault Detection
Employing Differential Global Positioning (DGPS), Inertial and Vision Sensors
www.mdpi.com/journal/sensors Publish online

Fault Detection of Vehicle Suspension System Using Wavelet Analysis
DOI: 10.1080/00423110802094298

Fault Detection and Identification Automotive Engines using Neural Networks
Proc. of 16th IFAC World Congress in Prague, 4-8 July 2005.