2 Data foundation

The success of each Data Mining task strongly depends on the quality of the data that are used. Thus, assuring a high data quality before performing any data analysis is a crucial, maybe the most important step of any data analysis project. This is also pointed out by the popular saying "garbage in, garbage out" [10]. In other words, the results of any data analysis can not be better than the quality of the data.

Therefore, this chapter gives insights into the characteristics of the data that are studied in this work. It starts with the description of the data sources, i.e., it explains what data is analysed in this work and how it is generated. Hence, it introduces a special kind of data that is called *load spectrum data* and illustrates in detail how this data is derived from measurement signals. Then, it discusses how the data are enriched with additional information from the workshops.

Afterwards, it is explicated what preprocessing steps are performed to improve the data quality and to transform the data into an appropriate structure such that the studied Data Mining and Machine Learning algorithms can be applied to it.

At the end of this chapter, the most important properties of the two real-world datasets are provided that are analysed thoroughly in several distinct case studies in this thesis.

2.1 Data sources

During its lifetime, a modern vehicle produces a huge amount of data, such as signals from several sensors and from different ECUs, e.g., the engine control unit, which are communicated within the car through a controller area network (CAN). However, in-vehicle storage of data, i.e., data acquisition that takes place on-board of the vehicles, is still expensive. Thus, the memory capacity of modern vehicles is still limited severely. Moreover, equipping each vehicle with large scale on-board logging solutions would result in high development

costs, since these data loggers have to be provided an intelligent software, they have to be tested thoroughly and they have to be produced in mass, finally [93]. Amongst others, it is still not possible to record the complete data streams in modern customers' cars because of these reasons. Another issue is data confidentiality. It is prohibited by law to continuously log signals which would enable the vehicle manufactures to create individual, customer related driving profiles if the customer does not explicitly permit such a recording. For example, the GPS signal is among these critical signals.

Hence, only development cars and vehicles that are part of a preproduction test fleet are usually equipped with expensive data loggers to perform continuous signal recordings, while this is impossible for mass-production cars. However, the car manufacturers also require the knowledge about how their vehicles are driven and how the components of their cars are stressed under real-world conditions. Otherwise, it would be very difficult for them to improve the vehicles or their components in terms of several aspects, such as emission requirements that are given by the governments. Another example of such an aspect is the reliability of the vehicles, because it affects directly the satisfaction of the customers.

As a consequence, data are nowadays aggregated directly on-board in a customer's vehicle to account for the memory limitations and the restrictions given by data confidentiality reasons, as mentioned before. Then, there are two common ways how the vehicle manufactures extract this kind of aggregated data from each car: Either modern telematics services are used to transmit the data wirelessly to the companies' databases or the data are downloaded during a workshop visit and are sent to these databases, afterwards. This kind of logged and aggregated on-board data is called *load spectrum data* in this work. It is discussed in detail in the following subchapter.

2.1.1 On-board data: load spectrum data

Load spectrum data originate from Fatigue Analysis, where it is still the state-of-the-art data employed for calculating the fatigue life of components. Since the fatigue life of a component depends on the magnitude of the amplitudes of alternating stress and the frequency of occurrences, many counting methods have been developed to transform stress-time functions, such as signals from the ECU, to frequency distributions. However, these data reduction methods
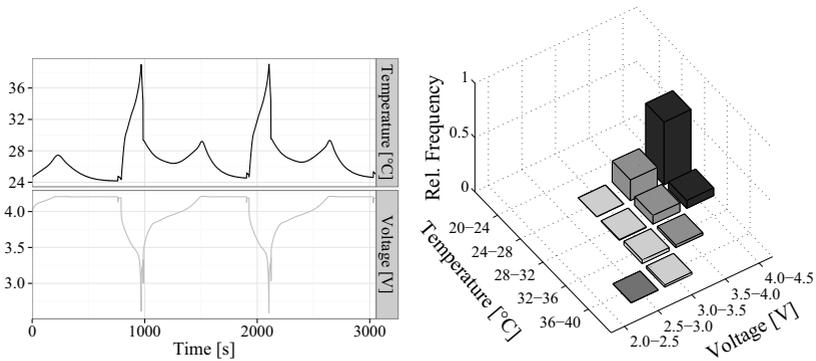
Figure 2.1: Measurements of the Li-ion battery signals *battery temperature* and *terminal voltage* and the corresponding relative load spectrum resulting from a two-parameter level distribution counting (cf. [9])

result in a loss of information, since some intrinsic characteristics of the signal data, like the chronology of certain events, are discarded. Thus, the responsibility lies with the analyst to decide for each use case individually whether such a data reduction is valid or not [61].

In dependence of the number of stress-time functions that are transformed simultaneously, the two main groups, in which these data reduction methods are frequently categorized in, are called the *one-* and *two-parameter counting methods*, respectively [104]. While in practise these are the most common groups, it has to be noted that in theory the majority of these counting methods is not limited by the number of signals they can process at the same time, i.e., they can be applied to even more than two signals. Regardless of the quantity of the processed signals, the outcome of a counting method is called a *load spectrum* in this work. However, it has to be mentioned that sometimes only the result of one-parameter counting is named load spectrum, whereas two-parameter counting leads to a *load matrix*. Since this distinction is irrelevant for the analysis presented in this study, the only term that is used in the following is *load spectrum*.

Since there are a plenty of different counting methods, the focus lies on the explanation of those techniques which are used to create the load spectrum data that are analysed in this work. The majority of them results from the *level distribution counting*, which is illustrated by Figure 2.1. The left chart

presents the curves of the two signals *temperature* and *terminal voltage* of a hybrid car battery, while the right one visualizes the corresponding normalized load spectrum that results from a two-parameter level distribution counting. Thereby, the measurements of these two signals are extracted from the publicly available *Battery Data Set* [102] by randomly selecting five runs of the lithium-ion (Li-ion) battery through the alternating operational profiles *charge* and *discharge* at room temperature. Moreover, breaks between these five runs are eliminated and it is additionally assumed that the signal values are measured at every second.

In general, the classes of interest, e.g. intervals, in which each of the two signals shall be grouped in have to be specified first, i.e., before the level distribution counting can be applied to these signals. Thereby, in the case of intervals, the widths do not necessarily have to be of the same size, but overlapping is not allowed. In the example shown in Figure 2.1, the five intervals that are used for partitioning the range of values of the battery's terminal voltage are $2.0 - 2.5V$, $2.5 - 3.0V$, $3.0 - 3.5V$, $3.5 - 4.0V$, and $4.0 - 4.5V$. Furthermore, the battery temperature is divided into the intervals $20 - 24°C$, $24 - 28°C$, $28 - 32°C$, $32 - 36°C$, and $36 - 40°C$. Afterwards, the total time of measurement is cut into equidistant time intervals. Then, the value of each signal is queried successively at each of the resulting points in time. In the example, the values are queried at every second. At the same time, the counter of the class formed by the corresponding intervals, to which the current values of the two signals belong to, is increased by one. Thus, the frequency counts of each class of interest are also a measure of the signals' operating time within the corresponding classes. Finally, the *normalized* or *relative load spectrum*, which is shown in the right chart of Figure 2.1, is obtained by dividing the frequency counts of each class by the whole measurement time, i.e., by the total sum of counts.

Another important counting method, which is underlying some of the studied load spectra, is the *rainflow-counting* algorithm. This method has been designed to account for cycle fatigue and is proposed in [81] for the first time, whereas [35] is the first publication in English about it. It appeared only a few years later.

The name of this method is based on its analogy with rainwater that is dripping down a pagoda roof. Figure 2.2 illustrates the basic principle of the original variant of this method. First, the (non-linear) time-series of recorded stress
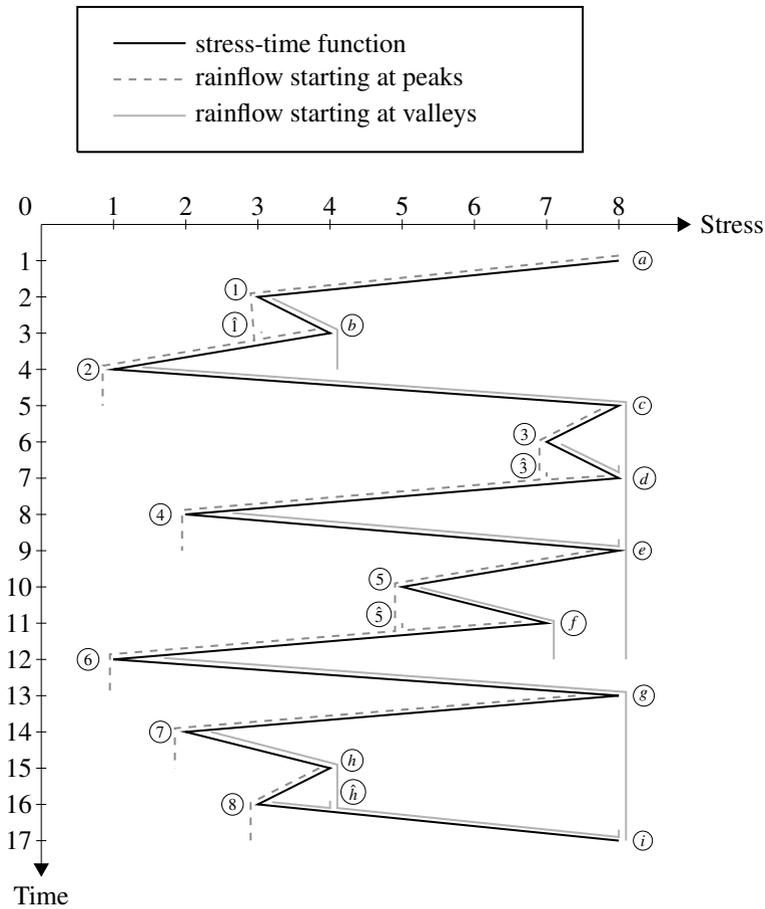
Figure 2.2: Basic principle of the rainflow-counting algorithm with peaks of
the stress-time function being tagged with letters, while valleys
are labelled with numbers

is reduced to a point process of peaks and valleys. Then, the stress history
is rotated clockwise by 90 degrees to symbolize the pagoda roofs. Next, it is
assumed that the rainwater flows down the pagoda, where each peak and each
valley of the process is imagined as a source of water. In Figure 2.2 peaks
are labelled with letters, while valleys are tagged with numbers to be able to

distinguish easily between them. Furthermore, each path of rain flow stops if one of the following conditions is satisfied [123]:

- If its source is a valley/peak, if its flow of water reaches a tip of a roof, and if it merges with another flow of rainwater that originates in a valley/peak that has a lower/higher stress value than its source; *or*

- If it merges with another flow that originates in a valley/peak of a higher roof level; *or*

- If it reaches the end of the time history.

In Figure 2.1, merging points of the water are indicated by letters or figures that are marked with the hat symbol, respectively. Here, the first rule applies, for example, for path $1 - b$, whereas the second one holds for $d - \hat{3}$.

Next, each resulting path of rain flow is counted as a half cycle and gets assigned a magnitude that equals the difference in stress between its starting and endpoint. Thus, the path $a - 2$ in Figure 2.2 gets assigned a stress value of 7, the path $5 - f$ a value of 2, and so forth. Finally, half cycles of the same magnitude and of the same location, i.e., if they have identical maximum and minimum values, are merged to a full cycle if their flow directions oppose each other. In Figure 2.2, the half cycles $a - 2$ and $2 - c$ build a full cycle, for example.

In order to obtain a load spectrum, the resulting counts are usually presented in form of a matrix or a triangular matrix. A full matrix is used to list the full cycles, their maxima and minima as well as their flow directions. However, if the information about the flow directions is not relevant, the full cycles can be stored in a triangular matrix that accounts for the maxima and minima. Figure 2.3 shows both variants of the load spectrum resulting from applying the rainflow-counting method to the stress history, that is illustrated by Figure 2.2.

Finally, it has to be mentioned that there are more than two possibilities to obtain the final load spectra. In the two representations that are shown in Figure 2.3, only full cycles are counted. However, if the stress-time function leads to half cycles that can not be merged to a full cycle, it would also be possible to count those separately.

In summary, load spectrum data contain aggregated information about the usage and stresses of a vehicle or its components. They are frequently computed
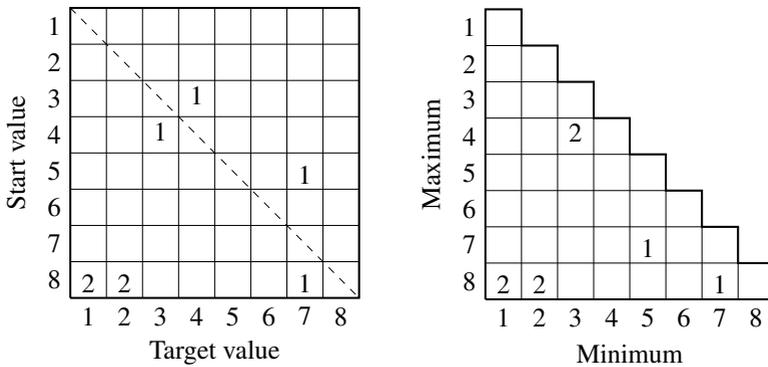
Figure 2.3: Two forms of representation of a load spectrum resulting from applying rainflow-counting to the stress-time function shown in Figure 2.2

directly on-board of each car with the help of a control unit. In this work, whenever a vehicle visits a workshop for service and repair that is authorized by an original equipment manufacturer (OEM), the current counts are downloaded and uploaded afterwards to a database that is managed by the OEM. However, the readouts do not provide any information about conducted repairs or replacements of any of the components of the vehicle. This information is collected in a different database which is described in the next subchapter.

2.1.2 Off-board data: workshop data

In this work, there is a database used which stores repair and maintenance information from OEM authorized workshops that facilitates the identification of vehicles suffering from a failure of a hybrid component. Table 2.1 shows a notional excerpt of such a database. It does not only contain information about which vehicle and what element was erroneous, but also provides some interesting facts about the date and type of repairing as well as the fault diagnosis that is made by the mechanist or by a control unit. Additionally, it shows the mileage of the vehicle at the point in time, when the car came into the workshop.

However, it has to be mentioned that the workshop data is mainly used for warranty issues and customer invoicing. Hence, it has unfortunately not been

Table 2.1: Conceptual illustration of the workshop information data

Car ID	Mileage [km]	Date of repair	Diagnosis	Failure location	Repair work
22	183450	2016-02-17	electric error	BMS	replacement of battery
815	17782	2014-10-03	contact fault	power electronics	cleaning of contacts
1103	64202	2015-08-13	brake wear	brakes	renewal of brake pads
⋮	⋮	⋮	⋮	⋮	⋮

designed for being merged with the database storing the load spectrum records. At least, both databases contain the attributes *CarID*, *Mileage*, and a time-stamp. In the database that stores the load spectrum data, the latter attribute stores the date, when the load spectrum data is read out from the control units of the vehicle. In the workshop database, this time-stamp refers to the manually entered date of repair. Since, the repair work does not necessarily have to take place on the same day as when the error memories of the vehicle are read out, there may be a mismatch of a few days between these to time-stamps of the two discussed databases.

The attribute mileage is also manually inserted by the mechanist in the workshop database, while this attribute stores the value that is revealed by the ECU in the load spectrum database. Therefore, typos may also lead to discrepancies in this attribute between the two databases.

All these issues have to be handled carefully, when the information stored in these two databases have to be merged. Corresponding dates are matched using the attributes *CarID*, *Mileage*, and the mentioned time-stamps in this work. In that way, individual readouts of the load spectrum data of a vehicle can be labelled with the information about the health status of the component of interest.

Table 2.2: Conceptual illustration of the dataset resulting from merging the load spectrum with the workshop information data

Car ID	Battery SoC [%]				Speed [*km/h*]				# ICE starts	Label
	0–25	25–50	50–75	75–100	0–10	10–20	...	240–250		
1	0.0	0.3	0.4	0.3	0.3	0.1	⋯	0.0	9832	faulty
2	0.1	0.4	0.4	0.1	0.2	0.2	⋯	0.1	78920	healthy
3	0.0	0.3	0.5	0.2	0.1	0.1	⋯	0.0	46010	healthy
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮	⋮	⋮

2.2 Preprocessing of data

After having merged the load spectrum data and the workshop information, several preprocessing steps are performed to achieve a high quality of the data. First, vehicles with a driven distance less than 1000 km are filtered out, because failures of components of these vehicles belong to the category "early failures" and are usually caused by manufacturing fault and not by loads of the vehicle or its components.

Moreover, sometimes the control units of a vehicle have to be flashed to install a new release of the operating software. As part of this process, the current values of the load spectra are downloaded and then reset to zero. Thus, the data before such a flash of a control unit have to be merged with the load spectrum data that are recorded after this event. However, if the readouts before and after a flash are not properly stored in the database, gaps can occur in the recordings of the load spectrum during the lifetime of a vehicle. Therefore, in order to ensure that the load spectrum data reflect the real usage and load patterns, vehicles are removed from the datasets if the corresponding load spectrum records do not cover at least 75% of their total mileage.

Since all the algorithms, which are studied in this work, require that the input is given in a matrix form, it has to be explained how such a data matrix is obtained. Hence, Table 2.2 presents the arrangement of the observed load spectrum data. All the recorded load spectra of each vehicle are concatenated

Table 2.3: Characteristics of the studied versions a) and b) of two real-world
datasets

	Dataset 1		Dataset 2	
Characteristic	a)	b)	a)	b)
Number of vehicles	6848	6670	8131	7576
Number of distinct operating countries	irrelevant	12	irrelevant	11
Minimum number of vehicles per country	irrelevant	25	irrelevant	100
Total number of load spectrum classes	737	737	823	823
Number of vehicles with a failure of the hybrid car battery	195	irrelevant	47	irrelevant

in a such a way that each row stores the observed load spectra values of an
individual vehicle, while each column contains the observed values of a partic-
ular class of a certain load spectrum. For example, the SoC of the hybrid car
battery of vehicle 1 has never been lower than 25%, while it has been in the
range from 50% to 75% for 40% of the operating time. Moreover, the ICE of
vehicle 1 has been started 9832 times.

2.3 Real-world datasets

Table 2.3 shows the main characteristics of the two datasets that are studied
intensively in this work. They result from two large HEV fleets, where the
versions a) and b) are created for each of them. Version b) is derived from
a) by filtering out vehicles that are driven in countries where the total number
of vehicles which are operated in the same country is less than 25 and 100
in dataset 1 and 2, respectively. These reduced datasets are necessary for the
visualization purposes of the analyses conducted in Chapter 4.

Each fleet contains only vehicles of the same type, whereas the car type is dif-
ferent in these two main datasets, i.e., the datasets 1 and 2 do not have any

vehicle in common. Moreover, individual characteristics which are not used for the studies carried out on the dataset, are flagged as "irrelevant".

Dataset 1 contains less vehicles and each of them is described by less features compared to those of dataset 2. Furthermore, it is known which vehicles included in the versions a) of both datasets suffer from a failure of the hybrid car battery. However, the types of defect leading to the failure of this component are manifold in dataset 1, whereas there is only a single error type predominant in dataset 2. Nevertheless, it is not differentiated between the distinct kinds of failures in dataset 1, i.e., all faulty vehicles are only labelled as "faulty". The reason is that there is not enough information available about the different types of failures that are contained in dataset 1.

Finally, there is a severe imbalance between the number of "healthy" and "faulty" vehicles in each of the two datasets. The imbalance ratio between these two classes is approximately 35:1 in dataset 1a), whereas it is 173:1 in dataset 2a). This special property may have a strong influence on the outcome of the algorithms that are studied in this work, as will be explained in Chapter 3.1.2 in detail.

2.4 Conclusion

In this work, the focus lies on performing classification, visualization, and rule learning on load spectrum data that have been enriched by information about repairs of particular components. Since the considered data sources already have been created for other purposes, there is no need for spending any additional money for collecting the data, for buying and setting up the required IT-infrastructure, or for installing new hardware within the vehicles of interest. However, a drawback is that the databases storing the load spectrum data and the workshop information have not been designed for being merged. Hence, merging these two databases is not straightforward.

Like in all data analysis tasks, the quality of the results strongly depends on the data quality. Thus, several pre-processing steps have been performed to create two main datasets where two variants are created from each of them. These two datasets will be studied thoroughly in several distinct case studies in the upcoming chapters.